Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Автоматизована система моніторингу подій безпеки в хмарній платформі AWS»

Виконавець: студентка IV курсу, групи КБ-43
Христина ЦИБАК
(підпис) (ім'я, прізвище)

	Підпис	**Ім'я, прізвище**
Керівник		Олександр ТОРОШАНКО
Нормоконтроль		Юрій БАБЕНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студентці **КБ-43** **Цибак Христині Василівні**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Автоматизована система моніторингу подій безпеки в хмарній платформі AWS

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Концепція хмарних обчислень, інформаційна безпека, моніторинг інцидентів, інфраструктура як код

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Ознайомлення з хмарними технологіями та їх вразливостями, аналіз сучасних загроз інформаційній безпеці в хмарі, огляд підходів до моніторингу подій безпеки, вибір та оцінка засобів автоматизованого моніторингу, реалізація інфраструктури як коду у хмарі AWS, тестування та оцінка ефективності розробленої системи.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Програмна реалізація дозволяє автоматизувати процес моніторингу подій безпеки в хмарі, оперативно виявляти інциденти інформаційної безпеки та реагувати на них у реальному часі.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняла до виконання _____ Христина ЦИБАК
(підпис) (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 08.01.2025	*виконано*
2	Аналіз літератури	09.01.2025 – 21.02.2025	*виконано*
3	Ознайомлення з хмарними технологіями та загрозами безпеки	21.02.2025 – 17.03.2025	*виконано*
4	Аналіз підходів до моніторингу подій та вибір інструментів	18.03.2020 – 04.04.2025	*виконано*
5	Проєктування архітектури моніторингової системи	05.04.2025 – 20.04.2025	*виконано*
6	Програмна реалізація системи моніторингу	21.04.2025 – 17.05.2025	*виконано*
7	Тестування системи та оцінка ефективності	18.05.2025 – 25.05.2025	*виконано*
8	Оформлення пояснювальної записки	26.05.2025 – 01.06.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	02.06.2025 – 13.06.2025	*виконано*

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняла до виконання _____ Христина ЦИБАК
(підпис) (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 72 сторінки основного тексту, 1 таблицю та 18 рисунків. Список використаних джерел містить 40 найменувань і займає 5 сторінок.

Метою роботи є розробка автоматизованої системи моніторингу подій безпеки в хмарному середовищі AWS з використанням вбудованих сервісів платформи для підвищення ефективності виявлення та аналізу інцидентів.

Для досягнення зазначеної мети поставлено наступні завдання:

- Дослідити принципи функціонування хмарних платформ та виявити актуальні проблеми інформаційної безпеки в таких середовищах.
- Оцінити можливості вбудованих сервісів AWS для автоматизованого моніторингу подій безпеки.
- Розробити архітектуру автоматизованої системи моніторингу подій безпеки в AWS.
- Реалізувати інфраструктуру моніторингу за допомогою засобів автоматизації.
- Перевірити працездатність розробленої системи в тестовому середовищі та провести оцінку її ефективності.

Об'єктом дослідження є процес моніторингу подій безпеки в хмарній платформі Amazon Web Services.

Предметом дослідження є підходи та інструменти автоматизованого моніторингу подій безпеки у хмарному середовищі AWS.

Практичною цінністю отриманих результатів є програмна реалізація системи моніторингу безпеки в AWS, що дозволяє своєчасно виявляти інциденти та спрощує розгортання завдяки використанню інфраструктурного коду.

Ключові слова: безпека, AWS, хмарні обчислення, моніторинг, інфраструктура як код.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AWS	–	Amazon Web Services
IaaS	–	Infrastructure as a Service
PaaS	–	Platform as a Service
SaaS	–	Software as a Service
APT	–	Advanced Persistent Threat
OS	–	Operating System
RBAC	–	Role-Based Access Control
ABAC	–	Attribute-Based Access Control
DDoS	–	Distributed denial-of-service
IaC	–	Infrastructure as Code
SIEM	–	Security Information and Event Management
API	–	Application Programming Interface
S3	–	Simple Storage Service
VPC	–	Virtual Private Cloud
NIST	–	National Institute of Standards and Technology
CIS	–	Center for Internet Security
ISO/IEC 27001	–	International Standard for Information Security Management
GDPR	–	General Data Protection Regulation
SOC 2	–	System and Organization Controls 2
PCI / DSS	–	Payment Card Industry Data Security Standard
IAM	–	Identity and Access Management
SNS	–	Simple Notification Service

MFA	–	Multi-factor authentication
JSON	–	JavaScript Object Notation

ВСТУП

В сучасних умовах стрімкого розвитку інформаційних технологій усе більше організацій переходять до використання хмарних платформ для зберігання, обробки та аналізу даних. Хмарні обчислення надають організаціям можливість масштабувати свої ресурси, знижувати витрати та спрощувати керування інформаційними системами, проте ці переваги супроводжуються новими викликами в сфері забезпечення конфіденційності, цілісності та доступності інформації. Атаки на хмарну інфраструктуру, витоки конфіденційної інформації, неправильні налаштування доступу – усе це створює серйозні ризики для бізнесу.

Динамічний характер хмарних інфраструктур, зокрема таких як Amazon Web Services, вимагає впровадження високоефективного моніторингу подій безпеки. За відсутності належного моніторингу організації можуть втратити контроль над тим, що відбувається у середовищі, не помітити ознак компрометації або не відреагувати вчасно на інцидент. Тому впровадження автоматизованих систем моніторингу подій безпеки є критично важливим етапом забезпечення інформаційної безпеки організацій. Amazon Web Services як одна з провідних хмарних платформ пропонує широкий набір сервісів для контролю, аналізу та реагування на інциденти безпеки. Правильна інтеграція цих сервісів та використання автоматизації у розгортанні хмарних рішень дозволяє знизити ризики та підвищити рівень захищеності інфраструктури.

Багаторівневий підхід до моніторингу подій безпеки в хмарному середовищі дозволяє гнучко адаптувати інфраструктуру до поточних потреб завдяки параметризованому розгортанню, автоматизованій обробці подій та масштабувати обсяг контролю залежно від обраного рівня, оптимізуючи використання ресурсів. Такий підхід забезпечує вищу ефективність контролю безпеки порівняно з типовими рішеннями та може застосовуватись у різних

галузях, зокрема де застосовуються хмарні технології, в умовах підвищених вимог до контролю подій безпеки.

Метою даної роботи є розробка автоматизованої системи моніторингу подій безпеки в хмарному середовищі AWS з використанням вбудованих сервісів платформи для підвищення ефективності виявлення та аналізу інцидентів.

Для досягнення зазначеної мети поставлено наступні завдання:

- Дослідити принципи функціонування хмарних платформ та виявити актуальні проблеми інформаційної безпеки в таких середовищах.
- Оцінити можливості вбудованих сервісів AWS для автоматизованого моніторингу подій безпеки.
- Розробити архітектуру автоматизованої системи моніторингу подій безпеки в AWS.
- Реалізувати інфраструктуру моніторингу за допомогою засобів автоматизації.
- Перевірити працездатність розробленої системи в тестовому середовищі та провести оцінку її ефективності

Об’єктом дослідження є процес моніторингу подій безпеки в хмарній платформі Amazon Web Services.

Предметом дослідження є підходи та інструменти автоматизованого моніторингу подій безпеки у хмарному середовищі AWS.

Методи дослідження – аналіз літературних джерел, системний аналіз, порівняння, моделювання архітектури рішення, аналіз методів автоматизації.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ФУНКЦІОНУВАННЯ ХМАРНИХ СЕРЕДОВИЩ ТА РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОГО ЗАХИСТУ

1.1 Сучасні тенденції переходу до хмарних технологій

Зростання обсягів даних, цифровізація бізнес-процесів та глобальний перехід до онлайн-моделей роботи сформували нову технічну реальність, компанії все частіше шукають рішення, які забезпечують масштабованість, гнучкість та доступність та все частіше звертаються до хмарних обчислень як до базової інфраструктурної моделі. Хмарні технології забезпечують не лише зберігання даних, а й створюють повноцінне середовище для запуску, масштабування та інтеграції цифрових рішень, забезпечуючи зростання не як тимчасовий тренд, а фундаментальну зміну моделі побудови інформаційних систем [1]. Поняття хмарних обчислень почало формуватись ще в 1990-х роках, однак практичну реалізацію отримало лише у 2006 році з появою Amazon Web Services – першої масштабної платформи інфраструктури як послуги, згодом на ринку з’явилися Microsoft Azure, а у 2011 році – Google Cloud Platform та інші провайдери. Саме в цей період хмарні сервіси перестали бути лише альтернативою локальним серверам і стали основою для побудови складних IT-систем [2].

Після 2015 року хмарні сервіси активно почали впроваджуватись у державному секторі, освітніх закладах та промисловості, а пандемія COVID-19 у 2020 році стала каталізатором для різкого зростання попиту на хмарні рішення. Вимушений перехід на дистанційну роботу, необхідність забезпечення доступності сервісів 24/7 та скорочення витрат на інфраструктуру спричинили впровадження хмарних технологій у всіх галузях.

Аналітичні прогнози підтверджують стабільне зростання ринку хмарних послуг, зокрема, за даними Gartner, у 2025 році глобальні витрати кінцевих

користувачів на публічні хмарні сервіси сягнуть 723 мільярдів доларів США, що на 21,5% більше, ніж у 2024 році. Подібну динаміку фіксує й Precedence Research, як показано на рис. 1.1, очікується, що вже у 2025 році обсяг ринку перевищить 900 мільярдів доларів США, а до 2034 року може сягнути 5,15 трильйона доларів [3].

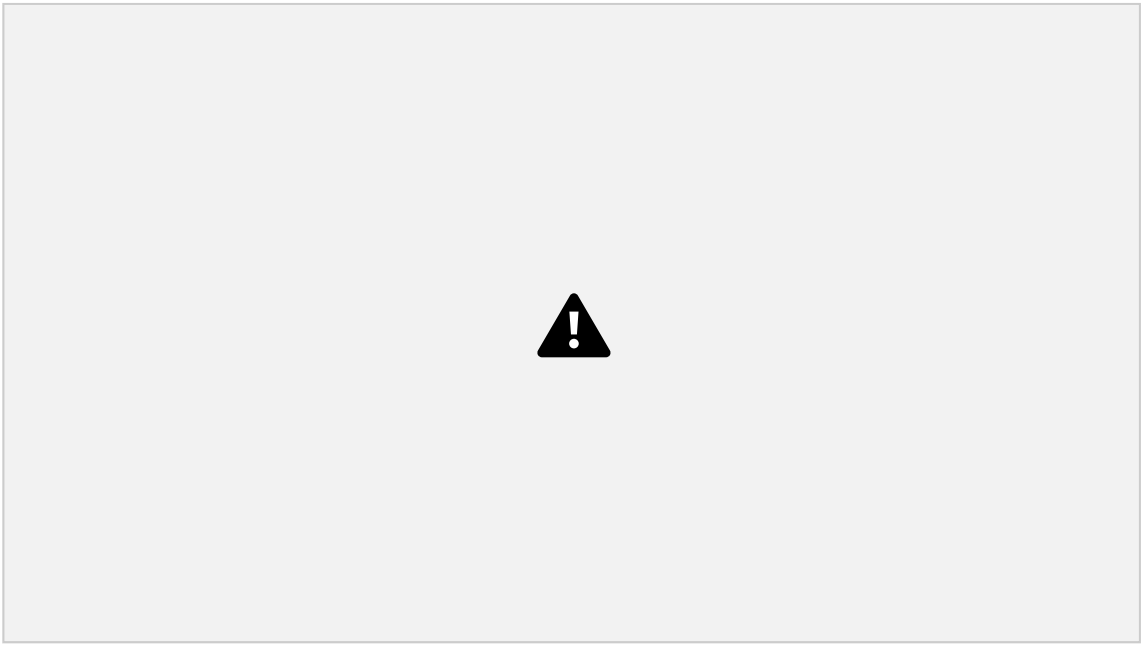

Рисунок 1.1 – Прогнозована динаміка зростання глобального ринку хмарних обчислень за даними Precedence Research

Близько 85% середніх і великих компаній частково або повністю перенесли свою IT-інфраструктуру у хмару, що свідчить про перетворення хмарних сервісів із допоміжного інструменту на ключовий елемент цифрової стратегії бізнесу [4].

Наприклад, компанія Netflix повністю перейшла до хмарної інфраструктури, вибудувала власну систему, здатну адаптуватися до змін і працювати на глобальному рівні без збоїв. Airbnb з самого початку використовував хмарні платформи для швидкого зростання та глобального охоплення. Capital One у 2015 році став першим великим банком, що повністю мігрував до публічної хмари, компанія реалізувала мікросервіси, безсерверні

обчислення та посилену безпеку, що дозволило знизити витрати на інфраструктуру на 50% та пришвидшити розгортання нових продуктів [5].

1.2 Теоретичні основи хмарних обчислень.

Хмарні обчислення – це модель доступу до обчислювальних ресурсів через інтернет, яка дозволяє організаціям використовувати інфраструктуру, платформи або програмне забезпечення як послугу без необхідності локального розгортання. Це модель, за якої ресурси, такі як сервери, сховища, бази даних, мережі або програмне забезпечення, надаються через Інтернет користувачеві, ключова ідея полягає в тому, що обчислювальні потужності стають сервісом, а не власністю.

Однією з головних переваг хмарних платформ є масштабованість, організації можуть швидко адаптувати свої ресурси до зростання або скорочення навантаження, не вкладаючи кошти у власні сервери. Це поєднується з гнучкістю – розгортання нових рішень або тестових середовищ відбувається за лічені хвилини, що особливо важливо в умовах швидкої зміни бізнес-вимог. Додатковою перевагою є оптимізація витрат, таким чином замість високих капітальних інвестицій у власні дата-центри, компанії платять лише за ті ресурси, які насправді використовують, що значно знижує бар'єр входу. Хмарна інфраструктура також забезпечує високу доступність і надійність, завдяки географічно розподіленим дата-центрам, аварійне відновлення та безперервність бізнесу стали стандартом навіть для малих підприємств. Сучасні хмарні провайдери інвестують у безпеку значні ресурси, пропонуючи такі механізми як шифрування даних, контроль доступу, моніторинг у режимі реального часу, відповідність міжнародним стандартам захисту інформації [6].

Сервіси в хмарному середовищі реалізуються за трьома основними моделями надання послуг – IaaS, PaaS та SaaS, які відрізняються рівнем керування ресурсами та гнучкістю для користувача.

Інфраструктура як послуга (IaaS) – це базовий рівень хмарного сервісу, що передбачає надання користувачеві віртуалізованих обчислювальних ресурсів, таких як серверів, сховищ, мережевої інфраструктури. У цій моделі користувач має максимальний контроль над операційною системою, програмним забезпеченням і конфігурацією середовища, однак не несе відповідальності за фізичне обслуговування обладнання. Типовими прикладами таких сервісів є Amazon EC2, Google Compute Engine, Microsoft Azure Virtual Machines. IaaS дозволяє будувати інфраструктуру з нуля і є популярним серед компаній, що потребують високої кастомізації або мають нестандартні потреби.

Платформа як послуга (PaaS) – це більш високий рівень абстракції, де постачальник хмарних послуг надає вже налаштоване середовище для розробки, тестування, розгортання та масштабування застосунків. Розробники можуть зосередитись на логіці застосунку, не переймаючись питаннями оновлень ОС, масштабування чи безпеки інфраструктури. Прикладами PaaS є Google App Engine, Heroku, AWS Elastic Beanstalk. Такий підхід особливо вигідний для стартапів і команд розробників, які цінують швидкість виведення продукту на ринок.

Програмне забезпечення як послуга (SaaS) – це модель, за якої кінцевий користувач отримує повністю готовий до використання програмний продукт, що працює у хмарі та доступний через вебінтерфейс або мобільний застосунок. Користувачу не потрібно дбати про інфраструктуру, налаштування чи оновлення – усе це забезпечує постачальник. Прикладами SaaS-рішень є Google Workspace, Microsoft 365, Dropbox, Salesforce. SaaS широко використовується як в особистих цілях, так і в корпоративному середовищі, оскільки забезпечує простий доступ, низький поріг входу та гнучку модель підписки.

Таким чином, моделі IaaS, PaaS і SaaS представляють собою різні рівні абстракції хмарних послуг, а саме від базової інфраструктури до готових бізнес-застосунків, вони не є взаємовиключними, а швидше доповнюють одна одну, де на основі інфраструктури (IaaS) будується платформа (PaaS), на якій розробляються та працюють прикладні сервіси (SaaS). Багато компаній

одночасно використовують усі три підходи, створюючи багаторівневу хмарну екосистему, яка охоплює як технічні компоненти, так і прикладні бізнес-процеси [7]. Цю ієрархічну структуру можна зобразити у вигляді піраміди, як це показано на рис. 1.2, де нижній рівень – це обчислювальні ресурси на вимогу, середній – середовище для розробки та управління застосунками, а верхній – готові програмні продукти, інтегровані в бізнес-процеси.

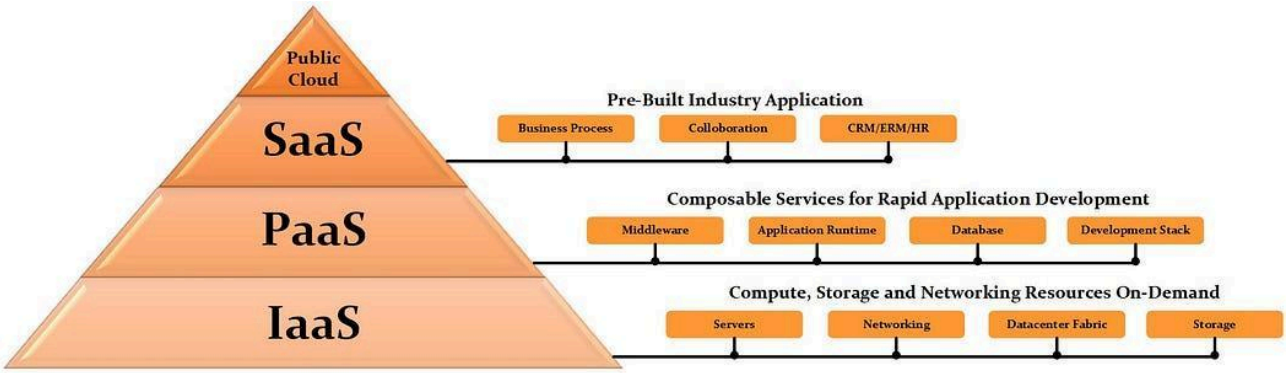

Рисунок 1.2 – Моделі хмарних сервісів за рівнем абстракції

Залежно від типу розгортання, хмари поділяються на публічні, приватні та гібридні. Публічна хмара – це інфраструктура, яку надає провайдер багатьом клієнтам одночасно, приватна хмара створюється виключно для однієї організації й може бути розміщена як на стороні провайдера, так і в межах самої організації. Гібридна модель дозволяє об'єднувати ресурси з різних джерел, наприклад, тримати конфіденційні дані в приватній хмарі, а додаткові обчислення – у публічній [8].

