Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка експертної системи підтримки прийняття рішень у сфері кібербезпеки
(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-24-1
Загоруйко А.А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)
(повна назва спеціалізації)
Керівник доц. Золотухін О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лариса ЧАЛА
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані (Data Science)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Загоруйку Антону Анатолієвичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка експертної системи підтримки прийняття рішень у сфері кібербезпеки

затверджена наказом університету від 24 листопада 20 25 р. № 1057Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 20 25 р.

3. Вихідні дані до роботи операційна система: Windows 11; мова програмування Python 3.10; технології ШІ: символьний штучний інтелект на основі продукційної моделі представлення знань, алгоритм прямого логічного висновку; бази даних та сховища: * NoSQL СУБД MongoDB; веб-технології та інтерфейс: фреймворк Flask, WebSocket, Bootstrap 5; стандарти та дані: класифікація атак за методиками OWASP Top 10 (зокрема Injection) та сценарії Brute Force.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Розробка архітектури системи
3) Практична реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	24.11.2025	виконано
2	Вибір та обґрунтування методики дослідження	25.11.2025-30.11.2025	виконано
3	Вибір технології розробки та інструментальних	1.12.2025-5.12.2025	виконано
	засобів		
4	Розробка та відлагодження програмного	6.12.2025-12.2025	виконано
	забезпечення		
5	Аналіз отриманих результатів	12.12.2025-16.12.2025	виконано
6	Подання кваліфікаційної роботи керівникові та її	17.12.2025	виконано
	попередній захист		
7	Подання кваліфікаційної роботи на рецензування	22.12.2025	виконано
8	Захист перед ЕК	23.12.2025	

Дата видачі завдання 24 листопада 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Золотухін О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 69 с., 4 рис., 1 табл., 1 дод., 18 джерел.

БАЗА ЗНАНЬ, ЕКСПЕРТНІ СИСТЕМИ, КІБЕРБЕЗПЕКА, ПОЯСНИМИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ПРАВИЛА ВИРОБНИЦТВА, РЕАГУВАННЯ НА ІНЦИДЕНТИ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єктом дослідження є процеси моніторингу та оперативного реагування на інциденти інформаційної безпеки в сучасних комп'ютерних мережах, що функціонують під впливом різноманітних типів цілеспрямованих атак.

Предметом дослідження виступають методи, моделі та програмні засоби побудови експертних систем на основі продукційних правил, а також алгоритми прямого логічного висновку та механізми формалізації експертних знань у галузі кіберзахисту.

Мета роботи полягає у підвищенні ефективності функціонування систем захисту інформації шляхом створення програмного прототипу експертної системи, яка забезпечує автоматизований аналіз подій безпеки у реальному часі та генерацію верифікованих рекомендацій для оператора, що дозволяє мінімізувати вплив «людського фактору» на швидкість детекції загроз.

Методи дослідження базуються на комплексному використанні теорії штучного інтелекту та інженерії знань для побудови логічної структури системи. У процесі проєктування було застосовано методи системного аналізу для декомпозиції функцій захисту, принципи об'єктно-орієнтованого програмування для реалізації модульної архітектури, а також методи математичної статистики та експериментального тестування для оцінки швидкодії та точності роботи розроблених алгоритмів на контрольних сценаріях атак.

ABSTRACT

Master’s thesis contains: 69 p., 4 fig., 1 tabl., 1 ann., 18 references.

ARTIFICIAL INTELLIGENCE, CYBERSECURITY, EXPERT SYSTEMS, EXPLAINABLE ARTIFICIAL INTELLIGENCE, INCIDENT RESPONSE, KNOWLEDGE BASE, PRODUCTION RULES.

The object of the study is the processes of monitoring and operational response to information security incidents in modern computer networks that operate under the influence of various types of targeted attacks.

The subject of the study is methods, models and software for building expert systems based on production rules, as well as direct logical inference algorithms and mechanisms for formalizing expert knowledge in the field of cyber security.

The purpose of the work is to increase the efficiency of information protection systems by creating a software prototype of an expert system that provides automated analysis of security events in real time and generation of verified recommendations for the operator, which allows minimizing the impact of the "human factor" on the speed of threat detection.

The research methods are based on the integrated use of the theory of artificial intelligence and knowledge engineering to build the logical structure of the system. In the design process, systems analysis methods were used to decompose protection functions, the principles of object-oriented programming to implement a modular architecture, as well as methods of mathematical statistics and experimental testing to assess the speed and accuracy of the developed algorithms on control attack scenarios.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 9
Вступ 11
1 Аналіз предметної галузі 13
1.1 Кібербезпека та управління ризиками 13
1.1.1 Фундаментальні концепції кібербезпеки: тріада CIA 13
1.1.2 Концепція управління ризиками як основа прийняття рішень 14
1.2 Типологія та аналіз сучасних кіберзагроз 15
1.2.1 Шкідливе програмне забезпечення (Malware) 16
1.2.2 Соціальна інженерія та фішинг 16
1.2.3 Розподілені атаки на відмову в обслуговуванні (DDoS) 17
1.2.4 Атаки нульового дня (Zero-day attacks) 17
1.3 Процес прийняття рішень у сфері кібербезпеки 18
1.3.1 Реагування на інциденти та управління кризовими ситуаціями 18
1.3.2 Оцінка та пріоритезація вразливостей 19
1.3.3 Вибір та оптимізація захисних механізмів 20
1.4 Проблематика людського фактора в управлінні кібербезпекою 20
1.4.1 Проблема надвеликих обсягів даних (Big Data) та когнітивне перевантаження 21
1.4.2 Швидкість змін та асиметрія часу реакції 22
1.4.3 Складність інфраструктурних взаємозв'язків 22
1.4.4 Суб'єктивність та психологічні фактори 23
1.5 Концепції експертних систем (ЕС) та систем підтримки прийняття рішень (СППР) 23
1.5.1 Системи підтримки прийняття рішень (СППР) 24
1.5.2 Експертні системи (ЕС) 24
1.5.3 Відмінності від звичайних інформаційних систем 25
1.6 Архітектура та основні компоненти експертної системи 26

1.6.1 База знань (Knowledge Base) 26
1.6.2 Машина логічного висновку (Inference Engine) 28
1.6.3 Інтерфейс користувача (User Interface) 28
1.6.4 Підсистема набуття знань (Knowledge Acquisition Subsystem) 29
1.7 Методи представлення знань в експертних системах 29
1.7.1 Продукційні правила (If-Then Rules) 30
1.7.2 Семантичні мережі (Semantic Networks) 31
1.7.3 Фрейми (Frames) 31
1.8 Огляд та аналіз існуючих рішень автоматизації кіберзахисту 32
1.8.1 Системи управління подіями та інформацією про безпеку (SIEM) 32
1.8.2 Платформи оркестрації, автоматизації та реагування (SOAR) 33
1.8.3 Системи управління вразливостями (Vulnerability Management) 34
1.8.4 Висновок та обґрунтування необхідності власної розробки 34
2 Розробка архітектури системи 36
2.1 Постановка задачі та визначення вимог до системи 36
2.1.1 Формалізація задачі 36
2.1.2 Функціональні вимоги 37
2.1.3 Нефункціональні вимоги 37
2.2 Розробка архітектури програмного забезпечення 38
2.2.1 Модуль збору та попередньої обробки даних (Data Ingestion & Pre-processing) 38
2.2.2 Ядро логічного висновку (Inference Engine Core) 39
2.2.3 Підсистема управління базою знань (Knowledge Management) 39
2.2.4 Інтерфейс та візуалізація (Presentation Layer) 40
2.3 Моделювання бази знань та структур даних 41
2.3.1 Формалізація фактів (модель даних) 42
2.3.2 Формалізація правил (модель знань) 42
2.3.3 Приклад структурної реалізації (JSON) 43

2.4 Вибір та обґрунтування засобів програмної реалізації 45
2.4.1 Мова програмування Python 45
2.4.2 Система управління базами даних (NoSQL / MongoDB) 46
2.4.3 Веб-фреймворк та інтерфейс користувача 46
2.4.4 Середовище розробки та контролю версій 47
3 Практична реалізація та оцінка ефективності 48
3.1 Реалізація алгоритмічного ядра штучного інтелекту 48
3.1.1 Алгоритм циклу розпізнавання-дії (Recognize-Act Cycle) 48
3.1.2 Реалізація механізму евристичного пошуку 49
3.2 Програмна реалізація модулів та архітектури системи 50
3.2.1 Реалізація класів ядра системи 50
3.2.2 Програмна реалізація циклу висновку 53
3.2.3 Реалізація API та інтеграція 55
3.2.4 Реалізація підсистеми збору та нормалізації даних (Parser Module) 55
3.3 Реалізація інтерфейсу користувача та підсистеми пояснень 58
3.3.1 Інформаційна панель (Security Dashboard) 59
3.3.2 Реалізація механізму пояснень (Explanation Facility) 60
3.4 Методика проведення експерименту та стенд 61
3.5 Сценарії тестування та отримані дані 62
3.6 Порівняльний аналіз ефективності та візуалізація 63
Висновки 65
Перелік джерел посилання 67
Додаток А Відомість кваліфікаційної роботи 69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БЗ – база знань;

БД – база даних;

ЕС – експертна система;

ІБ – інформаційна безпека;

ІТ – інформаційні технології;

МЛВ – машина логічного висновку;

ООП – об’єктно-орієнтоване програмування;

СУБД – система управління базами даних;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс взаємодії між компонентами системи;

BSON – Binary JSON – бінарний формат обміну даними, що використовується в СУБД MongoDB;

DoS – Denial of Service – атака, спрямована на відмову у звичайному обслуговуванні легітимних користувачів;

DDoS – Distributed Denial of Service – розподілена атака на відмову в обслуговуванні;

IDS – Intrusion Detection System – система виявлення вторгнень;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

MTTR – Mean Time to Respond – середній час реагування на інцидент;

NoSQL – Not Only SQL – підхід до побудови баз даних, що не використовує реляційні моделі;

REST – Representational State Transfer – архітектурний стиль взаємодії компонентів через протокол HTTP;

SOAR – Security Orchestration, Automation, and Response – технології автоматизації та координації процесів безпеки;

SSH – Secure Shell – протокол захищеного віддаленого доступу;

SPA – Single Page Application – односторінковий веб-застосунок;

TTL – Time to Live – час життя даних (факту) у пам’яті системи;

TTD – Time to Detect – час виявлення інциденту;

TTR – Time to Respond – час реагування на інцидент;

UI – User Interface – інтерфейс користувача;

XAI – Explainable Artificial Intelligence – пояснювальний (прозорий) штучний інтелект.

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімкою та всеосяжною цифровізацією, що, поряд із беззаперечними перевагами, створює нові виклики у сфері кібербезпеки. Зростання кількості та складності кібератак, а також розширення ландшафту загроз призводять до того, що традиційні засоби захисту генерують колосальні обсяги даних моніторингу. Це створює парадоксальну ситуацію, коли надлишок інформації не підвищує, а навпаки, знижує рівень реальної захищеності, оскільки оператори центрів безпеки фізично не спроможні якісно обробити цей потік у ручному режимі. Інформаційне перевантаження фахівців призводить до професійного вигорання, збільшення кількості помилок внаслідок «людського фактору» та критичного зростання часу між виявленням загрози та реагуванням на неї, що надає зловмисникам часову перевагу для нанесення шкоди.

Актуальність теми дипломної роботи зумовлена гострою необхідністю скорочення цього розриву шляхом автоматизації інтелектуальних процесів аналізу даних безпеки. В умовах дефіциту висококваліфікованих кадрів на ринку праці виникає потреба у засобах, здатних акумулювати та автоматично застосовувати експертний досвід. Найбільш перспективним напрямком вирішення цієї проблеми є застосування методів символьного штучного інтелекту, зокрема експертних систем. На відміну від алгоритмів машинного навчання, які часто працюють за принципом «чорної скриньки», експертні системи базуються на формалізованих правилах прийняття рішень, забезпечуючи прозорість та інтерпретованість результатів, що є критично важливим фактором довіри у сфері інформаційної безпеки.

Метою даної роботи є підвищення ефективності процесів виявлення та нейтралізації кіберзагроз шляхом розробки та впровадження експертної системи підтримки прийняття рішень, здатної в автоматизованому режимі

аналізувати події безпеки та генерувати верифіковані рекомендації для оператора.

