ІЗОЛЯЦІЯ ДІЙ АГЕНТА: РІВНІ ОБМЕЖЕННЯ ПОБІЧНИХ ЕФЕКТІВ

Киселевич Володимир Володимирович
асистент кафедри комп'ютерних наук та інформаційних технологій
Житомирський державний університет імені Івана Франка, Україна

Постановка проблеми. Агентні програмні системи на основі великих мовних моделей відрізняються від діалогових тим, що частина їхнього виходу автоматично перетворюється на виклик зовнішньої функції, а саме на запис до файлової підсистеми, запит до мережі, зміну записів бази даних або надсилання повідомлення. Помилковий чи зловмисно нав'язаний текст сам собою не змінює стану зовнішнього світу, тому небезпечним він стає лише в момент, коли його перетворено на дію. Практика захисту таких систем натомість зосереджена переважно на перевірці виходу, тобто на класифікаторах підозрілих інструкцій, застереженнях у системній підказці та повторному опитуванні моделі. Такий бар'єр є ймовірнісним, бо він оцінює текст, а множина формулювань, які приводять до однієї й тієї самої дії, практично необмежена. Обмеження середовища виконання має іншу природу: дія, для якої у процесі агента немає ні системного виклику, ні мережевого сокета, ні права запису, не відбудеться незалежно від того, наскільки переконливо агента ввели в оману. Ця асиметрія відома окремим інженерним колективам, проте вона не впорядкована у шкалу, придатну для проєктних рішень, і саме брак такого впорядкування становить розглянуту проблему.

Аналіз останніх досліджень. Слабкість перевірки виходу задокументована емпірично. Бенчмарк непрямих інструкційних ін'єкцій InjecAgent охоплює 1054 тестові випадки, 17 інструментів користувача та 62 інструменти нападника і показує, що зовнішній вміст, який агент опрацьовує, є повноцінним каналом керування ним [1]. Динамічне середовище AgentDojo фіксує, що наявні атаки порушують частину властивостей безпеки, але не всі, і що передові моделі не виконують значної частки завдань навіть за відсутності атаки [2]. Отруєння інструментів переміщує нападника ще раніше, на етап реєстрації: набір MCPTox побудовано на 45 наявних серверах протоколу MCP і 353 автентичних інструментах, де зловмисні інструкції вбудовано у метадані інструмента [3]. Паралельно розвивається лінія системних обмежень. Архітектура IsolateGPT пропонує виконавчу ізоляцію застосунків усередині агентної системи замість спільного контексту з вільною взаємодією [4]. Пісочниця Sandlock обмежує код агента непривілейованими примітивами Linux з контролем на рівні системних викликів, тоді як контейнери та мікровіртуальні машини збільшують витрати на запуск і керування образами [5]. Система Progent подає привілей як політику із символьних правил щодо імен інструментів та їхніх аргументів і перевіряє кожен виклик детерміністичною процедурою, реалізуючи принцип найменших привілеїв [6]. Синхронна авторизація перед викликом інструмента дає медіанну

затримку 53 мс на 1000 вимірювань, а в живому змагальному стенді, що охопив 4437 рішень авторизації у 1151 сесії, соціальна інженерія проти моделі мала успіх у 74,6 відсотка випадків за дозвільної політики і нульовий успіх на 879 спробах за обмежувальної [7]. Контроль інформаційних потоків із динамічним стеженням за позначками формалізує клас властивостей, які можна забезпечити системно, а не текстово [8]. Виконання політик у момент виклику інструмента MCP знижує успішність атак з 40,0 до 5,0 відсотка проти 35,0 відсотка для найсильнішої суто підказкової основи, причому вилучення поширення позначок між кроками підвищує частку хибних пропусків на рівні викликів на 26,4 пункту [9]. Наведені тут і далі числові оцінки походять переважно з препринтів [5; 6; 7; 8; 10; 11; 12; 13], тому їх слід трактувати як попередні, до завершення рецензування.