1.3 Основні положення та проблематика безпеки хмарних сервісів

Розвиток хмарних технологій значно змінив підходи до побудови та експлуатації інформаційних систем, хмарні обчислення забезпечують масштабованість, економічність та доступність ресурсів, що робить їх

привабливими для організацій різного масштабу, проте ці переваги супроводжуються низкою нових викликів, зокрема у сфері інформаційної безпеки. Особливості хмарної моделі, а саме віртуалізація, спільне використання ресурсів, розподілений доступ, а також часткова передача контролю зовнішньому провайдеру, створюють додаткові вектори атак і підвищують вимоги до захисту даних. Зростання залежності організацій від хмарних сервісів супроводжується підвищеними вимогами до захисту інформації, що передається та обробляється в цих середовищах. Ризики витоку даних обумовлені як зовнішніми загрозами, так і внутрішніми вразливостями – зокрема, помилками користувачів, слабкими механізмами автентифікації та потенційною наявністю шкідливих дій з боку інсайдерів [9].

Сучасні хмарні платформи активно інвестують у забезпечення високих стандартів безпеки. Виробники таких рішень як Amazon Web Services, Microsoft Azure або Google Cloud, впроваджують багаторівневі засоби захисту, включаючи шифрування даних, контроль доступу, розмежування прав користувачів, безперервний моніторинг активності та аудит системних подій. Більшість провайдерів хмарних сервісів проходять регулярну сертифікацію на відповідність міжнародним стандартам безпеки, таким як ISO/IEC 27001 або SOC 2, що формує певний рівень довіри до наданих ними інструментів. Разом із розвитком захисних механізмів у хмарних платформах зростає і складність ризиків, які постають перед користувачами. Однією з ключових проблем є поступове розширення поверхні атаки, що зумовлене зростаючим числом інтеграцій, відкритих інтерфейсів та віддаленого доступу до ресурсів. У хмарному середовищі кожен додатковий сервіс, кожна помилка конфігурації або неправильно налаштований API підвищують імовірність успішного вторгнення. Крім того, модель спільної відповідальності, яка є базовим принципом безпеки в хмарі, часто стає джерелом непорозумінь. Хоч постачальник хмарних послуг забезпечує захист самої інфраструктури, відповідальність за правильну конфігурацію сервісів, захист облікових даних, управління доступом та

шифрування даних покладається на кінцевого користувача. На практиці саме порушення цієї межі відповідальності стає причиною більшості інцидентів.

Особливу загрозу становить ризик витоку даних, як через дії зловмисників, так і через помилки або недбалість користувачів. Публічне розгортання приватних даних, відсутність багатофакторної автентифікації чи недостатнє шифрування є поширеними причинами втрати конфіденційної інформації.

Додатково слід враховувати регуляторні виклики, пов’язані з дотриманням нормативних вимог, у випадках, коли дані зберігаються або передаються між юрисдикціями, виникають труднощі із забезпеченням відповідності міжнародним стандартам, таким як GDPR, HIPAA чи ISO/IEC 27001. Обмежена прозорість і складність контролю над фізичним розміщенням даних ускладнюють управління правовими ризиками.

Окремо варто згадати про обмежену видимість внутрішніх процесів хмарної інфраструктури, а саме користувачі, як правило, мають доступ лише до частини логів і телеметрії, що не дозволяє повною мірою реалізувати аудит, інцидент-менеджмент та ретельне розслідування подій. Така ситуація знижує оперативність реагування на загрози та ускладнює впровадження ефективної системи моніторингу безпеки [10].

Отже, попри високий рівень захищеності, задекларований хмарними провайдерами, безпечність використання хмари значною мірою залежить від дотримання належних практик з боку користувача. Це підкреслює необхідність формування комплексної стратегії безпеки, яка враховує як технологічні, так і організаційні аспекти взаємодії з хмарною інфраструктурою.

1.3.1 Поширені загрози безпеці хмарних сервісів

Попри високий рівень технічної досконалості сучасних хмарних платформ, більшість інцидентів безпеки в таких середовищах є наслідком не вразливостей інфраструктури, а недоліків у її використанні. Помилки

конфігурації, неналежне управління обліковими даними, відсутність моніторингу або контрольних механізмів – усе це створює передумови для реалізації типових загроз, які становлять серйозну небезпеку для конфіденційності, цілісності та доступності даних. Розглянемо ключові типи загроз, що виникають при використанні хмарних сервісів, а також фактори, які сприяють їх реалізації.

Витік даних – є однією з найпоширеніших загроз, у більшості випадків він виникає внаслідок неправильної конфігурації ресурсів, наприклад, відкриття публічного доступу до хмарного сховища або відсутності шифрування критичної інформації. Додатковим фактором ризику виступає недостатній контроль над ідентифікацією та автентифікацією користувачів, зокрема використання слабких паролів чи відсутність багатофакторного захисту. У хмарних середовищах, де обробка й зберігання даних здійснюється через численні компоненти та сервіси, навіть одна вразливість може призвести до компрометації великих обсягів інформації. Показовим прикладом такого інциденту є витік даних у компанії Uber у 2016 році, коли зловмисники отримали доступ до облікових даних AWS, знайдених у відкритому репозиторії GitHub, що використовувався внутрішніми розробниками. Це дозволило їм скомпрометувати дані понад 57 мільйонів користувачів та водіїв. Компанія не повідомила про інцидент відразу, а натомість заплатила зловмисникам 100 тисяч доларів, щоб приховати факт витоку. Цей випадок став прикладом того, як неналежне управління обліковими даними в хмарному середовищі може призвести до значних репутаційних і юридичних наслідків, навіть без прямого зламу інфраструктури хмарного провайдера [11].

Неналежна конфігурація хмарної інфраструктури – становить одну з найпоширеніших причин порушення безпеки в хмарному середовищі. Через складність архітектури, велику кількість параметрів конфігурації та часті зміни, пов’язані з масштабуванням і автоматизацією, адміністратори або користувачі можуть несвідомо створити вразливе середовище. Типовими прикладами є відкритий доступ до сховищ даних, неправильні політики доступу IAM,

відсутність логування подій або шифрування. Подібні помилки не обов’язково є наслідком зловмисних дій, але в умовах хмарної інфраструктури навіть незначне відхилення від рекомендованих налаштувань може призвести до масштабних наслідків.

Атаки типу відмова в обслуговуванні (DoS) становлять суттєву загрозу для хмарної інфраструктури, оскільки спрямовані на порушення доступності сервісів шляхом навмисного перевантаження їх запитами. У хмарному середовищі така атака може мати подвійний ефект, з одного боку – унеможливити обслуговування легітимних користувачів, з іншого – спричинити надмірне автоматичне масштабування ресурсів, що веде до фінансових збитків для компанії. Розподілені DoS-атаки є особливо небезпечними, оскільки залучають велику кількість джерел трафіку, що ускладнює фільтрацію та блокування. Навіть короткочасне припинення доступу до критичних хмарних сервісів може мати серйозні наслідки для бізнесу, особливо в умовах залежності від безперервної роботи онлайн-систем.

Внутрішні загрози – це загрози, які виникають унаслідок дій осіб, що мають авторизований доступ до хмарних ресурсів, це можуть бути співробітники, адміністратори, підрядники чи партнери. Такі особи можуть умисно або випадково спричинити витік, зміну чи знищення даних, особливо небезпечними є дії привілейованих користувачів, оскільки їхня активність часто не обмежується звичайними політиками доступу і не викликає підозри з боку систем виявлення загроз. У хмарному середовищі контроль над діями таких користувачів ускладнюється обмеженою видимістю, відсутністю повноцінного журналювання або недостатньою сегментацією доступу.

Розширені постійні загрози (APT) – це складні, тривалі атаки, спрямовані на непомітне проникнення до інфраструктури з метою збору даних або підготовки до подальших дій. У хмарному середовищі такі атаки часто використовують фішинг, викрадення облікових даних або вразливості в API. APT можуть залишатися невиявленими тривалий час через розподілену

архітектуру та обмежену видимість. Протидія вимагає комплексного захисту: поведінкового аналізу, моніторингу та ізоляції середовищ.

Вразливості API у хмарних сервісах відкривають можливості для несанкціонованого доступу, обходу автентифікації та маніпуляцій із даними. Через API відбувається керування більшістю хмарних ресурсів, тому будь-яка помилка в логіці, перевірці прав або фільтрації запитів може бути використана зловмисниками. Особливу небезпеку становлять публічні API з надто широкими дозволами або відсутністю обмежень швидкості. Ефективний захист вимагає суворої політики доступу, логування запитів, валідації введених даних і регулярного аналізу безпеки API [12].

1.3.2 Найкращі практики безпеки хмарного середовища

Безпека в хмарній інфраструктурі вимагає системного підходу, який враховує як технічні, так і організаційні аспекти. У зв’язку з високою динамікою хмарних середовищ та великою кількістю точок доступу, основним завданням стає не лише впровадження окремих механізмів захисту, а й реалізація цілісної стратегії управління ризиками. Найкращі практики, що склалися у провідних постачальників хмарних послуг і підтримуються міжнародними стандартами безпеки, дозволяють мінімізувати ймовірність порушення конфіденційності, цілісності та доступності даних.

Одним із ключових принципів є мінімізація прав доступу, це означає, що кожен користувач, сервіс або процес повинен мати лише ті привілеї, які необхідні для виконання його функцій. Надмірні або необмежені права становлять серйозну загрозу, оскільки у разі компрометації облікового запису зловмисник може отримати доступ до всієї інфраструктури. Для реалізації цього підходу впроваджуються централізовані системи керування ідентифікацією та доступом, зокрема моделі на основі ролей (RBAC) або атрибутів (ABAC).

Другим критично важливим елементом є багатофакторна автентифікація, що дозволяє значно знизити ризик несанкціонованого доступу навіть у випадку

компрометації пароля, оскільки вимагає додаткового фактора підтвердження особи, наприклад, тимчасового коду або біометричних даних. Ця практика повинна застосовуватись як для доступу до консолі керування, так і для облікових записів з адміністративними правами.

Ще одним обов'язковим компонентом є шифрування даних як під час передавання, так і в стані зберігання. Для цього використовуються криптографічні алгоритми з відповідним рівнем стійкості та централізоване управління ключами, наявність шифрування дозволяє зберігати конфіденційність інформації навіть у разі витоку або фізичного доступу до носіїв.

У межах комплексного підходу важливу роль відіграє безперервний моніторинг та логування подій, а саме усі дії користувачів, запити до ресурсів, зміни конфігурацій та спроби автентифікації повинні реєструватися, зберігатися та аналізуватися. Це дає змогу вчасно виявляти ознаки вторгнення або порушень політик безпеки. Автоматизовані системи моніторингу, зокрема рішення класу SIEM, дозволяють здійснювати кореляцію подій, виявляти аномалії та формувати сповіщення у реальному часі [13].

Крім того, однією з сучасних практик є впровадження моделі нульової довіри, ця концепція передбачає, що жоден користувач або пристрій не вважається безпечним за замовчуванням, навіть якщо він знаходиться всередині корпоративної мережі. Кожен запит до ресурсу має проходити повторну автентифікацію, авторизацію та перевірку політик, що дозволяє ефективно захищати хмарну інфраструктуру в умовах розподіленої архітектури та гібридного доступу.

Також до найкращих практик належить регулярний аудит налаштувань безпеки. Важливо не лише одноразово налаштувати захист, а й постійно перевіряти відповідність конфігурацій встановленим вимогам та стандартам, наприклад, CIS Benchmarks або NIST 800-53. Використання спеціалізованих інструментів аналізу дозволяє своєчасно виявляти відхилення від політик та усувати потенційні вразливості.

На завершення слід відзначити важливість організаційних заходів, таких як розробка політик безпеки, проведення навчань для персоналу, періодичне тестування на проникнення та реагування на інциденти. Тільки поєднання технічних рішень із правильною організаційною культурою дозволяє ефективно забезпечити захист хмарного середовища [14].

1.3.3 Реалізація принципів безпеки в провідних хмарних платформах

Провідні постачальники хмарних послуг активно впроваджують багаторівневі стратегії захисту, адаптовані до особливостей своїх інфраструктур. Вони не лише забезпечують базовий набір інструментів для контролю доступу, шифрування та моніторингу, а й дотримуються міжнародних стандартів безпеки та відповідності. Ці платформи послідовно реалізують принципи моделі спільної відповідальності, згідно з якою клієнт несе відповідальність за безпеку налаштувань, доступу та даних у межах власних ресурсів, тоді як провайдер гарантує безпеку фізичної інфраструктури, мережі та гіпервізора.

Amazon Web Services пропонує один із найповніших наборів засобів безпеки. Основу захисту становлять такі сервіси, як AWS Identity and Access Management – для контролю прав доступу, AWS Key Management Service – для управління шифруванням, AWS CloudTrail – для журналювання подій, а також Amazon GuardDuty і AWS Security Hub – для виявлення загроз і централізованого аналізу стану безпеки. AWS має численні сертифікації, зокрема ISO/IEC 27001, SOC 1, 2, 3, PCI DSS, що підтверджує відповідність сервісів загальновизнаним стандартам [15].

Microsoft Azure реалізує комплексну модель захисту на всіх рівнях, включаючи Azure Active Directory для керування доступом та ідентифікацією, Azure Key Vault для зберігання секретів, Azure Defender for Cloud для оцінки безпеки ресурсів і виявлення вразливостей. Особливістю Azure є глибока

інтеграція з корпоративними середовищами та потужні можливості централізованої політики безпеки. Крім того, Microsoft надає набір шаблонів для відповідності численним регуляторним стандартам, включаючи GDPR, NIST, HIPAA [16].

Google Cloud Platform також впроваджує принципи нульової довіри, забезпечує шифрування даних за замовчуванням та пропонує розвинуті інструменти безпеки, такі як Cloud IAM, Security Command Center, Cloud DLP API для виявлення конфіденційної інформації, а також Chronicle для глибокого аналізу загроз. GCP активно просуває концепцію Data Access Transparency, яка дозволяє клієнтам бачити, коли і ким були отримані доступи до їхніх даних. Також Google сертифікований відповідно до стандартів ISO 27001, SOC 2, FedRAMP тощо [17].

Загалом, усі провідні хмарні платформи реалізують фундаментальні принципи безпеки – розмежування доступу, шифрування, моніторинг, відповідність – та забезпечують користувачів засобами для впровадження індивідуальних політик безпеки. Водночас ефективність цих засобів напряму залежить від рівня їх використання самим клієнтом, що підкреслює важливість належної конфігурації та безперервного контролю з боку організації.

Висновки до розділу 1

У першому розділі було розглянуто еволюцію хмарних технологій та їх вплив на сучасну IT-інфраструктуру, встановлено, що перехід до моделі інфраструктура як сервіс забезпечив організаціям гнучкість, масштабованість і економічну доцільність, проте одночасно створив нові виклики у сфері безпеки. Проведено аналіз основних типів хмарних обчислень, а також акцентовано увагу на проблемах захисту даних у хмарному середовищі. Особливої актуальності набули питання безпеки доступу, контролю подій, управління конфігураціями та своєчасного реагування на інциденти. Також було окреслено сучасні загрози хмарним платформам – зокрема, неправильні налаштування,

витоки даних, атаки типу DDoS, зловживання API тощо. Таким чином, у розділі обґрунтовано необхідність впровадження автоматизованих систем моніторингу подій безпеки, що дозволяють своєчасно виявляти ризики та підвищувати рівень захищеності хмарної інфраструктури. Це і визначає напрям подальшого дослідження.

РОЗДІЛ 2
ОГЛЯД ІНСТРУМЕНТІВ МОНІТОРИНГУ ПОДІЙ БЕЗПЕКИ У ХМАРНОМУ СЕРЕДОВИЩІ AWS

2.1 Загальна характеристика платформи Amazon Web Services

Amazon Web Services – це одна з найпопулярніших та наймасштабніших хмарних платформ у світі, що надає широкий спектр послуг для обчислень, зберігання даних, мережевої інфраструктури, аналітики, штучного інтелекту та кібербезпеки. Платформа була запущена у 2006 році компанією Amazon і з того часу стала беззаперечним лідером хмарних обчислень, обслуговуючи мільйони клієнтів у понад 190 країнах, серед основних користувачів AWS – NASA, Netflix, Adobe, Samsung, General Electric, Zoom та багато інших. Платформа Amazon Web Services охоплює всі основні моделі хмарних обчислень, що робить її універсальним середовищем для розгортання різноманітних IT-рішень. У моделі інфраструктури як послуги (IaaS) AWS надає клієнтам можливість використовувати віртуальні сервери, сховища, мережеві ресурси та інші базові компоненти, зокрема сервіси Amazon EC2, Amazon S3 та Amazon VPC. У моделі платформи як послуги (PaaS) пропонуються готові середовища для розробки й розгортання програмних рішень без необхідності ручного керування інфраструктурою – прикладами таких сервісів є AWS Lambda, Elastic Beanstalk та Amazon RDS. Крім того, AWS включає і елементи програмного забезпечення як послуги (SaaS), зокрема такі сервіси, як AWS Security Hub, Amazon QuickSight та Amazon Chime, що надають користувачам готовий функціонал у вигляді завершених продуктів, доступних через інтернет.

AWS запровадила модель хмарних обчислень, у якій ресурси, а саме обчислення, сховище, мережа, надаються користувачам у міру потреби й оплачуються за фактом споживання. Інфраструктура платформи охоплює десятки дата-центрів по всьому світу, кожен із яких працює під цілодобовим

контролем і регулярним технічним обслуговуванням, це забезпечує стійкість до збоїв і дає змогу зберегти цілісність даних навіть у разі виходу з ладу окремого регіону. AWS забезпечує гнучке масштабування ресурсів, дозволяючи швидко адаптувати інфраструктуру під зміну навантаження без втрати продуктивності або стабільності. Такий підхід особливо актуальний для компаній, що динамічно розвиваються або працюють з піковими навантаженнями. Другою вагомою перевагою є широкий вибір сервісів – понад 200 повнофункціональних рішень для обчислень, зберігання даних, аналітики, машинного навчання, розробки програмного забезпечення, обробки відео, IoT тощо. Це дозволяє компаніям реалізовувати різні типи IT-проєктів у межах однієї платформи, без потреби у сторонніх інструментах або сервісах. Третім фактором є глобальна інфраструктура AWS, яка включає десятки регіонів і зон доступності по всьому світу, що гарантує високу доступність сервісів, мінімальні затримки доступу до ресурсів і можливість розгортання рішень ближче до кінцевого користувача [18].

У сфері безпеки AWS випереджає конкурентів завдяки ширшому переліку сертифікатів відповідності міжнародним стандартам (ISO 27001, SOC 2, NIST, PCI DSS тощо) і наявності власної моделі спільної відповідальності, яка чітко розмежовує зони контролю провайдера й клієнта. Amazon Web Services забезпечує безпеку за багаторівневою моделлю, яка охоплює фізичну інфраструктуру, мережі, шифрування, контроль доступу та моніторинг. Ключовим принципом є модель спільної відповідальності, де AWS відповідає за безпеку хмари, а користувач – за безпеку власних ресурсів у ній. Платформа надає інструменти для управління доступом (IAM), журналювання (CloudTrail), перевірки конфігурацій (Config) і виявлення загроз (GuardDuty) [19].

