ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

ЧАЙКОВСЬКИЙ МАКСИМ ЮРІЙОВИЧ

УДК 004.75:004.49:004.3

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ВИЯВЛЕННЯ ПОЛІМОРФНИХ ВІРУСІВ ЗА ДОПОМОГОЮ ГІБРИДНОЇ МУЛЬТИАГЕНТНОЇ СИСТЕМИ

123 Комп'ютерна інженерія
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ М.Ю. Чайковський
(підпис)

Науковий керівник: Савенко Олег Станіславович, доктор технічних наук, професор

Хмельницький – 2025

АНОТАЦІЯ

Чайковський М. Ю. Методи та засоби виявлення поліморфних вірусів за допомогою гібридної мультиагентної системи. - Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії з галузі знань 12 Інформаційні технології за спеціальністю 123 Комп'ютерна інженерія. – Хмельницький національний університет, Хмельницький. 2025.

На сьогоднішній день досить гострою є проблема виявлення зловмисного програмного забезпечення, а особливо поліморфних вірусів, які завдають значної шкоди комп'ютерним системам, а також постійно вдосконалюються. Тому, створення гібридної мультиагентної системи для виявлення поліморфних вірусів є актуальною науково-практичною задачею.

У дисертації здійснено аналіз поліморфних вірусів за рівнями складності, методів виявлення поліморфних вірусів, проведено аналіз мультиагентних систем виявлення зловмисного програмного забезпечення. В роботі розроблено архітектуру гібридної мультиагентної системи виявлення поліморфних вірусів, яка здійснює аналіз середовища, досліджує темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення; розроблено метод виявлення поліморфних вірусів, який дозволяє визначати ймовірність їх належності до класів поліморфних вірусів із використанням алгоритмів пошуку рядка, інтелектуального аналізу даних, аналізу в пісочниці, машинного навчання, методу розробки структурних функцій та ймовірнісних логічних мереж; удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який дає змогу досліджувати вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі; удосконалено метод нечіткої класифікації поліморфних вірусів, який дозволяє не лише класифікувати поліморфні віруси за рівнями їх складності, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності.

Об'єкт дослідження – процес виявлення поліморфних вірусів у комп'ютерних системах.

Предмет дослідження – моделі, методи і програмні засоби мультиагентних систем для виявлення поліморфних вірусів у комп'ютерних системах.

Метою дисертаційного дослідження є покращення ефективності виявлення поліморфних вірусів у комп'ютерних системах за допомогою гібридних мультиагентних систем.

Наукова новизна отриманих результатів полягає в наступному:

1) вперше розроблено архітектуру гібридних мультиагентних систем виявлення поліморфних вірусів, яка каскадно делегуючи повноваження, здійснює аналіз середовища, встановлює темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення;

2) розроблено новий метод виявлення поліморфних вірусів, який на відміну від існуючих, дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів;

3) удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який, на відміну від існуючих, дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати;

4) удосконалено метод класифікації поліморфних вірусів, який, на відміну від існуючих, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обгрунтованості та ефективність обраних стратегій.

Практичне значення отриманих результатів. За результатами виконаних досліджень розроблено архітектуру гібридних мультиагентних систем виявлення поліморфних вірусів, здійснено реалізацію розроблених методів та гібридну мультиагентну систему, яка дає змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення. В результаті проведених експериментальних досліджень з гібридною мультиагентною системою, в основу функціонування якої покладено розроблений підхід, отримано наступні результати: при роботі з поліморфними вірусами першого рівня складності ефективність гібридної мультиагентної системи становить 98,87 %; при роботі з поліморфними вірусами першого і другого рівнів складності – 98,15 %; перших трьох рівнів складності – 97,45 %, перших чотирьох – 96,34 %, перших п’яти – 95,94 % та підтверджують ефективність запропонованого рішення.

У результаті проведених експериментальних досліджень з розробленою системою було підтверджене коректне функціонування гібридної мультиагентної системи, можливість застосування її до виявлення поліморфних вірусів.

Теоретичні та практичні результати дослідження впроваджені в в ТОВ «ІТТ» (м. Хмельницький), ПП «НОЛТ ТЕХНОЛОДЖИС» (м. Хмельницький), а також, в освітньому процесі Хмельницького національного університету при викладанні дисциплін на кафедрі комп’ютерної інженерії та інформаційних систем для спеціальності 123 Комп’ютерна інженерія, зокрема в курсах «Теорія і проєктування комп'ютерних та кіберфізичних систем і мереж», «Теорія і технології проєктування

спеціалізованих операційних систем», «Методи розв’язування наукових задач комп’ютерної інженерії».

У вступі представлено обгрунтування актуальності наукової задачі створення мультиагентної системи для виявлення поліморфних вірусів. Перспективним напрямом досліджень визначено гібридну мультиагентну систему, яка міститиме інтелектуальні агенти, які дозволять здійснити аналіз середовища, встановити темпи поширення, виявити, класифікувати поліморфні віруси та, в результаті, сформує оптимальні стратегії прийняття рішення, враховуючи типи загроз. Також, представлено зв’язок тематики дослідження з напрямками наукових досліджень дослідників цієї проблеми в світі та подано основні наукові результати роботи, її практичне значення, перелік підприємств та установ, в яких впроваджено результати роботи.

У першому розділі проаналізовано поняття, типи поліморфних вірусів згідно рівнів складності, проаналізовані методи їх виявлення, програмні засоби діагностування комп’ютерних систем на наявність поліморфного зловмисного програмного забезпечення, проаналізовано існуючі мультиагентні системи виявлення зловмисного програмного забезпечення. Також, підведено підсумки проведеного аналізу та здійснено постановку задачі дослідження.

У другому розділі побудовано моделі класів поліморфних вірусів різних рівнів складності їх будови, які враховують, що віруси першого рівня поліморфізму використовують постійні значення для різних розшифровувачів; другого рівня - віруси, розшифровувачі яких мають постійну одну або декілька інструкцій; третього - віруси, розшифровувачі яких мають в розшифровувачі команди, які не приймають участі в розшифруванні вірусного коду, чи «команд-сміття»; четвертого - використовують в розшифровувачі взаємозамінювані інструкції без зміни алгоритму розшифрування; п’ятого - використовують високорозвинені методи маскування та зміни свого коду; шостого рівня - нешифровані віруси, що складаються з програмних частин, які «перемішуються» всередині тіла вірусу. Також запропонована нечітка класифікація поліморфних вірусів за рівнями складності із використанням нечіткого логічного висновку. Розроблена архітектура гібридної мультиагентної системи

виявлення поліморфних вірусів та модель інтелектуального агента в її структурі. Розроблена архітектура гібридної мультиагентної системи є основою для розроблення нових систем із підвищеною ефективністю виявлення поліморфних вірусів.

У третьому розділі розроблено метод виявлення поліморфних вірусів, який дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому (у різному порядку застосування) – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів. Удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати. Удосконалено метод класифікації поліморфних вірусів, який дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обгрунтованості та ефективність обраних стратегій.

У четвертому розділі розроблено гібридну мультиагентну систему виявлення поліморфних вірусів, запропоновано методику визначення ефективності її функціонування, опис експериментального середовища, оцінювання ефективності функціонування системи, а також підведено підсумки з отриманих результатів.

У висновках представлено отримані наукові та практичні результати дослідження.

У додатках представлено наукові публікації, в яких відображено наукові результати роботи, акти впровадження результатів роботи, лістинг програмного забезпечення.

Ключові слова: комп'ютерні системи, комп'ютерні мережі, поліморфні віруси, методи виявлення поліморфних вірусів, інтелектуальний агент, мультиагентна система, нечітка класифікація.

ANNOTATION

Chaikovskiy M. Yu. Methods and means of detecting polymorphic viruses using a hybrid multi-agent system. - Qualification scientific work in the form of a manuscript.

Dissertation for the degree of Doctor of Philosophy in the field of knowledge 12 Information Technologies in the specialty 123 Computer Engineering. – Khmelnytskyi National University, Khmelnytskyi. 2025.

Today, the problem of detecting malicious software, especially polymorphic viruses, which cause significant damage to computer systems, and are also constantly being improved, is quite acute. Therefore, the creation of a hybrid multi-agent system for detecting polymorphic viruses is an urgent scientific and practical task.

The dissertation analyzes polymorphic viruses by levels of complexity, methods for detecting polymorphic viruses, and analyzes multi-agent systems for detecting malicious software. The work develops the architecture of a hybrid multi-agent system for detecting polymorphic viruses, which analyzes the environment, investigates the rates of spread, detects, classifies polymorphic viruses and uses various decision-making strategies; a method for detecting polymorphic viruses has been developed, which allows determining the probability of their belonging to polymorphic virus classes using string search algorithms, data mining, sandbox analysis, machine learning, the method for developing structural functions and probabilistic logical networks; a method for determining the rate of spread of polymorphic viruses based on the Lotka-Volterra model has been improved, which allows investigating the influence of the number of methods used to detect polymorphic viruses on the rate of their spread in an oscillatory process; a method for fuzzy classification of polymorphic viruses has been improved, which allows not only to classify polymorphic viruses by their complexity levels, but also to determine the probability of their belonging

to fuzzy terms at the levels of low, below average, average, above average, high for each complexity level.

The object of the study is the process of detecting polymorphic viruses in computer systems.

The subject of the study is models, methods and software tools of multi-agent systems for detecting polymorphic viruses in computer systems.

The purpose of the dissertation research is to improve the efficiency of detecting polymorphic viruses in computer systems using hybrid multi-agent systems.

The scientific novelty of the obtained results is as follows:

1) for the first time, an architecture of hybrid multi-agent systems for detecting polymorphic viruses has been developed, which, by cascading delegation of authority, analyzes the environment, establishes the rate of spread, detects, classifies polymorphic viruses and uses various decision-making strategies, taking into account the types of threats, which made it possible to determine and use the optimal set of methods for detecting polymorphic viruses, as well as to classify polymorphic viruses by levels of complexity, which increases the level of effectiveness of the selected decision-making strategies;

2) a new method for detecting polymorphic viruses has been developed, which, unlike existing ones, allows not only to increase the efficiency of detecting polymorphic viruses, but also to determine the probability of belonging of detected viruses to the class of polymorphic ones and provides for a three-stage combined application, of which, at the first stage, string search algorithms are used, at the second - intelligent data analysis, sandbox analysis, machine learning, a method for developing structural functions, at the third - probabilistic logical networks, which made it possible to classify detected threats according to the level of probability of their belonging to polymorphic viruses;

3) a method for determining the rate of spread of polymorphic viruses has been improved based on the use of the Lotka-Volterra model, which, unlike existing ones, allows investigating the influence of the number of used methods for detecting polymorphic viruses on the rate of their spread in an oscillatory process and allows determining the number of methods for detecting polymorphic viruses that need to be used;

4) the method for classifying polymorphic viruses has been improved, which, unlike existing ones, allows not only to classify polymorphic viruses by the levels of complexity of their structure, but also to determine the probability of their belonging to fuzzy terms at the levels of low, below average, average, above average, high for each level of complexity, which made it possible to increase the level of validity and effectiveness of the selected strategies.

Practical significance of the results obtained. Based on the results of the research, the architecture of hybrid multiagent systems for detecting polymorphic viruses was developed, the developed methods were implemented, and a hybrid multiagent system was developed, which allows determining and using the optimal set of methods for detecting polymorphic viruses, as well as classifying polymorphic viruses by complexity levels, which increases the level of effectiveness of the selected decision-making strategies. As a result of experimental studies with a hybrid multiagent system, the functioning of which is based on the developed approach, the following results were obtained: when working with polymorphic viruses of the first level of complexity, the efficiency of the hybrid multiagent system is 98.87%; when working with polymorphic viruses of the first and second levels of complexity - 98.15%; the first three levels of complexity - 97.45%, the first four - 96.34%, the first five - 95.94%, and confirm the effectiveness of the proposed solution.

As a result of experimental studies conducted with the developed system, the correct functioning of the hybrid multi-agent system and the possibility of its application to the detection of polymorphic viruses were confirmed.

The theoretical and practical results of the study are implemented in ITT LLC (Khmelnytskyi), PE "NOLT TECHNOLOGY" (Khmelnytskyi) and also in the educational process of Khmelnytskyi National University when teaching subjects at the Department of Computer Engineering and Information Systems for specialties 123 Computer Engineering, in particular in the courses «Theory and Design of Computer and Cyber-Physical Systems and Networks», «Theory and Technologies of Design of Specialized Operating Systems», «Methods of Solving Scientific Problems in Computer Engineering».

The introduction presents the justification of the relevance of the scientific task of creating a multi-agent system for detecting polymorphic viruses. A promising direction of

research is a hybrid multi-agent system, which will contain intelligent agents that will allow analyzing the environment, determining the rate of spread, detecting, classifying polymorphic viruses and, as a result, forming optimal decision-making strategies, taking into account the types of threats. Also, the connection of the research topic with the directions of scientific research of researchers of this problem in the world is presented and the main scientific results of the work, its practical significance, a list of enterprises and institutions in which the results of the work have been implemented are presented.

The first section analyzes the concept, types of polymorphic viruses according to levels of complexity, analyzes methods for their detection, software tools for diagnosing computer systems for the presence of polymorphic malicious software, analyzes existing multi-agent systems for detecting malicious software. Also, the results of the analysis are summarized and the research task is formulated.

In the second section, models of classes of polymorphic viruses of different levels of complexity of their structure are built, which take into account that viruses of the first level of polymorphism use constant values for different decryptors; viruses of the second level, whose decryptors have one or more constant instructions; viruses of the third level, whose decryptors have commands in the decryptor that do not participate in decrypting the virus code, or “garbage commands”; viruses of the fourth level, whose decryptors use interchangeable instructions without changing the decryption algorithm; viruses of the fifth level, whose decryptors use highly developed methods of masking and changing their code; viruses of the sixth level, whose decryptors consist of program parts that are “mixed” inside the virus body. A fuzzy classification of polymorphic viruses by levels of complexity using fuzzy logical inference is also proposed. The architecture of a hybrid multi-agent system for detecting polymorphic viruses and a model of an intelligent agent in its structure are developed. The developed architecture of the hybrid multi-agent system is the basis for the development of new systems with increased efficiency of polymorphic virus detection.

In the third section, a method for detecting polymorphic viruses is developed, which allows not only to increase the efficiency of detecting polymorphic viruses, but also to determine the probability of belonging of the detected viruses to the class of polymorphic and provides for a three-stage combined application, of which, at the first stage, string search

algorithms are used, at the second (in different orders according to the situation) - data mining, sandbox analysis, machine learning, a method for developing structural functions, at the third - probabilistic logical networks, which made it possible to classify the detected threats according to the level of probability of their belonging to polymorphic viruses. The method for determining the rate of spread of polymorphic viruses based on the Lotka-Volterra model is improved, which allows us to investigate the influence of the number of methods used to detect polymorphic viruses on the rate of their spread in the oscillatory process and allows us to determine how many methods for detecting polymorphic viruses need to be used. The method for classifying polymorphic viruses has been improved, which allows not only to classify polymorphic viruses by the levels of complexity of their structure, but also to determine the probability of their belonging to fuzzy terms at the level of low, below average, average, above average, high for each level of complexity, which made it possible to increase the level of validity and effectiveness of the selected strategies for their neutralization.

In the fourth section, a hybrid multi-agent system for detecting polymorphic viruses is developed, a methodology for determining the effectiveness of its functioning is proposed, a description of the experimental environment, an assessment of the effectiveness of the system's functioning are proposed, and the results obtained are summarized.

The conclusions present the obtained scientific and practical results of the study.

The appendices present scientific publications that reflect the scientific results of the work, acts of implementation of the results of the work, and a software listing.

Keywords: computer systems, computer networks, polymorphic viruses, methods for detecting polymorphic viruses, intelligent agent, multi-agent system, fuzzy classification.

Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Чайковський М. Комплексний підхід до виявлення та аналізу поліморфного зловмисного програмного забезпечення. *Measuring and Computing Devices in*

Technological Processes. 2024. № 2. С. 42-50. DOI: https://doi.org/10.31891/2219-9365-2024-78-5

2. Савенко О., Чайковський М. Метод нечіткої класифікації зловмисного програмного забезпечення з використанням інтелектуального агента. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2024. № 3. С. 140-148. DOI: https://doi.org/10.32782/IT/2024-3-15

3. Чайковський М. Модель мультиагентної системи для виявлення поліморфних вірусів. *Herald of Khmelnytskyi National University. Technical Sciences*. 2024. № 347(1). С. 543-547. DOI: https://doi.org/10.31891/2307-5732-2025-347-74

4. Чайковський М. Архітектура та засоби мультиагентної системи виявлення поліморфних вірусів в комп'ютерних мережах. *Measuring and Computing Devices in Technological Processes.* 2025. № 1. С. 278–286. DOI: https://doi.org/10.31891/2219-9365-2025-81-34

Праці, які засвідчують апробацію матеріалів дисертації

5. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Comprehensive approach to the detection and analysis of polymorphic malware. *CEUR Workshop Proceedings (ISSN 1613-0073)*. 2024. Vol.3736. P.312-323. URL: https://ceur-ws.org/Vol-3736/paper23.pdf (Scopus)

6. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Modeling the detection process of polymorphic malware based on the Lotka-Volterra Model. *CEUR Workshop Proceedings (ISSN 1613-0073)*. 2025. Vol.3899. P. 244-253. URL: https://ceur-ws.org/Vol-3899/paper22.pdf (Scopus)

7. Чайковський М.Ю. Прогнозування кількості атак зловмисного програмного забезпечення у світі. *Актуальні проблеми комп'ютерних наук АПКН-2024 : матеріали XVI всеукр. наук.-практ. конф., м. Хмельницький, 15-16 лист. 2024 р.* / Хмельницький національний університет. Хмельницький, 2024. С. 534-536. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ЗПЗ	–	Зловмисне програмне забезпечення
МАС	–	Мультиагентна система
ІА	–	Інтелектуальний агент
ПЗ	–	Програмне забезпечення
DL	–	Глибоке навчання
CNN	–	Згорткові нейронні мережі
GRR	–	Платформа Google Rapid Response
PLN	–	Ймовірнісні логічні мережі
MASFMMS	–	Multi Agent Systems Framework for Malware Modeling and Simulation
LPV	–	Рівень складності поліморфного вірусу
CE	–	Рівень шифрування коду
CO	–	Рівень обфускації коду
SDC	–	Рівень використання самозмінюваного та динамічного коду
PAS	–	Рівень захисту від аналізу та обходу пісочниць
CNB	–	Рівень складності мережевої поведінки та взаємодії
ISR	–	Рівень взаємодії із системою та реєстром
ABS	–	Рівень адаптивної поведінки та самозахисту
FIS	–	Fuzzy Inference System
TPR	–	True Positive Rate

ВСТУП

Актуальність роботи. Пошук та знешкодження комп'ютерних вірусів з кожним роком стає все більш актуальною та складною проблемою, адже вони несуть загрозу безперешкодному функціонуванню комп'ютерних систем, які використовуються у все більш критичних сферах діяльності людства. Тому, розроблення нових методів та засобів знешкодження зловмисного програмного забезпечення (ЗПЗ) є одним із перспективних та пріоритетних завдань досліджень у сфері комп'ютерних наук. Незважаючи на постійне вдосконалення антивірусного програмного забезпечення, генерація та поширення ЗПЗ збільшується з року в рік. Одна з найсерйозніших проблем, з якою стикається розробники антивірусного програмного забезпечення (ПЗ) – це автоматична мутація коду зловмисної програми. Методи мутації та перестановки коду зловмисної програми називаються поліморфізмом. Поліморфні віруси неможливо ідентифікувати сигнатурним аналізом. Тому, для цього необхідно використовувати нові, удосконалені методи аналізу сучасного ЗПЗ, а також комплексне поєднання існуючих методів та підходів.

Питаннями виявлення ЗПЗ та комп'ютерних атак, а також розробленням принципів, методів, систем і засобів захисту та забезпечення безпеки комп'ютерних і кіберфізичних систем, займаються науковці: Альшамрані С. [90], Альшері А. [90], Амін Р. [90], Ахтар М. [7], Ван С. [71], Дудикевич В. [137], Євсєєв С. [131, 133], Карлсон А. [115], Карпінський М. [53], Корченко А. [133], Корченко О. [72], Лахно В. [65], Летичевський О. [64,138], Лю С.[71], Макрані Х. [114], Мілов О. [133], Мухін В. [87], Наєм М. [90], Нгуєн В. [92], Саченко А. [60, 61], Саяді Х. [114], Сінгх М. [115], Смірнов О. [64], Сохор Т. [54, 55], Сун К. [71], Сян Ю. [22], Фенг П. [71], Фенг Т. [7], Харченко В. [56, 57, 86], Хохлачова Ю. [58, 59], Чакраборті А. [26], Чезаре С. [22].

Разом з тим існуючі підходи вимагають їх удосконалення у розрізі мінімізації недоліків запропонованих методів, а також необхідності врахування рівнів складності поліморфних вірусів, тобто удосконалення методу їх класифікації. Тому необхідним є системний підхід до аналізу, виявлення, класифікації, дослідження темпів поширення

поліморфних вірусів, що лежать в основі гібридних мультиагентних систем виявлення поліморфних вірусів і є досить актуальною науково-практичною задачею.

Зазначена науково-прикладна задача відповідає предметній області Стандарту вищої освіти України зі спеціальності 123 Комп'ютерна інженерія для третього (освітньо-наукового) рівня вищої освіти, зокрема, таким об'єктам вивчення та діяльності: методи та способи подання, отримання, зберігання, передавання, опрацювання та захисту інформації в комп'ютерних системах, архітектура та організація їх функціонування, інтерфейси та протоколи взаємодії їх компонентів.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконувалось у рамках науково-дослідної тематики Хмельницького національного університету, а саме держбюджетної науково-дослідної теми №2Б-2024 «Система виявлення ЗПЗ та комп'ютерних атак в корпоративних мережах з використанням хибних об'єктів атак та пасток» (№ держреєстрації 0124U000980), в якій автор дисертації був виконавцем.

Мета і завдання дослідження.

Об'єкт дослідження – процес виявлення поліморфних вірусів у комп'ютерних системах.

Предмет дослідження – моделі, методи і програмні засоби гібридних мультиагентних систем для виявлення поліморфних вірусів у комп'ютерних системах.

Метою дисертаційного дослідження є підвищення ефективності виявлення поліморфних вірусів у комп'ютерних системах за допомогою гібридних мультиагентних систем.

Завдання дослідження формулюються в роботі наступним чином:

1) провести аналіз поліморфних вірусів за різними рівнями складності їх структури, методів та засобів їх виявлення за допомогою мультиагентних систем;

2) розробити моделі класів поліморфних вірусів згідно різних рівнів поліморфізму;

3) розробити архітектуру гібридних мультиагентних систем для виявлення, аналізу та класифікації поліморфних вірусів, а також модель інтелектуального агента в архітектурі гібридних мультиагентних систем для виявлення поліморфних вірусів;

4) удосконалити метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра для дослідження впливу кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі;

5) розробити метод виявлення поліморфних вірусів для покращення ефективності виявлення поліморфних вірусів та визначення ймовірності належності виявлених вірусів до класу поліморфних;

6) удосконалити метод класифікації поліморфних вірусів, який дозволить не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначатиме ймовірність їх належності до нечітких термів кожного рівня складності;

7) розробити гібридну мультиагентну систему виявлення поліморфних вірусів, яка здійснюватиме аналіз середовища, встановлюватиме темпи поширення, виявлятиме, класифікуватиме поліморфні віруси та використовуватиме різні стратегії прийняття рішення і провести з нею експериментальні дослідження щодо встановлення ефективності функціонування та достовірності виявлення поліморфних вірусів і впровадити її у виробництво.

Методи дослідження. Для розв’язання поставлених задач використовуються:

1) для виявлення поліморфних вірусів - комплекс методів у їх системному поєднанні: алгоритм пошуку рядка, інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, ймовірнісні логічні мережі;

2) для встановлення темпу поширення поліморфних вірусів – модель Лотки-Вольтерра;

3) для класифікації поліморфних вірусів – нечітка логіка (нечіткий логічний висновок Мамдані).

Наукова новизна одержаних результатів полягає в наступному:

1) вперше розроблено архітектуру гібридних мультиагентних систем виявлення поліморфних вірусів, яка каскадно делегуючи повноваження, здійснює аналіз середовища, встановлює темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення

поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення;

2) розроблено новий метод виявлення поліморфних вірусів, який на відміну від існуючих, дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів;

3) удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який, на відміну від існуючих, дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати;

4) удосконалено метод класифікації поліморфних вірусів, який, на відміну від існуючих, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обгрунтованості та ефективність обраних стратегій.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій. Наукові положення, висновки і рекомендації дисертаційної роботи забезпечуються коректністю поставленої наукової задачі і виконанням досліджень з використанням математичного апарату, програмною реалізацією розробленої гібридної мультиагентної системи виявлення поліморфних вірусів, алгоритмами здійснення діагностування, ефективним практичним впровадженням результатів дисертаційних досліджень на підприємствах та організаціях, яке продемонструвало відповідність теоретичних досліджень із реальними результатами.

Практичне значення отриманих результатів. За результатами виконаних досліджень розроблено архітектуру гібридних мультиагентних систем виявлення поліморфних вірусів, здійснено реалізацію розроблених методів та гібридну мультиагентну систему, яка дає змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення. В результаті проведених експериментальних досліджень з гібридною мультиагентною системою, в основу функціонування якої покладено розроблений підхід, отримано наступні результати: при роботі з поліморфними вірусами першого рівня складності ефективність мультиагентної системи становить 98,87 %; при роботі з поліморфними вірусами першого і другого рівнів складності – 98,15 %; перших трьох рівнів складності – 97,45 %, перших чотирьох – 96,34 %, перших п’яти – 95,94 % та підтверджують ефективність запропонованого рішення.

У результаті проведених експериментальних досліджень з розробленою системою було підтверджене коректне функціонування гібридної мультиагентної системи, можливість застосування її до виявлення поліморфних вірусів.

Теоретичні та практичні результати дослідження впроваджені в ТОВ «ІТТ» (м. Хмельницький), ПП «НОЛТ ТЕХНОЛОДЖИС» (м. Хмельницький), а також, в освітньому процесі Хмельницького національного університету при викладанні дисциплін на кафедрі комп’ютерної інженерії та інформаційних систем для спеціальності 123 Комп’ютерна інженерія, зокрема в курсах «Теорія і проєктування комп'ютерних та кіберфізичних систем і мереж», «Теорія і технології проєктування спеціалізованих операційних систем», «Методи розв’язування наукових задач комп’ютерної інженерії».

Особистий внесок здобувача. Всі основні результати дисертаційного дослідження, які представлені до захисту, одержані автором особисто. В роботах, опублікованих одноосібно автором, отримано наступні результати: [153] - розроблено метод виявлення поліморфних вірусів, який передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка,

на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів, а також максимізувати ймовірність успішного виявлення поліморфних вірусів; [154, 152] - розроблено архітектуру гібридної мультиагентної системи виявлення поліморфних вірусів, яка здійснює аналіз середовища, встановлює темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз; [155] – здійснено прогнозування кількості атак зловмисного програмного забезпечення.

У роботах, які опубліковані у співавторстві, автору належать основні ідеї, теоретична та практична розробка положень, відображених у характеристиці наукової новизни отриманих результатів, а саме: [151] - удосконалено метод класифікації поліморфних вірусів, який дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів кожного рівня складності; [23] – розроблено комплексний підхід до виявлення та аналізу поліморфних вірусів; [24] – удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра.

У роботах, які опубліковані у співавторстві, співавторам належать такі результати: [151] – концепція класифікації поліморфних вірусів за рівнями складності їх будови (Савенко О. С.); [23] – проведення аналізу поліморфних вірусів за складністю їх будови (Martyniuk I., Lyhun O.), застосування елементів з технологій приманок для активізації поліморфних вірусів (Sochor T.), опрацювання результатів експерименту (Chaikovska I.); [24] – підготовка моделі Лотки-Вольтерра для здійснення моделювання (Chaikovska I., Sochor T.), вибір поліморфних вірусів за складністю їх будови як об’єктів дослідження на моделі Лотки-Вольтерра (Martyniuk I., Lyhun O.).

Апробація результатів дисертації. Апробацію основних положень, ідей, висновків дисертаційної роботи проведено на науковому семінарі кафедри комп’ютерної інженерії та інформаційних систем у Хмельницькому національному університеті. Наукові результати роботи доповідалися на таких конференціях: 1st

International Workshop on Advanced Applied Information Technologies (AdvAIT-2024) (Khmelnytskyi, Ukraine - Zilina, Slovakia, December 5, 2024); XVI Всеукраїнська науково-практична конференція «Актуальні проблеми комп'ютерних наук АПКН-2024» (Хмельницький, Україна, 15-16 листопада 2024); 1st International Workshop on Intelligent & CyberPhysical Systems (ICyberPhyS-2024) (Khmelnytskyi, Ukraine, June 28, 2024).

Публікації. За результатами проведених досліджень основні наукові результати опубліковано у 4 наукових статтях у трьох фахових наукових журналах України [151, 152, 153, 154]. Апробація засвідчена публікаціями 3 праць в матеріалах міжнародних та всеукраїнських конференцій [23, 24, 155], з яких 2 праці проіндексовані у наукометричній базі Scopus [23, 24].

Структура та обсяг дисертації. Дисертаційна робота складається з анотації, змісту, переліку умовних скорочень, вступу, чотирьох розділів, висновку, списку використаних джерел та додатків. Повний обсяг роботи містить 211 сторінок друкованого тексту, з них анотація – на 11 стор., зміст – на 2 стор., основний текст – на 144 стор., список із 155 використаних джерел – на 17 стор., додатки – на 35 стор. Дисертація містить 35 рисунків та 36 таблиць.

РОЗДІЛ 1.
АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ВИЯВЛЕННЯ ПОЛІМОРФНИХ ВІРУСІВ ЗА ДОПОМОГОЮ МУЛЬТИАГЕНТНИХ СИСТЕМ

1.1. Поняття поліморфних вірусів та системи їх виявлення

Світ кібербезпеки, що постійно розвивається – це постійна боротьба між кіберзлочинцями та професіоналами з безпеки. Про важливість та актуальність проблеми захисту від зловмисного програмного забезпечення (ЗПЗ) свідчать і статистичні дані. На рис. 1 зображено діаграму щодо кількості атак ЗПЗ в усьому світі (за інформацією статистичної компанії Statista [119]).

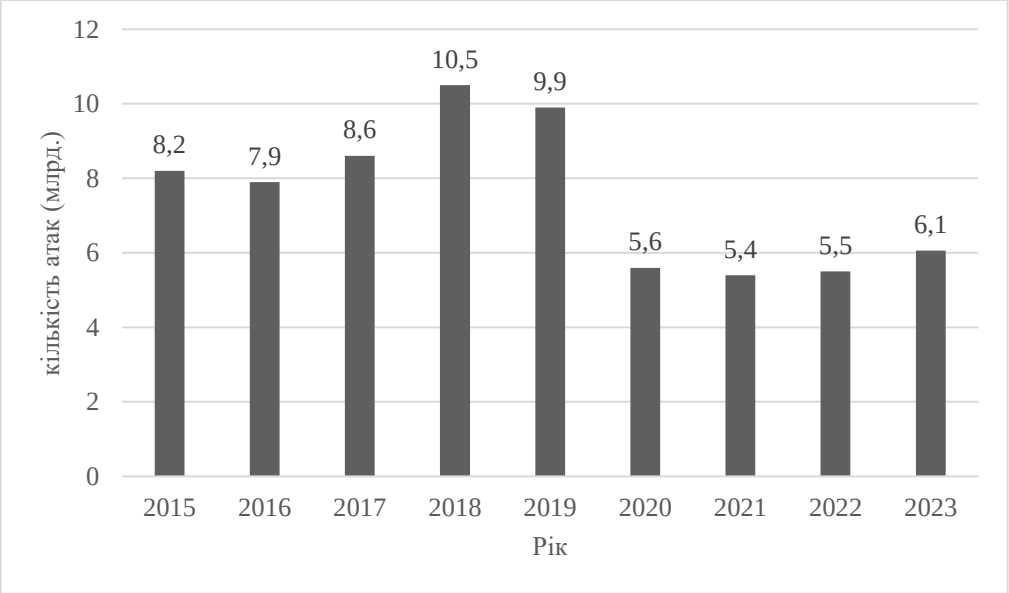

Рисунок 1.1 – Щорічна кількість атак ЗПЗ з 2015 по 2023 рік (у мільярдах) [119]

З метою відображення подальших тенденцій стосовно кількості атак ЗПЗ, даний показник було спрогнозовано на 2024-2025 роки. Для цього було побудовані лінії тренду (табл. 1.1). В якості критерію обрання моделі для прогнозування виступив коефіцієнт детермінації. Обрана поліноміальна лінія тренду (3) для прогнозування, оскільки коефіцієнт детермінації (R^2) для неї максимальний (рис. 1.2).

Таблиця 1.1

Лінії тренду кількості атак ЗПЗ

Назва моделі	Модель	R^2
Лінійна лінія тренду	$y = -0,451x + 9,7728$	0,3974
Експонентна лінія тренду	$y = 10,056e^{-0,064x}$	0,3659
Логарифмічна лінія тренду	$y = -1,337\ln(x) + 9,4197$	0,2409
Степенева лінія тренду	$y = 9,6613x^{-0,198}$	0,2076
Поліноміальна лінія тренду (2)	$y = -0,0981x^2 + 0,5297x + 7,9748$	0,4939
Поліноміальна лінія тренду (3)	$y = 0,0611x^3 - 1,014x^2 + 4,3888x + 3,9448$	0,6669

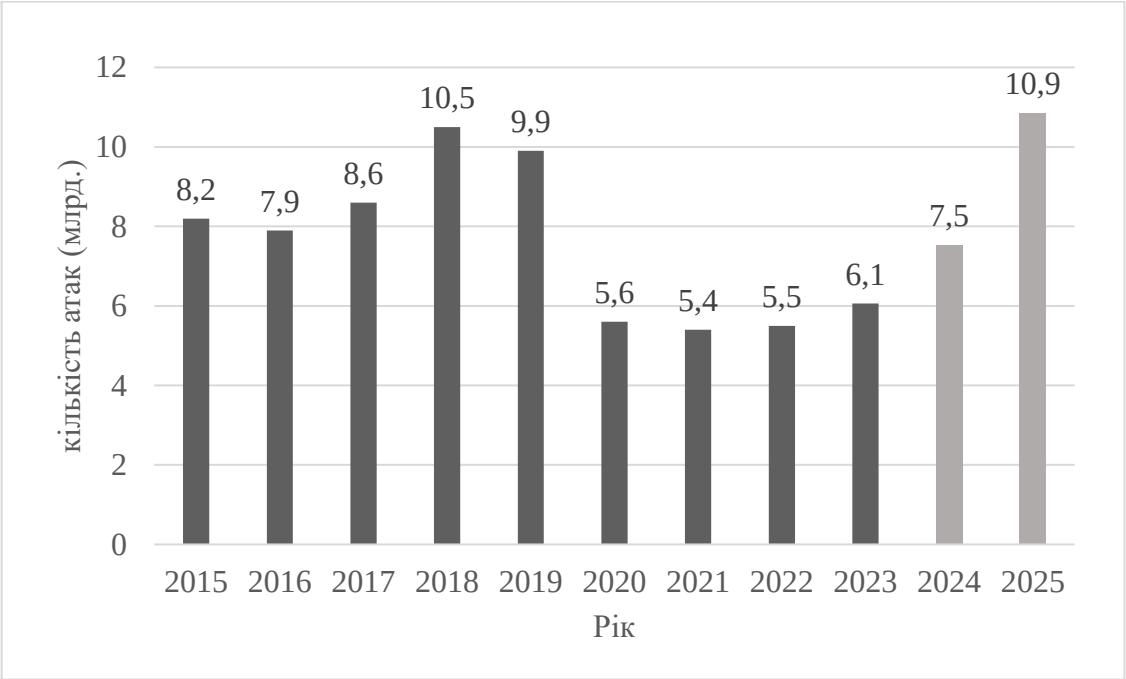

Рисунок 1.2 – Прогнозована кількість атак ЗПЗ на 2024-2025 роки

Встановлено, що у 2024-2025 роках спостерігатиметься значне збільшення кількості атак ЗПЗ та даний показник становитиме 7,5 та 10,9 мільярдів відповідно, що вимагає удосконалення методів виявлення та знешкодження ЗПЗ.

Поліморфні віруси є одним із найдосконаліших типів загроз, які є досить складними для виявлення та аналізу. Ці віруси можуть змінювати свій код або зовнішній вигляд із кожною новою інфекцією, що ускладнює розпізнавання антивірусним програмним забезпеченням (ПЗ) на основі статичної сигнатури. Ці

віруси зазвичай мають механізм мутації, що заснований на методах обфускації коду, упаковки та метаморфізму, які можуть шифрувати або дешифрувати код вірусу, щоразу створюючи унікальний програмний код [21, 142, 149]. Поліморфні віруси використовують кілька адаптивних стратегій, щоб гарантувати, що вони не будуть виявлені та нейтралізовані. Однією з найпоширеніших стратегій є шифрування коду за допомогою унікальних алгоритмів шифрування [115]. Це шифрування ускладнює виявлення вірусу антивірусним ПЗ, оскільки він виглядає як неінфікований файл. Також, можуть використовуватися процедури запобігання зворотній інженерії [11].

Отже, поліморфне ЗПЗ використовує методи, щоб уникнути виявлення. Наприклад, методи обфускації коду. Для цього використовуються алгоритми шифрування, стиснення тощо. Поліморфне ЗПЗ може приховати свою справжню природу від ПЗ безпеки та використовувати різні ключі шифрування для кожного нового екземпляру, що ускладнює ідентифікацію ЗПЗ на основі фіксованого шаблону для засобів виявлення на основі сигнатур. Поліморфне ЗПЗ може здійснювати зміну структури коду. Тоді, воно може заплутати засоби безпеки комп'ютерних систем, які покладаються на статичні підписи для виявлення. Поліморфне ЗПЗ може змінювати свою поведінку або шаблони виконання, щоб злитися зі звичайними системними процесами, що ускладнює виявлення загрози методами поведінкового виявлення.

Поліморфний вірус відрізняється від звичайного вірусу тим, як він маскується. Програмний код поліморфної зловмисної програми змінюється з кожним новим зараженням за допомогою шифрування. Існує стільки інфекцій, скільки існує варіацій одного вірусу [50]. Але кожна модифікація, по суті, є новою копією вірусу. Для шифрування в поліморфних вірусах можна використовувати складні криптографічні алгоритми. Тому, наприклад, антивірусний захист на основі баз сигнатур [45] не є ефективним засобом проти поліморфних вірусів. Щоб «лікувати» ПК від таких поліморфних вірусів, необхідна повна розшифровка їхнього «тіла» [100, 124].

Поліморфні віруси зазвичай класифікують за рівнями поліморфізму [92]. Таких рівнів в класифікації з роботи [92] введено шість. Найпростіші - олігоморфні віруси, з однаковими ділянками коду, які можна ідентифікувати за допомогою баз сигнатур. Найбільш складні віруси використовують код перестановки: вони постійно

змінюються на рівні підпрограм інсталятора, шифрувальника, обробника переривань тощо. Тому, класифікація ЗПЗ є досить актуальним [127, 129, 134] завдання, бо впливає на виокремлення унікальних характеристик класів для використання їх в подальшій ідентифікації поліморфних вірусів.

У роботі [1] запропоновано гібридний статичний класифікатор на основі CNN і двонаправленого LSTM для завдань класифікації ЗПЗ в IoT. У роботі [5] запропоновано DL-FHMC, детальний ієрархічний підхід до навчання для виявлення та класифікації ЗПЗ з можливістю виявлення 88,52%. У роботі [8] представлено метод оптимізації гібридного глибокого навчання (DL) шляхом використання інтеграції алгоритмів Backpropagation (BP) і Particle Swarm Optimization (PSO) для забезпечення оптимальних рішень для виявлення зловмисних програм. У роботі [10] використано метод ESMO-CBLSTMAE. Метод ESMO-CBLSTMAE використовує модель згорткового двонаправленого автоматичного кодування короткочасної пам’яті (CBLSTM-AE) для процесу виявлення ЗПЗ та його класифікації. У роботі [12] запропоновано підхід, який використовує переваги методології глибокого перенесення та включає метод тонкого налаштування та різні стратегії поєднання для підвищення продуктивності виявлення та класифікації без необхідності розробки моделей навчання з нуля. У роботі [25] запропоновано модель згорткової нейронної мережі (CNN) на основі глибокого навчання (Deep Learning, DL) для виконання класифікації ЗПЗ в бінарних файлах Portable Executable (PE) з використанням підходу набору функцій fusion. У роботі [40] запропоновано легку модель згорткової нейронної мережі для класифікації ЗПЗ на основі зображень, яку можна запускати на вбудованих системах, таких як Jetson Nano.

Використовуючи добре відомий підхід ансамблевого ML, який називається зваженим голосуванням, у дослідженні [51] виконано динамічний аналіз ознак для мультикласифікації. У роботі [63] пропонується метод класифікації ЗПЗ на основі Word2Vec і багаторівневого сприйняття (MLP). У роботі [90] запропоновано розумну систему та представлено нову модель глибокого навчання для класифікації та мультикласифікації сімейства зловмисних програм. У дослідженні [117] використовувався набір даних про ЗПЗ на основі зображень [Malimg] і

використовувалися різні алгоритми глибокого навчання, CNN, Caps-Net, VGG16, ResNet і InceptionV3, для виявлення ЗПЗ.

Проте, питання класифікації поліморфних вірусів залишається актуальним.

Отже, існують різні рівні складності поліморфних вірусів [92], які по суті і визначають окремі класи. Розглянемо їх.

Рівень 1 характеризується тим, що вибирається схема з множини готових алгоритмів шифрування/дешифрування для створення поліморфного вірусу. Примірник вірусу матиме одну з цих схем у вигляді звичайного тексту. Відкритий ключ для цього шифрування можна надати багатьом користувачам для шифрування повідомлення. Цей простий вірус називається «напівполіморфним».

Рівень 2 базується на характерній процедурі дешифрування вірусу, яка містить одну або декілька постійних інструкцій, а решта є змінними. Наприклад, алгоритм використовує змінні x_1 і x_2, але не використовує змінну x_3, що дозволяє x_3 нескінченно змінюватися.

Рівень 3 характеризується тим, що дешифратор вірусів містить невикористовувані функції або інструкції, такі як NOP, CLI та STI тощо.

Рівень 4 характеризується тим, що дешифрувальник вірусів використовує взаємозамінні інструкції та змінює їх порядок (перемішування інструкцій).

Рівень 5 передбачає формування поліморфного вірусу з використанням усіх методів з рівнів 1-4. Крім того, алгоритм дешифрування можна змінити.

Рівень 6 є найвищим рівнем поліморфних вірусів. Віруси цього рівня ще називають поліморфними вірусами тіла або метаморфними вірусами. На цьому рівні вважається, що він можете змінитиу весь код ядра вірусу (в поколіннях).

Проблема виявлення та протидії комп’ютерним вірусам залишається актуальною, про що свідчить значна кількість досліджень у цьому напрямку [3, 35, 37, 43, 54, 55, 95, 105]. Розглянемо сучасні системи виявлення ЗПЗ.

Аналітики використовують різноманітні інструменти для виявлення ЗПЗ, для захисту та прогнозування майбутніх атак [123]. Серед систем аналізу ЗПЗ з відкритим кодом можна відзначити наступні.

Платформа Google Rapid Response (GRR) [45] – це система реагування на інциденти, розроблена дослідниками безпеки в Google, яка визначає поширені робочі станції ЗПЗ, зосереджені на дистанційній криміналістичній експертизі. Вона складається з програми, встановленої в цільовій системі для зв'язку з агентом, і серверної інфраструктури. GRR – це клієнт (агент) Python, який встановлено на цільових системах, і серверна інфраструктура Python, яка може керувати клієнтами та спілкуватися з ними. Після розгортання сторони сервера та агента вони можуть стати клієнтами GRR і розпочати отримувати повідомлення від серверів. GRR був розроблений для роботи в масштабі, щоб аналітики могли легко отримувати та обробляти дані з великої кількості комп'ютерів.

REMnux [103] – це безкоштовний набір інструментів для Linux, призначений для допомоги аналітикам ЗПЗ у зворотному проєктуванні ЗПЗ. Це робиться для того, щоб користувачі за результатами нештатних випадків могли продовжувати використовувати різноманітне безкоштовне ПЗ, яке може аналізувати ПЗ-вимагач, хоча його може бути важко знайти чи налаштувати. Цей набір інструментів Linux розроблено як універсальний магазин для дослідників, які шукають приклади зворотного проектування ЗПЗ. REMnux орієнтований на Ubuntu та об'єднує кілька ресурсів в один для швидкого аналізу ЗПЗ на основі Windows і Linux. Наріжним каменем проєкту є система REMnux Linux на основі Ubuntu. Цей легкий дистрибутив надає різні ресурси для виявлення програм-вимагачів Windows і Linux, перегляду вразливостей у веб-переглядачі, таких як обфускований JavaScript, дослідження незвичних текстових файлів і видалення інших зловмисних об'єктів. Це допомагає дослідникам досліджувати ЗПЗ в браузері, проводити криміналістику пам'яті, аналізувати численні зразки ЗПЗ, витягувати та декодувати підозрілі елементи тощо.

Cuckoo Sandbox [32] – це платформа з відкритим вихідним кодом, яка автоматизує аналіз ЗПЗ для Windows, Linux і Android і пропонує вичерпну та практичну інформацію про те, як кожен представлений файл працює в ізольованому середовищі. А оскільки це ПЗ з відкритим кодом, то розробники постійно пишуть плагіни, які надають розширені функції. Cuckoo використовується фірмами з виявлення ЗПЗ та безпеки, щоб полегшити навантаження від ручного перегляду

потенційно зловмисних даних. Модульна конструкція дозволяє легко налаштувати етапи запису та аналізу. За останні роки він став одним із найбільш часто використовуваних інструментів з відкритим кодом.

Zeek Network Security Monitor (попередня назва - Bro) [135] – це універсальна мережева аналітична система, яка перетворює мережевий трафік у події, що викликають сценарії. Її можна порівняти з IDS (системою виявлення вторгнень), оскільки вона надає користувачам можливість переглядати їхню мережеву активність, використовуючи як моніторинг на основі сигнатур (шукає правила чи тенденції задокументованого зловмисного трафіку), так і моніторинг на основі аномалій (виявляє незвичайну активність). Тим не менш, його функції ширші ніж звичайних IDS. Хоча Zeek зосереджується на відстеженні безпеки мережі, він також надає комплексний форум для більш загального аналізу мережевого трафіку.

Yara Rules [131] - інструмент ідентифікації ЗПЗ з відкритим вихідним кодом, який може ідентифікувати зразки ЗПЗ на основі текстових або бінарних тенденцій після їх тестування в Cuckoo. Дослідники використовують Yara для створення визначень сімейств ЗПЗ на основі шаблонів. YARA означає «ще один рекурсивний акронім», оскільки описи називаються правилами. Це допомагає дослідникам ідентифікувати та класифікувати очевидно подібні типи ЗПЗ та може бути адаптовано для використання в Cuckoo. Його можна використовувати для Windows і для Linux. Правила Yara додано до багатьох систем виявлення та реагування на кінцеві точки, щоб допомогти їм ідентифікувати зразки ЗПЗ, з якими вони стикаються, класифікувати їх і пізніше поділитися своїми висновками з клієнтами та спільнотою.

Також, існують інструменти аналізу ЗПЗ для мобільних пристроїв: APKTool [12], Mobile-Sandbox [84], Dex2Jar [35].

Virustotal [128] – це служба, яка аналізує підозрілі файли та URL-адреси і допомагає швидко виявляти віруси, worm-віруси, трояни та різні зловмисні програми, виявлені антивірусними механізмами. На додаток до різноманітних методів видалення сигналів із досліджуваного матеріалу, VirusTotal перевіряє продукти за допомогою понад 70 антивірусних сканерів і служб чорного списку URL/доменів. Кожен може використовувати свій браузер, щоб вибрати файл зі свого пристрою та

надіслати його до VirusTotal. VirusTotal пропонує різні методи завантаження даних, включаючи загальнодоступний веб-портал за замовчуванням, завантажувачі з робочого столу, розширення браузера та програмний API. Веб-інтерфейс має найбільший пріоритет сканування серед загальнодоступних форм програми. Специфікації можна створити за допомогою загальнодоступного API на основі HTTP будь-якою мовою програмування. Він також пропонує низку інших функцій, у тому числі спільноту VirusTotal. Це мережа, яка дозволяє користувачам повідомляти про файли та URL-адреси та обмінюватися коментарями один з одним. Це може бути корисним для виявлення зловмисного вмісту, а також для пошуку хибних спрацьовувань – звичайних і незловмисних об’єктів, визначених одним або кількома сканерами як небезпечні.

Malzilla [78] – це інструмент пошуку ЗПЗ для аналізу вебсайтів, які містять зловмисний код. Вебсторінки, які містять експлойти, часто використовують послідовність перенаправлень і заплутаний код, щоб їх було важко відстежити. Це дозволяє користувачам отримувати доступ до вебсайтів і отримувати весь їхній вихідний код, наприклад wget, не відвідуючи сайт і потенційно не пошкоджуючи свій пристрій. Ця програма має можливість перемикання агента користувача та вибору власного реферера користувача. Це показує повний список вебсторінок для браузерів і всі заголовки для HTTP. Він також має проксі-функції, складні декодери та, що особливо важливо, деобфускацію коду JavaScript, все в одній програмі.

Також, можна виділити і ряд антивірусних програм, які дозволяють виявити поліморфні віруси. Серед них найбільш відомими є Symantec (США), Bitdefender (Румунія), McAfee (США), ESET NOD32 (Словаччина), Trend Micro (Японія), Sophos (Великобританія), Malwarebytes (США), Avast (Чехія), F-Secure (Фінляндія) (табл. 1.2).

Таким чином, проблема поширення ЗПЗ продовжує залишатись актуальною. Особливо складним ЗПЗ є поліморфні віруси. Наявні засоби виявлення є різноспрямованими і не завжди можуть виявити унікальні характерні особливості ЗПЗ, в основі якого реалізовано певні рівні поліморфізму.

Таблиця 1.2

Порівняння антивірусних програм стосовно використання методів виявлення та можливостей аналізу темпів поширення та класифікації поліморфних вірусів

Антивірусна програма	Аналіз темпів поширення поліморфних вірусів	Методи виявлення поліморфних вірусів						Класифікація за рівнями складності (рівнями поліморфізму)	Час реакції	Виявлення нових загроз
		Евристичний аналіз	Аналіз поведінки	Машинне навчання	Хмарний аналіз	Анти-обфускація	Probabilistic Logic Network (PLN)			
Symantec (США) [121]	Середній	+	+	+	+	+	-	Середній	Середній	Високе
Bitdefender (Румунія) [18]	Високий	+	+	+	+	+	-	Високий	Швидкий	Високе
McAfee (США) [87]	Середній	+	+	+	+	+	-	Середній	Швидкий	Високе
ESET NOD32 (Словаччина) [40]	Середній	+	+	+	-	+	-	Середній	Швидкий	Високе
Trend Micro (Японія) [125]	Високий	+	+	+	-	+	-	Високий	Швидкий	Високе
Sophos (Великобританія) [118]	Середній	+	+	+	-	+	-	Середній	Швидкий	Високе
Malwarebytes (США) [78]	Низький	+	+	+	-	-	-	Низька	Середній	Високе
Avast (Чехія) [14]	Високий	+	+	+	-	+	-	Середній	Швидкий	Високе
F-Secure (Фінляндія) [43]	Середній	+	+	+	-	+	-	Середній	Середній	Високе

1.2. Методи виявлення поліморфних вірусів

Значна кількість досліджень присвячена різним методам, технікам і підходам до аналізу та виявлення ЗПЗ [9, 75, 91, 137, 140, 141, 143, 146, 147, 148, 150]. В роботі [3] подано комплексний огляд досліджень підходів до виявлення ЗПЗ. Серед них можна відзначити машинне навчання [7, 28, 62, 106], згорткові нейронні мережі (CNN) [26, 89], метод, який заснований на механізмі уваги [29], динамічний метод виявлення

зловмисних програм (AMD) [52], метод аналізу головних компонент [60], система на основі емуляції під назвою UBER для покращення пісочниці аналізу ЗПЗ [71], техніка захисту від ухилення на основі DNS [74], напівконтрольована нечітка кластеризація c-Means [76], аналіз основних компонент (PCA) для вилучення функцій, комбінація вихідних кодів з виправленням помилок (ECOC) і адаптивної системи нейронечіткого висновку (ANFIS) для класифікації [77], модель класифікатору випадкового лісу для вибору ознак і модель опорних векторних машин (SVM) для виявлення ЗПЗ [93], алгоритми Boyer-Moore, Aho–Corasick, Naïve String пошуку, Rabin Karp String Search та Knuth–Morris–Pratt [94], архітектура системи наближеної до традиційної нейро-нечіткої системи Takagi-Sugeno-Kang [96], наскрізна модель для виявлення мережевих атак і класифікації мережевих атак з використанням рекурентних моделей на основі глибокого навчання [102], підхід для виявлення DNS-тунелів ботнету, який враховує всі особливості та архітектурні характеристики ботнетів [110].

У роботі [4] пропонується система інтелектуального агента (IA) для виявлення DDoS-атак за допомогою автоматичного вибору та вибору функцій. У роботі [12] запропоновано підхід, який використовує переваги методології глибокого перенесення та включає метод тонкого налаштування та різні стратегії комбінування для покращення продуктивності виявлення та класифікації без необхідності розробки моделей навчання з початку. У роботі [32] порівнюються методи виявлення зловмисних програм на основі статичного, динамічного та гібридного аналізу. У роботі [35] пропонується новий системний підхід до ідентифікації сучасного ЗПЗ з використанням методів динамічного глибокого навчання в поєднанні з евристичними підходами для класифікації та виявлення п’яти сучасних сімейств ЗПЗ: рекламне ПЗ; Radware; руткіт; ЗПЗ для SMS; програми-вимагачі. В роботі [39] запропоновано інтегровану структуру для впровадження IoT з технологією блокчейн для гарантування високого рівня безпеки та процесу перевірки на основі інтеграції між консенсусними алгоритмами блокчейну (PBFT і Tangle). У роботі [55] розроблено концептуальну модель багатокомп’ютерних систем, яка призначена для забезпечення функціонування антивірусних приманок і пасток для виявлення зловмисних програм. У роботах [69, 70] запропоновано новий підхід виявлення шляхом генерації

структурних особливостей шляхом обчислення потоку фрагментів байтів з використанням коефіцієнта стиснення, ентропії, коефіцієнта подібності Жаккара та статистичного тесту Хі-квадрат. У роботі [81] представлено підхід до виявлення метаморфних вірусів на основі аналізу ознак їх обфускації. У роботі [83] для виявлення ЗПЗ використовувався алгоритм K-NN. У роботі [93] для виявлення ЗПЗ була використана модель опорного вектора машини (SVM). Динамічний аналіз ЗПЗ з підкріпленням навчання було проведено в роботі [100]. У роботі [109] запропоновано метод визначення ефективності розподіленої системи виявлення аномальних проявів. У роботі [112] запропоновано метод виявлення невідомих метаморфічних вірусів, який базується на аналізі потенційно підозрілої поведінки програм на хості, а в роботі [111] – метод виявлення метаморфічних вірусів, який базується на пошуку еквівалентних функціональних блоків. В роботі [108] розглянуто проблеми, пов’язані з виявленням App-DDoS, і представлено високоефективне та адаптоване рішення для виявлення різних типів App-DDoS-атак.

