DOI: https://doi.org/10.26565/2304-6201-2026-69-02

УДК (UDC) 004.056.5:004.4

Братко Дмитро Вячеславович *магістрант, курсант*
Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", вул. Верхньоключова 4, м. Київ, Україна, 03056
e-mail: loading.2285@gmail.com
https://orcid.org/0009-0001-0863-9285

Кубрак Володимир Олександрович *PhD, Старший викладач Спеціальної кафедри № 1*
Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", вул. Верхньоключова 4, м. Київ, Україна, 03056
e-mail: v.kubrak@kpi.ua;
https://orcid.org/0000-0001-8877-5289

Матійко Александра Андріївна *PhD, Доцент Спеціальної кафедри №1*
Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", вул. Верхньоключова 4, м. Київ, Україна, 03056
e-mail: alexm1710@ukr.net;
https://orcid.org/0000-0002-6947-5958

Автономна оркестрована система реагування на інциденти на основі SIEM

Актуальність. Сучасні інформаційні системи генерують події безпеки з різних джерел, таких як журналів операційних систем і сервісів, мережевих сенсорів, сканерів вразливостей та інших засобів моніторингу. У таких умовах SIEM дає змогу централізовано збирати, індексувати та корелювати телеметрію, однак сам перехід від результату аналітики до практичного реагування часто залишається недостатньо формалізованим. Це призводить до затримок, залежності від ручних дій, складнощів із повторюваністю процедур і відсутності єдиного механізму підтвердження виконаних реакцій. Додатковою проблемою є безпечний автономний доступ до кінцевих вузлів під час інциденту, коли неприпустимими є як ручні підтвердження SSH-з'єднання, так і небезпечна довіра до першого ключа. У зв'язку з цим актуальною є побудова архітектурного моста між SIEM-аналітикою та системою оркестрації, здатного забезпечити кероване, відтворюване й аудитоване реагування на інциденти незалежно від первинного джерела подій.
Метою роботи є обґрунтування та експериментальна перевірка архітектурного підходу до автономного оркестрованого реагування на інциденти, у межах якого результати аналітики SIEM перетворюються на структурований інцидентний запис і далі використовуються для запуску процедур реагування в системі оркестрації. Для досягнення цієї мети передбачено опис детектора у декларативному вигляді, стандартизацію інцидентного запису, керування повторними спрацюваннями через унікальний ключ інциденту та інтервал блокування повторного запуску, узгодження цільових активів із даними inventory, журналювання результатів виконання, а також реалізацію безпечного доступу до кінцевих вузлів на основі SSH Host CA. Демонстраційним прикладом обрано сценарій виявлення та реагування на SSH brute-force.
Результати. У результаті дослідження сформовано й експериментально перевірено архітектурний підхід, який поєднує SIEM-аналітику з автоматизованим виконанням реакційних дій на цільових активах. Показано, що результат аналітичного запиту в SIEM може бути послідовно перетворений на інцидентний запис, використаний для побудови унікального ключа інциденту, перевірки політик повторного запуску, узгодження активу з inventory та передачі параметрів до playbook. Реалізований прототип підтвердив технічну можливість побудови повного циклу від виявлення події в SIEM до виконання реакційної процедури на кінцевому вузлі та фіксації результату у структурованому журналі. Окремо підтверджено, що використання SSH Host CA дає змогу забезпечити безпечний автономний доступ до кінцевих вузлів без ручного підтвердження під час інциденту. Отримані результати також показали, що запропонована архітектура може бути масштабована на інші сценарії реагування за умови зміни правил виявлення та процедур виконання.
Висновки. Отримані результати підтверджують, що поєднання SIEM-аналітики з системою оркестрації дає змогу реалізувати керований контур автономного реагування на інциденти. Результат аналітики в SIEM перетворюється на інцидентний запис, який використовується для контролю повторних запусків, узгодження цільового активу з inventory та запуску сценарію реагування. Практична перевірка на прикладі SSH brute-force підтвердила технічну здійсненність

такого підходу: реалізовано повний цикл від виявлення події до виконання реакції та фіксації її результату в журналі. Запропонована архітектура придатна для реагування на інциденти, зафіксовані з використанням різних джерел подій, якщо їх результати агрегуються та корелюються в SIEM. Використання SSH Host CA забезпечує безпечний автономний доступ до кінцевих вузлів без ручного підтвердження під час інциденту. Подальший розвиток роботи доцільно пов'язати з програмною реалізацією модуля-міста, розширенням бібліотеки сценаріїв реагування та передаванням журналу реагування до SIEM для подальшого аналізу.
Ключові слова: *Splunk, Ansible, SIEM, оркестроване реагування, автономне реагування, SSH brute-force, SSH Host CA.*

Як цитувати: Братко Д. В., Кубрак В. О., Матійко А. А. "Автономна оркестрована система реагування на інциденти на основі SIEM". *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2026. вип. 69. С.20-32. https://doi.org/10.26565/2304-6201-2026-69-02

How to quote: D. Bratko, V. Kubrak, and A. Matiiko, "Autonomous Orchestrated Incident Response System Based on SIEM", Bulletin of V. N. Karazin Kharkiv National University, series Mathematical Modelling. Information Technology. Automated Control Systems, vol. 69, pp. 20–32, 2026. https://doi.org/10.26565/2304-6201-2026-69-02 [in Ukrainian]

Вступ

Стрімке ускладнення сучасних інформаційних інфраструктур і зростання кількості джерел телеметрії суттєво змінюють підходи до виявлення та обробки інцидентів інформаційної безпеки. У практичних середовищах події, що можуть свідчити про інцидент, надходять із журналів операційних систем і сервісів, мережевих сенсорів, сканерів вразливостей, засобів контролю доступу та інших систем моніторингу. За таких умов саме SIEM-платформи дедалі частіше виступають центральним рівнем збору, індексації та кореляції подій, оскільки дозволяють звести різнорідну телеметрію до єдиного аналітичного простору та виявляти підозрілі закономірності на основі узагальнених правил і запитів [1–4]. Водночас саме виявлення інциденту ще не гарантує своєчасного реагування, оскільки на практиці між аналітичним результатом SIEM і фактичним виконанням дій на цільовому активі часто існує розрив.

