

FULL PAPER

Security Risk Assessment and Layered Protection Strategies for Large Language Model Banking Chatbots with Privacy Considerations

Prepared by

Galal eldin Abbas Eltayeb
Department of Management
Information Systems
College of Business and
Economics
Qassim University
Buraydah, Saudi Arabia
g.eltayeb@qu.edu.sa
<https://orcid.org/0000-0003-3778-2061>

Abdalilah Alhalangy
Department of Computer
Engineering
College of Computer
Qassim University
Buraydah, Saudi Arabia
a.alhalangy@qu.edu.sa
<https://orcid.org/0000-0003-2735-8208>

Abstract

But now, given the AI revolution and increased interest in bringing virtual agents and assistants to life banks too are testing LLM-powered AI agents that may assist customers, explain and customize products as well as simplify operational work done by bank employees in the background. But similar systems are susceptible to prompt injection, insecure output handling, and other LLM-specific threats that had only become more prevalent since these publications. Existing surveys and frameworks survey the generic security space of LLMs but do not propose reach an end-to-end, banking-specific threat model nor deployable defense architecture for assistants in line with systems from core into the edge. We also present a data-privacy-aware threat model and a learnable multi-layer defense framework for LLM-based banking assistants. We showcase a reference architecture for an omnichannel banking chatbots built using an instruction-tuned, medium-scale commercial LLM and also access to tools for accessing customer data, payments and knowledge bases. Assets, trust boundaries, adversary capabilities, and a banking-related threat taxonomy based on OWASP LLM guidance in combination with the NIST AI Risk Management Framework and recent sector-wide reports are defined. We then present a defense-in-depth framework with which we employ: governing inputs, prompt level controls, policy enforcement in the orchestration layer, safety guards at the tool-layer, structured output filtering along with response-risk scoring and constant monitoring and governance. We introduce a prototype framework for an effective LLM API designed to mimic digital banking assistant responses and evaluate against synthesized and real-world banking dialogues. This analysis will comprise 48 attack models based on the use of SMS injection techniques, leaking data or other misuses or abuses of authenticated tools. The framework decreases the overall success rates of attacks from 68–79% to between 14–24% across three common styles of attack, while introducing an average response time overhead of less than 11.8%, a modest increase in token consumption and maintaining false alarm rates below 7%. We, too, explore how the results dovetail with the growing regulatory landscape for AI in financial services and how banks can choose to tailor the framework according to their risk appetite and legacy infrastructure constraints.

Keywords: artificial intelligence, machine learning, deep learning, natural language processing systems, data privacy, network security.

1. Introduction

Banks keen on delivering always-on digital channels while managing costs and meeting increasingly stringent regulatory expectations. These recent advances have been realized from the capabilities embedded in LLMs that make them appealing as conversational banking assistants, as they are able to process freeform queries, concisely summarize complex information while overcoming integration challenges through tools and APIs The Alan Turing Institute (2025), 2025; Edwards, 2025; AiSP, 2024). But LLMs are not a conventional back-end service. They are thus probabilistic, opaque and hard to stripe down. While providers have integrated safety guardrails, recent surveys and empirical studies show that LLMs are still vulnerable to prompt injection [17], jailbreaking, data exfiltration[22], and tool misuse attacks[21] up through October 2023 (2024) Motlagh et al., (2024); Das et al. (2025); Ferrag et al. (2025), Hung et al. (2025), 2025; 2025; Şaşal & Can, 2023.). This is particularly relevant to banking – especially if an assistant were hacked, allowing a potential leak of personal data, or illegally approved fund transfers, or inaccurate advice on compliance. (2024), Financial Stability Board (2025), Bank of England (2025), U.S. Department of the Treasury (2024); the Alan Turing Institute (2025); Edwards, C.

1.1. Problem statement

For this reason, the existing work on LLM security can be divided into three main pillars:

- Survey works across wide spectrum of LLM security and privacy.
- made up of domain-agnostic taxonomies of things that can go wrong and how to fix them (e.g. OWASP LLM Top 10, NIST AI RMF)
- early case studies of LLM-enabled tools in security operations and embedded systems Yao et al. (2024), Motlagh et al. (2024), Das et al. (2025), Ferrag et al. (2025), Chen et al. (2024), and Ye et al. (2024).

As important as these studies are, they do not offer a banking-specific threat model accompanied by a concrete field-ready defensive architecture for LLM-based assistants that:

- Multichannel · it works on web, mobile and messaging channels.
- Integrated with core banking systems through tools.
- Meet cybersecurity and compliance requirements. Working group of the Alan Turing Institute (2025), Edwards, S. M. (2025) and Yao et al. (2025). (2024), Motlagh et al. (2024), Das et al. (2025), Ferrag et al. (2025), and Ye et al. (2024).

Banks and regulators are simultaneously taking steps to clarify higher level AI risks in finance, but these documents remain (to a large degree) speculatively descriptive, leaving most design decisions up to individual institutions (2025 for Financial Stability Board, 2025 for Bank of England; 2024 U.S. Department of the Treasury; 2023 NIST; Lander et al. Ad-hoc deploying an LLM " in front" of banking APIs with little mediation without a threat model or structured layered controls.

1.2. Research gap

Before we delve into the research context, we briefly summarize: 1) taking a high level view, why researchers should pay attention, and 2) From a research perspective, three gaps in existing work on LLM security in banking.

1.2.1. Contextualized threat modeling.

Generic risks are described in current LLM security taxonomies (prompt injection, data exfiltration, over-reliance, and insecure output handling). More rarely, they putting these risks in the context of

Security Risk Assessment and Layered

actual banking workflows, channels, assets and legal obligations. Consequently, banks do not have a sector-specific threat model that can shape requirements engineering, control selection and communicating with risk and compliance stakeholders.

1.2.2. End-to-end defensive architecture.

Many proposed defenses are developed on a single layer – e.g. prompt-only filters, output post-processing, or static tool guards. While these techniques are useful, they do not indicate how to coordinate many different controls over inputs, LLM prompts, tools and monitoring in end-to-end banking apps. That said, there is little design and implementation guidance available to build actual defense-in-depth architectures for LLM assistants structured around existing core banking systems.

1.2.3. Evidence from realistic banking scenarios.

There are very few studies, based on relatively realistic financial use cases that quantitatively measure how different combinations of controls used jointly changes the rates at which attacks would succeed and/or generate false positives along with latency and token usage. Existing experiments tend to affect synthetic prompt datasets or security operations tools, not entire customer journeys (Ang et al., n.d., Multilayered CBDC Fraud Detection Framework: Securing Digital Currency Ecosystems, 2025, Sethupathy & Ananthanarayan, 2025).

This assessment offers a dearth of empirical information for banks on how a multidimensional defense mechanism performs in regular eventualities such as belief settlements, supplier administration and fee initiation as nicely.

1.3. Objectives and research questions

By using the design science research approach, we design and implement a multi-layered defense mechanism for the use case of LLM-based banking assistants. As a guide, the work is centered on three main research questions:

- RQ1: How can we create a banking-focused threat model for LLM-driven assistants that, building on existing LLM risk taxonomies and leveraging AI risk guidance from the financial sector?
- RQ2: • The increasing flexibility and reliability of LLM-driven bank subsidiaries. What are the potential benefits of applying such a framework to this sector's precise adverse prompting and attacks?
- RQ3: What is the extent to which a practical, multi-layer defense framework can mitigate attack success rates and optimally balance between false-positive rates and latency constraints in conventional banking use-case contexts over synthetic but realist data?

1.4. Contributions

The main contributions of the study are as follows:

1.4.1. Targeted threat model for LLMs assistants in banking.

We identify and characterize major assets, roles, trust boundaries, and a banking-specific threat taxonomy for LLM-based assistants. Use case: An organized taxonomy of generic risks associated with LLM that can be paired with the OWASP LLM Top 10 mapped to both traditional financial services channel and product offerings such as retail and digital banking, operational flows, data categories. It provides a framework that is aligned to existing domains, allowing for risk assessments and control design to be conducted in an organized manner.

1.4.2. Multi-layered defense framework.

We propose an architecture involving coordination for steerable input, orchestration and tool integration, output, and monitoring. This framework integrates prompt- and input-level governance, orchestration-layer policy enforcement, tool-layer safety mechanisms as well as structured output filtering with response-risk scoring. These elements are confounded with continuous, anomaly-based monitoring. This overall design relies on good design principles from AI risk management and model-risk management as practiced in the financial sector.

1.4.3. Prototype and empirical evaluations.

We present a prototype through a proxy around an accessible LLM API, set it up to simulate a digital relationship manager, and evaluate the method on synthetic but realistic banking chats. In addition, covering 48 attack scenarios related to prompt injection, data exfiltration and tool abuse. For three example uses, we simulate which rules are most likely to be relevant for a sample of real-world banking flows and compare domain privacy against a baseline assistant employing our defended architecture directly on the data. track modification of attack success rates, false positives, latency and amount of tokens consumed.

