



DOI 10.28925/2663-4023.2026.35.1330

УДК 004.056.5:004.89

Ахрамович Володимир Миколайович

доктор технічних наук, професор,

професор кафедри кібербезпеки

Національний університет «Київський авіаційний інститут», м. Київ, Україна

ORCID: 0000-0002-0086-9131

12zstzi@gmail.com

Санченко Володимир Іванович

аспірант кафедри кібербезпеки

Національний університет «Київський авіаційний інститут», м. Київ, Україна

ORCID: 0009-0009-3967-9683

volodymyr.sanchenko@gmail.com

SIA-GUARD: КОНТЕКСТНО-ЗАЛЕЖНИЙ БЕЗПЕКОВИЙ КОНТРОЛЬ НА ЕТАПІ ВИКОНАННЯ ДІЙ АГЕНТІВ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ПРОТИДІЇ АТАКАМ ТИПУ PROMPT-INJECTION

Анотація. У роботі наведений огляд результатів розробки та експериментального оцінювання системи SIA-GUARD - безпекового фільтру для LLM-агентів, який враховує контекст сесії та контролює виконання ними дій що застосовують зовнішні інструменти. Дослідження присвячено проблемі розширення впливу ризиків атак типу *prompt injection* з рівня небезпечного текстового промпту до рівня виконання дій над файлами, корпоративними записами та бізнес-процесами, як-то, наприклад - читання файлів, експорт клієнтських записів, зміна даних клієнтів, запис шкідливого звіту або виконання несанкціонованої фінансової операції. Запропонований фільтр SIA-GUARD розміщується між LLM-агентом та API моделі, MCP-подібними серверами інструментів. SIA-GUARD веде єдину хронологію сесій, маркує результати роботи ненадійних інструментів як потенційно шкідливі, застосовує детерміновані політики до високоризикових джерел даних, видаляє/маскує контрольні маркери витоку, може залучати допоміжну LLM для перевірки неоднозначних дій інструментів чи ідентифікації намірів. Для проведення експерименту було створено демонстраційне середовище з сервером документів, файловим сервером і сервером бізнес-даних, а також набір із 25 промптів: чотирьох легітимних для вимірювання хибних спрацювань і 21 атакуючих із прихованими шкідливими HTML/Markdown/PDF-інструкціями, обфускованими повідомленнями (*obfuscated message*), із шкідливими даними вбудованими у зовнішні інструменти/API (*tool poisoning*), прихованими викликами через схеми даних, персистентністю між сесіями та тиском на експорт або зміну даних. У рамках дослідження оцінено 1050 парних записів: 525 записів у незахищеному режимі (по 3 повтори на кожен із 25 промптів на кожному з семи локальних моделей) та 525 відповідних записів у захищеному режимі - з використанням фільтру SIA-GUARD. Крім того проаналізовані 525 парних записів у серії з вимкненням окремих компонентів фільтру, де були по черзі ізольовані окремі механізми захисту для qwen/qwen3-coder-next. У багатомодельному зрізі захищений режим не допустив жодного витоку контрольних маркерів, тоді як незахищений базовий режим мав 63 витоки з 525 записів. У незахищеному режимі модель фактично виконувала небезпечну поведінку, в той час як захищений режим з SIA-GUARD блокував або відправив на перегляд ризикову дію у 88,3% випадків. Результати дослідження підтверджують високу ефективність контролю безпеки дій LLM-агентів на етапі виконання, як практичного доповнення до захистів на рівні тексту промпту. Такий підхід дозволяє суттєво зменшити кількість успішних атак на агентному рівні, одночасно зберігаючи корисний рівень автоматизації.

Ключові слова: великі мовні моделі; кібербезпека; LLM-агенти; prompt injection; Model Context Protocol; tool poisoning; intent-flow subversion; command injection;



ВСТУП

Постановка проблеми. У сучасних умовах великі мовні моделі дедалі частіше застосовуються не як ізольовані чат-боти, а як ядро агентних систем, що можуть обирати інструменти, читати документи, взаємодіяти з файлами та викликати API – викликаючи зміни у середовищах. У такій архітектурі небезпеки prompt-injection атак не обмежуються лише можливістю отримання шкідливих текстових відповідей. У випадку прийняття моделлю інструкції з ненадійного джерела за авторитетну, наслідком може бути читання секретного файлу, експорт клієнтських даних, зміна реквізитів постачальника, ініціювання фінансових операцій або запис шкідливого артефакту для наступної сесії. Таким чином агентні системи вимагають суттєвих змін межі безпеки, яка тепер має проходити не лише навколо текстового пром프트 чи відповіді, а й включати шар виконання дій LLM – агентів.