Ця проблема особливо помітна в середовищах, де реагування повинно бути швидким, повторюваним і незалежним від первинного джерела події. Більшість існуючих контурів автоматизації або орієнтовані на окремий тип телеметрії, або вимагають побудови окремих інтеграцій для кожного нового джерела даних. У результаті організація отримує не єдиний механізм реагування, а набір розрізнених сценаріїв, які складно супроводжувати, масштабувати та перевіряти з позиції аудиту. Крім того, навіть за наявності коректно налаштованих детекторів, автономне реагування ускладнюється необхідністю узгодження ідентифікаторів активів між SIEM та системою оркестрації, контролем повторних запусків однієї й тієї самої процедури, а також потребою фіксації результатів виконаних дій у вигляді доказового журналу.

Окрему проблему становить безпечний доступ до кінцевих вузлів у момент реагування. Якщо система оркестрації під час інциденту залежить від ручного підтвердження SSH-з'єднання, автоматизація втрачає оперативність. Якщо ж довіра до вузла формується через автоматичне прийняття першого ключа, виникає ризик підміни хоста та компрометації всього контуру реагування. Саме тому задача побудови автономного оркестрованого реагування повинна розглядатися не лише як проблема запуску playbook або скриптів, а як комплексна архітектурна задача, що охоплює уніфікацію інцидентного запису, керування повторюваністю дій, узгодження цільових активів і формування безпечного довіреного середовища для виконання процедур реагування.

У цьому контексті доцільним є використання проміжного архітектурного модуля-міста між SIEM та системою оркестрації, який дозволяє перетворити результат аналітики на структурований інцидентний запис, застосувати до нього визначені політики запуску та передати підготовлений набір параметрів до системи реагування. Такий підхід дає змогу побудувати універсальний контур автономного реагування для інцидентів, зафіксованих із використанням різних джерел подій, якщо їхні результати агрегуються та корелюються в SIEM. Як демонстраційний приклад у роботі розглянуто сценарій виявлення та реагування на SSH brute-force, оскільки він є наочним, відтворюваним і водночас практично значущим для сучасних інформаційних систем.

Метою статті є обґрунтування та експериментальна перевірка архітектурного підходу до автономного оркестрованого реагування на інциденти, у межах якого результати аналітики в

SIEM використовуються для формування інцидентного запису, запуску процедур реагування та журналювання їх виконання.

1. Теоретичні основи інтеграції SIEM та системи оркестрації

У сучасних інформаційних системах важливо не лише виявляти інциденти в SIEM, а й швидко та коректно перетворювати результат аналітики на практичну дію в інфраструктурі. Це особливо актуально в середовищах, де події надходять із різних джерел журналів операційних систем і сервісів, мережевих сенсорів, сканерів вразливостей та інших засобів моніторингу. Саме тому доцільним є використання проміжного архітектурного модуля, який поєднує аналітичний рівень SIEM із системою оркестрованого реагування та забезпечує керований перехід від результатів аналізу до виконання процедури на цільовому активі. [1,3,6]

Запропонований підхід поєднує аналітичний рівень SIEM, модуль-міст, систему оркестрації та контур керованої довіри до активів. Для цього інцидент подається у стандартизованому вигляді, повторні спрацювання контролюються через унікальний ключ інциденту та інтервал блокування повторного запуску, а безпечний доступ до кінцевих вузлів забезпечується без ручного підтвердження SSH-з'єднання під час інциденту. Загальну схему взаємодії цих компонентів наведено на рисунку 1.

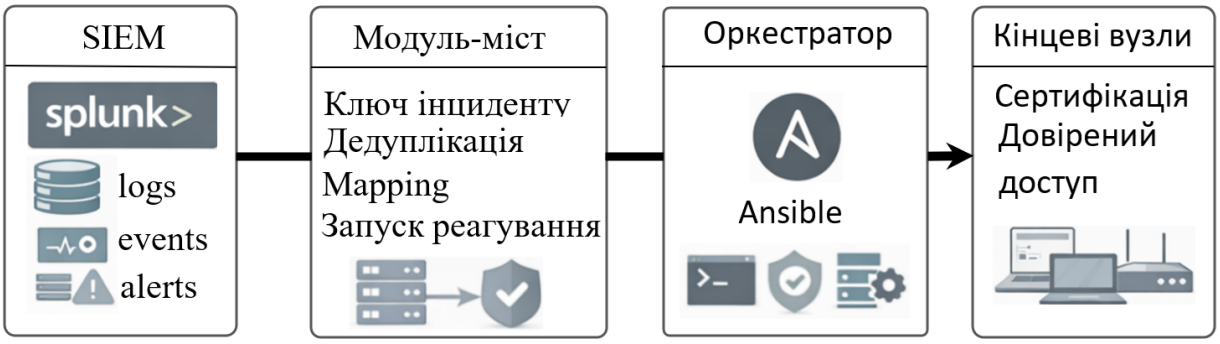

Рис. 1 Архітектура системи оркестрованого реагування на інциденти в SIEM-середовищі
Fig. 1 Architecture of an orchestrated incident response system in a SIEM environment

У межах запропонованої архітектури SIEM-шар виконує функції збору, індексації та аналітичної обробки подій, модуль-міст відповідає за інтерпретацію результатів пошуку, формування інцидентного запису, контроль повторних запусків і передачу параметрів до оркестратора, а система оркестрації виконує безпосередні дії реагування на визначених активах. Окремо функціонує керований контур довіри, побудований на основі SSH Host CA, який забезпечує безпечне підключення до кінцевих вузлів без ручного підтвердження ключів під час інциденту. Результати виконання процедур реагування фіксуються в журналі подій, що може виступати основою для подальшого аудиту та, за потреби, повторної індексації в SIEM, що дозволяє будувати точні умови запуску сцинаріїв реагувань.