Серед провідних хмарних платформ, таких як Amazon Web Services, Microsoft Azure та Google Cloud Platform , саме AWS вирізняється найбільш зрілою та стабільною інфраструктурою, широким спектром сервісів і найрозвиненішими засобами автоматизації. На відміну від конкурентів, AWS першою запровадила модель масштабованого хмарного обчислення з оплатою

за використання і на сьогодні пропонує понад 200 повнофункціональних сервісів. Її глобальна інфраструктура охоплює найбільшу кількість регіонів і зон доступності, що дозволяє забезпечувати високу доступність сервісів у будь-якій точці світу. За даними Synergy Research Group, що зображено на рис. 2.1, станом на третій квартал 2022 року її частка склала 34%, що значно перевищує показники Microsoft Azure (21%) та Google Cloud (11%).

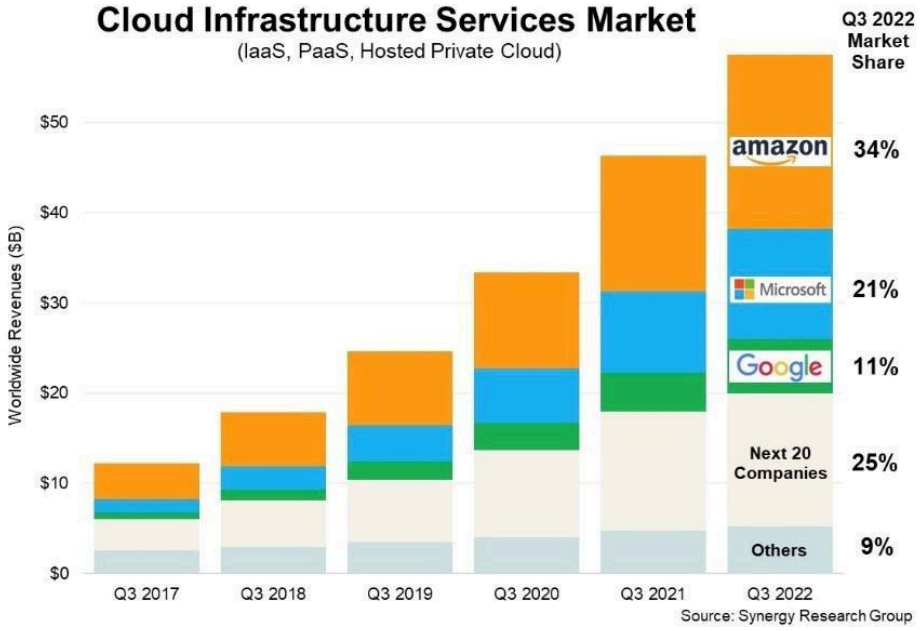

Рисунок 2.1 – Розподіл ринку хмарної інфраструктури за даними Synergy Research

Хоча Amazon Web Services є потужною та функціонально багатою платформою, вона має низку обмежень, які варто враховувати. Технічна підтримка у базовому пакеті є обмеженою, а повноцінна допомога доступна лише у межах платних підписок. Платформа має складну структуру тарифікації, де ціни залежать від багатьох змінних , а саме типів сервісів, регіону, обсягу трафіку та часу використання. Це створює ризик неконтрольованого зростання витрат у разі недостатнього моніторингу або неправильного налаштування ресурсів. Ще однією проблемою є дефіцит кваліфікованих фахівців з досвідом роботи в AWS, що може ускладнювати процес впровадження рішень або масштабування команди. Окрім того, платформа встановлює стандартні

обмеження на обсяг ресурсів, наприклад, кількість EC2-інстансів або обсяг запитів, які потрібно змінювати вручну через запити в підтримку, що додає адміністративного навантаження [20].

Незважаючи на вказані обмеження, AWS залишається найбільш комплексним і стабільним середовищем для розгортання хмарних рішень. Її гнучкість, масштабованість, наявність великої кількості вбудованих сервісів та підтримка міжнародних стандартів роблять платформу оптимальним вибором для реалізації систем автоматизованого моніторингу подій безпеки. Сукупність переваг значною мірою переважає над технічними й організаційними труднощами, які здебільшого можна подолати за рахунок грамотного проєктування архітектури та використання засобів автоматизації.

2.2 Критерії вибору інструментів для моніторингу безпеки

У хмарному середовищі основні загрози пов’язані з несанкціонованим доступом, неправильними конфігураціями, активністю з невідомих джерел та порушенням політик безпеки. Тому першочерговим завданням моніторингу є виявлення дій, які можуть свідчити про спроби порушення цілісності або конфіденційності системи. Інструменти, які застосовуються для цього, мають відповідати низці практичних вимог, одним із головних завдань було обрати такі рішення, які не потребують ручної інтеграції чи складних доопрацювань, а мають підтримку всіх ключових сервісів AWS та взаємодіють між собою через внутрішні механізми – події, логування, сповіщення.

Особливу увагу приділено можливості автоматичного виявлення змін у середовищі, таких як нові ресурси, зміни у правах доступу, підозріла активність, спроби обходу політик. У традиційних системах ці зміни часто фіксуються вручну або із затримкою, тоді як у хмарі необхідне постійне сканування стану інфраструктури, оцінка відповідності та запуск дій у відповідь. Саме тому серед пріоритетів були автоматизовані сервіси, що здатні не лише фіксувати порушення, але й самостійно ініціювати реакцію – надсилати

сповіщення, блокувати доступ, фіксувати інцидент у центральному репозиторії [21].

Ще одним важливим критерієм стало охоплення. Обрані інструменти мали покривати як події доступу, так і зміни конфігурацій, створення ресурсів, використання ключів, роботу з мережевими структурами. Моніторинг лише обмеженого набору подій не дозволяє побудувати повну картину стану безпеки, тому враховувались сервіси, які працюють на рівні облікового запису, мережі, API, логів та внутрішніх політик одночасно.

Не менш важливим було врахування відповідності галузевим стандартам, зокрема NIST і CIS. Інструменти мали надавати готові перевірки, аналітику щодо відхилень та інтеграцію з системами контролю відповідності. Також бралась до уваги прозорість вартості, можливість прогнозувати обсяг використання та уникати непередбачуваних витрат, що часто є проблемою у хмарних моделях.

Отже, вибір інструментів ґрунтувався не лише на функціональності, а й на здатності працювати безперервно, автоматично, вбудовано в екосистему AWS, з підтримкою стандартів та адекватною моделлю оплати. Такий підхід дозволив сформувати набір рішень, які відповідають сучасним вимогам до безпеки у хмарному середовищі [22].

2.3 Огляд вбудованих сервісів AWS для моніторингу подій безпеки

Ефективне забезпечення безпеки у хмарному середовищі вимагає постійного моніторингу подій, автоматичного виявлення аномалій, аналізу конфігурацій і оперативного реагування на інциденти. У цьому контексті Amazon Web Services пропонує комплекс вбудованих сервісів, які забезпечують повноцінний життєвий цикл управління подіями безпеки – від збору даних до автоматизованих дій. Ці сервіси тісно інтегровані між собою, охоплюють різні аспекти безпеки – від аналізу мережевого трафіку до перевірки відповідності ресурсів галузевим стандартам – і можуть бути легко масштабовані під потреби

як малих, так і великих організацій. AWS також забезпечує можливість централізації результатів виявлення подій і подальшої інтеграції з системами автоматизованого реагування або зовнішніми SIEM-рішеннями.

Amazon Web Services впроваджує низку спеціалізованих сервісів, які формують єдину екосистему для моніторингу подій безпеки у хмарному середовищі. Ці сервіси забезпечують безперервне спостереження за активністю, аналіз конфігурацій ресурсів, виявлення загроз, формування інцидентів і запуск автоматизованих дій у відповідь на потенційні ризики. Для цього AWS використовує такі сервіси як Amazon GuardDuty, AWS Config, AWS Security Hub, Amazon CloudWatch, AWS CloudTrail, AWS Lambda та VPC Flow Logs. Кожен із них виконує окрему роль у системі захисту, а їх комбінація дозволяє створити багаторівневу систему моніторингу та реагування на події безпеки в хмарній інфраструктурі [23].

2.3.1 AWS CloudTrail

AWS CloudTrail є базовим компонентом аудиту в інфраструктурі Amazon Web Services, призначеним для автоматичного протоколювання усіх дій, що виконуються користувачами, сервісами або зовнішніми системами в межах облікового запису AWS. Він фіксує звернення до API незалежно від того, чи вони здійснені через AWS Management Console, SDK, CLI або інші засоби, включаючи міжсервісну взаємодію. CloudTrail забезпечує детальну історію змін і подій у хмарному середовищі, що є необхідною умовою для реалізації системного моніторингу безпеки, забезпечення відповідності нормативним вимогам, розслідування інцидентів та впровадження механізмів автоматизованого реагування [24].

Після кожної взаємодії з сервісами CloudTrail створює структурований запис події у форматі JSON. Цей запис включає:

- ідентифікатор користувача або ролі (IAM principal)

- назву виконаної дії (API method)
- час запиту (timestamp)
- IP-адресу джерела
- регіон виконання
- сервіс і ресурс, до якого було звернення
- результат виконання (успішний чи з помилкою)

CloudTrail розрізняє два основні типи подій:

- Management events – дії, пов’язані з керуванням ресурсами: створення, видалення, зміна конфігурацій, оновлення IAM-політик, запуск інстансів, зміна параметрів мережі тощо. Вони журналюються за замовчуванням.
- Data events – детальні дії над ресурсами: звернення до об'єктів у S3, виконання Lambda-функцій, доступ до DynamoDB тощо. Журналювання таких подій вмикається окремо, оскільки вони мають вищу деталізацію і можуть створювати великі обсяги логів.

Використання AWS CloudTrail є доцільним та необхідним у контексті забезпечення контролю, прозорості та відповідності в хмарному середовищі. Адже CloudTrail забезпечує повну трасованість дій, що дозволяє відтворити хронологію змін і встановити джерело будь-якого впливу на ресурси. А також, сервіс є необхідною умовою для відповідності міжнародним стандартам безпеки – CIS AWS Foundations Benchmark, ISO/IEC 27001, NIST SP 800-53 та SOC 2 – у частині аудиту доступу та контролю змін. Крім того, CloudTrail інтегрується з іншими сервісами AWS, що дозволяє створювати комплексну систему виявлення й реагування на інциденти без використання сторонніх рішень.

AWS CloudTrail широко застосовується у практиці управління інформаційною безпекою для виявлення, розслідування та попередження інцидентів. Його журнали подій слугують надійною основою для аудиту дій користувачів і сервісів, а також для ретроспективного аналізу змін у хмарному середовищі. У корпоративних сценаріях CloudTrail використовується для

верифікації змін конфігурації, виявлення несанкціонованого доступу, вивчення аномальної активності та забезпечення відповідності вимогам міжнародних стандартів [25].

Для ефективного використання AWS CloudTrail у системах безпеки важливо дотримуватись перевірених практик, що підвищують спостережуваність, мінімізують ризики та сприяють відповідності нормативним вимогам. CloudTrail доцільно активувати в усіх підтримуваних регіонах AWS, це забезпечує виявлення дій, що можуть бути здійснені поза межами основного середовища, зокрема в цілях обходу безпекових механізмів. Журнали подій рекомендується зберігати в окремому, захищеному S3-бакеті з мінімально необхідними правами доступу, для гарантії їхньої цілісності слід увімкнути валідацію цифрових підписів і шифрування за допомогою AWS KMS. Варто також використовувати інтеграцію CloudTrail з CloudWatch Logs або EventBridge, що дозволяє реалізувати механізми оперативного реагування: надсилання сповіщень, блокування підозрілих дій або запуск автоматизованих сценаріїв через Lambda. Системне впровадження цих практик дозволяє використовувати CloudTrail як повноцінний компонент моніторингової та аналітичної інфраструктури, що відповідає сучасним вимогам до кібербезпеки в хмарному середовищі [26].

2.3.2 AWS CloudWatch

Amazon CloudWatch – це інтегрований сервіс моніторингу та спостережуваності, розроблений для забезпечення безперервного контролю за станом ресурсів, продуктивністю застосунків та поведінкою інфраструктури в межах платформи AWS. Його функціональність охоплює збір і обробку метрик, журналів, трасування подій, а також автоматичне сповіщення та реагування на задані умови. На відміну від традиційних систем моніторингу, CloudWatch функціонує нативно всередині екосистеми AWS та забезпечує мінімальні

затримки в зборі телеметрії, кожен підтримуваний сервіс AWS автоматично передає базові метрики до CloudWatch – зокрема, вбудовані метрики використання ресурсів і навантаження. Крім того, користувачі можуть передавати кастомні метрики, що дозволяє будувати моніторинг під специфіку конкретних застосунків[27].

Ключовим компонентом є CloudWatch Logs, який забезпечує централізоване збирання, зберігання та аналіз лог-файлів з сервісів CloudTrail, VPC Flow Logs, Route 53 та інших. Логи зберігаються в лог-групах із можливістю контролю життєвого циклу, доступу й архівації, що дозволяє виявляти події, які мають ознаки відхилень від очікуваної поведінки – наприклад, нестандартні запити, підозрілі з'єднання чи повторні спроби доступу до ресурсів.

CloudWatch надає можливість створення оповіщень, що спрацьовують при досягненні заданих порогів або статистичних аномаліях, наприклад виявлення нетипового обсягу мережевого трафіку або великої кількості невдалих спроб аутентифікації може ініціювати автоматизовану дію через AWS Lambda, надсилання сповіщення засобами Amazon SNS, або активацію сценарію реагування через AWS Systems Manager. Таким чином, CloudWatch стає не лише засобом моніторингу, а й важливою складовою інфраструктури виявлення й автоматичного реагування на інциденти.

CloudWatch дозволяє реалізувати реагування в реальному часі на події, зафіксовані в логах або метриках, його інтеграція з AWS CloudTrail, Security Hub та GuardDuty забезпечує наскрізний моніторинг інцидентів, а також формує основу для реалізації автоматизованих сценаріїв реагування. Таким чином, Amazon CloudWatch виступає не лише інструментом технічного моніторингу, а й ключовим елементом в архітектурі захисту хмарних середовищ, забезпечуючи комплексну спостережуваність, виявлення аномалій і реалізацію заходів оперативного реагування [28].

2.3.3 AWS Lambda

AWS Lambda – це безсерверна обчислювальна служба, яка дозволяє виконувати код у відповідь на події без потреби управління інфраструктурою, для цілей контролю безпеки в хмарному середовищі, Lambda виконує критичну функцію – автоматизовану реакцію на інциденти безпеки та обробку подій з інших сервісів AWS. Завдяки гнучкій моделі запуску, функції можуть ініціюватися подіями з CloudTrail, CloudWatch, Amazon S3, AWS Config, Security Hub, GuardDuty, EventBridge тощо.

Lambda дозволяє оперативно реалізовувати реакцію на загрози в режимі реального часу – без затримки, притаманної ручному втручанню [29]. Наприклад, при спрацьовуванні CloudWatch Alarm через підозрілу зміну мережевого трафіку або надмірне використання привілейованого API, Lambda може автоматично:

- Заблокувати доступ до ресурсу;
- Надіслати сповіщення в Amazon SNS, Slack, Microsoft Teams або SIEM-систему;
- Відкотити зміни конфігурації через AWS Config;
- Деактивувати IAM-користувача або ключ доступу;
- Записати інцидент у централізоване сховище або журнал подій для подальшого аналізу.

Або при спрацьовуванні служби Amazon GuardDuty, яка виявляє аномальну активність – зокрема спроби входу з незвичних геолокацій або дії, що свідчать про ескалацію привілеїв, – функція Lambda може автоматично втрутитися. Залежно від сценарію, вона змінює політику доступу до ресурсу, блокує джерело трафіку або створює запис про інцидент у зовнішній системі керування подіями. Інший поширений приклад – реагування на помилки конфігурації, як-от ненавмисне надання публічного доступу до S3-бакета. У такому випадку подія фіксується через AWS CloudTrail або AWS Config,

передається до EventBridge, після чого викликається Lambda-функція. Вона автоматично модифікує політику доступу до S3, фіксує порушення у логах CloudWatch і надсилає відповідне сповіщення адміністраторам безпеки [30].

AWS Lambda може бути ефективним інструментом для створення кастомних перевірок відповідності конфігурації інфраструктури, що дозволяє виявляти нетипові або небезпечні налаштування ресурсів і автоматично вживати заходів для їх усунення. Такі Lambda-функції можуть запускатися як періодично, так і подієво – при створенні, зміні або видаленні ресурсу. Це дозволяє впроваджувати гнучку політику відповідності, яка враховує унікальні вимоги середовища. AWS Lambda дозволяє реалізовувати критично важливі кастомні перевірки конфігурацій, зокрема:

- Виявлення публічного доступу до S3-бакетів – перевірка ACL і політик на наявність доступу.
- Аналіз правил у Security Groups – виявлення відкритих портів (SSH, RDP, HTTP) для 0.0.0.0/0.
- Контроль шифрування змінних середовища – перевірка, чи використовують Lambda-функції шифрування через AWS KMS.
- Перевірка IAM-політик – виявлення занадто широких дозволів.
- Оцінка шифрування даних – перевірка ввімкнення шифрування в S3, EBS та RDS.

Такі перевірки дозволяють автоматизувати контроль відповідності політикам безпеки та оперативно усувати ризики в хмарній інфраструктурі [31].

2.3.4 AWS GuardDuty

Amazon GuardDuty – це спеціалізований керований сервіс, розроблений для автоматизованого виявлення потенційних загроз у хмарній інфраструктурі AWS. Його основна функція полягає у безперервному моніторингу облікових записів, мережевого трафіку, системних логів і поведінкових шаблонів

користувачів для виявлення зловмисної або аномальної активності. Сервіс не вимагає попередньої конфігурації інфраструктури або ручної обробки логів, що забезпечує високу швидкість впровадження та масштабованість.

Для ідентифікації загроз GuardDuty обробляє дані з таких джерел:

- AWS CloudTrail – журнал викликів API, включаючи як керуючі, так і подієві дії;
- VPC Flow Logs – метадані про вхідний та вихідний мережевий трафік між ресурсами всередині віртуальної приватної хмари;
- DNS-запити – інформація про доменні звернення, що дозволяє виявити зв’язки з потенційно шкідливими доменами;

Сервіс застосовує комбінований підхід, що включає сигнатурний аналіз, машинне навчання та евристичні алгоритми, які дозволяють класифікувати та пріоритезувати знайдені загрози за категоріями. Кожна знахідка (finding) містить структуровану інформацію: тип загрози, ступінь критичності (low, medium, high), ідентифікатор ресурсу, часову мітку, а також рекомендації щодо реагування. GuardDuty підтримує інтеграцію з іншими сервісами AWS, зокрема AWS Security Hub, Amazon EventBridge, AWS Lambda та сторонніми системами типу SIEM, що дає змогу будувати комплексні сценарії автоматичного реагування. GuardDuty є повністю керованим рішенням, яке не потребує розгортання або підтримки інфраструктури безпеки, увімкнення сервісу можливе одним кліком, без необхідності ручного збору або агрегації логів. Оскільки GuardDuty використовує існуючі лог-джерела і працює асинхронно, він не впливає на продуктивність хмарної інфраструктури користувача. Також, сервіс автоматично масштабується під потреби будь-якого середовища – від одного акаунта до сотень, без потреби у ручному втручанні [32].

