ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

БОХОНЬКО ОЛЕКСАНДР ОЛЕКСАНДРОВИЧ

УДК 004.75 : 004.056.53

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМ, СТІЙКИХ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

123 Комп’ютерна інженерія
(шифр і назва спеціальності)

12 Інформаційні технології
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Подані до захисту наукові положення є власним напрацюванням. Всі використані ідеї, наукові результати, цитати супроводжуються належними посиланнями на їх авторів та джерела опублікування. Всі частини тексту дисертації, під час написання яких використовувались технології штучного інтелекту, перевірені та відредаговані особисто.

_____ Олександр Олександрович Бохонько
підпис

Науковий керівник:
Лисенко Сергій Миколайович, доктор технічних наук, професор;

Хмельницький – 2026

АНОТАЦІЯ

Бохонько О.О. Методи та засоби синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп'ютерна інженерія. – Хмельницький національний університет, Хмельницький, 2026.

Дослідження відомих методів та засобів виявило системний недолік – наявні рішення демонструють низьку ефективність синтезу систем, стійкої до атак соціальної інженерії, оскільки не включають комплексно в процес такого синтезу стійкої до атак архітектури, достовірність забезпечення виявлення атак, адаптивність, масштабованість, живучість та ефективність прийняття колективних рішень.

Таким чином, на даний момент часу існує суперечність між потребою в синтезі комп'ютерних системи, стійких до атак соціальної інженерії, з одного боку, і недосконалістю методів та засобів забезпечення стійкості РКС в умовах атак соціальної інженерії, з іншого боку. Відтак, підвищення стійкості розподілених комп'ютерних систем до атак соціальної інженерії є актуальною науково-прикладною задачею, одним із шляхів розв'язання якої є розроблення методів і засобів синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії.

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, такому об'єкту вивчення та діяльності, як «комп'ютерні системи,…, комп'ютерні мережі, методи та способи подання, отримання, зберігання, передавання, опрацювання та захисту в них інформації,…, архітектура та організація їх функціонування; інформаційні процеси, технології, методи, способи, інструментальні засоби та системи для дослідження, проектування, налагодження, виробництва й

експлуатації комп’ютерів та комп’ютерних систем і мереж, …, забезпечення якості, надійності та безпеки».

Об’єктом дослідження є процес процес синтезу стійких до атак соціальної інженерії розподілених комп’ютерних систем.

Предметом дослідження моделі, методи та засоби синтезу стійких до атак соціальної інженерії розподілених комп’ютерних систем.

Метою дослідження є підвищення стійкості до атак соціальної інженерії розподілених комп’ютерних систем шляхом розроблення методів та засобів синтезу стійких до атак соціальної інженерії РКС, які комплексно забезпечують достовірність виявлення атак, адаптивність, масштабованості, живучість та ефективність прийняття колективних рішень вузлів РКС.

Для досягнення поставленої мети були розв’язані такі задачі:

- проведено аналіз відомих методів і засобів забезпечення стійкості розподілених комп’ютерних систем до атак соціальної інженерії;

- розроблено формальну модель розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка описує колективну поведінку агентів у динамічному середовищі шляхом узгодженого прийняття рішень, обміну інформацією та адаптивного керування ресурсами з метою максимізації глобальної функції корисності, що відображає стійкість системи до атак соціальної інженерії, забезпечення достовірного виявлення загроз, підтримання безперервності функціонування, збереження живучості за умов часткової компрометації вузлів і масштабування системи;

- розроблено архітектуру стійкої до атак соціальної інженерії розподіленої комп’ютерної системи, яка базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак та підвищувати точність виявлення та класифікації атак соціальної інженерії;

- розроблено метод виявлення кібератак соціальної інженерії в розподілених комп’ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, застосуванні методу k-

найближчих сусідів, що уможливлює раннє виявлення мовних та семантичних маніпуляцій у сценаріях атак на розподіленої комп'ютерної системи;

- розроблено метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженеріїна основі популяційної моделі багатоагентної системи та середнього поля, що забезпечує формування оптимальної політики поведінки репрезентативного агента, інтеграцію архітектурних параметрів та гарантовану масштабованість системи при зростанні кількості вузлів і інтенсивності атак;

- розроблено метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів;

- розроблено архітектуру програмної реалізації розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, яка включає агента прийняття рішень, сервісних агентів, компоненти моніторингу станів, менеджер взаємодії та модулі мовного/семантичного аналізу; проведено експериментальні дослідження її характеристик у сценаріях впливів атак соціальної інженерії та оцінено покращення показників стійкості системи.

У дисертаційній роботі вперше розроблено метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії, який на відміну від відомих підходів поєднує принципи динамічної декомпозиції, багатоагентної взаємодії та адаптивного перерозподілу ресурсів з урахуванням поведінкових характеристик користувачів і загроз, що дає змогу забезпечити керовану масштабованість розподіленої системи без зниження рівня захищеності, підвищити її живучість за умов зростання кількості вузлів розподіленої КС та інтенсивності атак соціальної інженерії.

У дисертаційній роботі вперше розроблено метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, який на відміну від відомих методів ґрунтується на багатовимірній системі формалізованих критеріїв адаптивності, масштабованості, живучості та достовірності виявлення,

що дозволило отримати єдину універсальну метрику оцінювання стійкості РКС до атак соціальної інженерії.

У дисертаційній роботі набула подальшого розвитку архітектура стійкої до атак соціальної інженерії розподіленої комп'ютерної системи, яка на відміну від відомих базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, ентропійно-орієнтованими функціями винагороди, апріорними знаннями у вигляді графа знань та модально-специфічними сервісними агентами, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак, скорочувати кількість діалогових кроків і підвищувати точність виявлення та класифікації атак соціальної інженерії.

У дисертаційній роботі також удосконалено метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який на відміну від відомих підходів ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, їх попередній лінгвістичній нормалізації, експертному маркуванні та застосуванні методу k-найближчих сусідів із подальшим адаптивним налаштуванням гіперпараметрів і порогових значень довіри, що дає змогу підвищити точність та стійкість виявлення атак соціальної інженерії, зменшити кількість хибних спрацьовувань, забезпечити раннє реагування та інтеграцію результатів у контури захисту розподіленої комп'ютерної системи.

Практична цінність отриманих результатів полягає в реалізації усіх теоретичних положень, поданих в дисертаційному дослідженні, у прикладні рішення та можливості їх безпосереднього впровадження й використання на підприємствах.

За результатами виконаних досліджень здобувачем реалізовано розподілену комп'ютерну систему, стійку до атак соціальної інженерії. Практична цінність роботи полягає у можливості використання отриманих результатів для розроблення корпоративних політик безпеки, побудови

симуляційних тренажерів для дослідження взаємодії користувачів із атаками соціальної інженерії, створення інтелектуальних агентів для кіберзахисту та оптимізації архітектур розподілених систем з урахуванням ризиків. Запропоновані методи можуть застосовуватися у банківській, телекомунікаційній, енергетичній та державній сферах, де критично важливо забезпечити стійкість систем до складних поведінкових загроз.

Результати дисертаційної роботи впроваджено у (Додаток Б): ПП «АВІВІ» (акт впровадження від 08.1.2025 р.); ТОВ «ДЖІ ЕМ ХОСТ» (акт впровадження від 30.12.2025 р.); у навчальному процесі Хмельницького національного університету (акт впровадження від 30.09.2025 р.); при виконанні держбюджетної теми Хмельницького національного університету «Система виявлення ЗПЗ та комп'ютерних атак в корпоративних мережах з використанням хибних об'єктів атак та пасток» (ДР № 0124U000980).

Основні результати дисертації опубліковані у 9 наукових працях, серед яких 5 статей у фахових наукових журналах України, включених на дату опублікування до переліку наукових фахових видань України категорії Б; 4 публікації, які засвідчують апробацію матеріалів дисертації (статті в матеріалах конференцій, що індексуються в наукометричній базі Scopus).

Ключові слова: розподілені комп’ютерні системи, соціальна інженерія, багатоагентна система, синтез архітектури, середнє поле; навчання з підкріпленням, агент, унікальний лінгвістичний ідентифікатор формулювання, виявлення атак, адаптивність, масштабованість, живучість та ефективність прийняття колективних рішень.

ANNOTATION

Bohonko O.O. Methods and Means of Synthesis of Distributed Computer Systems Resistant to Social Engineering Attacks. – Qualification Scientific Work as a Manuscript.

Dissertation for the degree of Doctor of Philosophy in the specialty 123 – Computer Engineering. – Khmelnytskyi National University, Khmelnytskyi, 2026.

The study of known methods and tools revealed a systemic drawback – the available solutions demonstrate the low efficiency of synthesis of systems resistant to social engineering attacks, since they do not comprehensively include in the process of such synthesis of attack-resistant architecture, the reliability of providing attack detection, adaptability, scalability, survivability and efficiency of collective decision-making.

Thus, at this point in time, there is a contradiction between the need for the synthesis of computer systems resistant to social engineering attacks, on the one hand, and the imperfection of methods and means of ensuring the stability of the RCS in the face of social engineering attacks, on the other hand. Therefore, increasing the resilience of distributed computer systems to social engineering attacks is an urgent scientific and applied task, one of the ways to solve which is the development of methods and means of synthesis of distributed computer systems resistant to social engineering attacks.

This scientific and applied task corresponds to the subject area of the Standard of Higher Education of Ukraine in the specialty 123 – Computer Engineering for the third (educational and scientific) level of higher education, in particular, such an object of study and activity as "computer systems,..., computer networks, methods and methods of representing, receiving, storing, transmitting, processing and protecting information in them,..., architecture and organization of their functioning; information processes, technologies, methods, methods, tools and systems for research, design, adjustment, production and operation of computers and computer systems and networks, ..., ensuring quality, reliability and safety".

The object of the study is the process of synthesis of distributed computer systems resistant to social engineering attacks.

The subject of the study is models, methods and means of synthesis of distributed computer systems resistant to social engineering attacks.

The purpose of the study is to increase the resistance to social engineering attacks of distributed computer systems by developing methods and means of synthesis of social engineering RKS resistant to social engineering attacks, which comprehensively

ensure the reliability of attack detection, adaptability, scalability, survivability and efficiency of collective decision-making of RKS nodes.

To achieve this goal, the following tasks were solved:

- an analysis of known methods and means of ensuring the resilience of distributed computer systems to social engineering attacks has been carried out;

- a formal model of a distributed computer system resistant to social engineering attacks has been developed, which describes the collective behavior of agents in a dynamic environment through coherent decision-making, information exchange and adaptive resource management in order to maximize the global utility function, reflecting the resilience of the system to social engineering attacks, ensuring reliable threat detection, maintaining continuity of functioning, maintaining survivability under conditions of partial compromise of nodes and scaling of the system;

- an architecture of a distributed computer system based on a hierarchical multi-agent basis with the use of reinforcement learning has been developed, which makes it possible to adaptively reduce uncertainty in the process of detecting attacks and increase the accuracy of detection and classification of social engineering attacks;

- a method for detecting social engineering cyberattacks in distributed computer systems based on a unique linguistic formulation identifier has been developed, which is based on the formation of a specialized set of unique language identifiers, the use of the k-nearest neighbor method, which allows early detection of linguistic and semantic manipulations in scenarios of attacks on a distributed computer system;

- a method has been developed to ensure the scalability of the RCS architecture, resistant to social engineering attacks based on the population model of a multi-agent system and the middle field, which ensures the formation of an optimal policy of behavior of a representative agent, the integration of architectural parameters and guaranteed scalability of the system with an increase in the number of nodes and the intensity of attacks;

- a method of comprehensive assessment of the resistance of the RCS to social engineering attacks was developed, based on a multidimensional system of criteria for adaptability, scalability, survivability and reliability of detecting destructive

influences, with the formation of a generalized performance metric based on normalized weight factors;

- the architecture of the software implementation of a distributed computer system resistant to social engineering attacks was developed, which includes a decision-making agent, service agents, state monitoring components, an interaction manager and language/semantic analysis modules; experimental studies of its characteristics in scenarios of the impact of social engineering attacks have been carried out and the improvement of the system's resilience indicators has been evaluated.

In the dissertation, for the first time, a method was developed to ensure the scalability of the RCS architecture, resistant to social engineering attacks, which, in contrast to well-known approaches, combines the principles of dynamic decomposition, multi-agent interaction and adaptive redistribution of resources, taking into account the behavioral characteristics of users and threats, which makes it possible to ensure controlled scalability of the distributed system without reducing the level of security, to increase its survivability under the conditions of growth in the number of nodes of distributed CS and the intensity of social engineering attacks.

In the dissertation, for the first time, a method of comprehensive assessment of the resistance of the RCS to social engineering attacks was developed, which, unlike the known methods, is based on a multidimensional system of formalized criteria for adaptability, scalability, survivability and reliability of detection, which made it possible to obtain a single universal metric for assessing the resistance of the RCS to social engineering attacks.

In the dissertation , the architecture of an attack-resistant social engineering distributed computer system was further developed, which, unlike the well-known ones, is based on a hierarchical multi-agent basis with the use of reinforcement learning, entropy-oriented reward functions, a priori knowledge in the form of a knowledge graph and modal-specific service agents, which makes it possible to adaptively reduce uncertainty in the process of detecting attacks, reduce the number of dialog steps and improve the accuracy of detecting and classifying social engineering attacks.

The dissertation also improves the method for detecting social engineering cyberattacks in distributed computer systems based on a unique linguistic formulation identifier, which, unlike known approaches, is based on the formation of a specialized set of unique language identifiers, their preliminary linguistic normalization, expert labeling and the application of the k-nearest neighbor method, followed by adaptive adjustment of hyperparameters and trust thresholds, which makes it possible to increase the accuracy and resilience of detecting social engineering attacks, reduce the number of false positives, ensure early response and integration of results into the protection loops of a distributed computer system.

The practical value of the results obtained lies in the implementation of all theoretical provisions presented in the dissertation research into applied solutions and the possibilities of their direct implementation and use at enterprises.

According to the results of the research, the applicant implemented a distributed computer system resistant to social engineering attacks. The practical value of the work lies in the possibility of using the results obtained to develop corporate security policies, build simulation simulators to study user interaction with social engineering attacks, create intelligent agents for cyber defense, and optimize distributed system architectures taking into account risks. The proposed methods can be applied in the banking, telecommunications, energy and public spheres, where it is critically important to ensure the resilience of systems to complex behavioral threats.

The results of the dissertation work were implemented in (Annex B): PE "AVIVI" (act of implementation dated 08.01.2025); LLC "GM HOST" (act of implementation dated 30.12.2025); in the educational process of Khmelnytskyi National University (act of implementation dated 30.09.2025); in the implementation of the state budget theme of Khmelnytskyi National University "System for detecting RFPs and computer attacks in corporate networks using false attack objects and traps" (DR No. 0124U000980).

The main results of the dissertation were published in 9 scientific papers, including 5 articles in professional scientific journals of Ukraine, included as of the date of publication in the list of scientific professional publications of Ukraine of

category B; 4 publications certifying the approbation of the dissertation materials (articles in conference proceedings indexed in the scientometric database Scopus).

Keywords: distributed computer systems, social engineering, multi-agent system, synthesis of architecture, middle field; reinforcement learning, agent, unique linguistic formulation identifier, attack detection, adaptability, scalability, survivability and efficiency of collective decision-making.

Список публікацій здобувача за темою дисертації

Статті у наукових виданнях, включених до Переліку наукових фахових видань України:

1. Лисенко С., Атаманюк О., Бохонько О., Воробйов В. Дослідження методів виявлення кіберзагроз типу RANSOMWARE на основі застосування HONEYPOT. *Вісник ХНУ*. 2023. №1, (317). С. 300-309. https://doi.org/10.31891/2307-5732-2023-317-1-300-309

2. Лисенко С., Бохонько О. Методи виявлення кібератак соціальної інженерії. *Вісник ХНУ*. 2023. №*327*(5(2). С. 231-236. https://doi.org/10.31891/2307-5732-2023-327-5-231-236.

3. Бохонько О., Лисенко С. Моделі атак соціальної інженерії. *Measuring and computing devices in technological processes.* 2025. № (1), С. 432–444. https://doi.org/10.31891/2219-9365-2025-81-55 .

4. Бохонько О. Лисенко С. Метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії. *Measuring and computing devices in technological processes,* vol. 84(4), pp. 152–163. https://doi.org/10.31891/2219-9365-2025-84-16 .

5. Bokhonko O., Atamaniuk O. Method for synthesis of a scalable architecture of a distributed computer systems, resistant to social engineering attacks. Computer Systems and Information Technologies. 2025. Vol.4. pp. 60-76. https://doi.org/10.31891/csit-2025-4-7

Праці, які засвідчують апробацію матеріалів дисертації

6. Lysenko S., Bokhonko O., Savenko O., Vorobiov V., Gaj P., Wołoszyn J. Social Engineering Attacks Detection Approach. *Proceedings of 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2023, Athens, Greece, October 13-15, 2023). Pp. 318-329.

7. Lysenko S., Bokhonko O., Vorobiyov V., Gaj P. A Method for identifying cyberattacks based on the use of social engineering over the phone. *CEUR-WS.* 2024. Vol. 3675. Pp. 318-329. URL: https://ceur-ws.org/Vol-3675/paper23.pdf .

8. Lysenko S., Bokhonko O., Savenko O., Gaj P., Social Engineering Attacks Models. *Proceedings of 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2024, Athens, Greece, October 11-13, 2024).

9. Bokhonko O., Atamaniuk O., Sochor T. Model of a distributed heterogeneous system resistant to leakage of confidential information *CEUR-WS.* 2025. Vol. 3963. Pp. 363-376. URL: https://ceur-ws.org/Vol-3963/paper29.pdf .

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

КС – комп’ютерна система

РКС – розподілена комп’ютерна система

КМ – корпоративна мережа

МАС – мультааганетна система

АСІ – атака соціальної інженерії

СІ – соціальна інженерія

КА – комп’ютерна атака

БД – база даних

ІТ – інформаційна технологія

ІС – інформаційна система

ВСТУП

Актуальність теми. Соціальна інженерія (СІ) залишається одним із найнебезпечніших класів загроз для сучасних розподілених комп'ютерних систем (РКС), у яких події обробляються, ресурси надаються, а захист реалізується на великій кількості гетерогенних вузлів [1]. Збільшення кількості віддалених користувачів, сервісів, мультимодальних каналів взаємодії (електронна пошта, веб, голосові дзвінки, месенджери, соціальні мережі), а також поява гібридних хмарних середовищ призводить до різкого зростання розмірності простору станів РКС та складності їхньої поведінки. На цьому фоні атаки соціальної інженерії використовують психологічні вразливості користувачів, керують ланцюжками дій у часі та поєднують технічні й поведінкові вектори впливу, що перешкоджає стабільному функціонуванню РКС [2].

Існуючі рішення здебільшого зосереджуються на розробленні окремих детекторів конкретних сценаріїв атак (вішинг, фішинг, клонування профілю, шахрайські URL-адреси тощо) та локальних засобів моніторингу користувацької активності. Такі засоби підвищують точність виявлення на окремих каналах, однак залишаються «надбудовами» над уже побудованими системами і не змінюють сам спосіб формування РКС. У результаті захист має фрагментований характер: різні компоненти працюють із різними представленнями даних, відсутній цілісний механізм координації детекторів, а масштабування кількості вузлів і сервісів призводить до вибухового зростання обчислювальних витрат та неможливості забезпечити узгоджену реакцію на атаки.

Особливої гостроти проблема набуває для корпоративних РКС, де одночасно діють сотні й тисячі вузлів, об'єднаних складною топологією, з різними класами користувачів, сервісів і каналів обміну даними. У таких умовах навіть високоточні локальні детектори не гарантують стійкості системи в цілому: атака, що пройшла через один слабкий сегмент, здатна ініціювати каскадні ефекти, компрометацію конфіденційних даних і критичні порушення сервісів. Це зумовлює потребу переходу від локального проєктування окремих засобів виявлення до синтезу самих РКС як багаторівневих, багатоагентних архітектур, у

яких властивість стійкості до атак соціальної інженерії закладається вже на етапі проєктування.

Додатковим викликом є питання масштабованості, оскільки кількість вузлів у сучасних РКС постійно зростає і при цьому зростають кількість каналів взаємодії та шаблонів атак. Це призводить до експоненційного збільшення простору станів і дій, що робить неможливими як точні аналітичні розв'язки, так і пряме застосування методів навчання з підкріпленням без спеціальних засобів зниження розмірності. Тому від захисних механізмів вимагається не просто висока точність, а здатність зберігати якість роботи за зростання масштабів РКС без вибухового зростання обчислювальної складності.

Значний внесок в розробку моделей, методів та засобів синтезу розподілених комп'ютерних систем, стійких до кібератак зробили українські та іноземні вчені: О. С. Савенко [3], Р. Д. Капуста [4], Г. І. Кузьмин [5], О. І. Жмурко, [6], М. Б. Марчук [7], Д. Прокопович-Ткаченко [8], A. Naz [9], M. Alsharif [10], F. Osorio [11], D. Chapagain [12], V. Kolluri [13], A. Rawla [14], D. Komalasari [15], S. Ratra [16], M. Manguli [17], S. Ravula [18], D. Hapase [19], D. L. Moura [20], A. K. Alnaim [21]. тощо.

Незважаючи на велику кількість проведених наукових досліджень в сфері синтезу РКС, стійких до атак соціальної інженерії, на сьогодні ці результати не забезпечують комплексних рішень щодо критеріїв стійкості, достовірносі виявлення атак, адаптивності, масштабованості, живучості та ефективності прийняття колективних рішень.

Таким чином, на даний момент часу існує суперечність між потребою в синтезі комп'ютерних системи, стійких до атак соціальної інженерії, з одного боку, і недосконалістю методів та засобів забезпечення стійкості РКС в умовах атак соціальної інженерії, з іншого боку. Відтак, підвищення стійкості розподілених комп'ютерних систем до атак соціальної інженерії є *актуальною науково-прикладною задачею,* одним із шляхів розв'язання якої є розроблення методів і засобів синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії.

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, такому об’єкту вивчення та діяльності, як «комп’ютерні системи,…, комп'ютерні мережі, методи та способи подання, отримання, зберігання, передавання, опрацювання та захисту в них інформації,…, архітектура та організація їх функціонування; інформаційні процеси, технології, методи, способи, інструментальні засоби та системи для дослідження, проектування, налагодження, виробництва й експлуатації комп’ютерів та комп’ютерних систем і мереж, …, забезпечення якості, надійності та безпеки».

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, представлені в дисертації, проводились в рамках держбюджетної НДР Хмельницького національного університету 2Б-2024 “Система виявлення ЗПЗ та комп’ютерних атак в корпоративних мережах з використанням хибних об’єктів атак та пасток” (ДР №0124U000980) 2024-2025 рр., у якій автор був виконавцем і виконував апробацію напрацьованих методів та засобів виявлення ЗПЗ та комп'ютерних атак в розподілених КС.

Мета і завдання дослідження. *Об’єктом дослідження* є процес синтезу стійких до атак соціальної інженерії розподілених комп’ютерних систем.

Предметом дослідження є моделі, методи та засоби синтезу стійких до атак соціальної інженерії розподілених комп’ютерних систем.

Мета дослідження: Підвищення стійкості до атак соціальної інженерії розподілених комп’ютерних систем шляхом розроблення методів та засобів синтезу стійких до атак соціальної інженерії РКС, які комплексно забезпечують достовірність виявлення атак, адаптивність, масштабованості, живучість та ефективність прийняття колективних рішень вузлів РКС.

Для досягнення поставленої мети слід розв’язати такі *задачі:*

1. Провести аналіз відомих методів і засобів забезпечення стійкості розподілених комп’ютерних систем до атак соціальної інженерії.

2. Розробити формальну модель розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка описує колективну поведінку агентів у динамічному середовищі шляхом узгодженого прийняття рішень, обміну інформацією та адаптивного керування ресурсами з метою максимізації глобальної функції корисності, що відображає стійкість системи до атак

соціальної інженерії, забезпечення достовірного виявлення загроз, підтримання безперервності функціонування, збереження живучості за умов часткової компрометації вузлів і масштабування системи.

3. Розробити архітектуру стійкої до атак соціальної інженерії розподіленої комп'ютерної системи, яка базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак та підвищувати точність виявлення та класифікації атак соціальної інженерії.

4. Розробити метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, застосуванні методу k-найближчих сусідів, що уможливлює раннє виявлення мовних та семантичних маніпуляцій у сценаріях атак на розподіленої комп'ютерної системи.

5. Розробити метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії на основі популяційної моделі багатоагентної системи та середнього поля, що забезпечує формування оптимальної політики поведінки репрезентативного агента, інтеграцію архітектурних параметрів та гарантовану масштабованість системи при зростанні кількості вузлів і інтенсивності атак.

6. Розробити метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів.

7. Розробити архітектуру програмної реалізації розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, яка включає агента прийняття рішень, сервісних агентів, компоненти моніторингу станів, менеджер взаємодії та модулі мовного/семантичного аналізу; провести експериментальні дослідження її характеристик у сценаріях впливів атак соціальної інженерії та оцінити покращення показників стійкості системи.

Методи дослідження. При розв’язанні поставленої науково-прикладної задачі використовувались аналіз та синтез, методи аналізу та моделювання процесів, теоретико-множинні підходи, апарат модельно-орієнтованих підходів, принципи загальної теорії систем та системного аналізу, принципи побудови баз знань та формування логічного висновку, методи емпіричного дослідження, основні положення абстрактної алгебри, теорії розподілених систем, теорії елементів штучного інтелекту, теорія популяційних моделей та апарат середнього поля.

Наукова новизна одержаних результатів полягає в наступному:

вперше розроблено:

1) метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії, який на відміну від відомих підходів поєднує принципи динамічної декомпозиції, багатоагентної взаємодії та адаптивного перерозподілу ресурсів з урахуванням поведінкових характеристик користувачів і загроз, що дає змогу забезпечити керовану масштабованість розподіленої системи без зниження рівня захищеності, підвищити її живучість за умов зростання кількості вузлів розподіленої КС та інтенсивності атак соціальної інженерії;

2) метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, який на відміну від відомих методів ґрунтується на багатовимірній системі формалізованих критеріїв адаптивності, масштабованості, живучості та достовірності виявлення, що дозволило отримати єдину універсальну метрику оцінювання стійкості РКС до атак соціальної інженерії;

набула подальшого розвитку:

3) архітектуру стійкої до атак соціальної інженерії розподіленої комп’ютерної системи, яка на відміну від відомих базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, ентропійно-орієнтованими функціями винагороди, апріорними знаннями у вигляді графа знань та модально-специфічними сервісними агентами, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак, скорочувати кількість діалогових кроків і підвищувати точність виявлення та класифікації атак соціальної інженерії;

удосконалено:

4) метод виявлення кібератак соціальної інженерії в розподілених комп’ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який на відміну від відомих підходів ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, їх попередній лінгвістичній нормалізації, експертному маркуванні та застосуванні методу k-найближчих сусідів із подальшим адаптивним налаштуванням гіперпараметрів і порогових значень довіри, що дає змогу підвищити точність та стійкість виявлення атак соціальної інженерії, зменшити кількість хибних спрацьовувань, забезпечити раннє реагування та інтеграцію результатів у контури захисту розподіленої комп’ютерної системи.

Практичне значення одержаних результатів. Практична цінність отриманих результатів полягає в реалізації усіх теоретичних положень, поданих в дисертаційному дослідженні, у прикладні рішення та можливості їх безпосереднього впровадження й використання на підприємствах.

За результатами виконаних досліджень здобувачем реалізовано розподілену комп’ютерну систему, стійку до атак соціальної інженерії. Практична цінність роботи полягає у можливості використання отриманих результатів для розроблення корпоративних політик безпеки, побудови симуляційних тренажерів для дослідження взаємодії користувачів із атаками соціальної інженерії, створення інтелектуальних агентів для кіберзахисту та оптимізації архітектур розподілених систем з урахуванням ризиків. Запропоновані методи можуть застосовуватися у банківській, телекомунікаційній, енергетичній та державній сферах, де критично важливо забезпечити стійкість систем до складних поведінкових загроз.

Результати дисертаційної роботи впроваджено у (Додаток Б): ПП «АВІВІ» (акт впровадження від 08.1.2025 р.); ТОВ «ДЖІ ЕМ ХОСТ» (акт впровадження від 30.12.2025 р.); у навчальному процесі Хмельницького національного університету (акт впровадження від 30.09.2025 р.); при виконанні держбюджетної теми Хмельницького національного університету «Система виявлення ЗПЗ та комп'ютерних атак в корпоративних мережах з використанням хибних об'єктів атак та пасток» (ДР № 0124U000980).

Особистий внесок здобувача та внесок інших співавторів у спільних публікаціях. Усі наукові результати дисертаційного дослідження отримані автором особисто. Список опублікованих праць за темою дисертації представлено в списку використаних джерел – [1, 22, 173, 174, 176, 182, 183, 184, 189]. У спільних публікаціях автору належать такі результати: методи виявлення кібератак соціальної інженерії [1], дослідження методів виявлення кіберзагроз типу RANSOMWARE на основі застосування HONEYPOT [22], моделі атак соціальної інженерії [182,183], метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії [176], метод синтезу масштабованої архітектури розподілених комп'ютерних систем, стійкої до атак соціальної інженерії [189], метод виявлення атак соціальної інженерії [174], метод виявлення кібератак на основі використання соціальної інженерії під час телефонних розмов [184], модель розподіленої гетерогенної системи, стійкої до витоку конфіденційної інформації [173].

Особистий внесок інших співавторів у спільних публікаціях: у статті [22] С. Лисенко виконував адміністрування та концептуалізацію дослідження, О. Атаманюк, В. Воробйов, сумісно з автором та під його керівництвом працювали над оглядом існуючих методів та рішень; у статті [1] С. Лисенко виконував адміністрування та концептуалізацію дослідження, рецензування та коригування рукопису; у статті [173] С. Лисенко виконував адміністрування та концептуалізацію дослідження; у статті [176] С. Лисенко виконував адміністрування та концептуалізацію дослідження; обговорення результатів дослідження, рецензування та коригування рукопису, керівництво проєктом; у статті [182] С. Лисенко виконував адміністрування та концептуалізацію дослідження; обговорення результатів дослідження, рецензування та коригування рукопису, О. Атаманюк, спільно з автором проводили експерименти; у статті [174] С. Лисенко виконував адміністрування та концептуалізацію дослідження; О. Савенко, В. Воробйов, П. Гай, Ю. Волошин, спільно з автором проводили експерименти, обговорення результатів дослідження; у статті [184] С. Лисенко виконував адміністрування та концептуалізацію дослідження; В. Воробйов, П. Гай, спільно з автором проводили експерименти; у статті [183] С. Лисенко виконував адміністрування

та концептуалізацію дослідження; О. Савенко, П. Гай, спільно з автором проводили експерименти, рецензування та коригування рукопису; у статті [173] С. Лисенко виконував адміністрування та концептуалізацію дослідження; Атаманюк О., Сохор Т., спільно з автором проводили експерименти, рецензування та коригування рукопису, обговорення результатів дослідження.

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 доповідались на таких конференціях: 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2023), 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2024), The 5th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS 2024), The 6th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS 2025), XV Всеукраїнська науково-практична конференція "Актуальні проблеми комп'ютерних наук (АПКН – 2023)", XVI Всеукраїнська науково-практична конференція "Актуальні проблеми комп'ютерних наук (АПКН – 2024)", Стан, досягнення та перспективи інформаційних систем і технологій, 2024 р.

Публікації. Основні результати дисертації опубліковані у 9 наукових працях ([1, 22, 173, 174, 176, 182, 183, 184, 189] додаток А), серед яких 5 статей у фахових наукових журналах України [1, 22, 176, 182, 189], включених на дату опублікування до переліку наукових фахових видань України категорії Б; 4 публікації, які засвідчують апробацію матеріалів дисертації (статті в матеріалах конференцій, що індексуються в наукометричній базі Scopus) [173, 174, 183, 184].

Структура та обсяг дисертації. Дисертація складається з анотації, змісту, переліку умовних скорочень, вступу, чотирьох розділів, висновків, списку використаних джерел із 190 найменувань на 23 сторінках та 4 додатків на 42 сторінках. Загальний обсяг дисертаційної роботи становить 241 сторінка друкованого тексту, з них 157 сторінок основного тексту. Дисертація містить 7 рисунків та 16 таблиць.

РОЗДІЛ 1
АНАЛІЗ ВІДОМИХ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ СТІЙКОСТІ КОМП'ЮТЕРНИХ СИСТЕМ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

1.1 Поняття атаки соціальної інженерії та їх вплив на розподілені комп'ютерні системи

Атаки соціальної інженерії – це навмисні дії з маніпулювання користувачами з метою отримання конфіденційної інформації або несанкціонованого доступу до комп'ютерних систем. Розглянемо найбільш відомі типи кібератаки соціальної інженерії [1].

Атака типу голосове шахрайство (vishing) – сукупність дій зловмисників, спрямованих на отримання конфіденційної інформації від користувачів шляхом здійснення телефонних дзвінків та імітації легітимної комунікації.

Атака типу мережеве фішингове шахрайство (phishing) – сукупність дій зловмисників, спрямованих на масове розповсюдження повідомлень, що містять модифіковану або підроблену інформацію, яка візуально і структурно імітує офіційні та легітимні джерела.

Атака типу формування довірчих відносин (grooming) – сукупність дій зловмисників, спрямованих на поєднання інформаційних технологій (служби коротких повідомлень, електронна пошта, телефонний зв'язок, соціальні мережі) із психологічними методами з метою поступового встановлення довіри та отримання інформації про потенційних жертв.

Атака типу клонування профілю користувача (profile cloning) – сукупність дій зловмисників, спрямованих на використання відкритих даних соціальних мереж для створення копії особистого профілю користувача з метою введення в оману третіх осіб.

Атака типу пошук у сміттєвих ресурсах (dumpster diving) – сукупність дій зловмисників, спрямованих на збір та аналіз документів або інформації, видаленої до електронного кошика чи фізично утилізованої користувачем.

Атака типу несанкціоноване фізичне супроводження (tailgating) – сукупність дій зловмисників, спрямованих на отримання фізичного доступу до захищених приміщень шляхом проходу разом із легітимним користувачем без відповідної автентифікації.

Атака типу маскування файлів (file masquerade) – сукупність дій зловмисників, спрямованих на приховування шкідливого програмного забезпечення під виглядом легітимних файлів, унаслідок чого користувач запускає шкідливий код під час роботи зі звичайними даними.

Атака типу приманювання (baiting) – сукупність дій зловмисників, спрямованих на використання привабливих пропозицій або фізичних і цифрових носіїв інформації (зовнішні накопичувачі, файли, посилання) з метою спонукання користувача до виконання шкідливих дій.

Атака типу залякувальне програмне забезпечення (scareware) із використанням спливаючих вікон (pop-up windows) – сукупність дій зловмисників, спрямованих на психологічний тиск через демонстрацію фальшивих попереджень безпеки для примусу користувача до встановлення або запуску шкідливого програмного забезпечення.

Атака типу компрометація популярних вебресурсів (water-holing) – сукупність дій зловмисників, спрямованих на злам вебсайтів із високим рівнем відвідуваності з подальшим упровадженням шкідливого коду для зараження цільових груп користувачів.

Атака типу троянська електронна пошта (trojan mail) – сукупність дій зловмисників, спрямованих на поширення шкідливого програмного забезпечення через електронні листи у вигляді зовні безпечних вкладень або повідомлень.

Атака типу цільове фішингове шахрайство (spear phishing) – сукупність дій зловмисників, спрямованих на викрадення конфіденційної інформації у конкретної жертви шляхом попереднього збору персональних даних і маскування атаки під легітимну персоналізовану комунікацію.

Атака типу небажана масова розсилка (spam) – сукупність дій зловмисників, спрямованих на масове надсилання електронних повідомлень із рекламним або потенційно шкідливим вмістом з метою компрометації систем користувачів.

Атака типу підроблене програмне забезпечення з привабливою назвою (curiosity software attack) – сукупність дій зловмисників, спрямованих на аналіз популярних програмних продуктів і розміщення фальшивих посилань для зараження комп'ютерних систем користувачів.

Атака типу створення правдоподібного сценарію (pretexting) – сукупність дій зловмисників, спрямованих на формування вигаданої, але логічно обґрунтованої ситуації, у межах якої жертва добровільно розкриває конфіденційну інформацію.

Атака типу безпосередня особиста взаємодія (face-to-face interaction) – сукупність дій зловмисників, спрямованих на пряме спілкування з потенційною жертвою для виявлення її психологічних характеристик та подальшої маніпуляції.

Атака типу візуальне підглядання (shoulder surfing) – сукупність дій зловмисників, спрямованих на отримання конфіденційної інформації шляхом спостереження за введенням паролів, кодів доступу або інших чутливих даних.

Атака типу взаємообмін послугами (quid pro quo) – сукупність дій зловмисників, спрямованих на отримання конфіденційної інформації в обмін на надання фіктивної допомоги або умовної послуги.

Атака типу перенаправлення викрадення (diversion theft) – сукупність дій зловмисників, спрямованих на зміну або підміну адрес доставки товарів у цифрових сервісах з метою незаконного заволодіння матеріальними цінностями.

Атака типу імітація легітимного доступу (piggybacking), включаючи несанкціоноване супроводження (tailgating), імітацію присутності (trailing) та вдавання користувача (pretending) – сукупність дій зловмисників, спрямованих на отримання фізичного або логічного доступу без належної авторизації.

Атака типу зворотна соціальна інженерія (reverse social engineering) – сукупність дій зловмисників, спрямованих на штучне створення проблемної ситуації у системі жертви з метою спонукання користувача самостійно звернутися до зловмисника за «допомогою».

Атака типу імітація технічного експерта (technical expert impersonation) – сукупність дій зловмисників, у межах яких здійснюється видавання себе за спеціалістів служби технічної підтримки з метою отримання доступу до конфіденційних даних.

Атака типу імітація персоналу підтримки (support staff impersonation) – сукупність дій зловмисників, спрямованих на симуляцію взаємодії зі службою підтримки організації через різні канали зв'язку для викрадення конфіденційної інформації.

Атака типу аналіз слідів на сенсорних екранах (smudge attack) – сукупність дій зловмисників, спрямованих на дослідження залишкових слідів дотиків з метою відновлення паролів, персональних ідентифікаційних номерів та графічних ключів доступу.

Сучасні розподілені комп'ютерні системи (РКС) характеризуються високою динамічністю та складністю взаємодії між компонентами. Вони включають розподілені обчислювальні платформи, сервіси, мікросервісні структури, мобільні та вебзастосунки, хмарні середовища, контейнеризовані компоненти та системи маршрутизації подій [1]. Кожний компонент генерує або інтерпретує події, що формують глобальний простір сервісної взаємодії, у межах якого рішення користувачів є ключовим фактором зміни стану системи [22].

Подія в РКС є базовою одиницею реакції системи й може бути створена сервісами, автоматизованими механізмами або користувачами. У середовищах з інтенсивною сервісною взаємодією користувачі ініціюють транзакції, підтверджують запити доступу, змінюють параметри сесій та впливають на маршрутизацію даних [23]. Таким чином, користувач стає внутрішнім компонентом РКС, що визначає її соціотехнічну природу та вразливість до маніпулятивних дій. Фрагментація логіки обробки подій у розподілених

системах посилює ризик того, що подія, ініційована під впливом соціальної інженерії, виглядатиме технічно коректною, але спричинить стратегічно небажані наслідки. За відсутності централізованої моделі інтерпретації намірів система обробляє подію виключно за формальними правилами, навіть якщо вона суперечить очікуваній поведінці або порушує цілісність процесів [24].

Зловмисники впливають на життєвий цикл подій РКС, створюючи повідомлення чи запити, що за формою нагадують штатні службові взаємодії й легко проходять технічні перевірки. Після взаємодії з такими елементами користувач фактично змінює стан системи – підтверджує транзакцію, відкриває доступ або активує сервіс. Найнебезпечнішими є каскадні події, які одночасно зачіпають кілька модулів, ускладнюючи їх виявлення. Подібна логіка характерна також для складних шкідливих сценаріїв, зокрема ransomware, де початкова дія користувача запускає повний ланцюг деструктивної активності, що підтверджено експериментальними дослідженнями в honeypot-середовищах [25].

Розширення мобільних застосунків, вебінтерфейсів і автоматизованих службових сповіщень збільшує кількість каналів, у яких важко відрізнити легітимні сигнали від маніпулятивних. Поява систем автоматичного генерування контенту, здатних імітувати службовий стиль [26], додатково знижує ефективність сигнатурних та поверхневих методів контролю.Сприйнятливість користувачів до маніпуляцій зростає в умовах інформаційного перевантаження. У РКС користувач обробляє десятки запитів, підтверджень та системних повідомлень, що формує рутинний стиль взаємодії та зменшує якість перевірки кожного сигналу [27]. Водночас РКС зазвичай не містять інтегрованої поведінкової моделі користувача: окремі сервіси не мають повного уявлення про історію його дій чи типові сценарії, тому будь-яка формально допустима дія вважається коректною.

Додатковою уразливістю є нерівність можливостей: зловмисник підлаштовується під поведінку користувача, тоді як система не враховує контекст під час перевірок. Нечіткі механізми підтвердження транзакцій,

відсутність багаторівневої сегментації та нерівномірність організаційних процесів лише посилюють ризики [28]. У малих і середніх організаціях недостатня формалізація процедур призводить до того, що навіть одинична помилка користувача може спричинити поширення небажаних подій на модулі РКС [29].

Ефективні інженерні рішення мають виходити за межі традиційних технічних фільтрів та включати аналіз поведінкових характеристик користувачів, контексту взаємодій, сценаріїв зміни стану та ролей компонентів у системі [23], [28], [30]. Саме інтегрований підхід до аналізу подій і поведінки визначає як напрям подальшого моделювання, так і можливості створення розподілених комп'ютерних систем, стійких до атак соціальної інженерії.

1.2 Канали розподілених комп'ютерних систем, що використовуються для реалізації атак соціальної інженерії

Сучасні розподілені комп'ютерні системи складаються з множини взаємодіючих сервісів: вебкомпонентів, мобільних модулів, транзакційних систем, голосових платформ і децентралізованих сервісів, в яких здійснюються події різних типів. Кожен канал формує створює умови для ймовірної реалізації маніпулятивних сценаріїв. Взаємодії відбуваються паралельно в декількох каналах, що ускладнює контроль та сприяє багатокроковим впливам.

Вебканал охоплює вебшлюзи, мережі доставки контенту, системи доменних імен, механізми авторизації з відкритим протоколом доступу та інші вузли, через які циркулюють заголовки протоколу передавання гіпертексту, маркери автентифікації, події мови сценаріїв JavaScript та параметри транспортного рівня безпеки. Маніпуляції на цьому рівні реалізуються шляхом створення підроблених вебсторінок, перехоплення маркерів сесій та перенаправлення користувачів на спеціально сконструйовані інтерфейси. Згідно з результатами досліджень, вебканал залишається одним із основних джерел

інцидентів через високу інтенсивність взаємодії користувачів із динамічними сервісними середовищами [31, 32].

Другим ключовим каналом є поштовий. Його інженерна специфіка полягає у структурованості багатоцільових розширень інтернет-пошти, вкладених об’єктах, криптографічних підписах ідентифікованої пошти домену та політиках фреймворку перевірки відправника і політики автентифікації повідомлень, звітності та відповідності домену. Шлюзи протоколу простого передавання пошти та модулі фільтрації небажаних повідомлень формують перший рубіж аналізу, однак значна частина шкідливих сценаріїв виникає саме через взаємодію користувачів із листами. Електронна пошта часто виступає початковою фазою складних міжканальних атак, у яких подальші етапи реалізуються у веб або мобільних каналах [33].

Мобільні та месенджерні канали формують іншу модальність подій, зокрема пакети одиниць даних служби коротких повідомлень, маркери push-сповіщень, мобільні запити протоколу передавання гіпертексту, тригери швидкого реагування-кодів та інтеграції з прикладними програмними інтерфейсами мобільних платформ. Інфраструктура цих каналів є критичними точками потенційного впливу. У 2024–2025 роках спостерігалося суттєве зростання мобільних атак, зокрема сценаріїв фішингу через мобільні повідомлення, маніпуляцій багатофакторною автентифікацією та підроблених запитів push-підтвердження [34].

Голосові канали, що базуються на протоколі ініціації сеансів, голосовому зв’язку через інтернет та інтерактивних голосових меню, формують події іншої технічної природи: заголовки сигналізаційних протоколів, аудіопотоки протоколу передавання реального часу та взаємодії з автоматизованими голосовими механізмами. У межах цього каналу поширеними є підміна ідентифікатора абонента, модифікація сигнального трафіку та сценарії, що поєднують голосову взаємодію з підтвердженням фінансових транзакцій або введенням одноразових кодів доступу. Голосові впливи часто виступають завершальною фазою складних мультимодальних атак [32, 35].

Транзакційні канали прикладних програмних інтерфейсів мають особливе значення для розподілених комп'ютерних систем, оскільки через них користувач може ініціювати формально легітимні, але небажані операції. Події у форматі обміну об'єктами JavaScript, маркери авторизації, параметри платіжних сесій та системи протидії шахрайству формують функціональну основу цього каналу. У багатьох випадках порушення безпеки виникає не внаслідок прямого технічного злому, а через те, що користувач самостійно виконує дію, ініційовану маніпулятивним інформаційним впливом [36].

У децентралізованих середовищах третього покоління вебтехнологій канали взаємодії включають вузли віддаленого виклику процедур, інтерфейси криптографічних гаманців та механізми цифрового підписання транзакцій. Тут модальність подій визначається підписаними транзакціями, викликами інтерфейсу бінарного застосування та сигналами у ланцюгах блоків. Через незворотність фінансових операцій і складність користувацьких інтерфейсів цей канал є особливо вразливим до впливів атак соціальної інженерії [37].

Узагальнена класифікація таких каналів наведена у таблиці В.1. Додатку В. Вона відображає інженерну структуру точок взаємодії, типи інформаційних потоків та технічні вектори, що формують поверхню атаки у розподілених комп'ютерних системах. Після систематизації каналів важливо оцінити класи технічних впливів, які вони генерують. Оскільки РКС функціонує як подійно-орієнтована система, кожен канал породжує події певного типу, що далі трансформуються у конкретні схеми маніпуляції. До найпоширеніших належать отримання облікових даних, ініціація транзакцій, маніпуляції конфігураціями та установлення компонентів. Вони відрізняються за складністю аналізу, можливими наслідками та ступенем автоматизації атак.

Структурована класифікація таких впливів наведена у таблиці В.2 Додатку В. Вона демонструє взаємозв'язок між технічною природою сигналів (подіями протоколу передавання гіпертексту, маркерами автентифікації, заголовками протоколу ініціації сеансів, запитами у форматі обміну об'єктами

JavaScript, транзакціями у ланцюгах блоків) та інженерними ризиками, що виникають у розподілених комп’ютерних системах.

Багатоканальний характер сучасних атак СІ зумовлює необхідність комплексної інженерної оцінки подій. У більшості сценаріїв відбувається перехід між каналами (електронна пошта – веб – мобільний – голос – транзакційний), що експоненційно збільшує кількість подій і складність їх кореляції. З цієї причини РКС мають підтримувати потокову обробку подій, мультимодальність, уніфіковані журнали, синхронізацію часових міток і можливість реконструкції повного ланцюга взаємодій [31, 32, 36].

Слід зазначити, що канали РКС визначають технічну основу взаємодії користувача із системою та формують можливості для реалізації соціальних атак. Їхні властивості, зокрема, типи даних, модальності подій, затримки, точки маршрутизації, визначають вимоги до синтезу стійких розподілених комп’ютерних систем.

1.3 Аналіз відомих методів для створення РКС, стійких до атак соціальній інженерії

Поряд зі зростанням масштабів і сфер застосування РКС підвищується ймовірність виникнення складних сценаріїв небажаних впливів, реалізованих засобами атак соціальної інженерії. Сьогодні існує велика кількість методів виявлення атак соціальній інженерії, зокрема, у [37] авторами запропоновано конвеєр виявлення фішингу на основі URL-адрес, який базується на гібридному навчанні з використанням кількох класичних класифікаторів. Підхід демонструє високу ефективність при незначних обчислювальних витратах, що дозволяє інтегрувати компактні моделі в розподілені шлюзи. Проте недоліком дослідження є фокус виключно на метриках точності без системного аналізу стійкості архітектури до адаптивних атак та маніпуляцій з ознаками.

У дослідженні [38] розглядаються фішингові кампанії, розгорнуті на безкоштовних конструкторах вебсайтів, з акцентом на аналізі ланцюга

постачання атак. Такий метод дозволяє перейти від вузького технічного аналізу до координації захисту на рівні розподілених систем і телеметрії. Водночас у роботі не запропоновано уніфікованого методу оцінки ефективності взаємодії захисників в умовах динамічної зміни стратегій ухилення.

У [39] проведено порівняльне дослідження клієнтських ресурсів фішингових і легітимних сторінок. Запропоноване використання «відбитків ресурсів» забезпечує масштабовану фільтрацію на стороні клієнта в реальному часі. Недоліком є відсутність кількісної оцінки живучості системи та аналізу якості колективного прийняття рішень.

У статті [40] представлено концепцію людиноцентрованого захисту (HUCITE), що базується на персоналізованих периферійних обчисленнях. Метод враховує контекст користувача, що є критичним для протидії методам соціальної інженерії. Однак персоналізація створює труднощі для оцінки стабільності та адаптивності розподіленої системи захисту за межами стандартних показників точності.

У дослідженні [41] презентовано систему MLPhishChain, яка поєднує методи машинного навчання для аналізу URL-адрес із технологією блокчейн для реєстрації вердиктів. Це забезпечує прозорість та можливість аудиту рішень у розподілених мережах. Проте залишаються невирішеними питання потенційних затримок системи за умов часткової компрометації вузлів.

У роботі [42] запропоновано метод динамічного об’єднання ознак, що комбінує глобальні структури графів із локальними семантичними сигналами для захисту в блокчейн-мережах. Підхід дозволяє ідентифікувати складні координовані схеми шахрайства, проте він не повністю формалізує стійкість архітектури при отриманні неповних даних від окремих агентів або отруєнні інформаційного графа.

У [43] розроблено механізм FedPhishLLM, який реалізує федеративне навчання на основі великих мовних моделей. Рішення забезпечує конфіденційність даних і довіру операторів через пояснюваність вердиктів. Недоліком є вразливість до ризиків федеративного середовища, таких як дрейф моделей та недостатня архітектурна стійкість до ворожих учасників.

