УДК 004.056:004.8

АРХІТЕКТУРА ШЛЮЗУ БЕЗПЕКИ ДЛЯ ЗАХИСТУ МУЛЬТИАГЕНТНИХ СИСТЕМ НА ОСНОВІ ХМАРНИХ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

А. В. Хав'юк[1], І. В. Стьопочкіна[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

Розглянуто проблему забезпечення безпеки та керованості мультиагентних систем штучного інтелекту, що спираються на хмарні великі мовні моделі. Показано, що захист таких систем не може обмежуватися фільтрацією текстових запитів, оскільки агенти використовують пам'ять, викликають зовнішні інструменти та можуть виконувати дії з потенційно критичними наслідками. Метою роботи є розробка моделі архітектури проміжного шлюзу безпеки, який забезпечує контроль взаємодії між агентами та інструментами виконання. На основі аналізу існуючих рішень визначено ключові класи загроз і запропоновано багатошарову модель шлюзу, що поєднує верифікацію ідентичності агента, обмеження частоти викликів, рольовий контроль доступу, перевірку параметрів запиту, контроль вихідних потоків даних, узгодження критичних дій та журналювання. Для демонстрації практичної реалізовності підходу розроблено прототип системи. Запропонована модель архітектури може бути використана як інтегруючий компонент захисту мультиагентних LLM-систем, що забезпечує централізований контроль дій агентів, обмеження їхніх повноважень і простежуваність рішень.

Ключові слова: Agentic AI, мультиагентні системи, великі мовні моделі, шлюз безпеки, рольовий контроль доступу, prompt injection, спостережуваність агентів

Вступ

Розвиток великих мовних моделей (LLM) спричинив появу нового класу інтелектуальних систем — агентних AI (Agentic AI), які не лише відповідають на запити, а й планують послідовність дій, викликають зовнішні інструменти та виконують багатоетапні задачі за умов обмеженого нагляду людини [1]. Доступність хмарних LLM та можливість інтеграції з іншими послугами хмарних середовищ зробила дані системи привабливими для практичних застосувань, зокрема і в кібербезпеці — для моніторингу подій, аналізу журналів, реагування на інциденти та автоматизованої підтримки аналітика центру кіберзахисту [2].

Разом зі зростанням функціональних можливостей зростає і кількість ризиків. Агентна система, яка може блокувати мережеві з'єднання, читати журнали, звертатись до зовнішніх API, потребує значно жорсткішого контролю, ніж звичайна генеративна модель. У літературі дедалі частіше використовується не лише поняття безпеки, а й керованості — тобто обмеження повноважень, підзвітності та контролю над життєвим циклом дій агента [3]. Аналіз джерел показує, що існуючі підходи до захисту мультиагентних систем здебільшого є фрагментованими: одні зосереджуються на захисті підказок, інші на ізоляції виконання, на журналюванні подій, проте цілісного механізму, що поєднував би всі ці рівні, поки що не сформовано [4]. Саме цей розрив між наявними рішеннями та потребою в інтегрованому захисному контурі обґрунтовує актуальність дослідження. Метою роботи є розробка моделі архітектури проміжного шлюзу безпеки, який поєднує контроль ідентичності агента, рольову авторизацію, перевірку параметрів виклику, контроль вихідних даних, узгодження критичних дій та аудит у єдиному контурі керування взаємодією між агентами та інструментами.

1. Agentic AI як об'єкт захисту

Принциповою відмінністю Agentic AI від звичайних LLM-застосунків є те, що поряд із самою моделлю в системі присутні шар оркестрації, пам'ять, виклики інструментів та механізми оцінювання проміжних рішень [3]. Агент є не просто «розширеним чат-інтерфейсом», а виконавцем складеного процесу, де ризики виникають не лише у вхідному запиті, а й у переходах між компонентами, у записах до пам'яті та у викликах зовнішніх сервісів. Перехід до багатоагентної організації посилює цю проблему: компрометація ідентичності одного агента здатна вплинути на роботу всієї системи [5].