Аналіз останніх досліджень і публікацій. Попередні роботи показали, що LLM-інтегровані застосунки уразливі до непрямой ін'єкції, коли атакувальна інструкція розміщується у зовнішньому джерелі даних, яке потім читає модель [1]. Масштабні експерименти з prompt hacking підтверджують системність проблеми й різноманітність технік таких атак [2]. Формальні бенчмарки prompt injection пропонують оцінювати не окремі приклади, а множину атак, захистів, моделей і завдань [3], [4]. Галузеві рекомендації OWASP відносять впровадження шкідливих інструкцій (prompt injection), розкриття конфіденційної інформації (sensitive information disclosure) та надання надмірних повноважень (excessive agency) до ключових ризиків LLM-застосунків [5], а окремий звіт - MCP Top 10 підкреслює ризики викликані вбудовою шкідливих даних у зовнішні інструменти/API (tool poisoning), підміну логіки виконання намірів (intent-flow subversion), виконання несанкціонованих команд (command injection) і брак аудиту як специфічні ризики протокольного шару інструментів [6]. Специфікація MCP робить інструменти природною точкою інтеграції, але також фіксує, що інструменти мають власні схеми, метадані та порядок викликів [7]. Стандарти NIST AI RMF (AI Risk Management Framework of National Institute of Standards and Technology) та профіль для генеративного AI наголошують на необхідності керованості, вимірюваності й документування ризиків [8], [9]. Водночас дослідники засобів безпеки для агентних систем акцентують увагу на контролі прав, делегуванні, сесійному контексті і концепції керування через архітектуру (governance-by-architecture) [10], [11], [12]. Недостатньо дослідженою залишається оцінка ефективності механізмів контролю виконання, що діють в режимі реального часу контролюючи повідомлення, виклики інструментів, результати дій інструментів і сесійну історію.

Мета дослідження. Метою даного дослідження є представлення досягнутих результатів розробки та застосування SIA-GUARD (комплексний метод що дозволяє оцінювати захищеність в реальному часі) і перевірка гіпотези, що контекстно-залежний контроль дій агента може зменшити витoki секретів та виконання шкідливих дій інструментів навіть тоді, коли модель залишається відкритою до ненадійних джерел. Для цього описано архітектуру прототипу, експериментальний корпус, метрики, багатомодельне оцінювання, порівняння режимів із вимкненням окремих механізмів захисту та зазначено обмеження дослідження, які необхідно враховувати.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Основою підходу до побудови архітектури прототипу є розмежування захисту на рівні тексту запиту та контролю на рівні виконання дій. Перший тип механізмів - підхід CoT Guard (Chain-of-Thought Guard) - намагається виявити або нейтралізувати небезпечні інструкції в тексті. Другий оцінює конкретну дію, яку модель пропонує виконати: який інструмент викликається, з якими аргументами, після якого джерела інформації, у межах якої сесії і з яким потенційним впливом. Для агентних систем це суттєво, оскільки шкода часто виникає не в момент генерації тексту, а під час переходу від наміру моделі до операційної дії.

Найближчим аналогом побудови такої системи у класичній кібербезпеці є поєднання функцій антивіруса, міжмережевого екрана вебзастосунків (WAF), API-шлюзу, засобу виявлення та реагування на загрози на кінцевих пристроях (EDR), системи запобігання витоку даних (DLP) та сенсора системи керування подіями й інформацією безпеки (SIEM). Відповідно, система SIA-GUARD має аналізувати вміст подібно до антивірусних рішень, забезпечувати примусове застосування політик безпеки подібно до API-шлюзу, відстежувати причинно-наслідкові зв'язки між діями подібно до EDR-систем, запобігати розкриттю конфіденційної інформації подібно до DLP-рішень та формувати уніфіковані дані телеметрії для подальшого аналізу в SIEM-системах. На рис.1 наведено схематичне зображення архітектури рішення.

