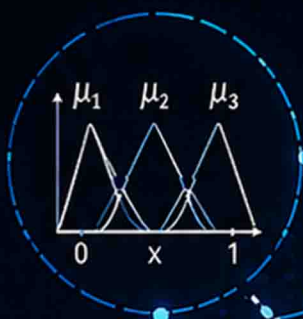


ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

ПІДТРИМКИ ПРИЙНЯТТЯ
РІШЕНЬ У КІБЕРБЕЗПЕЦІ:
від нечіткої логіки до LLM-агентів



ISG-KONF.COM

Хавіна І.П.
Лимаренко В.В.
Гнусов Ю.В.
Можаєв О.О.
Цуранов М.В.

Khavina I., Limarenko V., Hnusov Yu., Mozhayev O., Tsuranov M.

**INTELLIGENT DECISION SUPPORT
SYSTEMS IN CYBERSECURITY: FROM
FUZZY LOGIC TO LLM AGENTS**

Monograph

2026

UDC 004.056:004.896:510.644

Approved for publication by the Academic Council of the Kharkiv National University of Internal Affairs (protocol No. 10 dated 26.05.2026)

Authors:

Inna KHAVINA, ORCID: 0000-0002-1856-1186
Vyacheslav LIMARENKO, ORCID: 0009-0003-7170-9797
Yurii HNUSOV, ORCID: 0000-0002-9017-9635
Oleksandr MOZHAEV, ORCID: 0000-0002-1412-2696
Mykhailo TSURANOV, ORCID: 0000-0002-2115-7029

Reviewers:

Georgy Kuchuk – Doctor of Technical Sciences, Professor, Laureate of the National Prize of Ukraine named after B. Paton, Professor of the Department of "Computer Engineering and Programming" of the National Technical University "Kharkiv Polytechnic Institute".

Andriy Kovalenko – Doctor of Technical Sciences, Professor, Head of the Department of Electronic Computing Machines of the Kharkiv National Institute of Radio Electronics.

Khavina I., Limarenko V., Hnusov Yu., Mozhayev O., Tsuranov M. Intelligent decision support systems in cybersecurity: from fuzzy logic to LLM agents. Monograph. – Primedia eLaunch, Boston, USA, 2026. – 126 p.

Library of Congress Cataloging-in-Publication Data

ISBN – 979-8-90553-934-3

DOI – 10.46299/979-8-90553-934-3

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, distributed, or transmitted, in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher. The content and reliability of the articles are the responsibility of the authors. When using and borrowing materials reference to the publication is required.

UDC 004.056:004.896:510.644

ISBN – 979-8-90553-934-3

© Khavina I., Limarenko V., Hnusov Yu.,
Mozhayev O., Tsuranov M.

SUMMARY

The monograph presents a holistic scientific concept of intelligent cybersecurity management at the enterprise level. The mathematical apparatus of hierarchical structuring of security parameters and strategic selection of protection systems using the analysis of hierarchies (AHI) method by T. Saati is highlighted. The development of decision support systems (DSS) based on the Mamdani fuzzy inference algorithm in the MATLAB environment is described in detail. Decentralized multi-agent systems built on the basis of the cooperative problem solving (CPS) model and optimized by auction algorithms for task distribution based on the “regret” criterion are considered. Special attention is paid to the integration of autonomous artificial intelligence agents and the development of corporate search-augmented generation (Agentic RAG) systems for proactive detection of zero-day threats and anomalous user behavior. The publication is intended for scientists, cybersecurity and information protection specialists, developers of intelligent systems, as well as graduate students and senior students in relevant specialties.

ЗМІСТ

ВСТУП.....	8
1. ТЕОРЕТИЧНІ ОСНОВИ ОЦІНКИ РИЗИКІВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ.....	10
1.1 Аналіз методів оцінки ризиків безпеки та обґрунтування використання нечітких підходів.....	10
2. СИСТЕМНИЙ АНАЛІЗ ТА ІЄРАРХІЧНА СТРУКТУРИЗАЦІЯ ПАРАМЕТРІВ КІБЕРБЕЗПЕКИ НА ОСНОВІ ЕКСПЕРТНИХ ОЦІНОК.....	17
2.1 Побудова ієрархій (структуризація відносин).....	18
2.2 Метод аналізу ієрархій.....	24
2.3 Стратегічний вибір систем кіберзахисту методом МАІ.....	30
3. РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ІНФОРМАЦІЙНОЇ БЕЗПЕКИ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ.....	34
3.1 Основні припущення теорії нечіткої логіки.....	34
3.2 Припущення та методи з галузі кібербезпеки.....	37
3.3 Розробка бази нечітких правил СППР.....	44
3.4 Запропонований метод урахування вагових коефіцієнтів нечітких лінгвістичних змінних.....	45
4. РОЗРОБКА КОНЦЕПЦІЇ СИСТЕМИ ВІЯВЛЕННЯ ВТОРГНЕНЬ НА ОСНОВІ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ.....	47
4.1 Системи автоматизації захисту інформації.....	47
4.2 Сигнатури атак та їх застосування.....	53
4.3 Приклади застосування ШІ для аналізу мережевої активності.....	59
4.4 Розробка концептуальної структури системи виявлення вторгнення.....	62
5. АГЕНТНЕ МОДЕЛЮВАННЯ У ГАЛУЗІ КІБЕРБЕЗПЕКИ.....	67
5.1 Програмні агенти.....	67
5.2 Мультиагентні системи.....	77
5.2.1 Децентралізоване та централізоване управління.....	82
5.2.2 Визначення колективної моделі реальності (КМР).....	83
5.2.3 Оптимальне управління коаліцією за допомогою аукціону.....	85
5.3 Програмні агенти у кібербезпеці.....	90
5.3.1 LLM: розробка та застосування.....	90
5.3.2 Автоматизоване виявлення загроз за допомогою агентів LLM..	91
5.3.3 Виявлення аномалій за допомогою контекстного розуміння.....	92
5.3.4 Розпізнавання патернів у мережевому трафіку.....	92
5.3.5 Аналіз поведінки користувачів.....	92
5.3.6 Виявлення загроз нульового дня.....	93
5.3.7 Автоматизоване реагування на загрози.....	93

5.3.8	Впровадження адаптивної політики безпеки.....	94
5.4	Багатоагентні системи для розподіленого виявлення та реагування на загрози.....	94
5.5	Розробка архітектури MAC СУБ.....	95
5.6	Розробка корпоративної системи RAG.....	100
	ВИСНОВКИ.....	102
	Список джерел інформації.....	105
	Додаток А Процес створення СППР.....	111
	Додаток Б Код асистента AI.....	116

СПИСОК СКОРОЧЕНЬ

ALE (Annual Loss Expectancy) – Очікувані річні збитки (фінансова метрика для оцінки ризику)

Fuzzy Logic Toolbox – Спеціалізований набір інструментів у середовищі MATLAB для проектування систем нечіткої логіки

IDS (Intrusion Detection System) – Система виявлення вторгнень

LLM Agents – Програмні агенти на базі великих мовних моделей, що здатні до автономного аналізу та реагування на загрози

Mamdani (Алгоритм Мамдані) – Один із основних методів нечіткого висновку, що використовується в СППР для перетворення нечітких вхідних даних у чітке рішення

RAG – (Retrieval-Augmented Generation) технологія пошуково-доповненої генерації

ROI (Return on Investment) – Метод повернення інвестицій (використовується для розрахунку критерію фінансових збитків)

SEM (Security Event Management) – Управління подіями безпеки у реальному часі

SIEM (Security Information and Event Management) – Система управління інформацією та подіями безпеки

STRIDE – Модель класифікації загроз (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege)

Агент на базі LLM – автономна програма, що використовує ШІ для розуміння контексту та виконання дій (наприклад, сортування тривог)

АС – Автоматизована система

Аукціон (в МАС) – Метод оптимального розподілу завдань або ресурсів між коаліцією агентів (роботів)

Аукціон – математичний метод розподілу завдань між агентами для досягнення максимальної ефективності

ВП – Відношення погодженості (в методі МАІ)

Дефазифікація – процес отримання чіткого числового значення з нечіткого висновку

Децентралізоване управління – Система, у якій агенти приймають рішення самостійно без єдиного керівного центру

ІБ – Інформаційна безпека

ІП – Індекс погодженості

ІС – Інформаційна система

Коллективна модель реальності (КМР) – спільне представлення середовища, яке формується групою агентів

ЛЗ (Лінгвістична змінна) – змінна, значеннями якої є слова (наприклад, «високий», «низький»)

МАІ – Метод аналізу ієрархій (автор Т. Сааті)

Метод Мамдані – популярний алгоритм нечіткого логічного висновку

Мультиагентні системи (МАС) – системи, що складаються з багатьох взаємодіючих інтелектуальних агентів

ОПР – Особа, що приймає рішення

ОР – Оцінка ризиків

СППР – Система підтримки прийняття рішень

ТНЧ (Трикутне нечітке число) – функція належності у формі трикутника для перетворення якісних оцінок у кількісні

Фазифікація – перетворення чітких даних у нечіткі лінгвістичні терми

ІІ – Штучний інтелект

ВСТУП

Актуальність роботи зі створення системи підтримки прийняття рішень у галузі інформаційної безпеки на основі оцінки ризиків безпеки інформаційної системи (ІС) обумовлена наявністю постійної інформаційної загрози та гострою необхідністю швидкого реагування, виявлення та запобігання вторгненням. Оцінка ризиків (ОР) інформаційної безпеки (ІБ) відіграє ключову роль у практиці управління системою захисту інформації та життєздатністю підприємства. Зростання складності інформаційних систем, поширення цифрових сервісів та збільшення кількості кіберзагроз зумовлюють необхідність системного підходу до аналізу можливих ризиків та їх наслідків.

У таких умовах особливої актуальності набуває розроблення методів оцінювання ризиків, здатних враховувати багатофакторний характер загроз і невизначеність вихідних даних. Завдання оцінки ризиків інформаційної безпеки ІС ускладнюється такими факторами, як їх різноманітність, складність, неоднорідність та інше. Для рішення завдання необхідно використовувати спеціалізовані методи оцінки ризиків та враховувати контекст та особливості конкретної системи (розділ 1).

У роботі для оцінки ризиків ІБ пропонується використати нечітку логіку, яку застосовують у ситуаціях коли є недостатність знань про досліджувану систему, неможливість отримання потрібного обсягу інформації та коли інформація заснована на вхідних даних експертів.

Прикладів успішного застосування нечіткого підходу у сфері кібербезпеки вже достатньо багато як у вітчизняних так і іноземних виданнях.

Для того щоб скористатися нечіткою логікою потрібно отримати вхідні дані та відповідним чином їх представити. Теорія систем надала концептуальну основу для побудови методології, що дозволяє, описувати систему і її проблеми в термінах взаємозалежної ієрархічної структури (розділ 2).

Розробка системи підтримки прийняття рішень з інформаційної безпеки, що заснована на нечіткій логіці надано у розділі 3. Показано формування

вихідних даних для оцінювання ризиків на основі експертних оцінок, які відображають специфіку досліджуваної інформаційної системи. Застосування нечітких чисел дало змогу врахувати лінгвістичний характер експертних суджень та зменшити вплив суб'єктивності на результати оцінки.

У ході дослідження у розділі 4 було проаналізовано сучасні методи виявлення кібератак, визначено їхні переваги та недоліки. Наведено ряд методологій для моделювання інформаційних ризиків та запропоновано концептуальну структуру системи виявлення вторгнення. Планується, що запропонована система поєднає в собі методи машинного навчання, алгоритми аналізу трафіку та STRIDE-моделювання загроз, що дозволить досягти високої точності у виявленні атак.

Розділ 5 присвячено парадигмі мультиагентних систем та LLM-агентів. Обґрунтовано перехід до децентралізованих мультиагентних систем (МАС), де компоненти системи здатні самостійно координувати дії. Встановлено, що інтеграція агентів на базі великих мовних моделей (LLM) дозволяє автоматизувати сортування тривоги, аналізувати неструктуровані дані журналів та виявляти загрози «нульового дня» за рахунок контекстного розуміння.

Показано впровадження пошуково-доповненої генерації (RAG) завдяки якій вдалося підключити локальні корпоративні бази знань без ризику їх оприлюднення та звести до мінімуму вигадки (так звані галюцинації) й гарантувати аналітикам отримання свіжої та точної інформації.

1. Теоретичні основи оцінки ризиків інформаційної безпеки

1.1 Аналіз методів оцінки ризиків безпеки та обґрунтування використання нечітких підходів

У роботі для оцінки ризиків ІБ пропонується використати нечітку логіку, яку застосовують у наступних ситуаціях: недостатність знань про досліджувану систему; неможливість отримання потрібного обсягу інформації; інформація заснована на вхідних даних експертів, дані подані некоректно або не є достатньо точними.

У практиці управління ризиками інформаційної безпеки експертні судження відіграють визначальну роль, оскільки достовірні статистичні дані щодо ймовірності реалізації кіберзагроз або масштабів потенційних збитків часто відсутні. Експерти, як правило, формулюють свої оцінки у лінгвістичній формі, використовуючи поняття «низький», «середній», «високий» рівень ризику або впливу. Класичні методи, що оперують точними числовими значеннями, не здатні безпосередньо працювати з такими оцінками, що призводить до втрати інформації або спотворення результатів.

При створенні систем оцінки ризиків ІБ необхідно опиратися на ряд вимог та рекомендацій щодо стандартів та нормативних документів технічного захисту інформації.

Стандарт ДСТУ ISO/IEC 27001:2023 «Інформаційні технології. Методи захисту системи управління інформаційною безпекою» визначає вимоги до проектування, впровадження, підтримки та постійного вдосконалення системи управління інформаційною безпекою з урахуванням обставин організації.

Подолання зазначених обмежень стало можливим завдяки розвитку теорії нечітких множин і нечіткої логіки, запропонованої Л. Заде. Основною ідеєю цієї теорії є можливість опису нечітких, неточних і суб'єктивних понять за допомогою функцій належності, значення яких змінюються в інтервалі від 0 до 1. Це дозволяє формалізувати експертні судження та здійснювати їх математичну обробку без втрати смислового навантаження.

Прикладів застосування нечіткого підходу у сфері кібербезпеки вже

достатньо багато як у вітчизняних видання так і іноземних. Наприклад, у роботах [2-3] відзначено, що метод оцінки ризиків інформаційної безпеки на основі теорії нечітких множин навіть за недостатнього обсягу вхідних даних дозволяє побудувати адекватну модель впливу загроз на ресурс, що підлягає захисту.

Цей підхід застосували у [4] для попередніх розрахунків оцінки ризиків ще тільки створюваної АС. Візуалізація ризиків дала можливість ранжувати рівні ризиків та підготувати відповідний план пом'якшення наслідків.

У роботах [5-6] предметом дослідження також є процес оцінки рівня ризику інформаційної безпеки, що реалізується за допомогою апарату нечіткої логіки.

Щоб покращити інформаційну безпеку на основі наборів нечітких правил, у роботі [7] пропонується алгоритм вилучення правил нечіткої асоціації на основі нечіткої матриці та застосовується для кореляції подій безпеки. Ця представлена методологія дає можливість перевести отримані результати оцінки ризику з математичної мови у лінгвістичну форму, більш зрозумілу для особи, яка приймає рішення, що підвищує ефективність управління механізмами захисту автоматизованих інформаційних систем.

Л. Заде розширив класичне поняття множини, припустивши, що характеристична функція $\mu_A(x)$ (функція належності елемента множини) може приймати будь-які значення в інтервалі $[0; 1]$, а не тільки значення 0 чи 1. Такі множини були названі нечіткими (fuzzy) та розроблені операції над ними [8, 9].

Існують прямі та непрямі методи визначення функції належності до множини $\mu_A(x)$ [10, 11].

Як правило, прямі методи завдання функції приладдя використовуються для вимірних понять, таких як швидкість, час, відстань, тиск, температура і т. д., або коли виділяються полярні значення. При прямих методах використовуються також групові прямі методи, коли, наприклад, групі експертів пред'являють конкретне значення, і кожен повинен дати один з двох відповідей так чи ні (визначаючи суб'єктивну оцінку значення), тоді кількість

ствердних відповідей, поділену на загальну кількість експертів, дає колективну оцінку значення.

Непрямі методи визначення значень функції належності застосовують у випадках, коли немає елементарних вимірних властивостей, через які визначається нечітка множина, як правило, це методи попарних порівнянь.

Концепція нечіткою та лінгвістичних змінних використовується при описі об'єктів та явищ за допомогою нечітких множин.

Для визначення результату застосовуються нечіткі висновки у вигляді сукупності нечітких предикатних правил виду Якщо $x \in A_1$, тоді $y \in B_1$.

Загальний логічний висновок здійснюється за такі чотири етапи.

1. Нечіткість (введення нечіткості, фазифікація).
2. Логічний висновок.
3. Композиція.
4. На закінчення робиться приведення до чіткості (дефазифікація).

Відповідно ISO/IEC 27005:2023, ризик – це міра ступеня, у якому суб'єкту загрожує потенційна обставина чи подія, і, зазвичай, функція – несприятливі впливу, які можуть виникнути у разі обставини чи події, і навіть ймовірність виникнення. І ризик – це добуток ймовірності виникнення інциденту безпеки та шкоди, який буде заподіяно організації через інцидент.

Також, відповідно до стандарту, критерії, призначені для оцінки ймовірності виникнення загроз, включають: стратегічне значення процесу бізнес-інформації; критичність залучених інформаційних активів; операційна та бізнес важливість доступності, конфіденційності та цілісності; очікування та уявлення зацікавлених сторін, а також негативні наслідки для доброї волі та репутації.

Оцінка ризиків інформаційної безпеки є складовою системи управління інформаційною безпекою та використовується для прийняття обґрунтованих управлінських рішень щодо вибору заходів захисту. У сучасній практиці застосовується значна кількість методів оцінювання ризиків, які відрізняються за рівнем формалізації, обсягом необхідних даних та складністю реалізації.

Вибір конкретного підходу залежить від особливостей організації, доступності інформації та рівня зрілості системи управління безпекою.

Якісні методи базуються на експертних оцінках і використовують словесні шкали для визначення рівня ризику. Такі методи широко застосовуються на початкових етапах впровадження системи управління інформаційною безпекою, оскільки не потребують значних обчислювальних ресурсів і можуть бути реалізовані навіть за відсутності статистичних даних. Рівень ризику в межах якісного підходу, як правило, визначається за допомогою таких категорій, як низький, середній або високий, що дозволяє швидко ідентифікувати критичні загрози.

Разом з тим, якісні методи мають низку суттєвих обмежень. Основним недоліком є високий рівень суб'єктивності, зумовлений залежністю результатів від досвіду та компетентності експертів. Відсутність чітких числових показників ускладнює порівняння результатів оцінки ризиків між різними системами або в динаміці часу. Крім того, якісні методи не завжди дозволяють визначити пріоритети впровадження заходів захисту у разі наявності кількох ризиків з однаковим рівнем значущості.

Кількісні методи оцінки ризиків передбачають використання числових значень для опису ймовірності реалізації загроз і величини можливих збитків. У межах таких підходів ризик зазвичай визначається як добуток імовірності інциденту та його наслідків. Кількісні методи дозволяють отримати більш формалізовані результати, які можуть бути використані для економічного обґрунтування витрат на заходи інформаційної безпеки та порівняння альтернативних варіантів управлінських рішень [12].

Незважаючи на очевидні переваги, застосування кількісних методів у сфері інформаційної безпеки часто ускладнюються відсутністю достовірних вихідних даних. Ймовірність реалізації кіберзагроз, як правило, складно оцінити на основі статистики через швидку зміну ландшафту загроз і обмежений доступ до інцидентної інформації. Оцінка збитків також має значну невизначеність, оскільки нематеріальні втрати, такі як репутаційна шкода або втрата довіри

клієнтів, важко піддаються точному кількісному вимірюванню. Це знижує практичну цінність кількісних методів у реальних умовах.

У зв'язку з зазначеними обмеженнями на практиці часто застосовуються комбіновані підходи, які поєднують елементи якісного та кількісного аналізу. Такі методи дозволяють використовувати числові оцінки там, де це можливо, і водночас зберігати гнучкість експертних суджень. Прикладом комбінованих підходів є методики, що ґрунтуються на матрицях ризиків, де ймовірність та вплив подій оцінюються за шкалами з обмеженою кількістю рівнів.

Проте навіть комбіновані методи не завжди забезпечують достатній рівень адекватності оцінки ризиків у складних інформаційних системах. Основною причиною цього є неможливість точного формалізованого опису експертних знань та нечіткий характер вихідної інформації. Експерти, як правило, мислять у термінах лінгвістичних оцінок, а не точних числових значень, що створює розрив між реальним процесом ухвалення рішень і математичними моделями, які використовуються для його опису.

Міжнародні стандарти з управління ризиками інформаційної безпеки, зокрема ISO/IEC 27005 та NIST SP 800-30, визначають загальну методологію процесу оцінки ризиків, однак не регламентують конкретні математичні методи їх обчислення. Стандарти орієнтовані на опис етапів аналізу ризиків, таких як ідентифікація активів, загроз і вразливостей, а також оцінка ймовірності та впливу інцидентів. А вибір конкретних методів оцінювання залишається на розсуд організації, що зумовлює необхідність пошуку підходів, здатних адекватно працювати в умовах невизначеності та обмеженості вихідних даних.

Ключову роль у формуванні ризик-орієнтованого підходу до забезпечення інформаційної безпеки відіграє міжнародний стандарт ISO/IEC 27001, який встановлює вимоги до системи управління інформаційною безпекою (ISMS). Відповідно до цього стандарту, управління ризиками є основою для вибору та впровадження заходів захисту інформації, а оцінка ризиків розглядається як безперервний процес, що має враховувати специфіку інформаційних активів, загроз і вразливостей організації.

Стандарт ISO/IEC 27001 не визначає конкретних математичних методів оцінювання ризиків, залишаючи організаціям свободу вибору інструментарію з урахуванням рівня зрілості системи управління безпекою та доступності вихідних даних. Такий підхід зумовлює необхідність використання методів, здатних адекватно працювати в умовах невизначеності та базуватися на експертних судженнях.

У цьому контексті застосування нечітких методів оцінки ризиків є методологічно обґрунтованим та повністю узгоджується з вимогами ISO/IEC 27001. Нечітка логіка дозволяє формалізувати лінгвістичні оцінки експертів, а багатокритеріальні методи ухвалення рішень забезпечують інтеграцію різнорідних критеріїв у межах єдиної моделі оцінювання ризиків, що відповідає ризик-орієнтованій концепції стандарту.

Подолання зазначених проблем стало можливим завдяки розвитку теорії нечітких множин та нечіткої логіки. Нечіткі підходи дозволяють формалізувати лінгвістичні оцінки експертів шляхом використання нечітких чисел і функцій належності. Це забезпечує плавний перехід від якісних суджень до кількісних результатів без втрати смислового наповнення інформації. У контексті оцінки ризиків інформаційної безпеки нечіткі методи є особливо доцільними, оскільки дозволяють враховувати невизначеність та суб'єктивність експертних оцінок.

Оцінка ризиків інформаційної безпеки зазвичай не може бути зведена до аналізу одного показника, оскільки рівень ризику формується під впливом сукупності різнорідних факторів. Ймовірність реалізації загрози, масштаб можливих наслідків, рівень уразливості системи та ефективність наявних заходів захисту мають різну природу та значущість, що ускладнює їх узагальнення в межах традиційних одновимірних моделей. У таких умовах доцільним є використання багатокритеріальних методів ухвалення рішень, які дозволяють формалізувати процес оцінки ризиків з урахуванням важливості кожного критерію.

Аналіз традиційних методів оцінки ризиків інформаційної безпеки свідчить про їх обмежену ефективність у умовах високої невизначеності та

складності сучасних інформаційних систем. Нечіткі підходи, зокрема методи нечіткого ухвалення рішень, дозволяють подолати зазначені недоліки та забезпечують більш гнучкий і реалістичний механізм оцінювання ризиків.

На основі аналізу першого розділу наданого документа можна зробити такі висновки:

- традиційні кількісні методи часто втрачають точність через неможливість адекватно опрацювати лінгвістичні оцінки (наприклад, «низький» чи «високий» рівень ризику). Нечіткі множини дозволяють математично формалізувати ці судження без втрати їхнього змістового навантаження.

- використання нечіткої логіки для оцінки ризиків інформаційної безпеки є доцільним у ситуаціях з високим рівнем невизначеності: за браком знань про систему, неможливості отримати точні статистичні дані або коли вхідна інформація базується на суб'єктивних експертних судженнях.

- запропонований підхід повністю узгоджується з вимогами стандарту ISO/IEC 27001, який орієнтований на ризик-менеджмент, але не обмежує організацію у виборі конкретних математичних методів обчислення.

- для реалізації оцінки пропонується використовувати концепцію лінгвістичних змінних Л. Заде та чотири етапний процес логічного висновку: фазифікацію (введення нечіткості), власне логічний висновок, композицію та дефазифікацію (приведення до чіткого числа).

- оцінка ризику розглядається не як один показник, а як результат аналізу сукупності факторів (ймовірність загрози, масштаб наслідків, вразливість системи), що вимагає застосування багатокритеріальних методів прийняття рішень.

2. Системний аналіз та ієрархічна структуризація параметрів кібербезпеки на основі експертних оцінок

Будь-яка проблема являє собою складний об'єкт, що має ієрархічну структуру. При аналізі такого об'єкта, звичайно, зіштовхуються зі складною системою взаємодії компонент проблеми (ресурси, мети, впливові особи й групи, політичні, економічні й інші фактори), які потрібно проаналізувати [11, 12].

Забезпечення стійкості сучасних інформаційно-комунікаційних систем до кібервпливів є багатофакторною задачею, що розв'язується в умовах високої структурної та функціональної складності об'єктів захисту. Ключовою проблемою при побудові систем моніторингу та оцінювання безпеки є не стільки дефіцит даних, скільки відсутність чіткої структури їх взаємозв'язків. Велика кількість різномірних параметрів – від технічних характеристик трафіку до організаційних чинників – створює перенасичене інформаційне поле, де прямий аналіз окремих показників не дає цілісного розуміння стану захищеності.

У цьому контексті критичного значення набуває залучення висококваліфікованих експертів, здатних виділити релевантні чинники та визначити їхню роль у загальній архітектурі безпеки. Проте експертні знання за своєю природою є нелінійними та часто фрагментарними. Тому першочерговим завданням дослідження стає перехід від набору розрізнених думок фахівців до побудови цілісної ієрархічної моделі параметрів, яка відображає реальну топологію загроз та вразливостей.

Даний розділ присвячений методології виявлення та систематизації ключових параметрів кібербезпеки на основі інтелектуальної обробки експертних даних. Основна увага зосереджена на декомпозиції складної проблеми безпеки на простіші, керовані компоненти, що дозволяє встановити причинно-наслідкові зв'язки між технічними вразливостями та їхнім впливом на критичні бізнес-процеси.

В межах розділу розглядаються наступні фундаментальні питання.

Формування репрезентативної множини параметрів: критичний аналіз та відбір показників, що найбільш повно характеризують стан інформаційної системи, з урахуванням специфіки сучасних ландшафтів загроз.

Побудова багаторівневої ієрархії чинників: Визначення домінуючих та підпорядкованих факторів, що дозволяє структурувати систему захисту від низькорівневих технічних індикаторів до високорівневих метрик ризику.

Виявлення та аналіз взаємозв'язків: Дослідження кореляції та взаємовпливу між параметрами, що дозволяє ідентифікувати ланцюги поширення атак та потенційні точки каскадних відмов у системі захисту.

Валідація та верифікація структури: обґрунтування адекватності побудованої ієрархічної моделі через процедури експертного узгодження та аналіз консистентності їхніх оцінок.

Результати цього розділу закладають логічний каркас дослідження. Сформована ієрархічна структура та виявлені взаємозв'язки між параметрами є необхідним підґрунтям для подальшого математичного моделювання та розробки інтелектуальних систем підтримки прийняття рішень у сфері кіберзахисту.

2.1 Побудова ієрархій (структуризація відносин)

Ієрархія – це система, що складається з об'єктів (елементів), згрупованих у незалежні підмножини (групи). Об'єкти i -ої групи знаходяться під впливом об'єктів $(i+1)$ групи i , у той же час, впливають на об'єкти $(i-1)$ групи. Ці групи, розташовані певним чином (над або під іншою групою), називаються рівнями (або кластерами). Вважається (для багатьох задач), що елементи одного рівня незалежні [13, 14].

Існує кілька видів ієрархій, два з яких представлені на рис. 2.1: домінантні ієрархії (рис. 2.1 а, б); холархії (рис. 2.1 в). Домінантні ієрархії бувають повними й неповними. Повними називаються такі, у яких кожний елемент нижнього $(i+1)$ рівня зв'язаний з кожним елементом i -го рівня (рис. 2.1, а), а неповними називаються такі ієрархії, для яких ця умова не виконується, тобто деякі елементи $(i+1)$ рівня зв'язані не з всіма елементами i -го рівня.

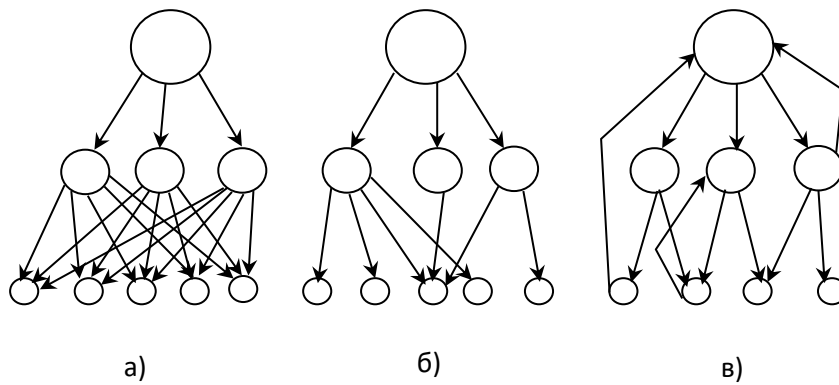


Рисунок 2.1 – Види ієрархій: а) домінантна повна;
б) домінантна неповна; в) холархія [13]

Холархіями називають домінантні ієрархії зі зворотним зв'язком (рис. 2.1, в). Холархії також бувають повними й неповними.

Існує ще один вид ієрархій, який називають іноді «китайською шкатулкою». Він представляє таке співвідношення між класами об'єктів, коли один клас об'єктів є підмножиною більш могутньої множини, що, у свою чергу, є підмножиною наступних ще більш могутніх множин і т.д. Таку ієрархічну структуру має навколишній нас світ з усіма його складними об'єктами.

Структурно-логічне моделювання та обґрунтування стратегічного вибору систем кіберзахисту критичної інфраструктури

Припустимо, що мається деяка множина елементів, між якими існують визначені відносини.

Опис такої системи може бути реалізовано у двох взаємозалежних формах: у виді бінарної матриці й у виді спрямованого графа.

Бінарна матриця може бути представлена матрицею досяжності, що визначається по матриці залежності.

Матриця залежності B заповнюється в такий спосіб. Якщо множина вершин H визначена, то за допомогою бінарного відношення «залежить від» можна заповнити матрицю так, що відповіді «так» фіксують «одиницею», а відповіді «ні» фіксують «нулем», тобто елемент b_{ij} матриці дорівнює:

$$b_{ij} = \begin{cases} 1, \text{ якщо } i \text{ залежить від } j \\ 0, \text{ якщо } i \text{ не залежить від } j \end{cases}$$

Побудувавши в такий спосіб матрицю, переходимо до формування матриці досяжності. Для цього формуємо бінарну матрицю $(I+B)$, (де I – одинична матриця) і зводимо її у деякий ступінь k , такий що виконується умова:

$$(I+B)^{k-1} \leq (I+B)^k = (I+B)^{k+1}.$$

Матриця $(I+B)^k = (I+B)^{k+1}$ і буде матрицею досяжності.

