

РОЗРОБКА ШІ-АГЕНТА ДЛЯ АВТОМАТИЗАЦІЇ ТРІАЖУ КІБЕРІНЦИДЕНТІВ БЕЗПЕКИ

О. К. Новіцький¹, М. В. Коломицев¹

¹ Навчально-науковий Фізико-технічний інститут

Анотація

У роботі досліджується проблема автоматизації тріажу кіберінцидентів в операційних центрах безпеки (SOC) із застосуванням AI-агента на основі великих мовних моделей (LLM). Проаналізовано обмеження класичних rule-based SOAR-систем та явище alert fatigue. Запропоновано архітектуру AI-агента, що інтегрується з існуючою SOAR-платформою та здатний виконувати контекстуальну класифікацію, пріоритизацію та збагачення алертів з урахуванням історичних тікетів і організаційного контексту. Проведено порівняльний аналіз сучасних LLM-моделей та визначено систему метрик для оцінки якості роботи агента.

Ключові слова: SOC, тріаж, кіберінциденти, LLM, AI-агент, SOAR, alert fatigue, автоматизація.

Вступ

Операційні центри безпеки (SOC) є ключовими структурними одиницями, що забезпечують моніторинг, виявлення та реагування на кіберінциденти в організаціях [1]. Аналітики SOC щоденно стикаються з великим потоком алертів від SIEM-систем (Security Information and Event Management), EDR-рішень (Endpoint Detection and Response) та інших інструментів моніторингу. При цьому значна частка цих алертів є хибно позитивними спрацюваннями, що призводить до явища alert fatigue — когнітивного виснаження аналітиків внаслідок хронічного перевантаження [2].

Тріаж — первинна обробка, класифікація та пріоритизація вхідних алертів — є найбільш рутинним і водночас критично важливим етапом реагування на інциденти [3]. Класичним інструментом автоматизації тріажу є SOAR-платформи (Security Orchestration, Automation and Response), що базуються на rule-based підході [4]. Проте їх ефективність обмежена: вони здатні реагувати лише на заздалегідь описані сценарії, а підтримка актуальності правил вимагає постійних інженерних зусиль.

У більшості SOC аналітики розподілені за рівнями [1]: Tier 1 займається цілодобовим моніторингом алертів у реальному часі, витрачаючи від однієї до п'ятнадцяти хвилин на кожну подію; Tier 2 приймає ескальовані інциденти та проводить детальне розслідування; Tier 3 займається найскладнішими випадками та проактивним пошуком загроз. Саме аналітики Tier 1 є найбільш вразливими до alert fatigue, оскільки обробляють найбільший обсяг алертів при найменшому часі на кожен із них.

Метою роботи є проектування AI-агента на основі великих мовних моделей (LLM), здатного викону-

вати контекстуальний тріаж алертів кібербезпеки, адаптуватись до нових типів загроз та формувати обґрунтовані рекомендації для аналітиків SOC.

1. Проблема alert fatigue та обмеження існуючих підходів

Продуктивність аналітиків SOC залежить від обсягу, швидкості надходження, достовірності та різноманітності алертів. SIEM-системи, налаштовані на мінімізацію хибно негативних спрацювань, генерують надмірну кількість хибно позитивних результатів. Дослідження [2] виділяють чотири основні причини alert fatigue: нестача кваліфікованого персоналу, висока частка хибних спрацювань, фрагментованість інформаційних панелей та неефективні операційні процедури.

Alert fatigue безпосередньо впливає на два ключові показники ефективності SOC: MTTD (Mean Time to Detect) — середній час від виникнення загрози до її виявлення, та MTTR (Mean Time to Respond) — середній час від виявлення до повного усунення інциденту. Зростання беклогу необроблених алертів збільшує обидва показники, а виснажені аналітики схильні приймати менш виважені рішення.