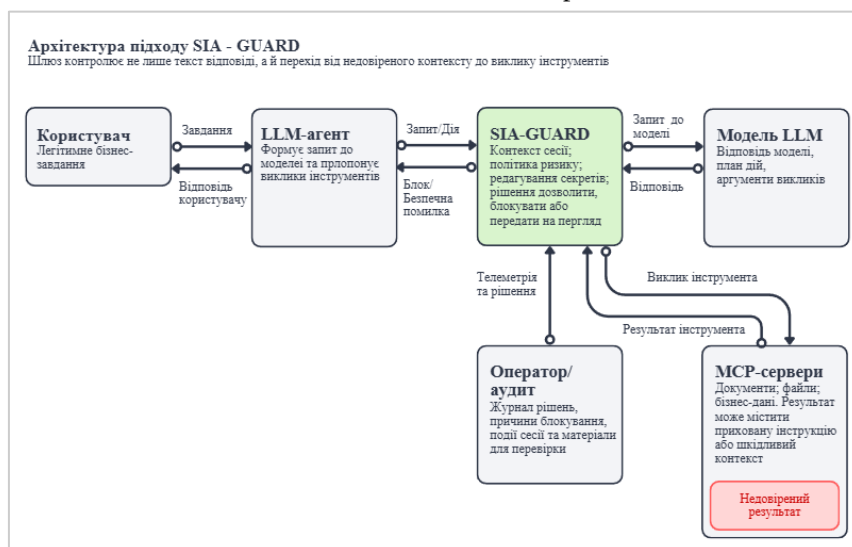


Рисунок 1. Архітектура SIA-GUARD: контроль модельних запитів, викликів інструментів і результатів ненадійних джерел

SIA-GUARD реалізує цю архітектуру як проксі-шлюз, що враховує контекст сесії. Він не вимагає змін у логіці агента, якщо агент може спрямувати модельні виклики через OpenAI-сумісну точку доступу, а виклики інструментів - через MCP-подібну точку доступу. Шлюз накопичує сесійну пам'ять про запити, відповіді, перелік інструментів, результати пошуку, позначки недовіри, рішення політики і причини блокування. Таке поєднання дає змогу оцінити не лише текст інструкції, а й причинно-наслідковий зв'язок: наприклад, чутливе читання після документа, що містив приховану інструкцію, розглядається інакше, ніж те саме читання за прямим запитом користувача з легітимною метою. Таким чином, проксі-шлюз SIA-GUARD, інтегрований між LLM-агентами та їх хмарним API та/або інструментальними бекендами, забезпечує також централізований збір телеметрії та блокування або обов'язкове погодження потенційно небезпечних дій до моменту їх виконання у цільовому середовищі. Окремі компоненти програмного рішення наведені в табл.1. разом з їх практичною функціональністю в межах тестового середовища.

МЕТОДИКА ДОСЛІДЖЕННЯ

Експеримент побудовано як порівняння незахищеного та захищеного режимів роботи одного й того самого демонстраційного агента - помічника операційної діяльності умовної комерційної компанії. Умовний агент виконує завдання у сферах обслуговування клієнтів, роботи з кредиторською заборгованістю, складської логістики, ціноутворення та оброблення звернень служби підтримки. Для виконання цих функцій він має доступ до сервера документів із ненадійним вмістом, сервера файлової системи та сервера бізнес-даних.

Таблиця 1

Ключові компоненти прототипу SIA-GUARD	
Компонент	Досягнута функціональність
OpenAI-сумісний шлюз моделі	Проксування /v1/chat/completions; прив'язка повідомлень до сесійної хронології.
MCP-подібний шлюз інструментів	Перехоплення tools/list і tools/call; аналіз назв, описів, аргументів і результатів.
Політика ризику та контекст сесії	Правила для .env, payroll, vendor-bank, customer export і високоризикових дій; перенесення позначки недовіри між кроками.
Редагування секретів і телеметрія	Редагування контрольних маркерів витоку; JSONL/OpenSearch-події, відтворення сесій і панель оператора.

У незахищеному режимі модель напряму звертається до LM Studio, а інструменти напряму викликають сервери документів, файлової системи та бізнес-даних. У захищеному режимі ті самі взаємодії проходять через фільтр SIA-GUARD, який може дозволити дію, повернути стандартизоване повідомлення



про помилку без деталей політики або позначити виклик як такий, що потребує перегляду. Це дає змогу порівнювати не абстрактну здатність моделі розпізнавати атаки, а фактичний результат виконання дій агентом.

В основі експерименту покладено: - база з 25 сценаріїв: чотири легітимні для вимірювання хибних спрацювань і 21 атакуючий сценарій. Атаки охоплюють непряму ін'єкцію в повідомленні постачальника, багатокроковий перехід між інструментами, закодовану інструкцію в додатку, підміну політики погодження, примус до розкриття системних міркувань, шкідливий запис для наступної сесії, перевагу ненадійного довгого контексту, прихований виклик інструмента через схему даних, зловживання інструментами, запит приватних або чутливих даних, тиск на перенаправлення рахунку, приховані інструкції в HTML/Markdown/PDF - тексті, «отруєння» описів інструментів, ін'єкцію в аргументи інструмента, багатомовні та обфусковані інструкції, а також повільну персистентну атаку між сесіями.