У більшості сучасних сценаріїв агентні системи спираються на хмарні LLM, передаючи до моделі значно ширший контекст, ніж звичайний застосунок: проміжні міркування, фрагменти внутрішніх даних та результати викликів інструментів. Хмарна залежність також формує специфічну границю довіри: жодна підказка у системному повідомленні не може вважатись надійним механізмом примусового виконання політик — адже вона передається

Таблиця 1. Класи загроз та шари захисту шлюзу

Клас загрози	**Шар шлюзу та функція**
Підробка ідентичності агента	Верифікація за спільним секретом у заголовку запиту
Зловживання інструментами	RBAC: матриця дозволів «агент → інструмент»
Аномальна поведінка агента	Обмеження частоти викликів (стан у сховищі)
Ін'єкційні атаки (prompt injection)	Payload Guard: сигнатурна перевірка аргументів
Витік даних у запиті	Memory Guard: пошук токенів, ключів, PII
Витік даних у відповіді інструмента	Memory Guard на виході (egress-фільтр)
Незворотні дії	Human-in-the-loop: підтвердження адміністратора
Відсутність простежуваності	Журнал аудиту з ідентифікатором трасування

тій самій моделі, яка потенційно може бути об'єктом ін'єкційного впливу. Саме тому виконання захисних перевірок поза межами моделі — на рівні окремого компонента — є принциповим архітектурним рішенням.

2. Загрози та обмеження існуючих підходів

2.1. Класи загроз для мультиагентних систем

На основі аналізу робіт [3, 5, 6, 7] виділено чотири ключові класи загроз для мультиагентних систем на основі хмарних LLM: підробка ідентичності агента, порушення рольового контролю доступу, ін'єкційні атаки через зовнішні дані та витік чутливої інформації через контекст або відповіді інструментів. Окремо виділяють також ризики, пов'язані з пам'яттю агента: неконтрольовані записи в неї можуть впливати на подальшу поведінку системи навіть тоді, коли початковий шкідливий стимул уже відсутній [5]. Зіставлення виділених класів загроз із шарами захисту наведено в табл. 1.

2.2. Обмеження існуючих підходів до захисту

Аналіз сучасних джерел показує, що захист агентних систем не може спиратися на один вузький механізм. Традиційні засоби фільтрації змісту створюють конфлікт між корисністю та безпекою: одна й та сама команда може бути припустимою в одному сценарії й небезпечною в іншому [4]. Тому для агентних систем доцільніше перевіряти не лише зміст запиту, а й контекст виконання, права агента, тип інструмента та очікуваний наслідок дії. Тому пропонується розглянути принцип посередництва, за яким агент не повинен мати прямого та безумовного доступу до всіх можливих інструментів, а між агентом і критичними операціями має існувати окремий перевіряючий компонент [7].

Окрему групу підходів становить підвищення спостережуваності та підзвітності агентних систем [8], де як основні механізми пропонується використовувати ідентифікатори агентів, моніторинг у реальному часі та реєстрацію дій. Однак варто наголосити, що для агентних систем недостатньо аналізувати лише окрему модель або окремий сценарій взаємодії, оскільки ризики виникають на рівні всієї системи, де поєднуються міркування, пам'ять, інструменти, зовнішні сервіси та зворотний зв'язок [9]. На практиці це означає, що навіть наявність окремих захисних механізмів не гарантує безпеки, якщо вони не зведені до єдиного контуру керування. Окремо слід зазначити, що звичайний API шлюз, який є типовим компонентом хмарних архітектур, не вирішує цієї задачі: він контролює технічні параметри HTTP-викликів (квоти, маршрутизацію, базову авторизацію), але не оперує поняттями ролі агента в мультиагентній системі, семантики аргументів виклику чи вмісту відповіді інструмента.

Таким чином, існуючі рішення утворюють перехід від захисту окремого текстового інтерфейсу до захисту всієї багатокрокової системи, однак універсального рішення, яке поєднувало б дотримання політик, посередництво інструментів, гігієну пам'яті та повноцінний аудит, у літературі поки що не простежується.