Отже, наявний досить широкий вибір методів виявлення ЗПЗ, які реалізовано відновідними засобами, але проблема забезпечення повного виявлення поліморфного ЗПЗ залишається актуальною. Методи виявлення поліморфного ЗПЗ можна комбінувати в залежності від оточуючого середовища, швидкості його поширення тощо. В такому підході виникає завдання вибору методів, які будуть застосовані першочергово, і методів, застосування яких буде потрібне пізніше. Тобто, частина методів може дати позитивний результат на початку виявлення ЗПЗ, але може пропустити частину ЗПЗ, яке може мати певні спроможності для уникнення виявлення і обходу засобів виявлення. Тоді, можуть бути застосовані методи для більш детальнішого аналізу. Для забезпечення такого дослідження виконуваних програм використовують моделі, що дають змогу врахувати параметри дослідження об’ктів та обрати відповідні методи для різних етапів їх дослідження. Однією з таких моделей є модель Лотки-Вольтерра (модель «хижак-жертва») [18, 97], яка знайшла широке застосування в різних сферах: у космічних дослідженнях [27], біології [31, 34], у багатьох галузях техніки [68], медицини [78], оцінюванні безпеки кіберфізичних

систем [131]. Застосування даної моделі для дослідження процесу виявлення та ідентифікації поліморфного ЗПЗ є доцільним та актуальним.

Отже, серед методів аналізу сучасного ЗПЗ [55, 81, 99, 112, 109] є методи, в основі яких наявні алгоритми штучного інтелекту (machine learning), що аналізують зловмисну програму у віртуальній машині. Віртуальна машина може запускати запакований потенційно небезпечний файл і динамічно аналізувати його, автоматично тестуючи код і поведінку. Крім того, перспективним для розвитку є дослідження, в якому антивірусне ПЗ використовує сучасні методи машинного навчання та аналіз поведінки в реальному часі в поєднанні зі статичними методами для виявлення підозрілої активності та запобігання загрозам. Такий підхід до виявлення зловмисних програм називають гібридним [32].

Виявлення поліморфних вірусів, також, вимагає використання комбінації методів статичного та динамічного аналізу [17]. Наприклад, у роботі [42] запропоновано виявлення поліморфних кібератак за допомогою системи нечіткого висновку. Хоча статичний аналіз може дати початкове розуміння поведінки ЗПЗ, але він часто неефективний через швидку зміну поліморфного коду ЗПЗ. Тому, методи динамічного аналізу важливі для ефективного виявлення та нейтралізації загроз. Динамічний аналіз передбачає запуск ЗПЗ в контрольованому середовищі, такому як віртуальна машина або пісочниця, для спостереження за його поведінкою [125]. Відстежуючи дії системи, модифікації файлів, мережеві підключення та інші показники, аналітики безпеки можуть ідентифікувати підозрілу поведінку та відповідним чином класифікувати ЗПЗ [99]. Методи аналізу поведінки часто використовуються для покращення можливостей виявлення. Ці методи включають моніторинг поведінки файлу під час виконання, аналіз його дій та оцінку ризиків, які він становить. Порівнюючи поведінку потенційно зловмисного виконуваного файлу з відомими шаблонами та евристиками, інструменти безпеки можуть швидко ідентифікувати екземпляри поліморфного ЗПЗ. Крім того, алгоритми машинного навчання відіграють важливу роль у виявленні поліморфних зловмисних програм. Завдяки моделям і великим наборам даних відомого ЗПЗ ці алгоритми вчаться ідентифікувати зловмисні прогарми та відрізняти поліморфне ЗПЗ від законного ПЗ. Цей підхід забезпечує

ефективне та масштабоване рішення для боротьби з постійно зростаючою загрозою поліморфного ЗПЗ. Оскільки поліморфне ЗПЗ продовжує розвиватися та ухиляється від традиційних методів виявлення та виправлення, то впровадження ефективних контрзаходів стає дедалі гострішою потребою. Нездатність пом'якшити загрозу поліморфного ЗПЗ може призвести до наслідків, таких як витік даних, фінансові втрати та репутаційна шкода. Тому, необхідним є узагальнення основних методів виявлення поліморфних вірусів.

Метод пошуку ЗПЗ є ефективним методом, який використовується в кібербезпеці для виявлення потенційного ЗПЗ в системі [47, 135]. Він передбачає сканування двійкового коду або коду програми для пошуку певних рядків даних, які зазвичай асоціюються зі ЗПЗ. Одним із найпоширеніших інструментів для пошуку рядків є команда strings у системах на базі Unix. Ця команда сканує файл і виводить будь-які послідовності друкованих символів, які часто вказують на зрозумілі людині рядки в коді програми. У контексті виявлення зловмисних програм ці рядки можуть надати цінну інформацію про потенційну поведінку підозрілого файлу. Наприклад, вони можуть виявити наявність підозрілих викликів API, шляхів до файлів, URL-адрес або ключів реєстру, які часто пов'язані зі зловмисною діяльністю. Однак пошук рядків не є надійним методом. Досвідчені зловмисники з розроблення ЗПЗ часто використовують методи обфускації, щоб приховати свої рядки, або вони можуть взагалі уникати використання підозрілих рядків. Крім того, легальні програми також можуть містити підозрілі на вигляд рядки випадково. Таким чином, хоча пошук рядків може бути корисним першим кроком у аналізі ЗПЗ, але важливо підтвердити результати за допомогою інших методів. Це може включати динамічний аналіз (спостереження за поведінкою програми під час виконання), статичний аналіз (перевірка коду програми без її запуску) або евристичний аналіз (порівняння поведінки програми або шаблонів коду з відомими сигнатурами ЗПЗ).

Одним із найбільш перспективних способів виявлення ЗПЗ є використання методів аналізу даних. Ці методи включають аналіз великих наборів даних для виявлення шаблонів, асоціацій або аномалій, які можуть вказувати на зловмисну діяльність [16, 107, 114].

Першим кроком з виявлення у методі інтелектуального аналізу даних є збір даних. Це передбачає збір широкого діапазону даних, таких як журнали мережевого трафіку, відстеження системних викликів і дії користувачів. Дані можуть бути зібрані з однієї комп’ютерної станції або мережі комп’ютерів для більш широкого аналізу. Коли дані зібрані, то їх часто попередньо обробляють, щоб перетворити у відповідний формат для аналізу даних. Наприклад, необроблені дані, можливо, потрібно буде перетворити в числовий формат або відфільтрувати нерелевантні дані. Потім, попередньо оброблені дані піддаються алгоритмам інтелектуального аналізу даних. Існує кілька типів методів інтелектуального аналізу даних, які можна використовувати, включаючи класифікацію, кластеризацію, регресію та виявлення аномалій. Ці методи можуть допомогти виявити шаблони або аномалії, які можуть свідчити про наявність ЗПЗ. Нарешті, результати можуть бути представлені у форматі, який аналітики з комп'ютерної безпеки легко інтерпретують, наприклад, візуальна панель або система сповіщень. Класифікація, наприклад, передбачає навчання моделі розпізнаванню характеристик відомого ЗПЗ, а потім використання цієї моделі для класифікації нових даних як безпечних або зловмисних. З іншої сторони, кластеризація групує подібні дані разом, що може допомогти виявити шаблони в даних, які можуть вказувати на атаку. Після процесу інтелектуального аналізу даних результати часто піддаються постобробці, щоб видалити будь-які хибні позитивні чи негативні результати. Це може включати перехресну перевірку результатів за допомогою інших методів виявлення або ручну перевірку виявлення ЗПЗ. Хоча видобуток даних може бути потужним інструментом для виявлення ЗПЗ, але він не безпомилковий. Іноді він може видавати хибні спрацювання чи давати негативну відповідь. Тому він може виявити не всі типи ЗПЗ. Однак у поєднанні з іншими методами виявлення інтелектуальний аналіз даних може значно підвищити здатність системи виявляти загрози ЗПЗ та реагувати на них.

Аналіз ЗПЗ в пісочниці – це техніка, яка використовується фахівцями з кібербезпеки для аналізу та розуміння поведінки ЗПЗ в контрольованому середовищі [15, 118]. Він передбачає запуск ЗПЗ у віртуальному або ізольованому середовищі, відомому як пісочниця, для спостереження за його діями та збору цінної інформації.

Метою аналізу ЗПЗ є розкриття можливостей ЗПЗ, ідентифікація потенційних загроз і розробка ефективних заходів протидії. Виконуючи ЗПЗ в контрольованому середовищі, аналітики можуть вивчати його взаємодію з операційною системою, мережею та іншими компонентами ПЗ.

Під час аналізу використовуються різні динамічні та статичні методики. Динамічний аналіз включає моніторинг поведінки ЗПЗ під час виконання, наприклад, модифікації файлової системи, мережевий зв'язок і системні виклики. З іншої сторони, статичний аналіз зосереджується на дослідженні коду та структури ЗПЗ без виконання. Інформація, зібрана в результаті аналізу поведінки зловмисної програми в ізольованому програмному середовищі, допомагає визначити вектори зараження, інфраструктуру та методи керування, механізми доставки корисного навантаження та потенційні методи викрадення даних. Ці знання мають вирішальне значення для розробки ефективних методів виявлення, оновлення елементів керування безпекою та пом'якшення впливу атак зловмисних програм. Тому, аналіз в ізольованому програмному середовищі є важливою складовою сучасних практик кібербезпеки. Він надає цінну інформацію про поведінку та характеристики ЗПЗ, дозволяючи організаціям в сфері кібербезпеки покращувати свої механізми захисту та розробляти перспективні методи протидії новим загрозам.

Традиційні методи виявлення ЗПЗ не встигають реагувати на стрімку зміну ландшафту атак ЗПЗ. Методи машинного навчання стали новим інструментом для покращення виявлення ЗПЗ та боротьби з цими загрозами. Алгоритми машинного навчання можуть аналізувати великі обсяги даних і витягувати шаблони та функції, які можна використовувати для виявлення зловмисної поведінки [28, 62]. Навчаючи моделі на відомих зразках ЗПЗ та законному ПЗ, алгоритми машинного навчання можуть навчитися розрізняти їх і точно класифікувати нові та невідомі файли. Однією з ключових переваг використання машинного навчання для виявлення ЗПЗ є його здатність адаптуватися та вчитися на нових загрозах. З появою нових типів ЗПЗ моделі машинного навчання можна оновлювати та перенавчати для ефективного виявлення цих нових загроз.

Існує кілька підходів до виявлення ЗПЗ за допомогою машинного навчання, включаючи статичний аналіз і динамічний аналіз. Статичний аналіз передбачає перевірку коду та структури файлу без його виконання, тоді як динамічний аналіз передбачає запуск файлу в контрольованому середовищі для спостереження за його поведінкою. Обидва підходи можуть надати цінну інформацію для виявлення ЗПЗ.

В роботі [19] запропоновано метод класифікації поліморфного ЗПЗ під назвою Malwise (рис. 1.3), який використовує емуляцію на рівні програми для розпакування коду ЗПЗ [22]. Однак, важливо зазначити, що виявлення ЗПЗ за допомогою машинного навчання не завжди дає змогу досягти достатнього результату. Змагальні атаки, коли зловмисники маніпулюють ЗПЗ, щоб уникнути виявлення, можуть становити значну проблему. Крім того, великий обсяг даних і необхідність постійного оновлення та перенавчання моделей вимагають значних обчислювальних ресурсів. Тому, машинне навчання пропонує перспективні рішення для виявлення ЗПЗ, використовуючи свою здатність аналізувати великі обсяги даних і виявляти шаблони. Постійно вдосконалюючи та оновлюючи моделі, машинне навчання може підвищити безпеку комп'ютерних систем і мереж від нових загроз ЗПЗ.

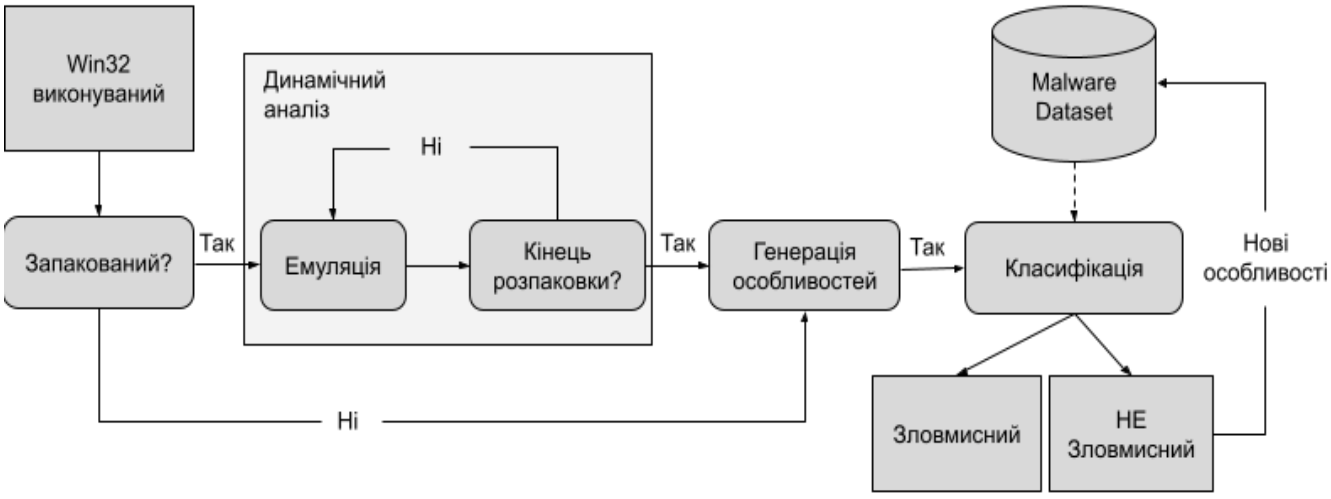

Рисунок 1.3 - Блок-схема системи класифікації ЗПЗ [22]

Розробка структурних функцій є ключовим аспектом ефективних моделей виявлення ЗПЗ [69, 70, 83]. Витягаючи значущі функції зі структурованих даних, аналітики та дослідники даних можуть підвищити точність і надійність своїх систем

виявлення ЗПЗ. У наступних кроках описано метод розробки структурних функцій, спеціально розроблений для виявлення ЗПЗ.

Крок 1. Розуміння даних. Отримання повного розуміння структури та характеристик даних ЗПЗ, а також визначення відповідних змінних, їх типів та будь-які шаблони чи зв'язки, присутні в наборі даних.

Крок 2. Ідентифікація функцій, яка полягає у визначенні функцій, що можуть бути інформативними для виявлення ЗПЗ. Цього можна досягти за допомогою знання домену, дослідницького аналізу даних або статистичних методів, які спеціально розроблені для виявлення ЗПЗ.

Крок 3. Вилучення функцій, тобто витягнення вибраних функцій з необроблених даних ЗПЗ та перетворення їх у відповідний формат для аналізу. Для реалізації цього кроку застосовують методи математичних перетворень, масштабування, нормалізації або кодування для попередньої обробки функцій.

Крок 4. Побудова функцій полягає в створенні нових функцій, поєднуючи або змінюючи існуючі функції таким чином, щоб охопити важливі аспекти поведінки ЗПЗ. Це може включати агрегації, математичні операції або взаємодію між змінними.

Крок 5. Вибір функцій. Обрання найбільш релевантних функцій для виявлення ЗПЗ, що допомагає зменшити розмірність, підвищити ефективність і точність моделі виявлення.

Крок 6. Кодування ознак забезпечується кодуванням категоріальних ознак в числові представлення, які можуть бути оброблені алгоритмами машинного навчання. Використання таких методів, як одноразове кодування, кодування міток або цільове кодування використовують для ефективного представлення категоріальних змінних.

Крок 7. Масштабування функцій здійснюють для приведення їх значень до загального діапазону, щоб переконатися, що вони мають порівнювані величини. Для реалізації цього кроку можна використовувати методи стандартизації та нормалізації.

Крок 8. Перевірка функцій необхідна для оцінювання їх продуктивності у моделі виявлення ЗПЗ. Використання таких методів, як перехресна перевірка та показники оцінки моделі необхідні для того, щоб виміряти ефективність розроблених функцій і за необхідності їх ітеративно вдосконалення.

Дотримуючись цього методу розроблення структурних особливостей, аналітики та дослідники даних можуть підвищити точність і надійність своїх систем виявлення ЗПЗ, що призведе до покращених заходів кібербезпеки та захисту від ЗПЗ.

Недоліки розглянутих методів вимагають нових підходів до виявлення та аналізу ЗПЗ. Серед них - виявлення ЗПЗ за допомогою імовірнісних логічних мереж (PLN). Використання PLN [48, 104, 101] надає потужний підхід до виявлення та пом'якшення загроз ЗПЗ. PLN поєднують імовірнісне міркування з логічним висновком для моделювання складних взаємозв'язків і залежностей у виявленні ЗПЗ. PLN – це гібридна архітектура, яка об'єднує ймовірнісні графічні моделі з логікою першого порядку. Вони забезпечують гнучке та виразне представлення для фіксації невизначеності та міркування про складні сфери. PLN використовують сильні сторони як імовірнісного міркування, так і логічного висновку, що робить їх придатними для виявлення ЗПЗ. Однією з ключових переваг PLN у виявленні ЗПЗ є їх здатність обробляти невизначену та неповну інформацію. Призначаючи ймовірності різним гіпотезам, PLN можуть оцінювати ймовірність присутності ЗПЗ та приймати обґрунтовані рішення. Це ймовірнісне міркування дозволяє використовувати більш точні та адаптивні механізми виявлення. PLN вловлюють складну поведінку та моделі зловмисних програм. Вони можуть представляти як статичні, так і динамічні характеристики ЗПЗ, включаючи структуру коду, взаємодію системи та механізми розповсюдження. Моделюючи таку поведінку, PLN можуть ефективно відрізняти законне ПЗ від ЗПЗ. Щоб навчити PLN виявляти ЗПЗ, потрібен великий набір даних відомих зразків ЗПЗ та законного ПЗ. Для вивчення параметрів і структури PLN з цих даних можна використовувати методи машинного навчання. Ітеративно удосконалюючи PLN за допомогою навчальних прикладів, його можна налаштувати для точного виявлення та класифікації ЗПЗ.

У роботах [143, 144] запропоновано метод оцінювання безпеки кіберфізичних систем на основі моделі Лотки-Вольтерра із плином часу, який передбачає створення класифікатора загроз на кіберфізичні системи (оцінка синергізму, гібридності), запропоновано алгоритм визначення економічних витрат (модель з урахуванням обчислювальних можливостей та спрямованостей кібератак, модель з урахуванням

можливої конкуренції зловмисників щодо «жертви», модель з урахуванням можливості групування зловмисників з метою досягнення цілей кібератаки, модель з урахуванням кіберзв'язків між «видами жертв» і «видами хижаків».

Таким чином, перспективними для розроблення методів виявлення поліморфного ЗПЗ є методи з використанням динамічного аналізу виконуваних програм та здійснення такого аналізу згідно моделі Лотки-Вольтерра (модель «хижак-жертва»). Даний підхід, також, дає змогу організувати процес дослідження виконуваних програм набором різних методів з урахуванням змінюваних параметрів на різних етапах дослідження та аналізу.

1.3. Аналіз мультиагентних систем виявлення зловмисного програмного забезпечення

Агент – це комп'ютерна система, розміщена в зовнішньому середовищі, здатна взаємодіяти з ним, виконуючи автономні раціональні дії для досягнення цілей [4, 67, 121]. Зазвичай для того, щоб вважатися «розумним», агент повинен мати такі властивості: реактивність (reactivity), тобто агент повинен відчувати зовнішнє середовище і реагувати на зміни в ньому, виконуючи дії, спрямовані на досягнення цілей; проактивність (pro-activeness) – агент повинен проявляти цілеспрямовану поведінку, проявляти ініціативу, виконувати дії, спрямовані на досягнення цілей; комунікабельність (соціальна здатність) – агент повинен взаємодіяти з іншими суб'єктами зовнішнього середовища (іншими агентами, людьми тощо) для досягнення цілей.

Формальну модель інтелектуального агента (ІА) в багатоагентних системах можна описати як систему окремих агентів, які взаємодіють один з одним. Кожен агент представлений як дискретна сутність із визначеним станом. Стан агента може змінюватися з часом відповідно до набору правил або функцій, які базуються на поточному стані агента та стані його середовища. Ці правила або функції можна виразити за допомогою різних дискретних математичних структур, таких як графи, послідовності та набори. Наприклад, зв'язки між агентами можна представити у

вигляді графа, де кожен вузол представляє агента, а кожне ребро – можливу взаємодію між двома агентами.

Крім того, послідовність дій, які виконує агент, можна представити у вигляді послідовності станів, де кожному стану відповідає дія. Набір усіх можливих дій для агента можна представити як набір, а стратегію агента можна визначити як функцію, яка відображає поточний стан агента на дію в цьому наборі. Ця формальна модель дозволяє аналізувати та прогнозувати поведінку багатоагентних систем за допомогою дискретних математичних методів. Це дозволяє досліджувати різні властивості системи, такі як стабільність, конвергенція та оптимальність.

Концепція багатоагентних систем (MAC) стала важливою темою інтересів у сфері штучного інтелекту [6, 38, 49]. В основі цих систем лежать інтелектуальні агенти (ІА), які є суб'єктами, що здатні сприймати оточуюче середовище та виконувати дії самостійно або у співпраці з іншими для досягнення конкретних цілей. Формальна модель ІА пропонує основу для розуміння, проєктування та вдосконалення цих складних систем [30, 98, 113].

Мультиагентна система, по суті, є набором кількох взаємодіючих ІА. Кожен агент, у своїй найпростішій формі, є обчислювальною сутністю, яка відчуває оточуюче середовище та реагує відповідно до заданих алгоритмів. Ці агенти здатні до автономної поведінки, можуть вчитися на своєму досвіді та мають здатність взаємодіяти з іншими агентами в системі. Відносини між агентами всередині системи можна представити у вигляді графа. Кожен вузол представляє агента, а кожне ребро представляє потенційну взаємодію між двома агентами. Це візуальне представлення дозволяє зрозуміти та проаналізувати взаємодію всередині системи. Подібним чином ряд дій, які виконує агент, можна представити у вигляді послідовності станів. Кожен стан у цій послідовності відповідає певній дії, яку виконує агент. Цей послідовний підхід забезпечує покрокове представлення дій агента. Як альтернатива, повний діапазон можливих дій, які доступні для агента в системі, може бути представлений як набір. Використовуючи цей підхід, стратегію агента можна визначити як функцію, яка відображає поточний стан агента на дію в цьому наборі. Цей метод забезпечує повний огляд усіх потенційних дій, сприяючи глибшому розумінню поведінки агента.

У роботі [84] розроблено структуру на основі багатоагентних систем (MASFMMS - Multi Agent Systems Framework for Malware Modeling and Simulation) для моделювання ефекту та поширення ЗПЗ в комп'ютерній мережі (рис. 1.4). Авторами було включено параметри із таксономії, наданої Weaver та ін. Переваги даного підходу полягають у масштабованості при моделюванні великої кількості комп'ютерів, врахуванні нематеріальних параметрів, таких як поведінка людей (як агентів) на комп'ютерах, і можливості включення різноманітних параметрів ЗПЗ. Експерименти демонструють різноманітність, з якою можна використовувати MASFMMS, і тепер можна вводити нові параметри в моделювання та динамічно контролювати агента та середовище. Поточні зусилля щодо розробки спрямовані на розширення цієї основи для прогнозування поширення ЗПЗ в мережі та впливу коригувальних дій, вжитих для його стримування.

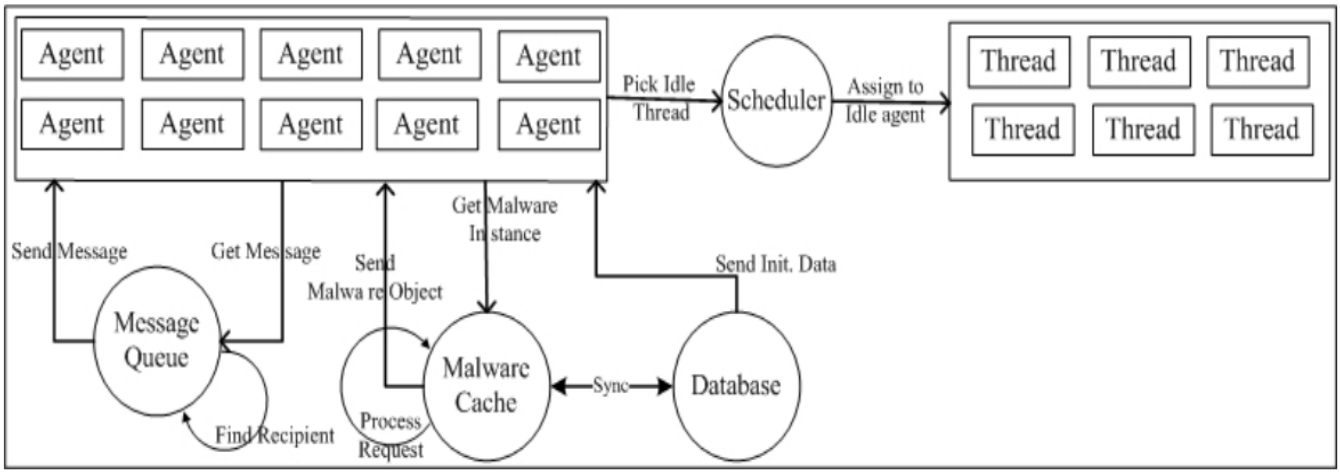

Рисунок 1.4 - Архітектура MASFMMS [84]

Недоліком даного підходу є обмеженість у виявленні ЗПЗ.

У роботі [20] запропоновано розподілену систему виявлення вторгнень (MAD-IDS), яка об'єднує бажані функції, що надані багатоагентною методологією, з високою точністю методів інтелектуального аналізу даних. Запропонована система спирається на набір ІА, які збирають і аналізують мережеві з'єднання, а методи інтелектуального аналізу даних виявляються корисними для виявлення вторгнень. Розподілена архітектура MAD-IDS складається з різних кооперативних, комунікативних і спільних агентів для збору й аналізу великих обсягів мережевого трафіку, що називаються:

Sniffer; Filter; Misuse Detection; Anomaly Detection; Rule Mining; Reporter Agent. На рис. 1.5 зображено загальну архітектуру MAD-IDS.

Недоліком даного підходу є обмеженість у виявленні та класифікації поліморфних вірусів.

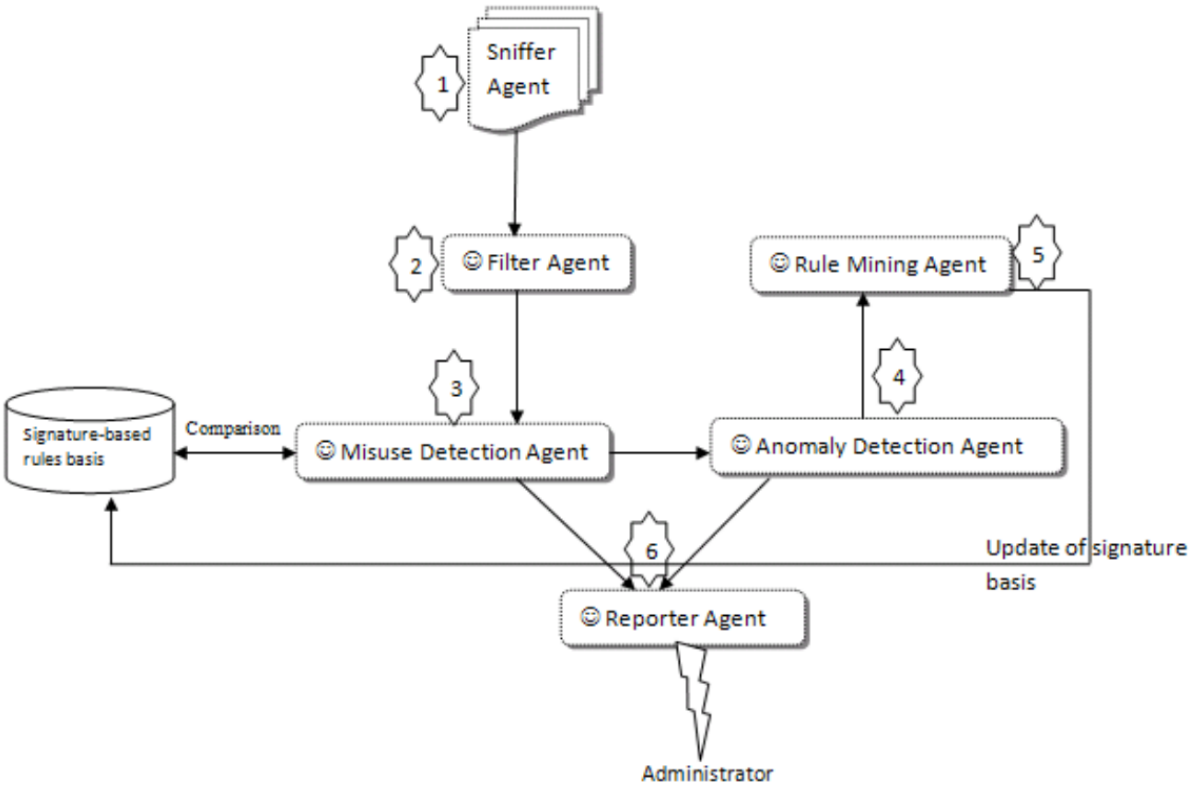

Рисунок 1.5 - Архітектура MAD-IDS [20]

У роботі [2] запропоновано інтелектуальну та поширену платформу контролю безпеки інформаційної системи (ІС) на основі МАС, яка виявляє та класифікує події ЗПЗ, що виникають на програмних і апаратних компонентах, і надсилає сповіщення в режимі реального часу користувачам смартфонів (рис. 1.6). Пропонована платформа складається з двох компонентів: центральний програмний компонент (CSC) - програма супервізора, яка керує пристроями, оцінює журнали та сигнали тривоги контрольованих компонентів; вбудований програмний компонент (ESC) на мобільному пристрої (MD) з ОС ANDROID. На платформі реалізовано агенти на ESC за допомогою JADE-LEAP android. ESC призначений для смартфонів і планшетів, які працюють під керуванням операційної системи Android, JADE API на CSC, і зв'язок

між цими двома компонентами здійснюється за допомогою Secure Протокол Socket Layer (SSL), реалізований за допомогою плагіна JADE-S.

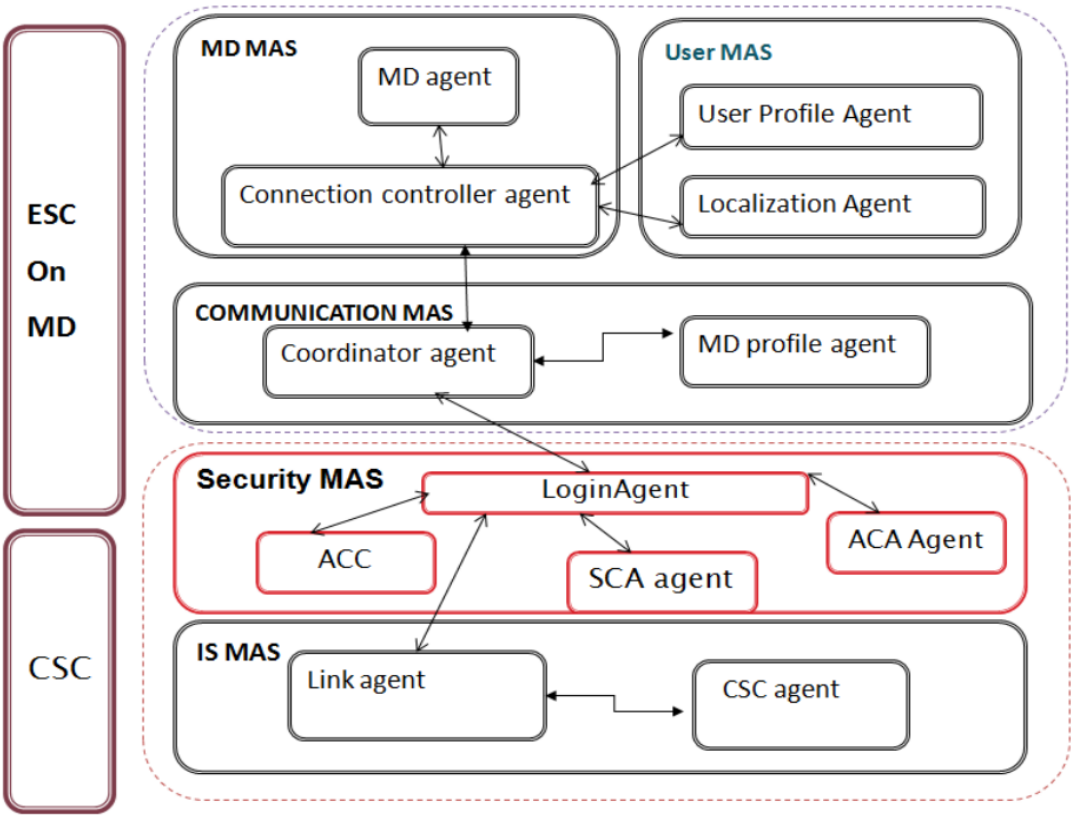

Рисунок 1.6 – Архітектура MAS [2]

У роботі [82] запропонувано нову архітектуру багатоагентного детектора ексфільтрації даних (MADEX), який синтезовано по аналогії з механізмами та функціями імунної системи людини. MADEX прагне виявити дії з викрадання даних, які здійснюються невидимими та прихованими зловмисними програмами, що приховують зловмисний трафік від зараженого хоста в мережах із низькою затримкою.

У роботі [116] представили підхід до розробки багатоагентної схеми RAS проти системних таємних кібератак. Зокрема, запропоновано дворівневу ієрархічну архітектуру, яка складається з розподілених локальних контролерів RAS (RASc) як локальних агентів, що працюють у різних зонах/областях, які постійно контролюються центральним агентом (рис. 1.7).

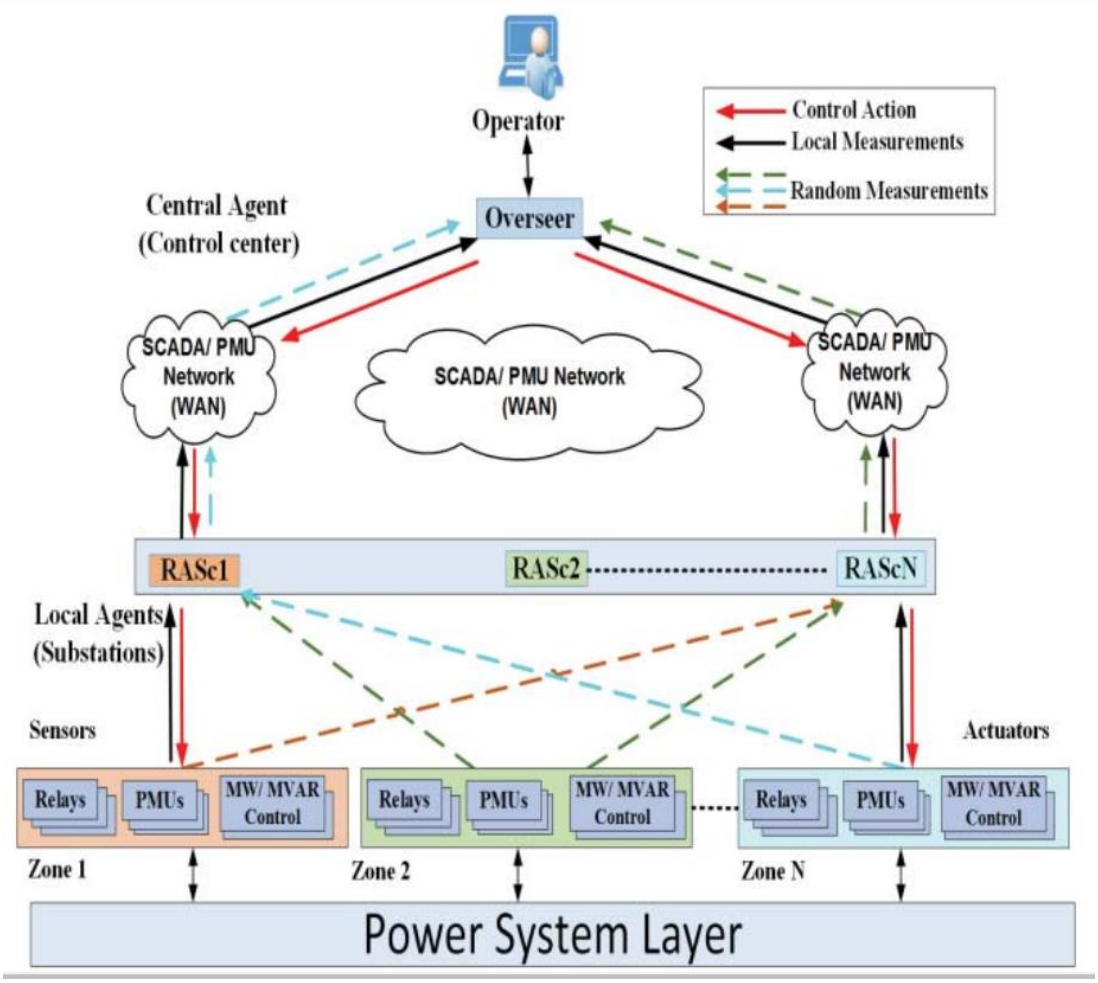

Рисунок 1.7 – Архітектура для мультиагентної RAS [116]

Таким чином, аналізовані рішення щодо виявлення ЗПЗ є перспективними для розроблення та застосування. При цьому кожне окреме рішення є недосттанім в контексті повного виявлення і є потреба в застосуванні комплексного підходу. Для його забезпечення доцільним є розроблення мультиагентних систем, які б забезпечували координацію агентів, в яких реалізовано певні методи виявлення ЗПЗ.

1.4. Постановка задачі дослідження

Для вирішення задачі покращення ефективності виявлення поліморфних вірусів необхідно розробити архітектуру та реалізувати гібридну мультиагентну систему виявлення поліморфних вірусів, яка здійснюватиме аналіз середовища, досліджуватиме темпи поширення, виявлятиме, класифікуватиме поліморфні віруси

та використовуватиме різні стратегії прийняття рішення, враховуючи типи загроз. Для цього необхідно вирішити наступні завдання:

1) провести аналіз поліморфних вірусів за різними рівнями складності їх структури, методів та засобів їх виявлення за допомогою мультиагентних систем;

2) розробити моделі класів поліморфних вірусів згідно різних рівнів поліморфізму;

3) розробити архітектуру гібридних мультиагентних систем для виявлення, аналізу та класифікації поліморфних вірусів, а також модель інтелектуального агента в архітектурі гібридних мультиагентних систем для виявлення поліморфних вірусів;

4) удосконалити метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра для дослідження впливу кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі;

5) розробити метод виявлення поліморфних вірусів для покращення ефективності виявлення поліморфних вірусів та визначення ймовірності належності виявлених вірусів до класу поліморфних;

6) удосконалити метод класифікації поліморфних вірусів, який дозволить не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначатиме ймовірність їх належності до нечітких термів кожного рівня складності;

7) розробити гібридну мультиагентну систему виявлення поліморфних вірусів, яка здійснюватиме аналіз середовища, встановлюватиме темпи поширення, виявлятиме, класифікуватиме поліморфні віруси та використовуватиме різні стратегії прийняття рішення і провести з нею експериментальні дослідження щодо встановлення ефективності функціонування та достовірності виявлення поліморфних вірусів і впровадити її у виробництво.

Під ефективністю функціонування гібридної мультиагентної системи будемо розуміти відсоток виявлених поліморфних вірусів у загальній кількості згенерованих (із врахуванням рівнів складності поліморфних вірусів).

1.5. Висновки до першого розділу

Аналіз існуючих методів та технологій виявлення ЗПЗ та поліморфних вірусів зокрема, дозволив зробити наступні висновки:

1. У 2024-2025 роках спостерігатиметься значне збільшення кількості атак ЗПЗ та даний показник становитиме 7,5 та 10,9 мільярдів відповідно, що вимагає удосконалення методів виявлення та знешкодження ЗПЗ.

2. Поліморфні віруси є одним із найдосконаліших типів загроз, які є досить складними для виявлення та аналізу.

3. Поліморфні віруси зазвичай класифікують за рівнями поліморфізму і таких рівнів шість визначено за певною класифікацією.

4. Аналітики використовують різноманітні інструменти для виявлення ЗПЗ, для захисту та прогнозування майбутніх атак, але кожен з них не є досконалим.

5. Найбільш поширеними методами виявлення поліморфних вірусів є метод пошуку ЗПЗ, методи аналізу даних, аналіз ЗПЗ в пісочниці, алгоритми машинного навчання, розробка структурних функцій.

6. Недоліки розглянутих методів вимагають нових підходів до виявлення та аналізу ЗПЗ, серед яких - виявлення ЗПЗ за допомогою імовірнісних логічних мереж (PLN), а також їх комплексне використання.

7. Аналіз існуючих мультиагентних систем виявлення ЗПЗ свідчить про їх недоліки для виявлення та класифікації поліморфних вірусів, що вимагає розробки гібридних мультиагентних систем на основі комплексу розглянутих методів виявлення поліморфних вірусів з урахуванням моделі Лотки-Вольтера для залучення різних методів виявлення на різних етапах дослідження виконуваних програм на наявність інфікування поліморфними вірусами.

Основні результати розділу опубліковані у працях [23, 24, 151, 152, 153, 154, 155].

РОЗДІЛ 2.
АРХІТЕКТУРА ГІБРИДНИХ МУЛЬТИАГЕНТНИХ СИСТЕМ ВИЯВЛЕННЯ ПОЛІМОРФНИХ ВІРУСІВ ТА МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОГО АГЕНТУ

2.1. Моделі класів поліморфних вірусів

Поліморфні віруси є однією з найбільш складних і небезпечних категорій зловмисних програм, які постійно еволюціонують, адаптуючись до нових методів виявлення та боротьби з ними. Їхня здатність змінювати свою архітектуру та кодову базу при кожному зараженні робить їх надзвичайно важким об'єктом для детекції традиційними антивірусними засобами. Виникає необхідність у математичній формалізації поліморфних вірусів, що дозволить розробляти більш ефективні методи для їх виявлення, класифікації та нейтралізації.

Отже, поліморфний комп'ютерний вірус - це ЗПЗ, яке здатне модифікувати свою форму (код) кожної ітерації, при якій воно заражає нову систему або виконується. Його динамічна поведінка ускладнює його виявлення традиційними методами. Розглянемо формалізацію поліморфного вірусу у вигляді моделі.

Поліморфний вірус задамо як п'ятірку:

$$V=(C, M_A, E, P_A, A), \tag{2.1}$$

де C - множина можливих кодів вірусу; M_A – множина алгоритмів мутації, що задають зміну вірусом свого коду:

$$M_A : C \times K \rightarrow C, \tag{2.2}$$

де K – множина ключів або параметрів мутації; E – множина алгоритмів виконання, що визначає дію вірусу, наприклад, знищення файлів, викрадення даних, поширення тощо; P_A – множина алгоритмів поширення, які визначають правила зараження інших файлів або систем; A – множина алгоритмів приховування, які змінюють поведінку вірусу залежно від детекторів.

В формулу (2.1) не включено типи виконуваних програм, оскільки будемо розглядати лише один тип виконуваних програм PE-файл (portable executable), на які спрямовані зловмисні впливи поліморфних вірусів.

Формалізуємо поведінку, яка властива поліморфному вірусу. Кожного разу, коли вірус інфікує виконувану програму, то його код змінюється за допомогою функції мутації $M_{A,f}$ так:

$$C_{i+1} = M_{A,f}(C_i, k_i), C_i \in C, k_i \in K, \quad (2.3)$$

де C_i - код вірусу на і-му кроці (до мутації); $i = 1,2, \dots, N_{C_i}$; N_{C_i} – кількість кроків зміни коду вірусу до мутації; k_i - випадковий ключ мутації, що може генеруватися на основі випадкових чисел або параметрів середовища.

Механізм мутації M має забезпечувати збереження функціональності вірусу після зміни форми:

$$E(C_i) = E(C_{i+1}), \quad (2.4)$$

де $E(C_i)$ - функція виконання вірусу, яка залишається незмінною.

Для кожного поліморфного вірусу функція E буде постійною, тобто згідно неї будуть виконуватись різні дії з мутації, але вона сама безпосередньо за своїми кроками буде їх повторювати кожного разу при виклику. Це важлива характеристика поліморфних вірусів, яка забезпечує їх стійкість. Звичайно, для різних поліморфних вірусів вона буде різною за синтаксисом, але суть її семантики буде незмінюваною.

Поширення вірусу задамо ймовірнісною множиною функцій P_A, що визначає потенційні заражені виконувані програми або системи, так:

$$P_A : F \times S \rightarrow \{0,1\}, \quad (2.5)$$

де F - множина виконуваних програм, які доступні для зараження; S - множина станів системи (наприклад, рівень доступу).

Згідно формули (2.5) поширення поліморфного вірусу в системі через інфікування виконуваних програм будемо оцінювати ймовірністю зараження виконуваних програм. Тобто, конкретний поліморфних вірус поширюється в системі і його поширення буде оцінено через ймовірності зараження виконуваних програм.

Якщо буде поширюватись в системі більше, ніж один поліморфний вірус, тоді ймовірність для кожної виконуваної програми буде визначатись для кожного вірусу окремо. Але як правило одна виконувана програма не інфікується двома різними поліморфними вірусами. Для розглядуваної моделі поширення поліморфних вірусів вважатимемо допустимим таку подію, тоді ймовірність інфікування однієї виконуваної програми для двох різних поліморфних вірусів приймемо як дві різні ймовірності. Це надасть змогу відокремити поліморфні віруси як різні.

Задамо залежність ймовірності зараження виконуваних програм $f \in F$ від стану системи $s \in S$ так:

$$P(f, s) = P, \tag{2.6}$$

де P - імовірність успішного зараження виконуваної програми; F – множина виконуваних програм; S – множина станів системи.

Стан усієї системи в поточний момент часу суттєво впливає на процес зараження виконуваних програм. Тобто, при певному стані ймовірність зараження виконуваної програми висока, а при іншому – низька. Стан системи може характеризуватись різними складовими і бути комплексною характеристикою. Тоді, наприклад, засоби виявлення поліморфних вірусів, які можуть бути одним із параметрів комплексної характеристики, в певний момент часу не виконують свого завдання, і в такому випадку ймовірність зараження виконуваної програми висока. А коли виконують, то ймовірність зараження виконуваної програми низька.

Задамо множину функцій A_f для характеристики типу алгоритму приховування так:

$$A_f : C \times D \rightarrow C, \tag{2.7}$$

де D - набір характеристик детекторів (антивірусних алгоритмів); C - множина можливих кодів вірусу.

Метою функцій приховування з множини функцій A_f є модифікація коду виконуваних програм з множини C так, щоб уникнути виявлення. Тоді, задамо таке відношення як цільову функцію так:

$$P_v(C_i) \rightarrow min, \tag{2.8}$$

де P_v - ймовірність виявлення коду C_i.

Задамо алгоритм виконання E, який визначає дію вірусу, так:

$$E : C \rightarrow D_A, \tag{2.9}$$

де D_A – множина дій згідно алгоритму і може включати такі дії: знищення файлів, викрадення даних, виконання зловмисного коду в системі.

Виділення дій алгоритму окремо в окрему множину потрібно в контексті того, що дії та їх кількість по суті є детермінованими, а різні алгоритми можуть вимагати виконання дій саме з цієї множини і ці дії будуть в них повторюватись. Це дає змогу деталізувати потенційні дії поліморфних вірусів з метою їх виявлення за проявами цих дій.

Здійснимо розроблення динамічної моделі поліморфного вірусу. В її основу приймемо загальний процес роботи поліморфного вірусу. Він переважно може містити такі кроки: ініціалізація; поширення; мутація; приховування; виконання. Задамо відношення та процеси для кожного з кроків окремо так:

1) ініціалізація базована на початковому коді вірусу C_0, який розроблено зловмисником як унікольний код або синтезовано згідно наявних у відкритому доступі модулів та функцій;

2) поширення (формула (2.6)) поліморфних вірусів пов’язане з кожною доступною виконуваною програмою f ∈ F, для встановлення відношення між якими задамо співвідношення так:

$$if\ P(f, s)=1,\ than\ f\ \overleftarrow{P_A}\ \mathrm{C}, \tag{2.10}$$

де P_A – множина алгоритмів поширення, які визначають правила зараження інших файлів або систем; C - множина можливих кодів вірусу; P - імовірність успішного зараження виконуваної програми; F – множина виконуваних програм; S – множина станів системи; $f \in F;\ s \in S;$

3) мутація виконується після кожного зараження виконуваних програм (формула (2.3));

4) приховування (формула (2.7)) необхідне поліморфним вірусам для мінімізації ймовірності виявлення і воно здійснюється через зміну його коду C_{i+1} так:

$$C_{i+1}=A_f(C_i, D), \tag{2.11}$$

де C_i - код вірусу на i-му кроці (до мутації); $i = 1,2, \dots, N_{C_i}$; N_{C_i} – кількість кроків зміни коду вірусу до мутації; D - набір характеристик детекторів (антивірусних алгоритмів); A_f - множина функцій для певного типу алгоритму приховування;

5) виконання здійснюється через функцію $E(C_{i+1})$ при активації вірусу (формула (2.9)).

Таким чином, формалізовано та встановлено співвідношення для ініціалізації, поширення, мутації, приховування та виконання поліморфних вірусів згідно їх задання за формулою (2.1). Така деталізація дає змогу відобразити поведінку поліморфних вірусів в операційному середовищі для її ідентифікації при їх виявленні. Крім того, потребують деталізації властивості поліморфних вірусів, які впливають на їх поділ на підмножини з метою уточнення відношення до певного класу та покращення точності виявлення в цілому.

Виділимо такі властивості поліморфних вірусів:

1) властивість функціональної інваріантності, яка означає, що мутації зберігають функціональність вірусу (формула (2.4)):

$$\forall i,\ E(C_i) = \text{незмінне;} \tag{2.12}$$

2) ентропія коду характеризує те, що кожна мутація збільшує різноманітність коду, що ускладнює його виявлення:

$$H(C_{i+1}) > H(C_i), \tag{2.13}$$

де $H(C_i)$ - ентропія коду; C_i - код вірусу на i-му кроці (до мутації); $i = 1,2, \dots, N_{C_i}$; N_{C_i} – кількість кроків зміни коду вірусу до мутації;

3) ймовірність виявлення характеризується рівнем приховування та ймовірністю виявлення коду P_v, причому справедливе таке співвідношення:

$$P_v(C_{i+1}) < P_v(C_i), \tag{2.14}$$

де P_v - ймовірність виявлення коду C_i.

Властивості поліморфних вірусів, які задано за формулами (2.12)-(2.14) дають змогу характеризувати їх класи та особливості щодо мутації.

Розглядатимемо поліморфні віруси в класифікації, яка передбачає поділ на 6 рівнів поліморфізму, деталізація яких подана в роботі [92]. Розробимо моделі класів усіх рівнів поліморфізму з метою деталізації особливостей поліморфних вірусів для виокремлення в них унікальних особливостей для їх використання при виявленні чи ідентифікації.

Віруси першого рівня поліморфізму використовують постійні значення для різних розшифровувачів. Їх можна виявити за деякими постійними ділянкам коду розшифровувача.

Приймемо модель поліморфних вірусів кортежем:

$$P_1 = (B, A, D, H, K, \xi, \sigma, \rho, R), \tag{2.15}$$

де B – сукупність інструкцій користувацької програми чи системного процесу, вірогідно інфікованих поліморфним вірусом, $B = \{b_i \mid i=1,2, \ldots, n\}$; A – сукупність алгоритмів розшифровування присутніх у поліморфному вірусі, $A = \{a_i \mid i=1,2 \ldots, x\}$; D – сукупність інструкцій вірусу та розшифровувача, $D = \{d_i \mid i=1,2 \ldots, y\}$; H – сукупність інструкцій вірусу для вибору одного з наявних у вірусі алгоритмів розшифровування, $H = \{h_i \mid i=1,2, \ldots, j\}$; K – сукупність команд, які є основним кодом вірусу, $K = \{k_i \mid i=1,2 \ldots, v\}$; ξ - функція вибору a_i розшифровувача, $\xi: H \to A$; σ – функція створення зловмисних команд (тіла вірусу) шляхом виконання команд $d_{a_i} \in D$ розшифровувача a_i, $\sigma: D_{x_i} \to K$; ρ – функція створення поведінки поліморфного вірусу R шляхом вкорінення тіла вірусу K в команди програми В, $\rho: B \times K \to R$.

Таким чином, поліморфні віруси мають поведінку, що формується з певної послідовності команд. Базуючись на цьому можна побудувати поведінку вірусу у вигляді послідовності.

Поведінка вірусу R_1^A першого рівня поліморфізму, який утворений вкоріненням зловмисних команд U в команди програми A, і поведінка вірусу R_1 утворена без вкорінення зловмисних команд U в команди програми A задамо послідовностями:

$$R_1^A = g_{\phi_{x_\varepsilon}} \dots g_{\eta_{x_\varepsilon}}\, a_1 \dots a_n\, u_1 \dots\, u_w; \tag{2.16}$$

$$R_1 = g_{\phi_{x_\varepsilon}} \dots g_{\eta_{\chi_\varepsilon}}\, u_1 \dots u_w, \tag{2.17}$$

де значення ϕ, η визначають, що можливі вірусні команди розшифровувача $g_{\phi_{\chi_\varepsilon}} \dots g_{\eta_{\chi_\varepsilon}}$ можуть бути різними для різних розшифровувачів x_ε , ε – номер обраного розшифровувача.

До другого рівня поліморфізму відносять віруси, розшифровувачі яких мають постійну одну або декілька інструкцій. Наприклад, він може використовувати різні регістри, деякі альтернативні інструкції в розшифровувачі. Такі віруси також можна розпізнати за визначеною сигнатурою в розшифровувачі. Приймемо модель поліморфних вірусів другого рівня кортежем:

$$P_2 = (B, L, K, \rho, \varepsilon, R), \tag{2.18}$$

де B – сукупність інструкцій користувацької програми чи системного процесу, вірогідно інфікованих поліморфним вірусом, $B = \{b_i \mid i=1,2, \dots, n\}$; L – сукупність інструкцій, які складають код розшифровувача, $L = \{l_i, \mid i=1,2, \dots, x\}$; K – сукупність команд, які є основним кодом вірусу, $K = \{k_i \mid i=1,2 \dots, v\}$; ρ – функція створення поведінки поліморфного вірусу R шляхом вбудовування тіла вірусу K в програму B, $\rho: B \times K \to R$; ε – функція створення інструкцій основного коду вірусу шляхом вибору команд розшифровувача, $\varepsilon: L \to K$.

Поведінки вірусу другого рівня поліморфізму R_2^A і R_2 можна представити у вигляді таких послідовностей:

$$R_2^A = e_k \dots e_\lambda\, a_1 \dots a_n\, u_1 \dots u_w; \tag{2.19}$$

$$R_2 = e_k \dots e_\lambda\, u_1 \dots u_w, \tag{2.20}$$

де значення κ, λ визначають, що можливі вірусні команди розшифровувача $e_k \dots e_\lambda$ можуть бути різними при кожному запуску вірусу.

Поліморфні віруси, що використовують в розшифровувачі інструкції, які не мають ролі в розшифруванні вірусного коду, чи «команди-сміття», відносять до

третього рівня поліморфізму. Ці віруси, також, можна виявити за допомогою сталої сигнатури, якщо провести фільтрацію та видалення команд-сміття.

Поліморфні віруси четвертого рівня використовують в розшифровувачі взаємозамінювані інструкції без зміни алгоритму розшифровування.

Віруси четвертого рівня використовують в розшифровувачі взаємозамінювані інструкції без зміни алгоритму розшифрування.