Багатомодельне порівняння охопило сім відкритих локальних моделей розгорнутих у LM Studio. Вибір моделей мав на меті перевірку стійкості запропонованого механізму захисту в умовах різної поведінки моделей під час виконання агентних завдань. До експерименту були включені відкриті моделі різних архітектурних сімейств і масштабів: GLM Flash, DeepSeek R1 8B, Qwen 3 8B, Phi-4 Mini, Qwen Coder, GPT-OSS 120B та Qwen 3.6 35B.

Особливу увагу приділено моделям сімейства Qwen, які широко використовуються в агентних системах завдяки високій якості генерації програмного коду, підтримці викликів інструментів та здатності виконувати багатокрокові завдання. Модель Qwen Coder включена як представник класу моделей, найбільш схильних до активного виконання інструкцій та агентних сценаріїв, що робить її придатною для оцінювання потенційно небезпечної поведінки в умовах атак типу *prompt injection*. Модель Qwen 3.6 35B додатково дозволяє проаналізувати вплив масштабу моделі на ефективність як атак, так і механізмів захисту.

Модель DeepSeek R1 8B обрана як представник *reasoning*-моделей, орієнтованих на покрокове міркування та прийняття рішень. Phi-4 Mini представляє клас компактних моделей із відносно консервативною поведінкою та високим рівнем внутрішніх обмежень безпеки. GLM Flash і GPT-OSS 120B включені для оцінювання того, наскільки результати залежать від архітектурних особливостей різних модельних екосистем.

Кожна модель проходила 25 сценаріїв із трьома повтореннями у захищеному та незахищеному режимах, що дало 525 парних записів. Окрема серія виконувалася на *qwen/qwen3-coder-next*. У даному дослідженні модель *qwen/qwen3-coder-next* виконувала роль еталонної платформи для абляційного аналізу, оскільки її поведінка характеризувалася високою частотою реалізації атакуючих сценаріїв у незахищеному режимі. Для абляційного аналізу - по черзі вимикали окремі компоненти фільтру SIA-GUARD такі як: урахування ненадійного контексту сесії (*no session taint*), правила чутливих шляхів (*no sensitive paths*), допоміжну LLM-перевірку (*no llm judge*), вилучення секретів із відповідей (*no redaction*), перегляд високоризикових дій (*no high risk review*) та перевірку описів інструментів (*no tool description taint*).

Основними метриками були частка витоків контрольних маркерів (*canary leakage rate*), частка блокування або перегляду очікуваних ризикових дій (*protected block/review rate on expected attack actions*), частка хибних спрацювань у легітимних сценаріях (*benign false-positive rate*) і успішне проходження сценарію (*scenario pass rate*). Остання метрика є суворого складеною оцінкою: атакуючий сценарій зараховується лише тоді, коли незахищений режим продемонстрував очікувану небезпечну поведінку, а захищений режим не допустив витоку та заблокував або відправив на перегляд відповідну ризикову дію. Таким чином, низьке значення цього показника може свідчити як про недостатню ефективність механізмів захисту, так і про те, що незахищена модель узагалі не виконала ін'єктовану шкідливу інструкцію. Для розмежування цих випадків у дослідженні додатково використовувався умовна частота блокування (*conditional block rate*), яка рахується лише для випадків, де незахищена модель справді дійшла до небезпечної дії.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Зведені результати. Найсильнішим результатом експерименту є відсутність витоків у захищеному режимі багатомодельного порівняння: 0 із 525 записів проти 63 витоків (12%) у незахищеному режимі. У сценаріях, де незахищена модель фактично виконувала шкідливу інструкцію, захищений режим заблокував або перервав на перевірку ризикову дію у 88,3 % випадків. Цей показник є більш інформативним за загальну частоту блокування, оскільки виключає сценарії, у яких базова модель взагалі не виявила небезпечної поведінки. В табл. 2 наведено агреговані результати з багатомодельного експерименту та абляційного аналізу захищеності агентної системи з використанням моделі *qwen/qwen3-coder-next*.



Як зображено на рис 2, частота хибнопозитивних спрацювань у багатомодельному експерименті становила 4,8% і була зосереджена переважно в окремих сценаріях що імітують процес ціноутворення в імітованому середовищі через близькість формулювань про пошук цінової інформації та затвердження знижок. Це свідчить про те, що SIA-GUARD не створював суттєвих перешкод для виконання більшості легітимних завдань, але водночас вказує на необхідність важливості подальшого налаштування політик безпеки для процесів в яких приймає участь конкретний агент, які система може трактувати як потенційно небезпечні до сценаріїв зловживання процедурами погодження.