Матриця досяжності може бути побудована і більш простим шляхом, безпосередньо по вихідному спрямованому графі. У цьому графі дуга виходить із залежного елемента. Заповнення матриці бінарними елементами здійснюється по рядку (ліворуч праворуч) за правилом:

$$d_{ij} = \begin{cases} 1, & \text{якщо з } i \text{ можна потрапити в } j \\ 0, & \text{інакше} \end{cases}$$

Наявність матриці досяжності дозволяє розділити множину вершин на підмножини рівнів.

Для цього вершини поділяють на досяжні й попередні.

Вершину h_j називають досяжною з вершини h_i , якщо в орієнтованому графі існує шлях із h_i до h_j . Позначимо підмножину вершин, досяжних із вершини h_i через $R(h_i)$. Вершину h_j називають попередній вершині h_i , якщо можливо досягнення h_i із h_j . Позначимо підмножину вершин, що передують вершині h_i через $A(h_i)$.

Множина тих вершин $A(h_i) = R(h_i) \cap A(h_i)$, для яких виконується умова недосяжності з кожної з вершин множини, що залишилися. H , може бути позначена як рівень ієрархії. Тобто, для структуризації деякої множини елементів H , зв'язаних визначеними відносинами залежності, необхідно виконати наступні процедури.

1. Скласти спрямований граф відносин між елементами множини H .
2. Сформувати матрицю досяжності по спрямованому графі.
3. Сформувати таблицю з елементами h_i , $R(h_i)$, $A(h_i)$ і $R(h_i) \cap A(h_i)$.

Для формування підмножини $R(h_i)$ з i -ої рядка матриці досяжності виписуються номери тих елементів, що містять одиниці. Для формування підмножини $A(h_i)$ з i -го стовпця матриці досяжності виписуються номери тих

елементів, що містять одиниці.

Підмножина $R(h_i) \cap A(h_i)$ формується як логічне перетинання (сполучення) елементів двох підмножин.

4. Знайти елементи в таблиці, для яких виконується умова:

$$A(h_i) = R(h_i) \cap A(h_i).$$

5. Викреслити отримані на першій ітерації елементи і застосувати вищеописані процедури (пункт 1 – 4) знову.

Ітерації повторюються доти, поки залишається більш одного елемента.

Описану вище методику структуризації продемонструємо на такому прикладі. Припустимо, що необхідно ієрархічно структурувати такі компоненти (елементи) як: економічна безпека (ЕНБ); військова безпека (ВБ); екологічна безпека (ЕЛБ); сільськогосподарський сектор економіки (СХ); сектор обчислювальної техніки (ОТ); сектор машинобудування (МШ); сектор енергетичний (ЕН).

Спрямований граф відносин між елементами, розташованими довільним образом, показаний на рис. 2.2.

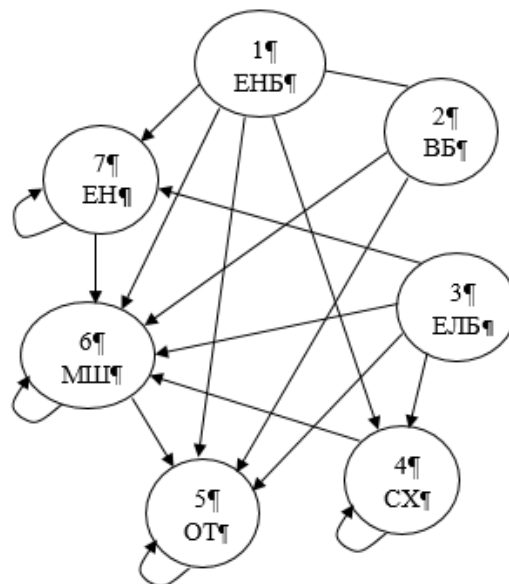


Рисунок 2.2 – Вихідний спрямований граф [14]

Матриця залежності для цієї схеми має вигляд (табл. 2.1).

Таблиця 2.1 – Матриця залежності

		1	2	3	4	5	6	7
		ЕНБ	ВБ	ЕЛБ	СХ	ОТ	МШ	ЕН
1	ЕНБ	0	0	0	1	1	1	1
2	ВБ	1	0	0	0	1	1	0
3	ЕЛБ	0	0	0	1	1	1	1
4	СХ	0	0	0	1	0	1	0
5	ОТ	0	0	0	0	1	0	0
6	МШ	0	0	0	0	1	1	0
7	ЕН	0	0	0	0	0	1	1

Матриця досяжності має вигляд (табл. 2.2).

Таблиця 2.2 – Матриця досяжності

		1	2	3	4	5	6	7
		ЕНБ	ВБ	ЕЛБ	СХ	ОТ	МШ	ЕН
1	ЕНБ	1	0	0	1	1	1	1
2	ВБ	1	1	0	1	1	1	1
3	ЕЛБ	0	0	1	1	1	1	1
4	СХ	0	0	0	1	1	1	0
5	ОТ	0	0	0	0	1	0	0
6	МШ	0	0	0	0	1	1	0
7	ЕН	0	0	0	0	1	1	1

Використовуючи матрицю досяжності, будуємо табл. 2.3, що є першою ітерацією аналізу.

Таблиця 2.3 – Перша ітерація

h_i	$R(h_i)$	$A(h_i)$	$R(h_i) \cap A(h_i)$
1	1,4,5,6,7	1,2	1
2	1,2,4,5,6,7	2	2
3	3,4,5,6,7	3	3
4	4,5,6	1,2,3,4	4
5	5	1,2,3,4,5,6,7	5
6	5,6,	1,2,3,4,6,7	6
7	5,6,7	1,2,3,7	7

З таблиці 2.3 видно, що рівність $A(h_i) = R(h_i) \cap A(h_i)$ виконується для елементів 2 і 3. Отже, вони і є елементами першого рівня.

Викреслюючи з таблиці рядки з номерами 2 і 3, а також викреслюючи з усіх послідовностей цифри 2 і 3, одержуємо другу ітерацію, у якій критеріальна рівність виконується для елемента 1. Він і є елементом другого рівня.

Повторюючи ітерації, одержуємо остаточно п'ять рівнів елементів, що представлені на табл. 2.4 та рис. 2.3. Таке представлення початкової моделі є більш наочним із погляду аналізу залежностей одних елементів від інших. Результат ієрархічної структуризації дозволяє зробити висновок, що з погляду забезпечення безпеки держави критичним є обчислювальна техніка та машинобудівна галузі економіки.

Таблиця 2.4 – Розподіл елементів за рівнями ієрархії

Номер рівня	Розподіл по галузях
I (вершина)	2 (ВБ), 3 (ЕЛБ)
II	1 (ЕНБ)
III	4 (СХ), 7 (ЕН)
IV	6 (МШ)
V (основа)	5 (ОТ)

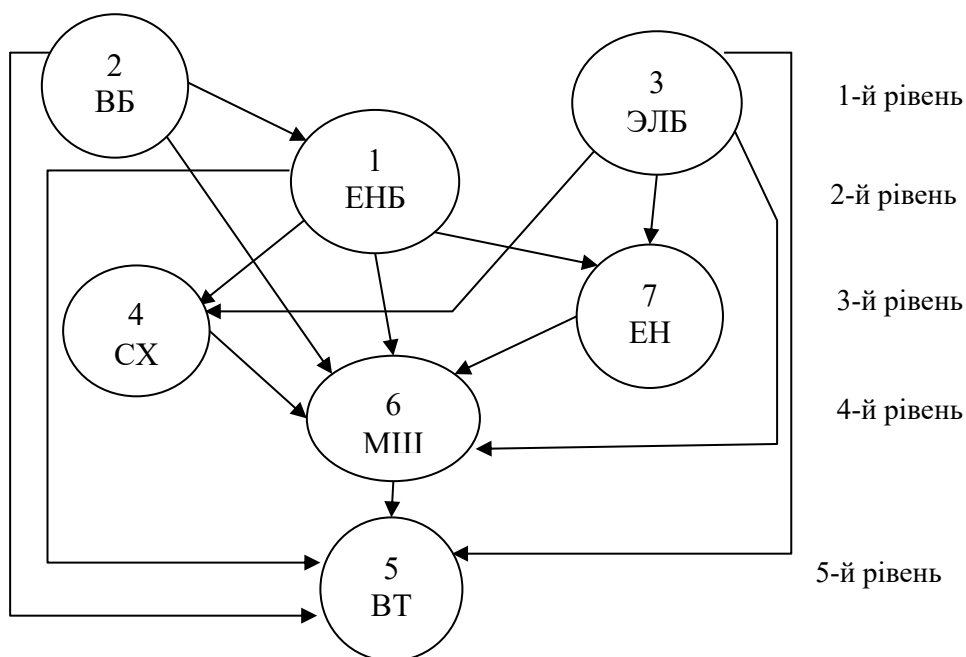


Рисунок 2.3 – Ієрархічна структура вихідного графа [14]

2.2 Метод аналізу ієрархій

Метод аналізу ієрархій (МАІ) є систематичною процедурою для ієрархічного представлення компонентів проблеми [15]. Метод становить у декомпозиції проблеми на усе більш прості складові і подальшій обробці послідовності суджень особи, що приймає рішення (ОПР), по парних порівняннях. У результаті може бути отриманий відносний ступінь (інтенсивність) взаємодії (впливу) компонентів нижнього i -го рівня на компоненти верхнього $(i-1)$ -го рівня або i -го рівня на самий верхній (нульовий) рівень. Ці оцінки виражаються потім чисельно. МАІ включає процедури синтезу множинних суджень, одержання пріоритетності критеріїв і пошуку альтернативних рішень.

Метод аналізу ієрархій включає наступні основні етапи:

- декомпозиція проблеми;
- побудова ієрархічної структури моделі проблеми;
- експертне оцінювання переваг;
- побудова локальних пріоритетів;
- оцінка погодженості суджень;
- синтез локальних пріоритетів;

- висновки й пропозиції для прийняття рішень.

Метод аналізу ієрархій при побудові єдиної шкали для різних компонентів проблеми використовує міру ступеня впливу кожного фактору одного рівня на фактори верхнього рівня або на кінцеву мету. Ця міра утвориться в результаті висловлення суджень про ступінь впливу (важливості) цих факторів. Американський фахівець із системного аналізу Т. Сааті запропонував шкалу відносної важливості (значимості, переваги) представлену в таблиці 2.5.

Таблиця 2.5 – Шкала відносної важливості

Ступінь переваги одного об'єкта перед іншим	Міра важливості (значимості) переваги
Рівна важливість (значимість). Немає переваги	1
Слабка перевага по важливості. Слабка перевага.	3
Істотна або сильна перевага по важливості (значимості). Сильна перевага.	5
Дуже сильна або значна перевага по важливості (значимості). Дуже сильна перевага.	7
Абсолютна перевага.	9
Проміжна оцінка міри переваги між сусідніми значеннями	2, 4, 6, 8

Вибір дискретної шкали «1 – 9» для оцінки порівняльної міри важливості (значимості або рівня переваг), одержуваної в результаті висловлення суджень експертом, ґрунтується на наступних передумовах:

1. Якісні розходження значимі на практиці і мають елемент точності, коли величина порівнюваних об'єктів (предметів, явищ, процесів, видів діяльності) одного порядку або об'єкти близькі щодо властивості, по якій вони порівнюються.

2. Психометричні властивості людини дозволяють досить добре проводити якісні розмежування мір властивостей порівнюваних об'єктів по наступним рівням: немає розходження, слабке розходження, сильне розходження, дуже

сильне розходження, абсолютне розходження. З огляду на компромісні оцінки розходження між перерахованими вище рівнями значимості (важливості), одержуємо дев'ять рівнів (ступенів) розходження, що можуть бути добре погоджені.

3. У психології існує поняття психологічної межі здатності людини одночасно розрізнати якесь число предметів по якійсь властивості. Ця межа дорівнює 7 ± 2 , тобто для створення шкали, на якій ці предмети будуть розбірливі, необхідно 9 точок.

Метод парних порівнянь

Для побудови шкали пріоритетів (переваг), одержуваної при експертному висловленні суджень про рівень розходження між порівнюваними об'єктами в МАІ застосовується метод парних порівнянь. Якщо для порівняння обране n (A_1, A_2, \dots, A_n) об'єктів, то результати порівнянь заносяться в квадратну n -мірну матрицю виду:

	A_1	A_2	...	A_j	...	A_n
A_1	a_{11}	a_{12}	...	a_{1j}	...	a_{1n}
A_2	a_{21}	a_{22}	...	a_{2j}	...	a_{2n}
...
A_i	a_{i1}	a_{i2}	...	a_{ij}	...	a_{in}
...
A_n	a_{n1}	a_{n2}	...	a_{nj}	...	a_{nn}

Причому $a_{ij} = a_{ij} / a_{ij}$.

Елементом цієї матриці a_{ij} є міра переваги об'єкта A_i у порівнянні з об'єктом A_j . Таким чином, i -й рядок матриці показує міру переваги i -го об'єкта над іншими $(n-1)$ об'єктами і над самим собою. Міра переваги виражається експертом у шкалі Сааті і приймає значення від 1 до 9, якщо об'єкт A_i більш важливий, чим об'єкт A_j . У випадку, коли $i=j$, міра переваги дорівнює 1, тобто діагональні елементи матриці парних порівнянь завжди дорівнюють 1.

Варто враховувати, що для матриці парних порівнянь виконується наступна умова $a_{ji} = \frac{1}{a_{ij}}$.

Це означає, що якщо по шкалі Сааті об'єкт A_i більш важливий, чим об'єкт A_j і ця міра переваги дорівнює a_{ij} (наприклад $a_{ij}=5$), то міра переваги A_j -го об'єкта в порівнянні з об'єктом A_i – величина зворотна a_{ij} (тобто $a_{ji}=1/5$). Експертом заповнюється тільки верхня над діагональна частина матриці парних порівнянь і матриця здобуває наступний вид (наприклад, для чотирьох порівнюваних об'єктів):

	A_1	A_2	A_3	A_4
A_1	1	a_{12}	a_{13}	a_{14}
A_2	$1/a_{12}$	1	a_{23}	a_{24}
A_3	$1/a_{13}$	$1/a_{23}$	1	a_{34}
A_4	$1/a_{14}$	$1/a_{24}$	$1/a_{34}$	1

На практиці експерт сам формує матрицю A , причому передбачається, що діагональні елементи дорівнюють 1, а для елементів симетричних щодо діагоналі $a_{ij} = 1/(a_{ji})$, тобто якщо один елемент оцінюється в α разів сильніше, ніж другий, то цей останній повинен бути в $1/\alpha$ разів сильніше, ніж перший. В загальному разі завдання зводиться до пошуку вектор w , що задовольняє рівнянню виду $Aw = \lambda_{max}w$, де λ_{max} – найбільше власне значення матриці A . Оскільки матриця A позитивна по побудові, рішення даної завдання існує і є позитивним.

Міра погодженості

У загальному випадку під погодженістю мається на увазі те, що при наявності основного масиву неопрацьованих даних усі інші дані можуть бути логічне отримані з них. Якщо порівнюються n об'єктів, то досить $(n-1)$ судження, у яких порівнювані об'єкти представлені, принаймні, один раз. Усі інші судження (у випадку погодженості суджень) можуть бути виведені з них.

Повна погодженість включає як порядкову погодженість, що називають ще властивістю транзитивності (якщо A_i має перевагу над A_j , а A_j має перевагу над A_k , то A_i має перевагу над A_k), так і кардинальну погодженість ($a_{ij} \cdot a_{jk} = a_{ik}$).

Очевидно, що домогтися повної погодженості матриці парних порівнянь при експертних оцінках об'єктів неможливо.

Природно після експертних оцінок по методу парних порівнянь порушити питання про ступінь погодженості отриманих оцінок.

Як міру погодженості розглядають два показники: індекс погодженості (ІП) та відношення погодженості (ВІ).

З теорії матриць відомо, що погодженість зворотно симетричної матриці (яка виходить як результат застосування експертом методу парних порівнянь по шкалі Сааті) еквівалентна вимоги рівності її максимального власного значення λ_{max} і числа порівнюваних об'єктів n ($\lambda_{max} = n$).

Тому як міру неузгодженості розглядають нормоване відхилення λ_{max} від n , називане індексом погодженості ІП

$$IP = \frac{\lambda_{max} - n}{n - 1}.$$

Для того щоб оцінити, чи є отримане узгодження прийнятним чи ні, його порівнюють із випадковим індексом (ВІ).

Випадковим індексом називають індекс погодженості, розрахований для квадратної n -мірної позитивної зворотно симетричної матриці, елементи якої генеровані датчиком випадкових чисел, розподілених по рівномірному закону для інтервалу значень: 1/9, 1/8, 1/7, 1/6, 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Для матриці з фіксованим значенням n індекс розраховується як середнє значення для вибірки N (наприклад, $N=100$).

Нижче представлена таблиця 2.6 величин випадкового індексу для різних матриць порядку від 2 до 10.

Таблиця 2.6 – Величини випадкового індексу (ВІ)

Порядок матриці (n×n)	2	3	4	5	6	7	8	9	10
Випадковий індекс (ВІ)	0	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

Одержавши в результаті розрахунку індекс погодженості i , вибравши з таблиці випадковий індекс для заданого порядку матриці, розраховують відношення погодженості ВІ

$$ВП = \frac{III}{VI}.$$

Якщо величина ВП менше 0,1, то ступінь погодженості варто вважати гарною. У деяких випадках прийнятним ступенем погодженості можна вважати діапазон (0,1 – 0,3). Це, як правило, відноситься до проблем, для яких прийняті по експертних висновках рішення не спричиняють серйозних негативних наслідків. У протилежному випадку (якщо $ВП > 0,1 \div 0,3$) експерту рекомендується переглянути свої судження. Для цього необхідно виявити ті позиції в матриці суджень, що вносять максимальний вклад у величину відносини погодженості, і спробувати змінити міру непогодженості в меншу сторону на основі більш глибокого аналізу питання.

Вектор пріоритетів

Виконуємо обробку матриці парних порівнянь у шкалі Сааті з метою одержання вектору пріоритетів ВП порівнюваних об'єктів [15].

З математичної точки зору задача зводиться до обчислення головного власного вектору, що після нормалізації стає вектором пріоритетів. Точний спосіб обчислення головного власного вектору матриці парних порівнянь полягає в зведенні матриці в достатньо великі ступені і поділ суми кожного рядка на загальну суму елементів матриці.

Скористаємося іншим, більш простим, способом, що дає добре наближення (табл. 2.7). Це обчислення:

- компоненти головного власного вектору (обчислюється як середнє геометричне значень у рядку матриці

$$V_i = \sqrt[n]{\prod_{j=1}^n a_{ij}} ; \quad (2.1)$$

- компоненти вектору пріоритетів (обчислюється як нормоване значення головного власного вектору)

$$P_i = \frac{V_i}{\sum_{i=1}^n V_i} . \quad (2.2)$$

Таблиця 2.7 – Компоненти головного власного вектору та вектору пріоритетів

	A_1	A_2	...	A_n	Головний власний вектор	Вектор пріоритетів
A_1	a_{11}	a_{12}	...	a_{1n}	V_1	P_1
A_2	a_{21}	a_{22}	...	a_{2n}	V_2	P_2
...
A_n	a_{n1}	a_{n2}	...	a_{nn}	V_n	P_n

Наближені значення λ_{max} для оцінки відносини погодженості можна розрахувати за наступною формулою

$$\lambda_{max} = \sum_{j=1}^n M_j P_j,$$

де $M_j = \sum_{i=1}^n a_{ij}$ – сума елементів i -го стовпця матриці;

P_j – вектор пріоритетів заданої матриці.

2.3 Стратегічний вибір систем кіберзахисту методом МАІ

Для демонстрації прикладу побудови вектору пріоритетів визначимо оптимальну систему захисту кінцевих точок напрямку обчислювальної техніки (рис. 2 3).

Припустимо, що є вибір з трьох альтернатив захисту:

A1 – Альфа (класу преміум),

A2 – Бета (клас баланс),

A3 – Гамма (класу бюджет).

Критерії вибору K1 – рівень детекції порушень, K2 – вартість та K3 – швидкість попередження. Причому встановлюємо:

- детекція (K1) суттєво важливіша за вартість (K2) - 5 балів.
- детекція (K1) помірно важливіша за швидкість (K3) - 3 бала.
- швидкість (K3) помірно важливіша за вартість (K2) – 3 бала.

Таблиця 2.8 – Матриця попарних порівнянь критеріїв К1, К2, К3

Критерії	К1	К2	К3	GM_i	w_i
К1	1	5	3	$\sqrt[3]{1 * 5 * 3} = 2,466$	0,637
К2	0,2	1	0,333	$\sqrt[3]{0,2 * 1 * 0,333} = 0,405$	0,105
К3	0,333	3	1	$\sqrt[3]{0,333 * 3 * 1} = 1,000$	0,258
Сума	1,533	9	4,333	3,871	1,000

Перевірка узгодженості

$$\lambda_{max} = (1,533 * 0,637) + (9 * 0,105) + (4,333 * 0,258) = 3,0395$$

Індекс узгодженості (CI)

$$CI = \frac{\lambda_{max} - n}{n - 1} = \frac{3,039 - 3}{3 - 1} = 0,01975,$$

де n (випадковий індекс) обираємо на основі табл. 2.6: так як матриця 3x3 то n = 3.

Відношення узгодженості (CR)

$$CR = \frac{0,0195}{0,58} = 0,034 < 0,1$$

Так, як відношення узгодженості $CR < 0,1$ то матриця вважається узгодженою, а отримані вагові коефіцієнти – математично узгодженими та логічне обґрунтованими.

Розрахунок локальних пріоритетів альтернатив за критерієм К1 (рівень детекції): А1 краща за А2 (бал 3) та суттєво краща за А3 (бал 7).

Таблиця 2.9 – Матриця попарних порівнянь критеріїв К1

Альтернативи	А1	А2	А3	GM_i	a_i (вага)
А1	1	3	7	$\sqrt[3]{1 * 3 * 7} = 2,759$	0,669
А2	0,333	1	5	$\sqrt[3]{0,333 * 1 * 5} = 1,186$	0,287
А3	0,143	0,2	1	$\sqrt[3]{0,143 * 0,2 * 1} = 0,306$	0,044
Сума	1,476	4,2	13	4,251	1,000

Вагу для А1 маємо $A1 = \frac{2,759}{4,251} = 0.669$.

Розрахунок локальних пріоритетів альтернатив за критерієм К2 (вартість): критерій вартості є оберненим: чим дешевша система, тим вищий її бал. А3 – найвигідніша (бал 9 порівняно з А1).

Таблиця 2.10 – Матриця попарних порівнянь критеріїв К2

Альтернативи	А1	А2	А3	GM_i	b_i (вага)
А1	1	0,2	0,111	$\sqrt[3]{1 * 0,2 * 0,111} = 0,281$	0,063
А2	5	1	0,333	$\sqrt[3]{5 * 1 * 0,333} = 1,186$	0,265
А3	9	3	1	$\sqrt[3]{9 * 3 * 1} = 3,000$	0,672
Сума	15	4,2	1,444	4,467	1,000

Розрахунок локальних пріоритетів альтернатив за критерієм К3 (швидкість): за швидкістю системи А1 та А2 ідентичні (бал 1) та значно переважають бюджетну А3 (бал 5).

Таблиця 2.11 – Матриця попарних порівнянь критеріїв К3

Альтернативи	А1	А2	А3	GM_i	c_i (вага)
А1	1	1	5	$\sqrt[3]{1 * 1 * 5} = 1,710$	0,455
А2	1	1	5	$\sqrt[3]{1 * 1 * 5} = 1,710$	0,455
А3	0,2	0,2	1	$\sqrt[3]{0,2 * 0,2 * 1} = 0,342$	0,090
Сума	2,2	2,2	11	3,762	1,000

Глобальний синтез результатів

Для отримання підсумкового балу (S) перемножуємо ваги критеріїв (w_i) з табл. 2.8 на отримані локальні ваги альтернатив a_i, b_i, c_i табл. 2.9, 2.10, 2.11.

Таблиця 2.12 – Зведена матриця глобальних пріоритетів

Альтернативи	К1 (0,637)	К2 (0,105)	К3 (0,258)	Підсумковий бал S	Рейтинг
А1	0,669	0,063	0,455	0,5497	I
А2	0,287	0,265	0,455	0,3281	II
А3	0,044	0,672	0,090	0,1218	III

Математичний розрахунок за методом Т. Сааті однозначно вказує на те, що найбільш обґрунтованим вибором є система А1 (бал 0,5497).

Незважаючи на те, що А1 є найдорожчою системою (мінімальний бал за К2 – 0,063), її перевага у критично важливих критеріях детекції та швидкості повністю нівелює фінансові витрати. Система А3, попри свою економічну привабливість (0,672), посідає останнє місце через неприпустимо низький рівень захисту (0,044), що є недопустимим для ієрархії безпеки (рівня ВТ).

Цей розрахунок доводить, що стратегія кіберзахисту має базуватися на технологічній перевазі, а не на економії ресурсів.

На основі аналізу другого розділу документа можна зробити такі висновки щодо системного аналізу та ієрархічної структуризації параметрів кібербезпеки:

- для опису складних систем захисту використовується теорія графів та матричний аналіз. Проблема декомпозується на ієрархічні рівні (від технічних індикаторів до стратегічних метрик), що дозволяє виявити причинно-наслідкові зв'язки між вразливостями та бізнес-ризиками.

- визначено різні види структур для моделювання безпеки: домінантні (повні та неповні) та холархії (ієрархії зі зворотним зв'язком). Це дозволяє адекватно відобразити реальну архітектуру загроз.

- метод аналізу ієрархій (МАІ) є ключовим інструментом моделювання. Він дозволяє перетворити суб'єктивні парні порівняння експертів у математично обґрунтовані вагові коефіцієнти (пріоритети). Важливим етапом є перевірка відношення узгодженості (CR), яке має бути менше 0,1 для підтвердження логічності оцінок.

- на прикладі вибору системи захисту кінцевих точок доведено, що найбільш обґрунтованою є стратегія, яка базується на технологічній перевазі (рівень детекції та швидкість), а не на мінімізації фінансових витрат. Навіть найдорожча система може бути оптимальною, якщо вона нівелює критичні ризики;

- розроблено математичний апарат для обчислення глобального синтезу результатів, що дозволяє порівнювати різноманітні альтернативи (преміум, баланс, бюджет) за комплексом критеріїв.

3. Розробка системи підтримки прийняття рішень з інформаційної безпеки на основі нечіткої логіки

Завданням розділу є розробка структури нечіткої системи прийняття рішень на основі оцінки ризиків безпеки інформаційної системи та апробація запропонованого підходу на прикладі визначення оцінки ризику ІС тестового підприємства.

3.1 Основні припущення теорії нечіткої логіки

Концепція нечіткої та лінгвістичної змінних використовується при описі об'єктів та явищ за допомогою нечітких множин.

Нечітка змінна характеризується трійкою

$$(\alpha, X, A),$$

де X – універсальна множина (область визначення α); – A найменування змінної; A нечітка множина на X , що описує обмеження (тобто $\mu_A(x)$) на значення нечіткою змінної α).

Лінгвістичної змінною (ЛЗ) називається набір

$$\langle \beta, T, X, G, M \rangle,$$

де: β – найменування лінгвістичної змінної; T – множина її значень (терм-множина), що представляє собою найменування нечітких змінних, областю визначення кожною з яких є множина X .

Множина T називається базовим *терм-множиною* лінгвістичної змінної; G – синтаксична процедура, що дозволяє оперувати елементами терм-множини T , зокрема , генерувати нові терми (значення).

M – семантична процедура, що дозволяє перетворити кожне нове значення лінгвістичної змінної, що утворюється процедурою G , в нечітку змінну, тобто сформувати відповідне нечітку множину.

Л.Заде [8] запропонував концепцію нечітких множин для врахування невизначеності змінних та параметрів. Для побудови функцій належності значення, потрібне визначити її вигляд: трикутна, трапецієвидна, дзвоноподібна чи інша. Наприклад, трикутне нечітке число (ТНЧ)

використовується в різних дослідженнях для перетворення якісних стверджень на кількісні [8]. Для цього інколи достатньо побудувати графік функцій та побачити трикутник (рис. 3.1).

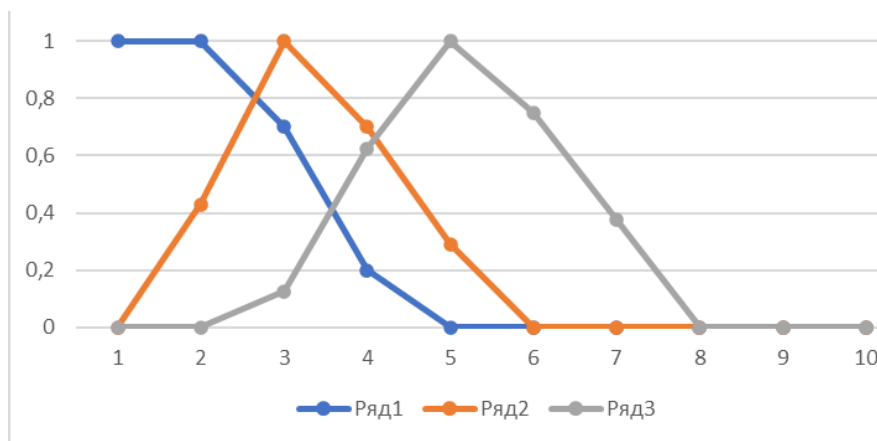


Рисунок 3.1 – Графік залежності значення функцій належності термів лінгвістичної змінної

ТНЧ представляє кожну цифру трьома цифрами. Перше, друге та третє цілі числа, що визначають нечітке число, відображають найнижче, найбільше та найвище потенційні значення. Рівняння (3.1) визначає функцію належності типу трикутника для нечітких чисел.

$$\mu_A(x) = \begin{cases} 0, & x < l \\ \frac{x-l}{m-l}, & l \leq x \leq m \\ \frac{u-x}{u-m}, & m \leq x \leq u \\ 0, & x > u \end{cases} \quad (3.1)$$

Для визначення результату застосовуються нечіткі висновки у вигляді сукупності нечітких предикатних правил виду Якщо $x \in A_1$, тоді $y \in B_1$.

П₁. якщо $x \in A_1$, тоді $y \in B_1$,

П₂. якщо $x \in A_2$, тоді $y \in B_2$,

....

П_n. якщо $x \in A_n$, тоді $y \in B_n$,

де - x вхідна змінна (ім'я для відомих значень даних), y – змінна виводу (ім'я для значення даних, яке буде обчислено); A і B – функції належності

відповідно до x і y .

Загальний логічний висновок здійснюється за наступні чотири етапи.

1. Нечіткість (введення нечіткості, фазифікація (fuzzification)). Функції належності, визначені на входних змінних застосовуються до їхніх фактичним значень для визначень ступеня істинності кожної передумови кожного правила.

2. Логічний висновок. Обчислене значення істинності для передумов кожного правила застосовується до висновків кожного правила. Це призводить до одної нечіткої підмножини, яке буде призначено кожною змінною висновку кожному правилу. Як правила логічного висновку зазвичай використовуються тільки операції \min (МІНІМУМ) або prod (ДОБУТОК).

3. Композиція. Усі нечіткі підмножини, призначені до кожної змінної висновку (у всіх правилах), об'єднуються разом, щоб формувати одну нечітку підмножину для кожної змінної висновку. За такого об'єднання зазвичай використовуються операції \max (МАКСИМУМ) або sum (СУМА).

4. На закінчення робиться *приведення до чіткості* (дефазифікація, defuzzification) для перетворення нечіткого набору висновків у чітке число. Існує велика кількість методів приведення до чіткості.

Приклад роботи методом Мамдані за трьома правилами (рис. 3.2):

P_1 : Якщо $x \in A_1$, та $y \in B_1$ тоді $Z \in C_1$,

P_2 : Якщо $x \in A_2$, та $y \in B_2$ тоді $Z \in C_2$,

P_3 : Якщо $x \in A_3$, та $y \in B_3$ тоді $Z \in C_3$,

Для моделювання задачі у роботі використовується MATLAB та редактор Fuzzy Logic Toolbox для проектування систем нечіткої логіки.