SOAR-платформи частково вирішують цю проблему через автоматизацію рутинних кроків тріажу за допомогою плейбуків [4]. SOAR-рішення складаються з трьох ключових компонентів: оркестрація безпеки — координація апаратних і програмних інструментів через API та плагіни; автоматизація — виконання рутинних задач, зокрема відкриття тікетів, збагачення подій та пріоритизація алертів; реагування на інциденти — централізоване управління процесом розслідування.

Однак принципове обмеження rule-based підходу

полягає в тому, що будь-який новий тип загрози або нетипова комбінація подій залишається поза межами автоматизації, доки інженер вручну не створить відповідний сценарій. Навіть добре налаштована система поступово втрачає актуальність: зміни в інфраструктурі, поява нових інструментів або еволюція тактик зловмисників вимагають постійного перегляду правил. Окрім того, rule-based системи позбавлені здатності до контекстуального розуміння — вони не можуть оцінити ситуацію в ширшому контексті або зробити логічний висновок на основі неповних даних. Це формує запит на якісно інший підхід до автоматизації тріажу.

Серед існуючих способів боротьби з alert fatigue також виділяють автоматичну пріоритизацію алертів на основі машинного навчання (ML), зменшення хибних спрацювань через аналіз графів причинно-наслідкових залежностей, а також інструменти візуалізації та пояснюваного ШІ (зокрема, методи SHAP — SHapley Additive exPlanations та LIME — Local Interpretable Model-agnostic Explanations), що полегшують прийняття рішень аналітиками [2]. Проте більшість цих підходів потребують значних інженерних зусиль для підтримання ефективності та стикаються з фундаментальним протиріччям: агресивне фільтрування алертів неминуче супроводжується зростанням ризику пропуску реальних загроз.

2. Концепція AI-агента для тріажу

Подолання обмежень rule-based підходу стає можливим завдяки застосуванню AI-агента на основі LLM, здатного в реальному часі оцінювати кожен новий алерт. На відміну від класичного підходу, де логіка прийняття рішень жорстко закодована у вигляді правил, AI-агент здатен до узагальнення — тобто до прийняття обґрунтованих рішень стосовно ситуацій, що не були явно передбачені під час його налаштування.

Ключовою характеристикою агента є здатність до роботи з контекстом. Один і той самий алерт може мати зовсім різне значення залежно від того, на якому хості він виник, яка політика безпеки застосовується та чи спостерігалась схожа активність раніше. Архітектура агента передбачає три вхідні потоки даних: безпосередньо сам алерт із технічними атрибутами; організаційний контекст — інформація про інфраструктуру, критичні хости та політики безпеки; історичний контекст — інформація про попередні алерти та інциденти, що дозволяє виявляти патерни багаторічних атак.

AI-агент розглядається не як повна заміна аналітика, а як інструмент, що бере на себе первинну обробку великого потоку алертів, зменшуючи когнітивне навантаження та скорочуючи час на кожну подію. Результатом роботи агента є не просто мітка з категорією і пріоритетом, а розгорнуте пояснення того, чому було прийняте саме таке рішення, разом із конкретними рекомендаціями для аналітика — що саме варто перевірити далі та які заходи

реагування застосувати. Така пояснюваність є принципово важливою: якщо агент видає лише результат без обґрунтування, аналітик змушений витратити час на самостійну перевірку висновків, що частково знецінює автоматизацію.

3. Архітектура запропонованого рішення

Архітектура системи побудована навколо двох основних компонентів: SOAR-платформи та AI-агента (рис. 1). SOAR-платформа виступає центральним сховищем та точкою інтеграції — вона приймає алерти з різноманітних джерел (SIEM, EDR, IDS — системи виявлення вторгнень, тощо), нормалізує їх та зберігає як тікети. AI-агент взаємодіє з SOAR через API, виконуючи три основні операції: отримання нових алертів, пошук пов'язаних історичних тікетів та запис результатів аналізу.