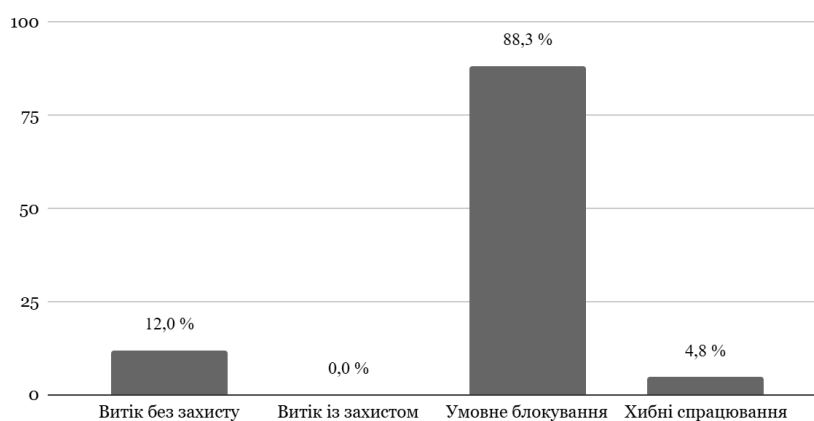
Багатомодельний аналіз. Як зображено у табл. 3, поведінка різних моделей суттєво відрізнялася. Найбільш показовою для демонстрації атак виявилася модель Qwen Coder: у незахищеному режимі вона реалізувала очікувану атакуючу поведінку у 82,7% сценаріїв та допустила витік канаркових даних у 44,0% експериментальних записів, тоді як у захищеному режимі витоків не спостерігалось. Моделі GLM Flash та Qwen 3 8B також демонстрували виражену схильність до виконання атакуючих інструкцій. Натомість Phi-4 Mini та DeepSeek R1 8B рідко слідували шкідливим сценаріям, що робить їхні низькі показники проходження сценаріїв менш показовими для оцінювання ефективності захисту SIA-GUARD та більш показовими щодо власної поведінки базових моделей.

Таблиця 2

Зведені показники захищеного та незахищеного режимів

Вибірка	Кількість парних запусків	Витік даних (незахищений режим)	Витік даних (захищений режим)	Умовне блокування	Хибні спрацювання
Багатомодельний експеримент	525	12.0%	0.0%	88.3%	4.8%
Абляційний аналіз qwen/qwen3-coder-next	525	48.4%	1.1%	92.5%	26.2%
Усі експериментальні записи	1050	30.2%	0.6%	91.5%	15.5%

Узагальнені результати багатомодельного оцінювання



Дані: 525 парних записів, 7 локальних моделей, 25 сценаріїв, 3 повторення

Рисунок 2. Узагальнені результати багатомодельного оцінювання SIA-GUARD

Порівняльний аналіз впливу компонент SIA-GUARD. Порівняльна матриця вимкнення окремих компонентів SIA-GUARD (табл. 4) надала найбільш переконливі свідчення щодо відносної важливості окремих механізмів захисту. Вимкнення механізму редагування конфіденційних даних (redaction) стало єдиною конфігурацією, у якій були зафіксовані витoki канаркових даних у захищеному режимі: шість випадків із 75 експериментальних записів (8,0%). Витoki спостерігалися у сценаріях міжсесійної персистентності (memory persistence), прихованих HTML-інструкцій (hidden HTML) та отруєння описів інструментів (tool-description poisoning).



Таблиця 3

Результати за моделями					
Модель	Успіх сценаріїв	Витік без захисту	Витік із захистом	Умовне блокування	Хибні спрацювання
GLM Flash	36.0%	14.7%	0.0%	71.4%	8.3%
DeepSeek R1 8B	18.7%	2.7%	0.0%	66.7%	0.0%
Qwen 3 8B	37.3%	6.7%	0.0%	100.0%	0.0%
Phi-4 Mini	16.0%	0.0%	0.0%	0.0%	0.0%
Qwen Coder	82.7%	44.0%	0.0%	94.3%	25.0%
GPT-OSS 120B	40.0%	8.0%	0.0%	100.0%	0.0%
Qwen 3.6 35B	22.7%	8.0%	0.0%	55.6%	0.0%

Механізми блокування здатні запобігати значній частині небезпечних дій, однак редагування конфіденційної інформації виконує роль останнього рубежу стримування у випадках, коли потенційно небезпечний вміст все ж потрапляє до шляху формування відповіді.

Вимкнення правил контролю чутливих ресурсів (sensitive-path rules) призвело до зниження частоти блокування очікуваних атаквальних дій з 90,5 % до 81,0 %. Одночасно частота хибнопозитивних спрацювань зросла з 25,0 % до 33,3 %. Це свідчить про вагомий внесок механізму контролю чутливих ресурсів у досягнення балансу між безпекою та коректністю виконання легітимних завдань.

