CEUR-WS.org/Vol-4141/paper5.pdf

A method for synthesising system architecture for IT infrastructure resistant to social engineering attacks

Sergii Lysenko[1,*], Oleksandr Bokhonko[1], Tomas Sochor[2], Olha Atamaniuk[1], Nadiia Lysenko[1] and Jiri Balej[3]

[1]*Khmelnytsky National University, Instytutska street 11, 29016, Khmelnitsky, Ukraine*
[2]*European Research University, Ostrava, Czech Republic*
[3]*Mendel University in Brno, 1665/1, Zemědělská, Brno, 613 00, Czech Republic*

Abstract
The primary purpose of this study is to develop a robust method for synthesizing IT infrastructure architecture resilient to social engineering attacks, addressing the critical vulnerability of the human factor in cybersecurity. Current detection mechanisms, ranging from rule-based filters to standard machine learning models, often fail to adapt to personalized attack strategies and suffer from high false-positive rates, creating a significant gap in effective defense. In this work, we propose a hierarchical multi-agent system that decouples high-level decision-making from modality-specific evidence gathering, utilizing reinforcement learning guided by entropy reduction and knowledge-graph priors to coordinate specialized agents for email and web content. Experimental results on a combined corpus of 80,000 samples demonstrate that the proposed architecture achieves an Area Under the ROC Curve of 0.96 and an F1-score of 0.92, significantly outperforming single-agent baselines while reducing the false-positive rate at a 95% true-positive rate to 0.08 and cutting the number of actions per sample by 47%. We conclude that this principled coordination of specialized agents offers a scalable and efficient path to engineering adaptable, multi-level defenses that maintain high accuracy and low latency against evolving social engineering threats.

Keywords
Distributed systems, social engineering, computer systems, machine learning, multi-agent systems

1. Introduction

The increasing reliance on digital communication and online services has been accompanied by a steady rise in social engineering attacks. Unlike purely technical exploits, these attacks target human behavior, exploiting trust and psychological manipulation to gain unauthorized access to information of IT infrastructure. Common techniques, such as phishing, pretexting, and baiting, are frequently used as entry points for more complex intrusions, including ransomware infections and advanced persistent threats [1, 2, 3].

Conventional security measures, including firewalls and intrusion detection systems, are effective against technical vulnerabilities but provide limited protection when the human factor is exploited. This has made the detection and prevention of social engineering attempts a pressing issue for both practitioners and researchers in the cybersecurity field [4]. Effective solutions require a multidisciplinary approach that combines technical defenses with behavioral and contextual analysis [5, 6].

Current research efforts have produced a range of detection methods, from simple rule-based filters to models employing natural language processing and machine learning [7, 8]. While these tools have demonstrated promise, many struggle to adapt to new or highly personalized attack strategies and often suffer from issues such as high false positive rates. These shortcomings highlight the need for more sophisticated and adaptive approaches [9, 10].

ExplAI-2025: Advanced AI in Explainability and Ethics for the Sustainable Development Goals, November 07, 2025, Khmelnytskyi, Ukraine

*Corresponding author.

sirogyk@ukr.net (S. Lysenko); booweb24@gmail.com (O. Bokhonko); tomas.sochor@eruni.org (T. Sochor); olhaatamaniuk12@gmail.com (O. Atamaniuk); nadialysenko07@gmail.com (N. Lysenko); jiri.balej@mendelu.cz (J. Balej)
0000-0001-7243-8747 (S. Lysenko); 0000-0002-7228-9195 (O. Bokhonko); 0000-0002-1704-1883 (T. Sochor); 0000-0002-6054-8700 (J. Balej)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2025-12-22

The proposed approach is intended to build the an efficient method for synthesising system architecture for IT infrastructure resistant to social engineering attacks, which will be able to improve the accuracy and adaptability of social engineering detection while maintaining usability for end users. The results presented here contribute to the broader effort to develop reliable defenses against one of the most persistent and difficult-to-detect threats in modern cybersecurity.

2. Related works

A significant body of research has focused on improving the detection of social engineering and intrusion attempts by leveraging multi-agent systems, multimodality, and federated learning paradigms. Recent studies highlight diverse approaches that combine technical feature extraction, behavioral analysis, and adaptive architectures to enhance detection accuracy while ensuring robustness and scalability.

In [11], the authors propose PhishDebate, a modular framework where specialized agents independently analyze different layers of a webpage (URL, HTML, semantic content, brand imitation signals). Their outputs are consolidated through a debate-style interaction involving Moderator and Judge roles. This structure reduces the risk of single-channel errors while improving interpretability through structured arguments. The debate process forces agents to reconcile contradictions (e.g., legitimate semantics vs. suspicious domains), resulting in high completeness and TPR rates (98.2%), outperforming single-agent and chain-of-thought (CoT) baselines. Importantly, the framework supports flexible reconfiguration and discusses deployment strategies such as automated parsing, input normalization, and an “argument language” for inter-agent communication.

The work in [12] extends this idea by integrating five cooperative agents (text, URL, metadata, explanation, and adversarial) whose contributions are weighted via proximal policy optimization (PPO). A key innovation is the inclusion of an adversarial agent that generates paraphrased or restructured phishing emails, enabling the detection system to adapt to evolving attacks. An explanation-simplifier agent translates technical reasoning into user-friendly summaries, thereby enhancing trust and auditability. Experimental results demonstrate a balanced performance with 97.89% accuracy, FP of 2.73%, and FN of 0.20%, along with robustness against URL shorteners and style variations. The system also demonstrates scalability and compatibility with existing email security infrastructure.

In [13], three specialized models separately process URL strings, webpage content, and DOM structures, after which the final classification is made based on the highest confidence score. This parallel feature processing reduces the likelihood of correlated errors and yields high accuracy (99.21%) with low false positives. The architecture is resilient to previously unseen websites due to a combination of generic and domain-specific indicators. Practical deployment considerations include server-side scalability and container-based accelerated execution, with a roadmap for reinforcement learning integration.

Multimodality is further explored in [14], where the authors combine multilingual large language models (MLLMs) with online and offline knowledge bases. Multimodal queries are constructed from visual (logos, favicons) and structural (HTML/DOM) signals, with a top-k retrieval strategy from offline knowledge improving detection of rare brands. This approach increases recall and reduces FP/FN rates with low latency. The study also addresses security concerns, such as trust in external knowledge and resilience against manipulated data sources, framing the method as a step toward explainable multimodal detection.

