КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.29.868
УДК 004.8

Артеменко Андрій Олександрович
бакалавр спеціальності 121 «Інженерія програмного забезпечення»
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID ID: 0009-0006-8047-1030
st7436326@stud.duikt.edu.ua

Худік Богдан Олександрович
доктор філософії, доцент кафедри інженерії програмного забезпечення
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID ID: 0009-0005-1914-6418
b.khudik@duikt.edu.ua

ОРГАНІЗАЦІЯ БЕЗПЕЧНОГО ФУНКЦІОНУВАННЯ ГОЛОСОВИХ АГЕНТІВ НА БАЗІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Анотація. Станом на сьогодні, набуває поширення практика впровадження голосових агентів на базі LLM (англ. «Large Language Models» — великі мовні моделі) у корпоративні Це створює нові виклики та небезпеки для архітектури програмного забезпечення, а також безпеки інформаційних систем. Традиційні підходи до розробки не враховують специфіку інтеграції LLM як компонента, що може бути скомпрометований зловмисниками через промпт-ін'єкції та інші атаки на рівні обробки природної мови. Стаття представляє огляд архітектурного підходу до безпечної інтеграції голосових агентів, що базується на принципі розділення основних бізнес-правил від логіки взаємодії з мовними моделями. Запропонована архітектура дозволяє побудувати багаторівневу систему захисту через: 1) реалізацію принципу найменших привілеїв для LLM-компонентів; 2) незалежну валідацію бізнес-операцій на рівні доменних сервісів; 3)використання великої мовної моделі як медіатора між користувачем та критично важливими системами. Дослідження, наведене у даній статті, демонструє ефективність архітектурного підходу у протидії типовим атакам –зокрема: промпт-ін'єкціям з метою обходу авторизації; діям, направленим на витік чутливих даних; несанкціонованим операціям, які здійснюються через function calling; діям, направленим на порушення цілісності бізнес-процесів тощо. Медіаторна архітектура значно спрощує розуміння та управління системою завдяки чіткому розподілу ролей між компонентами. Така архітектура підвищує передбачуваність системи та забезпечує можливість застосування традиційних методів валідації та аудиту. Архітектурна ізоляція LLM від критичних операцій дозволяє організаціям впроваджувати провідні AI-технології (від англ. «AI technologies» — технології штучного інтелекту) без кардинальної перебудови систем безпеки, які на даний момент вже впроваджені у компанії. В той же час, забезпечується стійкість до нових вразливостей та гнучкість у виборі технологічних рішень. Результати дослідження мають практичне значення для розробки корпоративних систем з інтеграцією голосових агентів та сприяють формуванню безпечних практик використання великих мовних моделей у критично важливих застосунках компанії.

Ключові слова: голосові агенти; великі мовні моделі; архітектура програмного забезпечення; безпека інформаційних систем.

ВСТУП

Актуальність теми. Динамічний розвиток технологій сприяє активному впровадженню голосових агентів на базі великих мовних моделей в різноманітні інформаційні системи, які використовуються на базі компаній та корпорацій. Таке впровадження здійснюється з метою досягнення додаткової автоматизації процесів.

Після впровадження, велика мовна модель стає ключовою складовою в контексті дотримання бізнес-правил, а також з точки зору організації безпеки системи. У компанії з'являється можливість сфокусуватися на підвищенні безпеки систем та відкриває простір для точкових усунень вразливостей. Важливо зазначити, що в наявних дослідженнях не розглянуто архітектурні рішення, де великі мовні моделі виступають лише як посередники (медіатори) та є частиною загальної системи бізнес-правил, тоді як основна логіка реалізована в інших модулях. Розробка таких архітектур є актуальною задачею, особливо з огляду на поширення агентських систем.