Мета. Мета роботи полягає в тому, щоб обґрунтувати перенесення центру надійності агентної системи з оцінювання тексту на обмеження наслідків дії, упорядкувати відомі засоби ізоляції за обсягом наслідків, які вони фізично унеможливлюють, для кожного рівня вказати наслідки, не охоплені ним принципово, і сформулювати перевірну умову достатності вибраного рівня ізоляції для заданої задачі.

Виклад основного матеріалу. Розмежуємо вихід і дію. Агент породжує текст, а виконавче середовище відображає цей текст у виклик інструмента з конкретними аргументами. Перевірка виходу діє до відображення, її результат є оцінкою тексту, а показники успішності залежать від вибірки атак, на якій їх отримано. Обмеження середовища діє на саме відображення та на доступні йому ресурси, тому воно є детерміністичним щодо цілого класу дій. Звідси випливає практичний висновок: підвищення точності фільтра зменшує ймовірність небажаної дії, а вилучення відповідної можливості із середовища зводить цю ймовірність до нуля для всього класу. Систематичність ризикованих дій підтверджує емульована пісочниця ToolEmu, у якій на матеріалі 36 інструментів високого ризику та 144 тестових випадків 68,8 відсотка виявлених збоїв визнано правдоподібними в реальних умовах [10]. Відповідно порівнювати перевірку виходу та обмеження середовища за одним показником успішності атак некоректно, бо перший засіб змінює ймовірність події, а другий змінює склад можливих подій.

Запропоновано шкалу з шести рівнів обмеження побічних ефектів. Рівень вважається вищим за попередній, якщо множина наслідків, які він фізично унеможливлює, включає відповідну множину попереднього рівня. Рівень P0 означає відсутність обмежень: агент діє з правами процесу застосунку, і будь-який виклик, доступний цьому процесу, доступний агентові. Рівень P1 обмежує перелік інструментів, тобто агент викликає лише задекларовані функції, а решта дій недосяжна через відсутність відповідного перехідника. Цей рівень не охоплює зловживання дозволеним інструментом, причому вибір надлишково привілейованого інструмента є вимірюваним явищем: агенти обирають інструмент з вищими правами і переходять до нього після тимчасових відмов навіть тоді, коли достатньою є альтернатива з нижчими правами [11]. Рівень P2

обмежує аргументи і додає дозвільні правила, які перевіряються детерміністичною процедурою перед кожним викликом, тому обхід правила вимагає не переконливого формулювання, а помилки в самому правилі [6; 7]. Тут доречні політики, залежні від призначення дії, а не від формулювання виходу [14]. Ціну такого рівня можна виміряти: на рівні завдань зафіксовано 30,0 відсотка хибних відмов, зумовлених відмовами токенів найменших привілеїв, передбачених самим задумом, а не помилками правил, тоді як на розширеному наборі з 30 легітимних завдань правила не дали жодної хибної відмови [9].

Рівень P3 обмежує права в межах хоста, а саме окремого непривілейованого користувача, файлову підсистему лише для читання поза виділеним каталогом, обмежений перелік адрес для мережевих з'єднань і контроль на рівні системних викликів [5]. Рівень P4 переносить виконання в ізольоване середовище, тобто в контейнер або віртуальну машину з обмеженим доступом до основної системи [4]. Принципова межа цього рівня полягає в тому, що пісочниця сама є програмою з уразливостями: оцінювання на вкладеній архітектурі, де зовнішній шар не має відомих уразливостей, показує, що після внесення уразливостей моделі знаходять і використовують їх, тому контейнер не може бути єдиним бар'єром [12]. Рівень P5 передбачає одноразове середовище без мережі та без сталого сховища, у якому наслідки зникають разом із середовищем, а результат передається назовні лише через один вузький канал відповіді. Зведення рівнів, унеможливлених наслідків і принципових прогалин подано в табл. 1.