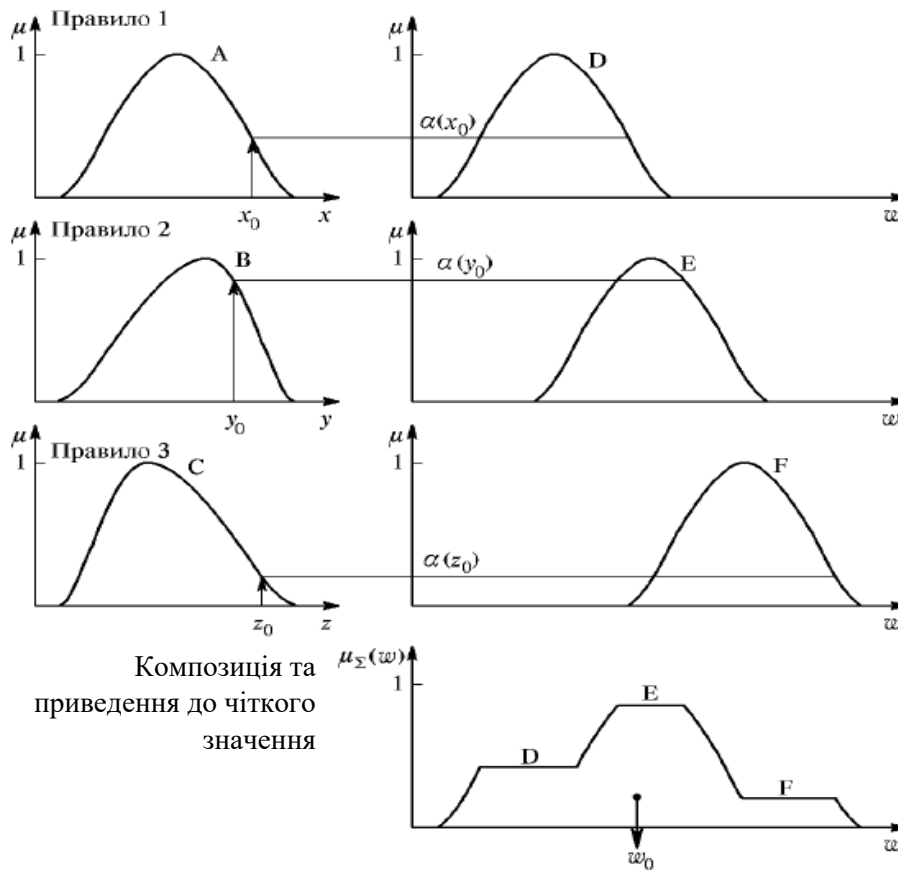


Рисунок. 3.2 – Ілюстрація до процедури логічного висновку Мамдані [13]

3.2 Припущення та методи з галузі кібербезпеки

Відповідно до ДСТУ ISO/IEC 27005:2023, у галузі кібербезпеки, ризик – це добуток ймовірності виникнення інциденту безпеки та шкоди, який буде заподіяно організації через інцидент

$$R = Y1 \cdot Y2,$$

де R – рівень ризику, $Y1$ – ймовірність виникнення загрози, $Y2$ – рівень заподіяної шкоди. Імовірність виникнення інциденту безпеки залежить від ймовірності виникнення загрози та ймовірності того, що загроза зможе використати вразливість у ІС.

Для визначення критеріїв впливу необхідно розробити та конкретизувати з точки зору ступеня шкоди або витрат для ІС, спричинених подією інформаційної безпеки, такі фактори:

1. Привабливість активів;
2. Доступність активів;
3. Вартість активів;

4. Конфіденційність активів;
5. Цілісність активів;
6. Програмне забезпечення та технічні засоби контролю;
7. Адміністративний контроль;
8. Процедурний контроль;
9. Відповідність заходів контролю за стандартами інформаційної безпеки;
10. Попередні інциденти.

Привабливість активу – це характеристика, яка вказує, наскільки привабливим є конкретний актив для потенційних зловмисників, які можуть спробувати несанкціонованого доступу до активу або його інформації.

Доступність активу – це характеристика, яка описує, наскільки легко можна отримати доступ до активу.

Вартість активів – це характеристика, яка показує, наскільки важливим є актив для організації. Якщо актив має високу вартість, це може спричинити більш високу ймовірність реалізації загрози.

Конфіденційність активу – це характеристика, яка відбиває ступінь важливості збереження конфіденційності інформації, пов'язаної з активом. Таким чином, якщо актив є джерелом конфіденційної інформації, це може зробити його привабливішим для потенційних зловмисників.

Цілісність активу – це характеристика, що показує, що актив залишається у своєму вихідному стані і не зазнає несанкціонованих змін або пошкодження.

Перші п'ять критеріїв (привабливість активів, доступність активів, вартість активів, конфіденційність та цілісність активів) були об'єднані в один критерій – значимість активів, що полегшує аналіз та розуміння безпеки активів ІС. Включення цих критеріїв в один комплексний критерій оцінки активів обумовлено їхньою взаємозалежністю та впливом на оцінку активів у контексті інформаційної безпеки.

Програмно-технічний контроль, адміністративний контроль, процедурний контроль та дотримання стандартів інформаційної безпеки були об'єднані в другому критерію – існуючий контроль, де описується рівень контролю, що існує всередині організації.

Третій критерій враховує попередні інциденти. Цей критерій оцінює, чи раніше було скоєно атаки на конкретний актив системи. Якщо так, то ймовірність виникнення загрози буде вищою, оскільки зловмисник уже знайомий із цим активом і може використати минулий досвід атак для наступної.

Перші три фактори в сукупності впливають на фінансові втрати тому об'єднані у четвертий критерій – фінансові збитки.

Критерій п'ять – репутаційні збитки відображає фактор можливих негативних наслідків для репутації компанії у разі ризикованих ситуацій.

З урахуванням обраних критеріїв була розроблена структура СППР для оцінки рівня інформаційного ризику, що містить три послідовні етапи, де:

$Y1$ – ймовірність виникнення загроз; $Y2$ – оцінка шкоди, завданої активам системи, що захищаються; K_1 – привабливість активу; K_2 – існуючий контроль; K_3 – попередні інциденти; K_4 – фінансові витрати; K_5 – втрата репутації, змінні від яких залежить $Y1$ та $Y2$; R – чіткий показник втрат ІС; R_1 – терм, що відповідає показнику R ; f_1, f_2 – нечіткі функціональні залежності лінгвістичної змінної (ЛЗ), що визначаються за допомогою нечітких правил (рис. 3.3).

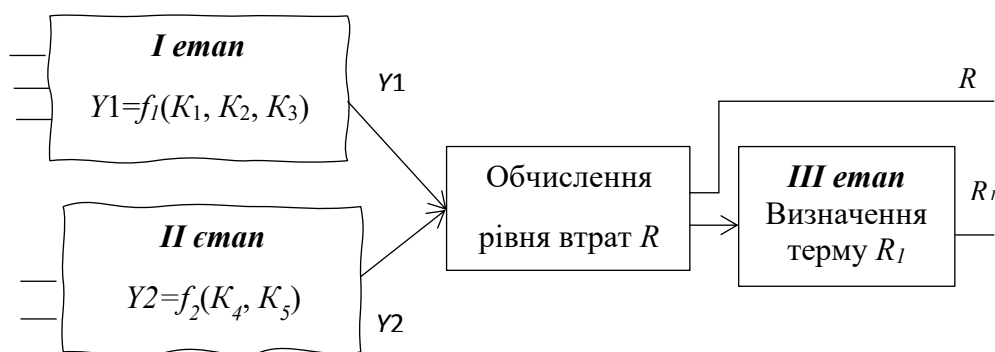


Рисунок 3.3 – Структура системи підтримки прийняття рішень

Вихідна змінна $Y1$ – ймовірності виникнення загрози, приймаємо ЛЗ «Ймовірність виникнення загрози»: «Дуже низька» (0-0,3), «Низька» (0,2-0,5), «Середня» (0,4-0,7), «Висока» (0,6-0,9), «Дуже висока» (0,8-1). В результаті задані вихідні лінгвістичні змінні для першого етапу, визначено множину

термів та їх функції належності.

Значення $Y_1 = f_1(K_1, K_2, K_3)$ залежить від змінних K_1 , K_2 та K_3 тому розглянемо їх розрахунки.

Для оцінки K_1 – привабливість активів, використовувалася таблиця з відповідями десяти експертів за 10-бальною шкалою. Приклад розрахунку надано у розділі 2.

Критерій K_2 – існуючий контроль визначаємо за формулою

$$K_2 = \frac{N_m}{8},$$

де N_m – кількість методів захисту інформації, що є у ІС.

Критерій K_3 – попередні інциденти. Числові значення змінних K_3 варіюються в діапазоні $[0, 100]$ і визначаються відсотком комп'ютерів, атакованих в системі за рік

$$K_3 = \frac{N_p}{40\%},$$

де N_p – відсоток атакованих комп'ютерів системі за рік, а 40 % – максимально допустиме у галузі безпеки значення.

Для оцінки рівня завданої шкоди Y_2 були використані наступні функції:

$$K_4 = \frac{ALE}{D} 100\%,$$

де ALE – очікувані річні збитки; D – річний дохід підприємства. Для K_4 розрахунку використовуємо метод повернення інвестицій [16].

Отримаємо нормовані значення критерію: «Дуже низький» (Незначний збиток, менше 1% річного доходу), «Низький» (Низький збиток, 2-4 % річного доходу), «Середній» (Помітний збиток, 4-7% річного доходу), «Високий» (Велика шкода, 7-10 % річного доходу), «Дуже високий» (Дуже великий збиток, більше 10 % річного доходу).

$$K_5 = \frac{P}{D} 100\%,$$

де P відображає збитки через негативне ставлення клієнтів, партнерів та

інвесторів до компанії за останній рік; D – річний дохід [17].

Визначимо умови для ЛЗ «Репутаційна шкода»: «Дуже низький» (Незначний збиток, менше 1 % річного доходу), «Низький» (Низький збиток, 2–4 % річного доходу), «Середній» (Помітний збиток, 4–7 % річного доходу), «Високий» (Висока шкода, 7–8 % річного доходу), «Дуже високий» (Велика шкода, 7–10 % річного доходу).

Вихідна змінна Y_2 – рівень завданої шкоди та їх значення для вихідної змінної «Рівень завданої шкоди»: «Дуже низький» (0–0,3), «Низький» (0,2–0,5), «Середній» (0,4–0,7), «Високий» (0,6–0,9), «Дуже високий» (0,6–0,9). За допомогою цих змінних визначається рівень завданої шкоди.

При прийнятті рішень і прогнозуванні можливих результатів доволі часто складність полягає в опрацюванні великого обсягу вхідних даних, тому необхідно визначити вагу кожної вхідної ЛЗ, що входять до правил.

У роботі розрахунок ваги кожного критерію здійснюється за допомогою методу парних порівнянь Сааті – статистичного інструменту, який використовується для оцінки відносних переваг між різними варіантами або альтернативами [15].

Приклад розрахунку надано у розділі 2, а отримані результати надано у табл. 3.1 – 3.3.

Таблиця 3.1 – Результати оцінок експертів за коефіцієнтами парного порівняння, власним вектором і ваговими значеннями за критеріями K_1 , K_2 та K_3 .

		Привабливість активів (K_1)	Існуючий контроль (K_2)	Попередні інциденти (K_3)
1	K_1	1,0	2,0	0,3
	K_2	0,5	1,0	0,5
	K_3	3,0	2,0	1,0
	e_i	1,14	1,59	0,55
	ω_i	0,3487	0,4836	0,1677
2	K_1	1,0	1,0	0,3
	K_2	1,0	1,0	0,5
	K_3	2,0	3,0	1,0
	e_i	1,26	1,14	0,55
	ω_i	0,3474	0,4434	0,1692
3	K_1	1,0	0,5	1,0
	K_2	2,0	1,0	0,5
	K_3	1,0	2,0	1,0
	e_i	1,26	1,0	0,79
	ω_i	0,4126	0,79	0,2599
4	K_1	1,0	0,5	0,5
	K_2	0,5	1,0	0,3
	K_3	2,0	3,0	1,0
	e_i	1,0	1,14	0,55
	ω_i	0,3711	0,4247	0,2042
5	K_1	1,0	0,5	1,0
	K_2	2,0	1,0	0,5
	K_3	1,0	2,0	1,0
	e_i	1,26	1,0	0,79
	ω_i	0,5396	0,2970	0,1634
	1 ω_i	0,3487	0,4836	0,1677
	2 ω_i	0,3474	0,4434	0,1692
	3 ω_i	0,4126	0,3275	0,2599
	4 ω_i	0,3711	0,4247	0,2042
	5 ω_i	0,5396	0,2970	0,1634
	ω_i	0,4039	0,3952	0,1929

Таблиця 3.2 – Результати оцінок експертів за ваговими значеннями за критеріям K_4, K_5 .

Експерт	Фінансові витрати (K_4)	Збитки репутації (K_5)
1	0,6667	0,3333
2	0,3333	0,6667
3	0,6667	0,3333
4	0,7500	0,25
5	0,5	0,5
ω_i	0,5417	0,4167

Таблиця 3.3 – Вагові коефіцієнти впливу критеріїв K_1, K_2 та K_3 .

	Привабливість активів (K_1)	Існуючий контроль (K_2)	Попередні інциденти (K_3)
ω_i	0,4226	0,3952	0,1929

На рис. 3.4 показана побудова лінгвістичних змінних з застосуванням набору інструментів MATLAB Fuzzy Logic для визначення Y_1 – ймовірності виникнення загроз.

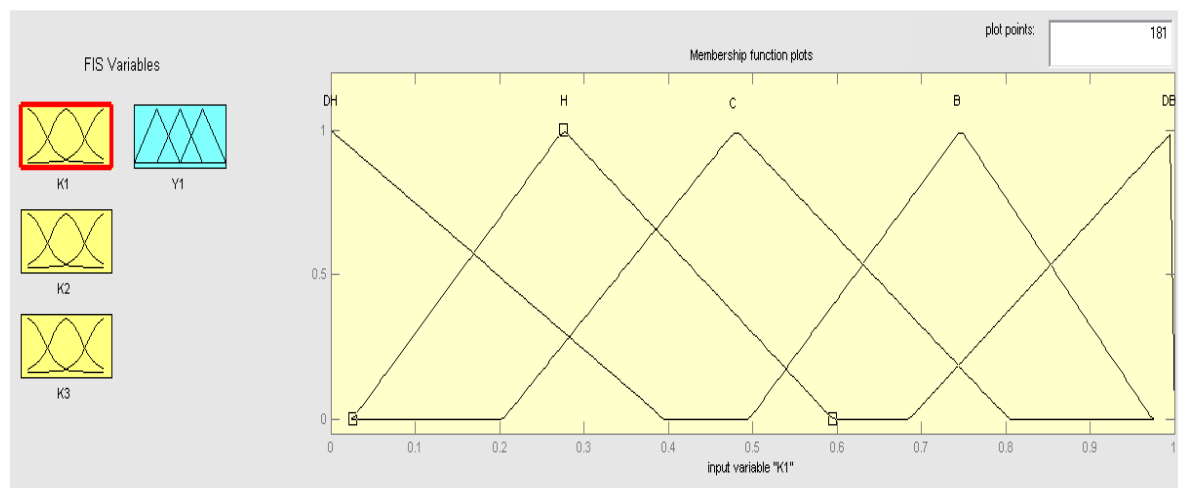


Рисунок 3.4 – Побудова системи ЛЗ для визначення «Привабливість активу»

Для решти критеріїв обираємо форму ЛЗ у вигляді трапеції, бо присутність на вершині площини характеризує постійність значення, що логічне при описі функцій для критеріїв $K_2, - K_5$ та Y_1 та Y_2 .

Аналогічним чином будується система для розрахунку оцінки шкоди, завданої активам, що захищаються – Y_2 .

3.3 Розробка бази нечітких правил СППР

Для роботи двох систем нечіткого висновку потрібно сформулювати логічні правила, за якими системи і будуть функціонувати.

База правил систем нечіткого висновку формується на основі попередньо визначених вхідних і вихідних лінгвістичних змінних.

Інформаційна база нечітких продукційних правил для оцінки Y_1 ймовірність виникнення загроз спочатку містила 125 правил. Після агрегації стало 40 правил. На рис. 3.5 надано фрагмент правил для визначення Y_1 .

1. If (K1 is H) and (K2 is DH) and (K3 is DH) then (Y1 is DH) (1)
2. If (K1 is H) and (K2 is C) and (K3 is DH) then (Y1 is H) (1)
3. If (K1 is H) and (K2 is C) and (K3 is H) then (Y1 is H) (1)
4. If (K1 is C) and (K2 is DH) and (K3 is DH) then (Y1 is H) (1)
5. If (K1 is C) and (K2 is DH) and (K3 is H) then (Y1 is H) (1)
6. If (K1 is C) and (K2 is DH) and (K3 is C) then (Y1 is H) (1)
7. If (K1 is C) and (K2 is DH) and (K3 is H) then (Y1 is H) (1)
8. If (K1 is C) and (K2 is H) and (K3 is DH) then (Y1 is H) (1)
9. If (K1 is C) and (K2 is H) and (K3 is H) then (Y1 is H) (1)
10. If (K1 is H) and (K2 is DH) and (K3 is DH) then (Y1 is H) (1)

Рисунок 3.5 – Фрагмент бази правил для визначення Y_1 – ймовірності виникнення загроз

При цьому градації K_1 : дуже низький – 0,2, Низький – 0,4, Середній – 0,6, Високий – 0,8; Дуже висока – 1,0.

Розрахункові значення меж термінів вихідної лінгвістичної змінної Y_1 – ймовірність виникнення загрози: дуже низька – [0,0; 0,3], низький – (0,3; 0,5], середній – (0,5; 0,7], високий – (0,7; 0,9], дуже високий – (0,9; 1,0]).

У результаті отримано основи нечітких правил для визначення ймовірності виникнення загрози та оцінки рівня завданої шкоди. Ці два параметри використовуються для розрахунку чіткого вихідного значення рівня ризику R .

Розроблено алгоритми для визначення ймовірності виникнення загроз, оцінки рівня збитків, а також базу нечітких правил.

Моделювання виконано за допомогою MATLAB, у додатку А надано процес створення першої підсистеми нечіткого висновку (рис. А.1-А.8).

Результати показали, що розроблена система здатна забезпечити адекватну оцінку ризиків інформаційної безпеки.

3.4 Запропонований метод урахування вагових коефіцієнтів нечітких лінгвістичних змінних

Для урахування вагових коефіцієнтів нечітких лінгвістичних змінних що входять бази правил нечіткої системи у роботі запропоновано задіяти величину $\sup_{x \in X} \mu_a(x)$, що визначає висоту нечіткої множини та зазвичай, для нормальної множини, дорівнює 1. Чітке рішення формується за допомогою, наприклад, алгоритму Mamdani, де на основі бази правил формується площина можливих рішень і чітке рішення – це проекція геометричного центру площини на вісь X.

Припустимо, що при значенні $\sup_{x \in X} \mu_a(x) = 1$ всі критерії мають однаковий вплив на рішення, тоді отримані раніше вагові коефіцієнти впливу $\omega_i, i = 1, \dots, 5$ для критеріїв $C_1 - C_5$ можливо застосувати для обчислення зміни висоти відповідної лінгвістичної змінної таким чином

$$\mu_i^{\text{впл}}(C_i) = (1 - \omega_i), i = 1, \dots, 5$$

Для випадку застосування алгоритму Mamdani (метод центроїди) це припущення працює і для тестового прикладу для перевірки підходу було створено два варіанта системи підтримки прийняття рішень що надано на рис. 2.1. За допомогою першого варіанту рішення було отримано без урахування вагових коефіцієнтів, а другого з ваговими коефіцієнтами.

Отримані рішення надано у табл.3.4. – 3.5.

Таблиця 3.4 – Результати обчислення Y1 (без вагових коефіцієнтів)

K_1	K_2	K_3	Y1
0.343	0.783	0.102	0.393
0.0783	0.861	0.102	0.425
0.476	0.127	0.849	0.33
0.476	0.849.	0.849	0.499
0.46	0.331	0.488	0.389

Таблиця 3.5 – Результати обчислення Y1 (з ваговими коефіцієнтами)

K_1	K_2	K_3	Y1
0,5	0,5	0,5	0,435
0.343	0.783	0.102	0.425
0.0783	0.783	0.102	0.495
0.476	0.127	0.849	0.425
0.44	0.331	0.488	0.432

Аналіз результатів показав, що запропонований підхід до урахування нечітких вагових коефіцієнтів дійсно працює. Результати тестової задачі показали зміну рішення від 3 до 10 %. Для технічної галузі це велика частка. Для подальшого розвитку даного підходу потрібно провести додаткові дослідження на реальних прикладах застосування.

Результати дослідження цього розділу надано у [18].

У розділі показана практична побудова системи підтримки прийняття рішень (СППР), де:

- оцінка ризику реалізована як послідовний процес: спочатку обчислюється ймовірність загроз, потім оцінюється потенційна шкода, а на фінальному етапі визначається чіткий показник втрат та відповідний йому лінгвістичний терм.

- інтеграція критеріїв: об'єднання 10 різнорідних факторів безпеки у п'ять ключових комплексних критеріїв (привабливість активів, існуючий контроль, попередні інциденти, фінансові витрати та втрата репутації), що спростило структурування нечітких правил.

- запропоновано новий метод урахування вагових коефіцієнтів через зміну висоти нечіткої множини. Це дозволяє системі «зважувати» думки експертів, що підвищує точність результату на 3–10% порівняно зі звичайними моделями нечіткого висновку.

- сформовано базу нечітких правил, яка після агрегації містить 40 правил виду «Якщо... То...». Це дозволяє системі імітувати логіку досвідченого аналітика безпеки при обробці великих обсягів вхідних даних.

Апробація підходу в середовищі MATLAB (Fuzzy Logic Toolbox) довела працездатність системи та її здатність надавати адекватні оцінки ризиків навіть для складних, багатофакторних систем.

4. Розробка концепції системи виявлення вторгнень на основі методів штучного інтелекту

4.1 Системи автоматизації захисту інформації

Сучасний світ значною мірою залежить від інформаційних технологій, що робить питання кібербезпеки надзвичайно важливим [19]. Різноманітні кібератаки стають дедалі складнішими, що загрожує як окремим користувачам, так і цілим організаціям. Тому необхідність створення систем захисту (IDS) та розробки ефективних методів виявлення атак на основі методів штучного інтелекту є актуальною.

Згідно [20], інцидент ІБ стосується однієї або серії небажаних чи неочікуваних подій ІБ, які мають значну ймовірність поставити під загрозу бізнес-операції та створювати загрозу. У свою чергу, подія ІБ – це ідентифікована (спостережувана) подія стану системи, послуги або мережі, що вказує на негативний наслідок, такий як можливе порушення ІБ, політики, стандартної практики безпеки або збій контролю, або раніше невідома ситуація, яка може бути загрозою.

Автоматизовані системи (програмне забезпечення) для управління подіями ІС – SIEM-системи – використовуються для вирішення проблеми контролю потоку подій ІС, що надходять від інтегрованих систем контролю якості (ІРТ), та для комп'ютеризації процесу управління інцидентами ІС. Ці системи є критично важливими для ІС організації. Організаціям будь-якого розміру зазвичай потрібна SIEM-система для цілей дотримання вимог, щоб автоматично генерувати звіти, що підтверджують дотримання організацією різних вимог до дотримання вимог. Для повного та правильного виконання поставлених завдань SIEM-системи потребують частого налаштування та адаптації, оскільки вони працюють у постійно мінливому, динамічному середовищі. Індикатори (у вигляді відповідних шаблонів) компрометації ресурсів інтрамережі розгортаються як сповіщення в SIEM-системах.

Системи SEM збирають події в режимі реального часу, проводять їх аналіз майже в режимі реального часу, надсилають сповіщення та відображають

інформацію на консолі оператора для швидшого вжиття захисних заходів. SEM орієнтований на негайність, тоді як SIM більше орієнтований до ведення історичних записів. Комбінована система SIEM збирає журнали та іншу інформацію, пов'язану з ІС, для аналізу.

Ключові функції цих трьох систем можна підсумувати наступним чином:

SIM – збір журналів, архівування, історична звітність, криміналістика;

SEM – звітність у режимі реального часу, збір журналів, нормалізація, кореляція, агрегація;

SIEM – збір журналів, нормалізація, кореляція, агрегація, звітність.

Збір записів подій у журналах з різних джерел в інтрамережі надає інструменти комп'ютерної криміналістики та допомагає виконувати вимоги щодо звітності щодо відповідності. Нормалізація відображає повідомлення журналів з численних систем у спільну модель даних, що дозволяє організаціям поєднувати та аналізувати пов'язані події, незважаючи на їхні початково різні формати джерел журналів. Кореляція пов'язує журнали та події з різних систем або програм, пришвидшуючи виявлення загроз ІС та реагування на них. Агрегація зменшує обсяг даних про події шляхом консолідації дублікатів записів подій. Звітність представляє корельовано, агреговані дані про події в режимі моніторингу в режимі реального часу та довгострокових зведень.

У SIEM-системах використовуються два основні підходи: 1) безагентний, коли хост, що генерує журнали, безпосередньо передає свої журнали до SIEM або проміжного сервера журналювання задіяні, такі як сервер системних журналів; та 2) агентний, з програмним агентом, встановленим на кожному хості, який генерує журнали та відповідає за вилучення, обробку та передачу даних на сервер SIEM.

Більшість SIEM-систем працюють, розгортаючи кілька агентів збору даних в ієрархічному порядку для збору подій, пов'язаних з ІС, з ІРТ, мережевого обладнання, пристроїв кінцевих користувачів, серверів тощо. Колектори пересилають події до централізованої консолі керування, яка виконує перевірки та позначає аномалії.

Системи SIEM можуть бути засновані на правилах (очевидними недоліками цього підходу є трудомісткість підтримки сотень правил в актуальному стані, занадто багато хибнопозитивних та хибнонегативних результатів для постійно інноваційних методів зловмисників), засновані на політиках або мати механізм статистичної кореляції для встановлення зв'язків між записами журналу подій [21, 22].

Усі дані, що обробляються системою SIEM, повинні бути захищені, оскільки вони містять дуже конфіденційна інформація, необхідна для цифрової криміналістики та реагування на інциденти ІБ.

Завдяки інтеграції з іншими IPT, системи SIEM можуть служити єдиним вікном для контролю інцидентів ІС будь-якої організації. Системи SIEM, що використовуються для постійного моніторингу подій та активності користувачів, можуть виявляти та обробляти інциденти ІС шляхом агрегації великих обсягів машинних даних у режимі реального часу для управління ризиками ІС та суттєво покращувати цю автоматизацію.

Системи SIEM також можуть візуалізувати загрози ІС для ресурсів інтрамережі, які в іншому випадку могли б бути приховані. Саме тому вони вважаються ядром SOC. Визначимо SOC як централізований підрозділ, який займається питаннями безпеки на організаційному рівні, плюс команда, що складається переважно з аналітиків безпеки (плюс кількох операторів), організованих для виявлення, аналізу, реагування, звітування та запобігання інцидентам ІС з метою мінімізації ризиків ІС. Ризик ІС пов'язаний з можливістю того, що загрози ІС використовуватимуть вразливості інформаційного активу/групи інформаційних активів і тим самим завдадуть шкоди організації. Ризик - це наслідок невизначеності щодо цілей [20].

Системи SIEM можуть допомогти досягти наступних цілей в управлінні ІС інтрамережі:

- комп'ютеризувати діяльність з аналізу та звітності ІС інтрамережі відповідно галузевих та міжнародних стандартів й норм;
- отримувати інформацію про реальний стан ІБ по всій інтрамережі

організації та певних активів;

- пришвидшувати реагування на виникнення подій та інцидентів, пов'язаних з ІБ, та забезпечувати інтрамережвий зв'язок ІБ цілодобово шляхом автоматизованого реагування на події ІБ відповідно до заздалегідь визначених правил обробки та кореляції;

- проводити обґрунтовану оцінку ризиків ІС в організації та своєчасно їх усувати визначати або зменшувати ризики ІС на основі цієї оцінки;

- виявляти відмінності та приводити активи й бізнес-процеси інтрамережі у відповідність до внутрішніх політик ІС, вимог регуляторних органів та аудиторів;

- формалізувати та впровадити ефективне прийняття рішень у сфері ІБ;

- усунути необхідність збільшення кількості слухань щодо ІБ відповідно до зростання кількості інформаційних груп, що надають інформацію про події, пов'язані з інтелектуальною безпекою;

Зменшити витрати на навчання персоналу ІС, оскільки виникає необхідність вивчати лише інтерфейс такої системи, а не всі різноманітні ПТТ, що використовуються в інтрамережах.

Нова реальність частіших та складніших атак і «хакінгу як послуги» робить злом інтрамереж більш професійним, доступним та небезпечно ефективним. Багато організацій починають усвідомлювати, що недостатньо просто аналізувати дані, пов'язані з ІС; вони також повинні вживати заходів щодо них.

Вони повинні протистояти цьому за допомогою належним чином розроблених та централізованих систем управління ІС. Це може робитися напівавтоматично або автоматично, щоб отримувати практичну інформацію, і може включати профілактичне обслуговування, наприклад, для моніторингу своїх активів на наявність проблем на основі минулих шаблонів або правил та ініціювання сповіщень, які можуть покращити безпеку.

Впровадження такого підходу та максимально автоматизація всіх рутинних операцій та реагування на інциденти ІС, що не потребують прийняття рішень експертами, є нагальною потребою для будь-якої сучасної організації у

створенні центру управління ІС, який би вирішував ці проблеми та був би більш просунутим, ніж традиційний SOC. Так званий SIC з інтегрованою архітектурою захисту від атак забезпечує повну видимість, контроль та контекстно-орієнтовану аналітику безпеки в одному місці для тимчасового вирішення подій ІС на мережевому рівні та важливіших подій вищого рівня. Впроваджуючи SIC, організації отримують цілісне поглиблене уявлення про «стан ІС» своєї інтрамережі та здатні не лише виявляти та розпізнавати атаки, але й ефективно реагувати на загрози ІС, перш ніж вони завдадуть шкоди, та запобігати інцидентам ІС, постійно збираючи та обробляючи знання про мережеві атаки.

Завдяки власному приватному SIC, заснованому на системі SIEM другого покоління, організація може отримати персоналізоване управління мережевою безпекою.

Системи SIEM другого покоління (SIEM 2.0) почали працювати наприкінці 2000-х років та виконували поведінковий та контекстний аналіз і реалізувати такі основні функції:

1. Виявлення в режимі реального часу та централізований збір інформації про події ІС з усіх джерел розподіленої гетерогенної інтрамережі: ПТПП (програмних та апаратних), мережевих пристроїв, програм, баз даних, файлів конфігурації тощо;

2. Моніторинг активності користувачів та програм у певному контексті;

3. Обробка зібраної інформації (включаючи її фільтрацію, агрегацію, нормалізацію, кореляцію тощо) у певному контексті, враховуючи попередню та поточну активність користувачів і програм, а також накопичену статистику;

4. Відстеження всього життєвого циклу кожного інциденту ІБ – аналіз та автоматичне виконання певних дій у відповідь на події ІБ, класифіковані як інциденти ІБ;

5. Автоматизована генерація звітів та рекомендацій щодо обробки інцидентів та подій ІБ (включаючи відстеження статусу їх вирішення) та зведених звітів на рівні ІБ інтрамережі;

6. Використання великих даних з аналітикою в пам'яті та базі даних, масивним паралельним програмуванням тощо (замість реляційних баз даних) для масштабованої аналітики ІС.