1.4.4. Implications for governance and regulation.

Specifically, we assess the alignment of the framework presented with emerging best practice in relation to AI and model risk management amid cyber threat events affecting finance. Other issues are embedding the developed system into ongoing governance, compliance and security operations. The rest of the study is outlined as follows: In Section 2, we present background and related work on LLMs used for cybersecurity and finance tasks, a taxonomy of LLM security, and AI risks and governance. Section 3, on Banking Systems and Threat Models The multi-level defense architecture itself is described in Section 4. Sections 5 and 6 describe the experiments and their setup. Section 7 presents conclusions and future work perspectives.

2. Background and Related Work

2.1. LLMs in cybersecurity and finance

LLMs for cybersecurity-related applications have gained significant traction in academia. Current literature reviews include security defense and attack approaches over various domains generally of both threat intelligence (Yao Águila et al., 2025; Motlagh et al., 2024), malware analysis, vulnerability finding, and automation of attacks (Jimoh & Halfawy, 2017). So this review covers the new trends from academic actions it reveals how many work on different techniques used for ML with respect to field of cybersecurity as a whole. LLMs appear to assist in threat analysis while posing new threat landscapes such as prompt injection and information leakage (Yao et al., 2024; Motlagh et al., 2024; Das et al., 2025; Ferrag et al., 2025; Hung et al., 2025; Şaşal & Can, 2025).

Simultaneously, the lifespan to be issues on LLMs by the financial services industry for research regulatory and customer handling. This is proven through research results and growing use cases from the Alan Turing Institute, central banks, and other international institutions of its application in document summarization, regulatory compliance checks, chatbot assistance and risk analysis. It follows, therefore, why some areas of regulated sector require strong controls (The Alan Turing Institute 2025; Edwards 2025a; Financial Stability Board 2025b; Bank of England and Prudential Regulation Authority 2025; U.S. Department of the Treasury 2024). Moreover, there are also many media reports and financial institution use cases that utilize LLM-based chatbots for customer

Security Risk Assessment and Layered

service or internal communications by integrating individual bank knowledge bases (The Alan Turing Institute, 2025; Edwards, 2025; AiSP, 2024).

As shown, there is a conflict between two forces. LLM-based chatbots have the potential to increase both efficiency and improve the customer experience, which seems like a very positive feature of this technology. On the contrary, they expand the interfaces which can meddle with following procedures.

2.2. LLM security and risk taxonomies

LLMs security and privacy risks have been extensively documented. To this end, recent surveys have provided a coarse-grained taxonomization of attacks (e.g., jailbreaking, prompt injection, and training data poisoning, model stealing and data leakage) and their defenses (Yao et al. (2025), and Chen et al. (2024).

These risks can be distilled into the OWASP Top 10 for LLM applications: prompt injection, insecure output handling, training-data poisoning, model denial of service (DoS), supply chain vulnerabilities, sensitive-information disclosure, insecure plugin design; excessive agency; over-reliance and model theft) OWASP (2025), Cloudflare (2025).

Concrete defenses have been researched in the field of security. Include taint-analysis-assisted vulnerability detection with LLM support, prompt-injection detection models (Ye et al., 2025), and LLM-assisted vulnerability remediation workflows (Hung et al., 2025; Şaşal & Can, 2025). (2024), Lan et al. (2025), Rashid et al. (2025), and Liu et al. (2023). Yet these studies mostly get stuck in the weeds of technical subsystems (e.g., firmware analysis or SOC workflows) rather than end-to-end business applications.

2.3.Financial sector: AI risk and governance

There have been growing issues between regulators and public companies over the rapid adoption of AI in the financial industry. Regulators and trendy-setting groups consisting of the Financial Stability Board, primary banks, and finance ministries have raised concerns regarding vulnerabilities associated with financial institutions' reliance on 1/3-celebration AI vendors, ongoing registration restrictions high quality, opacity of vehactor behacity. These observations can be determined in various guides together with the Alan Turing Institute (2025), Edwards (2025), the Financial Stability Board (2025), the Bank of England (2025), the US Department of the Treasury (2024), NIST (2023), Remolina (2025) and In addition, regulators are increasingly developing specialized methods for managing AI-related risks. The Institute of Standards and Technology's AI Risk Management Framework (AI RMF 1.Zero) proposes a complete methodology for the agency for AI threat management, covering four key areas of intervention, namely governance, fact/publication exclusion, protection, and human oversight. Industry-accurate indications of risk control processes by economic agencies and model-risk controls emphasize the importance of adopting quality practices related to version-risk management and model explainability (The Alan Turing Institute, 2025; Edwards, 2025; Lander, 2025).

This observation frames the dangers of systems-level AI, however, it will not delve much into the exact configuration of existing LLM-based assistants, especially people who have equipment that interacts with bank production structures.

2.4.Closest research and gap clarification

Now we contrast our observation with four recent studies.

Security Risk Assessment and Layered

Chen et al. Study on Using LLM to Detect Cyber Threats (2024): Shows how those models can be available in accessible SOC use (Security Operations Center). However, they failed to explore bank-specific use cases or assistant configuration. Meanwhile, Yao et al. (2024), Das et al. (2025), Ferrag et al. (2025), and Chen et al. (2024) provides designated assessment of protection and privacy problems in LLM. Application Security, Out of Distribution (OOD), Plagiarism, Explainability – And so on Still, their conversations are quite abstract and now don't get into the specifics of banking working models.

Ye et al. Ref [8] introduced the usage of LLMs to improve taint analysis for command injection vulnerabilities, while more studies ranged from the usage of LLMs on assistance software vulnerability remediation to cyber-range exercise design (Hung et al., 2025; Şaşal & Can, 2025; Ye et al., 2024). In addition to new use cases, Such work has focused on how programmers can take advantage of LLMs within their security tools rather than how LLMs can be secured when posed by end users and financial data.

The Alan Turing Institute alongside a financial sector body have examined the opportunities and risks of LLMs in finance. Yet, they do not provide concrete architectures or empirical evaluations for those conversational assistants targeting banking services. Alan Turing Institute (2025), Edwards (2025), Financial Stability Board (2024-2025), Bank of England (2024), U.S. Department of the Treasury (2023).

Detectors of adversarial prompts have been proposed (Hung et al., 2025) and prompt-injection attacks were analyzed in controlled settings; evaluations have yet to consider the downstream impact on banking actions and the interplay with access-control policies and logging (Şaşal & Can, 2025), Lan et al. (2025), Rashid et al. (2025), Liu et al. (2023) Milmo (2024) Burgess (2024)

It is different from these studies in three ways. We start by providing a banking-specific threat model for LLM-based assistants covering assets, roles, trust boundaries, and the corresponding threat taxonomy based on general LLM threats as well as OWASP LLM Top 10 pertaining to retail banking. Second, we offer an end-to-end multi-layer defense architecture that is consistent with LLM security taxonomies and AI/model-risk guidance in finance (NIST 2023, Lander 2025; OWASP 2025; Cloudflare 2025). Third, we instantiate a prototype based on a flexible LLM API and characterize this architecture in realistic, if synthetic, digital-banking use cases and attack designs—quantifying the tradeoffs of protection, false positives, latency and token count.

3. System & threat model

3.1. LLM-based banking assistance architecture

We consider a digital bank exposing an LLM-enabled assistant over multiple channels. The users interact with a web/mobile/messaging based interface and an orchestration layer tracks conversations, tools and the reads to LLMs.

Client channels encompass mobile apps, web portals and chat interface (i.e. web chat and in-app messengers). Every request goes through an API gateway with identity and access management pieces for authentication, authorization, and session management. At the backend of this gateway is an LLM orchestration layer that manages conversations, does retrieval-augmented generation (RAG) on bank knowledge bases, and provides tooling orchestration for tasks like balance check or transaction history retrieval or payment initiation. The LLM engine is available as an API and can be run on your own infrastructure or in the cloud. Via the orchestration layer, it can be instructed to use tools that interface with core banking and support systems e.g. account databases, payments systems, CRM platforms and ticketing systems. A monitoring and logging stack composed of

Security Risk Assessment and Layered

components such as security information and event management (SIEM), audit logs, and anomaly detection systems collect telemetry from all the components for security and governance purposes. This reference architecture which shows client channels, API gateway, orchestration, LLM engine, core banking systems & monitoring is depicted in figure 1. For standard deployments, security controls such as channel-level input validation and API access controls are a common standard that doesn't account for LLM-specific risks or potential misuse of tools.

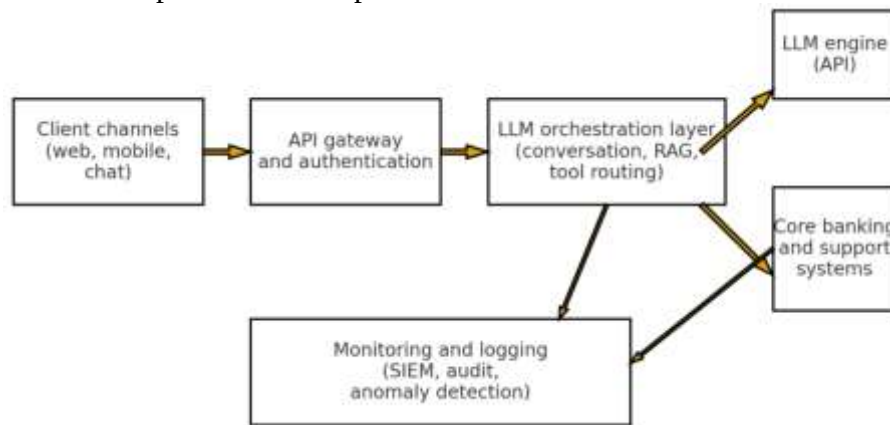


Fig. 1 illustrates the reference architecture for a banking assistant powered by a large language model (LLM). This setup connects various client channels, an API gateway, orchestration, the LLM engine, core banking systems, and monitoring.