3. Архітектура шлюзу безпеки як проміжного контуру керування

3.1. Концепція проміжного шлюзу та новизна

Виявлені обмеження існуючих підходів обґрунтовують доцільність побудови окремого захисного компонента — проміжного шлюзу безпеки, через який обов'язково проходять усі виклики інструментів від агентів. Концептуально такий шлюз реалізує принцип посередництва: агент не має прямого доступу до інструментів, натомість кожен виклик оцінюється за набором правил перед виконанням та після нього [7]. Це відповідає класичним принципам інформаційної безпеки — найменших привілеїв, розподілу обов'язків та ізольованого виконання, які тепер адаптуються до середовища агентних ШІ (Agentic AI).

Новизна запропонованої моделі полягає в тому, що шлюз перевіряє не лише технічні параметри HTTP-виклику, як це робить звичайний API шлюз, а оперує поняттями агентного контексту: він зіставляє заявлену агентом роль із попередньо узгодженою ідентичністю, перевіряє відповідність викликаного інструмента ролі агента, аналізує семантику аргументів запиту на наявність чутливих даних та сигнатур ін'єкційних впливів, а також контролює вміст відповідей інструментів перед їх передачею назад до контексту LLM. У межах одного компонента по-

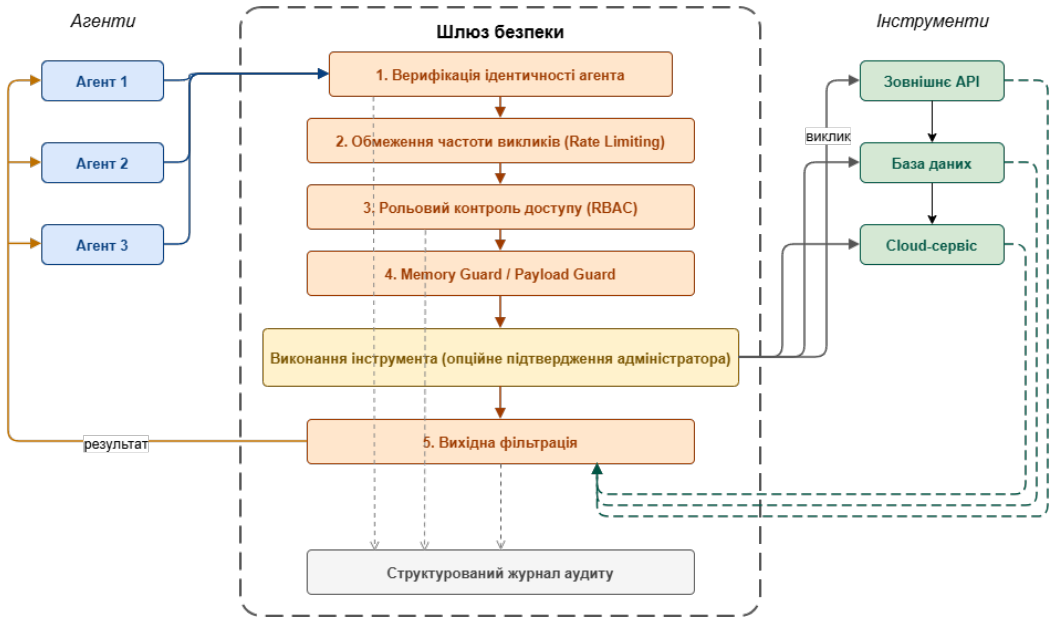

Рис. 1. Архітектура проміжного шлюзу

єднуються механізми, які зазвичай розглядаються в літературі окремо: контекстно-залежна авторизація, посередництво інструментів, контроль пам'яті, узгодження критичних дій та аудит. Загальну схему запропонованої архітектури наведено на рис. 1.

3.2. Послідовність шарів захисту

Кожен шар, наведений у табл. 1, вмонтовується у шлюз як окрема перевірка, що виконується послідовно. Запит від агента приходить у вигляді HTTP-повідомлення з полями `agent_id`, `tool_name`, `arguments` та заголовком, що містить попередньо спільний секрет агента. Шари 1 – 4 виконуються до фактичного виклику інструмента, шар 5 (вихідна фільтрація) — після нього, перед поверненням результату агенту. Якщо будь-який шар фіксує порушення, виконання припиняється з відповідним кодом помилки, причина зберігається в журналі.

