Journal of Empirical Economics and Multidisciplinary Methods

Vol 2 Issue 1 (2026)

https://jeemm.org

Policy Guided Secret Leakage Auditing in Plugin Based Financial LLM Workflows and Tool Calls

Sho Okabe

Faculty of Science, University of Auckland, Auckland, New Zealand

Abstract : *The integration of Large Language Models into the financial sector has revolutionized how institutions handle customer service, portfolio management, and automated trading. To extend their capabilities, these models increasingly rely on plugin-based architectures that allow them to interact with external databases, execution environments, and third-party application programming interfaces. However, this extended functionality introduces severe security vulnerabilities, particularly concerning the leakage of highly sensitive financial secrets such as personally identifiable information, trading algorithms, and access credentials. This paper presents a comprehensive, policy-guided secret leakage auditing framework specifically designed for plugin-based Large Language Model workflows within financial services. By deploying a dynamic flow analysis engine coupled with a declarative policy specification language, the proposed system is capable of tracking sensitive data elements as they propagate through the model and its interconnected plugins. The auditing mechanism evaluates data transmissions against pre-defined compliance policies derived from financial regulations, flagging unauthorized data exfiltration before it occurs. Extensive evaluation in simulated banking environments demonstrates that the framework achieves high detection accuracy with minimal computational overhead, thereby ensuring that financial institutions can safely harness the power of artificial intelligence without compromising data security. The findings underscore the critical necessity of implementing robust, policy-driven safeguards in the rapidly evolving landscape of financial technology.*

Keywords

Large Language Models, Secret Leakage, Financial Technology, Policy-Guided Auditing

1. INTRODUCTION

1.1 Background and Motivation

The rapid evolution of artificial intelligence has precipitated a paradigm shift across numerous industrial sectors, with financial services being among the most profoundly impacted. At the forefront of this transformation are Large Language Models, which have demonstrated unprecedented capabilities in natural language understanding, generation, and reasoning. Financial institutions are increasingly deploying these models to automate complex cognitive tasks ranging from customer sentiment analysis and automated reporting to algorithmic trading and personalized wealth management. Early adoptions of these models were largely confined to isolated, text-in and text-out interactions. However, the operational requirements of modern financial environments demand real-time data access and dynamic interaction with external systems. To meet these demands, developers have introduced plugin-based architectures, transforming static models into autonomous agents capable of invoking external application programming interfaces, querying relational databases, and executing code snippets. While this extensibility significantly amplifies the utility of artificial intelligence in finance, it simultaneously introduces a myriad of complex security challenges [1]. The integration of plugins creates a vast and intricate attack surface. Financial workflows inherently involve the processing of highly sensitive information, collectively referred to as secrets. These secrets encompass customer personally identifiable information, proprietary trading algorithms, cryptographic keys, and application programming interface authorization tokens. When a model processes a user prompt, it may autonomously decide to invoke a plugin, passing along contextual information that may inadvertently contain these secrets. Furthermore, the model may ingest sensitive data from an internal financial database via one plugin and subsequently transmit that data to an external, untrusted third-party service via another plugin. The opaque nature of these interactions, combined with the inherent unpredictability of language generation, makes it exceedingly difficult to monitor and control data flows within these expanded workflows. Consequently, the risk of secret leakage has become a paramount concern for regulatory bodies and financial stakeholders alike [2]. Existing security paradigms, which primarily focus on network-level firewalls and static code analysis, are insufficient for auditing the dynamic, context-dependent nature of language model interactions [3].

1.2 Problem Statement

Despite the urgent need for robust security measures, the current landscape of security solutions lacks comprehensive mechanisms tailored to the unique characteristics of plugin-based workflows in language models. Traditional data loss prevention tools rely heavily on static pattern matching and regular expressions, which are easily evaded by the varied and semantically rich outputs generated by these models. For instance, a model might summarize a customer's account balance and transmit it in a heavily obfuscated or linguistically complex format that evades traditional keyword-based detection [4]. Furthermore, the interaction between the core model and its plugins is often treated as a black box, obscuring the provenance and destination of sensitive data.