Маючи все це в рамках SIC, організації отримують такі ключові переваги:

Узгодження управління ризиками ІС з потребами бізнесу на основі попередньо визначених змістовних показників ІС та цілодобового охоплення безпеки, що поєднує локальні спостереження моніторингу, безперервно записану історію, зовнішню та внутрішню аналітику загроз ІС, міжканальний огляд в єдиному вигляді та аналіз зв'язків між об'єктами для виявлення прихованих зв'язків та підозрілих асоціацій між користувачами, обліковими записами або іншими об'єктами на ранніх стадіях їхнього життєвого циклу в одному місці без необхідності повного штату;

Цілісний проактивний та прогнозний підхід з поведінковою крос-кореляцією та розширеною контекстною (не подійно-орієнтованою) аналітикою, що означає, що організація розглядає кожен аспект управління загрозами ІС у зв'язку з усіма іншими аспектами, розглядає ІС як більше, ніж просто питання зниження ризику ІС шляхом виявлення та виправлення вразливостей мережевих активів, а також врахування можливостей і мотивів загроз ІС проти активів інтрамережі;

Наразі навіть ідея SIC не встигає за зростаючою кількістю складних загроз ІС у високогетерогенному пов'язаному світі. Наступний еволюційний крок до створення більш ефективної структури управління ІС для захисту ІТ-активів організацій очікується як ніколи раніше. Ця прогресивна структура повинна об'єднати всі переваги SIC з багаторічним досвідом управління мережевими операціями, впровадженим у NOC – центрах мережевих операцій.

Більш цілеспрямований підхід зосереджує свої ресурси на конкретних загрозах мережевій безпеці, надаючи пріоритет надлишковим численним рівням ІС, запозиченим зі стратегій «глибинної оборони» у військових справах, та стратегічно організуючи ці рівні.

Кіберзлочинці використовують різні типи атак для компрометації систем,

наприклад [20]:

- DoS/DDoS-атаки – перевантажують ресурси системи, роблячи їх недоступними для користувачів;

- SQL-ін'єкції – дозволяють атакувальникам маніпулювати базами даних через вразливі веб-додатки;

- XSS (міжсайтовий скриптинг) – дозволяє впроваджувати шкідливий код у веб-додатки для крадіжки даних користувачів;

- Атаки грубої сили – націлені на підбір паролів через автоматизоване введення численних варіантів;

- Атаки «людина посередині» (Man-in-the-Middle) – дозволяють зловмисникам перехоплювати та змінювати комунікації між двома сторонами без їхнього відома.

Для виявлення атак використовують сигнатурний аналіз, що базується на порівнянні вхідного трафіку з вже відомими сигнатурами атак, що дозволяє швидко ідентифікувати загрози.

Сигнатура атаки – це унікальний «відбиток» або шаблон (набір байтів, послідовність команд або патерн трафіку), який дозволяє системам захисту (IDS/IPS, антивірусам, WAF) розпізнати відому загрозу.

4.2 Сигнатури атак та їх застосування

Приклади сигнатур для різних типів атак:

1. Мережеві атаки (IDS/IPS)

Сигнатури часто пишуться на мові системи Snort. Вони перевіряють заголовки пакетів і їх зміст. Наприклад, пошук SQL-ін'єкції в HTTP-запиті:

```
snort
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SQL Injection detected"; content:"select"; nocase; content:"from"; nocase; sid:1000001;)
```

Фрагмент програми шукає наявність слів *select* і *from* (без урахування регістру) в TCP-пакетах, наявних на веб-сервері []. SecurityLab.ru

2. Веб-атаки (WAF)

Сигнатури націлені на конкретні аномалії в URL або запиті.

Атака Cross-Site Scripting (XSS) - шаблон шукає тег <script> або обробники подій (наприклад, onload):

сигнатура: *<script>.*alert\(.*\).*

Атака Path Traversal (обхід директорії): - спроба вийти за межі корневої папки сайту:

сигнатура: (\\.\\.|\\.\\.\\|)+ (пошук послідовностей ../ або ..\)

3. Шкідливе ПО (Антивіруси) - антивіруси шукають конкретні послідовності байтів у файлах (бінарні сигнатури), наприклад, спрощена байтова сигнатура:

Якщо вірус завжди містить код E8 00 00 00 00 58 05 33, антивірус помітить будь-який файл із такою послідовністю як загрозу.

Хеш-сума: найпростіший вид сигнатури – це MD5 або SHA-256 хеш шкідливого файлу. Якщо хеш співпадає, файл блокується.

4. Сигнатури логов (SIEM)

Відслідковування підозрілих дій в системних журналах, наприклад, підбір пароля (Brute-force): умова: Більше 5 подій "Невдалий вхід" (Event ID 4625 в Windows) від одного IP за 60 секунд.

Перевагами застосування сигнатур – висока точність і мінімальна кількість помилкових обробок на вже відомі атаки.

Однак цей метод неефективний проти нових або модифікованих атак [20].

Поведінковий аналіз аналізує звичні патерни поведінки користувачів та системи, виявляючи аномалії, які можуть свідчити про атаку.

Наприклад, детектування сканування портів відрізняється від пошуку тексту пакеті. Система захисту (IDS) має не просто дивитися всередину одного пакета, а аналізувати поведінку за проміжок часу.

Найпопулярніший вид сканування – TCP SYN-сканування (приховане). Хакер посилає SYN-пакет, чекає на відповідь і, не завершуючи з'єднання, переходить до наступного порту.

1. Логіка сигнатури (Thresholding)

У сучасних системах (типу Snort 3 або Suricata) це реалізується через поріг

спрацьовування (threshold).

Для фіксації ситуації, що з однієї IP-адреси надходить багато запитів на різні порти за короткий час, можна застосувати таке правило:

```
snort
alert tcp $EXTERNAL_NET any -> $HOME_NET any (
  msg:"Possible TCP Port Scan";
  flags:S;
  threshold:type threshold, track by_src, count 20, seconds 10;
  sid:1000003;
  rev:1;
)
```

де flags:S; - шукає пакети лише з прапором SYN (запит на відкриття з'єднання).

- track by_src - система запам'ятовує джерело (IP атакуючого).

- count 20, seconds 10 - якщо з цього IP надійшло 20 SYN-пакетів за 10 секунд, правило спрацює.

3. Детектування "Дивних" сканувань

Іноді сканери (наприклад Nmap) використовують некоректні комбінації прапорів, щоб обійти прості фільтри.

Пакет Xmas Scan

```
alert tcp any any -> $HOME_NET any (msg:"Xmas Scan Detected"; flags:FPU;
sid:1000004;)
```

Тут FPU означає, що включені прапори FIN, PUSH та URG одночасно - у нормальному трафіку так не буває).

Null Scan (немає прапорців):

```
snort
alert tcp any any -> $HOME_NET any (msg:"Null Scan Detected"; flags:0; sid:1000005;)
```

Головна проблема сигнатур на порти – хибні спрацьовування. Наприклад, якщо поштовий сервер або активний веб-клієнт відкриває багато з'єднань відразу, IDS може прийняти це за атаку. Тому параметри count та seconds завжди підлаштовують під конкретну мережу.

Але, можна виявити таке сканування, зробивши статистичний аналіз

логів в SIEM-системі.

У SIEM-системах (наприклад, ELK Stack, Splunk чи Wazuh) працюють з готовими подіями від міжмережевого екрану (Firewall), який зафіксував відмову в з'єднанні (Deny/Reject). Логіка ясна якщо один зовнішній IP стукає в закриті порти на різних внутрішніх адресах, - це сканування:

Це типовий приклад текстових повідомлень від Cisco ASA. SIEM отримує їх у такому вигляді:

```
Oct 12 10:01:05 fw-01 %ASA-6-305011: Built dynamic TCP translation... Повідомлення про створення трансляції (дозвіл з'єднання).
```

```
Oct 12 10:01:06 fw-01 %ASA-4-106023: Deny tcp src outside:203.0.113.5 dst inside:192.168.1.50/22... Блокування спроби підключення до порту 22 (SSH).
```

```
Oct 12 10:01:07 fw-01 %ASA-4-106023: Deny tcp src outside:203.0.113.5 dst inside:192.168.1.50/23... Блокування спроби підключення до порту 23 (Telnet).
```

2. Приклад правила (Sigma)

Щоб SIEM зрозуміла, що це не просто випадкові помилки, а сканування портів, використовуємо правило Sigma. Воно групує ці події за джерелом (IP) та підраховує кількість різних портів, до яких звертався зловмисник.

```
title: Виявлення сканування портів (з виключеннями)
```

```
logsource:
```

```
  product: firewall
```

```
  service: cisco_asa
```

```
detection:
```

```
  selection:
```

```
    event_id: '106023'
```

```
    action: 'deny'
```

```
  filter_internal_scanners:
```

```
    src_ip:
```

```
      - '192.168.1.100' # IP вашого Nessus або Rapid7
```

```
      - '10.0.5.50'    # IP адміністратора мережі
```

```
  condition: selection and not filter_internal_scanners | count(dst_port) by src_ip > 10
```

```
timeframe: 1m
```

```
level: medium
```

Програма постійно моніторить потік логів з фаєрвола (cisco_asa). Шукає лише події з кодом 106023 та статусом deny. Це критично важливо, щоб не аналізувати дозволений трафік і не перевантажувати пам'ять SIEM.

Механізм фільтрації:

Перевірка на легітимність: Перед тим як підозрювати IP в атаці, програма звіряє src_ip із переліком «довірених» адрес. Якщо IP співпадає з вашим сканером (наприклад, 192.168.1.100), програма миттєво відкидає цю подію пів вона не потрапляє в лічильник.

Агрегація та поріг:

Це «мозок» правила. SIEM створює в оперативній пам'яті тимчасову таблицю. Групує всі заблоковані спроби за адресою відправника (by src_ip). Рахує кількість унікальних портів призначення (count(dst_port)), до яких цей відправник намагався підключитися.

Умова спрацювання: Якщо за 60 секунд (1m) один IP отримав відмову на 11 або більше різних портів, генерується Alert.

Основний недолік висока кількість хибно позитивних спрацювань [20].

Евристичний аналіз застосовує евристичні правила та алгоритми для ідентифікації потенційних загроз, навіть якщо вони не мають явних сигнатур. Цей підхід використовується у сучасних антивірусних системах та засобах захисту мереж [21].

Імітаційне моделювання дозволяє тестувати нові підходи та методи для виявлення атак, аналізувати поведінку загроз і оцінювати ефективність захисних механізмів без загрози реальним системам [22].

Метою даного розділу є аналіз існуючих методів виявлення атак та визначення застосування можливих методів штучного інтелекту.

Сучасні розробки систем виявлення вторгнень (IDS – Intrusion Detection Systems) застосовують алгоритми аналізу трафіку, виявлення аномалій та, для підвищення ефективності виявлення атак, використовують комбінацію алгоритмів штучного інтелекту [23-27]:

- методи машинного навчання (наприклад, дерева рішень, нейронні

мережі) для аналізу поведінкових патернів трафіку;

- системи глибокого навчання для розпізнавання складних атак на основі великих обсягів даних;

- методи кластеризації та аномального детектування, такі як DBSCAN або Isolation Forest, для виявлення відхилень від норми.

Імітаційне моделювання у кібербезпеці є ключовим інструментом для аналізу загроз і розробки ефективних методів захисту. Воно дозволяє створювати штучні середовища, що імітують реальні інформаційні системи, у яких можна тестувати різні типи атак та методи їхнього виявлення.

Основними принципами цього підходу є:

- реалістичність моделювання: створення сценаріїв, що максимально наближені до реальних загроз;

- гнучкість: можливість адаптації моделей до нових загроз та змін у кіберпросторі;

- автоматизація: використання програмних засобів для відтворення та аналізу атак у великому масштабі.

IDS являє собою програмний або апаратний засіб, призначений для виявлення фактів неавторизованого доступу в комп'ютерну систему або мережу, або несанкціонованого управління ними головним чином через Інтернет. Системи виявлення вторгнень використовуються для виявлення деяких типів шкідливої активності, яка може порушити безпеку комп'ютерної системи. До такої активності відносяться мережеві атаки проти вразливих сервісів, атаки, спрямовані на підвищення привілеїв, неавторизований доступ до важливим файлам, а також дії шкідливого програмного забезпечення (комп'ютерних вірусів, троянів та черв'яків).

На сьогоднішній день існує багато різних методологій для моделювання інформаційних ризиків: STRIDE, PASTA, Trike, VAST та OCTAVE. Найпопулярніша з них STRIDE дозволяє класифікувати загрози за характеристиками [24]:

- spoofing (підробка ідентичності) – визначення спроб

несанкціонованого доступу;

- tampering (зміна даних) – виявлення змін у переданій інформації;
- repudiation (відмова від відповідальності) – аналіз логів для виявлення дій, які можуть бути заперечені користувачем;
- information Disclosure (розкриття інформації) – моніторинг витоків конфіденційних даних;
- Denial of Service (відмова в обслуговуванні) – виявлення атак типу DoS/DDoS.
- Elevation of Privilege (підвищення привілеїв) – відстеження спроб отримати доступ із розширеними правами.

У сфері кібербезпеки та моніторингу логів (SIEM), штучний інтелект (ШІ) та машинне навчання (ML) потрібні там, де статичні правила (як Sigma) стають занадто складними або неефективними.

4.3 Приклади застосування ШІ для аналізу мережевої активності

1. Пошук аномалій (Anomaly Detection)

Замість того, щоб вручну прописувати поріг «10 спроб на хвилину», ШІ вивчає базову лінію (Baseline) мережі. Система знає, що сервер А зазвичай спілкується з 5 серверами. Раптом він починає звертатися до 50 нових вузлів. ШІ піднімає тривогу про «горизонтальне переміщення» (Lateral Movement) зловмисника, навіть якщо він діє дуже повільно і не підпадає під звичайні фільтри.

2. Виявлення алгоритмів генерації доменів (DGA)

Шкідливе ПЗ часто використовує динамічні домени (наприклад, asdfg123.com), щоб зв'язатися з командним центром (C2). Нейромережа (наприклад, LSTM або CNN) аналізує назви доменів у логах. Вона здатна відрізнити «людську» назву (google.com) від випадково згенерованої «машинної» назви, яку неможливо передбачити заздалегідь чорними списками.

3. Кластеризація подій та боротьба з «алертною втомою»

Якщо на фаєрволі відбувся збій, він може висипати мільйон логів за хвилину. Аналітик фізично не зможе їх прочитати. Алгоритми кластеризації

групуєть мільйон схожих логів в один інцидент. ШІ скаже: «Це не мільйон подій, це одна проблема з сервером X, ось її першопричина».

4. Передбачення наступного кроку (Predictive Analytics)

На основі зібраних даних (наприклад, фреймворку MITRE ATT&CK), ШІ може оцінити ланцюжок дій.

Приклад: «Зафіксовано сканування портів -> зафіксовано вхід під адміном -> ШІ прогнозує спробу вивантаження даних (Exfiltration) і пропонує автоматично заблокувати обліковий запис».

5. Аналіз поведінки користувачів (UEBA)

ШІ створює «цифровий портрет» кожного співробітника. Оператор зазвичай працює з 9:00 до 18:00 і використовує лише 1С. Якщо о 3-й ночі з-під цього акаунта почнеться сканування мережі (навіть успішне), ШІ зрозуміє, що це аномальна поведінка, хоча пароль введено вірно.

Приклад коду на Python, який за допомогою простого ML-алгоритму шукає аномалії в IP-адресах. Застосуємо алгоритм Isolation Forest (Ізоляційний ліс) що знаходить «викиди» (anomalies) у даних, не потребуючи попереднього навчання на «поганих» прикладах.

Приклад виявлення аномальної активності IP.

```
python
import pandas as pd
from sklearn.ensemble import IsolationForest
# 1. Імітуємо дані з логів (IP та кількість запитів за хвилину)
data = {
    'ip': ['192.168.1.1', '192.168.1.2', '192.168.1.3', '192.168.1.4', '203.0.113.5'],
    'requests_per_min': [12, 15, 10, 14, 500], # 500 – явна аномалія
    'unique_ports': [2, 3, 1, 2, 85]      # 85 портів – теж підозріло
}
df = pd.DataFrame(data)
# 2. Готуємо ознаки для навчання (тільки цифри)
features = df[['requests_per_min', 'unique_ports']]
# 3. Ініціалізуємо модель ШІ
# contamination=0.1 означає, що ми очікуємо ~10% аномалій
```

```
model = IsolationForest(contamination=0.2, random_state=42)
# 4. Шукаємо аномалії
df['anomaly_score'] = model.fit_predict(features)
# Результат: -1 це аномалія, 1 це норма
df['status'] = df['anomaly_score'].map({1: 'Normal', -1: 'ANOMALY'})
print(df[['ip', 'requests_per_min', 'unique_ports', 'status']])
```

Як це працює з точки зору ШІ:

Алгоритм будує «дерева», щоб ізолювати кожну точку. Звичайні точки (10–15 запитів) знаходяться «глибоко» в лісі, їх важко відділити одну від одної. Аномалія (500 запитів) стоїть окремо, тому алгоритм ізолює її дуже швидко (за кілька «кроків»). ШІ автоматично позначає IP 203.0.113.5 як аномальний, навіть якщо ви не вказували йому конкретну цифру порогу.

Переваги алгоритму:

адаптується – якщо завтра вся мережа стане працювати швидше (наприклад, 50 запитів стане нормою), ШІ сам підлаштується під новий Baseline.

- бачить комбінації – окремо 50 запитів може бути нормою, і 10 портів нормою, але разом вони можуть дати сигнал аномалії.

Таку Python-перевірку просто інтегрувати безпосередньо у потік даних SIEM.

Для сфери детектування загроз (Detection Engineering) ШІ найчастіше використовується у двох ролях: як аналітик-помічник (Co-pilot) для написання правил та як двигун для виявлення аномалій, які неможливо описати жорстким кодом.

Приклад, як ШІ допомагає виявити одну з найскладніших атак – Beaconing (шкідливого зв'язку) коли вірус «стукає» до хакера через рівні проміжки часу. Звичайне правило (Sigma) тут безсиле, бо хакер може змінювати порти, IP та час. ШІ ж використовує математичну статистику.

1. Що бачить ШІ в логах фаєрвола:

Він аналізує не один рядок, а серію подій від одного IP за 24 години:

Запит о 10:00:01 (розмір пакета 500 байт)

Запит о 10:01:01 (розмір пакета 500 байт)

Запит о 10:02:01 (розмір пакета 500 байт)

2. Як працює алгоритм (на прикладі аналізу часових рядів):

ШІ обчислює ентропію (ступінь хаосу) інтервалів між запитами.

Звичайна людина заходить у браузер хаотично (через 5 сек, потім через 2 хв, потім через 10 сек). Ентропія висока. Вірус (Bot): Стукає рівно кожні 60 секунд. Ентропія майже нульова.

Приклад коду на Python.

```
python
import numpy as np
# Інтервали між запитами в секундах
human_behavior = [5, 120, 30, 450, 10] # Хаос
virus_behavior = [60, 61, 59, 60, 60] # Чітка періодичність
def detect_beaconing(intervals):
    # Рахуємо стандартне відхилення
    std_dev = np.std(intervals)
    if std_dev < 2: # Якщо похибка менше 2 секунд – це робот
        return "⚠️ АНОМАЛІЯ: Виявлено Beaconing (Бот)"
    return "✅ Нормальна активність"
print(detect_beaconing(virus_behavior))
```

Автоматичне збагачення (Enrichment): Коли прилітає лог з підозрілим доменом x7z-update.com, ШІ перевіряє дату реєстрації (якщо сьогодні – це атака) та репутацію без участі людини.

Класифікація логів: ШІ може розпізнати тип лога, навіть якщо розробник софту змінив його формат, і автоматично розкласти його по полях (IP, User, Action).

Переклад запитів: пишемо чат-боту: "Знайди мені всіх, хто качав більше 1ГБ за ніч", і ШІ сам генерує складний SQL або SPL запит до бази даних.

4.4 Розробка концептуальної структури системи виявлення вторгнення

На рис. 4.1 надано запропоновану концептуальну структуру системи

виявлення вторгнення, що складається з кількох ключових компонентів:

- модуль збору даних – здійснює моніторинг трафіку в реальному часі, збираючи інформацію про підозрілі активності;
- модуль попередньої обробки – фільтрує та нормалізує дані, видаляючи зайвий шум і підготовлюючи їх до аналізу;
- аналітичний модуль – включає алгоритми машинного навчання та методи виявлення аномалій для ідентифікації загроз;
- модуль ухвалення рішень – визначає рівень ризику атаки та пропонує відповідні заходи реагування.

Первинний збір даних здійснюють сенсори. Реєстраційна інформація може витягуватися з системних або прикладних журналів або видобуватися з мережі за допомогою відповідних механізмів активного мережевого обладнання або шляхом перехоплення пакетів посередництвом встановленої в режим моніторингу мережевої карти.

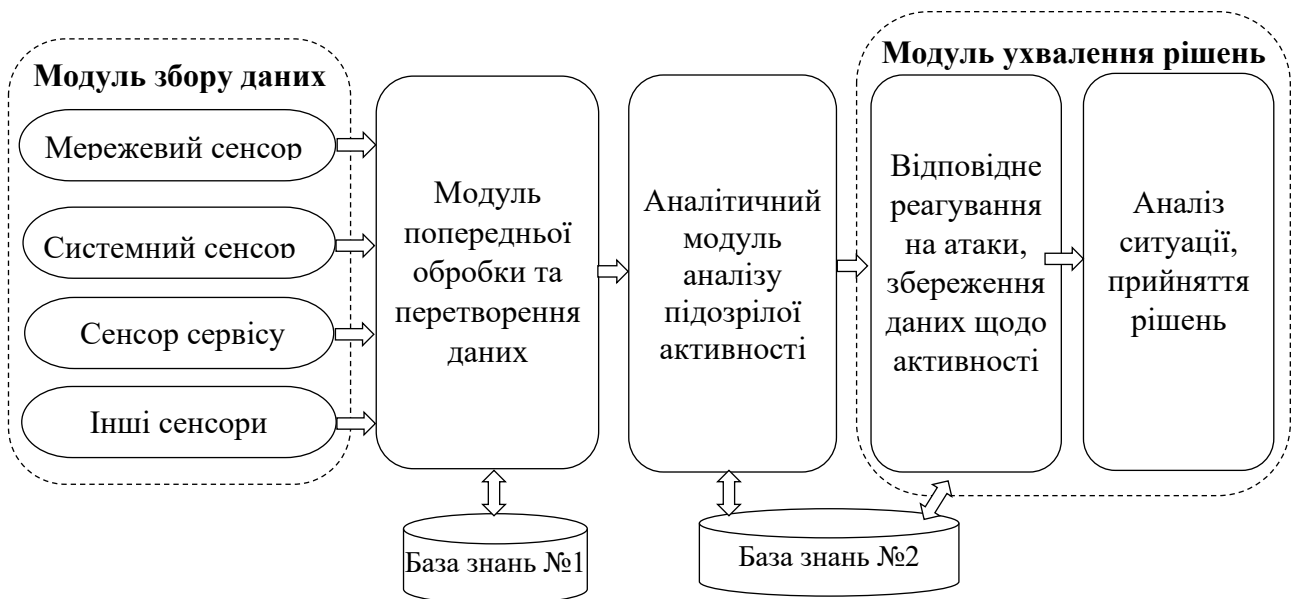


Рисунок 4.1 – Концептуальна структура системи виявлення вторгнення

Основою виявлення вторгнень є база знань № 1, побудова якої здійснюється на етапі початкового навчання системи.

Аналітичний модуль аналізу на основі сигнатур навчальної вибірки буде моделлю для класифікації реальної активності мережі. Це дозволяє системі розділити виявлені атаки на атаки мережного, транспортного та прикладного

рівнів та атаки каналного рівня.

Система має працювати у двох режимах: режим навчання, завантажується набір сигнатур а на основі цього набору будується модель, що класифікує загрозу; режим роботи, коли значення параметрів трафіку подаються як вхідні дані на підсистему сенсорів.

За допомогою побудованої на попередньому етапі класифікуючої моделі визначається, чи відповідають показання сенсорів нормальному стану або тій чи іншій атаці, і передає результат модулю ухвалення рішень, який, у разі атаки або підозрілої активності, видає оповіщення на консоль управління (пасивне виявлення) та організує збереження даних щодо активності у базу знань № 2 та формує команду реагування (активне блокування) для аналізу ситуації та прийняття рішень.

Розглянемо методи інтелектуального аналізу даних, що становлять основу алгоритму побудови пропонованої класифікуючої моделі системи [28].

Метод опорних векторів відноситься до методів лінійної класифікації. Кожен стан системи представляється як точки у багатовимірному просторі, координатами якого є характеристики системи.

Як переваги даного методу можна виділити високу точність, здатність до узагальнення та низьку обчислювальну складність прийняття рішення. Недоліком є відносно велика обчислювальна складність побудови моделі, що класифікує.

Метод k -найближчих сусідів – метод класифікації, основним принципом якого є надання об'єкту того класу, який є найпоширенішим серед сусідів даного об'єкта. Сусіди утворюються з множини об'єктів, класи яких вже відомі, і, виходячи із заданого значення k ($k \geq 1$), визначається, який із класів найбільш численний серед них. Якщо $k = 1$, об'єкт просто належить до класу єдиного найближчого сусіда. Метод є одним із найпростіших методів інтелектуального аналізу даних. Недоліком є те, що він чутливий до локальної структури даних.

Нейронні мережі дозволяють вирішувати практичні завдання, пов'язані з розпізнаванням та класифікацією образів. Нейронна мережа складається з взаємопов'язаних нейронів, утворюючих вхідний, проміжні (приховані) та вихідний шари. Навчання відбувається шляхом коригування значень ваги

нейронів для мінімізації помилки класифікації.

Перевагами нейронних мереж є їхня здатність набувати знань у процесі навчання, а також здатність до узагальнення, основний недолік – чутливість до шуму у вхідних даних.

Дерева прийняття рішень є деревоподібною структурою з «листя» і «гілок». На ребрах («гілках») дерева прийняття рішень записані атрибути, від яких залежить цільова функція, в «листях» записані значення цільової функції, а інших вузлах – атрибути, якими розрізняються об'єкти. Щоб класифікувати новий об'єкт, треба спуститися по дереву від кореня до листя та отримати відповідний клас, тобто шлях від кореня до листя виступає правилами класифікації з урахуванням значень атрибутів об'єкта.

Переваги дерев ухвалення рішень – простий принцип їх побудови, хороша інтерпретованість результатів, недолік – невисока точність класифікації.

У ході дослідження було проаналізовано сучасні методи виявлення атак, визначено їхні переваги та недоліки. Наведено ряд методологій для моделювання інформаційних ризиків та запропоновано концептуальну структуру системи виявлення вторгнення.

У розділі запропоновано функціональну структуру IDS, що складається з чотирьох ключових модулів: збору даних (сенсорів), аналітичного двигуна, бази знань та інтерфейсу керування. Це дозволяє системі не просто фіксувати атаки, а аналізувати їх у контексті поточної політики безпеки.

Особливістю системи є використання розроблених раніше нечітких моделей для класифікації аномалій. Це дає змогу виявляти нові типи атак (Zero-day), які не мають сигнатур у стандартних базах даних.

Система поєднує аналіз сигнатур (для відомих загроз) та статистичний аналіз аномалій (для нетипової поведінки мережі), що значно знижує кількість хибнопозитивних спрацювань (False Positives).

Завдяки механізму зворотного зв'язку, система може автоматично оновлювати «вектори пріоритетів» (з 2 розділу) та «бази правил» (з 3 розділу) при зміні ландшафту загроз або ІТ-інфраструктури підприємства.

Система не лише сигналізує про загрозу, а й пропонує адміністратору варіанти протидії, оцінюючи критичність ризику в режимі реального часу.

У ході дослідження було обґрунтовано необхідність впровадження інтелектуальних систем захисту в умовах зростаючої складності кібератак.

Розглянуто функціонування систем SIM, SEM та SIEM, які є ядром сучасних центрів моніторингу безпеки (SOC). Встановлено, що традиційні методи, засновані на жорстких правилах та сигнатурах, мають обмежену ефективність проти нових загроз (Zero-day) та генерують велику кількість хибнопозитивних спрацювань.

Визначено, що інтеграція методів машинного навчання (ML) (таких як Isolation Forest, нейронні мережі, дерева рішень) дозволяє системі адаптуватися до змін у мережевому середовищі. Використання ШІ забезпечує виявлення аномалій через аналіз поведінкових патернів (UEBA), детектування алгоритмів генерації доменів (DGA) та боротьбу з "алертною втомою" шляхом кластеризації подій.

Запропоновано чотири модульну структуру системи, що включає модулі збору даних (сенсори), попередньої обробки, аналітики та ухвалення рішень. Особливістю архітектури є поєднання сигнатурного аналізу для відомих загроз та статистичного/поведінкового аналізу для виявлення нетипової активності.

Запропонований підхід, що базується на моделюванні загроз за методологією STRIDE та використанні нечітких моделей, дозволяє не лише фіксувати інциденти, а й оцінювати ризики в режимі реального часу, пропонуючи адміністратору конкретні варіанти протидії.

Результатом розділу є теоретичне та практичне підґрунтя для створення адаптивної системи виявлення вторгнень, яка здатна мінімізувати вплив людського фактору та забезпечити цілісний проактивний захист інформаційних активів організації.

5. Агентне моделювання у галузі кібербезпеки

5.1 Програмні агенти

Програмні агенти (ПА) – нова парадигма програмування, яка дозволяє перейти на новий, більш інтелектуальний рівень взаємодії користувача з програмним і апаратним забезпеченням (рис. 5.1) [29]. Вона сприяє підвищенню ефективності праці та дозволяє користувачам доручити ІС виконання досить складних завдань. Необхідні інтелектуальні, інтерактивні та автономні програмні системи, які здатні до співпраці з користувачем для вирішення його задач. Такі складні задачі не можуть бути вирішені засобами одного наукового напрямку та потребують використання міждисциплінарного підходу, який містить технології з інформатики, штучного інтелекту, когнітивної науки та математики [29].

Теорія ПА використовує результати, отримані у процесі досліджень інших наукових дисциплін: теорії керування, розподіленого ШІ та когнітивної психології. ПА – програмні сутності, здатні діяти автономно та цілеспрямовано у динамічному середовищі для того, щоб виконати завдання користувачів [29].

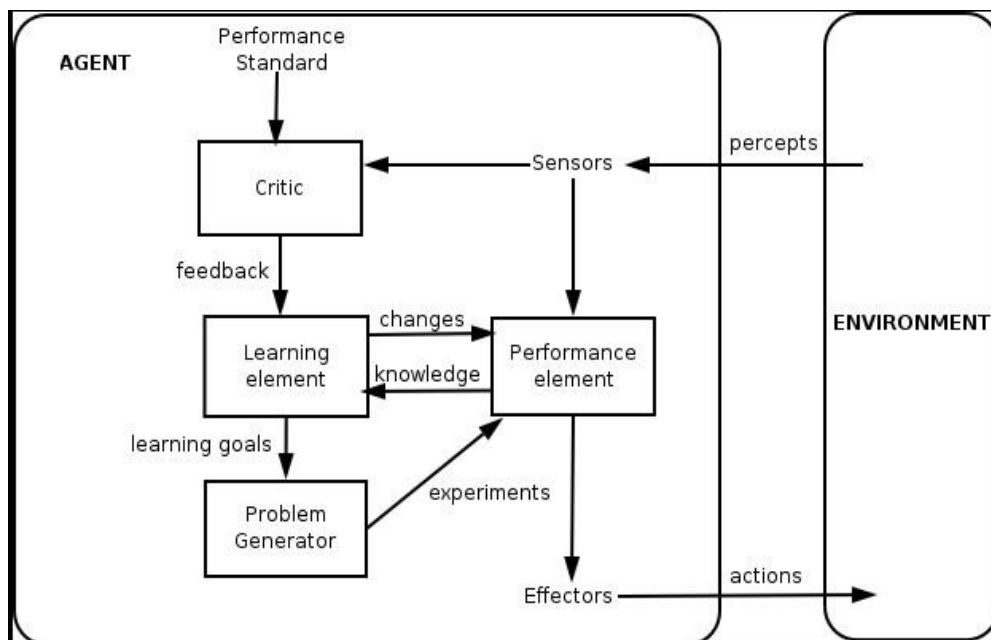


Рисунок 5.1 – Інтелектуальний агент [29]

Розподілене керування даними та обчислювальними процесами в умовах глобальних та корпоративних мереж спонукало до створення нової концепції

середовища функціонування ПЗ як середовища взаємодії мультиагентних систем (МАС), кооперації і конкуренції інтелектуальних агентів.