Several works have emphasized practical user-side deployment. In [15], the authors propose a browser extension + Docker + BiLSTM with attention pipeline for real-time URL classification, including detection of hidden redirects from HTML/JS. Three aggregation strategies are evaluated (SPhS, MSS, WeAS), with WeAS achieving the best trade-off. The system is particularly effective against Browser-in-the-Browser attacks by analyzing both primary and embedded URLs. Engineering considerations such as caching, containerization, and timeout management are presented to ensure integration with minimal infrastructure changes.

Beyond phishing websites, [16] addresses online social networks (OSN) by combining semantic profile attributes, text stylometry, and relational/graph features to detect impersonation and account cloning. The ensemble method achieves ∼95% accuracy, outperforming individual classifiers and maintaining robustness with incomplete or noisy attributes. Ablation studies highlight the contribution of each feature group, improving interpretability and deployment optimization. Cross-platform applicability and privacy policy constraints are also considered.

Recent work also explores federated learning (FL) for intrusion detection systems (IDS). In [17], decentralized sensor-level detection is performed with continuous local updates and knowledge aggregation through FL under non-IID traffic conditions. Sequential packet labeling enables early detection, with CNNs exceeding 95% detection and LSTMs capturing attacks within the first 15 packets. Communication cost analysis demonstrates trade-offs between synchronization frequency and detection quality. Similarly, [18] surveys FL paradigms for IDS across domains such as IoT, healthcare, finance, and enterprise networks, highlighting challenges such as model poisoning, gradient anomalies, and secure aggregation. The paper outlines a roadmap for scalable and reliable FL-based IDS.

Other works focus on hierarchical and anomaly-based methods. In [19], the authors design a multi-layer architecture with behavior-driven traffic sampling and hidden Markov models (HMMs) to predict multi-stage attacks. The system reduces event mixing and achieves up to 7.5× processing acceleration over centralized methods, addressing scalability in high-throughput networks. In [20], a hybrid detection pipeline is introduced that combines min–max normalization, feature reduction via Equilibrium Optimizer, and an LSTM-AE model tuned by an improved Snow Ablation Optimizer. On CIC-IDS2017, this approach achieves 99.46% accuracy, surpassing state-of-the-art anomaly detection benchmarks. A coarse-to-fine filtering scheme is proposed in [21], where a primary particle filter removes obvious anomalies and a secondary filter targets subtler deviations. This reduces false alerts while preserving sensitivity and scalability for large traffic volumes.

The paper [22] provides a comprehensive survey of IDS methods, classifying them into misuse and anomaly-based approaches and mapping architectural patterns (centralized, distributed, edge-based). The paper discusses datasets, deployment scenarios (cloud, IoT, ICS), and integration with AI/ML and blockchain for log integrity. A comparative threat-method matrix is introduced, along with directions for future work such as multi-agent hierarchies and privacy-preserving FL protocols.

Taken together, these studies demonstrate a strong need of the construction of the efficient method for synthesising system architecture for IT infrastructure resistant to social engineering attacks. While high accuracy has been achieved, challenges remain in adaptability to evolving attacks, interpretability of decisions, and practical deployment under resource constraints.

3. Methods and material

3.1. Fundamentals of the method

The study proposes a method for synthesizing the architecture of an IT-infrastructure system resilient to social-engineering attacks, based on a hierarchical multi-agent system employing reinforcement learning. The method enables construction of a system capable of detecting social-engineering attacks and relies on the interaction of specialized components that reflect the key processes of analyzing the information environment.

At the core is an agent which, in coordination with an attack classifier, formulates the dialogue policy between the system, the user, and other sources of threat. The agent directs the collection of contextual and technical features of a message or interaction, taking into account input information that may indicate a social engineering attack, for example, lexical–semantic features of an email's text, network connection parameters, user behavioral characteristics within the communication channel, etc. The attack classifier operates at two levels:

1. At first level it determines whether the incoming information exhibits signs of social engineering.
2. At second level it refines the assessment to identify the type (class) of attack and the specific tactic employed by the adversary.

An important role is played by the Condition Monitor Component (CMC), which accumulates all collected indicators and provides a holistic representation of the current interaction context. This prevents redundant checks, preserves the history of gathered features, and dynamically updates the probabilistic estimate of a message's belonging to a given attack category. Complementing this, a Simulation Engine Component (SEC) models user responses or attacker behaviors, creating conditions for training agents in an environment characterized by high uncertainty and a diversity of possible adversary strategies.

The whole system is conceived as a hierarchical structure in which a high-level agent decides which specific actions to assign to specialized lower-level sub-agents. These actions may include analysis of URL structure, domain-reputation checks, semantic assessment of a message, or detection of indicators of psychological pressure in voice calls. In this way, a multi-layered decision-making system is formed that can respond flexibly to novel social-engineering tactics and reduce risks to the IT infrastructure by proactively collecting relevant features and progressively decreasing the entropy of the information state.

The method includes the usage of the Interaction Manager Component (IMC) – a full-fledged agent capable of goal-oriented interaction of making the dialogs with the information environment. At each time step τ the agent receives from the Condition Monitor Component the current environment configuration, which reflects the set of indicators and behavioral characteristics collected so far. This state is then mapped to the selection of an action α_τ from the set of possible actions A according to the policy $\rho(\alpha|\sigma(\tau))$.

A Simulation Engine Component generates a response that serves as the basis for computing the reward signal ω_τ, which is returned to the agent. As a result, the environment transitions to a new state $\sigma(\tau + 1)$, and thus unfolds a sequence of iterations that simulate the real dynamics of communication. The goal of the agent is to find the optimal policy that ensures the maximization of the expected cumulative reward. The reward ω_τ is defined by the function $\Omega(\sigma(\tau), \alpha_\tau)$, where the coefficient $\lambda \in [0, 1]$ serves as a discount factor that reduces the weight of temporally distant outcomes, while Θ denotes the maximum number of dialogue steps.