Компонент SIEM і формування виходу аналітики. SIEM у запропонованому контурі виконує роль «точки нормалізації»: незалежно від того, чи подія походить із журналів ОС, мережевого сенсора або прикладного сервісу, вона потрапляє в єдиний аналітичний простір із можливістю кореляції. Детектор описується як запит до індексованих подій, а його результат має бути достатньо структурованим, щоб стати входом для реагування. Для Splunk типовим є опис детектора мовою SPL і виконання пошуку як у веб-інтерфейсі, так і через CLI або механізми alerting/scheduling. [6,7,12]

Для демонстрації працездатності підходу використано сценарій виявлення невдалих спроб SSH-автентифікації на цільовому вузлі. У межах експерименту аналітика будується на пошуку подій типу «Failed password» у журналах хоста, після чого виконується вилучення імені користувача та IP-адреси джерела, далі агрегація подій за полями host, src_ip та user. Такий підхід дає змогу виділити повторювані невдалі спроби входу, які можуть свідчити про brute-force активність. У запропонованому прикладі порогове спрацювання встановлено на рівні 5 невдалих спроб, що дозволяє виділити повторювану активність із ознаками brute-force та сформувати структурований результат для подальшого реагування. Отриманий результат далі

використовується для формування інцидентного запису та запуску відповідної процедури реагування. [6,7]

Лістинг 1. Приклад SPL-запиту
Listing 1. Example of an SPL query

```
index=* host=ub7* "Failed password"
| rex field=_raw "Failed password for (?:invalid user )?(?<user>\S+) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| stats count as failed_attempts latest(_time) as last_seen by host src_ip user
| where failed_attempts >= 5
| sort - failed_attempts
```

Наведений запит демонструє, як на основі журналів SSH можна виявити повторювані невдалі спроби автентифікації та звести їх до структурованого аналітичного результату. Агрегація за полями host, src_ip та user дає змогу отримати компактний набір ознак, достатній для формування інцидентного запису, оцінювання інтенсивності події та подальшого запуску процедури реагування. Використання порогової умови дозволяє відокремити поодинокі помилки входу від активності, що має ознаки brute-force, а сортування результатів спрощує пріоритезацію виявлених випадків. Таким чином, цей запит виступає не лише засобом виявлення, а й джерелом вхідних даних для модуля-міста, який перетворює результат SIEM-аналітики на керовану дію в інфраструктурі. Результат SPL-агрегації та поля, які формують інцидентний запис, наведено на рисунку 2 [5, 6].

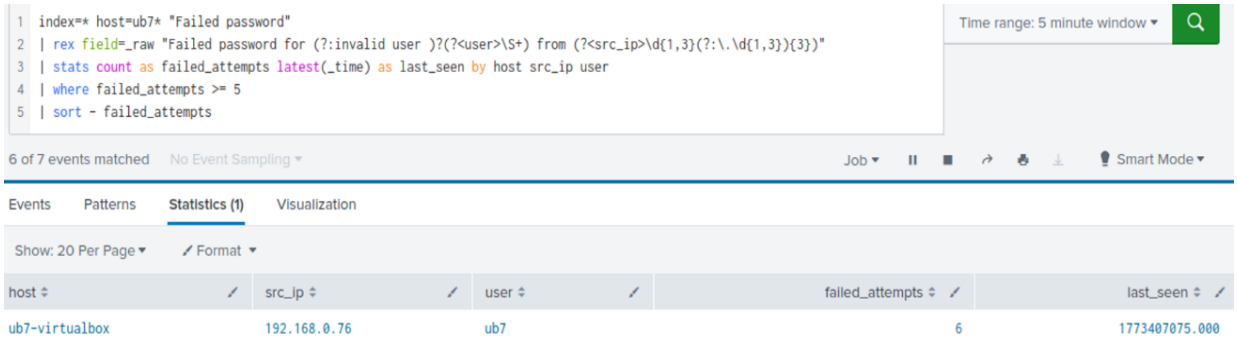

Рис. 2 Результат агрегації подій невдалих SSH-входів у Splunk
Fig. 2 Result of Aggregating Failed SSH Login Events in Splunk

Результат виконання запиту в SIEM містить поля, достатні для переходу від виявлення до реагування. Значення host описує цільовий актив, src_ip є адресою яка ініціювала подію, user є обліковим записом, щодо якого фіксуються спроби входу, failed_attempts це інтенсивність події, а last_seen її часову актуальність. На основі цього структурованого результату модуль-міст формує інцидентний запис, який використовується для логічної перевірки, оркестрації реагування та журналювання результатів. Саме в такому вигляді результат SIEM-аналітики переходить від етапу виявлення до етапу практичного реагування [6].

Таким чином можна сказати, що SIEM доцільно використовувати як централізований аналітичний рівень, у межах якого події з різнорідних джерел можуть бути зведені до узгодженого результату пошуку й кореляції. Для переходу від аналітики до практичної дії в інфраструктурі необхідний проміжний модуль, що забезпечує формалізацію інциденту, узгодження активів, керування повторними спрацюваннями та передавання параметрів до системи оркестрації. Така побудова створює основу для універсального контуру автономного реагування на інциденти, зафіксовані з використанням різних джерел подій [1, 2, 8].

Принципи формування інцидентного запису та функціонування модуля-міста

Робота модуля-міста визначається набором налаштувань, які задають правила інтерпретації аналітичного результату. До них належать назва правила виявлення, аналітичний запит, перелік потрібних полів, шаблон формування ключа інциденту, параметри контролю повторних запусків, спосіб узгодження активів із inventory та прив’язка до відповідної процедури реагування. Це

дозволяє модулю-місту працювати не як окремому скрипту під один сценарій, а як узагальненому механізму, який можна адаптувати до різних типів інцидентів шляхом зміни конфігурації. [5,6,7,8]

У межах реалізованого прототипу інцидентний запис формується у вигляді впорядкованого набору параметрів, що надалі передаються до модуля прийняття рішення та системи оркестрації. Параметри інцедентного запису отримуються зі SIEM системи, та формуються у json форматі після виявлення настання відповідного інциденту.

Лістинг 2. Приклад інцидентного запису у форматі JSON
Listing 2. Example of an Incident Record in JSON Format

```
{
  "rule_name": "ssh_bruteforce_multi_host",
  "host": "ub7-virtualbox",
  "src_ip": "192.168.0.76",
  "user": "ub7",
  "failed_attempts": 5,
  "last_seen": "1772905855.000"
}
```

Після формування інцидентного запису модуль-міст переходить до етапу логічної перевірки, у межах якого для кожного виявленого випадку формується унікальний ключ інциденту. У межах реалізованого прототипу такий ключ формується на основі назви правила та основних атрибутів події цільового активу, джерела активності й облікового запису користувача. Приклад формування такого ключа наведено в лістингу 3 [3, 5].