Таблиця 1. Шкала рівнів ізоляції дій агента за обсягом унеможливлених наслідків

Рівень	Що унеможливлює фізично	Чого не покриває принципово	Джерело
P0, без обмежень	Нічого не унеможливлює	Будь-яку дію у межах прав процесу	[4]
P1, перелік інструментів	Виклик незадекларованої функції	Зловживання дозволеним інструментом	[6; 11]
P2, аргументи і політики	Виклик з недозволеними аргументами	Дозволену дію за хибних підстав	[6; 9; 14]
P3, права в межах хоста	Запис і читання поза виділеним каталогом	Дії через дозволені мережеві канали	[5; 11]
P4, ізоляція середовища	Доступ до ресурсів основної системи	Вихід за межі пісочниці через уразливість	[12]
P5, одноразове середовище	Сталий слід і зовнішню передачу даних	Наслідки через канал відповіді користувачеві	[10; 13]

Джерело: розробка автора

Виокремлено три класи наслідків, які не покриває навіть найвищий рівень шкали. Перший клас утворюють наслідки через дозволений канал: агент повертає користувачеві хибну рекомендацію, а користувач виконує її власними

правами, тому дія відбувається поза ізольованим середовищем. Другий клас становить розкриття тих даних, які агентові навмисно передано на вхід, бо ізоляція обмежує вихідні канали, а не обізнаність агента, і саме тут потрібен контроль інформаційних потоків із позначками походження даних [8]. Третій клас виникає тоді, коли сама задача вимагає незворотної дії, наприклад надсилання листа або переказ платежу: за таких умов зростання рівня ізоляції суперечить призначенню системи, і замість заборони потрібні відкат та обмеження зони ураження, тобто перевірка результату після виконання дії, яку в цитованій праці обґрунтовано як простішу за перевірку до виконання [13]. Отруєні метадані інструментів показують, що навіть декларація дозволеного інструмента є недовіреним входом, тому реєстр інструментів потребує окремої перевірки [3].

Сформульовано умову достатності рівня ізоляції. Позначимо через U множину наслідків, які вибраний рівень не унеможливлює за заданої конфігурації, а через R множину наслідків, оборотних засобами самої системи, тобто таких, для яких існує зафіксована процедура повернення до попереднього стану у межах контрольованого часу. Рівень ізоляції достатній для задачі, якщо U є підмножиною R. Умова має три практичні наслідки. Якщо перевірка виявляє наслідок з U поза R, то допустимі лише два коректні кроки, а саме підняття рівня ізоляції доти, доки цей наслідок не буде унеможливлено, або розширення R побудовою процедури відкату для нього [13]. Умова робить неприйнятною поширену практику, за якої відсутність фіксованого відкату компенсують посиленням фільтра виходу, бо фільтр змінює ймовірність, а не належність наслідку до R. Умова також задає порядок проєктування: спершу перелічують незворотні наслідки задачі, потім вибирають найнижчий рівень, за якого вони унеможливлені або оборотні, і лише після цього додають перевірку виходу як додатковий, а не основний бар’єр [1; 2].

Обмеження. Робота не містить власного експерименту, тому наведена шкала є класифікаційною побудовою, а не вимірюванням. Усі числові оцінки взято з робіт інших авторів, причому більшість з них має статус препринта. Стенди й набори завдань у цих роботах різні, тому їхні показники не зведені до спільної основи і не порівнювані безпосередньо [7; 9; 12]. Твердження про детерміністичність обмежень середовища є умовним: воно справджується лише за припущення про коректну реалізацію самого обмежувача, і оцінювання виходів за межі пісочниці показує, що це припущення порушується [12]. Умова достатності сформульована як теоретико-множинне співвідношення, проте способу автоматично й повно побудувати множину наслідків U для довільної задачі у роботі не запропоновано, а її ручне визначення залежить від повноти моделі загроз. Статистичної перевірки жодного твердження не виконано, оцінок вартості обчислень для кожного рівня не отримано. Опрацьовано обмежену вибірку джерел, а саме чотирнадцять праць за період з 2023 до 2026 року, дібраних за темами ізоляції, привілеїв та інструкційних ін’єкцій, тому шкала може не враховувати засобів, поширених у промислових системах без наукових публікацій.