Аналіз останніх досліджень і публікацій. Світова практика налічує велику кількість випадків запровадження політик безпеки у агентських системах на базі великих мовних моделей. Науковці Alessio Bucaioni, Martin Weyssow, Junda He, Yunbo Lyu та David Lo у публікації «A Functional Software Reference Architecture for LLM-Integrated Systems» представили попередню функціональну архітектуру з шарами Presentation, Application Logic, LLM Integration і Data Management, а також боковими компонентами Monitoring і Guardrail для організації безпеки, приватності і якості окремих сервісів. В свою чергу, науковці Wen Wang, Zhenyue Zhao та Tianshu Sun в праці «Customizing Large Language Models for Business Context: Framework and Experiments» описують основні принципи інтеграції агентів в існуючі системи з комплексним доменом, але запропоноване рішення не бере до уваги валідацію дій LLM системою, а пропонує покладатися на валідацію людиною. В даному розрізі є нерозкритим є питання інтеграції агентів з супутніми сервісами. В результаті такої інтеграції утворюються системи, в яких відбувається комунікація саме у реальному часі — а, отже, валідація людиною не є оптимальним рішенням.

Мета і задачі дослідження. Метою дослідження є проектування архітектури програмного забезпечення, яка дозволяє виокремити основу домену від гнучких бізнес-правил діалогу з користувачем (тим самим, роблячи систему значно менш вразливою від атак на агентську частину, на кшталт промпт-ін'єкцій). Для досягнення поставленої мети дослідження, було сформоване наступне завдання:

1. розглянути способи інтеграції агентів в існуючі системи з можливістю забезпечення подальшого контролю над супутніми сервісами;
2. спроєктувати архітектуру для побудови програмного забезпечення, в якому перевірки безпеки закладені окремо від правил для великих мовних моделей.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Швидке впровадження голосових агентів на базі великих мовних моделей у корпоративні системи створює нові виклики для архітектури програмного забезпечення. Традиційні підходи до розробки не враховують специфіку інтеграції таких моделей як компонента, який може бути скомпрометованим зловмисниками через промпт-ін'єкції та інші атаки на рівні обробки природної мови. Це вимагає перегляду архітектурних рішень з метою мінімізації ризиків безпеки.

Забезпечення цілісності дотримання правил у корпоративному програмному забезпеченні набуває критичного значення при інтеграції агентських систем. Корпоративні системи оперують чутливими даними та виконують бізнес-критичні операції, тому будь-яка вразливість може призвести до значних фінансових втрат або порушення конфіденційності. Відокремлення основних бізнес-правил від логіки взаємодії з великими мовними моделями дозволяє створити додатковий рівень захисту, при якому навіть у випадку компрометації

агентської частини, критичні операції залишаються під контролем перевірених модулів системи. Такий підхід забезпечує принцип найменших привілеїв для компонентів великої мовної моделі та дозволяє застосовувати традиційні методи валідації та авторизації незалежно від поведінки мовної моделі.

Агентські системи з голосовим інтерфейсом користувача демонструють значний потенціал для трансформації способів взаємодії з корпоративними застосунками. Такі системи характеризуються природним та інтуїтивним способом доступу до складних систем, ліквідуючи при цьому «бар'єри» для користувачів та підвищуючи продуктивність роботи. Голосові агенти можуть обробляти складні запити в реальному часі, інтегруватися з множиною backend-сервісів та надавати персоналізовані відповіді на основі контексту користувача. Однак, даний підхід вимагає ретельного проектування архітектури, де велика мовна модель частково виступає як контролер бізнес-логіки — але, в той же час, містить в собі відомості про домен.

Сучасні підходи до інтеграції LLM у агентських системах базуються на механізмах function calling — технології, яка дозволяє мовним моделям взаємодіяти із зовнішніми інструментами через структуровані запити. Цей процес передбачає генерацію мовними моделями JSON-об'єктів із параметрами, що обробляються на стороні сервісу, котрий здійснює виклик мовної моделі. Після чого, результат для необхідного інструменту надсилається з наступним викликом API-моделі (від англ. «Application Programming Interface models» — моделі прикладного програмного інтерфейсу). На рис. 1 зображено отримання даних великою мовною моделлю за допомогою function calling та генерація результату на базі отриманих даних.