Використання Amazon GuardDuty у хмарному середовищі AWS дозволяє досягти значного підвищення рівня інформаційної безпеки за рахунок автоматизованого виявлення загроз та своєчасного інформування про інциденти. Завдяки безперервному моніторингу ключових джерел подій сервіс забезпечує оперативне виявлення як відомих типів атак, так і поведінкових

аномалій, що можуть свідчити про спроби компрометації облікових записів або використання ресурсів для зловмисної діяльності. Однією з головних переваг використання GuardDuty є суттєве скорочення середнього часу виявлення інцидентів, що досягається шляхом автоматичного аналізу подій у реальному часі. Це дозволяє оперативно виявляти такі загрози як несанкціонований доступ з IP-адрес, сканування портів, підозрілу мережеву активність або використання вкрадених облікових даних. Крім того, GuardDuty не лише ідентифікує загрозу, але й класифікує її за рівнем критичності, надаючи рекомендації щодо подальших дій, що істотно спрощує роботу фахівців з безпеки.

Інтеграція з такими сервісами як AWS Lambda, Amazon EventBridge та AWS Security Hub, дає змогу реалізовувати сценарії автоматичної відповіді на інциденти без участі людини. Крім того, завдяки підтримці централізованого моніторингу GuardDuty дає змогу створити уніфіковану архітектуру безпеки в межах усієї організації.

Розглянемо типовий сценарій використання Amazon GuardDuty у хмарній інфраструктурі AWS, припустимо, в обліковому записі AWS запущено EC2-інстанс, який зазвичай виконує функції веб-сервера та не ініціює зовнішніх з'єднань. У певний момент цей інстанс починає встановлювати мережеві з'єднання з численними IP-адресами в різних географічних регіонах, зокрема на нестандартних портах, що не відповідає його типовому профілю поведінки. Amazon GuardDuty, здійснюючи моніторинг VPC Flow Logs, виявляє, що інстанс встановлює з'єднання з IP-адресами, які внесено до списку відомих командно-контрольних серверів ботнет-мереж. На основі цього сервіс формує знахідку з високим рівнем критичності (High), що вказує на потенційне скомпрометування ресурсу. У структурі події GuardDuty зазначає ідентифікатор інстансу, час, тип загрози та рекомендації щодо реагування. Знахідка автоматично передається до Amazon EventBridge, де спрацьовує відповідне правило автоматизації [33]. Згідно з цим правилом викликається функція AWS Lambda, яка виконує наступні дії:

- змінює налаштування групи безпеки EC2-інстансу, блокуючи весь вихідний трафік;
- додає тег Compromised=true для позначення ресурсу як потенційно ураженого;
- надсилає сповіщення через Amazon SNS команді безпеки для подальшого розслідування.

Такий підхід суттєво скорочує час реагування на інцидент, обмежує зону ураження та підвищує загальний рівень захисту хмарного середовища.

Значущість Amazon GuardDuty як компонента стратегії хмарної безпеки підтверджується і практикою його використання у великих корпораціях. Зокрема, компанія Volkswagen впровадила GuardDuty у масштабах усієї організації, застосовуючи власну систему керування обліковими записами, яка забезпечувала централізований контроль та розмежування доступу між бізнес-підрозділами. На основі цього досвіду команда AWS реалізувала підтримку AWS Organizations у GuardDuty, що дало змогу централізовано активувати сервіс, агрегувати виявлені загрози з кількох облікових записів і забезпечити єдину точку контролю для аналізу інцидентів безпеки у великих хмарних середовищах [34].

2.3.5 AWS Config

AWS Config – це спеціалізований сервіс для централізованого відстеження стану ресурсів AWS, аудиту змін конфігурацій і автоматизованої перевірки на відповідність вимогам безпеки. Його ключова функціональність полягає у фіксації поточного стану об'єктів інфраструктури, записі всіх змін та верифікації того, чи відповідають ресурси визначеним політикам або стандартам. На відміну від сервісів, орієнтованих на виявлення загроз, AWS Config виконує структурний моніторинг – тобто контролює, як саме налаштовано ресурси, а не лише що з ними відбувається. Завдяки цьому

забезпечується контроль над конфігурацією мереж, облікових записів, політик доступу, шифрування даних, використання ключів KMS та інших критичних параметрів.

Сервіс веде історію змін конфігурацій, формуючи повний хронологічний ланцюг подій: хто, коли і як змінив параметри ресурсу. Це дозволяє виявляти невідповідності, відновлювати попередній стан інфраструктури та проводити технічний аудит у разі інциденту безпеки.

Центральне місце в AWS Config займає механізм Config Rules – набір правил, які перевіряють конфігурації ресурсів на відповідність ресурсів визначеним критеріям безпеки. Ці критерії формуються на основі міжнародних стандартів таких як CIS AWS Foundations Benchmark, NIST SP 800-53, ISO/IEC 27001, PCI DSS та внутрішніх політик організації. Його використання дозволяє забезпечити автоматизований аудит конфігураційної відповідності у хмарному середовищі та оперативно виявляти відхилення від очікуваного стану інфраструктури [35].

Кожне правило AWS Config визначає умову або набір умов, яким має відповідати певний ресурс або група ресурсів. У процесі оцінювання сервіс перевіряє поточний стан ресурсу та присвоює один із трьох статусів:

- COMPLIANT – ресурс відповідає вимогам;
- NON_COMPLIANT – виявлено порушення політики;
- NOT_APPLICABLE – правило не застосовується до даного типу ресурсу.

Залежно від способу створення, правила поділяються на два типи:

- Керовані правила – це стандартизовані шаблони, попередньо визначені AWS, які охоплюють типові вимоги безпеки: шифрування, доступність, конфігурації мережі, контроль ключів доступу тощо. Вони орієнтовані на відповідність нормативним стандартам, зокрема CIS AWS Foundations Benchmark, PCI DSS, HIPAA, NIST 800-53.
- Користувацькі правила – визначаються індивідуально користувачем і реалізуються за допомогою функцій AWS Lambda, що дозволяє створювати

правила довільної складності з урахуванням контексту, бізнес-логіки або специфіки середовища.

Припустимо, у середовищі організації створено правило s3-bucket-level-public-access-prohibited, яке перевіряє чи не надає жоден S3-бакет публічного доступу через політику або ACL. Після активації правила AWS Config автоматично сканує всі бакети в акаунті та визначає, які з них порушують політику. У випадку виявлення порушення, адміністратор отримує подію, яка може бути спрямована до AWS Security Hub, Amazon SNS або використовуватись для запуску AWS Lambda функції, що відкликає публічні дозволи. Таким чином, забезпечується негайна реакція на порушення стандартів безпеки.

Для забезпечення базового рівня відповідності з хмарної безпеки рекомендується впровадження мінімального набору таких правил, який забезпечує покриття найбільш поширених і потенційно вразливих конфігураційних параметрів [36]. У таблиці 2.1 наведено 6 рекомендованих AWS Config Rules, які вважаються best practices та є доцільними для впровадження у будь-якому середовищі AWS.

Таблиця 2.1

Рекомендований мінімальний набір керованих правил AWS Config для контролю конфігурацій хмарної інфраструктури

Номер правила	**Назва правила**	**Опис / Дія**
1	cloudtrail-enabled	Перевірка, чи увімкнено AWS CloudTrail для журналювання дій в акаунті
2	s3-bucket-level-public-acess-prohibited	Виявлення S3-бакетів із публічним доступом

Продовження табл. 2.1

3	s3-bucket-server-side-encryption-enabled	Контроль використання серверного шифрування для S3-бакетів
4	ec2-instance-no-public-ip	Заборона надання EC2-інстансам публічних IP-адрес
5	vpc-default-security-group-closed	Перевірка наявності відкритих правил у дефолтних security group у VPC
6	iam-root-access-key-check	Заборона наявності access-ключів у root-користувача

Таким чином, AWS Config виступає ключовим інструментом для забезпечення контролю конфігурацій, відповідності стандартам та відстеження змін у хмарному середовищі. Завдяки підтримці керованих правил, інтеграції з іншими сервісами AWS та масштабованості в межах AWS Organizations, сервіс дозволяє реалізувати безперервний аудит стану інфраструктури у режимі реального часу.

2.3.6 AWS Security Hub

AWS Security Hub є централізованим сервісом управління подіями безпеки у хмарній інфраструктурі AWS, який забезпечує агрегацію, нормалізацію та подальший аналіз інцидентів із різноманітних джерел. Його

ключове призначення полягає у створенні єдиного інформаційного простору, де всі виявлення, результати перевірок відповідності та повідомлення про загрози зводяться до уніфікованого формату, що дозволяє організації здійснювати узгоджене управління безпековими подіями, зменшуючи фрагментованість інформації між сервісами.

Після активації сервісу Security Hub автоматично інтегрується з іншими засобами виявлення загроз у межах AWS, такими як Amazon GuardDuty, AWS Config, IAM Access Analyzer, а також із рішеннями сторонніх вендорів, підключених через стандартизований API. Усі події конвертуються у формат AWS Security Finding Format, що дозволяє забезпечити однакову структуру та інтерпретацію даних незалежно від джерела. Кожна така подія містить уніфікований опис проблеми, ідентифікатор ресурсу, ступінь критичності, часову мітку, статус та рекомендації щодо усунення загрози [37].

Security Hub постійно виконує перевірку ресурсів на відповідність галузевим стандартам безпеки, використовуючи інтегровані стандарти перевірки, що включають:

- CIS AWS Foundations Benchmark – набір рекомендованих налаштувань безпеки для хмарних ресурсів AWS, розроблений Center for Internet Security. До основних положень належать вимоги до активації журналювання CloudTrail, обмеження публічного доступу до об’єктів у S3, використання багатофакторної автентифікації для root-користувача, моніторингу змін у ресурсах та інших критично важливих параметрів безпеки;
- AWS Foundational Security Best Practices – перелік перевірок, сформований на основі кращих практик, виявлених самою компанією AWS при аналізі типових загроз у хмарі. Ці перевірки охоплюють широкий спектр сервісів AWS, зокрема EC2, S3, IAM, Lambda, RDS, і дозволяють автоматично виявляти порушення ключових практик безпеки – наприклад, відсутність шифрування, неправильно налаштовані політики доступу, відкриті порти або неконтрольований обіг облікових даних;

- PCI DSS – специфікації безпеки для середовищ, які обробляють, зберігають або передають платіжні дані, відповідно до вимог Payment Card Industry Data Security Standard.

Крім аналізу подій, Security Hub регулярно виконує контроль ресурсів AWS на відповідність наборам вимог, таких як CIS AWS Foundations Benchmark або AWS Foundational Security Best Practices. Наприклад, якщо S3-бакет створено з відкритим доступом, а контрольна перевірка визначає таку конфігурацію як небезпечну, Security Hub створить відповідне повідомлення про невідповідність з рівнем критичності та детальним описом.

Реальний сценарій демонструє, як Security Hub може використовуватись для автоматизованого виявлення та реагування на інцидент безпеки. Наприклад, в одному з середовищ AWS інженер помилково створює S3-бакет із відкритим публічним доступом до об'єктів. Сервіс AWS Config, який відстежує конфігурації ресурсів у реальному часі, фіксує цю подію як порушення політики безпеки, далі ця інформація автоматично передається до Security Hub, який вже має активовані перевірки відповідності до CIS AWS Foundations та AWS Foundational Security Best Practices. У результаті створюється інцидент з високим пріоритетом, у якому вказано тип порушення, уражений ресурс, час виявлення та рекомендація щодо усунення – відключити публічний доступ. Після цього Security Hub через EventBridge запускає AWS Lambda-функцію, яка автоматично оновлює політику бакета, закриваючи доступ ззовні, та надсилає звіт адміністратору. Подібний сценарій дозволяє не лише виявити небезпечну конфігурацію, а й оперативно її нейтралізувати відповідно до внутрішніх вимог організації [38].

У результаті AWS Security Hub формує основу для побудови комплексної архітектури безпеки в хмарному середовищі, об’єднуючи моніторинг, перевірку відповідності та автоматизоване реагування в єдину систему. Завдяки інтеграції з іншими сервісами AWS, підтримці галузевих стандартів і можливості централізованого контролю на рівні організації, він значно підвищує

керованість, спостережуваність і оперативність реагування на інциденти безпеки у хмарній інфраструктурі.

2.4 Інтеграція з зовнішніми SIEM-системами

Із зростанням складності хмарних інфраструктур виникає об’єктивна потреба у централізованому та глибокому моніторингу подій безпеки, стандартні засоби, які пропонує AWS, CloudTrail, GuardDuty, Config чи Security Hub, дозволяють отримати базову інформацію про активність та потенційні загрози в середовищі, проте мають обмеження у гнучкості, глибині аналізу й міжсистемній кореляції. Це зумовлює необхідність інтеграції з зовнішніми SIEM-системами, які здатні об’єднувати події з різних джерел, надавати контекст, виявляти складні багатоступеневі атаки та автоматизувати реагування.

SIEM (Security Information and Event Management) – це потужний інструмент для обробки подій безпеки в реальному часі, що забезпечує збір, нормалізацію, зберігання, кореляцію та аналітику подій з усього IT-простору організації. У хмарному середовищі SIEM виконує роль центрального вузла моніторингу, об'єднуючи логіку з локальної інфраструктури, хмарних облікових записів, мережевих пристроїв, систем керування доступом та інше. Їхнє використання дозволяє вчасно виявити підозрілі дії, відстежити розвиток загроз і побудувати автоматизовані ланцюжки реагування. Основна причина, чому варто інтегрувати AWS із SIEM-платформою, – це централізація моніторингу. Події із сервісів хмарної платформи можуть надходити в одне місце, де їх можна не тільки переглядати, а й аналізувати в контексті загальної картини безпеки. Це дає змогу оперативно виявляти аномалії, складні атаки та зловмисну активність, яку неможливо побачити, аналізуючи окремі логи вручну [39].

Серед найпопулярніших SIEM-рішень на ринку можна виділити Splunk, IBM QRadar, Microsoft Sentinel, Elastic SIEM та LogRhythm. Усі ці системи підтримують інтеграцію з хмарними платформами, зокрема AWS, і надають

розширені можливості для візуалізації даних, створення алертів, кореляції подій і навіть автоматичного реагування. Найчастіше для побудови глибокої інтеграції з AWS використовується Splunk, завдяки готовим додаткам таких як Splunk Add-on for AWS, Splunk App for AWS і підтримці протоколу HEC, ця система дозволяє ефективно обробляти події з CloudTrail, GuardDuty, Config, VPC Flow Logs і Security Hub. Події надсилаються у структурованому форматі, обробляються через призначені пайплайни обробки, нормалізуються й використовуються для побудови дашбордів, створення алертів і сценаріїв реагування. У більшості випадків інтеграція передбачає використання AWS Lambda для трансформації подій і їх прямої відправки в Splunk через HTTPS.

Водночас інтеграція з SIEM має і низку викликів, одним із них є вартість, оскільки хмарні сервіси можуть генерувати великі обсяги логів. Зокрема, VPC Flow Logs або CloudTrail створюють значну кількість подій, і якщо не оптимізувати політики збору, це призводить до перевитрат на ліцензії. Інша проблема – це різнорідність форматів логів від різних сервісів, що ускладнює їх нормалізацію і кореляцію. І нарешті, важливо враховувати безпекові аспекти самої інтеграції – зокрема, автентифікацію, шифрування переданих даних, надійність каналів комунікації.

Таким чином, інтеграція хмарної платформи AWS з зовнішньою SIEM-системою є не лише доцільною, а й критично важливою для організацій, що прагнуть мати повну ситуаційну обізнаність про події безпеки. Це дозволяє значно підвищити рівень захищеності середовища за рахунок своєчасного виявлення інцидентів, кращого розуміння контексту подій і впровадження автоматизованих заходів реагування.

2.5 Використання Terraform для автоматизації моніторингу безпеки в AWS

У контексті побудови ефективної системи моніторингу подій безпеки в хмарному середовищі важливим завданням є гарантування стабільності та

контрольованості конфігурацій, особливо це актуально для складних і багаторівневих архітектур, де одночасно задіяно декілька сервісів AWS із численними параметрами та взаємозв'язками. Інструмент terraform – це засіб інфраструктури як коду, який дозволяє описувати всю інфраструктуру у вигляді декларативних конфігураційних файлів.

На відміну від звичайних інтерфейсів керування, де ресурси створюються вручну через консоль, terraform надає можливість автоматизованого розгортання ресурсів. У сфері безпеки це критично важливо, оскільки дає змогу уникати ручних помилок, забезпечити контроль змін і досягати відповідності вимогам стандартів.

Наприклад, неправильно вказаний IAM дозвіл або пропущене налаштування логування може створити серйозні вразливості. Завдяки terraform конфігурація таких сервісів, як AWS CloudTrail, GuardDuty, AWS Config або Security Hub, фіксується в коді, який проходить перевірку, версіонування та ревізію.

Основні переваги Terraform у контексті безпеки:

- Автоматизація розгортання безпекових сервісів. Завдяки Terraform можна програмно ініціалізувати такі сервіси, як CloudTrail, AWS Config, GuardDuty, Security Hub тощо, з наперед заданими параметрами, що відповідають політикам організації.
- Контроль змін і відстеження історії. Всі зміни в інфраструктурі фіксуються у конфігураційних файлах, що дозволяє вести аудит, аналізувати внесені зміни та забезпечувати прозорість налаштувань.
- Зменшення людського фактору. Один раз налаштовані модулі можна багаторазово застосовувати для різних середовищ, наприклад, dev/test/prod, що мінімізує ризики помилок при ручному налаштуванні.
- Сумісність із політиками відповідності. Важливі компоненти, наприклад, логування дій користувачів або контроль публічного доступу до S3-бакетів можуть бути налаштовані автоматично згідно з вимогами стандартів.

Terraform підтримує модульність, що дозволяє розділяти інфраструктуру на логічні блоки, а саме модулі та перевикористовувати їх у різних частинах архітектури. Це особливо корисно при побудові багаторазових шаблонів для безпечного розгортання моніторингових сервісів, зокрема з логуванням, аудитом, алертингом та контролем відповідності [40].

Крім того, terraform забезпечує механізм plan та apply, який дозволяє користувачам попередньо переглянути всі зміни, що будуть внесені, перш ніж вони фактично застосуються. Такий підхід значно підвищує надійність впровадження змін в інфраструктурі та сприяє зниженню ризиків у безпековому контексті.

Отже, terraform є потужним засобом управління хмарною інфраструктурою, який не лише спрощує процеси розгортання, але й підвищує загальний рівень безпеки за рахунок стандартизації, автоматизації та прозорості всіх дій в AWS-середовищі.

Висновки до розділу 2

У другому розділі було детально проаналізовано ключові сервіси AWS, що використовуються для моніторингу подій безпеки та виявлення загроз у хмарному середовищі. Розглянуто принципи роботи таких інструментів, як Amazon CloudTrail, GuardDuty, Security Hub, AWS Config, а також можливості CloudWatch для збору логів і створення оповіщень. Показано, що ці сервіси забезпечують як базовий рівень видимості активності, так і поглиблений аналіз інцидентів у режимі реального часу.