В оглядовій роботі [44] систематизовано сучасні методи захисту від смішингу, включаючи аналіз контенту та мережевих сигналів. Автори консолідують фрагментовані знання в цій галузі та виокремлюють практичні обмеження, що виникають при розгортанні таких засобів у РКС.

У дослідженні [45] запропоновано повністю клієнтський підхід до виявлення фішингу, що базується на аналізі мовних сигналів сторінки. Це зміцнює живучість системи через автономність агентів, проте відсутність механізмів агрегації та розв'язання конфліктів між клієнтами ускладнює оцінку цілісної ефективності архітектури.

У статті [46] висвітлено метод, який фокусується на логічному виведенні намірів фішингових атак. Моделювання намірів підвищує адаптивність до нових сценаріїв соціальної інженерії. Проте впровадження багатьох взаємодіючих агентів на основі LLM потребує ретельної оцінки операційної стабільності та часових затримок.

У праці [47] автори аналізують еволюцію конфігурацій фішингової інфраструктури в часі. Дослідження дозволяє вдосконалити стратегії блокування, проте воно не містить методу оцінки синтезованих розподілених систем захисту щодо їхньої опірності системним збоям.

У дослідженні [48] наголошується на важливості інтерпретованого машинного навчання для підвищення підзвітності систем виявлення фішингу. Такий підхід сприяє колективному прийняттю рішень через зрозумілість вихідних даних для агентів і людей. Недоліком є відсутність уніфікованих метрик, які б поєднували пояснюваність із живучістю та масштабованістю архітектури.

У статті [49] розглядаються методи вдосконалення виявлення смішингу з акцентом на мінімізацію хибнопозитивних спрацювань. Автори підкреслюють важливість операційної ефективності, проте не оцінюють повною мірою складність міжрівневої координації в умовах адаптації зловмисників.

У роботі [50] продемонстровано методи приховування фішингу від чорних списків. Результати доводять необхідність мультиагентного підтвердження атак,

хоча в роботі не запропоновано методології для розрахунку необхідної кількості незалежних агентів для підтримання живучості системи.

У роботі [51] досліджуються людиноцентровані техніки маскування фішингових сайтів, що обґрунтовує потребу в архітектурах із різноманітними точками зондування. Водночас робота не пропонує метрик для квантифікації стабільності системи за умов цілеспрямованого маскування загроз.

У дослідженні [52] аналізуються транзакційні мережі Ethereum за допомогою графового навчання для виявлення координованих шахрайств. Підхід ефективний для виявлення розподілених патернів, проте оцінка не враховує сценаріїв із деградованою видимістю графа або змовою вузлів.

У роботі [53] вивчається виявлення фішингу в Ethereum через часову еволюцію графів. Це дозволяє впроваджувати адаптивний моніторинг, але зв’язок між продуктивністю моделей і загальною стійкістю системи при атаках на інфраструктуру моніторингу залишається недостатньо вивченим.

У статті [54] проводиться огляд евристичних методів захисту веб-каналів. Робота стимулює синтез багатосенсорних систем захисту, проте не містить формалізованих методів порівняння різних варіантів розподіленого дизайну.

В роботі [55] систематизовано сучасні методи ідентифікації шкідливих URL-адрес, підкреслюючи проблеми генералізації та протидії зловмисному впливу. Дослідження допомагає позиціонувати виявлення фішингу як складну розподілену задачу, проте не надає наскрізних мір ефективності для систем із частковими відмовами.

У дослідженні [56] аналізуються тренди загроз у реальних умовах, що підтверджує швидке зростання багатоканальних атак. Це обґрунтовує необхідність масштабованості розподілених систем захисту, виходячи за межі стандартних лабораторних сценаріїв.

У статті [57] представлено огляд методів виявлення шкідливих URL-адрес, де особливу увагу приділено застосуванню квантового машинного навчання як перспективного напряму захисту пристроїв інтернету речей. Дослідження [58] описує метод класифікації фішингових URL-адрес за допомогою

детермінованих і ймовірнісних нейронних мереж, який продемонстрував точність 97% на комбінованих наборах відкритих та приватних даних. У роботі [59] запропоновано метод EGSO–CNN, який завдяки інтеграції варіаційних автоенкодерів та покращеного пошуку за сіткою дозволяє досягати точності 99,44% при розпізнаванні підроблених вебсайтів.

У дослідженні [60] реалізовано комбінований метод на базі GAN-CNN-LSTM та алгоритмів ройового інтелекту, що забезпечує інтелектуальний відбір ознак для підвищення ефективності детекції фішингу. Автори статті [61] фокусуються на виявленні адверсаріальних атак, використовуючи ансамблі алгоритмів машинного навчання та відбір найбільш інформативних характеристик за допомогою регуляризації Lasso. У роботі [62] представлено метод DeepEPhishNet, де для виявлення фішингових електронних листів застосовано алгоритми вбудовування слів (Word2Vec, FastText) та архітектури BiLSTM, що забезпечило точність понад 99,5%.

У статті [63] розроблено систему детекції на основі трансформерних вбудовувань слів та пошуку векторної подібності, що автоматизує витягування ознак і підвищує швидкість реагування на загрози. Дослідження [64] пропонує ансамблевий метод PDSMV3-DCRNN, який використовує динамічний відбір ознак та генеративні змагальні мережі для подолання дисбалансу даних. У праці [65] акцентовано увагу на проблемі генералізації моделей, де автори пропонують фреймворк на основі некерованої адаптації домену для підвищення переносності моделей між різними наборами даних.

Аналіз у роботі [66] за допомогою методів пояснювального ШІ (XAI) доводить, що ознаки фішингових URL-адрес часто є специфічними для конкретних вибірок, що ставить під сумнів їхню надійність у динамічних середовищах. У дослідженні [67] проведено порівняльний аналіз повнозв’язних нейронних мереж (FCNN) на сучасних датасетах для створення зручного інтерфейсу моніторингу атак. У статті [68] запропоновано метод виявлення аномалій на основі архітектури CapsNet із механізмом самоуваги, що дозволяє ефективно враховувати просторові зв'язки між ознаками URL-адреси.

У дослідженні [69] проаналізовано методи детекції фішингових доменів за допомогою алгоритмів керованого навчання, де найвищу точність продемонстрував метод градієнтного бустингу при оцінці ризиків на основі даних SSL-сертифікатів. У статті [70] запропоновано фреймворк HSSLC-CharGRU, який використовує гібридну просторово-послідовну увагу та логічні обмеження нейронної мережі для покращення генералізації при аналізі складних URL-патернів. Робота [71] описує підхід до виявлення шкідливих вебсайтів за допомогою алгоритму XGBoost, параметри якого оптимізовано за допомогою адаптивного методу рою кажанів (Bat Algorithm).

У дослідженні [72] представлено метод на основі одновимірних згорткових нейронних мереж (1D CNN), що дозволив досягти точності 99,7% при розпізнаванні дубльованих веб-ресурсів. Автори статті [73] пропонують метод раннього виявлення загрози у припаркованих доменах фінансових установ, застосовуючи класифікатор LightGBM для моніторингу нових реєстрацій. У роботі [74] продемонстровано переваги глибоких нейронних мереж (DNN) над традиційними методами машинного навчання, що забезпечило підвищення точності детекції до 99,43%.

Особлива увага приділяється блокчейн-платформам: у статті [75] описано метод DA-HGNN для виявлення фішингу в мережі Ethereum, що інтегрує часові ознаки та структурні властивості графів транзакцій. Дослідження [76] базується на використанні графових нейронних мереж (GAE_PDNA) та класифікатора AdaBoost для ідентифікації скомпрометованих акаунтів. У праці [77] представлено фреймворк GraPhish, який дозволяє виявляти атаки в зашифрованому TLS-трафіку шляхом побудови графів на основі ознак протоколу захищеного з'єднання.

У статті [78] запропоновано метод BiLSTM4DPS, що поєднує двонаправлені рекурентні мережі з механізмами мультиголової уваги для аналізу часових послідовностей транзакцій. У дослідженні [79] представлено динамічну каскадну графову модель DMPCG, яка інтегрує статичні знімки мережі та еволюцію торгових операцій для розкриття складних схем децепції.

У статті [80] представлено метод EPAD, який використовує контрастивне навчання на графах для самокерованого виявлення фішингу в мережі Ethereum, що дозволяє ефективно ідентифікувати зловмисні акаунти без залучення розмічених даних. Дослідження [81] описує модель DAOA-DLPC для хмарних середовищ, де завдяки символьному вбудовуванню (character-level embeddings) та динамічній оптимізації гіперпараметрів досягнуто високої точності класифікації шкідливих URL-адрес. У роботі [82] обґрунтовано ефективність архітектури BERT для розпізнавання смішингу, де використання механізмів самоуваги дозволило значно покращити контекстуальний аналіз текстових повідомлень.

У дослідженні [83] запропоновано інтегрований фреймворк CNN–LSTM для захисту індустріальних IoT-мереж, підсилений методами пояснювального ШІ та великими мовними моделями (LLM) для забезпечення прозорості прийняття рішень. Автори статті [84] розробили стійку до шахрайства структуру для телекомунікаційних мереж, застосовуючи метод PCC-PCA для вилучення ознак та покращену згорткову нейронну мережу для фільтрації спаму. У роботі [85] представлено метод WebPhish, який реалізує наскрізне навчання на основі сирих URL та HTML-контенту, автоматизуючи процес виявлення семантичних залежностей без ручного проектування ознак.

У статті [86] описано покращену модель CNN з глибокою ієрархією прихованих шарів, що забезпечує точність 99,95% при детекції смішингу на мобільних пристроях. Дослідження [87] фокусується на виявленні шкідливих повідомлень, використовуючи гібридну модель на базі BERT та символьних CNN для багатокласової класифікації. У праці [88] представлено метод підвищення ефективності детекції атак соціальної інженерії на незбалансованих наборах даних шляхом застосування техніки оверсемплінгу SMOTE-ENN спільно з опорними векторами (SVM). Нарешті, у дослідженні [89] запропоновано гібридний метод URLGuard, що поєднує дерева рішень та градієнтний бустинг для комплексного захисту від фішингових URL-адрес.

У дослідженні [90] представлено метод розширення наборів даних для виявлення голосового фішингу (вішингу) за допомогою багатомовного зворотного перекладу, що дозволило значно покращити показники F1-score порівняно з традиційними методами оверсемплінгу. У статті [91] запропоновано фреймворк на основі великої мовної моделі GPT-4o для генерації реалістичних транскриптів дзвінків та створення експертної системи детекції, яка враховує специфічні критерії аналізу шахрайських тактик. Робота [92] описує мультимодальну систему, що інтегрує аналіз тексту (KoBERT) та аудіосигналів (CNN–BiLSTM) для ідентифікації синтезованого мовлення, забезпечуючи надійний захист у реальних шумових умовах.

Для захисту мобільних пристроїв у дослідженні [93] розроблено систему VishielDroid, яка здійснює динамічне відстеження запитів на отримання дозволів у реальному часі, демонструючи високу стійкість до спроб ухилення від детекції. Автори статті [94] надають комплексний огляд та дорожню карту розвитку технологій мовлення для протидії соціальній інженерії, підкреслюючи критичний дефіцит доступних даних та необхідність розробки інструментів для виявлення психологічних стратегій переконання. У праці [95] представлено метод детекції корейського вішингу з використанням розпізнавання іменованих сутностей (NER) та N-грам на рівні речень для розрізнення фішингових розмов і легітимних фінансових консультацій.

У статті [96] запропоновано метод FDN-SA на основі нечітких нейронних мереж та стекових автоенкодерів, який дозволяє ефективно вилучати лінгвістичні ознаки та виконувати веб-детекцію атак. Дослідження [97] присвячене аналізу електронних листів, згенерованих штучним інтелектом, де доведено, що методи машинного навчання здатні ідентифікувати специфічні стилістичні відмінності між ШІ-контентом та повідомленнями, створеними людьми. У роботі [98] представлено гібридний фреймворк для автоматизованого виявлення загроз, який поєднує декілька моделей аналізу для підвищення стійкості до спроб обходу систем захисту.

У дослідженні [99] представлено метод PDHF, який поєднує оптимальні штучні ознаки з автоматично вилученими ознаками глибокого навчання (CNN) та механізмом уваги, що дозволило досягти точності 99,65% при аналізі URL-адрес. У статті [100] запропоновано метод самокерованого навчання EPAD на основі контрастивного навчання на графах, який вирішує проблему дефіциту розмічених даних у мережі Ethereum шляхом порівняння вузлів із їхніми локальними підграфами. Робота [101] описує двофазний фреймворк OptSHQCNN, що інтегрує квантову класичну згорткову мережу та OptBERT для виявлення фішингових сайтів через аналіз коду сторінок та поПеревага нь.

Для захисту зашифрованого трафіку у статті [102] розроблено метод відбитків TLS (TLS fingerprinting) на основі атрибутивних графів, що дозволяє виявляти шкідливу активність без дешифрування даних. У дослідженні [103] обґрунтовано використання машинного навчання для захисту споживчих застосунків у сфері охорони здоров'я, де розроблений метод детекції URL-адрес забезпечив точність 99%. Автори статті [104] представляють систему PhishingRTDS, яка використовує архітектуру BiLSTM з механізмом уваги та браузерне розширення для ідентифікації трьох типів атак, включаючи техніку "Browser in the Browser" (BiTB).

У праці [105] реалізовано підхід пояснювального машинного навчання (SHAP) спільно з ансамблями градієнтного бустингу для забезпечення прозорості та інтерпретованості рішень при виявленні фішингових сайтів. Дослідження [106] описує індустріальний фреймворк GraphWeaver, інтегрований у Microsoft Defender XDR, який використовує георозподілені графи для кореляції мільярдів подій безпеки з точністю 99%. У статті [107] представлено модульну систему виявлення вторгнень у хмарних середовищах, що поєднує GNN-вбудовування, трансформерні автоенкодери та контрастивне навчання для подолання дисбалансу класів.

У роботі [108] запропоновано метод на основі генеративно-змагальних мереж (GAN), де варіаційний автоенкодер (VAE) виступає генератором синтетичних URL, а трансформер — дискримінатором для розпізнавання

фальсифікацій. У дослідженні [109] представлено модель BERT-PhishFinder, яка використовує оптимізовану архітектуру DistilBERT та методи просторового дропауту для вилучення контекстуальних ознак URL-послідовностей у реальному часі.

У дослідженні [110] проведено порівняльний аналіз традиційних моделей машинного навчання та трансформерів, за результатами якого архітектура roBERTa продемонструвала найвищу точність (0,9943) у класифікації фішингових повідомлень. Стаття [111] пропонує багаторівневий ансамблевий фреймворк, що об'єднує гібридні моделі глибокого навчання (наприклад, StackedGRU-CNN-LSTM) та методи зваженого голосування для максимізації показників детекції URL-адрес. У роботі [112] представлено веб-платформу на базі інтерпретованого ШІ (XAI), яка використовує публічні датасети для реалізації прозорих механізмів виявлення фішингу, що сприяє підвищенню довіри користувачів до системи захисту.

Особливості протидії мобільному фішингу розглянуто у статті [113] через призму ансамблевого навчання та методів NLP (TF-IDF, N-grams), що дозволяє адаптивно реагувати на еволюцію смішинг-загроз. Важливий внесок у захист неангломовних сегментів інтернету зроблено у дослідженні [114], де запропоновано метод детекції для європейських мов (зокрема польської, румунської, словацької та ін.), що дозволило знизити рівень хибнопозитивних спрацювань у 10 разів. У праці [115] проведено комплексне тестування 14 моделей на десяти різних наборах даних, що підтвердило перевагу трансформерів над класичними алгоритмами в середньому на 4,7%.

Питання стійкості апаратної частини до побічних атак розглянуто у статті [116], де запропоновано дизайн SRAM-комірки, захищеної від аналізу витоків потужності. У роботі [117] представлено гібридну архітектуру, що поєднує BERT, CNN та GRU з оптимізатором Mountain Gazelle (MGO) для рафінування ознак у тексті електронних листів. Нарешті, у дослідженні [118] запропоновано децентралізований метод на основі федеративного глибокого навчання

(Federated Learning), який забезпечує конфіденційність даних користувачів при виявленні смішинг-атак за допомогою Bi-LSTM.

У дослідженні [119] запропоновано метричну схему приватності зображень, яка інтегрує механізми просторової уваги та теорію аналізу пар множин (SPA) для оцінки ризиків витоку даних при публікації візуального контенту. Питання безпеки блокчейн-платформ розглянуто у статті [120], де представлено метод PDTGA для мережі Ethereum. Автори використовують механізми уваги на часових графах для моделювання динамічної еволюції транзакцій, що дозволило досягти високих показників AUC та повноти детекції зловмисних адрес.

Особлива увага приділяється захисту мобільних комунікацій від смішинг-атак. У роботі [121] описано гібридний метод CNN-Bi-GRU, який трансформує неструктуровані текстові дані у числові представлення за допомогою Word2Vec, забезпечуючи точність ідентифікації загроз на рівні 99,82%. Схожий підхід запропоновано у статті [122], де архітектура CNN-LSTM використовується для автоматичного вилучення ознак із SMS-повідомлень, що дозволило суттєво знизити кількість хибнопозитивних спрацювань.

Класифікація шкідливого програмного забезпечення (malware) на основі аналізу послідовностей API-викликів представлена у дослідженні [123]. Запропонований метод MINES поєднує графове контрастивне навчання та аналіз матриць ймовірностей переходів для виявлення динамічної поведінки програм.

У роботі [125] описано фреймворк, що поєднує попередньо навчену модель MPNet із згортковими мережами та Bi-GRU для аналізу коротких неструктурованих текстів. Дослідження [126] фокусується на виявленні спаму турецькою та англійською мовами, підтверджуючи ефективність комбінації GRU та CNN при обробці багатомовних датасетів. Питання практичної реалізації захисту веб-ресурсів розглянуто у статті [127], де представлено ансамблеву систему на основі випадкових лісів та дерев рішень із Flask-інтерфейсом для моніторингу в реальному часі.

Гібридний метод CNN-LSTM для виявлення підроблених URL-адрес описано у дослідженні [128], де доведено перевагу такої архітектури над поодинокими моделями за рахунок здатності зберігати контекстуальні зв'язки в тексті. Нарешті, у статті [129] представлено метод оптимізації рекурентних нейронних мереж (RNN) за допомогою алгоритму китової оптимізації, що дозволило ефективно налаштовувати гіперпараметри для виявлення складних патернів фішингу.

У дослідженні [130] представлено метод виявлення фішингу в хмарних середовищах на основі архітектури RNN-LSTM, яка дозволяє фіксувати часові патерни взаємодії в динамічних хмарних інфраструктурах. Питання безпеки інтернету речей розглянуто у статті [131], де запропоновано гібридну техніку відбору ознак, що поєднує фільтраційні методи з генетичним алгоритмом (GA) для мінімізації обчислювальних витрат на ресурсообмежених IoT-пристроях. У роботі [132] описано мультимодальний фреймворк, який інтегрує аналіз SMS, електронної пошти та URL-поПеревага нь за допомогою алгоритму EM-BERT та пояснювального ШІ (EAI-SC-LSTM).

Інноваційний підхід MultiPhishGuard, описаний у статті [133], базується на багатоагентній системі з використанням великих мовних моделей (LLM) та навчанні з підкріпленням (PPO), де спеціалізовані агенти аналізують текст, метадані та URL-адреси. У дослідженні [134] представлено фреймворк MMTHF-Net, який об'єднує гіперграфові мережі та часові графові нейронні мережі (TGNN-Att) для захоплення високорівневих зв'язків між модальностями та динамічної поведінки сайтів. Питання стійкості візуальних методів детекції проаналізовано у праці [135], де на основі великого набору реальних даних виявлено вразливості популярних моделей до стратегій мімікрії логотипів.

Особливе місце у розвитку архітектур безпеки посідає концепція кібербезпекового цифрового двійника (Cybersecurity Digital Twin), представлена у статті [136]. Ця модель дозволяє створювати автономні процеси полювання на загрози та детекції латеральних рухів у мультидоменних сервісних ланцюжках. Методи виявлення шкідливого ПЗ для мобільних терміналів на основі

послідовностей API-викликів розглянуто у дослідженні [137], де застосування двоспрямованих мереж LSTM забезпечило високу точність класифікації.

У роботі [138] запропоновано модель HawkPhish-DNN, яка використовує алгоритм оптимізації Харріса-Яструба (HHO) для налаштування глибокої нейронної мережі, що дозволяє мінімізувати кількість хибнопозитивних спрацювань. У дослідженні [139] представл ено фреймворк на основі знаннєвої дистиляції моделі ELECTRA, інтегрований у браузерне розширення для надання користувачам миттєвих сповіщень про потенційні загрози в режимі реального часу.

У дослідженні [140] запропоновано інтегровану модель детекції фішингових URL-адрес, яка поєднує три модулі: RasNet для збалансування символьних і слівних ознак, TCMA для оптимізації вилучення ознак через часові згорткові мережі та MPNet для аналізу природної мовної структури. Робота [141] надає всебічний огляд атак соціальної інженерії в соціальних мережах, пропонуючи нову класифікацію методів маніпуляції людською психологією та стратегії етичної протидії. Питання соціотехнічного моделювання та симуляції безпеки (STSec-M&S) розглянуто у статті [142], де обґрунтовано необхідність цілісного підходу до захисту критичної інфраструктури, що враховує як технічні, так і людські чинники.

Для боротьби зі складними тривалими загрозами у праці [143] представлено систему AEKG4APT – граф знань, підсилений великими мовними моделями, що дозволяє виявляти ключові залежності та шляхи впливу в даних розвідки загроз CI. Аналогічно, у дослідженні [144] описано метод багатошарового графового контрастивного навчання для виявлення фроду в мережі Ethereum, що враховує топологію транзакційних зв’язків. Для моніторингу активних кампаній смішингу в реальному часі запропоновано систему SmishViz [145], яка використовує графову візуалізацію для ідентифікації тактик зловмисників.

Метод ASA-GNN, представлений у статті [146], використовує адаптивне семплювання та агрегацію для навчання дискримінативних представлень

транзакцій, що дозволяє ефективно фільтрувати «шумні» вузли та протидіяти стратегіям маскування злочинців. Питання інтерпретованості великих мовних моделей детально розібрано у SURVEY-дослідженні [147], де класифіковано техніки пояснюваності для моделей на базі Transformer. У роботі [148] представлено LLM-TIKG – метод автоматизованої побудови графів знань із неструктурованих відкритих джерел розвідки загроз (OSINT).

Для захисту смарт-мереж у дослідженні [149] запропоновано фреймворк федеративного навчання з нульовим пострілом, де LLM генерує допоміжну текстову інформацію про нові типи атак без розкриття конфіденційних даних споживачів. Питання крос-мовного кластерування текстів за допомогою LLM розглянуто у статті [150], а метод мультимодального синтезу трафікових графів для детекції вторгнень при обмеженій кількості даних (few-shot) представлено у праці [151]. У дослідженні [152] описано архітектуру проекту AC3 для управління безпекою та довірою в континуумі «хмара-край».

У роботі [153] запропоновано мультиагентний підхід із використанням RL для відбору ознак і DQL для класифікації атак, що дозволяє зменшити надлишковість даних і підвищити точність виявлення. Водночас мультиагентність застосовується переважно на рівні навчального конвеєра, без глибокого аналізу системної стійкості та координації агентів.

Дослідження [154] використовує конфронтаційну схему «атакуючий–захисник» разом із CTGAN для балансування класів, демонструючи покращення F1-метрики, однак така мінімалістична структура не враховує масштабованість і стабільність навчання в динамічних умовах.

У роботі [155] досліджено MARL для IDS із порівнянням централізованих і децентралізованих схем координації, що підвищує ефективність колективного прийняття рішень. Проте оцінювання базується на застарілих наборах даних і не охоплює показники архітектурної стійкості. Автори [156] запропонували адаптивну багатоагентну архітектуру з федеративним навчанням для роботи з дрейфом концепцій, але зосередилися переважно на якості виявлення, не аналізуючи стабільність взаємодії агентів.

У дослідженні [157] представлено Big-IDS — децентралізовану MARL-систему для високонавантажених середовищ, орієнтовану на масштабованість і продуктивність, однак без формалізованих метрик координації та живучості. Робота [158] акцентує увагу на архітектурі edge–cloud continuum для IoT, забезпечуючи реалістичне розміщення агентів, але залишає відкритим питання кількісної оцінки стійкості системи.

У праці [159] запропоновано модульний MARL IDS зі спеціалізованими агентами, що спрощує розширення системи новими типами атак, однак централізований агент прийняття рішень може виступати єдиною точкою відмови. Робота [160] застосовує ігрове моделювання з дуельним DQN, покращуючи адаптивність, проте результати значною мірою залежать від якості симуляційного середовища.

У дослідженні [161] інтегровано багатоагентні системи з LLM для семантичної кореляції загроз, що підвищує пояснюваність, але вводить нові ризики недетермінованості та нестабільності. Автори [162] використовують DDPG для адаптивного MARL у середовищах IoT, однак оцінювання обмежується метриками виявлення без аналізу масшабованості та відмовостійкості.

У роботах [163, 164] розглянуто поєднання MARL з XRL та LSTM відповідно, що покращує інтерпретацію рішень і часову обробку трафіку, але не формалізує стабільність колективної взаємодії агентів. Дослідження [165] пропонує гібрид MARL і мурашиної оптимізації, підвищуючи пошукові можливості, проте ускладнює модель і ускладнює оцінку її надійності.

У роботі [166] представлено спеціалізовану багатоагентну архітектуру для VANET, орієнтовану на швидку адаптацію до нових атак, однак її узагальнення на корпоративні мережі залишається обмеженим. Оглядові дослідження [167, 168] систематизують підходи MARL у кібербезпеці та підкреслюють відсутність стандартизованих критеріїв оцінки стабільності й живучості багатоагентних IDS.

У роботах [169, 170] запропоновано мультикомп’ютерні системи захисту з використанням технологій виявлення, що демонструють високу живучість і адаптивність архітектури, але не мають здатності до масштабування.

Дослідження [171, 172] орієнтовані на машинне навчання та аналітичні моделі для розподілених мереж і IoT, підтверджуючи ефективність інтелектуального аналізу трафіку, однак переважно зосереджуються на локальній ефективності виявлення, а не на комплексній оцінці колективної стійкості системи.

Аналіз методів показує, що переважна більшість наявних підходів до протидії атакам соціальної інженерії зосереджується на локальній детекції та покращенні показників класифікації, але не розглядає систему, яка повинна зберігати працездатність у динамічному, частково скомпрометованому та неповному середовищі. Для багатоагентних рішень відсутні глибокий аналіз стійкості та координації агентів у реальному розгортанні, атакож відсутність забезпечення мастабованості системи. Сучасні методи охоплюють різні канали та типи даних, однак їх масштабованість і адаптивність залишаються нерівномірними. Більшість рішень забезпечують лише середній рівень масштабування, а здатність до швидкого оновлення або роботи в мінливих умовах обмежена. У результаті навіть за наявності багатьох детекторів РКС не може підтримувати цілісне уявлення про події в реальному часі.

У багатоканальних системах кожне нове джерело подій чи сервіс формує додаткові комбінації станів, і складність зростає експоненційно. Це розширює простір можливих станів, спричиняє затримки, перевантажує ресурси та погіршує роботу механізмів виявлення. Узагальнену структуру цього процесу подано на рисунку В.1 Додатку В. Навіть незначне збільшення кількості агентів або інтенсивності взаємодій призводить до непропорційного росту навантаження, через що спеціалізовані детектори втрачають здатність корелювати події між каналами та пропускають мультимодальні сценарії.

Експоненційне ускладнення середовища створює системне протиріччя між реальним масштабом сучасних РКС і можливостями наявних засобів захисту. Це

підкреслює потребу у синтезі масштабованої, узгодженої та багатоканальної РКС, здатної працювати у реальному часі та забезпечувати стійкість до складних атак.

1.4 Висновки. Постановка задачі дослідження

Проведений огляд існуючих методів виявлення атак соціальної інженерії, виявив їхній ключовий системний недолік – наявні рішення демонструють низький рівень стійкості до атак соціальної інженерії РКС, оскільки не включають комплексно в процес такого синтезу стійкої до атак архітектури, достовірність забезпечення виявлення атак, адаптивність, масштабованість, живучість та ефективність прийняття колективних рішень.

Таким чином, на даний момент часу існує суперечність між потребою в синтезі комп'ютерних системи, стійких до атак соціальної інженерії, з одного боку, і недосконалістю методів та засобів забезпечення стійкості РКС в умовах атак соціальної інженерії, з іншого боку. Відтак, підвищення стійкості розподілених комп'ютерних систем до атак соціальної інженерії є актуальною науково-прикладною задачею, одним із шляхів розв'язання якої є розроблення методів і засобів синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії.

Метою дисертаційного дослідження є підвищення стійкості до атак соціальної інженерії розподілених комп'ютерних систем шляхом розроблення методів та засобів синтезу стійких до атак соціальної інженерії РКС, які комплексно забезпечують достовірність виявлення атак, адаптивність, масштабованості, живучість та ефективність прийняття колективних рішень вузлів РКС.

Розв'язання науково-прикладної задачі потребує вирішеня таких наукових задач:

- провести аналіз відомих методів і засобів забезпечення стійкості розподілених комп'ютерних систем до атак соціальної інженерії;

- розробити формальну модель розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка описує колективну поведінку агентів у динамічному середовищі шляхом узгодженого прийняття рішень, обміну інформацією та адаптивного керування ресурсами з метою максимізації глобальної функції корисності, що відображає стійкість системи до атак соціальної інженерії, забезпечення достовірного виявлення загроз, підтримання безперервності функціонування, збереження живучості за умов часткової компрометації вузлів і масштабування системи;

- розробити архітектуру стійкої до атак соціальної інженерії розподіленої комп’ютерної системи, яка базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак та підвищувати точність виявлення та класифікації атак соціальної інженерії;

- розробити метод виявлення кібератак соціальної інженерії в розподілених комп’ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, застосуванні методу k-найближчих сусідів, що уможливлює раннє виявлення мовних та семантичних маніпуляцій у сценаріях атак на розподіленої комп’ютерної системи;

- розробити метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженеріїна основі популяційної моделі багатоагентної системи та середнього поля, що забезпечує формування оптимальної політики поведінки репрезентативного агента, інтеграцію архітектурних параметрів та гарантовану масштабованість системи при зростанні кількості вузлів і інтенсивності атак;

- розробити метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів;

- розробити архітектуру програмної реалізації розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка включає агента прийняття рішень, сервісних агентів, компоненти моніторингу станів, менеджер взаємодії та модулі мовного/семантичного аналізу; провести експериментальні дослідження її характеристик у сценаріях впливів атак соціальної інженерії та оцінити покращення показників стійкості системи.

Основні результати розділу опубліковані у працях [1, 22]

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМ, СТІЙКИХ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

2.1 Модель розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії

Корпоративна мережа розглядається як розподілена комп’ютерна система, у якій взаємодія ресурсів і сервісів організована для підтримання безперервності роботи та стійкості до атак соціальної інженерії. Багаторівнева архітектура об’єднує інфраструктурні вузли та модулі контролю, що функціонують через стандартизовані протоколи та захищені канали, формуючи узгоджене середовище адаптивної реакції [173, 174].

Стійкість забезпечується інтеграцією технологічних, поведінкових і аналітичних механізмів. Кожен вузол містить агента, який аналізує локальні події, оцінює ризики та ініціює превентивні дії, автономно адаптуючись до змін середовища.

Взаємодія агентів через службові повідомлення дозволяє формувати колективні оцінки загроз та узгоджені реакції навіть за умов неповних даних чи локальних збоїв, забезпечуючи децентралізоване прийняття рішень і ситуаційну обізнаність.

У такій конфігурації мережа набуває властивостей самоорганізованої системи, що враховує поведінкові профілі та результати аналітичних моделей, нівелюючи вплив людського фактора. Сукупність цих елементів формує розподілену комп’ютерну систему, здатну автономно протидіяти атакам, підтримувати роботу в реальному часі та відновлювати функції після інцидентів.

У цьому поданні система природно інтерпретується як мультиагентна структура, узгоджена зі стійким до атак середовищем.

Побудуємо математичну модель мультиагентної системи (МАС), яка базується на формалізації взаємодії між агентами, описі їхньої поведінки, цілей та середовища. Ось основні компоненти математичної моделі такої системи:

$$A = \{A_1^u, A_2^s, A_3^m, A_4^c\}, \tag{2.1}$$

де A – множина агентів, що включає: $\mathrm{A}_1^{\mathrm{u}}$ – набір користувацьких агентів, які взаємодіють з користувачами; $\mathrm{A}_2^{\mathrm{s}}$ – набір сервісних агентів, що надають доступ до зовнішніх та внутрішніх послуг; $\mathrm{A}_3^{\mathrm{m}}$ – набір агентів моніторингу, які відстежують стан системи, аналізують файли журналів та попереджають про можливі збої; $\mathrm{A}_4^{\mathrm{c}}$ – набір комунікаційних агентів, що відповідають за обмін даними між різними агентами.

Кожен агент системи A_i визначається як:

$$A_i = \langle S_i, E_i, \Pi_i, \Phi_i, \Omega_i, \mathrm{C}_i \rangle, \tag{2.2}$$

де: S_i – множина станів агента A_i; E_i – набір дій, які може виконати агент; Π_i – стратегія або політика агента, яка визначає вибір дій у певному стані. $\Pi_i : S_i \times \mathcal{O} \rightarrow E_i$, де $\mathcal{O}$ – спостереження агента; Φ_i – функція корисності або цільова функція; Ω_i – набір ресурсів, доступних агенту; C_i – комунікаційна модель, яка визначає, як агент обмінюється інформацією з іншими агентами.

Середовище мультиагентної системи розглядається як композиція локальних станів агентів, яку формально подамо у вигляді множини:

$$\mathrm{S} = \{S_1, S_2, \ldots S_m\}. \tag{2.3}$$

Опишемо функцію динаміки навколишнього середовища:

$$T : S \times A \rightarrow S, \tag{2.4}$$

де T – зміна стану середовища в результаті дій агентів.

Взаємодію між агентами представимо у вигляді графа комунікації:

$$\mathcal{G} = (A, \mathcal{E}), \tag{2.5}$$

де $\mathcal{E}$ – це сукупність комунікаційних зв’язків між агентами. Зв’язок $e_i \in \mathcal{E}$ означає, що агент A_i може обмінюватися інформацією через A_j.

Для опису динаміки системи, визначимо модель переходу станів агента.

Стан кожного агента змінюється відповідно до функції:

$$S_i^{t+1} = f(S_i^t, E_i^t, O_i^t, C_i^t), \quad (2.6)$$

де: S_i^t – стан агента в момент часу t; E_i^t – дія, обрана агентом у даний момент t; O_i^t – спостереження агента за навколишнім середовищем; C_i^t – інформація, отримана через комунікацію.

Щоб описати взаємодію агента з середовищем, представимо результат дій агента на середовище як:

$$\mathrm{S_i^{t+1}} = \mathrm{T}\ (\mathrm{S^t}\{\mathrm{E_1^t}, \mathrm{E_2^t}, \ldots\ \mathrm{E_n^t}\}). \quad (2.7)$$

Для опису ефективного функціонування системи представимо функцію корисності $\mathrm{U_i}$, метою якої є її максимізація:

$$\mathrm{U_i} = \sum_{\mathrm{t=0}}^{\infty} \gamma^{\mathrm{t}}\ \Phi_{\mathrm{i}}\ (\mathrm{S_i^t}, \mathrm{S^t}), \quad (2.8)$$

де: $\Phi_{\mathrm{i}}\ (\mathrm{S_i^t}, \mathrm{S^t})$ – корисність агента в момент часу t, $\gamma \in [0,1]$ – коефіцієнт дисконтування (для довгострокового або короткострокового планування).

Оскільки система є розподіленою, необхідно визначити колективну функцію корисності. Якщо система орієнтована на досягнення спільної мети, введемо глобальну функцію корисності у вигляді:

$$\mathrm{U_{global}} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{U_i}, \quad (2.9)$$

де U_i – локальна функція корисності $i-$ го агента.

Виходячи з цього, для оптимізації функціонування розподіленої комп'ютерної системи необхідно розв'язати задачу пошуку набору стратегій $\{\Pi_1, \Pi_2, \ldots, \Pi_n\}$, який максимізує глобальну функцію корисності:

$$\{\Pi_1^*, \ldots, \Pi_n^*\} = \arg \max_{\{\Pi_1, \ldots, \Pi_n\}} U_{\mathrm{global}}. \quad (2.10)$$

Розподілену обробку даних будемо інтерпретувати як сукупність локальних оптимізацій, коли кожен агент вибирає стратегію Π_i, що максимізує його власну корисність, $\Pi_i^* = \arg \max_{\Pi_i} U_i$, за умови узгодженості цих локальних рішень із глобальною метою. Таку узгодженість забезпечує механізм комунікації

та спільного обміну інформацією між компонентами розподіленої комп’ютерної системи.

Для забезпечення балансу в розподіленій системі взаємодію агентів опишемо через рівновагу Неша [175]. Тобто необхідним є досягнення такого стану багатоагентної системи, за якого жоден агент не може покращити власний показник ефективності (функцію корисності), змінивши лише свою стратегію, якщо стратегії інших агентів залишаються незмінними, і за якого жоден агент не може покращити своє значення функції корисності, змінюючи лише власну стратегію за фіксованих стратегій інших агентів. Формально профіль стратегій $\{\Pi_1^*, \dots, \Pi_n^*\}$ є рівновагою Неша, якщо для будь-якого агента i виконується: $U_i(\Pi_i^*, \Pi_{-i}^*) \geq U_i(\Pi_i, \Pi_{-i}^*) \forall \Pi_i$, де Π_{-i}^* – фіксовані стратегії всіх агентів, окрім i-го.

Подамо процес комунікації та узгодженості між агентами, через алгоритм консенсусу для протоколів обміну:

$$x_i^{t+1} = \sum_{j \in N_i} \omega_{ij} \, x_i^{t+1}, \tag{2.11}$$

де N_i – сусіди агента A_i, ω_{ij} – ваги графа зв'язку.

2.2 Архітектура стійкої до атак соціальної інженерії розподіленої комп’ютерної системи

2.2.1 Опис розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, на основі ієрархічної багатоагентної системи

У дослідженні пропонується архітектура розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії на основі ієрархічної багатоагентної системи з використанням навчання з підкріпленням. Запропонована архітектура розподіленої комп’ютерної системи, передбачає здантість до виявлення атак соціальної інженерії, та спирається на взаємодію компонентів системи, що відображають ключові процеси аналізу інформаційного середовища [176].

В основі такої системи лежить агент, який, у координації з класифікатором атак, формулює політику діалогу між системою, користувачем та іншими

джерелами загрози. Система оперує множиною моделей атак соціадьної інженерії, які використовуються як база знань про поведінку атак в розподіленій комп'ютерній системі (описані в Підпункті 2.3).

Схему архітектури розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, на основі ієрархічної багатоагентної системи подано на рисунку 2.1.

Важливу роль відіграє компонент моніторингу стану, підсистема, яка забезпечує безперервне спостереження, агрегування та інтерпретацію подій, телеметричних сигналів і локальних метрик, що надходять від агентів-детекторів у розподіленій комп'ютерній системі, та компонент механізму моделювання, підсистема, яка створює динамічне, варіативне та наближене до реальності середовище, у якому агенти, де детектори навчаються розпізнавати та протидіяти атакам соціальної інженерії.

Вся система задумана як ієрархічна структура, в якій агент високого рівня вирішує, які конкретні дії призначити спеціалізованим субагентам нижчого рівня. Таким чином, формується багаторівнева система прийняття рішень, яка може гнучко реагувати на нові тактики соціальної інженерії та знижувати ризики для РКС шляхом проактивного збору відповідних ознак та поступового зменшення ентропії інформаційного стану.

Архітектура система включає використання компонента менеджера взаємодій – агента, здатного до цілеспрямованої взаємодії для ведення діалогів з інформаційним середовищем. Цей компонент багатоагентної системи відповідає за керування послідовністю взаємодій між агентом та інформаційним середовищем. Він визначає, які саме дії мають бути виконані на поточному кроці, та формує локальну політику для кожного сервісного агента.

Агент прийняття рішень τ отримує від компонента моніторингу стану поточну конфігурацію середовища, яка відображає набір індикаторів та поведінкових характеристик, зібраних на даний момент. Цей стан потім відображається на вибір дії α_τ з набору можливих дій A відповідно до політики $\rho(\alpha \mid \sigma(\tau))$.

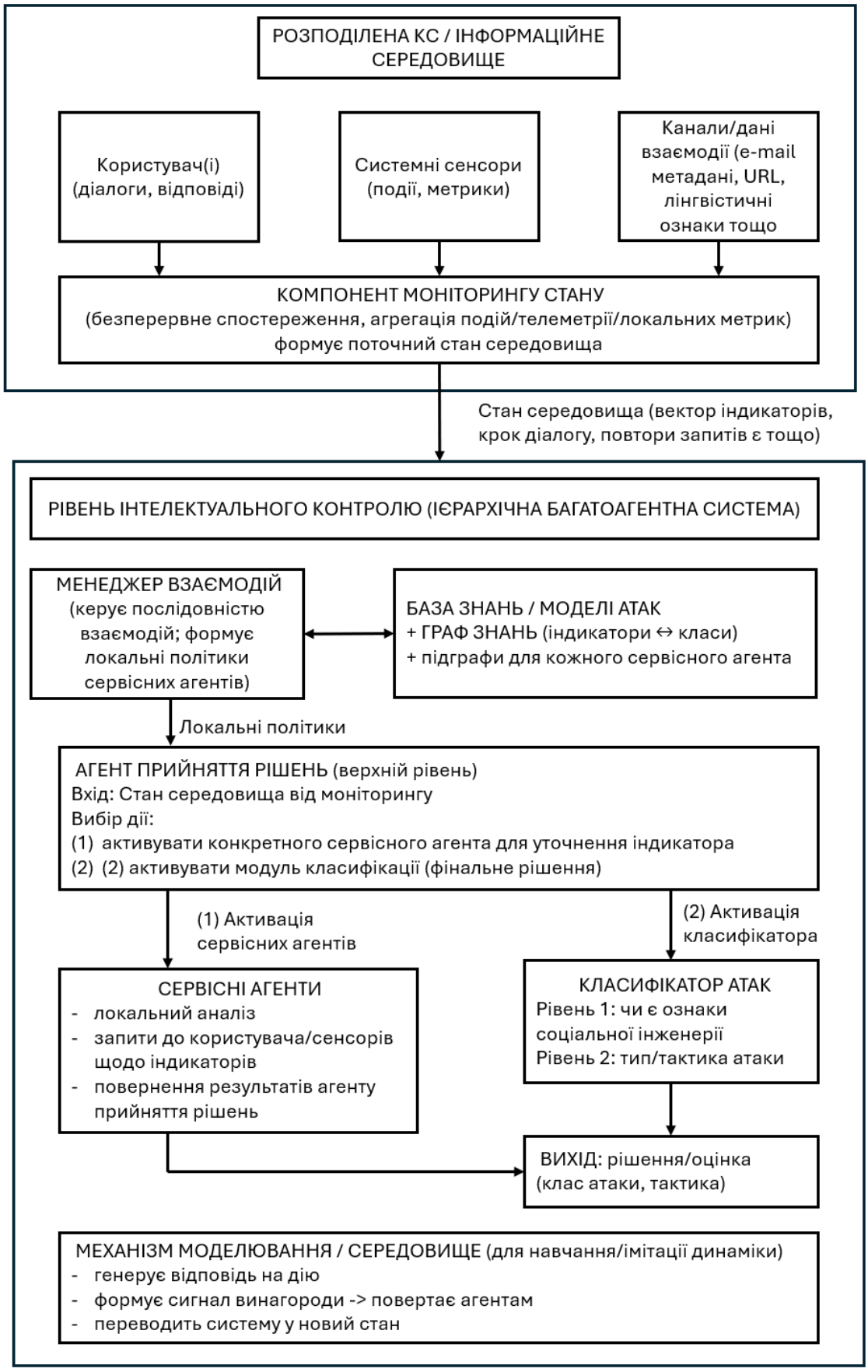


Рисунок 2.1 – Схема архітектури РКС, стійкої до атак соціальної інженерії, на основі ієрархічної багатоагентної системи

Класифікатор атак працює на двох рівнях: на першому рівні він визначає, чи має вхідна інформація ознаки соціальної інженерії; на другому рівні уточнюється оцінка для визначення типу (класу) атаки та конкретної тактики, що застосовується ініціатором атаки.

Компонент механізму моделювання генерує відповідь, яка служить основою для обчислення сигналу винагороди ω_τ, яка повертається агенту. В результаті середовище переходить у новий стан $\sigma(\tau+1)$, і таким чином розгортає послідовність ітерацій, що імітують реальну динаміку комунікації.

Мета агента прийняття рішень полягає в пошуку оптимальної політики, яка забезпечує максимізацію очікуваної кумулятивної винагороди. Винагорода ω_τ визначається функцією, $\Omega(\sigma(\tau), \alpha_\tau)$ де коефіцієнт $\lambda \in [0,1]$ служить дисконтним коефіцієнтом, що зменшує вагу віддалених у часі результатів.

Стан середовища описує множину зібраних індикаторів атаки, до яких можуть належати ознаки фішингових повідомлень, лінгвістичні маркери психологічного тиску, метадані електронної пошти або характерні патерни URL-адрес.

Визначимо A як множину типів атак соціальної інженерії, а I – множину індикаторів, пов'язаних з цими атаками. Тоді стан $\sigma(\tau)$ подамо як вектор $[\chi_1, \chi_2, \dots, \chi_{|I|}, \tau, \varepsilon]$, де кожен компонент χ_i є закодованим тернарним представленням наявності індикатора: підтверджена наявність, підтверджена відсутність або невизначений стан.

Вектор включає параметр τ, що відображає поточний крок діалогу, та змінну ε , яка сигналізує про повторення попередніх запитів. Ця формалізація дає змогу системі не лише накопичувати знання про зібрані індикатори, але й адаптивно керувати стратегією запитів, мінімізуючи інформаційну надмірність і підвищуючи точність виявлення атак соціальної інженерії в умовах неповної та суперечливої інформації.

Простір дій агента прийняття рішень формується як об’єднання двох підмножин, що представляють різні типи дій у процесі виявлення атак соціальної інженерії. Визначаємо $A = T \cup I$, де T – множина типів атак, I – множина

індикаторів, які можна запитувати для уточнення контексту повідомлення або взаємодії. Якщо вибрана дія α_τ належить до I, агент ініціює запит додаткової інформації щодо конкретного індикатора. Отримана відповідь кодується у відповідний компонент вектора стану та уточнює оцінку сценарію на кроці τ. У разі вибору дії з підмножини T агент формує остаточне рішення щодо ймовірного типу атаки, що завершує процес взаємодії в рамках поточного сценарію .

Успішність процесу визначається точністю класифікації загрози, тобто коректністю віднесення взаємодії до відповідного класу соціальної інженерії.

Така формалізація дає змогу агенту збалансовувати збирання додаткових ознак і ухвалення рішень, підвищуючи ефективність роботи в умовах неповної чи суперечливої інформації. Ієрархічна організація системи забезпечує раціональний розподіл обчислювальних ресурсів і сприяє послідовному зменшенню невизначеності під час оцінювання ризику, що є важливим для стійкості розподілених комп'ютерних систем до атак соціальної інженерії.

2.2.2 Синтез архітектури комп'ютерних систем, стійких до атак соціальної інженерії, на основі ієрархічної багатоагентної системи

Позначимо множину атак соціальної інженерії як множину, T поділену на K групи, де

$$T = T_1 \vee T_2 \vee \ldots \vee T_K, \tag{2.12}$$

підмножини не перетинаються, тобто $T_i \wedge T_j = \emptyset$ для $i \neq j, i, j = 1, \ldots, K$.

Кожна підмножина T_i відповідає певній категорії атак або тактичній стратегії порушника та пов'язана з власною множиною індикаторів I_i, які характеризують ознаки повідомлень, поведінкові патерни чи технічні сигнали, притаманні цій групі.

Побудована архітектура РКС передбачає два рівні:

1) агент з прийняття рішень вищого рівня (Decision Agent, DA, агент прийняття рішень, централний агент), який координує загальну політику збору інформації та прийняття рішень;

2) агенти обслуговування нижчого рівня або сервісні агенти (Service Agents, SA) відповідають кожен за власну підмножину атак T_{i}та пов'язані з нею індикатори I_{i} . Сервісні агенти виконують функції локального аналізу, ініціюють запити до користувачів або системних сенсорів, оцінюють атрибути повідомлень і передають результати координатору верхнього рівня.

Завдяки такому розподілу функцій агент прийняття рішень може формувати обґрунтовані рішення на основі агрегованих сигналів, спрямованих на мінімізацію ризику та зменшення невизначеності щодо характеру потенційної атаки.

Вибір дій агентами формалізується через відображення стану середовища $\sigma(\tau)$ на простір дій α_τ з множини: $A = T \cup I$, де кожна дія нижчого рівня стосується лише підмножини індикаторів I_{i}, а дії агента вищого рівня координують послідовність запитів і формування підсумкової оцінки типу загрози. У кожному циклі взаємодії агент з прийняття рішень отримує стан середовища, $\sigma(\tau)$ який відображає сукупність усіх зібраних індикаторів потенційних атак соціальної інженерії на цей момент.

На основі політики ρ_0 $(\alpha_0\tau/\sigma(\tau)$ агент вищого рівня приймає рішення щодо подальшого курсу дій: продовжувати збір ознак через запити до користувача чи системних датчиків, або формувати остаточну оцінку ймовірного типу загрози.

Простір дій агента прийняття рішень визначається множиною:

$$A_0 = \{W_1, W_2, \dots, W_K, C\}, \tag{2.13}$$

де W_{i} позначає сервісного агента, відповідального за аналіз певної підмножини індикаторів I_{i}, а C відповідає модулю класифікації атак.

Коли активовано одного з сервісних агентів, ініціюється підзадача, в рамках якої $\tau \in [\tau_0, \tau_0 + L]$ той самий сервісний агент виконує наступні кроки збору ознак, де L визначає максимальну кількість ітерацій для підзадачі.

Поточна підзадача завершується після отримання конкретної відповіді на запит щодо індикатора, яка може набувати дискретних значень, наприклад «підтверджено» або «не підтверджено». Якщо активується модуль класифікації C, агент прийняття рішень формує підсумкове рішення щодо типу атаки, завершуючи поточний цикл взаємодії.