Для досягнення поставленої мети в роботі послідовно вирішується комплекс взаємопов'язаних завдань, що включає аналіз сучасного стану проблеми та методів виявлення загроз, обґрунтування вибору архітектури експертної системи, проєктування структури бази знань для формалізації евристик безпеки, програмну реалізацію прототипу системи з використанням сучасного технологічного стека, а також проведення експериментального дослідження для кількісної оцінки ефективності розробленого рішення. Об'єктом дослідження виступає процес моніторингу та реагування на інциденти інформаційної безпеки в корпоративних мережах, а предметом дослідження є методи, моделі та програмні засоби побудови експертних систем для автоматизації цього процесу. Методологічну основу роботи складають методи системного аналізу, інженерії знань, об'єктно-орієнтованого проєктування та експериментального тестування програмного забезпечення. Практична цінність одержаних результатів полягає у створенні дієвого інструментарію, який дозволяє розвантажити аналітиків безпеки від рутинних операцій, мінімізувати вплив людського фактору на прийняття рішень та суттєво скоротити час реакції на типові кібератаки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Кібербезпека та управління ризиками

Сучасний стан розвитку інформаційних технологій характеризується тотальною цифровізацією всіх сфер людської діяльності, що, поряд із беззаперечними перевагами, створює нові вектори загроз для національної безпеки, економічної стабільності та приватності особистості. У цьому контексті кібербезпека перестає бути суто технічним питанням і перетворюється на складну багатогранну дисципліну, що охоплює стратегічне управління, правове регулювання та розробку інтелектуальних систем захисту. Аналіз предметної галузі вимагає глибокого розуміння фундаментальних засад, на яких будується захист інформаційного простору, а також усвідомлення того, що будь-яка система безпеки є насамперед інструментом мінімізації ризиків у динамічному середовищі [1].

1.1.1 Фундаментальні концепції кібербезпеки: тріада CIA

В основі сучасної теорії захисту інформації лежить класична концептуальна модель, відома як тріада CIA, яка охоплює три взаємопов’язані властивості: конфіденційність, цілісність та доступність. Ці компоненти визначають цілі безпеки для будь-якого інформаційного активу, а їхнє порушення розцінюється як інцидент кібербезпеки різного ступеня тяжкості.

Конфіденційність інформації визначається як стан, за якого доступ до даних обмежений суворим колом авторизованих суб'єктів. В умовах сучасних кіберзагроз забезпечення конфіденційності вимагає впровадження комплексних методів ідентифікації, автентифікації та авторизації, а також використання стійких алгоритмів шифрування. Основна проблема полягає у запобіганні несанкціонованому розголошенню чутливих даних, таких як

персональна інформація, комерційна таємниця або державні секрети. Важливо розуміти, що конфіденційність стосується не лише зберігання даних, але й процесу їх передачі через відкриті та захищені канали зв'язку, де ризик перехоплення інформації є постійним фактором впливу.

Цілісність виступає гарантом того, що інформація зберігається у первинному вигляді та не була піддана несанкціонованій, випадковій або зловмисній модифікації. Забезпечення цілісності є критично важливим для систем, де точність даних безпосередньо впливає на процеси прийняття рішень, наприклад, у фінансовому секторі, медицині або системах управління критичною інфраструктурою. Механізми контролю цілісності базуються на використанні контрольних сум, криптографічних хеш-функцій та цифрових підписів. Будь-яка невідповідність між вихідним та поточним станом даних має бути негайно виявлена системою захисту, оскільки спотворення інформації може призвести до катастрофічних наслідків, навіть якщо сама інформація залишається доступною та конфіденційною.

Доступність замикає тріаду і полягає у забезпеченні своєчасного та безперебійного доступу легітимних користувачів до інформаційних ресурсів та сервісів. У сучасному бізнес-середовищі, де час простою вимірюється значними фінансовими збитками, доступність стає одним із найпріоритетніших завдань. Загрози доступності можуть мати різну природу: від цілеспрямованих DDoS-атак, що мають на меті виснаження ресурсів системи, до технічних збоїв обладнання або стихійних лих [2].

1.1.2 Концепція управління ризиками як основа прийняття рішень

Перехід до розробки експертної системи підтримки прийняття рішень неможливий без детального аналізу процесів управління ризиками. У сфері кібербезпеки ризик розглядається не просто як ймовірність негативної події, а як комплексна величина, що виникає на перетині існуючих активів,

потенційних загроз та наявних вразливостей. Управління ризиками – це ітераційний процес, спрямований на ідентифікацію потенційних небезпек, оцінку їхнього можливого впливу на організацію та вибір оптимальної стратегії реагування.

Основна складність управління ризиками в кіберпросторі полягає в асиметрії знань та високій швидкості появи нових методів атак. Експертна система в цьому контексті має виступати аналітичним ядром, яке здатне обробляти великі масиви вхідних даних про стан системи та зовнішнє середовище загроз. Вона повинна допомагати адміністратору безпеки ранжувати ризики за ступенем їхньої критичності, враховуючи як імовірність успішної атаки, так і потенційні фінансові чи репутаційні втрати.

Процес прийняття рішень у сфері захисту інформації завжди пов’язаний із пошуком балансу між вартістю впровадження засобів захисту та цінністю активів, які вони захищають. Саме тому розробка інтелектуальної системи є актуальним завданням, адже людський фактор часто стає слабкою ланкою через обмежену здатність до оперативного опрацювання великої кількості факторів у стресових ситуаціях. Таким чином, аналіз предметної галузі підтверджує, що ефективна кібербезпека сьогодні базується на інтеграції фундаментальних принципів CIA з гнучкими методологіями управління ризиками, що реалізуються за допомогою передових систем підтримки прийняття рішень.

1.2 Типологія та аналіз сучасних кіберзагроз

Динамічний розвиток інформаційних технологій та глобальна цифровізація процесів управління зумовлюють безперервну еволюцію методів здійснення деструктивних впливів у кіберпросторі. Класифікація сучасних загроз є критично важливою умовою для побудови ефективної експертної системи, оскільки характер загрози безпосередньо визначає

логіку прийняття рішень щодо її нейтралізації. Сучасний ландшафт загроз характеризується високим ступенем адаптивності та часто комбінованим характером атак, де технічні вразливості поєднуються з методами психологічного маніпулювання, що робить задачу їх своєчасного виявлення пріоритетною для систем інтелектуальної підтримки.

1.2.1 Шкідливе програмне забезпечення (Malware)

Шкідливе програмне забезпечення на сучасному етапі трансформувалося з відносно простих вірусів у складні багатокомпонентні екосистеми, здатні до тривалого прихованого функціонування в інфраструктурі жертви. Особливу небезпеку становлять програми-вимагачі (ransomware), які використовують стійкі криптографічні алгоритми для блокування доступу до критичних даних, що безпосередньо загрожує цілісності та доступності інформаційних активів. Сучасне Malware активно застосовує технології обфускації коду та поліморфізму для обходу традиційних засобів захисту, які базуються на сигнатурному аналізі. Крім того, поширення набули безфайлові атаки, що виконуються безпосередньо в оперативній пам'яті, мінімізуючи залишення слідів у файловій системі. Для експертної системи аналіз такої загрози вимагає переходу від статичних методів виявлення до динамічного моніторингу поведінкових аномалій процесів.

1.2.2 Соціальна інженерія та фішинг

Соціальна інженерія репрезентує групу загроз, орієнтованих на експлуатацію людського фактора, який часто виявляється найбільш вразливою ланкою в периметрі безпеки. Фішинг еволюціонував від масових нецільових розсилань до високотехнологічних атак типу «спір-фішинг», які розробляються під конкретну особу чи організацію на основі попереднього

збору розвідувальних даних. Використовуючи методи психологічного тиску та імітацію легітимних запитів від довірених джерел, зловмисники маніпулюють діями користувачів для отримання конфіденційних даних або ініціації шкідливих процесів. Розпізнавання таких загроз в межах експертної системи потребує аналізу контекстуальних параметрів повідомлень, перевірки автентичності джерел та оцінки ризику на основі ідентифікації типових патернів маніпулятивної поведінки [3].

1.2.3 Розподілені атаки на відмову в обслуговуванні (DDoS)

Атаки типу DDoS мають на меті цілеспрямоване порушення доступності інформаційних ресурсів шляхом створення штучного надлишкового навантаження на мережеву інфраструктуру або прикладні сервіси. Сучасні атаки такого типу відрізняються значною потужністю, що досягається за рахунок використання розгалужених ботнетів, які часто складаються з мільйонів скомпрометованих пристроїв інтернету речей (IoT). Окрім атак на рівні об'ємного трафіку, все частіше фіксуються інтелектуальні атаки на рівні додатків (Layer 7), які імітують легітимні запити користувачів і вимагають значно меншої смуги пропускання для виведення системи з ладу. Експертна система підтримки прийняття рішень у цьому випадку повинна забезпечувати оперативний аналіз мережевої телеметрії для швидкого розмежування шкідливого та легітимного трафіку та пропонувати стратегії динамічного перерозподілу ресурсів або фільтрації запитів.

1.2.4 Атаки нульового дня (Zero-day attacks)

Атаки нульового дня представляють категорію найбільш критичних загроз, оскільки вони експлуатують вразливості в програмному забезпеченні, про які ще не відомо розробнику. Відсутність офіційних

патчів та сигнатур робить традиційні системи захисту неефективними проти таких впливів. Непередбачуваність цих атак вимагає від інтелектуальних систем підтримки прийняття рішень застосування проактивних методів аналізу, таких як евристичне моделювання та виявлення відхилень від профілю нормального функціонування системи. Оскільки атаки нульового дня часто є частиною складних цілеспрямованих кампаній кібершпигунства, їх ідентифікація в реальному часі дозволяє мінімізувати потенційні збитки ще до моменту широкого розповсюдження інформації про вразливість.

1.3 Процес прийняття рішень у сфері кібербезпеки

Процес прийняття рішень (ППР) у сфері кібербезпеки є фундаментальною складовою забезпечення стійкості інформаційних систем до зовнішніх та внутрішніх впливів. У сучасних реаліях цей процес трансформувався з дискретних кроків у безперервний цикл аналізу, що здійснюється в умовах високої невизначеності та дефіциту часу. Основна складність ППР полягає у необхідності обробки колосальних обсягів телеметрії, що генерується засобами моніторингу, та зіставлення її з актуальними базами знань про загрози. Когнітивні обмеження людини-оператора часто стають критичним фактором, що призводить до затримок або помилок у реагуванні. Впровадження інтелектуальної підтримки дозволяє формалізувати досвід провідних експертів, забезпечуючи швидке та обґрунтоване прийняття рішень на всіх рівнях управління безпекою.

1.3.1 Реагування на інциденти та управління кризовими ситуаціями

Оперативне реагування на інциденти є найбільш критичним аспектом ППР, де ціна помилки вимірюється цілісністю даних та безперервністю бізнес-процесів. На цьому етапі особа, що приймає рішення, повинна

миттєво класифікувати подію, оцінити масштаб компрометації та обрати відповідну стратегію стримування. Ключова дилема полягає у виборі між негайною ізоляцією атакованого активу, що може призвести до зупинки сервісів, та продовженням спостереження для збору доказової бази та виявлення всіх точок присутності зловмисника.

Процес прийняття рішень у таких ситуаціях доцільно моделювати на основі циклу OODA (спостереження, орієнтація, рішення, дія), де ефективність системи захисту прямо пропорційна швидкості ітерацій. Експертна система дозволяє автоматизувати виконання типових сценаріїв (playbooks), пропонуючи оператору найбільш ймовірні варіанти розвитку подій та відповідні заходи протидії [4].

1.3.2 Оцінка та пріоритезація вразливостей

У сфері управління вразливостями процес прийняття рішень спрямований на раціональний розподіл обмежених ресурсів служби безпеки. Оскільки кількість виявлених вразливостей у великих інфраструктурах зазвичай перевищує технічні можливості персоналу щодо їх негайного усунення, виникає потреба у глибокій пріоритезації. Рішення про черговість встановлення патчів базується не лише на базовій оцінці критичності за шкалою CVSS, а й на контекстуальних факторах: доступності вразливого сервісу з мережі Інтернет, наявності публічних експлойтів та бізнес-цінності активу. Експертна система підтримки прийняття рішень здатна інтегрувати ці різнорідні параметри, вираховуючи динамічний рейтинг ризику. Це дозволяє перейти від простого реагування на технічні слабкості до стратегічного управління ризиками, де рішення приймається на користь усунення тих вразливостей, які мають найбільший потенційний вплив на безпеку організації.

1.3.3 Вибір та оптимізація захисних механізмів

Стратегічний рівень прийняття рішень охоплює питання проектування архітектури безпеки та вибору технічних засобів захисту. Це рішення вимагає комплексного порівняння різнорідних засобів, таких як системи виявлення вторгнень, антивірусні рішення нового покоління (EDR/XDR) та засоби криптографічного захисту. Процес вибору базується на аналізі ефективності механізму проти актуальних моделей загроз у поєднанні з оцінкою вартості впровадження та експлуатації. Важливим аспектом ППР тут є розрахунок показника повернення інвестицій у безпеку (ROSI), який визначається як:

$$ROSI = \frac{(ALE \times P) - Cost}{Cost}, \quad (1.1)$$

де *ALE* – очікувані річні збитки;

P – відсоток зниження ризику після впровадження засобу;

Cost – вартість механізму захисту.

Експертна система допомагає змоделювати різні конфігурації захисту, прогнозуючи рівень залишкового ризику для кожної з них, що дозволяє керівництву приймати обґрунтовані рішення щодо інвестування в кібербезпеку [5].