Лістинг 3. Приклад формування унікального ключа інциденту
Listing 3. Example of Generating a Unique Incident Key

```
incident_key = (
    f"ssh_bruteforce_multi_host|"
    f"host={host}|src_ip={src_ip}|user={user}"
)
```

Слід зазначити, що наведене формування унікального ключа інциденту (incident_key) не є жорстко прив’язаним лише до сценарію SSH brute-force. У межах повноцінної програмної реалізації генерація таких ключів має виконуватися автоматизовано на основі шаблону, що відповідає конкретному типу інциденту. Для різних класів подій склад інцидентного ключа може відрізнятися, оскільки він повинен включати саме ті атрибути, які дозволяють однозначно відокремити новий випадок від повторного спрацювання. Отже, зі збільшенням кількості підтримуваних сценаріїв реагування система має масштабувати і механізм формування ключів, зберігаючи при цьому єдиний принцип їх побудови [3].

Після формування ключа інциденту модуль виконує перевірку умов запуску. Якщо подія з таким самим ключем не оброблялася раніше або для неї вже завершився інтервал блокування повторного запуску, система автономно ініціює відповідну процедуру реагування. Якщо ж інцидент уже був оброблений у межах заданого інтервалу, повторний запуск не виконується. Таким чином, модуль-міст забезпечує автономне оркестроване реагування, за якого після виявлення інциденту система самостійно приймає рішення щодо запуску процедури відповідно до визначених політик. Це дає змогу поєднати аналітичний результат SIEM із керованим виконанням дій у цільовій інфраструктурі та забезпечити стійкість, повторюваність і практичну придатність усього контуру реагування [3].

Наступним етапом після перевірки умов запуску є узгодження цільового активу з даними інвентаризації системи оркестрації. Це необхідно тому, що ідентифікатор host, отриманий із SIEM, не завжди повністю збігається з тим, як відповідний вузол описаний у системі керування інфраструктурою. У SIEM актив може бути представлений у вигляді hostname, FQDN, IP-адреси або іншого позначення, сформованого джерелом журналювання, тоді як в inventory оркестратора той самий вузол може бути заданий під іншим ім’ям або через окремий параметр підключення. Саме тому перед запуском процедури реагування необхідно виконати етап зіставлення, у межах

якого аналітичний ідентифікатор активу приводиться у відповідність до запису, придатного для системи оркестрації [8].

Таке узгодження дає змогу уникнути ситуацій, коли інцидент коректно виявлено, але реакція не може бути виконана через невідповідність між позначенням вузла в SIEM та його описом в inventory. У межах запропонованого підходу цей механізм виступає важливою ланкою між етапом аналітики та етапом практичного реагування, оскільки саме він забезпечує коректну передачу інциденту на рівень виконання дій. У більш загальному вигляді такий підхід може бути реалізований через таблицю відповідностей, інвентаризаційний довідник або інший механізм зіставлення активів, що дає змогу масштабувати систему на різні типи вузлів і різні джерела подій.

Узгодження активу з inventory ще не гарантує можливості безпечного виконання реакційної процедури. Після того як модуль-міст визначив цільовий вузол, система повинна отримати автономний, але безпечний адміністративний доступ до нього. Якщо оркестратор під час інциденту залежить від ручного підтвердження SSH-підключення, підхід втрачає автономність. Якщо ж довіра до вузла формується через автоматичне прийняття першого ключа, це створює ризик підміни хоста. Саме тому безпечний доступ до кінцевих вузлів слід розглядати як одну з центральних умов працездатності всього контуру реагування [1,8,10].

Для розв’язання цієї задачі в архітектурі системи передбачено окремий контур довіри, який включає SIEM-сервер, сервер оркестрації, сервер сертифікації та кінцеві вузли. Особливу роль у цій схемі відіграє сервер оркестрації, оскільки саме він поєднує аналітичний рівень із практичним виконанням дій реагування. Водночас безпечний доступ до кінцевого вузла забезпечується не під час інциденту, а завдяки попередньо сформованому довіреному середовищу. Для цього використовується SSH Host CA, що дозволяє відмовитися від небезпечної довіри до першого ключа та зберегти обов’язкову перевірку SSH-ключа хоста перед підключенням [8,9,10].

Отже, сертифікація має передувати автономному реагуванню: вузли, на які потенційно можуть поширюватися реакційні дії, повинні бути заздалегідь підготовлені до безпечної взаємодії з оркестратором. Саме завдяки цьому контур довіри не прив’язується до одного сценарію, а стає універсальною основою для автономного реагування на інциденти різних типів [1,2].

Після завершення етапів інтерпретації інциденту, перевірки політик запуску, узгодження цільового активу з inventory та встановлення довіреного контуру доступу система переходить безпосередньо до запуску процедури реагування. На цьому етапі сервер оркестрації передає до playbook параметри, сформовані на основі інцидентного запису. До таких параметрів можуть належати ідентифікатор цільового вузла, тип або назва правила, IP-адреса джерела активності, ім’я користувача, кількість зафіксованих спроб, час останнього прояву події, а також службові атрибути, необхідні для журналювання та подальшого аналізу. У результаті playbook отримує не абстрактну команду на виконання, а структурований набір даних, який безпосередньо пов’язаний із конкретним інцидентом [8,9].

Подальше виконання playbook на кінцевому вузлі залежить від типу інциденту та обраної політики реагування. У загальному випадку така процедура може виконувати одну або кілька дій: фіксацію факту реагування, збір додаткових артефактів, зміну параметрів захисту, тимчасове обмеження доступу або інші керовані адміністративні операції. У межах реалізованого прототипу для підтвердження працездатності підходу playbook виконує контрольну дію на цільовому вузлі та залишає артефакт, який дозволяє однозначно перевірити факт виконання реакції. Такий підхід є принципово важливим, оскільки дає змогу не лише ініціювати процедуру реагування, а й підтвердити, що вона справді була виконана на визначеному активі [1,8,9,11].