Формальна математична модель поліморфних вірусів третього та четвертого рівнів визначається сукупністю:

$$P_{3,4} = (B, L, K, T, \varphi, \phi\ , R), \tag{2.21}$$

де B – сукупність інструкцій користувацької програми чи системного процесу, вірогідно інфікованих поліморфним вірусом, $B = \{b_i \mid i=1,2, \ldots, n\}$; L – сукупність інструкцій, які складають код розшифровувача, $L = \{l_i, \mid i=1,2, \ldots, x\}$; K – сукупність команд, які є основним кодом вірусу, $K = \{k_i \mid i=1,2 \ldots, v\}$; T – множина інструкцій, які називаються «команди-сміття», $T = \{t_i \mid i = 1,2, \ldots, y\}$; φ – функція створення основного коду вірусу засобами розшифровувача, який вбудовує «команди-сміття» в сукупність зловмисних команд, $\varphi: L \times T \to K$; ϕ – функція створення поведінки поліморфного вірусу R шляхом вбудовування тіла вірусу K в користувацьку програму B, $\phi = B \times K \to R$.

Поведінка вірусів третього та четвертого рівнів поліморфізму R_3^A та R_4^A, які утворені засобами розшифровувача, що інтегрує «команди-сміття» в зловмисні команди та вставляє зловмисні команди U і поведінки вірусу в команди програми A, то поведінку поліморфних вірусів R_3 та R_4 утворена без вкорінення зловмисних команд U в команди програми A можна задати такими послідовностями:

$$R_3^A = e_1 \ldots e_\theta a_1 \ldots a_n u_1 b_\rho \ldots u_w b_\xi; \tag{2.22}$$

$$R_3 = e_1 \ldots e_\theta u_1 b_\rho \ldots u_w b_\xi; \tag{2.23}$$

$$R_4 = e_1 \ldots e_\theta u_\vartheta b_\rho \ldots u_\sigma b_\xi; \tag{2.24}$$

$$R_4^A = e_1 \ldots e_\theta a_1 \ldots a_n u_\vartheta b_\rho \ldots u_\sigma b_\xi, \tag{2.25}$$

де значення $\rho, \xi, \vartheta, \sigma$ визначають, що можливі «команди-сміття» і команди поліморфних вірусів $u_\vartheta b_\rho \dots u_\sigma b_\xi$ можуть бути різними для кожного нового запуску вірусу.

П’ятий рівень поліморфізму включає в себе властивості всіх перерахованих рівнів і розшифровувач може використовувати різні алгоритми розшифровування вірусного коду.

Формальна математична модель поліморфних вірусів п’ятого рівня визначається сукупністю:

$$P_5 = (B, A, H, K, T, G, \xi, \theta, \eta, R), \tag{2.26}$$

де B – сукупність інструкцій користувацької програми чи системного процесу, вірогідно інфікованих поліморфним вірусом, $B = \{b_i \mid i=1,2, \dots, n\}$; A – сукупність алгоритмів розшифровування присутніх у поліморфному вірусі, $A = \{a_i \mid i=1,2 \dots, x\}$; H – сукупність інструкцій вірусу для вибору одного з наявних у вірусі алгоритмів розшифровування, $H = \{h_i \mid i=1,2, \dots, j\}$; K – сукупність команд, які є основним кодом вірусу, $K = \{k_i \mid i=1,2 \dots, v\}$; T – множина інструкцій, які називаються «команди-сміття», $T = \{t_i \mid i = 1,2, \dots, y\}$; G – сукупність інструкцій вірусу a_i та розшифровувача, $G = \{g_i | i = 1,2, \dots, j\}$; ξ - функція вибору a_i розшифровувача, $\xi: H \to A$; θ – функція створення зловмисних інструкцій засобами вибору a_i розшифровувача інструкціями $g_{x_i} \in G$ і створення чіткого порядку їх виконання, $\theta: T \in G_{x_i} \to K$; η – функція створення поведінки поліморфного вірусу R шляхом вбудовування зловмисних команд K в команди програми B, $\eta: B \times K \to R$.

Поведінки поліморфних вірусів п'ятого рівня поліморфізму R_5^A та R_5 можна задати послідовностями так:

$$R_5^A = g, \tag{2.27}$$

де значення ρ, ξ визначають, що можливі вірусні команди розшифровувача $g_{\phi_{x_\varepsilon}} \dots g_{\eta_{x_\varepsilon}}$ можуть бути різними для різних розшифровувачів x_ε; ϑ - номер обраного розшифровувача; значення $\rho, \xi, \vartheta, \sigma$ визначають, що можливі «команди-сміття» і команди вірусу $u_\vartheta b_\rho \dots u_\sigma b_\xi$ можуть бути різними для кожного нового запуску вірусу.

Шостий рівень поліморфізму вірусів характеризується нешифрованими частинами, що складаються з підпрограм, які «перемішуються» всередині тіла вірусу. Такі віруси називаються - пермутуючі віруси.

Формальна модель поліморфних вірусів шостого рівня визначається сукупністю:

$$P_6 = (B, L, K, \vartheta, R), \tag{2.28}$$

де B – сукупність інструкцій користувацької програми чи системного процесу, вірогідно інфікованих поліморфним вірусом, $B = \{b_i \mid i=1,2, \ldots, n\}$; L – сукупність інструкцій, які складають код розшифровувача, $L = \{l_i, \mid i=1,2, \ldots, x\}$; K – сукупність команд, які є основним кодом вірусу, $K = \{k_i \mid i=1,2 \ldots, v\}$; ϑ – функція створення поведінки поліморфного вірусу R шляхом розміщення інструкцій коду, інструкцій розшифровування, інструкцій тіла вірусу блоками за заданим алгоритмом порядком, $\vartheta: B \times L \times K \to R$.

Поведінки поліморфних вірусів шостого рівня поліморфізму R_6^A та R_6 можуть бути задані такими послідовностями:

$$R_6^A = a_1, \tag{2.29}$$

де значення $\phi, \eta, \vartheta, \sigma$ визначають, що можливі команди розшифровувача та зловмисні команди $e_\phi \ldots e_\eta \, u_\vartheta \ldots u_\sigma$ можуть бути різними для кожного нового запуску поліморфних вірусів.

Таким чином, розроблено моделі класів поліморфних вірусів на основі властивої їм поведінки в процесі функціонування та з урахуванням їх властивостей (узагальнення на рис. 2.1). Розроблені моделі класів поліморфних вірусів згідно рівнів складності (6 рівнів) для забезпечення їх виявлення та ідентифікації згідно їх унікальних особливостей, що відображені в моделях. Даний підхід до визначення класів поліморфних вірусів є досить важливим та є основою для обрання методу їх класифікації. Проте, не менш важливим є питання виявлення поліморфних вірусів та визначення, які та скільки методів їх виявлення у системному їх поєднанні є ефективним. Для забезпечення їх поділу за класами необхідно розробити модель класифікації поліморфних вірусів згідно із рівнями складності.

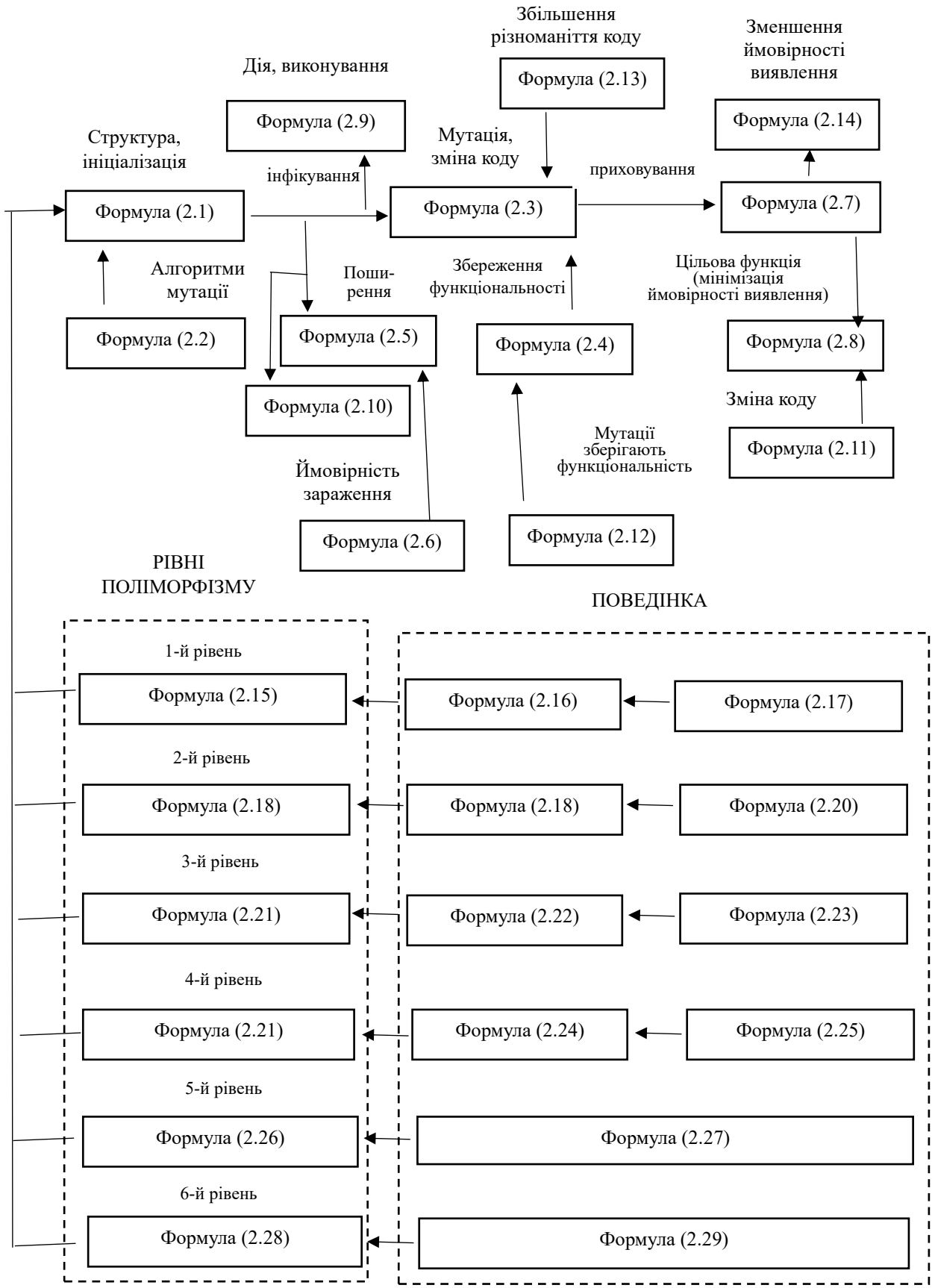

Рисунок 2.1 – Моделі класів поліморфних вірусів на основі властивої їм поведінки в процесі функціонування, з урахуванням їх властивостей і складності будови

2.2. Класифікація поліморфних вірусів за рівнями складності

Для класифікації поліморфних вірусів згідно рівня складності, доцільним є здійснення нечіткої класифікації на основі використання нечіткого логічного висновку. Нечіткий логічний висновок є апроксимацією залежності «входи-вихід» за допомогою лінгвістичних виразів типу <якщо-тоді> та логічних операцій над нечіткими множинами. Цей метод використовується для моделювання об'єктів із неперервним або дискретним виходом. Зазвичай в архітектурі системи нечіткого логічного висновку містяться такі модулі:

- фазифікатор, який перетворює фіксований вектор факторів, що впливають (X) у вектор нечітких множин $\tilde{X}$, необхідних для нечіткого висновку;

- нечітка база знань, яка містить інформацію про залежність $Y = f(X)$ у вигляді лінгвістичних правил <якщо-тоді>;

- функції приналежності, які використовуються для представлення лінгвістичних термів у вигляді нечітких множин;

- машина нечіткого логічного висновку, котра на основі правил бази знань визначає значення вихідної змінної у вигляді нечіткої множини $\tilde{Y}$, яка відповідає нечітким значенням вхідних змінних ($\tilde{X}$);

- дефазифікатор, який перетворює вихідну нечітку множину $\tilde{Y}$ в чітке число Y.

Нечіткою базою знань називається сукупність нечітких правил <якщо-тоді>, які задають взаємозв’язок між входами та виходами досліджуваного об’єкту. Формат нечітких правил такий:

якщо <антецедент правила>, тоді <консеквент правила>.

Алгоритми нечіткого висновку розрізняються в основному виглядом правила нечіткої імплікації, яке використовується. Якщо, наприклад, базу знань організують два нечітких правила виду:

- $П_1$: якщо $x \in A_1$ та $y \in B_1$, тоді $z \in C_1$;

- $П_2$: якщо $x \in A_2$ та $y \in B_2$, тоді $z \in C_2$,

де x та y – імена вхідних змінних; z – ім’я змінної виведення; $A_1, A_2, B_1, B_2, C_1, C_2$ –

деякі нечіткі множини, задані функціями належності $\mu_{A1}(x)$, $\mu_{A2}(x)$, $\mu_{B1}(y)$, $\mu_{B2}(y)$, $\mu_{C1}(z)$, $\mu_{C2}(z)$, при цьому чітке значення *z0* необхідно визначити на основі приведеної інформації і чітких значень *x0, y0.*

Отже, будемо використовувати нечіткий логічний висновок Мамдані, тому що у ньому найбільш прозоро задаються значення змінних нечіткими термами та найкраще інтерпретуються. Результати нечіткого висновку Мамдані традиційно дефазифікуються за методом центру тяжіння. Алгоритм Мамдані задається так:

1) введення нечіткості, для якої знаходяться ступені істинності для передумов кожного правила $\mu_{A1}(x0)$, $\mu_{A2}(x0)$, $\mu_{B1}(y0)$, $\mu_{B2}(y0)$;

2) нечіткий висновок, при якому знаходяться рівні "відсікання" для передумов кожного з правил (з використанням операції МІНІМУМ):

$$\begin{aligned} \alpha_1 &= \mu_{A_1}(xo) \wedge \mu_{B_1}(yo), \\ \alpha_2 &= \mu_{A_2}(xo) \wedge \mu_{B_2}(yo), \end{aligned} \tag{2.30}$$

де $\mu_{A_1}(xo)$ - ступінь належності *x0* до нечіткого підмножини A_1; $\mu_{B_1}(yo)$ - ступінь належності *y0* до нечіткого підмножини B_1; $\wedge$ означає операцію «і» за допомогою мінімуму; $\mu_{A_2}(xo)$ - ступінь належності *x0* до нечіткого підмножини A_2; $\mu_{B_2}(yo)$- ступінь належності *y0* до нечіткого підмножини B_2;

нехай задано

- $П_1$: якщо $x \in A_1$ та $y \in B_1$, тоді $z \in C_1$;
- $П_2$: якщо $x \in A_2$ та $y \in B_2$, тоді $z \in C_2$,

де через «∧» позначено операцію логічного мінімуму (min), а потім знаходяться «усічені» функції належності так:

$$\begin{aligned} \mu_{C\prime 1} &= (\alpha_1 \wedge \mu_{C_1}(z)), \\ \mu_{C\prime 1} &= (\alpha_2 \wedge \mu_{C_2}(z)); \end{aligned} \tag{2.31}$$

де $\mu_{C_1}(z)$ - функція належності вихідної змінної за першим правилом; $\mu_{C\prime 1}$ - її усічена версія, тобто значення не вище, ніж α_1;

продовжувати так далі для всіх правил;

3) композиція, при якій відбувається об'єднання знайдених усічених функцій з використанням операції МАКСИМУМ (max, позначена далі як "v"), що приводить до

отримання підсумкової нечіткої підмножини для змінної виходу з функцією належності:

$$\mu_C(z) = \mu_{C'1}(z) \vee \mu_{C'2}(z) = (\alpha_1 \wedge \mu_{C1}(z)) \vee = (\alpha_2 \wedge \mu_{C2}(z)), \quad (2.32)$$

де $\mu C(z)$ - підсумкова функція належності для вихідної змінної z після об'єднання всіх правил; $\mu_{C'1}(z)$, $\mu_{C'2}(z)$ - усічені функції належності для першого і другого правила відповідно; α_1 та α_2 - рівні відсікання для першого і другого правила, які обчислюються через операцію МІНІМУМ;

4) приведення до чіткості (для знаходження *z0*) проводиться у випадку пропонованого рішення центроїдним методом (чітке значення вихідної змінної визначається як центр тяжіння для кривої $\mu_{\sum}(z)$), тобто:

$$z0 = \frac{\int z \cdot \mu_{\Sigma}(z) dz}{\int \mu_{\Sigma}(z) dz}, \quad (2.33)$$

де $\mu_{\Sigma}(z)$ або просто $\mu_{\mathrm{C}}(z)$ - це підсумкова функція належності, яку ти отримав після об'єднання всіх усічених функцій на попередньому кроці; чисельник $\int z \cdot \mu_{\Sigma}(z) dz$ - це «момент відносно початку координат» (тобто сума всіх z, зважених на їхню належність); знаменник $\int \mu_{\Sigma}(z) dz$ - це «площа під кривою» (тобто загальна вага).

У цьому випадку база знань про вплив характеристик досліджуваного вірусу X = (x_1,x_2,…x_n) та висновок стосовно його належності до певного рівня складності поліморфних вірусів D задамо таким чином:

якщо $x_1 = a_1^{j1}$ та $x_2 = a_2^{j1}$ та… та $x_n = a_n^{j1}$ з вагою w_{j1}, або

$x_1 = a_1^{j2}$ та $x_2 = a_2^{j2}$ та… та $x_n = a_n^{j2}$ з вагою w_{j2}, або

… або

$x_1 = a_1^{jk_j}$ та $x_2 = a_2^{jk_j}$ та… та $x_n = a_n^{jk_j}$ з вагою w_{jk_j},

тоді $D = d_j, j = \overline{1,m}$,

де a_i^{jp} - нечіткий терм, що оцінює значення фактора x_i в правилі з номером jp, $j = \overline{1,m}$, $p = \overline{1,k_j}$, $I = \overline{1,n}$; d_j – класи (належність до певного рівня складності поліморфних вірусів) $j = \overline{1,m}$; m – кількість можливих класів оцінки; k_j – число правил, в яких $D = d_j$, $j = \overline{1,m}$; $w_{jp} \in [0,1]$ – ваговий коефіцієнт, який відповідає впевненості експерта в

достовірності правила з номером *jp*, $j = \overline{1,m}$, $p = \overline{1,k_j}$.

За допомогою операцій ⋃ (АБО) і ⋂ (ТА) нечітку базу запишемо у компактній формі:

$$\bigcup_{p=1}^{k_j}[\bigcap_{i=1}^{n}(x_i = a_i^{jp}) \text{ з вагою } w_{jp}] \rightarrow D = d_j, \quad j = \overline{1,m}. \tag{2.34}$$

де p=1,…,k_j - індексує правила для одного висновку d_j (може бути кілька шляхів отримання одного висновку); i=1,…,n – індексує вхідні фактори; $(x_i = a_i^{jp})$ - зв'язок між i-м фактором та передумовою a_i^{jp} у p-му правилі для висновку d_j; w_{jp} - вага p-го правила для висновку d_j; ⋂ (ТА) - перетин усіх передумов в одному правилі; ⋃ (АБО) - об'єднання всіх правил, що ведуть до одного висновку d_j; $\rightarrow D=d_j$ - висновок: якщо виконуються передумови, тоді вихід D дорівнює d_j.

Наведену вище нечітку базу знань часто представляють таблицею, яку ще називають матрицею знань.

Нечіткий висновок проводиться по системі нечітких логічних рівнянь. Вони будуються по нечітких базах знань. Для переходу від нечіткої бази знань до системи нечітких логічних рівнянь введемо такі позначення: $\mu^{a_i^{jp}}(x_i)$ - функція належності змінної x_i до нечіткого терма a_i^{jp} (j = $\overline{1,m}$, p = $\overline{1,k_j}$, i = $\overline{1,n}$); $\mu^{d_j}(x_1, x_2, \ldots, x_n)$ - функція належності вектору параметрів (x_1,x_2, ...x_n) до рішення d_j, $j = \overline{1,m}$.

Зв’язок між цими функціями представляється ізоморфною до нечіткої бази знань системою нечітких логічних рівнянь:

$$\mu^{d_j}(x_1, x_2, \ldots, x_n) = \vee_{p=1}^{k_j}\left(w_{pj} \cdot \wedge_{i=1}^{n}\mu^{a_i^{jp}}(x_i)\right), \text{ j}=\overline{1,m}, \tag{2.35}$$

де $\mu^{d_j}(x_1, x_2, \ldots, x_n)$ - функція належності для j-го висновку залежно від усіх вхідних змінних $x_1, x_2, \ldots, x_n$; p=1,…,k_j – індекс по різних правилах, які ведуть до одного і того ж висновку d_j; w_{pj} - вага p-го правила, яке формує j-й висновок; $\wedge_{i=1}^{n}\mu^{a_i^{jp}}(x_i)$ - мінімум функцій належності передумов p-го правила для всіх входів; $\mu^{a_i^{jp}}(x_i)$ - функція належності i-го вхідного фактору x_i до нечіткого поняття a_i^{jp} у p-му правилі для висновку d_j; $\vee_{p=1}^{k_j}$- максимум серед всіх p-правил, які ведуть до одного й того ж

висновку d_j.

Підставляючи у (2.35) вектор $X = (x_1,x_2,...x_n)$ значень факторів впливу, отримуємо таку нечітку множину вихідної змінної:

$$\tilde{y} = \left(\frac{\mu^{d_1}(X)}{d_1}, \frac{\mu^{d_2}(X)}{d_2}, \ldots, \frac{\mu^{d_m}(X)}{d_m}\right), \quad (2.36)$$

де $\tilde{y}$ – нечітка множина вихідної змінної (результат обчислення для заданого X); $\mu^{d_1}(X)$ - ступінь активності (ступінь істинності) j-го правила для заданого вектора X; d_j – конкретне чітке значення виходу, яке відповідає j-му правилу; $\frac{\mu^{d_m}(X)}{d_m}$ - запис пари (ступінь належності / значення).

Як розв'язок обирається розв'язок з максимальним ступенем належності:

$$y = \underset{\{d_1,d_2,\ldots,d_m\}}{arg\ max}\left(\mu_{d_1}(X), \mu_{d_2}(X), \ldots, \mu_{d_m}(X)\right), \quad (2.37)$$

де $d_1,d_2,...,d_m$ - це можливі значення вихідної змінної D; $\mu_{d_m}(X)$ - ступінь належності вихідного значення d_j відповідно до результатів обчислення нечіткого висновку для вхідного вектора X; arg max – оператор, який обирає той d_j, для якого значення $\mu_{d_m}(X)$ найбільше серед усіх.

Далі побудовано систему нечітких логічних рівнянь, перед формуванням якої необхідно врахувати такі особливості баз знань: наявність правил у "неповному" форматі та рівність одиниці вагових коефіцієнтів усіх правил. В неповних правилах довільне значення відповідних змінних не змінює істинності правил, тому вилучено функції належності цих змінних з логічних рівнянь. Значення вагових коефіцієнтів усіх правил дорівнює одиниці, тому для зручності вагові коефіцієнти було вилучено з усіх логічних рівнянь.

Після розв'язання рівнянь отримано ряд термів – результатів висновку за кожним правилом. За допомогою операції мінімуму обчислено значення вихідних термів та об'єднаємо функції належності за допомогою операції максимум.

Отже, класифікація виявлених поліморфних вірусів (нечітка класифікація) складається з кроків, які відображені схемою на рис. 2.2.

Отже, необхідно здійснити нечітку класифікацію поліморфних вірусів згідно рівнів складності із використанням нечіткого логічного висновку. Дана модель передбачає виконання наступних кроків: визначення характеристик виявлених поліморфних вірусів та формування дерева логічного висновку; опис лінгвістичних змінних; визначення функцій належності лінгвістичних термів; формування бази знань системи нечіткого висновку; отримання ймовірності належності досліджуваного файлу до поліморфних вірусів різних рівнів складності; нечітка класифікація поліморфних вірусів.

Для виявленні та класифікації поліморфних вірусів необхідно розглянути модель інтелектуального агента.

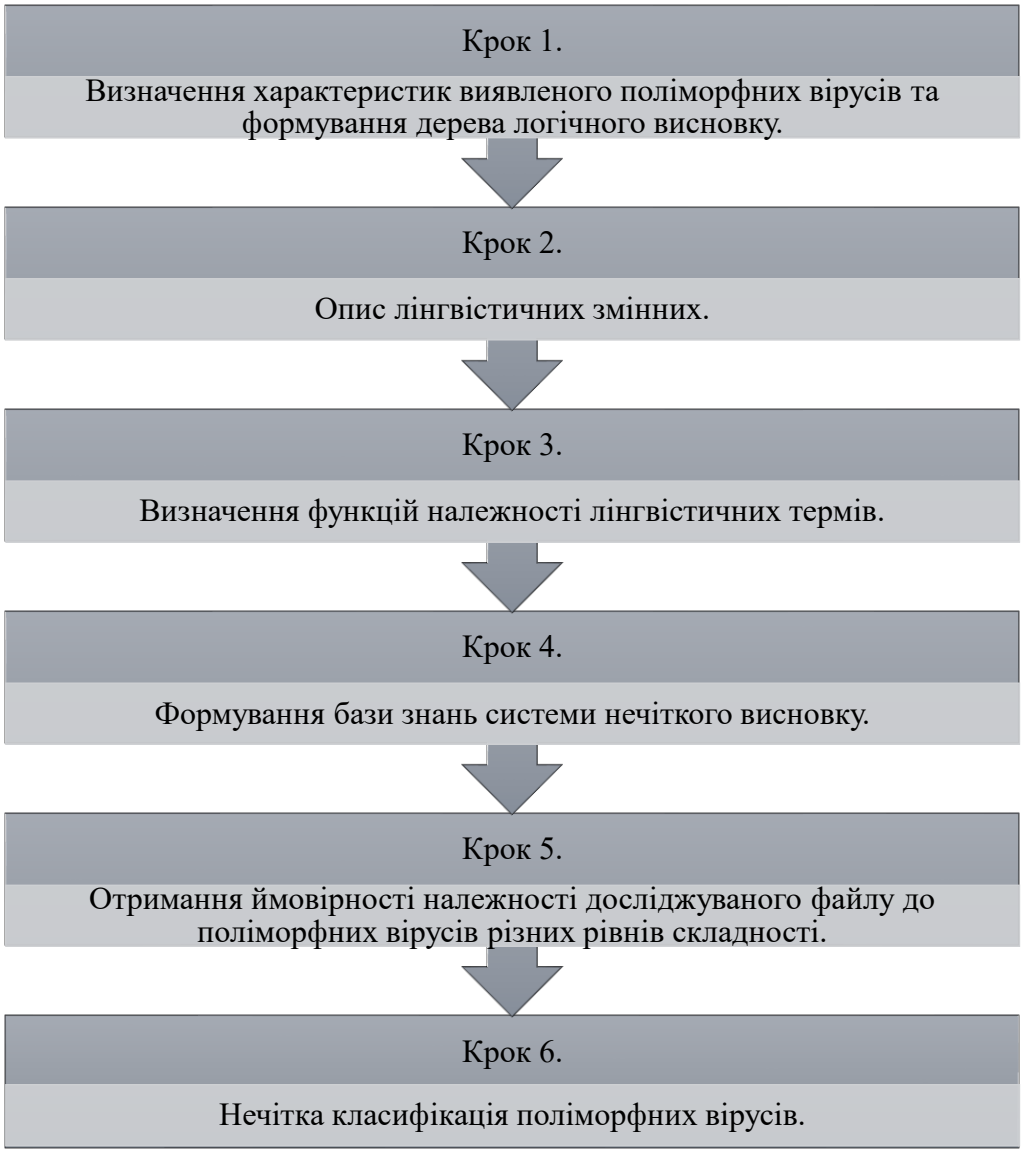

Рисунок 2.2 - Алгоритм нечіткої класифікації поліморфних вірусів

2.3. Модель інтелектуального агента

Інтелектуальний агент (ІА) для виявлення поліморфних вірусів є необхідним інструментом для забезпечення високого рівня кібербезпеки. Його здатність адаптуватися, швидко реагувати, ефективно виявляти нові загрози і мінімізувати хибні спрацьовування робить його надзвичайно важливим для протидії сучасними зловмисними програмами.

Перевагами використання ІА для виявлення та класифікації поліморфних вірусів є:

1) адаптивність до нових загроз, бо ІА мають здатність адаптуватися до нових типів вірусів, навіть якщо вони не були заздалегідь зафіксовані в базах даних сигнатур і замість того, щоб покладатися на старі сигнатури, агенти аналізують поведінку програм і можуть виявити нові, ще невідомі загрози, навіть якщо код вірусу змінюється;

2) поведінковий аналіз, бо віруси часто проявляються через нетипову або підозрілу поведінку в системі, а ІА може виявляти аномалії у поведінці програм, що може вказувати на наявність вірусу і це дозволяє виявити навіть віруси, які змінюють свою архітектуру або маскуються під легітимні програми;

3) самонавчання та постійне вдосконалення, бо ІА здатні до самонавчання за допомогою методів машинного навчання, тобто вони аналізують нові зразки ПЗ та оновлюють свої моделі для кращого виявлення нових загроз, що дає змогу адаптуватися до швидко змінюваного ландшафту кіберзагроз;

4) швидка реакція в реальному часі, бо ІА можуть миттєво реагувати на знайдені загрози і це може включати в себе ізоляцію заражених файлів, блокування зловмисних процесів або навіть автоматичне оновлення системи безпеки для запобігання подальшим атакам, що може забезпечити оперативність таких дій важлива для мінімізації шкоди;

5) мультиагентна взаємодія в системах, що використовують кілька агентів, коли кожен агент може виконувати різні функції (наприклад, сканування мережевого

трафіку, аналіз програм, моніторинг файлів), що дозволяє досягти більш високого рівня безпеки через паралельну роботу;

6) підвищення ефективності виявлення завдяки використанню евристичних методів і штучного інтелекту, бо ІА здатні виявляти не лише відомі віруси, але й нові варіанти, зокрема поліморфні або метаморфні віруси, які змінюють свою архітектуру, щоб уникнути традиційного виявлення;

7) мінімізація хибних спрацьовувань, оскільки ІА враховують контекст і поведінку програм, то вони здатні знижувати кількість хибних спрацьовувань порівняно з традиційними методами, такими як сигнатурні бази даних, що зменшує потребу в ручному втручанні користувачів або адміністраторів;

8) інтеграція з іншими системами безпеки, бо ІА можуть інтегруватися з іншими системами безпеки, наприклад, системами виявлення вторгнень (IDS) або системами управління безпекою інформації (SIEM), що дозволяє централізовано моніторити та реагувати на загрози в реальному часі.

9) зменшення навантаження на користувача та адміністраторів завдяки автоматизованому процесу виявлення та реагування на поліморфні віруси і, тому, інтелектуальний агент може значно зменшити потребу в постійному моніторингу системи людьми, дозволяючи їм зосередитися на важливіших задачах.

ІА I_a описується як система:

$$I_a = \langle S, E, P, A_t, U, M, L \rangle, \tag{2.38}$$

де S – множина можливих станів середовища (інформація про файли, процеси, мережеву активність тощо); $E : S \rightarrow O$ – функція сприйняття середовища, що перетворює стан середовища S у набір спостережень O; $P : O \rightarrow H$ – функція попередньої обробки спостережень, яка будує гіпотези H (наприклад, про наявність вірусної активності); A_t – множина можливих дій агента (сканування файлів, ізоляція, видалення ЗПЗ); $U : H \times A_t \rightarrow R$ – функція оцінки корисності дій у відповідь на певну гіпотезу; M - пам’ять агента, яка містить дані, правила або моделі (наприклад, сигнатури вірусів, історія атак); $L : M \times H \rightarrow M'$ - функція навчання, яка оновлює пам'ять M агента на основі досвіду.

Мета агента полягає у максимізації ефективності у пошуку і нейтралізації поліморфних вірусів. Формалізуємо її так:

$$arg\max_{A_t} U(H, A_t), \tag{2.39}$$

де A_t - дія (або стратегія дії) агента у момент часу *t; H* - поточний стан середовища (ситуація в мережі, поведінка об'єктів тощо); $U(H, A_t)$ – функція корисності (utility function), яка оцінює «вигідність» або «успішність» обраної дії A_t у ситуації *H*; arg max – означає, що потрібно знайти таку дію A_t, яка максимізує значення корисності *U*.

Процес роботи агента передбачає, що на кожному кроці t відбуваються такі події, які потребують активізації агенту:

1) агент отримує результати спостереження (сприйняття) в поточний момент часу, які формалізуємо так:

$$o_t = E(s_t), \tag{2.40}$$

де $s_t \in S$ – поточний стан середовища; o_t – значення результату спостереження в поточний момент часу *t;* $E : S \rightarrow O$ – функція сприйняття середовища (формула (2.38)), що перетворює стан середовища *S* у набір спостережень *O*;

2) розроблення гіпотези на основі аналізу визначимо так:

$$h_t = P(o_t), \tag{2.41}$$

де, наприклад, h_t - підозріла виконувана програма або підозрілий процес; $P : O \rightarrow H$ – функція попередньої обробки спостережень (формула (2.38)), яка будує гіпотези *H* (наприклад, про наявність вірусної активності);

3) прийняття рішення щодо вибору дій визначатимемо так:

$$a_t = arg\max_{a \in A_t} U(h_t, a), \tag{2.42}$$

де $U : H \times A_t \rightarrow R$ – функція оцінки корисності дій (формула (2.38)) у відповідь на певну гіпотезу;

4) навчання на основі оновлення моделі визначимо так:

$$M' = L(M, h_t), \tag{2.43}$$

де, наприклад, допустимо додавання нової сигнатури або шаблону поведінки вірусу; $L : M \times H \rightarrow M'$ - функція навчання (формула (2.38)), яка оновлює пам'ять M агента на основі досвіду;

навчання може бути визначено також так, якщо агент адаптує свою політику π і модель на основі отриманих нагород $R(s_t, a_t)$ з урахуванням, наприклад, оновлення функції цінності Q у методі Q-навчання:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_{a' \in A_t} Q(s_{t+1}, a') - Q(s_t, a_t) \right], \qquad (2.44)$$

де s_t - стан середовища в момент часу t, a_t - дія агента у стані s_t; $R(s_t,a_t)$ - нагорода за виконання дії a_t у стані s_t; s_{t+1} - новий стан після виконання дії a_t; A_{t+1} - множина можливих дій у новому стані s_{t+1}; $\alpha \in (0,1]$ - коефіцієнт навчання (learning rate), який визначає, наскільки сильно нова інформація змінює старі знання; $\gamma \in [0,1]$ - коефіцієнт дисконтування (discount factor), який визначає важливість майбутніх нагород.

5) виконання обраної дії a_t (наприклад, ізоляція чи сканування).

Таким чином, визначено основні події, що визначають процес функціонування ІА. Згідно визначених дій на основі введених виразів протягом всього часу функціонування ІА в операційному середовищі можна оцінювати та визначати його подальші дії з врахуванням певного накопиченого досвіду (формула (2.43)) та оцінювання наступних дій на основі гіпотез (формула (2.41)). Формули (2.40)-(2.43) та основні визначені події дають змогу розробляти ІА, які можуть вирішувати завдання з виявлення поліморфних вірусів. При їх застосуванні важливим є також оцінювання корисності роботи ІА.

Визначимо корисність роботи ІА так:

$$U(h,a) = P_v \cdot R_a - C_a, \qquad (2.45)$$

де $U : H \times A_t \rightarrow R$ – функція оцінки корисності дій (формула (2.38)) у відповідь на певну гіпотезу; P_v – ймовірність того, що загроза є вірусом; C_a – вартість виконання дії a (ресурсомісткість); R_a – очікувана вигода (захист системи від вірусу).

Поведінкову модель ІА задамо системою з такими основними компонентами наступним чином:

$$A = (S, O, A_t, R, T, \pi), \tag{2.46}$$

де $S = \{s_1, s_2, \ldots, s_n\}$ - множина станів середовища; $O = \{o_1, o_2, \ldots, o_m\}$ – множина спостережень агента; $A_t = \{a_1, a_2, \ldots, a_k\}$ – множина можливих дій агента в момент часу t; $R : S \times A_t \rightarrow R$ – функція нагороди, яка задає корисність дій для кожного стану; $T : S \times A_t \rightarrow S'$ - функція переходів, що описує зміну стану середовища внаслідок виконання дій; $\pi : O \rightarrow A_t$ – політика агента, що визначає правило вибору дій на основі спостережень.

Визначимо наступну політику ІА так:

1) детермінована політика:

$$\pi(o_t) = a_t,\ a_t \in A_t, \tag{2.47}$$

де o_t - спостереження агента у момент часу t (тобто часткова або повна інформація про стан середовища); A_t - множина можливих дій у момент часу t; a_t - конкретна дія, що однозначно визначається спостереженням o_t;

2) стохастична політика:

$$\pi(a|o_t) = P(a|o_t),\ \sum_{a \in A_t} \pi(a|o_t) = 1, \tag{2.48}$$

де $\pi(a|o_t)$ - ймовірність вибору дії a, коли агент спостерігає стан o_t; $P(a|o_t)$ - це саме ймовірність вибору дії a при спостереженні o_t; A_t - множина всіх можливих дій в стані o_t.

Виділимо наступні поведінкові моделі інтелектуального агента:

1) рефлексивний агент (прості правила), тобто коли агент миттєво реагує на спостереження без складного аналізу стану:

$$a_t = f(o_t); \tag{2.49}$$

де o_t - спостереження агента в момент часу t (стан середовища чи інші вхідні дані); a_t - дія, яку агент вибирає у відповідь на спостереження o_t; $f(o_t)$ - функція, яка однозначно визначає дію a_t на основі спостереження o_t;

2) агент на основі моделі середовища, тобто коли використовує модель переходів T для прогнозування наслідків своїх дій;

3) агент, що навчається, покращує політику π за допомогою функції нагород R:

$$\pi \to \pi', \text{де } \pi' = arg\max_{\pi} E[R], \tag{2.50}$$

де π - поточна політика агента; π' - нова політика агента після навчання; E[R] - очікуване значення нагороди, яке агент отримує від політики π.

Ця формалізація дозволяє описати і аналізувати поведінку інтелектуального агента для вирішення задач у динамічно змінюваному середовищі.

З метою визначення кількості та певних методів виявлення поліморфних вірусів потрібно використати ІА. Розглянемо модель наявності та поширення поліморфних вірусів $M_{n,v}$ у вигляді кортежу:

$$M_{n,v}=(\text{P, M, } \alpha, \beta, \gamma, \delta), \tag{2.51}$$

де $M_{n,v}$ – модель наявності та поширення поліморфних вірусів; P – початкова кількість поліморфних вірусів; M – кількість методів виявлення поліморфних вірусів, яка застосовується; α – ймовірність того, що кількість поліморфних вірусів зростатиме; β – ймовірність того, що поліморфні віруси різних рівнів складності будуть виявлені за допомогою обраних методів, технологій та інструментів; γ – ймовірність того, що деякі з обраних методів, технологій та інструментів не будуть ефективними при виявленні поліморфних вірусів різних рівнів складності в результаті появи нових їх різновидів; δ – ймовірність того, що поліморфні віруси різних рівнів складності вимагатимуть комплексного використання обраних методів, технологій та інструментів, а також новітніх підходів.

Показники α, β, γ, δ задаються експертами. При використанні експертного методу важливо визначити коефіцієнт узгодженості думок експертів (коефіцієнт конкордації) за загальноприйнятою формулою:

$$W = \frac{\sigma_{\phi}^2}{\sigma_{\max}^2} = \frac{\sum_{i=1}^{m}\left\{a_i - \frac{1}{2}n\cdot(m+1)\right\}^2}{\frac{1}{12}n^2\cdot m\cdot(m^2-1)}, \tag{2.52}$$

де σ_{ϕ}^2 - фактична дисперсія підсумкових оцінок, які надані експертами; $\sigma_{\max}^2$ - дисперсія підсумкових оцінок за умови, що думки експертів повністю збігаються; a_i – сумарна

оцінка, одержана *i*-м об’єктом; *m* – кількість досліджуваних об’єктів; *n* – кількість експертів.

Коефіцієнт конкордації може змінюватися в межах від 0 до 1. Якщо його значення дорівнює нулю, це означає, що між оцінками експертів немає жодного зв'язку, тобто відсутня узгодженість їхніх думок. Якщо ж значення коефіцієнта дорівнює одиниці, то оцінки експертів повністю збіжні.

Для спрощення прийнято вважати думки експертів узгодженими за $W > 0{,}5$ і добре узгодженими, якщо $W > 0{,}7$.

Отримане значення показника з множини $M_{n,v}$ доцільно інтерпретувати за допомогою елементів нечіткої логіки.

Трикутну функцію належності у загальному випадку задамо аналітично наступним виразом:

$$\mu(u) = \begin{Bmatrix} 0, c \le u \le a \\ \dfrac{u-a}{b-a}, a < u \le b \\ \dfrac{c-u}{c-b}, b < u < c \end{Bmatrix}, \tag{2.53}$$

де *a, b, c* – деякі числові параметри, які приймають довільні дійсні значення і впорядковані відношенням: $a \le b \le c$; *(a, c)* – песимістична оцінка нечіткого числа; *b* – координата максимуму – оптимістична оцінка нечіткого числа.

В якості функцій належності доцільно використати трикутну функцію належності (рис. 2.3) із наступними лінгвістичними термами: Low (L), Low Medium (LM), Medium (M), High Medium (HM), High (H), розподіленими, на шкалі від 0 до 100 балів.

Даній функції належності з лінгвістичними термами відповідають наступні системи рівнянь:

$$L : \mu_1(x) = \begin{cases} \dfrac{25-x}{25}, 0 \le x \le 25 \\ 0{,}25 \le x \end{cases}, \tag{2.54}$$

$$LM:\mu_2(x)=\begin{cases}\dfrac{x}{25},0\le x\le 25\\ \dfrac{50-x}{25},25\le x\le 50,\\ 0,50\le x\end{cases} \tag{2.55}$$

$$M:\mu_3(x)=\begin{cases}0,x\le 25\\ \dfrac{x-25}{25},25\le x\le 50\\ \dfrac{75-x}{25},50\le x\le 75\\ 0,75\le x\end{cases}, \tag{2.56}$$

$$HM:\mu_4(x)=\begin{cases}0,x\le 50\\ \dfrac{x-50}{25},50\le x\le 75\\ \dfrac{100-x}{25},75\le x\le 100\end{cases}, \tag{2.57}$$

$$H:\mu_5(x)=\begin{cases}0,x\le 75\\ \dfrac{x-75}{25},75\le x\le 100\end{cases}. \tag{2.58}$$

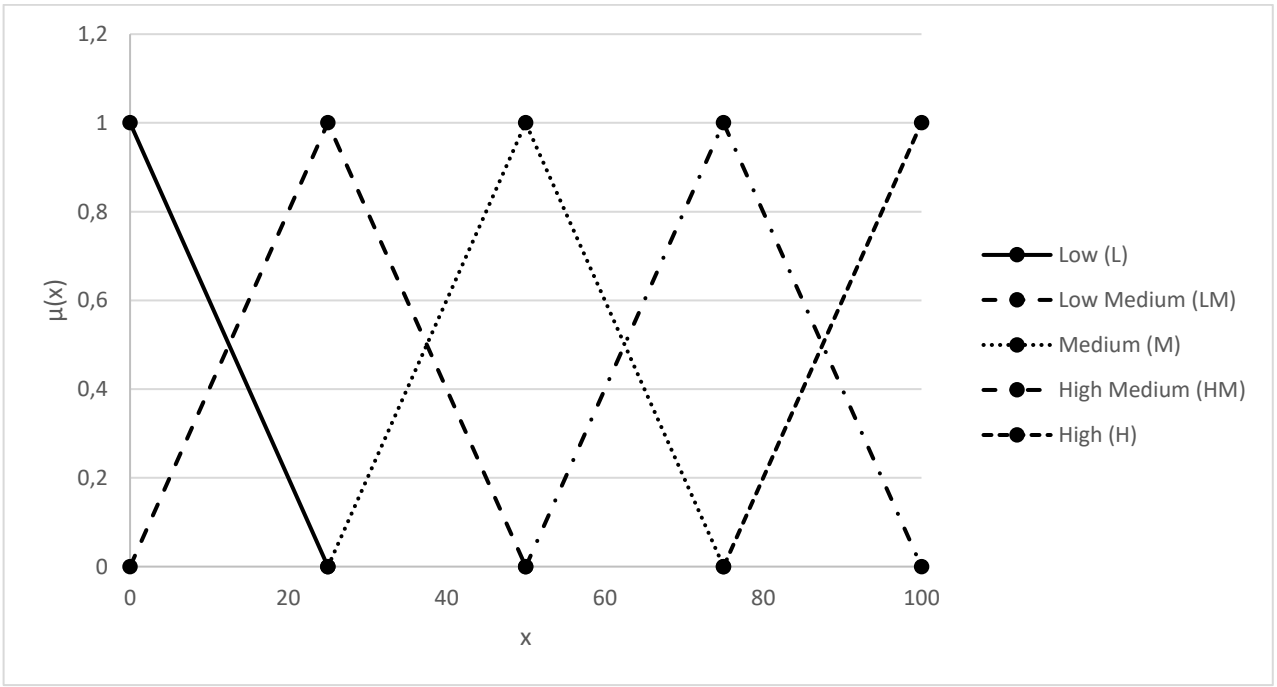

Рисунок 2.3 - Трикутні функції належності з лінгвістичними термами Low (L), Low Medium (LM), Medium (M), High Medium (HM), High (H)

Процес виявлення поліморфних вірусів агентом визначимо алгоритмом.

1. Збір даних (спостереження).

Агент отримує спостереження o_t з системи на кроці t. Даний етап включає сигнатурний аналіз, поведінковий аналіз та мережевий аналіз.

Сигнатурний аналіз передбачає пошук збігів у коді зі списком відомих сигнатур:

$$\sigma : C \rightarrow \{0,1\},\ \sigma(C) = 1, \text{ якщо код збігається із сигнатурою.}$$

Поведінковий аналіз передбачає виявлення аномальних поведінок:

$$\delta : B \rightarrow \{0,1\},\ \delta(B) = 1, \text{ якщо поведінка є аномальною.}$$

Мережевий аналіз передбачає виявлення аномалій у мережевій активності, таких як: підозрілі з'єднання, надмірне використання ресурсів.

При цьому спостереження формалізується як множина:

$$o_t=(o_{сигн}, o_{повед}, o_{мереж}), \tag{2.59}$$

де o_t – це структуроване спостереження мережевої активності в момент часу ttt, яке можна аналізувати для виявлення аномалій; $o_{сигн}$ - характеристики сигнатурних ознак (наприклад, співпадіння з відомими шаблонами атак); $o_{повед}$ - поведінкові характеристики (аномальна поведінка користувачів, серверів, додатків тощо); $o_{мереж}$ - мережеві характеристики (обсяг трафіку, кількість з'єднань, типи протоколів, час життя сесій тощо).

2. Класифікація стану системи.

На основі спостережень o_t агент визначає поточний стан s_t:

$$s_t = f(o_t), \tag{2.60}$$

де $f : O \rightarrow S$ - функція класифікації; s_t – це поточний стан системи (наприклад, «норма», «підозра», «атака» тощо); $f(o_t)$ – це функція класифікації, яка на основі спостереження o_t визначає цей стан. Агент аналізує o_t – всі сигнатурні, поведінкові й мережеві характеристики; функція f (яка може бути, наприклад, правилом, алгоритмом машинного навчання, деревом рішень чи нечіткою логікою) приймає це спостереження і повертає оцінку стану s_t.

Класифікація здійснюється через алгоритми машинного навчання, наприклад, дерева рішень, байєсівські моделі або нейронні мережі.

3. Рішення і дія.

На даному етапі агент застосовує політику П, щоб обрати дію a_t так:

$$a_t = П(o_t), \tag{2.61}$$

де a_t - це дія, яку агент виконує у момент часу t (наприклад: заблокувати IP, сповістити адміністратора, ізолювати сегмент мережі, просто записати подію тощо); $П$ - це політика прийняття рішень, тобто набір правил або стратегій, за якими агент визначає, що робити на основі спостереження o_t (сформоване спостереження про стан мережі).

Типовими діями є:

1) $a_{аналіз}$ - розширений аналіз коду;

2) $a_{блокування}$ - зупинка підозрілого процесу;

3) $a_{ізоляція}$ - переміщення заражених файлів у карантин;

4) $a_{навчання}$ - оновлення моделей f і $П$.

Після виконання дії система переходить у новий стан:

$$s_{t+1} = T(s_t, a_t), \tag{2.62}$$

де s_t – це поточний стан системи на час t; a_t – це дія, яку агент виконав у цьому стані; T – це функція переходу станів; s_{t+1} – це новий стан системи після виконання дії.

Поліморфний вірус змінює свій код, але зберігає ключові поведінкові характеристики. Агент аналізує поведінку через аномалії, які формалізуємо так:

а) модель поведінки, для якої кожну поведінку B_t задамо множиною властивостей:

$$B_t = \{p_1, p_2, \ldots, p_n\}, \tag{2.63}$$

де p_i – окрема характеристика, наприклад, створення файлів у системних директоріях, надмірне навантаження на ЦП;

б) функція аномалії, за якою агент визначає ступінь аномальності поведінки як:

$$A(B_t) = \frac{\sum_{i=1}^{n} w_i \cdot p_i}{\sum_{i=1}^{n} w_i}, \tag{2.64}$$

де w_i - ваговий коефіцієнт, що відображає значущість p_i; i – це індекс, який нумерує елементи (ознаки поведінки), i=1,2,3,...,n. Наприклад, i=1 - це перша характеристика (наприклад, кількість нових підключень), i=2 - це друга характеристика (наприклад, обсяг трафіку), i=3 - третя характеристика (наприклад, тип трафіку) і так далі.

в) поріг виявлення, де стан вважається аномальним, якщо:

$$A(Bt)>\theta, \tag{2.65}$$

де $A(Bt)$ – це обчислений ступінь аномальності у момент часу t; θ – це поріг виявлення аномалії (наприклад, 0,7); якщо ступінь аномальності більший за поріг ($A(Bt)>\theta$), то система вважає стан аномальним.

Таким чином, визначені математична модель ІА, формалізована мета ІА, відображений покроковий процес роботи ІА та формалізовано етапи його поведінки, формалізована корисність роботи ІА, відображена поведінкова модель ІА та різновиди його поведінкових моделей, відображена модель поширення поліморфних вірусів, формалізовано процес виявлення поліморфних вірусів агентом, які є основою для синтезу мультиагентних систем виявлення поліморфних вірусів.

2.4. Архітектура гібридних мультиагентних систем виявлення поліморфних вірусів

Гібридна МАС - це система, яка комбінує в собі різні типи агентів для вирішення комплексної задачі виявлення поліморфних вірусів, використовуючи різні методи та засоби. Гібридні МАС забезпечують високу ефективність, масштабованість, адаптивність та здатність до колективного аналізу загроз, тому їх застосовують в якості основи для розроблення систем виявлення поліморфних вірусів в комп’ютерних мережах. Вони дозволяють створювати розподілені, інтегровані рішення для виявлення та нейтралізації вірусів, що є особливо важливим у протидії сучасним, складним та змінюваним загрозами.

До основних переваг використання гібридних МАС для виявлення поліморфних вірусів є паралельне оброблення подій, розподілене виявлення загроз,

інтелектуальна взаємодія між агентами, адаптивність до нових загроз, прогнозування та виявлення аномалій, масштабованість, динамічне реагування на загрози, зниження навантаження на одну точку системи, розподілене навчання на основі досвіду.

Паралельне оброблення подій реалізується тоді, коли кілька агентів, з яких складаються гібридні МАС, працюють одночасно, виконуючи різні функції. Це дозволяє здійснювати паралельну обробку великої кількості даних (файлів, мережевого трафіку, системних журналів), що значно підвищує ефективність виявлення вірусів. Кожен агент може відповідати за окрему задачу (аналіз файлів, моніторинг процесів, сканування трафіку), що пришвидшує виявлення та нейтралізацію загроз.

Розподілене виявлення загроз реалізується тоді, коли агенти гібридних МАС можуть бути розподілені по різних сегментах мережі або різних частинах комп'ютерної системи. Це дозволяє забезпечити глобальний моніторинг системи на всіх рівнях, що дає змогу виявляти загрози навіть у найвіддаленіших її частинах, які можуть бути не помічені традиційними методами.

Інтелектуальна взаємодія між агентами у гібридних МАС доступна через можливість взаємодії для обміну інформацією та прийняття рішень спільно. Якщо один агент виявляє потенційну загрозу, інші агенти можуть бути повідомлені, щоб проаналізувати її з різних аспектів (наприклад, один агент може перевірити мережевий трафік, інший – аналізувати поведінку програм). Це дозволяє швидко і точно виявляти нові види вірусів, які можуть маскуватися різними способами.

Адаптивність до нових загроз проявляється в результаті включення в структуру гібридних МАС агентів, здатних самонавчатися, застосовуючи методи машинного навчання або еволюційні алгоритми. Це дозволяє агентам адаптуватися до нових типів загроз, виявляти невідомі віруси та реагувати на зміни у поведінці вірусів, що змінюють свою структуру або використовують нові техніки атак.

Масштабованість гібридних МАС проявляється тоді, коли у систему додаються нові агенти для розширення можливостей виявлення вірусів. Це робить такі системи зручними для великих організацій, хмарних платформ або великих мереж.

Здатність гібридних МАС до прогнозування та виявлення аномалій відображається через можливість в одному агенті реалізувати методи для прогнозування поведінки системи, а у іншому – виявляти аномалії в реальному часі. Це дозволяє виявити навіть загрози, які можуть не мати чіткої сигнатури або діяти дуже непомітно, наприклад, через використання нестандартних шляхів проникнення.

Гібридні МАС також дозволяють організувати динамічну реакцію на вірусні загрози. Кожен агент може мати свою стратегію реагування: ізоляція заражених файлів, блокування підозрілих процесів, повідомлення адміністратора або навіть автоматичне видалення зловмисних файлів. Це дає змогу швидко нейтралізувати загрози без ручного втручання.

Гібридні МАС можуть використовувати спільне навчання та обмін досвідом між агентами, що покращує виявлення нових загроз. Один агент може поділитися знаннями про новий тип вірусу, а інші агенти можуть швидко застосувати ці знання в інших частинах системи.

Також гібридні МАС дозволяють знижувати навантаження на одну точку системи. Замість того, щоб покладатися на один центральний процес, який перевіряє всі дані, гібридна МАС дозволяє розподіляти навантаження, що підвищує продуктивність та ефективність виявлення вірусів. Це особливо корисно в великих, складних або розподілених системах.

Отже, гібридні МАС для виявлення ЗПЗ є потужним підходом для забезпечення безпеки комп'ютерних мереж і систем. Такі системи використовують кілька агентів, кожен з яких має свою специфічну роль у процесі виявлення та нейтралізації поліморфних вірусів. Основні компоненти таких систем зображені на рис. 2.4.