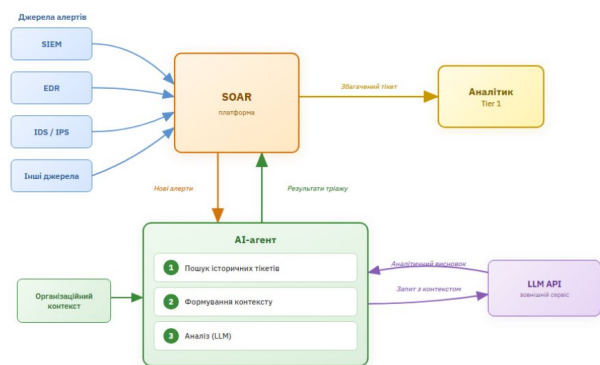


Рис. 1. Загальна архітектура системи AI-агента для тріажу.

Потік обробки алерту складається з кількох послідовних кроків. Спочатку агент отримує новий тікет із SOAR та витягує з нього ключові атрибути: ім'я хосту, обліковий запис, IP-адреси та назву правила детектування. Далі виконується пошук пов'язаних історичних тікетів за цими атрибутами за принципом часткового збігу — достатньо, щоб хоча б один атрибут співпадав. З-поміж знайдених тікетів відбираються до десяти найбільш релевантних.

На етапі формування контексту агент об'єднує чотири логічні блоки у єдиний запит до LLM: системну інструкцію (роль, формат відповіді, обмеження), організаційний контекст (критичність хостів, ролі акаунтів, політики), історичні тікети та дані поточного алерту. Такий порядок — від загального контексту до конкретної задачі — обрано для того, щоб модель мала уявлення про середовище на момент аналізу алерту.

Після генерації відповіді агент валідує її структуру та записує результат до SOAR як оновлення тікету, що включає класифікацію, пріоритет, текстове пояснення та рекомендації. Оброблений тікет стає частиною історичної бази, замикаючи цикл збагачення контексту.

4. Порівняльний аналіз та вибір LLM

Вибір мовної моделі здійснювався за шістьма критеріями: якість аналізу, розмір контекстного вікна, якість роботи з довгим контекстом, швидкість генерації, вартість використання та політика конфіденційності. Дані для порівняння отримано з незалежної платформи Artificial Analysis [5]. Результати порівняння наведено у табл. 1.

Таблиця 1. Порівняння характеристик LLM-моделей.

Модель	Intel. Index	Контекст, тис. т.	\$/1M вх.	Шв., т/с
GPT-5.4 (xhigh)	57	1100	2,50	81
GPT-5.4 mini (xhigh)	49	400	0,75	173
Claude Sonnet 4.6	44	1000	3,00	44
Claude 4.5 Sonnet	37	1000	3,00	42
Gemini 2.5 Pro	35	1000	1,25	128
Gemini 2.5 Flash	21	1000	0,30	185

За результатами аналізу обрано модель GPT-5.4 з рівнем reasoning effort xhigh. Обґрунтування базується на пріоритетності якості аналізу для задачі тріажу, де ціна помилки є асиметричною: пропущений інцидент має значно серйозніші наслідки, ніж зайва перевірка хибного спрацювання. GPT-5.4 (xhigh) з оцінкою 57 балів за Intelligence Index є найсильнішою серед розглянутих моделей. Різниця у 8 балів порівняно з GPT-5.4 mini (49 балів) та у 13 балів порівняно з Claude Sonnet 4.6 (44 бали) є суттєвою і може проявлятися саме у складних випадках — нетипових алертах, багатоетапних атаках та ситуаціях з неповними даними.