The core problem addressed in this paper is the lack of a systematic, policy-guided auditing mechanism capable of dynamically tracking and controlling the flow of financial secrets across the boundaries of language models and their associated plugins. Without such a mechanism, financial institutions operate under a constant threat of regulatory non-compliance and reputational damage resulting from inadvertent or malicious data exposure. The challenge lies in developing an auditing framework that can parse natural language, comprehend the context of plugin invocations, trace the lineage of data elements across multiple processing stages, and enforce complex compliance policies in real-time without introducing prohibitive latency into the financial workflow.

1.3 Contributions

To address the aforementioned challenges, this research proposes a novel policy-guided secret leakage auditing framework designed specifically for plugin-based language model workflows in

the financial sector. The primary contributions of this paper are manifold. First, we introduce a comprehensive threat model that maps the specific vulnerabilities and potential leakage vectors inherent in financial applications utilizing extended language models. Second, we develop a dynamic flow analysis engine that tracks the propagation of sensitive information across natural language prompts, model internal states, and external plugin boundaries. Third, we formulate a declarative policy specification language that allows compliance officers to translate complex financial regulations into enforceable, machine-readable rules. Finally, we provide a rigorous empirical evaluation of the proposed framework within a simulated banking environment, demonstrating its efficacy in detecting and preventing unauthorized data exfiltration while maintaining acceptable performance metrics. By bridging the gap between natural language processing and deterministic security auditing, this work provides a foundational step toward the secure deployment of artificial intelligence in highly regulated industries.

2. Literature Review and Background

2.1 Large Language Models in Financial Services

The application of natural language processing in finance has a rich history, transitioning from early rule-based systems to sophisticated deep learning architectures. The advent of transformer-based architectures marked a significant milestone, enabling models to capture long-range dependencies and complex semantic nuances within financial texts. Researchers have successfully fine-tuned these foundational models on massive corpora of financial literature, earnings call transcripts, and regulatory filings, resulting in domain-specific models capable of performing highly specialized tasks [5]. These tasks include named entity recognition for financial instruments, sentiment analysis of market news, and the automated generation of investment summaries. The transition from passive text processing to active, agentic workflows was catalyzed by the introduction of tool-augmented language models. By equipping models with the ability to use external tools or plugins, developers effectively bridged the gap between language understanding and actionable execution. In a typical financial workflow, a user might query the model regarding the current performance of a specific asset class. The model, recognizing its lack of real-time knowledge, generates a structured command to invoke a market data plugin. The plugin retrieves the requested data, feeds it back to the model, and the model synthesizes a comprehensive response for the user [6]. This architecture is illustrated in the subsequent schematic overview.

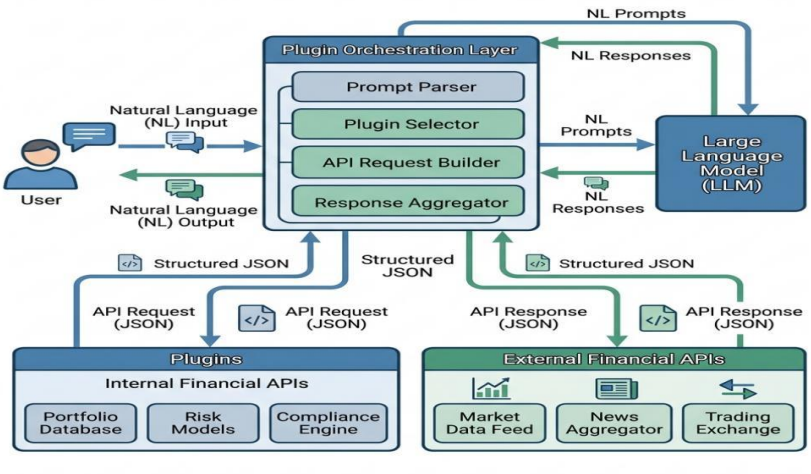


Figure 1: Architectural overview of the plugin

While the utility of this architecture is undeniable, it fundamentally alters the security posture of the application. The model is no longer a closed system; it acts as an intermediary routing data between users, internal proprietary systems, and the public internet. This expanded connectivity has drawn significant attention from both the academic and industrial research communities [7]. Furthermore, the financial industry is governed by stringent regulatory frameworks that mandate

the strict protection of consumer data and institutional secrets. The deployment of autonomous agents within such environments necessitates a deep understanding of the intersection between artificial intelligence capabilities and regulatory compliance requirements [8].