Агент прийняття рішень координує роботу сервісних агентів, оптимізуючи запити та процес класифікації, тоді як сервісні агенти виконують локальні підзадачі зі збирання ознак. Така організація зменшує обчислювальну складність і підвищує точність виявлення складних сценаріїв. Архітектура забезпечує адаптивне коригування стратегій залежно від поточного стану, мінімізує невизначеність і підсилює стійкість розподіленої КС до атак соціальної інженерії.

Коли сервісного агента активовано, він отримує стан середовища $\sigma(\tau)$ та витягує відповідні індикатори υ_τ, що відповідають призначеній йому підгрупі загроз:

$$\upsilon_\tau = [\chi_1, \chi_2, \dots, \chi_{|I_I|}, \tau, \varepsilon]. \quad (2.14)$$

Кожен сервісний агент працює в межах локального підпростору ознак, що дає йому змогу зосереджуватися на окремих категоріях атак соціальної інженерії та відповідних поведінкових або технічних сигналах.

На основі локальної політики, сформованої компонентом менеджера взаємодій $\rho_i(\alpha_i\tau/\upsilon_\tau)$, сервісний агент обирає наступний індикатор для уточнення або перевірки та виконує дії зі збирання додаткової інформації, яка уточнює ймовірність конкретного типу атаки.

Простір дій кожного сервісного агента MAC визначається множиною:

$$A_i = \upsilon_1, \upsilon_2, \dots, \upsilon_{|I_i|}, \quad (2.15)$$

де кожен елемент відповідає певному індикатору або шаблону. Така локалізація дій дає змогу сервісним агентам поступово накопичувати знання щодо конкретної категорії загроз. Ієрархічна взаємодія з агентом прийняття рішень дозволяє агрегувати ці локальні результати та коригувати загальну політику системи, підвищуючи адаптивність та стійкість РКС до атак соціальної інженерії.

Агент прийняття рішень отримує зовнішню винагороду $\omega_e\tau$ на кожному циклі взаємодії, яка відображає ефективність прийнятих рішень у виявленні потенційних атак соціальної інженерії. Після виконання підзавдання, агент прийняття рішень розраховує накопичену винагороду $\Omega_0\tau$ за даний цикл, враховуючи внесок кожного сервісного агента та результати локальної оцінки показників.

Нехай K позначає кількість сервісних агентів, а $\lambda \in [0,1]$ – коефіцієнт дисконтування, який зменшує вагу віддалених у часі результатів. Тоді накопичена винагорода визначається як:

$$\Omega_0\tau = \begin{cases} \sum_{\kappa=1}^{L} \lambda^{\kappa}\omega^{e}_{(\tau+\kappa)}, \text{якщо } \alpha_0\ \tau = W_{\text{і}}; \\ \omega^{e}_{(\tau)}, \text{якщо } \alpha_0\tau = C. \end{cases}, \quad (2.16)$$

де $W_{\text{і}}$ – сервісний агент, відповідальний за підзадачу аналізу підмножини індикаторів $I_{\text{і}}$, C – модуль класифікації типів атак. Така формалізація дає змогу агенту прийняття рішень інтегрувати результати локального аналізу, оцінювати ефективність дій на всіх рівнях та адаптивно коригувати політику збору даних і прийняття рішень. Використання коефіцієнта дисконту λ забезпечує баланс між короткостроковими та довгостроковими наслідками.

Мета функції $L(\Theta_0)$ визначається через дискретну динаміку навчання, що ґрунтується на підкріплювальному сигналі та історії взаємодій. Нехай σ' – наступний стан середовища після виконання дії $\alpha_0{}'$, $\Theta_0{}'$ – параметри цільової мережі, Θ_0 – параметри поточної мережі та B_0 – буфер досвіду агента прийняття рішень.

Тоді функція втрат визначається як:

$$L(\Theta_0) = E_{\sigma,\alpha_0,\Omega_0,\sigma'\sim B_0}((\Omega_0 + \lambda max_{\alpha_0{}'} Q_0(\sigma', \alpha_0{}'; \Theta_0{}') - Q_0(\sigma', \alpha_0; \Theta_0))^2), \quad (2.17)$$

де $Q_0(\sigma, \alpha_0; \Theta_0)$ – оцінка очікуваної кумулятивної винагороди згідно з поточною політикою, $\lambda max_{\alpha_0{}'} Q^0(\sigma', \alpha^{0\prime}; \Theta^{0\prime})$ забезпечує врахування довгострокових наслідків дій. Така формалізація дає змогу агенту прийняття рішень ефективно інтегрувати інформацію від сервісних агентів та модуля класифікації, оцінюючи як короткострокові, так і довгострокові наслідки своїх рішень.

Сервісні агенти отримують внутрішню винагороду $\omega_i\tau$ на кожному циклі взаємодії, яка оцінює ефективність їх локальної роботи з підмножиною індикаторів $I_{\text{і}}$, закріпленою за певною категорією атак соціальної інженерії.

Метою сервісних агентів є максимізація цих винагород, що забезпечує точне та ефективне виконання підзадач зі збору інформації.

Функція втрат кожного сервісного агента $L(\Theta_{\text{і}})$ визначається на основі локальної політики та історії взаємодії в буфері відтворення B_i, де Θ_i' – параметри цільової мережі, $\Theta_{\text{і}}$ – параметри поточної мережі:

$$L(\Theta_i) = E_{\sigma_i,\alpha_i,\Omega_i,\sigma' \sim B_i}((\Omega_i + \lambda max_{\alpha_i'} Q_i(\sigma_i', \alpha_i'; \Theta_i') - Q_0(\sigma_i, \alpha_i; \Theta_i))^2), \quad (2.18)$$

де $Q_{\text{і}}(\upsilon_{\text{і}}, \alpha_{\text{і}}; \Theta_{\text{і}})$ – очікувана кумулятивна винагорода за поточною локальною політикою, $\lambda max_{\alpha_i'} Q_i(\upsilon_i', \alpha_i'; \Theta_i')$ – довгострокові наслідки дій агентів.

Така формалізація дає змогу сервісним агентам адаптивно коригувати власну стратегію збору ознак, зосереджуючись на локальних підзадачах, і поступово підвищувати точність оцінювання ризиків для конкретних категорій атак. Координація з агентом прийняття рішень дає можливість агрегувати локальні результати та забезпечує оптимізацію загальної політики системи.

2.2.3 Розрахунок винагороди

У побудованій архітектурі РКС, починаючи з початкового стану σ_0, для основного класифікатора загроз та допоміжних класифікаторів, що спеціалізуються на окремих категоріях атак соціальної інженерії, формуються початкові ймовірнісні розподіли. Для агента прийняття рішень позначимо цей розподіл як $\phi_0^D = [\phi_1, \phi_2, \dots, \phi_k]$, де кожен компонент ϕ_i – апостеріорна оцінка належності поточного сценарію до певного класу загроз. Для сервісних агентів визначимо аналогічний розподіл $\phi_0^S = [\phi_1, \phi_2, \dots, \phi_j]$, де кожен елемент представляє ймовірність виявлення відповідної підкатегорії атаки в межах множини можливих дій A.

Важливою характеристикою цих розподілів є ентропія, яка відображає ступінь невизначеності під час прийняття рішень. Початкова ентропія агента прийняття рішень, що працює на рівні інтегральної оцінки загрози, визначається виразом:

$$J^D(\sigma_0) = -\sum_{i=1}^{\kappa} - \phi_i \log(\phi_i), \tag{2.19}$$

тоді як початкова ентропія сервісних агентів розраховується за формулою:

$$J^S(\sigma_0) = -\sum_{i=1}^{|A_i|} \phi_i \log(\phi_i). \tag{2.20}$$

Обидві метрики ентропії слугують нормалізаторами, запобігаючи зміщенням у винагороді між епізодами, що важливо за суттєвих відмінностей у характері вхідної інформації. Використання ентропії як керувального параметра дає багатоагентній системі змогу адаптивно балансувати між точністю виявлення атак і стійкістю до інформаційного шуму, характерного для реальних РКС.

Аналогічно, у момент часу τ обчислюється інформаційна ентропія основного класифікатора агента прийняття рішення та $J^D(\sigma_\tau)$ класифікатора $J^S(\sigma_\tau)$ агента обслуговування . Після того, як система виконує дію та переходить у новий стан $\sigma_{\tau+1}$, оновлені значення ентропії визначаються як $J^D(\sigma_{\tau+1})$ та $J^S(\sigma_{\tau+1})$. Різниця між цими значеннями стає основою для розрахунку винагороди, яка характеризує зменшення невизначеності в роботі агента прийняття рішення та агентів обслуговування.

Для агента прийняття рішення значення винагороди визначається співвідношенням:

$$\rho^D_{\text{entropy}} = max(\frac{(J^D(\sigma_\tau) - J^D(\sigma_{\tau+1}))}{J^H}(\sigma_0), 0). \tag{2.21}$$

Розмір винагороди сервісних агентів визначається за формулою:

$$\rho^S_{\text{entropy}} = max(\frac{(J^S(\sigma_\tau) - J^S(\sigma_{\tau+1}))}{J^S}(\sigma_0), 0). \tag{2.22}$$

У цих виразах відношення різниці ентропій до початкового значення виконує нормалізацію функціонала, що дає змогу коректно порівнювати

інформаційний приріст між епізодами. Використання оператора max унеможливлює появу від’ємних значень, які можуть виникати за неповних або зашумлених даних, забезпечуючи стабільне навчання в умовах підвищеної невизначеності, характерної для атак соціальної інженерії. За такої структури винагороди багатоагентна система отримує механізм самоадаптації, орієнтований на мінімізацію ентропії стану та формування однозначного діагностичного висновку щодо наявності індикаторів атаки.

Зовнішню функцію винагороди ρ_{t}^{ext} визначимо наступним чином:

$$\rho_{\text{t}}^{ext} = \sum_{\iota=1}^{n} N + \rho_{\text{entropy}}^{D}, \tag{2.23}$$

де N є додатним числом, якщо агент прийняття рішень видає остаточне рішення, і це рішення є правильним (результат «успіх»); N є від'ємним числом, якщо вибрана дія на кроці $t_{,}$ повторює попередньо виконану дію в тому ж епізоді (виявлено повторення дії, надлишкові запити караються), або якщо досягнуто максимально дозволеної кількості кроків взаємодії без остаточного рішення (бюджет епізоду вичерпано, нерішучість карається); N дорівнює 0 у всіх інших випадках (взаємодія триває, винагорода пропорційна зменшенню невизначеності, додаткова вигода від інформативних дій, що зменшують невизначеність, масштабується) $\rho_{\text{entropy}}^{D} > 0$.

2.2.4 Внутрішня винагорода для сервісних агентів системи

Внутрішня винагорода $\rho_{\text{t}}^{\text{int}}$ для сервісного агента визначається наступним чином:

$$\rho_{\text{t}}^{\text{int}} = \sum_{\iota=1}^{n} N + \rho_{\text{entropy}}^{S}, \tag{2.24}$$

де N є додатним числом, якщо запит сервісного агента дає остаточне та правильне рішення цільового індикатора або ознаки (спостерігається «правильий збіг»); N є від'ємним числом, якщо срвісний агент видає повторний або надлишковий запит, який не приносить нової інформації (повторення дій); N дорівнює 0 у всіх інших випадках (поступове зменшення невизначеності без остаточного збігу) $\rho_{\text{entropy}}^{S} > 0$.

Максимальна кількість кроків взаємодії визначає тривалість епізоду й використовується як умова тайм-ауту зовнішньої винагороди. Нормалізація за початковою ентропією забезпечує порівнюваність винагород у сценаріях з різним рівнем невизначеності. Результат «успіх» узгоджує поведінку агента верхнього рівня з вимогою точної класифікації, тоді як локальний «правильний збіг» спрямовує сервісних агентів на точне уточнення ознак. Штрафи за повторення дій виконують роль регуляризаторів, що запобігають циклам і надмірним запитам. У сукупності така структура збалансовує прагнення до швидкої взаємодії з необхідністю вимірюваного зменшення невизначеності.

Ієрархічна система винагород узгоджує розподіл функцій між рівнями: агент ухвалення рішень координує процес оцінювання, а сервісні агенти виконують роль локальних фільтрів. Це забезпечує багаторівневу стійкість РКС до атак соціальної інженерії та підтримує адаптацію системи в динамічних умовах.

Окрім винагороди, що базується на зменшенні невизначеності ентропії, основний агент отримує додаткові підкріплення, які генеруються залежно від результату прийнятого рішення. Якщо остаточна класифікація сценарію атаки успішна, то застосовується позитивний приріст винагороди, який позначається як σ^+. У разі некоректної ідентифікації або досягнення максимально допустимої кількості ітерацій діалогу вводиться штрафна складова σ^-, яка відображає неефективність стратегії агента.

Для сервісних агентів, що відповідають за локальне уточнення ознак, винагорода визначається за іншим принципом. Якщо обраний сервісний агент вірно уточнює ознаку, що характеризує можливу атаку, та отримує однозначну відповідь від середовища, то він отримує позитивний приріст τ^+. Якщо відбувається повторний запит, який не зменшує ентропію інформаційного стану та призводить до інформаційного шуму, то вводиться негативний штраф τ^-.

Формалізація сигналів підкріплення на різних рівнях архітектури системи поєднує глобальну оцінку продуктивності агента прийняття рішень з локальною точністю роботи сервісних агентів. Використання параметрів σ^+, σ^-, τ^+ та τ^-

дозволяє збалансувати стратегічні та тактичні аспекти функціонування системи, створюючи умови для адаптивного навчання в динамічному середовищі.

2.2.5 Побудова знань для компонента системи менеджера взаємодії

У запропонованій РКС знання про сценарії соціальної інженерії подаються у вигляді графа знань, інтегрованого в менеджер взаємодій. Вузли графа відповідають індикаторам та класам атак, а ребра відображають кореляційні зв'язки між ними. Кожному ребру призначається вага, що характеризує силу такого зв'язку і визначається на основі ймовірнісної статистики, отриманої з навчальних і тестових сценаріїв.

Оскільки система використовує багатоагентну структуру, сервісні агенти відповідають за обробку підмножин можливих індикаторів та класів атак. Для кожного такого агента будується окремий підграф знань, що містить набір індикаторів O_i та набір атак A_i, що стосуються його підсистеми. Це забезпечує локалізацію обчислень та зменшує загальну складність синтезу архітектури, оскільки окремі агенти спеціалізуються на певних класах загроз.

Вагові коефіцієнти ребер у підграфах трактуються як умовні ймовірності появи певної ознаки за умови реалізації відповідного класу атаки. Таку залежность представлено у формі наступного рівняння:

$$q(o_u a_v) = \frac{m(a_v, o_u)}{\sum_{i=1}^{n} m(a_v, o_z)}, \tag{2.25}$$

де $m(a_v, o_u)$ – кількість зареєстрованих сценаріїв, a_v) – атака з проявом ознаки o_u.

Обчислюючи значення ймовірності для кожної пари «атака – ознака», побудовано матрицю умовних ймовірностей $\Lambda_i \in R^{|O_i| \times |A_i|}$, яка використовується в процесі навчання агента прийняття рішень. Це дає змогу відобразити багатовимірні залежності між типами атак соціальної інженерії та їх ознаками, забезпечуючи адаптивне навчання агентів під час взаємодії з потенційно ворожими сценаріями.

На основі матриці умовних ймовірностей, введеної вище, поточний розподіл ймовірностей проявів ознак для сервісних агентів визначається як:

$$\pi(o_u) = \Lambda_i \cdot \pi(a_v) \tag{2.26}$$

де $\pi(a_v) \in R^{|A_i|\times 1}$ вектор ймовірностей відповідних класів атак, який обчислюється підпорядкованим класифікатором для підгрупи i, та $\pi(o_u) \in R^{|O_i|\times 1}$– розподіл ймовірностей для локальних ознак, що стосуються цього агента. Ця операція перетворює ймовірності класів атак на ймовірності прояву конкретних ознак, забезпечуючи основу для прийняття рішень агентом.

Далі цей розподіл ймовірностей поєднується з оцінками Q значень, обчислених сервісними агентами, що дозволяє враховувати як попередні знання про ймовірності появи ознак, так і досвід навчання агента в динамічному середовищі. Формально вибір наступної дії агента виражається як:

$$\alpha_\tau = argmax_\omega(\sigma(Q_\omega(\sigma_\tau)) + \sigma(\pi(o_u))), \tag{2.27}$$

де $Q_\omega(\sigma_\tau)$ – розподіл значень Q, обчислених сервісним агентом для стану σ_τ а σ – сигмоїдна функція нормалізації, яка відображає значення в діапазон [0,1], функція argmax визначає клас ознак або дій, на якому агент зупиняє свій вибір для подальшого запиту або оцінки.

Поєднання ймовірностей та Q значень , дає агенту прийняття рішень змогу враховувати як попередній досвід, так і оновлену інформацію про поведінку користувача чи системи, що є важливим для протидії атакам соціальної інженерії в межах РКС.

Класифікатор атак працює на двох рівнях: на першому рівні він визначає, чи має вхідна інформація ознаки соціальної інженерії; на другому рівні уточнюється оцінка для визначення типу (класу) атаки та конкретної тактики, що застосовується ініціатором атаки.

Компонент механізму моделювання генерує відповідь, яка служить основою для обчислення сигналу винагороди ω_τ, яка повертається агенту. В результаті середовище переходить у новий стан $\sigma(\tau + 1)$, і таким чином розгортає послідовність ітерацій, що імітують реальну динаміку комунікації.

Мета агента прийняття рішень полягає в пошуку оптимальної політики, яка забезпечує максимізацію очікуваної кумулятивної винагороди. Винагорода ω_τ визначається функцією, $\Omega(\sigma(\tau), \alpha_\tau)$ де коефіцієнт $\lambda \in [0,1]$ служить дисконтним коефіцієнтом, що зменшує вагу віддалених у часі результатів.

Стан середовища описує множину зібраних індикаторів атаки, до яких можуть належати ознаки фішингових повідомлень, лінгвістичні маркери психологічного тиску, метадані електронної пошти або характерні патерни URL-адрес.

Визначимо A як множину типів атак соціальної інженерії, а I – множину індикаторів, пов'язаних з цими атаками. Тоді стан $\sigma(\tau)$ подамо як вектор $[\chi_1, \chi_2, \dots, \chi_{|I|}, \tau, \varepsilon]$, де кожен компонент χ_i є закодованим тернарним представленням наявності індикатора: підтверджена наявність, підтверджена відсутність або невизначений стан.

Вектор включає параметр τ, що відображає поточний крок діалогу, та змінну ε , яка сигналізує про повторення попередніх запитів. Ця формалізація дає змогу системі не лише накопичувати знання про зібрані індикатори, але й адаптивно керувати стратегією запитів, мінімізуючи інформаційну надмірність і підвищуючи точність виявлення атак соціальної інженерії в умовах неповної та суперечливої інформації.

Простір дій агента прийняття рішень формується як об’єднання двох підмножин, що представляють різні типи дій у процесі виявлення атак соціальної інженерії. Визначаємо $A = T \cup I$, де T – множина типів атак, I – множина індикаторів, які можна запитувати для уточнення контексту повідомлення або взаємодії. Якщо вибрана дія α_τ належить до I, агент ініціює запит додаткової інформації щодо конкретного індикатора. Отримана відповідь кодується у відповідний компонент вектора стану та уточнює оцінку сценарію на кроці τ. У разі вибору дії з підмножини T агент формує остаточне рішення щодо ймовірного типу атаки, що завершує процес взаємодії в рамках поточного сценарію .

Успішність процесу визначається точністю класифікації загрози, тобто коректністю віднесення взаємодії до відповідного класу соціальної інженерії.

Така формалізація дає змогу агенту збалансовувати збирання додаткових ознак і ухвалення рішень, підвищуючи ефективність роботи в умовах неповної чи суперечливої інформації. Ієрархічна організація системи забезпечує раціональний розподіл обчислювальних ресурсів і сприяє послідовному зменшенню невизначеності під час оцінювання ризику, що є важливим для стійкості розподілених комп'ютерних систем до атак соціальної інженерії.

2.2.2 Синтез архітектури комп'ютерних систем, стійких до атак соціальної інженерії, на основі ієрархічної багатоагентної системи

Позначимо множину атак соціальної інженерії як множину, T поділену на K групи, де

$$T = T_1 \vee T_2 \vee \ldots \vee T_K, \tag{2.12}$$

підмножини не перетинаються, тобто $T_i \wedge T_j = \emptyset$ для $i \neq j, i, j = 1, \ldots, K$.

Кожна підмножина T_{i} відповідає певній категорії атак або тактичній стратегії порушника та пов'язана з власною множиною індикаторів I_{i}, які характеризують ознаки повідомлень, поведінкові патерни чи технічні сигнали, притаманні цій групі.

Побудована архітектура РКС передбачає два рівні:

1) агент з прийняття рішень вищого рівня (Decision Agent, DA, агент прийняття рішень, централний агент), який координує загальну політику збору інформації та прийняття рішень;

2) агенти обслуговування нижчого рівня або сервісні агенти (Service Agents, SA) відповідають кожен за власну підмножину атак T_{i}та пов'язані з нею індикатори I_{i} . Сервісні агенти виконують функції локального аналізу, ініціюють запити до користувачів або системних сенсорів, оцінюють атрибути повідомлень і передають результати координатору верхнього рівня.

Завдяки такому розподілу функцій агент прийняття рішень може формувати обґрунтовані рішення на основі агрегованих сигналів, спрямованих на мінімізацію ризику та зменшення невизначеності щодо характеру потенційної атаки.

Вибір дій агентами формалізується через відображення стану середовища $\sigma(\tau)$ на простір дій α_τ з множини: $A = T \cup I$, де кожна дія нижчого рівня стосується лише підмножини індикаторів I_{i}, а дії агента вищого рівня координують послідовність запитів і формування підсумкової оцінки типу загрози. У кожному циклі взаємодії агент з прийняття рішень отримує стан середовища, $\sigma(\tau)$ який відображає сукупність усіх зібраних індикаторів потенційних атак соціальної інженерії на цей момент.

На основі політики ρ_0 $(\alpha_0\tau/\sigma(\tau)$ агент вищого рівня приймає рішення щодо подальшого курсу дій: продовжувати збір ознак через запити до користувача чи системних датчиків, або формувати остаточну оцінку ймовірного типу загрози.

Простір дій агента прийняття рішень визначається множиною:

$$A_0 = \{W_1, W_2, \dots, W_K, C\}, \tag{2.13}$$

де W_{i} позначає сервісного агента, відповідального за аналіз певної підмножини індикаторів I_{i}, а C відповідає модулю класифікації атак.

Коли активовано одного з сервісних агентів, ініціюється підзадача, в рамках якої $\tau \in [\tau_0, \tau_0 + L]$ той самий сервісний агент виконує наступні кроки збору ознак, де L визначає максимальну кількість ітерацій для підзадачі.

Поточна підзадача завершується після отримання конкретної відповіді на запит щодо індикатора, яка може набувати дискретних значень, наприклад «підтверджено» або «не підтверджено». Якщо активується модуль класифікації C, агент прийняття рішень формує підсумкове рішення щодо типу атаки, завершуючи поточний цикл взаємодії.

Агент прийняття рішень координує роботу сервісних агентів, оптимізуючи запити та процес класифікації, тоді як сервісні агенти виконують локальні підзадачі зі збирання ознак. Така організація зменшує обчислювальну складність

і підвищує точність виявлення складних сценаріїв. Архітектура забезпечує адаптивне коригування стратегій залежно від поточного стану, мінімізує невизначеність і підсилює стійкість розподіленої КС до атак соціальної інженерії.

Коли сервісного агента активовано, він отримує стан середовища $\sigma(\tau)$ та витягує відповідні індикатори υ_τ, що відповідають призначеній йому підгрупі загроз:

$$\upsilon_\tau = [\chi_1, \chi_2, \dots, \chi_{|I_I|}, \tau, \varepsilon]. \quad (2.14)$$

Кожен сервісний агент працює в межах локального підпростору ознак, що дає йому змогу зосереджуватися на окремих категоріях атак соціальної інженерії та відповідних поведінкових або технічних сигналах.

На основі локальної політики, сформованої компонентом менеджера взаємодій $\rho_i(\alpha_i\tau/\upsilon_\tau)$, сервісний агент обирає наступний індикатор для уточнення або перевірки та виконує дії зі збирання додаткової інформації, яка уточнює ймовірність конкретного типу атаки.

Простір дій кожного сервісного агента MAC визначається множиною:

$$A_i = \upsilon_1, \upsilon_2, \dots, \upsilon_{|I_i|}, \quad (2.15)$$

де кожен елемент відповідає певному індикатору або шаблону. Така локалізація дій дає змогу сервісним агентам поступово накопичувати знання щодо конкретної категорії загроз. Ієрархічна взаємодія з агентом прийняття рішень дозволяє агрегувати ці локальні результати та коригувати загальну політику системи, підвищуючи адаптивність та стійкість РКС до атак соціальної інженерії.

Агент прийняття рішень отримує зовнішню винагороду $\omega_e\tau$ на кожному циклі взаємодії, яка відображає ефективність прийнятих рішень у виявленні потенційних атак соціальної інженерії. Після виконання підзавдання, агент прийняття рішень розраховує накопичену винагороду $\Omega_0\tau$ за даний цикл, враховуючи внесок кожного сервісного агента та результати локальної оцінки показників.

Нехай K позначає кількість сервісних агентів, а $\lambda \in [0,1]$ – коефіцієнт дисконтування, який зменшує вагу віддалених у часі результатів. Тоді накопичена винагорода визначається як:

$$\Omega_0\tau = \begin{cases} \sum_{\kappa=1}^{L} \lambda^{\kappa}\omega^{e}_{(\tau+\kappa)}, \text{якщо } \alpha_0\ \tau = W_{\text{і}}; \\ \omega^{e}_{(\tau)}, \text{якщо } \alpha_0\tau = C. \end{cases}, \quad (2.16)$$

де $W_{\text{і}}$ – сервісний агент, відповідальний за підзадачу аналізу підмножини індикаторів $I_{\text{і}}$, C – модуль класифікації типів атак. Така формалізація дає змогу агенту прийняття рішень інтегрувати результати локального аналізу, оцінювати ефективність дій на всіх рівнях та адаптивно коригувати політику збору даних і прийняття рішень. Використання коефіцієнта дисконту λ забезпечує баланс між короткостроковими та довгостроковими наслідками.

Мета функції $L(\Theta_0)$ визначається через дискретну динаміку навчання, що ґрунтується на підкріплювальному сигналі та історії взаємодій. Нехай σ' – наступний стан середовища після виконання дії $\alpha_0{}'$, $\Theta_0{}'$ – параметри цільової мережі, Θ_0 – параметри поточної мережі та B_0 – буфер досвіду агента прийняття рішень.

Тоді функція втрат визначається як:

$$L(\Theta_0) = E_{\sigma,\alpha_0,\Omega_0,\sigma'\sim B_0}((\Omega_0 + \lambda max_{\alpha_0'} Q_0(\sigma', \alpha_0'; \Theta_0') - Q_0(\sigma', \alpha_0; \Theta_0))^2), \quad (2.17)$$

де $Q_0(\sigma, \alpha_0; \Theta_0)$ – оцінка очікуваної кумулятивної винагороди згідно з поточною політикою, $\lambda max_{\alpha_0'} Q^0(\sigma', \alpha^{0\prime}; \Theta^{0\prime})$ забезпечує врахування довгострокових наслідків дій. Така формалізація дає змогу агенту прийняття рішень ефективно інтегрувати інформацію від сервісних агентів та модуля класифікації, оцінюючи як короткострокові, так і довгострокові наслідки своїх рішень.

Сервісні агенти отримують внутрішню винагороду $\omega_i\tau$ на кожному циклі взаємодії, яка оцінює ефективність їх локальної роботи з підмножиною індикаторів $I_{\text{і}}$, закріпленою за певною категорією атак соціальної інженерії.

Метою сервісних агентів є максимізація цих винагород, що забезпечує точне та ефективне виконання підзадач зі збору інформації.

Функція втрат кожного сервісного агента $L(\Theta_{\text{і}})$ визначається на основі локальної політики та історії взаємодії в буфері відтворення B_i, де Θ_i' – параметри цільової мережі, $\Theta_{\text{і}}$ – параметри поточної мережі:

$$L(\Theta_i) = E_{\sigma_i,\alpha_i,\Omega_i,\sigma' \sim B_i}((\Omega_i + \lambda max_{\alpha_i'} Q_i(\sigma_i', \alpha_i'; \Theta_i') - Q_0(\sigma_i, \alpha_i; \Theta_i))^2), \quad (2.18)$$

де $Q_{\text{і}}(\upsilon_{\text{і}}, \alpha_{\text{і}}; \Theta_{\text{і}})$ – очікувана кумулятивна винагорода за поточною локальною політикою, $\lambda max_{\alpha_i'} Q_i(\upsilon_i', \alpha_i'; \Theta_i')$ – довгострокові наслідки дій агентів.

Така формалізація дає змогу сервісним агентам адаптивно коригувати власну стратегію збору ознак, зосереджуючись на локальних підзадачах, і поступово підвищувати точність оцінювання ризиків для конкретних категорій атак. Координація з агентом прийняття рішень дає можливість агрегувати локальні результати та забезпечує оптимізацію загальної політики системи.

2.2.3 Розрахунок винагороди

У побудованій архітектурі РКС, починаючи з початкового стану σ_0, для основного класифікатора загроз та допоміжних класифікаторів, що спеціалізуються на окремих категоріях атак соціальної інженерії, формуються початкові ймовірнісні розподіли. Для агента прийняття рішень позначимо цей розподіл як $\phi_0^D = [\phi_1, \phi_2, \dots, \phi_k]$, де кожен компонент ϕ_i – апостеріорна оцінка належності поточного сценарію до певного класу загроз. Для сервісних агентів визначимо аналогічний розподіл $\phi_0^S = [\phi_1, \phi_2, \dots, \phi_j]$, де кожен елемент представляє ймовірність виявлення відповідної підкатегорії атаки в межах множини можливих дій A.

Важливою характеристикою цих розподілів є ентропія, яка відображає ступінь невизначеності під час прийняття рішень. Початкова ентропія агента прийняття рішень, що працює на рівні інтегральної оцінки загрози, визначається виразом:

$$J^D(\sigma_0) = -\sum_{i=1}^{\kappa} - \phi_i \log(\phi_i), \quad (2.19)$$

тоді як початкова ентропія сервісних агентів розраховується за формулою:

$$J^S(\sigma_0) = -\sum_{i=1}^{|A_i|} \phi_i \log(\phi_i). \quad (2.20)$$

Обидві метрики ентропії слугують нормалізаторами, запобігаючи зміщенням у винагороді між епізодами, що важливо за суттєвих відмінностей у характері вхідної інформації. Використання ентропії як керувального параметра дає багатоагентній системі змогу адаптивно балансувати між точністю виявлення атак і стійкістю до інформаційного шуму, характерного для реальних РКС.

Аналогічно, у момент часу τ обчислюється інформаційна ентропія основного класифікатора агента прийняття рішення та $J^D(\sigma_\tau)$ класифікатора $J^S(\sigma_\tau)$ агента обслуговування . Після того, як система виконує дію та переходить у новий стан $\sigma_{\tau+1}$, оновлені значення ентропії визначаються як $J^D(\sigma_{\tau+1})$ та $J^S(\sigma_{\tau+1})$. Різниця між цими значеннями стає основою для розрахунку винагороди, яка характеризує зменшення невизначеності в роботі агента прийняття рішення та агентів обслуговування.

Для агента прийняття рішення значення винагороди визначається співвідношенням:

$$\rho^D_{\text{entropy}} = max(\frac{(J^D(\sigma_\tau) - J^D(\sigma_{\tau+1}))}{J^H}(\sigma_0), 0). \quad (2.21)$$

Розмір винагороди сервісних агентів визначається за формулою:

$$\rho^S_{\text{entropy}} = max(\frac{(J^S(\sigma_\tau) - J^S(\sigma_{\tau+1}))}{J^S}(\sigma_0), 0). \quad (2.22)$$

У цих виразах відношення різниці ентропій до початкового значення виконує нормалізацію функціонала, що дає змогу коректно порівнювати інформаційний приріст між епізодами. Використання оператора max унеможливлює появу від'ємних значень, які можуть виникати за неповних або зашумлених даних, забезпечуючи стабільне навчання в умовах підвищеної невизначеності, характерної для атак соціальної інженерії. За такої структури винагороди багатоагентна система отримує механізм самоадаптації,

орієнтований на мінімізацію ентропії стану та формування однозначного діагностичного висновку щодо наявності індикаторів атаки.

Зовнішню функцію винагороди ρ_{t}^{ext} визначимо наступним чином:

$$\rho_{\mathrm{t}}^{ext} = \sum_{\iota=1}^{n} N + \rho_{\mathrm{entropy}}^{D}, \tag{2.23}$$

де N є додатним числом, якщо агент прийняття рішень видає остаточне рішення, і це рішення є правильним (результат «успіх»); N є від'ємним числом, якщо вибрана дія на кроці $t_{,}$ повторює попередньо виконану дію в тому ж епізоді (виявлено повторення дії, надлишкові запити караються), або якщо досягнуто максимально дозволеної кількості кроків взаємодії без остаточного рішення (бюджет епізоду вичерпано, нерішучість карається); N дорівнює 0 у всіх інших випадках (взаємодія триває, винагорода пропорційна зменшенню невизначеності, додаткова вигода від інформативних дій, що зменшують невизначеність, масштабується) $\rho_{\mathrm{entropy}}^{D} > 0$.

2.2.4 Внутрішня винагорода для сервісних агентів системи

Внутрішня винагорода $\rho_{\mathrm{t}}^{\mathrm{int}}$ для сервісного агента визначається наступним чином:

$$\rho_{\mathrm{t}}^{\mathrm{int}} = \sum_{\iota=1}^{n} N + \rho_{\mathrm{entropy}}^{S}, \tag{2.24}$$

де N є додатним числом, якщо запит сервісного агента дає остаточне та правильне рішення цільового індикатора або ознаки (спостерігається «правильий збіг»); N є від'ємним числом, якщо срвісний агент видає повторний або надлишковий запит, який не приносить нової інформації (повторення дій); N дорівнює 0 у всіх інших випадках (поступове зменшення невизначеності без остаточного збігу) $\rho_{\mathrm{entropy}}^{S} > 0$.

Максимальна кількість кроків взаємодії визначає тривалість епізоду й використовується як умова тайм-ауту зовнішньої винагороди. Нормалізація за початковою ентропією забезпечує порівнюваність винагород у сценаріях з різним рівнем невизначеності. Результат «успіх» узгоджує поведінку агента верхнього рівня з вимогою точної класифікації, тоді як локальний «правильний

збіг» спрямовує сервісних агентів на точне уточнення ознак. Штрафи за повторення дій виконують роль регуляризаторів, що запобігають циклам і надмірним запитам. У сукупності така структура збалансовує прагнення до швидкої взаємодії з необхідністю вимірюваного зменшення невизначеності.

Ієрархічна система винагород узгоджує розподіл функцій між рівнями: агент ухвалення рішень координує процес оцінювання, а сервісні агенти виконують роль локальних фільтрів. Це забезпечує багаторівневу стійкість РКС до атак соціальної інженерії та підтримує адаптацію системи в динамічних умовах.

Окрім винагороди, що базується на зменшенні невизначеності ентропії, основний агент отримує додаткові підкріплення, які генеруються залежно від результату прийнятого рішення. Якщо остаточна класифікація сценарію атаки успішна, то застосовується позитивний приріст винагороди, який позначається як σ^+. У разі некоректної ідентифікації або досягнення максимально допустимої кількості ітерацій діалогу вводиться штрафна складова σ^-, яка відображає неефективність стратегії агента.

Для сервісних агентів, що відповідають за локальне уточнення ознак, винагорода визначається за іншим принципом. Якщо обраний сервісний агент вірно уточнює ознаку, що характеризує можливу атаку, та отримує однозначну відповідь від середовища, то він отримує позитивний приріст τ^+. Якщо відбувається повторний запит, який не зменшує ентропію інформаційного стану та призводить до інформаційного шуму, то вводиться негативний штраф τ^-.

Формалізація сигналів підкріплення на різних рівнях архітектури системи поєднує глобальну оцінку продуктивності агента прийняття рішень з локальною точністю роботи сервісних агентів. Використання параметрів σ^+, σ^-, τ^+ та τ^- дозволяє збалансувати стратегічні та тактичні аспекти функціонування системи, створюючи умови для адаптивного навчання в динамічному середовищі.

2.2.5 Побудова знань для компонента системи менеджера взаємодії

У запропонованій РКС знання про сценарії соціальної інженерії подаються у вигляді графа знань, інтегрованого в менеджер взаємодій. Вузли графа відповідають індикаторам та класам атак, а ребра відображають кореляційні зв'язки між ними. Кожному ребру призначається вага, що характеризує силу такого зв'язку і визначається на основі ймовірнісної статистики, отриманої з навчальних і тестових сценаріїв.

Оскільки система використовує багатоагентну структуру, сервісні агенти відповідають за обробку підмножин можливих індикаторів та класів атак. Для кожного такого агента будується окремий підграф знань, що містить набір індикаторів O_i та набір атак A_i, що стосуються його підсистеми. Це забезпечує локалізацію обчислень та зменшує загальну складність синтезу архітектури, оскільки окремі агенти спеціалізуються на певних класах загроз.

Вагові коефіцієнти ребер у підграфах трактуються як умовні ймовірності появи певної ознаки за умови реалізації відповідного класу атаки. Таку залежность представлено у формі наступного рівняння:

$$q(o_u a_v) = \frac{m(a_v, o_u)}{\sum_{i=1}^{n} m(a_v, o_z)}, \tag{2.25}$$

де $m(a_v, o_u)$ – кількість зареєстрованих сценаріїв, a_v) – атака з проявом ознаки o_u.

Обчислюючи значення ймовірності для кожної пари «атака – ознака», побудовано матрицю умовних ймовірностей $\Lambda_i \in R^{|O_i| \times |A_i|}$, яка використовується в процесі навчання агента прийняття рішень. Це дає змогу відобразити багатовимірні залежності між типами атак соціальної інженерії та їх ознаками, забезпечуючи адаптивне навчання агентів під час взаємодії з потенційно ворожими сценаріями.

На основі матриці умовних ймовірностей, введеної вище, поточний розподіл ймовірностей проявів ознак для сервісних агентів визначається як:

$$\pi(o_u) = \Lambda_i \cdot \pi(a_v) \tag{2.26}$$

де $\pi(a_v) \in R^{|A_i|\times 1}$ вектор ймовірностей відповідних класів атак, який обчислюється підпорядкованим класифікатором для підгрупи i, та $\pi(o_u) \in R^{|O_i|\times 1}$– розподіл ймовірностей для локальних ознак, що стосуються цього агента. Ця операція перетворює ймовірності класів атак на ймовірності прояву конкретних ознак, забезпечуючи основу для прийняття рішень агентом.

Далі цей розподіл ймовірностей поєднується з оцінками Q значень, обчислених сервісними агентами, що дозволяє враховувати як попередні знання про ймовірності появи ознак, так і досвід навчання агента в динамічному середовищі. Формально вибір наступної дії агента виражається як:

$$\alpha_\tau = argmax_\omega(\sigma(Q_\omega(\sigma_\tau)) + \sigma(\pi(o_u))), \quad (2.27)$$

де $Q_\omega(\sigma_\tau)$ – розподіл значень Q, обчислених сервісним агентом для стану σ_τ а σ – сигмоїдна функція нормалізації, яка відображає значення в діапазон [0,1], функція argmax визначає клас ознак або дій, на якому агент зупиняє свій вибір для подальшого запиту або оцінки.

Поєднання ймовірностей та Q значень , дає агенту прийняття рішень змогу враховувати як попередній досвід, так і оновлену інформацію про поведінку користувача чи системи, що є важливим для протидії атакам соціальної інженерії в межах РКС.

Класифікатор атак працює на двох рівнях: на першому рівні він визначає, чи має вхідна інформація ознаки соціальної інженерії; на другому рівні уточнюється оцінка для визначення типу (класу) атаки та конкретної тактики, що застосовується ініціатором атаки.

Компонент механізму моделювання генерує відповідь, яка служить основою для обчислення сигналу винагороди ω_τ, яка повертається агенту. В результаті середовище переходить у новий стан $\sigma(\tau + 1)$, і таким чином розгортає послідовність ітерацій, що імітують реальну динаміку комунікації.

Мета агента прийняття рішень полягає в пошуку оптимальної політики, яка забезпечує максимізацію очікуваної кумулятивної винагороди. Винагорода ω_τ визначається функцією, $\Omega(\sigma(\tau), \alpha_\tau)$ де коефіцієнт $\lambda \in [0,1]$ служить дисконтним коефіцієнтом, що зменшує вагу віддалених у часі результатів.

Стан середовища описує множину зібраних індикаторів атаки, до яких можуть належати ознаки фішингових повідомлень, лінгвістичні маркери психологічного тиску, метадані електронної пошти або характерні патерни URL-адрес.

Визначимо A як множину типів атак соціальної інженерії, а I – множину індикаторів, пов'язаних з цими атаками. Тоді стан $\sigma(\tau)$ подамо як вектор $[\chi_1, \chi_2, \dots, \chi_{|I|}, \tau, \varepsilon]$, де кожен компонент χ_i є закодованим тернарним представленням наявності індикатора: підтверджена наявність, підтверджена відсутність або невизначений стан.

Вектор включає параметр τ, що відображає поточний крок діалогу, та змінну ε, яка сигналізує про повторення попередніх запитів. Ця формалізація дає змогу системі не лише накопичувати знання про зібрані індикатори, але й адаптивно керувати стратегією запитів, мінімізуючи інформаційну надмірність і підвищуючи точність виявлення атак соціальної інженерії в умовах неповної та суперечливої інформації.

Простір дій агента прийняття рішень формується як об’єднання двох підмножин, що представляють різні типи дій у процесі виявлення атак соціальної інженерії. Визначаємо $A = T \cup I$, де T – множина типів атак, I – множина індикаторів, які можна запитувати для уточнення контексту повідомлення або взаємодії. Якщо вибрана дія α_τ належить до I, агент ініціює запит додаткової інформації щодо конкретного індикатора. Отримана відповідь кодується у відповідний компонент вектора стану та уточнює оцінку сценарію на кроці τ. У разі вибору дії з підмножини T агент формує остаточне рішення щодо ймовірного типу атаки, що завершує процес взаємодії в рамках поточного сценарію .

Успішність процесу визначається точністю класифікації загрози, тобто коректністю віднесення взаємодії до відповідного класу соціальної інженерії.

Така формалізація дає змогу агенту збалансовувати збирання додаткових ознак і ухвалення рішень, підвищуючи ефективність роботи в умовах неповної чи суперечливої інформації. Ієрархічна організація системи забезпечує

раціональний розподіл обчислювальних ресурсів і сприяє послідовному зменшенню невизначеності під час оцінювання ризику, що є важливим для стійкості розподілених комп'ютерних систем до атак соціальної інженерії.

2.2.2 Синтез архітектури комп'ютерних систем, стійких до атак соціальної інженерії, на основі ієрархічної багатоагентної системи

Позначимо множину атак соціальної інженерії як множину, T поділену на K групи, де

$$T = T_1 \vee T_2 \vee \ldots \vee T_K, \tag{2.12}$$

підмножини не перетинаються, тобто $T_i \wedge T_j = \emptyset$ для $i \neq j, i, j = 1, \ldots, K$.

Кожна підмножина T_{i} відповідає певній категорії атак або тактичній стратегії порушника та пов'язана з власною множиною індикаторів I_{i}, які характеризують ознаки повідомлень, поведінкові патерни чи технічні сигнали, притаманні цій групі.

Побудована архітектура РКС передбачає два рівні:

1) агент з прийняття рішень вищого рівня (Decision Agent, DA, агент прийняття рішень, централний агент), який координує загальну політику збору інформації та прийняття рішень;

2) агенти обслуговування нижчого рівня або сервісні агенти (Service Agents, SA) відповідають кожен за власну підмножину атак T_{i}та пов'язані з нею індикатори I_{i} . Сервісні агенти виконують функції локального аналізу, ініціюють запити до користувачів або системних сенсорів, оцінюють атрибути повідомлень і передають результати координатору верхнього рівня.

Завдяки такому розподілу функцій агент прийняття рішень може формувати обґрунтовані рішення на основі агрегованих сигналів, спрямованих на мінімізацію ризику та зменшення невизначеності щодо характеру потенційної атаки.

Вибір дій агентами формалізується через відображення стану середовища $\sigma(\tau)$ на простір дій α_τ з множини: $A = T \cup I$, де кожна дія нижчого рівня

стосується лише підмножини індикаторів I_{i}, а дії агента вищого рівня координують послідовність запитів і формування підсумкової оцінки типу загрози. У кожному циклі взаємодії агент з прийняття рішень отримує стан середовища, $\sigma(\tau)$ який відображає сукупність усіх зібраних індикаторів потенційних атак соціальної інженерії на цей момент.

На основі політики ρ_0 $(\alpha_0\tau/\sigma(\tau)$ агент вищого рівня приймає рішення щодо подальшого курсу дій: продовжувати збір ознак через запити до користувача чи системних датчиків, або формувати остаточну оцінку ймовірного типу загрози.

Простір дій агента прийняття рішень визначається множиною:

$$A_0 = \{W_1, W_2, \dots, W_K, C\}, \tag{2.13}$$

де W_{i} позначає сервісного агента, відповідального за аналіз певної підмножини індикаторів I_{i}, а C відповідає модулю класифікації атак.

Коли активовано одного з сервісних агентів, ініціюється підзадача, в рамках якої $\tau \in [\tau_0, \tau_0 + L]$ той самий сервісний агент виконує наступні кроки збору ознак, де L визначає максимальну кількість ітерацій для підзадачі.

Поточна підзадача завершується після отримання конкретної відповіді на запит щодо індикатора, яка може набувати дискретних значень, наприклад «підтверджено» або «не підтверджено». Якщо активується модуль класифікації C, агент прийняття рішень формує підсумкове рішення щодо типу атаки, завершуючи поточний цикл взаємодії.

Агент прийняття рішень координує роботу сервісних агентів, оптимізуючи запити та процес класифікації, тоді як сервісні агенти виконують локальні підзадачі зі збирання ознак. Така організація зменшує обчислювальну складність і підвищує точність виявлення складних сценаріїв. Архітектура забезпечує адаптивне коригування стратегій залежно від поточного стану, мінімізує невизначеність і підсилює стійкість розподіленої КС до атак соціальної інженерії.

Коли сервісного агента активовано, він отримує стан середовища $\sigma(\tau)$ та витягує відповідні індикатори υ_τ, що відповідають призначеній йому підгрупі загроз:

$$\upsilon_\tau = [\chi_1, \chi_2, \dots, \chi_{|I_I|}, \tau, \varepsilon]. \quad (2.14)$$

Кожен сервісний агент працює в межах локального підпростору ознак, що дає йому змогу зосереджуватися на окремих категоріях атак соціальної інженерії та відповідних поведінкових або технічних сигналах.

На основі локальної політики, сформованої компонентом менеджера взаємодій $\rho_i(\alpha_i\tau/\upsilon_\tau)$, сервісний агент обирає наступний індикатор для уточнення або перевірки та виконує дії зі збирання додаткової інформації, яка уточнює ймовірність конкретного типу атаки.

Простір дій кожного сервісного агента MAC визначається множиною:

$$A_i = \upsilon_1, \upsilon_2, \dots, \upsilon_{|I_i|}, \quad (2.15)$$

де кожен елемент відповідає певному індикатору або шаблону. Така локалізація дій дає змогу сервісним агентам поступово накопичувати знання щодо конкретної категорії загроз. Ієрархічна взаємодія з агентом прийняття рішень дозволяє агрегувати ці локальні результати та коригувати загальну політику системи, підвищуючи адаптивність та стійкість РКС до атак соціальної інженерії.

Агент прийняття рішень отримує зовнішню винагороду $\omega_e\tau$ на кожному циклі взаємодії, яка відображає ефективність прийнятих рішень у виявленні потенційних атак соціальної інженерії. Після виконання підзавдання, агент прийняття рішень розраховує накопичену винагороду $\Omega_0\tau$ за даний цикл, враховуючи внесок кожного сервісного агента та результати локальної оцінки показників.

Нехай K позначає кількість сервісних агентів, а $\lambda \in [0,1]$ – коефіцієнт дисконтування, який зменшує вагу віддалених у часі результатів. Тоді накопичена винагорода визначається як:

$$\Omega_0\tau = \begin{cases} \sum_{\kappa=1}^{L} \lambda^\kappa \omega^e_{(\tau+\kappa)}, \text{якщо } \alpha_0\ \tau = W_\text{i}; \\ \omega^e_{(\tau)}, \text{якщо } \alpha_0\tau = C. \end{cases}, \quad (2.16)$$

де $W_{\text{і}}$ – сервісний агент, відповідальний за підзадачу аналізу підмножини індикаторів $I_{\text{і}}$, C – модуль класифікації типів атак. Така формалізація дає змогу агенту прийняття рішень інтегрувати результати локального аналізу, оцінювати ефективність дій на всіх рівнях та адаптивно коригувати політику збору даних і прийняття рішень. Використання коефіцієнта дисконту λ забезпечує баланс між короткостроковими та довгостроковими наслідками.

Мета функції $L(\Theta_0)$ визначається через дискретну динаміку навчання, що ґрунтується на підкріплювальному сигналі та історії взаємодій. Нехай σ' – наступний стан середовища після виконання дії $\alpha_0{}'$, $\Theta_0{}'$ – параметри цільової мережі, Θ_0 – параметри поточної мережі та B_0 – буфер досвіду агента прийняття рішень.

Тоді функція втрат визначається як:

$$L(\Theta_0) = E_{\sigma,\alpha_0,\Omega_0,\sigma'\sim B_0}((\Omega_0 + \lambda max_{\alpha_0'} Q_0(\sigma', \alpha_0{}'; \Theta_0{}') - Q_0(\sigma', \alpha_0; \Theta_0))^2), \quad (2.17)$$

де $Q_0(\sigma, \alpha_0; \Theta_0)$ – оцінка очікуваної кумулятивної винагороди згідно з поточною політикою, $\lambda max_{\alpha_0'} Q^0(\sigma', \alpha^{0\prime}; \Theta^{0\prime})$ забезпечує врахування довгострокових наслідків дій. Така формалізація дає змогу агенту прийняття рішень ефективно інтегрувати інформацію від сервісних агентів та модуля класифікації, оцінюючи як короткострокові, так і довгострокові наслідки своїх рішень.