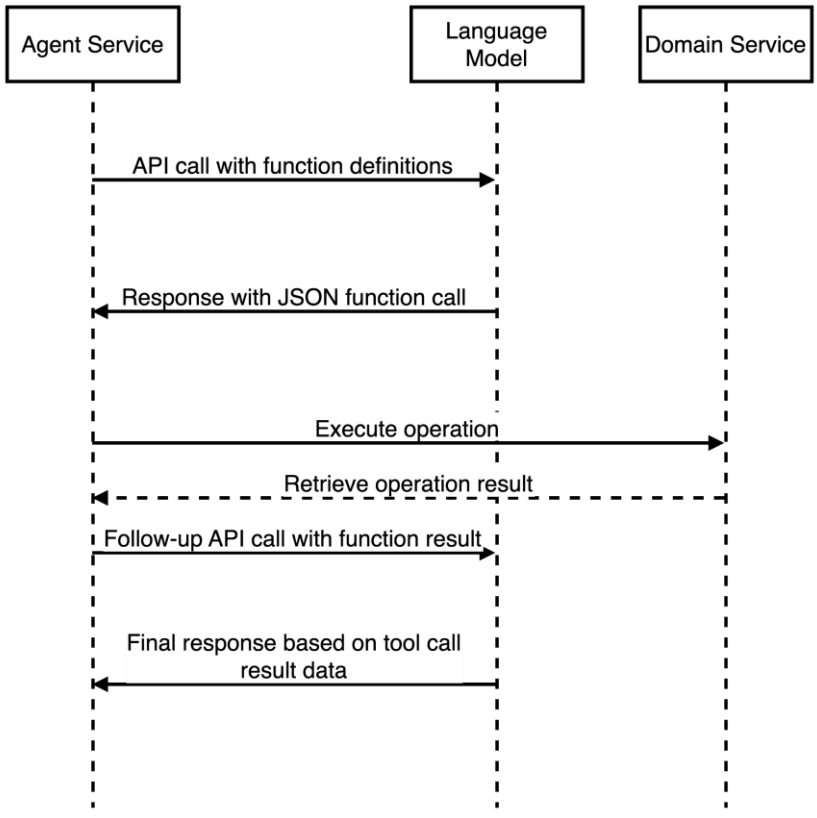

Рис. 1. Діаграма послідовності function calling

Ще одним способом надати агентському рішенню доступ до зміни стану системи через виклик структурованих команд є протокол MCP (Model Context Protocol). Цей протокол представляє собою відкритий стандарт, розроблений компанією Anthropic для забезпечення так званої «безшовної» інтеграції між великими мовними моделями та зовнішніми джерелами даних і інструментами. Протокол використовує архітектуру «клієнт-хост-сервер», де хост-процес створює та керує множинними клієнтськими екземплярами, кожен з яких підтримує ізольоване з'єднання з окремим сервером. Протокол MCP базується на JSON-RPC 2.0-повідомленнях та забезпечує стандартизований спосіб обміну контекстом між компонентами системи через три основні примітиви: ресурси, промпти та інструменти (функції для виконання LLM) [17].

Протокол реалізує систему узгодження можливостей, в якій клієнти та сервери явно декларують свої підтримувані функції під час ініціалізації, що забезпечує чітке розуміння доступного функціоналу та підтримує розширюваність протоколу. Ключовою особливістю протоколу MCP є принцип ізоляції, згідно з яким сервери не мають доступу до повної історії комунікації та не можуть мати доступ до конфігурації моделі, що організовує безпеку та дозволяє хост-процесу зберігати межі між компонентами.

Голосовий агент — це автономна програмна система, яка в реальному часі перетворює мовні запити користувача на текст, аналізує їх за допомогою великої мовної моделі і одразу генерує звукову відповідь природною мовою, мінімізуючи паузи між діалоговими ходами. Для розширення функціональності такі агенти використовують механізми виклику зовнішніх інструментів (function calling, Model Context Protocol тощо), які дозволяють оперативно збирати необхідні дані з бекенд-сервісів, запускати бізнес-логіку чи оновлювати стан системи. Завдяки цьому голосовий агент може виконувати складні запити, інтегруватися з базами даних, CRM чи IoT-пристроями та віддавати результат у вигляді природної мови з мінімальною затримкою, створюючи враження спілкування з людиною.

Фундаментальний принцип безпечної інтеграції голосових агентів полягає у чіткому відокремленню основних бізнес-правил від логіки взаємодії з мовними моделями. Цей підхід створює архітектурний бар'єр, що обмежує потенційний вплив атак на компоненти великих мовних моделей та на критично важливі операції системи. Функціональна референтна архітектура для LLM-інтегрованих систем пропонує багатошарову структуру з виділеними компонентами для моніторингу та захисту.