Як зображено на рис. 2.4, ІА складається з наступних модулів:

1) модуль аналізу сканує систему на предмет наявності підозрілих або зловмисних активностей, аналізує файлову систему, пам’ять, мережевий трафік, поведінку процесів і при цьому можуть використовуватися сигнатурний аналіз (порівняння з базою відомих вірусів), аналіз поведінки (пошук аномалій, підозрілих дій), евристичний аналіз (виявлення нових вірусів на основі ознак), тобто

відбувається збір даних, пов'язаних із ЗПЗ, і проводиться детальний аналіз для визначення природи загрози, її джерела та потенційного впливу;

2) модуль класифікації поліморфних вірусів за рівнями складності здійснює нечітку класифікацію виявлених поліморфних вірусів згідно його належності до різних рівнів складності та визначає їх рівень небезпеки;

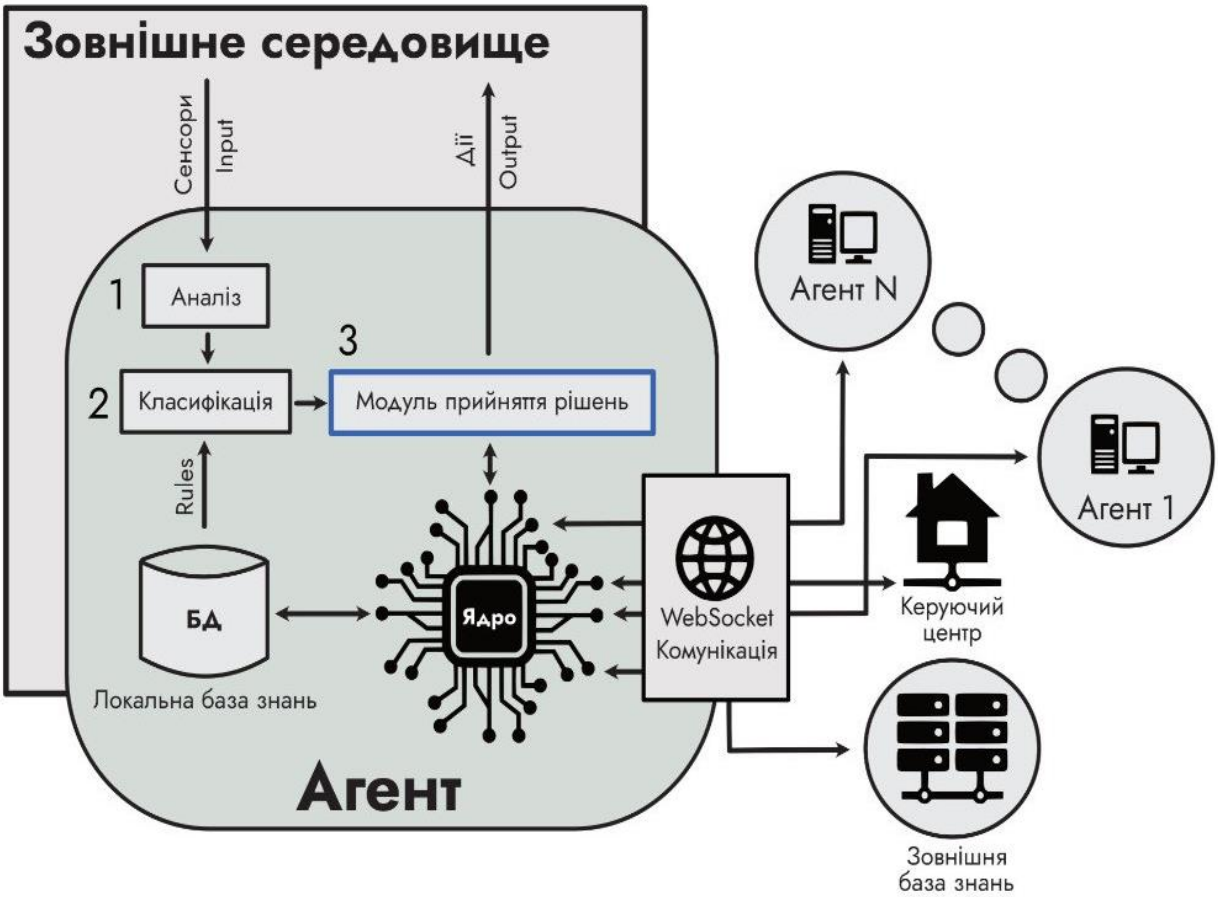

Рисунок 2.4 - Архітектура гібридних МАС виявлення поліморфних вірусів

3) модуль прийняття рішення визначає оптимальну реакцію та нейтралізацію (карантин, видалення, оповіщення, блокування), здійснює оновленння бази знань та стратегій виявлення.

Загальний алгоритм роботи гібридних МАС зображено на рис. 2.5.

За допомогою динамічної логіки розглянуто логіку дій у системі.

1. Виявлення вірусу агентом:

$$\langle \alpha_{\text{сканування}} \rangle p_{\text{виявлення}},$$

де $\langle \alpha_{\text{сканування}} \rangle$ - дія сканування, агент виконує сканування мережі чи файлів на наявність вірусів; $p_{\text{виявлення}}$ - властивість або умова виявлення вірусу (тобто вірус виявлено).

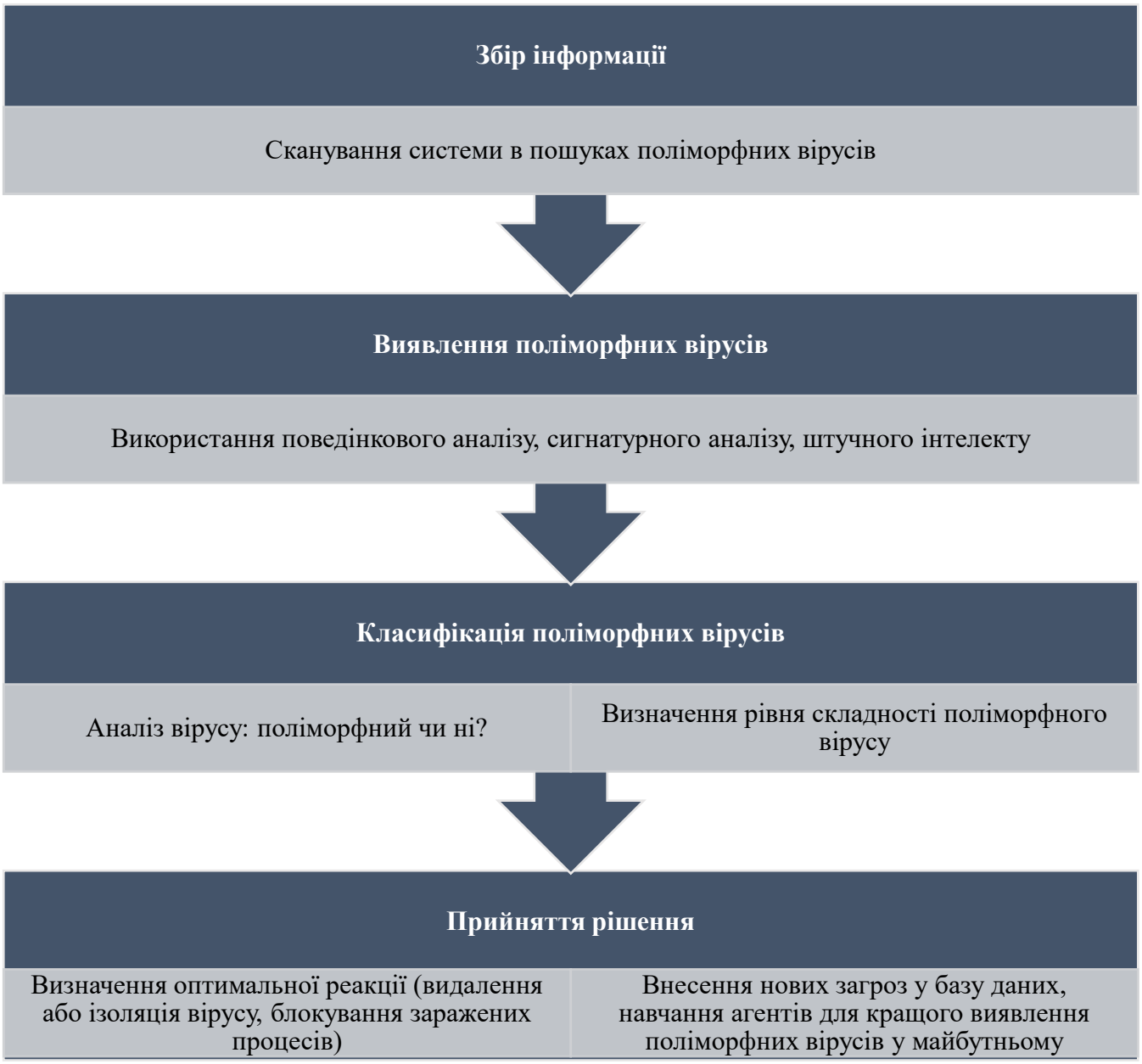

Рисунок 2.5 – Загальний алгоритм роботи гібридних МАС

2. Реакція вірусу на детекцію:

$$[p_{\text{виявлення}}] \langle \alpha_{\text{мутація}} \cup \alpha_{\text{самознищення}} \rangle T,$$

де $[p_{\text{виявлення}}]$ - виявлення вірусу (стан, коли агент знайшов вірус); $\alpha_{\text{мутація}}$ - дія мутації вірусу (зміна своєї форми, щоб уникнути повторного виявлення); $\alpha_{\text{самознищення}}$ - дія

самознищення вірусу (видалення себе, щоб не залишити слідів); ∪ - операція об'єднання (тобто або мутація, або самознищення); T - істина (тобто будь-який можливий стан; успішне виконання дії).

3. Цикл розповсюдження поліморфного вірусу:

$$((\alpha_{розповсюдження};\ \alpha_{мутація})^*)p_{маскування},$$

де $\alpha_{розповсюдження}$ - дія розповсюдження вірусу (наприклад, зараження нових файлів або пристроїв); $\alpha_{мутація}$ - дія мутації (вірус змінює свою структуру, щоб залишитися непоміченим); $p_{маскування}$ - стан маскування вірусу (тобто вірус заховав себе і став невидимим для простого виявлення); «;» - оператор послідовного виконання дій; «*» - послідовність може виконуватися нуль або більше разів.

4. Цикл боротьби агента з вірусом:

$$((\alpha_{сканування};\ \alpha_{аналіз};\ \alpha_{знешкодження})^*)\ p_{нейтралізація},$$

де $\alpha_{сканування}$ - дія сканування (агент шукає підозрілі об'єкти у системі); $\alpha_{аналіз}$ - дія аналізу (агент аналізує знайдені підозрілі об'єкти, щоб зрозуміти їхню природу); $\alpha_{знешкодження}$ - дія знешкодження (агент видаляє, блокує або лікує вірус); $p_{нейтралізація}$ - стан нейтралізації вірусу (загроза усунена); «;» - оператор послідовного виконання дій; «*» - повторення послідовності нуль або більше разів.

Розглянемо узагальнені переходи станів системи.

1. Початковий стан (чиста система):

$$s_0 = (p_{чистий}, \neg p_{інфіковано}),$$

де s_0 - початковий стан системи; $p_{чистий}$ – «система чиста» (відсутнє ЗПЗ); $\neg p_{інфіковано}$ – «система не інфікована» (знак ¬ означає логічне «не»).

2. Якщо поліморфний вірус заражає систему:

$$T(s,\ \alpha_{розповсюдження},\ s') \Rightarrow s' = (p_{інфіковано}, p_{маскування}),$$

де $T(s, \alpha, s')$ - це функція переходу з початкового стану s через дію α система переходить у новий стан s'; $\alpha_{розповсюдження}$ - дія розповсюдження вірусу; s' - новий стан системи після зараження; $p_{інфіковано}$ - система тепер інфікована; $p_{маскування}$ - вірус маскує свою присутність (тобто стає менш помітним для виявлення).

3. Якщо агент знаходить вірус:

$$T(s,\alpha_{сканування}, s') \Rightarrow s' = (p_{виявлення},p_{інфіковано}),$$

де $T(s, \alpha, s')$ - це функція переходу з початкового стану s через дію α система переходить у новий стан s'; $\alpha_{сканування}$ - дія сканування системи агентом; s' - новий стан після сканування; $p_{виявлення}$ - вірус виявлено агентом; $p_{інфіковано}$ - система залишається інфікованою, навіть якщо вірус уже знайдено.

4. Якщо агент знешкоджує вірус:

$$T(s, \alpha_{знешкодження},s') \Rightarrow s' = (p_{чистий},\neg p_{інфіковано}),$$

де де Т(s, α, s') - це функція переходу з початкового стану s через дію α система переходить у новий стан s'; $\alpha_{знешкодження}$ - дія знешкодження вірусу агентом; s' - новий стан після знешкодження; $p_{чистий}$ - система знову чиста; $\neg p_{інфіковано}$ - система більше не інфікована.

Гібридні МАС моделюються як сукупність агентів $A = \{A_1, A_2,..., A_n\}$, які взаємодіють із комп'ютерними системами та один з одним. Нехай комп'ютерна система представлена множиною можливих станів S, множиною дій агентів A та функцією переходів T. Тоді, гібридну МАС визначимо кортежем так:

$$MAC = \langle A, S, D, T, R, O, P, \pi \rangle, \quad (2.66)$$

де $A = \{A_1, A_2, ... , An\}$ – множина агентів; S – множина станів комп'ютерної системи; D_i – множина можливих дій агента A_i; $T : S \times A_1 \times A_2 \times ... \times A_n \rightarrow P(S)$ – функція переходу між станами; $R : S \times A_1 \times A_2 \times ... \times A_n \rightarrow R$ – функція винагороди, яка оцінює ефективність вибраних дій; $O : A \times S \rightarrow P(O)$ – функція спостереження, яка визначає, яку інформацію отримує кожен агент; $P(s'/s, a_1, ..., a_n)$ – ймовірність переходу в стан s' після виконання дій агентами; $\pi_i : S \rightarrow A_i$ – стратегія агента A_i, яка визначає, яку дію він обирає в кожному стані.

Агенти діють у середовищі та можуть співпрацювати або конкурувати. Вони функціонують згідно алгоритму, в якому основні кроки такі:

1. Агент спостерігає стан s частково через $O(A_i, s)$.

2. Агент вибирає дію a_i відповідно до своєї стратегії $\pi_i(s)$.

3. Агент отримує винагороду $R\ (s, a_1, \ldots, a_n)$.

4. Система переходить у новий стан s' згідно з функцією переходу $P(s'\ /s, a_1, \ldots, a_n)$.

Формально, оптимальна стратегія для агента A_i визначається як:

$$\pi_i^* = arg\ \max_{\pi_i} E(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_{1,t}, \ldots, a_{n,t}), \tag{2.67}$$

де π_i^* - оптимальна стратегія для агента A_i; $arg\ \max_{\pi_i} E$ - знаходження такої стратегії π_i, яка максимізує очікувану суму винагород, $E(\ldots)$ - математичне сподівання (очікуване значення); $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_{1,t}, \ldots, a_{n,t})$ – сума винагород за всі моменти часу t у майбутньому (нескінченна послідовність кроків); γ^t – коефіцієнт дисконтування на t-у кроці ($0<\gamma\leq 1$) - чим далі в майбутнє, тим менш важлива винагорода; $R(s_t, a_{1,t}, \ldots, a_{n,t})$ - функція винагороди, яка залежить від стану системи s_t, дій усіх агентів у момент часу t: $a_{1,t}, \ldots, a_{n,t}$.

Модулі агентів можуть бути розділені на підгрупи відповідно до їхніх функцій. Модуль аналізу M_A аналізує файли, пам’ять, мережевий трафік.

Вхідні дані: $O_D(s)$ – журнали процесів, сигнатури файлів, поведінкові аномалії. Дії: M_D = {сканувати, аналізувати, ігнорувати}. Функція винагороди:

$$R_A(s, a_A) = \begin{Bmatrix} +10, \text{якщо знайдено поліморфний вірус} \\ -5, \text{якщо хибна тривога} \\ -10, \text{якщо поліморфний вірус не виявлено} \end{Bmatrix}, \tag{2.68}$$

де s - поточний стан системи; a_A - дія агента A; $R_A(s, a_A)$ - винагорода або штраф для агента залежно від результату його дії.

Модуль класифікації M_K використовується для класифікації поліморфних вірусів.

Вхідні дані: вихід модуля M_A. Дії: M_K = {класифікувати, перевірити поведінку, відправити в карантин}.

Функція винагороди:

$$R_{\text{К}}(s, a_{\text{К}}) = \begin{Bmatrix} +20, \text{вірно класифікований поліморфний вірус} \\ -10, \text{ хибна класифікація} \\ -5, \text{здійснена зайва перевірка} \end{Bmatrix}, \quad (2.69)$$

де *s* - поточний стан системи; a_K - дія агента *K*; $R_K(s, a_K)$ - винагорода або штраф для агента залежно від результату його дії.

Модуль прийняття рішення M_M виконує нейтралізацію загрози. Дії: M_M = {видалити, карантин, ігнорувати}.

Функція винагороди:

$$R_M(s, a_M) = \begin{Bmatrix} +50, \text{поліморфний вірус успішно видалено} \\ -20, \text{ видалено безпечний файл} \\ -50, \text{ поліморфний вірус залишився активним} \end{Bmatrix}, \quad (2.70)$$

де *s* - поточний стан системи; a_M - дія агента *M*; $R_M(s, a_M)$ - винагорода або штраф для агента залежно від результату його дії.

Кожен агент обирає дію, максимізуючу сумарно винагороду функціонування його модулів. Оскільки система динамічна та стохастична, агенти використовують алгоритми підкріплювального навчання:

Q-навчання для оновлення цінності станів:

$$Q_i(s, a_i) \leftarrow Q_i(s, a_i) + \alpha[R(s, a_i) + \gamma maxQ_i(s', a_i') - Q_i(s, a_i)], \quad (2.71)$$

де $Q_i(s, a_i)$ - цінність (value) для агента *i* у стані *s* при виконанні дії a_i; α – коефіцієнт навчання (learning rate), який визначає, наскільки сильно нове спостереження повинно коригувати поточне значення *Q*; $R(s, a_i)$ - винагорода за виконання дії a_i у стані *s*; γ - коефіцієнт дисконтування, що визначає важливість майбутніх винагород (де $0 \leq \gamma \leq 1$); $maxQ_i(s', a_i')$ - максимальна цінність з усіх можливих дій a_i' у новому *s'*; $Q_i(s', a_i')$ - цінність для агента *i* у новому стані s' після виконання дії a_i'.

Стратегія вибору дій:

$$\text{a}_\text{i} = arg \max_{a_i'} Q_i(s, a_i'), \quad (2.72)$$

де a_i - вибрана дія для агента *i*; *s* - поточний стан системи; $Q_i(s, a_i')$ - Q-цінність для агента *i* для дії a_i' у стані *s*; argmax - це операція, яка вибирає дію a_i' з усіх можливих дій a_i' $Q_i(s, a_i')$ максимальне.

Таким чином, розроблена архітектура гібридних МАС виявлення поліморфних вірусів, описані модулі ІА, такі як модуль аналізу, модуль класифікації, модуль прийняття рішення. Представлено загальний алгоритм роботи гібридних МАС. Відображена логіка дій у системі за допомогою динамічної логіки. Розглянуто узагальнені переходи станів системи. Відображений алгоритм дій ІА в гібридних МАС, формалізована оптимальна стратегія ІА в системі. Встановлена функція винагороди для кожного модуля ІА. Це дозволило розробити архітектуру гібридних МАС виявлення поліморфних вірусів, які мають змогу здійснювати аналіз середовища, встановлювати темпи поширення, виявлявлення, класифікації поліморфних вірусів та використовувати різні стратегії прийняття рішення, враховуючи типи загроз.

2.5. Висновки до другого розділу

Поліморфний вірус подано як п'ятірку, що включає множину можливих кодів вірусу, механізм мутації, що задає алгоритм зміни коду вірусу, механізм виконання, що визначає дію вірусу, наприклад, знищення файлів, викрадення даних, поширення тощо, механізм поширення, який визначає правила зараження інших файлів або систем, алгоритм приховування, який змінює поведінку вірусу залежно від детекторів. Розроблено моделі класів поліморфних вірусів.

Запропоновано здійснити нечітку класифікацію поліморфних вірусів згідно рівнів складності із використанням нечіткого логічного висновку, що включає визначення характеристик виявленого поліморфного ЗПЗ та формування дерева логічного висновку, опис лінгвістичних змінних, визначення функцій належності лінгвістичних термів, формування бази знань системи нечіткого висновку, отримання ймовірності належності досліджуваного файлу до поліморфного ЗПЗ різних рівнів складності, нечітку класифікацію поліморфних вірусів.

Визначена математична модель ІА, формалізована мета ІА, відображений покроковий процес роботи ІА та формалізовано етапи його поведінки, формалізована корисність роботи ІА, відображена поведінкова модель ІА та різновиди його

поведінкових моделей, відображена модель поширення поліморфних вірусів, описаний процес виявлення поліморфних вірусів агентом. Це дає змогу для створення та практичної реалізації ІА для виявлення поліморфних вірусів.

Розроблена архітектура гібридних МАС виявлення поліморфних вірусів, які каскадно делегуючи повноваження, здійснюють аналіз середовища, встановлюють темпи поширення, виявляють, класифікують поліморфні віруси та використовують різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності.

Основні результати розділу опубліковані у працях [151, 153, 154].

РОЗДІЛ 3.
КОМПЛЕКСНИЙ ПІДХІД ДО ВИЯВЛЕННЯ, АНАЛІЗУ ТА КЛАСИФІКАЦІЇ ПОЛІМОРФНИХ ВІРУСІВ

3.1. Метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра

Запропоновано комплексний підхід до виявлення, аналізу та класифікації поліморфних вірусів (рис. 3.1), який складається з трьох етапів:

1) етап «0» моделювання процесу поширення поліморфних вірусів на основі моделі Лотки-Вольтерра;

2) етап «1» синтезу методів для виявлення поліморфних вірусів (алгоритму пошуку рядка, інтелектуального аналізу даних, аналізу в пісочниці, машинного навчання, методу розробки структурних функцій, ймовірнісних логічних мереж);

3) етап «2» нечіткої класифікації виявлених поліморфних вірусів.

Використання інструментів та методів виявлення поліморфного ЗПЗ можна порівняти з класичною моделлю «хижак-жертва». Модель Лотки-Вольтери (модель «хижак-жертва») описує популяцію, котра складається з двох видів, які взаємодіють між собою. Жертви вимирають зі швидкістю, котра дорівнює числу зустрічей хижаків та жертв, які є пропорційними чисельності обох популяцій. Хижаки розмножуються зі швидкістю, яка є пропорційна кількості жертв, які з’їли хижаки. Система рівнянь, котра описує таку популяцію називається моделлю Лотки-Вольтерра. За умовами моделі жертви їдять рослини, а хижаки – жертв. Використаємо дану модель для моделювання процесу виявлення поліморфного ЗПЗ. Поліморфні віруси у комп’ютерній системі виступатимуть у вигляді «жертви», інструменти та методи виявлення поліморфного ЗПЗ виступатимуть у ролі «хижака».

Даний підхід є нульовим етапом у запропонованому комплексному підході до виявлення, аналізу та класифікації поліморфного ЗПЗ (рис. 3.1).

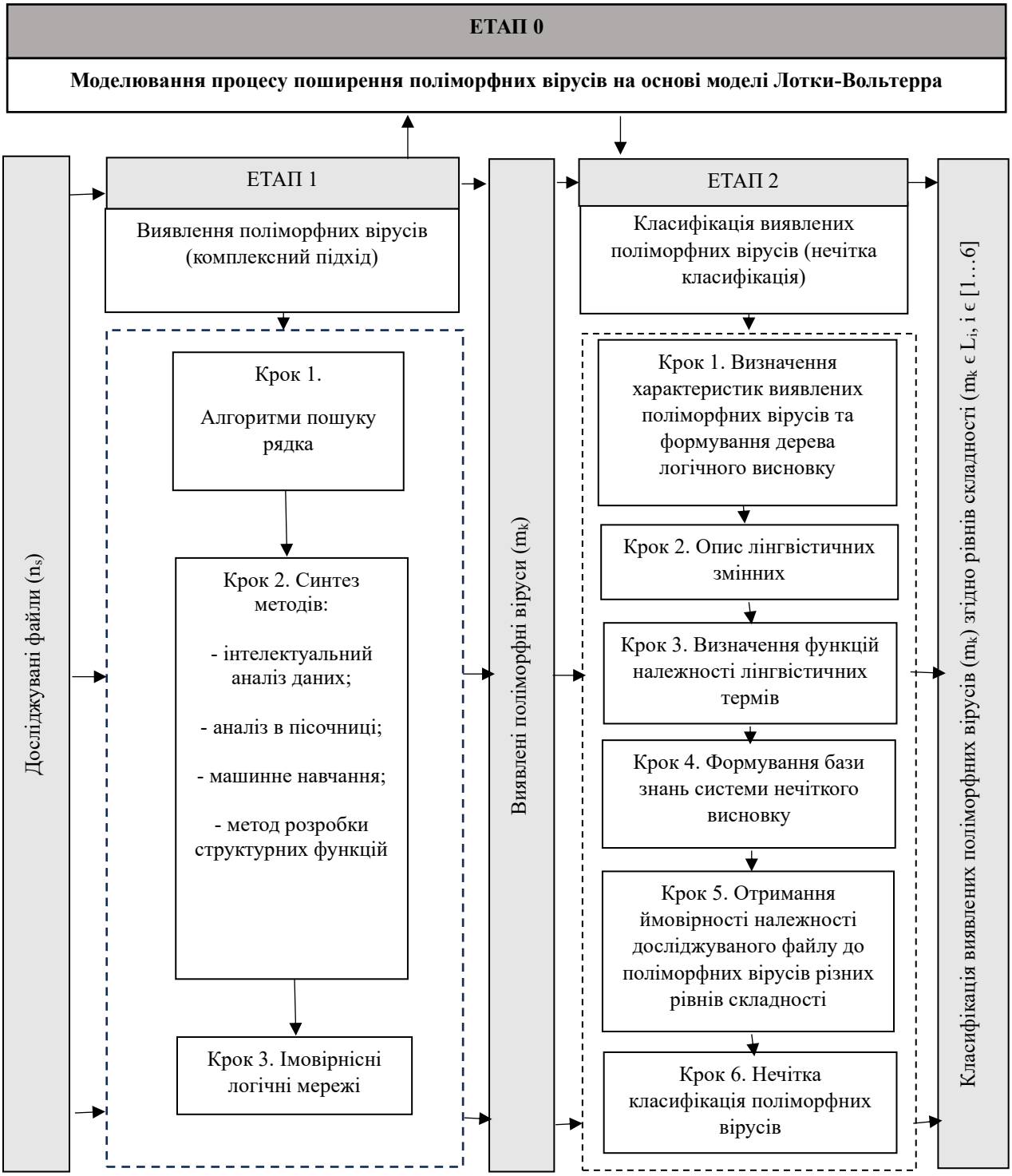

Рисунок 3.1 - Комплексний підхід виявлення, аналізу та класифікації поліморфних вірусів

У загальному вигляді модель міжвидової конкуренції подають наступним чином:

$$\begin{cases} \frac{dx}{dt} = (\alpha - \beta y)x \\ \frac{dy}{dt} = (-\gamma + \delta x)y \end{cases}, \quad (3.1)$$

де x – кількість жертв (кількісний вимір поліморфних вірусів на момент часу t); y – чисельність хижаків (кількісний вимір доступних технологій, методів та інструментів виявлення поліморфних вірусів на момент часу t); t – час (час виявлення поліморфних вірусів); $\alpha, \beta, \gamma, \delta$ – коефіцієнти, котрі відображають взаємодію між видами.

У випадку адаптації моделі для моделювання процесу виявлення поліморфних вірусів $\alpha, \beta, \gamma, \delta$ можуть відображати наступне: α – ймовірність того, що кількість поліморфних вірусів зростатиме; β – ймовірність того, що поліморфні віруси різних рівнів складності будуть виявлені за допомогою обраних методів, технологій та інструментів; γ – ймовірність того, що деякі з обраних методів, технологій та інструментів не будуть ефективними при виявленні поліморфних вірусів різних рівнів складності в результаті появи нових їх різновидів; δ – ймовірність того, що поліморфні віруси різних рівнів складності вимагатимуть комплексного використання обраних методів, технологій та інструментів, а також новітніх підходів.

Із системи відразу випливає, що якщо поліморфні віруси відсутні ($x = 0$), то кількість необхідних методів, технологій та інструментів їх виявлення буде зменшуватися експоненціально з певним початковим коефіцієнтом (γ відповідно до формули (3.1)).

$$\dot{y} = -\gamma \cdot y \rightarrow y = C_1 \cdot e^{-\gamma \cdot t}, C_1 \epsilon R, \quad (3.2)$$

де $\dot{y}$ - це похідна функції $y(t)$ за часом t, тобто швидкість зміни кількості методів/технологій; $y(t)$ – кількістьметодів, технологій та інструментів для виявлення вірусів на час t; γ – ймовірність того, що деякі з обраних методів, технологій та інструментів не будуть ефективними при виявленні поліморфних вірусів різних рівнів складності в результаті появи нових їх різновидів; C_1 - константа, яка визначається початковими умовами задачі (можна знайти за початковими значеннями y_0); t - час.

Схожу ситуацію отримуємо при повній відсутності методів, технологій та інструментів виявлення поліморфних вірусів ($y = 0$):

$$\dot{x} = \alpha \cdot x \to x = C_2 \cdot e^{\alpha \cdot t}, C_2 \epsilon R, \tag{3.3}$$

де $\dot{x}$ - похідна функції *x(t)* за часом *t*, тобто швидкість зміни кількості поліморфних вірусів; *x(t)* - кількість поліморфних вірусів на час *t*; α - ймовірність того, що кількість поліморфних вірусів зростатиме; C_2 - константа, яка визначається початковими умовами задачі (можна знайти за початковими значеннями x_0).

Дане рівняння (3.3), також, називають рівнянням Мальтуса.

Отже, зростання поліморфних вірусів виходить експоненціальним з певною, заздалегідь заданою, константою (α).

Варто відзначити, що в моделі Лотки-Вольтерри приймаються кілька припущень:

1) відбувається постійна поява поліморфних вірусів;

2) поліморфні віруси, а також технології їх виявлення знаходяться у комп'ютерній системі;

3) враховується лише наявність поліморфних вірусів та технологій їх виявлення у комп'ютерній системі.

Знайдемо особливі точки, якими володіє система:

$$\begin{cases} (\alpha - \beta y)x = 0 \\ (-\gamma + \delta x)y = 0 \end{cases} \to \begin{cases} \alpha x = \beta xy \\ \gamma y = \delta xy \end{cases} \to \begin{cases} y(0) = \frac{\alpha}{\beta} \\ x(0) = \frac{\gamma}{\delta}, \end{cases} \tag{3.4}$$

де *x* – кількість жертв (кількісний вимір поліморфних вірусів на момент часу *t*); *y* – чисельність хижаків (кількісний вимір доступних технологій, методів та інструментів виявлення поліморфних вірусів на момент часу *t*); *t* – час (час виявлення поліморфних вірусів); *α, β, γ, δ* – коефіцієнти, котрі відображають взаємодію між видами.

Зрозуміло, що при *x*(0) = 0, *y*(0) = 0 особливою точкою буде точка (0, 0), але цей випадок тривіальний, тому що в нульовий момент часу поліморфні віруси та технології їх виявлення відсутні і, що логічно, далі не з'являються.

Складні події відбуваються в ненулевому випадку. Залежно від початкових параметрів буде змінюватися особлива точка, тобто кількість вірусів та технологій їх виявлення, коли обидва показники залишаються незмінними і збалансованими.

Якщо ж початкова умова не потрапляє в особливу точку, фазові криві будуть знаходитися навколо неї, утворюючи нескінченну циклічне коливання згідно доказів Лотка і Вольтерра. Тобто, кількість поліморфних вірусів буде зростати, а кількість ефективних методів їх виявлення зменшуватись, а потім навпаки, і так протягом необмеженої кількості часу.

Розглянемо реалізацію моделі «хижак-жертва» для моделювання процесу виявлення поліморфних вірусів із використанням Lotka-Volterra equation solver [73].

Для позначення параметрів x та y використано наступну шкалу (табл. 3.1). Показник β сформовано на основі ефективності комплексного використання наведених методів виявлення поліморфних вірусів.

Таблиця 3.1

Бальна шкала для вхідних параметрів

Бальна шкала	x	y	β
до 1	Поліморфні віруси 1-го рівня складності	Використано 1 метод (алгоритм пошуку рядка)	0,1
2	Поліморфні віруси 2-го та нижчих рівнів складності	Використано 2 методи (алгоритм пошуку рядка + 1 з методів (інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій)	0,3
3	Поліморфні віруси 3-го та нижчих рівнів складності	Використано 3 методи (алгоритм пошуку рядка + 2 з методів (інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій)	0,4
4	Поліморфні віруси 4-го та нижчих рівнів складності	Використано 4 методи (алгоритм пошуку рядка + 3 з методів (інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій)	0,5
5	Поліморфні віруси 5-го та нижчих рівнів складності	Використано 5 методів (алгоритм пошуку рядка, інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій)	0,6
6 і більше	Поліморфні віруси 6-го та нижчих рівнів складності	Використано 6 і більше методів (алгоритм пошуку рядка, інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, імовірнісні логічні мережі)	0,9

Експеримент 1 передбачає наступні вхідні показники (рис.3.2, рис. 3.3): $\alpha = 0{,}2$; $\beta = 0{,}3$ (використано 2 методи для виявлення поліморфних вірусів); $\gamma = 0{,}7$; $\delta = 0{,}3$; $x = 1$; $y = 1$; *max_time* = 100 (секунд); $t = 1$.

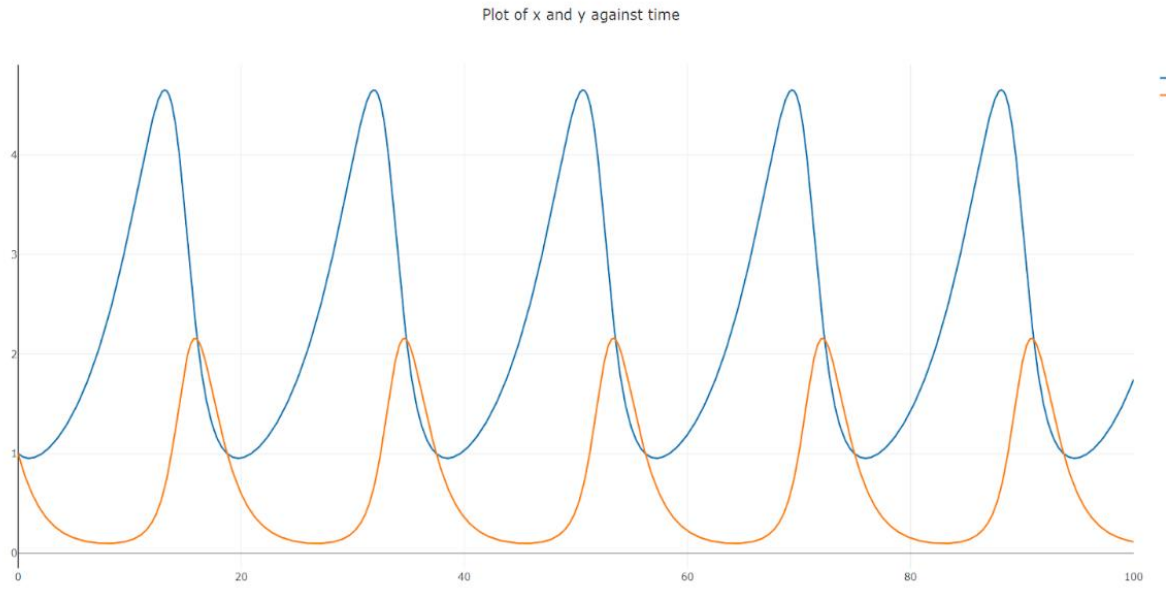

Рисунок 3.2 - Часові функції системи «хижак-жертва» (вісь абсцис – час, вісь y – бальна шкала), експеримент 1 (побудовано згідно вихідних даних за допомогою [73])

Всановлено на рис. 3.2 та рис. 3.3, що процес є коливальним. При однакових початкових значеннях кількості поліморфних вірусів та методів їх виявлення за бальною шкалою на рівні 1 балу. За цих вхідних значень кількість поліморфних вірусів зростає, а кількість і ефективність методів виявлення поліморфних вірусів падає. Коли значення y досягає величини β = 0,3, то відбувається часткове виявлення поліморфних вірусів та їх кількість починає спадати.

Зменшення кількості поліморфних вірусів через певний час починає позначатися на y, і кількість поліморфних вірусів досягає величини (у бальному вираженні) γ/δ = 0,7/0,3 = 2,33, кількість використаних методів виявлення поліморфного ЗПЗ також починає зменшуватися разом із зменшенням кількості поліморфних вірусів. Зменшення кількості поліморфних вірусів та методів його виявлення знижується до тих пір, поки y не досягне величини α/β = 0,2/0,3 = 0,66. У цей момент починає зростати кількість поліморфних вірусів, а через певний проміжок часу і методів їх виявлення. Даний процес постійно повторюється з певним періодом.

На рисунках можна чітко простежується періодичність процесу. Кількість поліморфних вірусів та методів їх виявлення коливається біля величин $x = 2{,}33$, $y = 0{,}66$ відповідно.

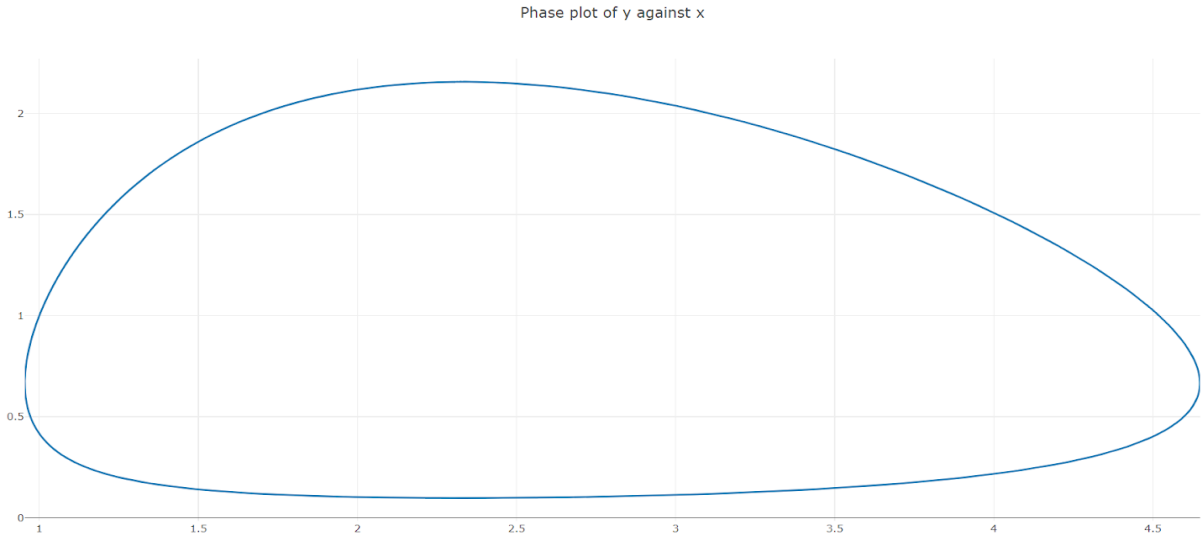

Рисунок 3.3 - Фазовий портрет системи «хижак-жертва», експеримент 1 (побудовано згідно вихідних даних за допомогою [73])

Періодичність процесу добре спостерігається на фазовій кривій ($x(t)$, $y(t)$), котра є замкнутою лінією. Крайня ліва точка цієї кривої є точкою, в котрій кількість поліморфних вірусів досягає свого мінімального значення, а в крайній правій – максимального. Поміж цих точок кількість ЗПЗ (відносна частка) спочатку спадає до нижньої точки фазової кривої, а потім зростає до верхньої точки фазової кривої. Фазова крива охоплює точку $x = 2{,}33$ та $y = 0{,}66$. В цій точці система має стаціонарний стан ($dx/dt = 0$, $dy/dt = 0$). Якщо в початковий момент система знаходилась у цій точці, то з плином часу $x(t)$ та $y(t)$ не будуть змінюватися та залишаться постійними, у всіх інших випадках буде спостерігатися коливальний процес. За даних вихідних значень максимальне значення методів виявлення поліморфних вірусів (у бальному вираженні) становитиме 2,33 бали.

Можна помітити, що обрані методи виявлення вірусів не є ефективними та призводять до поширення вірусів до рівня майже 5 балів.

Експеримент 2 передбачає наступні вхідні показники (рис. 3.4, рис. 3.5): $\alpha = 0{,}5$; $\beta = 0{,}9$ (використано 6 методів); $\gamma = 0{,}3$; $\delta = 0{,}7$; $x = 1$; $y = 1$; *max_time* = 100 (секунд); $t = 1$.

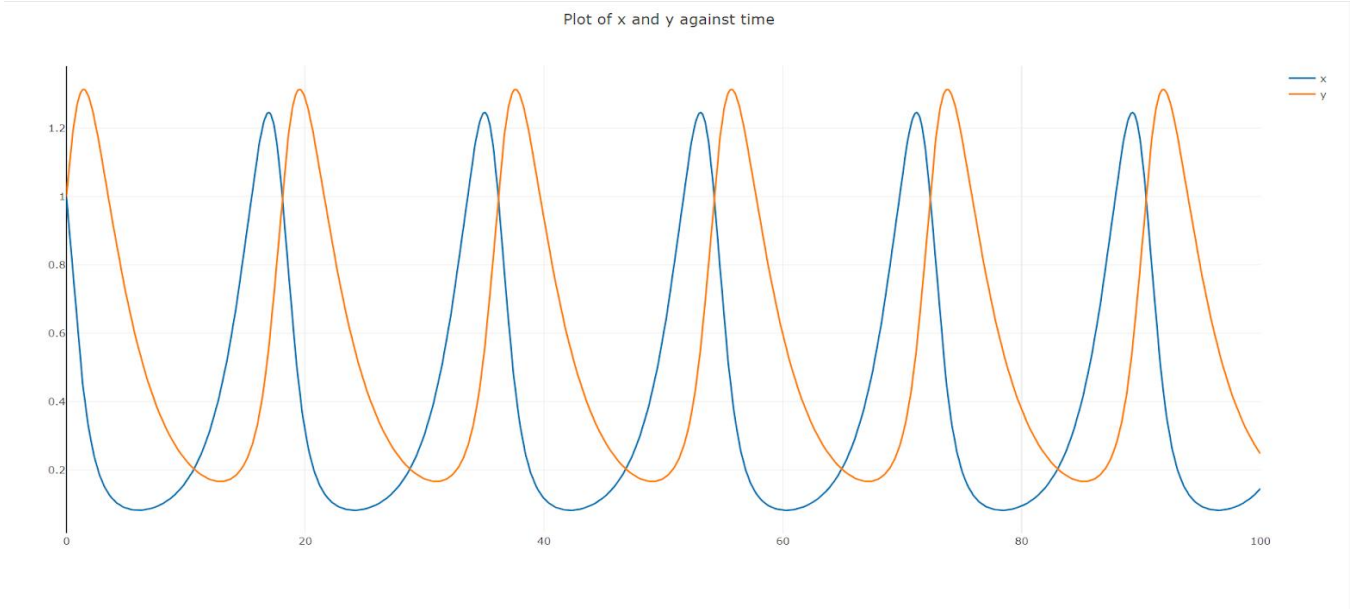

Рисунок 3.4 - Часові функції системи «хижак-жертва» (вісь абсцис – час, вісь у – бальна шкала), експеримент 2 (побудовано згідно вихідних даних за допомогою [73])

Обрані методи виявлення вірусів (6) є ефективними та призводять до поширення вірусів трішки більше 1 балу.

Експеримент 3 передбачає наступні вхідні показники (рис.3.6, рис.3.7): $\alpha = 0{,}2$; $\beta = 0{,}6$ (використано 5 методів); $\gamma = 0{,}2$; $\delta = 0{,}3$; $x = 1$; $y = 1$; *max_time* = 100 (секунд); t = 1.

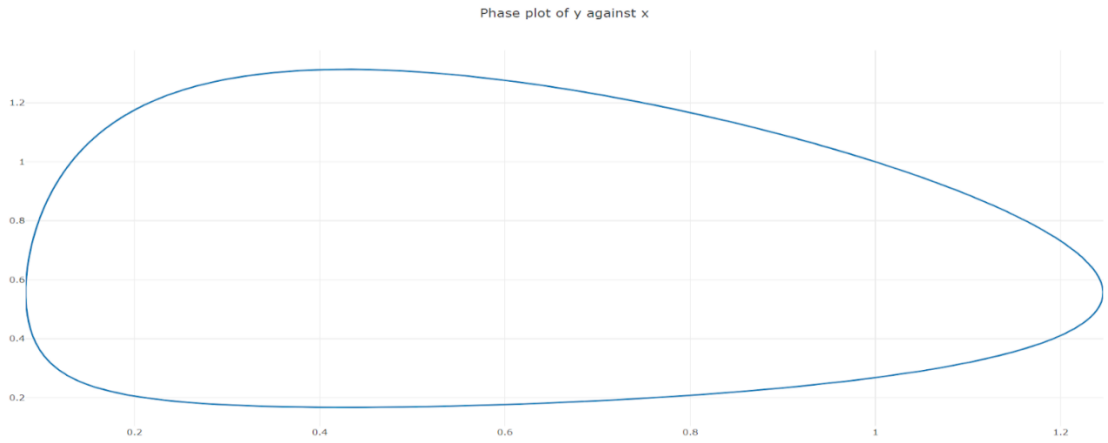

Рисунок 3.5 - Фазовий портрет системи «хижак-жертва», експеримент 2 (побудовано згідно вихідних даних за допомогою [73])

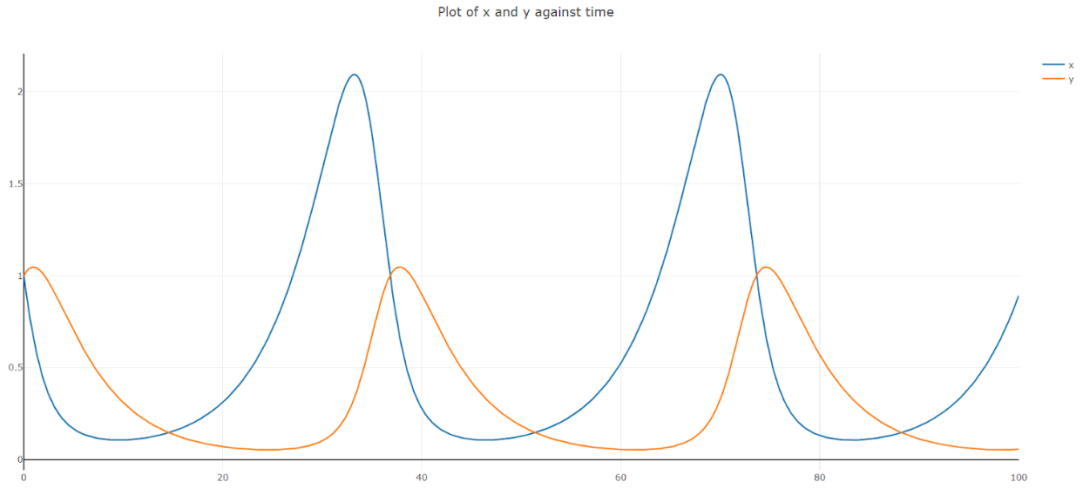

Рисунок 3.6 - Часові функції системи «хижак-жертва» (вісь абсцис – час, вісь у – бальна шкала), експеримент 3 (побудовано згідно вихідних даних за допомогою [73])

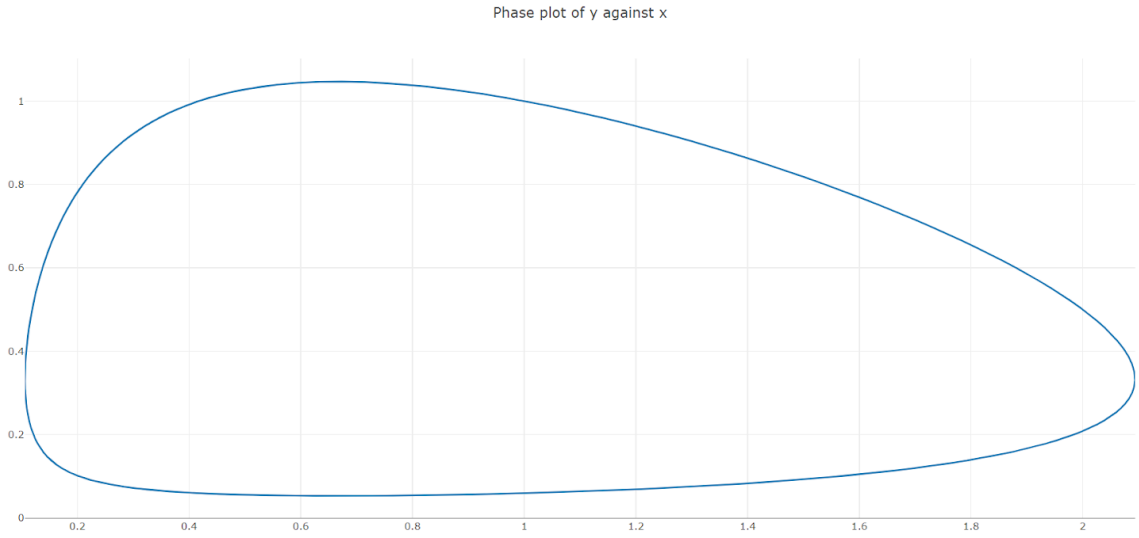

Рисунок 3.7 - Фазовий портрет системи «хижак-жертва», експеримент 3 (побудовано згідно вихідних даних за допомогою [73])

Можна встановити, що дані методи (5) виявлення вірусів є ефективними та призводять до поширення вірусів більше 2 балів.

Отже, запропоновано використання моделі Лотки-Вольтерра для моделювання рівня поширення поліморфних вірусів. Запропоновано показник α розглядати як ймовірність того, що кількість поліморфних вірусів зростатиме; β - ймовірність того, що поліморфні віруси різних рівнів складності будуть виявлені за допомогою обраних методів, технологій та інструментів; γ - ймовірність того, що деякі з обраних методів, технологій та інструментів не будуть ефективними при виявленні поліморфних

вірусів різних рівнів складності в результаті появи нових їх різновидів; δ - ймовірність того, що поліморфні віруси різних рівнів складності вимагатимуть комплексного використання обраних методів, технологій та інструментів, а також новітніх підходів; x - кількісний вимір поліморфних вірусів на момент часу t; y - кількісний вимір доступних технологій, методів та інструментів виявлення поліморфних вірусів на момент часу t. Досліджено вплив вхідних показників на максимальний темп поширення та виявлення поліморфних вірусів у його коливальному процесі. Даний підхід підтверджує доцільність використання для виявлення поліморфного зловмисного програмного забезпечення комплексу із шести методів: алгоритми пошуку рядка, інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, імовірнісні логічні мережі.

Таким чином, здійснено удосконалення методу встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який надає змогу дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати.

3.2. Метод виявлення поліморфних вірусів на основі комплексного підходу

Розглянемо більш детально перший етап запропонованого комплексного підходу, а саме синтез методів для виявлення поліморфних вірусів (алгоритму пошуку рядка, інтелектуального аналізу даних, аналізу в пісочниці, машинного навчання, методу розробки структурних функцій, ймовірнісних логічних мереж) (рис.3.8).

Даний етап складається з трьох кроків: на першому використовуються алгоритми пошуку рядка; на другому – комплекс методів (який передбачає використання методів у різному порядку, залежно від станів в системі, що дозволяє забезпечити адаптивінсть системи), серед яких інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій; на третьому кроці пропонується використання PLN, які дозволять встановити ймовірність належності ПЗ до поліморфного ЗПЗ. Використання запропонованого комплексного

підходу також дозволить визначити необхідні методи для знешкодження виявленого ЗПЗ.

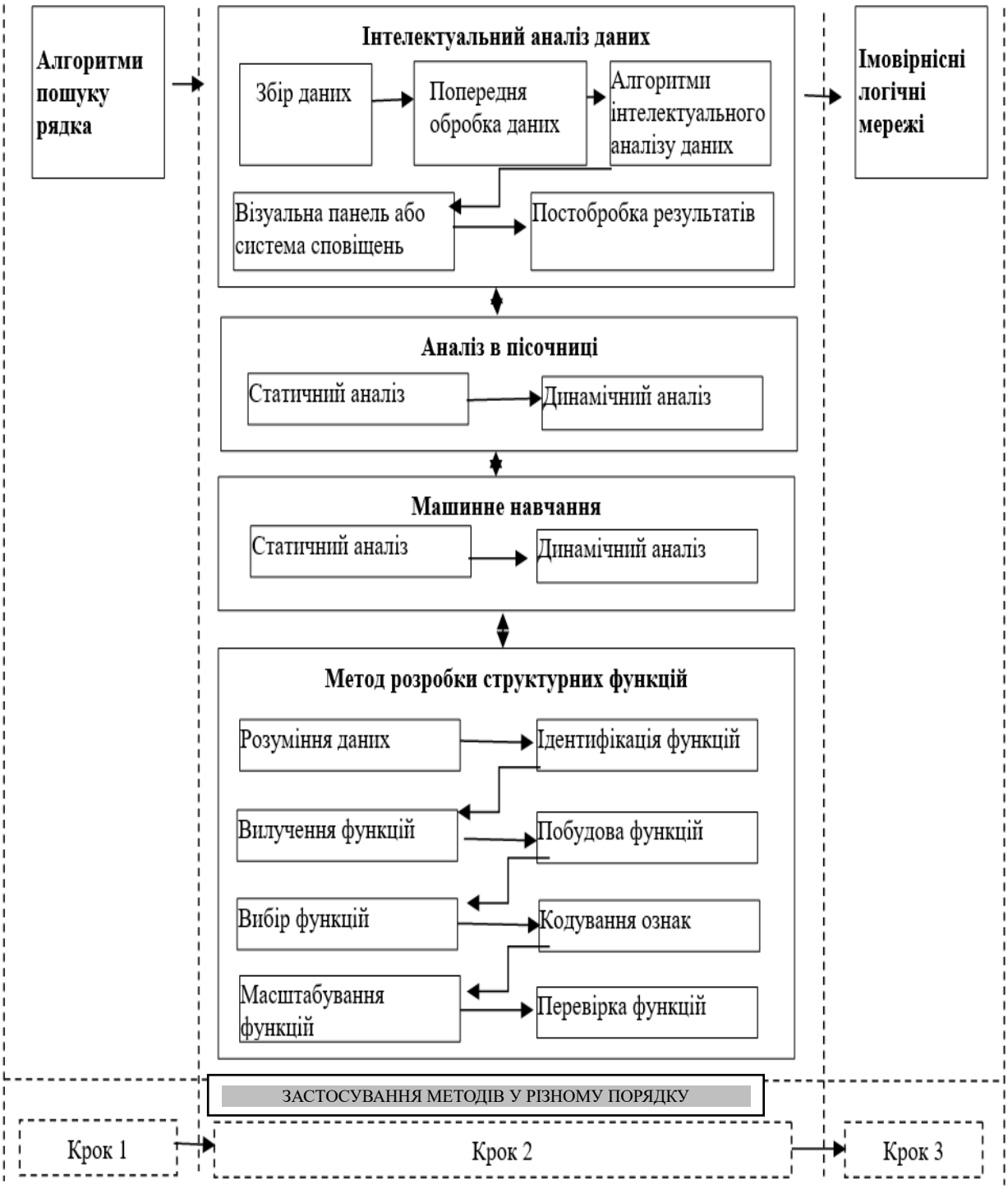

Рисунок 3.8 - Комплексний підхід до виявлення та аналізу поліморфних вірусів

Для виявлення поліморфних вірусів через пошук рядків (наприклад, у коді або байтових послідовностях) можна застосовувати комбіновані методи пошуку та аналізу патернів. Оскільки поліморфні віруси змінюють свою структуру, традиційний

пошук за фіксованими підписами (наприклад, хешами або фіксованими рядками) не завжди працює. Тому, для таких вірусів необхідні адаптовані підходи, що включають пошук схожих рядків з урахуванням змін у їхній структурі.

Відобразимо алгоритм та узагальнену формулу для алгоритму пошуку рядка в контексті виявлення поліморфних вірусів.