1.4 Проблематика людського фактора в управлінні кібербезпекою

Незважаючи на стрімкий прогрес у розвитку технологій захисту інформації, центральним елементом контуру прийняття рішень історично залишалася людина. Однак сучасна парадигма кіберзагроз демонструє фундаментальну суперечність між біологічними обмеженнями когнітивних здібностей оператора та експоненційним зростанням складності цифрового середовища.

Аналіз проблематики свідчить, що самостійне прийняття оптимальних рішень фахівцем стає неможливим через вплив низки об'єктивних факторів. Це створює стратегічну вразливість, оскільки ефективність системи безпеки починає залежати не від технічної досконалості засобів захисту, а від фізіологічних та психологічних меж персоналу.

1.4.1 Проблема надвеликих обсягів даних (Big Data) та когнітивне перевантаження

Сучасні системи моніторингу безпеки, такі як SIEM (Security Information and Event Management), IDS/IPS та міжмережеві екрани, генерують колосальні масиви телеметрії.

У середньому корпоративна інфраструктура може продукувати мільйони записів логів та подій безпеки щоденно. Для людини-аналітика обробка такого потоку інформації створює ефект критичного когнітивного перевантаження. Обмежений обсяг оперативної пам'яті мозку не дозволяє утримувати та корелювати тисячі розрізнених фактів для виявлення прихованих патернів складних цілеспрямованих атак (APT) [6].

Внаслідок інформаційного шуму виникає небезпечне явище, відоме як «втома від сповіщень» (alert fatigue). Через велику кількість хибнопозитивних спрацьовувань аналітик починає підсвідомо ігнорувати попередження або переглядати їх поверхнево, що призводить до пропуску реальних інцидентів.

Статистичні методи аналізу показують, що зі зростанням обсягу вхідних даних якість прийняття рішень людиною знижується за нелінійним законом, тоді як машинний інтелект здатний ефективно масштабуватися відповідно до навантаження.

1.4.2 Швидкість змін та асиметрія часу реакції

Критичним фактором сучасної кібервійни є часовий розрив між виникненням загрози та реакцією на неї. Кібератаки сьогодні є високоавтоматизованими процесами: шкідливе програмне забезпечення, черв'яки та ботнети діють зі швидкістю виконання машинних інструкцій, розповсюджуючись мережею за мілісекунди. Натомість цикл прийняття рішення людиною включає етапи виявлення аномалії, когнітивного усвідомлення проблеми, аналізу альтернатив та фізичної реалізації контрзаходів, що сумарно може займати від хвилин до годин.

В умовах атаки типу «нульового дня» або стрімкого шифрування даних вірусом-вимагачем кожна секунда зволікання призводить до геометричної прогресії збитків. Людина фізіологічно не здатна конкурувати зі швидкістю програмних алгоритмів атаки. Ця асиметрія зумовлює необхідність впровадження експертних систем, здатних реагувати в режимі реального часу (Real-time), автоматично блокуючи вектори атаки ще до того, як оператор встигне усвідомити факт інциденту.

1.4.3 Складність інфраструктурних взаємозв'язків

Сучасні інформаційні системи характеризуються високим рівнем гетерогенності, об'єднуючи хмарні сервіси, локальні сервери, мобільні пристрої, елементи Інтернету речей (IoT) та успадковане обладнання (legacy systems). Взаємозв'язки між цими компонентами часто є нелінійними, неочевидними та динамічно змінюваними. Приймаючи рішення про ізоляцію певного вузла чи блокування порту для локалізації загрози, адміністратор не завжди може передбачити каскадні наслідки для суміжних систем та бізнес-процесів.

Людський мозок має обмежену здатність до ментального моделювання складних мережевих топологій. Спроба інтуїтивного

вирішення проблеми в такій заплутаній архітектурі часто призводить до помилок конфігурації, коли засоби захисту завдають організації більшої шкоди, ніж сама атака (наприклад, ненавмисна зупинка критичного виробничого конвеєра). Експертна система ж здатна миттєво прорахувати граф залежностей і запропонувати безпечний варіант реагування.

1.4.4 Суб'єктивність та психологічні фактори

Окрім технічних обмежень, процес прийняття рішень людиною суттєво залежить від психоемоційного стану. Ситуація реального кіберінциденту неминуче супроводжується високим рівнем стресу та відповідальності. У такому стані фахівець стає схильним до когнітивних викривлень, таких як «тунельний зір» (фокусування на одній деталі при ігноруванні загальної картини) або «упередженість підтвердження» (пошук доказів, що підтверджують попередню гіпотезу, та ігнорування фактів, що їй суперечать).

Також на якість рішень впливають втома, час доби та рутинність завдань. Експертна система, на відміну від людини, позбавлена емоційного впливу та суб'єктивності. Вона забезпечує стабільно високу якість аналізу та суворе дотримання політик безпеки незалежно від зовнішніх обставин, що є вирішальним аргументом на користь автоматизації процесів підтримки прийняття рішень.

1.5 Концепції експертних систем (ЕС) та систем підтримки прийняття рішень (СППР)

В умовах інформаційного перевантаження та необхідності оперативної обробки складних даних, традиційні методи управління стають недостатньо ефективними. Це зумовлює перехід від систем, що просто реєструють та зберігають дані, до інтелектуальних систем, здатних

генерувати нове знання та рекомендувати оптимальні варіанти дій. Розуміння теоретичного базису Систем Підтримки Прийняття Рішень (СППР) та Експертних Систем (ЕС) є необхідною передумовою для проєктування ефективного засобу кіберзахисту.

1.5.1 Системи підтримки прийняття рішень (СППР)

Системи підтримки прийняття рішень (Decision Support Systems – DSS) визначаються як інтерактивні автоматизовані системи, що допомагають особам, які приймають рішення (ОПР), використовувати дані, документи, знання та математичні моделі для ідентифікації та вирішення проблем, а також прийняття рішень. Ключовою особливістю СППР є їхня орієнтація на роботу зі слабоструктурованими або неструктурованими задачами. Це задачі, де неможливо побудувати чіткий алгоритм розв’язання через невизначеність, неповноту даних або наявність якісних (нечислових) факторів, що характерно для сфери кібербезпеки.

Головна мета СППР полягає не в автоматичній заміні людини, а в посиленні її інтелектуальних можливостей. Система бере на себе рутинні операції з обробки інформації, моделювання наслідків та порівняння альтернатив, надаючи ОПР обґрунтовані рекомендації. У класифікації СППР виділяють системи, керовані даними (Data-driven), моделями (Model-driven) та знаннями (Knowledge-driven). Саме останній тип, який використовує бази знань для формулювання висновків, є перехідною ланкою до експертних систем.

1.5.2 Експертні системи (ЕС)

Експертні системи (Expert Systems) являють собою клас систем штучного інтелекту, призначених для вирішення складних задач у вузькоспеціалізованій предметній галузі шляхом імітації логіки міркувань

людини-експерта. На відміну від класичних СППР, які часто виступають пасивним інструментом аналізу, ЕС здатні не лише надавати рекомендації, а й пояснювати хід своїх міркувань, що є критично важливим для довіри до системи.

Фундаментальною особливістю архітектури ЕС є чітке відокремлення бази знань (де зберігаються факти та правила «Якщо-Тоді») від машини висновків (програмного механізму, що застосовує ці правила до вхідних даних). Це дозволяє легко масштабувати та оновлювати систему без переписання програмного коду: для навчання системи новим типам кібератак достатньо додати нові правила в базу знань, не змінюючи алгоритм висновку. Таким чином, Експертна Система фактично є різновидом інтелектуальної СППР, де «підтримка» реалізується через надання готового експертного висновку.

1.5.3 Відмінності від звичайних інформаційних систем

Розуміння специфіки СППР та ЕС неможливе без порівняння їх із традиційними інформаційними системами (ІС), такими як системи обробки транзакцій (TPS) або управлінські інформаційні системи (MIS). Відмінності лежать у площині мети, природи задач та методів обробки інформації.

По-перше, відрізняється мета функціонування. Звичайні ІС орієнтовані на підвищення ефективності обробки даних: їхнє завдання – відповісти на питання «Що сталося?» або «Який поточний стан?». Вони фіксують факти (наприклад, логи доступу). Натомість СППР та ЕС орієнтовані на ефективність прийняття рішення: вони відповідають на питання «Що робити?» та «Що буде, якщо?», генеруючи рекомендації на основі наявних фактів.

По-друге, існує принципова різниця у характері алгоритмів. Традиційні ІС базуються на детермінованих алгоритмах: одні й ті самі вхідні дані завжди призводять до одного й того ж результату за чітко визначеною

формулою. Експертні системи використовують евристичні та ймовірнісні методи. Вони оперують знаннями, які можуть бути неповними або нечіткими (fuzzy logic), і здатні знаходити задовільне рішення навіть в умовах невизначеності, що недосяжно для стандартного процедурного програмування.

По-третє, відрізняється взаємодія з користувачем. У звичайних системах користувач отримує фіксовані звіти. У СППР та ЕС взаємодія є ітеративною та діалоговою: користувач може уточнювати запити, вводити додаткові параметри «на льоту» та запитувати пояснення («Чому система вирішила, що це DDoS-атака?»). Це перетворює систему з «калькулятора» на інтелектуального партнера [7].

1.6 Архітектура та основні компоненти експертної системи

Ефективність функціонування експертної системи (ЕС) визначається її архітектурною побудовою, яка забезпечує чітке розмежування між предметними знаннями та алгоритмами їх обробки. На відміну від традиційного процедурного програмування, де логіка жорстко зашита в код, архітектура ЕС базується на модульному принципі. Типова структура складається з чотирьох ключових елементів: Бази знань, Машини логічного висновку, Інтерфейсу користувача та Підсистеми набуття знань. Взаємодія цих компонентів дозволяє моделювати процес мислення експерта при вирішенні задач кібербезпеки.

На рисунку 1.1 наведено концептуальну схему архітектури експертної системи.

1.6.1 База знань (Knowledge Base)

База знань виступає центральним сховищем інтелектуального ресурсу системи, визначаючи її компетентність та здатність вирішувати поставлені

задачі. На відміну від звичайних баз даних, які зберігають лише пасивну інформацію, база знань акумулює формалізований досвід експертів, представлений у вигляді фактів та правил. У контексті кібербезпеки наповнення цього модуля є гетерогенним і поділяється на декларативну та процедурну складові. Декларативні знання описують статичний стан об’єкта захисту, включаючи топологію мережі, перелік активів, відомі вразливості та конфігурації обладнання. Процедурні знання, своєю чергою, моделюють логіку прийняття рішень та причинно-наслідкові зв'язки, найчастіше реалізуючись через продукційні правила типу «Якщо-Тоді». Саме якість, повнота та несуперечливість інформації у базі знань є лімітуючим фактором ефективності всієї системи, оскільки навіть досконалий алгоритм висновку не зможе згенерувати вірну рекомендацію на основі хибних або неповних передумов.

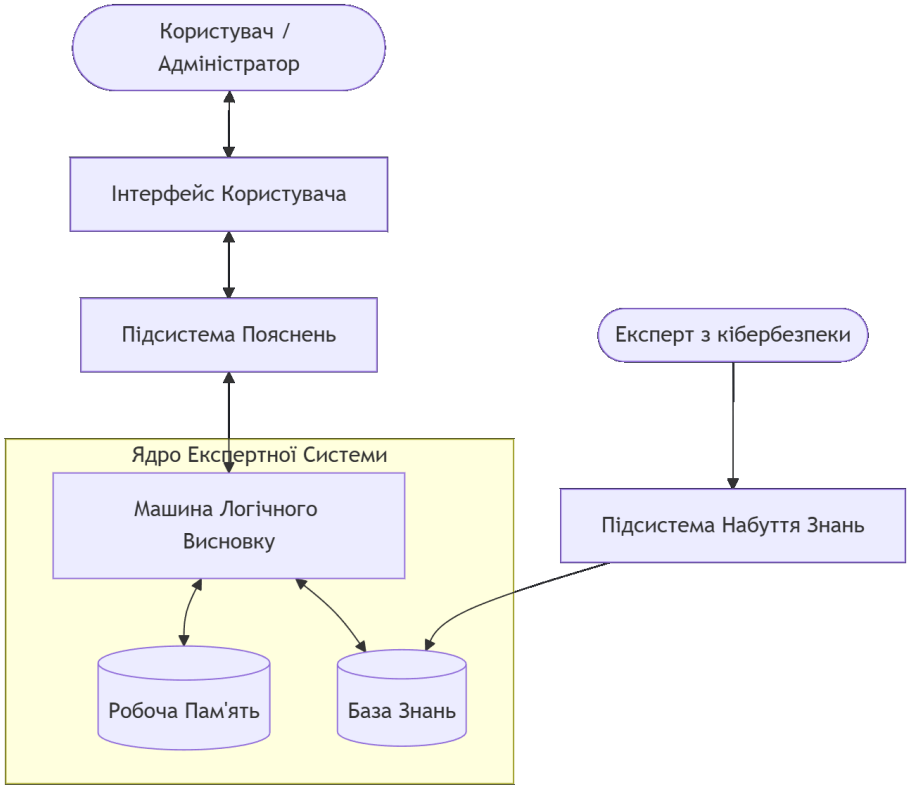

Рисунок 1.1 – Концептуальна схема архітектури експертної системи

1.6.2 Машина логічного висновку (Inference Engine)

Машина логічного висновку, яку також називають інтерпретатором або вирішувачем, є програмним ядром системи, що відповідає за моделювання ходу міркувань експерта. Функціонально цей модуль виступає диспетчером, який циклічно аналізує поточний стан робочої пам'яті, де зберігаються дані про інцидент, та зіставляє їх із правилами, закладеними в базу знань. Процес роботи машини висновку полягає у виборі та активації тих правил, умови яких задовольняються поточними фактами. Залежно від характеру задачі, машина може застосовувати стратегію прямого висновку, рухаючись від виявлених симптомів атаки до встановлення діагнозу, що є характерним для систем моніторингу реального часу. Альтернативно може використовуватися зворотний висновок, коли система перевіряє гіпотезу про можливий злом, намагаючись знайти підтверджуючі факти, що є ефективним методом при проведенні цифрових розслідувань. Важливою характеристикою машини висновку є її інваріантність щодо предметної галузі, що дозволяє використовувати один і той самий програмний рушій для різних задач захисту інформації.