Memory Guard на вході та фільтрування на виході використовують єдиний набір регулярних виразів для виявлення чутливих даних: адрес електронної пошти, номерів кредитних карт, ідентифікаторів соціального страхування, ключів API хмарних провайдерів, токенів облікових записів розробника та фрагментів приватних ключів. Payload Guard окремо перевіряє аргументи на наявність типових сигнатур ін'єкційних впливів. Для критичних дій із незворотним ефектом шлюз не виконує операцію автоматично, а повертає статус, що вимагає підтвердження адміністратора [10], що узгоджується з концепцією обмеженої автономії.

3.3. Спостережуваність та аудит

Кожне рішення шлюзу фіксується у структурованому журналі аудиту [8]. Запис містить унікальний ідентифікатор трасування, ідентифікатор агента, назву інструмента, статус рішення (дозволено, відхилено, очікує підтвердження) та причину. Оскільки кожна дія агента в будь-якому разі проходить через шлюз, такий журнал стає повним відтворюваним слідом роботи всієї агентної системи, який згодом використовується для аудиту, аналізу інцидентів та подальшого вдосконалення політик безпеки.

3.4. Прототип системи

Для перевірки практичної реалізовності запропонованої моделі архітектури розроблено прототип системи автоматизованого центру кіберзахисту (Automated SOC) на базі трьох агентів, що використовують Vertex AI та модель `gemini-2.5-pro`. Перший агент є аналізатором подій, який відповідає за читання записів журналу подій безпеки та делегування підозрілих індикаторів. Другий є агентом аналізу загроз, який перевіряє індикатори через зовнішні сервіси репутації (VirusTotal, MetaDefender). Третій є агентом реагування, який додає шкідливі IP-адреси у блокувальні правила, для чого використовується запис у Google Firestore. Сам шлюз розгорнуто як окремий сервіс у Google Cloud Run з відповідними кінцевими точками для перевірки вхідних викликів та контролю вихідних потоків, лічильники обмеження кількості запитів зберігаються у Google Cloud Firestore, журнал аудиту — у Google Cloud Logging. Тестування підтвердило працездатність моделі: усі визначені у табл. 1 класи загроз коректно перехоплюються відповідними шарами шлюзу, а легітимні виклики виконуються без затримок, помітних для користувача.

Пілотна перевірка підтвердила працездатність запропонованої багаторівневої архітектури: у 27 із 30 тестових сценаріїв система сформувала коректну реакцію, зокрема забезпечила обмеження всіх перевірених шкідливих або потенційно небезпечних запитів на рівні шлюзу безпеки або системних інструкцій мовної моделі. Усі 23 шкідливі або потен-

ційно небезпечні запити були обмежені на одному з рівнів захисту: 8 — безпосередньо шлюзом безпеки, 15 — на рівні системних інструкцій мовної моделі. Всі легітимні запити були коректно пропущені. Виявлені 3 хибно позитивні спрацювання зумовлені неповною реалізацією функціоналу агентів у прототипі, тоді як 1 хибно негативний результат стосувався сценарію автоматизованого масового блокування IP-адрес. Це вказує на необхідність посилення human-in-the-loop контролю для дій, здатних впливати на доступність системи.

4. Місце шлюзу в загальній моделі захисту

Запропонована архітектура є інтегруючим компонентом, що поєднує контекстно-залежний контроль дій, посередництво між агентом та інструментами, спостережуваність та принцип обмеженої автономії у межах одного компонента, уникаючи типової фрагментації механізмів захисту. Водночас шлюз не замінює інших засобів — зокрема, не вирішує задач захисту моделі від навчальних атак чи конфіденційного виконання інференсу — і повинен використовуватись у поєднанні з ними.

З точки зору архітектурних патернів безпеки, запропонований підхід є реалізацією принципу нульової довіри (zero trust) стосовно агентів: жоден агент не вважається довіреним за замовчуванням, кожен виклик верифікується незалежно, і доступ надається лише до інструментів, явно дозволених для конкретної ролі. Це узгоджується з тенденціями у захисті хмарних архітектур, де периметрова безпека поступається місцем перевірці на рівні кожної транзакції.