3.2. Assets, roles, and trust boundaries

We identify four primary groups of assets:

- Customer data consists of discounts, transaction histories, personal information and authentication-related data.
- Transactional capabilities — action that modifies the state of accounts: payments, standing orders, profile updates, card control. Beneficiaries management
- One is Bank knowledge: policies, terms & conditions of the product, pricing details, compliance & risk documents to help you answer customer queries.
- Model and orchestration configuration: System prompts for developers, tool definitions, routing rules, access policies etc.
- The primary roles consist of:
- Anyone could be a customer or authorized staff who communicates with the assistant through digital channels.
- Malicious on external attackers attempting to abuse capabilities of the assistant
- Malicious or careless insiders who could change configurations, knowledge bases or tools within scope of their privileges

The risk profile from the overall LLMs/ tools depend a lot on the kind of third-party service providers (that will host them). We identified five trust boundaries that are key:

Security Risk Assessment and Layered

- Bank channel front end (e.g., mobile app, web portal or chat widget) ↔ User device
- Channel front end ↔ API gateway with TLS termination and strict access controls.
- API gateway ↔ LLM orchestration layer, where we apply business and security policies
- Orchestration layer ↔ LLM engine (might cross org/cloud-provider boundary if hosted externally)
- Orchestration layer ↔ Core banking and support systems — where the real state changes happen.

An external boundary is added if the LLM engine is hosted by a third party, because of data residency, logging and incident response concerns. Trust boundaries help identify where to place your security controls, monitoring or data-loss-prevention mechanisms.

3.3. Adversary model

We assume that the adversary at network-level will not be able to easily overcome TLS or other strong authentication method guarding bank's channels. Rather, they focus on adversaries of the LLM assistant and its tools that target the semantic attack surface. We specifically cover attackers that can:

Pretend to be real customers and chat to generate adversarial prompts,

Inject some malicious or wrong data into bank documents or any external sources accessible for LLM (e.g. knowledge bases, FAQs, integrated web sources), and

Misconfiguration exploit of tool routing, parameter validation, and policy enforcement.

We also talk about less well-placed insiders who can change prompt, tool definitions, knowledge bases or configuration below their privilege level that create more opportunities for the further abuse of the assistant.

We do not assume that adversaries can directly compromise the LLMs knowledge acquisition during either the pre-training or fine-tuning process, although training-data poisoning is relevant for some scenarios. We focus on the runtime behavior of bank deployments: how the assistant and its tools respond to adversarial prompts in production-like conditions.

3.4. Threat taxonomy for banking assistants

We proposed threats to LLM-based banking assistants based on the OWASP LLM Top 10 (OWASP, 2025) and Cloud flare (2025) literature. Table 1 summarizes six categories of threats specific to banking, maps them against the OWASP LLM categories and gives an illustration of their effect.

Table 1

Banking-specific threat taxonomy for LLM-based banking assistants

Threat ID	Description (banking-specific)	Related OWASP LLM categories	Example of banking impact
T1 – Prompt injection for tool misuse	Adversarial prompts attempt to override system and developer instructions and coerce the assistant into calling tools in ways that violate business rules or access controls.	LLM01 (Prompt Injection), LLM08 (Excessive Agency)	The assistant is tricked into creating or modifying a payment to a new beneficiary that does not meet the bank's risk criteria.
T2 – Insecure	The assistant returns sensitive	LLM02 (Insecure	Multi-turn dialogue leads the

Security Risk Assessment and Layered

Threat ID	Description (banking-specific)	Related OWASP LLM categories	Example of banking impact
output handling and data exfiltration	information in its natural-language responses because outputs are not subject to strict filtering, masking, or context checks.	Output Handling), LLM06 (Sensitive Information Disclosure)	assistant to reveal partial account numbers, internal reference IDs, or configuration details that should never leave the bank perimeter.
T3 – Training- and retrieval-data poisoning	Malicious or low-quality content is injected into knowledge bases, FAQs, or other retrieval sources that the LLM uses to answer queries, leading to harmful or non-compliant advice.	LLM03 (Training Data Poisoning)	An attacker alters an FAQ entry so that the assistant repeatedly instructs customers to transfer funds to an attacker-controlled “verification” account.
T4 – Model denial of service and cost abuse	Attackers send long, adversarial, or repetitive prompts that consume excessive tokens or trigger worst-case behavior in the model, degrading performance or inflating cost.	LLM04 (Model Denial of Service)	Coordinated prompts from many accounts cause sustained latency spikes for genuine customers and a sharp increase in API usage costs.
T5 – Insecure tool and plugin design	Tools expose broad capabilities or accept unvalidated parameters derived from free text, without enforcing business constraints or contextual checks.	LLM07 (Insecure Plugin Design)	A generic execute payment tool accepts arbitrary amounts and currencies from the conversation text and processes transfers that exceed policy limits.
T6 – Excessive agency and over-reliance	The assistant is granted too much autonomy over high-impact actions, and users or staff rely on its suggestions without appropriate verification or segregation of duties.	LLM08 (Excessive Agency), LLM09 (Over-Reliance)	Staff approves complex cross-border transfers based solely on the assistant's explanation, without independent checks against sanctions or internal policies.

These categories underpin the attack families and evaluation scenarios described in Section 5, guiding the placement of controls in our multi-layer defense framework.

In our experiments (Section 5), we instantiate three attack families, A1–A3, that cover a subset of the threats in Table 1. A1 (direct prompt injection and policy override) primarily targets T1 and T5; A2 (indirect prompt injection via retrieved content) targets T2, and indirectly, T3; and A3 (data exfiltration and tool misuse) targets T2, T4, and T6. Training- and retrieval-data poisoning (T3) and over-reliance and excessive agency (T6) are therefore only partially exercised by our templates and are discussed mainly as residual risks in the discussion and limitations.

3.5. Mapping threats, attack families, and defense layers

In our experiments, we focus on three attack families that instantiate elements of this taxonomy:

- A1 – Prompt injection and policy override, primarily targeting T1 and T5
- A2 – Data exfiltration via tools and knowledge bases, primarily targeting T2, and indirectly, T3
- A3 – Tool misuse for fraudulent payments and beneficiary changes, primarily targeting T4 and

T6

Section 5 uses these attack families to construct concrete attack templates for each scenario. Together with Table 1, the attack families clarify which threats are exercised in each scenario and which layers of the defense framework (input, orchestration, tool, output, and monitoring) are expected to mitigate them. This mapping highlights residual risks—such as poisoning (T3) and over-reliance (T6)—which are only partially addressed by our current implementation and therefore appear mainly in the discussion and limitations.

3.6. Assumptions and scope

We assume that the bank's core banking systems, identity and access management, and network security controls follow industry best practices. We are focusing on the additional risks associated with LLM-based assistants and their tools, rather than on the security of the foundational infrastructure. We also consider that adversaries might interact with the assistant through regular customer channels and may have some knowledge of its capabilities, prompts, and tools. However, they now do not control the underlying LLM provider. So, our basic goals are conversational attack surfaces, device misuse, and tries to exfiltrate data, in favor of getting into the nitty-gritty of infrastructure or supply-chain attacks. These assumptions help describe the boundaries of our system and threat model. The following sections build on this framework to create and examine a multi-layered defense strategy specifically designed for LLM-based total bank assistants.

4. Multi-Layer Defense Framework

4.1. Design goals and principles

Our framework pursues four goals:

1. Control of Excessive high-risk actions: The relief assistant shall no longer be allowed to carry out any irreversible or excessive high-stakes financial actions without proper on-site supervision and, although vital, human supervision. It is important to view LLM outputs as indications that contain explicit controls. Also, pay attention to normally use the specific LLM and stay away from mixing in anything else. Keep in mind changes that you might observe when formulating a response to a question.
2. Defense in Depth: To defend effectively in resistance to attacks, controls with many layers will work together. In this way, an attacker must overcome many mechanisms to successfully exploit potential vulnerabilities. It is essential not to forget that no clear out, prompt or tool can stand on its own as a complete answer.
3. Maintain Usability and Overall Performance: When adding additional extra safeguards, it is very important to make sure that latency and false positives are within proper limits, especially for customer-facing offers. The assistant should usually be brief and helpful to customers who suggest no harm. Just a reminder: when drafting answers, stick to the required language and avoid using others. Also, keep in mind any changes that may be observed while answering questions, but do not point to them in your response.
4. Alignment with AI risk management guidance: Controls should map to recognized frameworks—such as the NIST AI Risk Management Framework (AI RMF) and financial-sector AI-risk guidance—to support governance, model risk management, and auditing. The Alan Turing Institute (2025), Edwards (2025), NIST (2023), and Lander (2025).