Також досліджено інтеграції з зовнішніми SIEM-системами, зокрема Splunk, що дозволяє централізувати події безпеки з різних джерел, корелювати їх між собою, виявляти складні атаки та формувати гнучкі сценарії реагування.

Окрему увагу приділено інфраструктурному інструменту terraform, який дає змогу автоматизовано розгортати та масштабувати сервіси моніторингу в

AWS, забезпечуючи гнучкість, повторюваність і контроль над середовищем. Такий підхід дозволяє ефективно керувати компонентами безпеки в рамках принципів інфраструктури як коду.

Таким чином, поєднання вбудованих сервісів AWS, інструментів автоматизації та зовнішніх систем аналітики створює надійну основу для побудови адаптивної та масштабованої системи моніторингу подій безпеки у хмарному середовищі.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ ПОДІЙ БЕЗПЕКИ В AWS

3.1 Архітектура програмної реалізації системи моніторингу

Розроблена система автоматизованого моніторингу подій безпеки призначена для виявлення потенційних інцидентів, фіксації критичних подій, аналізу конфігурацій хмарних ресурсів і передачі повідомлень відповідальним особам або зовнішнім системам реагування. Її основна мета забезпечити постійне спостереження за станом хмарної інфраструктури, оперативно сповіщати про підозрілі або невідповідні дії, а також створити централізовану точку контролю подій безпеки. Для цього вона фіксує зміни у хмарному середовищі, наприклад створення ресурсів, зміну конфігурацій доступу, відкриття публічних доступів або порушення політик відповідності. У разі виявлення події, яка потенційно загрожує безпеці, система автоматично формує повідомлення та передає його через встановлені канали, наприклад, електронною поштою або до зовнішньої SIEM-системи. Уся логіка реагування є подієвою та безперервною, що дозволяє забезпечити базовий рівень захисту в режимі реального часу.

Система побудована за модульним принципом, що дозволяє логічно та функціонально розділити її на незалежні компоненти. Вона сформована на основі вбудованих сервісів AWS і власних кастомних перевірок, які розгортаються за допомогою коду та взаємодіють між собою, утворюючи цілісну, узгоджену систему моніторингу. Кожен компонент виконує окрему роль у межах загального процесу моніторингу, це є реєстрація подій, зберігання та обробка інформації, перевірка конфігурацій, генерація сповіщень або агрегація інцидентів. Усі ці компоненти реалізовано за допомогою використання методу інфраструктура як коду, написаного на terraform, відповідно до найкращих

практик, зокрема, через використання окремих модулів для кожної функціональної частини.

Ключова ідея реалізації полягає у багаторівневій побудові, яка полягає в тому, що користувач або адміністратор має змогу обрати один із трьох рівнів моніторингу залежно від потреб – базовий, стандартний або розширений. Вибір рівня визначається як масштабом інфраструктури, яку потрібно контролювати, так і фінансовими можливостями, оскільки всі складові системи, такі як журнали подій, зберігання логів, безсерверні обчислення, аналітичні механізми, реалізовано за допомогою платних сервісів AWS. Наприклад, постійний моніторинг мережевого трафіку чи агрегація подій у централізованому сховищі потребує значно більше ресурсів, ніж базовий аудит активності. Кожен рівень містить відповідний обсяг логіки, обробки та аналітики:

- Базовий рівень забезпечує мінімальний рівень збору подій, орієнтований на загальнодоступний аудит і логування.
- Стандартний рівень передбачає розширене спостереження за конфігурацією хмарних ресурсів, контроль відповідності політикам безпеки та виявлення аномальних активностей.
- Розширений рівень додає глибоку мережеву аналітику, централізовану агрегацію результатів з усіх компонентів та можливість інтеграції з зовнішніми системами, зокрема SIEM.

Для зручності розгортання реалізовано механізм динамічного вибору рівня, що дозволяє не розгортати всю систему повністю, а запускати лише необхідні її частини. Такий підхід також дає змогу швидко масштабувати систему, наприклад перейти з базового рівня до розширеного без необхідності змінювати базову логіку чи переписувати конфігурації.

Архітектура реалізованої системи моніторингу побудована як повністю автоматизована платформа, що розгортається у хмарному середовищі AWS на основі коду. Кожен рівень моніторингу має власну директорію, а всередині неї набір підмодулів, які відповідають за окремі функціональні блоки (див. Додаток А). Структура дозволяє незалежно запускати, оновлювати або видаляти

будь-який компонент без впливу на інші частини системи. У центрі архітектури – python-скрипт deploy.py, який виконує роль стартового інтерфейсу для користувача (див. Додаток Б). Після запуску скрипт запитує в користувача бажаний рівень моніторингу – базовий, стандартний або розширений. Обраний рівень фіксується у вигляді змінної level у спеціальному файлі terraform.tfvars.json, що генерується скриптом, після цього той самий скрипт автоматично ініціює послідовність команд для запуску terraform, який, у свою чергу, розгортає лише ті ресурси, що відповідають вибраному рівню.

Проєкт поділено на три основні каталоги – basic/, standard/ і advanced/, що показано на рис. 3.1, кожен із яких містить підкаталог services/. У services/ розміщені окремі модулі, які відповідають за створення конкретних сервісів AWS. Наприклад, у рівні basic до services/ входять модулі cloudtrail/, cloudwatch/, lambda-checkers/, sns/, які реалізують базову фіксацію дій, логування та обробку подій, рівень standard включає aws-config/ та guardduty/, alerting/, а в advanced реалізовано vpc-flowlogs/, securityhub/, alerting/.

```
PS C:\Users\Христина\Desktop\Диплом\cloud-security-monitoring\terraform> tree  /a | findstr /v /i ".terraform"
Folder PATH listing
Volume serial number is 5003-2FDA
C:.
|   +---modules
|   \---providers
|           \---hashicorp
|               +---aws
|               |   \---5.97.0
|               |       \---windows_amd64
|               \---random
|                   \---3.7.2
|                       \---windows_amd64
\---modules
    +---advanced
    |   \---services
    |       +---alerting
    |       |   \---lambda
    |       +---securityhub
    |       \---vpc-flowlogs
    +---basic
    |   \---services
    |       +---cloudtrail
    |       +---cloudwatch
    |       +---lambda_checkers
    |       |   \---lambda
    |       \---sns
    \---standard
        \---services
            +---alerting
            +---aws-config
            \---guardduty
PS C:\Users\Христина\Desktop\Диплом\cloud-security-monitoring\terraform>
```

Рисунок 3.1 – Архітектурна структура terraform-проєкту системи моніторингу

Архітектура розроблена таким чином, щоб повністю автоматизувати процеси розгортання, зменшити вплив людського фактора, забезпечити

стабільність інфраструктури в різних AWS-регіонах і облікових записах, а також надати можливість швидкого розширення функціональності за рахунок незалежних модулів.

3.1.1 Базовий рівень моніторингу

Базовий рівень моніторингу передбачає впровадження мінімального, але функціонально достатнього набору інструментів для фіксації й реагування на критичні події безпеки в хмарній інфраструктурі AWS, архітектура якого зображена на рис. 3.2.

```
PS C:\Users\Христина\Desktop\Диплом\cloud-security-monitoring\terraform\modules\basic> tree /f /a
Folder PATH listing
Volume serial number is 5003-2FDA
C:.
|   main.tf
|   outputs.tf
|   variables.tf
|
\---services
    +---cloudtrail
    |       main.tf
    |       outputs.tf
    |       variables.tf
    |
    +---cloudwatch
    |       main.tf
    |       outputs.tf
    |       variables.tf
    |
    +---lambda_checkers
    |   |   main.tf
    |   |   output.tf
    |   |   variables.tf
    |   |
    |   \---lambda
    |           mfa_check.py
    |           mfa_check.zip
    |           s3_check.py
    |           s3_check.zip
    |           sg_check.py
    |           sg_check.zip
    |
    \---sns
            main.tf
            outputs.tf
            variables.tf
```

Рисунок 3.2 – Архітектурна структура базового рівня системи моніторингу

Цей рівень орієнтований на забезпечення базового аудиту, а також виявлення найпоширеніших помилок конфігурації або зловмисних дій, не потребуючи при цьому значних обчислювальних ресурсів чи фінансових витрат. На цьому рівні задіяні наступні ключові компоненти:

- AWS CloudTrail – базовий механізм журналювання, який фіксує усі API-запити до ресурсів AWS. У рамках реалізації CloudTrail налаштовується так, щоб журнали подій зберігалися у визначений S3-бакет і паралельно дублювалися в лог-групу Amazon CloudWatch Logs.
- Amazon CloudWatch – обробляє потік логів від CloudTrail, де створюються фільтри метрик, які виділяють критичні події в даній роботі було реалізовано DeleteTrail, StopLogging, PutUserPolicy, CreateAccessKey тощо. Ці фільтри прив'язані до CloudWatch Alarm, що спрацьовують за заданими умовами, а саме при вході під root-обліковим записом, несанкціонованому доступі до API, невдалих спробах входу в консоль, зупинці логування CloudTrail або зміні політик доступу до S3.
- AWS Lambda Checkers – окремі функції, які виконуються періодично та здійснюють цільові перевірки конфігурацій:

mfa_check.py – дозволяє виявляти облікові записи без двофакторної автентифікації;

s3_check.py – виявлення S3-бакетів із публічним доступом;

sg_check.py – для перевірки відкритих Security Groups (SG). Вона виявляє групи безпеки, які дозволяють трафік з будь-якої IP-адреси 0.0.0.0/0, тобто публічно відкриті порти, що створює загрозу безпеці.

Всі Lambda-функції працюють автономно, конфігуруються через Terraform, запускаються за допомогою EventBridge, а результати передають назад у CloudWatch Logs.

- Amazon SNS – централізований механізм оповіщення, який дозволяє передавати повідомлення від CloudWatch Alarm або інших джерел до заздалегідь визначених отримувачів. У типовій конфігурації повідомлення надсилаються на електронну пошту адміністратора, проте користувач може змінити тип підписки, обравши інші підтримувані протоколи – зокрема SMS для мобільних повідомлень, HTTPS для інтеграції з вебсервісами.

Загальна логіка роботи базового рівня моніторингу працює наступним чином, користувач або процес виконує дію, сервіс CloudTrail реєструє цю дію як

подію і передає в CloudWatch Logs. Якщо подія відповідає заданим умовам одного з лог-фільтрів, вона перетворюється на метрику, яка відслідковується відповідною CloudWatch Alarm і у разі перевищення порогових значень тривога спрацьовує й ініціює сповіщення через SNS, забезпечуючи оперативне інформування відповідальних осіб. Наприклад, при здійсненні трьох послідовних невдалих спроб входу до AWS Management Console, кожна з яких супроводжується помилковим введенням облікових даних, як показано на рис. 3.3, події такого типу фіксуються як ConsoleLogin, сервіс CloudTrail передає ці записи до CloudWatch Logs, де спрацьовує налаштований лог-фільтр.

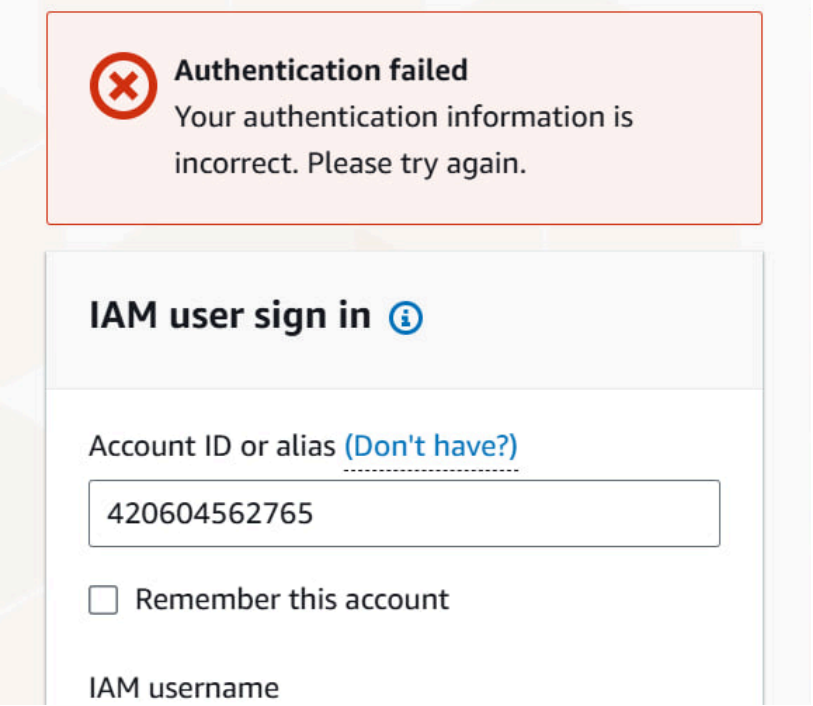

Рисунок 3.3 – Приклад невдалого входу до облікового запису AWS

Він ідентифікує шаблон події та перетворює його на числову метрику. У разі перевищення порогового значення, наприклад 3 спроб за 5 хвилин, активується CloudWatch Alarm, як продемонстровано на рисунку 3.4.

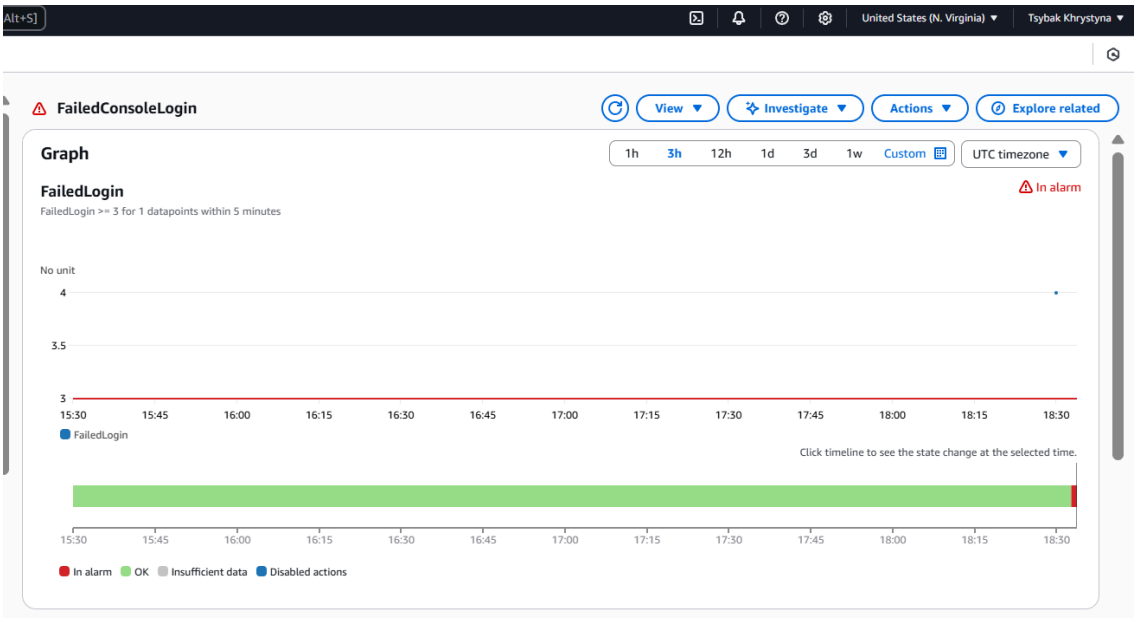

Рисунок 3.4 –CloudWatch Alarm при перевищенні порогу невдалих входів

У відповідь на це система автоматично ініціює сповіщення, яке показано на рис. 3.5 через сервіс Amazon SNS, надсилаючи повідомлення відповідальному спеціалісту.

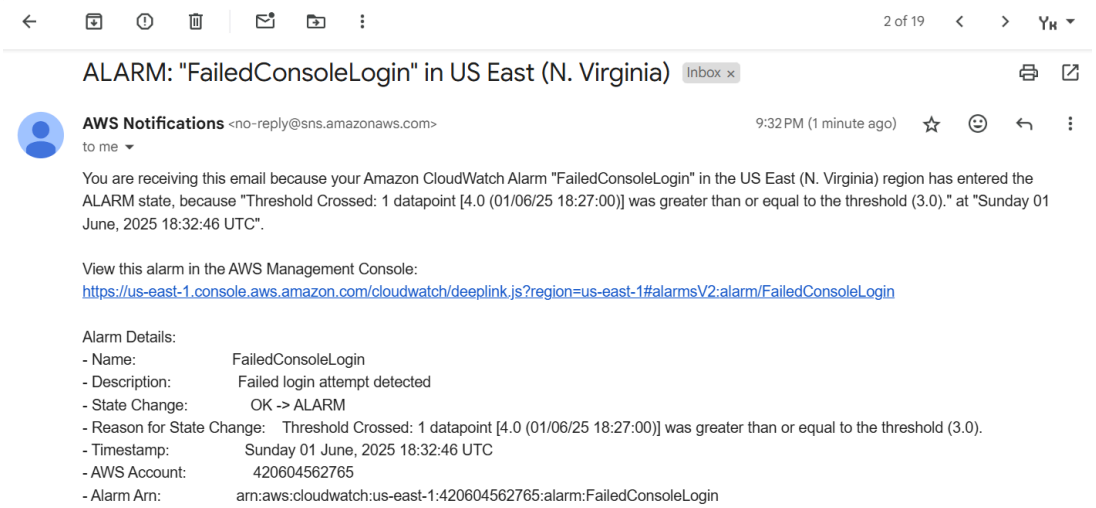

Рисунок 3.5 – Email-сповіщення про спрацювання CloudWatch Alarm при невдалих спробах входу

У листі міститься інформація про час спрацювання, джерельну IP-адресу, тип події та її критичність. Завдяки такій реакції потенційна спроба

несанкціонованого доступу не залишається непоміченою, що дозволяє оперативно вжити заходів захисту.

3.1.2 Стандартний рівень моніторингу

На відміну від базового рівня, стандартний рівень моніторингу безпеки в хмарному середовищі AWS передбачає розширення можливостей виявлення та реагування на інциденти завдяки використанню спеціалізованих сервісів для глибшого аналізу стану інфраструктури, до наявних компонентів додаються два важливі сервіси, а саме AWS Config та Amazon GuardDuty. Архітектура цього рівня зображена на рисунку 3.6.