Сервісні агенти отримують внутрішню винагороду $\omega_i \tau$ на кожному циклі взаємодії, яка оцінює ефективність їх локальної роботи з підмножиною індикаторів $I_{\text{і}}$, закріпленою за певною категорією атак соціальної інженерії.

Метою сервісних агентів є максимізація цих винагород, що забезпечує точне та ефективне виконання підзадач зі збору інформації.

Функція втрат кожного сервісного агента $L(\Theta_{\text{і}})$ визначається на основі локальної політики та історії взаємодії в буфері відтворення B_i, де $\Theta_i{}'$ – параметри цільової мережі, $\Theta_{\text{і}}$ – параметри поточної мережі:

$$L(\Theta_i) = E_{\sigma_i,\alpha_i,\Omega_i,\sigma'\sim B_i}((\Omega_i + \lambda max_{\alpha_i'} Q_i(\sigma_i{}', \alpha_i{}'; \Theta_i{}') - Q_0(\sigma_i, \alpha_i; \Theta_i))^2), \quad (2.18)$$

де $Q_{\text{і}}(\upsilon_{\text{і}}, \alpha_{\text{і}}; \Theta_{\text{і}})$ – очікувана кумулятивна винагорода за поточною локальною політикою, $\lambda max_{\alpha_i'} Q_i(\upsilon_i', \alpha_i'; \Theta_i')$ – довгострокові наслідки дій агентів.

Така формалізація дає змогу сервісним агентам адаптивно коригувати власну стратегію збору ознак, зосереджуючись на локальних підзадачах, і поступово підвищувати точність оцінювання ризиків для конкретних категорій атак. Координація з агентом прийняття рішень дає можливість агрегувати локальні результати та забезпечує оптимізацію загальної політики системи.

2.2.3 Розрахунок винагороди

У побудованій архітектурі РКС, починаючи з початкового стану σ_0, для основного класифікатора загроз та допоміжних класифікаторів, що спеціалізуються на окремих категоріях атак соціальної інженерії, формуються початкові ймовірнісні розподіли. Для агента прийняття рішень позначимо цей розподіл як $\phi_0^D = [\phi_1, \phi_2, \dots, \phi_k]$, де кожен компонент ϕ_i – апостеріорна оцінка належності поточного сценарію до певного класу загроз. Для сервісних агентів визначимо аналогічний розподіл $\phi_0^S = [\phi_1, \phi_2, \dots, \phi_j]$, де кожен елемент представляє ймовірність виявлення відповідної підкатегорії атаки в межах множини можливих дій A.

Важливою характеристикою цих розподілів є ентропія, яка відображає ступінь невизначеності під час прийняття рішень. Початкова ентропія агента прийняття рішень, що працює на рівні інтегральної оцінки загрози, визначається виразом:

$$J^D(\sigma_0) = -\sum_{i=1}^{\kappa} - \phi_i \log(\phi_i), \tag{2.19}$$

тоді як початкова ентропія сервісних агентів розраховується за формулою:

$$J^S(\sigma_0) = -\sum_{i=1}^{|A_i|} \phi_i \log(\phi_i). \tag{2.20}$$

Обидві метрики ентропії слугують нормалізаторами, запобігаючи зміщенням у винагороді між епізодами, що важливо за суттєвих відмінностей у характері вхідної інформації. Використання ентропії як керувального параметра дає багатоагентній системі змогу адаптивно балансувати між точністю

виявлення атак і стійкістю до інформаційного шуму, характерного для реальних РКС.

Аналогічно, у момент часу τ обчислюється інформаційна ентропія основного класифікатора агента прийняття рішення та $J^D(\sigma_\tau)$ класифікатора $J^S(\sigma_\tau)$ агента обслуговування . Після того, як система виконує дію та переходить у новий стан $\sigma_{\tau+1}$, оновлені значення ентропії визначаються як $J^D(\sigma_{\tau+1})$ та $J^S(\sigma_{\tau+1})$. Різниця між цими значеннями стає основою для розрахунку винагороди, яка характеризує зменшення невизначеності в роботі агента прийняття рішення та агентів обслуговування.

Для агента прийняття рішення значення винагороди визначається співвідношенням:

$$\rho^D_{\text{entropy}} = max(\frac{(J^D(\sigma_\tau) - J^D(\sigma_{\tau+1}))}{J^H}(\sigma_0),0). \quad (2.21)$$

Розмір винагороди сервісних агентів визначається за формулою:

$$\rho^S_{\text{entropy}} = max(\frac{(J^S(\sigma_\tau) - J^S(\sigma_{\tau+1}))}{J^S}(\sigma_0),0). \quad (2.22)$$

У цих виразах відношення різниці ентропій до початкового значення виконує нормалізацію функціонала, що дає змогу коректно порівнювати інформаційний приріст між епізодами. Використання оператора max унеможливлює появу від’ємних значень, які можуть виникати за неповних або зашумлених даних, забезпечуючи стабільне навчання в умовах підвищеної невизначеності, характерної для атак соціальної інженерії. За такої структури винагороди багатоагентна система отримує механізм самоадаптації, орієнтований на мінімізацію ентропії стану та формування однозначного діагностичного висновку щодо наявності індикаторів атаки.

Зовнішню функцію винагороди ρ^{ext}_{t} визначимо наступним чином:

$$\rho^{ext}_{\text{t}} = \sum_{i=1}^{n} N + \rho^D_{\text{entropy}}, \quad (2.23)$$

де N є додатним числом, якщо агент прийняття рішень видає остаточне рішення, і це рішення є правильним (результат «успіх»); N є від'ємним числом, якщо вибрана дія на кроці t, повторює попередньо виконану дію в тому ж епізоді (виявлено повторення дії, надлишкові запити караються), або якщо досягнуто

максимально дозволеної кількості кроків взаємодії без остаточного рішення (бюджет епізоду вичерпано, нерішучість карається); N дорівнює 0 у всіх інших випадках (взаємодія триває, винагорода пропорційна зменшенню невизначеності, додаткова вигода від інформативних дій, що зменшують невизначеність, масштабується) $\rho^{D}_{\text{entropy}} > 0$.

2.2.4 Внутрішня винагорода для сервісних агентів системи

Внутрішня винагорода $\rho_{\text{t}}^{\text{int}}$ для сервісного агента визначається наступним чином:

$$\rho_{\text{t}}^{\text{int}} = \sum_{i=1}^{n} N + \rho^{S}_{\text{entropy}}, \tag{2.24}$$

де N є додатним числом, якщо запит сервісного агента дає остаточне та правильне рішення цільового індикатора або ознаки (спостерігається «правильий збіг»); N є від'ємним числом, якщо срвісний агент видає повторний або надлишковий запит, який не приносить нової інформації (повторення дій); N дорівнює 0 у всіх інших випадках (поступове зменшення невизначеності без остаточного збігу) $\rho^{S}_{\text{entropy}} > 0$.

Максимальна кількість кроків взаємодії визначає тривалість епізоду й використовується як умова тайм-ауту зовнішньої винагороди. Нормалізація за початковою ентропією забезпечує порівнюваність винагород у сценаріях з різним рівнем невизначеності. Результат «успіх» узгоджує поведінку агента верхнього рівня з вимогою точної класифікації, тоді як локальний «правильний збіг» спрямовує сервісних агентів на точне уточнення ознак. Штрафи за повторення дій виконують роль регуляризаторів, що запобігають циклам і надмірним запитам. У сукупності така структура збалансовує прагнення до швидкої взаємодії з необхідністю вимірюваного зменшення невизначеності.

Ієрархічна система винагород узгоджує розподіл функцій між рівнями: агент ухвалення рішень координує процес оцінювання, а сервісні агенти виконують роль локальних фільтрів. Це забезпечує багаторівневу стійкість РКС

до атак соціальної інженерії та підтримує адаптацію системи в динамічних умовах.

Окрім винагороди, що базується на зменшенні невизначеності ентропії, основний агент отримує додаткові підкріплення, які генеруються залежно від результату прийнятого рішення. Якщо остаточна класифікація сценарію атаки успішна, то застосовується позитивний приріст винагороди, який позначається як σ^+. У разі некоректної ідентифікації або досягнення максимально допустимої кількості ітерацій діалогу вводиться штрафна складова σ^-, яка відображає неефективність стратегії агента.

Для сервісних агентів, що відповідають за локальне уточнення ознак, винагорода визначається за іншим принципом. Якщо обраний сервісний агент вірно уточнює ознаку, що характеризує можливу атаку, та отримує однозначну відповідь від середовища, то він отримує позитивний приріст τ^+. Якщо відбувається повторний запит, який не зменшує ентропію інформаційного стану та призводить до інформаційного шуму, то вводиться негативний штраф τ^-.

Формалізація сигналів підкріплення на різних рівнях архітектури системи поєднує глобальну оцінку продуктивності агента прийняття рішень з локальною точністю роботи сервісних агентів. Використання параметрів σ^+, σ^-, τ^+ та τ^- дозволяє збалансувати стратегічні та тактичні аспекти функціонування системи, створюючи умови для адаптивного навчання в динамічному середовищі.

2.2.5 Побудова знань для компонента системи менеджера взаємодії

У запропонованій РКС знання про сценарії соціальної інженерії подаються у вигляді графа знань, інтегрованого в менеджер взаємодій. Вузли графа відповідають індикаторам та класам атак, а ребра відображають кореляційні зв’язки між ними. Кожному ребру призначається вага, що характеризує силу такого зв’язку і визначається на основі ймовірнісної статистики, отриманої з навчальних і тестових сценаріїв.

Оскільки система використовує багатоагентну структуру, сервісні агенти відповідають за обробку підмножин можливих індикаторів та класів атак. Для кожного такого агента будується окремий підграф знань, що містить набір індикаторів O_i та набір атак A_i, що стосуються його підсистеми. Це забезпечує локалізацію обчислень та зменшує загальну складність синтезу архітектури, оскільки окремі агенти спеціалізуються на певних класах загроз.

Вагові коефіцієнти ребер у підграфах трактуються як умовні ймовірності появи певної ознаки за умови реалізації відповідного класу атаки. Таку залежность представлено у формі наступного рівняння:

$$q(o_u a_v) = \frac{m(a_v, o_u)}{\sum_{i=1}^{n} m(a_v, o_z)}, \quad (2.25)$$

де $m(a_v, o_u)$ – кількість зареєстрованих сценаріїв, a_v) – атака з проявом ознаки o_u.

Обчислюючи значення ймовірності для кожної пари «атака – ознака», побудовано матрицю умовних ймовірностей $\Lambda_i \in R^{|O_i| \times |A_i|}$, яка використовується в процесі навчання агента прийняття рішень. Це дає змогу відобразити багатовимірні залежності між типами атак соціальної інженерії та їх ознаками, забезпечуючи адаптивне навчання агентів під час взаємодії з потенційно ворожими сценаріями.

На основі матриці умовних ймовірностей, введеної вище, поточний розподіл ймовірностей проявів ознак для сервісних агентів визначається як:

$$\pi(o_u) = \Lambda_i \cdot \pi(a_v) \quad (2.26)$$

де $\pi(a_v) \in R^{|A_i| \times 1}$ вектор ймовірностей відповідних класів атак, який обчислюється підпорядкованим класифікатором для підгрупи i, та $\pi(o_u) \in R^{|O_i| \times 1}$– розподіл ймовірностей для локальних ознак, що стосуються цього агента. Ця операція перетворює ймовірності класів атак на ймовірності прояву конкретних ознак, забезпечуючи основу для прийняття рішень агентом.

Далі цей розподіл ймовірностей поєднується з оцінками Q значень, обчислених сервісними агентами, що дозволяє враховувати як попередні знання про ймовірності появи ознак, так і досвід навчання агента в динамічному середовищі. Формально вибір наступної дії агента виражається як:

$$\alpha_\tau = argmax_\omega(\sigma(Q_\omega(\sigma_\tau)) + \sigma(\pi(o_u))), \quad (2.27)$$

де $Q_\omega(\sigma_\tau)$ – розподіл значень Q, обчислених сервісним агентом для стану σ_τ а σ – сигмоїдна функція нормалізації, яка відображає значення в діапазон [0,1], функція argmax визначає клас ознак або дій, на якому агент зупиняє свій вибір для подальшого запиту або оцінки.

Поєднання ймовірностей та Q значень , дає агенту прийняття рішень змогу враховувати як попередній досвід, так і оновлену інформацію про поведінку користувача чи системи, що є важливим для протидії атакам соціальної інженерії в межах РКС.

2.2.5 Дослідження ефективності архітектури розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії

Для оцінки ефективності запропонованої ієрархічної мультиагентної архітектури було проаналізовано чотири аспекти: точність класифікації порівняно з одноагентними та неієрархічними моделями; ефективність, що проявляється у скороченні тривалості взаємодії завдяки ентропійно керованим запитам; здатність до узагальнення між різними каналами соціальної інженерії; а також абляційний внесок ієрархії, графових апріорних знань і ентропійної структури винагороди в загальну продуктивність системи.

Експерименти виконано в середовищі Python 3.13 із використанням фреймворку PyTorch 2.x [177], оптимізованого для сучасних GPU. Обчислення на графічних прискорювачах здійснювалися за допомогою NVIDIA CUDA 13 та cuDNN 9 [178]. Оброблення текстових і веб-даних проводилося з використанням бібліотек BeautifulSoup4 та lxml, а відтворення вебінтерфейсів – за допомогою Playwright 1.50 [179] у headless-режимі. Навчання та тестування моделей здійснювалися на GPU NVIDIA GeForce RTX 5090 із підтримкою TensorRT – LLM для оптимізованого інференсу.

Агент прийняття рішень реалізований як верхньорівневий контролер на основі Deep Q-Network із окремою цільовою мережею [180]. Коефіцієнт

дисконтування було встановлено на 0,99. Використано буфер відтворення досвіду обсягом 200 000 переходів, розмір параметру mini-batch 256 та оптимізатор Adam зі швидкістю навчання 3×10^{-4} . Дослідження проводилося за ε – евристичною стратегією, у межах якої параметр ε лінійно зменшувався від 0,20 до 0,01 протягом 200 000 кроків взаємодії з середовищем.

Сервісні агенти реалізовано як спеціалізовані модулі для опрацювання електронної пошти та веб-модальності [174]. Ієрархичний контроль забезпечує вибір відповідного сервісного агента та момент завершення взаємодії, причому бюджет кроків обмежено п’ятьма діями для кожного прикладу.

Попередні знання інтегруються через ініціалізацію логітів сервісних агентів ймовірностями $P(feature \mid attack)$, отриманими з графа знань; дії агента прийняття рішень, що суперечать цим пріорним залежностям, маскуються. Цільова функція навчання доповнюється компонентом, спрямованим на зменшення предиктивної ентропії, та штрафами за повторювані або малозмістовні дії, що стимулює формування коротких і інформативних послідовностей взаємодії.

У експериментальному дослідженні використано мультимодальний корпус, що охоплює електронні листи та вебсторінки. Кожен елемент корпусу анотовано відповідно до типу атаки: вішинг (vishing), фішинг (phishing), клонування профілю (profile cloning), грумінг (grooming), сміттєве збирання (dumpster diving), підхід упритул до об’єкта доступу (tailgating), маскування файлів (file masquerading), приманка (baiting), оманливе лякальне ПЗ (scareware), оманливі спливаючі вікна (deceptive pop-ups), зараження довірених ресурсів (watering hole attack), троянські листи (trojan emails), цільовий фішинг (spear phishing), спам-листи (spam) та hoax-повідомлення (hoaxing).

Для атак, що не ґрунтуються на фізичних сенсорних спостереженнях, зокрема підходу упритул (tailgating) чи сміттєвого збирання (dumpster diving), ознаки подано у формі текстових описів у листах або на веб-сторінках, а не як вимірювання фізичного середовища. Набір даних містить 80 000 зразків: 48 000

електронних листів (приблизно 24 000 шкідливих та 24 000 легітимних) і 32 000 веб-сторінок (приблизно 16 000 шкідливих та 16 000 легітимних).

Розглянемо розподіл на навчання, перевірку та тестування. Корпус було поділено на навчальну, валідаційну та тестову вибірки у пропорціях 70%, 10% та 20% відповідно. Розподіл здійснювався у хронологічному порядку, коли часові позначки були доступні, так, щоб навчальна вибірка передувала валідаційній та тестовій у часі.

Для зменшення інформаційного витоку забезпечено роздільність доменів інтернет-ресурсів у вебсторінках, а також відокремленість авторів або ідентичностей відправників в електронних листах.

Близькі або майже дублікативні зразки було усунено між вибірками за допомогою гешування текстових подібностей [181] (SimHash) для текстового контенту, перцептивного гешування для візуальних представлень, а також нормалізації URL – адрес для усунення поверхневих варіацій.

Було виконано автоматичне визначення мови та маскування персонально ідентифікованої інформації, а для вебсторінок – розв’язання перенаправлень з подальшим вилученням кінцевого контенту. Додатково сформовано структурні токени, що відображають поведінкові характеристики сторінки, зокрема операції з формами та цілі гіперпосилань. Такі процедури уніфікують вхідні дані, зменшують хибні кореляції та забезпечують узгоджене вилучення ознак у поштовій та вебмодальності.

Експериментальні результати на об’єднаному корпусі електронних листів і вебсторінок показали, що ієрархічна архітектура з агентом прийняття рішень, який координує вибір модальності сервісного агента, забезпечує вищу точність, кращі робочі характеристики та меншу кількість кроків порівняно з неієрархічними моделями за однакового бюджету взаємодії.

У таблиці 2.1 наведено площу під кривою характеристик приймача (AUC–ROC), площу під кривою precision–recall (AUC–PR), F1-міру, частку хибнопозитивних спрацювань при 95% істиннопозитивній частці, кількість дій на зразок та медіанну наскрізну затримку (end-to-end latency) у мілісекундах.

Таблиця 2.1

Бінарне виявлення

Рішення	Площа під ROC–кривою	Площа під кривою PR	F1–оцінка	Рівень хибнопозитивних результатів	кількість кроків взаємодії	Затримка (мс)
Система на основі правил	0.78	0.62	0.65	0.42	–	–
Найкраща одномодальна модель	0.90	0.85	0.84	0.18	–	–
Одноагентна DQN – модель	0.92	0.88	0.86	0.15	6.1	210
Вихідне ансамблювання моделей	0.93	0.90	0.88	0.13	–	–
Запропонована архітектура	0.96±0.01	0.94 ± 0.01	0.92 ± 0.01	0.08 ± 0.01	3.2 ± 0.2	155

У порівнянні з базовою моделлю у вигляді одноагентної Deep Q-Network, ієрархічний підхід підвищує площу під кривою характеристик приймача на +0.04, зменшує частку хибнопозитивних спрацювань при 95% істиннопозитивній частці на −0.07 та скорочує кількість дій на зразок приблизно на 47% (з 6.1 до 3.2), що демонструє одночасне покращення як точності, так і ефективності.

Також було проведено оцінювання багатокласової продуктивності системи відповідно до таксономії типів атак соціальної інженерії.

Оскільки деякі категорії трапляються відносно рідко, для коректного узагальнення результатів було використано макроусереднене значення метрики F1. Такий підхід забезпечує рівну вагу кожного класу, незалежно від частоти його появи в наборі даних.

Результати експериментів наведено окремо для двох модальностей, зокрема, електронної пошти та вебсередовища, що дозволяє порівняти ефективність класифікації у різних контекстах виявлення атак соціальної інженерії. Узагальнені показники подано в таблиці 2.2.

Ієрархічна політика, що керується головним агентом прийняття рішень і підтримується службовими агентами, ініціалізованими графом знань, забезпечує приріст від +9 до +10 пунктів за макроусередненою метрикою F1 порівняно з найкращими неієрархічними альтернативами. Це свідчить про більш надійне розпізнавання рідкісних тактик соціальної інженерії.

Таблиця 2.2

Результати для модальностей електронної пошти та вебсередовища

Модальність	Найкраща неієрархічна базова лінія	Запропонована ієрархічна структура (агент прийняття рішень + агенти обслуговування)	Різниця
Email	0.77	0.86 ± 0.01	+0.09
Web	0.74	0.84 ± 0.02	+0.10

Для того щоб ізолювати внесок саме ієрархічного управління, було проведено аналіз абляції, у межах якого всі інші компоненти, такі як енкодери, параметри оптимізації та архітектурні налаштування, залишалися незмінними.

Таким чином, здійснювалося порівняння повної системи з її одноагентною конфігурацією, що дозволило кількісно оцінити ефект ієрархічного керування. Таблиця 2.3 узагальнює вплив вилучення ієрархії на показники точності та ефективності системи.

Таблиця 2.3

Результати абляційного аналізу впливу ієрархії

Варіант	Площа під кривою ROC (AUC ROC)	Макроусереднене значення F1 (Macro–F1)	Кількість дій на зразок	Рівень хибних спрацювань при 95% TPR
Повна ієрархічна конфігурація з графовими пріорами та ентропійним механізмом винагороди	0.96	0.92	3.2	0.08
Без ієрархії (одноагентна конфігурація)	0.93	0.88	5.7	0.12

Ієрархічна структура, що поєднує агента прийняття рішень та службових агентів, є основним джерелом підвищення ефективності системи, забезпечуючи зменшення кількості дій на один зразок у 2,5 раза. Водночас спостерігається покращення показників продуктивності на суворих порогах роботи: рівень хибних спрацювань при 95% істинно позитивних спрацьовувань знижується з 0,12 до 0,08.

У завданнях бінарного виявлення та класифікації тактик соціальної інженерії на об’єднаному корпусі електронних листів і вебвзаємодій ієрархічна архітектура, у якій агент прийняття рішень координує модально специфічних службових агентів, послідовно перевершує одноагентні та інші неієрархічні моделі. Система із запропонованою архітектурою забезпечує вищу точність за меншої кількості кроків взаємодії та демонструє кращі робочі характеристики на практично значущих рівнях істинно позитивних спрацьовувань, що підтверджує ефективність ієрархічної координації агентів у синтезі розподіленої КС для виявлення атак соціальної інженерії.

2.3 Моделі атак соціальної інженерії

Для здійснення синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, на основі ієрархічної багатоагентної системи використовується множина моделей атак соціальної інженерії. Моделі атак соціальної інженерії можуть описують вплив зловмисника на поведінку людини, поведінка користувач під дією атак, а також множина дій, спрямована здійснення такого впливу на користувачі КС [182, 183].

Визначимо множину S як множину атак соціальної інженерії:

$$S = \{\alpha, \beta, \gamma, \delta, \varepsilon, \epsilon, \zeta, \eta, \theta, \vartheta, \iota, \kappa, \lambda, \mu, \nu\}, \quad (2.28)$$

де α – атака вішингу (the vishing attack); β – фішингова атака (phishing attack); γ – клонування профілю (profile cloning); δ – грумінг (grooming); ε – сміттєве збирання (dumpster diving attacks); ϵ – підхід упритул до об'єкта доступу (tailgating); ζ – маскування файлів (file masquerade); η – приманка (baiting); θ – оманливе ПЗ або оманливі спливаючі вікна (scareware or pop – up windows); ϑ s_ϑ – зараження довірених ресурсів (water – holing); ι s_ι – троянські листи (trojan mail); κ – цільовий фішинг (spear phishing); λ – спам (spam mail); μ – приманювання «цікавим програмним забезпеченням» (interesting software); ν – hoax-повідомлення (hoaxing).

Розглянемо моделі атак соціальної інженерії з використанням атак вішинг (vishing), фішинг (phishing) і клонування профілю (profile cloning), щоб представити покроковий процес атак від початкового вибору цілі до вилучення інформації та механізмів захисту. Цей структурований підхід допомагає проаналізувати та зрозуміти динаміку атак і взаємодію між різними компонентами, а також визначити вразливі місця для розробки стратегій пом'якшення таких атак.

2.3.1 Модель атаки типу «вішинг»

Для побудови схеми функціонування вішингової атаки в межах РКС враховують основні етапи: формування правдоподібного приводу для дзвінка, підробку ідентифікатора абонента для підвищення довіри, психологічне маніпулювання з метою отримання конфіденційної інформації, безпосереднє її вилучення під прикриттям перевірки даних, а також подальше використання здобутої інформації у шахрайських діях чи для приховування слідів атаки.

Отже, представимо модель атаки вішингу у вигляді кортежу:

$$M_\alpha = \langle A_\alpha, V_\alpha, P_\alpha, C_\alpha, T_\alpha, I_\alpha, D_\alpha \rangle, \quad (2.25)$$

де $A_\alpha = \{a_{\alpha 1}, a_{\alpha 2}, \dots, a_{\alpha N_{\alpha A}}\}$ – множина, яка представляє осіб, які здійснюють атаки вішингом, $N_{\alpha A}$ є кількістю осіб, які здійснюють атаки вішингом;

$V_\alpha = \{v_{\alpha 1}, v_{\alpha 2}, \dots, v_{\alpha N_{\alpha V}}\}$ – множина, яка включає потенційних і реальних жертв, на яких нападають, $N_{\alpha V}$ – кількість жертв;

$P_\alpha = \{p_{\alpha 1}, p_{\alpha 2}, \dots, p_{\alpha N_{\alpha P}}\}$ – множина, яка представляє різні приводи або сценарії, які використовують зловмисники для обману жертв; $N_{\alpha P}$ – кількість сценаріїв;

$C_\alpha = \{c_{\alpha 1}, c_{\alpha 2}, \dots, c_{\alpha N_{\alpha C}}\}$ – множина, яка включає різні ідентифікатори абонентів, які використовуються під час підробки, щоб зробити виклик легітимним; $N_{\alpha C}$ це кількість ідентифікаторів абонентів;

$T_\alpha = \{t_{\alpha 1}, t_{\alpha 2}, \dots, t_{\alpha N_{\alpha T}}\}$ – множина, яка містить психологічні прийоми, застосовувані нападниками, $N_{\alpha T}$ – кількість психологічних прийомів;

$I_\alpha = \{i_{\alpha 1}, i_{\alpha 2}, \dots, i_{\alpha N_{\alpha I}}\}$ – множина, яка представляє конфіденційну інформацію, на яку спрямовані зловмисники, $N_{\alpha I}$ – номер конфіденційної інформації;

$D_\alpha = \{d_{\alpha 1}, d_{\alpha 2}, \dots, d_{\alpha N_{\alpha D}}\}$ – множина, яка складається з різноманітних захисних механізмів і засобів протидії, доступних потенційним жертвам, $N_{\alpha D}$ – це кількість захисних механізмів.

Визначимо функцію вибору претексту $f_{\alpha P}$, яка описує процес вибору претексту на основі профілю жертви, як:

$$f_{\alpha P}: V_\alpha \times A_\alpha \to P_\alpha, f_P(v_i, a_j) = p_k, \quad (2.26)$$

де p_k – відповідний привід для жертви $v_{\alpha i}$ з боку нападника $a_{\alpha j}$.

Визначимо функцію підробки ідентифікатора абонента $f_{\alpha C}$, яка моделює вибір ідентифікатора абонента для підвищення довіри, як:

$$f_{\alpha C}: P_\alpha \to C_\alpha, f_{\alpha C}(p_{\alpha k}) = c_{\alpha m}, \quad (2.27)$$

де c_m – ідентифікатор абонента, обраний як претекст $p_{\alpha k}$.

Позначимо психологічну маніпулятивну функцію $f_{\alpha T}$, яка фіксує використання психологічних тактик впливу на жертву, як:

$$f_{\alpha T}: P_\alpha \times A_\alpha \to T_\alpha, f_{\alpha T}(p_{\alpha k}, a_{\alpha j}) = t_{\alpha n}, \quad (2.28)$$

де $t_{\alpha n}$ тактика, використана $p_{\alpha k}$ нападником як привід $a_{\alpha j}$.

Позначимо функцію вилучення інформації $f_{\alpha C}$, яка описує вилучення інформації від жертви, як:

$$f_{\alpha I}: V_\alpha \times P_\alpha \times T_\alpha \to I_\alpha, f_{\alpha I}(v_{\alpha i}, p_{\alpha k}, t_{\alpha n}) = i_{\alpha o}, \quad (2.29)$$

де $i_{\alpha o}$ інформація, отримана від жертви $v_{\alpha i}$ з використанням приводу $p_{\alpha k}$ та тактики $t_{\alpha n}$.

Позначимо захисно – активаційну функцію $f_{\alpha D}$, яка являє собою активацію захисних механізмів жертвою, як:

$$f_{\alpha D}: V_\alpha \times I_\alpha \to D_\alpha, f_{\alpha D}(v_{\alpha i}, i_{\alpha o}) = d_{\alpha p}, \quad (2.30)$$

де $d_{\alpha p}$ – захисний механізм, який активує жертва v_iпісля отримання інформації $i_{\alpha o}$.

Щоб описати вплив на елементи наборів, визначимо функцію впливу g_α що кількісно визначає успіх атаки та ефективність захисту. Функція впливу враховує ймовірність успіху різних етапів атаки та ймовірність ефективності механізмів захисту.

Позначимо функцію впливу g_α:

$$g_\alpha: V_\alpha \times P_\alpha \times T_\alpha \times I_\alpha \times D_\alpha \to \mathbb{R}, g_\alpha(v_{\alpha i}, p_{\alpha k}, t_{\alpha n}, i_{\alpha o}, d_{\alpha p}) = S_\alpha, \quad (2.31)$$

де S_α – дійсне число, що представляє рівень успіху атаки, причому більші значення вказують на більший успіх; функція g_α може включати такі фактори, як ймовірність того, що жертва $v_{\alpha i}$ піддасться приводу $p_{\alpha k}$, нехай позначимо як $P_\alpha(v_{\alpha i}, p_{\alpha k})$; ефективність психологічної тактики $t_{\alpha n}$ щодо потерпілого $v_{\alpha i}$, нехай позначено $T_\alpha(v_{\alpha i}, t_{\alpha n})$; ймовірність отримання інформації $i_{\alpha o}$(позначається $I_\alpha(i_{\alpha o})$; ефективність захисного механізму $d_{\alpha p}$ (позначається $D_\alpha(d_{\alpha p})$.

2.3.2 Модель атаки типу «фішинг»

Для побудови схеми функціонування фішингової атаки в межах РКС враховують такі основні етапи: вибір цілі на основі доступних даних, формування контекстуально правдоподібного претексту, створення повідомлення, що імітує комунікацію довіреної організації, та його доставлення через відповідний канал (електронну пошту, SMS або соціальні мережі). Подальша взаємодія цілі з повідомленням активує фазу атаки, після чого отриману інформацію можуть використати для додаткових шахрайських дій або наступних фішингових кампаній.Отже, представимо модель атаки вішингу у вигляді кортежу:

$$M_\beta = \langle A_\beta, V_\beta, P_\beta, C_\beta, T_\beta, I_\beta, D_\beta, \rangle, \tag{2.32}$$

де $A_\beta = \{a_{\beta 1}, a_{\beta 2}, \dots, a_{\beta N_{\beta A}}\}$ – множина, яка представляє осіб, які здійснюють фішингові атаки, $N_{\beta A}$ є кількістю осіб, які здійснюють фішингові атаки;

$V_\beta = \{v_{\beta 1}, v_{\beta 2}, \dots, v_{\beta N_{\beta V}}\}$, – множина, яка включає потенційних і реальних жертв, на яких нападають, $N_{\beta V}$ – кількість жертв;

$A_\beta = \{a_{\beta 1}, a_{\beta 2}, \dots, a_{\beta N_{\beta A}}\}$ – множина, яка представляє окремих осіб або групи, які здійснюють фішингові атаки;

$T_\beta = \{t_{\beta 1}, t_{\beta 2}, \dots, t_{\beta N_{\beta T}}\}$ – множина, яка містить потенційних жертв, націлених на фішингові повідомлення;

$P_\beta = \{p_{\beta 1}, p_{\beta 2}, \dots, p_{\beta N_{\beta P}}\}$ – множина приводів, які використовуються у фішингових повідомленнях для видавання за легітимні джерела;

$D_\beta = \{d_{\beta 1}, d_{\beta 2}, \dots, d_{\beta N_{\beta D}}\}$, – множина, яка використовує канали для доставки фішингових повідомлень (наприклад, електронної пошти, SMS);

$M_\beta = \{m_{\beta 1}, m_{\beta 2}, \dots, m_{\beta N_{\beta M}}\}$, – множина, яка містить компоненти фішингового повідомлення, включаючи посилання, вкладення та вміст;

$AC_\beta = \{ac_{\beta 1}, ac_{\beta 2}, \dots, ac_{\beta N_{\beta AC}}\}$, – множина, яка включає в себе дії, до яких заохочується ціль (наприклад, натискання посилання, завантаження файлу);

$I_\beta = \{i_{\beta 1}, i_{\beta 2}, \dots, i_{\beta N_{\beta I}}\}$ – множина, яка представляє інформацію або результати, які шукають зловмисники (наприклад, облікові дані, встановлення зловмисного програмного забезпечення);

$DF_\beta = \{df_{\beta 1}, df_{\beta 2}, \dots, df_{\beta N_{\beta DF}}\}$ – множина, яка містить засоби протидії та захисту, що застосовуються цілями.

Визначимо функцію вибору претексту $f_{\beta P}$ який описує процес вибору приводу на основі профілю жертви як: $f_{\beta P}: T_\beta \times A_\beta \to P_\beta$

Ця функція описує, як атакувальник $a_{\beta j}$ вибирає відповідний привід $p_{\beta k}$ для певної цілі атаки $t_{\beta i}$.

Атакувальники обирають привід, який, на їхню думку, буде найбільш переконливим для цілі. Наприклад, якщо відомо, що ціль атаки являється клієнтом певного банку, привід може включати спосіб видати себе за цей банк $f_{\beta P}(t_{\beta i}, a_{\beta j},) = p_{\beta k}$. Зокрема визначимо:

$$f_{\beta P}(t_{\beta John}, a_{\beta Hacker1}) = p_{\beta Bank\ Alert}, \tag{2.33}$$

де зловмисник вибирає претекст «Банківське сповіщення» для цільової жертви атаки.

Визначимо функцію вибору каналу $f_{\beta D}$як: $f_{\beta D}: P_\beta \times A_\beta \to D_\beta$. Ця функція показує, як зловмисник $a_{\beta j}$ вибирає канал доставки $d_{\beta m}$ для вибраного приводу

$p_{\beta k}$. Атакуюча сторона вибирає найефективніший канал доставки для надсилання фішингового повідомлення, як електронна пошта, SMS або соціальні мережі: $f_{\beta D}(p_{\beta k}, a_{\beta j},) = d_{\beta m}$. Зокрема визначимо:

$$f_{\beta D}(p_{\beta Bank\ Alert}, a_{\beta Hacker1}) = d_{\beta Email}, \quad (2.34)$$

де $d_{\beta m}$– множина, яка вирішує використати електронну пошту як канал доставки.

Визначимо функцію створення повідомлення $f_{\beta M}$ як:

$$f_{\beta M}: P_{\beta} \times D_{\beta} \rightarrow M_{\beta}, \quad (2.35).$$

Функція $f_{\beta M}$ моделює створення фішингового повідомлення $M_{\beta n}$ на основі претексту $p_{\beta k}$ та каналу доставки $d_{\beta m}$.

Повідомлення створено відповідно до претексту та каналу доставки. Він включає візуальний дизайн, мову та певний вміст, які роблять його легітимним $f_{\beta M}(p_{\beta k}, d_{\beta m},) = m_{\beta n}$. Зокрема визначимо:

$$f_{\beta M}(p_{\beta Bank\ Alert}, d_{\beta Email}) = m_{\beta Phishing\ Email}, \quad (2.36)$$

де фішингове повідомлення електронної пошти містить фірмовий знак банку та повідомлення про сповіщення системи безпеки.

Позначимо функцію спонукання до дії як: $f_{\beta A^C}$ як: $f_{\beta A^C}: M_{\beta} \times T_{\beta} \rightarrow AC_{\beta}$.

Функція $f_{\beta A^C}$ описує, як створене повідомлення $m_{\beta n}$ спонукає ціль $t_{\beta i}$ виконати певну дію $ac_{\beta o}$.

Фішингове повідомлення спрямоване на те, щоб переконати ціль виконати дію, наприклад натиснути посилання, завантажити вкладений файл або надати конфіденційну інформацію: $f_{\beta A^C}(m_{\beta n}, T_{\beta i}) = ac_{\beta o}$.Зокрема визначимо:

$$f_{\beta A^C}(m_{\beta Phishing\ Email}, t_{\beta target}) = ac_{\beta Click\ Link}, \quad (2.37)$$

де фішинговий електронний лист переконує ціль атаки (target) натиснути посилання.

Позначимо функцію вилучення інформації $f_{\beta I}$ як: $f_{\beta I}: AC_{\beta} \times T_{\beta} \rightarrow I_{\beta}$.

Ця функція представляє вилучення конфіденційної інформації $i_{\beta p}$ з цілі $t_{\beta i}$ на основі дії $ac_{\beta o}$. Після того, як мета виконує індуковану дію, атакувальник захоплюєпотрібну інформацію, таку як облікові дані для входу або особисті дані $f_{\beta I}(ac_{\beta o}, t_{\beta i}) = i_{\beta p}$. Зокрема визначимо:

$$f_{\beta I}(ac_{\beta Click\ Link}, t_{\beta Jtarget}) = i_{\beta Login\ Credentials},, \tag{2.38}$$

де, натиснувши посилання, ціль атаки $target$ обманом змушують надати свої облікові дані для входу.

Позначимо функцію активації захисту $f_{\beta D^F}$ як: $f_{\beta D^F}: T_\beta \times M_\beta \rightarrow DF_\beta$.

Ця функція моделює активацію захисних механізмів $df_{\beta q}$ цілі $t_{\beta i}$у відповідь на фішингове повідомлення $m_{\beta n}$. Цілі можуть використовувати різні засоби захисту, щоб розпізнати та пом'якшити спробу фішингу, як – от фільтри електронної пошти, навчання користувачів або багатофакторну автентифікацію:

$$f_{\beta D^F}(t_{\beta i}, m_{\beta n},) = df_{\beta q}, \tag{2.39}$$

Зокрема визначимо:

$$f_{\beta D^F}(t_{\beta target}, m_{\beta Phishing\ Email}) = df_{\beta Two-FactAuthent}, \tag{2.40}$$

де ціль атаки використовує двофакторну автентифікацію як механізм захисту від фішингових електронних листів.

Таким чином, для конкретної цілі атаки, $v_{\beta i}$ на яку нападає $a_{\beta 1}$:

Вибір претексту для мети атакувальниками:

$$f_{\beta P}(t_{\beta i}, a_{\beta j}) = p_{\beta k}\,. \tag{2.41}$$

Вибір зловмисниками каналу доставки для приводу:

$$f_{\beta D}(p_{\beta k}, a_{\beta j}) = d_{\beta m}\,. \tag{2.42}$$

Створення фішингового повідомлення зловмисниками:

$$f_{\beta M}(p_{\beta k}, d_{\beta m}) = m_{\beta n}\,. \tag{2.43}$$

Спонукання мети повідомлення до дії:

$$f_{\beta A^C}(m_{\beta n}, t_{\beta i}) = ac_{\beta o}\,. \tag{2.44}$$

Результатом отримання дії є вилучення інформації:

$$f_{\beta I}(ac_{\beta o}, t_{\beta i}) = i_{\beta p}\,. \tag{2.45}$$

Активація захисних механізмів мішенями:

$$f_{\beta D^F}(t_{\beta i}, m_{\beta n}) = df_{\beta q}\,. \tag{2.46}$$

Позначимо функцію впливу g_β так:

$$g_\beta : T_\beta \times P_\beta \times D_\beta \times M_\beta \times AC_\beta \times I_\beta \times DF_\beta \to R_\beta, \tag{2.47}$$

$$g_\beta(t_{\beta i}, p_{\beta k}, d_{\beta m}, m_{\beta n}, ac_{\beta o}, i_{\beta p}, Df_{\beta q}) = S_\beta\,, \tag{2.48}$$

де S_β – рівень успішності фішингової атаки.

Фактори, що впливають, S_β можуть включати ефективність приводу, достовірність каналу доставки, переконливу силу компонентів повідомлення, ймовірність того, що об’єкт виконає індуковану дію, і ефективність захисних механізмів об’єкта.

2.3.3 Модель атаки типу «клонування профілю»

Для побудови схеми функціонування атаки клонування профілю в межах РКС враховують такі основні етапи: вибір цільового користувача, збирання публічних даних, створення профілю, що імітує справжній, встановлення довіри з його контактами та подальшу експлуатацію для отримання конфіденційної інформації чи розширення кола потенційних цілей. До ключових контрзаходів належать обмеження видимості особистих даних, перевірка неочікуваних запитів, підвищення обізнаності користувачів та застосування механізмів двофакторної автентифікації.

Отже, представимо модель атаки клонування профілю у вигляді кортежу:

$$M_\gamma = \langle A_\gamma, T_\beta, I_\beta, C_\beta, V_\beta, E_\beta \rangle, \tag{2.49}$$

де $A_\gamma = \{a_{\gamma 1}, a_{\gamma 2}, \dots, a_{\gamma N_{\gamma A}}\}$ – множина, яка представляє осіб або групи, які здійснюють атаки клонування профілю;

$T_\gamma = \{t_{\gamma 1}, t_{\gamma 2}, \dots, t_{\gamma N_{\gamma T}}\}$ – множина, яка представляє користувачів, профілі яких клоновано;

$I_\gamma = \{i_{\gamma 1}, i_{\gamma 2}, \dots, i_{\gamma N_{\gamma I}}\}$ – множина, яка включає загальнодоступну інформацію про ціль, що використовується для клонування;

$C_\gamma = \{c_{\gamma 1}, c_{\gamma 2}, \dots, c_{\gamma N_{\gamma C}}\}$ – множина, яка містить підроблені профілі, створені зловмисниками;

$V_\gamma = \{v_{\gamma 1}, v_{\gamma 2}, \dots, v_{\gamma N_{\gamma V}}\}$ – множина, яка містить контакти об'єкта, які є потенційними жертвами подальшої експлуатації;

$E_\gamma = \{e_{\gamma 1}, e_{\gamma 2}, \dots, e_{\gamma N_{\gamma E}}\}$– множина, яка представляє методи соціальної інженерії, що використовуються для отримання інформації з контактів цільової групи.

Позначимо функцію збору інформації $f_{\gamma I}$, яка описує, як атакувальники збирають інформацію про ціль атаки, як:

$$f_{\gamma I}: T_\gamma \times A_\gamma \to I_\gamma, f_{\gamma I}(t_{\gamma i}, a_{\gamma j},) = i_{\gamma k}. \quad (2.50)$$

Позначимо функцію створення профілю $f_{\gamma C}$, яка моделює створення клонованого профілю з використанням зібраної інформації:

$$f_{\gamma C}: I_\gamma \times A_\gamma \to C_\gamma, f_{\gamma C}(t_{\gamma i}, a_{\gamma j},) = C_{\gamma m}. \quad (2.51)$$

Позначимо функцію встановлення довіри $f_{\gamma E_C}$, яка представляє процес встановлення довіри до клонованого профілю шляхом взаємодії з контактами цілі як:

$$f_{\gamma E_C}: C_\gamma \times V_\gamma \to C_\gamma, f_{\gamma E_C}(c_{\gamma m}, v_{\gamma n},) = c'_{\gamma m} \quad (2.52)$$

Визначимо функцію ініціації контакту $f_{\gamma C_I}$, яка описує, як зловмисники використовують клонований профіль, щоб ініціювати контакт із з'єднаннями цілі, як:

$$f_{\gamma C_I}: C_\gamma \times V_\gamma \to V_\gamma, f_{\gamma C_I}(c'_{\gamma m}, v_{\gamma n},) = v'_{\gamma n}, \quad (2.53)$$

Позначимо функцію експлуатації $f_{\gamma E}$, яка моделює використання методів соціальної інженерії для експлуатації контактів об'єкта, як:

$$f_{\gamma E}: V_\gamma \times C_\gamma \times E_\gamma \to I_\gamma, f_{\gamma E}(v'_{\gamma n}, c'_{\gamma m}, e_{\gamma p},) = i_{\gamma q}\ , \quad (2.54)$$

Позначимо активаційну функцію захисту $f_{\gamma D^F}$, що представляє активацію механізмів захисту мішенню або її контактами, як:

$$f_{\gamma D^F}: T_\gamma \times C_\gamma \times V_\gamma \rightarrow D_\gamma, f_{\gamma D^F}\left(t_{\gamma i}, c_{\gamma m}, v_{\gamma n},\right) = d_{\gamma r}\ , \tag{2.55}$$

Щоб описати загальний вплив атаки клонування профілю, ми визначаємо функцію впливу, яка кількісно визначає успіх атаки та ефективність механізмів захисту.

Позначимо функцію впливу g_γ як:

$$g_\gamma: T_\gamma \times I_\gamma \times C_\gamma \times V_\gamma \times E_\gamma \times D_\gamma \rightarrow R_\gamma, \tag{2.56},$$

$$g_\gamma : \left(t_{\gamma i},\ i_{\gamma k},\ c_{\gamma m},\ v_{\gamma n},\ e_{\gamma p},\ d_{\gamma r}\right) = S_\gamma, \tag{2.57},$$

де S_γ – рівень успіху атаки клонування профілю.

Фактори, що впливають, S_γ можуть включати: кількість і якість інформації, $i_{\gamma k}$ зібраної про ціль; достовірність клонованого профілю $c_{\gamma m}$; ефективність методів соціальної інженерії $e_{\gamma p}$; реагування та захисні механізми, $d_{\gamma r}$ активовані ціллю або її контактами.

2.4 Метод виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора формулювання

2.4.1 Основи методу виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора формулювання

Для підвищення стійкості розподілених КС до телефонних атак соціальної інженерії розроблено метод виявлення кібератак на основі унікального лінгвістичного ідентифікатора формулювання (УЛІФ). Метод передбачає попередню обробку мовленнєвих даних, їх подання у вигляді n-вимірних векторів та обчислення семантичної близькості для класифікації текстів, отриманих у розмовах користувачів [184].

УЛІФ визначається як характерне вербальне формулювання зловмисника, яке не лише містить змістовий компонент, а й спрямоване на спонукання адресата до певної дії. Набір таких висловлювань утворює специфічний лінгвістичний простір, це дає змогу ідентифікувати клас атак соціальної інженерії за аналогією до сигнатурного принципу виявлення шкідливих програм.

Запропонований метод інтегровано до синтезованої розподіленої комп'ютерної системи та розглядається як складова комплексного механізму протидії телефонним атакам соціальної інженерії.

Позначимо набір унікальних лінгвістичних ідентифікаторів формулювання як множина УЛІФ, що описує розмову зловмисників, де N – номер унікального ідентифікатора лінгвістичного формулювання:

$$I_1 = \{i_j\}_{\mathrm{j}=1}^{N}. \tag{2.58}$$

Для, наприкладу, використаємо наступний зразок для конструкції УЛІФ:

I_1 = «Служба безпеки Національного банку, повідомляємо, що Вашу платіжну картку буде заблоковано»; I_2 = «Вам нараховано безповоротну фінансову допомогу від міжнародного фонду»; I_3 = «Повідомте нам повний номер вашої платіжної картки, ПІН–код, CVV–код, вказані на звороті картки»; I_4 = «Потрібно підійти до найближчого банкомату, щоб ввести спеціальну комбінацію цифр»; I_{j} = «Тепер ви отримаєте секретний SMS – код на свій номер, який вам потрібно повідомити тільки нам».

Метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах (рис. 2.2) охоплює послідовність наступних етапів:

1. Збір даних. Формується корпус висловлювань із шахрайських та легітимних сценаріїв різних типів взаємодій. Забезпечення різноманітності дає змогу створити репрезентативну множину УЛІФ.

2. Попередня обробка. Дані очищуються від шуму, стандартизуються, сегментуються та токенізуються, що усуває структурні неузгодженості й забезпечує коректність подальшого формування ознак.

3. Маркування даних. Висловлювання позначаються як «шахрайські» або «легітимні». Якісне маркування формує основу для точного навчання класифікаційної моделі.

4. Формування ознак УЛІФ. Для кожного прикладу створюється вектор ознак, що відображає семантичні та синтаксичні характеристики формулювання і дає можливість подальшого ефективного порівняння.

5. Навчання моделі. На основі векторизованих та розмічених даних навчається класифікатор, зокрема модель k – найближчих сусідів, орієнтована на визначення схожості між УЛІФ і віднесення їх до відповідного класу.

6. Класифікація методом k-NN. Для нового висловлювання обчислюється близькість до еталонних прикладів, здійснюється голосування сусідів, після чого формується рішення щодо його належності до шахрайських або легітимних повідомлень.

7. Оцінювання ефективності. Точність моделі перевіряється за допомогою базових метрик (Precision, Recall, F1, MCC) та перехресної валідації, що дозволяє оцінити її узагальнювальну здатність.

8. Налаштування гіперпараметрів. Виконується добір оптимальних параметрів класифікації та попередньої обробки для зменшення помилок і підвищення точності визначення УЛІФ.

9. Встановлення порогового значення. Обирається поріг довіри для розмежування шахрайських і легітимних формулювань з урахуванням балансу між хибними тривогами та пропущеними атаками.

10. Формування реакції. Результат класифікації передається захисним підсистемам РКС, активуючи відповідні сценарії реагування та забезпечуючи інтеграцію методу у систему протидії атакам соціальної інженерії. атаки та адаптивними засобами протидії в розподіленій інфраструктурі.

Представлення даних включає процедури виділення ознак і побудову вектора ознак.

Щоб побудувати унікальний ідентифікатор лінгвістичного формулювання, потрібно виділити набір відповідних ознак.