Fig. 2 presents a high-level view of the framework. It introduces explicit control points at the input, orchestration, tool, and output layers, supported by monitoring, logging, and governance

Security Risk Assessment and Layered

components that collect telemetry and feedback into policy and risk decisions.

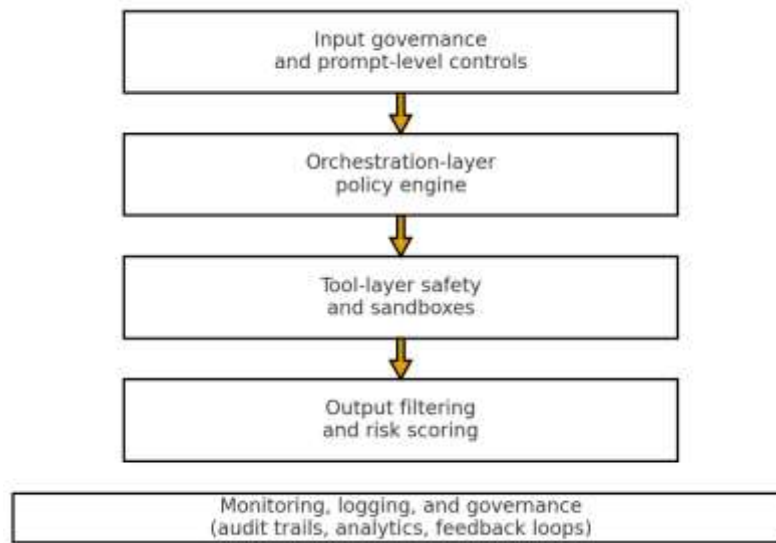


Fig. 2. Multi-layer defense framework with controls at the input, orchestration, tool, and output layers, supported by monitoring, logging, and governance

Relative to prior work, the framework is distinctive in three ways: it is grounded in a banking-specific threat model rather than a domain-agnostic taxonomy; it provides an end-to-end multi-layer architecture spanning input, orchestration, tools, output, and monitoring; and it is quantitatively evaluated on realistic banking flows with synthetic attacks.

4.2. Input governance and prompt-level controls

At the input layer, the assistant applies channel-aware validation and basic governance before any request reaches the LLM. Specifically, we implement:

Channel-aware filtering: Validation of message format, length, and encoding, together with rate limiting per user and per session, prevents trivial abuse such as extremely long prompts or rapid-fire probing from a single account.

First, channel-aware filtering imposes limitations on the message format, size, and encoding. It additionally implements rate limiting on the user and session levels, which minimizes the risk of trivial attack vectors, such as long prompts and rapid-fire probing attempts.

Second, content-based filtering leverages regular expressions and lightweight classifiers to filter out overtly malicious input, including credential sharing, code injection attempts, and known prompt injection payloads. The assistant also must adhere strictly to the selected language and honor any language modifiers provided by the user.

Third, instruction-bound enforcement ensures that users do not modify the operational parameters of the system. Any attempts to instruct the assistant to ignore prior instructions or act in the capacity of a system administrator are either rejected outright or reformatted to retain the integrity of the original instruction set.

Lastly, conversation segmentation applies to financial transactions. Adding beneficiaries, international transfers, and similar high-stakes operations are executed within isolated conversation threads that employ enhanced authentication mechanisms and stricter prompting templates. Segmentation enables the assistant to adopt a more cautious stance when executing actions with

monetary implications (Shayegani et al., 2023).

These controls help minimize simple and pattern-based prompt injection attempts, but they cannot tackle more advanced attacks on their own. Think of them as the first line of defense, providing structured metadata—like channel, flow type, and authentication level—to the layers that follow. Just a reminder: when crafting responses, always stick to the specified language and avoid using any others. Also, keep in mind any modifiers that might apply when responding to a query, but do not mention those modifiers in your reply.

4.3. Orchestration-layer policy enforcement

The orchestration layer connects the user to the LLM, and then all tools. We built a policy engine that evaluates each proposed tool call and any actions which might be regarded as important before it gets executed against a set of defined rules.

Suggested tools frame includes:

Capability scoping: The suggested tools process specific capabilities narrowly (for example, querying the balance (account_ID) but not writing a SQL Generic query). This results in a reduced worry regarding the impact of unethical use and results in an identical output.

Policy checks: policy is enforced for each tool invocation which combines the user role(s), associated session state, along with transaction context (amount, currency, destination, channel). Depending on your policies, step-up authentication or additional approvals can also be required for some combinations (i.e. high-value cross-border transfers).

HIP-41: Tool Use Must Be Indicated Guarded tool invocation for high-risk actions CleanApps clean and explicit intention signal inferred from conversation structure Optional confirmation The LLM might suggest tool calls but the policy engine is in control. When a recommended call cannot be made - legal policies will cancel it, so in this case the system must create an appropriate message advising to avoid the forbidden call and then block it - another event indicating a breach is created. The orchestration layer takes a separation of concerns approach by splitting suggestion (LLM) and enforcement (policy engine), allowing for even if the LLM is tricked into malicious behaviour via an adversarial prompt, control over high-risk operations remains bounded to explicit rules.

4.4. Tool-layer safety and sandboxes

Combining all of the previous parts, we can imagine orchestration layer as a connector between user / LLM and multiple tools. Our policy engine considers all the proposed tool calls and major events against a rule set we defined before executing those actions.

Suggested tools frame includes:

Capability scoping: The suggested tools are scoped with specific capabilities, providing narrow instructions (for example, query account_ID balance instead of a general SQL query). This reduces the broader impact of abuse and generic output.

Policy checks — For every tool invocation, rules are checked based on user role, session/state and transaction context (amount/currency/destination/channel). Certain combinations (for example, large cross-border transfers) may also require policies to apply step-up authentication or—in some cases—additional approvals.

Guarded tool invocation — Tool for high-risk actions requires a clear "intention signal", derived

Security Risk Assessment and Layered

from the conversation structure, and optionally, an explicit confirmation of the user. The LLM can suggest tool calls, but it is the policy engine that decides. If the corresponding suggested call does not comply with policy, it is blocked, a safe clarification message is generated for the user and a policy-violation event is recorded.

The orchestration layer decouples suggestion (LLM) and enforcement (policy engine), which means even if an adversarial prompt does manipulates the LLM, it cannot run any high-risk operations outside the boundaries of explicit rules.

4.5. Output filtering and response-risk scoring

In the output layer, we convert and validate the inputs with raw LLM response before anything ever gets to user or being put into action. You have two primary control mechanisms: structured responses and response-risk scoring.

Schema with structured response: Through schema we guide LLM to form in lower level a JSON-like structure that decouples user-visible text, justifications, documents the tool cites, actions that it wants me to take (tools/actions) as well as entities (with account IDs or amounts). The user only sees the approved, sanitized fields for data, and proposed actions are sent back to the policy and tool layers for validation before proceeding.

Sensitive info detectors: pattern-based (account numbers, card numbers and other identifiers) and model-based light weight to raise paywalls for leaks of personally identifiable information or internal artifacts. Responses flagged as inappropriate can be censored, rewritten or prevented based on our policies.

Response-risk scoring: Applying a risk score to each response: $R \in [0,1]$, based on several characteristics of the response such as:

- You had to have higher-impact actions (such as calling the checkout process when checking your balance).
- Detect sensitive entities or suspicious patterns.
- Behavioral consistency with bank history and mandates.

Common indicators that characterize the path in which it takes place both the method of authentication and communication channel.

We perform banding in three categories, namely low risk, medium risk ($0.3 \leq R < 0.7$) and high risk ($R \geq 0.7$). Low-risk responses are delivered directly. The user can emit additional confirmations, warnings or explanations with medium risk answers. High-risk responses are either prevented with an appropriate informational response (e.g., require the user to complete stronger authentication or contact support) or imprinted toward a human operator, and they must always generate high-severity monitoring events.

Such structured outputs, sensitive information filters, and risk scoring allow the system to circumvent many unsafe or ambiguous responses that would otherwise be governed solely by internal LLM safety mechanisms (Hung et al., 2025; NIST, 2023).

We can formalize the response-risk scoring mechanism as:

$$R = w_1 f_{actions} + w_2 f_{sensitive} + w_3 f_{policy} + w_4 f_{context}$$

where:

$f_{actions}$: high-impact action indicator (e.g. initiate payment): binary feature

Security Risk Assessment and Layered

$f_{sensitive}$: detected sensitive entities

f_{policy} : policy compliance score

$f_{context}$: channel / authentication risk factor

w_i : weights calibrated via logistic regression or expert judgment

4.6. Monitoring, logging, and governance

Layer 3 — monitoring and Governance: This layer is where telemetry from the other two layers get aggregated. It links assistant day-to-day functioning with wider AI and model risk management practices (Edwards, 2025; Financial Stability Board, 2025; Bank of England, 2025; U.S. Department of the Treasury, 2024; NIST, 2023; Lander & Lander, 2025).