Таким чином, система оркестрації виступає завершальною ланкою переходу від аналітичного результату SIEM до практичної дії в інфраструктурі. Якщо SIEM формує ознаки інциденту, а модуль-міст приймає рішення щодо доцільності запуску, то playbook реалізує безпосереднє застосування обраного сценарію до цільового вузла. Саме на цьому етапі автономне оркестроване реагування набуває завершеного вигляду, оскільки виявлена подія не лише фіксується, а й перетворюється на конкретну контрольовану дію з можливістю подальшого аудиту.

Програмна реалізація модуля-міста та практична перевірка запропонованого підходу

Для практичної перевірки запропонованого підходу на поточному етапі було реалізовано прототип модуля-міста у вигляді Python-скрипта. Його призначення полягає у підтвердженні технічної здійсненності методу та відтворенні основної логіки взаємодії між SIEM-аналітикою і системою оркестрації. У межах реалізованого прототипу скрипт приймає структурований

результат аналітичного запиту, формує інцидентний запис, будує унікальний ключ інциденту, перевіряє інтервал блокування повторного запуску, виконує зіставлення активу з inventory та передає підготовлений набір параметрів до playbook. Подальші лістинги демонструють реалізацію цих етапів у прототипному скрипті.

Наведений інцидентний запис у лістингу 2 показує, що кожне поле виконує окрему функцію в подальшій логіці обробки. Значення rule_name використовується для прив'язки запису до конкретного правила виявлення та відповідної процедури реагування. Поле host визначає вузол, до якого потенційно має бути застосована реакція, а src_ip характеризує джерело події. Поле user уточнює, щодо якого облікового запису зафіксовано підозрілу активність. Значення failed_attempts дозволяє оцінити інтенсивність події, а last_seen задає її часовий контекст. Саме на основі цих атрибутів надалі формується унікальний ключ інциденту, перевіряються політики повторного запуску та готуються параметри для передачі до playbook.

Після формування інцидентного запису скрипт переходить до побудови унікального ключа інциденту, який використовується для логічного відокремлення нового випадку від повторного спрацювання. Такий ключ формується на основі найбільш суттєвих атрибутів події, що дозволяють однозначно ідентифікувати конкретний інцидент у межах обраного сценарію. У розглянутому прикладі до складу ключа входять назва правила, цільовий актив, джерело активності та обліковий запис користувача. Використання саме цих полів дає змогу пов'язати між собою всі повторювані прояви одного й того самого випадку та, водночас, відокремити їх від інших подібних подій. Таким чином, incident_key виступає не лише технічним ідентифікатором, а й основою для подальшої перевірки політик повторного запуску, що є необхідною умовою автономного оркестрованого реагування.

Після формування ключа інциденту скрипт виконує перевірку умов запуску реагування. Для цього використовується локальний стан системи, що зберігається в таблиці trigger_state бази SQLite. На основі пари rule_name та incident_key визначається, чи оброблявся такий інцидент раніше, а також чи не перебуває він у межах заданого інтервалу блокування повторного запуску. Якщо різниця між поточним часом і часом останнього спрацювання менша за значення cooldown_sec, повторний запуск не виконується. Такий механізм дозволяє уникнути багаторазового ініціювання однієї й тієї самої процедури реагування, зменшує навантаження на систему оркестрації та забезпечує керованість автономного реагування.

Лістинг 4. Перевірка інтервалу блокування повторного запуску
Listing 4. Checking the restart lockout interval

```
    def is_in_cooldown(
        conn: sqlite3.Connection,
        rule_name: str,
        incident_key: str,
        cooldown_sec: int,
    ) -> bool:
        cur = conn.execute(
            "SELECT last_triggered_epoch FROM trigger_state WHERE rule_name=? AND
incident_key=?",
            (rule_name, incident_key),
        )
        r = cur.fetchone()
        if not r:
            return False
        last_ts = int(r[0])
        now = int(dt.datetime.utcnow().timestamp())
        return (now - last_ts) < int(cooldown_sec)
```

У наведеному фрагменті перевірка виконується шляхом звернення до локальної таблиці стану trigger_state, де для кожної пари rule_name та incident_key зберігається час останнього успішного спрацювання. Якщо відповідний запис відсутній, система вважає інцидент новим. Якщо ж запис існує, скрипт порівнює поточний час із моментом останнього запуску та визначає, чи не потрапляє подія в межі інтервалу блокування повторного запуску.

Якщо інцидент уже оброблявся в межах заданого інтервалу, система не виконує повторний запуск і фіксує стан cooldown_skip у журналі. Якщо ж інцидент є новим або інтервал блокування вже завершився, формується набір параметрів для передачі до playbook, та ініціюється процедура реагування. Таким чином, саме на цьому етапі модуль-міст реалізує автономне оркестроване реагування, оскільки після виявлення події система самостійно вирішує, чи потрібно запускати відповідну дію, в подальшому можна реалізувати умови виконання, коли інцедент потребує додаткового підтвердження не лише за фактором часу, а й за особливостями інших подій, що дозволяє реагувати на специфічну кореляцію деяких подій.

Лістинг 5. Умовний перехід до запуску реагування
Listing 5. Conditional transition to trigger response

```
    if is_in_cooldown(conn, rule_name, incident_key, int(rule["cooldown_sec"])):
        log_json_line(
            log_file,
            {
                "ts": utc_now_iso(),
                "rule": rule_name,
                "status": "cooldown_skip",
                "incident_key": incident_key,
                "splunk_host": splunk_host,
                "target_host": mapped_target,
            },
        )
    continue
```

Окремим етапом роботи скрипта є зіставлення цільового активу, отриманого з результату SIEM-аналітики, з його представленням у системі оркестрації. Це необхідно, оскільки значення host у журналі подій не завжди збігається з тим, як відповідний вузол описаний в inventory Ansible. У межах реалізованого прототипу таке зіставлення виконується через таблицю відповідностей host_map, що дозволяє перетворити аналітичний ідентифікатор активу у формат, придатний для подальшої роботи з inventory і запуску playbook.

Лістинг 6. Приклад зіставлення ідентифікатора активу з даними inventory
Listing 6. Example of mapping asset identifier to inventory data

```
    def map_target_host(cfg: dict, splunk_host_value: str) -> str:
        host_map = cfg.get("host_map", {}) or {}
    return host_map.get(splunk_host_value, splunk_host_value)
```

У наведеному фрагменті скрипт звертається до словника host_map, у якому зберігаються відповідності між позначеннями активів у SIEM та системі оркестрації. Якщо для отриманого значення host існує явне зіставлення, використовується перетворене значення; якщо ж такого запису немає, скрипт залишає початковий ідентифікатор без змін.