Принцип найменших привілеїв набуває особливого значення в контексті LLM-інтеграції, при якій мовні моделі отримують доступ лише до мінімально необхідного набору функцій для виконання своїх завдань. Валідація дій на рівні окремих сервісів, незалежно від рішень мовної моделі, забезпечує додатковий рівень захисту від потенційних уражень. Цей принцип дозволяє застосовувати традиційні методи авторизації та аудиту стосовно дій, ініційованих агентськими компонентами.

Для забезпечення коректності авторизації запитів можна застосовувати автентифікації окремих запитів, чи застосування сесій, що будуть містити в собі інформацію про конкретного співрозмовника. Такі сесії можуть існувати на рівні бізнес-логіки голосового асистента та стати обов'язковою складовою всіх викликів сторонніх сервісів. Такий підхід вирішує проблему недостатньої кількості даних в модулі голосового асистента в цілому, але, в той же час, дозволяє ізолювати мовні моделі від чутливих даних.

В рамках дослідження було створено архітектуру голосового асистента, що взаємодіє з CRM Manager Service, що має свої власні бізнес правила. Завдяки коректній організації роботи Voice Agent Application Layer, є можливість напряму керувати сесією та набором інструментів, які доступні для виклику великою мовною моделлю.

ISSN 2663 - 4023

Інтегруючи Application Layer напряму з pipeline, забезпечується простота системи, що дозволяє уникнути створення незграбних абстракцій та забезпечити достатній рівень контролю над всією системою.

Детально архітектуру спроєктованої системи показано на рис. 2.