Нехай $S = [s_1, s_2, \ldots, s_m]$ – масив символів або байт, що представляють потенційно заражений файл або програму; $P = [p_1, p_2, \ldots, p_n]$ – шаблон або підпис, який є частиною вірусу або ознакою його присутності, але може змінюватися через поліморфізм; Δ – множина змін, що може виникати через поліморфні зміни (наприклад, змінений порядок байт, додані випадкові байти, зміни в інструкціях тощо); *Sim (a, b)* – функція схожості між двома підрядками *a і b* (може бути заснована на метриках, таких як LCS (Longest Common Subsequence), Levenshtein distance або інші).

Алгоритм пошуку рядка з врахуванням поліморфізму наступний.

1. Зміна шаблону під поліморфізм. Поліморфні віруси можуть модифікувати свої рядки чи байти, тому не обов'язково використовувати точний шаблон. Замість цього можна використовувати шаблон, що допускає деякі зміни:

$$P' = ModifyTemplate(P, \Delta), \tag{3.5}$$

де P' – модифікований шаблон, який враховує потенційні зміни; P - оригінальний шаблон (рядок або набір байтів, що відповідає певному вірусу); Δ - набір змін або варіацій, які дозволяються в шаблоні (наприклад, зміна символів, додавання випадкових байтів, шифрування тощо); операція ModifyTemplate може включати в себе такі кроки, як дозволені зміни в байтах, допущення помилок, шифрування або маскування деяких частин шаблону.

2. Пошук схожих рядків. Пошук у тексті S за допомогою адаптованої функції схожості:

$$Match(S, P') = \max_{i \in [1, m-n+1]} Sim(S[i, i+n-1], P'), \tag{3.6}$$

де S - текст, в якому здійснюється пошук (можливо, файл або програмний код, що містить віруси); P' - адаптований шаблон, змінений для врахування поліморфізму вірусів; $S[i, i+n-1]$ – підрядок довжиною n з тексту S, *а Sim (a, b)* – міра схожості між підрядками *а і b;* *Sim(a,b)* – міра схожості між підрядками a і b (як правило, функція, що оцінює, наскільки схожі два рядки або байти один на одного); $\max_{i\in[1,m-n+1]} Sim(S[i, i+n-1], P')$ - максимізація міри схожості серед усіх можливих підрядків *S[i,i+n−1]* у тексті S, де m - довжина тексту S, а n - довжина шаблону P'.

3. Оцінка ймовірності вірусу:

Якщо схожість між підрядком і шаблоном більша за певний поріг θ, тоді є ймовірність того, що знайдений рядок є частиною поліморфного вірусу:

$$P(Virus|S) = \begin{cases} 1, if\ Sim(S[i, i+n-1], P') > \theta \\ 0, if\ Sim(S[i, i+n-1], P') \leq \theta \end{cases}, \quad (3.7)$$

де *S[i,i+n−1]* - підрядок тексту S, який перевіряється на схожість із шаблоном P'; *Sim(a,b)* - міра схожості між підрядком а і шаблоном *b;* θ - поріг схожості, що визначає, наскільки схожим має бути підрядок для того, щоб його вважати частиною вірусу; P(Virus|S) - ймовірність того, що знайдений підрядок є частиною поліморфного вірусу.

Оскільки поліморфні віруси змінюють свою структуру, то важливо дозволити шаблону бути гнучким. Модифікація шаблону (P') може включати: додавання випадкових байтів; зміну порядку байт; використання шифрування для зміни рядка.

Для точнішого виявлення поліморфних вірусів необхідно використовувати функцію, яка оцінює не тільки точний збіг, а й схожість між підрядками. Наприклад: Levenshtein distance (редакційна відстань) вимірює мінімальну кількість операцій для перетворення одного рядка в інший; Longest Common Subsequence (LCS) знаходить найбільшу спільну підпослідовність між двома рядками; Jaccard Similarity вимірює подібність між двома множинами. Встановлення порогу для ймовірності того, що знайдений рядок є частиною вірусу, залежить від кількості змін, які допускаються у поліморфному вірусі.

Відобразимо алгоритм та узагальнену формулу для виявлення поліморфних вірусів за допомогою інтелектуального аналізу даних. Нехай $X = [x_1, x_2, \ldots, x_n]$ – набір характеристик (фіч) файлів, де кожен x_i – це вектор ознак (наприклад, байти файлу, хеші, системні виклики тощо); μ – середнє значення характеристик для нормальних файлів; σ – стандартне відхилення для нормальних файлів; $f(X)$ – функція класифікації, яка приймає вектор характеристик та виводить ймовірність того, що файл є вірусом. Тоді, процес виявлення поліморфних вірусів можна записати наступним чином.

1. Попередня обробка та векторизація (включає векторизацію файлу у вигляді числових характеристик (байтових послідовностей, хешів тощо):

$$X_i = ExtractFeatures(f_i), \tag{3.8}$$

де f_i – файл, а X_i – вектор характеристик файлу після попередньої обробки. i - це індекс, що позначає елемент у послідовності чи масиві (індекс поточного файлу, поточного рядка, або іншої одиничної характеристики, яку обробляє алгоритм).

2. Оцінка аномалії (в характеристиках файлів через відстань між ними та нормальним середнім):

$$Anomaly\ Score(Xi) = \frac{\|X_i - \mu\|}{\sigma}, \tag{3.9}$$

де $\| X_i - \mu \|$ – евклідова відстань між характеристиками файлу та середнім значенням характеристик для нормальних файлів, а σ – стандартне відхилення для нормальних файлів; i - це змінна, що вказує на поточний елемент у наборі файлів або ознак, яку обробляє алгоритм.

3. Класифікація за допомогою моделі машинного навчання (використання моделей для прогнозування ймовірності того, що файл є вірусом):

$$P(Virus \mid X_{\mathrm{i}}) = f(X_{\mathrm{i}}), \tag{3.10}$$

де $P(Virus|X_i)$ – ймовірність того, що файл є вірусом, яку видає модель, навчену на основі попередніх даних; i - це індекс або позначення конкретного файлу або зразка, для якого обчислюється ймовірність того, що він є вірусом, за допомогою навченої моделі.

4. Фінальний результат (клас файлу) (приймається рішення про клас файлу (вірус чи ні) на основі порогового значення ймовірності):

$$Class(f_{\text{i}}) = \begin{Bmatrix} Virus, if\ P(Virus \mid X_{\text{i}}) > \theta \\ Non - Virus, if\ P(Virus \mid X_{\text{i}}) \leq \theta \end{Bmatrix}, \quad (3.11)$$

де *Class(fi)* - це клас файлу f_i, який визначається на основі ймовірності того, що файл є вірусом; *P(Virus/Xi)* – ймовірність того, що файл f_i є вірусом, яка була обчислена за допомогою даного підходу; θ – це граничне значення ймовірності, який використовується для прийняття рішення про те, чи є файл вірусом чи ні.

Якщо ймовірність більше за θ, то файл класифікується як вірус, а якщо менше або дорівнює - як не вірус.

Ця формула узагальнює процес виявлення поліморфних вірусів, поєднуючи методи попередньої обробки даних, аналізу аномалій та машинного навчання.

Аналіз в пісочниці (sandboxing) є потужним методом для виявлення поліморфних вірусів. Пісочниця – це ізольоване середовище, в якому запускаються підозрілі програми чи файли для спостереження за їхньою поведінкою. Цей метод дозволяє виявити шкідливу активність, навіть якщо програма змінює свою структуру, бо вірус виконується в умовах, коли його реальні наслідки можна оцінити без ризику для основної системи.

Відобразимомо алгоритм та узагальнену формулу даного методу для виявлення поліморфних вірусів. Нехай $S = [s_1, s_2, \ldots, s_m]$ – набір характеристик поведінки програми (файлу) під час її виконання в пісочниці. Це можуть бути системні виклики, зміни в реєстрі, створення або зміна файлів, мережеві з'єднання, використання пам'яті, тощо. Y – змінна, яка позначає клас файлу: $Y = 1$ – файл є вірусом, $Y = 0$ – файл не є вірусом. *P (Y/S)* – ймовірність того, що файл є вірусом, зважаючи на його поведінку S. $A = [a_1, a_2, \ldots, a_k]$ – набір умов, які визначають шкідливу активність (наприклад, необґрунтовані мережеві з'єднання, зміни в критичних системних файлах тощо). *P* (A_i */Y)* – ймовірність того, що шкідлива активність a_i буде спостерігатися, якщо файл є вірусом. *P(S/Y)* – ймовірність спостереження поведінки S, якщо файл є вірусом.

Алгоритм застосування методу аналізу в пісочниці для виявлення поліморфних вірусів.

1. Ізоляція програми.

Файл або програма запускається в ізольованому середовищі (пісочниці), де спостерігається її взаємодія з операційною системою і різними компонентами. Набір даних S містить інформацію про всі операції, виконувані під час виконання програми в пісочниці.

2. Збір поведінкових характеристик.

Протягом виконання програми в пісочниці реєструються всі її дії, які можуть включати виклики системних функцій (наприклад, доступ до файлів, реєстру, мережі), запуск або зміна інших програм, зміни в файловій системі, створення нових файлів або змінення існуючих, мережеві активності (наприклад, встановлення з'єднань, передача даних), використання пам'яті (наприклад, виділення пам'яті, доступ до несанкціонованих областей пам'яті). Таким чином, набір характеристик S складається з усіх спостережених поведінкових ознак.

3. Визначення зловмисної активності.

Під час виконання програми в пісочниці можуть бути виявлені ознаки зловмисної поведінки. Це може включати необґрунтовані мережеві з'єднання (наприклад, на незнайомі або підозрілі сервери), модифікацію критичних системних файлів (реєстр, системні бібліотеки), запуск або спроби прихованого виконання (наприклад, спроби обходу моніторингу). Позначимо ці ознаки як $A = [a_1, a_2, \ldots, a_k]$, де кожен a_i є конкретною ознакою зловмисної активності.

4. Оцінка ймовірності вірусу.

Після того, як програма виконана і її поведінка проаналізована, можна обчислити ймовірність того, що файл є вірусом, використовуючи умовні ймовірності:

$$P(Y = 1 \mid S) = \frac{P(S|Y=1) \cdot P(Y=1)}{P(S)}, \tag{3.12}$$

де S - це спостережувана поведінка або характеристики файлу, які збираються під час його виконання. Це може бути набір дій, викликів системних функцій, зміни в реєстрах, доступ до мережі, використання ресурсів, тощо, які можуть бути

індикаторами того, що файл є вірусом. Наприклад, *S* може включати дані про те, як файл взаємодіє з операційною системою або іншими компонентами системи під час виконання; *Y* - означення стану файлу, яке може бути двома можливими варіантами (*Y*=1, якщо файл є вірусом; *Y*=0, якщо файл не є вірусом); *P(S/Y=1)* – ймовірність спостереження такої поведінки, якщо файл є вірусом; *P(Y=1)* – апріорна ймовірність того, що файл є вірусом; *P(S)* – ймовірність спостереження такої поведінки в цілому.

Оцінка *P(Y=1/S)* дозволяє визначити, наскільки висока ймовірність того, що файл є вірусом на основі його поведінки в пісочниці.

5. Класифікація файлу.

Якщо ймовірність того, що файл є вірусом, перевищує певний поріг θ, то файл класифікується як вірус:

$$Class(Y) = \left\{\frac{1, if\ P(Y=1|S)>\theta}{0, if\ P(Y=1|S)\leq\theta}\right\}, \quad (3.13)$$

де *Y* - означення стану файлу, яке може бути двома можливими варіантами (Y=1, якщо файл є вірусом; *Y*=0, якщо файл не є вірусом); *S* - це спостережувана поведінка файлу під час виконання в пісочниці; *Class(Y)* - це клас файлу (1 – файл класифікується як вірус (якщо ймовірність того, що файл є вірусом, перевищує поріг *θ*); 0 - файл класифікується як не вірус (якщо ймовірність того, що файл є вірусом, менша або дорівнює порогу *θ*); *P(Y=1/S)* - умовна ймовірність того, що файл є вірусом, на основі спостережуваної поведінки *S* (яка була отримана під час виконання файлу в пісочниці).

Машинне навчання є потужним інструментом для виявлення поліморфних вірусів, оскільки ці віруси можуть змінювати свою структуру, але зберігають спільні характеристики в поведінці або функціонуванні. Основною метою є побудова моделі, яка на основі навчальних даних виявлятиме вірусні властивості в файлах, навіть якщо їх структура була змінена.

Відобразимо алгоритм та узагальнену формулу даного методу. $X = [x_1, x_2, \ldots, x_n]$ – набір ознак, які використовуються для опису файлів. Це можуть бути статистичні характеристики (наприклад, хеш-сум, довжина файлу), поведінкові характеристики (під час аналізу в пісочниці або за допомогою динамічного аналізу), виклики функцій,

операції з пам'яттю, шаблони коду або інші характеристики. Нехай Y – змінна, яка позначає клас файлу: $Y = 1$, якщо файл є вірусом; $Y = 0$, якщо файл не є вірусом. $P(Y|X)$ – ймовірність того, що файл є вірусом $Y = 1$, враховуючи ознаки X. Моделі машинного навчання використовують різні моделі, наприклад, лінійні моделі (логістична регресія), дерева рішень, CSupport Vector Machines (SVM), нейронні мережі, Random Forest, K-ближніх сусідів (KNN). Нехай $D = \{(X_i, Y_i)\}$ – набір навчальних даних, де кожен елемент є парою (X_i, Y_i), що містить ознаки X_i для файлу та його клас Y_i.

Алгоритм застосування машинного навчання для виявлення поліморфних вірусів.

1. Збір і підготовка навчальних даних.

Для навчання моделі використовуються великі набори даних, що містять як позитивні (вірусні), так і негативні (невірусні) приклади. Ознаки файлів можуть бути отримані з різних джерел, таких як: статичний аналіз файлів (хеші, метадані), динамічний аналіз (поведінкові ознаки файлів в пісочниці), структурний аналіз коду. Таким чином, навчальний набір D складається з пар (X_i, Y_i).

2. Вибір моделі.

Вибирається модель машинного навчання, яка підходить для цієї задачі. Наприклад, для лінійних моделей, таких як логістична регресія:

$$P(Y = 1 \mid X) = \frac{1}{1+e^{-(w^T x+b)}}, \tag{3.14}$$

де X - це набір числових або категоріальних ознак, що описують файл або програму, яку потрібно класифікувати; Y - це мітка класу або результат класифікації (Y=1 - файл є вірусом, Y=0 - файл є не вірусом); w - це вектор ваг (параметр моделі), що відповідає кожній характеристиці файлу; x - це вектор ознак (характеристик), таких як байти файлу, хеші, системні виклики тощо; b - це зміщення або константа, що додається до лінійної комбінації; $P(Y=1|X)$ - це ймовірність того, що файл є вірусом, за умови, що X – це вектор ознак (характеристик) файлу; $w^T x + b$ - це лінійна комбінація векторів w (ваг) та x (характеристик), до яких додається зміщення b; T – функція трансформації стану.

Для дерев рішень або Random Forest, де модель приймає рішення, базуючись на значеннях ознак:

$$Y = DecisionTree(X), \tag{3.15}$$

де Y - це результат класифікації або прогнозу, який модель приймає (наприклад, чи є файл вірусом або ні); X - це вектор характеристик файлу, що містить ознаки або дані, на основі яких буде здійснена класифікація (наприклад, хеші, байти, системні виклики).

Для нейронних мереж:

$$Y = \sigma(W_2 \cdot \sigma(W_1 \cdot X + b_1) + b_2), \tag{3.16}$$

де Y - це вихід моделі, тобто результат класифікації, який може вказувати на те, чи є файл вірусом (наприклад, ймовірність того, що файл є вірусом); X - вхідний вектор характеристик файлу, що містить ознаки, на основі яких нейронна мережа буде приймати рішення (це можуть бути байти файлу, хеші, системні виклики та інші дані); W_1, W_2 - ваги нейронної мережі, які є параметрами, що навчання моделі коригує під час тренування, щоб оптимізувати класифікацію; b_1,b_2 - зсуви (bias), які також є параметрами, що моделюють зміщення в результатах класифікації; σ - це активаційна функція, яка обчислює результат для кожного нейрону (найпоширеніші функції активації: ReLU, Sigmoid, Tanh), яка дозволяє мережі навчатися складних патернів, додаючи нелінійність у модель.

3. Навчання моделі.

Під час навчання моделі обчислюється оптимальне значення параметрів (ваг), щоб мінімізувати функцію втрат (наприклад, крос-ентропію для задачі класифікації). Для цього використовуються алгоритми оптимізації, такі як градієнтний спуск:

$$w^{t+1} = w^t - \eta \nabla L(w^t), \tag{3.17}$$

де $w^{(t+1)}$ - це оновлені значення ваг w після (t+1)-го кроку оптимізації (після оновлення); w^t - це значення ваг на поточному кроці t; η - це швидкість навчання, яка визначає, наскільки сильно ваги будуть змінюватися на кожному кроці; L – функція

втрат; $\nabla L(w^t)$ - це градієнт функції втрат за поточними вагами w^t, який дає напрямок, в якому потрібно змінювати ваги, щоб зменшити значення втрат.

4. Оцінка моделі.

Після тренування моделі необхідно оцінити її ефективність за допомогою тестових даних. Основними метриками оцінки є точність (accuracy), точність (precision), відгук (recall), F1-міра, ґриві ROC і AUC. Для поліморфних вірусів важливо, щоб модель могла правильно класифікувати вірус, навіть якщо його структура була змінена, тому застосування метрики, такої як AUC, є важливим для оцінки ефективності.

5. Класифікація нових файлів.

Після того, як модель навчена і оцінена, вона може бути застосована для класифікації нових файлів:

$$P(Y = 1 \mid X) > \theta \Rightarrow Y = 1(\text{вірус}), \tag{3.18}$$

де *P(Y=1|X)* – ймовірність того, що файл *X* є вірусом (це значення виходить від навченого класифікатора після того, як модель обробить вхідний вектор характеристик *X* нового файлу); θ - це порогове значення ймовірності, яке визначає межу для прийняття рішення (якщо ймовірність *P(Y=1|X)* перевищує θ, то файл класифікується як вірус, а інакше файл вважається нешкідливим; *Y*=1 - це класифікація файлу як вірус (тобто файл заражений або містить шкідливий код); *Y*=0 - якщо ймовірність *P(Y=1|X)* менша або дорівнює θ, то файл класифікується як Non-Virus (нешкідливий).

Якщо ймовірність того, що файл є вірусом, перевищує певний поріг θ, то файл класифікується як вірус.

Метод структурних функцій для виявлення поліморфних вірусів ґрунтується на аналізі та порівнянні структурних властивостей програмного коду або файлів. Поліморфні віруси змінюють свою структуру для того, щоб уникнути виявлення традиційними методами на основі сигнатур. Однак, навіть після змін у структурі, вони зберігають певні спільні функціональні або структурні ознаки, які можуть бути використані для їх виявлення. У цьому контексті структурні функції

використовуються для розробки методів аналізу, що дозволяють оцінювати, наскільки схожі файли за певними ознаками (наприклад, за структурою коду, операціями, викликами функцій тощо).

Відобразимомо алгоритм та узагальнену формулу даного методу для виявлення поліморфних вірусів. Нехай F – множина структурних функцій, що визначають поведінку програми або файлу (наприклад, операції над даними, виклики функцій, алгоритмічні патерни тощо). $S = [s_1, s_2, \ldots, s_m]$ – набір характеристик (структурних функцій) файлу або програми, де кожна sis_isi є конкретною характеристикою, що описує певний аспект програми. Y – змінна, що позначає клас файлу (наприклад, вірус або не вірус), де $Y = 1$ – файл є вірусом, $Y = 0$ – файл не є вірусом. $P(Y|S)$ – ймовірність того, що файл є вірусом, враховуючи набір структурних функцій S.

Алгоритм застосування методу структурних функцій для виявлення поліморфних вірусів.

1. Визначення структурних функцій.

Спочатку потрібно визначити, які саме структурні функції будуть використовуватися для опису файлів або програм. Це можуть бути різноманітні операції, що виконуються файлом (наприклад, арифметичні операції, маніпуляції з пам'яттю), виклики функцій (якщо мова йде про програмне забезпечення), алгоритми, що використовуються (наприклад, шифрування, декодування), шаблони коду, які можуть бути спільними для вірусів.

Тому, структурні функції можна позначити як:

$$F = \{f1, f2, \ldots, fn\}, \tag{3.19}$$

де F - множина структурних функцій, які використовуються для виявлення поліморфних вірусів (набір функцій, кожна з яких описує певну характеристику або особливість файлу, яка може бути корисною для виявлення вірусу); f_i – це окрема структурна функція, що визначає певну характеристику поведінки або структури коду.

2. Формалізація структурних функцій.

Структурні функції можуть бути представлені через різні формалізми:

- аналіз потоку даних полягає в тому, що для кожної структурної функції f_i, яка описує операції з даними, визначається набір параметрів $p_i = [p_{i1}, p_{i2}, \ldots, p_{in}]$, що описують операції, що виконуються функцією;

- аналіз викликів функцій передбачає, що для кожної структурної функції, яка є викликом системної або бібліотечної функції, визначається її ймовірність та вплив на функціонування програми.

Таким чином, кожна структурна функція f_i може бути представлена як функція:

$$f_i(p_i) = operation(p_i), \tag{3.20}$$

де $operation(p_i)$ - це конкретна операція, що виконується в межах кожного агента чи модуля.

3. Оцінка ймовірності на основі структурних функцій.

Метою є обчислення ймовірності того, що файл є вірусом $P(Y=1/S)$, базуючись на наборі структурних функцій S, що описують його поведінку. Це можна зробити за допомогою Бейєсівської теореми або методів машинного навчання, наприклад:

$$P(Y = 1 \mid S) = \frac{P(S|Y=1) \cdot P(Y=1)}{P(S)}, \tag{3.21}$$

де S - це набір структурних функцій, що описують поведінку файлу; Y - це цільова змінна або мітка класу для файлу. Вона може набувати значень (Y=1, якщо файл є вірусом (позитивний клас) або Y=0, якщо файл не є вірусом (негативний клас)). $P(S/Y=1)$ – ймовірність спостереження структурних функцій S, якщо файл є вірусом (це ймовірність, що структурні функції відповідають типовим характеристикам вірусних програм); $P(Y=1)$ – апріорна ймовірність того, що файл є вірусом; $P(S)$ – ймовірність спостереження характеристик S для будь-якого файлу.

4. Класифікація файлу.

Якщо ймовірність того, що файл є вірусом, перевищує певний поріг θ, то файл класифікується як вірус.

Probabilistic Logic Networks (PLN) – це метод, який поєднує логічне виведення і ймовірнісні моделі для вирішення складних задач, таких як виявлення поліморфних

вірусів. Він може бути використаний для побудови моделі, яка виявляє аномалії або вираховує ймовірність того, що файл є вірусом на основі характеристик файлів.

Розглянемо алгоритм та узагальнену формулу застосування PLN для виявлення поліморфних вірусів. Нехай $X_1, X_2, \ldots, X_n$ – ознаки (характеристики) файлу, такі як байти, хеші, виклики системних функцій, шаблони коду, антивірусні підписи тощо; Y – змінна, що позначає клас файлу, де Y=1 – вірус, $Y = 0$ – не вірус. $P(Y|X_1, X_2, \ldots, Xn)$ – ймовірність того, що файл є вірусом ($Y = 1$), враховуючи характеристики $X_1, X_2, \ldots, X_n$. $P(X_i|Y)$ – умовна ймовірність характеристики Xi для класу Y (вірус чи не вірус). $P(Y)$ – апріорна ймовірність того, що файл є вірусом або не є вірусом. $P(X_1,X_2, \ldots,X_n|Y)$ – ймовірність спостереження характеристик $X_1, X_2, \ldots, X_n$, зважаючи на клас Y.

Кроки для застосування PLN.

1. Моделювання зв'язків між ознаками та класами.

Першим кроком є побудова ймовірнісної логічної мережі, яка описує зв'язки між ознаками $X_1, X_2, \ldots, X_n$ і класами файлів Y (вірус або не вірус). Це можна зробити за допомогою логічних операторів, таких як:

$$P(Y \mid X_1, X_2, \ldots, X_n) = \frac{P(X_1,X_2,\ldots,X_n|Y)\cdot P(Y)}{P(X_1,X_2,\ldots,X_n)}, \quad (3.22)$$

де $X_1, X_2, \ldots, X_n$ - це вхідні ознаки (або характеристики), які описують файл (кожна ознака Xi може бути різним параметром або характеристикою файлу); Y - це цільова змінна або мітка класу для файлу, яка вказує на те, чи є файл вірусом, чи ні; $P(Y|X_1, X_2, \ldots, X_n)$ - ймовірність того, що файл є вірусом (або не вірусом), враховуючи спостережувані ознаки $X_1, X_2, \ldots, X_n$ (ймовірність, яку необхідно обчислити); $P(X_1,X_2,\ldots,X_n|Y)$ - умовна ймовірність спостереження ознак $X_1, X_2, \ldots, X_n$, якщо файл є вірусом (або не вірусом), що є ключовим етапом у побудові моделі, оскільки потрібно визначити, як саме кожна ознака впливає на клас файлу; $P(Y)$ - апріорна ймовірність того, що файл є вірусом або не є вірусом (може бути розраховано на основі статистики або історичних даних, які описують співвідношення вірусних і невірусних файлів у системі; $P(X_1,X_2,\ldots,X_n)$ - ймовірність спостереження всіх ознак $X_1,X_2,\ldots,X_n$, незалежно від класу файлу (часто називають нормалізуючим фактором, оскільки він гарантує, що ймовірність буде коректно нормалізована).

Тут ймовірність того, що файл є вірусом, обчислюється за допомогою Бейєсівської теореми, яка дає можливість обчислити умовні ймовірності для кожної ознаки в контексті класу.

2. Визначення умовних ймовірностей.

Умовні ймовірності $P(X_i/Y)$ є критичними для моделювання. Для цього можна використовувати статистичні або машинно-навчальні методи для оцінки ймовірності кожної ознаки у класах вірусу та не вірусу:

$$P(Xi \mid Y) = probability\ of\ feature\ Xi\ given\ the\ class\ Y, \quad (3.23)$$

де X_i - це конкретна ознака файлу (наприклад, доступ до системних файлів, мережевих ресурсів, або інші поведінкові характеристики); Y – це клас файлу (вірус або не вірус).

Для поліморфних вірусів ці ймовірності можуть змінюватися в залежності від того, як сильно змінюються байти, шифрується код чи змінюються інші характеристики.

3. Виведення логічних зв'язків.

Після визначення умовних ймовірностей мережа може використовувати логічні зв'язки між ознаками для визначення ймовірності наявності вірусу. Це може бути реалізовано через застосування логічних операторів (AND, OR, NOT) для комбінації ознак:

$$P(Y = 1 \mid X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P(X_i \mid Y = 1) \cdot P(Y = 1), \quad (3.24)$$

де кожен $P\ (X_i/Y=1)$ визначає ймовірність спостереження конкретної ознаки при наявності вірусу, $P\ (Y=1)$ – це апріорна ймовірність того, що файл є вірусом.

4. Обчислення ймовірності зараження.

Для кожного файлу можна обчислити ймовірність того, що він є вірусом, використовуючи PLN:

$$P(Y = 1 \mid X_1, X_2, \dots, X_n) = \frac{\prod_{i=1}^{n} P(X_i|Y=1) \cdot P(Y=1)}{\prod_{i=1}^{n} P(X_i)}, \quad (3.25)$$

де $P(Y=1/X_1,X_2,\dots,X_n)$ - ймовірність того, що файл є вірусом, враховуючи спостережувані ознаки $X_1,X_2,\dots,X_n$; $P(X_i/Y=1)$ - умовна ймовірність ознаки X_i при

умові, що файл є вірусом; $P(Y=1)$ - апріорна ймовірність того, що файл є вірусом (це може бути оцінено на основі статистики або історичних даних); $P(X_i)$ - ймовірність спостереження ознаки X_i, незалежно від того, чи є файл вірусом або не вірусом.

Якщо результат $P(Y = 1/X_1, X_2, \ldots, X_n) > \theta$, то файл класифікується як вірус.

Для визначення ефективності запропонованої методики була проведена серія експериментів. Для отримання модифікованих поліморфних версій вірусів, використовували різні типи поліморфних генераторів. Усі поліморфні версії, створені ними генератори були скомпільовані з опціями антиналагодження та антиемуляції. Для проведення першого експерименту було згенеровано 100 вірусів. Щоб оцінити ефективність запропонованої методики, визначалося відсоткове співвідношення виявлених вірусів на кожному кроці запропонованого у дослідженні комплексного підходу.

Результати проведеного експерименту подані у табл. 3.2.

Таблиця 3.2

Відсоткове співвідношення виявлених вірусів на кожному кроці запропонованого комплексного підходу

Кількість згенерованих вірусів	Відсоток виявлених вірусів алгоритмами пошуку рядка (крок 1)	Відсоток виявлених вірусів методами кроку 2	Відсоток виявлених вірусів із використанням PLN (крок 3)	Діапазон ймовірність належності вірусів до поліморфних вірусів	Кількість вірусів у діапазоні ймовірності належності до ЗПЗ
100	12 %	61 %	89 %	0-25 % (low)	9 %
				25-75 % (medium)	19 %
				75-100 % (high)	72 %

Отже, на кроці 1 було виявлено лише 12 % вірусів, на кроці 2 – 61 %, а на кроці 3 із використанням PLN – 89 %. Ефективність запропонованої методики згідно проведеного експерименту становить 28 % завдяки використанню PLN. Також з 89 % виявлених вірусів за допомогою PLN 9 % було віднесено у діапазон ймовірності належності до ЗПЗ на рівні 0-25 % (низький рівень), на рівні 25-75 % (середній рівень)

- 19 %, на рівні 75-100 % - 72 % (високий рівень). Використання PLN дозволило не лише збільшити ефективність виявлення ЗПЗ, але й класифікувати за рівнем ймовірності належності до ЗПЗ.

Таким чином запропоновано комплексний підхід до виявлення та аналізу поліморфного ЗПЗ. Даний підхід складається з трьох етапів. На першому використовуються алгоритми пошуку рядка. На другому – комплекс методів, серед яких інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій. На третьому кроці пропонується використання PLN, які дозволять встановити ймовірність належності ПЗ до поліморфного ЗПЗ. Ефективність запропонованої методики згідно проведеного експерименту становить 28 % завдяки використанню PLN. Використання PLN дозволило не лише збільшити ефективність виявлення ЗПЗ, але й класифікувати за рівнем ймовірності належності до ЗПЗ.

Отже, розроблено метод виявлення поліморфних вірусів, який дозволяє, крім виявлення поліморфних вірусів, визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій (передбачено використання методів у різному порядку при кожному новому застосуванні цього етапу, що дозволяє забезпечити адаптивінсть до ситуації із врахуванням попереднього досвіду), на третьому - ймовірнісні логічні мережі, що дає змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів.

3.3. Метод класифікації поліморфних вірусів на основі нечіткого логічного висновку

Нехай є множина файлів, які виконуються F_i, $i = \overline{1,K}$, і можуть містити зловмисні коди. Нехай є множина ІА A_j , $j = \overline{1,L}$, які мають розпізнати файли, які виконуються. Навколишнє середовище NS представляє собою операційне середовище

комп'ютера, в якому взаємодіють як файли, які виконуються F_i з ІА A_j, так і ІА між собою. Передбачається, що для кожного файлу, що виконується F_i , $i = \overline{1,K}$ існує вектор ознак $S_i = [s_{i,1}, s_{i,2}, \dots, s_{i,k}]$ з k елементів, який може містити зловмисні коди. У кожного ІА A_j, $j = \overline{1,L}$ також є вектор ознак $C_j = [c_{j,1}, c_{j,2}, \dots, c_{j,l}]$ з l елементів, який визначає його самостійні цілі. При цьому вектори ознак і особливості програмних агентів, так і виконуваних файлів можуть відрізнятися один від одного.

Передбачається, що у ІА A_j є здатність ідентифікувати виконувані файли F_i в сенсорних областях за допомогою двовимірного масиву значень спорідненості (подібності).

$$SIM_{A_j,F_i} = \frac{1}{(1+P_{j,i})};\ j = \overline{1,L};\ i = \overline{1,K}, \tag{3.26}$$

де $P_{j,i} = \|A_j - F_i\|$ – евклідова відстань між агентом A_j та виконуваним файлом F_i; A_j - вектор ознак для ІА j; F_i - вектор ознак для виконуваного файлу i; L - кількість інтелектуальних агентів у системі; K - кількість виконуваних файлів, що аналізуються; $j=1,\dots,L$ - індекс агента A_j, де L - загальна кількість агентів у системі; i=1,…,K - індекс виконуваного файлу F_i, де K - загальна кількість виконуваних файлів, що підлягають ідентифікації. Евклідова відстань визначає міру схожості між агентом та файлом, де чим менша відстань, тим вищий рівень подібності.

Крім того, ІА також мають здатність повідомляти інформацію про виконувані файли іншим програмним агентам у комунікаційних областях. Проте, у даному дослідженні буде розглянуто окремий ІА в структурі мультиагентної системи.

Отже, задачею ІА є виявлення ЗПЗ (у даному дослідженні – поліморфного ЗПЗ), а також досить важливим питанням є визначення рівня складності поліморфного ЗПЗ. Тому, необхідним є здійснення нечіткої класифікації виявлених поліморфних вірусів згідно рівнів складності (рис.3.1).

Класифікацію виявлених поліморфних вірусів (нечітка класифікація) здійснимо згідно наступних кроків:

Крок 1. Визначення характеристик виявлених поліморфних вірусів та формування дерева логічного висновку.

Крок 2. Опис лінгвістичних змінних.

Крок 3. Визначення функцій належності лінгвістичних термів.

Крок 4. Формування бази знань системи нечіткого висновку.

Крок 5. Отримання ймовірності належності досліджуваного файлу до поліморфних вірусів різних рівнів складності.

Крок 6. Нечітка класифікація поліморфних вірусів.

Деталізуємо кроки даного алгоритму.

На кроці 1 були встановлені наступні показники для аналізу виявлених поліморфних вірусів: рівень шифрування коду (*CE*), рівень обфускації коду (*CO*), рівень використання самозмінюваного та динамічного коду (*SDC*), рівень захисту від аналізу та обходу пісочниць (*PAS*), рівень складності мережевої поведінки та взаємодії (*CNB*), рівень взаємодії із системою та реєстром (*ISR*), рівень адаптивної поведінки та самозахисту (*ABS*).

Відобразимо дерево логічного висновку на рис. 3.9.

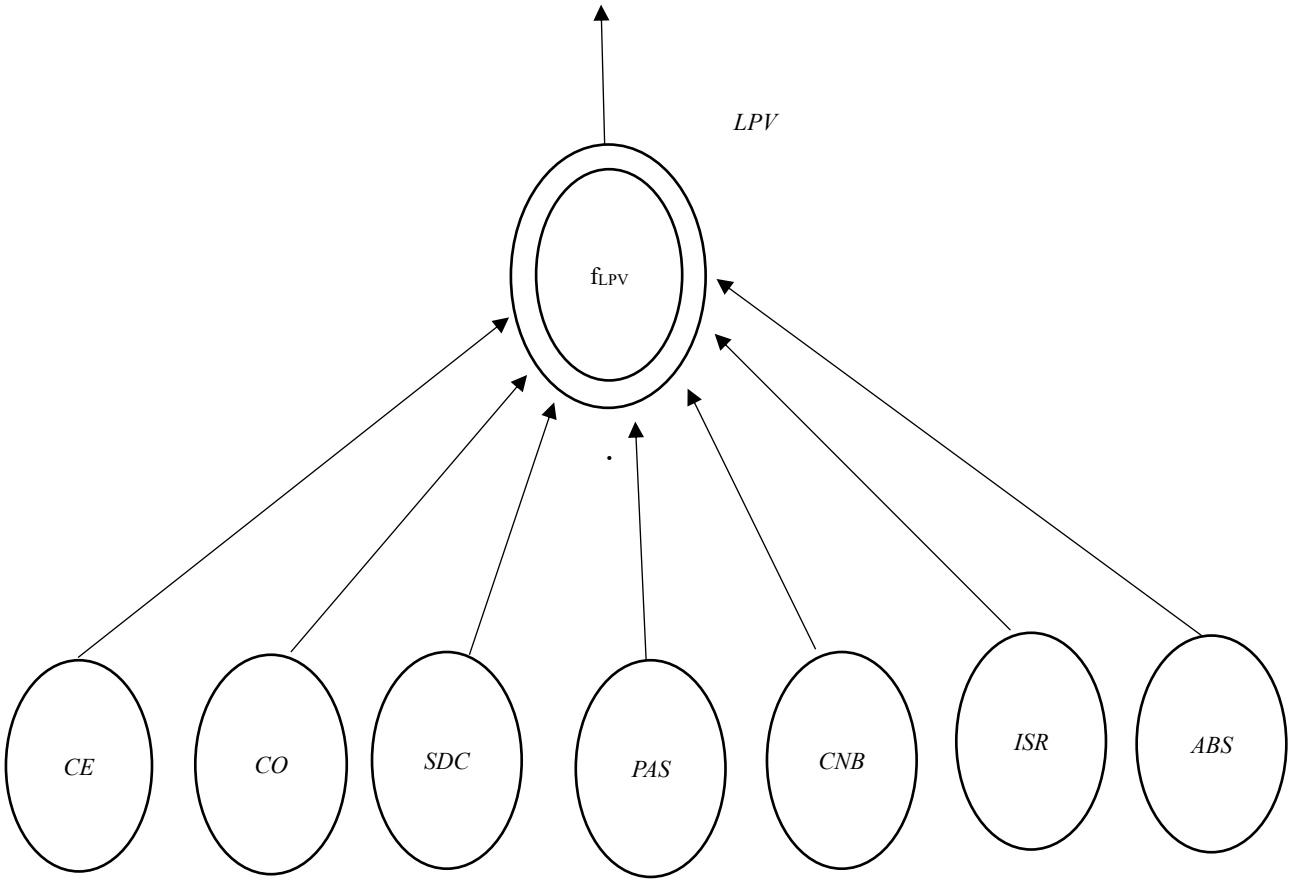

Рисунок 3.9 - Ієрархічне дерево логічного висновку для визначення рівня складності поліморфних вірусів (структурний вигляд моделі)

Наведеному на рис. 3.9 дереву логічного висновку відповідає таке співвідношення:

$$LPV = f_{LPV}(CE, CO, SDC, PAS, CNB, ISR, ABS), \quad (3.27)$$

де *LPV* – рівень складності поліморфного вірусу; *CE* - рівень шифрування коду; *CO* - рівень обфускації коду; *SDC* - рівень використання самозмінюваного та динамічного коду; *PAS* - рівень захисту від аналізу та обходу пісочниць; *CNB* - рівень складності мережевої поведінки та взаємодії; *ISR* - рівень взаємодії із системою та реєстром; *ABS* - рівень адаптивної поведінки та самозахисту.

FIS-структура співвідношення (3.27) у середовищі Matlab відображена на рис. 3.10.

На кроці 2 необхідно описати лінгвістичні змінні. Для цього деталізуємо кожен з обраних показників та його характеристику для різних рівнів складності поліморфних вірусів (табл. 3.3 – 3.9).

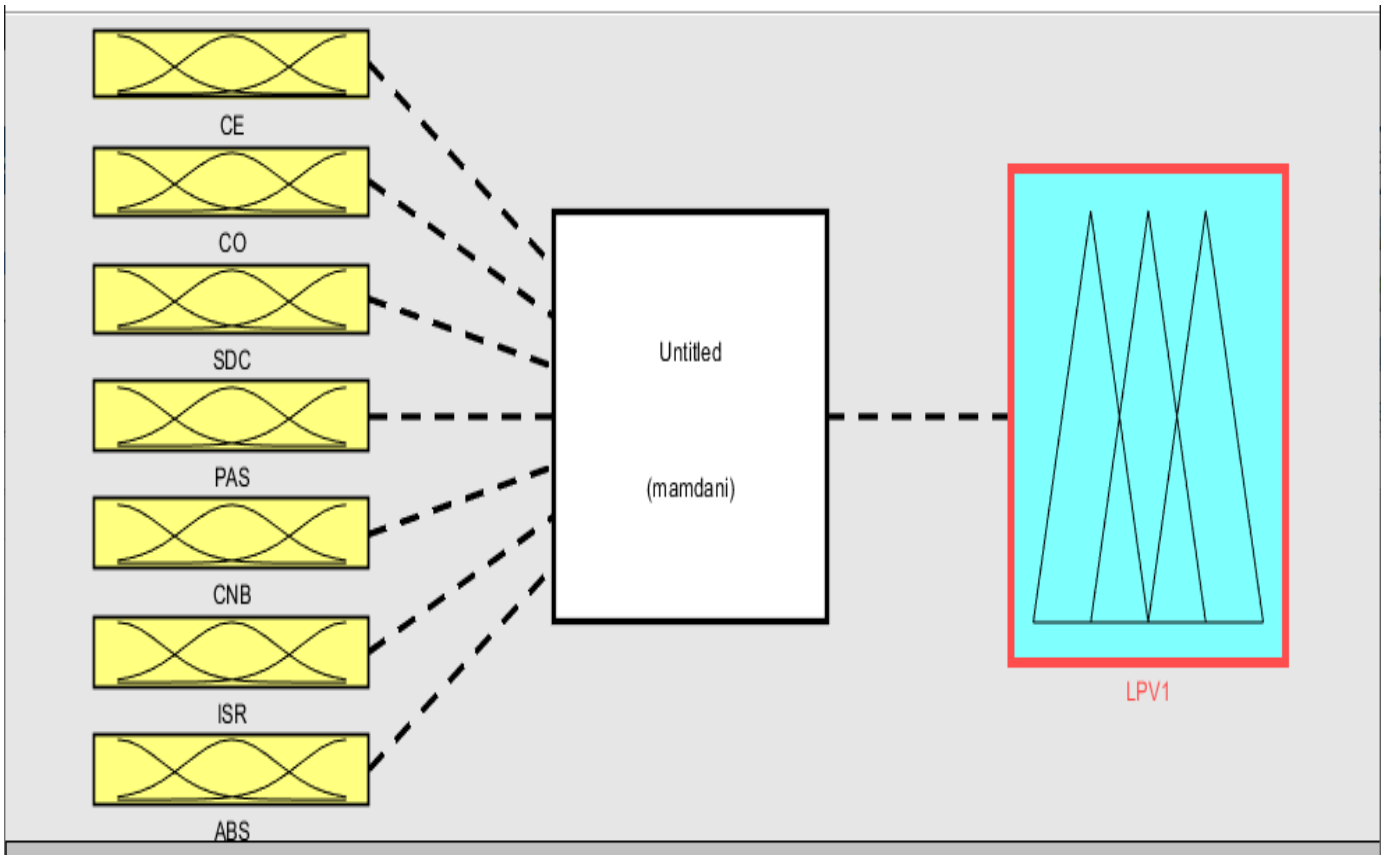

Рисунок 3.10 - FIS-структура співвідношення 3.27 у середовищі Matlab

Таблиця 3.3

Відповідність рівня шифрування коду рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Просте шифрування (наприклад, XOR, зсув бітів)	Рівень 1
Алгоритмічне шифрування (RC4, прості алгоритми)	Рівень 2
Використання асиметричних алгоритмів (RSA, ECC)	Рівень 3
Багаторівневе шифрування та динамічна генерація ключів	Рівень 4
Стеганографія для приховування даних і команд	Рівень 5
Самозмінюваний шифруючий код, який змінює структуру та алгоритми шифрування	Рівень 6

Таблиця 3.4

Відповідність рівня обфускації коду рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Проста обфускація (перейменування змінних, заміна команд)	Рівень 1
Середня обфускація (перепорядкування команд, приховування значень у рядках)	Рівень 2
Повна обфускація коду (заміна всіх функцій і змінних на випадкові значення, переписування логіки)	Рівень 3
Динамічна обфускація коду (обфускація на льоту, перемішування частин коду)	Рівень 4
Багаторівнева обфускація, приховані коди та алгоритми	Рівень 5
Повністю прихований код (змішування з іншими процесами, використання ядра)	Рівень 6

Таблиця 3.5

Відповідність рівня використання самозмінюваного та динамічного коду рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Просте змінювання коду (наприклад, додавання випадкових інструкцій)	Рівень 1
Генерація деяких частин коду під час виконання програми	Рівень 2
Повна динамічна генерація коду зі зміною алгоритмів	Рівень 3
Автоматичне оновлення або зміна коду з віддалених серверів	Рівень 4
Самоізмінювання коду, який адаптується до середовища, в якому він працює	Рівень 5
Повна самореплікація та переписування коду, що змінює саму природу вірусу	Рівень 6

Таблиця 3.6

Відповідність рівня захисту від аналізу та обходу пісочниць рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Простейший захист від дебаггерів (перевірка на наявність дебаггера)	Рівень 1
Використання базових методів захисту (наприклад, перевірка на віртуальні машини)	Рівень 2
Активне використання антидебаггінгових технік (перевірки на емулятори, зміна поведінки)	Рівень 3
Обхід пісочниць та емуляторів, динамічна поведінка	Рівень 4
Складні методи обходу пісочниць, робота в умовах суворої ізоляції	Рівень 5
Невиявленість у віртуальному середовищі та реальній системі (приховання в пам'яті, ядрі)	Рівень 6

Таблиця 3.7

Відповідність рівня складності мережевої поведінки та взаємодії рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Просте підключення до сервера або надсилання команд	Рівень 1
Використання нестандартних портів або протоколів для зв'язку	Рівень 2
Складні мережеві маніпуляції (маскування мережевої активності, використання прихованих каналів)	Рівень 3
Багаторівневий захист мережевого з'єднання (шифрування зв'язку, динамічна маршрутизація)	Рівень 4
Використання протоколів, прихованих за повсякденним трафіком (наприклад, DNS-тунелювання)	Рівень 5
Адаптація та приховання в мережевому середовищі, маскування серед нормальних операцій	Рівень 6

Таблиця 3.8

Відповідність рівня взаємодії з системою та реєстром рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Просте змінення метаданих або файлової структури	Рівень 1
Приховування файлів в системі або зміна реєстру	Рівень 2
Глибоке впровадження в систему, наприклад, через зміну системних процесів	Рівень 3
Повна модифікація реєстру та системних файлів, захист від змін	Рівень 4
Інтеграція з іншими програмами або процесами для приховування	Рівень 5
Впровадження в ядро або критичні системні процеси для приховування	Рівень 6

Таблиця 3.9

Відповідність рівня адаптивної поведінки та самозахисту рівням складності поліморфних вірусів

Характеристика показника	Рівні поліморфізму
Простіше змінювання поведінки залежно від системи	Рівень 1
Адаптація до різних типів операційних систем або мов програмування	Рівень 2
Динамічна зміна поведінки залежно від умов навколишнього середовища	Рівень 3
Адаптація вірусу в залежності від мережевої активності або поведінки користувача	Рівень 4
Використання складних алгоритмів ІІ для адаптації до атакуючих та зміни поведінки	Рівень 5
Повністю автономне поводження, яке самостійно приймає рішення на основі аналізу середовища та загроз	Рівень 6

У табл. 3.10 представлено узагальнення з різними рівнями прояву показників для кожного рівня поліморфізму.

Таблиця 3.10

Узагальнена таблиця з різними рівнями прояву показників для кожного рівня поліморфізму

Показник	Рівень 1	Рівень 2	Рівень 3	Рівень 4	Рівень 5	Рівень 6
Рівень шифрування коду	Просте шифрування (XOR)	Алгоритмічне шифрування (RC4)	Асиметричні алгоритми (RSA)	Багаторівневе шифрування	Стеганографія	Самоізмінюваний шифруючий код
Рівень обфускації коду	Проста обфускація	Середня обфускація	Повна обфускація	Динамічна обфускація	Багаторівнева обфускація	Повністю прихований код
Рівень використання самозмінюваного та динамічного коду	Просте змінювання коду	Генерація коду	Повна динамічна генерація	Автоматичне оновлення	Самоізмінюваний код	Повна самореплікація
Рівень захисту від аналізу та обходу пісочниць	Найпростіший захист	Базовий захист	Активне використання антидебаггінгових технік	Обхід пісочниць	Складні методи обходу пісочниць	Невиявленість у віртуальній середовищі
Рівень складності мережевої поведінки та взаємодії	Просте підключення	Нестандартні порти	Складні мережеві маніпуляції	Багаторівневий захист	Сховані канали	Адаптація та приховання в мережі
Рівень взаємодії із системою та реєстром	Просте змінення метаданих	Приховування файлів	Глибоке впровадження	Повна модифікація реєстру	Інтеграція з іншими програмами	Впровадження в ядро
Рівень адаптивної поведінки та самозахисту	Простіше змінювання поведінки	Адаптація до ОС	Динамічна зміна поведінки	Адаптація до мережевої активності	Алгоритми ІІ	Повністю автономне поводження

У табл. 3.11 подано опис лінгвістичних змінних.

Таблиця 3.11

Опис лінгвістичних змінних

Параметр	Назва лінгвістичної змінної (x)	Універсальна множина (U)	Лінгвістичні терми (T)
CE	Рівень шифрування коду	(0-6) балів	Level 1 (L1), Level 2 (L2), Level 3 (L3), Level 4 (L4), Level 5 (L5), Level 6 (L6)
CO	Рівень обфускації коду		
SDC	Рівень використання самозмінюваного та динамічного коду		
PAS	Рівень захисту від аналізу та обходу пісочниць		
CNB	Рівень складності мережевої поведінки та взаємодії		
ISR	Рівень взаємодії із системою та реєстром		
ABS	Рівень адаптивної поведінки та самозахисту		
LPV	Ймовірність належності вірусу до досліджуваного рівня складності поліморфних вірусів	(0-100) %	Low (L), Low Medium (LM), Medium (M), High Medium (HM), High (H)

На кроці 3 визначаються функції належності лінгвістичних термів.

Для всіх вхідних показників *CE, CO, SDC, PAS, CNB, ISR, ABS* використовувалася наступна функція належності (рис.3.11).

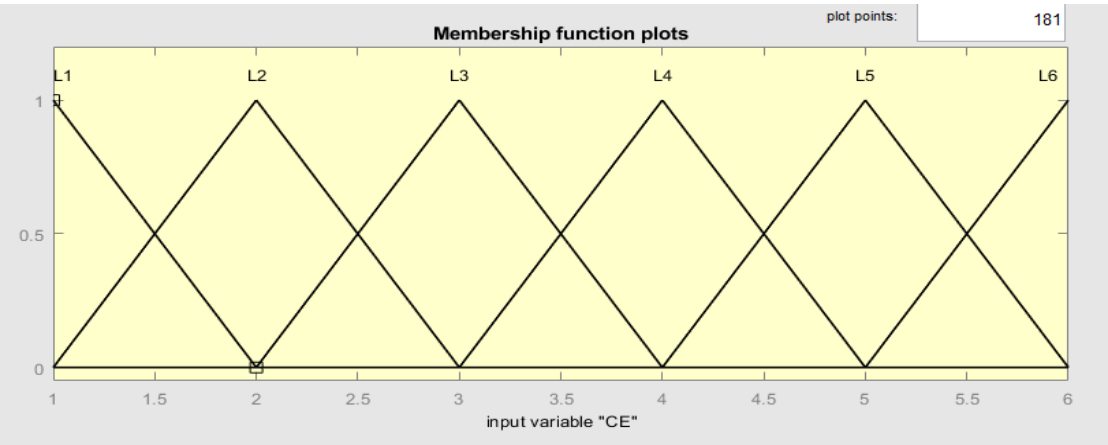

Рисунок 3.11 - Функція належності вхідних величин *CE, CO, SDC, PAS, CNB, ISR, ABS*

Належність поліморфного вірусу до певного рівня складності описується наступною множиною: B = {Low, Low Medium, Medium, High Medium, High}.

Приймаючи елементи множини як назви нечітких змінних, їх формально можна представити наступним чином:

$$B = \{b_1, b_2, b_3, b_4, b_5\}, \tag{3.28}$$

де b_1 – нечіткий терм Low; b_2 - нечіткий терм Low Medium; b_3 - нечіткий терм Medium; b_4 - нечіткий терм High Medium; b_5 - нечіткий терм High.

Нечіткі змінні:

$$<b_i, X_B, M^i_E>; \text{ де } M^i_E = \{x, \mu^i_E(x)\}; x \in X_B; X_B = [0; 100], \tag{3.29}$$

де b_i - нечіткий термін з множини B; X_B - область визначення, де значення змінної x лежать в інтервалі [0;100]; $M^i_E = \{x, \mu^i_E(x)\}$ - нечітка функція належності для змінної b_i, де $x \in X_B$ - значення змінної x в області визначення; $\mu^i_E(x)$ - функція належності для змінної b_i при значенні x.

На основі формули (3.29) введена лінгвістична змінна:

$$<"LPV", B, X_B>, \tag{3.30}$$

де "*LPV*" - рівень складності поліморфного вірусу; *B* містить дискретні терміни, які

описують рівень складності вірусу, що дозволяє нечітким чином класифікувати вірус;
X_B - область визначення, де значення змінної x лежать в інтервалі [0;100]

Набір функцій належності нечітких змінних (3.29) відображається у вигляді (рис. 3.12).

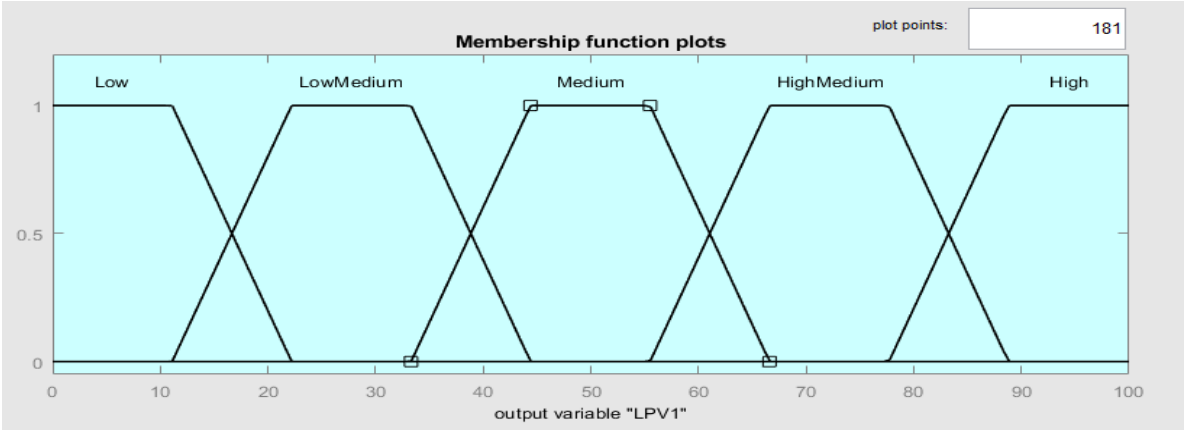

Рисунок 3.12 - Функції приналежності нечітких змінних лінгвістичної змінної ”*LPV*”

Таким чином, функції належності нечітких змінних мають такий узагальнений вигляд:

$$\mu^{\mathrm{i}}(\mathrm{x}) = \begin{Bmatrix} 0, \text{якщо } \mathrm{x}_i^d < x < x_i^a; \\ 1, \text{якщо } \mathrm{x}_i^b \le x \le x_i^c; \\ \frac{x - x_i^a}{x_i^b - x_i^a}, \text{якщо } \mathrm{x}_i^a \le x < x_i^b; \\ \frac{x_i^d - x}{x_i^d - x_i^c}, \text{якщо } \mathrm{x}_i^c < x \le x_i^d \end{Bmatrix}. \quad (3.31)$$

Для забезпечення високої ефективності роботи алгоритму класифікації найбільш важливим завданням є вибір області визначення функцій належності термів та способу завдання значень їхнього аргументу. Тому, пропонується використовувати для класифікації спосіб оцінювання лінгвістичної змінної, заснований на компонентах базових алгоритмів нечіткого висновку.