Після коректного зіставлення цільового активу з даними inventory система переходить до підготовки параметрів для запуску playbook. На цьому етапі сервер оркестрації передає до процедури реагування структурований набір даних, сформований на основі інцидентного запису та результатів зіставлення активу. У результаті playbook отримує не абстрактну команду, а пов’язаний із конкретним інцидентом контекст, необхідний для виконання дії на цільовому вузлі.

Лістинг 7. Формування параметрів для передавання до playbook
Listing 7. Preparation of Parameters for Passing to the Playbook

```
def build_extra_vars(cfg: dict, rule: dict, row: dict, incident_key: str) -> dict:
    extra_vars = {}

    for var_name, field_name in (rule.get("extra_vars_map", {}) or {}).items():
        extra_vars[var_name] = row.get(field_name, "")

    extra_vars.update(rule.get("extra_vars_static", {}) or {})

    if "target_host" in extra_vars:
        extra_vars["target_host"] = map_target_host(cfg, str(extra_vars["target_host"]))

    extra_vars["incident_key"] = incident_key
    extra_vars["rule_name"] = rule["name"]
  return extra_vars
```

У наведеному фрагменті скрипт формує словник extra_vars, у який включаються як значення, отримані з результату SIEM-пошуку, так і службові параметри, визначені конфігурацією правила. Окремо до цього набору додаються incident_key та назва правила, що дозволяє зберегти зв’язок між процедурою реагування і конкретним інцидентом.

Перед запуском дії реагування bridge-скрипт формує виклик ansible-playbook, у якому поєднуються шлях до цільового playbook, джерело інвентаря та набір параметрів інциденту, підготовлених на попередніх етапах обробки події. Змінні інциденту передаються у форматі JSON через механізм extra vars, що дає змогу динамічно підставляти до playbook значення цільового вузла. У лістингу 8 наведено фрагмент функції, яка реалізує запуск playbook та повертає результат виконання для подальшого аналізу й журналювання.

Лістинг 8. Запуск playbook із параметрами інциденту
Listing 8. Running the Playbook with Incident Parameters

```
def run_playbook(cfg: dict, rule: dict, extra_vars: dict) -> subprocess.CompletedProcess:
    ansible_cfg = cfg["ansible"]
    cmd = [
        "ansible-playbook",
        rule["playbook"],
        "-i",
        ansible_cfg["inventory"],
        "-e",
        json.dumps(extra_vars, ensure_ascii=False),
    ]
    timeout = int(ansible_cfg.get("playbook_timeout_sec", 300))
  return subprocess.run(cmd, capture_output=True, text=True, timeout=timeout)
```

Цей фрагмент показує, що запуск playbook виконується безпосередньо зі скрипта модуля-міста. До команди передаються шлях до playbook, інвентаризаційний файл і сформований набір параметрів extra_vars, що забезпечує прив’язку дій оркестратора до конкретного інциденту та цільового вузла.

Завершальним етапом роботи скрипта є журналювання результату виконання процедури реагування. Його призначення полягає в тому, щоб зафіксувати не лише факт запуску дії, а й її підсумковий стан. У межах реалізованого підходу журнал формується у структурованому вигляді та містить часову мітку, назву правила, ключ інциденту, статус виконання, цільовий актив і фрагменти службового виводу. Завдяки цьому він забезпечує простежуваність обробки інциденту та придатний для подальшого аудиту.

Важливо, що журнал охоплює не лише успішні випадки, а й інші стани роботи системи, зокрема відсутність збігів, блокування повторного запуску, помилки пошуку в SIEM і невдале виконання playbook. Таким чином, він відображає не лише результат реагування, а й логіку

прийняття рішення системою, що дозволяє використовувати його для технічної діагностики та, за потреби, повторної індексації в SIEM.

Лістинг 9. Формування журналу результатів виконання playbook
Listing 9. Generation of a Log of Playbook Execution Results

```
    payload = {
        "ts": utc_now_iso(),
        "rule": rule_name,
        "incident_key": incident_key,
        "status": "playbook_success" if result.returncode == 0 else "playbook_failed",
        "rc": result.returncode,
        "playbook": rule["playbook"],
        "target_host": extra_vars.get("target_host", ""),
        "splunk_host": splunk_host,
        "row": row,
        "stdout_tail": (result.stdout or "")[-1500:],
        "stderr_tail": (result.stderr or "")[-1500:],
    }
log_json_line(log_file, payload)
```

У наведеному фрагменті журнал формується як структурований запис, що містить часову мітку, назву правила, ключ інциденту, статус виконання, код завершення, ідентифікатор playbook, цільовий актив та фрагменти службового виводу. Поле status дозволяє одразу відрізнити успішне виконання від невдалого, а збереження stdout_tail і stderr_tail спрощує подальший аналіз причин помилок без необхідності повторного відтворення інциденту. Така структура журналу робить його придатним як для локального технічного контролю, так і для подальшої інтеграції в засоби централізованого моніторингу.

Окремою перевагою є те, що журналювання безпосередньо пов’язується з інцидентним записом через incident_key. Це означає, що кожен запис про виконання або пропуск процедури реагування може бути однозначно співвіднесений із конкретним виявленим випадком. У практичному сенсі така прив’язка дає змогу аналізувати не лише окремі події, а й повний життєвий цикл інциденту: від моменту його виявлення в SIEM до завершення реакційної процедури або відмови від її запуску за визначеними політиками. Саме це і забезпечує доказовий характер журналу реагування.