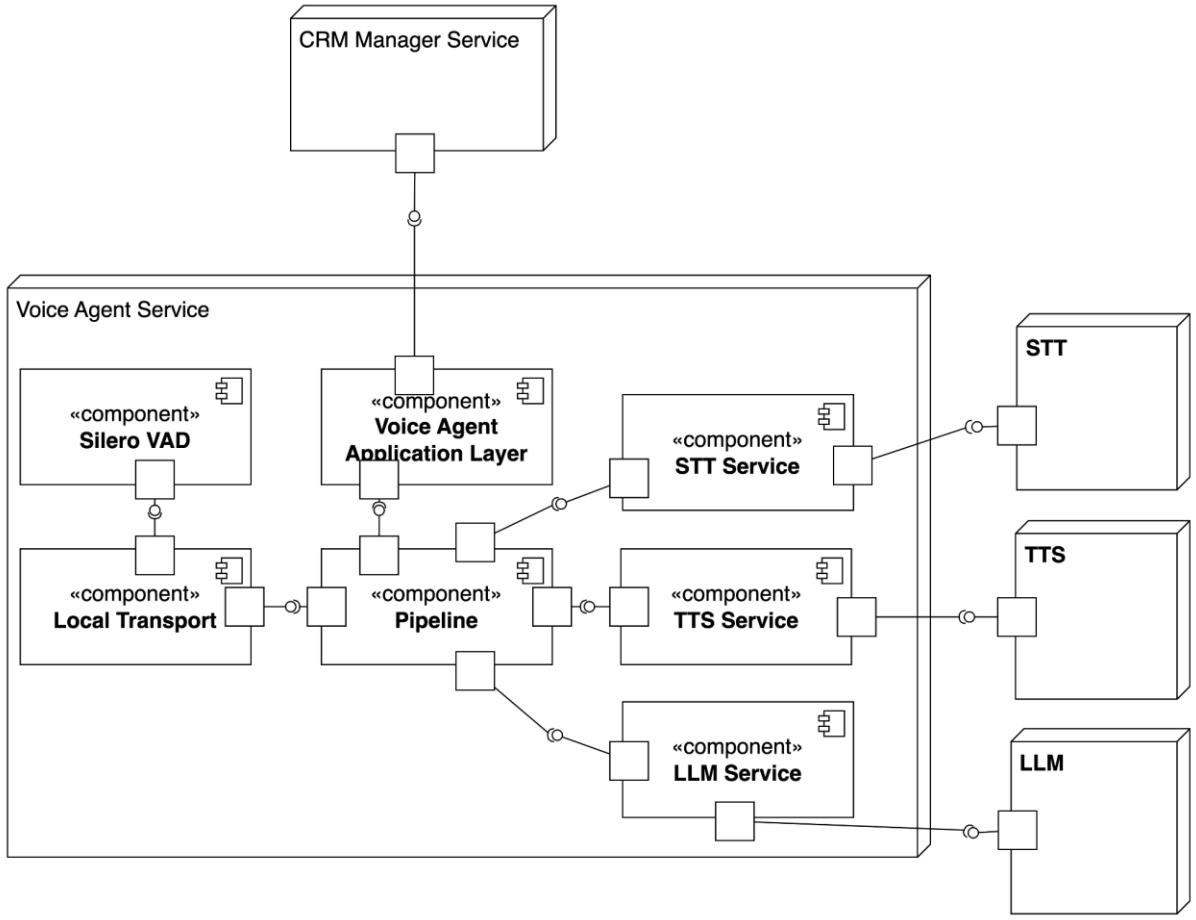

Рис. 2. Діаграма компонентів системи голосового асистента

Фундаментальним принципом розробленої архітектури є чітке розділення основних бізнес-правил від логіки взаємодії з великими мовними моделями. Цей підхід створює архітектурний бар'єр, що обмежує потенційний вплив атак на критично важливі операції системи через LLM-компоненти. Розділення реалізується через виділення окремого шару Voice Agent Application Layer, який виступає посередником між мовною моделлю та доменними сервісами.

Даний принцип безпосередньо пов'язаний з принципом найменших привілеїв та валідацією на рівні окремих сервісів, створюючи багаторівневу систему захисту. Така архітектура забезпечує, що навіть у випадку повної компрометації агентської частини, критичні бізнес-операції залишаються під контролем перевірених модулів системи.

У контексті інтеграції великих мовних моделей, принцип найменших привілеїв набуває особливого значення. Мовні моделі отримують доступ лише до мінімально необхідного набору функцій для виконання своїх завдань, що значно знижує ризики безпеки при потенційній компрометації агентської частини.

Цей принцип реалізується через архітектурне розділення та підсилюється механізмами сесійної авторизації. Application Layer контролює набір інструментів, доступних для

виклику великої мовної моделі, та забезпечує, що кожен запит містить необхідну інформацію про контекст користувача без надання прямого доступу до чутливих даних.

Валідація дій на рівні окремих доменних сервісів, незалежно від рішень мовної моделі, забезпечує додатковий рівень захисту від потенційних зловживань. Ця теза доповнює архітектурне розділення та створює можливість застосування традиційних методів авторизації та аудиту до дій, ініційованих агентськими компонентами.

Кожен доменний сервіс (наприклад, CRM Manager Service) містить власні бізнес-правила та механізми валідації, що працюють незалежно від логіки голосового агента. Це забезпечує цілісність бізнес-процесів навіть у випадку некоректної поведінки бізнес-шару голосового асистента.

Для забезпечення ще більшої ізольованості та передбачуваності агентської поведінки, пропонується використання архітектурного підходу з багатоагентними системами, де кожен етап обробки запиту виконується спеціалізованим агентом із чітко визначеним порядком дій. Такий підхід дозволяє сегментувати складні завдання на послідовні ланцюги операцій, де кожен агент має доступ лише до обмеженого набору інструментів та доменних знань, що суттєво знижує поверхню атаки та ризики компрометації [2], [3].

У сучасних дослідженнях все частіше наголошується на перевагах ієрархічної організації агентів, які виконують окремі функції у чітко визначеній послідовності. Це дозволяє не лише підвищити безпеку, але і забезпечити більшу передбачуваність роботи системи. Ієрархічна організація агентів із механізмами валідації проміжних станів уможливлює дотримання глобальних обмежень навіть при обробці складних багатоетапних запитів [3]. Кожен агент у такій системі виконує чітко визначену функцію:

- ініціалізація контексту — аналіз вхідного запиту та визначення необхідних доменних сервісів для виклику через інструменти;
- виконання доменних операцій — взаємодія з бекенд-сервісами через стандартизовані API;
- генерація відповіді — формування текстового/голосового відгуку з урахуванням контексту [3].