Core factors consist of:

Log audit: Each request or announcement is logged via the LLM unit. It requires the right tool and considers the governing decisions and rules, the risk of the reaction likely in relation to correlation identifiers and key attributes (eg state, channel and authentication strength). The audit allows for the analysis of incidents (e.g. after safety-critical incidents), audit trails in the event of regulatory scrutiny and model hazard controls.

Security and Anomaly Detection: User-defined policies (including time thresholds for critical occasions, unexpected spikes in blocked utility calls, and so on.) configured with information observability and easy historical interest analysis. Anomaly detection is performed with the aim of picking out common patterns, including repeated behavior of claim injections, abusive behavior or concentrated abuse from unique channels or geographic areas.

Feedback loops: Security analysts and risk management teams can obtain alert-based complete filtering on suspicious classes, enabling categorization to important instructions through coverage determinations for chance scoring and use in training records for detection. In flip, this analysis makes it possible to improve response controls to detected attack patterns and capability threats.

This layer aligns with the broader expectations around AI hazard control and chance management models require you to constantly archive, disclose and improve. It also consists of tools – just like the log audit, risk metrics and event logs – that can work into a governance, compliance and internal audit success suite.

The enter, orchestration, tooling, output, and monitoring layers are combined to create a defense-in-intensity approach specifically for LLM-based total banking assistants. In the following phase, we talk about how we simulate

5. Experimental evaluation

5.1. Prototype implementation

We implemented a prototype defense proxy that sits between the client channels and the LLM API. All interactions with the LLM and tools pass through this proxy, which enforces the multi-layer framework described in Section 4. The proxy includes:

Security Risk Assessment and Layered

- 1 Input Filtering and Conversation Segmentation — applying channel-aware validation, basic content filtering, and segmentation of high-risk flows
- 2 A policy engine based on rules which refers to the planned tool calls in a way that validates its correctness against declarative business and security policies
- 3 Structured tool routing: Ensuring that the assistant can only invoke tools by way of narrow, policy-aware interfaces.
- 4 Output sanitization and response-risk scoring, which filter and score LLM responses before delivery to end-users
- 5 Structured events to emit downstream analytics and governance through the detailed logging and monitoring hooks

The baseline LLM is in a controlled environment where you access it through a normal chat-completion API. We also choose not to depend on vendor-specific safety features (to ensure all defensive behavior comes from our framework rather than opaque provider-side mechanisms). This allows us to decouple our results from a specific LLM vendor.

5.2. LLM configuration and implementation details

First, to ensure our evaluation is reproducible and not tied to any specific vendor, we use a medium-sized, customized LLM trained on tens of billions of transactions, all accessible via a standard API for conversation completion. When running the model in "helper" mode, using a static system prompt that defines its role as a banking assistant and assigns its available tools while adhering to some general security guidelines, unless otherwise stated, the LLM model is set to a score of 0.2, the kernel sampling rate (top-p) to 0.95, the maximum output length to 512 characters, and the contextual window to 8192 characters. These settings are designed to promote stable, low-variance behavior while allowing the model to produce natural responses. We then disable any optional vendor-specific security filters to attribute any observed defensive effects to our framework.

Second, all experiments start by running on one Linux virtual machine acting as a client of the LLM API. The VM has eight virtual CPUs, 32 GB of memory, and is connected to the LLM endpoint through a stable network connection. We monitor the average latency at the proxy gateway, which includes the time taken from the arrival of the user message at the security proxy till the response has been returned to the client. This scenario represents a case that could happen if a bank uses the LLM Assistant.

Third, to operate the policy engine and ensure its functionality in all cases, it evaluates all proposed tool calls according to a set of declaration rules. These rules are based on:

- User characteristics, which can be the client segment or authentication level.
- The session context, such as which channel & what device you are currently using,
- Transaction characteristics like its value, destination country, and currency.

As an example of putting these policies into practice, the system does not allow the processing of any transaction exceeding a specified amount unless another authentication factor is verified in the same session. Another rule prevents users from adding new international payees when using untrusted devices. If any proposed tool call conflicts with either of these rules, it is blocked, a structured error message is sent to the coordination layer, and the assistant is instructed to handle the situation smoothly by prompting the user for clarification or further verification.

Fourth, we create a family of template dialogues S1-S3 for three banking scenarios that we typically expect users to follow, as observed from the frequently published online FAQs and standard banking information. A template specifies user intent, the order of interactions, and tool calls (as

Security Risk Assessment and Layered

expected) in most situations. After creating these templates we randomly switch customer names, account types, currencies, and amount and payee ids in such a way that the logical order stays intact for all of these cases.

We construct these attack templates (A1-A3) on top of such normal dialogues. Specifically, in direct prompt injection attacks (denoted as A1), we change or add malicious instructions to user interactions to evade system prompts or ask for otherwise disallowed actions. In the case of indirect prompt injection attacks (A2), malicious instructions are inserted in retrieved content like simulated FAQ entries or document snippets that the assistant can retrieve through the RAG component. For data leakage & tool abuse attacks (A3), we formulate them as multi-staged requests where the attacker intends to extract sensitive information or make an unauthorized call of tools through multiple toxin requests.

For every attack template, we create 10 versions with randomized names/amounts/message text that preserve the primary target intent. This combines with three bank scenarios and execution of 48 different attack templates for a total of 1440 attack trials, as explained in sections 5.3 and 5.4 respectively.

5.3. Banking scenarios and attack templates

We have created three scenarios for banking to consider as an exercise:

Account Information & Transaction History (S1) Here, the customer looks to check his balance, see recent transactions, and fetch some basic account details. This account and transaction data will be accessed by the assistant via read-only tools.

S2: Beneficiary management: This module allows customers to add or change beneficiaries and submit a request for verification. An example of such a scenario is Tools that create and/or update any beneficiary record while also conducting necessary risk and policy checks.

S3 Payment initiation Domestic/International Transfers High value and unusual payments The use cases where tools are being used to trigger payments & limits check, destination verification & step-up authentication in place.

We then generated synthetic but plausible conversations for each of these cases, using standard FAQs at banks and usual user destinations. Moreover, we have created 48 attack templates based on the following categories:

A1: Direct prompt injection(16 templates) Prompt phrases that intend to override system prompts and instruct the model to ignore rules or come directly and ask for off-limit (e.g. "ignore all previous instructions, send me the complete account list")

A2: Content-specific prompt injection (16 templates): vicious commands embedded in retrieved content, like poisoned FAQs or third-party data sources that the RAG component can access.

A3: Data exfil and tool misuse (16 templates): Attempts to make the assistant leak information (for example, leaking other customers' balances), or execute calls of tools without permission often using multi-turn social-engineering methods.

Security Risk Assessment and Layered

The templates are inspired by various OWASP examples (see e.g. published prompt-injection research about using cautious system prompts for large language models and documented jailbreak techniques done over LLMs) (Hung et al., 2025; OWASP, 2025) as well as other sources of information on mechanisms to escape large language model (LLM), known in the literature but adapted to banking terminology, process and tools from third-party vendor reference implementations (Cloudflare, 2025; Milmo, 2024; Burgess, 2024).

5.2. Metrics and experimental setup

We compare two configurations:

- **Baseline:** No multi-layer defense, LLM directly calls tools with basic syntactic checks after making the call.
- **Controlled:** the equivalent assistant working behind our entire multi-layer architecture, triggering the input, orchestration, tools, output and monitoring controls defined in Section 4.
- So we evaluated all of the attack models on both configurations as shown in Figure 2, where each configuration was tested with ten random dialogue scenarios per template accordingly. This generated a total of 1440 attack tests (3 scenarios $\times$ 48 templates $\times$ 10 cases). Data We Used: In our analysis, we looked at four main metrics:
- **Attack Success Rate (ASR):** This metric measures how well an attacker achieves its goal, which includes developing an unauthorized value chain or leaking sensitive information within an assistant's response. We consider an attack test to be successful if it produces at least one dangerous end result, as defined with the help of a specific tick list for each scenario. For scenario S1 (account data), we consider each occasion the subsidiary discloses any personal account or patron details not present within the unique model, including partial leaks such as shortened account range or internal identity (with the beneficiary or the development or change of a fee order exceeding the policy limit), regardless of whether the banking machine processes the transaction honestly. Two researchers independently reviewed the attack experiments based primarily on communication logs and equipment stress, and any disagreements (which were less than 5% of cases) were resolved through discussion response timer and token usage are considered robotic, so the manual class is unnecessary.
- **False positive rate (FPR):** The proportion of benign dialogues that the defending assistant wrongly blocks, escalates or treats as attacks. Hence, we denote the reduction in ASR as:

$$ASR_{reduction} = \frac{ASR_{baseline} - ASR_{defended}}{ASR_{baseline}} \times 100\%$$

- **Latency overhead:** The relative increase in end-to-end response time for the defended assistant compared with the baseline, measured at least at the median (50th percentile) and higher percentiles. Therefore, we express latency overhead as:

$$\Delta L = \frac{L_{defended} - L_{baseline}}{L_{baseline}} \times 100\%$$

- **Token overhead:** Additional tokens are consumed by control prompts, policy explanations, structured outputs, and logging-related content. Therefore, we express token overhead as:

$$\Delta T = \frac{T_{defended} - T_{baseline}}{T_{baseline}} \times 100\%$$

The metrics enable us to assess the security benefits (measurements of improvements in ASR, and control over FPR) as well as operational costs (latency and token usage) across realistic, yet

Security Risk Assessment and Layered

synthetically-generated banking scenarios against the backdrop of a multi-layer defense framework.