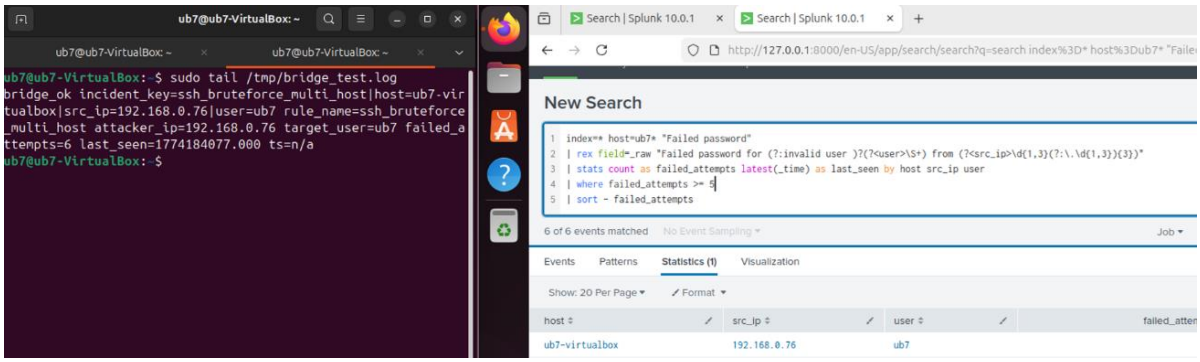

Рис. 3 Підтвердження виявлення інциденту у SIEM та виконання реакції на цільовому вузлі
Fig. 3. Confirmation of Incident Detection in the SIEM and Execution of the Response on the Target Host

На рисунку 3 наведено приклад виявлення повторюваних невдалих SSH-автентифікацій у Splunk, а також підтвердження того, що після цього на цільовому вузлі було виконано реакційну процедуру, результат якої зафіксовано у файлі bridge_test.log.

Отже, отримані результати підтверджують, що поєднання SIEM-аналітики з проміжним модулем-містом і системою оркестрації дає змогу реалізувати керований контур автономного реагування на інциденти. У ході дослідження показано, що результат аналітики в SIEM може бути послідовно перетворений на інцидентний запис, використаний для формування унікального ключа інциденту, перевірки політик повторного запуску, узгодження цільового активу з inventory та подальшого запуску сценарію реагування (playbook). Практична перевірка на прикладі SSH

brute-force підтвердила технічну здійсненність такого підходу: було реалізовано повний цикл від виявлення події в SIEM до виконання сценарію реагування на цільовому вузлі та фіксації результату в журналі реагування. Окремо підтверджено, що використання SSH Host CA дає змогу забезпечити безпечний автономний доступ до кінцевих вузлів без ручного підтвердження під час інциденту.

Висновки

По-перше, SIEM доцільно використовувати як централізований аналітичний рівень, у межах якого події з різнорідних джерел можуть бути зведені до узгодженого результату пошуку й кореляції. Для переходу від аналітики до практичної дії в інфраструктурі необхідний проміжний модуль, що забезпечує формалізацію інциденту, узгодження активів, керування повторними спрацюваннями та передавання параметрів до системи оркестрації. Саме така побудова створює основу для універсального контуру автономного реагування на інциденти, зафіксовані з використанням різних джерел подій. По-друге, система оркестрації є завершальною ланкою переходу від аналітичного результату SIEM до практичної дії в інфраструктурі. Якщо SIEM формує ознаки інциденту, а модуль-міст визначає доцільність запуску, то playbook реалізує безпосереднє застосування обраного сценарію до цільового вузла. Саме на цьому етапі автономне оркестроване реагування набуває завершеного вигляду, оскільки виявлена подія не лише фіксується і класифікується, а й перетворюється на конкретну контрольовану дію з можливістю подальшого аудиту її результатів. По-третє, поєднання SIEM-аналітики з проміжним модулем-містом і системою оркестрації забезпечує керований контур автономного реагування на інциденти. Результат аналітики в SIEM послідовно переходить в інцидентний запис, використовується для формування унікального ключа інциденту, перевірки політик повторного запуску, узгодження цільового активу з inventory та запуску сценарію реагування. Практична перевірка на прикладі SSH brute-force підтвердила технічну здійсненність такого підходу: реалізовано повний цикл від виявлення події в SIEM до виконання сценарію реагування на цільовому вузлі та фіксації результату в журналі реагування. Використання SSH Host CA забезпечує безпечний автономний доступ до кінцевих вузлів без ручного підтвердження під час інциденту.

Запропонований підхід вирізняється тим, що поєднує в єдиному контурі аналітичний рівень SIEM, модуль-міст, механізм контролю повторних запусків, засоби узгодження активів та систему оркестрації реагування. На відміну від рішень, орієнтованих на окремі типи телеметрії або вузькоспеціалізовані засоби захисту, така архітектура дозволяє будувати автономне реагування на інциденти, зафіксовані з використанням різних джерел подій, якщо їх результати агрегуються та корелюються в SIEM. У цьому контексті важливим є не лише саме виявлення події, а й перетворення результату аналітики на стандартизований інцидентний запис, придатний для подальшої логічної обробки, запуску процедури реагування та фіксації її результату.

Одержані результати розширюють уявлення про перехід від етапу виявлення інциденту до етапу практичного реагування в розподіленій інформаційній інфраструктурі. Запропонований підхід дає підстави розглядати автономне реагування не як набір окремих скриптів або локальних інтеграцій, а як цілісний керований контур із визначеними вхідними даними, умовами запуску, політиками повторного виконання та засобами аудиту. Практична цінність роботи полягає в тому, що така архітектура може бути використана як основа для реальних систем реагування, у яких необхідно не лише швидко перейти від виявлення до дії, а й зберегти керованість, відтворюваність і доказовість виконаних процедур.

Подальший розвиток роботи доцільно пов’язати зі створенням повноцінної програмної реалізації модуля-міста, розширенням бібліотеки сценаріїв реагування, підтримкою інших типів інцидентів, інтеграцією з довідниками активів та механізмами автоматичного збагачення інцидентного запису контекстною інформацією. Перспективним напрямом також є повторне індексування журналу реагування в SIEM для подальшого оцінювання ефективності політик і вдосконалення всього контуру автономного оркестрованого реагування.