Така послідовність етапів усуває необхідність надання LLM прямого доступу до критичних інструментів, оскільки кожен агент працює в рамках свого ізольованого середовища з обмеженим набором привілеїв [11].

Ключовим аспектом є динамічне формування набору доступних інструментів для кожного агента на основі:

- поточного етапу workflow (робочого процесу системи);
- рівня авторизації користувача;
- історичного контексту взаємодії.

Введення передбачуваних моделей для оцінки ризиків кожного виклику інструменту дозволяє проактивно блокувати потенційно небезпечні операції. Наприклад, агент нормалізації запитів може мати доступ лише до семантичного парсера, тоді як агент виконання операцій — виключно до доменних API з жорсткою валідацією вхідних параметрів [4].

Ще однією важливою перевагою послідовної багатоагентної архітектури є зменшення залежності від складних процедур донавчання великих мовних моделей. Завдяки вузькій спеціалізації кожного агента та стандартизованим шаблонам промптів, система може ефективно працювати навіть із моделями без доменної адаптації.

Послідовна архітектура усуває необхідність складного налаштування моделей завдяки спеціалізації агентів (оскільки кожен агент вирішує вузьке завдання, що не

вимагає універсальності великої мовної моделі), а також завдяки контекстній ізоляції, що забезпечує обмеження доменних знань у межах одного агента [9].

Велика мовна модель у запропонованій архітектурі виступає як інтелектуальний медіатор (посередник), що перетворює природномовні запити користувача в структуровані виклики API При цьому, LLM залишається ізольованою від критичних даних домену. Такий підхід об'єднує всі попередньо зазначені принципи, створюючи систему, де велика мовна модель є потужним інтерфейсом, а не носієм критичної бізнес-логіки.

ВИСНОВКИ

Запропонований підхід дозволяє організаціям впроваджувати передові AI-технології без кардинальної перебудови існуючих систем безпеки та бізнес-процесів. Великі мовні моделі стають потужним інструментом для покращення користувацького досвіду, зберігаючи при цьому надійність та безпеку корпоративних процесів. Це особливо важливо в контексті швидкого розвитку технологій великих мовних моделей, де нові вразливості можуть з'являтися регулярно. Архітектурна ізоляція забезпечує стійкість системи до кіберзагроз, дозволяючи організаціям користуватися перевагами AI-технологій з мінімальними ризиками для критично важливих операцій.

Запропонована архітектура також забезпечує гнучкість у виборі та заміні LLM-компонентів, оскільки основна бізнес-логіка залишається незалежною від конкретної реалізації мовної моделі. Це дозволяє організаціям адаптуватися до технологічних змін без значних витрат на рефакторинг існуючих систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bucaion, A. (2025). *A functional software reference architecture for LLM-integrated systems*.
2. Chan, C. M. (2024). *AgentMonitor: A plug-and-play framework for predictive and secure multi-agent systems*.
3. Chang, E., & Geng, L. (2025). *SagaLLM: Context management, validation, and transaction guarantees for multi-agent LLM planning*.
4. Chun, Y. E. (2025). *CREFT: Sequential multi-agent LLM for character relation extraction*.
5. Ginart, A. A. (2024). *Asynchronous tool usage for real-time agents*.
6. Guran, N. (2024). *Towards a middleware for large language models*.
7. Kalhor, B., & Das, S. (2023). *Evaluating the security and privacy risk postures of virtual assistants*.
8. Kim, M. (2024). *A multi-agent approach for REST API testing with semantic graphs and LLM-driven input*.
9. Li, X. (2024). *A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges*.
10. Ma, H. (2025). *Coevolving with the other you: Fine-tuning LLM with sequential cooperative multi-agent reinforcement learning*.
11. Nunez, A. (2024). *AutoSafeCoder: A multi-agent framework for securing LLM code generation through static analysis and fuzz testing*.
12. Suo, X. (2025). *Signed-Prompt: A new approach to prevent prompt injection attacks against LLM-integrated applications*.
13. Szeider, S. (2025). *MCP-Solver: Integrating language models with constraint programming systems*.
14. Wen, W., Zhenyue, Z., & Tianshu, S. (2023). *Customizing large language models for business context: Framework and experiments*.
15. Yi-Chang, C. (2024). *Enhancing function-calling capabilities in LLMs: Strategies for prompt formats, data integration, and multilingual translation*.
16. Zhang, Y. (2024). *Towards efficient LLM grounding for embodied multi-agent collaboration*.
17. Modelcontextprotocol.io. (2025, March 26). *Specification (latest version)*. https://modelcontextprotocol.io/specification/2025-03-26