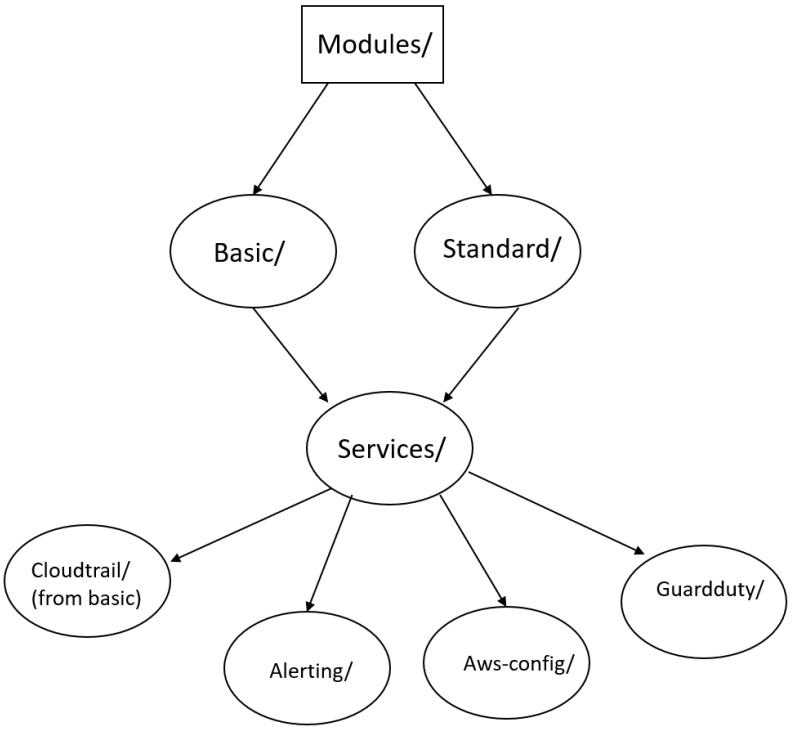

Рисунок 3.6 – Архітектурна структура стандартного рівня системи моніторингу

Основу моніторингу на цьому рівні становить сервіс AWS Config, який фіксує поточний стан ресурсів в обліковому записі та автоматично оцінює їх на відповідність заздалегідь визначеним правилам безпеки. У межах даної

реалізації застосовуються готові правила, рекомендовані Amazon як найбільш важливі з точки зору безпеки. Це, зокрема, перевірки на відсутність публічного доступу до S3-бакетів, обов’язкове шифрування даних, контроль за використанням ключів, а також наявність двофакторної автентифікації для користувачів IAM. Завдяки використанню таких стандартів, система здатна виявляти типові помилки конфігурації, що найчастіше призводять до інцидентів, і своєчасно повідомляти про них.

Паралельно з цим, Amazon GuardDuty здійснює постійний аналіз потоків мережевого трафіку, запитів до DNS, а також журналів CloudTrail, використовуючи сигнатурні методи, евристику та машинне навчання, таким чином сервіс здатен виявити ознаки компрометації, спроби сканування інфраструктури, підозрілі з’єднання з IP-адресами з відомих ботнет-мереж.

Події, які надходять із GuardDuty або AWS Config, передаються у сервіс Amazon EventBridge, де обробляються за попередньо визначеними шаблонами. Для кожного джерела подій створено відповідне правило, яке реагує, наприклад, на зміну стану ресурсу до "NON_COMPLIANT" або на виявлення загрози із рівнем небезпеки середнім і вище. Після цього подія автоматично надсилається у SNS-топік сповіщень.

Завдяки такій архітектурі, моніторинг на стандартному рівні переходить від пасивного спостереження до активного контролю за безпекою. Іншими словами, система не лише фіксує те, що відбувається, а й самостійно аналізує, чи відповідає поточний стан інфраструктури вимогам безпеки, а також виявляє дії, які потенційно є шкідливими. Стандартний рівень має беззаперечні переваги над базовим, зокрема – глибший аналіз, оцінка відповідності та автоматичне виявлення загроз. Проте варто враховувати і потенційні недоліки: зокрема, використання додаткових сервісів тягне за собою збільшення вартості моніторингу, а ефективність деяких механізмів потребує правильного налаштування політик та джерел логів. Незважаючи на це, стандартний рівень є обґрунтованим вибором для середовищ з підвищеними вимогами до інформаційної безпеки.

3.1.3 Розширений рівень моніторингу

Розширений рівень моніторингу передбачає комплексне охоплення усіх ключових аспектів безпеки хмарного середовища, включаючи контроль над користувацькими діями, конфігураціями ресурсів, поведінкою мережевого трафіку та централізованою обробкою подій безпеки. Основна мета цього рівня – забезпечити повноцінну видимість усіх безпекових процесів та автоматизувати реагування на виявлені інциденти.

У реалізації цього рівня застосовується комбінація як раніше використаних сервісів, таких як Amazon CloudTrail, AWS Config і Amazon GuardDuty, так і додаткових інструментів, які значно розширюють аналітичні та операційні можливості системи . Архітектуру цього рівня показано на рисунку 3.7.

Зокрема, інтеграція Amazon VPC Flow Logs дозволяє отримувати детальну інформацію про мережеву активність у межах віртуальної приватної хмари, цей сервіс фіксує дані про дозволені та заблоковані з’єднання між ресурсами, що дозволяє виявляти нетипову мережеву поведінку, зокрема спроби сканування портів, несанкціонований доступ або сплески активності.

Ключовим елементом архітектури виступає AWS Security Hub, який агрегує події безпеки з різних джерел, у тому числі з GuardDuty, AWS Config, CloudTrail, а також сторонніх рішень. Цей сервіс не тільки збирає та нормалізує Findings, але й дозволяє централізовано керувати ними, запускати автоматизовану обробку через події EventBridge та інтегрувати в робочі процеси безпеки організації. У контексті автоматизації важливу роль відіграє зв’язка з Amazon EventBridge, AWS Lambda та Amazon SNS. Security Hub створює подію, яка через EventBridge активує Lambda-функцію для фільтрації або трансформації даних, її логіка передбачає обробку лише тих findings, що мають високий або критичний рівень серйозності. Таким чином, незначні або

інформаційні сповіщення не потрапляють до каналу реагування, що дозволяє уникнути надмірного інформаційного навантаження та зосередитися на інцидентах, які дійсно потребують втручання. Далі результат передається в SNS, який відповідає за надсилання сповіщень користувачеві, в даній реалізації – пошті, або сторонній SIEM-системі.

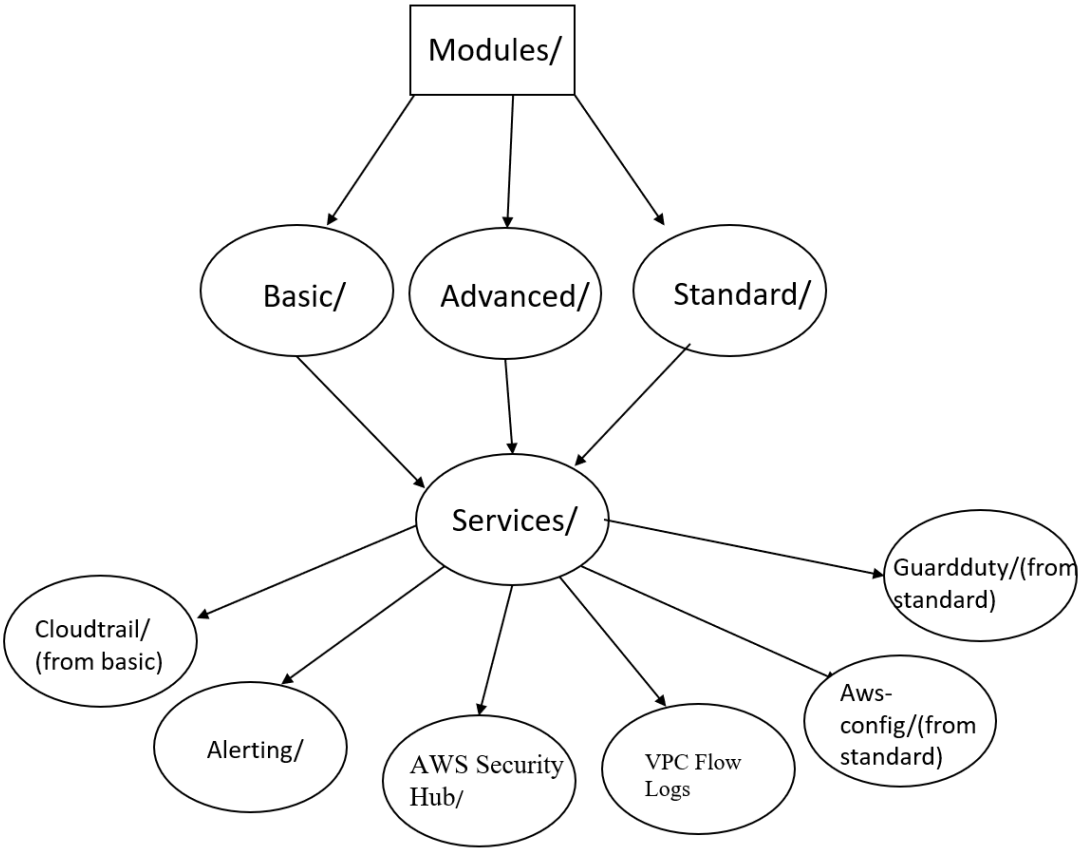

Рисунок 3.7 – Архітектурна структура розширеного рівня системи моніторингу

Порівняно з попередніми рівнями, розширений підхід забезпечує найбільш повний та глибокий моніторинг, він охоплює не лише контроль за діями користувачів і перевірку відповідності ресурсів політикам безпеки, а й аналіз поведінки мережі, виявлення комплексних загроз і централізовану реакцію на інциденти. Недоліком є відносна вартість конфігурації та підвищені вимоги до підтримки, однак ці фактори є виправданими для середовищ із критично важливими ресурсами або високими вимогами до інформаційної безпеки. Розширений рівень особливо доцільний для використання у великих

організаціях, фінансовому секторі чи в середовищах з обробкою чутливих персональних даних.

3.2 Аналіз працездатності та оцінка ефективності системи моніторингу

Для забезпечення гнучкості у налаштуванні та керованого запуску системи з вибором рівня моніторингу була реалізована автоматизована процедура розгортання, що виконується за допомогою скрипта deploy.py, виконання якого зображено на рис. 3.8.

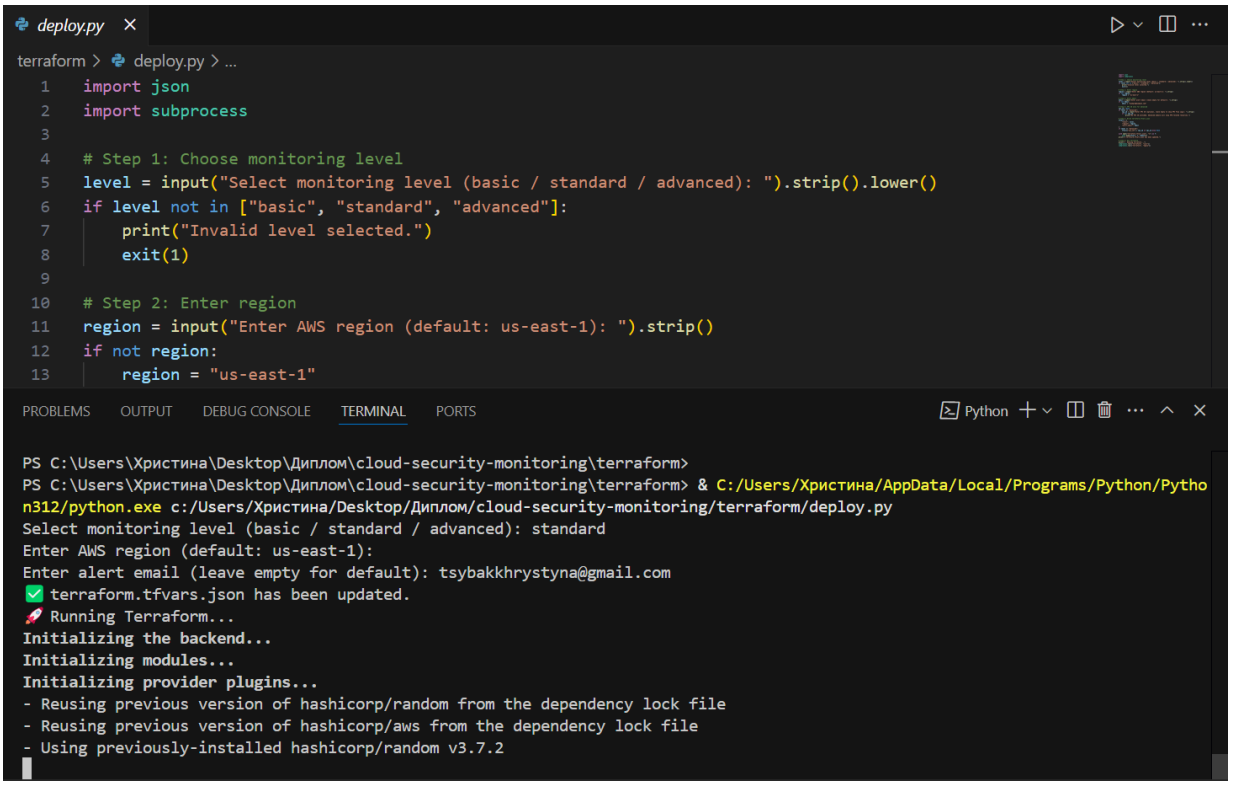

Рисунок 3.8 – Виконання скрипту deploy.py

Цей скрипт забезпечує взаємодію з користувачем у консольному режимі та дозволяє визначити конфігураційні параметри перед запуском основного інфраструктурного коду. Користувач, отримавши проект у вигляді структури terraform з відповідними модулями та python-скриптом, запускає файл deploy.py,

де відбувається покрокове введення параметрів. Далі користувач має можливість обрати один з трьох рівнів, таких як basic, standard або advanced, залежно від необхідної глибини аналізу подій безпеки, що визначає сервіси, які будуть активовані.

За замовчуванням для регіону AWS використовується значення us-east-1, однак користувач має змогу змінити його відповідно до потреб тестування. У разі надсилання повідомлень через сервіс SNS користувач вводить email, на який мають надходити сповіщення про інциденти безпеки, за відсутності введення використовується адреса за замовчуванням. Введення VPC ID – актуальне лише для advanced рівня, якщо користувач хоче активувати моніторинг трафіку в рамках VPC, необхідно вказати його ідентифікатор. Якщо поле лишається порожнім -відповідні ресурси не створюються. На основі введених параметрів скрипт генерує файл terraform.tfvars.json, який виступає джерелом змінних для terraform. Після цього автоматично виконується ініціалізація конфігурації через команду terraform init та її застосування – terraform apply, що забезпечує повне розгортання обраного рівня системи моніторингу.

Особливість полягає в тому, що всі інші ресурси, неактуальні для вибраного рівня, автоматично видаляються під час terraform apply, якщо були створені раніше. Це забезпечує гнучкість, контроль над витратами та чітке логічне зонування інфраструктури, наприклад, при переході з розширеного рівня на базовий, система автоматично видалить усі непотрібні ресурси, включаючи VPC Flow Logs чи інтеграцію з Security Hub, що виключає людський фактор і знижує ризик помилок конфігурації.

У ході тестування системи використовується розширений рівень моніторингу як основний сценарій перевірки функціональності. Це зумовлено тим, що даний рівень включає всі сервіси, реалізовані у проєкті, зокрема як власні компоненти, а саме vpc-flowlogs, securityhub, alerting, так і модулі, успадковані з попередніх рівнів cloudtrail із basic, guardduty та aws-config із standard. Таким чином, це дозволить протестувати всю архітектуру системи в

повному обсязі – від збору подій до генерації сповіщень. Крім того, розширений рівень реалізує найбільш комплексну логіку, зокрема включає обробку findings у Security Hub, маршрутинг подій через EventBridge, використання Lambda-функції для фільтрації загроз та доставку повідомлень через SNS.

Завдяки цьому, під час одного циклу розгортання можна перевірити:

- чи коректно фіксуються дії користувача через CloudTrail;
- чи спрацьовують політики відповідності AWS Config;
- як виявляються загрози GuardDuty;
- чи централізується інформація Security Hub;
- чи коректно запускаються правила EventBridge та функції Lambda;
- чи надходять сповіщення у вигляді email або до SIEM.

Для тестування було використано окремий AWS акаунт, у якому розгорталась ізольована інфраструктура. Таке середовище дозволило змоделювати типові порушення безпеки та перевірити, як система на них реагує.

3.2.1 Функціональна перевірка системи в умовах інцидентів

З метою перевірки ефективності роботи системи моніторингу в умовах реальних загроз, у тестовому середовищі було змодельовано серію інцидентів безпеки. Кожен із них відповідав поширеному типу помилок або порушень, які можуть виникати в хмарному середовищі. Одним із перших інцидентів, змодельованих для перевірки працездатності системи, стало створення S3-бакета з відкритим публічним доступом, політика доступу S3-бакета показана на рис. 3.9.

За допомогою Terraform було створено S3-бакет із прикріпленою політикою, що дозволяла загальнодоступне зчитування, а саме s3:GetObject для будь-якого користувача через конструкцію Principal: "*". Такий підхід є

серйозним порушенням рекомендацій AWS щодо безпеки, оскільки дозволяє будь-кому з Інтернету отримати доступ до вмісту бакета.

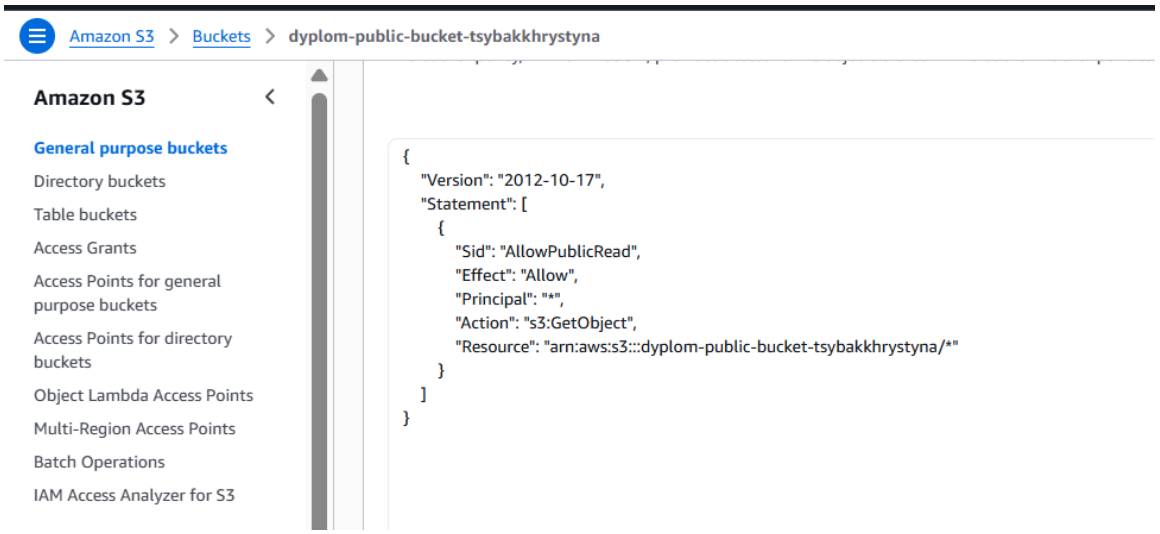

Рисунок 3.9 – s3-бакет “dyplom-public-bucket-tsybakkhrystyna”

Після застосування конфігурації Terraform, AWS Config автоматично зафіксував порушення політики, активувавши вбудоване правило s3-bucket-public-access-prohibited-advanced. Цей сервіс постійно відслідковує зміни конфігурацій ресурсів і порівнює їх із заздалегідь визначеними правилами відповідності і як тільки Config виявив, що бакет порушує вимоги до доступу, була створена відповідна подія, яка зображена на рис. 3.10.