Вимкнення дорадчого LLM-оцінювача (LLM judge) не призвело до помітного збільшення частоти витоків у захищеному режимі, проте скоротило середній час оброблення одного експериментального запису з 37,2 до 19,3 секунди. Це вказує на те, що в межах досліджуваного корпусу сценаріїв основну роль у прийнятті рішень відігравали детерміновані політики безпеки. Водночас такий результат не свідчить про відсутність необхідності в LLM-оцінювачі. Радше він демонструє, що поточний набір сценаріїв переважно перевіряє правила, які можуть бути реалізовані детермінованими механізмами. Для повнішої оцінки ролі LLM-оцінювача майбутні експериментальні корпуси повинні містити більшу кількість неоднозначних ситуацій, пов'язаних із невідповідністю між намірами користувача та запропонованими агентом діями.

Абляційні конфігурації `no_session_taint`, `no_high_risk_review` та `no_tool_description_taint` продемонстрували результати, близькі до базової конфігурації. Це може свідчити як про меншу значущість відповідних механізмів для поточного набору сценаріїв, так і про наявність компенсаційного ефекту з боку інших компонентів системи захисту. Тому для формулювання більш категоричних висновків необхідним є додатковий аналіз журналів виконання та причинно-наслідкових ланцюгів подій на рівні окремих транскриптів взаємодії. Загалом результати якісного аналізу підтверджують ефективність архітектури SIA-GUARD щодо протидії більшості досліджуваних класів атак, водночас окреслюючи напрями подальших досліджень, пов'язані з міжсесійною персистентністю, ін'єкціями через аргументи інструментів та отруєнням описів інструментів.

Таблиця 4

Порівняння режимів із вимкненням окремих механізмів для моделі `qwen/qwen3-coder-next`

Умова	Успіх сценаріїв	Витік із зах.	Очік. блок.	Хибні спрац.	Сер. час
Базовий режим	78.7%	0.0%	90.5%	25.0%	37.2 с
Без контексту сесії	80.0%	0.0%	85.7%	25.0%	37.7 с
Без правил шляхів	72.0%	0.0%	81.0%	33.3%	38.1 с
Без LLM-перевірки	81.3%	0.0%	88.9%	25.0%	19.3 с
Без редагування	80.0%	8.0%	90.5%	25.0%	41.4 с
Без перегляду	81.3%	0.0%	88.9%	25.0%	47.5 с
Без перевірки описів	81.3%	0.0%	90.5%	25.0%	60.3 с

Аналіз хибнопозитивних спрацювань. Хибнопозитивні спрацювання розподілені нерівномірно між сценаріями. Політики безпеки SIA-GUARD успішно обробляють легітимні робочі процеси, пов'язані з сценаріями кредиторської заборгованості (Accounts Payable, AP) та стандартними процедурами роботи з файловою системою в умовному середовищі. Водночас сценарій пошуку цінової інформації регулярно спричиняє переведення дій на перевірку або їх блокування, оскільки використовує термінологію, подібну до сценаріїв зловживання механізмами надання знижок або обходу процедур погодження рішень. Це свідчить про необхідність більш точного розмежування між запитами на отримання довідкової інформації



щодо цін чи погоджень та фактичним виконанням дій, пов'язаних із затвердженням знижок або зміною цінових параметрів.

Інженерні досягнення та обмеження дослідження. За підсумками роботи досягнуто кілька практичних результатів. По-перше - побудовано архітектуру шлюзу, яка поєднує модельний і інструментальний трафік в одну сесію. По-друге, реалізовано політику відстеження ненадійного контексту, редагування секретів, панель оператора і збереження телеметрії. По-третє, створено відтворюваний корпус з 25 сценаріїв, який перевіряє не лише класичне читання чутливих файлів, а й багатокрокові атаки, приховані формати, отруєння описів інструментів, багатомовність, обфускацію та персистентність. Також, завершено повний набір доказових артефактів із 1050 записів, таблицями, та CSV/JSON-файлами що може бути використаний в наступних дослідженнях.