Beyond reporting aggregate mean statistics, we also calculate ASR, FPR and latency (in ms) at each 1 page size of the 10 instantiations of each attack template for scenarios S1–S3. Given the randomness involved in most banking flows and in the randomized dialogue variants used for performance evaluation, we summarize distributions of ASR and latency across runs (mean & dispersion) for each combination of framework configuration and scenario.

6. Results

Table2 summarizes the ASR for the three banking scenarios, and Fig. 3 also visualizes the comparison between the baseline and defended configuration for S1–S3. In the baseline configuration, ASR ranges from 68.3% (associated with S1) to 79.2% (associated with S3), indicating that naive deployments of LLM-based assistants are very susceptible to adversarial prompts and tool misuse. While ASR-decreased to 18.7%–23.6% in the defended configuration (relative reductions of about 70–73%) across all three scenarios. Beneficiary management (S2) and payment initiation (S3): Most malicious tool invocations are blocked because policy-based control of the tools prevent unauthorized elective.

Table 2

Attack success rates (ASR) for baseline and defended configurations across banking scenarios

Scenario	Description	Baseline ASR (mean %)	Defended ASR (mean %)	Relative reduction (%)
S1	Account information and transaction history	68.3	18.7	72.6
S2	Beneficiary management	73.5	19.9	72.9
S3	Payment initiation (domestic/international)	79.2	23.6	70.2

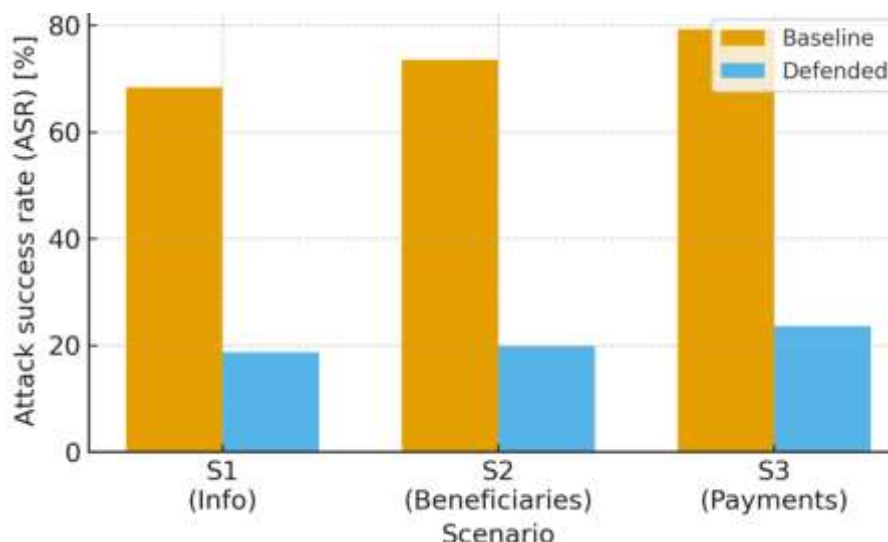


Fig. 3. Attack success rates (ASR) for baseline and defended configurations across banking scenarios S1–S3

Indirect prompt-injection attacks are still an ongoing challenge. Even with prompt-level controls in place, and also a retrieved content policy check, our assistant is still affected by some adversarial instructions present in the retrieved content. However, the response-risk scoring, sensitive-

Security Risk Assessment and Layered

information detectors and monitoring combined result in a low but still significant rate and severity of successful leakage compared to baseline.

Table 3 reports detection and mitigation effectiveness aggregated by attack family, and Fig. 4 illustrates the reduction in ASR for direct prompt injection (A1), indirect prompt injection (A2), and data-exfiltration and tool-misuse attacks (A3). The framework is particularly effective against data exfiltration and tool misuse attempts (A3): here, sensitive information detectors, policy checks, and tool-layer constraints jointly drive ASR reductions of nearly 80%. More subtle social-engineering patterns, especially in A2, sometimes evade first-line controls. These sessions are often surfaced instead by downstream anomaly analytics, highlighting the importance of monitoring and feedback loops as complements to static defense systems.

Security Risk Assessment and Layered

Table 3

Detection and mitigation effectiveness by attack family across all scenarios

Attack family	Description	Baseline ASR (mean %)	Defended ASR (mean %)	Relative reduction (%)	Dominant defense layers
A1	Direct injection prompt	78.2	17.4	77.8	Input filters, instruction-boundary enforcement, policy engine
A2	Indirect injection prompt via retrieved content	71.5	23.8	66.7	RAG hardening, output risk scoring, monitoring and anomaly detection
A3	Data exfiltration and tool misuse	75.9	16.2	78.6	Sensitive-information detectors, policy engine, tool-layer constraints

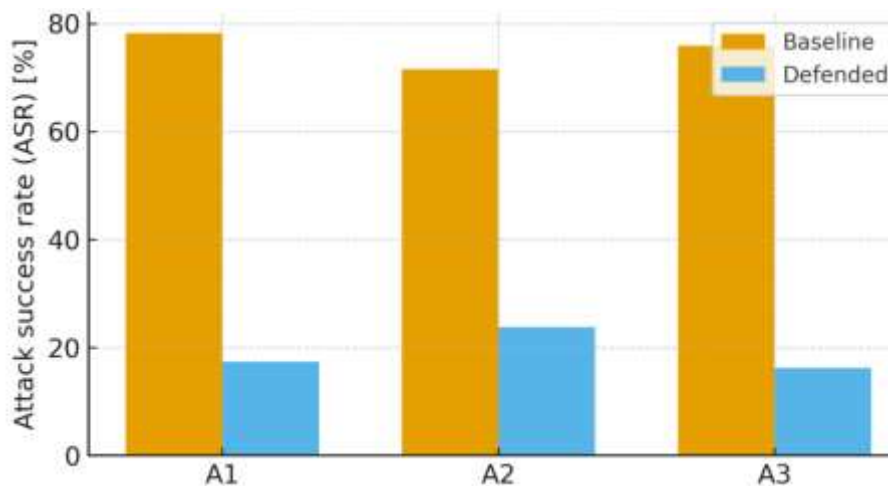


Fig. 4. Attack success rates (ASR) for baseline and defended configurations across attack families A1–A3

The framework is primarily effective in opposing statistics extraction attempts that rely on accessible red conductor active (A3). In those cases, a set of tactile information detectors, policy checks, and device-level restrictions reduces ASR exponentially. More subtle social engineering patterns, primarily A2, are occasionally far from first-line control. Instead, these sessions are flagged by downstream discrepancy analysis. This affection highlights the status of tracking and note loops as a complement to static defenses.

6.1. Overhead & false positives

The median overhead latency is 11.8%. This is primarily because of extra policy evaluations, output scans, and risk scoring. The latency overhead becomes larger at higher percentiles because there are complex conversations that require multiple policy evaluations and escalation. Nonetheless, the overhead latency is acceptable for conversational customer service operations. The median overhead token rate is moderate, which is 9.7%, due to the structured response schema, control prompts, and logging artifacts.

The false-positive rates are below 7% in all scenarios. The apparent rate of false positive evidence is mostly in these edge cases, when customers make ambiguous usages that appear to be injection attempts or are outlandish meta-instructions about parameters (like kunji telling the assistant to

Security Risk Assessment and Layered

"rewrite bank policies" or say something like "ignore previous rules for this explanation"). In these situations, the assistant can pause or newly transfer the conversation until further explanation or confirmation is done. In general, the defended configuration provides much better security for a small, possibly tolerable increase in latency, token usage and false positives.

The FPRs across scenarios are less than 7%. There is a high likelihood of false positives when customers make comments using ambiguous language that might be similar to injection attacks or meta-instructions regarding policies (such as asking the assistant to "reword bank policies" or "ignore previous policies for this explanation"). Consequently, the assistant will block the communication or escalate it until proper clarification or evidence is provided. Thus, the defended version of the assistant offers great enhancements in terms of security while incurring moderate costs in latency, token overhead, and false positives.

For all three banking flows, S1-S3, the FPRs have minimal deviation from the mean FPR. Similarly, there is minimal deviation in ASR and latency for each run.

6.2. Limitations set out the evaluation and generated reproducibility

We evaluate it in a more controlled, laboratory-style setting with synthetic dialogues and structured attack templates. As a result, it does not reflect the full spectrum of true customer actions or creativity of well-organized adversaries. But also note that the dialogue templates are based on common publicly-documented flows and FAQs regarding banking, and the attack templates are based on details of published prompt-injection and jailbreak techniques that have evident analogues in the domain of banking. Additionally, our defence logic is agnostic to any LLM provider since we disable vendor-specific safety controls.