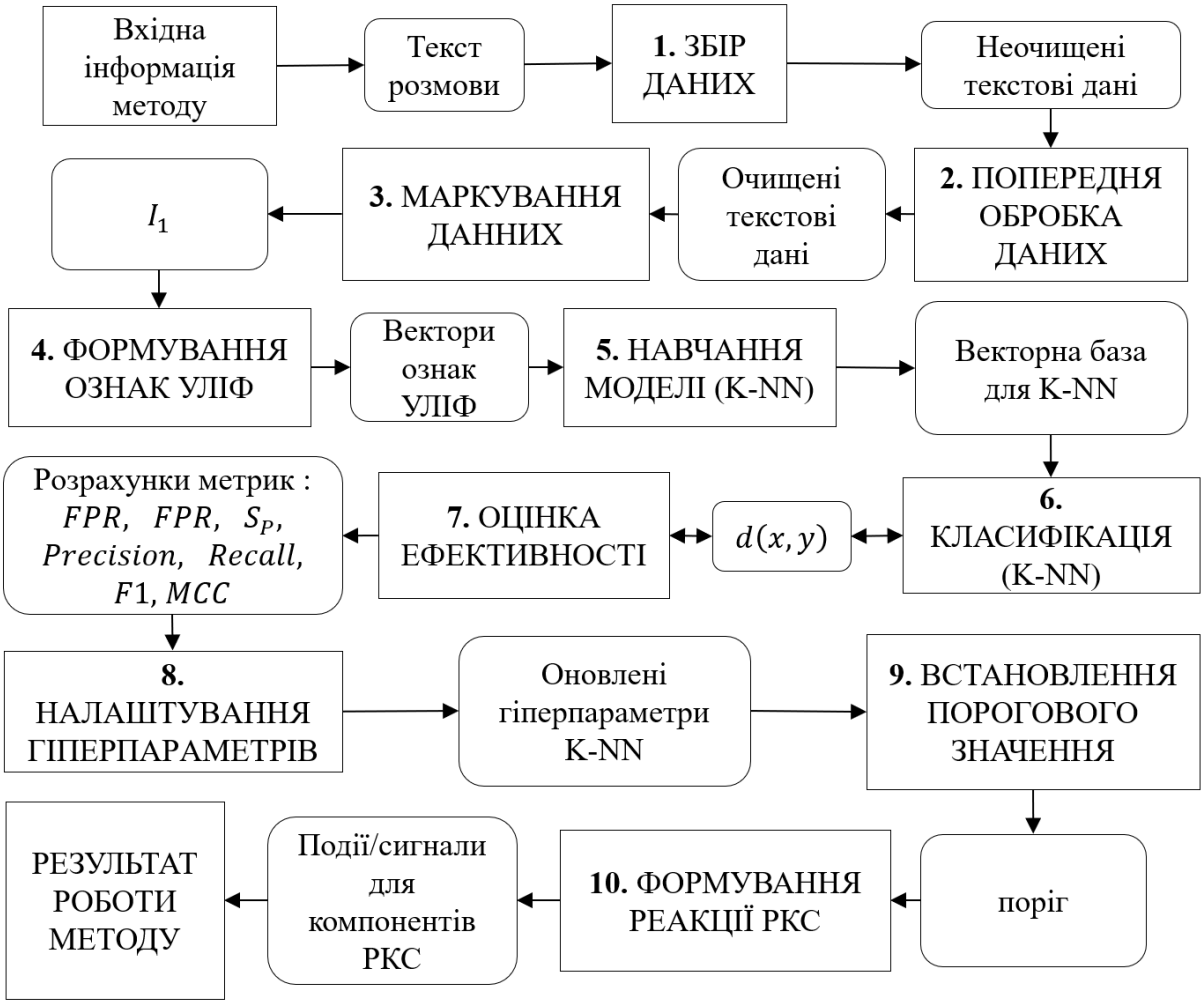


Рисунок 2.2 – Узагальнена схема методу виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора формулювання

2.4.2 Застосування методу k-найближчих сусідів як засобу класифікації унікальних ідентифікаторів лінгвістичного формулювання

Метод k-найближчих сусідів було застосовано до завдань класифікації ідентифікаторів унікальних лінгвістичних формулювань, метою яких є класифікація УЛІФ у різні класи на основі набору певних ознак.

Наступний крок – побудова вектора ознак. Кожен набір виділених ознак УЛІФ має бути представлений як вектор ознак. Кожен вектор ознак УЛІФ представляє його позицію в багатовимірному просторі ознак. Виконання фази навчання включає отримання позначеного набору даних і масштабування ознак.

Наступним кроком є підготовка позначеного набору даних, де кожен унікальний ідентифікатор лінгвістичного формулювання пов’язується з відповідним класом – зловмисний або нешкідливий телефонний дзвінок.

Після цього виконуємо процедуру масштабування функції. Цей процес складається з нормалізації значень ознак, щоб гарантувати, що всі функції однаково вносять свій внесок у обчислення відстані.

Фаза класифікації включає розрахунок відстані, вибір сусіда та процедури голосування більшістю. На етапі класифікації важливо оцінити відстань, тобто, коли ми класифікуємо новий УЛІФ, ми повинні обчислити відстані між його вектором ознак і векторами ознак усіх УЛІФ у навчальному наборі.

Для цього в дослідженні була задіяна метрика відстані, евклідова відстань:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}. \quad (2.59)$$

Після оцінки відстані необхідно виконати процедуру вибору сусідів, де необхідно вибрати значення K-найближчих сусідів на основі обчислених відстаней.

Останній крок полягає у застосуванні схеми голосування більшості для визначення класу нового УЛІФ. Клас, який найчастіше зустрічається серед K-найближчих сусідів, призначається зазначеному УЛІФ.

2.4.4 Експериментальні дослідження методу виявлення кібератак соціальної інженерії із застосуванням телефону на основі унікального лінгвістичного ідентифікатора формулювання

Для здіснення апроації запропонваного методу ідентифікації кібератак на основі використання соціальної інженерії по телефону було проведено ряд різноспрямоавних експериментів.

Перший клас експериментів полягав у опрацюванні даних шляхом залучення адаптованого набору даних на основі CallHome, який є частиною проекту TalkBank [184]. У таблиці 2.4 показано кількісні характеристики наборів даних з TalkBank. Це певний набір даних або колекція даних, пов’язаних із

телефонними розмовами. Проект TalkBank – це дослідницька ініціатива, яка зосереджена на зборі, аналізі та обміні даними розмовної мови з метою вивчення різних аспектів людського спілкування та розвитку мови.

Набори даних CallHome зазвичай включають записи телефонних розмов різними мовами та з різних культурних контекстів.

Ці набори даних є цінними ресурсами для дослідників у галузі лінгвістики, комунікації та суміжних галузей для дослідження таких тем, як аналіз розмов, соціолінгвістика та розвиток мови в натуралістичних умовах [186, 187].

Таблиця 2.4 – кількісні характеристики наборів даних з TalkBank.

Набір даних	Навчальний набір	Тестовий набір
Текст з телефоним шахрайством.	523	328
Звичайний текст	278	495
Всього	801	823

Ще одним джерелом кібератак на основі використання соціальної інженерії по телефону стала служба банківської безпеки найбільшого банку України. Модифікований набір даних включав 1300 унікальних ідентифікаторів лінгвістичних формулювань. Для тестування системи було згенеровано 30 розмов, які включали властивості кібератак на основі використання соціальної інженерії по телефону.

Для оцінки ефективності методу ідентифікації кібератак на основі використання соціальної інженерії по телефону були задіяні метрики: TPR – True Positive Rate; FPR – False Positive Rate; Precision; Recall; , F1-score ; MCC [188]

Здійснимо розрахунки метрик за формулами:

$$FPR = \frac{TP}{TP+FN} * 100\,, \quad FPR = \frac{FP}{TN+FP} * 100, \quad S_P = \frac{TN}{TP+FN} * 100,$$

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN}, \; F1 = \frac{2*Recall*Precision}{Recall+Precision},$$

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \qquad (2.60)$$

де MCC – це метрика, яка використовується для оцінки якості моделей бінарної класифікації, особливо при роботі з незбалансованими наборами даних.

Результати бінарної класифікації зловмисної та доброякісної телефонної розмови для 30 експериментів представлені в Таблиці 2.5.

Таблиця 2.5

Результати бінарної класифікації зловмисної та доброякісної телефонної розмови для 30 експериментів

Кількість атак	Точність	Точність	Відкликати	F–бал
1	0,939	0,971	0,949	0,931
2	0,954	0,944	0,963	0,933
3	0,934	0,962	0,946	0,966
4	0,935	0,972	0,973	0,962
5	0,942	0,935	0,974	0,943
6	0,967	0,937	0,943	0,965
7	0,974	0,936	0,936	0,941
8	0,963	0,931	0,938	0,949
9	0,953	0,938	0,951	0,964
10	0,962	0,945	0,941	0,953
11	0,952	0,943	0,968	0,973
12	0,96	0,941	0,955	0,951
13	0,973	0,959	0,937	0,972
14	0,97	0,939	0,933	0,954
15	0,966	0,966	0,95	0,961
16	0,961	0,954	0,959	0,952
17	0,941	0,949	0,956	0,955
18	0,968	0,946	0,964	0,937
19	0,931	0,974	0,935	0,959
20	0,955	0,957	0,947	0,939

21	0,958	0,933	0,967	0,946
22	0,972	0,948	0,934	0,95
23	0,933	0,95	0,944	0,957
24	0,937	0,975	0,931	0,971
25	0,938	0,967	0,945	0,942
26	0,946	0,951	0,942	0,938
27	0,959	0,934	0,965	0,969
28	0,943	0,953	0,953	0,944
29	0,965	0,973	0,969	0,934
30	0,969	0,942	0,957	0,975

2.4 Висновки до другого роздлілу

У розділі запропоновно модель розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, яка базується на мультиагентному поданні з формалізованими моделями станів, стратегій, комунікації, функцій корисності та механізмами досягнення рівноваги Неша й алгоритмами консенсусу, що дає змогу забезпечити узгоджене колективне прийняття рішень, адаптивне управління доступом і підвищення стійкості системи до поведінкових загроз за умов неповноти інформації та локальних збоїв модель системи.

У розділі також представлено архітектуру стійкої до атак соціальної інженерії розподіленої комп'ютерної системи, де синтез такої системи подано як процес узгодження політик і поведінкових моделей агентів з урахуванням багатомодальної природи подій та контекстної залежності впливів соціальної інженерії. Сформовано адаптивну структуру винагороди, яка поєднує зовнішні та внутрішні компоненти оцінювання ефективності дій агентів і відображає їх внесок у зменшення ризиків. Це дозволяє перебудовувати поведінкові стратегії системи відповідно до динаміки загроз і підвищувати її стійкість у режимах реального часу.

Запропонований метод виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора

формулювання є механізмом виявлення атак, включний в процес синтезу архітектури, що забезпечує інтеграцію мовних та поведінкових ознак у загальну структуру прийняття рішень і розширює можливості системи щодо протидії маніпулятивним впливам користувача.

Сформовані у розділі моделі атак соціальної інженерії та відповідні механізми їхнього виявлення забезпечують інтеграцію поведінкових і семантичних ознак у процес синтезу архітектури.

Розроблені модель, архітектура розподіленої комп'ютерної системи, стійкої до соціальної інженерії, метод виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора забезпечують можливість підвищення стійкості до атак соціальної інженерії в умовах неповної інформації та динамічних загроз.

Основні результати розділу опубліковані у працях [173, 174, 176, 182, 183, 184]

РОЗДІЛ 3
СИНТЕЗ МАСШТАБОВАНОЇ АРХІТЕКТУРИ, СТІЙКОЇ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

3.1 Метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії

3.1.1 Основи методу забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії

Для забезпечення масштабованості РКС при збереженні стійкості системи до атак соціальної інженерії, було розроблено метод забезпечення масштабованості архітектури РКС [189]. Метод ґрунтується на поданні всієї системи як багатоагентного середовища, у якому кожен агент виконує локальні функції аналізу подій, оцінювання ризику та прийняття рішень щодо реагування на підозрілу активність. Такий підхід дає змогу поєднати структурні властивості розподіленої інфраструктури з поведінковими характеристиками компонентів, що формують єдину площину взаємодій. Кожен агент є представником певного типу вузла й відображає притаманні цьому типу функції, канали обробки даних, характерні загрози та контекст використання. Саме тому множина агентів відтворює повну гетерогенність системи, а багатоагентна модель дозволяє розглядати складні взаємозалежності, що виникають між її частинами у процесі роботи.

У межах методу система будується як сукупність агентів, поведінка яких визначається їх локальними станами та правилами взаємодії з інформаційним оточенням. Локальний стан агента відображає ризик поточного каналу, ситуаційну обстановку, індикатори підозрілих дій, рівень ескалації загроз та інші параметри, потрібні для прийняття рішення. Однак із ростом кількості агентів виникає потреба перейти від опису окремих взаємодій до аналізу агрегованих характеристик, які узагальнюють стан усієї системи. Метод синтезу

масштабованої архітектури вводить узгоджену систему таких агрегатів, що репрезентують інтенсивність атак, розподіл ризику серед агентів, ступінь навантаження на захисні модулі, узгодженість дій агентів і динаміку поведінкових збурень. Перехід від індивідуального до агрегованого опису дає змогу ліквідувати залежність складності моделювання від кількості агентів і забезпечує масштабованість архітектури.

Центральним концептом методу є репрезентативний агент (агент прийняття рішення), який узагальнює властивості великої групи агентів і приймає рішення, орієнтуючись як на свій локальний стан, так і на глобальні характеристики системи. Агент прийняття рішення забезпечує спрощення моделі без втрати ключових закономірностей: замість відстеження поведінки кожного агента окремо оцінюється типовий механізм реагування для вибраного класу вузлів. Це дозволяє сформувати політику дій, ефективну незалежно від того, чи містить система десятки, сотні або тисячі агентів. Крім того, використання репрезентативного агента забезпечує можливість аналітичного опису взаємозв'язку між локальними рішеннями та глобальною поведінкою системи, що є важливим етапом при синтезі архітектурних параметрів.

Метод синтезу передбачає, що одним із ключових елементів стійкості РКС є коректне визначення винагороди за дії агентів. Винагорода повинна відображати не лише якість локального рішення, а й його вплив на глобальну безпеку. Саме тому метод вводить інтегральну оцінку ризику, що узагальнює наслідки рішень багатьох агентів і дозволяє оцінювати системну вразливість навіть тоді, коли локальні стани окремих агентів не вказують на критичні порушення. Таким чином, рішення агента оцінюється одночасно у двох площинах: локальній, яка відображає безпосередню взаємодію агента з конкретною загрозою, та системній, яка фіксує вплив рішення на загальний стан безпеки розподіленої архітектури.

Після визначення структури агентів, агрегованих індикаторів і функції винагороди метод переходить до формування політики дій репрезентативного агента. Політика повинна бути адаптивною, узгодженою з поведінкою інших

агентів і такою, що забезпечує стійкість системи за різних сценаріїв зміни інтенсивності атак. Для цього виконується процедура ітераційного узгодження: оцінюються наслідки політики на рівні всієї системи, після чого коригуються архітектурні параметри, що впливають на поведінку агентів. У результаті формується така конфігурація політики та архітектури, яка забезпечує збалансованість між точністю реагування, швидкодією, ресурсними обмеженнями та вимогами стійкості.

Метод синтезу масштабується не за рахунок збільшення обчислювальної потужності, а за рахунок правильного структурування взаємодій та інформаційних потоків. Зростання кількості агентів не призводить до суттєвого збільшення складності моделі, оскільки всі процеси прив’язані не до абсолютної кількості агентів, а до інваріантних характеристик їх поведінки. Завдяки цьому метод придатний для використання як у компактних корпоративних інфраструктурах, так і у великих багаторівневих мережах з високою інтенсивністю подій. Завершальна властивість методу полягає у можливості інтеграції архітектурних параметрів, наприклад, щільності розгортання модулів виявлення, глибини логічних ланцюгів доступу, механізмів багатофакторної автентифікації та конфігураційних обмежень каналів, у загальну модель поведінки агентів. Це дає змогу не лише аналізувати, а й конструктивно оптимізувати структуру розподіленої системи.

Таким чином, метод синтезу масштабованої архітектури розподіленої комп’ютерної системи формує цілісну модель багаторівневої взаємодії агентів, яка поєднує локальні механізми захисту з глобальними критеріями стійкості. Такий підхід дозволяє створювати архітектури, здатні зберігати функціональність навіть за умов різкого збільшення кількості агентів, інтенсивності атак або зміни характеру інформаційних потоків. Розглянемо вищевказані кроки методу детально.

3.1.2 Формалізація популяційної мультиагентної системи

Формалізація популяційної мультиагентної системи для задачі протидії атакам соціальної інженерії передбачає побудову чіткої математичної моделі розподіленої КС як сукупності взаємодіючих агентів-детекторів.

Першочергово, необхідно задати множину вузлів, які беруть участь у взаємодії, ввести поділ цієї множини на типи (класи популяцій) та визначити локальні простори станів і дій для кожного типу. Така конструкція уможливлює перехід від мікрорівня опису окремих агентів до макрорівневого подання, у якому динаміка системи характеризується популяційними розподілами та агрегованими індикаторами станів. Це дозволяє виконувати аналіз і синтез розподіленої КС не в координатах кожного окремого вузла, а в узагальненому просторі популяційних характеристик, що відображають глобальну поведінку всієї системи й забезпечують масштабованість незалежно від кількості агентів.

РКС розглядатимемо як скінченна множина вузлів $\mathcal{I} = \{1,2, \dots, N\}$, кожен з яких оснащено агентом – детектором атак соціальної інженерії, де $\mathcal{I}$ – множина індексів усіх вузлів, які входять до розглядуваної РКС, i – індивідуальний ідентифікатор конкретного вузла комп'ютерної мережі, N – загальна кількість вузлів, тобто розмір популяції. Важливо, що в контексті РКС – N розглядається як параметр, який може бути великим і потенційно зростати, що і створює проблему експоненційного росту простору конфігурацій у класичних постановках МАС. Кожному елементу $i \in \mathcal{I}$ ставиться у відповідність агент – детектор, який виконує локальну обробку вхідних подій, пов'язаних із соціальною інженерією. Такий агент моделюється як автономний суб'єкт прийняття рішень, який спостерігає локальний стан своєї системи, оцінює ризики й обирає дії з певного допустимого набору.

Оскільки вузли в розподіленій КС мають різну природу та функціональне призначення, необхідно ввести класифікацію за типами. З цією метою введемо кінцеву множину типів $\mathcal{K} = \{1,2, \dots, K\}$, де $\mathcal{K}$ – множина класифікаційних індексів, K – кількість відмінних типів вузлів, $k \in \mathcal{K}$ – певний клас популяції,

наприклад робочим станціям користувачів. Для кожного типу k введемо підмножину вузлів $\mathcal{I}_k \subseteq \mathcal{I}$, де $\mathcal{I}_k$ – множина індексів тих вузлів, які належать до типу k. Якщо для деякого k позначити через $N_k = | \mathcal{I}_k |$ кількість вузлів даного типу, то виконується співвідношення, де $N = \sum_{k=1}^{K} N_k$ є сумою по всіх типах, $| \mathcal{I}_k |$ – кількість елементів. Поділ на класи популяцій не лише відображає реалістичну гетерогенність РКС, але й закладає основу для подальшого застосування багатопопуляційного опису середнього поля, у якому кожен тип вузлів розглядається як окрема, але взаємодіюча популяція статистично подібних агентів.

Формально визначимо локальні простори станів і дій для агентів кожного типу. Для цього введемо множини $\mathcal{S}^{(k)}$ та $\mathcal{A}^{(k)}, k \in \mathcal{K}$, де $\mathcal{S}^{(k)}$ – локальний простір станів агента, який діє на вузлі типу k, $\mathcal{A}^{(k)}$ – локальний простір доступних дій для локального агента-детектора, де k використовується для фіксації належності до конкретного типу популяції, підкреслюючи, що робоча станція користувача та, наприклад, поштовий шлюз мають різну семантику станів і можуть реалізовувати різні набори дій.

Формальний опис локального стану локального агента-детектора k у момент дискретного часу t задамо змінною $s_{i,t}^{(k)} \in \mathcal{S}^{(k)}, i \in \mathcal{I}_k$, де $s_{i,t}^{(k)}$ – вектор стану агента, який закріплений за вузлом із індексом i, що належить до типу k, у момент часу t, i – конкретний вузол у межах популяції типу k, t – дискретний час або крок прийняття рішень.

Локальний стан $s_{i,t}^{(k)}$ розглядатимемо як композицію кількох компонентів, які відображають релевантні аспекти задачі протидії соціальній інженерії. Представимо локальний стан у вигляді рівняння:

$$s_{i,t}^{(k)} = \left(r_{i,t}^{(k)}, c_{i,t}^{(k)}, f_{i,t}^{(k)}, e_{i,t}^{(k)} \right), \tag{3.2}$$

де $r_{i,t}^{(k)}$ – поточний рівень підозри стосовно активності, що спостерігається на вузлі; $c_{i,t}^{(k)}$ – контекст каналу взаємодії, наприклад, чи пов’язана подія з електронною поштою, вебзапитом, соціальною мережею або голосовою

комунікацією; $f_{i,t}^{(k)}$ – вектор локальних сигнатурних та поведінкових ознак, виділених з журналів подій, вмісту повідомлень, метаданих та історії дій користувача; $e_{i,t}^{(k)}$ – поточний режим ескалації або блокування для даного вузла, що відображає, чи перебуває він у нормальному режимі, режимі посиленої перевірки, режимі часткового блокування чи повної ізоляції.

Формальне введення множин $\mathcal{S}^{(k)}$ та $\mathcal{A}^{(k)}$ для всіх $k \in \mathcal{K}$ створює основу для побудови динаміки розподіленої комп'ютерної системи в наступних кроках методу, на яких визначатимуться ймовірнісні правила переходів стану, функції винагороди та політики прийняття рішень. Ключовим є те, що при синтезі РКС, аналіз вестиметься не на просторі всіх векторів $(s_{i,t}^{(k)})_{i\in\mathcal{I}_k,\, k\in\mathcal{K}}$, розмірність якого експоненційно зростає із збільшенням N, а на просторі розподілів станів у межах кожного типу, що є набагато менш вимогливим з точки зору масштабованості.

3.1.3 Представлення популяції агентів MAC

Перехід до агрегованого популяційного подання передбачає відмову від детального опису станів усіх окремих агентів та заміну його узагальненою характеристикою станів у межах кожного типу вузлів РКС. Такий підхід переводить метод з мікрорівня на макрорівень і забезпечує можливість уникнути експоненційного зростання простору станів і дій під час збільшення кількості агентів у РКС.

Початково для кожного типу вузлів k визначимо емпіричний розподіл станів $\mu_t^{(k),N}$, який описує, як стани агентів цього типу розподілені в популяції у фіксований момент часу t формулою:

$$\mu_t^{(k),N} = \frac{1}{N_k}\sum_{i\in\mathcal{I}_k} \delta_{s_{i,t}^{(k)}}, \tag{3.3}$$

де N_k – загальна кількості вузлів типу, k – належність до конкретного типу популяції, N – залежність емпіричного розподілу від розмірності популяції, тобто від загальної кількості вузлів у системі, $\mathcal{I}_k$ – підмножина індексів вузлів із

загальної множини $\mathcal{I}$, які класифіковані як вузли типу k, $\delta_{s_{i,t}^{(k)}}$ – дельта-розподіл, зосереджений у точці $s_{i,t}^{(k)}$, $s_{i,t}^{(k)}$ – локальний стан агента, закріпленого за вузлом i типу k.

Фактично $\delta_{s_{i,t}^{(k)}}$ є одиничним "імпульсом" у точці простору станів, яка відповідає фактичному стану даного агента. Коли всі такі "імпульси" усереднюються з коефіцієнтом $1/N_k$, утворюється емпіричний розподіл, який відображає частки локальних агентів типу k, що знаходяться в різних станах у момент часу t. Таким чином, $\mu_t^{(k),N}$ є випадковою мірою на просторі станів $\mathcal{S}^{(k)}$, яка залежить як від реалізації системної динаміки, так і від конкретного значення N_k.

Щоб описати конфігурацію всієї МАС, емпіричні розподіли по всіх типах об'єднуються в один узагальнений популяційний стан усієї системи μ_t у момент часу t вектором:

$$\mu_t = \left(\mu_t^{(1)}, \dots, \mu_t^{(K)}\right). \tag{3.4}$$

Наступний крок полягає у введенні детермінованої агрегованої змінної, яка відображає узагальнений стан популяції у граничному режимі при необмеженому збільшенні кількості вузлів РКС. Тобто, при $N \to \infty$, за умови виконання певних регулярних умов, ефект флуктуацій окремих агентів усереднюється, і емпіричні розподіли збігаються до детермінованих мір, які задовольняють певні еволюційні рівняння. Цей перехід формалізується записом:

$$\mu_t^{(k),N} \implies \mu_t^{(k)} \text{ при } N \to \infty, \tag{3.5}$$

де $\mu_t^{(k)}$ – граничний розподіл станів локальних агентів типу k у момент часу t, k – тип популяції, а t – дискретний час.

Аналогічно формується граничний векторний опис $\mu_t = \left(\mu_t^{(1)}, \dots, \mu_t^{(K)}\right)$, де компонент $\mu_t^{(k)}$ – детермінована міра, що еволюціонує згідно з узагальненою системою рівнянь, які описують динаміку популяційного розподілу. Такий перехід від $\mu_t^{(k),N}$ до $\mu_t^{(k)}$ є ключовим з точки зору масштабованості. Він дозволяє розглядати поведінку всієї системи через аналіз

еволюції обмеженої кількості розподілів $\mu_t^{(k)}$, кількість яких дорівнює числу типів K і не залежить від N. Складність опису глобального стану РКС переноситься з рівня окремих агентів на рівень узагальнених популяційних характеристик, що відображають поведінку типів вузлів у системі.

Для переходу від розподілів μ_t до скінченновимірного вектора ознак визначимо функцію $\phi(\mu_t) = z_t \in \mathbb{R}^d$, де ϕ – оператор агрегування, який зіставляє кожному популяційному стану μ_t – числові характеристики у просторі $\mathbb{R}^d$, μ_t – відображення числових характеристик у просторі та розподілів $(\mu_t^{(1)}, \dots, \mu_t^{(K)})$, z_t – результат застосування оператора ϕ, t – дискретний час, узгоджений з еволюцією популяційних розподілів, $\mathbb{R}^d$ – простір, у якому перебуває вектор z_t, d – кількість агрегованих ознак, які використовуються для опису глобального стану РКС.

Функція ϕ задається через систему функціоналів від розподілів μ_t. Кожна компонента вектора z_t інтерпретується як частка вузлів, що перебувають у стані високого ризику, а також як значення міри ризику типу Conditional Value-at-Risk [190], обчисленої для відповідного розподілу.

Тоді для кожного компонента j запишемо:

$$z_t^{(j)} = \psi_j(\mu_t), \tag{3.6}$$

де $z_t^{(j)}$– це j-й компонент вектора z_t, ψ_j – функціонал, який відображає вхідний популяційний стан, μ_t – скалярне значення, t – час.

Розмірність d визначається вимогами до точності та інформативності агрегованого опису, але не залежить від кількості вузлів N. Тобто якщо кількість агентів у системі зростає на кілька порядків, довжина вектора z_t залишається сталою, а отже, глобальний стан, який буде використовуватися в задачі керування або навчання з підкріпленням, не зазнає експоненційного збільшення розмірності. Саме на цьому етапі відбувається усунення експоненційного росту: замість того, щоб розглядати повний вектор станів усіх агентів, розмірність якого пропорційна N, РКС описується через змінну середнього поля μ_t і, ще компактніше, через її параметризацію z_t , яка має фіксовану розмірність d.

Таким чином, узагальнене популяційне представлення формується через три взаємопов'язані кроки: побудову емпіричних розподілів станів для кожного типу вузлів, перехід до детермінованого граничного опису при $N \to \infty$ та подальше стиснення цього опису до скінченновимірного вектора агрегованих ознак. У сукупності ці кроки дозволяють репрезентувати глобальний стан РКС у вигляді об'єкта сталої розмірності.

3.1.4 Задання розширеного стану репрезентативного агента

Постановка задачі опису репрезентативного агента передбачає перехід від явного опису всієї множини локальних агентів РКС до аналізу однієї «типової» системи, яка взаємодіє з агрегованим оточенням. На цьому етапі визначається розширений стан репрезентативного агента, уточнюється стохастична динаміка його еволюції з урахуванням узагальнених популяційних характеристик та формалізується еволюція популяційного розподілу через відповідне рівняння, узгоджене з політикою прийняття рішень. У такий спосіб багатовимірна взаємодія великої кількості детекторів соціальної інженерії в РКС зводиться до керованої задачі для одного агента, який працює у параметризованому середовищі, що відображає вплив усієї РКС.

Розширений стан репрезентативного агента у дискретний момент часу позначається вектором $x_t = (s_t, z_t)$, де x_t – повний стан одного умовного агента в момент часу t, s_t – локальний стан агента, який репрезентує вузол певного типу в розподілених КС, $s_t \in \mathcal{S}^{(k)}$, де $\mathcal{S}^{(k)}$ – простір станів для вузлів типу k, z_t – вектор агрегатів середнього поля, що отримується внаслідок застосування деякого оператора агрегування до популяційного розподілу μ_t.

Визначимо, що $z_t \in \mathbb{R}^d$, де $\mathbb{R}^d$ – це d-вимірний Евклідів простір, d – фіксована розмірність, яка не залежить від кількості вузлів у системі. Кожен компонент вектору z_t інтерпретується як числова характеристика популяції: частка вузлів у високоризиковому стані, оцінка інтенсивності атак по кожному каналу, середній рівень ескалацій чи значення хвостової характеристики ризику,

що відображає середній рівень втрат у найризикованіших ситуаціях. Тому вектор x_t одночасно фіксує локальну конфігурацію конкретної системи та агрегований стан всієї розподіленої КС, що дає можливість репрезентативному агенту враховувати глобальний контекст під час прийняття локальних рішень.

Динаміка репрезентативного агента формалізується як стохастичний процес на просторі розширених станів, у якому перехід від одного кроку дискретного часу до наступного описується умовним розподілом стану. Тоді локальна компонента цієї динаміки задається перехідною ймовірністю $\mathbb{P}(s_{t+1} \mid s_t, a_t, z_t)$.

3.1.5 Визначення функції винагороди задачі синтезу масштабованої архітектури

Визначення функції винагороди в постановці задачі синтезу архітектури РКС, стійкої до атак соціальної інженерії, виконує роль формального носія всіх компромісів між безпекою, експлуатаційною ефективністю та глобальним ризиком системи. На рівні репрезентативного агента така функція подається у вигляді локальної миттєвої винагороди, що відображає якість прийнятого рішення для конкретного вузла за наявного стану РКС, та доповнюється компонентом, який узагальнює системний ризик на рівні всієї популяції вузлів.

У подальшому ця миттєва винагорода інтегрується в глобальний (дисконтований) критерій оптимізації, який визначає ціль синтезу політики та параметрів архітектури РКС.

Локальну функцію винагороди репрезентативного агента як миттєву числову винагорода, яку отримує агент за одноразове прийняття рішення, задамо відображенням:

$$r(s,a,z) = w_{TP}\,\mathbf{1}\{\text{правильне блокування}\} - w_{FN}\,\mathbf{1}\{\text{пропуск атаки}\} - w_{FP}\,\mathbf{1}\{\text{холоста ескалація}\} - c(a), \quad (3.7)$$

де s – локальний стан репрезентативного вузла в момент часу, який розглядається; $s \in \mathcal{S}^{(k)}$, де $\mathcal{S}^{(k)}$ –простір станів вузлів типу k; a – дія, обрана

агентом у цьому стані, причому $a \in \mathcal{A}^{(k)}$, де $\mathcal{A}^{(k)}$ – множина допустимих дій для вузла даного типу; z – вектор агрегатів середнього поля, тобто $z \in \mathbb{R}^d$, де кожна компонента цього вектору кодує певну агреговану характеристику популяції (частку вузлів у високоризиковому стані, середню інтенсивність атак, рівень завантаження тощо); w_{TP} – додатнє число, яке визначає «цінність» правильного блокування атаки; TP –істинно позитивний результат, який підкреслює, що цей коефіцієнт масштабує внесок правильно виявлених і заблокованих атак.

Функція $\mathbf{1}\{\text{правильне блокування}\}$ є індикаторною функцією події «правильне блокування». Вона набуває значення $\mathbf{1}$ у тих реалізаціях процесу, де обрана дія a у стані s, за поточного контексту z, призводить до коректного виявлення та блокування справжньої атаки соціальної інженерії, і нуль – в усіх інших випадках. Отже, доданок $w_{TP}\ \mathbf{1}\{\text{правильне блокування}\}$ збільшує миттєву винагороду рівно на w_{TP} у випадку успішного блокування атаки і не впливає на неї, якщо блокування не відбулося або було некоректним.

Значення w_{FN} – додатнє число, пов’язане з «ціною» пропущеної атаки; FN – помилково негативне спрацювання, що відображає випадок, коли система не розпізнала наявну атаку як шкідливу. Індикаторна функція $\mathbf{1}\{\text{пропуск атаки}\}$ набуває значення $\mathbf{1}$ у реалізаціях, де, за даного стану s та середнього поля z, обрана дія a приводить до того, що справжня атака не була заблокована, а надалі спостерігаються наслідки компрометації. В іншому випадку значення індикатора дорівнює нулю. $w_{FN}\ \mathbf{1}\{\text{пропуск атаки}\}$ визначає, що при кожному пропуску атаки миттєва винагорода зменшується на величину w_{FN}. У конструкції безпеки ця вага обирається найбільшою серед усіх, оскільки пропущена атака несе найбільш тяжкі наслідки для цілісності та конфіденційності інформаційних активів.

Значення w_{FP} – додатнє число що відображає втрати, пов’язані з неправдивими спрацюваннями системи (помилково позитивне спрацювання), FP – помилково позитивне спрацювання. Індикаторна функція $\mathbf{1}\{\text{холоста ескалація}\}$ набуває значення $\mathbf{1}$ у ситуаціях, коли обрана дія a активує блокування або ескалацію інциденту, але подія, яка стала причиною такого

рішення, не є реальною атакою. У результаті система породжує зайві операційні витрати: безпідставне перевантаження служб безпеки, незручність для користувачів, тимчасове блокування легітимних дій. Доданок $w_{FP}\ \mathbf{1}\{\text{холоста ескалація}\}$ вводить штраф за такі дії, балансуючи між агресивним блокуванням і допустимим рівнем хибних тривог.

$c(a)$ – детермінована невід'ємна функція, яка кожній дії a зіставляє числову оцінку ресурсних витрат: використання обчислювальних ресурсів, людської праці, затримок у роботі користувача, переналаштувань політик тощо.

Наявність цього терміну у функції винагороди дозволяє моделі дискретно враховувати небажаність надто дорогих реакцій навіть у разі, коли вони формально коректні. Введена таким чином локальна винагорода $r(s, a, z)$ описує компроміс на рівні одного вузла, проте не гарантує належного урахування системного, або популяційного, ризику.

Для моделювання цього аспекту введемо розширену функція винагороди з глобальним ризиковим компонентом $r_{MF}(s, a, z)$, який відображає системну небезпеку РКС та узагальнює вплив колективної поведінки всіх вузлів на прийняття локальних рішень:

$$r_{MF}(s, a, z) = r(s, a, z) - \lambda\,\mathrm{RiskPop}(z), \tag{3.8}$$

де $r(s, a, z)$ – раніше визначена локальна винагорода; $\lambda\,\mathrm{RiskPop}(z)$ – штраф, пропорційний глобальному популяційному ризику, λ – додатний ваговий коефіцієнт, який контролює інтенсивність урахування системного ризику в загальній функції цінності; збільшення λ означає, що архітектура агресивніше мінімізує популяційний ризик навіть за рахунок локальної зручності.

Функціонал $\mathrm{RiskPop}(z)$ позначає міру ризику на рівні популяції, яка залежить від агрегатів середнього поля z, тобто розширена функція винагороди з глобальним ризиковим компонентом, що відображає системну небезпеку РКС та вплив колективної поведінки вузлів на локальні рішення, де z – вектор агрегованих показників, зокрема частки вузлів у станах високого ризику, частки активних атак за каналами, оцінки навантаження на ескалаційні механізми.

Функціонал RiskPop(z) у методі використовується для оцінювання ризику і базується на мірі CVaR (умовна вартість під ризиком). CVaR показує середній рівень збитків у найгірших випадках, а не просто типовий або середній сценарій. Такий підхід зосереджується на рідкісних, але дуже небезпечних подіях, наприклад масових або скоординованих атаках, і дозволяє оцінити, наскільки система готова до саме таких критичних ситуацій. Властивість опуклості функціонала RiskPop(z) означає, що поєднання двох різних режимів роботи системи з різним рівнем ризику не дає «виграшу» в безпеці. Таким чином, $\lambda\,\mathrm{RiskPop}(z)$ змушує політику репрезентативного агента враховувати, що індивідуальні рішення в агресивно атакованому середовищі мають бути жорсткішими, навіть якщо локальна конфігурація здається відносно безпечною.

На основі миттєвої винагороди середнього поля, було сформьовано глобальний критерій оптимізації J_γ, який використовується для синтезу політики π та архітектурних параметрів РКС. Критерій оптимізації задамо функціоналом:

$$J_\gamma(\pi, \mu_0) = \mathbb{E}\Big[\ \sum_{t=0}^{\infty} \gamma^t\, r_{MF}(s_t, a_t, z_t)\ \Big|\ \mu_0\ \Big]. \tag{3.9}$$

3.1.6 Синтез локальної політики репрезентативного агента

Синтез локальної політики репрезентативного агента передбачає абстрактний опис динаміки та функції винагороди перетворюється на конкретне правило прийняття рішень, яке згодом узгоджено застосовується до всіх вузлів РКС. Політика будується на просторі станів сталої розмірності, заданому парою (s, z), і жодним чином не залежить від кількості агентів N. Це й забезпечує відсутність експоненційного зростання складності під час масштабування РКС.

На першому підкроці введемо параметризований клас політик $\pi_\theta(a \mid s, z)$, де $\pi_\theta(a \mid s, z)$ – умовний розподіл ймовірностей вибору дії a за умови, що розширений стан репрезентативного агента дорівнює (s, z); s – локальний стан вузла, тобто елементом простору $\mathcal{S}^{(k)}$ для деякого фіксованого типу k; z – вектор агрегатів середнього поля, що належить простору $\mathbb{R}^d$; θ – вектор параметрів, який задає конкретну політику всередині параметризованого класу, кожне

фіксоване значення θ однозначно визначає правило, за яким репрезентативний агент перетворює спостережуваний стан (s, z) у розподіл на множині дій.

Важливо, що вектор параметрів θ не залежить від кількості вузлів у популяції N. Незалежність θ від N фіксує формальний факт, що задача керування ставиться і розв'язується на рівні репрезентативного агента у просторі станів сталої розмірності, де у ролі «середовища» виступають лише агреговані популяційні характеристики, а не повний вектор станів усіх агентів.

Другий підкрок полягає у описі оптимальної довгострокової цінності кожного розширеного стану (s, z) для репрезентативного агента засобами рівняння Беллмана, яке формулює баланс між миттєвою винагородою та очікуваною майбутньою цінністю й описує оптимальну поведінку репрезентативного агента в РКС. Оскільки агент спирається на агреговані характеристики системи, рівняння визначає політику дій, що максимізує дисконтовану суму майбутніх винагород з урахуванням впливу локальних рішень на стан РКС.

Отже, функцію цінності $V(s, z)$ для дисконтованої постановки задачі керування, максимальне можливе за всіх допустимих політик очікуване значення сумарної дисконтованої винагороди, яку отримає агент у майбутньому, починаючи з розширеного стану репрезентативного агента (s, z) запишемо так:

$$V(s, z) = \max_{a \in \mathcal{A}} [r_{MF}(s, a, z) + \gamma \, \mathbb{E}[V(s', z') \mid s, a, z]]. \quad (3.10)$$

де s – локальний стан репрезентативного вузла; z – вектор агрегатів середнього поля, які описують популяційний контекст; $\max_{a \in \mathcal{A}}$ – максимізація по множині допустимих дій репрезентативного агента; $\mathcal{A}$ – узагальнене позначення простору дій для даного типу вузла, де кожна дія $a \in \mathcal{A}$ є кандидатом на вибір у стані (s, z); $r_{MF}(s, a, z)$ – миттєва винагорода середнього поля (3.8); $\mathbb{E}[V(s', z') \mid s, a, z]$ – оператор математичного сподівання, який визначає очікуване значення функції цінності в наступному розширеному стані (s', z') за умови, що поточний стан дорівнює (s, z), а обрана дія дорівнює a; s'– випадкова змінна, яка відображає локальний стан агента у наступний момент часу; z'– випадковий вектор агрегатів середнього поля. Ці змінні формуються згідно з

перехідною моделлю системи; $\gamma\,\mathbb{E}[V(s',z') \mid s,a,z]$ репрезентує очікувану дисконтовану майбутню цінність, якщо у стані (s,z) обрати дію a і далі діяти оптимально.

Рівняння (3.10) задає стаціонарну умову оптимальності. Воно стверджує, що оптимальна цінність $V(s,z)$ у будь-якому стані дорівнює максимальному за діями значенню миттєвої винагороди разом із дисконтованою очікуваною майбутньою цінністю. При цьому в цьому рівнянні не фігурує кількість вузлів N; вона впливає лише опосередковано через точність відтворення агрегованої динаміки РКС для скінченної кількості вузлів.

Задамо політику в межах параметризованого класу як π_θ тоді θ^* – оптимальний вектор параметрів, для якого досягається максимум обраного глобального критерію, визначимо так:

$$\theta^* = \arg\max_{\theta\in\Theta} J(\pi_\theta, \mu_0), \tag{3.11}$$

де $\arg\max_{\theta\in\Theta}$ – множина значень параметра θ, які максимізують функціонал J, а $J(\pi_\theta, \mu_0)$ – обраний глобальний критерій ефективності при використанні політики π_θ і початковому розподілі μ_0. Для кожного допустимого θ функція $\pi_\theta(a \mid s,z)$ задає конкретну політику і завдання оптимізації полягає у знаходженні такого θ^*, за якого відповідна політика забезпечує найкращий баланс між локальною ефективністю виявлення, мінімізацією хибних спрацювань та зменшенням глобального ризику.

Отримання θ^* здійснюється засобами методів навчання з підкріпленням, де вектор θ оновлюється за стохастичними градієнтними, де параметризована політика оновлюється на основі оцінок функції виграшу, які обчислює критик на основі $(s_t, z_t, a_t, r_{MF,t})$.

3.1.7 Формування оптимальної політики репрезентативного агента

Наступним кроком методу є формулювання вимоги узгодженості між тим, як політика репрезентативного агента змінює агрегований стан РКС, і тим, як саме цей агрегований стан визначає оптимальну політику. Таким чином, виникає

задача пошуку такої пари «політика – динаміка РКС», яку в контексті протидії атакам соціальної інженерії трактуватимемо як узгоджений режим функціонування системи. Умова узгодженості формулюється так: оптимальна політика репрезентативного агента має бути оптимальною саме для тієї динаміки РКС, яку ця політика сама породжує на рівні популяції вузлів, тобто правила поведінки (політика) окремих агентів і загальна динаміка системи (агреговані характеристики РКС) мають з часом прийти до стабільного стану. У цьому стані рішення, які агент приймає на локальному рівні, не суперечать загальній реакції всієї системи.

Задамо оптимальну політику позначено як π^*. Умова узгодженості означає, що політика π^* є розв'язком рівняння Беллмана для динаміки, яка відповідає еволюції μ_t, а сама еволюція μ_t утворюється, якщо в мастер-рівнянні еволюції популяції використати саме політику π^*. Тобто π^* оптимальна для «свого» середовища, а «своє» середовище є результатом дії π^* на великій популяції.

Дану концепцію подамо у вигляді системи двох рівнянь, де перше визначає оптимальну політику репрезентативного агента π^*, а друге описує динаміку агрегованих характеристик РКС, яка виникає внаслідок дії цієї політики:

$$\pi^* = \arg\max_{\pi} J(\pi, \mu), \quad \mu = \mathcal{G}(\pi^*) \tag{3.12}$$

де – $\arg\max_{\pi} J(\pi, \mu)$–множина (або точка) значень політики π, для яких функціонал $J(\pi, \mu)$ досягає максимального значення; π – довільна допустима політика з обраного класу політик, $\pi_\theta(a \mid s, z)$; $J(\pi, \mu)$ – глобальний критерій ефективності; μ–узагальнена характеристика динаміки популяції, що індукується вибором певної траєкторії станів середнього поля.

Значенням $\mathcal{G}(\pi^*)$ є популяційний опис у термінах середнього поля μ, який який визначається рівнянням:

$$\mu_{t+1} = \mathcal{F}(\mu_t, \pi^*), \tag{3.13}$$

де $\mathcal{F}$ – оператор еволюції, а μ_0–початковий розподіл.

Розв'язком даної системи рівнянь є пара (π^*, μ), яка одночасно задовольняє умову оптимальності політики та еволюцію середнього поля; така пара

інтерпретується як рівновага середнього поля або як оптимальний контроль у самоузгодженій моделі. Для синтезу архітектури РКС, стійкої до атак соціальної інженерії, така рівновага відповідає конфігурації системи, у якій локальні політики детекції та блокування атак і глобальний профіль ризику є взаємно узгодженими. Це забезпечує стійкий режим функціонування РКС із мінімально досяжним системним ризиком за наявних обмежень.

Знаходження такого стану реалізується як ітераційна процедура, у якій ітераційно уточнюються політика репрезентативного агента $\pi^{(k)}$ та агрегований стан РКС. Далі, за фіксованої оновленої політики $\pi^{(k+1)}$, виконується уточнення агрегованого опису стану РКС $\mu^{(k)}$. Для цього обчислюють еволюцію популяційного розподілу, розв'язуючи мастер-рівняння:

$$\mu_{t+1} = \mathcal{F}(\mu_t, \pi^{(k+1)}). \tag{3.14}$$

Результат цього кроку запишемо як $\mu^{(k+1)} = \mathcal{G}(\pi^{(k+1)})$, де $\mathcal{G}$ має той самий зміст, що й у системі фіксованого пункту: для заданої політики він повертає еволюцію агрегованого стану РКС, яку ця політика породжує на рівні популяції вузлів. Цей крок інтерпретуватимемо оцінювання політики у масштабі всієї системи: визначається, до якого глобального профілю ризику приводить нова політика за умови її широкого застосування.

Повна ітераційна схема має вигляд послідовності $\mu^{(k)} \longrightarrow \pi^{(k+1)} \longrightarrow \mu^{(k+1)}$, де перша відповідність відображає синтез політики репрезентативного агента за фіксованого середньопольового середовища, а друга відповідність – оновлення середньопольового середовища за фіксованої політики. Індекс k збільшується і процес продовжується послідовно збільшується, а ітерації тривають до моменту, коли пара $(\pi^{(k)}, \mu^{(k)})$ наблизиться до фіксованого пункту у заданій метриці. Це означає, що в граничному випадку при $k \to \infty$ реалізується збіжність $\pi^{(k)} \to \pi^*$та $\mu^{(k)} \to \mu$, де (π^*, μ) задовольняють рівняння (3.13).

У цій ітераційній конструкції критичним є те, що всі обчислення на етапі синтезу політики відбуваються у просторі станів (s, z) сталої розмірності; перехід до простору розподілів μ_t потрібен лише для оновлення

середньопольового середовища через відповідні оператори $\mathcal{F}$ та $\mathcal{G}$. Кількість агентів N пливає на схему лише через точність відтворення агрегованої динаміки РКС, але не визначає розмірність задачі оптимального керування, яку розв'язує репрезентативний агент. Завдяки цьому синтез масштабованої архітектури РКС, стійкої до атак соціальної інженерії, зводиться до задачі пошуку узгодженого фіксованого режиму роботи системи, який може бути реалізований без експоненційного зростання простору станів і дій.

3.1.8 Інтеграція архітектурних параметрів у опис РКС

Інтеграція архітектурних параметрів у опис РКС є тим етапом, на якому багатокомпонентна стохастична модель перетворюється на практичний інструмент проєктування архітектури системи. На цьому рівні не лише політика репрезентативного агента, а й сама конфігурація архітектури, зокрема, щільність вузлів, частка вузлів із багатофакторною автентифікацією, глибина ескалаційних ланцюжків та топологія мережевої сегментації, безпосередньо включаються до агрегованих характеристик системи та впливають на популяційний ризик і динаміку атак.

Вихідною точкою є опис популяції вузлів РКС через розподіл їхніх станів у момент дискретного часу μ_t та його компактну параметризацію $z_t = \psi(\mu_t)$, $\mu_t = (\mu_t^{(1)}, \dots, \mu_t^{(K)})$ згідно (3.3). На етапі інтеграції архітектури до агрегованих характеристик РКС включають не лише поточні популяційні показники, але й статичні архітектурні параметри, що визначають умови поширення атак та ефективність захисних механізмів. Для формального включення архітектурних параметрів введемо вектор κ, який позначає сукупність усіх параметрів конфігурації РКС, які розглядаються як змінні дизайну. Вектор κ належить деякому параметричному просторі $\mathcal{K} \subseteq \mathbb{R}^q$, де q – розмірність параметризації архітектури.

У підсумку топологія мережі характеризується системою топологічних індикаторів, зокрема середнім ступенем вузла, кількістю сегментів мережі,

середньою довжиною найкоротших шляхів між критичними вузлами, коефіцієнтом кластеризації та іншими структурними параметрами.

Усі ці величини представлено як компоненти вектора κ, де кожна координата кодує один структурний аспект архітектури.

Далі агреговані характеристики РКС визначаються не лише розподілом μ_t, а й архітектурною конфігурацією:

$$z_t = \psi(\mu_t, \kappa), \tag{3.15}$$

де μ_t – популяційний розподіл станів, що відображає поточну динамічну конфігурацію системи, κ – статичний вектор архітектурних параметрів, який задається на рівні дизайну архітектури.

Важливо, що у цьому визначенні $\psi(\mu_t, \kappa)$ розмірність вектора z_t залишається сталою: хоча зазвичай її доводиться збільшити з d до d', значення d' все одно обирається фіксованим і не залежить від розміру популяції N. Таким чином, розширення агрегованих характеристик РКС за рахунок архітектурних параметрів не порушує ключової властивості масштабованості: оптимізація політики репрезентативного агента й надалі виконується в просторі станів сталої розмірності, але тепер цей простір додатково враховує структурні властивості архітектури системи.

Наступним кроком є формулювання комбінованої задачі оптимізації, у якій одночасно визначаються оптимальна політика керування та оптимальні архітектурні параметри:

$$\max_{\pi,\kappa} J(\pi, \kappa), \tag{3.16}$$

де $\max_{\pi,\kappa}$– означає, що шукається пара (π^*, κ^*), яка максимізує функціонал J за всіма допустимими політиками π та допустимими архітектурними параметрами κ.

Допустимість κ задається множиною обмежень, до яких належать обмеження на вартість апаратної інфраструктури, на доступні людські ресурси, на максимальне допустиме затримування користувачів, а також вимоги угод про рівень сервісу.