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023

Andrii Artemenko
Bachelor, Software Engineering
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID ID: 0009-0006-8047-1030
st7436326@stud.duikt.edu.ua

Bohdan Khudik
PhD, Docent at Department of Software Engineering
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID ID: 0009-0005-1914-6418
b.khudik@duikt.edu.ua

ENSURING SECURE OPERATING OF VOICE AGENTS BASED ON THE LARGE LANGUAGE MODELS

Abstract. The rapid deployment of voice agents based on LLM (Large Language Models) in corporate systems creates new challenges for the software architecture and information system security. Traditional development approaches do not take into account the specifics of LLM integration as a component that can be compromised by attackers through prompt injections and other attacks at the natural language processing level. The article presents an architectural approach to secure voice agent integration based on the principle of separation of the basic business rules from the logic of interaction with language models. The proposed architecture creates a multi-level security system via 1. the implementation of the principle of least privilege for LLM components: 2. independent validation of business operations at the domain service level; 3. the usage of a large language model as a mediator between the user and critical systems. The research demonstrates the effectiveness of the architectural approach in countering typical attacks, including prompt injections to bypass authorisation, sensitive data leakage, unauthorised operations via function calling, and violation of business process integrity. The mediator architecture greatly simplifies the understandability and manageability of the system due to a clear division of roles between components, which increases the system predictability, and enables the use of traditional validation and audit methods. The architectural isolation of LLM from critical operations allows organisations to implement advanced AI technologies without a major overhaul of existing security systems, ensuring resilience to new vulnerabilities and flexibility in choosing of the technological solutions. The results of the research are of practical importance for the development of corporate systems with voice agent integration. Also, the resultscontribute to the forming of secure practices for the usage of large language models in applications of critical importance.

Keywords: voice agents; large language models; software architecture; information systems security.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Bucaion, A. (2025). *A functional software reference architecture for LLM-integrated systems*.
2. Chan, C. M. (2024). *AgentMonitor: A plug-and-play framework for predictive and secure multi-agent systems*.
3. Chang, E., & Geng, L. (2025). *SagaLLM: Context management, validation, and transaction guarantees for multi-agent LLM planning*.
4. Chun, Y. E. (2025). *CREFT: Sequential multi-agent LLM for character relation extraction*.
5. Ginart, A. A. (2024). *Asynchronous tool usage for real-time agents*.
6. Guran, N. (2024). *Towards a middleware for large language models*.
7. Kalhor, B., & Das, S. (2023). *Evaluating the security and privacy risk postures of virtual assistants*.
8. Kim, M. (2024). *A multi-agent approach for REST API testing with semantic graphs and LLM-driven input*.
9. Li, X. (2024). *A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges*.
10. Ma, H. (2025). *Coevolving with the other you: Fine-tuning LLM with sequential cooperative multi-agent reinforcement learning*.
11. Nunez, A. (2024). *AutoSafeCoder: A multi-agent framework for securing LLM code generation through static analysis and fuzz testing*.

12. Suo, X. (2025). *Signed-Prompt: A new approach to prevent prompt injection attacks against LLM-integrated applications*.
13. Szeider, S. (2025). *MCP-Solver: Integrating language models with constraint programming systems*.
14. Wen, W., Zhenyue, Z., & Tianshu, S. (2023). *Customizing large language models for business context: Framework and experiments*.
15. Yi-Chang, C. (2024). *Enhancing function-calling capabilities in LLMs: Strategies for prompt formats, data integration, and multilingual translation*.
16. Zhang, Y. (2024). *Towards efficient LLM grounding for embodied multi-agent collaboration*.
17. Modelcontextprotocol.io. (2025, March 26). *Specification (latest version)*. https://modelcontextprotocol.io/specification/2025-03-26