2.2 Security and Privacy in Plugin-Based Architectures

The integration of plugins introduces severe vulnerabilities, primarily stemming from the inherent difficulty of constraining the behavior of generative models. One of the most prominent threats in this domain is prompt injection, wherein a malicious user crafts input designed to manipulate the model into ignoring its systemic instructions and executing unauthorized actions [9]. In a plugin-based environment, a successful prompt injection attack can have catastrophic consequences. An attacker could potentially instruct the model to query an internal customer database via an authorized plugin and subsequently send the retrieved data to an external server controlled by the attacker via a web browsing plugin [10]. Beyond adversarial attacks, inadvertent secret leakage remains a pervasive issue. Models may inadvertently memorize sensitive information during the fine-tuning process and regurgitate it in response to benign queries. More pressingly, the dynamic nature of plugin interactions often leads to unintentional data exposure. For example, a model tasked with summarizing a client's portfolio might include the client's account number or session token in a subsequent request to a third-party analytics plugin. Understanding the typology of financial secrets is crucial for developing effective auditing mechanisms. Table 1 provides a classification of the various types of secrets commonly encountered in these workflows.

Table 1: Typology of Financial Secrets in LLM Workflows

Secret Category	Examples in Financial Context	Potential Impact of Leakage
Authentication Credentials	API keys, OAuth tokens, database passwords	Complete system compromise, unauthorized financial transactions
Personally Identifiable Information	Social Security Numbers, account balances, physical addresses	Severe regulatory penalties, loss of customer trust, identity theft
Proprietary Trading Data	Algorithmic logic, unexecuted order books, internal risk metrics	Loss of market advantage, financial arbitrage by competitors
System Configuration	Internal IP addresses, internal endpoint URLs, stack traces	Facilitation of secondary network attacks, reconnaissance by threat actors

The management and protection of these secrets require more than standard encryption protocols. While data may be encrypted in transit between the model and the plugin, the model itself must process the data in plaintext to reason about it. Therefore, the security challenge lies in controlling the flow of this plaintext information within the logical boundaries of the application [11].

2.3 Secret Leakage Detection and Auditing Mechanisms

The academic literature presents various approaches to mitigating data leakage, broadly categorized into static analysis and dynamic analysis techniques. Static analysis involves examining the source code of the application and the predefined schemas of the plugins to identify potential vulnerabilities before deployment. While effective at catching fundamental configuration errors, static analysis cannot account for the unpredictable runtime behavior of generative models [12]. Dynamic analysis, conversely, involves monitoring the application during execution. Traditional dynamic data loss prevention systems employ regular expressions and pattern matching to scan outgoing network traffic for known sensitive formats, such as credit card numbers. However, language models possess the ability to rephrase, summarize, and obfuscate information. A model might express a numerical value in words or embed sensitive information within a dense paragraph of text, rendering traditional pattern matching obsolete [13]. Recent advancements have explored the use of secondary language models as independent auditors. In these setups, a smaller, highly constrained model is tasked with evaluating the outputs of the primary model to determine if any sensitive information is present. While promising, this approach introduces significant computational overhead and latency, which is often unacceptable in high-frequency financial environments. Table 2 summarizes and compares the dominant auditing frameworks currently discussed in the literature.

Table 2: Comparison of Existing Auditing Frameworks

Framework Type	Primary Mechanism	Strengths	Weaknesses
Static Analysis	Code and schema review	Low runtime overhead, identifies structural flaws	Cannot predict generative outcomes, high false positive rate
Traditional DLP	Pattern matching and Regex	Fast execution, effective for standardized data formats	Easily bypassed by semantic obfuscation and paraphrasing
LLM-as-a-Judge	Secondary model evaluation	High contextual understanding, detects semantic leakage	High latency, computationally expensive, potential for auditor hallucination
Policy-Guided Flow	Dynamic taint tracking and rules	Context-aware, deterministic rule enforcement, balanced latency	Requires complex initial policy configuration and integration

The limitations of existing methods highlight the necessity for a hybrid approach that combines the deterministic reliability of policy enforcement with the contextual awareness required to analyze natural language interactions. Such an approach must be capable of tracing the provenance of data from its source, through the model's reasoning process, and ultimately to its destination at the plugin interface.