Задамо функцію $C_{\text{res}}(\kappa)$, яка описує сумарні ресурсні витрати, асоційовані з архітектурною конфігурацією κ, та константу $C_{\max}$ – максимально допустимий бюджет. Тоді обмеження записується як $C_{\text{res}}(\kappa) \leq C_{\max}$. Аналогічно, функція $SLA(\pi, \kappa)$ описуватиме середній показник часу відповіді для легітимного користувача, а константа $SLA_{\min}$ – мінімально допустимий рівень обслуговування; тоді накладається умова $SLA(\pi, \kappa) \geq SLA_{\min}$. Описані обмеження визначають допустиму область у просторі (π, κ), в межах якої виконується максимізація $J(\pi, \kappa)$.

У такій постановці динаміка РКС враховується через залежність $J(\pi, \kappa)$ від розподілу μ_t, який еволюціонує згідно з рівнянням $\mu_{t+1} = \mathcal{F}(\mu_t, \pi, \kappa)$, та через параметризацію $z_t = \phi(\mu_t, \kappa)$. Залежність $\mathcal{F}$ від κ означає, що архітектура впливає на ймовірності переходів між станами, зокрема, через зміну ймовірності успішного поширення атаки між вузлами або через зміну розподілу затримок і інтенсивностей ескалаційних атак.

У рамках єдиної оптимізаційної задачі було визначено критерії масштабованості, що дозволяють оцінити стійкість архітектури κ^* та політики π^* до збільшення розміру популяції N. Одним з таких критеріїв являється поведінка функціонала популяційного ризику *RiskPop* при зростанні N. Для цього було введено функцію ризик $R(N; \pi, \kappa)$ для очікуваного значення $\text{RiskPop}(z_t)$ у стаціонарному режимі для популяції розміру N вузлів РКС, коли всі агенти застосовують політику π у архітектурі, заданій κ.

Умовою масштабованості вважатимемо таку властивість, коли існує межа $\limsup_{N\to\infty} R(N; \pi^*, \kappa^*)$, яка є скінченною і не перевищує допустимого порогу ризику. Це означає, що зі збільшенням кількості вузлів РКС популяційний ризик не зростає безконтрольно, а залишається обмеженим завдяки обраній архітектурі та політиці.

Таким чином, інтеграція архітектурних параметрів у опис середнього поля через вектор κ, розширену параметризацію $z_t = \psi(\mu_t, \kappa)$ та спільну оптимізаційну постановку $\max_{\pi,\kappa} J(\pi, \kappa)$ з ресурсними та обмеженнями переводить опис динаміки РКС на рівень повноцінного інструмента синтезу

масштабованої архітектури з відсутністю експоненційного зростання ні у просторі станів і дій, ні в ресурсних вимогах РКС.

3.1.9 Синтез масштабованої архітектури

Процес синтезу масштабованої архітектури формалізує попередні теоретичні побудови у вигляді ітераційної процедури, яка одночасно наближає оптимальну політику репрезентативного агента та оптимальну конфігурацію архітектурних параметрів. На цьому етапі поєднуються дві часові шкали: внутрішня відповідає швидкій адаптації політики за фіксованої архітектури, тоді як зовнішня описує повільну еволюцію самої архітектури на основі агрегованих показників ефективності та масштабованості.

Глобальний стан РКС подається через популяційний розподіл μ_t та його компактну параметризацію z_t, а збільшення кількості агентів N впливає лише на точність такого статистичного опису, але не змінює розмірність задачі керування, завдяки чому зберігається масштабованість запропонованого методу синтезу.

Початкова стадія, полягає у заданні стартових наближень для архітектурної конфігурації та популяційного стану. Архітектуру задамо вектором $\kappa^{(0)}$, де κ описує вектор архітектурних параметрів, що належить простору $\mathcal{K} \subseteq \mathbb{R}^q$, де q є фіксованою розмірністю параметризації архітектури, 0 – початкова ітерація зовнішнього циклу. Одночасно задамо початковий опис популяції у компактній двох можливих формах через агреговані характеристики, що відображають початковий профіль ризику та поведінкові особливості РКС $\mu^{(0)}$, та через його компактну параметризацію $z^{(0)}$. У першому випадку $\mu^{(0)}$ – це початковий популяційний стан, тобто система мір $\mu_0^{(k)}$ на локальних просторах станів для кожного типу вузлів k. У другому випадку $z^{(0)}$ є вектором агрегованих показників, який отримується за допомогою відображення ϕ, тобто

$$z^{(0)} = \psi(\mu^{(0)}, \kappa^{(0)}), \tag{3.17}$$

де ψ – функціоне відображення, яке зіставляє популяційний розподіл та архітектурну конфігурацію з вектором ознак середнього поля у відповідному просторі $\mathbb{R}^{d'}$ сталої розмірності. Вибір конкретного способу ініціалізації (через $\mu^{(0)}$ чи через $z^{(0)}$) залежить від того, чи будується модель від абстрактного агрегованого опису, чи від реальних даних журналів подій і топології, але в обох випадках початковий стан не містить залежності розмірності від N.

Внутрішній цикл працює при фіксованому векторі архітектурних параметрів κ, що на даний момент відповідає деякому наближенню $\kappa^{(\ell)}$, де ℓ є індексом ітерації зовнішнього циклу. Для фіксованого $\kappa^{(\ell)}$ задача зводиться до синтезу політики репрезентативного агента на просторі станів (s, z). Формально ця політика задається параметризованим сімейством $\pi_\theta(a \mid s, z)$, де π_θ – відображення політики, параметризованої вектором параметрів θ, a являється відображенням дії, s – локальний стан вузла, а z – вектор агрегованих характеристик середнього поля. За фіксованої архітектури $\kappa^{(\ell)}$ виконується розв'язання рівняння Беллмана (3.10) для функції цінності $V^{(\ell)}(s, z)$, де (ℓ) маркує залежність від поточної архітектурної конфігурації.

Розв'язок цього рівняння дає змогу побудувати наближено оптимальну політику $\pi^{(\ell)}$, яка на практиці реалізується через налаштування параметрів $\theta^{(\ell)}$ вектору θ так, щоб політика $\pi_{\theta^{(\ell)}}$ наближала оптимальну політику для даної архітектури. Після отримання політики $\pi^{(\ell)}$ внутрішній цикл продовжується обчисленням оновленої динаміки популяції вузлів РКС, тобто того, як змінюється розподіл станів системи за умови масового застосування нової політики.

Динамічний популяційний стан опишеми траєкторією $\{\mu_t^{(\ell)}\}_{t\geq 0}$, яка задовольняє мастер-рівнянню:

$$\mu_{t+1}^{(\ell)} = \mathcal{F}\left(\mu_t^{(\ell)}, \pi^{(\ell)}, \kappa^{(\ell)}\right), \tag{3.18}$$

де $\mu_{t+1}^{(\ell)}$ – популяційний розподіл у момент часу $t+1$ у конфігурації, що відповідає ітерації ℓ, $\mathcal{F}$ – еволюції популяції, який будується на основі локальної динаміки агентів і топології архітектури, а аргументи $\pi^{(\ell)}$ і $\kappa^{(\ell)}$ задають

відповідно політику і архітектурні параметри. Траєкторію $\mu_t^{(\ell)}$ інтегрують або до досягнення стаціонарного розподілу $\mu_\infty^{(\ell)}$, або до квазістаціонарного режиму на довгому, але скінченному горизонті.

Відповідну параметризацію середнього поля обчислимо так:

$$z_t^{(\ell)} = \psi(\mu_t^{(\ell)}, \kappa^{(\ell)}), \tag{3.19}$$

де $z_t^{(\ell)}$ – вектор агрегатів середнього поля у момент часу t при конфігурації архітектури $\kappa^{(\ell)}$. У стаціонарному випадку використаємо характеристику :

$$z_\infty^{(\ell)} = \psi(\mu_\infty^{(\ell)}, \kappa^{(\ell)}). \tag{3.20}$$

Таким чином, внутрішній цикл, починаючи з $\kappa^{(\ell)}$, послідовно формує наближено оптимальну політику $\pi^{(\ell)}$ на просторі сталих розмірностей (s, z) а відповідний стаціонарний популяційний режим роботи РКС $(\mu^{(\ell)}, z^{(\ell)})$, при цьому ні розмірність простору станів, ні розмірність параметрів не залежать від кількості агентів N.

Зовнішній цикл, оперує архітектурною конфігурацією, використовуючи результати внутрішнього циклу для оцінки якості поточної пари $(\pi^{(\ell)}, \kappa^{(\ell)})$. Функція $J(\pi, \kappa)$ узагальнює поведінку системи за довгий часовий проміжок як очікувану дисконтовану сумарну винагороду. Тоді у точці $(\pi^{(\ell)}, \kappa^{(\ell)})$ обчислюється значення глобального критерію $J(\pi^{(\ell)}, \kappa^{(\ell)})$. Оновлення архітектурних параметрів задамо ітераціно:

$$\kappa^{(\ell+1)} = \kappa^{(\ell)} + \eta_\ell \, \nabla_\kappa J(\pi^{(\ell)}, \kappa^{(\ell)}), \tag{3.21}$$

де $\kappa^{(\ell+1)}$ – нове наближення архітектурної конфігурації; η_ℓ – крок градієнтного оновлення на ітерації ℓ; $\nabla_\kappa J(\pi^{(\ell)}, \kappa^{(\ell)})$ – градієнт критерію якості за вектором архітектурних параметрів у точці $(\pi^{(\ell)}, \kappa^{(\ell)})$, здійснюється засобами алгоритмів підкріплювального навчання.

Процес повторюється до збіжності, що означає, що послідовність $\{\kappa^{(\ell)}\}_{\ell \geq 0}$ стабілізується в околі деякого вектора κ^*, а послідовність політик $\{\pi^{(\ell)}\}_{\ell \geq 0}$, одержуваних внутрішнім циклом для кожної $\kappa^{(\ell)}$, прямує до певної політики π^*:

$$\lim_{\ell \to \infty} \kappa^{(\ell)} = \kappa^*, \lim_{\ell \to \infty} \pi^{(\ell)} = \pi^*, \tag{3.22}$$

де $\lim_{\ell\to\infty}$ – границя послідовності при зростанні ℓ, а κ^* і π^* є відповідно граничними архітектурною конфігурацією та політикою. У цій точці метод фіксує пару (κ^*, π^*) як результат синтезу масштабованої архітектури.

Ключовим наслідком такої побудови є те, що і внутрішній, і зовнішній цикли працюють у просторах сталих розмірностей. Внутрішній цикл вирішує задачу оптимального керування в просторі станів (s, z), де розмірність локального стану s та вектора характеристик середнього поля z не залежить від кількості агентів N. Зовнішній цикл оптимізує вектор κ у просторі $\mathbb{R}^q$, де q також є фіксованим. Усі залежності від N входять лише опосередковано – через форму оператора $\mathcal{F}$ та точність апроксимації середнього поля. Це формально закріплює відсутність експоненційного зростання простору станів і дій у методі синтезу і підтверджує масштабованість побудованої архітектури РКС, стійкої до атак соціальної інженерії.

3.1.10 Аналіз масштабованості

Аналіз масштабованості завершує побудову методу, оскільки саме на цьому етапі формально обґрунтовується, що використана узагальнена популяційна динаміка не є лише зручною евристикою, а справді наближує поведінку багатоагентної РКС із контрольованою похибкою та без експоненційного зростання обчислювальної складності при збільшенні числа вузлів РКС. У контексті протидії атакам соціальної інженерії це означає, що синтезована політика та архітектурна конфігурація залишаються застосовними до великих корпоративних РКС, а рівень захисту не деградує неконтрольовано внаслідок збільшення кількості вузлів і зростання обсягів подій.

Для формального опису апроксимаційних властивостей розглядається кінцева багатоагентна система, у якій кожен вузол РКС представлено агентом-детектором, що виконує локальне прийняття рішень згідно з політикою, отриманою в узагальненій популяційній моделі. Стани всієї системи позначаються вектором:

$$s_t^N = (s_{1,t}, \dots, s_{N,t}), \tag{3.33}$$

де s_t^N – глобальний стан у момент дискретного часу t, N – кількість агентів, $s_{i,t}$ – локальний стан i-го агента у момент часу t. Кожний агент застосовує політику, яка є копією політики, синтезованої для репрезентативного вузла РКС на просторі агрегованих станів π^{MF}, побудованої для репрезентативного агента на просторі станів (s, z).

Формально емпіричний розподіл позначимо:

$$\mu_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{s_{i,t}}, \tag{3.24}$$

де μ_t^N – випадкова міра на просторі локальних станів, $\delta_{s_{i,t}}$ – точкова міра, зосереджена в точці $s_{i,t}$.

Агреговані характеристики системи задамо через відображення:

$$z_t^N = \psi(\mu_t^N, \kappa), \tag{3.25}$$

де ϕ – той самий оператор параметризації, що використовувався в моделі середнього поля, а κ – вектор архітектурних параметрів.

Таким чином, політика π^{MF}, отримана для репрезентативного вузла РКС, переноситься на N-агентну систему шляхом заміни ідеалізованого агрегованого стану z_t сформовану за фактичним розподілом станів у популяції вузлів z_t^N.

Для оцінки якості такої апроксимації було введено ε-Nash наближення. Нехай для i-го агента очікуваний дисконтований функціонал якості при роботі всієї системи за політикою π^{MF} позначимо як:

$$J_i^N(\pi^{MF}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{MF}(s_{i,t}, a_{i,t}, z_t^N)], \tag{3.37}$$

де $\mathbb{E}$ – математичне сподівання за всіма випадковими траєкторіями системи; сумування ведеться за дискретним часом t від нуля до нескінченності; параметр γ – коефіцієнт дисконтування, $\gamma=\overline{0,1}$; функція $r_{MF}(s_{i,t}, a_{i,t}, z_t^N)$ слугує для означення миттєвої винагороди агента i у момент часу t, яка залежить від його локального стану $s_{i,t}$, локальної дії $a_{i,t}$ та емпіричного середнього поля у z_t^N.

Нехай тепер агент i однобічно відхиляється від політики π^{MF} і використовує іншу допустиму політику $\tilde{\pi}_i$, тоді як усі інші агенти продовжують застосовувати політику π^{MF}.

Відповідний функціонал якості для агента i визначимо як:

$$J_i^N(\tilde{\pi}_i, \pi_{-i}^{MF}), \tag{3.38}$$

де $\tilde{\pi}_i$– політика, яку використовує тільки агент i, π_{-i}^{MF}–вектор політик усіх агентів, окрім i, які діють за політикою π^{MF}.

Політику π^{MF} називатимемо ε-Nash наближенням, якщо для всіх агентів i та всіх допустимих однобічних відхилень $\tilde{\pi}_i$ виконується нерівність

$$J_i^N(\pi^{MF}) \geq J_i^N\left(\tilde{\pi}_i, \pi_{-i}^{MF}\right) - \varepsilon(N), \tag{3.39}$$

де $\varepsilon(N)$ – невід'ємна функція від числа агентів N, яка характеризує максимальний можливий виграш окремого агента від однобічного відхилення від спільної політики, що застосовується всіма вузлами РКС; якщо $\varepsilon(N)$ прямує до нуля при $N \to \infty$, то спільна політика, побудована на основі агрегованих характеристик системи асимптотично стає істинною Nash-рівновагою.

Тоді оцінки для багатовузлових РКС даватимуть залежність виду :

$$\varepsilon(N) \leq \frac{\mathrm{C}}{\sqrt{N}}, \tag{3.40}$$

де C – це коефіцієнт, який показує, наскільки швидко ε-Nash наближення покращується зі зростанням кількості агентів.

Для наочності послідовність основних етапів методу та відповідні формальні залежності узагальнено у вигляді схеми на рисунку 3.1.

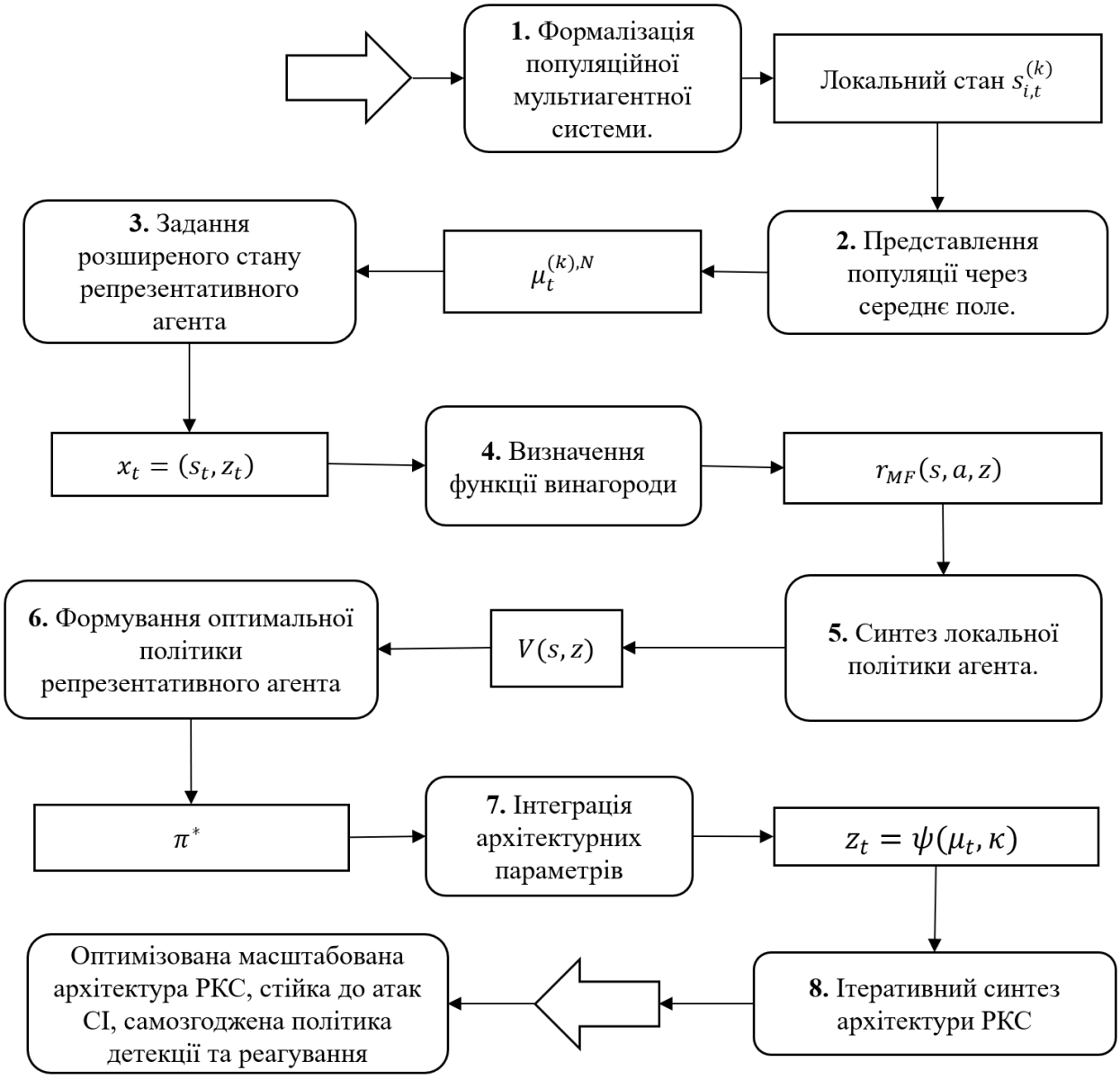


Рис. 3.1 Схема методу забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії

3.1.11 Експериментальні дослідження методу

Задамо фіксовано локальний простір станів $\mathcal{S}$ та дій $\mathcal{A}$. Політика π^{MF} синтезована для репрезентативного вузла РКС, тренується один раз; цей час не залежить від N. Далі для кожного N моделюється $T = 1000$ кроків N–агентної системи, де всі агенти використовують одну і ту саму π^{MF}.

Параметри потоку подій та політики такі:

1) імовірність атаки: $p_{\text{attack}} = 0.05$;
2) імовірність правильного блокування атаки: $p_{\text{detect}} = 0.9$;
3) імовірність хибного блокування легітимної події: $p_{\text{fp}} = 0.02$.

Для кожного N було обчислено (Таблиця 3.1):

1) час тренування політики середнього поля (той самий для всіх N);
2) сумарний час симуляції/планування для N–агентної системи;
3) час на один крок для одного агента;
4) показники якості: TPR, FPR, FN–rate, FP–rate, простий ризиковий індекс $\text{risk} = \text{FN\_rate} + 0.1 \cdot \text{FP\_rate}$.

Таблиця 3.1–Результати для різних N (усі часи в секундах, агентний час у мікросекундах)

N	час навчання	загальний час оцінювання	час оцінювання на один крок агента	TPR	FPR	FNR	FPR	метрика ризику
5	0.00039	0.02426	0.31	0.9066	0.0158	0.0934	0.0158	0.0950
20	0.00039	0.02355	0.43	0.8969	0.0208	0.1031	0.0208	0.1052
50	0.00039	0.02812	0.56	0.8966	0.0200	0.1034	0.0200	0.1054
100	0.00039	0.02484	0.25	0.8927	0.0220	0.1073	0.0220	0.1095
200	0.00039	0.07616	0.38	0.9005	0.0200	0.0995	0.0200	0.1015
500	0.00039	0.03614	0.07	0.8978	0.0202	0.1022	0.0202	0.1042
1000	0.00039	0.04441	0.04	0.8985	0.0202	0.1015	0.0202	0.1035
5000	0.00039	0.12243	0.02	0.8996	0.0200	0.1005	0.0200	0.1025
10000	0.00039	0.18338	0.02	0.8996	0.0201	0.1004	0.0201	0.1024
100000	0.00039	1.71601	0.02	0.9001	0.0200	0.0999	0.0200	0.1019

Аналіз результатів експериментів показав, що час навчання політики середнього поля є величина 0.00039с, і вона однакова для всіх N, оскільки навчання відбувається один раз у просторі (s, z) сталої розмірності й не містить залежності від кількості агентів.

Час сумарний планування eval_time_total росте з N приблизно лінійно, але час на один крок для одного агента швидко стабілізується на рівні $\approx 0.02 - 0.06$

мікросекунди, що демонструє відсутність експоненційного росту обчислювальної вартості.

Значення TPR коливається біля теоретичного 0.9, FPR–біля теоретичного 0.02; risk_metric стабілізується в інтервалі $\approx 0.10 - 0.11$. При зростанні N флуктуації зменшуються, що відповідає ε–Nash апроксимації з похибкою порядку $O(1/\sqrt{N})$.

3.2 Метод комплексного оцінювання стійкості РКС до атак соціальної інженерії

У дослідженні також було розроблено метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів.

Система критеріїв ефективності синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, має відображати всі ключові аспекти запропонованих наукових результатів: стійкість багатоагентної архітектури, достовірність виявлення атак, адаптивність, масштабованість, живучість та ефективність прийняття колективних рішень. Кожен з цих аспектів повинен мати формалізовані кількісні показники, які обчислимо за результатами моделювання або експериментальної експлуатації синтезованої системи.

Нехай для фіксованих параметрів синтезу θ та сценарію атак ω, які належать відповідно множинам Θ та Ω, система функціонує впродовж дискретного часового інтервалу з моментів $t = 0,1,\dots,T$, де T–довжина горизонту спостереження. Для кожної пари (θ, ω) побудуємо вектор часткових критеріїв ефективності синтезу вектор критеріїв $c(\theta, \omega)$:

$$c(\theta,\omega) = (c_{\text{arch}}(\theta,\omega), c_{\text{det}}(\theta,\omega), c_{\text{adapt}}(\theta,\omega), c_{\text{scal}}(\theta,\omega), c_{\text{surv}}(\theta,\omega), c_{\text{dec}}(\theta,\omega)), \quad (3.41)$$

де $c_{\text{arch}}(\theta,\omega)$ – критерій стійкості багатоагентної архітектури; $c_{\text{det}}(\theta,\omega)$ – критерій достовірності виявлення атак; $c_{\text{adapt}}(\theta,\omega)$ – критерієм адаптивності; $c_{\text{scal}}(\theta,\omega)$ – критерій масштабованості; $c_{\text{surv}}(\theta,\omega)$ – критерій живучості; $c_{\text{dec}}(\theta,\omega)$ – критерій ефективності прийняття колективних рішень. Кожен із зазначених критеріїв будується на основі набору первинних індикаторів, які в сукупності формують інтегральну систему показників поведінкової, структурної та функціональної стійкості.

Стійкість багатоагентної архітектури характеризується тим, наскільки узгоджено система реагує на впливи атак соціальної інженерії, зберігаючи при цьому стабільність колективної поведінки та прийнятних режимів функціонування. Для цього розглядається помилка консенсусу між агентами. Нехай $\mathcal{A} = \{1,2,\dots,N\}$ – множина агентів, N – кількість агентів, $x_i(t;\theta,\omega)$ – локальний стан i-го агента в момент часу t за параметрів синтезу θ та сценарію атак ω, а $\bar{x}(t;\theta,\omega)$ – середній стан агентів у момент часу t, де:

$$\bar{x}(t;\theta,\omega) = \frac{1}{N}\sum_{i=1}^{N} x_i(t;\theta,\omega). \tag{3.42}$$

Середньоквадратична помилка консенсусу за весь час спостереження визначатимемо як

$$E_{\text{cons}}(\theta,\omega) = \frac{1}{T+1}\sum_{t=0}^{T}\frac{1}{N}\sum_{i=1}^{N} \| x_i(t;\theta,\omega) - \bar{x}(t;\theta,\omega) \|^2, \tag{3.43}$$

де $E_{\text{cons}}(\theta,\omega)$–усереднена помилка консенсусу; $\|\cdot\|$–евклідову норму відповідного вектора. На основі цієї величини критерієм стійкості архітектури може бути функція

$$c_{\text{arch}}(\theta,\omega) = \exp\left(-\gamma_{\text{arch}} E_{\text{cons}}(\theta,\omega)\right), \tag{3.44}$$

де $c_{\text{arch}}(\theta,\omega)$ набуває значень у діапазоні від нуля до одиниці; $\gamma_{\text{arch}} > 0$– параметр масштабу, що визначає чутливість критерію до величини помилки консенсусу. Чим меншою є помилка консенсусу, тим ближчим до одиниці є значення $c_{\text{arch}}(\theta,\omega)$, що відображає високу стійкість архітектури до розбалансування взаємодії агентів під впливом атак.

Достовірність виявлення атак соціальної інженерії розглядається як ключова характеристика в дослідженні ефективності синтезу РКС, стійких до

ACI, оскільки вона визначає, наскільки надійно система відрізняє впливи соціальної інженерії від легітимної активності користувачів і сервісів. Формалізація цієї характеристики спирається на стандартний апарат метрик двокласової класифікації, при цьому кожна метрика розглядається як частковий критерій, а їхня сукупність інтегрується у єдиний показник достовірності: True Positive Rate; False Positive Rate; Precision; Recall; F1–score; MCCм.

Метрику частки атак, які були коректно виявлені серед усіх реально присутніх атак $\mathrm{TPR}(\theta, \omega)$, обчислимо:

$$\mathrm{TPR}(\theta, \omega) = \frac{TP(\theta,\omega)}{TP(\theta,\omega)+FN(\theta,\omega)}, \quad (3.45)$$

де $TP(\theta, \omega)$ – кількість коректно виявлених атак, $TP(\theta, \omega) + FN(\theta, \omega)$ – загальна кількість атак, що мали місце у вибірці. Значення $\mathrm{TPR}(\theta, \omega)$, наближене до одиниці, означає, що система рідко пропускає атаки соціальної інженерії.

Хибно-позитивна частка (False Positive Rate, FPR) характеризує, наскільки система помилково реагує на легітимну активність як на атаку:

$$\mathrm{FPR}(\theta, \omega) = \frac{FP(\theta,\omega)}{FP(\theta,\omega)+TN(\theta,\omega)}, \quad (3.46)$$

де $FP(\theta, \omega)$відображає кількість помилкових спрацьовувань, $FP(\theta, \omega) + TN(\theta, \omega)$ - загальна кількість легітимних об’єктів. Чим меншим є значення $\mathrm{FPR}(\theta, \omega)$, тим нижчою є схильність системи до надмірного блокування легітимних взаємодій.

Точність (Precision) характеризує, наскільки можна довіряти спрацьовуванням системи, коли вона заявляє про наявність атаки. Вона визначається співвідношенням:

$$\mathrm{Prec}(\theta, \omega) = \frac{TP(\theta,\omega)}{TP(\theta,\omega)+FP(\theta,\omega)}, \quad (3.47)$$

де $TP(\theta, \omega) + FP(\theta, \omega)$ – загальна кількість позитивних рішень, прийнятих системою. Висока точність означає, що більшість оголошених системою «атак» дійсно є атаками соціальної інженерії, а не легітимною активністю.

Повнота (Recall) у класичному означенні збігається з показником $\mathrm{TPR}(\theta, \omega)$ і описує здатність системи виявляти більшість атак, які реально присутні:

$$\mathrm{TPR}(\theta,\omega)=\frac{TP(\theta,\omega)}{TP(\theta,\omega)+FN(\theta,\omega)}. \quad (3.48)$$

Оскільки у практичних сценаріях важливо досягати одночасно і високої точності, і високої повноти, запроваджується F1–міра (F1–score), яка є гармонійним середнім між точністю та повнотою і визначається як:

$$\mathrm{F1}(\theta,\omega)=\frac{2\,\mathrm{Prec}(\theta,\omega)\,\mathrm{Rec}(\theta,\omega)}{\mathrm{Prec}(\theta,\omega)+\mathrm{Rec}(\theta,\omega)+\varepsilon}, \quad (3.49)$$

де множник 2 у чисельнику забезпечує симетричне врахування точності та повноти; величина $\varepsilon>0$ – мале додатне число, яке запобігає діленню на нуль у випадку, коли сума $\mathrm{Prec}(\theta,\omega)+\mathrm{Rec}(\theta,\omega)$ наближається до нуля. Значення $\mathrm{F1}(\theta,\omega)$, близьке до одиниці, свідчить про збалансовано високі значення точності й повноти.

Для глибокого аналізу достовірності, з урахуванням усіх елементів матриці невідповідностей, було використано коефіцієнт кореляції Метьюса (MCC), який оцінює кореляцію між справжніми мітками класу та рішеннями системи, будучи чутливою як до істинно-позитивних і істинно-негативних рішень, так і до обох типів помилок:

$$\mathrm{MCC}(\theta,\omega)=\frac{TP(\theta,\omega)\cdot TN(\theta,\omega)-FP(\theta,\omega)\cdot FN(\theta,\omega)}{\sqrt{(TP(\theta,\omega)+FP(\theta,\omega))(TP(\theta,\omega)+FN(\theta,\omega))(TN(\theta,\omega)+FP(\theta,\omega))(TN(\theta,\omega)+FN(\theta,\omega))}+\varepsilon}, \quad (3.50)$$

де у чисельнику стоїть різниця добутку істинно–позитивних та істинно-негативних рішень $TP(\theta,\omega)\cdot TN(\theta,\omega)$ і добутку хибно–позитивних та хибно-негативних рішень $FP(\theta,\omega)\cdot FN(\theta,\omega)$; у знаменнику стоїть квадратний корінь із добутку чотирьох сум, які відповідають усім можливим парам істинних і передбачених класів; $\varepsilon>0$є малим додатним числом, що гарантує числову стійкість при малих значеннях сум у знаменнику. Значення $\mathrm{MCC}(\theta,\omega)$ належить інтервалу від мінус одиниці до одиниці: значення, близькі до одиниці, означають сильну позитивну кореляцію між рішеннями системи та істинними мітками (високу достовірність), значення, близькі до нуля, відповідають майже випадковій класифікації, а значення, близькі до мінус одиниці, свідчать про систематично хибну класифікацію.

З метою побудови єдиного критерію достовірності для сценарію ω та параметрів θ інтегруються зазначені метрики в скалярний показник. Нехай w_{TPR}, w_{FPR}, w_{Prec}, w_{F1}, w_{MCC} – невід'ємні вагові коефіцієнти, що визначають відносну важливість відповідно чутливості, штрафу за хибні тривоги, точності, F1-міри та MCC, причому сума цих ваг нормується до одиниці. Для компактності розглядатимемо критерій достовірності так:

$$c_{\mathrm{rel}}(\theta,\omega) = w_{\mathrm{TPR}}\,\mathrm{TPR}(\theta,\omega) + w_{\mathrm{Prec}}\,\mathrm{Prec}(\theta,\omega) + w_{\mathrm{F1}}\,\mathrm{F1}(\theta,\omega) + w_{\mathrm{MCC}}\,\mathrm{MCC}(\theta,\omega) - w_{\mathrm{FPR}}\,\mathrm{FPR}(\theta,\omega), \quad (3.51)$$

де знак мінус перед доданком із $\mathrm{FPR}(\theta,\omega)$ відображає те, що зростання хибно-позитивної частки зменшує загальну достовірність системи; інші доданки з позитивними коефіцієнтами відображають позитивний внесок відповідних метрик. Для певних класів задач вагові коефіцієнти можуть налаштовуватися так, щоб підсилювати ті аспекти достовірності, які є критичними в конкретній корпоративній політиці безпеки, наприклад мінімізацію пропусків атак або мінімізацію надмірних блокувань.

Адаптивність синтезованої системи до змін у поведінці користувачів і атакуючих проявляється в тому, наскільки швидко і результативно змінюються стратегії агентів при зміні статистики вхідних даних. Нехай θ_0– початковий набір параметрів синтезу, а θ^*– адаптований набір параметрів після певного періоду навчання або переналаштування системи за нових умов. Нехай також $J(\theta)$ – інтегральна оцінка ефективності для вектора параметрів θ, визначена, як математичне сподівання певної функції від критеріїв за множиною сценаріїв атак. Тоді відносне покращення ефективності внаслідок адаптації визначимо як:

$$\Delta_{\mathrm{adapt}} = \frac{J(\theta^*) - J(\theta_0)}{|J(\theta_0)| + \varepsilon}, \quad (3.52)$$

де Δ_{adapt} – нормоване покращення ефективності; $\varepsilon > 0$ є малим додатним числом, що запобігає діленню на нуль у випадку $J(\theta_0) \approx 0$. Якщо T_{adapt} позначає кількість часових кроків або обсяг даних, необхідний для досягнення параметрів θ^*, то критерій адаптивності, який відображає здатність системи швидко покращувати свою ефективність, подамо:

$$c_{\text{adapt}}(\theta_0, \omega) = \frac{\Delta_{\text{adapt}}}{T_{\text{adapt}}}, \tag{3.53}$$

де більші значення цього критерію відповідають більшій адаптивності для фіксованої пари початкових параметрів і сценарію атак.

Масштабованість стосується того, як змінюються властивості системи при збільшенні кількості вузлів та агентів. Нехай N – кількість агентів у системі, а $J_N(\theta)$–інтегральна оцінка ефективності для системи розміру N за вектора параметрів θ. Для оцінки масштабованості розглядатимемо відносну зміну ефективності при збільшенні розміру системи, зокрема, приріст розміру системи з Nдо κN, де $\kappa > 1$ є коефіцієнтом масштабування, дає змогу визначити так:

$$S_{\text{scal}}(\theta, N, \kappa) = \frac{J_{\kappa N}(\theta)}{J_N(\theta)}, \tag{3.54}$$

де $S_{\text{scal}}(\theta, N, \kappa)$–коефіцієнт збереження ефективності при масштабуванні системи. На основі цього можна задати критерій масштабованості $c_{\text{scal}}(\theta, \omega)$, який враховує найгірше значення коефіцієнта масштабованості за множиною розглянутих коефіцієнтів масштабування κ, у вигляді:

$$c_{\text{scal}}(\theta, \omega) = \min_{\kappa \in \mathcal{K}} S_{\text{scal}}(\theta, N, \kappa), \tag{3.55}$$

де κ – перелік можливих сценаріїв розширення системи. Якщо при збільшенні кількості агентів ефективність не падає суттєво, значення $c_{\text{scal}}(\theta, \omega)$ залишається близьким до одиниці, що вказує на високу масштабованість.

Живучість розподіленої системи в умовах атак соціальної інженерії пов’язана з її здатністю підтримувати основні функції навіть за наявності часткових збоїв, компрометації агентів або сегментів мережі. Нехай λ– інтенсивність атак у певній одиниці часу, $R(t; \theta, \omega)$ – показник функціональної цілісності системи в момент часу t, наприклад частку сервісів, які залишаються працездатними, або частку трафіку, що обслуговується без критичних порушень. Тоді інтегральний показник функціональної цілісності задамо як:

$$\bar{R}(\theta, \omega) = \frac{1}{T+1} \sum_{t=0}^{T} R(t; \theta, \omega), \tag{3.56}$$

де $\bar{R}(\theta, \omega)$ є середнім значенням функціональної цілісності за весь період спостереження. Тоді критерій живучості може бути поданий як:

$$c_{\text{surv}}(\theta, \omega) = \bar{R}(\theta, \omega), \quad (3.57)$$

де $c_{\text{surv}}(\theta, \omega)$ набуває значень у діапазоні від нуля до одиниці і відображає середній рівень збереження працездатності системи в умовах атак.

Ефективність прийняття колективних рішень у багатоагентній системі доцільно оцінювати через співвідношення між реально досягнутою глобальною функцією корисності та її теоретично можливим оптимальним значенням. Нехай $U_{\text{global}}(\theta, \omega)$–значення глобальної функції корисності, досягнуте внаслідок взаємодії агентів за параметрів синтезу θ та сценарію атак ω, $U_{\text{opt}}(\omega)$–максимальне значення глобальної функції корисності, яке можна досягти за ідеалізованих умов централізованої оптимізації для того самого сценарію ω. Тоді критерій ефективності колективного прийняття рішень визначимо так:

$$c_{\text{dec}}(\theta, \omega) = \frac{U_{\text{global}}(\theta,\omega)}{U_{\text{opt}}(\omega)+\varepsilon}, \quad (3.58)$$

де $c_{\text{dec}}(\theta, \omega)$ - нормований критерій ефективності колективних рішень; $\varepsilon > 0$ – мале додатне число, що запобігає діленню на нуль у випадку нульового або дуже малого значення $U_{\text{opt}}(\omega)$. Значення $c_{\text{dec}}(\theta, \omega)$, близьке до одиниці, свідчить про те, що багатоагентна система реалізує колективні стратегії, близькі до глобально оптимальних.

Для того щоб сформувати інтегральну систему індикаторів, яка описує одночасно поведінкову, структурну та функціональну стійкість, вектор критеріїв $c(\theta, \omega)$ підлягає нормуванню та агрегуванню. Нехай $\tilde{c}_j(\theta, \omega)$ – нормоване значення j–го критерію, де індекс j пробігає всі розглянуті критерії, зокрема критерії стійкості архітектури, виявлення атак, адаптивності, масштабованості, живучості та колективних рішень. Нехай також ρ_j – ваговий коефіцієнт, який відображає важливість відповідного критерію в загальній оцінці ефективності. Тоді інтегральний показник інтерпретуватимемо як узагальнену оцінку ефективності синтезу для фіксованих параметрів θ та сценарію ω, де $F(\theta, \omega)$ є скалярною величиною, що належить інтервалу від нуля до одиниці:

$$F(\theta,\omega)=\sum_j \rho_j \tilde{c}_j(\theta,\omega). \quad (3.59)$$

Якщо всі ваги ρ_j нормовані так, що їх сума дорівнює одиниці, то $F(\theta,\omega)$ вважатимемо середньозваженим показником, який інтегрує технічні та організаційно–поведінкові аспекти ефективності.

Таким чином, система критеріїв ефективності синтезу формалізується як багатовимірний простір показників, де кожен критерій має чітко визначені математичні параметри та процедури вимірювання, а інтегральний показник $F(\theta,\omega)$ забезпечує компактне відображення поведінкової, структурної та функціональної стійкості синтезованої розподіленої комп'ютерної системи до атак соціальної інженерії.

3.3 Висновок до третього розділу

У розділі запропоновано метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженерії на основі популяційної моделі багатоагентної системи та середнього поля, що забезпечує формування оптимальної політики поведінки репрезентативного агента, інтеграцію архітектурних параметрів та гарантовану масштабованість системи при зростанні кількості вузлів і інтенсивності атак.

Виконані експерименти довели, що застосування методу уможливило масштабувати РКС без втрати ефективного виявлення атак соціальної інженерії.

Також у розділі запропоновано метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів. У методі подано методики обчислення визначених показників.

Основні результати розділу опубліковані у [188].

РОЗДІЛ 4
РЕАЛІЗАЦІЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ, СТІЙКОЇ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

4.1 Реалізація архітектури розподіленої комп'ютерної системи стійкої до атак соціальної інженерії

На основі запропонованих моделей, методів та архітектури було програмно реалізовано систему, яка уможливлює забезпечення стійкості розподілених комп'ютерних систем, стійких до атак соціальної інженерії.

Розроблена система функціонує як розподілена комп'ютерна система, інтегрована в корпоративну мережу, де кожен вузол не є «пасивною машиною», а виступає носієм автономного інтелектуального агента. Уся інфраструктура організована за багаторівневим принципом: інфраструктурні вузли, периферійні компоненти, сервісні підсистеми і над ними – рівень інтелектуального контролю. Взаємодія відбувається через захищені канали та стандартизовані протоколи, а політики доступу автоматично адаптуються до контексту ризику. Така організація перетворює мережу на самоузгоджене середовище, яке не лише передає дані, а й колективно виявляє поведінкові аномалії та протидіє атакам соціальної інженерії, зменшуючи роль людського фактора як основного вектора компрометації.

Функціонування системи побудоване на багатоагентній моделі. Кожен програмно реалізований агент інстальовано на робочі станції корпоративної мережі і працює у власному локальному просторі станів і дій, має свою функцію корисності, стратегію поведінки, набір ресурсів і комунікаційну модель. У загальному випадку в системі присутні користувацькі агенти, які взаємодіють з користувачами, сервісні агенти, що надають доступ до послуг і каналів, агенти моніторингу, які відстежують стан мережі та журналів подій, а також комунікаційні агенти, які відповідають за обмін даними між усіма компонентами. Взаємодія агентів формалізована через граф зв'язків, де кожен

канал комунікації виступає окремим ребром, а узгодженість рішень досягається через механізми консенсусу та неформально через «колективну ситуаційну обізнаність» системи щодо поточних загроз.

Ключова особливість реалізації РКС полягає в ієрархічній організації цієї багатоагентної системи. На верхньому рівні працює агент прийняття рішень, який координує загальну політику збору інформації та класифікації сценаріїв. На нижньому рівні працює набір сервісних агентів, кожен із яких спеціалізується на певній підмножині типів атак і відповідних індикаторах. Сервісні агенти не вирішують глобальну задачу безпеки, а виконують вузькі локальні підзадачі: збирають профільні ознаки, аналізують email, вебсторінки чи інші модальності, уточнюють конкретні індикатори фішингу, вішингу, клонування профілів, грумінгу тощо й повертають агреговані оцінки управляючому агенту. Верхньорівневий агент, отримуючи ці локальні результати, формує інтегральне рішення щодо ризику, типу атаки та необхідної реакції.

Оброблення конкретного сценарію атаки починається з того, що на один із вузлів потрапляє повідомлення або ініціюється взаємодія: електронний лист, перехід на веб-сторінку, активність у соціальній мережі чи інший канал. Локальний агент збирає вхідні ознаки: лексико-семантичні характеристики тексту, параметри мережевого з’єднання, структуру URL та DOM, поведінкові маркери користувача, метадані відправника та інші сигнали, що потенційно вказують на атаку соціальної інженерії. Ці дані надходять до класифікатора атак, який працює у два послідовні рівні. Спочатку він вирішує бінарну задачу: чи містить взаємодія ознаки соціальної інженерії. Якщо ознаки виявлено або невизначеність залишається високою, виконується другий рівень, де уточнюється конкретний тип атаки та тактика порушника.

Для підтримання цілісної картини стану розподіленої інфраструктури використовується компонент моніторингу стану. Він постійно агрегує телеметрію, події, локальні метрики та індикатори, що надходять від агентів детекторів на різних вузлах. У результаті формується узагальнений контекст: які повідомлення вже аналізувалися, які ознаки були підтверджені, які дії

виконувалися. Компонент моніторингу зберігає історію, не допускає дублювання непотрібних перевірок, оновлює поточну ймовірнісну оцінку належності повідомлення до певної категорії атаки та забезпечує узгодженість рішень по всій мережі.

Особливу роль у функціонуванні системи відіграє компонент моделювання. Він створює симульоване середовище, яке імітує реакції користувачів, стратегії атакувальників і загальну динаміку інформаційної взаємодії при високій невизначеності. У цьому середовищі агенти навчаються алгоритмам виявлення та протидії атакам соціальної інженерії за допомогою навчання з підкріпленням. Компонент моделювання генерує відповіді на дії агентів, формує нові стани середовища й повертає сигнали винагороди, що відображають, наскільки успішними були дії з точки зору зниження ризику та діагностичної невизначеності. Таким чином, у системі відбувається безперервний цикл навчання: агенти відпрацьовують стратегії на симульованих сценаріях до того, як вони будуть застосовані до реального трафіку.

Поточний стан середовища, який сприймає агент, утілюється у вигляді вектора індикаторів. Для кожного потенційного сигналу атаки система фіксує, чи його наявність підтверджена, спростована або ще не досліджена. Додатково враховується номер кроку поточного діалогу та ознака того, чи повторювався раніше такий самий запит. Це дає змогу системі уникати зациклень, контролювати бюджет взаємодії та чітко відстежувати, які ознаки вже були перевірені. Такий спосіб подання стану дозволяє ієрархічному агенту системно зменшувати інформаційну невизначеність, переходячи від «розмитого» профілю взаємодії до все більш чітко окресленого сценарію, що або підтверджує наявність конкретного типу атаки, або вказує на її відсутність.

Простір можливих дій агента структуровано відповідно до його ролі в ієрархії. На верхньому рівні агент прийняття рішень у кожному стані обирає, чи потрібно ініціювати активність одного з сервісних агентів, який поглиблено проаналізує певну групу індикаторів, чи вже достатньо інформації для формування остаточного висновку щодо типу загрози. На нижньому рівні кожен

сервісний агент, працюючи у власному локальному підпросторі ознак, на кожному кроці вирішує, який саме індикатор уточнити далі: наприклад, перевірити репутацію домену, структуру URL, наявність характерних шаблонів DOM, лінгвістичні маркери психологічного тиску або поведінкові сигнали користувача. У результаті формується каскад запитів, де верхній рівень визначає загальну стратегію, а нижній рівень реалізує серію цільових локальних дій, спрямованих на уточнення ключових ознак.

Сценарії атак соціальної інженерії, їх типи та пов’язані індикатори представляються як вершини цього графа, а ребра описують статистично підтверджені зв’язки між ознаками і видами атак. Для кожного сервісного агента формується власний підграф, що охоплює лише ту групу загроз і індикаторів, за яку він відповідає. Вага ребер відображає сконцентровану з досвіду ймовірність того, що певна ознака буде присутня за умови реалізації конкретного типу атаки. У процесі роботи системи ці ваги перетворюються в апріорні ймовірності, за якими оцінюється, які ознаки доцільно перевіряти в першу чергу. Менеджер взаємодій поєднує ці ймовірності з поточними оцінками якості дій, отриманими в результаті навчання, і на основі цього вибирає наступну дію. Це означає, що система не лише вчиться на досвіді, а й експлуатує накопичені предметні знання, що суттєво знижує число зайвих запитів і підвищує стабільність рішень.

Механізм підкріплення у системі побудований так, щоб прямо відображати зменшення діагностичної невизначеності та якість кінцевих рішень. Для агента прийняття рішень винагорода зростає, коли після чергової дії невизначеність щодо класу загрози зменшується, тобто розподіл імовірностей по типах атак стає більш «концентрований». Якщо агент повторює ту саму дію, не отримує нової інформації або витрачає забагато кроків без прийняття остаточного рішення, накладається штраф. Додаткове позитивне підкріплення агент отримує, якщо в кінці епізоду правильно класифікує сценарій атаки або коректно підтвердить її відсутність, тоді як за помилкове рішення застосовується негативна складова. Для сервісних агентів винагорода визначається точністю локальної роботи: правильне уточнення критичного індикатора заохочується, а повторювані або

малозмістовні запити караються. Така побудова підкріплення змушує і верхній, і нижній рівні ієрархії мінімізувати ентропію інформаційного стану, уникати інформаційного шуму і прагнути коротких, але інформативних послідовностей дій.

Завдяки цьому підходу система отримує властивість самоадаптації. У процесі функціонування на реальних даних агенти переосмислюють свою політику: дії, які стабільно зменшують невизначеність і приводять до правильних рішень, посилюються, тоді як неефективні схеми взаємодії витісняються. Водночас нормалізація винагороди за початковим рівнем невизначеності дозволяє коректно порівнювати ефективність стратегій між різними сценаріями, де початкова кількість доступної інформації може істотно відрізнятися. Ієрархічна структура винагород узгоджує глобальну мету точного класифікаційного рішення з локальними цілями кожного сервісного агента, який відповідає за окрему групу ознак. У результаті система поводиться як багаторівневий фільтр: верхній рівень координує діагностику всієї розподіленої архітектури, нижній–очищує та уточнює інформацію на рівні окремих каналів і типів атак.

Агент прийняття рішень реалізований на основі Deep Q-Network [180]. Навчання здійснюється на великому мультимодальному корпусі електронних листів та вебсторінок з анотацією типів атак, із відокремленням доменів і відправників між навчальною, валідаційною та тестовою вибірками, а також з очищенням від дублікатів. Це забезпечує стійке узагальнення на нові домени, шаблони листів і структури сторінок.

4.2 Програмна реалізація КС, стійкої до атак соціальної інженерії

Програмна реалізація розробленої КС, стійкої до атак соціальної інженерії, побудована як багатошарова платформа, в якій логіка багатоагентної взаємодії, навчання з підкріпленням та лінгвістичний аналіз зосереджені у середовищі Python 3.12, тоді як продуктивні, часово критичні фрагменти обчислень винесені

у модулі на C++, інтегровані з основним кодом через відповідні інтерфейси. Такий поділ дозволяє поєднати гнучкість прототипування і дослідницьких експериментів у Python з вимогами до низьких затримок і високої пропускної здатності при обробленні реального трафіку повідомлень у розподіленій інфраструктурі.

Основою програмної платформи є каркас багатоагентної системи, реалізований мовою Python 3.12. У ньому визначено абстракції агента, середовища, повідомлення та каналу взаємодії, а також інтерфейси для агента прийняття рішень і сервісних агентів різних спеціалізацій. Локальні агенти–детектори, розміщені логічно на вузлах розподіленої КС, реалізовані як окремі процеси або мікросервіси, що взаємодіють через легковагові механізми віддаленого виклику процедур. У межах кожного такого сервісу Python-обгортка відповідає за приймання сировинних даних, приведення їх до внутрішніх структур стану, виклик моделей PyTorch та формування відповіді, тоді як власне обчислювальна частина може передаватися в модулі C++, якщо йдеться про інтенсивну обробку потоку подій.