The environment state describes the set of collected attack indicators, which may include signs of phishing messages, linguistic markers of psychological pressure, email metadata, or characteristic URL patterns. Let us denote A as the set of social engineering attack types, and I – the set of indicators associated with these attacks. Then the state $\sigma(\tau)$ can be represented as a vector $[\chi_1, \chi_2, \dots, \chi_{|I|}, \tau, \varepsilon]$, where each component χ_i – is an encoded ternary representation of the indicator's presence: confirmed presence, confirmed absence, or uncertain state. The vector includes the parameter τ, that reflects the current dialogue step, and the variable ε, which signals the repetition of previous queries. This formalization allows the method not only to accumulate knowledge about the collected indicators but also to adaptively manage the querying strategy, minimizing informational redundancy and improving the accuracy of social engineering detection under conditions of incomplete and contradictory information.

The agent's action space is formed as the union of two subsets representing different types of actions in the process of detecting social engineering attacks. Let us present $A = T \cup I$, where T denotes the set of attack types, and I - the set of indicators that can be queried to clarify the context of the message or interaction. If the selected action α_τ belongs to I, the agent initiates a request for additional information regarding a specific indicator, for example, checking for suspicious domains, characteristic URL patterns, lexical markers of psychological pressure, or anomalous user behavior signals. In the case of selecting an action from the subset T the agent forms the final decision regarding the probable type of attack, which concludes the interaction process within the current scenario.

The success of the process is evaluated based on the accuracy of threat identification, that is, how correctly the agent classified a specific interaction into a particular class of social engineering. This formalization allows the agent to dynamically balance between actively gathering features and making decisions, optimizing interaction of making the dialogs with the information environment to achieve maximum efficiency under conditions of incomplete or contradictory information. At the same time, the hierarchical structure of the system provides flexibility in the allocation of computational resources and contributes to the gradual reduction of uncertainty in assessing the risk of each message or interaction, which is critically important for the resilience of IT infrastructure against social engineering attacks.

3.2. Synthesis of the architecture of an IT infrastructure system

Let us denote the set of social engineering attacks as the set T divided into K groups, where $T = T_1 \vee T_2 \vee \cdots \vee T_K$, the subsets do not intersect, that is $T_i \wedge T_j = \emptyset$ for $i \neq j, i, j = 1, \ldots, K$. Each subset T_i corresponds to a specific category of attacks or an adversary's tactical strategy and is associated with its own set of indicators I_i, which characterize the features of messages, behavioral patterns, or technical signs inherent to this group. Such organization allows lower-level agents of the system to focus on specific types of threats without spending resources on analyzing less relevant signals, while also maintaining the modularity and scalability of the architecture.

The hierarchical structure of the multi-agent system provides for two levels:

1. The upper-level Decision Agent (DA), which coordinates the overall policy of information gathering and decision-making;
2. The lower-level Service Agents, each are responsible for its own subset of attacks T_i and the indicators associated with it I_i.

Service Agents perform local analysis functions, initiate requests to users or system sensors, evaluate message attributes, and transmit the results to the upper-level coordinator. Due to this distribution of responsibilities, the Decision Agent can make informed decisions based on aggregated signals aimed at minimizing risk and reducing uncertainty regarding the nature of a potential attack.

The selection of actions by agents is formalized through the mapping of the environment state $\sigma(\tau)$ to the action space α_τ from the set $A = T \cup I$, where each lower-level action concerns only a subset of indicators I_i, and the actions of the upper-level agent coordinate the sequence of queries and the final assessment of the threat type. Such a hierarchical organization allows the system to gradually accumulate relevant features, adaptively adjust the data collection strategy, and ensure effective detection of complex social engineering scenarios even under conditions of incomplete or contradictory information.

In each interaction cycle, the central agent receives the state of the environment $\sigma(\tau)$, which reflects the set of all collected indicators of potential social engineering attacks at that moment. Based on the policy $\rho_0(\alpha_0\tau|\sigma(\tau))$ the upper-level agent makes a decision regarding the further course of action: to continue collecting features through queries to the user or system sensors, or to form the final assessment of the probable type of threat.

The action space of the Decision Agent is defined by the set $A_0 = \{W_1, W_2, \ldots, W_K, C\}$, where W_i denotes the Service Agent responsible for analyzing a specific subset of indicators I_i, and C corresponds to the attack classification module. When one of the Service Agents is activated, a subtask is initiated, within which the following $\tau \in [\tau_0, \tau_0 + L]$ feature collection steps are performed by the same Service Agent, where L defines the maximum number of iterations for the subtask. The current subtask ends after receiving a specific response to the indicator query, which can take discrete values, for example, "confirmed" or "not confirmed". If the classification module C is activated, the upper-level agent forms the final decision about the type of attack, completing the current interaction cycle.

The Decision Agent coordinates the work of all Service Agents, optimizes the sequence of queries and decision-making, while the Service Agents focus on local feature collection subtasks, which significantly reduces computational complexity and increases the accuracy of detecting complex social engineering scenarios. At the same time, this system architecture allows agents to adaptively change their strategy depending on the current state of the environment, minimizing uncertainty and increasing the resilience of the IT infrastructure to social engineering attacks.

When a Service Agent is activated, it receives the environment state $\sigma(\tau)$ and extracts the relevant indicators υ_τ, that correspond to its assigned threat subgroup, $\upsilon_\tau = [\chi_1, \chi_2, \ldots, \chi_{|I_i|}, \tau, \varepsilon]$. Each Service Agent operates within a local feature subspace, allowing it to focus on specific categories of social engineering attacks and the corresponding behavioral or technical signals. Based on the local policy, produced by the Interaction Manager Component $\rho_i(\alpha_i\tau|\upsilon_\tau)$ the Service Agent selects the next indicator for clarification or verification, performing steps to collect additional information that refine the probability of a specific type of attack.

The action space of each Service Agent is defined by the set $A_i = \upsilon_1, \upsilon_2, \dots, \upsilon_{|I_i|}$, where each element corresponds to a specific indicator or pattern that the agent can query or verify within its subtask. This localization of actions allows Service Agents to gradually accumulate knowledge about a specific threat category, reducing noise from less relevant signals and ensuring a more accurate risk assessment. Hierarchical interaction with the Decision Agent makes it possible to aggregate these local results and adjust the overall system policy, increasing the adaptability and resilience of the IT infrastructure to social engineering attacks even under complex conditions of incomplete or contradictory information.