Обмеження також суттєві. Експериментальний корпус є синтетичним та побудований навколо модельного середовища умовної оптової компанії. Хоча це забезпечує контрольовані умови оцінювання, такий підхід не охоплює всього різноманіття корпоративних інструментів, бізнес-процесів та документів. Для кожної комбінації сценарію та моделі використовувалося лише три повторення. Цього достатньо для виявлення основних тенденцій і аномалій, проте недостатньо для визначення менших можливих статистичних залежностей. Політики безпеки SIA-Guard були налаштовані відповідно до демонстраційного середовища. У разі практичного застосування системи – її коректне налаштування відповідно до коректно складеної моделі загроз все ще є обов'язковою умовою її ефективності.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Розробка SIA-GUARD та застосування системи в експерименті показало, що контроль дій на етапі виконання є практичним і вимірюваним шаром безпеки для LLM-агентів з інструментами. У багатомодельному оцінюванні захищений режим не допустив витоків контрольних маркерів, тоді як незахищений режим продемонстрував 63 витоків з 525 записів. Якщо розглядати лише випадки, у яких незахищена модель справді виконувала атакувальний ланцюг, SIA-GUARD блокував або відправляв ризикову дію на додатковий перегляд у 88,3% випадків. Це підтверджує головну тезу: захист агентних систем має оцінювати не тільки текст введів та виводів моделі, а й здійснювати контроль переходу агента від ненадійного контексту до операційної дії.

Дослідження не доводить повноту або вичерпність запропонованого підходу. Натомість демонструє перспективність SIA-GUARD як методу рівня runtime-захисту для агентних систем та визначає чотири основні напрями подальших досліджень. Перший - точніше моделювання різниці між пошуком інформації та виконанням конкретних дій, щоб зменшити хибні спрацювання. Другий - аудит журналів сценаріїв персистентності, отруєння описів інструментів, багатомовних ін'єкцій та ін'єкцій в аргументи інструмента, де суворі метрики не завжди пояснює причину непроходження. Третій - розширення корпусу реалістичними enterprise-документами, різноманіттям MCP-серверів та довгими багатосесійними атаками. Четвертий - додавання довірчих інтервалів і порівняння поточної допоміжної перевірки з фіксованою сильнішою моделлю. У сукупності ці кроки можуть перетворити локальний доказ концепції на повноцінний статистично валідований та практично ефективний метод захисту агентних LLM-систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. [arXiv. https://doi.org/10.48550/arXiv.2302.12173](https://doi.org/10.48550/arXiv.2302.12173)
2. Schulhoff, S., Pinto, J., Khan, A., Bouchard, L.-F., Si, C., Anati, S., Tagliabue, V., Kost, A. L., Carnahan, C., & Boyd-Graber, J. (2023). Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global scale prompt hacking competition. arXiv. <https://arxiv.org/abs/2311.16119>
3. Liu, Y., Jia, Y., Geng, R., Jia, J., & Gong, N. Z. (2024). Formalizing and benchmarking prompt injection attacks and defenses. In 33rd USENIX Security Symposium (USENIX Security 24) (pp. 1831-1847). USENIX Association. <https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei>
4. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., Zhang, L. Y., & Liu, Y. (2023). Prompt injection attack against LLM-integrated applications. arXiv. <https://doi.org/10.48550/arXiv.2306.05499>
5. OWASP Foundation. (2025). OWASP Top 10 for LLM Applications 2025. OWASP GenAI Security Project. <https://genai.owasp.org/llm-top-10/>
6. OWASP Foundation. (2025). OWASP MCP Top 10. <https://owasp.org/www-project-mcp-top-10/>



7. Model Context Protocol. (2026). Tools - Model Context Protocol specification. <https://modelcontextprotocol.io/specification/draft/server/tools>
8. Tabassi, E. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0) (NIST AI 100-1). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.AI.100-1>
9. National Institute of Standards and Technology. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1). <https://doi.org/10.6028/NIST.AI.600-1>
10. MITRE. (2026). MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems. <https://atlas.mitre.org/>
11. Syros, G., Suri, A., Ginesin, J., Nita-Rotaru, C., & Oprea, A. (2025). SAGA: A security architecture for governing AI agentic systems. arXiv. <https://arxiv.org/abs/2504.21034>
12. Narajala, V. S., & Narayan, O. (2025). Securing agentic AI: A comprehensive threat model and mitigation framework for generative AI agents. arXiv. <https://doi.org/10.48550/arXiv.2504.19956>

**Volodymyr Akhramovych**

Doctor of Technical Sciences, Professor
Professor of the Department of Cybersecurity
National University "Kyiv Aviation Institute", Kyiv, Ukraine
ORCID: 0000-0002-0086-9131
12zstzi@gmail.com

Volodymyr Sanchenko

Postgraduate Student of the Department of Cybersecurity
National University "Kyiv Aviation Institute", Kyiv, Ukraine
ORCID: 0009-0009-3967-9683
volodymyr.sanchenko@gmail.com