Вартість обробки одного алерту складає приблизно 4 центи (при 10 тис. входних та 1 тис. вихідних tokenів), що при обробці 200 алертів на добу дає приблизно 240 доларів на місяць — значно менше за витрати на додаткового аналітика Tier 1. Параметр reasoning effort xhigh передбачає найбільш глибокий аналіз, що є обґрунтованим для задачі без жорстких вимог до латентності в реальному часі. Усі три розглянуті провайдери (OpenAI, Anthropic, Google) у документації для API-продуктів зазначають, що дані клієнтів не використовуються для навчання моделей за замовчуванням [6, 7, 8].

5. Метрики оцінки якості

Для об'єктивної оцінки роботи агента визначено систему метрик, що охоплює три виміри.

Якість аналізу оцінюється через точність класифікації — частку правильних рішень true positive / false positive порівняно з еталонним рішенням аналітика. Окремо вимірюється здатність використовувати історичний контекст через спеціальні тестові сценарії, де правильний висновок неможливий без урахування історичних тікетів. Наприклад, одиничний алерт про невдалу автентифікацію є рутинною подією, але серія аналогічних спроб з різних IP-адрес

вказує на атаку перебором. Додатково проводиться експертна оцінка якості пояснень — їх конкретності, обґрунтованості фактами та практичної корисності рекомендацій.

Швидкодія вимірюється як загальний час обробки одного алерту з розбивкою по етапах: пошук історичних тікетів у SOAR, формування контексту, генерація відповіді LLM та запис результату. Така розбивка дозволяє оперативно діагностувати джерело затримок при деградації продуктивності системи.

Консистентність перевіряється через повторну обробку однакових алертів з однаковим контекстом. Відсоток випадків, коли класифікація та пріоритет залишаються незмінними, має бути максимально близьким до 100 %. Стабільність логіки пояснень оцінюється через ручне порівняння відповідей. Низька консистентність може вказувати на проблеми з промптом — недостатню конкретність інструкцій, що залишають моделі занадто багато простору для інтерпретації.

6. Обмеження та ризики LLM-підходу

Застосування великих мовних моделей у задачі тріажу пов'язане з низкою специфічних ризиків, що потребують врахування при проектуванні системи.

Першим ризиком є галюцинації — генерація правдоподібних, але фактично недостовірних тверджень. На відміну від детермінованих rule-based систем, LLM генерує відповідь на основі статистичних патернів, що може призводити до хибної класифікації нешкідливої події як загрози або, що значно небезпечніше, до помилкової оцінки справжнього інциденту як незначного.

Другим ризиком є prompt injection — атака, при якій зловмисник вбудовує шкідливі інструкції у дані, що обробляються моделлю [9]. Для агента тріажу ця загроза є особливо актуальною, оскільки він за своєю природою обробляє потенційно шкідливий вміст — алерти, логи та артефакти, що можуть бути навмисно сформовані зловмисником. Наприклад, зловмисник може вбудувати у назву процесу або параметри командного рядка текст, що імітує системну інструкцію і намагається змусити модель класифікувати інцидент як нешкідливий. Мінімізація цього ризику досягається на архітектурному рівні: чітким розмежуванням системних інструкцій та входних даних у промпті, обмеженням набору допустимих вихідних форматів та валідацією структури відповіді.

Третім ризиком є передача чутливих даних на сервери стороннього LLM-провайдера. Алерти можуть містити внутрішні IP-адреси, імена хостів, облікові дані та деталі вразливостей. При виборі провайдера перевагу слід надавати корпоративним API з гарантіями незберігання та невикористання даних клієнтів для навчання моделей. Важливо також враховувати, що політики зберігання даних різняться залежно від провайдера та типу акаунту: споживчі продукти, як правило, передбачають можливість використання даних для навчання, тоді як корпора-

тивні API зазвичай пропонують суворіші гарантії конфіденційності.

Окрім перелічених ризиків, слід зазначити питання відтворюваності результатів. Мовні моделі за своєю природою не є детерміністськими — навіть при однакових вхідних даних вони можуть генерувати різні відповіді. Для агента тріажу це означає, що один і той самий алерт при повторній обробці може отримати іншу класифікацію, що підриває довіру аналітиків до системи. Мінімізація цієї проблеми досягається через використання низьких значень температури генерації, чітких системних інструкцій з обмеженим набором допустимих вихідних значень та валідації структури кожної відповіді.