In the hierarchical architecture with reinforcement learning, the Decision Agent receives an external reward $\omega_e\tau$ at each interaction cycle, which reflects the effectiveness of the decisions made in detecting potential social engineering attacks. After the subtask is completed, Decision Agent calculates the accumulated reward $\Omega_0\tau$ for the given cycle, taking into account the contribution of each Service Agent and the results of the local evaluation of indicators. Let K denote the number of Service Agents, and $\lambda \in [0, 1]$ – the discount factor, that reduces the weight of temporally distant outcomes. Then, the accumulated reward is defined as:

$$\Omega_0\tau = \begin{cases} \sum_{\kappa=1}^{L} \lambda^{\kappa} \omega^{e}_{(\tau+\kappa)}, & \text{if } \alpha_0\tau = W_i; \\ \omega^{e}_{(\tau)}, & \text{if } \alpha_0\tau = C. \end{cases} \tag{1}$$

where W_i - the Service Agent responsible for the subtask of analyzing the subset of indicators I_i, C - the attack type classification module.

Such formalization allows the Decision Agent to integrate the results of local analysis, evaluate the effectiveness of actions at all levels, and adaptively adjust the policy for data collection and decision-making. The use of discounting λ ensures a balance between short-term and long-term outcomes, which is critically important for improving the accuracy of detecting complex social engineering scenarios.

The goal of the update $L(\Theta_0)$ is defined through the discrete learning dynamics based on the reinforcement signal and the interaction history. Let σ' denote the next state of the environment after performing the action α'_0, Θ'_0 - the parameters of the target network, Θ_0 – the parameters of the current network, and B_0 - the Decision Agent's experience replay buffer. Then the loss function is defined as:

$$L(\Theta_0) = E_{\sigma,\alpha_0,\Omega_0,\sigma'\sim B_0}((\Omega_0 + \lambda \max_{\alpha'_0} Q_0(\sigma', \alpha'_0; \Theta'_0) - Q_0(\sigma', \alpha_0; \Theta_0))^2), \tag{2}$$

where $Q_0(\sigma, \alpha_0; \Theta_0)$ evaluates the expected cumulative reward under the current policy, and the term $\lambda \max_{\alpha'_0} Q_0(\sigma', \alpha'_0; \Theta'_0)$ ensures the consideration of the long-term consequences of actions. Such a formalization allows the Decision Agent to effectively integrate information from the Service Agents and the Classification module, evaluating both the short-term and long-term effects of its decisions, which is critical for improving the accuracy of detecting complex social engineering attacks. The Service Agents, receive an internal reward $\omega_i\tau$ at each interaction cycle, which evaluates the effectiveness of their local work with the subset of indicators I_i, assigned to a specific category of social engineering attacks.

The goal of the Service Agents is to maximize these rewards, which ensures accurate and efficient execution of information gathering subtasks. The loss function of each Service Agent $L(\Theta_i)$ is defined based on the local policy and the interaction history in the replay buffer B_i, where Θ'_i denotes the parameters of the target network, and Θ_i - the parameters of the current network:

$$L(\Theta_i) = E_{\sigma_i,\alpha_i,\Omega_i,\sigma'\sim B_i}((\Omega_i + \lambda \max_{\alpha'_i} Q_i(\sigma'_i, \alpha'_i; \Theta'_i) - Q_i(\sigma_i, \alpha_i; \Theta_i))^2), \tag{3}$$

where $Q_i(\upsilon_i, \alpha_i; \Theta_i)$ evaluates the expected cumulative reward under the current local policy, and the term $\lambda \max_{\alpha'_i} Q_i(\upsilon'_i, \alpha'_i; \Theta'_i)$ takes into account the long-term consequences of the agents' actions. Such a formalization allows the Service Agents to adaptively adjust their own feature collection strategy, focusing on local subtasks, and gradually improves the accuracy of risk assessment for specific attack categories. Coordination with the Decision Agent allows aggregating local results and ensures the optimization of the overall system policy.

3.3. Reward calculation

Within the hierarchical system architecture, starting from the initial state σ_0, for the main threat classifier and auxiliary classifiers specializing in individual categories of social engineering attacks, the initial probabilistic distributions are formed. For the Decison Agent, let us denote this distribution as $\phi_0^D = [\phi_1, \phi_2, \ldots, \phi_k]$, where each component ϕ_i corresponds to the posterior estimate of the current scenario's belonging to a certain class of threats. For the Service Agents, we define a similar distribution $\phi_0^S = [\phi_1, \phi_2, \ldots, \phi_j]$, where each element represents the probability of detecting the corresponding attack subcategory within the set of possible actions A.

An important characteristic of these distributions is an entropy, which reflects the degree of uncertainty in decision-making. The initial entropy of the Decison Agent, operating at the level of the integral threat assessment, is defined by the expression:

$$J^D(\sigma_0) = -\sum_{i=1}^{k} \phi_i \log(\phi_i), \tag{4}$$

while the initial entropy of the Service Agents is calculated using the formula:

$$J^S(\sigma_0) = -\sum_{i=1}^{|A|} \phi_i \log(\phi_i). \tag{5}$$

Both entropy metrics serve as normalizers, preventing biases in rewards during different interaction episodes. This is especially important when the nature of information information coming from users or the system's internal components can vary significantly between scenarios. Thus, by using entropy as a control parameter, the hierarchical multi-agent system gains the ability to adaptively balance between attack detection accuracy and resilience to information noise, which is characteristic of real-world IT infrastructure operation conditions.

Similarly, at the time moment τ the information entropy of the main Decision Agent classifier is calculated $J^D(\sigma_\tau)$ and the Service Agent classifier $J^S(\sigma_\tau)$. After the system takes an action and transitions to a new state $\sigma_{\tau+1}$, the updated entropy values are determined as $J^D(\sigma_{\tau+1})$ and $J^S(\sigma_{\tau+1})$. The difference between these values becomes the basis for calculating the reward, which characterizes the reduction of uncertainty in the work of the Decision Agent and the Service Agents.