1.6.3 Інтерфейс користувача (User Interface)

Інтерфейс користувача у системах підтримки прийняття рішень виходить далеко за межі простого графічного відображення інформації, виконуючи роль інтелектуального посередника між людиною та машинним алгоритмом. Його основним завданням є забезпечення ефективної людино-машинної взаємодії, що дозволяє оператору вводити опис проблеми, коригувати вхідні параметри та отримувати результати аналізу у зрозумілій формі. Критично важливою складовою інтерфейсу є інтегрована підсистема пояснень (Explanation Facility). Вона надає можливість трасувати процес прийняття рішення, відповідаючи на запитання «Чому система надала таку

рекомендацію?» та «Як було отримано цей висновок?». У сфері кібербезпеки, де ціна помилки є надзвичайно високою, наявність прозорого механізму пояснень є обов'язковою вимогою, оскільки це дозволяє аналітику верифікувати логіку системи та уникнути «ефекту чорної скриньки», підвищуючи загальний рівень довіри до автоматизованого помічника.

1.6.4 Підсистема набуття знань (Knowledge Acquisition Subsystem)

Підсистема набуття знань є інструментальним засобом, що забезпечує життєвий цикл експертної системи та її адаптацію до змін у зовнішньому середовищі загроз. Оскільки методи кібератак еволюціонують надзвичайно швидко, база знань потребує постійного оновлення та розширення. Цей модуль надає інженеру зі знань або експерту можливість додавати нові правила та факти без необхідності втручання в програмний код ядра системи. Функціонал підсистеми включає засоби редагування правил, перевірки їх на синтаксичну коректність, а також виявлення логічних конфліктів та суперечностей з уже існуючими знаннями. У найбільш досконалих реалізаціях підсистема набуття знань може інтегрувати методи машинного навчання для напівавтоматичного вилучення закономірностей з історичних логів інцидентів, трансформуючи їх у нові евристичні правила для бази знань, що значно прискорює процес навчання системи.

1.7 Методи представлення знань в експертних системах

Ефективність функціонування експертної системи безпосередньо залежить від обраного формалізму представлення знань, який визначає спосіб кодування людського досвіду у структуру даних, придатну для машинної обробки. Проблема представлення знань полягає у необхідності перетворення нечітких, інтуїтивних та часто суперечливих уявлень експерта

про предметну галузь у строгу логічну модель. У сфері кібербезпеки, де об'єкти захисту та загрози мають складну структуру та численні взаємозв'язки, вибір методу представлення визначає не лише швидкість роботи машини висновку, а й здатність системи адаптуватися до нових викликів. Найбільш поширеними підходами до формалізації знань є продукційні правила, семантичні мережі та фреймові моделі [8].

1.7.1 Продукційні правила (If-Then Rules)

Продукційна модель є найбільш розповсюдженим та інтуїтивно зрозумілим методом представлення знань у системах діагностики та моніторингу, до яких належать і системи кібербезпеки. Основою цієї моделі є правила, що мають структуру імплікації: «ЯКЩО (умова), ТОДІ (дія/висновок)». Умова правила називається антецедентом і являє собою логічний вираз, який перевіряє наявність певних фактів у робочій пам'яті системи. Дія, або консеквент, визначає операцію, яку необхідно виконати, або новий факт, який слід додати до бази знань у разі істинності умови.

Популярність продукційних правил у сфері захисту інформації пояснюється їхньою природною відповідністю логіці роботи засобів безпеки, таких як міжмережеві екрани, IDS/IPS та SIEM-системи, де рішення про блокування трафіку приймається на основі збігу сигнатур. Важливою перевагою цього підходу є модульність, яка дозволяє додавати, змінювати або видаляти окремі правила без необхідності перегляду всієї бази знань. Це забезпечує високу гнучкість системи, дозволяючи оперативно реагувати на нові вразливості шляхом простого дописування нових продукцій. Однак при значному зростанні кількості правил може виникнути проблема узгодженості та непрозорості взаємозв'язків між ними, що вимагає ретельного проектування механізму управління конфліктами.

1.7.2 Семантичні мережі (Semantic Networks)

Семантичні мережі пропонують графічний підхід до представлення знань, де предметна галузь моделюється у вигляді орієнтованого графа. Вершини цього графа відповідають поняттям, об'єктам, подіям або властивостям, а дуги відображають семантичні відношення між ними. Найчастіше використовуються відношення типу «є частиною» (is-a), «має властивість» (has-property) або «спричиняє» (causes). У контексті кібербезпеки семантичні мережі є надзвичайно ефективними для моделювання топології інформаційної інфраструктури та аналізу впливу загроз.

За допомогою цього формалізму можна описати складні залежності між активами, наприклад: «Сервер А (вершина) забезпечує роботу Сервісу Б (вершина), Сервіс Б має Вразливість В (вершина), Вразливість В експлуатується Експлойтом Г (вершина)». Таке представлення дозволяє машині висновку здійснювати пошук шляхів по графу, визначаючи можливі вектори атак та прогнозуючи каскадні наслідки інциденту для пов'язаних систем. Головною перевагою семантичних мереж є наочність та здатність відображати структурні зв'язки, проте їх обробка вимагає складніших алгоритмів пошуку порівняно з лінійним перебором правил.

1.7.3 Фрейми (Frames)

Фреймова модель представлення знань, запропонована Марвіном Мінським, базується на концепції структурування інформації навколо стереотипних ситуацій або об'єктів. Фрейм являє собою абстрактний образ або структуру даних, що містить слоти (поля) для опису атрибутів об'єкта та приєднані процедури для їх обробки. Кожен слот може містити конкретне значення, значення за замовчуванням, посилання на інший фрейм або процедуру-демон, що запускається при зверненні до слоту.

У розробці експертних систем для кібербезпеки фрейми ідеально підходять для класифікації інцидентів та опису профілів загроз. Наприклад, фрейм «DDoS-атака» може містити слоти: «Джерело», «Тип трафіку», «Ціль», «Інтенсивність», «Рекомендована протидія». Важливою властивістю фреймових систем є механізм спадкування, який дозволяє створювати ієрархії понять від загальних до конкретних. Це дає можливість системі економити пам'ять та автоматично виводити властивості нових об'єктів на основі їх приналежності до певного класу. Поєднання фреймів для опису структури об'єктів та продукційних правил для опису логіки їх взаємодії часто використовується при побудові гібридних експертних систем, що дозволяє нівелювати недоліки кожного з методів окремо.

1.8 Огляд та аналіз існуючих рішень автоматизації кіберзахисту

Сучасний ринок засобів забезпечення інформаційної безпеки характеризується високим рівнем насиченості та фрагментації. Існує широкий спектр програмних продуктів, які частково реалізують функції підтримки прийняття рішень, починаючи від систем моніторингу подій і закінчуючи платформами автоматизованого реагування. Для обґрунтування доцільності розробки власної експертної системи необхідно провести детальний аналіз функціональних можливостей, переваг та обмежень існуючих класів рішень, серед яких ключове місце займають SIEM-системи, платформи класу SOAR та сканери вразливостей.

1.8.1 Системи управління подіями та інформацією про безпеку (SIEM)

Найбільш поширеним класом рішень для централізованого збору та аналізу даних безпеки є системи SIEM (Security Information and Event Management), яскравими представниками яких є Splunk Enterprise Security, IBM QRadar та ELK Stack (Elasticsearch, Logstash, Kibana). Основна

парадигма функціонування цих систем полягає в агрегації лог-файлів з різнорідних джерел, їх нормалізації та кореляції для виявлення аномалій.

Незважаючи на потужні аналітичні можливості, класичні SIEM-системи мають суттєві обмеження в контексті підтримки прийняття рішень. По-перше, вони орієнтовані переважно на діагностику (повідомлення про те, що сталося), а не на проскрипцію (рекомендацію, що робити). Оператор SIEM отримує повідомлення про інцидент, але рішення про метод реагування залишається повністю на його розсуд. По-друге, проблема високого рівня хибнопозитивних спрацьовувань (false positives) вимагає від аналітика значних часових витрат на фільтрацію шуму. Впровадження та налаштування правил кореляції у таких системах вимагає глибокої експертизи та постійної підтримки, що робить їх важкодоступними для організацій малого та середнього бізнесу.

1.8.2 Платформи оркестрації, автоматизації та реагування (SOAR)

Еволюційним розвитком ідей автоматизації стало виникнення класу рішень SOAR (Security Orchestration, Automation and Response), таких як Palo Alto Cortex XSOAR (раніше Demisto) або Splunk Phantom. Ці платформи націлені на вирішення проблеми розриву між виявленням загрози та реагуванням на неї. Головною перевагою SOAR є можливість створення «плейбуків» (playbooks) – автоматизованих сценаріїв, які виконують послідовність дій без участі людини (наприклад, ізоляція хоста при виявленні вірусу).

Однак, з точки зору експертних систем, SOAR-рішенням часто бракує інтелектуальної гнучкості. Їхня логіка зазвичай є жорстко детермінованою і базується на лінійних алгоритмах. Якщо ситуація виходить за межі прописаного сценарію, система зупиняється і чекає на втручання оператора. Крім того, вартість ліцензування комерційних SOAR-платформ є надзвичайно високою, а їх інтеграція вимагає наявності зрілої

інфраструктури безпеки, що обмежує їх застосування у навчальних цілях або в невеликих компаніях [9].

1.8.3 Системи управління вразливостями (Vulnerability Management)

Окремий сегмент ринку займають спеціалізовані сканери вразливостей, такі як Tenable Nessus, Rapid7 InsightVM або OpenVAS. Ці інструменти є незамінними для етапу ідентифікації технічних слабкостей системи. Вони використовують великі бази даних відомих вразливостей (CVE) для сканування мережевих вузлів та формування детальних звітів.

Проблематика використання цих засобів у контексті прийняття рішень полягає у відсутності контекстуалізації. Сканер може видати звіт на сотні сторінок із тисячами вразливостей, класифікованих лише за технічною шкалою CVSS. Він не враховує бізнес-значимість активу, наявність засобів захисту периметра або реальну ймовірність експлуатації в конкретному середовищі. В результаті адміністратор стикається з проблемою пріоритезації: система надає дані, але не надає знання, необхідні для прийняття оптимального рішення про черговість встановлення патчів.

1.8.4 Висновок та обґрунтування необхідності власної розробки

Проведений аналіз дозволяє зробити висновок, що, попри наявність потужних інструментів моніторингу та автоматизації, на ринку існує певний вакуум у ніші інтелектуальних систем підтримки прийняття рішень, доступних для широкого кола користувачів. Комерційні гіганти (SIEM/SOAR) є занадто дорогими та складними, а відкриті інструменти (OpenVAS, Snort) надають лише «сирі» дані, вимагаючи від користувача високої кваліфікації для їх інтерпретації.

Розробка спеціалізованої експертної системи, яка поєднувала б у собі механізми збору даних із гнучкою базою знань та пояснювальним інтерфейсом, дозволить вирішити проблему «інтелектуального розриву». Така система зможе не лише автоматизувати рутинні процеси, а й акумулювати експертний досвід для допомоги менш кваліфікованим фахівцям, забезпечуючи при цьому прозорість прийнятих рішень, чого часто бракує алгоритмам «чорної скриньки» у комерційних продуктах [10].

2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

2.1 Постановка задачі та визначення вимог до системи

Першим етапом проєктування експертної системи підтримки прийняття рішень є формалізація задачі та формування вичерпного переліку вимог, що визначають функціональний профіль майбутнього програмного продукту.

Метою розробки є створення автоматизованого інструментарію, здатного аналізувати дані про стан інформаційної безпеки, виявляти потенційні загрози на основі евристичних правил та надавати користувачеві рекомендації щодо їх нейтралізації. Об'єктом автоматизації виступає діяльність адміністратора безпеки або аналітика SOC (Security Operations Center), пов'язана з обробкою інцидентів та оцінкою ризиків.