Серед очевидних обмежень такого підходу слід відзначити те, що ефективність сигнатурного аналізу шкідливих корисних навантажень принципово обмежена відомими шаблонами, а більш складні форми ін'єкційних атак потребують поведінкових методів виявлення. Крім того, статичні матриці рольового контролю не покривають сценаріїв, де роль агента має змінюватись динамічно залежно від контексту задачі. Ці обмеження визначають напрями подальшого розвитку запропонованого підходу.

Висновки

У роботі обґрунтовано необхідність побудови окремого захисного компонента для мультиагентних систем на основі хмарних LLM. Запропонована модель архітектури проміжного шлюзу безпеки поєднує верифікацію ідентичності, обмеження частоти викликів, рольовий контроль доступу, перевірку вхідних та вихідних даних, узгодження критичних дій з адміністратором та структуроване журналювання у межах одного компонента. Новизна моделі полягає в тому, що шлюз оперує поняттями агентного контексту — ролі агента, семантики аргументів та вмісту відповідей інструментів, що відрізняє його від класичних API Gateway, які працюють лише на рівні технічних параметрів HTTP-викликів. Працездатність моделі підтверджена прототипом системи автоматизованого SOC на базі Vertex AI та Google Cloud. Подальші дослідження планується спрямувати на розширення можливостей шлюзу в напрямку поведінкового виявлення складніших форм ін'єкційних атак та переходу від статичної матриці дозволів до контекстно-залежного контролю на основі стану середовища.

Перелік використаних джерел

1. The emerged security and privacy of LLM agent: A survey with case studies / F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, P. S. Yu // ACM Computing Surveys. — 2025. — Т. 58, № 6. — С. 1—36.
2. A Survey of Agentic AI and Cybersecurity: Challenges, Opportunities and Use-case Prototypes / S. J. Lazer, K. Aryal, M. Gupta, E. Bertino. — 2026. — arXiv: 2601.05293. — URL: https://arxiv.org/abs/2601.05293.
3. *Narajala V. S.*, *Narayan O.* Securing Agentic AI: A Comprehensive Threat Model and Mitigation Framework for Generative AI Agents. — 2025. — arXiv: 2504.19956. — URL: https://arxiv.org/abs/2504.19956.
4. A Framework for Formalizing LLM Agent Security / V. Siu, J. He, K. Montgomery, Z. Wang, N. Z. Gong, C. Wang, D. Song. — 2024. — arXiv: 2410.02644. — URL: https://arxiv.org/abs/2410.02644.
5. Navigating the Risks: A Survey of Security, Privacy, and Ethics Threats in LLM-based Agents / Y. Gan, Y. Yang, Z. Ma, P. He, R. Zeng, Y. Wang, S. Ji. — 2024. — arXiv: 2411.09523. — URL: https://arxiv.org/abs/2411.09523.
6. From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows / M. A. Ferrag, N. Tihanyi, D. Hamouda, L. Maglaras, A. Lakas, M. Debbah // ICT Express. — 2025.
7. Design Patterns for Securing LLM Agents Against Prompt Injections / L. Beurer-Kellner, B. Buesser, A.-M. Cretu, E. Debenedetti, D. Dobos, D. Fabian, V. Volhejn. — 2025. — arXiv: 2506.08837. — URL: https://arxiv.org/abs/2506.08837.
8. Visibility into AI Agents / A. Chan, C. Ezell, M. Kaufmann, K. Wei, L. Hammond, H. Bradley, M. Anderljung // Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT). — 2024. — С. 958—973.
9. Agentic AI Needs a Systems Theory / E. Miehling, K. N. Ramamurthy, K. R. Varshney, M. Riemer, D. Bouneffouf, J. T. Richards, W. Geyer. — 2025. — arXiv: 2503.00237. — URL: https://arxiv.org/abs/2503.00237.
10. *Navneet S. K.*, *Chandra J.* Rethinking Autonomy: Preventing Failures in AI-Driven Software Engineering. — 2025. — arXiv: 2508.11824. — URL: https://arxiv.org/abs/2508.11824.