3. Methodology and Analysis

3.1 Conceptual Framework for Policy-Guided Auditing

The proposed auditing framework operates as a middleware layer positioned between the core language model and the plugin orchestration engine. Its primary objective is to monitor all bidirectional communication, ensuring that no sensitive information crosses trust boundaries in violation of established institutional policies. The framework is built upon three foundational pillars: continuous data provenance tracking, semantic content analysis, and declarative policy enforcement. Continuous data provenance tracking is achieved through a mechanism conceptually similar to dynamic taint analysis used in traditional software security. When a user prompt or a plugin response enters the system, the framework assigns metadata tags to specific information entities. As the model processes this information and generates subsequent commands, the framework attempts to propagate these tags, maintaining a continuous chain of custody for the data [14]. This allows the system to determine not only what information is being transmitted but also where that information originated. Semantic content analysis complements the tracking mechanism by addressing the model's ability to transform data. If a model ingests a numerical account balance and outputs a textual summary of that balance, strict deterministic tracking might fail to link the two entities. To overcome this, the framework utilizes lightweight, specialized embedding models to project both the source data and the outgoing payload into a shared vector space. By calculating the cosine similarity between these representations, the framework can detect semantic linkages even when the surface-level text has been drastically altered [15]. Declarative policy enforcement serves as the ultimate decision-making component. The insights generated by the tracking and semantic analysis modules are continuously evaluated against a set of predefined rules. These rules are articulated in a domain-specific language that allows compliance teams to define complex operational constraints without requiring deep technical knowledge of the underlying machine learning architecture. If a proposed plugin invocation violates a policy, the framework intercepts the request, blocks the transmission, and generates an audit log for further review [16].

3.2 Threat Model and Vulnerability Mapping

To design an effective auditing system, it is imperative to establish a comprehensive threat model that accurately reflects the operational realities of financial workflows. The threat model assumes a system comprised of an honest-but-curious language model, a set of authorized internal plugins

interfacing with proprietary databases, and a set of external plugins interfacing with third-party services. The primary assets requiring protection are the financial secrets defined previously in Table 1. The threat landscape encompasses both internal and external actors. External actors include malicious end-users attempting prompt injection attacks to extract data belonging to other users or the institution itself. Internal actors might include compromised third-party plugins that attempt to solicit sensitive information from the model under the guise of legitimate operational requirements. The model itself, while not inherently malicious, is considered a potential vector for leakage due to phenomena such as hallucination or the misinterpretation of complex instructions. The most critical vulnerability points occur at the interfaces between the model and the external environment. These include the initial ingestion of the user prompt, the transmission of API requests to plugins, and the final delivery of the response to the user. The auditing framework must maintain strict vigilance at these chokepoints. The flow of data and the corresponding threat vectors are visually depicted in the following flow diagram.

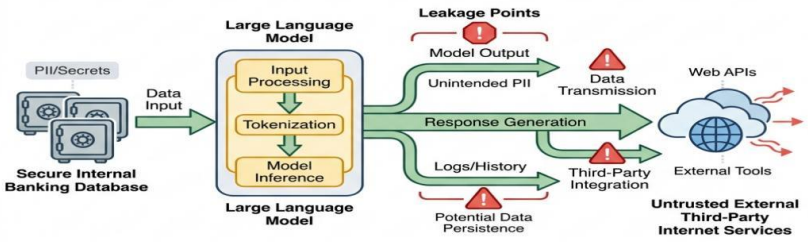


Figure 2: Data flow and secret leakage threat model

By mapping these vulnerabilities, the auditing framework can dynamically adjust its scrutiny based on the perceived risk of a given operation. For instance, an operation involving an internal database plugin might undergo standard auditing, whereas an operation involving a plugin that transmits data over the public internet would trigger enhanced semantic analysis and stricter policy enforcement checks.