Основні властивості програмних агентів

Терміни «агент» і «інтелектуальний агент» (ІА) мають два значення, і через це іноді виникає плутанина.

У штучному інтелекті, під терміном інтелектуальний агент розуміють розумні сутності, що спостерігають за навколишнім середовищем і діють у ньому, при цьому їхня поведінка раціональна в тому розумінні, що вони здатні до розуміння і їхні дії завжди спрямовані на досягнення якоїнебудь мети. Такий агент може бути як роботом, так і вбудованою програмною системою. Про інтелектуальність агента можна говорити, якщо він взаємодіє з навколишнім середовищем приблизно так само, як діяла би людина.

У комп'ютерній науці інтелектуальний агент – це програма, що самостійно виконує завдання, указане користувачем комп'ютера, протягом тривалих проміжків часу. Інтелектуальні агенти використовуються для сприяння операторові або для збирання інформації. Одним із прикладів завдань, виконуваних агентами, може служити завдання постійного пошуку й збору необхідної інформації в Інтернеті. Комп'ютерні віруси, боти, пошукові роботи – усе це також можна віднести до інтелектуальних агентів. Хоча такі агенти мають строгий алгоритм, «інтелектуальність» у цьому контексті розуміється як здатність пристосовуватися й навчатися [29].

Для програмістів Агент – це, насамперед, комп'ютерна програма. З цього випливають такі властивості, як коректність, повнота, ефективність, надійність. При цьому агент виконує певні функції людини, надаючи користувачу потрібні йому послуги.

Існують різні визначення ПА залежно від їх призначення та акцентування певних технічних властивостей. П.Маєс визначає автономні ПА як комп'ютерні системи, що існують у складному динамічному середовищі, сприймають зміни у ньому та діють автономно, реалізуючи набір цілей або задач, для виконання яких вони створені. Відзначають, що інтелектуальні ПА виконують три функції:

- сприйняття динамічних умов середовища;
- дії у відповідь на такі умови;
- міркування для інтерпретації сприйняття, рішення проблеми та визначення реакції.

Таким чином, можна визначити ПА як автономну фізичну або віртуальну обчислювальну одиницю, що базується на:

- власних ресурсах – знаннях та вміннях;
- засобах сприйняття середовища (сенсорах) та впливу на це середовище (ефекторах);
- моделі середовища, заснованої на знаннях про нього.

ПА забезпечують наступні функціональні можливості:

- ✓ вирішення задач або досягнення певних цілей на основі наявних ресурсів та навичок;
- ✓ вибір рішення між альтернативами та виконання цього рішення у певному середовищі;
- ✓ спрямована взаємодія з іншими агентами та середовищем, у якому функціонує ПА.

За визначенням FIPA (Federation of Intelligent Physical Agents), агент – це об'єкт, що знаходиться в певному середовищі, від якого він отримує дані про події в цьому середовищі, інтерпретує їх і виконує команди, що впливають на середовище. Такий агент може містити як програмні, так і апаратні компоненти. FIPA – це міжнародна організація, створена в 1996 році з метою впровадження агентної парадигми для розробки практичних застосувань.

Можна визначати ПА через множину його атрибутів. ПА – термін, що дозволяє об'єднати множину більш специфічних і обмежених типів агентів, які мають деякі з таких атрибутів:

- реактивність (reactivity) – зміна своєї поведінки залежно від конкретної ситуації;
- автономність (autonomy) – самостійне виконання розпоряджень користувача без детальних інструкцій;

- співробітництво (collaborative behavior) – здатність працювати разом з іншими агентами для досягнення спільної мети;

- спілкування на рівні знань (“knowledge level” communication ability) – спроможність спілкуватися з людьми й іншими агентами мовою, близькою до природної;

- здатність до логічного виведення (inferential capability) – обробка абстрактного опису задачі з використанням апріорних знань про цілі і найбільш придатні методи їх досягнення, спроможність будувати моделі власної сутності, свого користувача, ситуацій та інших агентів;

- безперервність у часі (temporal continuity) – стійкість ідентифікації і положення протягом тривалого періоду часу;

- персоналізація (personality) – наявність персоналізованих значень атрибутів власної поведінки;

- адаптивність (adaptivity) – навчання й удосконалення на основі власного досвіду;

- мобільність (mobility) – здатність самостійно переходити з однієї платформи на іншу.

- правдивість – припущення про те, що агент не буде свідомо поширювати помилкову інформацію;

- лояльність – намагання робити те, що потрібно іншим агентам;

- раціональність – виконання тільки тих дій, які приводять до досягнення цілей.

Класифікація ПА за призначенням:

- інтерфейсні агенти;

- Інтранет-агенти;

-Інтернет-агенти;

- гетерогенні агенти.

Інтерфейсні агенти можна розглядати як персональні асистенти, які допомагають користувачу працювати з різними програмними засобами. Такі агенти спостерігають за діями користувача та намагаються запропонувати йому

такі дії, які мають спростити його роботу. Вони здатні адаптуватися до індивідуальних особливостей та потреб конкретного користувача. Інтерфейсні ПА здатні до самонавчання, що приводить до ефективнішого виконання ними своїх функцій. Для самонавчання агенти використовують:

- спостереження за поведінкою користувача;
- отримання позитивних або негативних оцінок від користувача (зворотний зв'язок);
- безпосереднє отримання інструкцій від користувача;
- консультації з іншими агентами.

Представниками цієї групи агентів є такі системи, як A-Match та Verbal Software Robots.

Інтернет-агенти є найбільш численними представниками інтелектуальних ПА. Вони виникли як засіб обробки та транспортування інформаційних ресурсів Інтернету. На відміну від інтерфейсних агентів вони здатні не тільки створювати персоніфікований профіль користувача, але й відповідно класифікувати інформаційні ресурси.

Інтернет-агентів можна поділити на дві основні групи: статичні та мобільні. Статичні ПА звичайно вбудовані у браузер. Приміром, такі агенти можуть сортувати електронну пошту, повідомляти користувача про події та повідомлення, які, за наявними в агента відомостями, можуть його зацікавити. Мобільні Інтернет-агенти менш поширені. Прикладом такого агента є Jasper. Цей агент здатний не тільки знаходити інформацію, цікаву для його користувача, але й повідомляти про неї інших агентів.

Робота Інтранет-агентів схожа на роботу Інтернет-агентів, проте має власну специфіку. Характерні задачі, завдяки яким Інтранет-агентів виділяють у окрему групу, такі:

- автоматизація бізнес-процесів підприємства;
- виконання послуг для користувачів, пов'язаних з використанням інформації з бази даних підприємства.

Гетерогенні агенти інтегрують функції двох або більше агентів, які належать до різних типів.

На відміну від класичних систем ШІ агенти не тільки пропонують рішення проблеми, але й реально діють. Сукупність причин, через які ПА виконує певні дії, називають мотивацією. Для досягнення своїх цілей ПА конструює план дій, які мають призвести до виконання поставленого перед ним завдання.

Зараз визначилися два різних, проте пов'язаних один з одним підходи до визначення агента. Відповідно до першого підходу агент визначається головним чином своїми діями і не може бути цілком охарактеризований набором своїх атрибутів. У другому підході агент визначається атрибутами, якими він володіє.

Агент – це об'єкт, що сприймає середовище за допомогою сенсорів і діє в ньому за допомогою ефекторів.

ПА функціонують у багатовимірному просторі. Вони складаються з кількох шарів: комунікації; координації, організації; визначення; інтерфейсу API. Шар комунікацій розглядає низькорівневі деталі взаємодії між ПА. На координаційному шарі подаються соціальні властивості ПА, технології координації та переговорів. На організаційному шарі ПА визначається через відношення з іншими ПА, через ролі, які він виконує у взаємодії з цими агентами. На шарі визначення ПА визначається як автономна раціональна сутність, тобто в термінах механізмів міркування та навчання, цілей, ресурсів, здібностей, переконань тощо. Шар програмного інтерфейсу API пов'язує ПА з його фізичною реалізацією.

В обчислювальному середовищі кожний агент має свій життєвий цикл і власне ім'я. За допомогою використання доменів призначення агентів відбувається їх адміністрування.

Позначення імені агента забезпечує спосіб ідентифікації агента серед доменів інших груп агентів. Розробка програмних агентів вимагає застосування стандартизованих профілів агентів і методології розробки агентів для кожного з конкретних застосувань.

Профіль платформи агента – це кортеж <пароль, значення>, що описує послуги і властивості платформи. Онтологія керування агента визначає словник і семантику опису можливості платформи агента. Атрибути ПА включають назву платформи агента, адресу, призначення сервісів агента, інформацію про підтримку в мережі, ступінь пріоритетів сервісів і їх застосування.

З погляду користувача, основна перевага використання агентів полягає в спрощенні взаємодії з програмою – користувачу досить поставити загальну задачу перед агентом, не вдаючись у подробиці того, як саме агент має її вирішувати. Якщо агент має можливості для рішення цієї задачі, він вирішує її сам, а інакше запитує необхідні йому послуги в інших агентів.

Агент – це програмний об'єкт, який:

- забезпечує виконання однієї або кількох корисних послуг;
- надає опис семантики цих послуг іншим ПА;
- здатний функціонувати автономно без безпосередніх вказівок користувача;
- може інтерактивно взаємодіяти з іншими ПА та користувачами.

Функціональний зв'язок між агентом, середовищем, у якому цей агент функціонує, та його програмним кодом можна подати в такий спосіб:

$$A = f(X), \quad (5.1)$$

де A – агент; f – програма (функція); X – середовище.

Найпростіше визначення агента ґрунтується на моделі чорного ящика, що знаходиться у певному середовищі. Агент описується як функція f , що обробляє інформацію від сенсорів і вхідні повідомлення (рис. 5.2). Результат роботи агента – дії і вихідні повідомлення. Цей узагальнений підхід відповідає як біологічним, так і інтенціональним моделям агентів. Розходження між цими моделями полягають у способі визначення f .

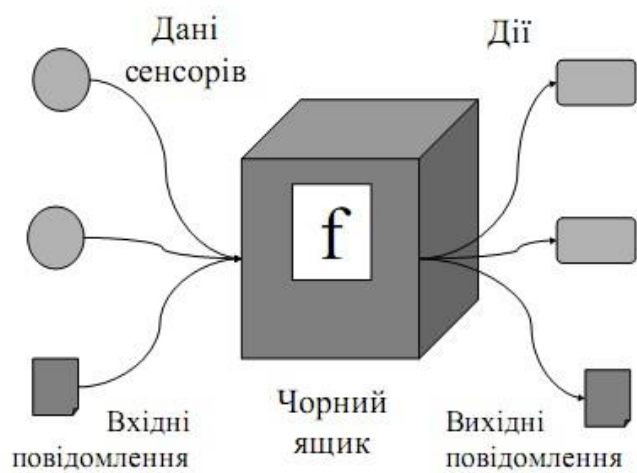


Рисунок 5.2 – Визначення агента через модель чорного ящика [29]

Однією з основних характеристик агента є комунікабельність – здатність до гнучкого спілкування як з агентами, так і з іншими програмними компонентами. Унаслідок цього агенти відіграють важливу роль у досягненні інтеоперабельності ПЗ, створеного незалежними розробниками в різний час. Агентифікація дозволяє поширити цю властивість на довільне ПЗ.

Агентифікація – перетворення довільного програмного забезпечення в ПА, приміром, у формі надбудови агентної оболонки над фрагментами програмного коду, яка забезпечує інтеоперабельність цього коду. Підкласом ПА є інтелектуальні агенти (ІА) (рис. 5.3).

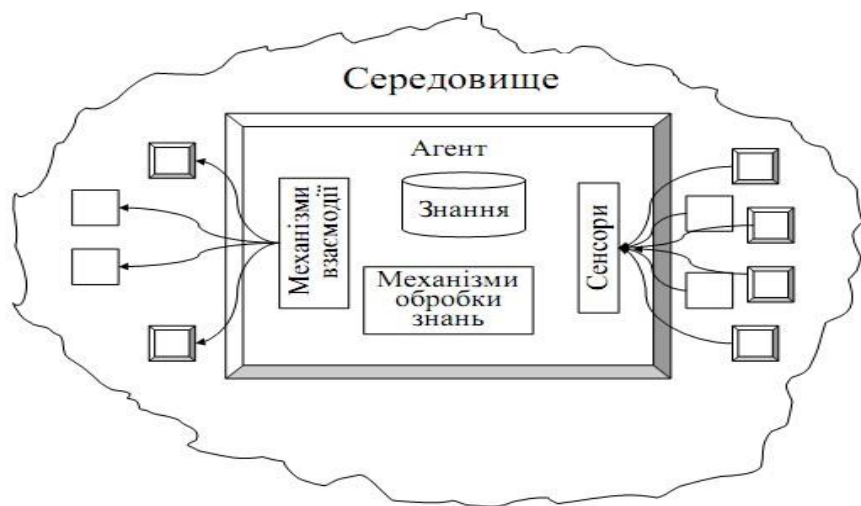


Рисунок 5.3 – Загальна концептуальна схема інтелектуального ПА [29]

Прийнято розрізняти вузьке ("сильне") та широке ("слабке") визначення терміну "інтелектуальний агент".

ІА у широкому розумінні – це ІС, що має такі ключові ознаки:

- автономність (autonomy) – функціонування значною мірою незалежно від втручання людини і контроль власних дій та внутрішнього стану;

соціальність (social ability) – інтелектуальна та конструктивна взаємодія з іншими агентами та людьми шляхом обміну з ними повідомленнями деякою загальнозрозумілою мовою комунікацій;

- реактивність (reactivity) – сприйняття зміни середовища і вчасне реагування на них;

- проактивність (pro-activity) – здатність агента генерувати цілі і діяти раціонально для їх досягнення, а не тільки реагувати на зовнішні події.

Зазвичай, поведінка людини прогнозується і аналізується через такі атрибути відношень, як переконання, бажання, надії, побоювання тощо, які називаються інтенціональними поняттями.

Більш строго ("сильне") розуміння терміну "інтелектуальний агент" вимагає наявності в агента ментальних властивостей (інтенціональних відношень), до яких належать:

- знання (knowledge) – стала частина інформації агента про себе, середовище й інших агентів, що не змінюється в процесі його функціонування;

- переконання (beliefs) – знання агента, які можуть змінюватися в процесі його функціонування і ставати хибними, про поточний стан світу і про зміни в ньому, до яких має привести виконання дій агента;

- бажання (desires) – ставлення агента до майбутніх станів світу та переваги, які він надає одним з них порівняно з іншими (агент може мати несумісні та недосяжні бажання і тому не очікує, що усі вони мають бути досягнуті);

- наміри (intentions) – підмножина цілей, які може досягти обмежений у ресурсах агент, і засіб їх досягнення;

- цілі (goals) – несуперечлива підмножина бажань, досягнення яких агент прийняв як поточну стратегію поведінки; зобов'язання (commitments) стосовно інших агентів – завдання, що агент виконує за дорученням інших агентів у рамках кооперації та співробітництва.

Перші два поняття – переконання та знання – називають “точкою зору” (attitudes) агента, інші характеризують в англomовній літературі загальним терміном “pro-attitude”, сутність яких полягає у тому, що вони спрямовують дії та поведінку агента.

Інтенціональні відношення поділяються на інформаційні (переконання та знання) і перед-відношення (бажання та емоції, намір, зобов'язання, цілі тощо). Перші стосуються інформації, що має агент про світ, у якому він існує, тоді як передвідношення – це відомості, які певним чином впливають на дії агента.

Перед- і інформаційні відношення тісно пов'язані, оскільки агенти можуть, наприклад, формувати наміри на основі наявної в них інформації про світ.

Деякі автори вважають, що інтелектуальному агенту мають бути притаманні такі властивості:

- мобільність (mobility) – здатність переміщуватися телекомунікаційними мережами (локальними або глобальними) для досягнення своїх цілей;
- доброзичливість (benevolence) – готовність агентів допомагати іншим агентам та виконувати доручення користувача;
- правдивість (veracity) – властивість не повідомляти іншим агентам та користувачу інформацію, про помилковість якої йому відомо;
- раціональність (rationality) – здатність виконувати саме ті дії, що приводять до досягнення його цілей у рамках наявних у агента знань і переконань.

Інтелектуальність ПА визначається його спроможністю міркувати і навчатися, наявністю моделі користувача, його потреб і механізму пошуку засобів їх задоволення.

Деякі автори вирізняють такі ознаки інтелектуальності агентів :

- автономне виконання своїх функцій;
- взаємодія з іншими агентами і користувачами;
- здатність стежити за оточенням;
- використання абстракції;

- застосування знання ПрО;
- адаптивність поведінки; навчання на власному досвіді; толерантність до помилок у вхідних сигналах;
- здатність працювати в реальному часі;
- спілкування природною мовою.

Використання агентів сприяє розвитку принципово нових інформаційних технологій.

5.2. Мультиагентні системи

Проблеми, для вирішення яких застосовують агентний підхід, можуть бути досить складними. Дослідження в галузі ШІ показали, що взаємодія і декомпозиція загальної задачі ефективно позначається на результатах виконання цієї задачі. Тому краще створювати окремі автономні блоки такої системи, а потім організувати їх спільне функціонування.

Термін мультиагентні системи використовується для позначення ІС, які складаються з множини автономних модулів ПА та мають такі властивості:

- кожен ПА є автономним, мобільним та інтероперабельним;
- ПА, що входять до складу МАС, здатні обмінюватися інформацією для досягнення спільних цілей;
- керування ПА може бути децентралізованим;
- джерела даних і доступ до них децентралізовані;
- робота агентів є асинхронною.

Теорія МАС походить від теорії відкритих систем, розподіленого ШІ і загальної теорії складних систем.

Розподілений штучний інтелект (РШІ) пов'язаний з аналізом систем, що складаються з окремих незалежних об'єктів, які взаємодіють один з одним, та механізмів їх координації. МАС теж є предметом розгляду РШІ. У цьому випадку незалежними об'єктами є ПА. Для вивчення поведінки МАС використовують методи таких наукових дисциплін:

розподілений ШІ (теорія розподілених систем, теорії прийняття рішень), що займається найбільш загальними аспектами колективної поведінки агентів;

теорія ігор, яка використовується для дослідження ситуацій, аналогічних до кооперативних ігор, стратегій ведення переговорів; теорія колективної поведінки автоматів, яка досліджує колективну поведінку великих груп автоматів з примітивними функціями, спроможних навчатися за допомогою системи штрафів і заохочень; біологічні, економічні та соціальні моделі.

У розробці МАС використовуються також результати з інших галузей теоретичних досліджень (рис. 5.4).

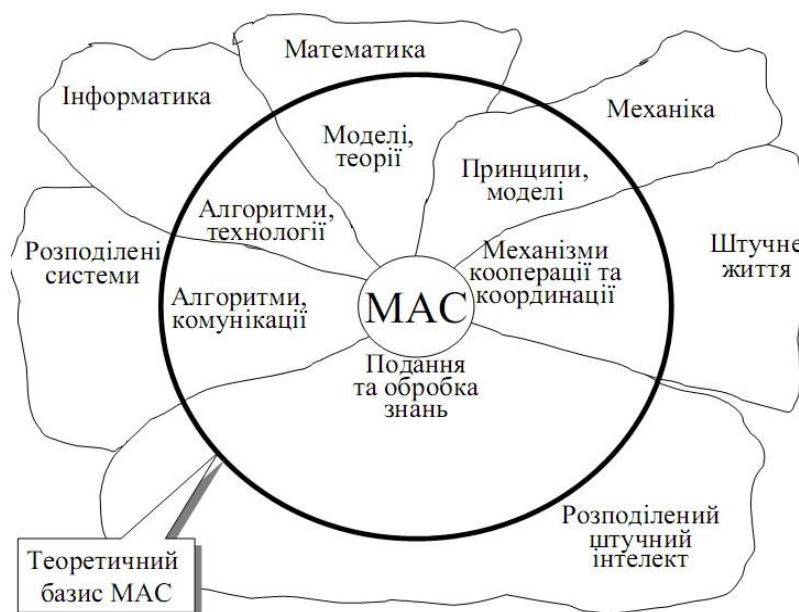


Рисунок 5.4 – Основні ПрО, що використовуються у розробці МАС [29]

МАС – співтовариство ПА, які пов’язані один з одним цілями та ресурсами. ПА, що входять до складу МАС, здатні взаємодіяти для того, щоб обмінюватися послугами, які потрібні їм для досягнення цілей, поставлених перед ними користувачами. Агент, не здатний самостійно вирішити задачу, поставлену перед ним користувачем, звертається до інших агентів, які можуть надати йому відповідні послуги.

МАС поширені в багатьох ПрО: керуванні виробничими процесами і промисловими підприємствами; плануванні рухом транспорту (повітряного, залізничного, автомобільного); аналізі та пошуку інформації; навчанні; бізнесі тощо.

МАС складається з множини ПА, які: взаємодіють шляхом комунікацій; здатні діяти у певному середовищі; мають певні сфери впливу, які можуть перетинатися або співпадати.

Можна розглядати МАС як слабкопов'язану мережу обчислювальних пристроїв, взаємодіючих для вирішення проблеми, що знаходиться поза індивідуальними можливостями або знаннями кожного з цих пристроїв.

Координація – одна з центральних проблем МАС. Наступні фактори визначають потребу в координації МАС:

- запобігання хаосу в децентралізованій МАС;
- наявність глобальних обмежень, яким має задовольняти МАС для успішного виконання завдання;
- розподіленість знань, ресурсів та інформації, які використовують ПА, що входять до складу МАС;
- залежність між цілями та діями ПА; забезпечення ефективності функціонування МАС.

Функціонування МАС пов'язане з кооперацією та конкуренцією агентів в процесі колективного вирішення задач. Агент, що не може вирішити власну задачу самостійно, має взаємодіяти з іншими ПА.

При цьому агенти можуть будувати плани спільних дій, ґрунтуючись не тільки на власних можливостях, але і аналізувати плани і наміри інших ПА (використовуючи різні комбінації інтенціональних відношень). Кооперативні дії ПА – важлива перевага МАС. При такій організації роботи група ПА проявляє новий, ефективніший тип поведінки.

Мотивація використання МАС базується на їх властивостях:

- здатність вирішувати проблеми, складні для одного централізованого ПА через обмеженість ресурсів;
- можливість взаємодії та інтеперабельності з різноманітними застосунками (ЕС, СППР тощо);
- вирішення дійсно розподілених проблем (приміром, керування рухом транспорту);

- знаходження рішень на основі розподілених ІР;
- вирішення проблем з розподіленою експертизою (приміром, медичний догляд);
- підвищення швидкості та надійності;
- здатність збільшувати кількість обчислювальних пристроїв (процесорів), що використовуються для виконання задачі;
- можливість опрацювання нечітких та неповних даних і знань.

Знання й уміння МАС здобуваються від великої кількості відносно простих ПА, що поєднуються разом деякою архітектурою. В процесі колективної роботи агенти мають будувати плани дій, ґрунтуючись не тільки на своїх можливостях, але й враховуючи плани і наміри інших агентів (на основі своїх знань та переконань щодо цих планів та намірів).

Причини взаємодії ПА: наявність сумісних цілей; нестача індивідуальних ресурсів ПА для досягнення цілей; нездатність ПА самотійно вирішити задачу; наявність взаємних зобов'язань.

Перехід до відкритих МАС надає можливість переходу на нову якість функціонування системи в силу того, що система (група агентів) більше, ніж сума властивостей її членів (агентів).

Системи, які складаються з досить простих програм, кожна з яких переслідує тільки свої примітивні цілі, у цілому здатні вирішувати дуже складні задачі. Однак моделювання колективної поведінки приводить до необхідності рішення багатьох проблем: формування спільних планів дій; врахування інтересів інших агентів; синхронізації спільних дій; вирішення конфліктів між цілями різних агентів; конкуренції за спільні ресурси; організації переговорів про спільні дії; розпізнавання необхідності кооперації; вибір партнерів; декомпозиції задач; поділу обов'язків тощо.

Багато робіт в сфері ШІ присвячені специфікації протоколів взаємодії між ПА. Дослідження переговорів у МАС засновані на ідеї контрактних мереж. Відповідно до цієї парадигми ПА, який бажає отримати певну послугу, запрошує на переговори інших ПА і вибирає послуги, які найбільше

відповідають його потребі. У складніших моделях ПА тільки запитує пропозиції, а інші спеціалізовані агенти оцінюють варіанти відповідей та обирають найкращу.

Інший вид взаємодії між ПА – формування коаліції для спільного виконання своїх намірів. Механізми формування коаліції, де агенти динамічно поєднуються з іншими агентами, інтенсивно вивчаються багатьма дослідниками.

Існує багато моделей кооперації агентів. Приміром, модель CPS (Cooperative Problem Solving) призначена для встановлення взаємодії між ПА, побудованих на основі BDI-архітектури. Ментальні поняття формалізуються за допомогою операторів темпоральної логіки для визначення таких понять, як потенціал кооперації, групові дії, досяжність мети агента тощо.

Найпростіший метод координації MAC – організаційне структурування – полягає у чітко визначених і довгострокових відношеннях між ПА. При цьому використовують ієрархічні структури master-slave або client-server. Організаційне структурування передбачає, що принаймні один ПА має глобальне уявлення про все співтовариства, однак для багатьох ПрО це не реально.

У процесі формування кооперативного рішення в CPS-моделі (Cyber-Physical Systems) виділяють чотири етапи:

1. Визначення потреби у кооперації (приміром, агент має певну мету, але переконаний у тому, що не може досягти самостійно).

2. Створення групи агентів. При успішному завершенні цього етапу створюється група ПА зі спільними зобов'язаннями, пов'язаними з виконанням колективних дій.

3. Формування спільного плану дій. ПА ведуть переговори для формування плану дій, який за їхніми переконаннями має привести до реалізації цілей кожного з них.

4. Спільні дії. Агенти виконують дії відповідно до плану, який вони прийняли на попередньому етапі, виконуючи прийняті на себе зобов'язанням.

Кожний з підходів до координації МАС має певні переваги і недоліки та може застосовуватися тільки для деяких специфічних областей. Універсального методу координації, що ідеально підходить для будь-якої реальної задачі, не існує.

5.2.1. Децентралізоване та централізоване управління.

Централізоване управління (рис. 5.5, а) передбачає наявність одного блоку управління, який збирає повну інформацію про всі об'єкти управління і генерує (розраховує, приймає рішення про) сигнали управління для всіх цих об'єктів. Децентралізоване управління (рис. 5.5, б) передбачає, що кожний з об'єктів управління оснащений власним блоком управління, який генерує для нього сигнали управління, виходячи з інформації лише про нього.

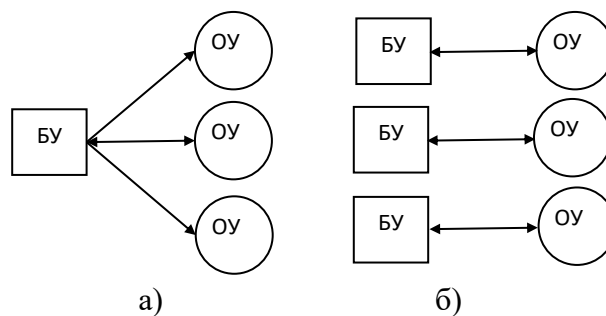


Рисунок 5.5 – Структура: а) централізованого та б) децентралізованого управління (БУ – блок управління, ОУ – об'єкт управління)[30]

Існують дві вимоги до систем управління.

1. Адекватність (цілеспрямованість, доцільність);
2. Оперативність управління (швидкість прийняття рішення).

Принцип дуальності управління – ці вимоги протирічать одна іншій, в наслідок чого має бути знайдено оптимальне співвідношення цих двох вимог. Це протиріччя набуває особливого змісту для розподілених систем.

Обидва алгоритми гарантують вирішення задачі. Алгоритм для централізованого управління є складним та більш швидко приводить до результату. Алгоритм для децентралізованого управління дуже простий та працює значно повільніше. Порівнюючи складність та швидкодію цих алгоритмів, можна зробити висновок про доцільність комбінації двох підходів

до управління. В цьому аспекті самоорганізація розуміється як пошук оптимального співвідношення централізованого та децентралізованого управління в залежності від поставленою перед системою задачі та умов, в яких ця задача вирішується.

5.2.2 Визначення колективної моделі реальності (КМР)

Модель реальності (world model): набір взаємопов'язаних даних про оточення агента та його дії в цьому оточенні. Індивідуальні моделі реальності агентів є вхідною інформацією для побудови КМР. Носіями КМР є окремі агенти. Окрема КМР передбачає наявність однакових для двох або більше агентів структур збереження даних та однакового для двох або більше агентів наповнення цих структур.

Самоорганізацію колективу можна розглядати, як розвиток, типізацію та розповсюдження колективних моделей реальності. Колективні моделі реальності визначають напрямок та швидкість самоорганізації та розвитку діяльності колективу. Дослідження агентом середовища та обмін інформацією між агентами породжує динамізм, видозмінність їхніх індивідуальних моделей реальності. В процесі цих видозмін за певним законом формуються колективні моделі.

Приклад організації пам'яті агента на основі його узагальненої функціональної структури

1. Модель зовнішнього середовища – будується на основі індивідуальних сенсорних даних та сенсорних даних, отриманих від інших агентів ("сусідів"). Містить дані про середовище. Приклад запису: відлік часу, координата, назва параметру, величина параметру.

2. Модель колективу – будується на основі даних про виявлених сусідніх агентів (вихідна інформація блоку детектування). Містить данні про колектив. Приклад запису: відлік часу, список ID виявлених сусідів.

3. Модель впливів на середовище – будується на основі звітності блоку реалізації прийнятих рішень. Містить дані про реалізовані/не реалізовані дії. Приклад запису: відлік часу, номер дії, відмітка про успішність реалізації.

4. Модель розвитку – будується на основі службових даних блоку прийняття рішень. Містить дані про стратегії поведінки, що їх використовує агент (пам'ять стратегій поведінки). Приклад запису: відлік часу, номер дії, величина виграшу.

5. Модель переконань – будується на основі службових даних блоку оцінки. Містить дані про функції оцінки (системи цінностей), що їх застосовує агент. Приклад запису: відлік часу, номер функції оцінки.

Коллективне знання – це функціонально повний набір колективних моделей та правил їх використання, що застосовуються усіма агентами колективу. Коллективне знання – не сума індивідуальних, а якісно нова інтегральна властивість. Індивідуальна пам'ять агента не здатна вмістити колективне знання. Вона містить або якусь його частину, або відбиває його в деякому узагальненому вигляді (звичайно і те, і інше).

Коллективне знання є цілісним єдиним утворенням, воно існує як самостійно діюча система, компоненти якої можуть виключатись, замінюватись та додаватись, рис 5.6, де: 1 – виділення з індивідуальних моделей загальних фрагментів; 2 – уточнення та видалення не збігів (критика та випробування); 3 – інтеграція індивідуальних спеціалізованих моделей; 4 – формування колективних моделей; 5 – оцінка та перевірка ефективності застосування колективних моделей; 6 – генерування індивідуальних рішень щодо модифікації колективної моделі; 7 – стимулювання індивідуальної поведінки щодо розвитку колективної моделі; 8 – контролювання цілісності та коректності колективних моделей.