2.1.1 Формалізація задачі

З математичної точки зору, функціонування розроблюваної системи можна представити як відображення множини вхідних векторів станів системи (подій безпеки, логів, результатів сканування) у множину вихідних рішень (рекомендацій).

Задача системи полягає у знаходженні такого рішення з множини альтернатив, яке максимізує показник захищеності інфраструктури при мінімальних витратах ресурсів. Система повинна працювати в умовах неповної інформації, використовуючи базу знань для заповнення прогалин у вхідних даних шляхом логічного висновку.

Ключовим завданням є мінімізація часу між виявленням ознак аномальної активності та генерацією валідного плану реагування.

2.1.2 Функціональні вимоги

На основі аналізу предметної галузі, проведеного у попередньому розділі, було визначено ключові функції, які повинна реалізовувати система. Насамперед програмний комплекс має забезпечувати можливість збору та первинної обробки вхідних даних. Це передбачає наявність парсера, здатного трансформувати неструктуровані текстові логи (наприклад, з веб-сервера або фаєрволу) у структурований формат, придатний для машинного аналізу. Наступною критичною вимогою є реалізація механізму ідентифікації загроз шляхом зіставлення отриманих даних із правилами бази знань. Система повинна вміти класифікувати подію (наприклад, розрізняти легітимний сплеск трафіку та DDoS-атаку) та визначати рівень її критичності.

Результатом роботи системи має бути не просто констатація факту загрози, а генерація конкретних рекомендацій. Вимоги до цього блоку включають надання чіткого алгоритму дій для користувача (наприклад, «заблокувати IP-адресу X», «ізолювати хост Y», «оновити сервіс Z»). Крім того, система повинна мати функцію пояснення прийнятого рішення, відображаючи ланцюжок правил, які призвели до формування конкретного висновку. Для забезпечення адаптивності системи необхідна наявність інтерфейсу адміністратора, що дозволяє додавати, редагувати та видаляти правила в базі знань без необхідності втручання у вихідний код програми.

2.1.3 Нефункціональні вимоги

Окрім функціональних можливостей, проєктована система повинна відповідати низці якісних характеристик. Вимога до продуктивності передбачає, що час обробки одного запиту (аналіз події та видача рекомендації) не повинен перевищувати допустимих меж, що дозволяє використовувати систему в режимі, наближеному до реального часу.

Надійність системи визначається її здатністю коректно обробляти помилкові або неповні вхідні дані, не припиняючи функціонування. Критично важливою є також вимога до ергономічності інтерфейсу: оскільки система призначена для підтримки прийняття рішень у стресових ситуаціях, візуалізація даних має бути інтуїтивно зрозумілою, не перевантажуючи оператора зайвою інформацією. Зважаючи на специфіку предметної галузі, сама експертна система повинна бути захищеною від несанкціонованого доступу, забезпечуючи цілісність бази знань та конфіденційність журналів аудиту.

2.2 Розробка архітектури програмного забезпечення

Проектування архітектури програмного засобу є етапом структурної декомпозиції системи на функціональні компоненти, що дозволяє реалізувати визначені вимоги найбільш ефективним способом. Для розробки експертної системи підтримки прийняття рішень у сфері кібербезпеки обрано модульну архітектуру, побудовану за принципом слабкої зв'язності (loose coupling). Такий підхід забезпечує гнучкість розробки, спрощує тестування окремих блоків та дозволяє в майбутньому масштабувати систему шляхом додавання нових джерел даних або правил без зміни ядра програми. Архітектура системи складається з чотирьох основних рівнів: рівня збору та нормалізації даних, рівня логічної обробки (ядро), рівня зберігання знань та рівня взаємодії з користувачем.

2.2.1 Модуль збору та попередньої обробки даних (Data Ingestion & Pre-processing)

Першим ешелоном архітектури, який відповідає за інтеграцію системи із зовнішнім середовищем, виступає модуль попередньої обробки. Оскільки вхідні дані про події безпеки надходять у різнорідних

форматах (syslog-повідомлення, JSON-об’єкти від вебсерверів, CSV-звіти сканерів), критично важливою задачею є їх уніфікація. Цей модуль виконує функцію парсингу «сирих» даних, виокремлюючи ключові сутності, такі як IP-адреси джерела та призначення, часові мітки, типи протоколів та коди помилок. Після екстракції відбувається процес нормалізації, під час якого дані приводяться до єдиного внутрішнього формату фактів, зрозумілого для машини висновку. Саме на цьому етапі відбувається фільтрація надлишкового інформаційного шуму, що дозволяє знизити навантаження на обчислювальне ядро системи.

2.2.2 Ядро логічного висновку (Inference Engine Core)

Центральне місце в ієрархії компонентів займає модуль логічного висновку, який реалізує алгоритмічну складову експертної системи. Цей модуль функціонує як безперервний цикл, що сканує оперативну пам'ять (робочу пам'ять) на наявність нових фактів. Архітектурно ядро складається з блоку зіставлення зразків (Pattern Matcher), який перевіряє відповідність поточних фактів умовам правил (IF-частина), та блоку виконання (Execution Unit), який ініціює дії, прописані у висновках правил (THEN-частина). Для оптимізації швидкодії планується використання алгоритмів, що дозволяють уникати повного перебору всіх правил на кожній ітерації (наприклад, спрощена реалізація алгоритму Rete), що є критичним для забезпечення роботи в режимі реального часу. Результатом роботи ядра є згенерований об'єкт «Рішення», що містить опис виявленої проблеми, рівень ризику та рекомендований сценарій реагування.

2.2.3 Підсистема управління базою знань (Knowledge Management)

Для забезпечення збереження та цілісності інтелектуального ресурсу системи проектується підсистема управління знаннями. Вона відповідає за

збереження правил у постійній пам'яті (файлова система або база даних) та їх завантаження в оперативну пам'ять під час ініціалізації системи. Архітектурно цей модуль розділяє фізичне зберігання правил (наприклад, у форматі JSON або YAML для зручності редагування людиною) та їх внутрішнє представлення (об'єктна модель). Важливою функцією цього компонента є валідація синтаксису: перед завантаженням нового правила в активне ядро система повинна перевірити його на відповідність структурі та відсутність логічних суперечностей, що запобігає збоям у роботі алгоритму висновку через помилку оператора.

2.2.4 Інтерфейс та візуалізація (Presentation Layer)

Взаємодія користувача із системою реалізується через веб-інтерфейс, побудований за архітектурою клієнт-сервер. Серверна частина (API) приймає запити від фронтенду та передає їх відповідним модулям ядра. Клієнтська частина надає інструменти для моніторингу стану безпеки (Dashboard), де в режимі реального часу відображаються згенеровані рекомендації та попередження. Окремим важливим компонентом інтерфейсу є редактор бази знань, який надає адміністратору графічну оболонку для створення та модифікації правил без необхідності написання коду. Також на цьому рівні реалізується механізм пояснення, який візуалізує для користувача ланцюжок логічних кроків, що призвели до конкретного висновку системи.

На рисунку 2.1 наведено діаграму компонентів, що ілюструє потоки даних у розроблюваній системі.

Така архітектура забезпечує чітке розмежування відповідальності між модулями: парсер «знає», як читати логи, ядро «знає», як думати, а інтерфейс «знає», як показувати. Це значно спрощує процес програмної реалізації та відлагодження системи.

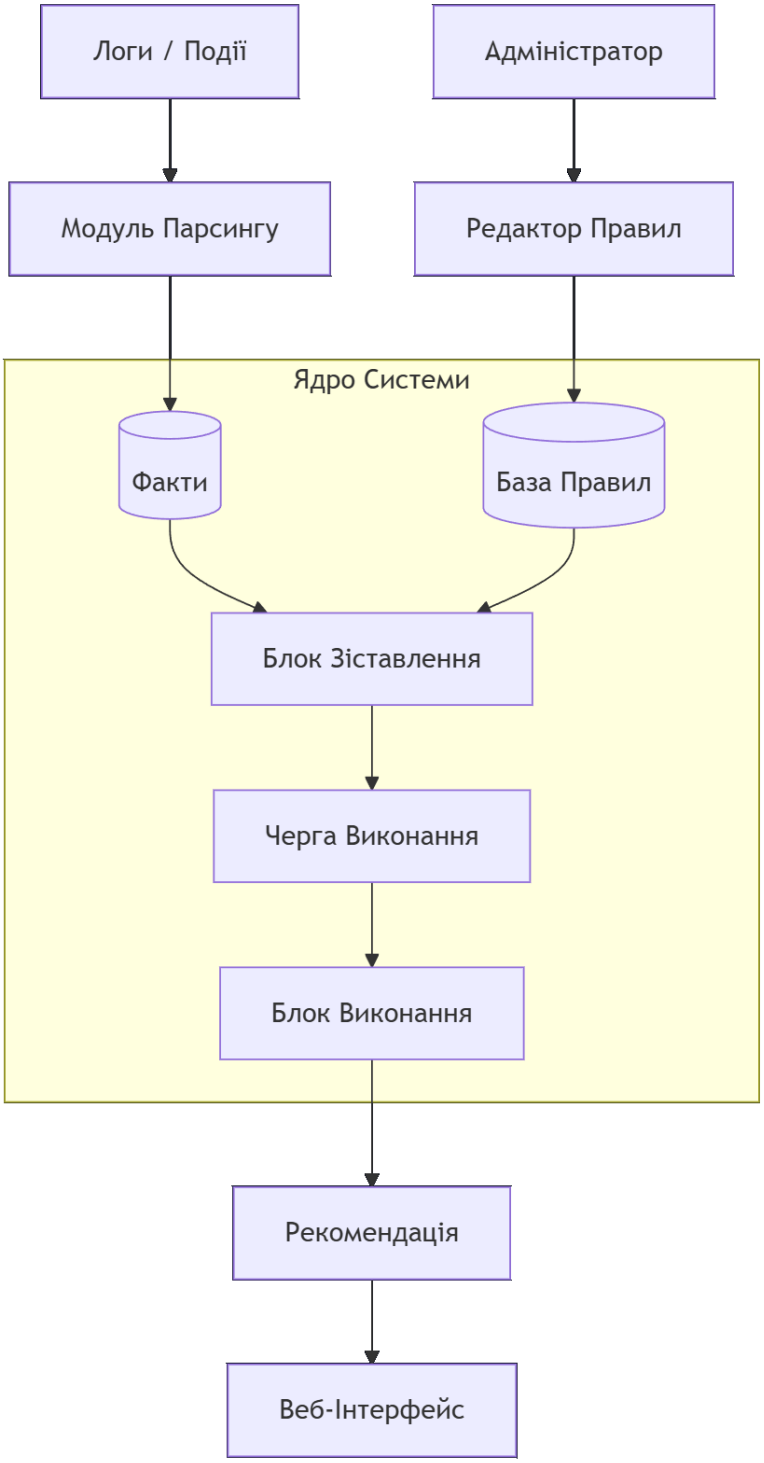

Рисунок 2.1 – Ілюстрація потоків даних у розроблюваній системі

2.3 Моделювання бази знань та структур даних

Етап моделювання бази знань є ключовим у процесі розробки експертної системи, оскільки він визначає спосіб формалізації предметної

галузі та правила, за якими функціонує машина логічного висновку. Метою цього етапу є створення уніфікованої структури даних, яка дозволяє однозначно описати стан інформаційної системи (факти) та алгоритми реагування на загрози (правила). Для реалізації системи обрано продукційну модель представлення знань, адаптовану для збереження у форматі JSON, що забезпечує високу швидкість серіалізації даних та зручність інтеграції з програмними модулями, написаними мовою Python.

2.3.1 Формалізація фактів (модель даних)

Факти в розроблюваній системі виступають атомарними одиницями інформації, що відображають поточний стан середовища безпеки. Кожен факт є наслідком обробки вхідної події (логу) або результатом спрацювання правила. Структурно факт моделюється як об’єкт, що містить набір типізованих атрибутів. Базова модель факту включає унікальний ідентифікатор події, часову мітку (timestamp), що дозволяє відслідковувати хронологію атак, та джерело походження інформації. Специфічні атрибути залежать від типу події: для мережевого трафіку це IP-адреси джерела та призначення, порти та протоколи транспортного рівня; для системних подій – імена користувачів, типи операцій (читання/запис) та шляхи до файлів. Важливим аспектом моделювання є введення атрибуту «час життя» (TTL) факту, що дозволяє автоматично очищувати робочу пам'ять від застарілої інформації, підтримуючи високу продуктивність системи.

2.3.2 Формалізація правил (модель знань)

Правила в системі представляють собою евристичні конструкції, що кодують експертний досвід виявлення та нейтралізації загроз. Логічна структура кожного правила базується на імплікації «ЯКЩО <Умова> ТОДІ <Дія>». Для програмної реалізації розроблено розширену структуру

правила, яка, окрім логічного ядра, містить метадані для управління процесом висновку.