Крок 4. Формування бази знань системи нечіткого висновку. У табл. 3.12-3.14 відображено нечітку базу знань для моделювання належності виявленого вірусу до першого рівня складності поліморфних вірусів.

Таблиця 3.12

База правил для показника LPV (1 рівень складності)

Номер	CE	CO	SDC	PAS	CNB	ISR	ABS	LPV
1	L1	L1	L1	L1	L1	L1	L1	H
2	L2	L1	L1	L1	L1	L1	L1	H
3	L1	L2	L1	L1	L1	L1	L1	H
4	L1	L1	L2	L1	L1	L1	L1	H
5	L1	L1	L1	L2	L1	L1	L1	H
6	L1	L1	L1	L1	L2	L1	L1	H
7	L1	L1	L1	L1	L1	L2	L1	H
8	L1	L1	L1	L1	L1	L1	L2	H
9	L2	L2	L1	L1	L1	L1	L1	HM
10	L1	L2	L2	L1	L1	L1	L1	HM
11	L1	L1	L1	L2	L1	L2	L1	HM
12	L1	L2	L2	L1	L2	L1	L1	HM
13	L1	L1	L1	L3	L1	L1	L1	HM
14	L2	L2	L1	L2	L1	L2	L1	M
15	L1	L2	L1	L1	L3	L1	L3	M
16	L2	L2	L1	L1	L2	L2	L2	LM
17	L2	L2	L2	L2	L2	L2	L2	L
18	L3	L2	L2	L2	L3	L2	L2	L
19	L3	L4	L5	L5	L3	L5	L4	L
20	L2	L3	L2	L2	L3	L4	L5	L

Далі побудовано систему нечітких логічних рівнянь, перед формуванням якої необхідно врахувати таку особливість баз знань: рівність одиниці вагових коефіцієнтів усіх правил. Значення вагових коефіцієнтів усіх правил дорівнює одиниці, тому для зручності вагові коефіцієнти було вилучено з усіх логічних рівнянь.

Нечіткі логічні рівняння для бази знань для показника LPV (1 рівень складності) задамо так:

$$\mu^{H}(LPV) = \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L2}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L2}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L2}(ABS);$$

$$\mu^{HM}(LPV) = \mu^{L2}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L1}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L3}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L1}(ABS);$$

$$\mu^{M}(LPV) = \mu^{L2}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L1}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L1}(ABS) \vee \mu^{L1}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L3}(CNB) \wedge \mu^{L1}(ISR) \wedge \mu^{L3}(ABS);$$

$$\mu^{M}(LPV) = \mu^{L2}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L1}(SDC) \wedge \mu^{L1}(PAS) \wedge \mu^{L2}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L2}(ABS);$$

$$\mu^{L}(LPV) = \mu^{L2}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L2}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L2}(ABS) \vee \mu^{L3}(CE) \wedge \mu^{L2}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L3}(CNB) \wedge \mu^{L2}(ISR) \wedge \mu^{L2}(ABS) \vee \mu^{L3}(CE) \wedge \mu^{L4}(CO) \wedge \mu^{L5}(SDC) \wedge \mu^{L5}(PAS) \wedge \mu^{L3}(CNB) \wedge \mu^{L5}(ISR) \wedge \mu^{L4}(ABS) \vee \mu^{L2}(CE) \wedge \mu^{L3}(CO) \wedge \mu^{L2}(SDC) \wedge \mu^{L2}(PAS) \wedge \mu^{L3}(CNB) \wedge \mu^{L4}(ISR) \wedge \mu^{L5}(ABS).$$

де μ – функція належності; H – лінгвістичний терм *High (H)*; $L1$ – лінгвістичний терм *Level 1 (L1); L2* – лінгвістичний терм *Level 2 (L2);* $\vee(\wedge)$ – операція максимуму (мінімуму); *CE* - рівень шифрування коду; *CO* - рівень обфускації коду; *SDC* - рівень

використання самозмінюваного та динамічного коду; *PAS* - рівень захисту від аналізу та обходу пісочниць; *CNB* - рівень складності мережевої поведінки та взаємодії; *ISR* - рівень взаємодії із системою та реєстром; *ABS* - рівень адаптивної поведінки та самозахисту.

Представлення бази правил з табл. 3.12 у середовищі Matlab зображено на рис. 3.13.

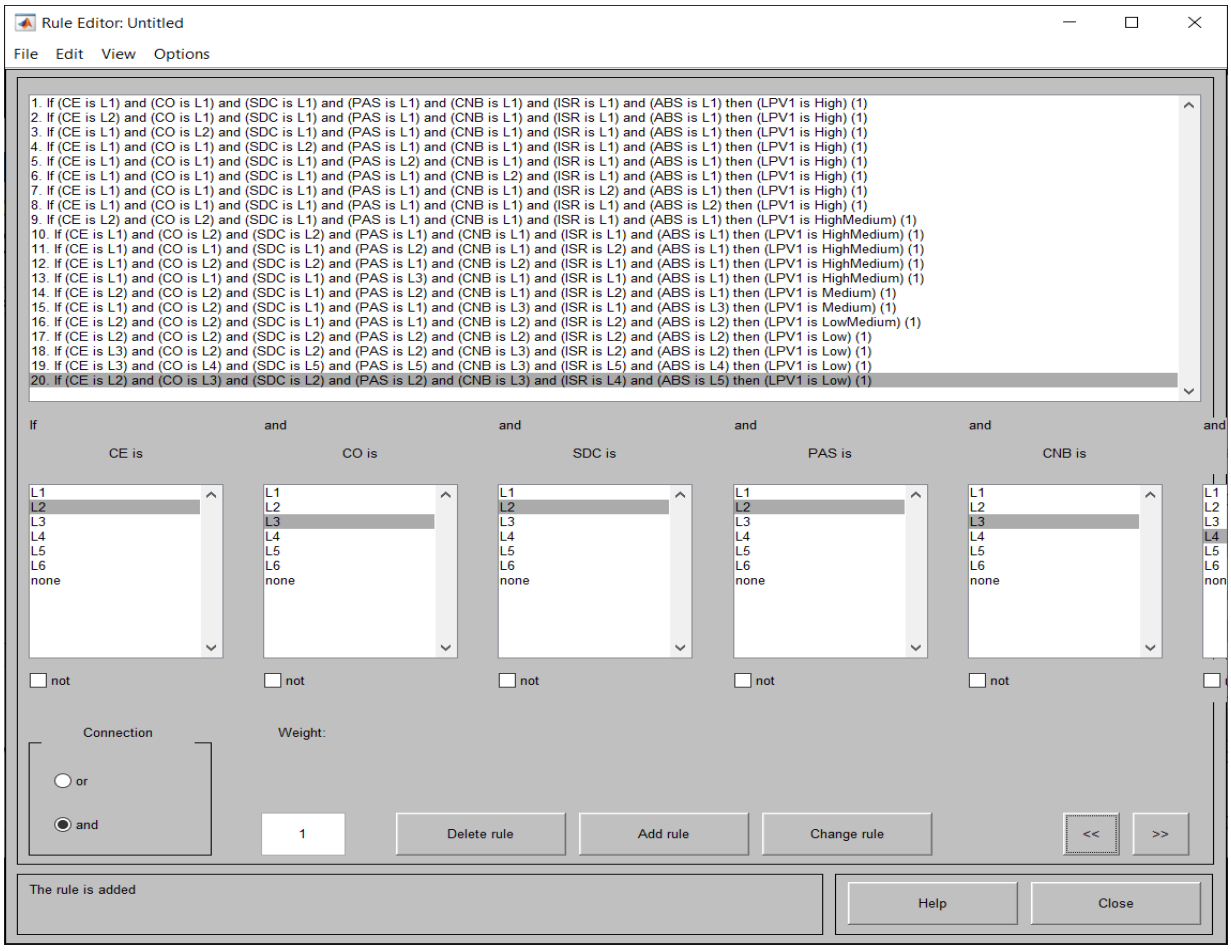

Рисунок 3.13 - Відображення бази правил у середовищі Matlab

На рис. 3.14 відображено графічне представлення результатів нечіткого логічного висновку в середовищі Matlab.

Також, були згенеровані бази знань для кожного з рівнів складності поліморфних вірусів, які враховують інтерпретацію значень складових вектору ознак $S_i = [s_{i,1}, s_{i,2}, \dots, s_{i,k}]$ у їх комплексному поєднанні для файлів, що виконуються.

В результаті використання нечіткого логічного висновку для кожного файлу, що виконується, було отримане числове значення (у %) ймовірності належності до

кожного з рівнів складності поліморфних вірусів.

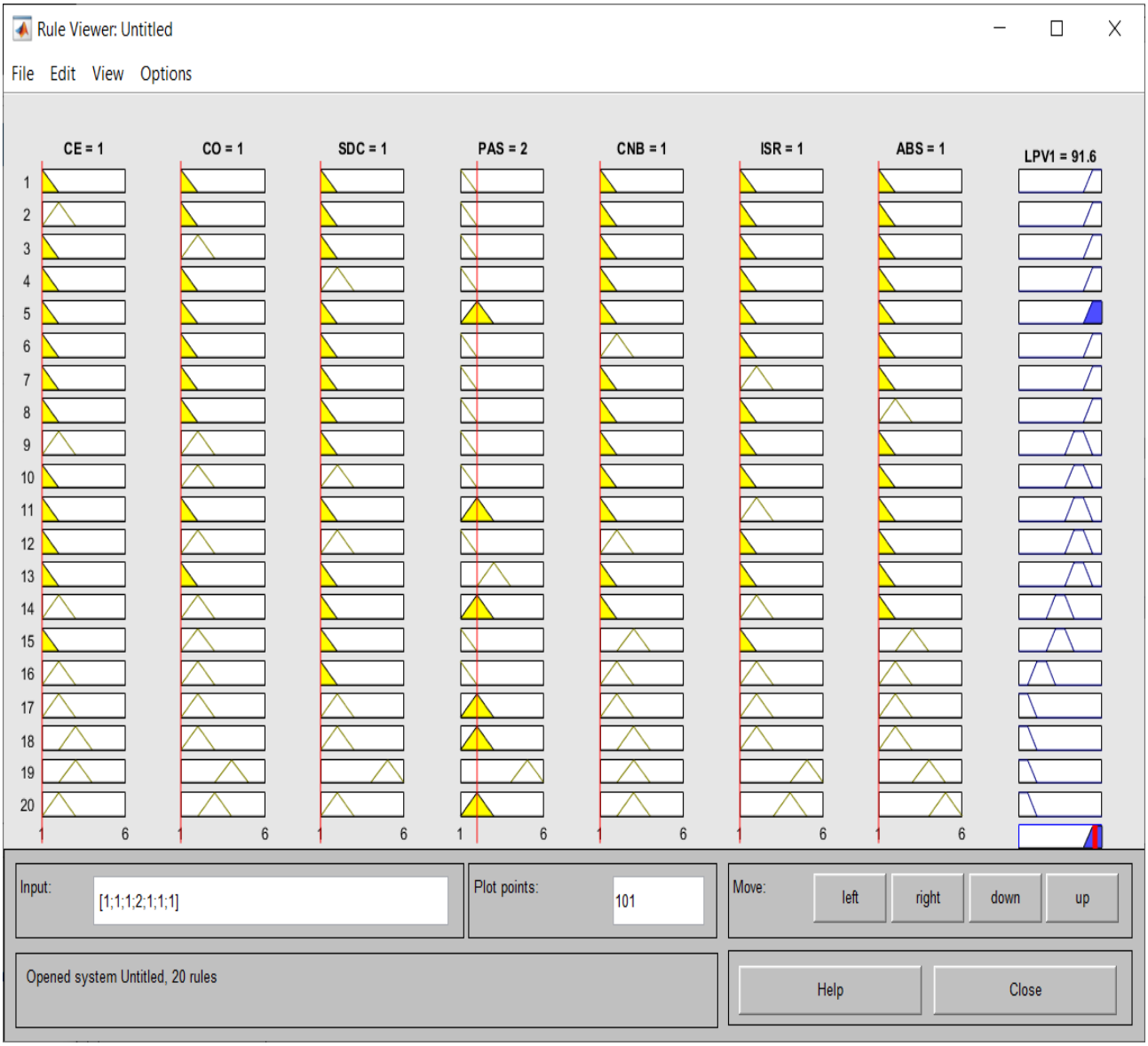

Рисунок 3.14 - Графічне представлення результатів нечіткого логічного висновку в середовищі Matlab

Для визначення ефективності запропонованої методики була проведена серія експериментів. З усіх виявлених поліморфних вірусів (89) даний підхід дозволив здійснити їх класифікацію згідно рівнів складності (всі 89). Також для перевірки надійності запропонованого підходу перевірялися файли, які не є поліморфним ЗПЗ. З 40 файлів, які не є поліморфним ЗПЗ, було отримано 100 % правильних висновків.

Нижче представлені точкові результати експерименту (табл. 3.13-3.15).

Таблиця 3.13

Результати експерименту для файлу, який не є поліморфним ЗПЗ

Рівень складності поліморфного зловмисного ПЗ	Розмірність універсуму (%)	Отримана ймовірність (%)	Значення функції належності для нечітких термів					Висновок
			Low	Low Medium	Medium	High Medium	High	
1	[0; 100]	0,00	0	0	0	0	0	Не є поліморфним ЗПЗ
2		0,00	0	0	0	0	0	
3		0,00	0	0	0	0	0	
4		0,00	0	0	0	0	0	
5		0,00	0	0	0	0	0	
6		0,00	0	0	0	0	0	

Таблиця 3.14

Результати експерименту для згенерованого поліморфного ЗПЗ (варіант 1)

Рівень складності поліморфного зловмисного ПЗ	Розмірність універсуму (%)	Отримана ймовірність (%)	Значення функції належності для нечітких термів					Висновок
			Low	Low Medium	Medium	High Medium	High	
1	[0; 100]	0,00	0	0	0	0	0	Є поліморфним ЗПЗ 2 рівня
2		75,00	0	0	0	1,00	0	
3		0,00	0	0	0	0	0	
4		0,00	0	0	0	0	0	
5		0,00	0	0	0	0	0	
6		0,00	0	0	0	0	0	

Таблиця 3.15

Результати експерименту для згенерованого поліморфного ЗПЗ (варіант 2)

Рівень складності поліморфного зловмисного ПЗ	Розмірність універсуму (%)	Отримана ймовірність (%)	Значення функції належності для нечітких термів					Висновок
			Low	Low Medium	Medium	High Medium	High	
1	[0; 100]	0,00	0	0	0	0	0	Є поліморфним ЗПЗ 4 рівня
2		0,00	0	0	0	0	0	
3		0,00	0	0	0	0	0	
4		64,00	0	0	0,24	0,76	0	
5		0,00	0	0	0	0	0	
6		0,00	0	0	0	0	0	

Таким чином, розроблено модель ІА із використанням нечіткої класифікації для виявлення та аналізу поліморфних ЗПЗ. Ефективність запропонованої методики, згідно проведеного експерименту, полягає в тому, що з усіх виявлених поліморфних вірусів (89) даний підхід дозволив здійснити їх класифікацію згідно рівнів складності (всі 89), а з 40 файлів, які не є поліморфним ЗПЗ, було отримано 100 % вірних

висновків. Тобто, даний підхід надав можливість виявити ті файли, які не є поліморфними вірусами, а із виявлених поліморфних вірусів здійснити їх класифікацію за рівнями складності із врахуванням належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього та високий, що є перевагою даного підходу. Виявлення належності поліморфного ЗПЗ до певного рівня складності дозволяє полегшити процес підбору необхідних методів для боротьби та їх знешкодження, у чому і полягає предмет наших подальших досліджень.

Особливістю розробленого методу класифікації поліморфних вірусів є не лише класифікація поліморфних вірусів за рівнями складності їх будови, але й можливість визначення ймовірності їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності.

3.4. Висновки до третього розділу

Запропоновано комплексний підхід до виявлення, аналізу та класифікації поліморфних вірусів, який складається з наступних етапів: етап моделювання процесу поширення поліморфних вірусів на основі моделі Лотки-Вольтерра; етап синтезу методів для виявлення поліморфних вірусів (алгоритму пошуку рядка, інтелектуального аналізу даних, аналізу в пісочниці, машинного навчання, методу розробки структурних функцій, ймовірнісних логічних мереж); етап нечіткої класифікації виявлених поліморфних вірусів.

Удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який, дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати.

Розроблено метод виявлення поліморфних вірусів, який дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми

пошуку рядка, на другому (використання методів у різному порядку при кожному наступному виконанні цього етапу, що дозволяє забезпечити адаптивінсть до ситуації) – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів.

Удосконалено метод класифікації поліморфних вірусів, який, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обгрунтованості та ефективність обраних стратегій для їх нейтралізації.

Основні результати розділу опубліковані у працях [23, 24, 151, 152, 153, 154].

РОЗДІЛ 4.
ГІБРИДНА МУЛЬТИАГЕНТНА СИСТЕМА ВИЯВЛЕННЯ ПОЛІМОРФНИХ ВІРУСІВ В КОМП'ЮТЕРНИХ МЕРЕЖАХ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

4.1. Гібридна мультиагентна система виявлення поліморфних вірусів в комп'ютерних мережах та методика визначення ефективності її функціонування

Гібридна МАС - це система, яка комбінує в собі різні типи агентів для вирішення комплексної задачі виявлення поліморфних вірусів, використовуючи різні методи та засоби. Дана гібридна МАС розгортається в локальній мережі (рис. 4.1), але також є можливість її використання в хмарі, системах IoT тощо. Нею можуть користуватись фахівці з кібербезпеки або професійні адміністратори корпоративних мереж.

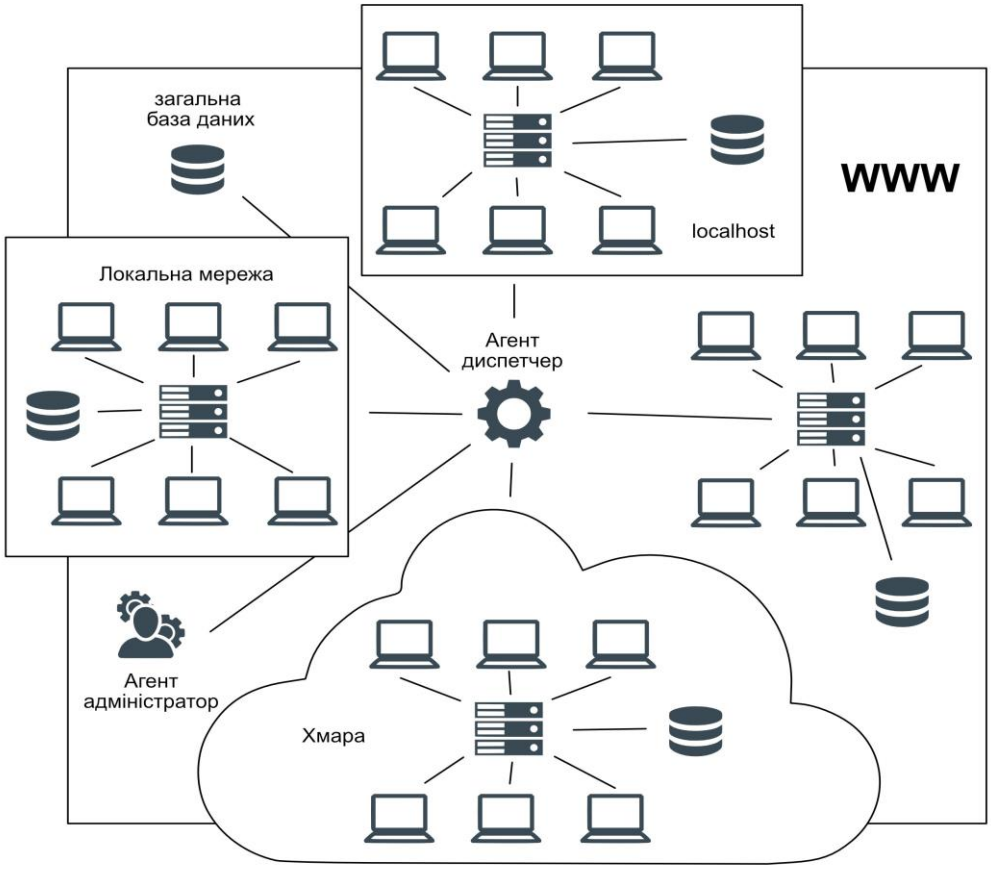

Рисунок 4.1 - Мережа з гібридною МАС

На рис. 4.2 зображена архітектура гібридної МАС для виявлення поліморфних вірусів. Гібридна МАС відображається як сукупність агентів $A = \{A_1, A_2, \ldots, A_n\}$, які взаємодіють із комп'ютерною системою та між собою.

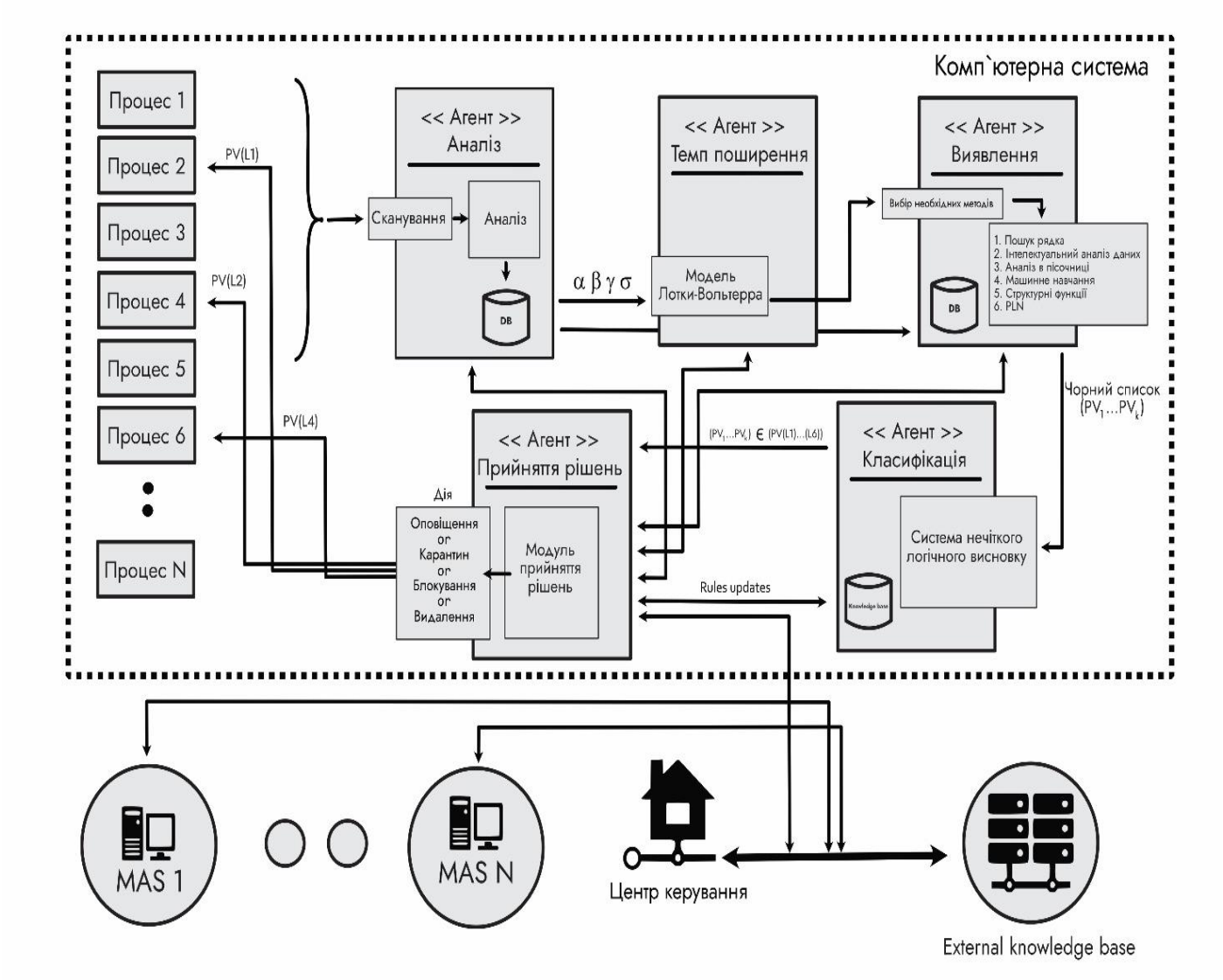

Рисунок 4.2 - Архітектура гібридної МАС виявлення поліморфних вірусів в комп'ютерних системах

Згідно рис. 4.2, запропонована гібридна МАС складається з наступних ІА: ІА «Аналіз»; ІА «Темп поширення»; ІА «Виявлення»; ІА «Класифікація»; ІА «Прийняття рішення». ІА виконують встановлені ролі та взаємодіють між собою.

ІА «Аналіз» перевіряє систему на наявність підозрілих або зловмисних дій, аналізуючи файлову систему, оперативну пам'ять, мережевий трафік та поведінку процесів. Він включає збір інформації, що стосується програмного забезпечення, та проведення глибокого аналізу для визначення характеру загрози, її джерела та

можливого впливу. В результаті аналізу системи ІА «Аналіз» формує та передає ІА «Темп поширення» значення коефіцієнтів $\alpha, \beta, \gamma, \delta$.

ІА «Темп поширення» здійснює моделювання процесу поширення поліморфних вірусів на основі моделі Лотки-Вольтерра. Даний агент досліджує темп поширення поліморфних вірусів та визначає, яку кількість та які методи виявлення поліморфних вірусів необхідно використати ІА «Виявлення».

ІА «Виявлення» використовує комплекс методів для виявлення поліморфних вірусів (кількість методів визначена ІА «Темп поширення»): алгоритми пошуку рядка; інтелектуальний аналіз даних; аналіз в пісочниці; машинне навчання; метод розробки структурних функцій; PLN, які визначені ІА «Темп поширення». Виявлені поліморфні віруси потрапляють у «чорний список», який надходить до ІА «Класифікація».

ІА «Класифікація» здійснює нечітку класифікацію виявлених поліморфних вірусів (ІА «Виявлення»), які потрапили у «чорний список» згідно шести рівнів їх складності.

Нечітка класифікація здійснюється за алгоритмом: визначення характеристик виявлених поліморфних вірусів та формування дерева логічного висновку; опис лінгвістичних змінних; визначення функцій належності лінгвістичних термів; формування бази знань системи нечіткого висновку; визначення ймовірності належності досліджуваного файлу до поліморфних вірусів різних рівнів складності; нечітка класифікація поліморфних вірусів.

Були встановлені наступні показники для аналізу виявлених поліморфних вірусів: рівень шифрування коду (CE); рівень обфускації коду (CO); рівень використання самозмінюваного та динамічного коду (SDC); рівень захисту від аналізу та обходу пісочниць (PAS); рівень складності мережевої поведінки та взаємодії (CNB); рівень взаємодії із системою та реєстром (ISR); рівень адаптивної поведінки та самозахисту (ABS).

В результаті використання нечіткого логічного висновку для кожного файлу, що виконується, було отримане числове значення (у %) ймовірності належності до кожного з рівнів складності поліморфних вірусів.

ІА «Прийняття рішення» використовує різні стратегії прийняття рішення (оповіщення, блокування, карантин, видалення), враховуючи типи загроз.

Алгоритм роботи запропонованої гібридної МАС зображено на рис. 4.3.

Спочатку спрацьовує ІА «Аналіз», в результаті роботи якого формується відповідь на питання «Чи є загроза (присутність поліморфних вірусів)?». Якщо загроза відсутня, то робота МАС припиняється із відповідним повідомленням про відсутність загроз. Якщо виявлена загроза, то здійснюється формування значень коефіцієнтів $\alpha, \beta, \gamma, \delta,$ які передаються ІА «Темп поширення».

ІА «Темп поширення» формує відповідь на питання «Яка кількість методів виявлення необхідна?» залежно від темпу поширення та поведінки поліморфних вірусів. Якщо відповідь «0», то робота гібридної МАС припиняється із відповідним повідомленням, що для виявлення поліморфних вірусів не потрібно використовувати жоден із методів. Якщо ж ІА згідно темпу поширення встановлює необхідну кількість n методів виявлення поліморфних вірусів (а їх може бути від 1 до 6, $n \in [1\ldots6]$), то інформація про їх встановлену кількість та обрані методи передається до ІА «Виявлення».

ІА «Виявлення» дозволяє визначити, з якою ймовірністю кожен з виявлених вірусів належить до поліморфних вірусів. В результаті роботи ІА дає відповідь на питання «Яка кількість поліморфних вірусів виявлена?». Якщо відповідь «0», то робота гібридної МАС припиняється із відповідним повідомленням, що жоден з виявлених вірусів не є поліморфним. В результаті застосування відповідних методів ІА формує кількість виявлених поліморфних вірусів m та передає до ІА «Класифікація».

ІА «Класифікація» на основі використання нечіткого логічного висновку формує відповідь на питання «Яка кількість поліморфних вірусів класифікована?». Якщо відповідь «0», то робота МАС припиняється із відповідним повідомленням, що жоден з вірусів не класифікований як поліморфний згідно рівнів складності. Якщо ж поліморфні віруси класифіковані ІА, то їх кількість k передається ІА «Прийняття рішення».

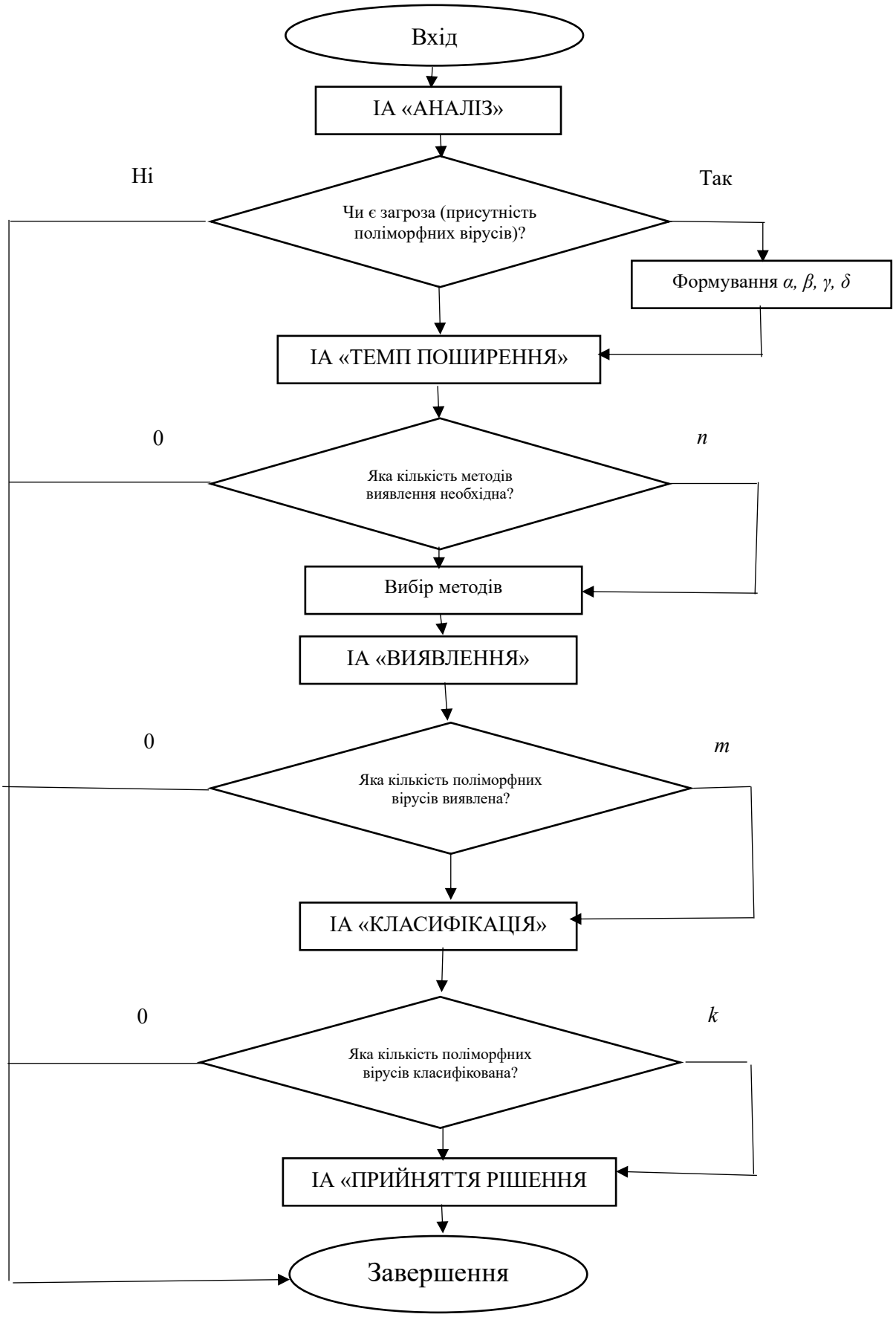

Рисунок 4.3 - Алгоритм роботи запропонованої гібридної МАС

ІА «Прийняття рішення» формує стратегії прийняття рішення згідно рівнів складності поліморфних вірусів (табл. 4.1).

Таблиця 4.1

Стратегії прийняття рішення ІА «Прийняття рішення» залежно від рівня складності поліморфних вірусів

Рівень складності поліморфних вірусів	Стратегії прийняття рішення
1	Оповіщення користувача, пропозиція ручної перевірки
2	Оповіщення користувача, логування
3	Автоматичне блокування процесу, карантин
4	Блокування, карантин, автоматичне надсилання на сервер аналізу
5	Видалення файлу, розрив підключення до мережі
6	Повна ізоляція системи, аналіз всієї системи, відновлення з резервної копії (за потреби)

Для визначення стратегії прийняття рішення використовуються продукційні правила типу:

IF (ймовірність належності до рівня 5 складності поліморфних вірусів > 70%) THEN видалити файл і відключити мережу.

Для розрахунку інтегрального показника ефективності гібридної МАС використано наступний підхід, який передбачає розгляд його в адитивній (E_A) та мультиплікативній формі (E_M).

В адитивній формі:

$$E_A = \sum_{i=1}^{n} a_i E_i^A, \tag{4.1}$$

де E_i^A – складові індикатори ефективності гібридної МАС для адитивної форми; n – кількість складових для визначення ефективності гібридної МАС; a_i – вагові коефіцієнти.

В мультиплікативній формі:

$$\text{с}\prod_{i=1}^{m}(E_i^M)^{a_i}, \tag{4.2}$$

де E_i^M – складові індикатори ефективності гібридної МАС для мультиплікативної форми; n – кількість складових для визначення ефективності гібридної МАС; a_i – вагові коефіцієнти.

Згідно формул (4.1) та (4.2) справедливе співвідношення:

$$\sum_{i=1}^{n} a_i = 1, a_i \geq 0, i = \overline{1,n}. \tag{4.3}$$

Також, в якості E_i^A та E_i^M можна розглядати індивідуальний показник ефективності кожного ІА в гібридній МАС. Для визначення ефективності запропонованої МАС розглядатимемо усереднене значення E_A та E_M:

$$E_{MAS} = \frac{E_A+E_M}{2}. \tag{4.4}$$

Отже, запропоновано використати двохвимірний підхід до визначення ефективності гібридної МАС.

Для визначення ефективності кожного ІА та визначення загальної ефективності гібридної МАС, необхідно розглянути основні показники ефективності для складових ІА в МАС (табл. 4.2).

Отже, базовими показниками ефективності гібридної МАС для виявлення поліморфних вірусів є показники, які можуть бути опрацьовані загальноприйнятними метриками.

Питома вага поліморфних вірусів, які були успішно виявлені гібридною МАС - True Positive Rate (*TPR*) або *Recall* (Повнота):

$$TPR(Recall) = \frac{TP}{TP+FN}, \tag{4.5}$$

де *TP* – True Positives (істинні позитиви), кількість поліморфних вірусів, які були вірно ідентифіковані МАС як загроза; *FN* – False Negatives (хибні негативи), кількість поліморфних вірусів, які система не розпізнала та помилково класифікувала як «чисті» файли.

Таблиця 4.2

Основні показники ефективності для кожного ІА

ІА	Мета	Показники ефективності
ІА «Аналіз»	Виявлення загроз і параметризація ($\alpha, \beta, \gamma, \delta$)	Точність виявлення (F1), швидкість аналізу, повнота
ІА «Темп поширення»	Правильність моделювання, стратегія реакції	Адекватність прогнозу, кореляція з реальним темпом
ІА «Виявлення»	Якість виявлення вірусів різними методами	Recall, FPR, час, точність при різних рівнях складності
ІА «Класифікація»	Точність нечіткої класифікації	Достовірність класифікації, стабільність при нових даних
ІА «Прийняття рішення»	Адекватна стратегія реагування	Частота помилкових рішень, співвідношення: блокування/хибна тривога

TPR є стимулятором, бо чим більше значення даного показника, тим краще працює система і тим менше поліморфних вірусів залишаються непоміченими.

Показник, що відображає хибні спрацьовування МАС – False Positive Rate (*FPR*):

$$FPR = \frac{FP}{FP+TN}, \tag{4.6}$$

де *FP* – False Positives (хибні позитиви), кількість «чистих» файлів, які гібридна МАС помилково ідентифікувала вірусами; *TN* – True Negatives (істинні негативи), кількість «чистих» файлів, які система правильно ідентифікувала як безпечні.

FPR є дестимулятором, бо чим менше значення даного показника, то тим краще працює система і тим менше хибних блокувань.

Показник точності (Precision):

$$Precision = \frac{TP}{TP+FP}. \quad (4.7)$$

Precision є стимулятором, бо чим більше значення даного показника, то тим краще працює система і тим менше помилкових тривог гібридної МАС.

Середнє гармонійне (*F1-score*):

$$F1 - score = 2 \cdot \frac{Precision \cdot TPR(Recall)}{Precision + TPR(Recall)}. \quad (4.8)$$

F1-score є стимулятором, бо чим більше значення даного показника, то тим краще працює система і тим краще система одночасно балансує між мінімізацією хибних спрацьовувань (*Precision*) і максимальним виявленням справжніх загроз (*Recall*).

Якщо *F1-score* = 1.0 (100 %), то це означає оптимальну роботу гібридної МАС, адже жодного пропущеного вірусу і жодної хибної тривоги не спостерігалося. Якщо *F1-score* → 0, то це означає, що гібридна МАС або постійно пропускає віруси (*Recall* ≈ 0), або постійно генерує хибні спрацьовування (*Precision* ≈ 0).

Точність загалом (*Accuracy*) відображає частку правильно класифікованих файлів серед усіх:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (4.9)$$

Чутливість до чистих (Specificity) програм відображає наскільки добре система уникає хибних тривог:

$$Specificity = \frac{TN}{TN+FP}. \quad (4.10)$$

NPV (Negative Predictive Value) – ймовірність того, що «чистий» результат дійсно чистий:

$$NPV = \frac{TN}{TN+FN}. \quad (4.11)$$

FNR (False Negative Rate) – питома вага пропущених загроз:

$$FNR = \frac{FN}{FN+TP} = 1 - Recall. \quad (4.12)$$

FDR (False Discovery Rate) – питома вага хибних тривог серед усіх тривог:

$$FDR = \frac{FP}{FP+TP} = 1 - Precision. \quad (4.13)$$

MCC (Matthews Correlation Coefficient) – це показник якості бінарної класифікації, який ураховує всі чотири комірки матриці сплутаності («+1» – оптимальна класифікація; «0» – випадкове вгадування; «-1» – усі класифікації невірні):

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(FP+TP)(TP+FN)(TN+FP)(TN+FN)}}. \quad (4.14)$$

Також, необхідно визначити вагові коефіцієнти кожного ІА в гібридній МАС. Для цього був використаний експертний метод. Було долучено 10 експертів (фахівців з кібербезпеки та захисту інформації ІТ-компаній). Для оцінювання рівня компетентності експертів при їх обранні було використано тестовий метод. Міра наближення еталонної шкали і шкали відповідей обраних експертів коливається в межах від 90 % до 100 %.

При експертному оцінюванні використано метод одночасного ранжування. Експертам запропоновано проранжувати всі ІА, які є складовими в гібридній МАС стосовно їх вагомості. Найважливіший, з точки зору експерта, ІА одержує ранг «1», менш суттєвий − ранг «2» і т.д. Якщо експерт вважає, що декілька ІА є однаково вагомими, то він може присвоїти їм однакові ранги.

У табл. 4.3 відображено результати роботи експертів (матриця ранжування).

Таблиця 4.3

Матриця ранжування

ІА	Експерт									
	1	2	3	4	5	6	7	8	9	10
ІА «Аналіз»	3	5	3	2	3	4	3	3	3	3
ІА «Темп поширення»	2	4	4	2	2	4	4	3	2	4
ІА «Виявлення»	1	1	1	1	1	1	1	1	1	1
ІА «Класифікація»	2	2	1	1	1	2	1	2	1	2
ІА «Прийняття рішення»	4	3	2	1	4	3	2	3	3	5

Оскільки в матриці ранжування (табл. 4.3) є ранги, які збіжні (так звані “зв’язані”), то приведемо її спочатку до нормального виду (табл. 4.4). У нормальній матриці ІА, які в оцінюванні експерта мають однакові ранги, надають ранг, що дорівнює середньому значенню тих місць, які ці ІА поділили між собою.

Так, експертом 1 для ІА «Темп поширення» та ІА «Класифікація» був наданий однаковий ранг – «2». У результаті приведення матриці до нормального виду їм надають ранг «2,5» ((2+3)/2=2,5), а для ІА «Аналіз» вже присвоюється ранг «4» (4 місце), а для ІА «Прийняття рішення» - ранг «5». Аналогічно перетворення відбулося і для результатів оцінок інших експертів.

Таблиця 4.4

Нормальна матриця ранжування ІА в гібридній МАС

ІА	Експерт										Сума рангів	Середній ранг	Ранг фактора	Ваговий коефіцієнт
	1	2	3	4	5	6	7	8	9	10				
ІА «Аналіз»	4	5	4	4,5	4	4,5	4	4	4,5	3	41,50	4,15	4	0,085
ІА «Темп поширення»	2,5	4	5	4,5	3	4,5	5	4	3	4	39,50	3,95	5	0,105
ІА «Виявлення»	1	1	1,5	2	1,5	1	1,5	1	1,5	1	13,00	1,30	1	0,370
ІА «Класифікація»	2,5	2	1,5	2	1,5	2	1,5	2	1,5	2	18,50	1,85	2	0,315
ІА «Прийняття рішення»	5	3	3	2	5	3	3	4	4,5	5	37,50	3,75	3	0,125

Після перетворення матриці ранжування на нормальну матрицю ранжування слід перевірити узгодженість думок експертів згідно коефіцієнта конкордації.

У даному випадку:

$$W = \frac{132{,}25 + 90{,}25 + 289 + 132{,}25 + 56{,}25}{\frac{1}{12} \cdot 100 \cdot 5 \cdot (5^2 - 1)} = \frac{700}{1000} = 0{,}7.$$

Значення W свідчить про узгоджені думки експертів ($W \geq 0{,}7$).

Для визначення коефіцієнта вагомості кожного ІА в структурі гібридної МАС використано формулу:

$$a_i = \frac{2(mn - S_i)}{mn(n-1)}, \qquad (4.15)$$

де m – кількість експертів, які прийняли участь в оцінюванні; n – кількість ІА, які оцінюються; S_i – сума рангів оцінок для i-го ІА.

В результаті у табл. 4.5 та на рис. 4.4 представлено отримані вагові коефіцієнти для ІА в структурі запропонованої МАС.

Отже, можна встановити, що найбільш вагомим, на думку експертів, є ІА «Виявлення», на другому місці – ІА «Класифікація», на третьому – ІА «Прийняття рішення», на четвертому – ІА «Темп поширення», на п’ятому – ІА «Аналіз».

Таблиця 4.5

Вагові коефіцієнти ІА в структкурі гібридної МАС

ІА	Ваговий коефіцієнт
ІА «Аналіз»	0,085
ІА «Темп поширення»	0,105
ІА «Виявлення»	0,370
ІА «Класифікація»	0,315
ІА «Прийняття рішення»	0,125

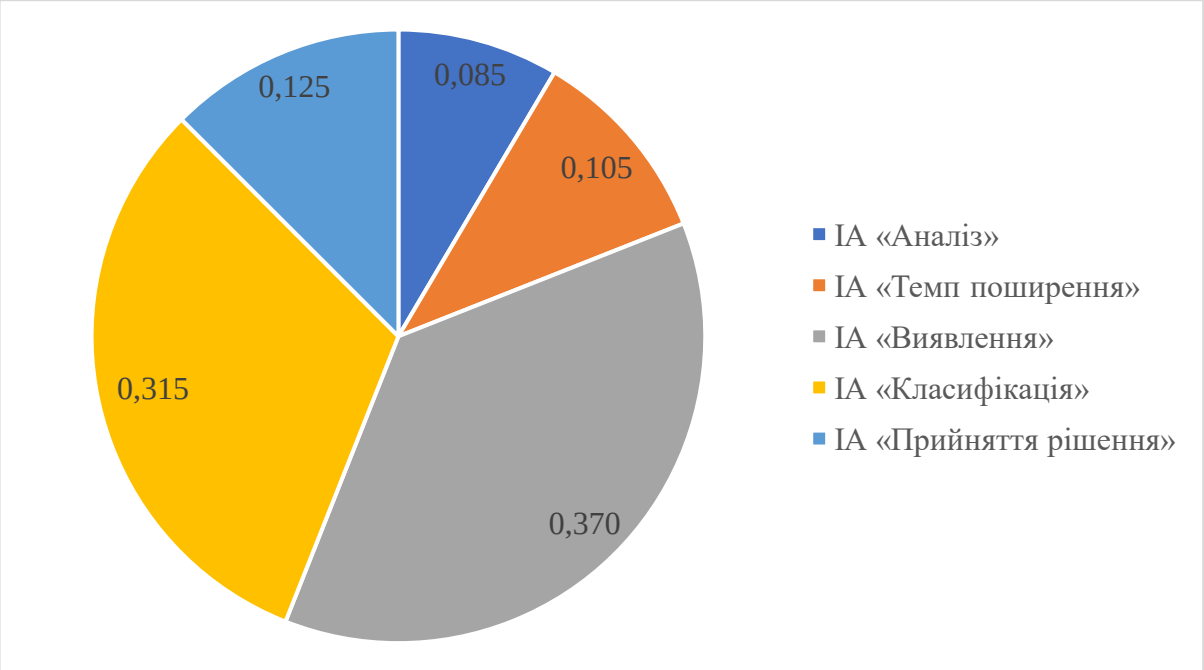

Рисунок 4.4 - Вагові коефіцієнти ІА в структкурі гібридної МАС

Для ІА «Аналіз» обрано базовий показник для оцінювання ефективності – TPR_A (при виявленні підозрілих файлів), для ІА «Темп поширення» - RE (частка виявлення

агресивних темпів поширення), для ІА «Виявлення» – TPR_P (при виявленні поліморфних вірусів), для ІА «Класифікація» - *CA* (Classification accuracy) (частка коректно класифікованих поліморфних вірусів), для ІА «Прийняття рішення» - *DM* (Decision making) (частка знешкоджених поліморфних вірусів серед виявлених).

Тому, узагальнена формула для визначення ефективності гібридної МАС у адитивній формі виглядатиме наступним чином:

$$E_A = TPR_A * 0{,}085 + RE * 0{,}105 + TPR_P * 0{,}37 + CA * 0{,}315 + DM * 0{,}125. \quad (4.16)$$

Узагальнена формула для визначення ефективності гібридної МАС у мультиплікативній формі виглядатиме наступним чином:

$$E_M = {TPR_A}^{0{,}085} \cdot RE^{0{,}105} \cdot {TPR_P}^{0{,}37} \cdot CA^{0{,}315} \cdot DM^{0{,}125}. \quad (4.17)$$

Далі розглянуто експерименти та результати експериментальних досліджень з розробленою гібридною МАС для виявлення та класифікації поліморфних вірусів.

Використаємо також другий підхід, де E_i^A – складові індикатори ефективності гібридної МАС для адитивної форми, E_i^M – складові індикатори ефективності гібридної МАС для мультиплікативної форми. Складовими індикаторами ефективності МАС розглянемо індикатор виявлення та індикатор класифікації.

Індикатор виявлення поліморфних вірусів передбачає розрахунок комплексної оцінки виявлення поліморфних вірусів, а індикатор класифікації - розрахунок комплексної оцінки класифікації поліморфних вірусів різних рівнів складності. Обрані показники для комплексної оцінки виявлення та класифікації поліморфних вірусів відображені у табл. 4.6.

Таблиця 4.6

Показники для комплексної оцінки виявлення та класифікації поліморфних вірусів

Індикатор ефективності гібридної МАС	Показники
Індикатор виявлення поліморфних вірусів (E_d)	TPR_P, *Precision, F1-score*
Індикатор класифікації поліморфних вірусів (E_c)	$TPR_{PV(L1)}$, $TPR_{PV(L2)}$, $TPR_{PV(L3)}$, $TPR_{PV(L4)}$, $TPR_{PV(L5)}$, $TPR_{PV(L6)}$

Показники $TPR_{PV(L1)}$, $TPR_{PV(L2)}$, $TPR_{PV(L3)}$, $TPR_{PV(L4)}$, $TPR_{PV(L5)}$, $TPR_{PV(L6)}$ відображають питому вага поліморфних вірусів кожного з шести рівнів складності, які були успішно виявлені гібридною МАС.

Для визначення вагових коефіцієнтів у даному підході запропоновано використати вагові коефіцієнти Фішберна. Для системи переваг, що знижуються N альтернатив:

$$p_i = \frac{2(N-i+1)}{(N+1)N}, i = 1..N. \quad (4.18)$$

А системі рівнозначних один одному N альтернатив – комплекс однакових ваг:

$$p_i = N^{-1}, i = 1..N. \quad (4.19)$$

Далі побудовано ієрархічне дерево індикаторів ефективності МАС та їх складових показників із використанням системи відношень переваг на основі опитувань експертів (рис. 4.5), при чому: }= - відношення переваги; ≈ - відношення рівноваги.

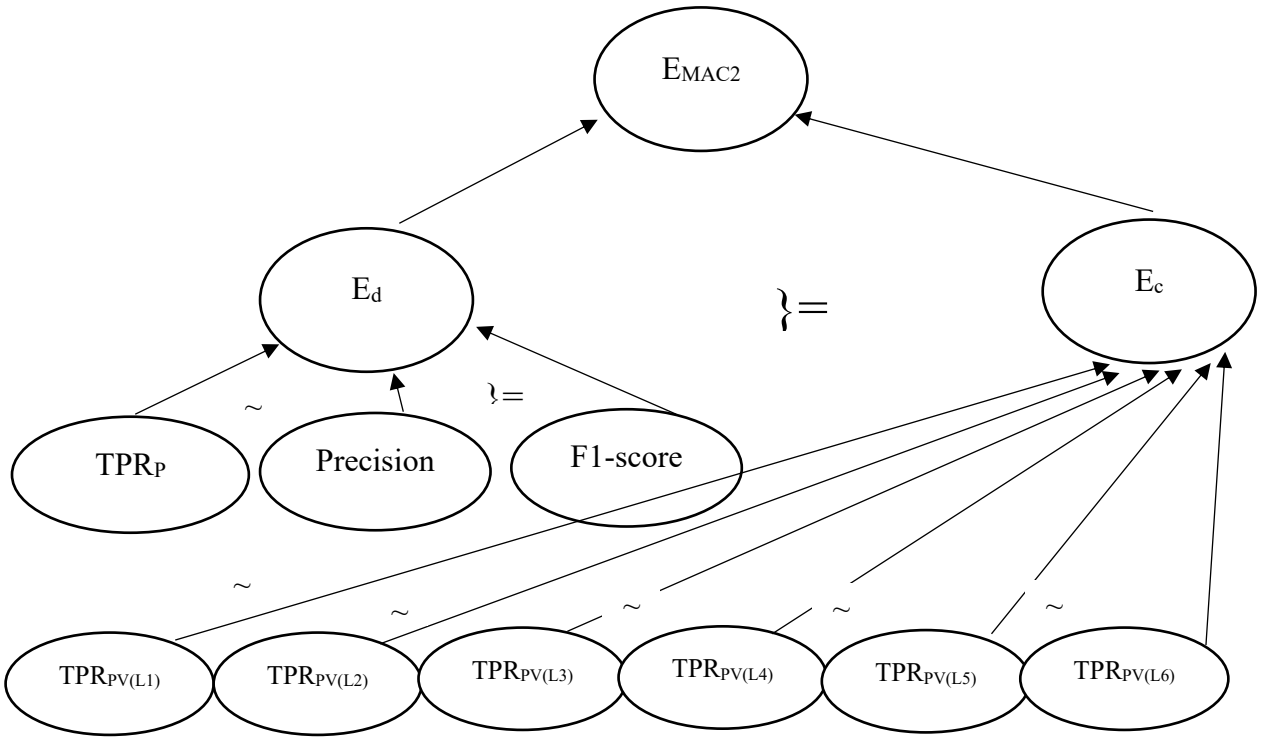

Рисунок 4.5 - Ієрархічне дерево показників ефективності гібридної МАС із зазначенням системи відношень переваг

Рис. 4.5 відповідає системі співвідношень Е так:

$$E=\{E_d\}E_c;\ TPR_P \approx Precision\}\ F1\text{-}score;\ TPR_{PV(L1)} \approx TPR_{PV(L2)} \approx$$
$$\approx TPR_{PV(L3)} \approx TPR_{PV(L4)} \approx TPR_{PV(L5)} \approx TPR_{PV(L6)}\} \quad (4.20)$$

Отримана система відношень переваг дозволяє використати для визначення вагових коефіцієнтів показників ефективності гібридної МАС ваг Фішберна.

Згідно формул (4.17) та (4.18), а також системи відношень переваг (рис. 4.4) сформована система ваг Фішера (табл. 4.7). Провівши згортку вагових коефіцієнтів різних рівнів ієрархії було отримано наступні узагальнені вагові коефіцієнти (рівень впливу) для показників: TPR_P - 0,267; *Precision* - 0,267; *F1-score* - 0,133; для $TPR_{PV(L1)}$, $TPR_{PV(L2)}$, $TPR_{PV(L3)}$, $TPR_{PV(L4)}$, $TPR_{PV(L5)}$, $TPR_{PV(L6)}$ – 0,056.

Таблиця 4.7

Система ваг Фішберна

N	*E*	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*
2	$E_d\}E_c$	2/3	1/3	-	-	-	-
3	$TPR_P \approx Precision\}$ *F1-score*	2/5	2/5	1/5	-	-	-
6	$TPR_{PV(L1)} \approx TPR_{PV(L2)} \approx TPR_{PV(L3)} \approx TPR_{PV(L4)} \approx \approx TPR_{PV(L5)} \approx TPR_{PV(L6)}$	1/6	1/6	1/6	1/6	1/6	1/6

На рис. 4.6 представлено встановлені вагові коефіцієнти показників індикаторів ефективності гібридної МАС.