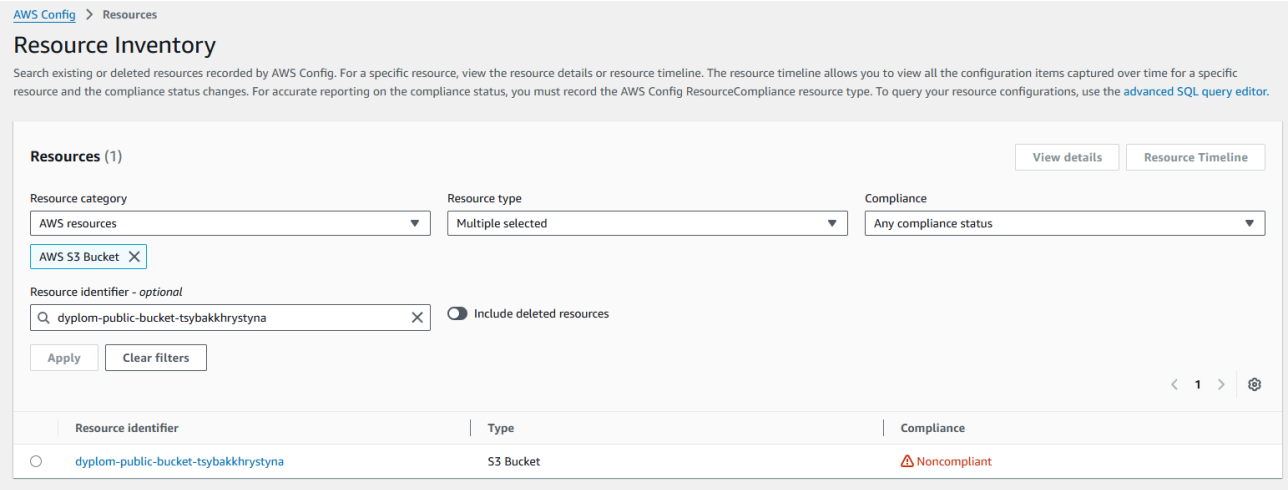

Рисунок 3.10 – Виявлення невідповідності конфігурації S3-бакета в AWS Config

Крім виявлення порушення конфігурації за допомогою AWS Config, інцидент також був ідентифікований сервісом Amazon GuardDuty. Цей сервіс аналізує журнали подій і мережеву активність на предмет аномальної або потенційно шкідливої поведінки, у даному випадку GuardDuty спрацював на відкритість ресурсу до зовнішнього світу, знайдений finding показано на рис. 3.11. Це підтверджує, що система моніторингу має багаторівневу логіку виявлення загроз, де одна й та сама подія може бути виявлена декількома механізмами, що значно підвищує надійність і зменшує ризик пропуску критичних ситуацій.

Після того як окремі сервіси, такі як AWS Config, GuardDuty чи CloudTrail виявляють інциденти, інформація про них надходить до AWS Security Hub. Цей сервіс виконує роль централізованого агрегатора, що об'єднує findings з різних джерел у єдиний уніфікований формат.

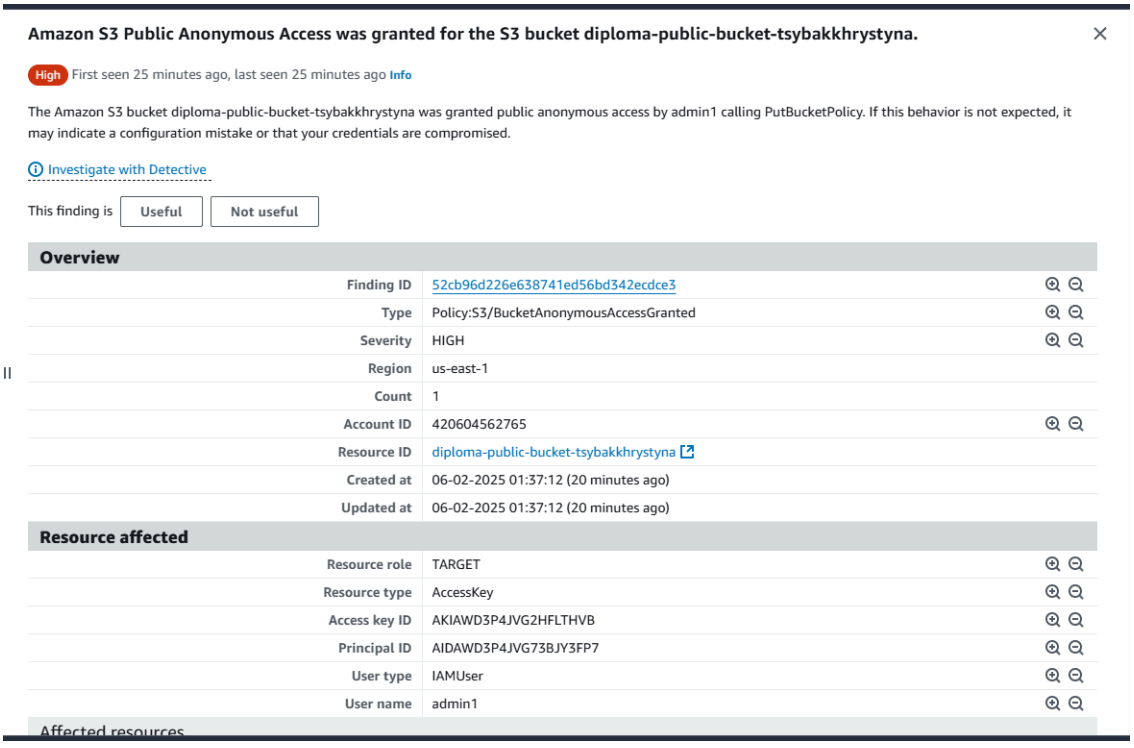

Рисунок 3.11 – Виявлення публічного доступу S3-бакета в AWS GuardDuty

Security Hub дозволяє не тільки переглядати всі інциденти в одному місці, а й автоматизовано фільтрувати події, оцінювати їх пріоритетність, критичність,

тип ресурсу, до якого вони належать, і час виявлення, що значно спрощує подальший аналіз та побудову логіки реагування. У межах розробленої системи саме Security Hub виступає основною точкою інтеграції для подальшої автоматизації, тому було налаштоване правило AWS EventBridge, яке реагує виключно на нові події типу Security Hub Finding. Таким чином, замість створення окремих механізмів для кожного джерела подій, уся логіка реагування централізована через один канал, що спрощує архітектуру, знижує ризик дублювання подій і дозволяє фільтрувати лише найбільш важливі з них.

Тож даний інцидент, а саме створення S3-бакета з відкритим публічним доступом, був успішно виявлений і переданий до AWS Security Hub із відповідним записом. У розділі findings сервісу було зафіксовано подію з назвою “S3 general purpose buckets should block public read access”, яка стосувалася бакета dyplom-public-bucket-tsybakkhrystyna2003. Події було надано рівень критичності CRITICAL, а статус відповідності був FAILED, що свідчить про порушення політики безпеки щодо відкритого доступу до об’єкта зберігання. Події Security Hub показані на рис. 3.12.

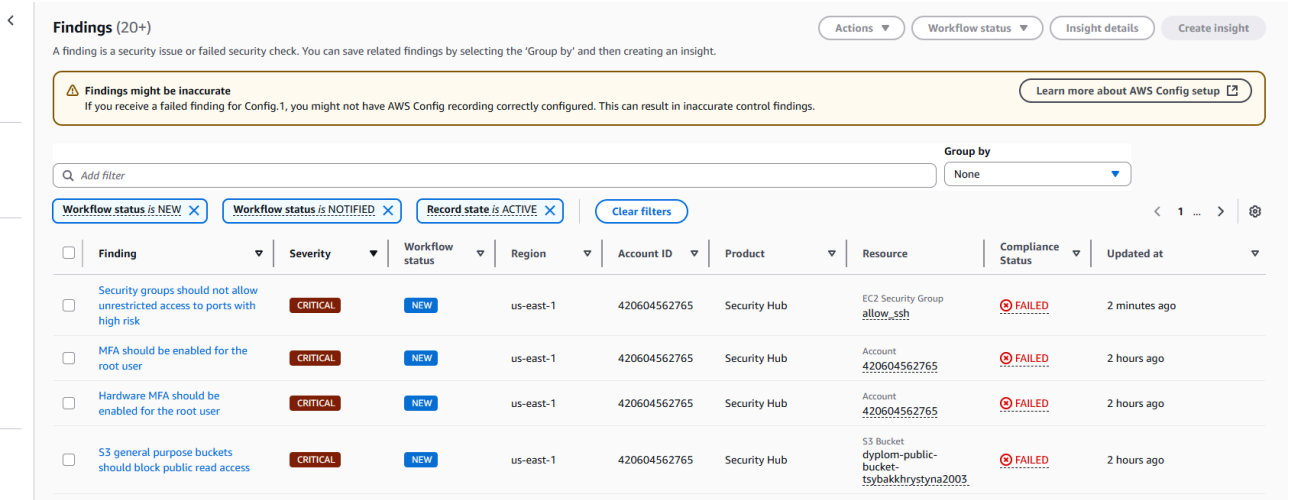

Рисунок 3.12 – Перелік подій безпеки, виявлених системою моніторингу в AWS Security Hub

Окрім цього інциденту, в Security Hub одночасно відображались й інші findings, які стосуються базових налаштувань безпеки в AWS акаунті, зокрема:

- Security groups should not allow unrestricted access to ports with high risk – подія, пов’язана з відкритим доступом до порту 22 з будь-якої IP-адреси, що є типовою помилкою конфігурації EC2 Security Group;
- MFA should be enabled for the root user – виявлено відсутність багатофакторної автентифікації для головного користувача облікового запису AWS;
- Hardware MFA should be enabled for the root user – рекомендація щодо більш надійного захисту за допомогою апаратного токена;
- VPC default security groups should not allow inbound or outbound traffic – виявлено , що група безпеки за замовчуванням у VPC дозволяє вхідний або вихідний трафік, що створює ризик неконтрольованого доступу;
- EC2 instances should not have a public IPv4 address – EC2-інстанс має публічну IP-адресу, що підвищує вірогідність зовнішніх атак.

Усі ці події мали статус NEW та були класифіковані як CRITICAL, що підкреслює їхню важливість для інформаційної безпеки. Їх наявність у Security Hub підтверджує, що система успішно агрегує findings із різних джерел, обробляє їх, і забезпечує централізований контроль за станом безпеки в хмарному середовищі.

Після підтвердження того, що змодельовані та автоматично виявлені інциденти були успішно зафіксовані в AWS Security Hub, наступним кроком стало тестування системи сповіщення. Метою цього етапу є переконатися, що критичні findings дійсно передаються з Security Hub до кінцевого отримувача, відповідно до реалізованої логіки. Протягом кількох секунд після появи події в Security Hub , було отримано email-повідомлення, що зображене на рис. 3.13 зі змістом, що включав:

- Назву інциденту та тип ресурсу;
- Ідентифікатор ресурсу;
- Ступінь ризику;

Це підтверджує, що ланцюжок реагування працює повністю: від виявлення події до її доставки відповідальній особі. У випадку підключення SIEM-системи, повідомлення також може бути дубльовано в зовнішнє середовище для подальшої кореляції подій.

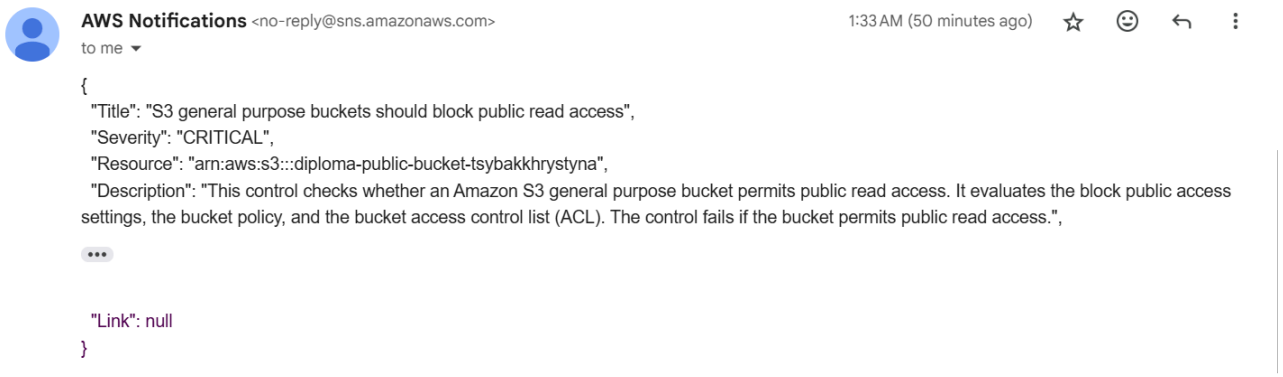

Рисунок 3.13 – Email-сповіщення з AWS SNS про критичне порушення безпеки

Для демонстрації інтеграції із SIEM системою була реалізована альтернативна версія Lambda-функції, що зображена на рис. 3.14, яка замість надсилання повідомлення на електронну пошту передає вміст події у SIEM-платформу Splunk за допомогою HTTP Event Collector.

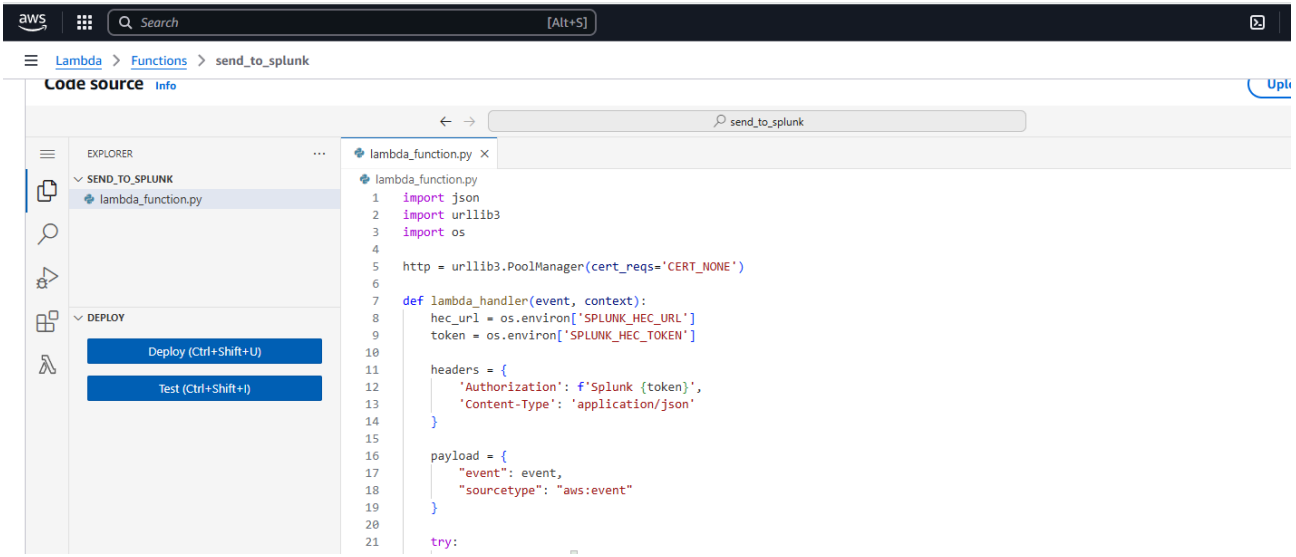

Рисунок 3.14 – Lambda-функція для надсилання подій безпеки у Splunk

Такий підхід дозволяє повністю автоматизувати обмін інформацією між системою моніторингу в AWS та платформою безпеки, яка використовується в організації для централізованого збору, аналізу та кореляції подій. Функція використовує бібліотеку urllib3 і передає повний JSON-вміст події у зазначений HEC endpoint, зберігаючи структуру даних для подальшої обробки. Доступ до конфігурацій, а саме URL і токен, забезпечується через змінні середовища, що відповідає принципам безпечного зберігання параметрів.

Після реалізації інтеграції було підтверджено, що події з AWS Security Hub успішно надходять до Splunk та зберігаються у відповідному індексі, що дозволяє переглядати findings у структурованому вигляді через інтерфейс Splunk, використовуючи стандартні засоби пошуку, побудову таблиць, графіків і часових діаграм. Приклад графіка зображений на рис 3.15.

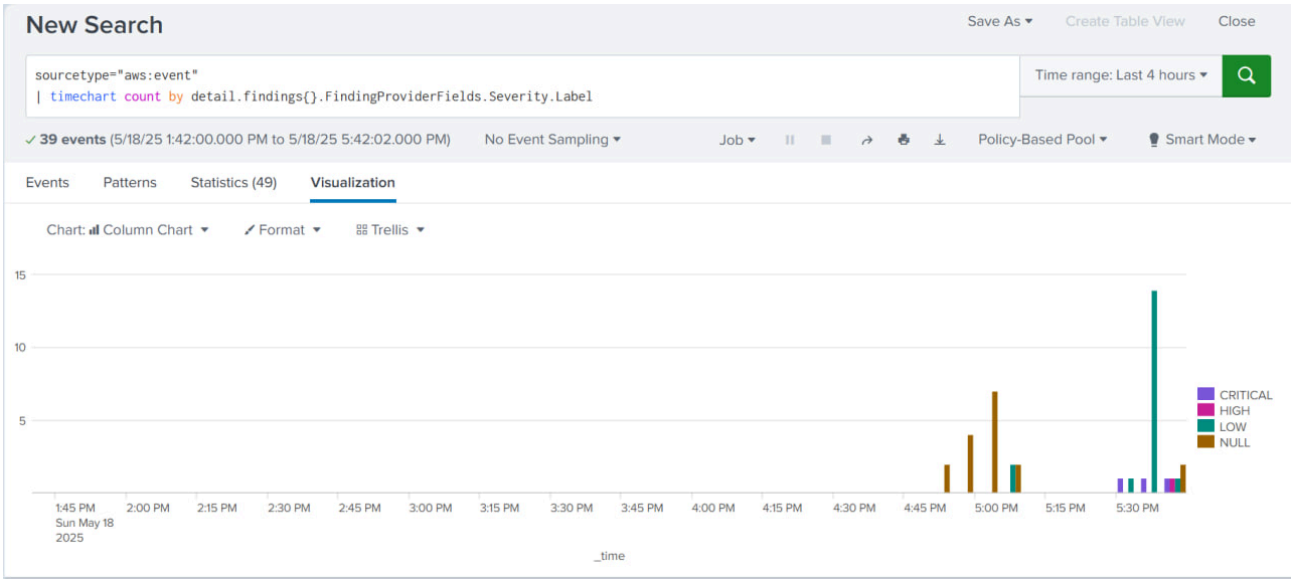

Рисунок 3.15 – Візуалізація подій безпеки з AWS Security Hub у Splunk за рівнем критичності

Після надходження в SIEM події можуть бути використані для побудови аналітичних дашбордів, що відображають динаміку інцидентів, їх розподіл за сервісами, регіонами або типами загроз. Також можна налаштувати правила кореляції, які виявляють складні атаки або повторювані патерни поведінки, що

не завжди помітні в межах одного середовища. Крім того, Splunk підтримує механізми реального часу оповіщення, наприклад, при виявленні критичних подій з певною частотою або у поєднанні з іншими ознаками загроз.