СПИСОК ЛІТЕРАТУРИ

1. C. Pascoe, S. Quinn, K. Scarfone. The NIST Cybersecurity Framework (CSF) 2.0. NIST Cybersecurity White Paper 29. Gaithersburg, MD, USA : National Institute of Standards and Technology, 2024. DOI: 10.6028/NIST.CSWP.29. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf
2. K. Rigopoulos, S. Quinn, C. Pascoe, J. Marron, A. Mahn, D. Topper. NIST Cybersecurity Framework 2.0: Resource & Overview Guide. NIST Special Publication 1299. Gaithersburg, MD, USA : National Institute of Standards and Technology, 2024. DOI: 10.6028/NIST.SP.1299. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1299.pdf
3. A. Nelson, S. Rekhi, M. Souppaya, K. Scarfone. Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile. NIST Special Publication 800-61 Rev. 3. Gaithersburg, MD, USA : National Institute of Standards and Technology, 2025. DOI: 10.6028/NIST.SP.800-61r3. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf
4. European Union Agency for Cybersecurity (ENISA). ENISA Threat Landscape 2024. 2024. (дата звернення: 23.03.2026) URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024
5. MITRE ATT&CK. Brute Force, Technique T1110 - Enterprise. (дата звернення: 23.03.2026) URL: https://attack.mitre.org/techniques/T1110/
6. Splunk. Search | Splunk Enterprise. (дата звернення: 23.03.2026) URL: https://docs.splunk.com/Documentation/Splunk/9.4.2/SearchReference/Search
7. Splunk. savedsearches.conf | Platform. (дата звернення: 23.03.2026) URL: https://docs.splunk.com/Documentation/Splunk/9.4.2/Admin/Savedsearchesconf
8. Ansible Project. How to build your inventory. https://docs.ansible.com/projects/ansible/latest/inventory_guide/intro_inventory.html
9. Ansible Project. ansible-playbook. (дата звернення: 23.03.2026) URL: https://docs.ansible.com/projects/ansible/latest/cli/ansible-playbook.html
10. OpenBSD. ssh-keygen(1). (дата звернення: 23.03.2026) URL: https://man.openbsd.org/OpenBSD-7.6/ssh-keygen.1
11. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Hoboken, NJ, USA : Wiley, 2020. https://www.wiley.com/en-cn/Security+Engineering%3A+A+Guide+to+Building+Dependable+Distributed+Systems%2C+3rd+Edition-p-9781119642787
12. Bejtlich R. The Tao of Network Security Monitoring: Beyond Intrusion Detection. Boston, MA, USA : Addison-Wesley, 2004. https://www.informit.com/store/tao-of-network-security-monitoring-beyond-intrusion-9780321246776

Bratko Dmytro Viacheslavovych *undergraduate, cadet*
Institute of Special Communication and Information Protection of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 4 Verkhnokliuchova St., Kyiv, Ukraine, 03056
e-mail: loading.2285@gmail.com;
https://orcid.org/0009-0001-0863-9285

Kubrak Volodymyr Oleksandrovych *PhD student, Senior Lecturer of Special Department № 1*
Institute of Special Communication and Information Protection of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 4 Verkhnokliuchova St., Kyiv, Ukraine, 03056
e-mail: v.kubrak@kpi.ua;
https://orcid.org/0000-0001-8877-5289

Matiiko Aleksandra Andriivna *PhD student, Associate Professor of Special Department № 1*
Institute of Special Communication and Information Protection of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 4 Verkhnokliuchova St., Kyiv, Ukraine, 03056
e-mail: alexm1710@ukr.net;
https://orcid.org/0000-0002-6947-5958

Autonomous orchestrated incident response system based on SIEM

Relevance. Modern information systems generate security events from various sources, including operating system and service logs, network sensors, vulnerability scanners, and other monitoring tools. In such conditions, SIEM enables the centralized collection, indexing, and correlation of telemetry; however, the transition from analytical results to practical response often remains insufficiently formalized. This leads to delays, dependence on manual actions, difficulties in ensuring the repeatability of procedures, and the absence of a unified mechanism for confirming executed response actions. An additional challenge is the provision of secure autonomous access to endpoints during an incident, when both manual confirmation of SSH connections and insecure trust on first use are unacceptable. In this context, the development of an architectural bridge between SIEM analytics and an orchestration system is highly relevant, as it can ensure controlled, repeatable, and auditable incident response regardless of the original source of events.
Goal. The purpose of this work is to substantiate and experimentally validate an architectural approach to autonomous orchestrated incident response, in which the results of SIEM analytics are transformed into a structured incident record and then used to initiate response procedures in an orchestration system. To achieve this goal, the detector is described in a declarative form, the incident record is standardized, repeated triggering is controlled through a unique incident key and a re-execution lockout interval, target assets are aligned with inventory data, execution results are logged, and secure access to endpoints based on SSH Host CA is implemented. The scenario of detecting and responding to an SSH brute-force attack was chosen as a demonstration case.
Results. As a result of the study, an architectural approach that combines SIEM analytics with the automated execution of response actions on target assets was developed and experimentally validated. It was shown that the result of an analytical query in a SIEM can be consistently transformed into an incident record, used to construct a unique incident key, verify re-execution policies, align the target asset with the inventory, and transfer parameters to a playbook. The implemented prototype confirmed the technical feasibility of building a complete cycle from event detection in the SIEM to the execution of a response procedure on an endpoint and the recording of the result in a structured log. It was also confirmed that the use of SSH Host CA makes it possible to provide secure autonomous access to endpoints without manual confirmation during an incident. The obtained results further demonstrated that the proposed architecture can be scaled to other response scenarios provided that the detection rules and execution procedures are adapted accordingly.
Conclusions. The obtained results confirm that the integration of SIEM analytics with an orchestration system makes it possible to implement a controlled framework for autonomous incident response. The result of SIEM analytics is transformed into an incident record, which is then used to control repeated executions, align the target asset with the inventory, and initiate a response scenario. Practical validation based on the SSH brute-force case confirmed the technical feasibility of this approach: a complete cycle was implemented, from event detection to response execution and the recording of its result in the log. The proposed architecture is suitable for responding to incidents identified from various event sources, provided that their results are aggregated and correlated in the SIEM. The use of SSH Host CA ensures secure autonomous access to endpoints without manual confirmation during an incident. Further development of this work should be associated with the software implementation of the bridge module, the expansion of the response scenario library, and the transfer of response logs back to the SIEM for further analysis.
Keywords: *Splunk, Ansible, SIEM, orchestrated response, autonomous response, SSH brute-force, SSH Host CA.*

Надійшла у першій редакції 07.02.2026; рекомендована до друку 11.03.2026; опублікована 03.04.2026.
The first version has been received on 07.02.2026; was recommended for printing on 11.03.2026; published 03.04.2026.