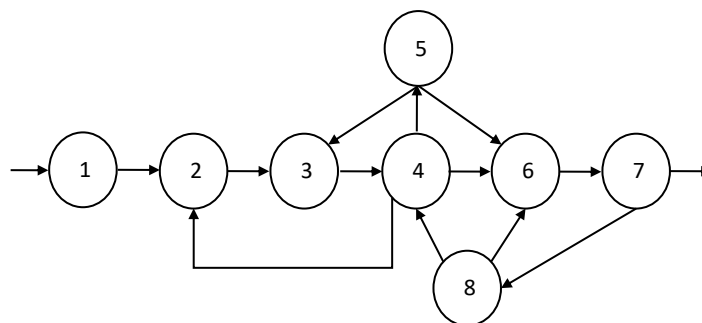


Рисунок 5.6 – Схема утворення колективного знання [30]

5.2.3. Оптимальне управління коаліцією за допомогою аукціону

Для вирішення завдання оптимального управління коаліцією роботів в роботі запропоновано використання алгоритмів аукціонів, де кожен агент взаємодіє з усіма агентами і сам визначає ставки та переможця [31-32].

Колектив з транспортних роботів, вільних в даний момент часу, становить тимчасову коаліцію. Хай за робочу зміну t_{cm} агенти-роботи беруть участь в роботі K коаліцій.

Тоді цільова функція оптимального керування транспортними роботами, які працюють в коаліції протягом зміни, має вигляд

$$\Phi_{tr}(U_{tr}) = \sum_{k=1}^K \Phi_k \Rightarrow \min, \quad (5.2)$$

де Φ_k – критерій якості роботи k -й ситуаційної коаліції агентів роботів, що представляє собою сумарний час виконання заявок N_k

$$\Phi_k = \sum_{i=1}^{N_k} t_{ik} \Rightarrow \min, \quad k = 1, \dots, K, \quad (5.3)$$

де N_k – кількість заказів, що виконані k -ю коаліцією; t_{ik} – час виконання i -го заказу в k -й коаліції

$$t_{ik} = \min_j(t_{ikj}), \quad j = 1, \dots, J_k, \quad i = 1, \dots, N_k, \quad (5.4)$$

де j – номер агента у k -й коаліції; t_{ikj} – час виконання j -м агентом з k -й коаліції i -го заказу; J_k – число агентів у k -й коаліції, а операція визначення мінімуму враховує тільки множину агентів $\{A_{tr1}, A_{tr2}, \dots, A_{trJ_k}\}$ не зайнятих у виконанні інших заявок.

Виконання критерія ефективності управління (5.2) забезпечується послідовним виконанням ситуаційних критеріїв ефективності (3) за рахунок умови (4) на всіх етапах управління.

Рішення задачі оптимального управління (5.2) – (5.4), що реалізовується в режимі реального часу шляхом поетапних аукціонів всередині ситуаційних коаліцій агентів-роботів, вимагає формування таких цільових функцій агентів,

які забезпечують найбільш ефективне рішення задачі (5.3) для кожної ситуаційної коаліції.

З іншого боку, цільові функції агентів повинні стимулювати участь вільних в поточний момент часу агентів у всіх формуються ситуаційних коаліціях і на всіх етапах аукціонів з розподілу поточних заявок.

Як компонент цільових функцій агентів транспортних робіт запропоновано використовувати дві складові.

Перша складова стимулює участь агента транспортного робота на всіх етапах аукціону з розподілу заявок всередині коаліції у всіх доступних йому коаліціях. Ця компонента є нормоване середнє значення відносного часу на транспортну операцію j -го агента при роботі в k -й коаліції

$$U_j^1 = \frac{1}{N_{kj}} \left[\sum_{i=1}^{N_{kj}} (t_{ijk_{\text{раб}}} / t_{ijk_{\text{max}}}) - 1 \right] \Rightarrow \min, \quad j = 1, \dots, J_k, \quad (5.5)$$

де N_{kj} – кількість заявок, які може виконати j -й робот в k -й коаліції; $t_{ijk_{\text{раб}}}$ – час j -го робота на i -ю заявку, яке визначається як

$$t_{ijk_{\text{раб}}} = t_{ijk_0} + t_{ijk_{\text{дост}}},$$

де t_{ijk_0} – час переміщення транспортного робота від поточного місця до місця складування по i -й заявці; $t_{ijk_{\text{дост}}}$ – час переміщення j -го транспортного робота від місця складування до місця i -й доставки; $t_{ijk_{\text{max}}}$ – максимальний час з набору.

Друга складова цільової функції агента-робота максимально стимулює його участь у всіх ситуаційних коаліціях K_j , де K_j – кількість коаліцій в яких брав участь j -й агент-робот і являє собою нормований час простою j -го транспортного робота за час робочої зміни

$$U_j^2 = 1 - \frac{1}{t_{\text{см}}} \sum_{k=1}^{K_j} t_{ik_{\text{раб}}} \Rightarrow \min, \quad j = 1, \dots, N_{\text{тр}}, \quad (5.6)$$

де $t_{\text{см}}$ – час робочої зміни; $N_{\text{тр}}$ – кількість транспортних робіт системи.

Критерії (5.5) та (5.6) повинні виконуватися при обмеженні на сумарний час

роботи j -агента транспортного робота

$$\sum_{k=1}^{K_j} \sum_{i=1}^{N_{jk}} t_{ik} \text{ раб} + t_j \text{ проф} \leq t_{\text{см}}, \quad (5.7)$$

де $t_j \text{ проф}$ – час, витрачений на профілактику j -го робота.

Завдання оптимального керування транспортними роботами (5.2) – (5.4) реалізується через рішення багатокритеріальної задачі оптимізації роботи транспортних роботів (5.5) – (5.7) шляхом формування ситуаційних коаліцій роботів і проведення цими коаліціями аукціонів [12] в режимі реального часу.

Для вирішення поставленого завдання оптимального управління (5.2) – (5.4) за рахунок коаліційної оптимальної координації роботи агентів-роботів, що реалізується шляхом вирішення задачі оптимізації (5.5) – (5.7) створена модифікована модель аукціону, яка складається з кінцевого числа послідовних раундів.

Прагнення до виконання критеріїв завдання (5.5) і (5.6) вимагає від кожного агента-робота участі у всіх ситуаційних коаліціях і у всіх раундах аукціону, за умови, що робот вільний.

Для демонстрації механізму аукціону представимо завдання у вигляді спрямованого графа для однієї коаліції з трьох транспортних роботів (рис. 5.7).

Розглядається окремий випадок, де кількість заявок $N = 3$ і кількість агентів $J_k = 3$. Час виконання заявок кожним j -м транспортним роботом задано згідно рис. 5.7,

$$\begin{aligned} t_{11} &= 20, t_{12} = 12, t_{13} = 6, \\ t_{21} &= 21, t_{22} = 10, t_{23} = 11, \\ t_{31} &= 10, t_{32} = 11, t_{33} = 8. \end{aligned}$$

Взаємодія j -го агента транспортного робота із середовищем протягом кожного поточного такту характеризується функцією оцінки (ставки) P_j , на основі якого він і приймає рішення про свої подальші дії, розраховується як різниця часів виконання агентом двох заявок з мінімальним часом $t_{\min 1}$, $t_{\min 2}$,

$t_{\min_1} \leq t_{\min_2}$ $P_j = (t_{j_{\min_2}} - t_{j_{\min_1}}) + \varepsilon$, де ε – премія за доставку (в задачі постійна величина).

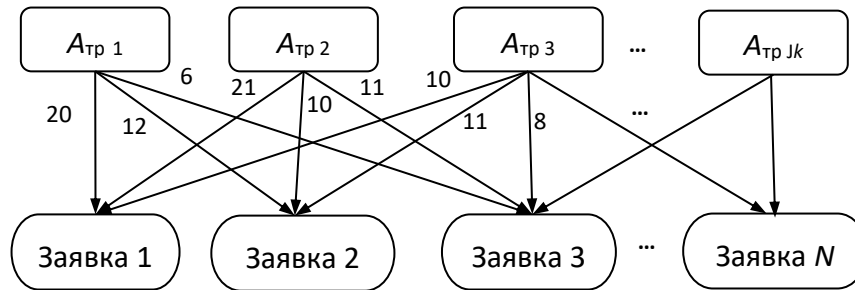


Рисунок 5.7 – Спрямований граф

Надалі ставка визначає величину накопичуваної підсумкової функції оцінки діяльності кожного агента, який бере участь в раунді, і на основі ставок агент приймає рішення про свої подальші дії. Чим ставка вище, тим менше часу агент-робот витратить на виконання заявки, і найбільша ставка буде відповідати найвигіднішою заявці раунду.

Ставка заснована на обчисленні значення відомого в теорії ігор під назвою "жалю" і в даному випадку визначає розмір втрат, яких зазнає вся коаліція агентів, якщо агент не стане переможцем. Такий підхід дозволяє оцінити розмір можливого програшу для кожного раунду і отримати оптимальне (квазіоптимальне) рішення.

У першому раунді аукціону (рис. 5.8) отримуємо:

для агента $A_{\text{тр}1}$ заявка 3 та часом t_{13} дає ставку $P_1 = 12 - 6 + 1 = 7$;

для $A_{\text{тр}2}$ заявка 2 та час t_{22} дає ставку $P_2 = 11 - 10 + 1 = 2$;

для $A_{\text{тр}3}$ заявка 3 та час t_{33} дає ставку $P_3 = 10 - 8 + 1 = 3$.

На виконання заявки 3 претендують відразу два агента 1 і 2, але ставка 1-го більше, тому заявку виграє перший агент.

На заявку 2 претендує тільки агент $A_{\text{тр}2}$, тому він її отримує.

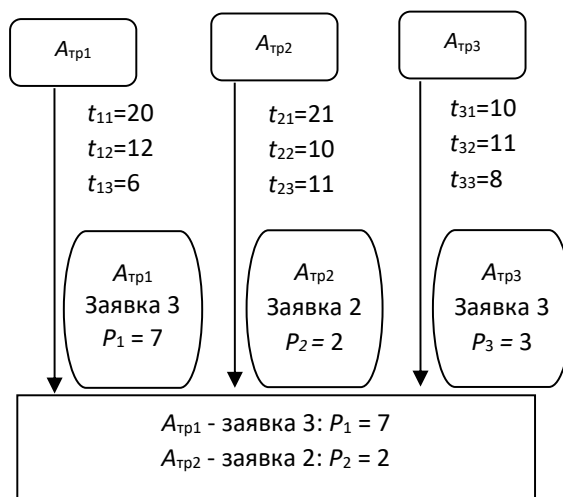


Рисунок 5.8 – Перший раунд аукціону

Так як не всі заявки розподілені, то починається наступний, другий раунд (рис. 5.9). Для двох розподілених заявок в наступному раунді підвищується значення часу їх заявок на величину їх ставок:

для $A_{тp1}$: заявка 2 $t_{12} = 12 + 2 = 14$;

для $A_{тp2}$: заявка 3 $t_{23} = 11 + 7 = 18$;

для $A_{тp3}$: заявка 2 $t_{32} = 11 + 2 = 13$ та
заявка 3 $t_{33} = 8 + 7 = 15$.

Для нерозподіленої 1-ї заявки час не змінюється.

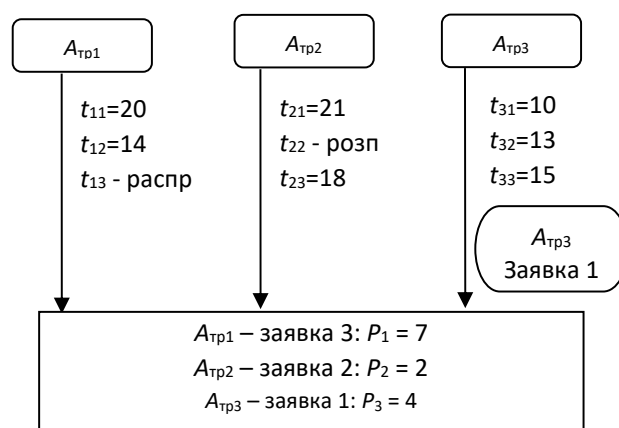


Рисунок 5.9 – Другий раунд аукціону

Після розрахунків другого раунду маємо:

для $A_{тp1}$ заявка 3, ставка $P_1 = 7$ залишається;

для $A_{тp2}$ заявка 2, ставка $P_2 = 2$ залишається;

для $A_{тp3}$ заявка 1, назначається ставка

$$P_3 = 13 - 10 + 1 = 4.$$

Аукціон завершується, тому що всі заявки завершені. Підсумковий час заявок для прикладу складає

$$\Phi_{\text{пр}} = 6 + 10 + 10 = 26.$$

При завданні з більшою розмірністю рішення являє собою квазіоптимальне рішення.

5.3 Програмні агенти у кібербезпеці

Еволюція інструментів виявлення та реагування на загрози після появи рішень IPS та IDS логічно привела до створення систем SIEM. Головна їхня фішка полягала в умінні збирати та зіставляти інформацію з кількох абсолютно різних джерел. Звісно, інтеграція першого покоління захисних систем на базі штучного інтелекту дала серйозний поштовх розвитку індустрії. Проте був і критичний мінус – суто реактивний підхід. Через це організації залишалися практично беззахисними перед атаками нульового дня та складними цільовими загрозами (APT). На сучасному етапі розробники перейшли до комплексної інтеграції захисних рішень. Саме так з'явилася технологія розширеного виявлення та реагування (XDR) – системи, які зв'язують розрізнені продукти безпеки в єдину, монолітну інфраструктуру [32].

5.3.1 LLM: розробка та застосування

У наукових працях часто можна зустріти абревіатуру LM (Language Model). Нею позначають класичні мовні моделі. Вони заточені під вузькі завдання – скажімо, вгадати наступне слово в рядку чи перевірити граматику. Логічно мислити такі системи не вміють. А от із LLM (Large Language Model) ситуація геть інша. Коли така модель розростається до певних масштабів, у ній раптово прокидаються так звані емерджентні властивості. Тобто з'являються вміння, яких модель ніхто навмисно не вчив: розгадувати логічні ребуси, писати код, створювати вірші або навіть пояснювати, чому анекдот смішний. Основою для цих моделей стали архітектури глибокого навчання, насамперед Transformer. Пропустивши крізь себе колосальні обсяги текстів, вони навчилися копіювати людський стиль письма у найрізноманітніших сферах. Сьогодні LLM активно проникають туди, де раніше про автоматизацію навіть не йшлося.

Взяти хоча б розробку софту чи обробку природної мови (NLP) – тут за допомогою великих моделей спокійно генерують чистий код і шукають приховані баги. Сфера кібербезпеки теж не стоїть осторонь. Зараз розробники масово впроваджують системи захисту на базі NLP, які згрібають інформацію про кіберзагрози з усіх можливих джерел. Завдяки цьому аналітики отримують прогнози щодо нових атак, готові довідкові бази, а заодно автоматизують рутину: розбір системних логів та фільтрацію тривожних сповіщень [33-38]. Ба більше, оскільки LLM добре розуміють «живу» мову людей, вони легко виловлюють фішинг у пошті, чатах чи соцмережах, а також помічають внутрішніх порушників і спроби просканувати мережу на вразливості [38]. Помічаючи дрібні аномалії чи ледь помітні відхилення від нормального стану системи, такі моделі допомагають зловити інцидент на самому початку [39]. Це дає командам безпеки дорогоцінний час для проактивного удару. До того ж, якщо постійно просіювати через LLM гігантські масиви даних із форумів, хакерських блогів та соцмереж, можна заздалегідь виявити свіжі схеми атак і набагато краще розуміти специфіку власного цифрового периметру [40]. Якщо ж ризик уже став реальністю, модель миттєво аналізує системні журнали й видає безпековій команді готовий покроковий план для ліквідації загрози [40].

5.3.2 Автоматизоване виявлення загроз за допомогою агентів LLM

Інтеграція великих мовних моделей (LLM) у захисні платформи кардинально змінила підхід до автоматичного пошуку кіберзагроз. Раніше для аналізу системних журналів зазвичай використовували прості алгоритми. Вони шукали збіги за ключовими словами або звіряли лог-файли із жорстко прописаними шаблонами атак. Проте LLM-агенти діють інакше. Вони здатні бачити глибинний контекст і давати семантичну (сміслову) оцінку кожному запису у журналі [41]. Якщо натренувати такого агента на реальних логах подій, він почне розпізнавати не просто окремі підозрілі фрази чи технічні маркери. Він бачитиме приховані взаємозв'язки між діями. Простий приклад: LLM-агент може легко виявити спробу витоку інформації за сукупністю дрібних дій у

системі, навіть якщо жоден окремий рядок у логах сам по собі не виглядав шкідливим.

5.3.3 Виявлення аномалій за допомогою контекстного розуміння

Ще одна потужна перевага LLM-агентів – вміння працювати з контекстом для точного пошуку аномалій. Стандартні статистичні методи аналізу зазвичай оцінюють лише поверхневі дані першого рівня. На противагу їм, інтелектуальні агенти здатні враховувати величезну кількість супутніх факторів, що формують загальну картину активності. Скажімо, такий агент може проаналізувати та «запам'ятати» щоденний графік конкретного працівника, його професійні звички, поточні завдання в проектах і навіть світові новини, які прямо зараз впливають на формат його роботи. Саме яке глибоке розуміння поточної ситуації дозволяє системі безпомилково відсіювати справді небезпечну поведінку зловмисників і водночас зводить до мінімуму кількість хибних тривог, які так втомлюють аналітиків безпеки [41].

5.3.4 Розпізнавання патернів у мережевому трафіку

На перший погляд, великі мовні моделі створені суто для роботи з текстом. Проте їхній математичний апарат чудово підходить і для розпізнавання прихованих закономірностей – наприклад, під час моніторингу мережевого трафіку. Головне – подати мережеві пакети у зрозумілому для LLM текстовому чи структурованому вигляді. Тоді розумні агенти зможуть проаналізувати потік даних та виявити аномальні шаблони, які вказують на хакерську активність. Якщо «прогнати» модель крізь великі масиви інформації як про нормальний трафік, так і про реальні інциденти, LLM навчиться розрізняти специфічні ознаки найрізноманітніших атак. Спектр тут величезний: від примітивного DDoS-флуду до хитромудрих цільових кібератак класу APT [42].

5.3.5 Аналіз поведінки користувачів

Створення детальних контекстних моделей, що описують звичні дії користувача, дозволяє LLM-агентам вивести аналіз активності на новий рівень. Такі моделі набагато гнучкіші за класичні жорсткі правила. Вони враховують

купу супутніх чинників: посадові обов'язки людини, її типові поведінкові звички, поточні робочі завдання та навіть внутрішню структуру компанії. Чітко розуміючи, що є нормою для кожного конкретного облікового запису, штучний інтелект набагато краще бачить тих, хто починає шкодити системі або чий акаунт банально зламали. Як це працює? LLM-агент може забити на сполох через послідовність дій, де кожен крок окремо виглядає цілком легально. Але варто оцінити їх комплексно – крізь призму ролі користувача та його поточних проєктів – як одразу стає зрозуміло, що це аномалія.

5.3.6 Виявлення загроз нульового дня

Найбільший головний біль для будь-якої команди кібербезпеки – це атаки нульового дня (*Zero-day*), які експлуатують свіжі, ніде не описані вразливості. Старі методи, що опираються виключно на бази відомих сигнатур, тут безсилі. А от LLM-агенти здатні ловити такі загрози завдяки постійному порівнянню поточного стану системи із її базовою (нормальною) поведінкою. Оскільки моделі володіють колосальним обсягом знань про архітектуру мереж та потенційні вектори зламу, вони можуть будувати логічні висновки про принципово нові шкідливі інструменти. Наприклад, агент здатний розпізнати нову небезпечну атаку просто тому, що зафіксована активність системи виглядає аномально і при цьому не збігається з жодною відомою сигнатурою легітимних процесів [43, 44].

5.3.7 Автоматизоване реагування на загрози

На рис. 5.10 показано структуру ієрархічного планувальника з агентами конкретних завдань (NPTSA). Ця архітектура тримається на трьох китах: безпосередньо ієрархічному планувальнику, агенті-координаторі (керівнику команди) та групі вузькоспеціалізованих експертних агентів. Сам планувальник взаємодіє із зовнішнім середовищем – наприклад, досліджує цільовий вебсайт, після чого формує покрокові інструкції для координатора [44].

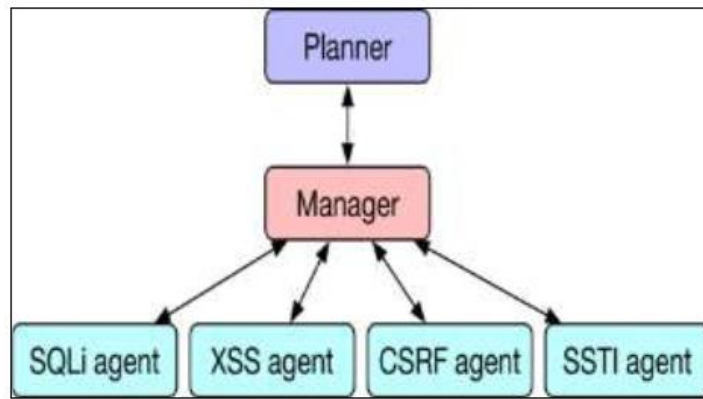


Рисунок 5.10 – Архітектура системи HPTSA на базі LLM: механізми ієрархічного планування та спеціалізовані агенти [44]

Орієнтуючись на вказівки від планувальника, агент-координатор розподіляє завдання між вільними експертними агентами та повністю контролює обмін даними всередині системи. Роль експертних агентів – точкове відпрацювання конкретних сценаріїв. Це може бути перевірка системи на вразливості типу SQL-ін'єкцій (SQLi) або міжсайтового скриптингу (XSS). Подібний підхід ідеально підходить для планового та максимально точного аудиту безпеки. Поєднання стратегічного планування верхнього рівня із глибокою спеціалізацією дозволяє гнучко й багатогранно тестувати вебдодатки на стійкість до зламів [37, 44-46].

5.3.8 Впровадження адаптивної політики безпеки

Ще одна корисна функція LLM-агентів – автоматичне коригування правил та політик безпеки «на льоту». Щойно ландшафт кіберзагроз змінюється, система реагує миттєво. Аналізуючи свіжі звіти про інциденти, нові вектори атак чи будь-які зміни всередині IT-інфраструктури підприємства, агенти здатні самостійно генерувати точні та актуальні рекомендації для оновлення захисних протоколів.

5.4 Багатоагентні системи для розподіленого виявлення та реагування на загрози

Масштаби та заплутана архітектура сучасних IT-систем змушують розробників шукати нові підходи. Одним із найперспективніших рішень тут є використання багатоагентних систем. Ідея проста: всередині інфраструктури

компанії розгортають цілу мережу взаємопов'язаних LLM-агентів. Кожен із них відповідає за свій вузький шматок роботи – наприклад, аналізує конкретний тип даних або контролює окремий периметр безпеки. Ключова фішка в тому, що ці агенти постійно комунікують між собою, обмінюються знахідками та діють як одна злагоджена команда. Утім, не варто ідеалізувати LLM-агентів, адже сфера кібербезпеки виявила низку серйозних проблем та обмежень. Перший ризик досить специфічний – це загроза атак на ШІ з боку інших, ворожих мовних моделей. Оскільки захисні системи стають усе більш автономними, хакери можуть цілеспрямовано атакувати самих агентів. Наприклад, намагатися заблокувати їх, перехопити контроль, спотворити алгоритми прийняття локальних рішень або банально підсунути скомпрометовані (отруєні) дані для навчання. Ще дві хронічні болячки штучного інтелекту – це інтерпретованість та пояснюваність (explainability). Звичайному користувачу чи навіть досвідченому аналітику буває вкрай складно розібратися, за якою саме логікою модель дійшла до того чи іншого висновку. Для кібербезпеки це величезний мінус. Тут кожен крок жорстко регулюється правилами підзвітності, а чітке розуміння причин інциденту є обов'язковим під час проведення розслідувань. Проте «чорна скринька» всередині LLM часто робить таке детальне пояснення просто неможливим [47].

5.5 Розробка архітектури МАС СУІБ

МАС – це набір з N інтелектуальних агентів, які діють разом для досягнення спільних цілей. Центр (користувач) – той, хто ставить завдання колективу.

Функціональна цілісність системи передбачає такий тип внутрішньої взаємодії її елементів при якому властивості цілого (тобто всієї системи) не можна звести до суми властивостей елементів.

Принцип колективної дії – гіпотеза про перевагу колективної дії над індивідуальними (за певних умов) – колективними діями можна досягти великих результатів ніж індивідуальними.

Роботу МАС СУІБ можливо уявити як взаємодії (суцільна лінія) в

реальному часі множини агентів, що відповідають за різні функції захисту, що реалізовані через їх поведінку в системі і формалізовані у вигляді відповідних цільових функцій та обмежень, та визначають оптимальну дію на основі аукціонів, що здійснюються за допомогою переговорів (пунктирна лінія) (рис. 5.11).

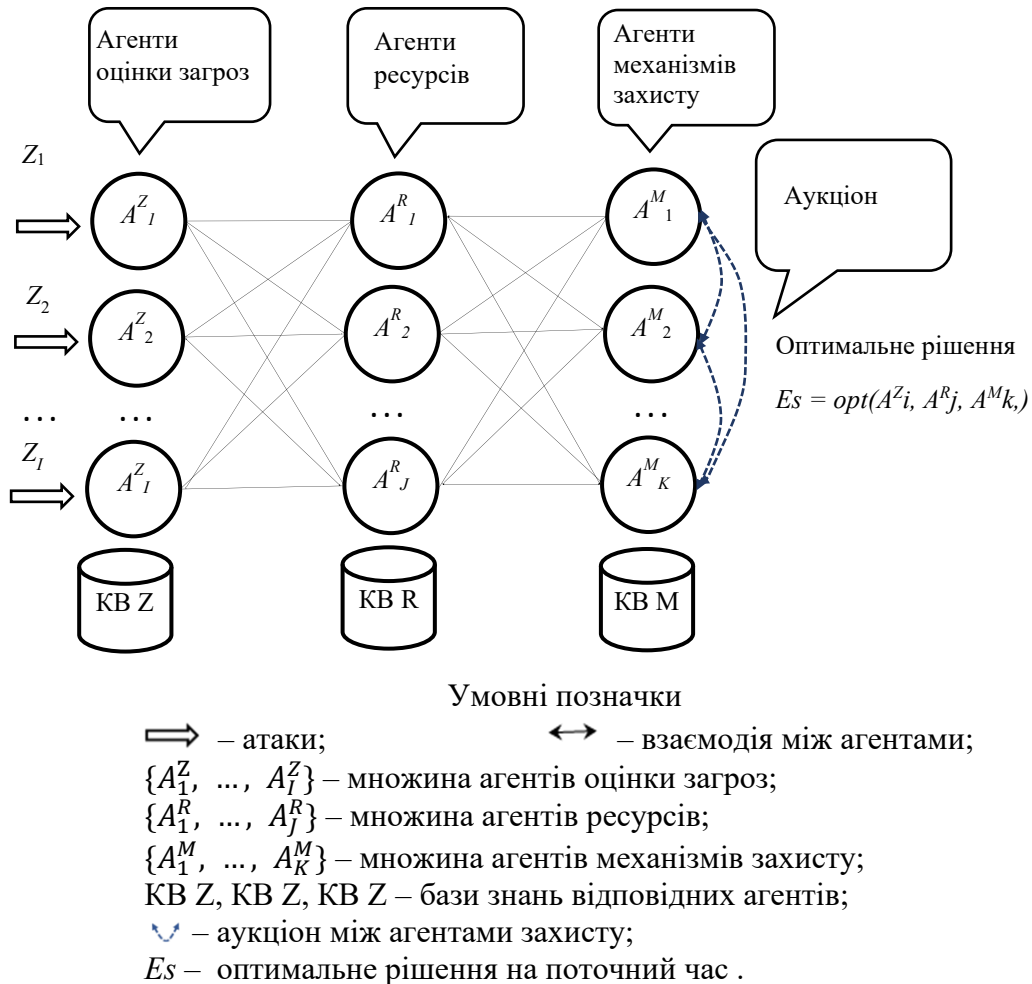


Рисунок 5.11 – Архітектура системи МАС СУІБ

Процес роботи МАС захисту інформації представлено наступними етапами (рис. 5.11).

1. Ідентифікація загроз $\{Z_1, \dots, Z_I\}$, де I – кількість загроз, що надійшли за наперед заданий фіксований період часу здійснюється множиною агентів оцінки загроз $\{A_1^Z, \dots, A_I^Z\}$, де I – кількість загроз.

Ідентифікація оцінки загроз і вразливостей полягає в послідовному виконанні наступних кроків: визначення загроз для ресурсів організації та

визначення рівня реалізації загроз без використання механізмів захисту [32].

База знань KB Z містить знання для забезпечення роботи множини агентів оцінки загроз $\{A_1^Z, \dots, A_J^Z\}$.

2. Множина агентів ресурсів $\{A_1^R, \dots, A_J^R\}$, де J – кількість ресурсів АС, згідно політики безпеки, встановлюють відповідний рівень конфіденційності, цілісності, доступності та спостережності для кожного ресурсу організації та ризику і їх пріоритети.

Рівень конфіденційності визначається ступенем важливості ресурсу і наслідками розголошення відповідної інформації, рівень цілісності ресурсу – ступенем пошкодження, фінансових втрат і можливістю відновлення, рівень доступності ресурсу – значенням максимального часу, на протязі якого недоступність ресурсу не впливає негативно на діяльність організації, рівень спостережності – ступенем повноти, якості і контролю використання ресурсу з боку авторизованих користувачів.

База знань KB R містить знання для забезпечення роботи множини агентів ресурсів $\{A_1^R, \dots, A_J^R\}$.

3. Множина агентів захисту $\{A_1^M, \dots, A_K^M\}$, де K – кількість механізмів захисту, визначають механізми захисту, що протидіють загрозам та входять до складу СЗІ, визначають рівень з яким механізми захисту зменшують загрозу, що діє на систему та вартість механізму захисту з урахуванням цінності інформації, що захищається у поточний час.

База знань KB M містить знання для забезпечення роботи множини агентів ресурсів $\{A_1^M, \dots, A_K^M\}$;

Після того, як були визначені загрози і вразливості для системи та встановлені ризику та їх пріоритети, наступним етапом є розробка впорядкованої і економічно обґрунтованої стратегії захисту.

На рис. 5.12 показано варіанти можливих дій агентів механізмів захисту згідно можливостей захисту ресурсу R_2 від усіх, діючих в поточний момент, загроз та можливі коаліції агентів захисту.

Для визначення коаліції агентів захисту K з множини усіх можливих механізмів захисту $\{A_1^M, \dots, A_K^M\}$, що протидіють загрозі та оцінки їх рівня захисту запропоновано застосувати аукціон агентів для пошуку оптимального складу множини коаліції агентів механізмів захисту за критерієм вартості захисту E_s та з урахуванням цінності інформації, що дозволить упорядкувати дії захисту, та протистояти усім поточним загрозам:

$$E_s = \text{opt}(A_i^Z, A_j^R, A_i^M) \quad m = 1, \dots, M,$$

де M – кількість коаліцій, що були відібрані для роботи МАС.

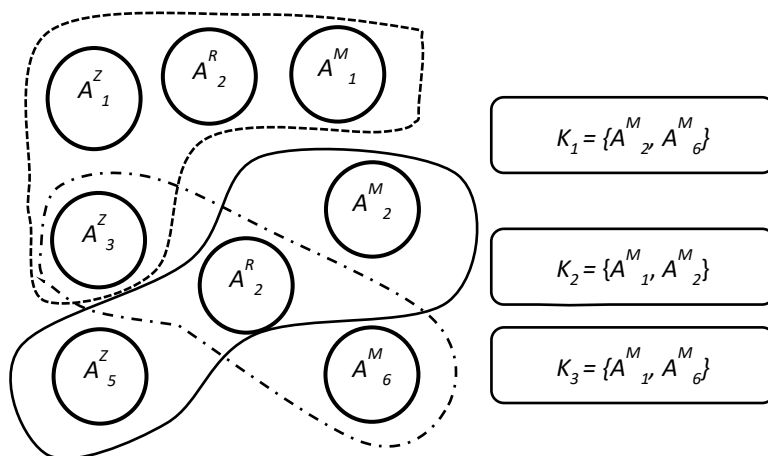


Рисунок 5.12 – Варіанти можливих коаліцій агентів механізмів захисту

Мультиагентна система захисту інформації має наступну структуру

$$MAS = \langle \{A_1^Z, \dots, A_l^Z\}, \{A_1^R, \dots, A_j^R\}, \{A_1^M, \dots, A_K^M\}, \{K_1, \dots, K_M\}, KB_1, \dots, KB_m \rangle$$

де A^Z – множина агентів загроз; A^R – множина агентів ресурсів; A^M – множина агентів механізмів захисту; $\{K_1, \dots, K_M\}$ – множина створених коаліцій агентів захисту під час роботи МАС, де M – кількість створених коаліцій; $\{KB_1, \dots, KB_M\}$ – бази даних та знань для роботи МАС.