Кожне правило описується набором полів: унікальним ідентифікатором (Rule ID), який використовується для трасування та логування спрацювань; рівнем пріоритету, що визначає черговість виконання у випадку конфлікту правил (коли кілька правил можуть бути застосовані одночасно); та описом (Description), що використовується підсистемою пояснень для надання зрозумілої людині інформації про сутність загрози.

Блок умов (Condition) моделюється як сукупність логічних предикатів, об'єднаних операторами AND/OR. Предикат перевіряє значення конкретного атрибута факту на відповідність заданому критерію (рівність, нерівність, входження в діапазон, регулярний вираз). Блок дій (Action) визначає набір інструкцій, які виконує система при істинності умов. Це може бути генерація нового факту (наприклад, присвоєння мітки «Підозрілий» для IP-адреси), створення сповіщення для адміністратора або запуск зовнішнього скрипта блокування.

2.3.3 Приклад структурної реалізації (JSON)

Для наочної демонстрації розробленої моделі наведемо приклад формалізації правила виявлення спроби підбору пароля (Brute-force attack) у форматі JSON, який буде використовуватися системою (лістинг 2.1).

Лістинг 2.1 Приклад формалізації правила при спробі підбору пароля

```
{
  "rule_id": "SEC_RULE_101",
  "name": "SSH Brute Force Detection",
  "priority": 50,
  "description": " An attempt to guess the password via SSH protocol was detected (more than 5 unsuccessful attempts)",
```

Продовження лістингу 2.1

```
  "conditions": {
    "logic": "AND",
    "predicates": [
      {
        "field": "event_type",
        "operator": "equals",
        "value": "failed_login"
      },
      {
        "field": "protocol",
        "operator": "equals",
        "value": "ssh"
      },
      {
        "field": "count_last_minute",
        "operator": "greater_than",
        "value": 5
      }
    ]
  },
  "actions": [
    {
      "type": "create_alert",
      "severity": "high",
      "message": "Brute force attack detected from IP
{source_ip}"
    },
    {
      "type": "add_fact",
      "fact_name": "malicious_ip",
      "value": "{source_ip}",
      "ttl": 3600
    }
  ]
}
```

Такий підхід до моделювання дозволяє досягти високої гнучкості: додавання нових правил не вимагає перекомпіляції програмного коду, а проста структура JSON дозволяє розробити зручний графічний інтерфейс для редагування бази знань навіть фахівцями без навичок програмування. Використання параметризованих дій (наприклад, {source_ip}) забезпечує динамічність реагування, адаптуючи контрзаходи до конкретних параметрів атаки.

2.4 Вибір та обґрунтування засобів програмної реалізації

Вибір технологічного стека є критичним етапом проєктування, що визначає не лише швидкість розробки прототипу, але й подальшу життєздатність, масштабованість та продуктивність експертної системи. Зважаючи на специфіку предметної галузі, яка вимагає обробки різнорідних даних та реалізації складних логічних алгоритмів, пріоритетними критеріями вибору інструментарію стали: наявність спеціалізованих бібліотек для інформаційної безпеки, підтримка роботи з неструктурованими даними, кросплатформеність та висока швидкість ітеративної розробки. На основі проведеного аналізу було сформовано комплекс засобів реалізації, що включає мову програмування Python, документоорієнтовану систему управління базами даних MongoDB та мікрофреймворк Flask.

2.4.1 Мова програмування Python

Основним інструментом розробки програмного ядра системи обрано мову високого рівня Python. Цей вибір зумовлений домінуванням Python у сфері кібербезпеки та штучного інтелекту, що забезпечує доступ до найширшої екосистеми спеціалізованих бібліотек. Інтерпретована природа мови дозволяє суттєво прискорити процес створення машини логічного

висновку, фокусуючись на алгоритмах обробки знань, а не на низькорівневому управлінні пам'яттю. Крім того, лаконічний синтаксис Python та підтримка різних парадигм програмування (об'єктно-орієнтованої та функціональної) спрощують реалізацію модульної архітектури, описаної у попередніх підрозділах. Для роботи з мережевими протоколами та аналізу трафіку планується використання бібліотек socket та scapy, а для математичних обчислень та маніпуляцій з даними – бібліотеки pandas, яка є стандартом де-факто в аналітиці даних.

2.4.2 Система управління базами даних (NoSQL / MongoDB)

Вибір системи зберігання даних продиктований структурою інформаційних потоків у системах моніторингу безпеки. Оскільки логи від різних джерел (фаєрволів, антивірусів, веб-серверів) мають різну структуру, а правила бази знань моделюються у форматі JSON, використання класичних реляційних баз даних (SQL) із жорсткою схемою створило б надлишкову складність при проєктуванні таблиць та зв'язків. Тому для реалізації підсистеми зберігання обрано документоорієнтовану СУБД MongoDB. Вона нативно підтримує формат BSON (бінарний JSON), що дозволяє зберігати факти та правила саме в тому вигляді, в якому вони використовуються програмним ядром, усуваючи необхідність витратної операції ORM-перетворення (Object-Relational Mapping). Висока швидкість операцій запису в MongoDB є критично важливою перевагою для системи, яка повинна фіксувати потенційно тисячі подій безпеки за короткий проміжок часу без втрати продуктивності.

2.4.3 Веб-фреймворк та інтерфейс користувача

Для реалізації рівня взаємодії з користувачем та візуалізації результатів прийнято рішення використати архітектуру веб-застосунку. В

якості серверної платформи обрано мікрофреймворк Flask. На відміну від ваговагових аналогів (наприклад, Django), Flask надає розробнику мінімалістичне ядро, що дозволяє гнучко конструювати архітектуру API без зайвих залежностей. Це ідеально відповідає задачі створення легкого та швидкого інтерфейсу для експертної системи. Взаємодія між клієнтською частиною (Front-end) та сервером (Back-end) здійснюватиметься за архітектурним стилем REST, що забезпечує універсальність доступу до функцій системи. Для створення клієнтського інтерфейсу будуть використані стандартні веб-технології HTML5, CSS3 та JavaScript, посилені бібліотекою Bootstrap для забезпечення адаптивності дизайну та коректного відображення на різних типах пристроїв.

2.4.4 Середовище розробки та контролю версій

Розробка програмного продукту здійснюватиметься у інтегрованому середовищі (IDE) Visual Studio Code, яке забезпечує потужні засоби статичного аналізу коду та відлагодження. Для управління життєвим циклом програмного коду та забезпечення можливості колективної роботи (або версійності змін) використовуватиметься система контролю версій Git. Це дозволить фіксувати етапи розробки машини висновку та безпечно експериментувати з новими алгоритмами, маючи можливість швидкого повернення до стабільної версії.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Реалізація алгоритмічного ядра штучного інтелекту

Ключовим етапом практичної реалізації є розробка машини логічного висновку (Inference Engine), яка виступає інтелектуальним ядром системи. На відміну від детермінованих алгоритмів, що використовуються у стандартному програмному забезпеченні, розроблене ядро реалізує методи символьного штучного інтелекту. Воно забезпечує відокремлення логіки прийняття рішень від обчислювальних процесів, дозволяючи системі оперувати знаннями, а не лише даними.

3.1.1 Алгоритм циклу розпізнавання-дії (Recognize-Act Cycle)

Функціонування інтелектуального агента базується на циклічній моделі обробки продукційних правил. Реалізований алгоритм працює за стратегією прямого висновку (Forward Chaining), що є оптимальним для задач моніторингу подій у реальному часі. Цикл роботи ядра складається з трьох послідовних фаз:

– зіставлення (Matching): на цьому етапі алгоритм сканує Робочу пам'ять (де зберігаються поточні факти про стан мережі) та порівнює їх з умовами (антецедентами) правил, що зберігаються в Базі знань. Для оптимізації цього процесу та уникнення повного перебору (що є обчислювально затратним при великій кількості правил), реалізовано модифікований механізм зіставлення зі зразком (Pattern Matching). Система індексує правила за типами подій, що дозволяє миттєво відсіювати нерелевантні правила ще до перевірки детальних умов;

– вибір (Selection / Conflict Resolution): у результаті етапу зіставлення формується так звана «конфліктна множина» (Conflict Set) – набір правил, умови яких задоволені поточними фактами. Оскільки одночасне виконання

всіх правил може призвести до суперечностей, впроваджено евристичний алгоритм вирішення конфліктів. Пріоритет надається правилам з вищим рівнем специфічності (більшою кількістю умов) та вищим рейтингом критичності (визначеним у полі priority). Це імітує поведінку експерта, який у критичній ситуації спочатку реагує на найбільш небезпечні симптоми.

– виконання (Action): система виконує дії, прописані у консеквенті (висновку) обраного правила. Це може бути модифікація робочої пам'яті (додавання нового факту, наприклад, «Підозріла активність підтверджена») або ініціація зовнішньої команди (блокування порту). Зміна стану робочої пам'яті автоматично перезапускає цикл, що дозволяє системі будувати ланцюжки висновків.

3.1.2 Реалізація механізму евристичного пошуку

Особливістю розробленого модуля штучного інтелекту є здатність працювати з неповною інформацією. Якщо для спрацювання правила не вистачає певних фактів, система не відкидає гіпотезу, а переводить її у статус «очікування». Реалізовано механізм часових вікон (Time Windows), який дозволяє корелювати події, рознесені у часі.

Наприклад, для виявлення розподіленої атаки (DDoS), AI-модуль не розглядає кожен запит окремо. Він агрегує факти в динамічному буфері пам'яті. Алгоритм відстежує швидкість надходження однотипних фактів і, якщо поріг перевищено, генерує новий факт вищого рівня абстракції – «Аномалія трафіку». Такий підхід дозволяє системі переходити від аналізу технічних даних (IP-пакети) до семантичного розуміння ситуації (тип атаки), що є ознакою інтелектуальної поведінки.

Окрім того, для підвищення адаптивності системи, в алгоритм закладено можливість оновлення ваг правил. Це створює підґрунтя для подальшого впровадження елементів машинного навчання, коли система

зможе самостійно коригувати пріоритети правил на основі зворотного зв'язку від оператора про хибні спрацювання.

3.2 Програмна реалізація модулів та архітектури системи

Програмна реалізація розробленої експертної системи виконана з дотриманням принципів об'єктно-орієнтованого програмування (ООП) мовою Python, що забезпечує високий рівень модульності та інкапсуляції логіки штучного інтелекту. Відповідно до архітектури, спроєктованої у другому розділі, кодова база системи розділена на логічні шари: шар доступу до даних (Data Access Layer), шар бізнес-логіки (Core Logic) та шар представлення (API/Web). Така структура дозволяє ізолювати алгоритми прийняття рішень від технічних деталей зберігання інформації чи візуалізації, що спрощує подальшу підтримку та масштабування програмного комплексу.

3.2.1 Реалізація класів ядра системи

Центральним елементом програмної реалізації є клас InferenceEngine, який виступає програмним втіленням машини логічного висновку. Цей клас ініціалізує робоче середовище, завантажуючи правила з бази знань у внутрішні структури даних. Для представлення знань розроблено спеціалізовані класи Rule та Fact (лістинг 3.1).

Клас Fact реалізує методи нормалізації вхідних даних, перетворюючи «сирі» словники (dictionary) з логів у типізовані об'єкти з атрибутами валідності та часовими мітками. Клас Rule містить логіку перевірки умов: у ньому реалізовано метод evaluate(), який приймає на вхід поточний стан робочої пам'яті та повертає булеве значення істинності.

Така архітектура дозволяє динамічно обчислювати складні логічні вирази, визначені у JSON-структурі правила, використовуючи вбудовані можливості Python для обробки операторів порівняння.

Лістинг 3.1 – Реалізація класів Rule та Fact

```
class Fact:
    """Клас, що представляє одиницю інформації (подію) в системі."""
    def __init__(self, data: dict, source: str):
        self.timestamp = datetime.now()
        self.source = source
        self.attributes = data  # Словник з даними (наприклад, ip, port, protocol)

class Rule:
    """Клас, що представляє продукційне правило."""
    def __init__(self, rule_structure: dict):
        self.id = rule_structure.get('rule_id')
        self.name = rule_structure.get('name')
        self.priority = rule_structure.get('priority', 0)
        self.conditions = rule_structure.get('conditions', {})
        self.actions = rule_structure.get('actions', [])

    def check_conditions(self, fact: Fact) -> bool:
        """Перевіряє, чи відповідає факт умовам правила."""
        logic = self.conditions.get('logic', 'AND')
        predicates = self.conditions.get('predicates', [])

        results = []
        for pred in predicates:
            field = pred['field']
            operator = pred['operator']
            target_value = pred['value']
```

Продовження лістингу 3.1

```
                # Отримуємо значення з факту (якщо атрибут
відсутній, умова False)
                fact_value = fact.attributes.get(field)
                if fact_value is None:
                    results.append(False)
                    continue

                # Порівняння значень (спрощена реалізація
операторів)
                if operator == 'equals':
                    results.append(fact_value == target_value)
                elif operator == 'greater_than':
                    results.append(fact_value > target_value)
                elif operator == 'contains':
                    results.append(target_value in fact_value)

            # Обчислення фінального результату залежно від
логіки (AND/OR)
            if logic == 'AND':
                return all(results)
            elif logic == 'OR':
                return any(results)
            return False
```

Окрему увагу приділено реалізації класу KnowledgeBaseManager, який відповідає за взаємодію з системою управління базами даних MongoDB. Використання бібліотеки pymongo дозволило реалізувати ефективний механізм серіалізації та десеріалізації об'єктів. Оскільки у Розділі 2 було обрано документоорієнтований підхід, цей менеджер завантажує JSON-документи правил безпосередньо в об'єкти Python, оминаючи етап складної трансформації, характерний для SQL-систем. Це забезпечує високу швидкодію системи при старті та оновленні бази знань «на льоту».