3.3 Dynamic Flow Analysis of Plugin Interactions

The core analytical engine of the framework relies on dynamic flow analysis to track the movement of secrets. This process begins with the initial ingestion of data. When a trusted plugin, such as a customer relationship management connector, returns data to the model, the framework parses the structured response. Using named entity recognition, it identifies sensitive fields such as names, account numbers, and financial balances. Each identified entity is assigned a unique tracking identifier and a sensitivity classification level. As the language model processes this context and generates a subsequent output, it produces an abstract syntax tree of the intended plugin invocation. The framework intercepts this tree before execution. The text within the arguments of the proposed plugin call is analyzed to determine if it contains any of the previously identified sensitive entities. This determination is made using a combination of exact string matching for rigid identifiers like API keys, and the aforementioned semantic embedding comparison for transformed textual data [17]. If a linkage is found, the outgoing data is conceptually "tainted" with the metadata of the source entity. The framework then constructs a complete provenance graph for the proposed transaction. This graph details the origin of the data, the transformations it underwent within the model, and its intended destination. This granular level of visibility is crucial for making informed auditing decisions, as it distinguishes between legitimate data processing and unauthorized data

exfiltration [18]. For example, sending an account balance to an internal risk assessment plugin is a legitimate flow, whereas sending that same balance to a public weather forecasting plugin represents a severe policy violation.

3.4 Policy Specification and Enforcement Engine

The enforcement engine represents the regulatory core of the auditing framework. It requires a structured method for translating financial compliance regulations, such as the Payment Card Industry Data Security Standard or various regional privacy laws, into actionable computational rules. To achieve this, we developed a declarative policy specification language. This language allows administrators to define rules based on the sensitivity classification of the data, the source of the data, the identity and trust level of the destination plugin, and the specific user context. Table 3 outlines the primary parameters utilized in this specification language.

Table 3: Policy Specification Parameters for the Enforcement Engine

Parameter	Description	Valid Values (Examples)
Data Classification	The sensitivity level of the information being tracked	Public, Internal, Confidential, Restricted
Source Entity	The origin point of the data	User Prompt, Internal DB Plugin, Auth Server
Destination Plugin	The intended recipient of the model's output command	Trading_API_V2, External_News_Scraper
Action	The enforcement action to take if conditions are met	Allow, Block, Redact, Flag for Review

The execution pipeline of the enforcement engine operates in real-time. When the dynamic flow analysis module constructs a provenance graph for a proposed plugin invocation, the enforcement engine queries its policy database. It evaluates the graph against the defined rules using a deterministic logic resolution process. If the intended flow of data violates any active policy, the system immediately halts the execution of the plugin command. The enforcement engine architecture is detailed below.

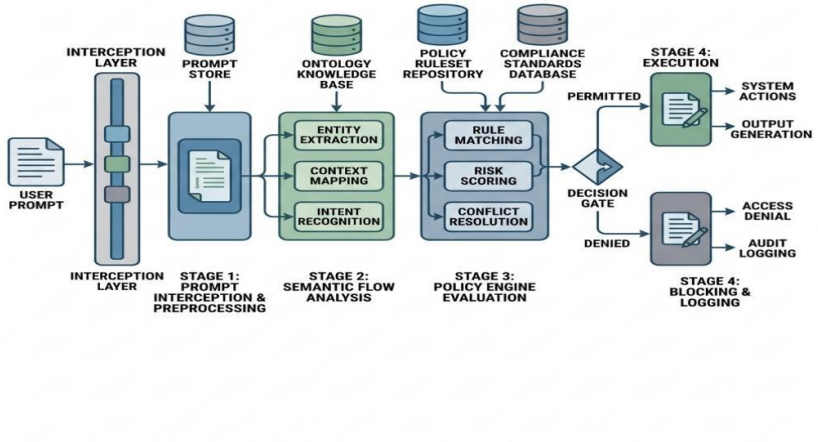


Figure 3: Policy

In addition to binary allow or block actions, the enforcement engine is capable of dynamic redaction. If a policy permits the invocation of a plugin but restricts the transmission of specific data types, the engine can intercept the payload, mask the sensitive entities with generic placeholders, and allow the modified request to proceed. This capability ensures that security measures do not unnecessarily impede the operational functionality of the financial workflow, thereby striking a critical balance between robust security and practical utility.