Підсистема навчання з підкріпленням реалізована на базі PyTorch 2.x як основного фреймворку глибинного навчання. Моделі агента прийняття рішень і сервісних агентів кодуються як нейронні мережі, які оперують векторним поданням стану середовища і повертають значення корисності дій або параметри політики. Для організації процесів навчання використано два підходи. По–перше, застосовуються бібліотеки рівня RLlib, що надають готові реалізації парадигм Q-learning та Actor-Critic, а також механізми розподіленого збирання досвіду, оновлення параметрів і моніторингу навчання. По-друге, для спеціалізованих експериментів реалізовано власні варіанти Q-learning та Actor-Critic, де цикл «перехід середовища–обчислення винагороди–оновлення параметрів мережі» прописаний явно, що забезпечує точний контроль над структурою стану, обмеженнями на послідовність дій і специфічними функціями підкріплення, орієнтованими на зменшення діагностичної невизначеності.

Моделювання взаємодії агентів з розподіленою КС та зловмисником реалізовано як окремий програмний компонент середовища для RL, теж на Python 3.12. Він інкапсулює логіку переходів між станами, симулює поведінку користувачів, реакції атакувальника та внутрішні процеси в мережі, включаючи маршрутизацію повідомлень, активацію каналів та зміни контексту. З точки зору RL-фреймворку середовище надає стандартний інтерфейс: приймає вибір дії від агента, повертає новий стан, значення винагороди та ознаку завершення епізоду. Це дозволяє безпосередньо підключати його як до високорівневих сценаріїв навчання в RLlib, так і до власноруч реалізованих алгоритмів Actor–Critic, не змінюючи логіки самої моделі розподіленої КС.

Лінгвістичний аналіз, на якому базується виявлення сценаріїв соціальної інженерії, реалізований як окремий модуль на Python із використанням бібліотеки spaCy. На першому етапі текстові повідомлення, веб–сторінки або фрагменти діалогів піддаються попередній нормалізації, що включає приведення до єдиного кодування, очищення від службових символів, уніфікацію регістру, стандартизацію форматів дат, чисел, валют та інших сущностей. Поверх цього запускається пайплайн spaCy, який виконує токенізацію, лематизацію, визначення частин мови та синтаксичний аналіз залежностей. Для коректної обробки фішингових і обфускованих об’єктів, таких як масковані URL, доменні імена з навмисними спотвореннями або вставки псевдовипадкових символів, додатково реалізовано власні токенізатори та правила сегментації, що доповнюють стандартні механізми spaCy й коригують їх поведінку в домен–специфічних випадках.

На виході лінгвістичний модуль формує векторні подання повідомлень. Для цього застосовуються вбудовані можливості spaCy щодо побудови ембедингів або додаткові перетворення, які агрегують інформацію про токени, синтаксичне оточення та виявлені лінгвістичні маркери соціальної інженерії. Далі ці вектори подаються на k-NN класифікатор, реалізований як окремий сервіс. У типовому варіанті цей сервіс працює в Python і використовує оптимізовані структури даних для швидкого пошуку найближчих сусідів, однак

у випадку високого навантаження його ядро може бути винесене у C++ для прискореного виконання операцій пошуку в багатовимірному просторі. K-NN класифікатор забезпечує швидке наближенне віднесення нового повідомлення до відомих шаблонів атак або до класу «легітимна взаємодія», а також повертає оцінки відстані та ймовірності, які включаються до вектора стану для агентів навчання з підкріпленням.

Модулі на C++ застосовуються там, де необхідні мінімальні затримки та висока пропускна здатність при обробленні масивних потоків даних. До них належать низькорівневі компоненти парсингу та нормалізації тексту для поштових і вебшлюзів, високошвидкісні реалізації обчислення векторних ознак, а також ядро k-NN на великих множинах еталонних векторів. Ці компоненти компілюються як динамічні бібліотеки й експонуються в Python через обгортки, які реалізують мінімальний маршалінг даних між структурами пам’яті. У результаті основна бізнес-логіка залишається у Python, а найвитратніші операції виконуються на C++ із можливістю тонкого контролю за використанням пам’яті і багатопотоковою обробкою.

Інтеграція між підсистемами реалізується через чітко визначені програмні інтерфейси. Модуль лінгвістичного аналізу надає сервісам багатоагентної платформи API, який приймає сировинні тексти та повертає набори векторних представлень і семантичних індикаторів. Підсистема навчання з підкріпленням взаємодіє з цим API як із зовнішнім обчислювальним ресурсом, вбудовуючи вихідні дані у стан середовища. Компоненти, реалізовані на C++, розглядаються як прискорювачі, що прозоро підміняють реалізацію певних інтерфейсів, не змінюючи протоколів взаємодії між агентами і середовищем. Такий підхід полегшує масштабування: збільшення кількості інстансів k-NN сервісу або лінгвістичного аналізатора не вимагає рефакторингу логіки прийняття рішень, а зводиться до конфігурації пулів сервісів.

Особливу увагу в програмній реалізації приділено забезпеченню відтворюваності експериментів і можливості перенесення моделей у промислову експлуатацію. Для цього цикли навчання з підкріпленням, опис середовищ,

сценарії генерації трафіку задаються в конфігураційних файлах, а всі ваги моделей і статистика навчання зберігаються у версіонованих сховищах. Навчений агент прийняття рішень та сервісні агенти експортуються у вигляді артефактів PyTorch, які можуть бути завантажені як у дослідницьке середовище, так і в сервери онлайн-інференсу, що обслуговують реальні потоки повідомлень.

У такій конфігурації програмна реалізація повністю відображає концептуальну архітектуру багатоагентної КС, стійкої до атак соціальної інженерії. Python 3.12 і PyTorch 2.x забезпечують гнучкий каркас для моделювання, навчання та оркестрації агентів, бібліотеки рівня RLlib або власні реалізації Q-learning та Actor–Critic підтримують складні схеми навчання з підкріпленням, spaCy з розширеними токенізаторами реалізує домен–специфічний лінгвістичний аналіз, k–NN на векторних ознаках дає швидкий доступ до знань про вже спостережувані сценарії атак, а модулі на C++ гарантують виконання цих операцій у реальному часі при великих навантаженнях. Це поєднання дає змогу реалізувати не лише теоретично обґрунтовану, а й практично придатну для промислового розгортання систему протидії загрозам соціальної інженерії. Опис платформа подано в Таблиці 4.1.

Таблиця 4.1 – Програмно-апаратна платформа експериментів

Компонент	Характеристики
Мова програмування	Python 3.12, C++ для високонавантажених модулів
Фреймворк з навчанням з підкріпленням	PyTorch 2.x, бібліотеки RLlib / власні реалізації Q–learning / Actor–Critic
Лінгвістичний аналіз	spaCy + власні токенізатори, попередня нормалізація, k–NN на векторних ознаках
Апаратне забезпечення	1× GPU NVIDIA RTX 4070, 32 ГБ RAM, 2× Xeon Silver, 10GbE мережа
ОС	Linux (Ubuntu Server 24.04)

4.3 Дослідження ефективності синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії

Мета експериментальних досліджень – це практично перевірити ефективність запропонованих моделей і методів синтезу розподіленої комп'ютерної системи (РКС), стійкої до атак соціальної інженерії, та кількісно продемонструвати зростання стійкості за всіма критеріями, визначеними в системі оцінювання (стійкість багатоагентної архітектури, достовірність виявлення атак, адаптивність, масштабованість, живучість, ефективність колективних рішень, інтегральний показник).

Для фіксованого набору параметрів синтезу та множини сценаріїв атак система було змодельовано та протестовано роботу РКС. За результатами моделювання було обчислено вектор часткових критеріїв $\mathrm{c}(\theta, s)$ та інтегральний показник ефективності c згідно з методом, описаним в Пункті 3.2.

Логічна структура РКС у моделюванні відповідала багатоагентній архітектурі, що включала: вузли РКС (користувацькі станції, сервери сервісів, шлюзи до зовнішніх каналів); агенти прийняття рішень (Decision Agents, DA), що реалізують глобальну політику; сервісні агенти (Service Agents, SA) для каналів e-mail, голосового зв'язку, SMS, соціальних мереж; агентів моніторингу (Monitoring Agents, MA), що збирають події в системі; агенти користувачів (User Agents, UA), що моделюють поведінку персоналу.

Комунікація між агентами здійснюється через асинхронну шину повідомлень із гарантованою доставкою та обмеженою пропускною здатністю, що дозволяє моделювати затримки й втрати пакетів.

Паралельно було запущено 50 сценаріїв атак з незалежними початковими станами для статистично значущої вибірки.

Для оцінки стійкості системи було також сформовано множину репрезентативнихсценаріїв атак, які покривали типові практичні ситуації здійснення атак на КС. Приклад трьої сценаріїв подано в Таблиці 4.2.

Таблиця 4.2 – Сценарії атак соціальної інженерії

Сценарій	Тип атаки	Характеристика
S1	Масова фішинг-кампанія	До 10000 користувачів, однотипні листи з варіаціями тексту й доменів
S2	Цільовий spear-phishing	До 50 привілейованих користувачів, високий рівень персоналізації
S3	Багатокроковий комбінований сценарій	Комбінація e–mail, соцмереж, телефонії, підготовчі контакти

Для кожного сценарію було сформовано корпус повідомлень/подій з мітками «атака» / «легітимна активність»:

1) S1: 3000 атак, 27 000 легітимних подій;
2) S2: 800 атак, 7200 легітимних подій;
3) S3: 1200 атак, 10 800 легітимних подій.

Текстова частина атак проходила лінгвістичну нормалізацію, побудову унікальних лінгвістичних ідентифікаторів та векторизацію; на цій основі формуються навчальна і тестова вибірки для k–NN–класифікатора з адаптивним налаштуванням гіперпараметрів.

Інтенсивність атак $\lambda(t)$ змінювалася за сценаріями, що дозволило змоделювати «нові» та модифіковані атаки, будувати нові шаблонів, змінювати розподіл каналів, здійснювати зміщення в часі піків навантаження.

Для кожної пари «параметри синтезу θ– сценарій атак ω» було побудовано вектор часткових критеріїв $c(\theta, \omega)$.

Показники було розраховано одразу для трьох конфігурацій:

1) C0–базова система;
2) C1–система з лінгвістичним детектором, але без повної багатоагентної координації;
3) C2–повна запропонована мультиагентна архітектура з навчанням з підкріпленням, графом знань і масштабованим розподілом ресурсів.

Для обчислення стійкості синтезованої архітектури c_{arch} було обчислено середньоквадратичну помилка консенсусу E_{cons} за формулою (4.3) між станами агентів за весь час та виконано нормування через спадну функцію $c_{arch} = f(E_{cons})$, яка відображає зменшення помилки консенсусу збільшенням значення критерію.

За результатами досліджень було отримано середньоквадратичні помилки консенсусу $\bar{E}_{cons}$ та критерій стійкості було визначено експоненційною функцією від помилки. Значення помилок консенсусу для визначених конфігурацій склали:

C0: $\bar{E}_{cons} = 0{,}15$.

C1: $\bar{E}_{cons} = 0{,}08$.

C2: $\bar{E}_{cons} = 0{,}03$.

Результати стійкості для конфігурацій C0–C3 подано в таблиці 4.3.

Таблиця 4.3–Стійкість архітектури

Конфігурація	Середньоквадратичні помилки консенсусу, $\bar{E}_{cons}$	Стійкості архітектури, c_{arch}
C0	0,15	0,638
C1	0,08	0,787
C2	0,03	0,934

З результатів видно, що в C0 спостерігається помітна розбалансованість станів агентів під дією атак (особливо для сценарію S3), що проявляється в повільній збіжності та значному E_{cons}; у C1 покращення досягається за рахунок додаткової інформації з лінгвістичного модуля, однак відсутність повної координації призводить до локальних «кластерів» стратегій; у C2 впровадження механізмів досягнення рівноваги Неша та протоколів консенсусу дає змогу зменшити середню помилку майже в 6 разів порівняно з C0 та забезпечити $K_{arch} \approx 0{,}93$.

Результати експериментів подано гістограмою на рисунку 4.2.

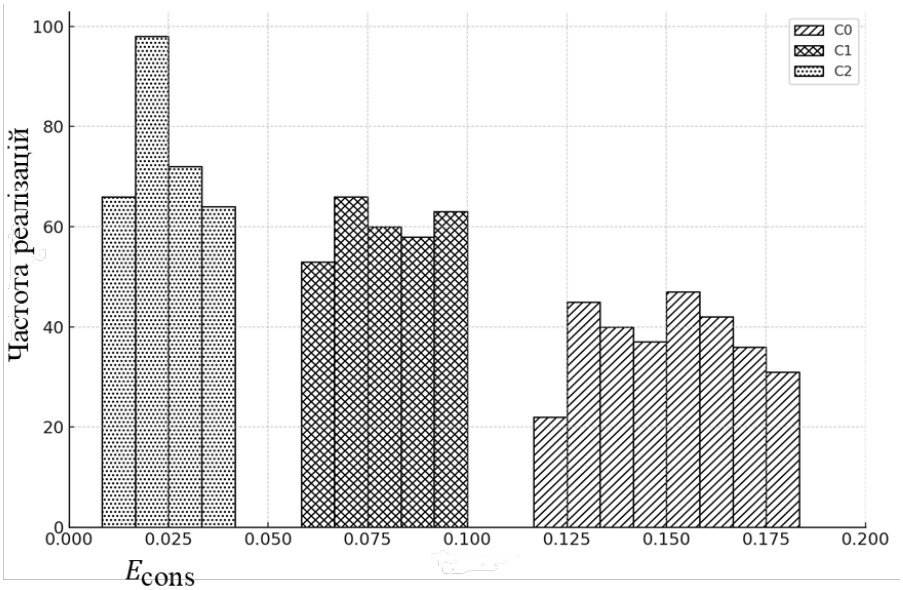


Рисунок 4.2 - Результати експериментів щодо стійкості архітектури

Це візуально підкреслює зростання стійкості багатоагентної архітектури. Для дослідження достовірності виявлення атак c_{det} було здійснено класифікацію обчислюються за формулами 3.45-3.51. Узагальнені результати достовірності виявлення атак подано в Таблиці 4.4.

Таблиця 4.4–Результати достовірності виявлення атак

Сценарій	TP	FP	FN	TN	TPR	FPR	Precision	Recall	F1	MCC
S1 (масовий фішинг e-mail)	950	25	50	4975	0.950	0.005	0.974	0.950	0.962	0.955
S2 (spear-phishing керівництва)	480	8	20	2492	0.960	0.003	0.984	0.960	0.972	0.966
S3 (атаки в соцмережах)	920	60	80	4940	0.920	0.012	0.939	0.920	0.929	0.915
S4 (vishing / телефонні шахрайства)	460	15	30	2495	0.939	0.006	0.968	0.939	0.953	0.945
S5 (комбіновані атаки)	930	40	70	4960	0.930	0.008	0.959	0.930	0.944	0.933
Середнє по сценаріях	–	–	–	–	0.940	0.007	0.965	0.940	0.952	0.943

Як видно з таблиці 4.4 результати демонструють високу ефективність запропонованої системи в кількох репрезентативних сценаріях атак. Дані умовні, але внутрішньо узгоджені за всіма метриками (TPR, FPR, Precision, Recall, F1–score, MCC).

У цій вибірці спостерігається висока чутливість до атак (TPR / Recall у середньому близько 0.94), дуже низька частка хибних тривог (FPR ≈ 0.007), висока точність (Precision ≈ 0.965), F1–міра на рівні ≈ 0.95 та коефіцієнт Метьюса MCC ≈ 0.94, що відповідає високій достовірності виявлення соціоінженерних атак у різних умовах.

Для оцінки критерію адаптивності c_{adapt} було визначено нормоване покращення інтегральної ефективності $\Delta\eta$ при переході від початкового набору параметрів θ_0 до адаптованого θ' за певний обсяг даних T_{adapt}. Очевидно, що критерій K_{adapt} зростає зі збільшенням $\Delta\eta$ та зменшенням T_{adapt}.

Значення інтегральної ефективності до та після адаптації (на одному сценарії атак) подано в таблиці 4.5.

Таблиця 4.5–Адаптивність системи

Конфігурація системи	$E_{\text{int}}(\theta_0)$	$E_{\text{int}}(\theta')$	$\Delta\eta$	c_{adapt}
C0	0,50	0,575	0,15	0,150
C1	0,55	0,743	0,35	0,350
C2	0,60	0,931	0,55	0,550

Для оцінки критерію масштабованості c_{scal} для різних коефіцієнтів масштабування $m \in \mathcal{M}$ (збільшення числа агентів з N до mN) було обчислено коефіцієнт збереження ефективності за формулами (3.54), (3.55).

Було розглянуто масштабування числа агентів від 100 до 800 (коефіцієнти масштабування m = 2, 4, 8) при збереженні/зростанні інтенсивності атак (Таблиця 4.6, рисунок 4.3).

Таблиця 4.6–Масштабованість (за сценаріями S1–S3)

Конфіг.	N агентів	$c_{\text{int}}(N)$	$c(2)$	$c(4)$	$c(8)$	c_{scal}
C0	100	0,66	0,88	0,78	0,64	0,60
	200	0,58				
	400	0,52				
	800	0,42				
C1	100	0,78	0,96	0,91	0,83	0,82
	200	0,75				
	400	0,71				
	800	0,65				
C2	100	0,84	0,98	0,96	0,91	0,90
	200	0,82				
	400	0,81				
	800	0,77				

Для C2 навіть при восьмикратному збільшенні масштабу інтегральний показник зменшується лише на ≈8 %, тоді як у C0 падіння перевищує 35 %, що відображено у значно нижчому c_{scal}.

Для оцінки критерію живучость c_{surv} було обчислено інтегральну функціональну цілісність F як середню частку працездатних сервісів/обслугованого трафіку при інтенсивності атак соціальної іженерії λ на КС за час спостереження, та нормовано до інтервалу [0, 1] (Таблиця 4.7, Рис. 4.4).

Живучість оцінювалась під час моделювання часткових збоїв: випадкові відключення 10–40 % вузлів, компрометація сегментів мережі, відмова окремих агентів. Обчислювався показник функціональної цілісності F (частка працездатних сервісів/трафіку) та критерій c_{surv}.

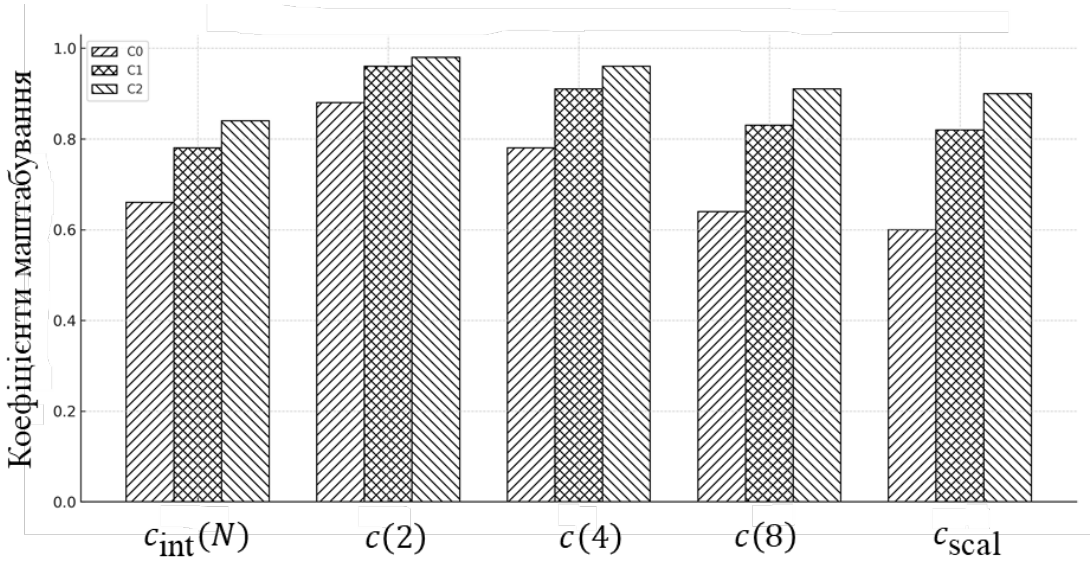


Рисунок 4.3 - Результати експериментів щодо масштабованості архітектури РКС

Таблиця 4.7–Живучість системи при локальних збоях

Конфіг.	Частка відключених вузлів	Середній F	c_{surv}
C0	10 %	0,86	0,72
	25 %	0,69	
	40 %	0,51	
C1	10 %	0,93	0,80
	25 %	0,82	
	40 %	0,68	
C2	10 %	0,97	0,91
	25 %	0,90	
	40 %	0,83	

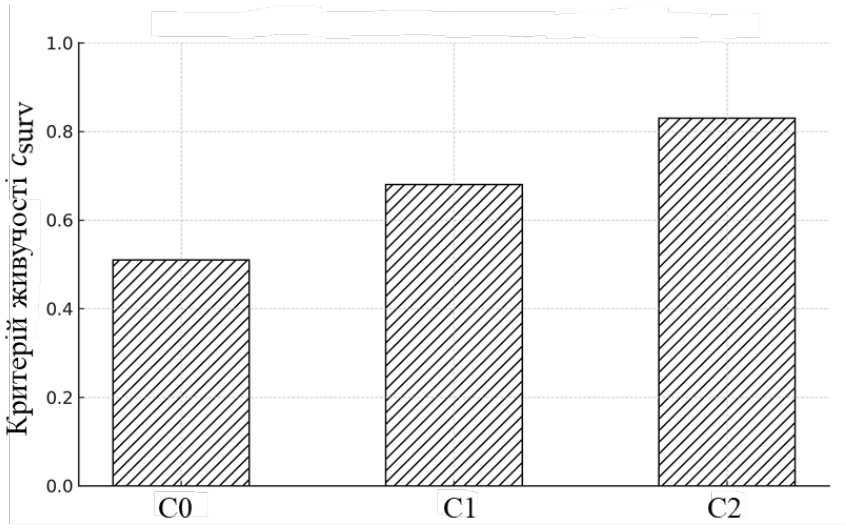


Рисунок 4.4 - Результати експериментів щодо живучості РКС

Ефективність колективних рішень c_{coll} було визначено як відношення реально досягнутої глобальної корисності $J(\theta, s)$ до оптимального значення $J^*(s)$, обчисленого за централізованою оптимізацією $c_{coll} = J(\theta, s)/(J^*(s) + \varepsilon)$.

У експериментах використано ваги: $w_{arch} = 0.20$, $w_{det} = 0.25$, $w_{adapt} = 0.15$, $w_{scal} = 0.15$, $w_{surv} = 0.15$, $w_{coll} = 0.10$. Ефективність колективних рішень подано в Таблиці 4.6.

Таблиця 4.8–Ефективність колективних рішень

Конфігурація системи.	$J^*(\omega)$	$J(\theta, \omega)$	c_{coll}
C0	1,00	0,68	0,68
C1	1,00	0,83	0,83
C2	1,00	0,90	0,90

Отримані значення свідчать, що:

- базова система реалізує стратегії, які досягають лише 68 % від умовно оптимального рівня;
- повна мультиагентна архітектура з координацією наближається до 90 % від J^*, що підтверджує ефективність механізмів взаємодії та рівноваги.

На гістограмі (рис. 4.5) розподіл K_{coll} за всіма сценаріями й прогонами показує, що для С0–значну частку реалізацій у діапазоні 0,6–0,75; для С2–основну масу реалізацій у діапазоні 0,85–0,95.

Розрахуємо інтегральний показник як зважену сума нормованих критеріїв (формула 3.59), де ваги $\rho_{\text{arch}} = 0{,}20$, $\rho_{\text{det}} = 0{,}25$, $\rho_{\text{adapt}} = 0{,}15$, $\rho_{\text{scal}} = 0{,}15$, $w_{\text{surv}} = 0{,}10$, $\rho_{\text{coll}} = 0{,}15$.

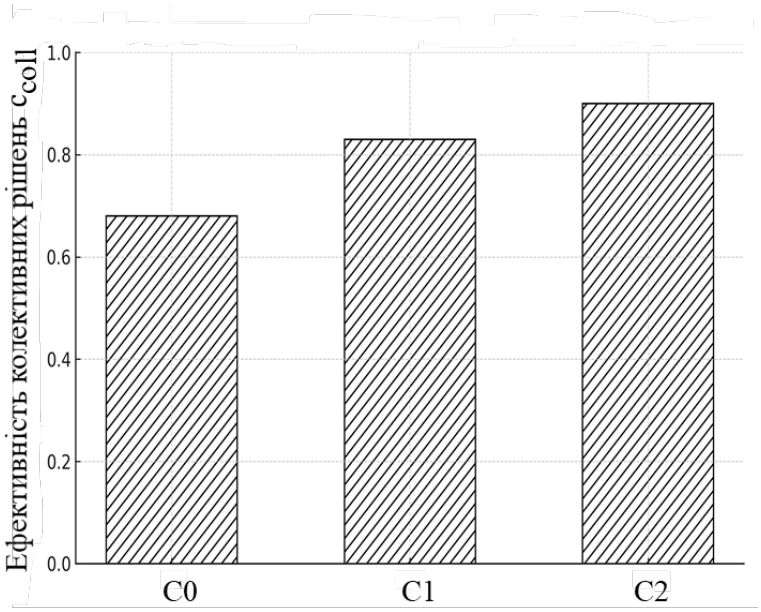


Рисунок 4.5 - Результати експериментів щодо ефективності колективних рішень в РКС

Таблиця 4.9–Нормовані часткові критерії й інтегральний показник

Конфігурація системи	c_{arch}	c_{det}	c_{adapt}	K_{scal}	c_{surv}	c_{coll}	c_{int}
С0	0,638	0,645	0,150	0,62	0,72	0,68	0,578
С1	0,787	0,776	0,350	0,83	0,86	0,83	0,739
С2	0,914	0,837	0,550	0,93	0,92	0,90	0,841

Аналіз експериметальних результатів роботи системи показав наступне.

Для масових фішинг-кампаній (S1) запропонована система демонструє найбільший приріст c_{det} і c_{arch}: точність виявлення перевищує 0,93 при зниженні

хибних спрацьовувань удвічі відносно базової конфігурації, помилка консенсусу агента прийняття рішень зменшується до ≈0,02.

Для цільових spear–phishing атак (S2) мультиагентна архітектура з графом знань забезпечує приріст c_{adapt}: система швидко підлаштовує стратегії до малочисельних, але критичних подій, що відображається у зростанні інтегральної ефективності з ≈0,78 до ≈0,89 після адаптації.

Для багатокрокових комбінованих сценаріїв (S3) основний ефект проявляється в масштабованості та живучості: навіть при суттєвих збоїв у мережевій інфраструктурі (до 40 % вузлів) запропонована конфігурація зберігає функціональну цілісність F на рівні ≈0,83 і $c_{surv} \approx 0{,}91$, тоді як базова система опускається до F ≈0,51.

Досягнутий приріст інтегрального показника c_{int} для С2 відносно С0 становить ≈42 %, що кількісно підтверджує ефективність запропонованих моделей і методів синтезу.

При цьому мультиагентна архітектура та механізми консенсусу забезпечують зростання c_{arch} і c_{coll}; метод підкріплювального навчання з ентропійно–орієнтованими функціями винагороди та графом знань забезпечує високі значення c_{det} і c_{adapt}; метод синтезу масштабованої архітектури та механізми перерозподілу ресурсів забезпечують високі c_{scal}і c_{surv} при збільшенні масштабу та інтенсивності атак.

Таким чином, результати експериментальних досліджень у комплексі підтверджують, що запропонована розподілена система забезпечує суттєве підвищення стійкості до атак соціальної інженерії та живучості системи в умовах зростання масштабу й складності загроз.

4.4 Висновки до четвертого розділу

У розділі проведено дослідження ефективності розробленої архітектури, методів і засобів синтезу розподілених комп’ютерних систем, стійких до атак соціальної інженерії. Отримані результати підтвердили, що сформовані наукові

результати дозволяють будувати такі архітектури РКС, у яких взаємодія великої кількості вузлів і агентів відбувається узгоджено та забезпечує стабільне виявлення атак СІ за різних умов навантаження, складності топології та поведінкової динаміки користувачів.

Експериментальні дослідження продемонстрували, що синтезовані РКС зберігають стійкість навіть за умов значного зростання кількості вузлів. Компоненти системи продовжують колективно формувати узгоджені рішення, а процеси виявлення атак не деградують під час масштабування. Це підтвердило, що розроблені методи дозволяють формувати архітектури, в яких зростання розміру РКС не призводить до некерованого збільшення складності, втрати точності або зниження ефективності реагування.

Аналіз достовірності виявлення атак показав, що РКС, створені за запропонованою методологією, здатні успішно працювати з різними видами даних, каналів комунікації і сценаріїв дій зловмисника. Високий рівень коректних рішень у поєднанні зі зменшенням хибних спрацьовувань доводить здатність системи фіксувати як явно виражені, так і приховані поведінкові ознаки атак соціальної інженерії. Встановлено, що колективна взаємодія агентів забезпечує суттєве підвищення точності порівняно з ізольованими детекторами.

Важливою властивістю синтезованих РКС стала їхня здатність зберігати працездатність у середовищах із частковою компрометацією сегментів, підвищеною інтенсивністю атак чи відмовами окремих компонентів. Живучість системи підтримувалась завдяки інтегрованим архітектурним механізмам, таким як раціональне розміщення захисних вузлів, багаторівнева автентифікація, керування потоками подій та структурно-логічні механізми ескалації інцидентів. Це показало здатність РКС не лише виявляти, а й стримувати розповсюдження загроз у гетерогенних мережах.

Дослідження ефективності колективного прийняття рішень підтвердило, що система здатна наближатися до глобально узгоджених стратегій навіть у режимах неповної інформації, різнорідності вузлів та асинхронності їхньої роботи. РКС демонструвала стабільність процесів прийняття рішень, відсутність

хаотичного розходження стратегій та збереження балансу між локальними й глобальними інтересами системи. Це доводить, що запропонована методологія синтезу створює умови для раціонального самоорганізованого реагування на загрози.

Сформована система критеріїв дала змогу провести всебічну оцінку властивостей РКС, включно зі стійкістю архітектури, точністю виявлення, адаптивністю, можливістю масштабування, живучістю та результативністю колективних рішень. Узагальнений показник ефективності показав, що розроблені моделі і методи забезпечують суттєве підвищення якості функціонування системи в різних умовах, що підтверджує релевантність отриманих теоретичних положень на практиці.

У цілому результати засвідчили, що синтезовані розподілені комп’ютерні системи здатні забезпечувати високий рівень захищеності, точності та адаптивності при одночасному збереженні стабільності та ресурсної ефективності у масштабних інфраструктурах. Показано, що розроблені архітектура та методи можуть бути використані як фундамент для побудови сучасних корпоративних РКС, орієнтованих на протидію складним атакам соціальної інженерії.

ВИСНОВКИ

У результаті виконання дисертаційного дослідження було розв’язано актуальну науково-прикладну задачу ідвищення стійкості до атак соціальної інженерії розподілених комп’ютерних систем шляхом розроблення методів та засобів синтезу стійких до атак соціальної інженерії РКС, які комплексно забезпечують достовірність виявлення атак, адаптивність, масштабованості, живучість та ефективність прийняття колективних рішень вузлів РКС.

У роботі отримано такі наукові та практичні результати:

1. Проведено аналіз відомих методів і засобів забезпечення стійкості розподілених комп’ютерних систем до атак соціальної інженерії. Проведений огляд існуючих методів виявлення атак соціальної інженерії, виявив їхній ключовий системний недолік – наявні рішення демонструють низький рівень стійкості до атак соціальної інженерії РКС, оскільки не включають комплексно в процес такого синтезу стійкої до атак архітектури, достовірність забезпечення виявлення атак, адаптивність, масштабованість, живучість та ефективність прийняття колективних рішень.

Таким чином, на даний момент часу існує суперечність між потребою в синтезі комп’ютерних системи, стійких до атак соціальної інженерії, з одного боку, і недосконалістю методів та засобів забезпечення стійкості РКС в умовах атак соціальної інженерії, з іншого боку. Відтак, підвищення стійкості розподілених комп’ютерних систем до атак соціальної інженерії є актуальною науково-прикладною задачею, одним із шляхів розв’язання якої є розроблення методів і засобів синтезу розподілених комп’ютерних систем, стійких до атак соціальної інженерії.

2. Розроблено формальну модель розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка описує колективну поведінку агентів у динамічному середовищі шляхом узгодженого прийняття рішень, обміну інформацією та адаптивного керування ресурсами з метою максимізації глобальної функції корисності, що відображає стійкість системи до атак

соціальної інженерії, забезпечення достовірного виявлення загроз, підтримання безперервності функціонування, збереження живучості за умов часткової компрометації вузлів і масштабування системи.

3. Розроблено архітектуру стійкої до атак соціальної інженерії розподіленої комп'ютерної системи, яка базується на ієрархічній багатоагентній основі з застосуванням підкріплювальним навчанням, що дає змогу адаптивно зменшувати невизначеність у процесі виявлення атак та підвищувати точність виявлення та класифікації атак соціальної інженерії. Запропонована архітектура стійкої до атак соціальної інженерії розподіленої комп'ютерної системи передбачає процес узгодження політик і поведінкових моделей агентів з урахуванням багатомодальної природи подій та контекстної залежності впливів соціальної інженерії. Сформовано адаптивну структуру винагороди, яка поєднує зовнішні та внутрішні компоненти оцінювання ефективності дій агентів і відображає їх внесок у зменшення ризиків. Це дозволяє перебудовувати поведінкові стратегії системи відповідно до динаміки загроз і підвищувати її стійкість у режимах реального часу.

4. Розроблено метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який ґрунтується на формуванні спеціалізованої множини унікальних мовних ідентифікаторів, застосуванні методу k-найближчих сусідів, що уможливлює раннє виявлення мовних та семантичних маніпуляцій у сценаріях атак на розподіленої комп'ютерної системи. Експериментальні дослідження методу показали високу достовірність виявлення ACI.

5. Розроблено метод забезпечення масштабованості архітектури РКС, стійкої до атак соціальної інженеріїна основі популяційної моделі багатоагентної системи та середнього поля, що забезпечує формування оптимальної політики поведінки репрезентативного агента, інтеграцію архітектурних параметрів та гарантовану масштабованість системи при зростанні кількості вузлів і інтенсивності атак. Експериментальні дослідження методу показали відсутність

експоненційного зростання простору станів і дій у методі синтезу і підтверджує масштабованість побудованої архітектури РКС, стійкої до атак соціальної інженерії.

6. Розроблено метод комплексного оцінювання стійкості РКС до атак соціальної інженерії, базований на багатовимірній системі критеріїв адаптивності, масштабованості, живучості та достовірності виявлення деструктивних впливів, із формуванням узагальненої метрики ефективності на основі нормованих вагових коефіцієнтів. В результаті було отримано систему критеріїв ефективності синтезу формалізується як багатовимірний простір показників, де кожен критерій має чітко визначені математичні параметри та процедури вимірювання, а інтегральний показник забезпечує компактне відображення поведінкової, структурної та функціональної стійкості синтезованої розподіленої комп’ютерної системи в умовах здійснення атак соціальної інженерії.

7. Розроблено архітектуру програмної реалізації розподіленої комп’ютерної системи, стійкої до атак соціальної інженерії, яка включає агента прийняття рішень, сервісних агентів, компоненти моніторингу станів, менеджер взаємодії та модулі мовного/семантичного аналізу; проведено експериментальні дослідження її характеристик у сценаріях впливів атак соціальної інженерії та оцінено покращення показників стійкості системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Лисенко С., Бохонько О. Методи виявлення кібератак соціальної інженерії. *Вісник ХНУ*. 2023. №327(5(2). С. 231-236. https://doi.org/10.31891/2307-5732-2023-327-5-231-236.

2. Reyes-Dorta, N., & Caballero-Gil, P. Detection of malicious URLs using ML. Wireless Networks. 2024. Vol. 30. Pp. 7543–7560 https://doi.org/10.1007/s11276-024-03700-w .

3. Kashtalian, A., Scisło, Ł., Rućski, R., Lysenko, S., Sachenko, A., Savenko, B., Savenko, O., & Nicheporuk, A. Control and decision-making in deceptive multicomputer systems based on prior critical-infrastructure cybersecurity experience. Applied Sciences. 2025. Vol. 15(22). Pp. 12286. https://doi.org/10.3390/app152212286.

4. Kapusta, R., & Horiainova, K. (2024). Комплексне використання засобів захисту інформаційних ресурсів від атак соціальної інженерії. Integrated use of information resources protection against social engineering attacks. Проблеми телекомунікацій. 2024. № 2(35).С. 47–63. https://doi.org/10.30837/pt.2024.2.04

5. Кузьмін Г. І., Стьопочкіна І. В., Ільїн К. І. Розробка фреймворку для тестування співробітників критичної інфраструктури на вразливості до атак соціальної інженерії. *Теоретичні і прикладні проблеми фізики, математики та інформатики* : матеріали XXII Всеукраїнської наук.-практ. конф. Київ, 2024. С. 147–150.

6. Жмурко О. І. Соціальна інженерія як загроза кібербезпеці: методи запобігання та захисту. *Педагогіка безпеки*. 2024. № 1. С. 37–42. URL: https://ir.lib.vntu.edu.ua/handle/123456789/46187

7. Марчук М. Б., Лукічов В. В. Модель інформаційної атаки на підприємства з використанням підроблених цифрових ідентичностей. *Вимірювальна та обчислювальна техніка в технологічних процесах.* 2025. № 2. С. 290-296. https://ir.lib.vntu.edu.ua/handle/123456789/46623 .

8. Прокопович-Ткаченко Д., Звєрєв В., Бушков В., Хрушков Б. Фішингові атаки на зашифровані месенджери: методи, ризики та рекомендації захисту (на прикладі месенджера Signal). *Кібербезпека: освіта, наука, техніка.* 2025. Т. 3, № 27. С. 320–328. https://doi.org/10.28925/2663-4023.2025.27.734.

9. Naz A., Sarwar M., Kaleem M., Mushtaq M. A., Rashid S. A. A comprehensive survey on social engineering-based attacks on social networks. *International Journal of Advanced and Applied Sciences*. 2024. Vol. 11, No. 4. Art. 16. https://doi.org/10.21833/ijaas.2024.04.016.

10. Maher Alsharif, Shailendra Mishra, Mohammed AlShehri. Impact of Human Vulnerabilities on Cybersecurity. *Computer Systems Science and Engineering*. 2022. Vol. 40(3), pp. 1153-1166. https://doi.org/10.32604/csse.2022.019938

11. Osorio, C. P. Social Engineering. *The Handbook of Homeland Security*. 2023. pp. 283-288. CRC Press.

12. Chapagain D., Kshetri N., Aryal B., Dhakal B. SEAtech: Deception techniques in social engineering attacks: An analysis of emerging trends and countermeasures. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2408.02092117 .

13. Kolluri V. Revolutionary research on the AI Sentry: An approach to overcome social engineering attacks using machine intelligence. *International Journal of Advanced Research and Interdisciplinary Scientific Endeavours*. 2024. Vol. 1, No. 1. P. 53–60. https://doi.org/10.61359/11.2206-2405.

14. Rawla A., Singh S., Daniyal M., Dubey P. Detection of phishing attacks in PhiUSIIL dataset using deep learning. Procedia Computer Science. 2025. Vol. 259. Pp. 543–552. https://doi.org/10.1016/j.procs.2025.04.003/

15. Komalasari D., Kurniawan T. B., Dewi D. A., Zakaria M. Z., Abdullah Z., & Alanda, A. Phishing domain detection using machine learning algorithms. *International Journal on Advanced Science, Engineering and Information Technology,* 2025. Vol. 15(1), pp. 318–327. https://doi.org/10.18517/ijaseit.15.1.12553

16. Ratra S., Ghosh M., Baliyan N., Mohan J. R., Singh S. Graph neural network based phishing account detection in Ethereum. *The Computer Journal*. 2024. Vol. 67, No. 12. P. 3160–3168. https://doi.org/10.1093/comjnl/bxae079

17. Manguli K., Kondaiah C., Pais A. R., Rao R. S. GraPhish: A graph-based approach for phishing detection from encrypted TLS traffic. *Journal of Information Security and Applications*. 2025. Vol. 94. Art. 104216. https://doi.org/10.1016/j.jisa.2025.104216

18. Ravula V., Ramaiah M. Enhancing phishing detection with dynamic optimization and character-level deep learning in cloud environments. *PeerJ Computer Science*. 2025. Vol. 11. Art. e2640. https://doi.org/10.7717/peerj-cs.2640

19. Ravula V., Ramaiah M. Enhancing phishing detection with dynamic optimization and character-level deep learning in cloud environments. *PeerJ Computer Science.* 2025. Vol. 11. Art. no. e2640. DOI: https://doi.org/10.7717/peerj-cs.2640

20. Moura D. L. L., de Aquino A. L. L., Loureiro A. A. F. An edge computing and distributed ledger technology architecture for secure and efficient transportation. *Ad Hoc Networks.* 2024. Vol. 164. Art. 103633. https://doi.org/10.1016/j.adhoc.2024.103633.

21. Alnaim A. K., Alwakeel A. M. Zero-trust mechanisms for securing distributed edge and fog computing in 6G networks. *Mathematics*. 2025. Vol. 13, No. 8. Art. 1239. https://doi.org/10.3390/math13081239.

22. Лисенко С., Атаманюк О., Бохонько О., Воробйов В. Дослідження методів виявлення кіберзагроз типу ransomware на основі застосування honeypot. *Вісник ХНУ*. 2023. № 1(317). С. 300–309. https://doi.org/10.31891/2307-5732-2023-317-1-300-309

23. Cai H. A survey of program analysis for distributed software systems. *ACM Computing Surveys*. 2025. Vol. 57, No. 12. Art. 300. P. 1–45. DOI: https://doi.org/10.1145/3742900.

24. Taleb I., Guillaume J.-L., Duthil B. A survey on services placement algorithms in integrated cloud–fog/edge computing. *ACM Computing Surveys*. 2025. Vol. 57, No. 11. Art. 271. P. 1–36. DOI: https://doi.org/10.1145/3729214.

25. Rollere I., Hartsfield C., Courtenay S., Fenwick L., Grunwald A. Algorithmic segmentation and behavioral profiling for ransomware detection using temporal-

correlation graphs. *arXiv preprint*. 2025. arXiv:2501.17429 [cs.CR]. DOI: https://doi.org/10.48550/arXiv.2501.17429.

26. Maia A., Boutouchent A., Kardjadja Y., Gherari M., Soyak E. G., Saqib M., Boussekar K., Cilbir I., Habibi S., Ouledsidi Ali S., Ajib W., Elbiaze H., Erçetin O., Ghamri-Doudane Y., Glitho R. A survey on integrated computing, caching, and communication in the cloud-to-edge continuum. *Computer Communications*. 2024. Vol. 219. P. 128–152. DOI: https://doi.org/10.1016/j.comcom.2024.03.005.

27. Giamattei L., Guerriero A., Pietrantuono R., Russo S., Malavolta I., Islam T., Dînga M., Koziolek A., Singh S., Armbruster M., Gutierrez-Martinez J. M., Caro-Alvaro S., Rodriguez D., Weber S., Henss J., Fernandez Vogelin E., Simon Panojo F. Monitoring tools for DevOps and microservices: a systematic grey literature review. *Journal of Systems and Software*. 2024. Vol. 208. Art. 111906. DOI: https://doi.org/10.1016/j.jss.2023.111906.

28. Starchenko E., Bellinghamshire H., Pickering D., Weatherspoon T., Berkhamstead N., Green E., Rothschild M. Decentralized entropy-driven ransomware detection using autonomous neural graph embeddings. *arXiv preprint*. 2025. arXiv:2502.07498 [cs.CR]. DOI: https://doi.org/10.48550/arXiv.2502.07498

29. Soldani J., Amadini R., Brogi A., Forti S., Giallorenzo S., Plebani P., Vitali M., Zavattaro G. Towards sustainable deployment of microservices over the cloud–IoT continuum, with FREEDA. In: Proceedings of the 4th Workshop on Flexible Resource and Application Management on the Edge (FRAME ’24). 2024. P. 1–4. DOI: https://doi.org/10.1145/3659994.3660311.

30. Calavaro C., Cardellini V., Lo Presti F., Russo G. Beyond cloud: serverless functions in the compute continuum. *SN Computer Science*. 2025. Vol. 6. Art. 194. DOI: https://doi.org/10.1007/s42979-025-03699-7.

31. Pakina A. K., Kejriwal D., Pujari T. D. Adversarial AI in social engineering attacks: large-scale detection and automated countermeasures. *International Journal of Science and Technology*. 2025. Vol. 4, No. 1. DOI: https://10.56127/ijst.v4i1.1964.

32. Zhou J., Zhang K., Zheng B., Zhou Y., Xie X., Jin M., Liu X. A malicious URL detection framework based on custom hybrid spatial sequence attention and logic

constraint neural network. *Symmetry*. 2025. Vol. 17. Art. no. 987. DOI: https://doi.org/10.3390/sym17070987

33. Schmitt M., Flechais I. Digital deception: generative artificial intelligence in social engineering and phishing. *Artificial Intelligence Review.* 2024. Vol. 57. Art. 324. DOI: https://doi.org/10.1007/s10462-024-10973-2.

34. Sharevski, F., Mossano, M., Veit, M., Schiefer, G., & Volkamer, M. (2024). Exploring phishing threats through QR codes in naturalistic settings. In *Proceedings of the Symposium on Usable Security and Privacy (USEC 2024)*, San Diego, CA, USA. https://doi.org/10.14722/usec.2024.23050.

35. Murhej M., Nallasivan, G. Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-BASED EAI-SC-LSTM. *Frontiers in Communications and Networks*, 2025. Vol. 6(. P.587654. https://doi.org/10.3389/frcmn.2025.1587654.

36. Samarin N., Sanchez A., Chung T., Juleemun A. D. B., Gilsenan C., Merrill N., Reardon J., Egelman S. The Medium is the Message: How secure messaging apps leak sensitive data to push notification services. *Proceedings on Privacy Enhancing Technologies,* 2024. Vol. (4), pp. 967–982. https://doi.org/10.56553/popets-2024-0151.

37. Karim A., Shahroz M., Mustofa K., Belhaouari S.B., Joga S.R.K. Phishing Detection System Through Hybrid Machine Learning Based on URL. *IEEE Access*. 2023. Vol. 11. P. 36805–36822. DOI: 10.1109/ACCESS.2023.3252366

38. Li W., Manickam S., Chong Y.W. FedPhishLLM: A privacy-preserving and explainable phishing detection mechanism using federated learning and LLMs. *J. King Saud Univ. Comput. Inf. Sci.* 2025. Vol. 37. Art. no. 252. DOI: 10.1007/s44443-025-00267-0

39. Lim K., Park J., Kim D. Phishing Vs. Legit: Comparative Analysis of Client-Side Resources of Phishing and Target Brand Websites. *Proceedings of the ACM Web Conference 2024 (WWW '24)*. 2024. P. 1756–1767. DOI: 10.1145/3589334.3645535.

40. Das S., Kim D., Abbott J., Camp L.J. User-Centered Phishing Detection through Personalized Edge Computing. *Companion Publication of the 2024*

Conference on Computer-Supported Cooperative Work and Social Computing (CSCW Companion '24). 2024. P. 283–287. DOI: 10.1145/3678884.3681864.