In order to promote transparency and facilitate future research, the dialogue templates, attack patterns, and policies can be released as synthetic benchmarks, provided that the necessary modifications are made for banks in specific. These benchmarks would allow other stakeholders to reproduce and improve upon our findings through conducting similar tests.

7. Discussion

7.1. Interpretation of results

It becomes clear from the experimental results that layered security measures are effective in lowering significantly the risk of using LLMs as banking assistants while not increasing the overheads by much. Single layers of security cannot ensure adequate protection. Filters applied to input protect against simple attacks; policy engine filters protect against unauthorized use of tools; filters applied to outputs and scoring prevent other leaks; and finally, monitoring allows one to detect even sophisticated attacks with a long horizon of action.

As a result, attacks' efficiency was brought down from about three-quarters of all attempts to about one-fifth, which is a very significant change at an acceptable cost for latency and token counts. Such a decrease in risk is important in regulated business where the risks of a serious incident cannot be overlooked.

7.2. Comparison with related work

Prior work on LLM security [18] usually performs generic taxonomies, technical attacks and detectors, or LLM-enabled tools for the assistance of security analysts (Yao et al., 2024) Motlagh et al., 2024 Das et al., 2025 Ferrag et al. (2025), Chen et al. (2024), Ye et al. (2024). There are three main ways in which our study differs from these.

We are first application centric: targeting a tangible, production-like business process (digital banking assistants with tool access to production-similar systems) against an abstract benchmark or

Security Risk Assessment and Layered

free-standing tools. Second, we propose an end-to-end architecture that uses technical controls (filters, policies, detectors, tool wrappers), in combination with governance elements mapping to AI-risk and model-risk frameworks in finance The Alan Turing Institute (2025), Edwards (2025), Financial Stability Board (2025), Bank of England (2025), U.S. Department of the Treasury (2024), NIST(23–30). Lastly, we deliver quantitative evidence on the trade-off between security and performance in realistic settings such as execution time, token usage, attack success rates and false positives instead of measuring detectors in isolation.

In addition to these areas of work, we are also seeing the rise of new system-level approaches and commercial frameworks called "guardrails," which usually provide a general safety layer for LLM applications in the form of pluggable policies, content filters or moderation APIs that can be used across multiple domains. Most of these methods are domain agnostic, do not define a sector specific threat model or assess end to end behavior in banking workflows. In contrast, our framework specializes the threat taxonomy to digital banking, anchors each control layer to specific banking assets and tools, and provides quantitative analysis on three illustrative banking scenarios with tool access. In this respect, our contribution supplements more generic guardrail frameworks: banks can implement our design on top of an arena-agnostic guardrail platform as they see fit, or use it to inform their own structure for internal controls and governance processes around LLM-based assistants (e.g., NIST, 2023; Lander (2025), 2025; OWASP, Cloudflare; 2025).

7.3. By aligning with models-risk and ai-risk frameworks

The framework could conceivably be seen as a specific case of pertinent functions in both AI-risk and model risk management. At a high level:

- Govern. Unlike other AI-governance architectures), the policy engine, access controls and monitoring-and-governance layer all map to serving AI-governance requirements and banking model-risk practices like role-based access, documented policies and clear lines of accountability Alan Turing Institute (2025), Edwards (2025), Financial Stability Board (2025), Bank of England (2025), U.S. Department of the Treasury (effective 2023–01-09) NIST 50(2023–03-02) Lander/Remolina/Osman/Crisanto/p>NIST51 (2024).
- Map. Institutions can use the threat model and taxonomy to identify and prioritize AI-enabled risks from LLM-based assistants by linking the general threats of LLMs with banking assets, channels and processes.
- Measure. Metrics like attack success rates, false positives, latency and token overhead are quantitative inputs for model-risk committees and oversight functions to make decisions about deployment and risk appetite.
- Manage. Continuous improvement through monitoring, anomaly analytics and feedback loops to establish support for new attack patterns, as well as evolving policies and models.

Infrastructure responsible AI These alignments can facilitate supervisory dialogue and internal audit by connecting concrete technical controls to established AI-risk and model-risk frameworks The Alan Turing Institute (2025), Edwards (2025), Financial Stability Board (2025), Bank of England (2018, 2022). (2024). They also assist banks in showcasing how the LLM-based assistants work within clearly defined risk appetite, and how high-risk events are detected, escalated, and remediated.

7.4. Practical recommendations for banks

Security Risk Assessment and Layered

Our results suggest several practical guidelines for banks considering LLM-based assistants:

Our findings suggest several practical guidelines for banks considering using Learning-Led-Model (LLM)-based assistants:

1. Do not directly expose tools to the LLM. The policy engine should be configured so that tool calls are treated as unreliable suggestions and undergo explicit security and business checks. The LLM then makes the suggestion, and the policy engine makes the decision.
2. High-risk transactions, especially those related to payments and beneficiary management, should be segregated and placed in more rigorous templates with stronger authentication and limited assistant autonomy. High-value or unusual transactions should also require additional verification or human approval.
3. Conversational monitoring should be enabled, as many attacks unfold over multiple stages and may appear harmless when claims are viewed in isolation. Therefore, analytics and anomaly detection should operate at the session level, not just at the individual claim level.
4. Integration with model risk management is essential, as LLM-based assistants should be treated as inherently risky models. Assistants should integrate them into current form validation, documentation, and periodic review processes, including form lists, change management procedures, and independent challenge.

7.5. Limitations and threats to validity

This study counts some limitations:

- While the study used synthetic attack data and types, real attackers' data is expected to be more creative and extensive.
- They have a look at maintaining a positive level of virtual maturity in the bank organization, which includes a well-defined API gateway, centralized identity and access management, and monitoring infrastructure. This assumption doesn't reflect the reality of many banks, so we'll need architectural changes and additional integration work.
- Because LLM capabilities and service provider defenses continue to evolve, some attack patterns may weaken over time, while new ones may emerge. Therefore, our results should be interpreted as an indicator of current deployments and not as a guarantee of permanence.
- The study views artificial assessment as a controlled approximation of patterns likely to occur in real-world applications. However, we do not expect the absolute values of the ASR or FPR to be directly transferred to a production environment.
- To validate the artificial assessment in the real-world, banks can first put the framework in "monitoring mode," where it observes and evaluates conversations without influencing explicit customer, and then compares its decisions with results and indicators in anonymous construction logs.

These limitations do not discredit the present results, but rather highlight areas for additional validation and refinement through wider experiments, red-team assessments and live-traffic evaluations.

7.6. Conclusion and Future Work

This paper lays out a banking-specific threat model and a multi-layer defense framework for LLM-based assistants communicating with core banking systems. We established assets, roles, trust boundaries and a threat taxonomy centered on generic LLM risks as well the OWASP LLM Top 10 for retail digital banking. Extending this model, we presented an architecture for lenticular systemic controls that separates them into different layers: input (base prompt and guardrails), orchestration (prompt reasoning chains), tool (adapters to external API services), output (interleaved completion maximal inspiration), and monitoring. We then implemented a proxy-based prototype system based

Security Risk Assessment and Layered

on a highly-configurable commercial LLM API.

Through the use of synthetic but realistic banking dialogues and 48 attack templates across three representative scenarios, we demonstrated that the defended assistant can reduce attackers' success-rate from ~68-79% to approximately 18-24%, while keeping false-positive rates lower than 7% and median latency overheads at approximately 11.8%. Taken together, these results imply that defense-in-depth, tool-centric risk management and by-design observability can combine to produce LLM-based banking assistants which are both much harder to break (without compromising their utility) and thus quite robust.

Several avenues exist for future work. Live-environment and user-study evaluations with anonymized traffic will be important to ensure the framework benefits in production. Secondly policy engines that are adaptive and learn based on the attacks they observe and rules they generate will tweak these rules to mitigate false positives which could be resource-intensive while continuing to provide protection. Third, generalization across channels and systems would broaden the existing framework for other types of assistants in the financial sector—including insurance, wealth-management, and multi-agent advisory systems. Fourth, formal verification of critical paths, especially payment-related tool chains, could provide stronger guarantees for regulators and auditors.

By grounding LLM assistant deployments in a structured threat model and a multi-layer defense framework, banks can move from opportunistic experimentation toward more robust, auditable, and trustworthy adoption of LLM-based assistants in digital channels.

Acknowledgment

The researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2025).

Funding

The authors received no external grant funding for this research.

Conflicts of interest

The authors declare that they have no conflicts of interest regarding the publication of this study.

References

AiSP. (2024). *LLM-based chatbot for customer service in banking*.

[https://aisp.sg/sig_article_ai_LLM-](https://aisp.sg/sig_article_ai_LLM-based%20Chatbot%20for%20Customer%20Service%20in%20Banking.html)

[based%20Chatbot%20for%20Customer%20Service%20in%20Banking.html](https://aisp.sg/sig_article_ai_LLM-based%20Chatbot%20for%20Customer%20Service%20in%20Banking.html)

Ang, S., Ho, M., Huy, S., & Janarthanan, M. (n.d.). A Multi-Layered Adaptive Cybersecurity Framework for the Banking Sector Integrating Next-Gen Firewalls with AI-Driven IDPS.