3.2.2 Програмна реалізація циклу висновку

Алгоритм зіставлення (Pattern Matching), описаний у теоретичній частині, реалізовано всередині методу run_cycle() класу InferenceEngine (лістинг 3.2). Цей метод виконує ітеративний прохід по списку активних правил.

Для оптимізації продуктивності впроваджено механізм попередньої фільтрації: система використовує хеш-таблиці для миттєвого пошуку правил, що стосуються конкретного типу події (наприклад, ssh_login або http_request), ігноруючи решту масиву бази знань.

Лістинг 3.2 – Реалізація машини логічного висновку (Inference Engine)

```
class InferenceEngine:
    def __init__(self):
        self.knowledge_base = []  # Список об'єктів Rule
        self.active_facts = []    # Робоча пам'ять

    def load_rules(self, db_manager):
        """Завантажує правила з MongoDB через менеджер БД."""
        rules_data = db_manager.get_all_rules()
        self.knowledge_base = [Rule(r) for r in rules_data]
        # Сортування правил за пріоритетом (найвищий пріоритет - перший)
        self.knowledge_base.sort(key=lambda x: x.priority, reverse=True)

    def analyze_event(self, event_data: dict) -> dict:
        """
        Основний цикл висновку (Forward Chaining).
        Приймає "сирі" дані, створює Факт і шукає відповідні правила.
        """
```

Продовження лістингу 3.2

```
            current_fact = Fact(event_data, source="SIEM_Collector")
            self.active_facts.append(current_fact)

            matched_rule = None

            # Етап зіставлення (Pattern Matching)
            for rule in self.knowledge_base:
                if rule.check_conditions(current_fact):
                    matched_rule = rule
                    break    # Знайдено правило з найвищим пріоритетом (Conflict Resolution)

            # Етап виконання (Action)
            if matched_rule:
                return {
                    "status": "ALERT",
                    "rule_triggered": matched_rule.name,
                    "description": matched_rule.id,
                    "actions": matched_rule.actions,
                    "timestamp": current_fact.timestamp.isoformat()
                }

            return {"status": "SAFE", "message": "No threats detected"}
```

Коли умови правила виконуються, система ініціює створення екземпляра класу Action. Цей клас реалізує патерн «Command», що дозволяє інкапсулювати всі необхідні параметри дії (блокування IP, надсилання сповіщення) в один об'єкт. Важливою особливістю реалізації є те, що виконання дій відбувається асинхронно, щоб не блокувати основний потік аналізу подій. Це досягається використанням модуля threading або

асинхронних можливостей фреймворку, що дозволяє системі продовжувати моніторинг навіть під час виконання тривалих операцій реагування.

3.2.3 Реалізація API та інтеграція

Для забезпечення взаємодії із зовнішнім світом реалізовано RESTful API на базі мікрофреймворку Flask. Основною точкою входу є ендпоінт /api/analyze, який приймає POST-запити з даними про події у форматі JSON. Обробник цього запиту виконує валідацію вхідних даних, створює новий об'єкт Fact і передає його в чергу обробки ядра InferenceEngine.

Варто зазначити, що інтерфейс користувача (Dashboard), реалізований на HTML/JavaScript, взаємодіє з сервером виключно через це API. Це підтверджує дотримання принципу слабкої зв'язності компонентів. Крім того, реалізовано механізм WebSockets (через розширення Flask-SocketIO) для забезпечення двостороннього зв'язку в реальному часі. Це дозволяє серверу миттєво «пушити» (push) сповіщення про виявлені загрози на екран адміністратора без необхідності постійного оновлення сторінки браузера, що є критичним для оперативного реагування на інциденти кібербезпеки.

Таким чином, програмна реалізація повністю відповідає спроектованій архітектурі, забезпечуючи надійну платформу для функціонування інтелектуальних алгоритмів аналізу загроз.

3.2.4 Реалізація підсистеми збору та нормалізації даних (Parser Module)

Першим етапом обробки інформації в розробленій системі є трансформація неструктурованих або напівструктурованих даних у уніфікований формат фактів. Цю задачу вирішує клас LogParser, реалізований з використанням модуля re (Regular Expressions) стандартної

бібліотеки Python (лістинг 3.3). Необхідність розробки власного парсера зумовлена гетерогенністю джерел подій: системні журнали Linux (syslog) мають один формат, веб-сервер Nginx – інший, а сучасні IDS-системи часто віддають дані у JSON.

Програмна логіка модуля базується на використанні бібліотеки патернів (Regex Patterns). При ініціалізації система завантажує набір регулярних виразів, кожен з яких відповідає певному типу події. Процес обробки вхідного рядка відбувається методом послідовного зіставлення: парсер намагається застосувати шаблони до вхідного рядка, і у випадку збігу екстрагує ключові сутності (групи захоплення).

Критично важливою функцією цього модуля є нормалізація даних. Різні системи можуть називати одні й ті самі сутності по-різному (наприклад, src_ip, sourceAddress, origin_ip). Парсер приводить ці розрізнені ключі до єдиного внутрішнього стандарту (наприклад, source_ip), що дозволяє правилам у базі знань бути універсальними та не залежати від специфіки джерела даних.

Лістинг 3.3 – Реалізація класу LogParser та нормалізації

```
import re
import json

class LogParser:
    """
    Модуль для перетворення сирих логів у структуровані події.
    """
    def __init__(self):
        # Компіляція регулярних виразів для підвищення продуктивності
        self.patterns = {
            'ssh_failed': re.compile(
```

Продовження лістингу 3.3

```
                    r'Failed password for (?P<user>[\w-]+) from
(?P<source_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})          port
(?P<port>\d+) (?P<protocol>\w+)'
                ),
                'nginx_access': re.compile(
                    r'(?P<source_ip>\d+\.\d+\.\d+\.\d+)   -   -
\[.*\]       "(?P<method>\w+)       (?P<url>.*)       HTTP/.*"
(?P<status_code>\d+) .*'
                )
            }

        def parse_line(self, log_line: str) -> dict:
            """
            Аналізує рядок логу та повертає нормалізований
словник даних.
            """
            # Спроба зіставити рядок з відомими шаблонами
            for event_type, pattern in self.patterns.items():
                match = pattern.search(log_line)
                if match:
                    #  Екстракція даних (перетворення named
groups у dict)
                    normalized_data = match.groupdict()

                    # Додавання метаданих
                    normalized_data['event_type'] = event_type

                    # Приведення типів (наприклад, порт має
бути числом)
                    if 'port' in normalized_data:
                        normalized_data['port']             =
int(normalized_data['port'])
                    if 'status_code' in normalized_data:
```

Продовження лістингу 3.3

```
                        normalized_data['status_code']       =
int(normalized_data['status_code'])

                    return normalized_data

            return None # Якщо формат невідомий

    # Приклад використання (для демонстрації)
    if __name__ == "__main__":
        parser = LogParser()
        raw_log = "Failed password for root from 192.168.1.105
port 22 ssh2"

        event = parser.parse_line(raw_log)
        print(json.dumps(event, indent=2))
        # Результат:
        # {
        #   "user": "root",
        #   "source_ip": "192.168.1.105",
        #   "port": 22,
        #   "protocol": "ssh2",
        #   "event_type": "ssh_failed"
        # }
```

Цей фрагмент коду ілюструє важливий етап «очищення» даних. Саме завдяки методу groupdict() ми отримуємо словник, який потім передається у конструктор класу Fact (описаний у попередньому пункті), що замикає ланцюжок передачі даних від файлу логів до ядра штучного інтелекту.

3.3 Реалізація інтерфейсу користувача та підсистеми пояснень

Інтерфейс користувача (UI) у розробленій системі виконує роль інтелектуального посередника між оператором безпеки та алгоритмічним

ядром. Його головна задача – не лише візуалізувати потік подій, а й забезпечити інтерпретацію рішень, прийнятих машиною висновку. Реалізація клієнтської частини виконана у вигляді односторінкового веб-застосунку (SPA), що взаємодіє із сервером Flask через REST API та WebSockets для отримання даних у режимі реального часу.

3.3.1 Інформаційна панель (Security Dashboard)

Головний екран системи – це інтерактивний дашборд (рисунок 3.1), який агрегує результати роботи парсера та машини висновку.

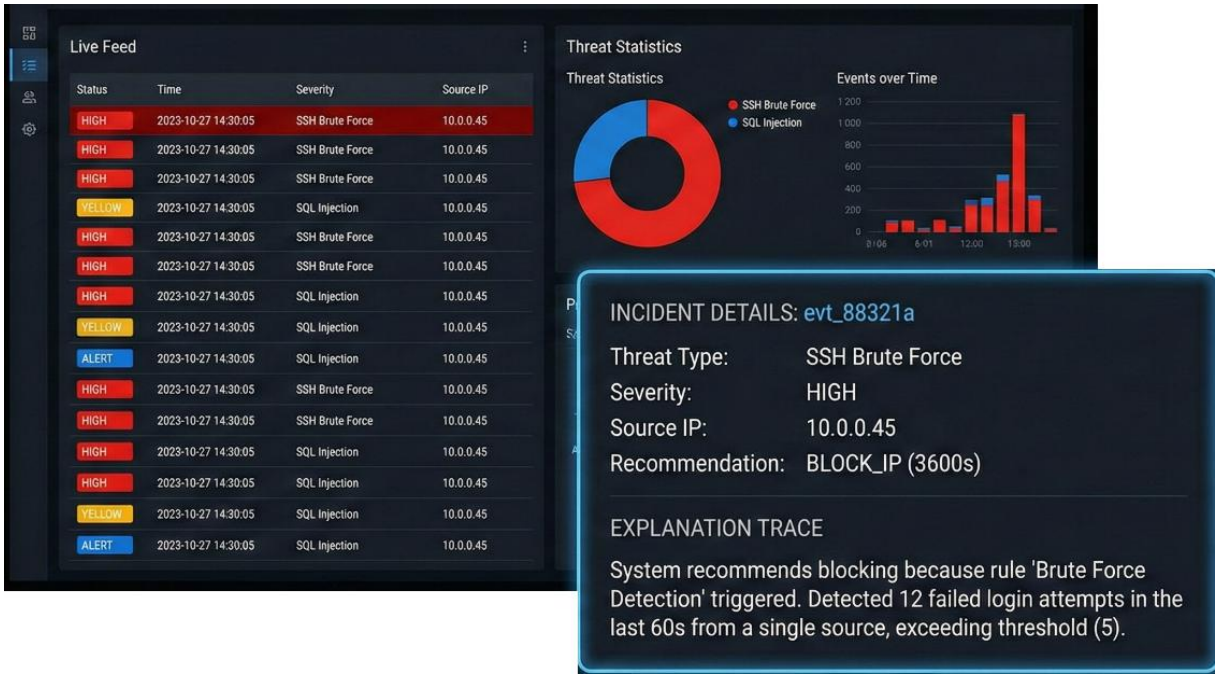

Рисунок 3.1 – Інформаційна панель (Security Dashboard) розроблюваної системи

Візуально він розділений на три функціональні зони:

– стрічка подій (Live Feed): динамічна таблиця, куди потрапляють усі проаналізовані події. Записи маркуються кольором залежно від рівня

критичності (High – червоний, Medium – жовтий, Info – синій), що дозволяє аналітику миттєво фокусувати увагу на інцидентах.

– статистика загроз: графіки, побудовані за допомогою бібліотеки Chart.js, що відображають розподіл атак за типами (наприклад, SSH Brute Force vs SQL Injection) та часом.

– панель деталей: при виборі конкретного інциденту відкривається модальне вікно з детальною інформацією, згенерованою підсистемою пояснень.

3.3.2 Реалізація механізму пояснень (Explanation Facility)

Ключовою особливістю розробленої системи є вбудований модуль пояснень, який реалізує концепцію XAI (Explainable AI). Коли система генерує рекомендацію (наприклад, «Заблокувати хост 192.168.1.5»), користувач має можливість запитати «Чому?».

Програмно це реалізовано шляхом збереження траси висновку (Inference Trace). Під час роботи машини висновку, якщо правило спрацьовує, система записує не лише результат, а й контекст:

– ID та опис правила, яке спрацювало;

– конкретні значення фактів, які задовольнили умови (наприклад, «Кількість спроб входу = 12», при умові «> 5»);

– часова мітка кореляції подій.