Таким чином, реалізована система моніторингу не обмежується лише виявленням подій у середовищі AWS, а стає частиною більшої корпоративної інфраструктури інформаційної безпеки, інтегруючись зі сторонніми засобами аналізу, аудиту та реагування. Це суттєво підвищує її практичну цінність, масштабованість і придатність до використання в умовах реального підприємства.

3.2.2 Підсумкова характеристика працездатності системи

Проведене тестування системи автоматизованого моніторингу подій безпеки в середовищі AWS дало змогу повноцінно оцінити її функціональність, стійкість до типових сценаріїв інцидентів, а також здатність до масштабованої інтеграції в інформаційну інфраструктуру підприємства. Перевірка охоплювала всі ключові компоненти, реалізовані в межах трьох рівнів моніторингу, що дозволило протестувати як базові функції фіксації подій, так і більш складні сценарії виявлення загроз, централізації обробки та автоматичного реагування.

Система продемонструвала цілісну роботу, а саме всі змодельовані порушення безпеки, серед яких були створення S3-бакета з публічним доступом, відкриття SSH-порту для всіх IP-адрес, надання надмірних IAM-дозволів та відсутність багатофакторної автентифікації для root-користувача, були своєчасно зафіксовані сервісами AWS Config, GuardDuty і CloudTrail. Дані про події надійшли до AWS Security Hub, де були класифіковані за ступенем критичності, типом ресурсу та джерелом. Це підтверджує правильну роботу компонентів збору, агрегації та нормалізації подій безпеки.

Подальша логіка реагування від Security Hub через EventBridge, Lambda та SNS працювала стабільно, без втрат подій чи збоїв у ланцюжку обробки. У випадку надсилання на електронну пошту сповіщення надходили в межах 30-60 секунд після виникнення інциденту. Окрім цього, реалізована альтернативна інтеграція з SIEM-платформою Splunk підтвердила можливість розширення системи за межі AWS.

Отже, тестування підтвердило, що система:

- забезпечує своєчасне виявлення інцидентів безпеки;
- коректно передає події до центрального хабу аналізу;
- автоматично сповіщає відповідальних осіб або зовнішні системи;
- сумісна з вимогами до корпоративних засобів захисту;
- є масштабованою, адаптивною та стійкою до помилок конфігурації.

З урахуванням результатів тестування, реалізоване рішення може бути рекомендоване для впровадження в організаціях, що використовують AWS як платформу для розміщення критичних сервісів.

Висновки до розділу 3

У даному розділі було здійснено програмну реалізацію автоматизованої системи моніторингу подій безпеки в хмарному середовищі AWS. Система побудована за модульним принципом з використанням інфраструктури як коду, а саме terraform та охоплює три рівні моніторингу – базовий, стандартний і розширений, кожен з яких забезпечує певний рівень глибини аналізу подій безпеки та масштабу контролю.

На базовому рівні реалізовано збір подій через CloudTrail, журналювання до CloudWatch та запуск перевірочних Lambda-функцій. Стандартний рівень розширює систему за рахунок підключення AWS Config та GuardDuty, що дозволяє виявляти конфігураційні порушення та потенційні загрози.

Розширений рівень включає додаткові механізми аналізу, зокрема Security Hub, VPC Flow Logs і інтеграцію зі SIEM-системами, такими як Splunk.

У результаті тестування підтверджено працездатність усіх компонентів системи, їхню здатність до виявлення критичних інцидентів безпеки та формування сповіщень у режимі, близькому до реального часу. Система також продемонструвала гнучкість, адаптивність і здатність до масштабування, що дозволяє рекомендувати її до використання в реальних корпоративних хмарних середовищах.

ВИСНОВКИ

У ході кваліфікаційної роботі було розв’язано задачу створення автоматизованої системи моніторингу подій безпеки в хмарному середовищі Amazon Web Services. Робота поєднує аналітичне вивчення хмарних технологій, методів забезпечення інформаційної безпеки та практичну реалізацію багаторівневої інфраструктури, здатної виявляти, обробляти та передавати події безпеки в режимі, наближеному до реального часу.

Результатом дослідження є реалізована автоматизована система моніторингу подій безпеки у хмарному середовищі AWS, яка підтримує три рівні гнучкого контролю, а саме базовий, стандартний, розширений, та представляє собою цілісне рішення, що об’єднує вбудовані сервіси платформи з додатковими механізмами перевірки та реагування. Вся інфраструктура описана кодом за підходом Infrastructure as Code, що забезпечує повну відтворюваність, контроль змін і зручність масштабування. Система дозволяє централізовано виявляти, фільтрувати й аналізувати події безпеки, автоматично надсилати сповіщення про інциденти та інтегруватися з зовнішніми SIEM-платформами. Проведене тестування підтвердило коректну роботу всіх компонентів, здатність системи виявляти події, обробляти інциденти й передавати їх до зовнішніх систем, що підтверджує її ефективність у практичному застосуванні.

У межах дослідження реалізовано такі ключові етапи:

1. Проведено аналіз сучасного стану безпеки хмарних середовищ. Встановлено, що хмарні платформи забезпечують гнучкість та масштабованість, однак супроводжуються ризиками – помилки конфігурацій, загрози витоку даних та несанкціонованого доступу, уразливості на стороні користувача, що зумовлює потребу у впровадженні багаторівневого моніторингу для своєчасного виявлення й реагування на інциденти безпеки.

2. Оцінено вбудовані сервіси для контролю подій безпеки в AWS. Досліджено функціональні можливості таких сервісів, як CloudTrail, GuardDuty,

Config, CloudWatch і Security Hub. Встановлено, що їх поєднання дозволяє забезпечити аудит дій, виявлення аномалій, контроль конфігурацій і централізоване управління подіями безпеки, що в свою чергу створює основу для побудови автоматизованої моніторингової системи.

3. Розроблено архітектуру багаторівневої системи моніторингу. Кожен рівень (базовий, стандартний, розширений) реалізує відповідну глибину аналізу, що дозволяє адаптувати систему під конкретні потреби організації.

4. Реалізовано інфраструктуру за допомогою інструменту terraform. Весь проєкт побудовано на принципах Infrastructure as Code, де модульна структура, параметризація, можливість контролю змін через код забезпечують ефективне керування конфігурацією.

5. Проведено тестування розробленої системи. У середовищі, наближеному до реального, змодельовано інциденти безпеки, система моніторингу коректно виконувала фіксацію, фільтрацію та передачу подій, що підтверджує її практичну придатність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Батаєв С. В., Мельник О. С. Аналіз принципів роботи, переваг та викликів у використанні хмарних технологій в умовах сьогодення // Інформатика, обчислювальна техніка та автоматизація. 2024. Том 35 (74). № 3. С. 31–36.

2. Slingerland C. The Simple Guide To The History Of The Cloud. Cloudzero. 15.12.2023. URL: https://www.cloudzero.com/blog/history-of-the-cloud/ (дата звернення: 10.01.2025).

3. Cloud Computing Market Size, Share, and Trends 2025 to 2034. Precedenceresearch. URL: https://www.precedenceresearch.com/cloud-computing-market. (дата звернення: 10.01.2025).

4. How Many Companies Use Cloud Computing in 2025? [10 Statistics and Insights]. Edgedelta. 17.05.2024. URL: https://edgedelta.com/company/blog/how-many-companies-use-cloud-computing (дата звернення: 10.01.2025).

5. The Best Cloud Migration Success Stories & What We Can Learn from Them. Smartocs. URL: https://www.smartosc.com/the-best-cloud-migration-success-stories-what-we-can-learn-from/ (дата звернення: 14.01.2025).

6. Андрощук О.В., Головченко О.В., Литовченко Г.Д., Петрушен М.В. АНАЛІЗ ПОНЯТТЯ ХМАРНІ ТЕХНОЛОГІЇ: ВИДИ, КАТЕГОРІЇ, ПЕРЕВАГИ ТА НЕДОЛІКИ // Молодий вчений. 2021. №6(94). С. 83–87.

7. Cloud Computing Mastery: Key Strategies and Success Stories for Transformation. Medium. URL: https://ip-specialist.medium.com/cloud-computing-mastery-key-strategies-and-success-stories-for-transformation-8f48c3506457 (дата звернення: 18.01.2025).

8. Ітан Райт. Типи хмарних обчислень – публічні, приватні, гібридні та спільноти. Guru99. 15.12.2023. URL: https://www.guru99.com/uk/types-of-cloud-computing.html (дата звернення: 20.01.2025).

9. Abhishek Arora. 15 Cloud Security Issues: Risks, Threats, and Challenges. CloudDefense.AI. URL: https://www.clouddefense.ai/cloud-security-issues-threats-challenges/ (дата звернення: 20.01.2025).

10. 17 Security Risks of Cloud Computing in 2025. SentinelOne. URL: https://www.sentinelone.com/cybersecurity-101/cloud-security/security-risks-of-cloud-computing/#17-security-risks-of-cloud-computing. (дата звернення: 23.01.2025).

11. Софія Пилипюк. В Uber приховували витік даних про 57 мільйонів користувачів. Village. URL: https://www.village.com.ua/village/business/news/265053-uber-prihovuvala-vitik-danih-pro-57-milyoniv-koristuvachiv. (дата звернення: 23.01.2025).

12. Cloud Security Threats: Top Threats and 3 Mitigation Strategies. Exabeam. URL: https://www.exabeam.com/explainers/cloud-security/cloud-security-threats-top-threats-and-3-mitigation-strategies/ . (дата звернення: 29.01.2025).

13. Dana Raveh. 20 Cloud Security Best Practices. 20.05.2024. URL: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/cloud-security-best-practices/ . (дата звернення: 29.01.2025).

14. Overcoming Cloud Security Challenges: Key Risks and Threats. Varonis. URL: https://www.varonis.com/blog/cloud-security-challenges#building-a-holistic-cloud-security-strategy. (дата звернення: 29.01.2025).

15. What is AWS Cloud Security? How Does it Work? Scalahosting. URL: https://www.scalahosting.com/blog/what-is-aws-cloud-security-how-does-it-work/. (дата звернення: 05.02.2025).

16. What is Azure Security? Orca-security. URL: https://orca.security/resources/blog/what-is-azure-security/. (дата звернення: 05.02.2025).

17. Google Cloud Security: A Complete Guide to GCP Security. SentinelOne. URL: https://www.sentinelone.com/cybersecurity-101/cloud-security/google-cloud-security/. (дата звернення: 05.02.2025).

18. Top 7 Benefits of AWS: Advantages and Disadvantages of AWS. Intellipaat. URL: https://intellipaat.com/blog/aws-benefits-and-drawbacks/. (дата звернення: 15.02.2025).

19. Szili D. Building a Threat Detection Strategy in AWS. SANS Institute, 2019. 19 с. URL: https://pages.awscloud.com/rs/112-TZM-766/images/Threat%20Detection_SANS%20and%20AWS%20Marketplace%20whitepaper.pdf. (дата звернення: 17.02.2025)

20. Comparing AWS, Azure, GCP. 16.08.2023. URL: https://www.digitalocean.com/resources/articles/comparing-aws-azure-gcp. (дата звернення: 20.02.2025)

21. Security Automation in AWS. Medium. 09.03.2022. URL: https://medium.com/@cloud_tips/security-automation-in-aws-22e45aa8aa56 (дата звернення: 25.02.2025)

22. Shackleford D. Cloud security automation: Benefits and best practices. Techtarget. 05.01.2024. URL: https://www.techtarget.com/searchsecurity/tip/4-steps-toward-cloud-security-automation (дата звернення: 25.02.2025)

23. Sylva I. Automating AWS Security with Code a Comprehensive Guide. Dev. 27.11.2024. URL: https://dev.to/ikoh_sylva/automating-aws-security-with-code-a-comprehensive-guide-69f. (дата звернення: 27.02.2025)

24. Amazon CloudTrail. Track user activity and API usage. URL: https://www.amazonaws.cn/en/cloudtrail/. (дата звернення: 02.03.2025)

25. Sumit K. AWS CloudTrail with Centralized Logging — Deployment with Terraform. 23.05.2023. URL: https://dev.to/ikoh_sylva/automating-aws-security-with-code-a-comprehensive-guide-69f. (дата звернення: 02.03.2025)

26. Surve Y. Leveraging the Power of AWS CloudTrail with AWS CloudWatch: A Complete Guide. 01.02.2023. URL: https://medium.com/@surveyogita1200/leveraging-the-power-of-aws-cloudtrail-with-aws-cloudwatch-a-complete-guide-e66d2b5b1bfa. (дата звернення: 02.03.2025)

27. What is Amazon CloudWatch Logs? URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html. (дата звернення: 06.03.2025)

28. Awad J. AWS CloudWatch Logs Deep Dive. 17.02.2025. URL: https://medium.com/@joudwawad/aws-cloudwatch-logs-deep-dive-d52b5bb7c40d. (дата звернення: 06.03.2025)

29. What is AWS Lambda? URL: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. (дата звернення: 10.03.2025)

30. Bitton Y. Securing AWS Lambda. How Misconfigurations Can Lead to Lateral Movement. 15.11.2024. URL: https://www.sentinelone.com/blog/lateral-movement-in-aws-lambda-environments/. (дата звернення: 10.03.2025)

31. Automated AWS Security Monitoring: A Python-Based AWS Security Tool. 10.02.2025. URL: https://dev.to/leonardkachi/automated-aws-security-monitoring-a-python-based-aws-security-tool-43nm. (дата звернення: 10.03.2025)

32. Amazon GuardDuty. URL: https://aws.amazon.com/guardduty/?nc1=h_ls. (дата звернення: 16.03.2025)

33. Best Practices for Amazon GuardDuty. URL: https://www.dashsdk.com/resource/best-practices-for-amazon-guardduty/ (дата звернення: 16.03.2025)

34. Amazon GuardDuty customers. URL: https://aws.amazon.com/guardduty/customers/?customer-references-cards.sort-by=item.additionalFields.sortDate&customer-references-cards.sort-order=desc&awsf.language=*all&awsf.content-type=*all&awsf.customer-references-location=*all&awsf.customer-references-segment=*all&awsf.customer-references-industry=*all&awsf.customer-references-use-case=*all&awsf.customer-references-tech-category=*all&awsf.customer-references-product=*all (дата звернення: 16.03.2025)

35. AWS Config best practices. URL: https://aws.amazon.com/blogs/mt/aws-config-best-practices/ (дата звернення: 19.03.2025)

36. 12 AWS Config rules that every account should have. 08.06.2023. URL: https://www.pluralsight.com/resources/blog/cloud/12-aws-config-rules-that-every-account-should-have. (дата звернення: 19.03.2025)

37. AWS Security Hub. URL: https://aws.amazon.com/security-hub/ (дата звернення: 23.03.2025)

38. Ceresnak R. Implementing Continuous Security Monitoring with AWS Security Hub. 18.04.2024. URL: https://medium.com/codex/implementing-continuous-security-monitoring-with-aws-security-hub-e6aba6772eb8 (дата звернення: 23.03.2025)

39. Що таке SIEM? Інформація про безпеку та управління подіями. URL: https://gridinsoft.ua/siem (дата звернення: 25.03.2025)

40. Douglas N. Terraform Security Best Practices. 21.03.2023. URL: https://sysdig.com/blog/terraform-security-best-practices/ (дата звернення: 27.03.2025)

Додаток А

Структура інфраструктурного коду системи моніторингу подій безпеки

```
PS C:\Users\Христина\Desktop\Диплом\cloud-security-monitoring\terraform> tree
/F /a | findstr /V ".terraform" | findstr /V "tfstate" | findstr /V "lock.hcl"
        Folder PATH listing
        Volume serial number is 5003-2FDA
        C:.
        |   deploy.py
        |   main.tf
        |   provider.tf
        |   variables.tf
        |
        |   +---modules
        |   |       modules.json
        |   |
        |   \---providers
        |           \---hashicorp
        |               +---aws
        |               |   \---5.97.0
        |               |       \---windows_amd64
        |               |               LICENSE.txt
        |               |
        |               \---random
        |                   \---3.7.2
        |                       \---windows_amd64
        |                               LICENSE.txt
        |
```

Продовження додатку А

```
\---modules
    +---advanced
    |   |   main.tf
    |   |   variables.tf
    |   |
    |   \---services
    |       +---alerting
    |       |   |   main.tf
    |       |   |   outputs.tf
    |       |   |   variables.tf
    |       |   |
    |       |   \---lambda
    |       |           forward_findings.py
    |       |           forward_findings.zip
    |       |
    |       +---securityhub
    |       |       main.tf
    |       |
    |       \---vpc-flowlogs
    |               main.tf
    |               outputs.tf
    |               variables.tf
    |
    +---basic
    |   |   main.tf
    |   |   outputs.tf
    |   |   variables.tf
    |   |
```

Продовження додатку А

```
|   \---services
|       +---cloudtrail
|       |       main.tf
|       |       outputs.tf
|       |       variables.tf
|       |
|       +---cloudwatch
|       |       main.tf
|       |       outputs.tf
|       |       variables.tf
|       |
|       +---lambda_checkers
|       |   |   main.tf
|       |   |   output.tf
|       |   |   variables.tf
|       |   |
|       |   \---lambda
|       |           mfa_check.py
|       |           mfa_check.zip
|       |           s3_check.py
|       |           s3_check.zip
|       |           sg_check.py
|       |           sg_check.zip
|       |
|       \---sns
|               main.tf
|               outputs.tf
|               variables.tf
```

Продовження додатку А

```
|
\---standard
    |   main.tf
    |   outputs.tf
    |   variables.tf
    |
    \---services
        +---alerting
        |       main.tf
        |       outputs.tf
        |       variables.tf
        |
        +---aws-config
        |       main.tf
        |       outputs.tf
        |       variables.tf
        |
        \---guardduty
                main.tf
```

PS C:\Users\Христина\Desktop\Диплом\cloud-security-monitoring\terraform>

Додаток Б

Скрипт автоматизованого розгортання інфраструктури deploy.py

```
import json
import subprocess

# Step 1. Choose monitoring level
level = input("Select monitoring level (basic / standard / advanced): ").strip().lower()
if level not in ["basic", "standard", "advanced"]:
    print("Invalid level selected.")
    exit(1)

# Step 2. Enter region
region = input("Enter AWS region (default: us-east-1): ").strip()
if not region:
    region = "us-east-1"

# Step 3. Alert email
email = input("Enter alert email (leave empty for default): ").strip()
if not email:
    email = "example@example.com"

# Step 4. VPC ID only for advanced
vpc_id = None
if level == "advanced":
    vpc_id = input("Enter VPC ID (optional, leave empty to skip VPC Flow Logs): 
").strip()
```

Продовження додатку Б

```
    if not vpc_id:
        print("No VPC ID provided. Advanced module will skip VPC-related
resources.")


# Step 5. Write terraform.tfvars.json
tfvars = {
    "level": level,
    "region": region,
    "alert_email": email
}
if level == "advanced":
    tfvars["vpc_id"] = vpc_id if vpc_id else None

with open("terraform.tfvars.json", "w") as f:
    json.dump(tfvars, f, indent=4)
print("terraform.tfvars.json has been updated.")

# Step 6. Run Terraform
print("Running Terraform...")
subprocess.run(["terraform", "init"])
subprocess.run(["terraform", "apply"])
```