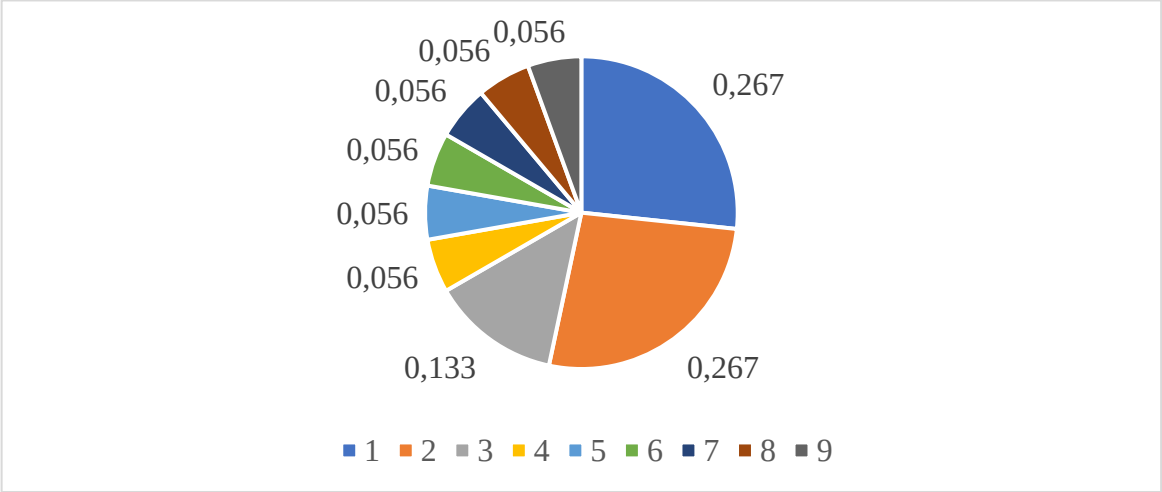

Рисунок 4.6 - Вагові коефіцієнти показників індикаторів ефективності гібридної МАС *(1 - TPR_P, 2 – Precision, 3 - F1-score, 4 - $TPR_{PV(L1)}$, 5 - $TPR_{PV(L2)}$, 6 - $TPR_{PV(L3)}$, 7 - $TPR_{PV(L4)}$, 8 - $TPR_{PV(L5)}$, 9 - $TPR_{PV(L6)}$)*

Згідно рис. 4.6 можна помітити, що вагові коефіцієнти для TPR_P та Precision на першому місці, для *F1-score* – на другому, а для $TPR_{PV(L1)}$, $TPR_{PV(L2)}$, $TPR_{PV(L3)}$, $TPR_{PV(L4)}$, $TPR_{PV(L5)}$, $TPR_{PV(L6)}$ – на третьому.

4.2. Постановка експериментів та результати експериментальних досліджень з гібридною мультиагентною системою виявлення поліморфних вірусів

Проведемо серію експериментів та відобразимо результати експериментальних досліджень з гібридною МАС виявлення поліморфних вірусів з метою визначення її ефективності. Експеримент 1 передбачає, що було згенеровано 1000 зразків даних (файлів), серед яких 300 – це поліморфні віруси, змінені з часом; 700 – чисті (не заражені) файли. Було згенеровано поліморфні віруси від першого до п’ятого рівнів складності.

Результати підходу 1 для визначення ефективності гібридної МАС із врахуванням ефективності ІА МАС відображено у табл. 4.8.

Таблиця 4.8

Ефективність інтелектуальних агентів гібридної МАС(експеримент 1)

Агент	Опис виконуваних функцій	Кількісні результати	Ефективність (%)	Формула / Розрахунок
ІА «Аналіз»	Перевірка 1000 файлів, попереднє виявлення підозрілих (аналіз файлової системи, пам’яті, мережі, процесів)	Із 300 вірусів правильно виявлено 294 (*TP*), помилково підозрює 20 чистих (*FP*)	98,0%	*TPR = TP / (TP + FN)* = 294 / 300
ІА «Темп поширення»	Визначення рівня загрози вірусів на основі моделі Лотки-Вольтерра	Агресивні темпи поширення точно визначено для 27 з 30 випадків	90,0%	*RE* = 27 / 30
ІА «Виявлення»	Поглиблене виявлення методами: ML, аналіз у пісочниці, PLN тощо	Із 294 підозрілих 289 підтверджено як поліморфні віруси, 5 чистих помилково (*FP*)	96,3%	*TPR = TP / (TP + FN)* = 289 / 300
ІА «Класифікація»	Нечітка класифікація за 5 рівнями складності (враховано показники *CE, CO, SDC*...)	276 з 289 коректно класифіковано	95,5%	*Classification accuracy* = 276 / 289
ІА «Прийняття рішення»	Реакція на виявлені загрози (блокування, карантин тощо)	Із 289 виявлених – 274 знешкоджено, 15 лише попереджено	94,8%	*Decision making* = 274 / 289

Згідно табл. 4.8 розраховано ефективність МАС:

$$E_{A1} = 0{,}98 * 0{,}085 + 0{,}9 * 0{,}105 + 0{,}963 * 0{,}37 + 0{,}955 * 0{,}315 + 0{,}948 * 0{,}125 = 0{,}9536.$$

$$E_{M1} = 0{,}98^{0{,}085} \cdot 0{,}9^{0{,}105} \cdot 0{,}963^{0{,}37} \cdot 0{,}955^{0{,}315} \cdot 0{,}948^{0{,}125} = 0{,}9534.$$

$$E_{MAC1} = \frac{0{,}9536 + 0{,}9534}{2} = 0{,}9535.$$

Результати розрахунків подано у табл. 4.9.

Таблиця 4.9

Ефективність гібридної МАС (підхід 1) із врахуванням ефективності ІА в МАС (експеримент 1)

ІА	Ефективність агента	Ваговий коефіцієнт агента	Ефективність MAS (адитивна система)	Ефективність MAS (мультиплікативна система)	Усереднена ефективність MAS (E_{MAS})
«Аналіз»	0,980	0,085	0,9536	0,9534	0,9535
«Темп поширення»	0,900	0,105			
«Виявлення»	0,963	0,370			
«Класифікація»	0,955	0,315			
ІА «Прийняття рішення»	0,948	0,125			

Згідно даних з табл. 4.9 можна встановити, що використання запропонованого підходу 1 (коли ефективність гібридної МАС визначається через визначення ефективності складових ІА), при роботі із поліморфними вірусами від першого до п’ятого рівнів складності, ефективність гібридної МАС при розгляді її як адитивної системи, складає 95,36 %, ефективність гібридної МАС при розгляді її як мультиплікативної системи – 95,34 %, а усереднене значення становить 95,35%, що є досить високим результатом.

У табл. 4.10 відображена деталізація кількості поліморфних вірусів за рівнями складності.

Визначимо TPR_P, *Precision, F1-score* згідно отриманих даних в цьому експерименті.

$$TPR(Recall) = \frac{289}{289 + 11} = 0{,}9633.$$

$$Precision = \frac{289}{289+5} = 0{,}9830.$$

$$F1 - score = 2 \cdot \frac{0{,}9830 \cdot 0{,}9633}{0{,}9830 + 0{,}9633} = 0{,}9731.$$

Таблиця 4.10

Таблиця з деталізацією кількості поліморфних вірусів за рівнями складності (експеримент 1)

Рівень складності вірусу	Кількість зразків	Правильно виявлено (ІА «Виявлення»)	Правильно класифіковано (ІА «Класифікація»)	Ефективність виявлення (%)	Ефективність класифікації (%)
Рівень 1	80	80	79	100,00%	98,75%
Рівень 2	70	69	67	98,57%	97,10%
Рівень 3	60	58	55	96,67%	94,83%
Рівень 4	50	46	42	92,00%	91,30%
Рівень 5	40	36	33	90,00%	91,67%
Усього / Середнє	300	289	276	95,45%	94,73%

Оскільки в даному експерименті було згенеровано поліморфні віруси п’яти рівнів складності, тому вагові коефіцієнти ефективності їх класифікації будуть на рівні 0,067 (видозміна рисунку 4.4 та зміна розрахунку ваг Фішберна, що представлено у табл. 4.11).

Таблиця 4.11

Система ваг Фішберна (при наявності поліморфних вірусів від першого до п’ятого рівнів складності)

N	*E*	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*
2	$E_d\}E_c$	2/3	1/3	-	-	-	-
3	$TPR_P \approx Precision\}$ *F1-score*	2/5	2/5	1/5	-	-	-
6	$TPR_{PV(L1)} \approx TPR_{PV(L2)} \approx TPR_{PV(L3)} \approx TPR_{PV(L4)} \approx \approx TPR_{PV(L5)}$	1/5	1/5	1/5	1/5	1/5	-

Згідно запропонованої методики (підхід 2) обчислимо ефективність гібридної МАС у адитивній формі із врахуванням індикаторів ефективності так:

$$E_{A2} = 0{,}267 * 0{,}9633 + 0{,}267 * 0{,}9830 + 0{,}133 * 0{,}9731 + 0{,}067 * (0{,}9875 + 0{,}9710 + 0{,}9483 + 0{,}9130 + 0{,}9167) = 0{,}9664 = 96{,}64\%.$$

Обчислимо ефективність МАС у мультиплікативній формі (підхід 2) так:

$E_{\text{M2}} = 0{,}9633^{0{,}267} \cdot 0{,}9830^{0{,}267} \cdot 0{,}9731^{0{,}133} \cdot (0{,}9875^{0{,}067} \cdot 0{,}9710^{0{,}067} \cdot 0{,}9483^{0{,}067} \cdot 0{,}9130^{0{,}067} \cdot 0{,}9167^{0{,}067}) = 0{,}9642 = 96{,}42\%$. Визначимо ефективність гібридної МАС за другим підходом, усереднивши ці значення:

$$E_{MAS2} = \frac{0{,}9664 + 0{,}9642}{2} = 0{,}9653 = 96{,}53\%$$

Зведемо результати екперименту 1 для підходу 1 та 2 у табл. 4.12 та на рис. 4.7.

Таблиця 4.12

Узагальнення результатів визначення ефективності гібридної МАС згідно запропонованих підходів (експеримент 1)

Ефективність гібридної МАС	Підхід 1 (врахована ефективність ІА в гібридній МАС)	Підхід 2 (враховані індикатори ефективності гібридної МАС)	Усереднене значення за двома підходами
E_a (адитивна)	95,36%	96,64%	96,00%
E_M (мультиплікативна)	95,34%	96,42%	95,88%
E_{MAS} (усереднена)	95,35%	96,53%	95,94%

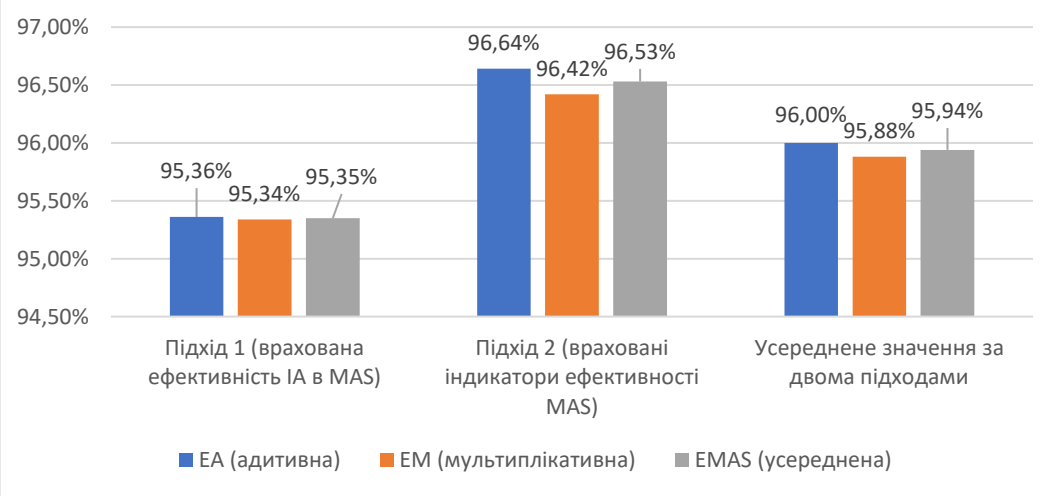

Рисунок 4.7 - Узагальнення результатів визначення ефективності гібридної МАС згідно запропонованих підходів (робота із поліморфними вірусами 1-5 рівнів складності)

Експеримент 2 передбачає, що було згенеровано 1000 зразків даних (файлів), серед яких 300 – це поліморфні віруси, змінені з часом; 700 – чисті (не заражені) файли. Було згенеровано поліморфні віруси 1 рівня складності.

Результати підходу 1 для визначення ефективності гібридної МАС із врахуванням ефективності ІА гібридної МАС відображено у табл. 4.13.

Таблиця 4.13

Ефективність інтелектуальних агентів гібридної MAS(експеримент 2)

Агент	Опис виконуваних функцій	Кількісні результати	Ефективність (%)	Формула / Розрахунок
ІА «Аналіз»	Перевірка 1000 файлів, попереднє виявлення підозрілих (аналіз файлової системи, пам'яті, мережі, процесів)	Із 300 вірусів правильно виявлено 298 (*TP*), помилково підозрює 10 чистих (*FP*)	99,3%	TPR = TP / (TP + FN) = 298 / 300
ІА «Темп поширення»	Визначення рівня загрози вірусів на основі моделі Лотки-Вольтерра	Агресивні темпи поширення точно визначено для 19 з 20 випадків	95,0%	*RE* = 19 / 20
ІА «Виявлення»	Поглиблене виявлення методами: ML, аналіз у пісочниці, PLN тощо	Із 298 підозрілих 295 підтверджено як поліморфні віруси, 3 чистих помилково (*FP*)	99,0%	*TPR = TP / (TP + FN)* = 295 / 298
ІА «Класифікація»	Нечітка класифікація за 1 рівнем складності (враховано показники *CE, CO, SDC*...)	292 з 295 коректно класифіковано	99,0%	*Classification accuracy* = 292 / 295
ІА «Прийняття рішення»	Реакція на виявлені загрози (блокування, карантин тощо)	Із 295 виявлених – 293 знешкоджено, 2 лише попереджено	99,3%	*Decision making* = 293 / 295

Згідно табл. 4.13 обчислено ефективність гібридної МАС так:

$$E_{A1} = 0{,}993 * 0{,}085 + 0{,}95 * 0{,}105 + 0{,}99 * 0{,}37 + 0{,}99 * 0{,}315 + 0{,}993 * 0{,}125 = 0{,}9864.$$

$$E_{M1} = 0{,}993^{0{,}085} \cdot 0{,}95^{0{,}105} \cdot 0{,}99^{0{,}37} \cdot 0{,}99^{0{,}315} \cdot 0{,}993^{0{,}125} = 0{,}9863.$$

$$E_{MAC1} = \frac{0{,}9864 + 0{,}9863}{2} = 0{,}98635.$$

Результати обчислень подано у табл. 4.14.

Таблиця 4.14

Ефективність гібридної МАС (підхід 1) із врахуванням ефективності ІА в МАС (експеримент 2)

ІА	Ефективність агента	Ваговий коефіцієнт агента	Ефективність гібридної МАС (адитивна система)	Ефективність гібридної МАС (мультиплікативна система)	Усереднена ефективність гібридної МАС (E_{MAC})
ІА «Аналіз»	0,993	0,085	0,9864	0,9863	0,98635
ІА «Темп поширення»	0,950	0,105			
ІА «Виявлення»	0,990	0,370			
ІА «Класифікація»	0,990	0,315			
ІА «Прийняття рішення»	0,993	0,125			

Згідно даних табл. 4.14 можна встановити, що використання запропонованого підходу 1 (коли ефективність гібридної МАС визначається через визначення ефективності складових ІА), при роботі із поліморфними вірусами першого рівня складності, ефективність гібридної МАС при розгляді її як адитивної системи, складає 98,64 %, ефективність гібридної МАС при розгляді її як мультиплікативної системи – 85,63 %, а усереднене значення становить 98,635%, що є досить високим результатом.

У табл. 4.15 відображена деталізація кількості поліморфних вірусів за рівнями складності.

Таблиця 4.15

Таблиця з деталізацією кількості поліморфних вірусів за рівнями складності (експеримент 2)

Рівень складності вірусу	Кількість зразків	Правильно виявлено (ІА «Виявлення»)	Правильно класифіковано (ІА «Класифікація»)	Ефективність виявлення (%)	Ефективність класифікації (%)
Рівень 1	300	298	295	99,00%	99,00%
Усього / Середнє	300	298	295	99,00%	99,00%

Визначимо значення TPR_P, *Precision, F1-score* згідно отриманих даних в даному експерименті.

$$TPR(Recall) = \frac{298}{298+2} = 0{,}993.$$

$$Precision = \frac{298}{298+3} = 0{,}990.$$

$$F1 - score = 2 \cdot \frac{0{,}990 \cdot 0{,}993}{0{,}900 + 0{,}993} = 0{,}9915.$$

Оскільки в даному експерименті було згенеровано поліморфні віруси першого рівня складності, тому вагові коефіцієнти ефективності їх класифікації будуть на рівні 0,333 (видозміна рис. 4.4 та зміна розрахунку ваг Фішберна, що представлено у табл. 4.16).

Таблиця 4.16

Система ваг Фішберна (при наявності поліморфних вірусів 1-го рівня складності)

N	*E*	*p1*	*p2*	*p3*	*p4*	*p5*	*p6*
2	$E_d\}E_c$	2/3	1/3	-	-	-	-
3	$TPR_P \approx Precision\} F1\text{-}score$	2/5	2/5	1/5	-	-	-
6	$TPR_{PV(L1)}$	1	-	-	-	-	-

Згідно запропонованої методики (підхід 2) обчислимо ефективність гібридної МАС у адитивній формі із врахуванням індикаторів ефективності так:

$$E_{A2} = 0{,}267 * 0{,}993 + 0{,}267 * 0{,}990 + 0{,}133 * 0{,}9915 + 0{,}333 * 0{,}990 = 0{,}9910 = 99{,}10\%.$$

Обчислимо ефективність МАС у мультиплікативній формі (підхід 2) так:

$$E_{\text{M2}} = 0{,}993^{0{,}267} \cdot 0{,}990^{0{,}267} \cdot 0{,}9915^{0{,}133} \cdot 0{,}990^{0{,}333} = 0{,}991 = 99{,}10\%.$$

Обчислимо ефективність гібридної МАС за другим підходом, усереднивши ці значення так:

$$E_{MAC2} = \frac{0{,}991 + 0{,}991}{2} = 0{,}991 = 99{,}10\%.$$

Зведемо результати екперименту 2 для підходу 1 та 2 у таблицю 4.17 та на рис. 4.8.

Таблиця 4.17

Узагальнення результатів визначення ефективності гібридної МАС згідно запропонованих підходів (експеримент 2)

Ефективність гібридної МАС	Підхід 1 (врахована ефективність ІА в гібридній МАС)	Підхід 2 (враховані індикатори ефективності гібридної МАС)	Усереднене значення за двома підходами
E_a (адитивна)	98,64%	99,10%	98,87%
E_M (мультиплікативна)	98,63%	99,10%	98,87%
E_{MAC} (усереднена)	98,635%	99,10%	98,87%

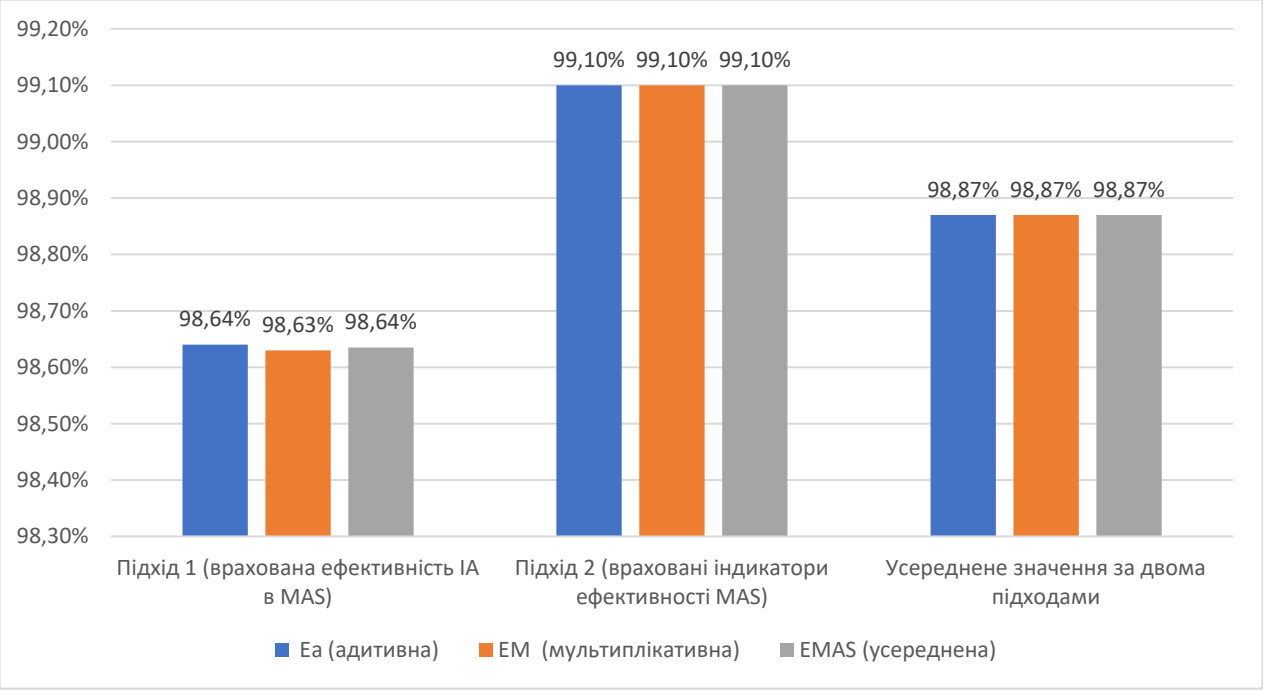

Рисунок 4.8 - Узагальнення результатів визначення ефективності гібридної МАС згідно запропонованих підходів (робота із поліморфними вірусами 1-го рівня складності)

Якщо порівняти результати експерименту 1 та 2, то можна помітити, що МАС має ефективність вищу (на рівні 98,87 %) при роботі лише з поліморфними вірусами першого рівня складності, а при роботі з поліморфними вірусами перших п’яти рівнів складності – 95,94 %.

Аналогічно експериментам 1 та 2 проведено експерименти стосовно роботи гібридної МАС з поліморфними вірусами 1 і 2 рівнів складності, 1-3 рівнів складності, 1-4 рівнів складності. Результати проведених експериментів відображено у табл. 4.18 та на рис. 4.9.

Таблиця 4.18

Порівняння ефективності гібридної МАС залежно від поліморфних вірусів різних рівнів складності

Рівні складності поліморфних вірусів	E_{MAC} (усереднена)
1 рівень	98,87%
1-2 рівні	98,15%
1-3 рівні	97,45%
1-4 рівні	96,34%
1-5 рівні	95,94%

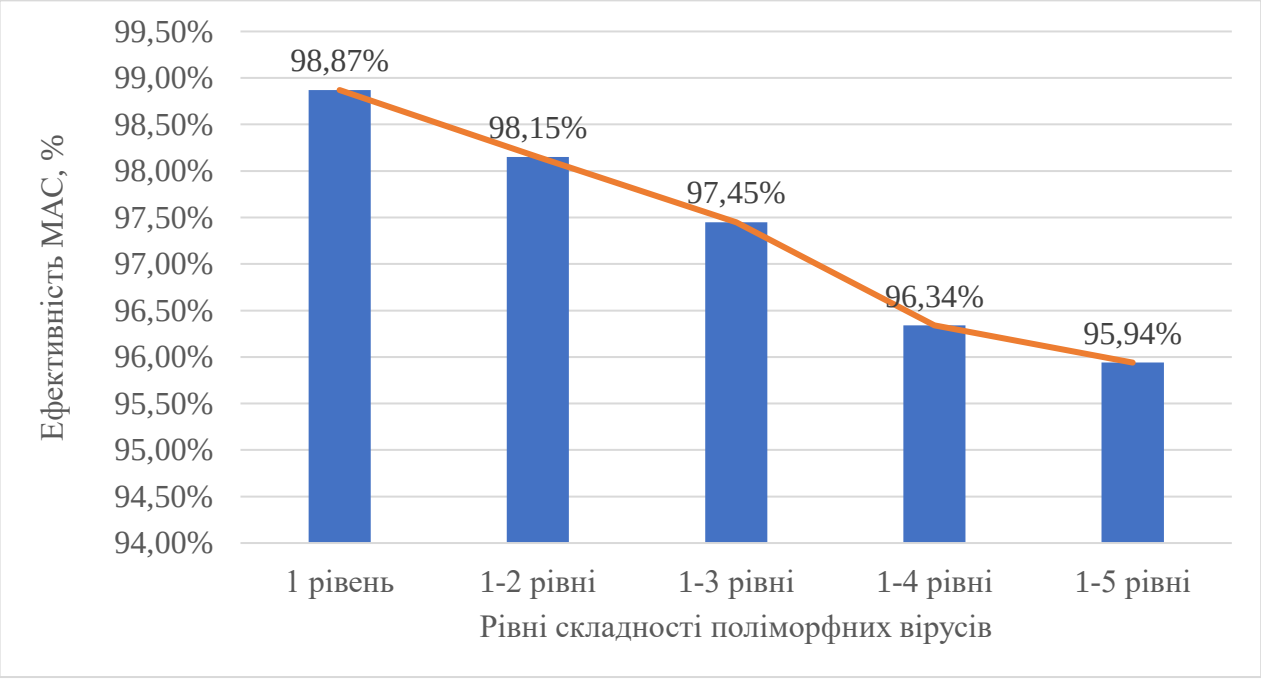

Рисунок 4.9 - Порівняння ефективності гібридної МАС залежно від поліморфних вірусів різних рівнів складності

Аналіз рис. 4.5 свідчить про те, що робота гібридної МАС характеризується досить високим рівнем ефективності, яка знижується з появою нового рівня складності поліморфних вірусів.

Також проаналізуємо ефективність даної гібридної МАС за часом виявлення та використанням ресурсу (E_{TR}). Основними показниками оберемо T_d (час виявлення, сек), T_r (час реакції, сек), T_A (час комунікації між агентами, сек), U_R (використання ресурсів (CPU, RAM), частка (0-1). Експерти встановили, що ці показники є рівновагомими (ваговий коефіцієнт кожного – 0,25).

Для визначення ефективності у адитивній та мультиплікативніфй формі скористаємося формулами (4.1) та (4.2). Нормалізацію показників здійснено за формулою:

$$Z_X = 1 - \frac{X}{X_{max}}, \tag{4.21}$$

де Z_X – нормалізованs значення показника; X – наявне значення показника; X_{max} – максимальне значення показника (згідно експертних оцінок).

У таблиці 4.19 подані значення показників для розрахунку ефективності гібридної МАС за часом виявлення та використанням ресурсу.

Таблиця 2.19

Значення показників для розрахунку ефективності гібридної МАС за часом виявлення та використанням ресурсу

Показник	Значення показника	Ваговий коефіцієнт	Максимальне значення показника (згідно експертних оцінок)	Нормалізоване значення
T_d	0,3 с	0,25	10 сек	$1 - \frac{0,3}{10} = 0,97$
T_r	0,2 с	0,25	10 сек	$1 - \frac{0,2}{10} = 0,98$
T_A	0,1 с	0,25	2 сек	$1 - \frac{0,1}{2} = 0,95$
U_R	5 % (0,05)	0,25	1 (100 %)	$1 - 0,05 = 0,95$

Тому, узагальнена формула для визначення ефективності гібридної МАС за часом виявлення та використанням ресурсу у адитивній формі (формула (4.16)) виглядатиме наступним чином:

$$E_{TRA} = T_d * 0{,}25 + T_r * 0{,}25 + T_A * 0{,}25 + U_R * 0{,}25 = 0{,}9625 = 96{,}25\ \%. \quad (4.16)$$

Узагальнена формула для визначення ефективності гібридної МАС за часом виявлення та використанням ресурсу у мультиплікативній формі (формула (4.17)) виглядатиме наступним чином:

$$E_{TRM} = {T_d}^{0{,}25} \cdot {T_r}^{0{,}25} \cdot {T_A}^{0{,}25} \cdot {U_R}^{0{,}25} = 0{,}9624 = 96{,}24\ \%. \quad (4.17)$$

Відзначимо майже ідентичні значення ефективності гібридної МАС за часом виявлення та використанням ресурсу у мультиплікативній та адитивній формі. Усереднене значення становить 96,245 %.

Такаим чином, запропонована архітектура гібридної МАС виявлення поліморфних вірусів в комп’ютерних мережах. запропонована МАС складається з наступних ІА: ІА «Аналіз», ІА «Темп поширення», ІА «Виявлення», ІА «Класифікація», ІА «Прийняття рішення», які виконують встановлені ролі та взаємодіють між собою.

Запропоновано, що ІА «Аналіз» перевіряє систему на наявність підозрілих або шкідливих дій, аналізуючи файлову систему, оперативну пам'ять, мережевий трафік та поведінку процесів; включає збір інформації, що стосується програмного забезпечення, та проведення глибокого аналізу для визначення характеру загрози, її джерела та можливого впливу. В результаті аналізу системи ІА «Аналіз» формує та передає ІА «Темп поширення» значення коефіцієнтів *α, β, γ, δ.*

Запропоновано, що ІА «Темп поширення» здійснює моделювання процесу поширення поліморфних вірусів на основі моделі Лотки-Вольтерра. Даний агент досліджує темп поширення поліморфних вірусів та визначає, яку кількість та які методи виявлення поліморфних вірусів необхідно використати ІА «Виявлення».

Запропоновано, що ІА «Виявлення» використовує комплекс методів для виявлення поліморфних вірусів (кількість методів визначена ІА «Темп поширення»): алгоритми пошуку рядка, інтелектуальний аналіз даних, аналіз в пісочниці, машинне

навчання, метод розробки структурних функцій, PLN, які визначені ІА «Темп поширення». Виявлені поліморфні віруси потрапляють у «чорний список», який надходить до ІА «Класифікація».

Запропоновано, що ІА «Класифікація» здійснює нечітку класифікацію виявлених поліморфних вірусів (ІА «Виявлення»), які потрапили у «чорний список» згідно 6 рівнів їх складності.

Відображено алгоритм роботи гібридної МАС.

Запропонована методика визначення ефективності гібридної МАС, яка складається з двох підходів: 1) визначення ефективності гібридної МАС із врахуванням ефективності складових ІА; 2) визначення ефективності гібридної МАС із врахування індикаторів ефективності (виявлення та класифікація).

Запропоновано в рамках кожного з підходів визначати ефективність гібридної МАС в адитивній та мультиплікативній формі. Як узагальнене значення обирається їх усереднене значення.

Проведені експерименти відображають високий рівень ефективності гібридної МАС, який залежить від роботи МАС з поліморфними вірусами різних рівнів складності. При роботі з поліморфними вірусами 1-го рівня складності ефективність гібридної МАС становить 98,87 %, при роботі з поліморфними вірусами 1-го і 2-го рівня складності – 98,15 %, 1-го – 3-го рівнів складності – 97,45 %, 1-го – 4-го – 96,34 %, 1-го – 5-го – 95,94 %.

4.3. Висновки до четвертого розділу

Реалізовано гібридну МАС, яка складається з наступних ІА: ІА «Аналіз», ІА «Темп поширення», ІА «Виявлення», ІА «Класифікація», ІА «Прийняття рішення», які виконують встановлені ролі та взаємодіють між собою.

Запропоновано методику визначення ефективності гібридної МАС. яка складається з двох підходів: визначення ефективності МАС із врахуванням ефективності складових ІА; визначення ефективності МАС із врахування індикаторів ефективності (виявлення та класифікація). Запропоновано в рамках кожного з

підходів визначати ефективність МАС в адитивній та мультиплікативній формі. Як узагальнене значення обирається їх усереднене значення.

Проведені експерименти відображають досить високий рівень ефективності гібридної МАС, який залежить від роботи МАС з поліморфними вірусами різних рівнів складності. При роботі з поліморфними вірусами 1-го рівня складності ефективність МАС становить 98,87 %, при роботі з поліморфними вірусами 1-го і 2-го рівня складності – 98,15 %, 1-го – 3-го рівнів складності – 97,45 %, 1-го – 4-го – 96,34 %, 1-го – 5-го – 95,94 %. Ефективність гібридної МАС за часом виявлення та використанням ресурсу становить 96,245 %. Отримані значення ефективності відображають досягнення мети роботи.

Основні результати розділу опубліковані у працях [152, 154].

ВИСНОВКИ

У результаті виконання дисертаційного дослідження було розв’язано актуальну науково-прикладну задачу підвищення ефективності виявлення поліморфних вірусів в комп’ютерних мережах за допомогою розробленої гібридної мультиагентної системи. У роботі отримано наступні наукові та практичні результати.

1. Проведено аналіз поліморфних вірусів за різними рівнями складності їх структури, методів та засобів їх виявлення за допомогою мультиагентних систем, що дозволило встановити необхідність розробки гібридної мультиагентної системи з метою підвищення ефективності виявлення поліморфних вірусів різних рівнів складності.

2. Розроблено моделі класів поліморфних вірусів згідно їх складності та, відповідно, різних рівнів поліморфізму, особливістю яких є те, що віруси першого рівня поліморфізму використовують постійні значення для різних розшифровувачів, віруси другого рівня в розшифровувачах мають постійну одну або декілька інструкцій, віруси третього рівня в розшифровувачах мають команди, які не приймають участі в розшифруванні вірусного коду, чи «команд-сміття», віруси четвертого рівня використовують в розшифровувачі взаємозамінювані інструкції без зміни алгоритму розшифрування, віруси п’ятого рівня використовують високорозвинені методи маскування та зміни свого коду, віруси шостого рівня використовують програмні частини, які «перемішуються» всередині тіла вірусу.

3. Розроблено архітектуру гібридних мультиагентних систем виявлення поліморфних вірусів на основі інтелектуального агента, яка каскадно, делегуючи повноваження, здійснює аналіз середовища, встановлює темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення, а також є основою для

розроблення нових систем із підвищеною ефективністю виявлення поліморфних вірусів.

4. Удосконалено метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, результат використання якого дозволив дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначати, яку кількість методів виявлення поліморфних вірусів необхідно використати.

5. Розроблено метод виявлення поліморфних вірусів, який дозволив не лише збільшити ефективність виявлення поліморфних вірусів, але й визначити ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів. Застосування методів на кожному наступному виконанні другого етапу здійснюється у різному порядку, що дозволило забезпечити адаптивінсть системи.

6. Удосконалено метод класифікації поліморфних вірусів, який дозволив не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обґрунтованості та ефективність обраних стратегій.

7. Розроблено гібридну мультиагентну систему виявлення поліморфних вірусів. В структуру даної системи входять інтелектуальні агенти: «Аналіз», «Темп поширення», «Виявлення», «Класифікація», «Прийняття рішення», які виконують встановлені ролі та взаємодіють між собою. Гібридну мультиагентну систему впроваджено у виробництво, а проведені експерименти відображають досить високий рівень її ефективності, який залежить від роботи мультиагентної системи з поліморфними вірусами різних рівнів складності. При роботі з поліморфними

вірусами першого рівня складності ефективність гібридної мультиагентної системи становить 98,87 %, при роботі з поліморфними вірусами першого і другого рівня складності – 98,15 %, перших трьох рівнів складності – 97,45 %, перших чотирьох – 96,34 %, перших п’яти – 95,94 %. Ефективність гібридної МАС за часом виявлення та використанням ресурсу становить 96,245 %. Отримані значення ефективності відображають досягнення мети роботи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abdullah M. A., Yu Y., Adu K., Imrana Y., Wang X., Cai J. HCL-classifier: CNN and LSTM based hybrid malware classifier for internet of things (IoT). *Future Generation Computer Systems.* 2023. Vol. 142. P. 41–58. https://doi.org/10.1016/j.future.2022.12.034

2. Abidar R., Moummadi K., Moutaouakkil F., Medromi H. Intelligent and Pervasive Supervising Platform for Information System Security Based on Multi-Agent Systems. *International Review on Computers and Software (IRECOS).* 2015. Vol. 10 (1). URL: https://doi.org/10.15866/irecos.v10i1.4699

3. Aboaoja F. A., Zainal A., Ghaleb F. A., Al-rimy B. A. S., Eisa T. A. E., Elnour A. A. H. Malware Detection Issues, Challenges, and Future Directions: A Survey. *Applied Sciences*. 2022. Vol. 12(17). https://doi.org/10.3390/app12178482

4. Abu Bakar R., Huang X., Javed M.S., Hussain S., Majeed M.F. An Intelligent Agent-Based Detection System for DDoS Attacks Using Automatic Feature Extraction and Selection. *Sensors.* 2023. Vol. 23. 3333. https://doi.org/10.3390/s23063333

5. Abusnaina A., M. Abuhamad, H. Alasmary, A. Anwar, R. Jang, S. Salem, D. Nyang, D. Mohaisen. Dl-fhmc: Deep learning-based fine-grained hierarchical learning approach for robust malware classification. *IEEE Transactions on Dependable and Secure Computing*. 2021. Vol. 19(5). P. 3432–3447. https://doi.org/10.48550/arXiv.2005.07145

6. Ahmed M., Kazar O., Harous S.Cyber-physical system model based on multi-agent system. *IET Cyber-Physical Systems: Theory & Applications*. 2024. P. 1-11. https://doi.org/10.1049/cps2.12096

7. Akhtar M. S., Feng T. Malware Analysis and Detection Using Machine Learning Algorithms. *Symmetry (Basel)*. 2022. Vol. 14(11). P. 2304. https://doi.org/10.3390/sym14112304

8. Al-Andoli M. N., Tan S. C., Sim K. S., Lim C. P., Goh P. Y. Parallel deep learning with a hybrid BP-PSO framework for feature extraction and malware classification. *Applied Soft Computing*. 2022. Vol. 131. P. 109756. https://doi.org/10.1016/j.asoc.2022.109756

9. Alenezi M. N., Alabdulrazzaq H., Alshaher A. A., Alkharang M. M. Evolution of Malware Threats and Techniques: A Review. *International Journal of Communication Networks and Information Security (IJCNIS)*. 2020. Vol. 12(3). P. 326–337. https://doi.org/10.17762/ijcnis.v12i3.4723

10. Alsubai S., Dutta A.K., Rahaman Wahab Sait A., Jaish Y. A. A., Alamer B. H., Saad H. E. H., Ayub R. Enhanced slime mould optimization with convolutional BLSTM autoencoder based malware classification in intelligent systems. *Expert Systems*. 2024. e13557. https://doi.org/10.1111/exsy.13557

11. Anderson B., McGrew D. OS fingerprinting: New techniques and a study of information gain and obfuscation. 2017 IEEE Conference on Communications and Network Security, CNS, Las Vegas, 2017. P. 1–9. doi: 10.1109/CNS.2017.8228647

12. APKTool. URL: https://apktool.org/

13. Atitallah S. B., Driss M., Almomani I. A novel detection and multi-classification approach for IoT-malware using random forest voting of finetuning convolutional neural networks. *Sensors*. 2022. Vol. 22(11). P. 4302. https://doi.org/10.3390/s22114302

14. Avast. Available at: https://www.avast.ua/ (accessed 21.02.2025).

15. Balazs Z. Malware Analysis Sandbox Testing Methodology. The Journal on Cybercrime and Digital Investigations. 2016. Vol. 1(1). P. 49-53. https://doi.org/10.18464/cybin.v1i1.3

16. Beg R., Pateriya R.K., Tomar D. S. ACMFNN: A Novel design of an augmented convolutional model for intelligent cross-domain malware localization via forensic neural networks. *IEEE Access XX*. 2017. doi: 10.1109/ACCESS.2023.3305274

17. Bilge L., Han Y., Dell'Amico M. Riskteller: Predicting the risk of cyber incidents. *2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, Texas, USA, 2017. P. 1299-1311. doi: 10.1145/3133956.3134022

18. Bitdefender Antivirus Plus. Available at: https://bitdefender.ua/ (accessed 21.02.2025).

19. Bonnard B., Rouot J. Feedback classification and optimal control with applications to the controlled Lotka–Volterra model. *Optimization*. 2024. P. 1–24. doi:10.1080/02331934.2024.2392209

20. Brahmi I., Ben Yahia S., Aouadi H., Poncelet P. Towards a Multiagent-Based Distributed Intrusion Detection System Using Data Mining Approaches. *Lecture Notes in Computer Science.* 2012. Vol. 7103. URL: https://doi.org/10.1007/978-3-642-27609-5_12

21. Brezinski K., Ferens K. Metamorphic Malware and Obfuscation: A Survey of Techniques, Variants, and Generation Kits. *Security and Communication Networks*. 2023. P. 8227751. doi: 10.1155/2023/8227751.

22. Cesare S., Xiang Y. Classification of malware using structured control flow. *8-th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010),* Brisbane, Australia, 2010. Vol. 107. P. 61-70. doi: 10.5555/1862294.1862301

23. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Comprehensive approach to the detection and analysis of polymorphic malware. *CEUR Workshop Proceedings*. 2024. Vol. 3736. P.312-323. URL: https://ceur-ws.org/Vol-3736/paper23.pdf

24. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Modeling the detection process of polymorphic malware based on the Lotka-Volterra Model. CEUR Workshop Proceedings. 2025. Vol.3899. P. 244-253. URL: https://ceur-ws.org/Vol-3899/paper22.pdf

25. Chaganti R., Ravi V., Pham T.D. A multi-view feature fusion approach for effective malware classification using deep learning. *Journal of Information Security and Applications*. 2023. Vol. 72. 103402. https://doi.org/10.1016/j.jisa.2022.1034

26. Chakraborty A., Kriti K., Yateendra, Bennet Praba M.S. Polymorphic Malware Detection by Image Conversion Technique. *International Journal of Engineering and Advanced Technology (IJEAT).* 2020. Vol. 9(3). P. 2898-2903. https://doi.org/10.35940/ijeat.B4999.029320

27. Chen Y., Ni J., Ong Y.C. Lotka–Volterra models for extraterrestrial self-replicating probes. *The European Physical Journal Plus*. 2022. Vol. 137. P. 1109. doi:10.1140/epjp/s13360-022-03320-3

28. Chiwariro R., Pullagura L. Malware Detection and Classification Using Machine Learning Algorithms. *International Journal for Research in Applied Science & Engineering Technology, IJRASET*. 2023. Vol. 11. P. 1727-1738. doi: 10.22214/ijraset.2023.55255

29. Choi S., Bae J., Lee C., Kim Y., Kim J. Attention-based automated feature extraction for malware analysis. *Sensors (Switzerland)*. 2020. Vol. 20(10). P. 1–17. https://doi.org/10.3390/s20102893

30. Chumachenko D., Sokolov O., Yakovlev S. Fuzzy recurrent mappings in multiagent simulation of population dynamics systems. *International Journal of Computing*. 2020. Vol. 19(2). P. 290-297. https://doi.org/10.47839/ijc.19.2.1773

31. Clenet M., Massol F., Najim J. Equilibrium and surviving species in a large Lotka–Volterra system of differential equations. *Journal of Mathematical Biology*. 2023. Vol. 87. P. 13. doi:10.1007/s00285-023-01939-z

32. Cuckoo Sandbox. URL: https://cuckoosandbox.org/

33. Damodaran A., Troia F.D., Visaggio C.A., Austin T. H., Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*. 2017. Vol. 13. P. 1–12. doi: 10.1007/s11416-015-0261-z

34. Davis J., Olivença D., Brown S., Voit E. Methods of quantifying interactions among populations using Lotka-Volterra models. *Frontiers in Systems Biology*. 2022. Vol. 2. P. 1021897. doi: 10.3389/fsysb.2022.1021897

35. Dex2Jar. URL: https://dex2jar.en.uptodown.com/android

36. Djenna A., Bouridane A., Rubab S., Marou I.M. Artificial intelligence-based malware detection, analysis, and mitigation. *Symmetry*. 2023. Vol. 15(3). P. 677. https://doi.org/10.3390/sym15030677

37. Dovbysh A., Liubchak V., Shelehov I., Simonovskiy J., Tenytska A. Information-extreme machine learning of a cyber attack detection system. *Radioelectronic and Computer Systems*. 2022. Vol. 3. P. 121-131. https://doi.org/10.32620/reks.2022.3.09

38. Dunets O., Wolff C., Sachenko A., Hladiy G., Dobrotvor I. Multi-agent system of IT project planning. *2017 9th IEEE International Conference on Intelligent Data*

Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, 2017. P. 548-552. https://doi.org/10.1109/IDAACS.2017.8095141

39. Emam O., Fahmy H., Mamdouh M. Securing IoT Systems using Blockchain Algorithms. *Communications on Applied Electronics*. 2020. Vol. 7(34). P. 10-17. doi:10.5120/cae2020652871

40. ESET NOD32 Antivirus. Available at: https://www.eset.com/ua/ (accessed 21.02.2025).

41. Fathurrahman A., Bejo A., Ardiyanto I. Lightweight convolution neural network for image-based malware classification on embedded systems. *2021 International seminar on machine learning, optimization, and data science (ISMODE). IEEE*. 2022. P. 12–16.

42. Fesokha V.V., Subach I.Y., Kubrak V.O., Mykytiuk A.V., Korotaiev S.O. Zero-day polymorphic cyberattacks detection using fuzzy inference system. *Austrian Journal of Technical and Natural Sciences*. 2020. Vol. 5-6. P. 8-13.

43. F-Secure. Available at: https://www.f-secure.com/en (accessed 21.02.2025).

44. Ganin A. A., Quach P., Panwar M., Collier Z. A., Keisler J. M., Marchese D., Linkov I. Multicriteria Decision Framework for Cybersecurity Risk Assessment and Management. *Risk Analysis*. 2020. Vol. 40(1). P. 183-199. https://doi.org/10.1111/risa.12891

45. Google Rapid Response. URL: https://github.com/google/grr/issues

46. Goyal M., Kumar R. AVMCT: API Calls Visualization based Malware Classification using Transfer Learning. *Journal of Algebraic Statistics*. 2022. Vol. 13(1). P. 31–41. https://doi.org/10.52783/jas.v13i1.59

47. Gundogan K., Gupta K., Garland L., Varol C., Shashidhar N. Identifying Malware Family with String Matching Algorithms Based on API Calls and Entire Strings. 12th International Symposium on Digital Forensics and Security, ISDFS, San Antonio, TX, USA, 2024. doi: 10.1109/ISDFS60797.2024.10527225.

48. Harrigan C., Goertzel B., Ikle M., Belayneh A., Yu G. Guiding Probabilistic Logical Inference with Nonlinear Dynamical Attention Allocation. *Lecture Notes in Computer Science*. 2014. Vol. 8598. P. 238-241.

49. Harshvardhan U., Rastgoftar H. Multi-Layer Continuum Deformation Optimization of Multi-Agent Systems. *IFAC-PapersOnLine. 2023. Vol*. 56 (2). P. 10222-10227. https://doi.org/10.1016/j.ifacol.2023.10.901

50. Hama Saeed M. A. Malware in Computer Systems: Problems and Solutions. *(IJID) International Journal on Informatics for Development*. 2020. Vol. 9(1). P. 1. https://doi.org/10.14421/ijid.2020.09101

51. Islam R., Sayed M.I., Saha S., Hossain M. J., Masud M. A. Android Malware Classification Using Optimum Feature Selection and Ensemble Machine Learning. *Internet of Things and Cyber-Physical Systems*. 2023. Vol. 3. P. 100–111. https://doi.org/10.1016/j.iotcps.2023.03.001

52. Jerbi M., Dagdia Z. C., Bechikh S., Said L. B. On the Use of Artificial Malicious Patterns for Android Malware Detection. *Computer & Security*. 2020. Vol. 92. 101743. https://doi.org/10.1016/j.cose.2020.101743

53. Karpinski M., Kuznetsov O., Oliynykov R. Security, Privacy, Confidentiality, and Trust in the Blockchain: From Theory to Applications. *Electronics*. 2025. Vol. 14. P. 581. https://doi.org/10.3390/electronics14030581

54. Kashtalian A., Lysenko S., Savenko B., Sochor T., Kysil T. Principle and method of deception systems synthesizing for malware and computer attacks detection. *Radioelectronic and Computer Systems*. 2023. Vol. 4. P. 112-151. https://doi.org/10.32620/reks.2023.4.10

55. Kashtalian A., Lysenko S., Savenko O., Nicheporuk A., Sochor T., Avsiyevych V. Multi-computer malware detection systems with metamorphic functionality. *Radioelectronic and Computer Systems*. 2024. Vol. 1. P. 152-175. https://doi.org/10.32620/reks.2024.1.13

56. Kharchenko V., Ponochovnyi Y., Abdulmunem A.-S. M. Q., Boyarchuk A. Security and Availability Models for Smart Building Automation Systems. *International Journal of Computing*. 2017. Vol. 16(4). P. 194-202. DOI:10.47839/ijc.16.4.907.

57. Kharchenko V., Ponochovnyi Y., Ivanchenko O., Fesenko H., Illiashenko O. Combining Markov and semi-Markov modelling for assessing availability and

cybersecurity of cloud and IoT systems. *Cryptography*. 2022. Vol. 6(44). DOI:10.3390/cryptography6030044.

58. Khoroshko V., Khokhlachova Y., Vyshnevska N. Choice of indicators for forecasting cyber protection of computer systems. Ukrainian Scientific Journal of Information Security. 2023. Vol. 29(1). P. 41–47.

59. Khoroshko V., Khokhlachova Y., Vyshnevska N. Decomposition of computer network technology in their design. *Ukrainian Scientific Journal of Information Security*. 2023. Vol. 29. Issue 3. P. 130-137. DOI: 10.18372/2225-5036.29.18072

60. Komar M., Golovko V., Sachenko A., Bezobrazov S. Intelligent system for detection of networking intrusion. 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Prague, Czech Republic, 2011. P. 374-377. https://doi.org/10.1109/IDAACS.2011.6072777

61. Komar M., Sachenko A., Golovko V., Dorosh V. Compression of Network Traffic Parameters for Detecting Cyber Attacks Based on Deep Learning. 9th IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT'2018), Kyiv, Ukraine, 2018. P. 44-48.

62. Kurian A. J., Santhosh A., Subin M. Enhanced malware detection framework leveraging machine learning algorithms. International Research Journal of Modernization in Engineering Technology and Science. 2024. Vol. 06(03). Vol. 3597-3603.

63. Qiao Y., Zhang W., Du X., Guizani M. Malware classification based on multilayer perception and Word2Vec for IoT security. *ACM Transactions on Internet Technology (TOIT)*. 2021. Vol. 22(1). P. 1–22. https://doi.org/10.1145/3436751

64. Lakhno, V., Akhmetov, B., Smirnov, O., Chubaievskyi, V., Khorolska, K., Bebeshko, B. Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm. In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. *Lecture Notes on Data Engineering and Communications Technologies.* 2023. Vol 131. Springer, Singapore. https://doi.org/10.1007/978-981-19-1844-5_2

65. Lakhno V. *et al.* Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information. In: Pal, S.K.,

Thampi, S.M., Abraham, A. (eds) Intelligent Informatics. ISI 2023. *Smart Innovation, Systems and Technologies*. 2025. Vol. 389. Springer, Singapore. https://doi.org/10.1007/978-981-97-2147-4_26

66. Letychevskyi O., Peschanenko V. Applying algebraic virtual machine to cybersecurity tasks. IEEE 9th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), IEEE, Hammamet, Tunisia, 2022. P. 161–169. DOI:10.1109/SETIT54465.2022.9875895.

67. Ligo A.K., Kott A., Linkov I. How to measure cyber-resilience of a system with autonomous agents: approaches and challenges. *IEEE Engineering Management Review*. 2021. Vol. 49(2). P. 89-97. https://doi.org/10.1109/EMR.2021.3074288

68. Lin Y., Din Q., Rafaqat M., Elsadany A. A., Zeng Y. Dynamics and Chaos Control for a Discrete-Time Lotka-Volterra Model. *IEEE Access.* (2020). Vol. 8. P. 126760-126775. doi: 10.1109/ACCESS.2020.3008522.

69. Ling Y. T., · Sani N. F. M., · Abdullah M. T., · Hamid N. A. W. A. Metamorphic malware detection using structural features andnonnegative matrix factorization with hidden markov model, Journal of Computer Virology and Hacking Techniques. 2022. Vol. 18. P. 183–203. doi: 10.1007/s11416-021-00404-z

70. Ling Y. T., Sani N. F. M., Abdullah M. T., Hamid N. A. W. A. Structural Features with Nonnegative Matrix Factorization for Metamorphic Malware Detection. *Computers & Security.* 2021. Vol. 104(2). P. 102216. doi: 10.1016/j.cose.2021.102216

71. Liu S., Feng P., Wang S., Sun K., Cao J. Enhancing malware analysis sandboxes with emulated user behavior. *Computers and Security*. 2022. Vol. 115. P. 102613. https://doi.org/10.1016/j.cose.2022.102613

72. Lobanchykova N.M., Pilkevych I.A., Korchenko O. Analysis of attacks on components of IoT systems and cybersecurity technologies. *QuaInT+ doors*. 2021. P. 83–96. https://ceur-ws.org/Vol-2850/paper6.pdf.

73. Lotka-Volterra equation solver. URL: https://fusion809.github.io/LotkaVolterra/

74. Lysenko S., Pomorova O., Savenko O., Kryshchuk A., Bobrovnikova K. DNS-based Anti-evasion Technique for Botnets Detection. *8-th IEEE International Conference*

on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, Poland, 2015. P. 453–458.

75. Lysenko S., Savenko B. Distributed Discrete Malware Detection Systems Based on Partial Centralization and Self-Organization. *International Journal of Computing*. 2023. Vol. 22. P. 117-139. DOI: https://doi.org/10.47839/ijc.22.2.3082

76. Lysenko S., Savenko O., Bobrovnikova K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. CEUR-WS. 2018. Vol.2104. P. 688-695.

77. Majidian Z., TaghipourEivazi S., Arasteh B., Babai S. An Intrusion Detection Method to Detect Denial of Service Attacks Using Error-Correcting Output Codes and Adaptive Neuro-Fuzzy Inference. *Computers and Electrical Engineering*. 2023. Vol. 106. 108600. https://doi.org/10.1016/j.compeleceng.2023.108600

78. Malwarebytes. Available at: https://www.malwarebytes.com/ (accessed 21.02.2025).

79. Malzilla. URL: https://malzilla.org/

80. Manikandan A. Investigative Study of the Behavior of Lotka-Volterra Model of COVID-19. *International Journal of Science and Research (IJSR).* 2021. Vol. 10(11). P. 556-558. doi: 10.21275/SR211110105514

81. Markowsky G., Savenko O., Lysenko S., Nicheporuk A. The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS*. 2018. Vol. 2104. P. 680–687.

82. Marques R.S., Epiphaniou G., Al-Khateeb H. et al. A flow-based multi-agent data exfiltration detection architecture for ultra-low latency networks. *ACM Transactions on Internet Technology*. 2021. Vol. 21 (4). 103. URL: https://doi.org/10.1145/3419103

83. Masabo E., Kaawaase K.S., Sansa-Otim J., Hanyurwimfura D. Structural Feature Engineering approach for detecting polymorphic malware. *15-th IEEE Intl Conf on Dependable, Autonomic and Secure Computing, 15-th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech*, 2017. P. 716-721. doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.125

84. Mobile-Sandbox. URL: https://mobilesandbox.com/

85. Monga R., Karlapalem K. MASFMMS: Multi Agent Systems Framework for Malware Modeling and Simulation. *Lecture Notes in Computer Science*. 2009. Vol. 5269. URL: https://doi.org/10.1007/978-3-642-01991-3_8

86. Moskalenko V., Kharchenko V., Moskalenko A., Kuzikov B. Resilience and resilient systems of artificial intelligence: Taxonomy, models and methods. *Algorithms*. 2023. Vol. 16. P. 165. DOI: 10.3390/a16030165.

87. McAfee Antivirus. Available at: https://www.mcafee.com/en-gb/index.html (accessed 21.02.2025).

88. Mukhin V., Kornaga Y., Bondarenko V., Zavgorodnii V., Herasymenko O., Sholokhov O. Mathematical model for heterogeneous databases parameters estimation in distributed systems with dynamic structure. *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*, Kyiv, Ukraine, 2020. P. 158–161. DOI:10.1109/ATIT50783.2020.9349331 .