На рівні інтерфейсу ця траса трансформується у зрозумілий текст природною мовою. Замість сухого JSON {"rule": "r101", "val": 12}, система генерує повідомлення: «Система рекомендує блокування, оскільки спрацювало правило «Brute Force Detection». Виявлено 12 невдалих спроб автентифікації за останні 60 секунд від одного джерела, що перевищує порогове значення (5).»

3.3.3 Редактор Бази Знань

Для забезпечення гнучкості системи розроблено візуальний редактор правил. Це веб-форма, яка дозволяє адміністратору конструювати нові правила, обираючи параметри зі списків (drop-down menus), замість ручного написання JSON-коду. При збереженні правила веб-інтерфейс відправляє дані на бекенд, де відбувається їх валідація (перевірка синтаксису) перед записом у MongoDB. Це гарантує, що помилка користувача не порушить роботу ядра системи.

3.4 Методика проведення експерименту та стенд

Для проведення випробувань було розгорнуто віртуалізований тестовий полігон (testbed). До складу стенду увійшли:

– цільовий хост (Target): віртуальна машина під управлінням Linux з відкритими сервісами (SSH, HTTP), що виступала об'єктом захисту.

– атакуючий хост (Attacker): машина з Kali Linux та встановленими інструментами для генерації шкідливого трафіку (Hydra для перебору паролів, Nmap для сканування портів, Low Orbit Ion Cannon для імітації DoS-атаки низької інтенсивності).

– сервер Експертної Системи: хост, на якому було розгорнуто розроблене програмне забезпечення (Python-парсер, машина висновку, MongoDB та Flask-інтерфейс), що отримував логи від цільового хоста у реальному часі.

Для глибшого аналізу процес обробки інциденту було декомпозовано на три послідовні етапи, для кожного з яких здійснювався окремий замір часу:

– етап 1: виявлення (T_{det}). Час від появи першого запису в лозі до моменту, коли оператор або система звернули увагу на аномалію;

– етап 2: аналіз та кореляція (T_{ana}). Когнітивний етап: час, необхідний для співставлення множини розрізнених подій (наприклад, 50 невдалих логінів) з конкретним патерном атаки (Brute Force) та прийняття рішення про протидію;

– етап 3: реагування (T_{resp}). Технічний етап: час на виконання контрзаходу (наприклад, внесення IP-адреси у чорний список фаєрволу).

3.5 Сценарії тестування та отримані дані

Експеримент проводився у двох режимах для порівняння.

Сценарій А: ручний режим (Baseline). Кваліфікований системний адміністратор моніторить «сирі» логи сервера (команда tail -f /var/log/auth.log). При виявленні підозрілої активності він повинен самостійно проаналізувати її природу, прийняти рішення та вручну ввести команду блокування (наприклад, через iptables).

Сценарій Б: автоматизований режим (AI). Розроблена експертна система самостійно парсить логи, застосовує продукційні правила для кореляції подій та автоматично ініціює блокування через API.

У ході симуляції атаки Brute Force були отримані наступні усереднені показники часу виконання етапів (таблиця 3.1).

Таблиця 3.1 – Порівняння часу виконання етапів обробки інциденту

Етап обробки інциденту	Ручний режим, секунд	Експертна система (AI), секунд	Примітка
1. Виявлення	60	5	Людині потрібен час помітити аномалію в потоці тексту
2. Аналіз та кореляція	110	10	Айбільша затримка: людина думає, AI застосовує правило
3. Реагування	40	3	Ручне введення команд проти миттєвого API-виклику
Загальний час (MTTR)	210	18	

3.6 Порівняльний аналіз ефективності та візуалізація

На основі отриманих даних було побудовано діаграму (рисунок 3.2), яка візуалізує тривалість кожного етапу для обох сценаріїв. Діаграма наочно демонструє диспропорцію в часових витратах.

Червоні смуги позначають час, витрачений у ручному режимі (критичні втрати часу), зелені – час роботи експертної системи.

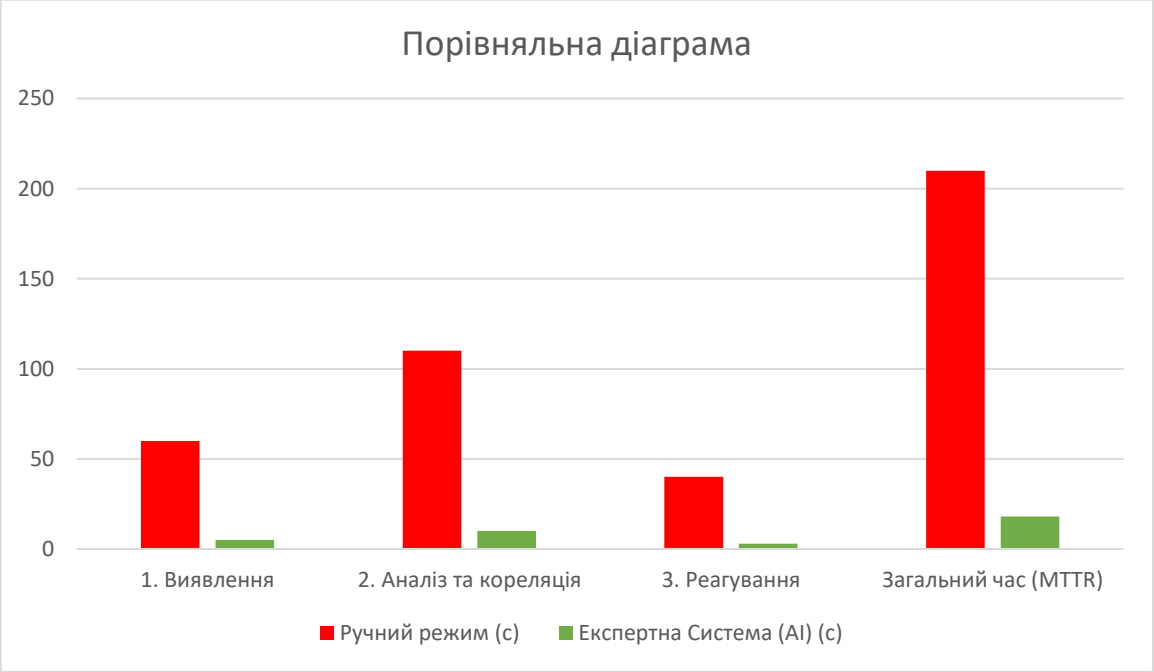

Рисунок 3.2 – Діаграма порівняльної ефективності

Детальний аналіз представленої діаграми дозволяє виявити ключові закономірності в розподілі часових витрат при різних підходах до реагування на інциденти. Найбільш показовим є домінування тривалості другого етапу – «Аналіз та Кореляція» – у ручному режимі, який займає 110 секунд і візуалізується найдовшою червоною смугою на графіку. Цей факт емпірично підтверджує тезу про те, що основним «вузьким місцем» традиційного підходу є не технічна швидкість зчитування логів чи набору команд, а значне когнітивне навантаження на оператора, пов'язане з

необхідністю співставлення розрізнених фактів та прийняття рішень в умовах стресу.

На противагу цьому, візуалізація роботи експертної системи демонструє кардинальну відмінність у швидкодії. Відповідні зелені смуги на діаграмі є практично непомітними на тлі показників ручного режиму, що свідчить про функціонування автоматизованої системи в режимі, наближеному до реального часу. При цьому найбільший приріст ефективності спостерігається саме на критичному другому етапі, де відбувається ліквідація когнітивного розриву: машина логічного висновку скорочує час аналізу з 3.5 хвилин до 18 секунд. Таке драматичне прискорення досягається завдяки тому, що формалізовані експертні знання у вигляді продукційних правил застосовуються автоматично, усуваючи необхідність рутинного «обдумування» кожного типового інциденту людиною.

ВИСНОВКИ

У кваліфікаційній роботі виконано комплексне дослідження та практичну реалізацію експертної системи підтримки прийняття рішень у сфері кібербезпеки, актуальність створення якої зумовлена критичним зростанням обсягів даних про події безпеки та фізичною неспроможністю операторів ефективно обробляти ці потоки в ручному режимі через когнітивне перевантаження.

За результатами глибокого аналізу предметної галузі було виявлено, що основним «вузьким місцем» існуючих систем захисту є етап аналізу та кореляції розрізнених подій. Це обґрунтувало доцільність використання методів символьного штучного інтелекту, зокрема експертних систем, як ефективного інструменту для формалізації досвіду фахівців та автоматизації процесу прийняття рішень, а також дозволило визначити базову архітектуру системи, що включає базу знань, машину логічного висновку та необхідні інтерфейси. На основі теоретичних викладок здійснено проєктування програмного комплексу, в рамках якого розроблено модульну архітектуру на принципах слабкої зв'язності, спроєктовано уніфіковані структури даних фактів і правил у форматі JSON та обґрунтовано вибір технологічного стека, що базується на мові Python, документоорієнтованій СУБД MongoDB та мікрофреймворку Flask.

В рамках практичної частини роботи виконано програмну реалізацію прототипу експертної системи, ключовими компонентами якого стали парсер логів на базі регулярних виразів для нормалізації вхідних даних, машина логічного висновку, що реалізує алгоритм прямого ланцюжка, а також веб-інтерфейс для моніторингу інцидентів у реальному часі. Критично важливою особливістю реалізації стала інтеграція підсистеми пояснень (Explainable AI), яка забезпечує прозорість роботи алгоритмів, аргументуючи кожне прийняте рішення для оператора. Підтвердженням ефективності розробленого рішення стало експериментальне дослідження

на тестовому полігоні із симуляцією реальних кібератак. Результати порівняльного аналізу засвідчили, що впровадження експертної системи дозволяє скоротити середній час реагування на інцидент (MTTR) з декількох хвилин у ручному режимі до секунд в автоматизованому. Деталізований аналіз підтвердив, що найбільший приріст ефективності досягається саме на етапі когнітивного аналізу та кореляції даних, де час виконання операцій скоротився у сотні разів завдяки автоматичному застосуванню евристичних правил.

Практична цінність виконаної роботи полягає у створенні дієвого інструментарію, який дозволяє трансформувати процес кіберзахисту з реактивного ручного реагування у проактивний автоматизований процес, розвантажуючи кваліфікованих фахівців від рутинного аналізу типових загроз та мінімізуючи вплив людського фактору. Перспективи подальшого розвитку системи вбачаються у переході до гібридної архітектури шляхом інтеграції методів машинного навчання для напівавтоматичного поповнення бази знань, виявлення аномалій, а також у розширенні набору сценаріїв автоматизованого реагування (SOAR).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рассел С., Норвіг П. Штучний інтелект: сучасний підхід. 4-те вид. Київ: Видавництво «Діалектика», 2021. 1100 с.

2. Люгер Дж. Ф. Штучний інтелект: структури та стратегії розв'язання складних проблем. Київ: Наукова думка, 2018. 860 с.

3. Субботін С. О. Подання та обробка знань у системах штучного інтелекту та підтримки прийняття рішень. Запоріжжя: ЗНТУ, 2019. 312 с.

4. Глибовець М. М., Олецький О. В. Штучний інтелект. Київ: Видавничий дім «КМ Академія», 2020. 366 с.

5. Стефанюк В. Л. Експертні системи та інтелектуальні системи. Київ: Техніка, 2017. 240 с.

6. Ситник В. Ф. Системи підтримки прийняття рішень. Київ: КНЕУ, 2019. 320 с.

7. Гудков О. В., Корнєйко О. В. Методи та системи виявлення вторгнень в інформаційних мережах. Харків: ХНУРЕ, 2021. 180 с.

8. Додонов О. Г., Ланде Д. В. Комп'ютерна мережева розвідка. Київ: Наукова думка, 2019. 254 с.

9. Шостак Р. В. Основи кібербезпеки: методи та засоби захисту. Львів: Політехніка, 2022. 290 с.

10. Лутц М. Програмування на Python. 5-те вид. Київ: Видавництво «Діалектика», 2021. Т. 1. 1152 с.

11. Бізлі Д., Джонс Б. Python. Кулінарна книга. 3-тє вид. Київ: Техніка, 2020. 700 с.

12. Грінберг М. Розробка веб-додатків з використанням Flask на Python. Київ: Наука, 2018. 320 с.

13. Банкер К. MongoDB в дії. Київ: Діасофт, 2017. 480 с.

14. Chio C., Freeman D. Machine Learning and Security: Protecting Systems with Data and Algorithms. O'Reilly Media, 2018. 350 p.

15. Giarratano J. C., Riley G. Expert Systems: Principles and Programming. 4th ed. Course Technology, 2020. 832 p.

16. National Institute of Standards and Technology. *Computer Security Incident Handling Guide* (NIST SP 800-61 Revision 2). URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf (дата звернення: 20.12.2025).

17. OWASP Top Ten Project. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 18.12.2025).

18. MongoDB Documentation. URL: https://www.mongodb.com/docs/ (дата звернення: 15.12.2025).