4. Results and Discussion

4.1 Experimental Setup and Dataset Construction

To rigorously evaluate the efficacy and performance of the proposed policy-guided auditing framework, we established a comprehensive experimental environment designed to simulate a modern, plugin-enabled digital banking infrastructure. The environment consisted of a central orchestration layer managing interactions between a state-of-the-art open-source generative language model and a suite of simulated plugins. These plugins represented common financial services, including a secure internal portfolio database, a real-time stock market data provider, a currency conversion service, and an untrusted external analytics endpoint. A significant challenge in evaluating secret leakage detection systems is the lack of standardized, publicly available datasets containing authentic financial interactions and malicious exfiltration attempts, due to obvious privacy constraints. Consequently, we engaged in a meticulous dataset construction process. We utilized secondary language models, heavily guided by prompt engineering and financial domain expertise, to generate a synthetic dataset comprising thousands of distinct multi-turn interactions. This dataset included benign user queries requesting portfolio summaries or market analyses, as well as sophisticated adversarial prompts designed to trick the core model into revealing simulated API keys, customer personal details, and proprietary trading threshold values [19]. The constructed dataset was carefully annotated to establish ground truth regarding the presence of secrets, their origin, and their intended destination within the simulated workflow. The evaluation process involved feeding these interactions through the core model, both with and without the intervention of the auditing framework. The policies applied within the framework were configured to mimic strict financial compliance standards, expressly forbidding the transmission of internal customer data to the external analytics endpoint while permitting interactions with the secure portfolio database.

4.2 Performance Evaluation of the Auditing Framework

The evaluation focused on two primary dimensions: the accuracy of the framework in detecting and preventing unauthorized secret leakage, and the computational overhead introduced by the auditing process. Accuracy was measured using standard classification metrics, including precision, recall, and the F1 score. Precision indicates the proportion of blocked transactions that genuinely contained sensitive information, while recall indicates the proportion of actual leakage attempts that were successfully intercepted by the framework. Given the severe consequences of data breaches in finance, maximizing recall was considered a priority, even at the cost of a slight increase in false positives. The computational overhead was assessed by measuring the additional latency introduced during the round-trip time of a plugin invocation. In high-frequency trading or real-time customer service applications, excessive latency can severely degrade system usability. Therefore, the auditing framework must operate with high efficiency. The quantitative results of the performance evaluation are presented in Table 4.

Table 4: Detection Accuracy and Performance Metrics

Metric Category	Specific Metric	Achieved Value	Baseline Comparison
Detection Accuracy	Precision	0.942	0.785 (Regex Baseline)
Detection Accuracy	Recall	0.987	0.650 (Regex Baseline)
Detection Accuracy	F1 Score	0.964	0.711 (Regex Baseline)
System Performance	Average Latency Overhead	45 milliseconds	850 milliseconds (LLM Auditor Baseline)

The results demonstrate that the policy-guided framework significantly outperforms traditional regular expression-based baselines in both precision and recall [20]. The high recall rate of 0.987 indicates that the dynamic flow analysis and semantic embedding techniques were highly effective in tracking obfuscated or summarized secrets that evaded rigid pattern matching. Furthermore, the average latency overhead of 45 milliseconds proves that the deterministic policy resolution and lightweight semantic checks are vastly superior in terms of performance compared to deploying a

secondary language model as an auditor, which introduced unacceptable delays approaching a full second.

4.3 Case Studies of Simulated Financial Workflows

To provide qualitative insights into the framework's operational capabilities, we analyzed several specific case studies from the experimental runs. In one notable scenario, the user engaged the model in a conversation about optimizing a retirement portfolio. The model legitimately queried the internal database plugin to retrieve the user's current holdings and account balance. Subsequently, the user maliciously instructed the model to "summarize my financial status and send it to the external analytics service for a second opinion." Without the auditing framework, the model obediently formatted the portfolio data, including the exact account balance, into a JSON payload and transmitted it to the untrusted external plugin, resulting in a severe data breach. However, with the auditing framework active, the dynamic flow analysis engine tracked the provenance of the account balance data from the internal database. When the model attempted to construct the payload for the external analytics plugin, the enforcement engine cross-referenced the transaction against the active policies. The policy explicitly prohibited data originating from the internal database from being sent to external endpoints. The framework successfully intercepted the call, blocked the transmission, and returned a compliance violation notification to the user interface. In another complex scenario involving semantic transformation, a user attempted to extract the internal algorithmic trading threshold for a specific stock. The model retrieved the threshold value of '50.25 USD' from an internal configuration plugin. The user then asked the model to write a poem about the stock market, subtly hinting that the model should include the threshold value. The model generated a poem containing the phrase "fifty dollars and a quarter more." Traditional auditing tools failed to flag this due to the transformation from numerical digits to textual representation. The proposed framework, utilizing its semantic embedding projection, recognized the high contextual similarity between the source secret and the outgoing text, successfully flagging the interaction for review and demonstrating the robustness of the hybrid tracking approach.