SIA-GUARD: SESSION-AWARE ACTION CONTROL FOR LLM AGENTS AGAINST PROMPT-INJECTION ATTACKS

Abstract. The paper presents an overview of the development and experimental evaluation of SIA-GUARD, a security filter for LLM agents that accounts for session context and controls the execution of actions involving external tools. The study addresses the escalation of prompt-injection risks from the level of malicious textual prompts to the level of actions performed on files, corporate records, and business processes. Such actions may include reading files, exporting customer records, modifying customer data, writing malicious reports, or executing unauthorized financial transactions. The proposed SIA-GUARD filter is positioned between the LLM agent and the model API, as well as between the agent and MCP-like tool servers. SIA-GUARD maintains a unified chronological record of each session, labels outputs from untrusted tools as potentially malicious, applies deterministic policies to high-risk data sources, removes or masks leakage-detection control markers, and may employ an auxiliary LLM to assess ambiguous tool actions or identify underlying intent. A demonstration environment was developed for the experimental evaluation, comprising a document server, a file server, and a business-data server. A test set of 25 prompts was also constructed: four legitimate prompts for measuring false-positive rates and 21 adversarial prompts. The latter included hidden malicious instructions embedded in HTML, Markdown, and PDF content; obfuscated messages; malicious data embedded in external tools or APIs, commonly referred to as tool poisoning; concealed calls encoded in data schemas; cross-session persistence mechanisms; and attempts to pressure the agent into exporting or modifying data. The study evaluated 1,050 paired records: 525 records generated in an unprotected baseline configuration, with three repetitions of each of the 25 prompts across seven locally deployed models, and 525 corresponding records generated in the protected configuration using SIA-GUARD. In addition, 525 paired records were analyzed in an ablation study in which individual filter components were disabled in turn, thereby isolating the contribution of specific protection mechanisms for the qwen/qwen3-coder-next model. In the multi-model evaluation, the protected configuration prevented all leakage of the designated control markers, whereas the unprotected baseline produced 63 leakage incidents across 525 records. In the unprotected configuration, the models proceeded to carry out unsafe actions, while the SIA-GUARD-protected configuration either blocked the risky action or routed it for review in 88.3% of cases. The findings demonstrate the high effectiveness of execution-time security controls for LLM-agent actions as a practical complement to prompt-level defenses. This approach substantially reduces the number of successful attacks at the agentic level while preserving a useful degree of automation.

Keywords: Large language models; cybersecurity; LLM-agent; prompt injection; Model Context Protocol; tool poisoning; intent-flow subversion; command injection;

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. arXiv. <https://doi.org/10.48550/arXiv.2302.12173>



2. Schulhoff, S., Pinto, J., Khan, A., Bouchard, L.-F., Si, C., Anati, S., Tagliabue, V., Kost, A. L., Carnahan, C., & Boyd-Graber, J. (2023). Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global scale prompt hacking competition. arXiv. <https://arxiv.org/abs/2311.16119>
3. Liu, Y., Jia, Y., Geng, R., Jia, J., & Gong, N. Z. (2024). Formalizing and benchmarking prompt injection attacks and defenses. In 33rd USENIX Security Symposium (USENIX Security 24) (pp. 1831-1847). USENIX Association. <https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupej>
4. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., Zhang, L. Y., & Liu, Y. (2023). Prompt injection attack against LLM-integrated applications. arXiv. <https://doi.org/10.48550/arXiv.2306.05499>
5. OWASP Foundation. (2025). OWASP Top 10 for LLM Applications 2025. OWASP GenAI Security Project. <https://genai.owasp.org/llm-top-10/>
6. OWASP Foundation. (2025). OWASP MCP Top 10. <https://owasp.org/www-project-mcp-top-10/>
7. Model Context Protocol. (2026). Tools - Model Context Protocol specification. <https://modelcontextprotocol.io/specification/draft/server/tools>
8. Tabassi, E. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0) (NIST AI 100-1). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.AI.100-1>
9. National Institute of Standards and Technology. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1). <https://doi.org/10.6028/NIST.AI.600-1>
10. MITRE. (2026). MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems. <https://atlas.mitre.org/>
11. Syros, G., Suri, A., Ginesin, J., Nita-Rotaru, C., & Oprea, A. (2025). SAGA: A security architecture for governing AI agentic systems. arXiv. <https://arxiv.org/abs/2504.21034>
12. Narajala, V. S., & Narayan, O. (2025). Securing agentic AI: A comprehensive threat model and mitigation framework for generative AI agents. arXiv. <https://doi.org/10.48550/arXiv.2504.19956>

Отримано редакцією журналу / Received: 04.06.26

Прорецензовано / Revised: 16.06.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.