For the Decision Agent the reward value is defined by the relation:

$$\rho^D_{\text{entropy}} = \max\left(\frac{(J^D(\sigma_\tau) - J^D(\sigma_{\tau+1}))}{J^H(\sigma_0)}, 0\right). \tag{6}$$

The the Service Agents reward value is defined by the formula:

$$\rho^S_{\text{entropy}} = \max\left(\frac{(J^S(\sigma_\tau) - J^S(\sigma_{\tau+1}))}{J^S(\sigma_0)}, 0\right). \tag{7}$$

In these expressions, the ratio of the entropy difference to the initial value normalizes the functional, allowing the system to adequately compare the level of information gain across different interaction episodes. The use of the max operator prevents the occurrence of negative values, as in practice there may be situations where, due to incomplete data or noisy signals, entropy increases, and thus uncertainty does not decrease. This step ensures stable learning aimed at gradually reducing informational chaos, which is characteristic of social engineering attacks where adversaries often create situations of cognitive confusion. With such a reward structure, the hierarchical multi-agent system gains an additional self-adaptation mechanism aimed at finding solutions that minimize the entropy of the state, gradually transitioning the infrastructure from an uncertain state to one with a clear diagnostic conclusion regarding the presence or absence of attack indicators.

3.3.1. External reward for the decision-making agent

The external reward ρ_t^{ext} is defined piecewise as follows:

$$\rho_t^{\text{ext}} = \sum_{i=1}^{n} N + \rho_{\text{entropy}}^{D}, \tag{8}$$

where N is a positive if the agent emits a final decision and that decision is correct (success condition is met); N is negative if the selected action at step t, repeats a previously executed action in the same episode (action repetition detected, redundant queries are penalized), or if the maximum permitted number of interaction steps is reached without a final decision (episode budget exhausted, indecision are penalized); N is equal to 0 in all other cases (ongoing interaction, reward proportional to the reduction of uncertainty, the incremental benefit of informative actions that reduce uncertainty are scaled); $\rho_{\text{entropy}}^{D} > 0$.

3.3.2. Internal reward for the service agents

The internal reward ρ_t^{int} for a Service Agent is defined as follows:

$$\rho_t^{\text{int}} = \sum_{i=1}^{n} N + \rho_{\text{entropy}}^{S}, \tag{9}$$

where N is a positive if the Service Agent's query yields a definitive and correct resolution of a targeted indicator or feature (correct match observed); N is a negative if the Service Agent issues a repeated or redundant query that does not bring new information (repetition in actions); N is equal to 0 in all other cases (progressive uncertainty reduction without a definitive match); $\rho_{\text{entropy}}^{S} > 0$.

The maximum number of interaction steps sets an upper bound on episode length and appears in the timeout condition for the external reward. The normalization by the initial entropy for both levels ensures that rewards are comparable across different scenarios with varying prior uncertainty. The "success" term aligns the upper-level agent with accurate final classification, while the local "match" term aligns service agents with accurate and efficient feature resolution. The repetition penalties at both levels act as regularizers against cycles and unnecessary queries. Together, these definitions balance the desire for short, decisive interactions with the requirement to reduce diagnostic uncertainty in a principled, quantifiable way.

The hierarchical reward structure ensures a harmonious distribution of roles between system levels, where the Decision Agent acts as the coordinator for IT-infrastructure security evaluation, and the subordinate agents serve as local filters, enabling the formation of multi-level protection against social engineering attacks, reducing the risks of cognitive manipulation, and ensuring effective system adaptation in a dynamic environment.

In addition to the reward based on the reduction of entropy uncertainty, the main agent receives additional reinforcements that are generated depending on the outcome of the decision made. If the final classification of the attack scenario is successful, that is, the system correctly identifies a social engineering attempt or confirms its absence, then a positive reward increment is applied, denoted as σ^{+}. In the case of incorrect identification or reaching the maximum allowable number of dialogue iterations, a penalty component is introduced σ^{-}, which reflects the inefficiency of the agent's strategy.

For the Service Agents performing localized tasks of analyzing individual user behavioral features, the reward is determined according to a different principle. If the selected agent correctly refines a feature characterizing a possible attack and receives an unambiguous response from the environment (for example, confirmation or denial of the feature, similar to "yes" or "no" responses in a medical dialogue), then it receives a positive increment τ^{+}. If a repeated query occurs that does not reduce the entropy of the information state and leads to informational noise, then a negative penalty is introduced τ^{-}.

The formalization of reinforcement signals at different levels of the system architecture combines the global assessment of the main agent's performance with the local accuracy of the subordinate agents' work. The use of parameters σ^{+}, σ^{-}, τ^{+} and τ^{-} makes it possible to balance the strategic and tactical aspects of the system's functioning, creating conditions for adaptive learning in a dynamic environment. In the context of countering social engineering attacks, this means that the main agent learns to distinguish the overall structure of manipulative scenarios, while the Service Agents focus on the smaller details of the interaction, which together form a holistic and resilient behavioral model of the IT infrastructure.

3.4. Knowledge development for interaction manager component

Within the proposed system architecture, knowledge about social engineering attack scenarios was be represented in the form of a knowledge graph integrated into the interaction manager component. In this graph, the nodes correspond to attack indicators and their manifestations, as well as classes of attack actions, while the edges describe the correlation dependencies between these elements. Each edge is assigned a weight that characterizes the degree of interconnection between a specific indicator and the attack class, with the weight value determined based on the probabilistic statistics obtained during the analysis of training and testing scenarios.

Since the method employs a hierarchical multi-agent structure, the Service Agents are responsible for processing subsets of possible indicators and attack classes. For each such agent, a separate knowledge subgraph is constructed, containing a set of indicators O_i and a set of attacks A_i, relevant to its subsystem. This ensures computation localization and reduces the overall complexity of architecture synthesis, as individual agents specialize in specific classes of threats.

The edge weights in the subgraphs are interpreted as conditional probabilities of observing a feature given the occurrence of a particular attack class. To formalize such a dependency, the following relation is introduced as following:

$$q(o_u \mid a_v) = \frac{m(a_v, o_u)}{\sum_{z=1}^{Z} m(a_v, o_z)}, \tag{10}$$

where $m(a_v, o_u)$ denotes the number of recorded scenarios in which an attack of type is accompanied by the manifestation of the feature o_u.

By calculating the probability value for each "attack–feature" pair, it is possible to construct a conditional probability matrix $\Lambda_i \in R^{|O_i| \times |A_i|}$, which is used in the agent training process. Such a formalization makes it possible to represent the complex multidimensional structure of dependencies between different types of social engineering attacks and their features, thereby providing the capability for adaptive agent learning in the process of interacting with potentially hostile scenarios.