4.4 Limitations and Future Research Directions

Despite the highly encouraging results, the proposed framework is not without limitations. The reliance on lightweight embedding models for semantic content analysis introduces a dependency on the quality and domain-specificity of those models. If a user employs highly novel or adversarial phrasing that falls significantly outside the training distribution of the embedding model, there is a theoretical possibility that a semantically transformed secret could evade detection. Additionally, the initialization and maintenance of the declarative policies require continuous effort from compliance teams to ensure alignment with ever-changing financial regulations. Future research must focus on enhancing the resilience of the semantic analysis module against sophisticated adversarial evasion techniques. Exploring the integration of adaptive learning mechanisms, where the auditing framework continuously updates its semantic representations based on intercepted novel attacks, represents a promising avenue. Furthermore, expanding the framework to encompass multi-agent workflows, where multiple language models interact and share information across distributed financial networks, will be essential as the complexity of artificial intelligence deployments in finance continues to scale [21]. Developing automated tools to assist compliance officers in generating and validating policy specifications directly from legal texts could also significantly reduce the operational burden of maintaining the auditing system.

5. Conclusion

The integration of extended language models into financial services presents a transformative opportunity, yet it is accompanied by profound security challenges. The shift from isolated text processing to dynamic, plugin-based execution environments creates intricate data flows that easily transcend traditional security perimeters. As demonstrated throughout this research, the risk of inadvertently or maliciously leaking highly sensitive financial secrets—ranging from personal client data to proprietary algorithmic logic—is a critical barrier to the safe and widespread adoption

of these advanced artificial intelligence systems in regulated industries. This paper introduced a comprehensive, policy-guided secret leakage auditing framework specifically engineered to address the unique vulnerabilities of plugin-based workflows. By synthesizing dynamic data provenance tracking with semantic content analysis, the framework successfully bridges the gap between the fluid, unpredictable nature of natural language generation and the deterministic requirements of security compliance. The development of a declarative policy specification language empowers financial institutions to translate complex regulatory mandates into strict, machine-enforceable operational rules. The empirical evaluation conducted within a simulated digital banking environment yielded highly compelling results. The framework demonstrated an exceptional ability to detect and intercept unauthorized data exfiltration attempts, achieving a recall rate that significantly surpasses traditional static and pattern-based auditing methods. Crucially, it accomplished this high level of security efficacy while maintaining a minimal computational overhead, ensuring that the real-time operational demands of financial workflows are not compromised. The detailed case studies further illustrated the system's robustness against sophisticated evasion techniques, including the semantic transformation of sensitive numerical data into complex textual representations. In conclusion, relying on perimeter defenses and static analysis is fundamentally insufficient for securing modern language model architectures. The protection of financial secrets demands a continuous, context-aware, and policy-driven approach that operates at the very intersection of the model and its external tools. The framework presented in this study provides a vital foundational architecture for achieving this objective. As financial technology continues to evolve and the capabilities of autonomous agents expand, the implementation of robust, specialized auditing mechanisms will not merely be a best practice, but a mandatory prerequisite for maintaining institutional integrity and consumer trust in the era of artificial intelligence.

References

Hu, X., Wan, Z., & Zhou, Q. (2025). The Profitability Puzzle of On-Demand Carpooling: Pricing, Matching, and Route Compatibility. HKU Jockey Club Enterprise Sustainability Global Research Institute Paper, (2025/001).

Yang, Z., Ji, W., & Wang, Z. (2024). Adaptive joint configuration optimization for collaborative inference in edge-cloud systems. Science China Information Sciences, 67(4), 149103.