Висновки

У роботі запропоновано архітектуру AI-агента на основі великих мовних моделей для автоматизації тріажу кіберінцидентів у SOC. Показано, що класичні rule-based SOAR-системи мають принципові обмеження — нездатність адаптуватись до нових загроз без ручного оновлення правил та відсутність контекстуального розуміння. Запропонований підхід дозволяє інтегрувати AI-агента поверх існуючої SOAR-інфраструктури без її заміни, що суттєво знижує поріг впровадження та робить рішення доступним навіть для невеликих організацій.

Архітектура агента забезпечує тривимірну контекстуалізацію алертів — з урахуванням поточних даних, історичних тікетів та організаційного контексту — що наближає якість аналізу до міркувань досвідченого аналітика. Замкнений цикл збагачення, при якому кожен оброблений тікет поповнює історичну базу, дозволяє системі накопичувати контекст та підвищувати якість аналізу з часом без перенавчання моделі. Проведений порівняльний аналіз шести LLM-моделей за шістьма критеріями обґрунтував вибір GPT-5.4 (xhigh) як оптимального рішення за критерієм співвідношення якості аналізу та вартості. Визначено систему метрик для об'єктивної оцінки якості, швидкодії та консистентності роботи агента.

Подальша робота спрямована на реалізацію прототипу системи, проведення експериментальної оцінки за визначеними метриками та дослідження стійкості агента до adversarial-впливів, зокрема prompt injection атак. Окремим перспективним напрямком є дослідження можливості використання менших спеціалізованих моделей, що можуть бути розгорнуті ло-

кально в інфраструктурі організації, повністю усуваючи ризик передачі чутливих даних зовнішнім провайдерам. Крім того, важливим залишається питання інтеграції зворотного зв'язку від аналітиків — механізму, який дозволив би системі враховувати корекції людини при обробці наступних подібних алертів, тим самим поступово підвищуючи точність класифікації без необхідності перенавчання моделі.

Перелік використаних джерел

- 10 Strategies of a World-Class Cybersecurity Operations Center / J. Madigan [et al.] ; The MITRE Corporation. — 2014. — URL: <https://giaiphamangh3t.com/wp-content/uploads/2020/07/10-strategies-cyber-ops-center.pdf>.
- Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities / S. Tariq [et al.] // ACM Computing Surveys. — 2025. — Vol. 57, no. 9. — P. 1–38. — DOI: [10.1145/3723158](https://doi.org/10.1145/3723158). — URL: <https://dl.acm.org/doi/10.1145/3723158>.
- National Institute of Standards and Technology. Incident Response Recommendations and Considerations for Cybersecurity Risk Management / NIST. — 2024. — SP 800–61r3. — URL: <https://csrc.nist.gov/pubs/sp/800/61/r3/final>.
- IBM. What is SOAR (security orchestration, automation and response)? — 2024. — URL: <https://www.ibm.com/think/topics/security-orchestration-automation-response>.
- Artificial Analysis. LLM Leaderboard and Comparison. — 2025. — URL: <https://artificialanalysis.ai/>.
- OpenAI. How OpenAI uses your data. — 2025. — URL: <https://platform.openai.com/docs/guides/your-data>.
- Anthropic. Is my data used for model training? — 2025. — URL: <https://privacy.claude.com/en/articles/7996868-is-my-data-used-for-model-training>.
- Google Cloud. Gemini for Google Cloud data governance. — 2025. — URL: <https://docs.cloud.google.com/gemini/docs/discover/data-governance>.
- OWASP. Prompt Injection. — 2024. — URL: <https://owasp.org/www-community/attacks/PromptInjection>.