Based on the conditional probability matrix introduced earlier, the current probability distribution of feature manifestations for the Service Agents is defined as:

$$\pi(o_u) = \Lambda_i \cdot \pi(a_v), \tag{11}$$

where $\pi(a_v) \in R^{|A_i| \times 1}$ the probability vector of the corresponding attack classes, which is calculated by the subordinate classifier for the subgroup i, and $\pi(o_u) \in R^{|O_i| \times 1}$ - the probability distribution for local features relevant to this agent. This operation transforms the probabilities of attack classes into the probabilities of specific feature manifestations, providing the basis for the agent's decision-making.

Next, this probability distribution is combined with the estimates Q - of the values computed by the Service Agents, which allows taking into account both prior knowledge about the probabilities of feature occurrences and the agent's learning experience in a dynamic environment. Formally, the choice of the agent's next action is expressed as:

$$\alpha_{\mathcal{T}} = \arg\max_{\omega}(\sigma(Q_\omega(\sigma_{\mathcal{T}})) + \sigma(\pi(o_u))), \tag{12}$$

where $Q_\omega(\sigma_{\mathcal{T}})$ the distribution Q - of the values computed by the Service Agent for the state $\sigma_{\mathcal{T}}\sigma$ - the sigmoid normalization function, which maps values into the [0,1] range, and *argmax* function determines the feature or action class on which the agent stops its choice for further querying or evaluation.

Due tothis combination of probabilities and Q - values, the agent gains the ability to take into account both prior experience and updated information about user or system behavior, which is critically important for countering social engineering attacks. This approach allows the agent to adaptively select priority features for verification, increasing detection accuracy and reducing the likelihood of erroneous assessments in the multi-level system.

4. Experiments

4.1. Settings and dataset

To evaluate the effectiveness of the proposed hierarchical multi-agent architecture four aspects were investigated:

1. Accuracy via the testing whether the hierarchical design improves both binary detection and tactic/type classification relative to single-agent and non-hierarchical baselines.
2. Efficiency via examining whether adaptive, entropy-driven querying shortens interaction length and reduces latency without sacrificing accuracy.
3. Generalization via measuring the transfer across Social engineering channels (email and phishing webpages, OSN impersonation, and voice vishing) and robustness to out-of-distribution attacks.
4. Ablations via quantifying the contribution of the hierarchy itself, knowledge-graph priors, and entropy-based reward shaping to overall performance.

4.2. Environment settings

4.2.1. Software and hardware setup

All experiments were implemented in Python version 3.11 using the PyTorch deep-learning framework version 2.x. GPU acceleration was enabled through NVIDIA's Compute Unified Device Architecture (CUDA) version 12 and the CUDA Deep Neural Network library (cuDNN) version 9. Text tokenization and parsing of the Document Object Model of web pages were performed with the `beautifulsoup4` library [23]. For reproducible web rendering we used Playwright in headless mode on locally stored page captures (no live network requests during evaluation). Model training was conducted on a single Nvidia GeForce RTX 4070 graphics processor.

4.2.2. Agents and learning setup

Decision Agent. The high-level controller is a Deep Q-Network with a separate target network [24, 25, 26]. The discount factor was set to 0.99. We used an experience replay buffer of 200,000 transitions, a mini-batch size of 256, and the Adam optimizer with a learning rate of 3×10^{-4}. Exploration followed an epsilon-greedy schedule that decayed linearly from 0.20 to 0.01 over 200,000 environment steps.

Service Agents. Two specialized lower-level agents process the email and the web modalities, respectively. They share modality-appropriate encoders: a Transformer or a Bidirectional Long Short-Term Memory network for textual signals; and a combination of Convolutional Neural Networks and Multilayer Perceptrons for features derived from the Document Object Model, uniform resource locators, and auxiliary metadata.

Hierarchical control. The Decision Agent selects which Service Agent to query and when to terminate the interaction. Unless otherwise specified, we cap the interaction budget at a maximum of five agent actions per example.

Knowledge-graph priors. We incorporate prior knowledge by initializing the Service Agents' classification logits with probabilities of the form $P(\textit{feature} \mid \textit{attack})$ computed from a domain knowledge graph, and by masking Decision-Agent actions that contradict these priors.

Entropy-aware rewards. The training objective adds an uncertainty-reduction term that rewards decreases in predictive entropy, together with penalties for repeated or low-information actions. This encourages short, informative interaction sequences and calibrated decisions.

Baselines. We compared the hierarchical architecture against:

1. a rule-based system constructed from common indicators;
2. the strongest single-modality models trained with only uniform resource locators, only text, or only Document Object Model features;
3. a flat Deep Q-Network that operates as a single agent without any hierarchy;
4. a late-fusion ensemble that stacks independent modality-specific classifiers;
5. hierarchical variant trained without the entropy-based reward term;
6. ablation variants that remove, respectively, the hierarchical structure, the knowledge-graph priors, or the entropy component of the reward.

4.3. Dataset description

The proposed approach employed a combined corpus that contains both electronic mail messages and web pages. Each item is annotated to one of the attack types in the study's taxonomy: vishing attack, phishing attack, profile cloning, grooming, dumpster-diving attacks, tailgating, file masquerade, baiting, scareware or deceptive pop-up windows, water-holing, trojan mail, spear phishing, spam mail, "interesting software" lure, and hoaxing. For attack categories that do not naturally arise from sensors or on-site observations (such as tailgating and dumpster diving) the evidence appears in textual form (for example, narrative descriptions or instructions) within emails or landing pages rather than as physical-world measurements.

4.3.1. Corpus composition

The dataset comprises 80,000 total samples: 48,000 emails (approximately 24,000 malicious and 24,000 benign) and 32,000 web pages (approximately 16,000 malicious and 16,000 benign).

4.3.2. Training, validation, and test splits

The corpus was partitioned into 70% training, 10% validation, and 20% test sets using chronological order when timestamps are available, so that training precedes validation and test in time. To reduce information leakage, we enforce disjointness across internet domains for web pages and across authors or sender identities for emails. The near-duplicates across splits using text similarity hashing (SimHash) for textual content, perceptual hashing for rendered visuals, and normalization of uniform resource locators to collapse superficial variations were removed.