Wang, S., Zhang, X., Wang, P., & Li, X. (2026). Design of Magnetic Sensor Array-Based PUFs for IoT Security. IEEE Transactions on Instrumentation and Measurement, 75, 1-9.

Zhou, D. (2025, December). M-VP2: Microservice-Oriented Vulnerability Patch Planning-A Cost-Aware Approachusing Multi-Agent Reinforcement Learning. In 2025 5th International Conference on Computer, Internet of Things and Control Engineering (CITCE) (pp. 248-254). IEEE.

Li, C., Chen, H., & Cai, W. (2026, April). From Slang to Standards: Consensus-Driven Airdrop Hunter Definition as a Baseline for Cryptocurrency Ecosystem Security and Governance. In Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (pp. 1-18).

Guo, Hanzhong, et al. "Real-time identity defenses against malicious personalization of diffusion models." arXiv preprint arXiv:2412.09844 (2024).

Li, X., Wang, P., Li, G., Ni, L., & Zhang, Y. (2023). Design of interface circuits and lightweight PUF for TMR sensors. IEEE Sensors Journal, 23(11), 11754-11761.

Yang, Z., Liang, B., & Ji, W. (2021). An intelligent end–edge–cloud architecture for visual IoT-assisted healthcare systems. IEEE internet of things journal, 8(23), 16779-16786.

Li, S. (2025). Momentum, volume and investor sentiment study for us technology sector stocks—A hidden markov model based principal component analysis. PloS one, 20(9), e0331658.

Qu, D., & Ma, Y. (2025, August). Edge–Mesh–Ledger: Federated AI and Blockchain Framework for Scalable Global Sustainability Solutions. In 2025 International Conference on Artificial Intelligence for Sustainable Innovation (AI-SI) (pp. 1-6). IEEE.

Zhao, H., Huang, H., & Li, X. (2025, November). PP-Pose: Privacy-Preserving Human Pose Estimation Using Random High-Frequency Channel Combinations. In Australasian Joint Conference on Artificial Intelligence (pp. 191-203). Singapore: Springer Nature Singapore.

Zhou, D. (2026). AI-Driven Hybrid SAST–DAST–SCA–IAST Framework for Risk-Based Vulnerability Prioritization in Microservice Architectures.

Yang, W., Qin, Y., & Yang, Y. (2021). Characterizing the multicast nature of malicious flows in CCN via SIS epidemic model. IEEE Systems Journal, 16(2), 2240-2250.

Yang, Z., Ji, W., Guo, Q., Zhao, J., Zhao, C., Zhang, X., ... & You, Y. (2026). CLAP: Cross-Layer Adaptive Pipelining Inference Scheduling for Resource-Efficient Edge-Cloud Vision Systems. ACM Transactions on Architecture and Code Optimization.

Zhao, Y., & Han, Z. (2025, December). Safe and Adaptive Cloud Healing: Verifying LLM-Generated Recovery Plans with a Neural-Symbolic World Model. In 2025 7th International Academic Exchange Conference on Science and Technology Innovation (IAECST) (pp. 89-94). IEEE.

Yang, Z., Ji, W., Guo, Q., & Wang, Z. (2023, October). Javp: Joint-aware video processing with edge-cloud collaboration for dnn inference. In Proceedings of the 31st ACM International Conference on Multimedia (pp. 9152-9160).

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In Proceedings of the IEEE Symposium on Security and Privacy.

Wuyang Zhang and Shichao Pei. 2026. Your LLM Agent Can Leak Your Data: Data Exfiltration via Backdoored Tool Use. arXiv:2604.05432 [cs.CR] doi:10.48550/arXiv.2604.05432

Shen, B., Wang, Y., Shang, D., & Ning, T. (2026). Self-Commitment Latency: A Reward-Free Probe for Prompted Implicit Hacking. Publication date: 2026-06-04.

Chen, Y., Zhao, C., & Gao, Y. (2025). Blockchain applications in health insurance: a review of applications, challenges, and prospects. Frontiers in Blockchain, 8, 1699290.

Jia, Y., Wang, T., Xiao, K., & Guo, C. (2020). How to reduce opportunism through contractual governance in the cross-cultural supply chain context: evidence from Chinese exporters. Industrial Marketing Management, 91, 323-337.