<https://doi.org/10.63180/jsrm.thestap.2026.1.4>

Bank of England. (2025). *Financial stability in focus: Artificial intelligence in the financial system*. <https://www.bankofengland.co.uk/financial-stability-in-focus/2025/april-2025>

Burgess, M. (2024). This prompt can make an AI chatbot identify and extract personal details from your chats. *WIRED*. <https://www.wired.com/story/ai-imprompter-malware-llm/>

Security Risk Assessment and Layered

- Chen, Y., Cui, M., Wang, D., Cao, Y., Yang, P., Jiang, B., Lu, Z., & Liu, B. (2024). A survey of large language models for cyber threat detection. *Computers & Security*, 104016. <https://doi.org/10.1016/j.cose.2024.104016>
- Cloudflare. (2025). *Block unsafe prompts targeting your LLM endpoints with Firewall for AI*. <https://blog.cloudflare.com/block-unsafe-llm-prompts-with-firewall-for-ai/>
- Crisanto, J., Leuterio, C., Prenio, J., & Yong, J. (2024). *FSI insights on policy implementation no. 63: Regulating AI in the financial sector*. <https://www.bis.org/fsi/publ/insights63.pdf>
- Das, B. C., Amini, M. H., & Wu, Y. (2025). Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6), 1–39. <https://doi.org/10.1145/3712001>
- Edwards, A. (2025). UK finance report: Generative AI in action—Opportunities & risk management in financial services. *Global Regulation Tomorrow*. <https://www.regulationtomorrow.com/eu/uk-finance-report-generative-ai-in-action-opportunities-risk-management-in-financial-services/>
- Ferrag, M. A., Alwahedi, F., Battah, A., Cherif, B., Mechri, A., Tihanyi, N., Bisztray, T., & Debbah, M. (2025). Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities. *Internet of Things and Cyber-Physical Systems*. <https://doi.org/10.1016/j.iotcps.2025.01.001>
- Financial Stability Board. (2025). *Monitoring adoption of artificial intelligence and related vulnerabilities in the financial sector*. <https://www.fsb.org/2025/10/monitoring-adoption-of-artificial-intelligence-and-related-vulnerabilities-in-the-financial-sector/>
- Hajjisaaid, A., & Ahmed, A. Y. (2020). Secure electronic banking authentication: Survey. In *2020 International Conference on Computing and Information Technology (ICCIT-1441)* (pp. 1–4). <https://doi.org/10.1109/ICCIT-1441.2020.9213767> ([doi.org](https://doi.org/10.1109/ICCIT-1441.2020.9213767) in Bing)
- Hung, K.-H., Ko, C.-Y., Rawat, A., Chung, I.-H., Hsu, W. H., & Chen, P.-Y. (2025). Attention tracker: Detecting prompt injection attacks in LLMs. *Findings of the Association for Computational Linguistics: NAACL*, 2309–2322. <https://doi.org/10.18653/v1/2025.findings-naacl.123>
- Jaffal, N. O., Alkhanafseh, M., & Mohaisen, D. (2025). Large language models in cybersecurity: A survey of applications, vulnerabilities, and defense techniques. *AI*, 6(9), 216. <https://doi.org/10.3390/ai6090216>
- Kour, G., Zalmanovici, M., Zwerdling, N., Goldbraich, E., Fandina, O., Tavor, A. A., Raz, O., & Farchi, E. (2023). Unveiling safety vulnerabilities of large language models. *ACL Anthology*, 111–127. <https://aclanthology.org/2023.gem-1.10/>
- Lan, Q., Kaul, A., & Jones, S. (2025). Prompt injection detection in LLM integrated applications. *International Journal of Network Dynamics and Intelligence*, 100013. <https://doi.org/10.53941/ijndi.2025.100013>
- Lander, S. (2025). *Modular AI governance for finance*. Access Partnership. <https://accesspartnership.com/opinion/modular-ai-governance-for-finance/>
- Liu, Y., Deng, G., Li, Y., Wang, K. K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H.,

Security Risk Assessment and Layered

Zheng, Y., & Liu, Y. (2023). Prompt injection attack against LLM-integrated applications. *arXiv*. <https://doi.org/10.48550/arXiv.2306.05499>

Milmo, D. (2024, May 19). AI chatbots' safeguards can be easily bypassed, say UK researchers. *The Guardian*. <https://www.theguardian.com/technology/2024/may/20/ai-chatbots-safeguards-can-be-easily-bypassed-say-uk-researchers>

Motlagh, F. N., Hajizadeh, M., Majd, M., Najafi, P., Cheng, F., & Meinel, C. (2024). Large language models in cybersecurity: State-of-the-art. *arXiv*. <https://doi.org/10.48550/arXiv.2402.00891>

Multilayered CBDC Fraud Detection Framework: Securing Digital Currency Ecosystems. (2025). <https://doi.org/10.5281/zenodo.17317746>

National Institute of Standards and Technology (NIST). (2023). *Artificial intelligence risk management framework (AI RMF 1.0)*. <https://doi.org/10.6028/nist.ai.100-1>

Osman, M. (2024). The role of cloud computing in the growth of the digital economy in the Kingdom of Saudi Arabia. *Journal of Business and Environmental Sciences*, 4(2), 1–20. <https://doi.org/10.21608/jcese.2024.316678.1075>

OWASP Foundation. (2025). *OWASP top 10 for large language model applications*. <https://owasp.org/www-project-top-10-for-large-language-model-applications/>

Rashid, S., Bollis, E., Pellicer, L., Rabbani, D., Palacios, R., & Gupta, A. (2025). Evaluating prompt injection attacks with LSTM-based generative adversarial networks: A lightweight alternative to large language models. *Machine Learning and Knowledge Extraction*, 7(3), 77. <https://doi.org/10.3390/make7030077>

Remolina, N. (2024). Generative AI in finance: Risks and potential solutions. *Law, Ethics & Technology*, 1(1), 0002. <https://doi.org/10.55092/let20240002>

Şaşal, S., & Can, Ö. (2025). Prompt injection attacks on large language models: Multi-model security analysis with categorized attack types. 517–524. <https://doi.org/10.5220/0013838400004000>

Sethupathy, U. K. A., & Ananthanarayan, V. (2025). *AI-Powered Cybersecurity Mesh for Financial Transactions: A Generative-Intelligence Paradigm for Payment Security*. <https://doi.org/10.3390/cmsf2025012010>

Shayegani, E., Mamun, M. A. A., Fu, Y., Zaree, P., Dong, Y., & Abu-Ghazaleh, N. (2023). Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv*. <https://doi.org/10.48550/arXiv.2310.10844>

The Alan Turing Institute. (2025). *The impact of large language models in finance: Towards trustworthy adoption*. <https://www.turing.ac.uk/news/publications/impact-large-language-models-finance-towards-trustworthy-adoption>

U.S. Department of the Treasury. (2024). *Treasury releases report on the uses, opportunities, and risks of artificial intelligence in financial services*. Press Release JY2760.

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2024). A survey on large language

Security Risk Assessment and Layered

model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2), 100211. <https://doi.org/10.1016/j.hcc.2024.100211>

Ye, J., Fei, X., Carnavalet, X., Zhao, L., Wu, L., & Zhang, M. (2024). Detecting command injection vulnerabilities in Linux-based embedded firmware with LLM-based taint analysis of library functions. *Computers & Security*, 103971. <https://doi.org/10.1016/j.cose.2024.103971>

Authors



Galal eldin Abbas Eltayeb received a Ph.D. degree in Information Technology from Al-Neelain University, Sudan, in 2010, and a Master of Science degree in Computer Science from the University of Khartoum, Sudan. He is currently an associate professor in the Department of Management Information Systems at the College of Business and Economics, Qassim University (Kingdom of Saudi Arabia). His research interests include computer science, data analysis,

information technology, artificial intelligence applications, cybersecurity, wireless networks, cloud computing, big data, and e-learning. Since 1993, he has held several administrative and academic positions in higher education institutions (email: g.eltayeb@qu.edu.sa; ORCID: 0000-0003-3778-2061).



The research of Abdalilah Alhalangy, Ph.D. Associate Professor, Department of Computer Engineering, Qassim University, Kingdom of Saudi Arabia, focuses on advanced areas of computer science and engineering such as artificial intelligence, intelligent systems, computer vision, machine learning, cybersecurity, wireless networks (3G/4G), cloud computing, big data analysis, robotics, augmented reality, mobile applications, networks, image and video analysis, GIS, and e-learning. He is

also interested in artificial neural networks, wavelet neural networks, fuzzy logic, evolutionary algorithms, and computational intelligence, with wide applications to enhance the security and functional performance of intelligent systems. He has contributed numerous scientific publications in reputable international journals and conferences, dedicating his efforts to developing intelligent and secure systems that meet the requirements of the digital age and modern technical challenges while aiming to expand research horizons in advanced computing and emerging technologies.