89. Naeem H., Alshammari B. M., Ullah F. Explainable artificial intelligence-based IoT device malware detection mechanism using image visualization and fine-tuned CNN-based transfer learning model. *Computational Intelligence and Neuroscience*. 2022. 7671967. https://doi.org/10.1155/2022/7671967

90. Naeem M. R., Amin R., Alshamrani S.S., Alshehri A. Digital forensics for malware classification: An approach for binary code to pixel vector transition. *Computational Intelligence and Neuroscience*. 2022. P. 1–12. https://doi.org/10.1155/2022/6294058

91. Nasir M.H., Khan S.A., Khan M.M., Fatima M. Swarm Intelligence Inspired Intrusion Detection Systems - A Systematic Literature Review. *Computer Networks*. 2022. Vol. 205(1). 108708. https://doi.org/10.1016/j.comnet.2021.108708

92. Nguyen V.T. A study of polymorphic virus detection. 2018. https://doi.org/10.13140/RG.2.2.19853.79842

93. Nwagwu C. B., Taylor O. E., Nwiabu N. D. A Model for Detection of Malwares on Edge Devices. *International Journal Of Engineering And Computer Science*. 2024. Vol. 13(07). P. 26274-26283. https://doi.org/10.18535/ijecs/v13i07.4846

94. Obeidat I., AlZubi M. Developing a Faster Pattern Matching Algorithms for Intrusion Detection System. *International Journal of Computing*. 2019. Vol. 18(3). P. 278-284. https://doi.org/10.47839/ijc.18.3.1520

95. Perepelitsyn A., Kulanov V. Analysis of Ways of Digital Rights Management for FPGA-as-a-Service for AI-Based Solutions. 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies DESSERT 2023. 2023. P. 1-5. https://doi.org/10.1109/DESSERT61349.2023.10416526

96. Perova I., Bodyanskiy Y. Fast medical diagnostics using autoassociative neuro-fuzzy memory. *International Journal of Computing*. 2017. Vol. 16(1). P. 34-40. https://doi.org/10.47839/ijc.16.1.869

97. Poley L., Baron J. W., Galla T. Generalized Lotka-Volterra model with hierarchical interactions. *Physical Review E*. 2023. Vol. 107. 024313. doi: 10.1103/PhysRevE.107.024313

98. Pomorova O., Savenko O., Lysenko S., Kryshchuk A. Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic. *Communications in Computer and Information Science*. 2013. Vol. 370. P.243-254. https://doi.org/10.1007/978-3-642-38865-1_16

99. Pomorova O., Savenko O., Lysenko S., Nicheporuk A. Metamorphic Viruses Detection Technique based on the the Modified Emulators. CEUR-WS. 2016. Vol. 1614. Vol. 375-383.

100. Potter K., Shad R. Dynamic Malware Analysis with Reinforcement Learning. *Journal of Cyber Security*. 2024, July 16. http://dx.doi.org/10.2139/ssrn.4897267

101. Qu M., Tang J. Probabilistic Logic Neural Networks for Reasoning. *33-rd Conference on Neural Information Processing Systems, NeurIPS 2019*, Vancouver, Canada, 2019. doi: 10.48550/arXiv.1906.08495

102. Ravi V., Chaganti R., Alazab M. Recurrent Deep Learning-Based Feature Fusion Ensemble Meta-Classifier Approach for Intelligent Network Intrusion Detection System. Computers & Electrical Engineering. 2022. Vol. 102(3). 108156. https://doi.org/10.1016/j.compeleceng.2022.108156

103. REMnux. URL: https://remnux.org/

104. Sadeghi K. M. M., Goertzel B. Uncertain Interval Algebra via fuzzy/probabilistic modeling. *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),* Beijing, China, 2014. doi: 10.1109/FUZZ-IEEE.2014.6891863

105. Safdari F., Gorbenko A. Theoretical and experimental study of performance anomaly in multi-rate IEEE802.11ac wireless networks. *Radioelectronic and Computer Systems.* 2022. Vol. 4. P. 85-97. https://doi.org/10.32620/reks.2022.4.06

106. Salem A., S. S. Banescu, Pretschner A. Maat: Automatically Analyzing VirusTotal for Accurate Labeling and Effective Malware Detection. ACM Transactions on Privacy and Security. 2021, Vol. 24. P. 1–35. https://doi.org/10.48550/arXiv.2007.00510

107. Santos I., Brezo F., Ugarte-Pedrero X., Bringas P.G. Opcode sequences as representation of executables for data mining-based unknown malware detection. *Information Sciences*. 2013. Vol. 231. P. 64-82.

108. Sharif D. M., Beitollahi H. Detection of application-layer DDoS attacks using machine learning and genetic algorithms. *Computers & Security*. 2023. Vol.135. 103511. doi: 10.1016/j.cose.2023.103511

109. Savenko B., Kashtalian A. A method for determining the effectiveness of a distributed system for detecting abnormal manifestations. *Computer Systems and Information Technologies*. 2022. Vol. 2. P. 14–22. doi: 10.31891/csit-2022-2-2

110. Savenko B., Lysenko S., Bobrovnikova K., Savenko O., Markowsky G. Detection DNS Tunneling Botnets. *2021 IEEE 11th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IDAACS'2021*, Cracow, Poland, 2021.

111. Savenko O., Lysenko S., Nicheporuk A., Savenko B. Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search. *CEUR-WS*. 2017. Vol. 1844. P. 555–569.

112. Savenko O., Lysenko S., Nicheporuk A., Savenko B. Approach for the Unknown Metamorphic Virus Detection. 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, Bucharest, Romania, 2017. P. 71–76. doi: 10.1109/IDAACS.2017.8095052

113. Savenko O., Sachenko A., Lysenko S., Markowsky G., Vasylkiv N. Botnet Detection Approach based on the Distributed Systems. *International Journal of Computing*. 2020. Vol. 19(2). P. 190-198. https://doi.org/10.47839/ijc.19.2.1761

114. Sayadi H., He Z., Makrani H. M., Homayoun H. Intelligent Malware Detection based on HardwarePerformance Counters: A Comprehensive Surve. *25-th International Symposium on Quality Electronic Design, ISQED'24*, San Francisco, California, 2024. doi: 10.1109/ISQED60706.2024.10528369

115. Singh M., Carlson A. Exploring Polymorphic Algorithms and Their Use in Cryptography. *2024 IEEE 14th Annual Computing and Communication Workshop and Conference, CCWC*, Las Vegas, NV, USA, 2024. doi: 10.1109/CCWC60891.2024.10427812.

116. Singh V. K., Ozen A., Govindarasu M. A Hierarchical Multi-Agent Based Anomaly Detection for Wide-Area Protection in Smart Grid. *2018 Resilience Week (RWS)*, Denver, CO, USA. 2018. P. 63-69. URL: https://doi.org/10.1109/RWEEK.2018.8473514

117. Sharif M. H. U., Jiwani N., Gupta K., Mohammed M. A., Ansari D. M. F. A deep learning based technique for the classification of malware images. *Journal of Theoretical and Applied Information Technology*. 2023. Vol. 101(1). P.135–160. URL: http://www.jatit.org/volumes/Vol101No1/12Vol101No1.pdf

118. Sophos. Available at: https://home.sophos.com/en-us (accessed 21.02.2025).

119. Statista. URL: https://www.statista.com/

120. Sun B., Fujino A., Mori T., Ban T., Takahashi T., Inoue D. Automatically Generating Malware Analysis Reports UsingSandbox Logs. *IEICE transactions on information and systems E101–D.* 2018. Vol. 11. P. 2622-2632. doi: 10.1587/transinf.2017ICP0011

121. Symantec Endpoint Security Complete. Available at: https://www.broadcom.com/products/cybersecurity/endpoint/end-user/complete (accessed 21.02.2025).

122. Taher F., AlFandi O., Al-kfairy M., Al Hamadi H., Alrabaee S. DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection. *Applied Sciences*. 2023. Vol. 13. 7720. https://doi.org/10.3390/app13137720

123. Talukder S. Tools and Techniques for Malware Detection and Analysis. arXiv: Cryptography and Security. 2020. URL: https://doi.org/10.48550/arXiv.2002.06819

124. Tayyab U.-H., Khan F. B., Durad M. H., Khan A., Lee Y. S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. *Journal of Cybersecurity and Privacy*. 2022. Vol. 2(4). P. 800–829. https://doi.org/10.3390/jcp2040041

125. Trend Micro. Available at: https://www.trendmicro.com/en_gb/business.html (accessed 21.02.2025).

126. Urooj U., Al-rimy B.A.S., Zainal A., Ghaleb F.A., Rassam M.A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Applied Sciences*. 2022. Vol. 12. P. 172. doi: 10.3390/ app12010172

127. Vasan D., Alazab M., Wassan S., Naeem H., Safaei B., Zheng Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*. 2020. Vol. 171. 107138. https://doi.org/10.1016/j.comnet.2020.107138

128. Virustotal. URL: https://www.virustotal.com/gui/home/upload

129. VX Heavens. URL: http://vxheaven.org/

130. Xiao G., Li J., Chen Y., Li K. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *Journal of Parallel and Distributed Computing*. 2020. Vol. 141. P. 49–58. https://doi.org/10.1016/j.jpdc.2020.03.012

131. Yara Rules. URL: https://yara.readthedocs.io/en/latest/writingrules.html

132. Yevseiev S., Pohasii S., Milevskyi S., Milov O., Melenti Y., Grod I., Berestov D., Fedorenko R., Kurchenko O. Development of a method for assessing the security of cyber-physical systems based on the Lotka–Volterra model. Eastern-European Journal of Enterprise Technologies. 2021. Vol. 5(9(113). P. 30–47. DOI: 10.15587/1729-4061.2021.241638

133. Yevseiev S., Melenti Y., Voitko O., Hrebeniuk V., Korchenko A., Mykus S., Milov O., Prokopenko O., Sievierinov O., Chopenko, D. Development of a concept for

building a critical infrastructure facilities security system, *Eastern-European Journal of Enterprise Technologies*. 2021. Vol. 3. P. 63–83. DOI:10.15587/1729-4061.2021.233533

134. Yuan B., Wang J., Wu P., Qing X. IoT malware classification based on lightweight convolutional neural networks. *IEEE Internet of Things Journal*. 2021. Vol. 9(5). P. 3770–3783. https://doi.org/10.1109/JIOT.2021.3100063

135. Zeek Network Security Monitor. URL: https://zeek.org/

136. Zhang Zh. Review on String-Matching Algorithm. *SHS Web of Conferences*. 2022. Vol. 144. 03018. doi: 10.1051/shsconf/202214403018

137. Дудикевич В. Б., Микитин Г. В., Ребець А. І. Квінтесенція інформаційної безпеки кіберфізичної системи. *Вісник Національного університету «Львівська політехніка». Інформаційні системи та мережі*. 2018. № 887. С. 58–68.

138. Летичевський О. О. Сучасні наукові проблеми кібербезпеки. Visnik Nacional noi academii nauk Ukraini. 2021. Vol. 2. С. 12–20. doi: 10.15407/visn2023.02.012.

139. Лисенко С.М., Щука Р.В.. Аналіз методів виявлення зловмисного програмного забезпечення в комп’ютерних системах. *Вісник Хмельницького національного університету. Технічні науки.* 2020. №2 (283). С. 101-107.

140. Нічепорук А. О. Використання нечіткої класифікації для виявлення метаморфних вірусів в корпоративній мережі. *Вісник Хмельницького національного університету*. 2016. №4 (239). С. 128-132.

141. Нічепорук А. О. Інформаційна технологія виявлення метаморфних вірусів у локальних комп'ютерних мережах : автореф. дис. на здобуття наук. ступеня канд. техн. наук : [спец.] 05.13.06 "Інформ. технології" / Нічепорук Андрій Олександрович ; Тернопіл. нац. екон. ун-т, [Хмельниц. нац. ун-т]. Тернопіль, 2018. 22 с.

142. Нічепорук А. О., Савенко О. С. Моделі життєвого циклу поліморфних вірусів. *Комп’ютерно-інтегровані технології: освіта, наука, виробництво*. 2013. № 11. С. 64–71.

143. Погасій С.С. Моделі і методи захисту інформації в кіберфізичних системах. – Рукопис. Дисертація на здобуття наукового ступеня доктора технічних

наук за спеціальністю 05.13.21 «Системи захисту інформації». – Державний університет інформаційно-комунікаційних технологій, Київ, 2024.

144. Погасій С. Моделі і методи захисту інформації в кіберфізічних системах. *Ukrainian Scientific Journal of Information Security*. 2022. № 28. С. 67-79. DOI: 10.18372/2225-5036.28.16951.

145. Савенко Б. О. Метод виявлення worm-вірусів згідно багатокласової класифікації. *Вісник Хмельницького національного університету. Серія: Технічні науки.* 2024. № 1 (331). С. 18-28. DOI: https://doi.org/10.31891/2307-5732-2024-331-2

146. Савенко Б. О. Метод синтезу математичних моделей рівнів безпеки для частково централізованих розподілених систем виявлення зловмисного програмного забезпечення. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки*. 2023. № 3. Ч. 1. С. 217-227. DOI: https://doi.org/10.32782/2663-5941/2023.3.1/34

147. Савенко О. С. Критерії класифікації методів виявлення зловмисного програмного забезпечення. *Вісник Хмельницького національного університету. Технічні науки.* 2018. № 1. С. 23–27.

148. Савенко О. С., Лисенко С. М., Крищук А. Ф. Діагностування комп'ютерних систем на наявність зловмисного програмного забезпечення на основі антивірусної мультиагентної системи. *Вісник Національного університету «Львівська політехніка».* 2011. № 5. С. 99–105.

149. Савенко О. С., Лисенко С. М., Нічепорук А. О. Моделі рівнів поліморфних комп'ютерних вірусів. *Вісник Вінницького політехнічного інституту*. 2015. № 2. С. 75–83.

150. Савенко О. С., Нічепорук А. О., Лисенко С. М. Метод виявлення поліморфних вірусів на основі модифікованих емуляторів. *Радіоелектронні і комп'ютерні системи.* 2016. № 6 (80). С. 35-40.

151. Савенко О., Чайковський М. Метод нечіткої класифікації зловмисного програмного забезпечення з використанням інтелектуального агента. *Information Technology: Computer Science, Software Engineering and Cyber Security.* 2024. № 3. С. 140-148. URL: https://journals.politehnica.dp.ua/index.php/it/article/view/644/574

152. Чайковський М. Архітектура та засоби мультиагентної системи виявлення поліморфних вірусів в комп'ютерних мережах. *Measuring and Computing Devices in Technological Processes*. 2025. № 1. С. 278–286. DOI: https://doi.org/10.31891/2219-9365-2025-81-34

153. Чайковський М. Комплексний підхід до виявлення та аналізу поліморфного зловмисного програмного забезпечення. *Measuring and Computing Devices in Technological Processes.* 2024. № 2. С. 42-50. DOI: https://doi.org/10.31891/2219-9365-2024-78-5

154. Чайковський М. Модель мультиагентної системи для виявлення поліморфних вірусів. *Herald of Khmelnytskyi National University. Technical Sciences*. 2024. № 347(1). С. 543-547. DOI: https://doi.org/10.31891/2307-5732-2025-347-74

155. Чайковський М.Ю. Прогнозування кількості атак зловмисного програмного забезпечення у світі. *Актуальні проблеми комп'ютерних наук АПКН-2024 : матеріали XVI всеукр. наук.-практ. конф*., м. Хмельницький, 15-16 лист. 2024 р. / Хмельницький національний університет. Хмельницький, 2024. С. 534-536.

ДОДАТКИ

ДОДАТОК А.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Чайковський М. Комплексний підхід до виявлення та аналізу поліморфного зловмисного програмного забезпечення. *Measuring and Computing Devices in Technological Processes*. 2024. № 2. С. 42-50. DOI: https://doi.org/10.31891/2219-9365-2024-78-5

2. Савенко О., Чайковський М. Метод нечіткої класифікації зловмисного програмного забезпечення з використанням інтелектуального агента. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2024. № 3. С. 140-148. DOI: https://doi.org/10.32782/IT/2024-3-15

3. Чайковський М. Модель мультиагентної системи для виявлення поліморфних вірусів. *Herald of Khmelnytskyi National University. Technical Sciences*. 2024. № 347(1). С. 543-547. DOI: https://doi.org/10.31891/2307-5732-2025-347-74

4. Чайковський М. Архітектура та засоби мультиагентної системи виявлення поліморфних вірусів в комп'ютерних мережах. *Measuring and Computing Devices in Technological Processes*. 2025. № 1. С. 278–286. DOI: https://doi.org/10.31891/2219-9365-2025-81-34

Праці, які засвідчують апробацію матеріалів дисертації

5. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Comprehensive approach to the detection and analysis of polymorphic malware. *CEUR Workshop Proceedings* (ISSN 1613-0073). 2024. Vol.3736. P.312-323. URL: https://ceur-ws.org/Vol-3736/paper23.pdf (Scopus)

6. Chaikovskyi M., Chaikovska I., Sochor T., Martyniuk I., Lyhun O. Modeling the detection process of polymorphic malware based on the Lotka-Volterra Model. *CEUR Workshop Proceedings* (ISSN 1613-0073). 2025. Vol.3899. P. 244-253. URL: https://ceur-ws.org/Vol-3899/paper22.pdf (Scopus)

7. Чайковський М.Ю. Прогнозування кількості атак зловмисного програмного забезпечення у світі. *Актуальні проблеми комп'ютерних наук АПКН-2024 : матеріали XVI всеукр. наук.-практ. конф., м. Хмельницький, 15-16 лист. 2024 р.* / Хмельницький національний університет. Хмельницький, 2024. С. 534-536. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

ДОДАТОК Б.

АКТИ ВПРОВАДЖЕННЯ

«Затверджую»
Проректор з науково-педагогічної роботи
Віктор ЛОПАТОВСЬКИЙ
«16» квітня 2025 р.

АКТ

про впровадження в навчальний процес Хмельницького національного університету результатів дисертаційної роботи аспіранта кафедри комп'ютерної інженерії та інформаційних систем Чайковського Максима Юрійовича «Методи та засоби виявлення поліморфних вірусів за допомогою гібридної мультиагентної системи»

Ми, комісія в складі: декана факультету інформаційних технологій, професора кафедри комп'ютерної інженерії та інформаційних систем, д.т.н., професора Говорущенко Т. О. (голова комісії), доцента кафедри комп'ютерної інженерії та інформаційних систем, к.т.н., доцента Нічепорука А. О., доцента кафедри комп'ютерної інженерії та інформаційних систем, к.т.н., доцента Капустян М. В. склала акт про те, що результати дисертаційної роботи Чайковського М.Ю. впроваджені та використовуються в освітньому процесі Хмельницького національного університету при викладанні дисциплін на кафедрі комп'ютерної інженерії та інформаційних систем для спеціальності 123 Комп'ютерна інженерія, зокрема в курсах «Теорія і проєктування комп'ютерних та кіберфізичних систем і мереж», «Теорія і технології проєктування спеціалізованих операційних систем», «Методи розв'язування наукових задач комп'ютерної інженерії».

При викладанні цих дисциплін викладачами кафедр використовувалися наступні матеріали досліджень, отримані Чайковським М.Ю. особисто:

1) модель архітектури гібридної мультиагентної системи виявлення поліморфних вірусів, яка каскадно делегуючи повноваження, здійснює аналіз середовища, досліджує темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів, а також здійснювати класифікацію поліморфних вірусів за рівнями складності, що підвищує рівень ефективності обраних стратегій прийняття рішення;

2) метод виявлення поліморфних вірусів, який дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне

комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів;

3) метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати;

4) метод класифікації поліморфних вірусів, який, на відміну від існуючих, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обґрунтованості та ефективність обраних стратегій для їх нейтралізації.

Отримані матеріали досліджень дозволили розробити лабораторні практикуми з використанням моделі архітектури та методу створення гібридних мультиагентних систем, методу виявлення та класифікації поліморфних вірусів.

Говорущенко Т. О.
Нічепорук А. О.
Капустян М. В.

«Затверджую»
Генеральний директор ПП «Нолт Технолоджис»
Олександр МЕЛЬНИЧЕНКО
«__» квітня 2025 р.

АКТ
про впровадження результатів дисертаційної роботи
аспіранта кафедри комп'ютерної інженерії та інформаційних систем
Хмельницького національного університету Чайковського Максима Юрійовича
«Методи та засоби виявлення поліморфних вірусів за допомогою гібридної мультиагентної системи»

Результати дисертаційної роботи аспіранта кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету Чайковського М.Ю. впроваджені на ПП «Нолт Технолоджис».

При розробленні мультиагентної системи виявлення зловмисного програмного забезпечення в комп'ютерних мережах, включаючи підсистему виявлення поліморфних вірусів, були використані на ПП «Нолт Технолоджис» такі результати, які одержані Чайковським М.Ю. особисто:

1) метод встановлення темпу поширення поліморфних вірусів на основі використання моделі Лотки-Вольтерра, який дозволяє дослідити вплив кількості використаних методів виявлення поліморфних вірусів на темп їх поширення у коливальному процесі та дає змогу визначити, яку кількість методів виявлення поліморфних вірусів необхідно використати;

2) метод виявлення поліморфних вірусів, який дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів;

3) метод класифікації поліморфних вірусів, який, на відміну від існуючих, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обґрунтованості та ефективність обраних стратегій для їх нейтралізації.

Виконані дослідження включають методи та гібридну мультиагентну систему виявлення поліморфних вірусів в комп'ютерних мережах за рахунок синтезу методів таким чином, щоб максимізувати ймовірність успішного виявлення поліморфних вірусів різних рівнів складності.

Отриманні результати покращують ефективність функціонування мультиагентних систем для виявлення поліморфних вірусів в комп'ютерних мережах порівняно із іншими системами приблизно на 8-11%.

Цей акт не є підставою для фінансових розрахунків.

Генеральний директор Олександр МЕЛЬНИЧЕНКО

«Затверджую»
Директор ТОВ «ІТТ»
Сімогук В.С.
«14» квітня 2025 р.

АКТ

про впровадження результатів дисертаційної роботи
аспіранта кафедри комп'ютерної інженерії та інформаційних систем
Хмельницького національного університету Чайковського Максима Юрійовича
«Методи та засоби виявлення поліморфних вірусів за допомогою гібридної мультиагентної системи»

Результати дисертаційної роботи Чайковського М.Ю. аспіранта кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету впроваджені на ТОВ «ІТТ».

При розробленні мультиагентної системи виявлення поліморфних вірусів були використані на ТОВ «ІТТ» такі результати, які одержані Чайковським М.Ю. особисто:

1) модель архітектури гібридної мультиагентної системи виявлення поліморфних вірусів, яка каскадно делегуючи повноваження, здійснює аналіз середовища, досліджує темпи поширення, виявляє, класифікує поліморфні віруси та використовує різні стратегії прийняття рішення, враховуючи типи загроз, що дало змогу визначати та використовувати оптимальний комплекс методів для виявлення поліморфних вірусів;

2) метод виявлення поліморфних вірусів, який дозволяє не лише збільшити ефективність виявлення поліморфних вірусів, але й визначати ймовірність належності виявлених вірусів до класу поліморфних та передбачає трьохетапне комбіноване застосування, з якого, на першому етапі використовуються алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів;

3) метод класифікації поліморфних вірусів, який, на відміну від існуючих, дозволяє не лише класифікувати поліморфні віруси за рівнями складності їх будови, але й визначати ймовірність їх належності до нечітких термів на рівні низький, нижче середнього, середній, вище середнього, високий кожного рівня складності, що дало змогу підвищити рівень обґрунтованості та ефективність обраних стратегій для їх нейтралізації.

Виконані дослідження включають модель архітектури гібридної

мультиагентної системи, яка використовує метод виявлення поліморфних вірусів, який передбачає комплексне поєднання на першому алгоритми пошуку рядка, на другому – інтелектуальний аналіз даних, аналіз в пісочниці, машинне навчання, метод розробки структурних функцій, на третьому - ймовірнісні логічні мережі, що дало змогу класифікувати виявлені загрози за рівнем ймовірності їх належності до поліморфних вірусів. А використання запропонованого методу нечіткої класифікації поліморфних вірусів дало змогу підвищити рівень обґрунтованості та ефективність обраних стратегій для їх нейтралізації.

Отриманні результати дозволили покращити ефективність функціонування мультиагентної системи виявлення поліморфних вірусів в комп'ютерних мережах порівняно із системами з відомими зловмисникам архітектурами приблизно на 9-10%.

Цей акт не є підставою для фінансових розрахунків.

Заступник директора Іванов О.В.

Системний адміністратор Веремеєнко В.А.

ДОДАТОК В.
ЛІСТИНГ ПРОГРАМНОГО КОДУ (ФРАГМЕНТ)

Основний клас агента

```
// thoth.ts

import WebSocket, { WebSocketServer } from 'ws';
import { v4 as uuidv4 } from 'uuid'; // Для ID запитів

// Типи повідомлень
type MessageType = 'handshake' | 'method_call' | 'method_response' |
'error' | 'text_message';
type MethodHandler = (...args: any[]) => any;
type TextMessageHandler = (senderName: string, message: string) =>
void;

// Уніфікований інтерфейс повідомлення
interface Message {
    type: MessageType;
    id?: string;       // RPC (запит/відповідь/помилка)
    name?: string;     // handshake (ім'я відправника)
    method?: string;   // method_call
    params?: any[];    // method_call
    result?: any;      // method_response
    error?: string;    // error
    content?: string;  // text_message
}

// Інтерфейси Rule та BeliefEntry залишаються, якщо потрібні для
майбутнього
interface Rule { /* ... */ }
interface BeliefEntry { /* ... */ }

// Стан для неідентифікованих з'єднань на сервері
type SocketState = 'pending_handshake' | 'identified';
interface ServerConnectionInfo {
    socket: WebSocket;
    state: SocketState;
```

```
    name?: string; // Ім'я буде встановлено після handshake
}

export class Agent {
    private name: string;
    private methods: Map<string, MethodHandler> = new Map();
    private pendingRequests: Map<string, { resolve: Function;
reject: Function; timer: NodeJS.Timeout }> = new Map();
    private server?: WebSocketServer;
    // Зберігаємо ідентифіковані з'єднання: Ім'я -> Сокет
    private connections: Map<string, WebSocket> = new Map();
    // Тимчасове сховище для сокетів на сервері до handshake
    private pendingServerSockets: Map<WebSocket,
ServerConnectionInfo> = new Map();
    private textMessageHandler?: TextMessageHandler;
    private rules: Map<string, Rule> = new Map(); // логіка
    private beliefs: Map<string, BeliefEntry> = new Map(); //
вірування
    private decisionThreshold: number = 0.5; // рішення
    private readonly requestTimeoutMs = 15000; // Таймаут для RPC
запитів (15 секунд)

    constructor(name: string, port?: number) {
        if (!name || typeof name !== 'string' || name.trim().length
=== 0) {
            throw new Error("Agent name must be a non-empty
string.");
        }
        this.name = name;
        if (port) {
            this.startServer(port);
        }
    }

    public getName(): string {
        return this.name;
    }

    // --- Керування сервером ---
    public startServer(port: number): void {
```

```
        if (this.server) {
            console.warn(`Agent ${this.name}: Server already
running.`);
            return;
        }
        this.server = new WebSocketServer({ port });
        console.log(`Agent ${this.name}: Starting server on port
${port}...`);

        this.server.on('connection', (socket: WebSocket) => {
            console.log(`Agent ${this.name}: Incoming connection
received. Waiting for handshake.`);
            const connectionInfo: ServerConnectionInfo = { socket,
state: 'pending_handshake' };
            this.pendingServerSockets.set(socket, connectionInfo);

            socket.on('message', (data: WebSocket.Data) => {
                this.handleServerMessage(connectionInfo, data);
            });

            socket.on('close', () => {
                this.handleServerDisconnection(connectionInfo);
            });

            socket.on('error', (error: Error) => {
                console.error(`Agent ${this.name}: WebSocket error
on connection (state: ${connectionInfo.state}, name:
${connectionInfo.name ?? 'N/A'}):`, error);
                this.handleServerDisconnection(connectionInfo); //
Обробляємо як дисконект
                // Додатково можна закрити сокет, якщо він ще не
закритий
                 try { socket.close(); } catch (e) { /* ігноруємо
помилку закриття */ }
            });

            // Додамо тайм-аут для handshake
            const handshakeTimeout = setTimeout(() => {
                if (connectionInfo.state === 'pending_handshake') {
```

```
                    console.warn(`Agent ${this.name}: Handshake
timeout for incoming connection. Closing socket.`);
                     this.pendingServerSockets.delete(socket);
                    socket.close();
                }
            }, 5000); // 5 секунд на handshake

            // Очистка тайм-ауту при закритті
            socket.on('close', () =>
clearTimeout(handshakeTimeout));

        });

        this.server.on('listening', () => {
             console.log(`Agent ${this.name}: Server listening
successfully on port ${port}.`);
        });

        this.server.on('error', (error) => {
            console.error(`Agent ${this.name}: Server error:`,
error);
            this.server = undefined; // Вважаємо сервер неактивним
        });
    }

    // Обробка повідомлень на стороні сервера (з логікою handshake)
    private handleServerMessage(connectionInfo:
ServerConnectionInfo, data: WebSocket.Data): void {
        let message: Message;
        try {
            message = JSON.parse(data.toString());
        } catch (error) {
            console.error(`Agent ${this.name}: Failed to parse
message from ${connectionInfo.name ?? 'pending handshake agent'}:`,
data.toString(), error);
             if (connectionInfo.state === 'pending_handshake') {
                 connectionInfo.socket.close(); // Закриваємо, якщо
запит до handshake
             }
            return;
```

```
        }

        // --- Логіка Handshake ---
        if (connectionInfo.state === 'pending_handshake') {
            if (message.type === 'handshake' && message.name &&
typeof message.name === 'string' && message.name.trim().length > 0)
{
                const remoteName = message.name;
                if (remoteName === this.name) {
                    console.warn(`Agent ${this.name}: Connection
attempt from agent with the same name refused.`);
                    this.pendingServerSockets.delete(connectionInfo
.socket);
                    connectionInfo.socket.close();
                    return;
                }
                 if (this.connections.has(remoteName)) {
                    console.warn(`Agent ${this.name}: Agent
'${remoteName}' is already connected. Refusing new connection.`);
                    // Якшошо, відправляємо помилку перед закриттям
(опціонально)
                    // this.sendError(connectionInfo.socket,
undefined, `Agent '${remoteName}' already connected.`);
                    this.pendingServerSockets.delete(connectionInfo
.socket);
                    connectionInfo.socket.close();
                    return;
                 }
                // Handshake успішний
                connectionInfo.state = 'identified';
                connectionInfo.name = remoteName;
                this.connections.set(remoteName,
connectionInfo.socket);
                this.pendingServerSockets.delete(connectionInfo.sock
et); // Видаляємо з очікуючих
                console.log(`Agent ${this.name}: Handshake
successful. Identified remote agent as '${remoteName}'.`);
                // Якшошо, відправляємо підтвердження handshake
(опціонально)
```

```
                // connectionInfo.socket.send(JSON.stringify({
type: 'handshake_ack' }));
            } else {
                // Неправильне перше повідомлення
                console.warn(`Agent ${this.name}: Received invalid
first message (expected handshake) from connecting agent. Closing
connection. Message:`, message);
                this.pendingServerSockets.delete(connectionInfo.sock
et);
                connectionInfo.socket.close();
            }
            return; // Не обробляємо handshake як звичайне
повідомлення
        }

        // --- Обробка звичайних повідомлень (після handshake) ---
        if (connectionInfo.state === 'identified' &&
connectionInfo.name) {
             this.handleIncomingData(connectionInfo.socket, message,
connectionInfo.name);
        } else {
             // Це не повинно статися, якщо логіка коректна
             console.error(`Agent ${this.name}: Received message on
socket in unexpected state: ${connectionInfo.state}`);
        }
    }

    // Обробка відключення на стороні сервера
    private handleServerDisconnection(connectionInfo:
ServerConnectionInfo): void {
         this.pendingServerSockets.delete(connectionInfo.socket); //
Видаляємо з очікуючих, якщо є
        if (connectionInfo.state === 'identified' &&
connectionInfo.name) {
            if (this.connections.get(connectionInfo.name) ===
connectionInfo.socket) {
                 this.connections.delete(connectionInfo.name);
                 console.log(`Agent ${this.name}: Agent
'${connectionInfo.name}' disconnected.`);
            }
```

```
        } else {
            console.log(`Agent ${this.name}: Unidentified agent
disconnected (or disconnected before handshake).`);
        }
    }


    // --- Керування клієнтськими з'єднаннями ---
    public async connectTo(agentName: string, host: string, port:
number): Promise<void> {
        if (agentName === this.name) {
             return Promise.reject(new Error("Cannot connect to
self."));
        }
        if (this.connections.has(agentName)) {
            const existingSocket = this.connections.get(agentName);
            if (existingSocket?.readyState === WebSocket.OPEN ||
existingSocket?.readyState === WebSocket.CONNECTING) {
                console.log(`Agent ${this.name}: Already connected
or connecting to '${agentName}'.`);
                return Promise.resolve();
            } else {
                this.connections.delete(agentName); // Видалити
старий закритий сокет
            }
        }

        console.log(`Agent ${this.name}: Attempting to connect to
'${agentName}' at ws://${host}:${port}...`);
        return new Promise((resolve, reject) => {
            const socket = new WebSocket(`ws://${host}:${port}`);
            let connectionEstablished = false; // Сімафор, щоб
reject викликався лише раз

            const openHandler = () => {
                connectionEstablished = true;
                this.connections.set(agentName, socket);
                console.log(`Agent ${this.name}: Successfully
connected to '${agentName}'. Sending handshake...`);
                // Відправляємо handshake
```

```
                try {
                    const handshakeMsg: Message = { type:
'handshake', name: this.name };
                    socket.send(JSON.stringify(handshakeMsg));
                    console.log(`Agent ${this.name}: Handshake sent
to '${agentName}'.`);
                     // Встановлюємо обробники після успішного open
і handshake
                    socket.on('message', (data: WebSocket.Data) => {
                        let message: Message;
                        try {
                            message = JSON.parse(data.toString());
                        } catch(error) {
                             console.error(`Agent ${this.name}:
Failed to parse message from '${agentName}':`, data.toString(),
error);
                             return;
                        }
                        // На клієнті ми знаємо ім'я віддаленого
агента
                        this.handleIncomingData(socket, message,
agentName);
                    });
                     resolve(); // З'єднання успішне
                } catch (sendError) {
                     console.error(`Agent ${this.name}: Failed to
send handshake to '${agentName}':`, sendError);
                     this.connections.delete(agentName); //
Видаляємо, якщо handshake не відправився
                     try { socket.close(); } catch(e) {}
                     reject(new Error(`Failed to send handshake to
${agentName}: ${sendError instanceof Error ? sendError.message :
String(sendError)}`));
                }
            };

            const closeHandler = () => {
                // Видаляємо з'єднання, тільки якщо воно ще існує і
це той самий сокет
                 if (this.connections.get(agentName) === socket) {
```

```
                    this.connections.delete(agentName);
                    console.log(`Agent ${this.name}: Disconnected
from agent '${agentName}'.`);
                }
                // Якщо з'єднання не було встановлено, відклоняємо
Promise
                if (!connectionEstablished) {
                    reject(new Error(`Connection to agent
${agentName} closed before handshake completed.`));
                }
            };

            const errorHandler = (error: Error) => {
                console.error(`Agent ${this.name}: WebSocket error
connecting to '${agentName}':`, error.message);
                if (this.connections.get(agentName) === socket) {
                    this.connections.delete(agentName);
                }
                // Відклоняємо Promise, тільки якщо помилка сталася
*до* успішного 'open'
                if (!connectionEstablished) {
                    reject(new Error(`Failed to connect to agent
${agentName}: ${error.message}`));
                }
                 // Спробувати закрити сокет, якщо ще не закритий
                try { if (socket.readyState !== WebSocket.CLOSED)
socket.close(); } catch(e) {}
            };

            socket.once('open', openHandler);
            socket.once('close', closeHandler);
            socket.once('error', errorHandler);
        });
    }

    // --- Загальний обробник вхідних даних (після ідентифікації) --
-
    private handleIncomingData(socket: WebSocket, message: Message,
senderName: string): void {
```

```
        // senderName вже відомий (з handshake на сервері або з
connectTo на клієнті)
        try {
            switch (message.type) {
                case 'method_call':
                    if (message.id && message.method) {
                        this.handleMethodCall(socket, message,
senderName);
                    } else {
                        console.warn(`Agent ${this.name}: Received
invalid method_call from '${senderName}':`, message);
                        if (message.id) this.sendError(socket,
message.id, 'Invalid method_call structure');
                    }
                    break;
                case 'method_response':
                    if (message.id) {
                        this.handleMethodResponse(message);
                    } else {
                         console.warn(`Agent ${this.name}: Received
invalid method_response (missing id) from '${senderName}':`,
message);
                    }
                    break;
                case 'error':
                     if (message.id) {
                        this.handleErrorResponse(message);
                     } else {
                         console.warn(`Agent ${this.name}: Received
error message without id from '${senderName}':`, message.error);
                     }
                    break;
                case 'text_message':
                    if (typeof message.content === 'string') {
                        this.handleTextMessage(senderName,
message.content);
                    } else {
                         console.warn(`Agent ${this.name}: Received
invalid text_message (missing/invalid content) from
'${senderName}':`, message);
```

```
                    }
                    break;
                case 'handshake':
                    // Ігноруємо повторні handshake після
ідентифікації
                     console.warn(`Agent ${this.name}: Received
unexpected handshake message from already identified agent
'${senderName}'. Ignoring.`);
                    break;
                default:
                    console.warn(`Agent ${this.name}: Received
message with unknown type '${(message as any).type}' from
'${senderName}':`, message);
            }
        } catch (error) {
            console.error(`Agent ${this.name}: Error processing
message from '${senderName}':`, error, "Original message:",
message);
            // Розглянути відправку помилки, якщо це був запит з ID
            if (message.type === 'method_call' && message.id) {
                this.sendError(socket, message.id, `Internal server
error processing message: ${error instanceof Error ? error.message :
String(error)}`);
            }
        }
    }

    // --- Обробка RPC ---
    public addMethod(name: string, handler: MethodHandler): void {
        this.methods.set(name, handler);
        console.log(`Agent ${this.name}: Method '${name}' added.`);
    }

    public removeMethod(name: string): boolean {
        const deleted = this.methods.delete(name);
        if (deleted) {
            console.log(`Agent ${this.name}: Method '${name}'
removed.`);
        }
        return deleted;
```

```
    }

    public listMethods(): string[] {
        return Array.from(this.methods.keys());
    }

    public async callRemoteMethod(agentName: string, methodName:
string, ...args: any[]): Promise<any> {
        const socket = this.connections.get(agentName);
        if (!socket) {
            return Promise.reject(new Error(`Not connected to agent
'${agentName}'`));
        }
        if (socket.readyState !== WebSocket.OPEN) {
            return Promise.reject(new Error(`Connection to agent
'${agentName}' is not open (state: ${socket.readyState})`));
        }

        return new Promise((resolve, reject) => {
            const id = this.generateRequestId();
            const message: Message = {
                type: 'method_call',
                id,
                method: methodName,
                params: args
            };

            const timeoutId = setTimeout(() => {
                // Перевіряємо, чи запит ще очікує
                if (this.pendingRequests.has(id)) {
                    this.pendingRequests.delete(id);
                    reject(new Error(`Request ${id} ('${methodName}'
on '${agentName}') timed out after ${this.requestTimeoutMs}ms`));
                }
            }, this.requestTimeoutMs);

            const cleanup = () => {
                 clearTimeout(timeoutId);
                 this.pendingRequests.delete(id);
            };
```

```
            const resolver = (result: any) => {
                cleanup();
                resolve(result);
            };
            const rejecter = (error: Error) => {
                cleanup();
                reject(error);
            };

            this.pendingRequests.set(id, { resolve: resolver,
reject: rejecter, timer: timeoutId });

            try {
                socket.send(JSON.stringify(message));
                console.log(`Agent ${this.name}: Sent RPC call
'${methodName}' [${id}] to '${agentName}'.`);
            } catch (sendError) {
                rejecter(new Error(`Failed to send RPC call to
${agentName}: ${sendError instanceof Error ? sendError.message :
String(sendError)}`));
            }
        });
    }

    // Обробник вхідного RPC виклику
    private async handleMethodCall(socket: WebSocket, message:
Message, senderName: string): Promise<void> {
        const method = this.methods.get(message.method!);
        console.log(`Agent ${this.name}: Received RPC call
'${message.method!}' [${message.id!}] from '${senderName}'.`);
        if (!method) {
            this.sendError(socket, message.id!, `Method
'${message.method!}' not found on agent ${this.name}`);
            return;
        }

        try {
            const result = await method(...(message.params || []));
            const response: Message = {
```

```
                type: 'method_response',
                id: message.id!,
                result
            };
             if (socket.readyState === WebSocket.OPEN) {
                  socket.send(JSON.stringify(response));
                  console.log(`Agent ${this.name}: Sent RPC response
for [${message.id!}] to '${senderName}'.`);
             } else {
                 console.warn(`Agent ${this.name}: Cannot send RPC
response for [${message.id!}] to '${senderName}', socket is not
open.`);
             }
        } catch (error) {
            console.error(`Agent ${this.name}: Error executing
method '${message.method!}' called by '${senderName}':`, error);
            this.sendError(socket, message.id!, `Error executing
method '${message.method!}': ${error instanceof Error ?
error.message : String(error)}`);
        }
    }

    // Обробник відповіді на наш RPC виклик
    private handleMethodResponse(message: Message): void {
        const request = this.pendingRequests.get(message.id!);
        if (!request) {
            console.warn(`Agent ${this.name}: Received response for
unknown/timed-out request ID: ${message.id}`);
            return;
        }
         console.log(`Agent ${this.name}: Received RPC response for
[${message.id!}].`);
        request.resolve(message.result); // cleanup відбувається
всередині resolve
    }

    // Обробник помилки на наш RPC виклик
    private handleErrorResponse(message: Message): void {
        const request = this.pendingRequests.get(message.id!);
        if (!request) {
```

```
            console.warn(`Agent ${this.name}: Received error for
unknown/timed-out request ID: ${message.id}`);
            return;
        }
         console.log(`Agent ${this.name}: Received RPC error for
[${message.id!}]: ${message.error}`);
        request.reject(new Error(message.error || `Unknown error
from remote agent for request ${message.id}`)); // cleanup
відбувається всередині reject
    }

    // Відправка повідомлення про помилку у відповідь на запит
    private sendError(socket: WebSocket, id: string, errorMessage:
string): void {
        if (!id) {
             console.error(`Agent ${this.name}: Cannot send error
response without original message ID. Error: ${errorMessage}`);
             return;
        }
        const response: Message = { type: 'error', id, error:
errorMessage };
        try {
            if (socket.readyState === WebSocket.OPEN) {
                 socket.send(JSON.stringify(response));
                 console.log(`Agent ${this.name}: Sent error
response for [${id}]: ${errorMessage}`);
            } else {
                 console.warn(`Agent ${this.name}: Cannot send error
response for [${id}], socket is not open.`);
            }
        } catch (sendError) {
            console.error(`Agent ${this.name}: Failed to send error
response for [${id}]:`, sendError);
        }
    }

    // Генерація ID
    private generateRequestId(): string {
        return uuidv4();
    }
```

```
    // --- Обробка текстових повідомлень ---
    public onTextMessage(handler: TextMessageHandler): void {
        this.textMessageHandler = handler;
        console.log(`Agent ${this.name}: Text message handler
registered.`);
    }

    public sendTextMessage(agentName: string, messageText: string):
void {
        const socket = this.connections.get(agentName);
        if (!socket) {
            throw new Error(`Not connected to agent
'${agentName}'`);
        }
        if (socket.readyState !== WebSocket.OPEN) {
            throw new Error(`Connection to agent '${agentName}' is
not open (state: ${socket.readyState})`);
        }

        const message: Message = { type: 'text_message', content:
messageText };
        try {
            socket.send(JSON.stringify(message));
            console.log(`Agent ${this.name}: Sent text message to
'${agentName}': "${messageText}"`);
        } catch (error) {
            console.error(`Agent ${this.name}: Failed to send text
message to '${agentName}':`, error);
            throw error; // Посилаємо помилку далі
        }
    }

    // Обробник вхідного текстового повідомлення
    private handleTextMessage(senderName: string, content: string):
void {
        console.log(`Agent ${this.name}: Received text message from
'${senderName}': "${content}"`);
        if (this.textMessageHandler) {
            try {
```

```
                this.textMessageHandler(senderName, content);
            } catch (error) {
                console.error(`Agent ${this.name}: Error in custom
text message handler for message from '${senderName}':`, error);
            }
        } else {
            console.log(`(Agent ${this.name}: No custom text message
handler registered)`);
        }
    }

    // --- Зупинка агента ---
    public stop(): void {
        console.log(`Agent ${this.name}: Stopping...`);
        // Закриваємо всі клієнтські з'єднання
        for (const [name, socket] of this.connections) {
            console.log(`Agent ${this.name}: Closing connection to
'${name}'.`);
            try {
                 if(socket.readyState === WebSocket.OPEN ||
socket.readyState === WebSocket.CONNECTING){
                     socket.close();
                 }
            } catch (e) {
                 console.error(`Agent ${this.name}: Error closing
connection to '${name}':`, e);
            }
        }
        this.connections.clear();

         // Закриваємо всі сокети, що очікують handshake на сервері
        for (const socket of this.pendingServerSockets.keys()) {
            console.log(`Agent ${this.name}: Closing pending
handshake connection.`);
             try {
                  if(socket.readyState === WebSocket.OPEN ||
socket.readyState === WebSocket.CONNECTING){
                      socket.close();
                  }
             } catch (e) {
```

```
                    console.error(`Agent ${this.name}: Error closing
pending handshake connection:`, e);
                }
            }
            this.pendingServerSockets.clear();


            // Зупиняємо сервер, якщо він запущений
            if (this.server) {
                console.log(`Agent ${this.name}: Stopping server...`);
                this.server.close((err) => {
                    if (err) {
                        console.error(`Agent ${this.name}: Error
stopping server:`, err);
                    } else {
                        console.log(`Agent ${this.name}: Server
stopped.`);
                    }
                    this.server = undefined;
                });
            }

            // Скасовуємо всі очікуючі RPC запити
            for (const [id, request] of this.pendingRequests) {
                clearTimeout(request.timer); // Очищуємо таймер
                 request.reject(new Error(`Agent ${this.name} is
stopping. Request ${id} cancelled.`));
            }
            this.pendingRequests.clear();

            console.log(`Agent ${this.name}: Stopped.`);
        }
} // --- Кінець класу Agent ---
```

Запуск агента А

```
// runAgentA.ts
import { Agent } from './thoth';

const agentA = new Agent('AgentA', 8080);

agentA.addMethod('ping', (senderName: string) => {
  console.log(`AgentA received ping from ${senderName}`);
  // Відповідаємо текстовим повідомленням
  agentA.sendTextMessage(senderName, "Pong received! Thanks.");
  return 'pong from AgentA';
});

// Реєструємо обробник для текстових повідомлень
agentA.onTextMessage((senderName, message) => {
  console.log(`---> AgentA received text from ${senderName}:
"${message}"`);
  // Можна додати логіку відповіді або дії
  if (message.toLowerCase().includes('hello')) {
      setTimeout(() => {
          agentA.sendTextMessage(senderName, `Hello ${senderName},
AgentA here!`);
      }, 500); // Невелика затримка для імітації обробки
  }
});


async function connectAndInteract() {
  try {
    console.log('AgentA attempting to connect to AgentB...');
    await agentA.connectTo('AgentB', 'localhost', 8081);
    console.log('AgentA connected to AgentB.');

    // Надсилаємо текстове повідомлення
    agentA.sendTextMessage('AgentB', 'Hello AgentB from AgentA!');

    // Викликаємо метод 'hello' на Агенті Б
    const response = await agentA.callRemoteMethod('AgentB',
'hello', agentA.getName());
```

```
    console.log(`AgentA received RPC response from AgentB:
${response}`);

  } catch (error) {
    console.error('AgentA failed to connect or interact with
AgentB:', error);
    // setTimeout(connectAndInteract, 5000); // Спробувати ще раз
  }
}

setTimeout(connectAndInteract, 3000);

console.log(`Agent ${agentA.getName()} is running.`);
process.on('SIGINT', () => { agentA.stop(); process.exit(0); });
```

Запуск агента B

```
// runAgentB.ts
import { Agent } from './thoth'

const agentB = new Agent('AgentB', 8081);

agentB.addMethod('hello', (senderName: string) => {
  console.log(`AgentB received hello RPC from ${senderName}`);
  agentB.sendTextMessage(senderName, "RPC 'hello' executed
successfully!");
  return `Hi ${senderName}, this is AgentB!`;
});

// Реєструємо обробник для текстових повідомлень
agentB.onTextMessage((senderName, message) => {
  console.log(`---> AgentB received text from ${senderName}:
"${message}"`);
   if (message.toLowerCase().includes('pong')) {
      setTimeout(() => {
          agentB.sendTextMessage(senderName, `Got your pong,
${senderName}!`);
      }, 500);
  }
});

async function connectAndInteract() {
  try {
    console.log('AgentB attempting to connect to AgentA...');
    await agentB.connectTo('AgentA', 'localhost', 8080);
    console.log('AgentB connected to AgentA.');

     // Надсилаємо текстове повідомлення
    agentB.sendTextMessage('AgentA', 'Hi AgentA, AgentB here!');

    // Викликаємо метод 'ping' на Агенті A
    const response = await agentB.callRemoteMethod('AgentA', 'ping',
agentB.getName());
    console.log(`AgentB received RPC response from AgentA:
${response}`);
```

```
  } catch (error) {
    console.error('AgentB failed to connect or interact with
AgentA:', error);
     // setTimeout(connectAndInteract, 5000); // Спробувати ще раз
  }
}

setTimeout(connectAndInteract, 4000); // Трохи пізніше за Агента А

console.log(`Agent ${agentB.getName()} is running.`);
process.on('SIGINT', () => { agentB.stop(); process.exit(0); });
```

Програма керування агентною системою (теж агент)

```
// cli.ts

import readline from 'readline';
import { Agent } from './thoth'; // Імпортуємо наш клас Agent

// --- Конфігурація ---
const CLI_AGENT_NAME = 'CLI-User';
const CONNECT_TO_AGENT_NAME = 'AgentA'; // До якого агента
підключаємось
const CONNECT_TO_HOST = 'localhost';
const CONNECT_TO_PORT = 8080; // Порт агента AgentA

const OTHER_AGENT_NAME = 'AgentB'; // Ім'я іншого агента для
надсилання повідомлень/RPC
// --------------------


// Створюємо екземпляр агента для нашого CLI
// НЕ запускаємо сервер для CLI агента (не передаємо порт у
конструктор)
const cliAgent = new Agent(CLI_AGENT_NAME);

// Створюємо інтерфейс для читання команд з консолі
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
  prompt: '> ' // Запрошення для вводу
});

// Функція для виведення повідомлень без порушення поточного вводу
користувача
function logMessage(message: string) {
    readline.cursorTo(process.stdout, 0); // Перемістити курсор на
початок рядка
    readline.clearLine(process.stdout, 0); // Очистити поточний
рядок (де може бути ввід користувача)
    console.log(message); // Вивести повідомлення
    rl.prompt(true); // Показати запрошення '>' знову
```

```
}

// Реєструємо обробник для текстових повідомлень, що надходять до
cliAgent
// Це повідомлення від AgentA (або AgentB, якщо вони пишуть напряму
CLI)
cliAgent.onTextMessage((senderName, message) => {
    // Ми очікуємо, що AgentA пересилатиме нам повідомлення від
AgentB,
    // але поки що просто показуємо те, що прийшло
    logMessage(`[${senderName} SAYS]: ${message}`);
});

// Метод для CLI, щоб інші агенти могли йому щось сказати
cliAgent.addMethod('displayMessage', (sender: string, message:
string) => {
    logMessage(`[${sender} RPC MSG]: ${message}`);
    return "Message displayed in CLI.";
});


// Обробка команд, введених користувачем
rl.on('line', async (line) => {
  const input = line.trim();
  if (!input) {
    rl.prompt(); // Якщо пустий рядок, просто показати запршення
знову
    return;
  }

  const parts = input.split(' ');
  const command = parts[0].toLowerCase();
  console.log(parts)
  const targetAgent = parts[1]; // Ім'я агента, якому надсилаємо
(зазвичай OTHER_AGENT_NAME)
  const content = parts.slice(2).join(' '); // Решта - це контент
або параметри

  switch (command) {
```

```
    case 'text': // Надіслати текстове повідомлення: text
<targetAgent> <message>
      if (!targetAgent || !content) {
        logMessage("Usage: text <agentName> <message content>");
      } else {
        try {
          cliAgent.sendTextMessage(targetAgent, content);
          logMessage(`Text message sent to ${targetAgent}.`);
        } catch (error: any) {
          logMessage(`Error sending text: ${error.message}`);
        }
      }
      break;

    case 'rpc': // Викликати метод: rpc <targetAgent> <methodName>
[param1] [param2] ...
      const methodName = parts[2];
      const params = parts.slice(3);
      if (!targetAgent || !methodName) {
        logMessage("Usage: rpc <agentName> <methodName> [param1]
[param2] ...");
      } else {
        try {
          logMessage(`Calling ${methodName} on ${targetAgent}...`);
          // Тут ми викликаємо метод на targetAgent через агента, до
якого підключені
          const result = await
cliAgent.callRemoteMethod(targetAgent, methodName, ...params);
          logMessage(`[RPC RESULT from ${targetAgent}]:
${JSON.stringify(result)}`);
        } catch (error: any) {
          logMessage(`Error calling RPC: ${error.message}`);
        }
      }
      break;

    case 'help':
        logMessage("Commands:");
        logMessage("  text <agentName> <message>  - Send a text
message");
```

```
        logMessage("  rpc <agentName> <method> [args...] - Call a
remote method");
        logMessage("  quit                          - Exit the
client");
        logMessage("Example text: text AgentB Hello there!");
        logMessage("Example RPC:  rpc AgentB hello CLI-User"); //
Викликати метод 'hello' на AgentB
        logMessage("Example RPC:  rpc AgentA ping CLI-User");   //
Викликати метод 'ping' на AgentA
      break;

    case 'quit':
    case 'exit':
      rl.close(); // Закриваємо readline, що викличе подію 'close'
      break;

    default:
      logMessage(`Unknown command: '${command}'. Type 'help' for
available commands.`);
      break;
  }

  rl.prompt(); // Показуємо запрошення для наступної команди
});

// Обробка закриття readline (наприклад, командою quit)
rl.on('close', () => {
  console.log(`Disconnecting ${cliAgent.getName()}...`);
  cliAgent.stop(); // Зупиняємо нашого CLI агента (закриваємо
з'єднання)
  console.log('Goodbye!');
  process.exit(0);
});

// Обробка Ctrl+C
process.on('SIGINT', () => {
    rl.close();
});
```

```
// --- Основна логіка запуску ---
async function startCli() {
    console.log(`Starting ${cliAgent.getName()}...`);
    try {
        console.log(`Attempting to connect to
${CONNECT_TO_AGENT_NAME} at
ws://${CONNECT_TO_HOST}:${CONNECT_TO_PORT}...`);
        // Підключаємо наш CLI агент до вказаного агента (AgentA)
        await cliAgent.connectTo(CONNECT_TO_AGENT_NAME,
CONNECT_TO_HOST, CONNECT_TO_PORT);
        logMessage(`Successfully connected to
${CONNECT_TO_AGENT_NAME}. Type 'help' for commands.`);
        rl.prompt(); // Показуємо запрошення для вводу після
успішного підключення

    } catch (error: any) {
        console.error(`Failed to connect to
${CONNECT_TO_AGENT_NAME}: ${error.message}`);
        console.error('Please ensure AgentA is running and
accessible.');
        process.exit(1); // Виходимо, якщо не вдалося підключитися
    }
}

// Запускаємо клієнт
startCli();
```