Висновки. Обґрунтовано, що надійність агентної системи визначає не якість перевірки виходу, а обсяг наслідків, які агент здатний спричинити: перевірка тексту є ймовірнісним бар'єром, а обмеження середовища є детерміністичним щодо класу дій [1; 8]. Побудовано шкалу з шести рівнів ізоляції, упорядкованих за множиною фізично унеможливлених наслідків, і для кожного рівня вказано наслідки, не охоплені ним принципово (табл. 1). Виокремлено три класи наслідків, недосяжних для ізоляції за побудовою, а саме дії користувача за хибною рекомендацією, розкриття навмисно переданих на вхід даних і незворотні дії, що є призначенням системи. Сформульовано умову достатності, за якою рівень ізоляції достатній тоді, коли множина неунеможливлених наслідків є підмножиною наслідків, оборотних засобами системи. Практичне значення полягає в тому, що умова перетворює вибір рівня ізоляції з оцінного судження на перевірний крок проєктування. Подальші дослідження доцільно спрямувати на вимірювання вартості кожного рівня та на автоматичне виведення множини наслідків з опису інструментів [6; 14].

Список літератури

1. Zhan Q., Liang Z., Ying Z. [та ін.] InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. Findings of the Association for Computational Linguistics ACL 2024. 2024. P. 10471–10506. DOI: 10.18653/v1/2024.findings-acl.624.

2. Debenedetti E., Zhang J., Balunović M. [та ін.] AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. Advances in Neural Information Processing Systems 37. 2024. P. 82895–82920. DOI: 10.52202/079017-2636.

3. Wang Z., Gao Y., Wang Y. [та ін.] MCPTox: A Benchmark for Tool Poisoning on Real-World MCP Servers. Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 40, No. 42. 2026. P. 35811–35819. DOI: 10.1609/aaai.v40i42.40895.

4. Wu Y., Roesner F., Kohno T. [та ін.] IsolateGPT: An Execution Isolation Architecture for LLM-Based Systems. Proceedings 2025 Network and Distributed System Security Symposium. 2025. DOI: 10.14722/ndss.2025.241131.

5. Wang C., Zheng Y. Sandlock: Confining AI Agent Code with Unprivileged Linux Primitives : препринт. arXiv:2605.26298. 2026. URL: https://arxiv.org/abs/2605.26298.

6. Shi T., He J., Wang Z. [та ін.] Progent: Securing AI Agents with Privilege Control : препринт. arXiv:2504.11703. 2025. URL: https://arxiv.org/abs/2504.11703.

7. Uchi U. Before the Tool Call: Deterministic Pre-Action Authorization for Autonomous AI Agents : препринт. arXiv:2603.20953. 2026. URL: https://arxiv.org/abs/2603.20953.

8. Costa M.F.M., Köpf B., Kolluri A. [та ін.] Securing AI Agents with Information-Flow Control : препринт. arXiv:2505.23643. 2025. URL: https://arxiv.org/abs/2505.23643.

9. Wang S., Zhu S., Li R.K. Runtime Policy Enforcement for MCP-Based LLM Agents. Electronics. Vol. 15, No. 13. 2026. P. 2829. DOI: 10.3390/electronics15132829.

10. Ruan Y., Dong H., Wang A. [та ін.] Identifying the Risks of LM Agents with an LM-Emulated Sandbox : препринт. arXiv:2309.15817. 2023. URL: https://arxiv.org/abs/2309.15817.

11. Yang K., Bu Y., Yi J. [та ін.] When Lower Privileges Suffice: Investigating Over-Privileged Tool Selection in LLM Agents : препринт. arXiv:2606.20023. 2026. URL: https://arxiv.org/abs/2606.20023.

12. Marchand R., Catháin A.Ó., Wynne J. [та ін.] Quantifying Frontier LLM Capabilities for Container Sandbox Escape : препринт. arXiv:2603.02277. 2026. URL: https://arxiv.org/abs/2603.02277.

13. Patil S.G., Zhang T., Fang V. [та ін.] GoEX: Perspectives and Designs Towards a Runtime for Autonomous LLM Applications : препринт. arXiv:2404.06921. 2024. URL: https://arxiv.org/abs/2404.06921.

14. Tsai L., Bagdasarian E. Contextual Agent Security: A Policy for Every Purpose. Proceedings of the Workshop on Hot Topics in Operating Systems. 2025. P. 8–17. DOI: 10.1145/3713082.3730378.