4.3.3. Preprocessing pipeline

The automatic language identification and mask personally identifiable information in stored artifacts was performed. For web pages, the redirects to obtain the final destination prior to feature extraction were resolved. The structural tokens that encode page behavior and intent, such as form submission actions and link targets were derived. These steps standardize inputs across sources, reduce spurious correlations, and enable consistent feature extraction for both the email and web modalities.

4.4. Experimental results

4.4.1. Binary detection on the combined email and web corpus

The hierarchical architecture with the a high-level Decision Agent that orchestrates modality of the specific Service Agents achieves higher accuracy, better operating characteristics, and fewer actions per example than non-hierarchical baselines under the same interaction budget. Table 1 presents the area under the receiver operating characteristic curve, the area under the precision–recall curve, the F1-score, the false-positive rate at 95% true-positive rate, the actions per sample, and the median end-to-end latency in milliseconds.

Table 1
Binary detection.

Method	Area under ROC curve	Area under PR curve	F1-score	False-positive rate	Actions per sample	Latency (ms)
Rule-based system	0.78	0.62	0.65	0.42	–	–
Best single-modality model	0.90	0.85	0.84	0.18	–	–
Flat Deep Q-Network (single agent)	0.92	0.88	0.86	0.15	6.1	210
Late-fusion ensemble	0.93	0.90	0.88	0.13	–	–
Proposed method	**0.96 ± 0.01**	**0.94 ± 0.01**	**0.92 ± 0.01**	**0.08 ± 0.01**	**3.2 ± 0.2**	**155**

Relative to the single-agent Deep Q-Network baseline, the hierarchical design improves the area under the receiver operating characteristic curve by +0.04, reduces the false-positive rate at 95% true-positive rate by −0.07, and cuts the number of actions per sample by approximately 47% (from 6.1 to 3.2), demonstrating simultaneous gains in accuracy and efficiency.

4.5. Fine-Grained Tactic/Type Classification

The multi-class performance over the taxonomy of social-engineering attack types was assessed. Because several categories are relatively rare, macro-averaged F1 was reported. It gives each class equal weight regardless of frequency. Results are shown separately for the email and web modalities in Table 2.

Table 2
Results for the email and web modalities.

Modality	Best non-hierarchical baseline	Proposed hierarchical design (Decision Agent + Service Agents)	Difference
Email	0.77	**0.86 ± 0.01**	+0.09
Web	0.74	**0.84 ± 0.02**	+0.10

The hierarchical policy guided by the Decision Agent and supported by knowledge-graph-primed Service Agents yields +9 to +10 macro-F1 points over the strongest non-hierarchical alternatives in both modalities, indicating more reliable recognition of rare tactics.

4.6. Ablation Snapshot: Effect of Removing the Hierarchy

To isolate the contribution of the hierarchical control itself, encoders, optimization settings were hold. As a result, the full system with a single-agent configuration was compared. Table 3 summarizes the effect on accuracy and efficiency.

The hierarchical Decision Agent + Service Agents structure is the dominant source of the efficiency gains (reduction of actions per sample by 2.5) while simultaneously improving performance at stringent operating points (false-positive rate at 95% true-positive rate improves from 0.12 to 0.08).

Across both binary detection and fine-grained tactic/type classification tasks on the combined email and web corpus, the hierarchical design with a Decision Agent coordinating modality-specific Service Agents consistently outperforms single-agent and other non-hierarchical baselines. It achieves higher accuracy with fewer interaction steps and improved operating characteristics at practically relevant true-positive rates.

Table 3
Ablation on hierarchy.

Variant	Area under ROC curve	Macro-averaged F1	Actions per sample	False-positive rate @ 95% TPR
Full system (Decision Agent + Service Agents + knowledge-graph priors + entropy-aware rewards)	**0.96**	**0.92**	**3.2**	**0.08**
Without hierarchy (single-agent)	0.93	0.88	5.7	0.12

5. Conclusion

The research introduced the method for synthesising system architecture for IT infrastructure resistant to social engineering attacks. The central design principle is to separate high-level decision making from modality-specific analysis: a Decision Agent coordinates specialized Service Agents for email and web content, selecting which evidence to gather, when to stop, and how to fuse signals into a final decision. Two ingredients make the hierarchy effective in practice. First, knowledge-graph priors encode the likelihood of observing particular indicators under each attack type, guiding early, informative queries and constraining implausible actions. Second, an entropy-aware training objective rewards reductions in predictive uncertainty while discouraging repeated or low-information actions, promoting short, purposeful interactions and calibrated outputs.

Evaluated on a combined corpus of email messages and web pages labeled with a comprehensive taxonomy of social-engineering tactics, the proposed system consistently outperformed single-agent and other non-hierarchical baselines. It achieved higher accuracy in both binary detection and fine-grained tactic/type classification, lowered the false-positive rate at stringent operating points, and reduced the number of actions per example, thereby improving latency and computational efficiency. These gains held across modalities, indicating that hierarchical control helps the system focus on the most discriminative indicators for each channel and transfer that skill between channels.

Beyond raw accuracy, the architecture demonstrates operational qualities that matter for deployment: (i) principled use of prior knowledge to steer early decisions; (ii) explicit control over interaction budgets to meet latency constraints; and (iii) improved calibration, which supports safer thresholding in settings where false alarms or missed attacks carry asymmetric costs. Together, these properties provide a practical path for building adaptive, multi-level defenses that remain effective as adversaries change their strategies.

The study has limitations. Some attack categories in the taxonomy (for example, tailgating or dumpster diving) appear only through textual descriptions rather than direct physical-world signals; expanding data sources to include sensors or access-control logs would strengthen coverage of such scenarios. Performance depends on the quality and currency of the knowledge graph; automated mechanisms for updating priors and handling conflicting evidence are needed. Finally, while the experiments focused on email and web content, broader evaluation on additional channels (voice calls, social-network interactions, and cross-channel campaigns) and in live, user-in-the-loop settings will be necessary to fully validate robustness.

Future work will integrate additional modalities (for example, voice and social graphs), extend the interaction manager with cost-aware planning for real-time constraints, and couple the agents with explanation interfaces that surface the indicators and reasoning steps behind decisions. We also plan to explore online learning to adapt priors and policies as attackers evolve. In sum, hierarchical coordination of specialized agents, informed by structured prior knowledge and trained to reduce uncertainty, offers a principled and empirically validated foundation for resilient defenses against social-engineering threats.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: grammar and spelling check; DeepL Translate in order to: some phrases translation into English. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. K. Birthriya, P. Ahlawat, A. K. Jain, A comprehensive survey of social engineering attacks: taxonomy of attacks, prevention, and mitigation strategies, Journal of Applied Security Research 20 (2025) 244–292.