Взаємодія множини агентів, що відповідають за різні функції системи, реалізована через їх поведінку в системі та формалізована у вигляді відповідних цільових функцій та обмежень, та визначають оптимальну дію на основі аукціонів, що здійснюються за допомогою переговорів.

Це буде оптимальне рішення для поточного стану АС. Через деякий час, коли відбудеться зміна середовища (сцени), здійсняться нові атаки і МАС

зробить перерахунок нової коаліції механізмів захисту K_m за подібним сценарієм для вже нових умов.

Висновок про ефективність системи захисту E_s робиться за рахунок глобального оптимального рішення за часом та складається з множини сукупної дії усіх коаліцій K_m :

$$E_s = (K_1 + K_2 + \dots + K_m), m = 1, \dots, M$$

де M – кількість коаліцій, що за поточний час були обрані як кращі за визначеними критеріями щодо коаліції.

Вважаючи можливості та здібності агентів до моделювання загроз, процесу виявлення, прогнозування та визначення загроз на рівні підприємства в майбутньому агент може приймати про-активні, а не реактивні рішення щодо безпеки. Це допоможе активізувати необхідні засоби захисту даних ще на ранніх стадіях процесу захисту інформації.

Також, при такому підході, множина агентів, що працює у реальному режимі часу, може протистояти віяловим атакам за рахунок великої кількості агентів та можливості їх навчання.

У ході дослідження було проаналізовано парадигму програмних агентів як інтелектуальної основи для створення сучасних систем захисту інформації.

Визначено, що програмний агент (ПА) є автономною сутністю, здатною сприймати середовище через сенсори та впливати на нього через ефектори.

Доведено, що для вирішення складних завдань кібербезпеки ефективнішим є використання децентралізованих спільнот агентів. Це забезпечує синергетичний ефект, де колективне знання системи перевищує суму знань окремих компонентів. Ключовою моделлю взаємодії визначено CPS (Cooperative Problem Solving), яка структурує процес формування кооперативного рішення через чотири етапи: від визначення потреби у співпраці до спільних дій.

Для оптимізації управління коаліціями (наприклад, групами роботів чи захисних модулів) запропоновано використання алгоритмів аукціонів. Метод, заснований на обчисленні "жалю" (втрат коаліції), дозволяє в режимі реального часу розподіляти завдання між вільними агентами, мінімізуючи сумарний час виконання та підвищуючи загальну ефективність системи.

Окрему увагу приділено новому поколінню агентів на базі великих мовних моделей (LLM). Встановлено, що вони здатні подолати обмеження класичних систем (SIEM, IDS), оскільки володіють контекстним розумінням, здатні аналізувати неструктуровані дані, виявляти загрози "нульового дня" та автоматизувати складні процеси реагування (триаж сповіщень, генерація планів стримування).

Запропоновано концептуальну схему мультиагентної системи управління інформаційною безпекою, що включає три спеціалізовані групи агентів: оцінки загроз, ресурсів та механізмів захисту. Така тривірнева структура дозволяє динамічно адаптувати стратегію захисту під конкретні бізнес-пріоритети та актуальний ландшафт загроз.

Перехід від жорстких алгоритмів до адаптивних мультиагентних систем дозволяє трансформувати кіберзахист із реактивного в інтелектуальний та проактивний, що є критично важливим для протидії динамічним загрозам у сучасних кіберфізичних середовищах.

5.6 Розробка корпоративної системи RAG

Велика мовна модель (LLM) володіє потужним інтелектуальним потенціалом, що дозволяє їй генерувати природні тексти та глибоко розуміти людську мову. Проте є суттєва проблема – залежність від статичних навчальних даних. Через це модель не здатна адекватно відповідати на динамічні запити в режимі реального часу, видаючи застарілу або відверто помилкову інформацію. Ефективним розв'язанням цієї проблеми стала технологія пошуково-доповненої генерації (RAG). Вона прокачує можливості LLM завдяки підключенню зовнішнього пошуку актуальних даних під кожен конкретний запит [48]. Але й класичні RAG-системи мають свої ліміти. Зазвичай вони обмежені жорсткими статичними сценаріями, тому їм бракує гнучкості для складного багатокрокового моделювання та управління комплексними процесами. Переступити ці обмеження дозволяє концепція агентного RAG (Agentic RAG). Її суть полягає у впровадженні автономних ШІ-агентів безпосередньо в архітектуру RAG. Такі агенти спираються на передові патерни проектування: рефлексію, стратегічне планування, використання зовнішніх інструментів та

міжагентну кооперацію. Це дає їм змогу динамічно керувати пошуковими алгоритмами, ітеративно уточнювати контекст та гнучко адаптувати робочі процеси під поточні завдання – від виконання послідовних кроків до складної адаптивної взаємодії. Зрештою, інтеграція Agentic RAG забезпечує системі високу масштабованість та чудове розуміння контексту в реальних бізнес-додатках [48].

RAG – це не якась універсальна «коробкова» технологія. Це цілий комплекс взаємопов'язаних модулів та налаштувань під конкретну предметну область. Стандартна архітектура RAG зазвичай складається з таких елементів [49-50]:

Модуль інжинірингу (прийому) даних. Тут інформація проходить попередню обробку, розбивається на фрагменти, переводиться у векторний вигляд (ембедінги) та завантажується для зберігання у векторну базу знань.

Пошуковий контур. Його завдання – оперативне витягувати з бази потрібні документи та ранжувати їх за рівнем релевантності до запиту користувача.

Модуль обробки запитів. Він зшиває початковий запит користувача із результатами пошуку у фінальний промпт, який і надсилається на обробку в LLM.

Приклад коду для локального корпоративного RAG-асистента, розробленого мовою Python та адаптованого під лекційні матеріали курсу ОШІ, наведено у додатку Б.1. Результати тестування інтерфейсу та фрагменти реального діалогу з цим асистентом можна переглянути у додатках Б.2 та Б.3 відповідно. Причому для генерації та фінального відлагодження цього програмного коду використовувався інструментарій сучасної мовної моделі Gemini від компанії Google (версія станом на червень 2026 року).

ВИСНОВКИ

На основі проведеного дослідження вдалося сформувати цілісну концепцію інтелектуального управління кібербезпекою. В її основу ліг синтез нечіткої логіки, мультиагентного моделювання та передових LLM-технологій. Основні наукові та практичні результати роботи зводяться до кількох ключових пунктів.

По-перше, доведено, що для аналізу сучасних кіберзагроз ефективним інструментом є використання нечіткої логіки та лінгвістичних змінних. Саме такий підхід дозволяє перевести суб'єктивні експертні оцінки (на кшталт «низький» чи «високий» ризик) у чітку математичну площину без втрати сенсу. Це повністю узгоджується з вимогами міжнародних стандартів ISO/IEC 27001 та 27005.

По-друге, завдяки методу аналізу ієрархій Сааті було структуровано складну архітектуру безпеки та математично обґрунтували вибір конкретних засобів захисту. Розрахунки підтвердили: оптимальна стратегія кіберзахисту підприємства має опиратися на технологічну перевагу (швидкість реакції та рівень детекції), а не на банальну мінімізацію фінансових витрат.

По-третє, практична реалізація системи підтримки прийняття рішень в середовищі MATLAB дозволила інтегрувати 10 факторів безпеки у 5 комплексних критеріїв. За допомогою інструментарію Fuzzy Logic Toolbox ми реалізували алгоритм нечіткого висновку Мамдані, який базується на 40 агрегованих правилах. Запропонований метод корекції висоти функцій належності дозволив підняти точність оцінки ризиків на 3–10 %.

По-четверте, у роботі детально обґрунтовано перехід до децентралізованих мультиагентних систем, де компоненти можуть самостійно координувати свої дії. Встановлено, що інтеграція автономних агентів на базі великих мовних моделей дозволяє автоматизувати сортування тривоги, глибоко аналізувати неструктуровані масиви лог-файлів та впевнено ловити загрози нульового дня завдяки розумінню контексту подій.

По-п'яте, для реалізації автономної взаємодії захисних модулів використано мову Python та математичний апарат аукціонних алгоритмів. Програмна реалізація механізму обчислення «жалю» (оцінки втрат коаліції) дозволяє в режимі реального часу розподіляти завдання між вільними агентами. Це оптимізує ресурси та суттєво скорочує час реакції на інциденти.

Нарешті, впровадження пошуково-доповненої генерації (RAG) стало, без перебільшення, тим самим критично важливим інструментом, завдяки якому вдалося нарешті витягнути сучасні мовні моделі з інформаційного вакууму та інтегрувати їх у реальне життя. Підключення локальних корпоративних баз знань розв'язує хронічну проблему ШІ – воно зводить до мінімуму вигадки (так звані галюцинації) й гарантує аналітикам отримання свіжої та точної інформації.

Проте класичний статичний RAG уже не задовольняє сучасні потреби захисту. Саме тому на зміну йому приходить концепція автономних агентних архітектур (Agentic RAG), яка відкриває якісно нові горизонти. Отримавши внутрішні функції самоаналізу та можливість покроково планувати власні дії, такі системи трансформувалися зі звичайних пошукових скриптів на повноцінних цифрових помічників, здатних самостійно розплутувати складні інциденти безпеки та гнучко підлаштовуватися під зміни в бізнес-процесах. І тут є принциповий момент: розгортати подібні платформи потрібно виключно локально, на власних обчислювальних потужностях підприємства (що ми й проілюстрували на прикладі створеного асистента).

Для сучасного бізнесу це єдиний адекватний спосіб зберегти реальну конфіденційність корпоративних даних та повністю нівелювати ризики витоку інформації через публічні хмарні API-сервіси.

Стрімка інтеграція інтелектуальних агентів у контур безпеки одразу ж підняла руба складну проблему «чорної скриньки». Адміністратори захисту не мають права сліпо довіряти автоматиці – їм потрібна чітка й покрокова пояснюваність кожного рішення, яке приймає модель. Крім того, на перший план виходить жорсткий контроль за упередженістю масивів даних для

навчання, оскільки дотримання принципів справедливості та прозорості є обов'язковим під час автоматизації систем захисту.

У підсумку цей підхід повністю перегортає уявлення про побудову захисту, трансформуючи його з пасивного очікування у проактивне полювання за кіберзагрозами. Поєднання гнучкого програмного коду на Python та MATLAB із потужним математичним апаратом нечіткої логіки та МАІ дозволяє створити адаптивну екосистему, яка практично повністю виключає людський фактор і надійно закриває цифрові активи компанії від будь-яких динамічних векторів загроз.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Прохорова О.М. Моделі і методи нечіткої логіки: навч. посіб. [Рукопис] / О.М. Прохорова, Н. В. Кальчук; Нац. аерокомс. ун-т ім. Н.Є. Жуковського “ХАІ”. – Х., 2021. – 166 с.
2. Аналіз методик оцінки ризиків інформаційної безпеки / А. Е. Лагун, Ю. П. Данилюк, О. Р. Мороз, Р. С. Юськів // Вісник Львівського державного університету безпеки життєдіяльності. – 2014. – № 9. – С. 45–52.
3. Tubis, A.; Werbinska-Wojciechowska, S.; Góralczyk, M.; Wróblewski, A.; Zietek, B. Cyber-Attacks Risk Analysis Method for Different Levels of Automation of Mining Processes in Mines Based on Fuzzy Theory Use. *Sensors* 2020, 20, 7210.
4. Kuzminykh, I.; Ghita, B.; Sokolov, V.; Bakhshi, T. Information Security Risk Assessment. *Encyclopedia* 2021, 1(3), 602-617; <https://doi.org/10.3390/encyclopedia1030050>.
5. Romashchenko N. The algorithm of information security risk assessment based on Fuzzy-multiple approach/ International competition of Student scientific works «Black Sea Science 2020, Information Technology, Automation and Robotics», 2020, p. 242-251. <https://card-file.ontu.edu.ua/handle/123456789/14490>
6. Soetanto, D. P. (2000). Implementing fuzzy logic in determining selling price. *Jurusan Teknik Industry*, 2(1), 42- 52. <https://doi.org/10.9744/jti.2.1.43-52>
7. Wentian, C.; Huijun, Y. Research on Information Security Risk Assessment Method Based on Fuzzy Rule Set. *Wirel. Commun. Mob. Comput.* 2021, <https://doi.org/10.1155/2021/9663520>.
8. Zadeh L. A. Fuzzy sets. *Information and Control*. 1965. Vol. 8, no. 3. P. 338–353.
9. Bellman R. E., Zadeh L. A. Decision-making in a fuzzy environment. *Management Science*. 1970. Vol. 17, no. 4. P. 141–164.
10. Теорія ризиків : навч. посіб. / В. В. Поповський та ін. Харків : НТУ «ХПІ», 2025. 408 с.

11. Бережний І. В. Теорія прийняття рішень : навч. посіб. Київ : Центр учбової літератури, 2019. 248 с.

12. Корченко О. Г. Побудова систем захисту інформації : підручник. Київ : Нац. авіац. ун-т, 2018. 320 с.

13. Федоров М. В. Системний аналіз : конспект лекцій для студентів 2 курсу денної форми навчання освітнього рівня «бакалавр» спеціальності 126 – Інформаційні системи та технології / М. В. Федоров, О. М. Хренов, О. М. Штельма ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 62 с.

14. Методичні рекомендації до проведення практичних занять із дисципліни «Системний аналіз» (для здобувачів першого (бакалаврського) рівня вищої освіти всіх форм навчання спеціальності 126 – Інформаційні системи та технології) / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова ; уклад. О. М. Штельма. – Харків : ХНУМГ ім. О. М. Бекетова, 2023. – 28 с.

15. Saaty T. L. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*. 2008. Vol. 1, No. 1. P. 83–98.

16. Arrow, K.J.; Kruz, M. *Public Investment, the Rate of Return, and Optimal Fiscal Policy*; RFF Press: New York, NY, USA, 2013; p. 218. <https://doi.org/10.4324/9781315064178>.

17. Gaultier-Gaillard, S.; Louisot, J.P.; Rayner, J. Managing reputational risk—From theory to practice. In *Reputation Capital: Building and Maintaining Trust in the 21st Century*; Springer: Berlin/Heidelberg, Germany, 2009. https://doi.org/10.1007/978-3-642-01630-1_11

18. Khavina I. P., Tsuranov M. V. Research on the fuzzy logic mechanism for assessing enterprise information risks//Modern research in science and education. Proceedings of the 4th International scientific and practical conference. BoScience Publisher. Chicago, USA. 2023. Pp. 329 -335. URL: <https://sci-conf.com.ua/iv-mizhnarodna-naukovo-praktichna-konferentsiya-modern-research-in-science-and-education-7-9-12-2023-chikago-ssha-arhiv/>

19. Shukla A, Katt B., Nweke L.O. et al. System security assurance: A systematic literature review Volume 45, 2022, p. 29. <https://doi.org/10.1016/j.cosrev.2022.100496>.

20. ISO/IEC 27000: Information technology – Security techniques – Information security management systems – Overview and vocabulary (2023)

21. IBM Corporation: IT Security Compliance Management Design Guide with IBM Tivoli Security Information and Event Manager. 2nd edn. (2010). <http://www.redbooks.ibm.com/abstracts/sg247530.html?Open>. Accessed 05 June 2017

22. Miller, D., Harris, S., Harper, A., VanDyke, S.: Security Information and Event Management (SIEM) Implementation. McGraw-Hill, New York (2010). 464 p.

23. Stallings W. Pearson EText for Cryptography and Network Security: Principles and Practice, Global Edition. Pearson Education, Limited, 2022. P. 833.

24. Khraisat, A., Gondal, I., Vamplew, P. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2, 20 (2019). p. 22. <https://doi.org/10.1186/s42400-019-0038-7>.

25. Сучасні інформаційні технології в кібербезпеці : монографія / А. С. Довбиш, В. К. Ободяк, І. В. Шелехов та ін. ; за ред. В. К. Ободяка, І. В. Шелехова. – Суми : Сумський державний університет, 2021. – 348 с.

26. Зоря І.С., Марущак А.В. Застосування штучного інтелекту для виявлення та реагування на кіберзагрози // Вісник Вінницького політехнічного інституту. № 1, 2023, 4-8.

27. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley & Sons, Incorporated, John, 2020. <https://onlinelibrary.wiley.com/doi/epdf/10.1002/9781119644682.index>.

28. Болюбаш Н. М. Інтелектуальний аналіз даних : навч. посіб. / Н.М. Болюбаш. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 320 с.

29. Плєскач В.Л., Рогушина Ю.В. Агентні технології: Монографія. – К.: Київ. нац. торг.-екон. ун-т, 2005. – 344 с.

30. Бочкарьов О.Ю., Голембо В.А. Моделі колективної поведінки вимірювальних агентів // Вісник НУ «Львівська політехніка» «Комп'ютерні системи та мережі». – 2002. – № 463. – С. 19-27.

31. Khavina I. P., Lymarenko V. V. DSS controlling a machine manufacturing // General and complex problems of technical sciences: experience of EU countries

and implementation in the practice of Ukraine : collective monograph. Riga : Baltija Publishing, 2019. Pp. 319–337.

32. Хавіна І. П., Гнусов Ю. В., Можаяев О. О. Розробка мультиагентної системи управління інформаційною безпекою. *Право і безпека*. 2022. № 4 (87). С. 171–183. DOI: <https://doi.org/10.32631/pb.2022.4.14>.

33. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions, *Electronics*, vol. 12, no. 6, pp. 1–42, Mar. 2023, doi:<https://doi.org/10.3390/electronics12061333>

34. Attri, Large Language Models: Basics, Working & Examples Attri AI Blog Attri.ai Blog, attri.ai, 2024. <https://attri.ai/blog/introduction-to-large-language-models>

35. R. Kaur, D. Gabrijelčić, and T. Klobučar, Artificial Intelligence for Cybersecurity: Literature Review and Future Research Directions, *Information Fusion*, vol. 97, no. 101804, p. 101804, Apr. 2023, doi:<https://doi.org/10.1016/j.inffus.2023.101804>.

36. IBM, What Are Large Language models? | IBM, www.ibm.com, 2023. <https://www.ibm.com/topics/largelanguage-models>

37. Dylan, Utilizing Generative AI and LLMs to Automate Detection Writing, Medium, May 10, 2024. <https://medium.com/@dylanhwilliams/utilizing-generative-ai-and-llms-to-automate-detection-writing-5e4ea074072e>.

38. S. Swamy, N. Tabari, C. Chen, and R. Gangadharaiyah, Contextual Dynamic Prompting for Response Generation in Task-oriented Dialog Systems, *ACLWeb*, May 01, 2023. <https://aclanthology.org/2023.eacl-main.226>

39. M. A. Teixeira, M. Zolanvari, K. M. Khan, R. Jain, and N. Meskin, Flow-based intrusion detection algorithm for supervisory control and data acquisition systems: A real-time approach, *IET Cyber-Physical Systems: Theory & Applications*, May 2021, doi: <https://doi.org/10.1049/cps2.12016>.

40. H. Xu *et al.*, Large Language Models for Cyber Security: A Systematic Literature Review, *arxiv.org*, 2018. <https://arxiv.org/html/2405.04760v1> (accessed Jul. 24, 2024).

41. Ramasankar Molleti, Vinod Goje, Puneet Luthra and Prathap Raghavan. Automated threat detection and response using LLM agents. *World Journal of Advanced Research and Reviews*, 2024, 24(2), 079-090. Article DOI: <https://doi.org/10.30574/wjarr.2024.24.2.3329>

42. P. Tseng, Z. Yeh, X. Dai, and P. Liu, Using LLMs to Automate Threat Intelligence Analysis Workflows in Security Operation Centers, *arxiv.org*, 2024. <https://arxiv.org/html/2407.13093v1#:~:text=By%20leveraging%20the%20advanced%20capabilities>

43. D. Kang, LLM Agents can Autonomously Exploit Zero-day Vulnerabilities, *Medium*, Jun. 05, 2024. <https://medium.com/@danieldkang/llm-agents-can-autonomously-exploit-zero-day-vulnerabilities-e4664d7c598e>

44. R. Fang, R. Bindu, A. Gupta, Q. Zhan, and D. Kang, Teams of LLM Agents can Exploit Zero-Day Vulnerabilities, *arxiv.org*, 2024. <https://arxiv.org/html/2406.01637v1#:~:text=Researchers%20have%20shown%20that%20LLM>

45. Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, A survey on Large Language Model (LLM) security and privacy: The Good, The Bad, and The Ugly, *High-Confidence Computing*, vol. 4, no. 2, p. 100211, Mar. 2024, doi: <https://doi.org/10.1016/j.hcc.2024.100211>.

46. Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security, *arxiv.org*. <https://arxiv.org/html/2401.05459v1>

47. V. du Preez *et al.*, From bias to black boxes: understanding and managing the risks of AI – an actuarial perspective, *British Actuarial Journal*, vol. 29, p. e6, Jan. 2024, doi: <https://doi.org/10.1017/S1357321724000060>.

48. Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Enhancing ai systems with agentic workflows patterns in large language model. In *IEEE World AI IoT Congress (AIIoT)*, pages 527–532, 2024.

49. Aditi Singh , Abul Ehtesham , Saket Kumar , Tala Talaei Khoei , Athanasios V. Vasilakos. Agentic retrieval-augmented generation: a survey on AGENTIC RAG. <https://arxiv.org/abs/2501.09136v4> 2501.09136v4.pdf, 2026.

50. Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

ДОДАТОК А

ПРОЦЕС СТВОРЕННЯ СППР

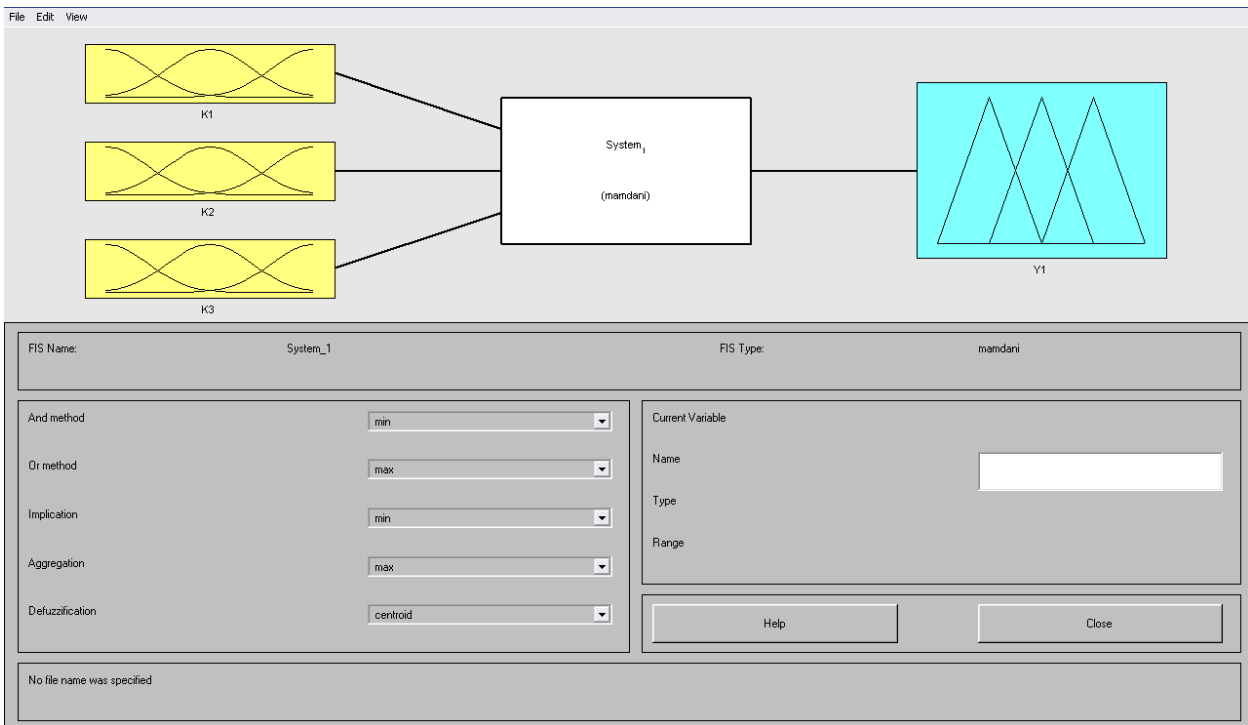


Рисунок А.1 – Структура системи System_1

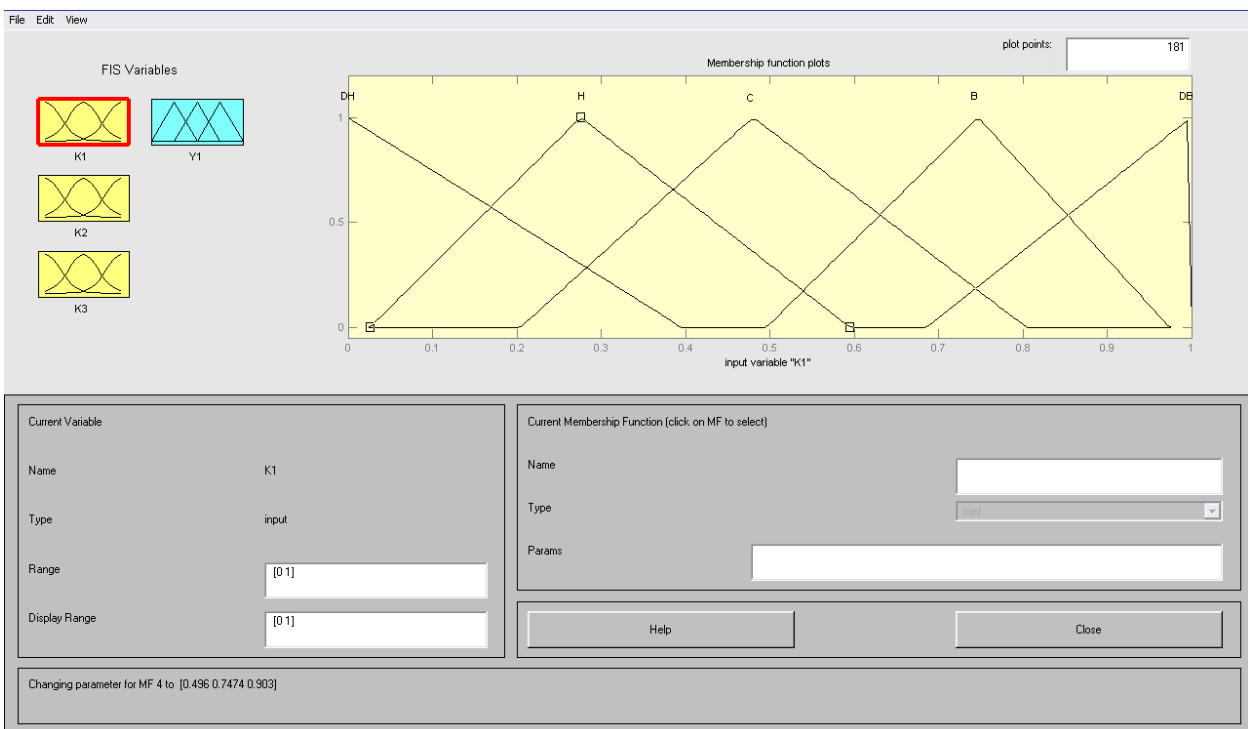


Рисунок А.2 – Визначення ЛЗ для K₁

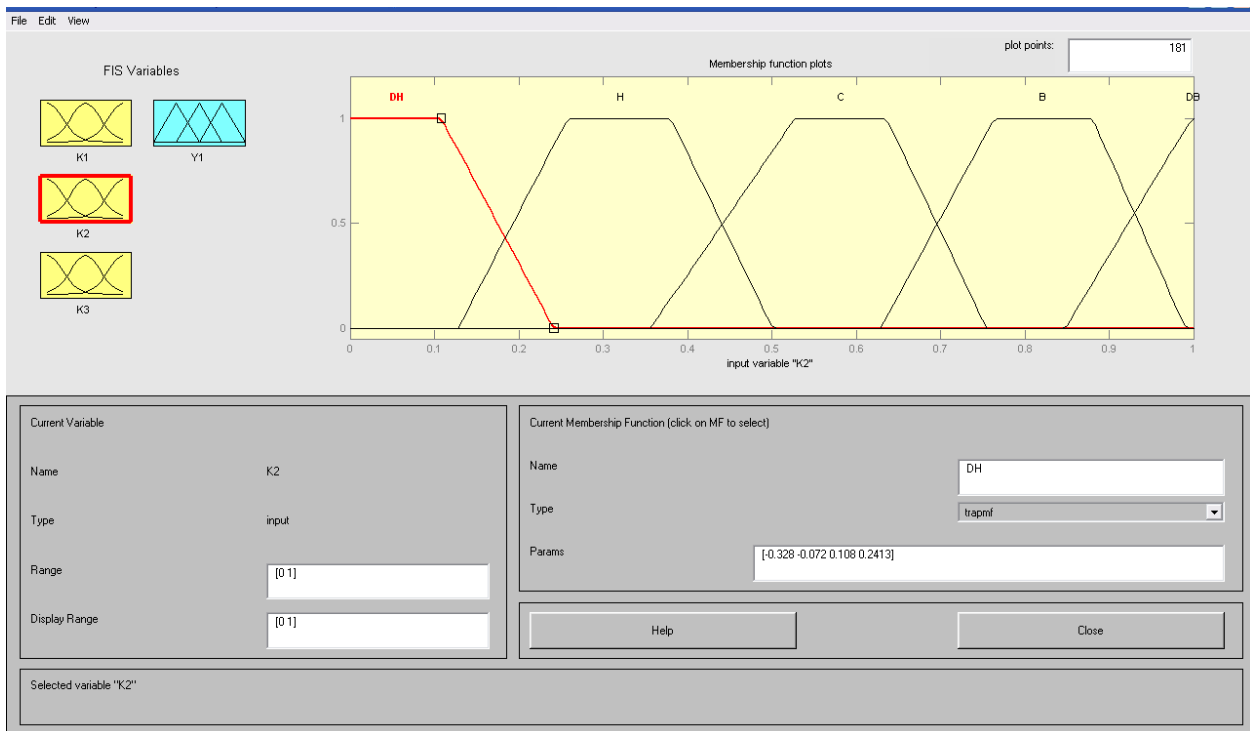


Рисунок А.3 – Визначення ЛЗ для K_2

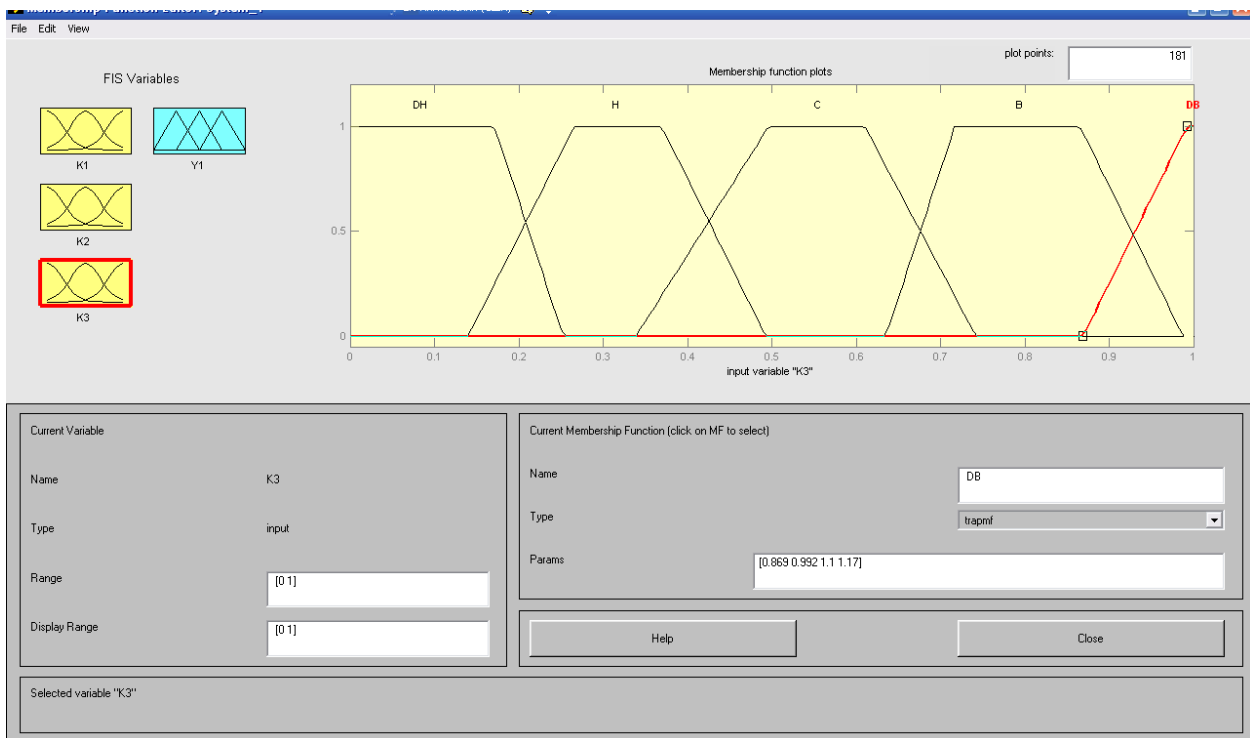


Рисунок А.4 – Визначення ЛЗ для K_3

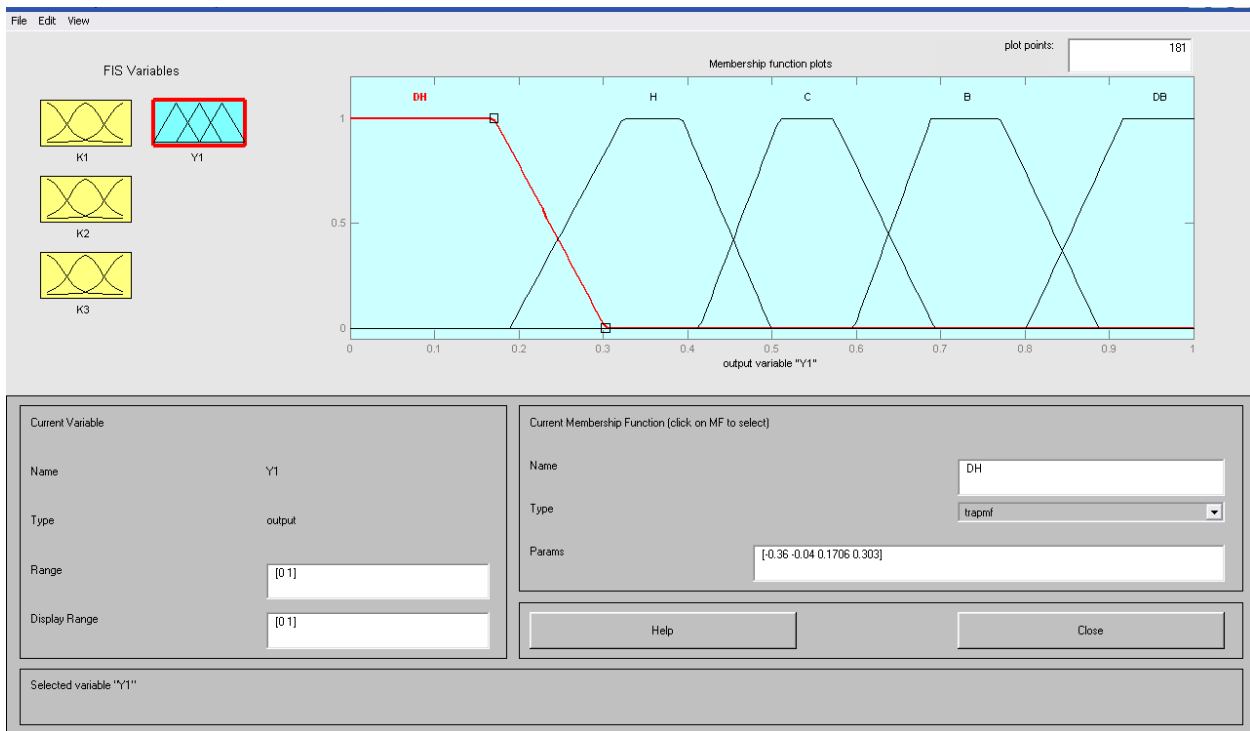


Рисунок А.5 – Визначення ЛЗ для Y_1

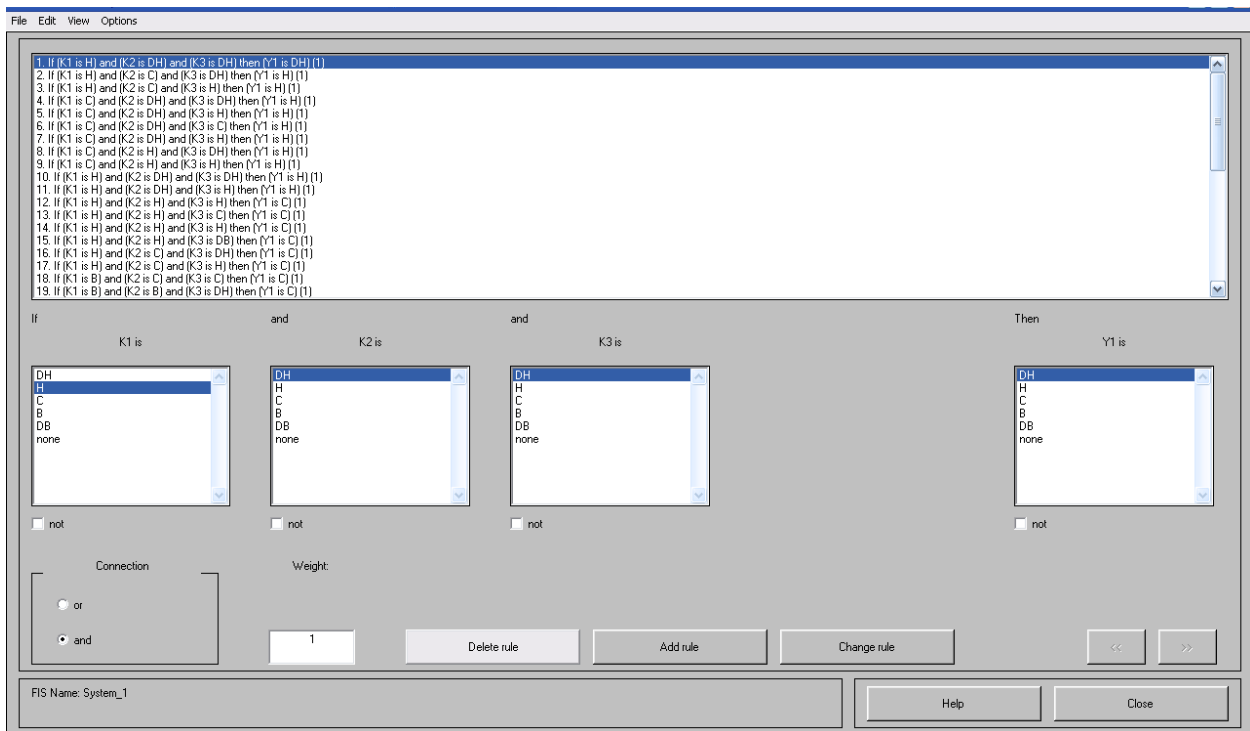


Рисунок А.6 – Нечіткі правила для визначення Y_1

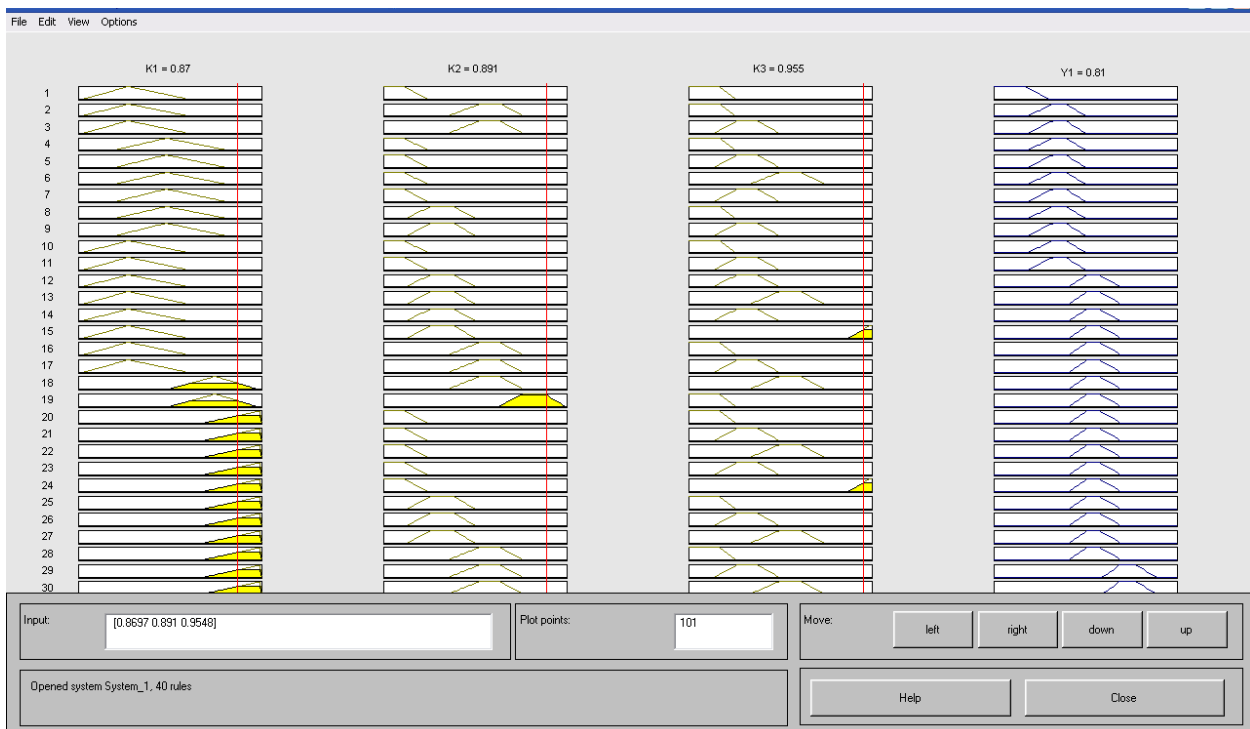
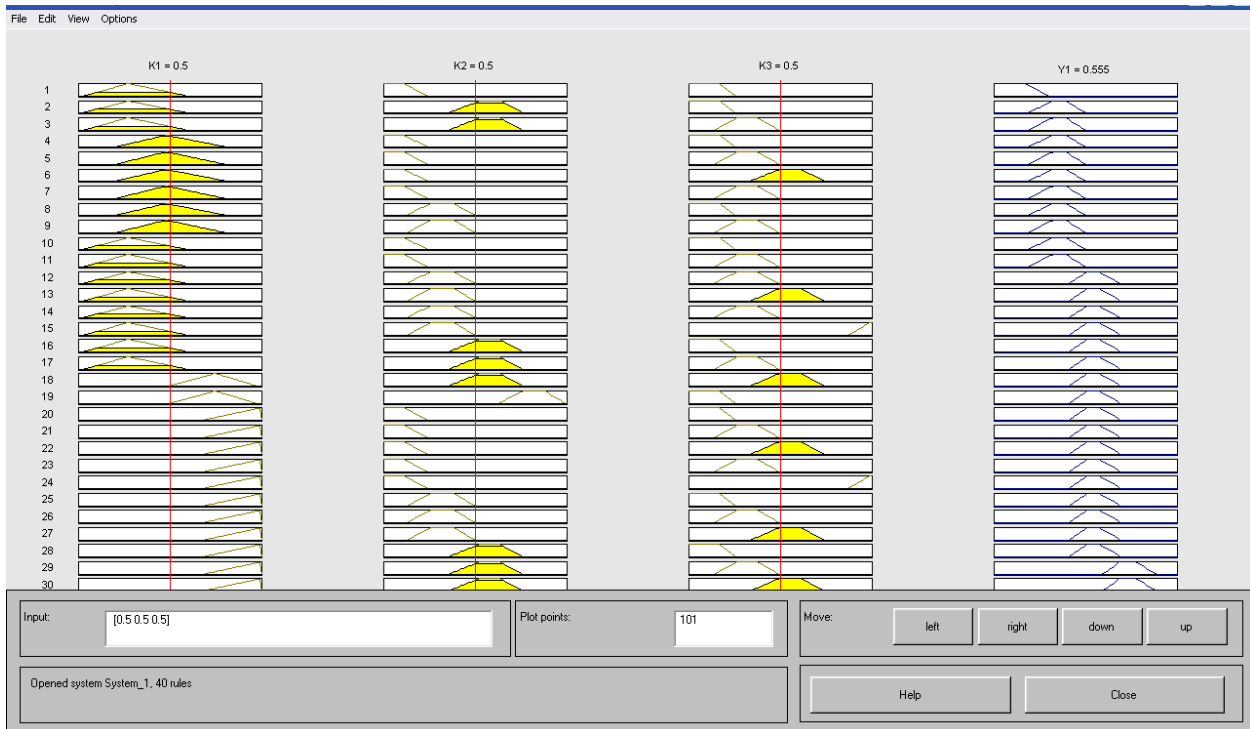


Рисунок А.7 – Приклади визначення Y_1 з залежності від K_1 , K_2 та K_3

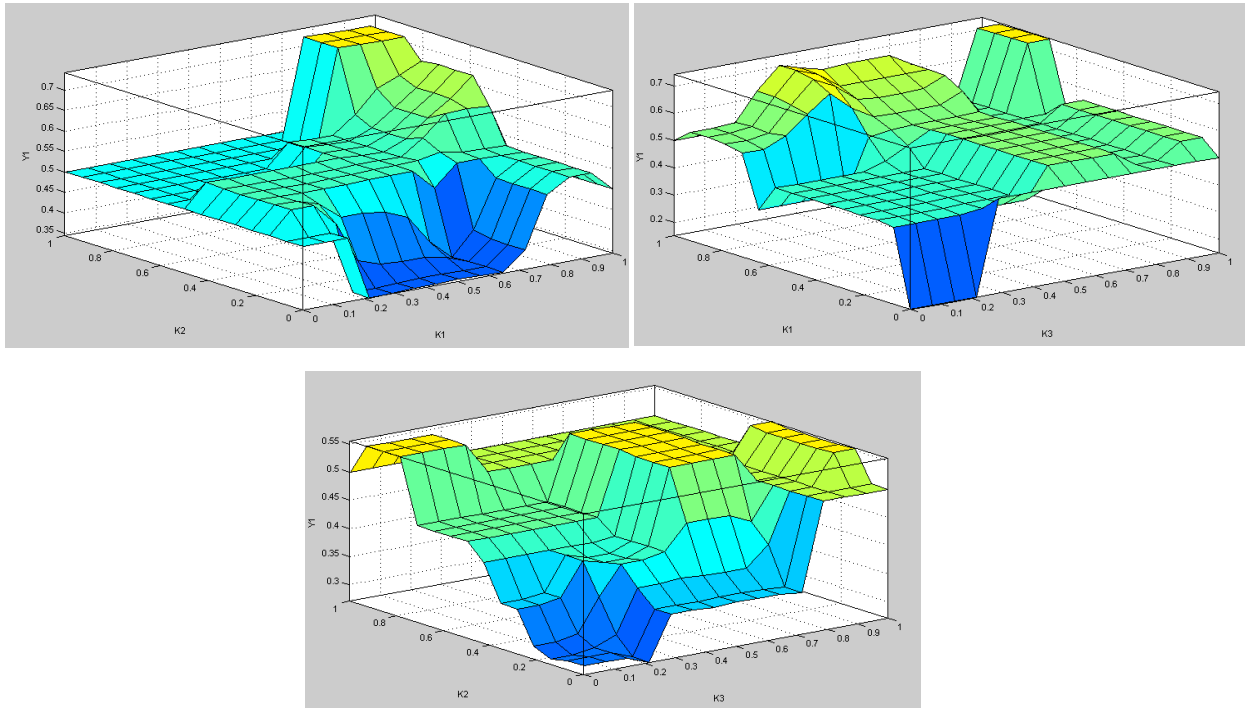


Рисунок А.8 – 3-D поверхні визначення Y_1

ДОДАТОК Б

Б.1 ПРОГРАМНИЙ АГЕНТ LM AI

```
import os
import sys
import io
import datetime
import time
import threading
import glob
import pydantic
import tkinter as tk
from tkinter import scrolledtext
import customtkinter as ctk
# 1. НАЛАШТУВАННЯ ТА КОДУВАННЯ
pydantic.v1 = pydantic
BASE_DIR = r"C:\D_2\AI_25-26\MY_AI_pro"
DB_DIR = os.path.join(BASE_DIR, "db_storage")
LOG_FILE = os.path.join(BASE_DIR, "answers.txt")
try:
    sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding='utf-8',
errors='replace')
except Exception:
    pass
from langchain_ollama import ChatOllama, OllamaEmbeddings
from langchain_community.document_loaders import PyPDFLoader,
DirectoryLoader
from langchain_chroma import Chroma
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_core.prompts import ChatPromptTemplate
```

```

from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

# Глобальна змінна для ланцюжка RAG
rag_chain = None

# 2. ГРАФІЧНИЙ ІНТЕРФЕЙС (ОПТИМІЗОВАНО ДЛЯ СТАТТІ)
ctk.set_appearance_mode("light")

class ChatApp(ctk.CTk):
    def __init__(self):
        super().__init__()
        self.title("AI Асистент ОШІ")
        self.geometry("1100x750")

        # Світло-сірий фон головного вікна
        self.configure(fg_color="#f8f9fa")
        self.stop_flag = False
        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)

        # Вікно чату: чистий білий фон, великий чорний текст (розмір 14)
        self.chat_history = scrolledtext.ScrolledText(
            self, wrap=tk.WORD, font=("Segoe UI", 14),
            bg="#ffffff", fg="#1a1a1a", insertbackground="#000000",
            highlightthickness=1, highlightbackground="#dbdbdb"
        )
        self.chat_history.grid(row=0, column=0, padx=20, pady=(20, 5),
            sticky="nsew")
        self.chat_history.config(state=tk.DISABLED)

        # Прогрес-бар

```

```

self.progress = ctk.CTkProgressBar(self, orientation="horizontal",
mode="indeterminate")

self.progress.grid(row=1, column=0, padx=20, pady=(0, 5), sticky="ew")

self.progress.set(0)

# Інформаційний рядок
self.info_label = ctk.CTkLabel(
    self, text=f"Історія запитів: {LOG_FILE} (натисніть, щоб відкрити)",
    font=("Segoe UI", 10), text_color="#555555", cursor="hand2"
)

self.info_label.grid(row=2, column=0, padx=20, pady=(0, 5), sticky="w")

self.info_label.bind("<Button-1>", lambda e: self.open_log_file())

# Панель вводу
self.input_frame = ctk.CTkFrame(self, fg_color="#ffffff",
border_color="#dbdbdb", border_width=1)

self.input_frame.grid(row=3, column=0, padx=20, pady=(0, 20), sticky="ew")

self.entry = ctk.CTkEntry(
    self.input_frame, placeholder_text="Зачекайте, завантаження бази
знань...",
    height=45, fg_color="#ffffff", text_color="#1a1a1a",
placeholder_text_color="#888888",
    border_color="#dbdbdb"
)

self.entry.pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(10, 10),
pady=10)

# Кнопки
self.send_btn = ctk.CTkButton(self.input_frame, text="Запитати", width=110,
height=45, fg_color="#1a73e8", hover_color="#1557b0", text_color="white",
command=self.send_message, state=tk.DISABLED)

self.send_btn.pack(side=tk.LEFT, padx=3, pady=10)

```

```

self.cancel_btn = ctk.CTkButton(self.input_frame, text="Скасувати",
width=110, height=45, fg_color="#f4b400", hover_color="#c39000",
text_color="white", command=self.cancel_question, state=tk.DISABLED)

self.cancel_btn.pack(side=tk.LEFT, padx=3, pady=10)

self.clear_btn = ctk.CTkButton(self.input_frame, text="Очистити",
width=110, height=45, fg_color="#00bac7", hover_color="#00949f",
text_color="white", command=self.clear_chat, state=tk.DISABLED)

self.clear_btn.pack(side=tk.LEFT, padx=3, pady=10)

self.exit_btn = ctk.CTkButton(self.input_frame, text="Вихід", width=110,
height=45, fg_color="#d93025", hover_color="#b3241b", text_color="white",
command=self.destroy)

self.exit_btn.pack(side=tk.LEFT, padx=10, pady=10)

self.start_ai_initialization()
def start_ai_initialization(self):
self.progress.start()

self.safe_update_chat(" ⚙️ Системне повідомлення: Завантаження бази
знань та моделей AI... Будь ласка, зачекайте.\n")

threading.Thread(target=self.async_init_rag, daemon=True).start()
def async_init_rag(self):
global rag_chain

try:
os.makedirs(DB_DIR, exist_ok=True)

current_pdfs = glob.glob(os.path.join(BASE_DIR, "*.pdf"))

if not current_pdfs and not (os.path.exists(DB_DIR) and
os.listdir(DB_DIR)):

self.safe_update_chat("❌ Помилка: PDF-файли не знайдено в
директорії, і база даних порожня!\n\n")

self.after(0, self.progress.stop)

return

embeddings = OllamaEmbeddings(model="nomic-embed-text")

llm = ChatOllama(model="llama3:8b-instruct-q4_K_M", temperature=0)

```

```

if os.path.exists(DB_DIR) and os.listdir(DB_DIR):

    self.safe_update_chat(f"📁 Знайдено готову базу. Швидко
завантаження ({len(current_pdfs)} файлів)...\\n")

    vectorstore = Chroma(persist_directory=DB_DIR,
embedding_function=embeddings)

else:

    self.safe_update_chat("👉 Створення нової бази знань з PDF-файлів
(це може зайняти час)...\\n")

    loader = DirectoryLoader(BASE_DIR, glob="*.pdf",
loader_cls=PyPDFLoader)

    docs = loader.load()

    text_splitter = RecursiveCharacterTextSplitter(chunk_size=700,
chunk_overlap=150)

    splits = text_splitter.split_documents(docs)

    vectorstore = Chroma.from_documents(documents=splits,
embedding=embeddings, persist_directory=DB_DIR)

    retriever = vectorstore.as_retriever(search_kwargs={"k": 2})

    template = """ДІЙ ЯК ЕКСПЕРТ. Використовуй ТІЛЬКИ цей текст:
{context}\\nВідповідай УКРАЇНСЬКОЮ. Стило.\\nПитання:
{question}\\nВідповідь: """

    prompt = ChatPromptTemplate.from_template(template)

    rag_chain = ({ "context": retriever, "question": RunnablePassthrough() } |
prompt | llm | StrOutputParser())

    self.safe_update_chat("✅ База знань успішно завантажена! Асистент
готовий до роботи.\\n\\n")

    self.after(0, self.enable_interface)

except Exception as e:

    self.safe_update_chat(f"❌ Помилка ініціалізації системи: {str(e)}\\n\\n")

    self.after(0, self.progress.stop)

def enable_interface(self):

    self.progress.stop()

```

```

self.progress.set(0)
self.entry.configure(state=tk.NORMAL, placeholder_text="Вставте питання тут...")
self.send_btn.configure(state=tk.NORMAL)
self.clear_btn.configure(state=tk.NORMAL)
self.entry.focus_set()
self.entry.bind("<Return>", lambda e: self.send_message())
self.entry._entry.bind("<Control-KeyPress>", self.check_ctrl_v)
def open_log_file(self):
    try:
        if os.path.exists(LOG_FILE):
            os.startfile(LOG_FILE)
        else:
            self.safe_update_chat(" ⚠ Файл логів ще не створено.\n\n")
    except Exception as e:
        print(f"Помилка відкриття файлу: {e}")
def check_ctrl_v(self, event):
    if event.keycode == 86 or event.keysym.lower() == 'v':
        self.custom_paste()
        return "break"
def custom_paste(self):
    try:
        text = self.clipboard_get()
        if text:
            clean_text = " ".join(text.split())
            if self.entry.get() and self.entry._entry.selection_present():
                self.entry._entry.delete(tk.SEL_FIRST, tk.SEL_LAST)
                self.entry.insert(tk.INSERT, clean_text)
    except Exception: pass

```

```

    return "break"

def clear_chat(self):
    self.chat_history.config(state=tk.NORMAL)
    self.chat_history.delete(1.0, tk.END)
    self.chat_history.config(state=tk.DISABLED)

def cancel_question(self):
    self.stop_flag = True

def safe_update_chat(self, text):
    self.after(0, lambda: self._update_chat_ui(text))

def _update_chat_ui(self, text):
    self.chat_history.config(state=tk.NORMAL)
    self.chat_history.insert(tk.END, text)
    self.chat_history.see(tk.END)
    self.chat_history.config(state=tk.DISABLED)

def send_message(self):
    query = self.entry.get().strip()
    if not query or not rag_chain:
        return
    self.entry.delete(0, tk.END)
    self.stop_flag = False
    self.send_btn.configure(state=tk.DISABLED)
    self.entry.configure(state=tk.DISABLED)
    self.cancel_btn.configure(state=tk.NORMAL)
    self.progress.start()

    # Початок відліку таймера та фіксація часу запиту (ДД-ММ-РРРР)
    now = datetime.datetime.now()
    date_str = now.strftime("%d-%m-%Y")
    time_request = now.strftime("%H:%M:%S")

```

```

start_time = time.time()

self.safe_update_chat(f"👤 Ви [{date_str} {time_request}]: {query}\n")
# Передаємо start_time як четвертий аргумент у потік
threading.Thread(
    target=self.async_query_rag,
    args=(query, date_str, time_request, start_time),
    daemon=True
).start()

def async_query_rag(self, query, date_str, time_request, start_time):
    global rag_chain
    try:
        if self.stop_flag:
            response_text = "⊘ Запит скасовано користувачем."
        else:
            response_text = rag_chain.invoke(query)
        if self.stop_flag:
            response_text = "⊘ Запит скасовано користувачем."
        # Фіксація часу завершення та пройденого часу генерації
        now_resp = datetime.datetime.now()
        date_resp_str = now_resp.strftime("%d-%m-%Y")
        time_response = now_resp.strftime("%H:%M:%S")
        elapsed_time = round(time.time() - start_time, 2)
        # Виведення великого тексту відповіді із точним часом генерації
        self.safe_update_chat(f"🤖 Асистент [{date_resp_str} {time_response}]
(Час генерації: {elapsed_time} сек):\n{response_text}\n\n")
        self.log_to_file(query, response_text, date_str, time_request, date_resp_str,
time_response, elapsed_time)
    except Exception as e:

```

```

        self.safe_update_chat(f'✘ Помилка генерації: {str(e)}\n\n")
    finally:
        self.after(0, self.disable_cancel_state)
def disable_cancel_state(self):
    self.progress.stop()
    self.progress.set(0)
    self.send_btn.configure(state=tk.NORMAL)
    self.entry.configure(state=tk.NORMAL)
    self.cancel_btn.configure(state=tk.DISABLED)
    self.entry.focus_set()
def log_to_file(self, question, answer, date_req, time_req, date_resp, time_resp,
elapsed):
    try:
        with open(LOG_FILE, "a", encoding="utf-8") as f:
            f.write(f"==== Запит від: {date_req} ==== \n")
            f.write(f"Час запиту: {time_req} \n")
            f.write(f"Питання: {question} \n")
            f.write(f"Дата відповіді: {date_resp} \n")
            f.write(f"Час відповіді: {time_resp} \n")
            f.write(f"Тривалість обробки: {elapsed} сек \n")
            f.write(f"Відповідь: {answer} \n")
            f.write("-" * 40 + "\n\n")
    except Exception as e:
        print(f"Помилка запису логу: {e}")
# 3. ЗАПУСК ДОДАТКУ
if __name__ == "__main__":
    app = ChatApp()
    app.mainloop()

```

Б.2 Фрагмент Log файлу Агента ОШІ

==== Запит від: 2026-06-01 ====

Час запиту: 14:47:52

Питання: що таке база знань?

Дата відповіді: 2026-06-01

Час відповіді: 14:51:03

Відповідь База знань - це структуралізована колекція даних, які описують об'єкти, їхні ознаки та полярні значення, що дозволяє здійснювати передбачення, класифікацію та інші задачі в області штучного інтелекту.

==== Запит від: 2026-06-01 ====

Час запиту: 14:52:39

Питання: Яка можлива структура інтелектуальної системи?

Дата відповіді: 2026-06-01

Час відповіді: 14:56:01

Відповідь: Є три різноманітні підходи для структуривання інтеліктних систем, які розпізнають та навчаються. Вони включають:

1. Структурно-евристичний підхід: Здебільшого підход, який ґрунтується на знаннях, що спостерігаються в поведінці об'єкта та розгляданих структур (і їх властивостей) мозку.

2. Багатоканальне розпізнавання складних зорових образів: У цьому підході використовується паралельний розгляд зорових образів, що може бути розпізнано системою.

3. Умова навчання умовним рефлексам: В цьому випадку система навчає умовними рефлексом, що можуть бути розміщеними в різноманітних місцях.

Також, система може працювати в трьох основних режимах: навчання з контролем (або \"учителем\"), самонавчання та розпізнавання.

Б.3 Інтерфейс Агента

AI Асистент ОШІ

Системне повідомлення: Завантаження бази знань та моделей AI... Будь ласка, зачекайте.
Знайдено готову базу. Швидке завантаження (9 файлів)...

База знань успішно завантажена! Асистент готовий до роботи.

Ви [01-06-2026 15:15:31]: властивості агентних систем
Асистент [01-06-2026 15:19:08] (Час генерації: 217.77 сек):
Як експерт у галузі агентних систем, я можу стверджувати, що дві основні форми агентних систем - це несамовиявлений колектив (без зв'язку між агентами) та самовиявлений колектив (з зв'язком між агентами).

Властивості агентних систем:

- * Автономність: кожен агент діє самостійно, не залежно від інших.
- * Діяльність: агенти можуть виконувати різні дії у відповідь на зміни в середовищі.
- * Оцінювання: агенти оцінюють результати своїх дій та змін у середовищі.
- * Взаємодія: агенти можуть взаємодіяти між собою, обмінюючи інформацію заданого змісту та формату.

Також, агентні системи можуть мати різні цільові функції, такі як формування колективної дії або максимізація значення функції.

Ви [01-06-2026 15:49:59]: Зроби тест з 5 питань по останній лекції
Асистент [01-06-2026 15:54:47] (Час генерації: 287.52 сек):
Як експерт у галузі штучного інтелекту, я створив тест на основі останньої лекції.

Тест складається з 5 питань:

1. Що таке нечіткі множини?
 - a) Множина з точними значеннями
 - b) Множина з невизначеними значеннями
 - c) Множина з дискретними значеннями

Віповідь: b) Множина з невизначеними значеннями

Історія запитів: C:\D_2\AI_25-26\MY_AI_pro\answers.txt (натисніть, щоб відкрити)

Запитати Скасувати Очистити Вихід