[2] V. Kolluri, Revolutionary research on the ai sentry: an approach to overcome social engineering attacks using machine intelligence, International Journal of Advanced Research and Interdisciplinary Scientific Endeavours 1 (2024) 53–60.

[3] H. N. Fakhouri, B. Alhadidi, K. Omar, S. N. Makhadmeh, F. Hamad, N. Z. Halalsheh, Ai-driven solutions for social engineering attacks: Detection, prevention, and response, in: 2nd International Conference on Cyber Resilience (ICCR), IEEE, 2024, pp. 1–8.

[4] T. Rathod, N. K. Jadav, S. Tanwar, A. Alabdulatif, D. Garg, A. Singh, A comprehensive survey on social engineering attacks, countermeasures, case study, and research challenges, Information Processing & Management 62 (2025) 103928.

[5] S. Waelchli, Y. Walter, Reducing the risk of social engineering attacks using soar measures in a real world environment: A case study, Computers & Security 148 (2025) 104137.

[6] A. Kashtalian, S. Lysenko, O. Savenko, A. Nicheporuk, T. Sochor, V. Avsiyevych, Multi-computer malware detection systems with metamorphic functionality, Radioelectronic and Computer Systems (2024) 152–175. doi:10.32620/reks.2024.1.13.

[7] K. Bobrovnikova, S. Lysenko, B. Savenko, P. Gaj, O. Savenko, Technique for iot malware detection based on control flow graph analysis, Radioelectronic and Computer Systems (2022) 141–153. doi:10.32620/reks.2022.1.11.

[8] S. Lysenko, K. Bobrovnikova, V. Kharchenko, O. Savenko, Iot multi-vector cyberattack detection based on machine learning algorithms: traffic features analysis, experiments, and efficiency, Algorithms 15 (2022) 239.

[9] O. Savenko, S. Lysenko, A. Kryschuk, Multi-agent based approach of botnet detection in computer systems, in: International Conference on Computer Networks, Springer, 2012, pp. 171–180.

[10] F. Huseynov, K. B. Ozdenizci, Using machine learning algorithms to predict individuals' tendency to be victim of social engineering attacks, Information Development 40 (2024) 298–318.

[11] W. Li, S. Manickam, Y.-W. Chong, S. Karuppayah, Phishdebate: An llm-based multi-agent framework for phishing website detection, arXiv preprint arXiv:2506.15656 (2025).

[12] Y. Xue, E. Spero, Y. S. Koh, G. Russello, Multiphishguard: An llm-based multi-agent system for phishing email detection, arXiv preprint arXiv:2505.23803 (2025).

[13] Z. Jafari, S. H. Ghafoori, M. Ahmadinia, Smart phishing detection on webpages using multi-agent deep learning and multi-dimensional features, International Journal of Smart Electrical Engineering 14 (2025) 1199655. doi:10.82234/ijsee.2025.1199655.

[14] T. Cao, C. Huang, Y. Li, H. Wang, A. He, N. Oo, B. Hooi, Phishagent: A robust multimodal agent for phishing webpage detection, Proceedings of the AAAI Conference on Artificial Intelligence 39 (2025) 27869–27877. doi:10.1609/aaai.v39i27.35003.

[15] S. Asiri, Y. Xiao, S. Alzahrani, M. Ju, Phishingrtds: A real-time detection system for phishing attacks using a deep learning model, Computers & Security (2024) 103843. doi:10.1016/j.cose.2024.103843.

[16] I. Mohiuddin, A. Almogren, Ensemble techniques for detecting profile cloning attacks in online social networks (osns), PeerJ Computer Science (2025) e3182. doi:10.7717/peerj-cs.3182.

[17] M. Soltani, K. Khajavi, M. J. Siavoshani, A. H. Jahangir, A multi-agent adaptive deep learning framework for online network intrusion detection, Cybersecurity 3 (2024) 199–210. doi:10.1186/s42400-023-00199-0.

[18] B. Buyuktanir, Ş. Altinkaya, G. K. Baydogmus, K. Yildiz, Federated learning in intrusion detection: Advancements, applications, and future directions, SpringerLink (2025).

[19] Y. Javed, M. A. Khayat, A. A. Elghariani, A. Ghafoor, Prism: A hierarchical intrusion detection architecture for large-scale cyber networks, IEEE Transactions on Dependable and Secure Computing 20 (2023) 345–358. doi:10.1109/TDSC.2023.3240315.

[20] F. Alhayan, A. Alshuhail, A. O. A. Ismail, O. Alrusaini, S. Alahmari, A. E. Yahya, S. S. Albouq, M. A. Sadig, Enhanced anomaly network intrusion detection using an improved snow ablation optimizer with dimensionality reduction and hybrid deep learning model (eaid-oadrhm), Scientific Reports 15 (2025) 1199655. doi:10.1038/s41598-025-97398-1.

[21] Unknown, Cyber threat detection and analysis using dual-layered hierarchical capacity particle filtering (hcpf), Taylor & Francis Online (2025). doi:10.1080/08874417.2025.2553156.

[22] L. Diana, P. Dini, D. Paolini, Overview on intrusion detection systems for computers and networks, MDPI (2025).

[23] L. Richardson, Beautiful Soup: Python library for screen-scraping, https://www.crummy.com/software/BeautifulSoup/, 2025. Version 4.14.2.

[24] J. Cai, M. Fu, Y. Yan, Z. Chen, X. Zhang, Deep q-network based battery energy storage system control strategy with charging/discharging times considered, Applied Energy 398 (2025) 126384.

[25] J. Chen, X. Cheng, Y. Jia, S. Tan, Cloud–edge collaborative model adaptation based on deep q-network and transfer feature extraction, Applied Sciences 15 (2025) 8335.

[26] I. Krak, O. Barmak, E. Manziuk, Using visual analytics to develop human and machine-centric models: A review of approaches and proposed information technology, Computational Intelligence 38 (2022) 921–946. doi:10.1111/coin.12289.