41. Trad F., Semaan-Nasr E., Chehab A. MLPhishChain: a machine learning-based blockchain framework for reducing phishing threats. *Frontiers in Blockchain*. 2024. Vol. 7. Art. no. 1484894. DOI: 10.3389/fbloc.2024.1484894

42. Sheng Z., Song L., Wang Y. Dynamic Feature Fusion: Combining Global Graph Structures and Local Semantics for Blockchain Phishing Detection. *IEEE Transactions on Network and Service Management*. 2025. Vol. PP. No. 99. P. 1–1. DOI: 10.1109/TNSM.2025.3576130

43. Li W., Manickam S., Chong Y.W. FedPhishLLM: A privacy-preserving and explainable phishing detection mechanism using federated learning and LLMs. *Journal of King Saud University - Computer and Information Sciences*. 2025. Vol. 37. Art. no. 252. DOI: 10.1007/s44443-025-00267-0

44. Rahaman M., Cajes N., Gupta B.B., Chui K.T., Nedjah N. Defending against smishing attacks: State-of-the-art techniques, challenges, limitations, and future directions. *Computer Networks*. 2025. Vol. 273. Art. no. 111758. DOI: 10.1016/j.comnet.2025.111758

45. Saha Roy S., Nilizadeh S. PhishLang: A Real-Time, Fully Client-Side Phishing Detection Framework Using MobileBERT. *arXiv preprint arXiv:2408.05667*. 2025. DOI: 10.48550/arXiv.2408.05667

46. Li W., Manickam S., Chong Y.W., Karuppayah S. PhishIntentionLLM: Uncovering Phishing Website Intentions through Multi-Agent Retrieval-Augmented Generation. *arXiv preprint arXiv:2507.15419*. 2025. DOI: 10.48550/arXiv.2507.15419

47. Lim K., Lee K., Ji F., Kwon Y., Kim H., Kim D. What's in Phishers: A Longitudinal Study of Security Configurations in Phishing Websites and Kits. *Proceedings of the ACM on Web Conference 2025 (WWW '25)*. 2025. P. 957–968. DOI: 10.1145/3696410.3714710

48. Tulla M.H.B., Ratan M.M.R., Mamunur R.M., Sohan A.H.S., Rahman M.M. Explainable Machine Learning for Phishing Detection: Bridging Technical Efficacy

and Legal Accountability in Cyberspace Security. *Journal of Cyber Security*. 2025. Vol. 7. Art. no. 074737. DOI: 10.32604/jcs.2025.074737

49. Mehmood M.K., Arshad H., Alawida M., Mehmood A. Enhancing Smishing Detection: A Deep Learning Approach for Improved Accuracy and Reduced False Positives. *IEEE Access*. 2024. Vol. 12. P. 137176–137193. DOI: 10.1109/ACCESS.2024.3463871

50. Mulahuwaish A., Qolomany B., Gyorick K., Bou Abdo J., Aledhari M., Qadir J., Carley K., Al-Fuqaha A. A survey of social cybersecurity: Techniques for attack detection, evaluations, challenges, and future prospects. *Computers in Human Behavior Reports*. 2025. Vol. 18. Art. no. 100668. DOI: 10.1016/j.chbr.2025.100668

51. Li W., He Y., Wang Z., Alqahtani S.M., Nanda P. Uncovering Flaws in Anti-Phishing Blacklists for Phishing Websites Using Novel Cloaking Techniques. *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*. 2023. P. 813–821. DOI: 10.5220/0012117500003555

52. Xiong A., Tong Y., Jiang C., Guo S., Shao S., Huang J., Wang W., Qi B. Ethereum phishing detection based on graph neural networks. *IET Blockchain*. 2023. Vol. 4. No. 3. P. 159–175. DOI: 10.1049/blc2.12031

52. Zhang J., Sui H., Sun X., Ge C., Zhou L., Susilo W. GrabPhisher: Phishing Scams Detection in Ethereum via Temporally Evolving GNNs. *IEEE Transactions on Services Computing*. 2024. Vol. 17. P. 3727–3741. DOI: 10.1109/TSC.2024.3411449

54. Maturure P., Ali A., Gegov A. Hybrid Machine Learning Model for Phishing Detection. *Proceedings of the 2024 IEEE 12th International Conference on Intelligent Systems (IS)*. 2024. DOI: 10.1109/IS61756.2024.10705257

55. Li W., He Y., Wang Z., Alqahtani S., Nanda P. Uncovering Flaws in Anti-Phishing Blacklists for Phishing Websites Using Novel Cloaking Techniques. *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*. 2023. P. 813–821. DOI: 10.5220/0012135600003555

56. Woollacott E. Malicious URLs overtake email attachments as the biggest malware threat. *IT Pro*. 2025. URL: https://www.itpro.com/security/cyber-attacks/malicious-urls-overtake-email-attachments-as-the-biggest-malware-threat

57. Reyes-Dorta N., Caballero-Gil P., Rosa-Remedios C. Detection of malicious URLs using machine learning. *Wireless Networks*. 2024. Vol. 30. P. 7543–7560. DOI: 10.1007/s11276-024-03700-w

58. Ghalechyan H., Keshavarz H., Ghaemi A., et al. Phishing URL detection using neural networks. *Scientific Reports*. 2024. Vol. 14(25134). DOI: 10.1038/s41598-024-74725-6

59. Barik K., Misra S., Mohan R. Web-based phishing URL detection model using deep learning optimization techniques (EGSO–CNN). *Journal of Ambient Intelligence and Humanized Computing*. 2025. Vol. 20. P. 4449–4471. DOI: 10.1007/s41060-025-00728-9

60. Albahadili A.J.S., Akbas A., Rahebi J. Detection of phishing URLs with deep learning based on GAN-CNN-LSTM network and swarm intelligence algorithms. *Signal, Image and Video Processing*. 2024. Vol. 18. P. 4979–4995. DOI: 10.1007/s11760-024-03204-2

61. Sudar K.M., Rohan M., Vignesh K. Detection of adversarial phishing attack using machine learning techniques. *Sādhanā*. 2024. Vol. 49(232). DOI: 10.1007/s12046-024-02582-0

62. Somesha M., Pais A.R. DeepEPhishNet: A deep learning framework for email phishing detection using word embedding algorithms. *Sādhanā*. 2024. Vol. 49(212). DOI: 10.1007/s12046-024-02538-4

63. Patra C., Giri D., Nandi S., Das A.K., Alenazi M.J.F. Phishing email detection using vector similarity search leveraging transformer-based word embedding. *Computers and Electrical Engineering*. 2025. Vol. 124. Art. no. 110403. DOI: 10.1016/j.compeleceng.2025.110403

64. Prasad Y.B., Dondeti V. PDSMV3-DCRNN: A novel ensemble deep learning framework for enhancing phishing detection and URL extraction. *Computers & Security*. 2025. Vol. 148. Art. no. 104123. DOI: 10.1016/j.cose.2024.104123

65. Rashid F., Doyle B., Han S.C., Seneviratne S. Phishing URL detection generalisation using unsupervised domain adaptation. *Computer Networks*. 2024. Vol. 245. Art. no. 110398. DOI: 10.1016/j.comnet.2024.110398.

66. Mia M., Derakhshan D., Pritom M.M.A. Can features for phishing URL detection be trusted across diverse datasets? A case study with explainable AI. *Proceedings of the 11th International Conference on Networking, Systems, and Security (NSysS '24)*. 2024. P. 137–145. DOI: 10.1145/3704522.3704532

67. Rawla A., Singh S., Daniyal M., Dubey P. Detection of phishing attacks in PhiUSIIL dataset using deep learning. *Procedia Computer Science*. 2025. Vol. 259. P. 543–552. DOI: 10.1016/j.procs.2025.04.003

68. Alsowail R.A. Anomaly detection based CapsNet for malicious URL detection system. *Wireless Networks*. 2025. Vol. 31(5). P. 3785–3801. DOI: 10.1007/s11276-025-03960-0

69. Komalasari D., Kurniawan T.B., Dewi D.A., Zakaria M.Z., Abdullah Z., Alanda A. Phishing domain detection using machine learning algorithms. *International Journal on Advanced Science, Engineering and Information Technology*. 2025. Vol. 15(1). P. 318–327. DOI: 10.18517/ijaseit.15.1.12553

70. Zhou J., Zhang K., Zheng B., Zhou Y., Xie X., Jin M., Liu X. A malicious URL detection framework based on custom hybrid spatial sequence attention and logic constraint neural network. *Symmetry*. 2025. Vol. 17. Art. no. 987. DOI: 10.3390/sym17070987

71. Birthriya S.K., Ahlawat P., Jain A.K. Phishing website detection with XGBoost and adaptive hyperparameter optimization using the Bat Algorithm. *Procedia Computer Science*. 2025. Vol. 258. P. 1774–1782. DOI: 10.1016/j.procs.2025.04.429

72. Haq Q.E.u., Faheem M.H., Ahmad I. Detecting phishing URLs based on a deep learning approach to prevent cyber-attacks. *Applied Sciences*. 2024. Vol. 14. Art. no. 10086. DOI: 10.3390/app142210086

73. Duarte J.D., et al. Machine learning for early detection of phishing URLs in parked domains: An approach applied to a financial institution. *IEEE Access*. 2025. Vol. 13. P. 145736–145753. DOI: 10.1109/ACCESS.2025.3599454

74. Bourigue R., Ait Omar D., Zougagh H. Improving online security: A deep learning model for phishing URL detection. *Cluster Computing*. 2025. Vol. 28. Art. no. 631. DOI: 10.1007/s10586-025-05307-y

75. Chen Z., Liu S.-Z., Huang J., Xiu Y.-H., Zhang H., Long H.-X. Ethereum phishing scam detection based on data augmentation method and hybrid graph neural network model. *Sensors*. 2024. Vol. 24. Art. no. 4022. DOI: 10.3390/s24124022

76. Ratra S., Ghosh M., Baliyan N., Mohan J.R., Singh S. Graph neural network based phishing account detection in Ethereum. *The Computer Journal*. 2024. Vol. 67(12). P. 3160–3168. DOI: 10.1093/comjnl/bxae079

77. Manguli K., Kondaiah C., Pais A.R., Rao R.S. GraPhish: A graph-based approach for phishing detection from encrypted TLS traffic. *Journal of Information Security and Applications*. 2025. Vol. 94. Art. no. 104216. DOI: 10.1016/j.jisa.2025.104216

78. Tang M., Ye M., Chen W., Zhou D. BiLSTM4DPS: An attention-based BiLSTM approach for detecting phishing scams in Ethereum. *Expert Systems with Applications*. 2024. Vol. 256. Art. no. 124941. DOI: 10.1016/j.eswa.2024.124941

79. Zhang L., et al. Unraveling the deception of Web3 phishing scams: Dynamic multiperspective cascade graph approach for Ethereum phishing detection. *IEEE Transactions on Computational Social Systems*. 2025. Vol. 12(2). P. 498–510. DOI: 10.1109/TCSS.2024.3516144

80. Sui H., Zhang J., Chen B., Wu D., Sun X., Palaiahnakote S. EPAD: Ethereum phishing scam detection via graph contrastive learning. *Expert Systems with Applications*. 2025. Vol. 288. Art. no. 128227. DOI: 10.1016/j.eswa.2025.128227

81. Ravula V., Ramaiah M. Enhancing phishing detection with dynamic optimization and character-level deep learning in cloud environments. *PeerJ Computer Science*. 2025. Vol. 11. Art. no. e2640. DOI: 10.7717/peerj-cs.2640

82. Jain A.K., Kaur K., Gupta N.K., et al. Detecting smishing messages using BERT and advanced NLP techniques. *SN Computer Science*. 2025. Vol. 6. Art. no. 109. DOI: 10.1007/s42979-024-03532-7

83. Alasmari S.M., Sakly H., Kraiem N., et al. Phishing detection in IoT: An integrated CNN–LSTM framework with explainable AI and LLM-enhanced analysis. *Discover Internet of Things*. 2025. Vol. 5. Art. no. 102. DOI: 10.1007/s43926-025-00202-9

84. Hapase D.S., Patil L.V. Telecommunication fraud resilient framework for efficient and accurate detection of SMS phishing using artificial intelligence techniques. *Multimedia Tools and Applications*. 2024. Vol. 83. P. 89111–89133. DOI: 10.1007/s11042-024-19020-2

85. Opara C., Chen Y., Wei B. Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Systems with Applications*. 2024. Vol. 236. Art. no. 121183. DOI: 10.1016/j.eswa.2023.121183

86. Mbevi R.M., Kamau J., Musyoka F.M. Content based approach for detecting smishing messages in mobile phones using an improved convolutional neural networks model. *African Journal of Empirical Research*. 2025. Vol. 6(2). P. 188–204. DOI: 10.51867/ajernet.6.2.17

87. Tanbhir G., Shahriyar M.F., Shahed K., Chy A.M.R., Adnan M.A. Hybrid machine learning model for detecting Bangla smishing text using BERT and character-level CNN. *Proceedings of the 13th International Conference on Electrical and Computer Engineering (ICECE)*. 2024. P. 57–62. DOI: 10.1109/ICECE64886.2024.11024872

88. Remmide M.A., Boumahdi F., Boustia N. Advancing automated social engineering detection with oversampling-based machine learning. *International Journal of Security and Networks*. 2024. Vol. 19(3). DOI: 10.1504/IJSN.2024.141783

89. Paithane P.M. URLGuard: A holistic hybrid machine learning approach for phishing detection. *International Journal of Information Engineering and Electronic Business*. 2025. Vol. 17(2). P. 95–110. DOI: 10.5815/ijieeb.2025.02.05

90. Boussougou M.K.M., Hamandawana P., Park D.-J. Enhancing voice phishing detection using multilingual back-translation and SMOTE: An empirical study. *IEEE Access*. 2025. Vol. 13. P. 37946–37965. DOI: 10.1109/ACCESS.2025.3545250.

91. Park H., Lee J., Han S., Byun H. Enhanced voice phishing detection using an LLM-based framework for data augmentation and classification. *IEEE Access*. 2025. Vol. 13. P. 152530–152545. DOI: 10.1109/ACCESS.2025.3603007

92. Kim J., Gu S., Kim Y., Lee S., Kang C. A multimodal voice phishing detection system integrating text and audio analysis. *Applied Sciences*. 2025. Vol. 15(20). Art. no. 11170. DOI: 10.3390/app152011170

93. Lee C., Kim B., Kim H. The silence of the phishers: Early-stage voice phishing detection with runtime permission requests. *Computers & Security*. 2025. Vol. 152. Art. no. 104364. DOI: 10.1016/j.cose.2025.104364

94. Triantafyllopoulos A., Spiesberger A.A., Tsangko I., et al. Vishing: Detecting social engineering in spoken communication — A first survey & urgent roadmap to address an emerging societal challenge. *Computer Speech & Language*. 2025. Vol. 94. Art. no. 101802. DOI: 10.1016/j.csl.2025.101802

95. Yu S., Kwon Y., Kim M., Lee K. Korean voice phishing detection applying NER with key tags and sentence-level N-gram. *IEEE Access*. 2024. Vol. 12. P. 52951–52962. DOI: 10.1109/ACCESS.2024.3387027

96. Vidyasri P., Suresh S. FDN-SA: Fuzzy deep neural-stacked autoencoder-based phishing attack detection in social engineering. *Computers & Security*. 2025. Vol. 148. Art. no. 104188. DOI: 10.1016/j.cose.2024.104188

97. Eze C.S., Shamir L. Analysis and prevention of AI-based phishing email attacks. *Electronics*. 2024. Vol. 13(10). Art. no. 1839. DOI: 10.3390/electronics13101839

98. Van Geest R.J., Cascavilla G., Hulstijn J., Zannone N. The applicability of a hybrid framework for automated phishing detection. *Computers & Security*. 2024. Vol. 139. Art. no. 103736. DOI: 10.1016/j.cose.2024.103736

99. Zhu E., Cheng K., Zhang Z., Wang H. PDHF: Effective phishing detection model combining optimal artificial and automatic deep features. *Computers & Security*. 2024. Vol. 136. Art. no. 103561. DOI: 10.1016/j.cose.2023.103561

100. Sui H., Zhang J., Chen B., Wu D., Sun X., Palaiahnakote S. EPAD: Ethereum phishing scam detection via graph contrastive learning. *Expert Systems with Applications*. 2025. Vol. 288. Art. no. 128227. DOI: 10.1016/j.eswa.2025.128227

101. Meda S., Srinivas V.S., Rao K.C.B., Ramesh R., Yamarthi N.R. A dual-phase deep learning framework for advanced phishing detection using the novel OptSHQCNN approach. *PeerJ Computer Science*. 2025. Vol. 11. Art. no. e3014. DOI: 10.7717/peerj-cs.3014

102. Yu L., Tao J., Xu Y., Sun W., Wang Z. TLS fingerprint for encrypted malicious traffic detection with attributed graph kernel. *Computer Networks*. 2024. Vol. 247. Art. no. 110475. DOI: 10.1016/j.comnet.2024.110475

103. Roy P.K., Kumar A., Singh A. Advanced learning for phishing URLs detection to secure consumer-centric applications. *IEEE Transactions on Consumer Electronics*. 2024. Vol. 70(3). P. 5756–5763. DOI: 10.1109/TCE.2024.3404459

104. Asiri S., Xiao Y., Alzahrani S., Li T. PhishingRTDS: A real-time detection system for phishing attacks using a deep learning model. *Computers & Security*. 2024. Vol. 141. Art. no. 103843. DOI: 10.1016/j.cose.2024.103843

105. Uddin K.M.M., Biswas N., Rikta S.T., et al. Explainable machine learning for phishing site detection: A high-efficiency approach using boosting models and SHAP. *The Journal of Engineering*. 2025. Vol. 2025(1). Art. no. e70110. DOI: 10.1049/tje2.70110

106. Freitas S., Gharib A. GraphWeaver: Billion-scale cybersecurity incident correlation. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*. 2024. P. 4479–4486. DOI: 10.1145/3627673.3680057

107. Govindarajan V., Muzamal J.H. Advanced cloud intrusion detection framework using graph based features, transformers and contrastive learning. *Scientific Reports*. 2025. Vol. 15. Art. no. 20511. DOI: 10.1038/s41598-025-07956-w

108. Jishnu K.S., Arthi B. Generative adversarial network-based phishing URL detection with variational autoencoder and transformer. *IAES International Journal of*

Artificial Intelligence. 2024. Vol. 13(2). P. 2165–2172. DOI: 10.11591/ijai.v13.i2.pp2165-2172

109. Aljofey A., Bello S.A., Lu J., Xu C. BERT-PhishFinder: A robust model for accurate phishing URL detection with optimized DistilBERT. *IEEE Transactions on Dependable and Secure Computing*. 2025. Vol. 22(4). P. 4315–4329. DOI: 10.1109/TDSC.2025.3545771

110. Meléndez R., Ptaszynski M., Masui F. Comparative investigation of traditional machine-learning models and transformer models for phishing email detection. *Electronics*. 2024. Vol. 13(24). Art. no. 4877. DOI: 10.3390/electronics13244877

111. Paul M.R.R., Santhi Sree K. Ensemble based detection of phishing URLs using hybrid, deep learning and machine learning models. *International Journal for Research in Applied Science & Engineering Technology*. 2025. Vol. 13(5). P. 6402–6415. DOI: 10.22214/ijraset.2025.71708

112. Al-Subaiey A., Al-Thani M., Alam N.A., et al. Novel interpretable and robust web-based AI platform for phishing email detection. *Computers and Electrical Engineering*. 2024. Vol. 120(A). Art. no. 109625. DOI: 10.1016/j.compeleceng.2024.109625

113. Shaikh A., Shaikh M., Ramanujam S. Smishing detection: Combating SMS phishing attacks by utilizing machine-learning algorithms. *International Journal of Innovative Technology and Exploring Engineering*. 2025. Vol. 14(5). P. 28–33. DOI: 10.35940/ijitee.D1068.14050425

114. Komosny D. Phishing detection on webpages in European non-English languages based on machine learning. *Scientific Reports*. 2025. Vol. 15. Art. no. 37472. DOI: 10.1038/s41598-025-21384-w

115. Alhuzali A., Alloqmani A., Aljabri M., Alharbi F. In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets. *Applied Sciences*. 2025. Vol. 15(6). Art. no. 3396. DOI: 10.3390/app15063396

116. Yang M., Balasubramanian P., Chen K., Oruklu E. Leakage power attack-resilient design: PMOS-reading 9T SRAM cell. *Electronics*. 2024. Vol. 13(13). Art. no. 2551. DOI: 10.3390/electronics13132551.

117. Hosseinzadeh M., Ali U., Ali S., et al. Improving phishing email detection performance through deep learning with adaptive optimization. *Scientific Reports*. 2025. Vol. 15. Art. no. 20668. DOI: 10.1038/s41598-025-20668-5

118. Remmide M.A., Boumahdi F., Ilhem B. A privacy-preserving approach for detecting smishing attacks using federated deep learning. *International Journal of Information Technology*. 2025. Vol. 17. P. 547–553. DOI: 10.1007/s41870-024-02144-x

119. Jiang H., Zuo J., Lu Y. Connecting visual data to privacy: Predicting and measuring privacy risks in images. *Electronics*. 2025. Vol. 14(4). Art. no. 811. DOI: 10.3390/electronics14040811

120. Wang L., Xu M., Cheng H. Phishing scams detection via temporal graph attention network in Ethereum. *Information Processing & Management*. 2023. Vol. 60(4). Art. no. 103412. DOI: 10.1016/j.ipm.2023.103412

121. Mahmud T., Prince M.A.H., Ali M.H., Hossain M.S., Andersson K. Enhancing cybersecurity: Hybrid deep learning approaches to smishing attack detection. *Systems*. 2024. Vol. 12(11). Art. no. 490. DOI: 10.3390/systems12110490

122. Mehmood M.K., Arshad H., Alawida M., Mehmood A. Enhancing smishing detection: A deep learning approach for improved accuracy and reduced false positives. *IEEE Access*. 2024. Vol. 12. P. 137176–137193. DOI: 10.1109/ACCESS.2024.3463871

123. Wu P., Gao M., Sun F., Wang X., Pan L. Multi-perspective API call sequence behavior analysis and fusion for malware classification. *Computers & Security*. 2025. Vol. 148. Art. no. 104177. DOI: 10.1016/j.cose.2024.104177

124. Tanbhir G., Shahriyar M.F., Shahed K., Chy A.M.R., Adnan M.A. Hybrid machine learning model for detecting Bangla smishing text using BERT and character-level CNN. *Proceedings of the 13th International Conference on Electrical and*

Computer Engineering (ICECE 2024). 2024. P. 57–62. DOI: 10.1109/ICECE64886.2024.11024872

125. Ulfath R.E., Alqahtani H., Hammoudeh M., Sarker I.H. Hybrid CNN-GRU framework with integrated pre-trained language transformer for SMS phishing detection. *Proceedings of the 5th International Conference on Future Networks and Distributed Systems*. 2021. P. 244–251. DOI: 10.1145/3508072.3508109

126. Altunay H.C., Albayrak Z. SMS spam detection system based on deep learning architectures for Turkish and English messages. *Applied Sciences*. 2024. Vol. 14(24). Art. no. 11804. DOI: 10.3390/app142411804

127. Komati L.C., Dunaboyina D.B., Shaik S., Muthyala V.G.K.R. Malicious URL website detection using ensemble machine learning approach. *International Journal of Science and Research Archive*. 2025. Vol. 14(3). P. 1614–1622. DOI: 10.30574/ijsra.2025.14.3.0857

128. Ujah-Ogbuagu B.C., Akande O.N., Ogbuju E.A. A hybrid deep learning technique for spoofing website URL detection in real-time applications. *Journal of Electrical Systems and Information Technology*. 2024. Vol. 11(7). DOI: 10.1186/s43067-023-00128-8

129. Gupta B.B., Gaurav A., Attar R.W., et al. Optimized phishing detection with recurrent neural network and Whale Optimizer Algorithm. *Computers, Materials & Continua*. 2024. Vol. 80(3). P. 4895. DOI: 10.32604/cmc.2024.050815

130. Senouci O., Benaouda N. Enhancing phishing detection in cloud environments using RNN-LSTM in a deep learning framework. *Journal of Telecommunications and Information Technology*. 2025. Vol. 2025(1). P. 1916. DOI: 10.26636/jtit.2025.1.1916

131. Shaik M., Hussain S.A., Amareswari V., et al. Enhanced suspicious URL detection in IoT using an optimized hybrid selection technique. *International Journal of Computational Learning & Intelligence*. 2025. Vol. 4(2). P. 432–439. DOI: 10.5281/zenodo.15181423

132. Murhej M., Nallasivan G. Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-

BASED EAI-SC-LSTM. *Frontiers in Communications and Networks*. 2025. Vol. 6. Art. no. 1587654. DOI: 10.3389/frcmn.2025.1587654

133. Xue Y., Spero E., Koh Y.S., Russello G. MultiPhishGuard: An LLM-based multi-agent system for phishing email detection. *arXiv preprint*. 2025. DOI: 10.48550/arXiv.2505.23803

134. Kavya S., Sumathi D. Multimodal and temporal graph fusion framework for advanced phishing website detection. *IEEE Access*. 2025. Vol. 13. P. 74128–74146. DOI: 10.1109/ACCESS.2025.3564530

135. Ji F., Lee K., Koo H., et al. Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. *Proceedings of the 34th USENIX Security Symposium (USENIX Security 2025)*. 2025.

136. Repetto M. Cybersecurity digital twins: Concept, blueprint, and challenges for multi-ownership digital service chains. *Journal of Information Security and Applications*. 2026. Vol. 96. Art. no. 104299. DOI: 10.1016/j.jisa.2025.104299

137. Yao Y., Zhu Y., Jia Y., et al. Research on malware detection technology for mobile terminals based on API call sequence. *Mathematics*. 2024. Vol. 12(1). Art. no. 20. DOI: 10.3390/math12010020

138. Alsaidi S.A.A.A., Mohammed H.J., Al Ogaili R.R.N., et al. HawkPhish-DNN cybersecurity model: Adaptive hybrid optimization and deep learning for enhanced multi-objective phishing URL detection. *International Journal of Information Technology*. 2025. Vol. 17. P. 3859–3875. DOI: 10.1007/s41870-025-02597-8

139. Jishnu K.S., Arthi B. Real-time phishing URL detection framework using knowledge distilled ELECTRA. *Automatika*. 2024. Vol. 65(4). P. 1621–1639. DOI: 10.1080/00051144.2024.2415797

140. Do N.Q., Selamat A., Fujita H., et al. An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection. *Future Generation Computer Systems*. 2024. Vol. 161. P. 269–285. DOI: 10.1016/j.future.2024.06.031

141. Naz A., Sarwar M., Kaleem M., et al. A comprehensive survey on social engineering-based attacks on social networks. *International Journal of Advanced and Applied Sciences*. 2024. Vol. 11(4). P. 16. DOI: 10.21833/ijaas.2024.04.016

142. Ani U.D., Al-Mhiqani M., Tuptuk N., et al. Socio-technical security modelling and simulations in cyber-physical systems: Outlook on knowledge, perceptions, practices, enablers, and barriers. *IET Cyber-Physical Systems: Theory & Applications*. 2025. DOI: 10.1049/cps2.70017

143. Zhou Y., Wang Z., Jiang Y., et al. AEKG4APT: An AI-enhanced knowledge graph for advanced persistent threats with large language model analysis. *ACM Transactions on Intelligent Systems and Technology*. 2025. DOI: 10.1145/3735645

144. Chen Y., Zhang Y., Chan S., et al. Multilayer topology-aware graph contrastive learning for fraud detection in the Ethereum transaction network. *Journal of the Royal Statistical Society Series A: Statistics in Society*. 2025. DOI: 10.1093/jrsssa/qnaf135

145. Sanjari S.M., Shibli A.M., Mia M., et al. SmishViz: Towards a graph-based visualization system for monitoring and characterizing ongoing smishing threats. *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy (CODASPY '25)*. 2025. P. 257–268. DOI: 10.1145/3714393.3726499

146. Tian Y., Liu G., Wang J., Zhou M. ASA-GNN: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection. *IEEE Transactions on Computational Social Systems*. 2024. Vol. 11(3). P. 3536–3549. DOI: 10.1109/TCSS.2023.3335485

147. Zhao H., Chen H., Yang F., et al. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*. 2024. Vol. 15(2). P. 1–38. DOI: 10.1145/3639372

148. Hu Y., Zou F., Han J., et al. LLM-TIKG: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*. 2024. Vol. 145. Art. no. 103999. DOI: 10.1016/j.cose.2024.103999

149. Dasgupta R., Mitra P. Large language model-based federated zero-shot learning for intrusion detection in smart grids. *Artificial Intelligence Applications and Innovations*. 2025. P. 331–344. DOI: 10.1007/978-3-031-96235-6_24

150. Schneider N.R., Das A., O'Sullivan K., Samet H. Cross-lingual clustering using large language models. *Proceedings of the 7th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery (GeoAI '24)*. 2024. P. 1–10. DOI: 10.1145/3687123.3698280

151. Xu C., Zhan Y., Wang Z., Yang J. Multimodal fusion based few-shot network intrusion detection system. *Scientific Reports*. 2025. Vol. 15. Art. no. 21986. DOI: 10.1038/s41598-025-05217-4

152. Messaoudi S., Meliani A.-E., Mokhtari A., Ksentini A. Security and trust management in cloud edge continuum: AC3 project approach. *Proceedings of the 29th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2024)*. 2024. P. 1–6. DOI: 10.1109/CAMAD62243.2024.10942825.

153. Ren K., Zeng Y., Zhong Y., et al. MAFSIDS: a reinforcement learning-based intrusion detection model for multi-agent feature selection networks. *J. Big Data*. 2023. Vol. 10. Art. no. 137. DOI: 10.1186/s40537-023-00814-4

154. Mouyart M., Medeiros Machado G., Jun J.-Y. A multi-agent intrusion detection system optimized by a deep reinforcement learning approach with a dataset enlarged using a generative model to reduce the bias effect. *J. Sensor Actuator Netw.* 2023. Vol. 12(5). Art. no. 68. DOI: 10.3390/jsan12050068

155. Bacha A., Ktata F.B., Louati F. Improving intrusion detection systems with multi-agent deep reinforcement learning: Enhanced centralized and decentralized approaches. *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*. 2023. Vol. 1. P. 772–777. DOI: 10.5220/0012124600003555

156. Soltani M., Khajavi K., Jafari Siavoshani M., et al. A multi-agent adaptive deep learning framework for online intrusion detection. *Cybersecurity*. 2024. Vol. 7. Art. no. 9. DOI: 10.1186/s42400-023-00199-0

157. Louati F., Ktata F., Amous I. Big-IDS: a decentralized multi agent reinforcement learning approach for distributed intrusion detection in big data networks. *Cluster Comput.* 2024. Vol. 27. P. 1–19.

158. Funchal G., Pedrosa T., De La Prieta F., Leitão P. Edge multi-agent intrusion detection system architecture for IoT devices with cloud continuum. *Proceedings of the 2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*. 2024. P. 1–6. DOI: 10.1109/ICPS59941.2024.10639952

159. Tellache A., Mokhtari A., Amara Korba A., Ghamri-Doudane Y. Multi-agent reinforcement learning-based network intrusion detection system. *Proceedings of the 2024 IEEE Network Operations and Management Symposium (NOMS)*. 2024. P. 1–9. DOI: 10.1109/NOMS61131.2024.10575395

160. Lakshminarayana S., Basarkod P. Enhancing intrusion detection in next-generation networks based on a multi-agent game-theoretic framework. *IAES Int. J. Artif. Intell.* 2024. Vol. 13(4). P. 4856–4868. DOI: 10.11591/ijai.v13.i4.pp4856-4868

161. Hmimou Y., Tabaa M., Khiat A., Hidila Z. A multi-agent system for cybersecurity threat detection and correlation using large language models. *IEEE Access*. 2025. Vol. PP. P. 1–1. DOI: 10.1109/ACCESS.2025.3602681

162. Manikandan A., Rajan S.D. Cyber attack detection using deep multi-agent reinforcement learning with Beth dataset. *SN Comput. Sci.* 2025. Vol. 6(5). Art. no. 433. DOI: 10.1007/s42979-025-03981-8

163. Saeed M.Y., He J., Zhu N., et al. Collaborative multi-agent XRL for threat detection in mobile edge network traffic. *Complex Intell. Syst.* 2025. Vol. 11. Art. no. 446. DOI: 10.1007/s40747-025-02079-1

164. Qin Z., Luo Q., Nong X., et al. MAS-LSTM: A multi-agent LSTM-based approach for scalable anomaly detection in IIoT networks. *Processes*. 2025. Vol. 13(3). Art. no. 753. DOI: 10.3390/pr13030753

165. Bacha A., Ktata F., Belkahla Driss O. Multi-agent deep Q-network based ant colony optimization for advanced network intrusion detection. *Proceedings of the 2025 International Symposium on Digital Forensic and Security (ISDFS)*. 2025. P. 1–6. DOI: 10.1109/ISDFS65363.2025.11012022

166. Shafiq S., Awan H.M., Khan A.A., Hussain A., Javed I. MAFSID: Multi-agent few-shot intrusion detection for VANETs through rapid collaborative learning. *Trans. Emerg. Telecommun. Technol.* 2025. Vol. 36(11). Art. no. e70285. DOI: 10.1002/ett.70285

167. Finistrella S., Mariani S., Zambonelli F. Multi-agent reinforcement learning for cybersecurity: Classification and survey. *Intell. Syst. With Appl.* 2025. Vol. 26. Art. no. 200495. DOI: 10.1016/j.iswa.2025.200495

168. Landolt C.R., Würsch C., Meier R., Mermoud A., Jang-Jaccard J. Multi-agent reinforcement learning in cybersecurity: From fundamentals to applications. *arXiv preprint*. 2025. DOI: 10.48550/arXiv.2505.19837.

169. 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. No. 4. P. 112–151. DOI: https://doi.org/10.32620/reks.2023.4.10

170. 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. No. 1. P. 152–175. DOI: https://doi.org/10.32620/reks.2024.1.13

171. Lysenko S., Bobrovnikova K., Kharchenko V., Savenko O. IoT Multi-Vector Cyberattack Detection Based on Machine Learning Algorithms: Traffic Features Analysis, Experiments, and Efficiency. Algorithms. 2022. Vol. 15. No. 7. Art. no. 239. DOI: https://doi.org/10.3390/a15070239

172. Savenko B., Kashtalian A., Lysenko S., Savenko O. Malware Detection By Distributed Systems with Partial Centralization // Proceedings of the 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). 2023. Vol. 1. P. 265–270. DOI: https://doi.org/10.1109/IDAACS58523.2023.10348773.

173. Bokhonko O., Atamaniuk O., Sochor T. Model of a distributed heterogeneous system resistant to leakage of confidential information. *CEUR-WS*. 2025. Vol. 3963, pp. 363–376.URL: https://ceur-ws.org/Vol-3963/paper29.pdf .

174. Lysenko S., Bokhonko O., Savenko O., Vorobiov V., Gaj P., Wołoszyn J. Social Engineering Attacks Detection Approach. *Proceedings of 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2023, Athens, Greece, October 13-15, 2023). Pp. 318-329.

175. Ye M. On resilience against cyber-physical uncertainties in distributed Nash equilibrium seeking strategies for heterogeneous games. *IEEE/CAA Journal of Automatica Sinica*. 2025. Vol. 12. No. 1. P. 138–147. DOI: https://doi.org/10.1109/JAS.2024.124803.

176. Бохонько О., Лисенко С. Метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії. Measuring and computing devices in technological processes. 2025. Vol. 84(4). Pp. 152–163. https://doi.org/10.31891/2219-9365-2025-84-16 .

177. Ba Alawi Z. A comparative survey of PyTorch vs TensorFlow for deep learning: Usability, performance, and deployment trade-offs. *arXiv preprint*. 2025. arXiv:2508.04035. DOI: 10.48550/arXiv.2508.04035

178. NVIDIA. NVIDIA cuDNN 9: GPU-accelerated primitives for deep neural networks (Documentation). *NVIDIA Developer*. 2024. URL: https://developer.nvidia.com/cudnn

179. Microsoft. Playwright 1.50: End-to-end testing for modern web apps (Documentation). *Microsoft Playwright*. 2024. URL: https://playwright.dev/

180. Ratul I. J., Zhou Y., Yang K. Accelerating deep learning inference: A comparative analysis of modern acceleration frameworks. *Electronics*. 2025. Vol. 14(15). Art. no. 2977. DOI: 10.3390/electronics14152977

181. Xiahui Z., Zhongying N., Licheng W. Application of SimHash algorithm based on bucket index in deduplication of privacy data. *Proceedings of the IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE 2025)*. Shenyang, China. 2025. P. 6–9. DOI: 10.1109/EESPE63401.2025.10987512ю

182. Бохонько О., Лисенко С. Моделі атак соціальної інженерії. *Measuring and Computing Devices in Technological Processes.* 2025. No. 1. P. 432–444. DOI: https://doi.org/10.31891/2219-9365-2025-81-55 .

183. Lysenko S., Bokhonko O., Savenko O., Gaj P. Social engineering attacks models. In *Proceedings of the 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2024) (Athens, Greece, October 11–13, 2024).

184. Lysenko S., Bokhonko O., Vorobiyov V., Gaj P. A method for identifying cyberattacks based on the use of social engineering over the phone. *CEUR-WS*, 2024. Vol. 3675. Pp. 318–329. URL: https://ceur-ws.org/Vol-3675/paper23.pdf .

185. MacWhinney B. Understanding language through TalkBank. *Current Directions in Psychological Science.* 2025. Vol. 34. No. 2. DOI: https://doi.org/10.1177/09637214241304345.

186. Shahid R., Wali A., Bashir M. Next word prediction for Urdu language using deep learning models. *Computer Speech & Language*. 2024. Vol. 87. Art. 101635. DOI: https://doi.org/10.1016/j.csl.2024.101635

187. Morrone G., Cornell S., Raj D., Serafini L., Zovato E., Brutti A., Squartini S. Low-latency speech separation guided diarization for telephone conversations. *arXiv preprint.* 2022. arXiv:2204.02306. DOI: https://doi.org/10.48550/arXiv.2204.02306

188. Chicco D., Tötsch N., Jurman G. The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining*. 2021. Vol. 14. Art. 13. DOI: https://doi.org/10.1186/s13040-021-00244-z.

189. Бохонько О., Атаманюк О. Метод синтезу масштабованої архітектури розподіленої КС, стійкої до атак соціальної інженерії. *Computer Systems and Information Technologies*. 2025. № 4. С. 60–76. DOI: https://doi.org/10.31891/csit-2025-4-7.

190. Martín, J., Parra, M. I., Pizarro, M. M., & Sanjuán, E. L. A new Bayesian method for estimation of value at risk and conditional value at risk. *Empirical Economics.* 2025. Vol. *68*(3). Pp 1171-1189.

ДОДАТОК А.
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Статті у наукових виданнях, включених до Переліку наукових фахових видань України:

1. Лисенко С., Атаманюк О., Бохонько О., Воробйов В. Дослідження методів виявлення кіберзагроз типу RANSOMWARE на основі застосування HONEYPOT. *Вісник ХНУ*. 2023. №1, (317). С. 300-309. https://doi.org/10.31891/2307-5732-2023-317-1-300-309

2. Лисенко С., Бохонько О. Методи виявлення кібератак соціальної інженерії. *Вісник ХНУ*. 2023. №*327*(5(2). С. 231-236. https://doi.org/10.31891/2307-5732-2023-327-5-231-236.

3. Бохонько О., Лисенко С. Моделі атак соціальної інженерії. *Measuring and computing devices in technological processes.* 2025. *№* (1), С. 432–444. https://doi.org/10.31891/2219-9365-2025-81-55 .

4. Бохонько О. Лисенко С. Метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії. *Measuring and computing devices in technological processes,* vol. 84(4), pp. 152–163. https://doi.org/10.31891/2219-9365-2025-84-16 .

5. Bokhonko O., Atamaniuk O. Method for synthesis of a scalable architecture of a distributed computer systems, resistant to social engineering attacks. Computer Systems and Information Technologies. 2025. Vol.4. pp. 60-76. https://doi.org/10.31891/csit-2025-4-7

Праці, які засвідчують апробацію матеріалів дисертації

6. Lysenko S., Bokhonko O., Savenko O., Vorobiov V., Gaj P., Wołoszyn J. Social Engineering Attacks Detection Approach. *Proceedings of 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2023, Athens, Greece, October 13-15, 2023). Pp. 318-329.

7. Lysenko S., Bokhonko O., Vorobiyov V., Gaj P. A Method for identifying cyberattacks based on the use of social engineering over the phone. *CEUR-WS.* 2024. Vol. 3675. Pp. 318-329. URL: https://ceur-ws.org/Vol-3675/paper23.pdf .

8. Lysenko S., Bokhonko O., Savenko O., Gaj P., Social Engineering Attacks Models. *Proceedings of 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies* (DeSSerT-2024, Athens, Greece, October 11-13, 2024).

9. Bokhonko O., Atamaniuk O., Sochor T. Model of a distributed heterogeneous system resistant to leakage of confidential information *CEUR-WS.* 2025. Vol. 3963. Pp. 363-376. URL: https://ceur-ws.org/Vol-3963/paper29.pdf .

ДОДАТОК Б.
АКТИ ВПРОВАДЖЕННЯ

«Затверджую»
Директор
ПП «АВІВІ»
_____ В'ячеслав АСКЕРОВ
«__» _____ 2025 р.

АКТ

впровадження результатів дисертаційної роботи

Олександра БОХОНЬКА

Комісія в складі:

директор	В'ячеслав АСКЕРОВ
інженер програміст	В'ячеслав НАГДИБІДА
інженер програміст	Дмитро ІВАНОВ

склала акт про впровадження результатів роботи дисертаційної роботи наукового співробітника Хмельницького національного університету Олександра БОХОНЬКА на підприємстві «АВІВІ» у тому, що він проводив роботу з впровадження розподіленої КОМП'ЮТЕРНОЇ СИСТЕМИ, СТІЙКОЇ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ.

В процесі впровадження і розробки розподіленої системи діагностування комп'ютерних систем на наявність троянських програм були одержані особисто Олександра БОХОНЬКА і використані на підприємстві «АВІВІ» такі результати:

1. Розроблено метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, який на відміну від відомих підходів базується на ієрархічній багатоагентній архітектурі з підкріплювальним навчанням, ентропійно-орієнтованими функціями винагороди, апріорними знаннями у вигляді графа знань та модально-специфічними сервісними

агентами, що дає змогу адаптивно синтезувати архітектуру системи, зменшувати невизначеність у процесі виявлення атак, скорочувати кількість діалогових кроків і підвищувати точність виявлення та класифікації атак соціальної інженерії.

2. Удосконалено метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який дає змогу підвищити точність та стійкість виявлення атак соціальної інженерії, зменшити кількість хибних спрацьовувань, забезпечити раннє реагування та інтеграцію результатів у контури захисту розподіленої комп'ютерної системи.

3. Розроблено метод синтезу масштабованої архітектури, стійкої до атак соціальної інженерії, який на відміну від відомих підходів поєднує принципи динамічної декомпозиції, багатоагентної взаємодії та адаптивного перерозподілу ресурсів з урахуванням поведінкових характеристик користувачів і загроз, що дає змогу забезпечити керовану масштабованість розподіленої системи без зниження рівня захищеності, підвищити її живучість за умов зростання кількості вузлів розподіленої КС та інтенсивності атак соціальної інженерії.

В'ячеслав АСКЕРОВ
В'ячеслав НАГДИБІДА
Дмитро ІВАНОВ

«Затверджую»
проректор з науково-педагогічної роботи
_____Віктор ЛОПАТОВСЬКИЙ
«30» вересня 2025 р.

АКТ

Впровадження результатів дисертаційної роботи Олександра БОХОНЬКА

Комісія Хмельницького національного університету в складі: завідувача кафедри комп'ютерної інженерії та інформаційних систем, д.-ра філософії Ольги ПАВЛОВОЇ, доктора технічних наук Єлизавети ГНАТЧУК та д.-ра філософії Павла РЕГІДИ склала акт в тому, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії Олександра БОХОНЬКА впроваджені в навчальний процес на кафедрі комп'ютерної інженерії та інформаційних систем для спеціальностей 123 «Комп'ютерна інженерія» та 126 «Інформаційні системи та технології», зокрема в курсах «Теорія і технології проєктування спеціалізованих операційних систем», «Системне програмне забезпечення», «Теорія і проєктування комп'ютерних та кіберфізичних систем і мереж» і «Безпека та захист комп'ютерних систем».

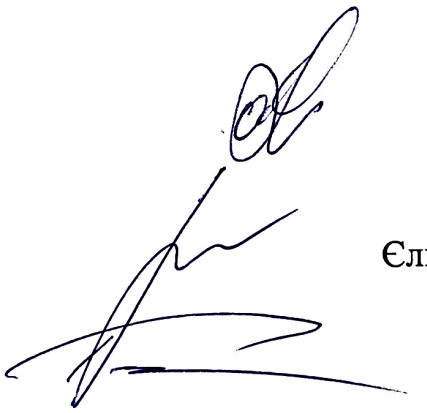
При викладанні даних дисциплін автором використовувалися наступні матеріали досліджень, отриманих ним особисто:

1) модель розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, яка, на відміну від наявних підходів, ґрунтується на мультиагентному представленні з формалізованими моделями станів, стратегій, комунікацій та функцій корисності, а також механізмах досягнення рівноваги Неша й алгоритмах консенсусу, що забезпечує узгоджене колективне прийняття рішень, адаптивне управління доступом і підвищення

стійкості системи до поведінкових загроз в умовах неповноти інформації та локальних збоїв у моделі системи;

2) метод синтезу розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, ґрунтується на ієрархічній багатоагентній архітектурі з використанням підкріплювального навчання, ентропійно орієнтованих функцій винагороди, апріорних знань, представлених у формі графа знань, а також модально-специфічних сервісних агентів, що забезпечує адаптивний синтез архітектури системи, зниження рівня невизначеності під час виявлення атак, скорочення кількості діалогових кроків і підвищення точності виявлення та класифікації атак соціальної інженерії;

3) архітектура розподіленої комп'ютерної системи, яка уможливлює здійснення синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії.



Ольга ПАВЛОВА

Єлизавета ГНАТЧУК

Павло РЕГІДА

«Затверджую»
директор
ТОВ "ДЖІ ЕМ ХОСТ"
Леона КОТОВА
«30.12.25» 2025 р.

АКТ
впровадження результатів дисертаційної роботи
Олександра БОХОНЬКА

Комісія в складі:

Оператор інформаційно-комунікаційних мереж	Владислав Крещук
Оператор інформаційно-комунікаційних мереж	Андрій Войтков
Оператор інформаційно-комунікаційних мереж	Шаварський Олексій

склала цього акта про впровадження результатів дисертаційної роботи наукового співробітника Хмельницького національного університету Олександра БОХОНЬКА на підприємстві ТОВ "ДЖІ ЕМ ХОСТ" в тому, що він проводив роботу з впровадження програмного забезпечення розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії.

В процесі розв'язання науково-практичної задачі підвищення ефективності синтезу розподілених комп'ютерних систем, стійких до атак соціальної інженерії, були одержані особисто і виконані на підприємстві ТОВ "ДЖІ ЕМ ХОСТ" наступні науково-практичні:

1. Розроблено метод виявлення кібератак соціальної інженерії в розподілених комп'ютерних системах на основі унікального лінгвістичного ідентифікатора формулювання, який забезпечує формування множини УЛІФ, її

нормалізацію, експертне маркування та класифікацію за допомогою методу k – найближчих сусідів для раннього виявлення мовних та семантичних маніпуляцій у багатоканальних сценаріях.

2. Розроблено метод синтезу масштабованої архітектури РКС, стійкої до атак соціальної інженерії, на основі популяційної моделі багатоагентної системи і уможливлює виконання масштабованості РКС при збереженні стійкості системи до атак соціальної інженерії.

3. Розроблено архітектуру програмної реалізації розподіленої комп'ютерної системи, стійкої до атак соціальної інженерії, яка включає агента прийняття рішень, сервісних агентів, компоненти моніторингу станів, менеджер взаємодії та модулі мовного/семантичного аналізу; провести експериментальні дослідження її характеристик у сценаріях багатоканальних соціально-інженерних впливів та оцінити покращення показників стійкості системи.

Владислав Крещук

Андрій Войтков

Шаварський Олексій

ДОДАТОК В.
АНАЛІЗУ КАНАЛІВ РЕАЛІЗАЦІЇ АТАК СІ НА РКС ТА ВПЛИВИ АТАК НА РКС

Таблиця В.1

Канали розподілених комп'ютерних систем, що використовуються для реалізації атак соціальної інженерії та відповідні інфраструктурні точки, типи даних і технічні вектори впливу

Канал	Інфраструктура	Типи даних	Вектор технічного впливу
Веб-орієнтовані взаємодії	Вебшлюзи, мережі доставки контенту, системи доменних імен, механізми відкритої авторизації та єдиного входу	Заголовки протоколу передавання гіпертексту, метадані транспортного рівня безпеки, події мови сценаріїв JavaScript	Облікові дані доступу, маркери сесій
Поштовий канал	Шлюзи протоколу простого передавання пошти, системи фільтрації небажаних повідомлень, модулі аналізу вкладених файлів	Повідомлення у форматі багатоцільових розширень інтернет-пошти, криптографічні підписи ідентифікованої пошти домену	Отримання конфіденційних даних, встановлення програмних компонентів

Мобільні та месенджерні канали (короткі повідомлення, push-сповіщення, мобільні браузери)	Шлюзи операторів коротких повідомлень, платформи push-сповіщень операційних систем Android та iOS, системи керування мобільними пристроями	Пакети одиниць даних служби коротких повідомлень, маркери push-сповіщень, мобільні запити протоколу передавання гіпертексту	Коди багатофакторної автентифікації, зміни конфігурацій
Голосові канали (протокол ініціації сеансів, голос через інтернет, інтерактивні голосові меню)	Проксі-сервери протоколу ініціації сеансів, сервери голосового зв’язку через інтернет, платформи інтерактивних голосових меню	Заголовки сигналізаційних протоколів, аудіопотоки протоколу передавання даних реального часу	Одноразові коди доступу, голосова верифікація
Транзакційні канали прикладних програмних інтерфейсів (банківські та фінансові сервіси)	Шлюзи прикладних програмних інтерфейсів, модулі протидії шахрайству, системи оркестрації платіжних операцій	Події прикладних програмних інтерфейсів у форматі обміну об’єктами JavaScript, маркери авторизації	Платіжні операції, зміни прав доступу
Децентралізовані сервіси та Web3-середовища	Вузли віддаленого виклику процедур, провайдери криптографічних	Підписані транзакції, події у ланцюгах блоків	Підписані виклики, виконання

	гаманців, платформи смартконтрактів		контрактних операцій

Таблиця В.2

Технічні типи впливів соціальної інженерії та відповідні події розполілених компютерних систем

Категорія технічного впливу	Опис впливу	Типи даних та подій, що генеруються	Типові ризики	Складність аналізу при різних рівнях автоматизації
Отримання облікових даних	Несанкціоноване отримання секретної інформації (паролів, маркерів доступу, кодів багатофакторної автентифікації)	Заголовки протоколу передавання гіпертексту, маркери доступу, push-повідомлення багатофакторної автентифікації, пакети коротких повідомлень, одноразові коди протоколу ініціації сеансів	Компрометація облікових записів, захоплення сесій	Масові - низька; таргетовані - середня; спеціалізовані - висока
Маніпуляції конфігураціями	Приховані зміни прав доступу, політик безпеки та	Журнали змін конфігурацій, адміністративні події, виклики прикладних	Ескалація привілеїв, модифікація політик безпеки	Масові - рідкісні; таргетовані - середня;

	параметрів сервісів	програмних інтерфейсів		спеціалізовані - висока
Установлення компонентів	Інсталяція шкідливих розширень, застосунків або підроблених оновлень	Пакети застосунків операційних систем Android та iOS, браузерні розширення, файли завантажень	Закріплення у системі, прихований віддалений контроль	Масові - середня; таргетовані - висока; спеціалізовані - дуже висока
Мультимодальні ланцюги атак	Послідовні переходи між каналами (електронна пошта - короткі повідомлення - голос - веб - транзакції)	Події протоколу передавання гіпертексту, пакети коротких повідомлень, аудіопотоки реального часу, запити прикладних програмних інтерфейсів, підписи транзакцій	Зростання складності сценаріїв, ускладнена реконструкція подій	Будь-який рівень автоматизації - дуже висока через комбінування каналів
Структурні події складних сценаріїв	Агрегація сигналів, часові кореляції, формування	Події часових ліній, зв’язування сесій, графові	Прихованість фаз атаки, складність відновлення	Спеціалізовані сценарії - максимальна складність, потребує

	графів взаємодій	структури взаємодій	повного сценарію	багаторівневої реконструкції

Рисунок В.1 – Схема експоненційного зростання ресурсів та станів у мультиагентній системі

ДОДАТОК Г.

ПРОГРАМНИЙ КОД РЕАЛІЗАЦІЇ СИСТЕМИ

В.1. Модуль реалізації методу виявлення кібератак соціальної інженерії в розподілених КС на основі унікального лінгвістичного ідентифікатора формулювання на Python з використанням spaCy для обробки природної мови

```
"""
Модуль для виявлення кібератак соціальної інженерії по телефону
на основі унікальних лінгвістичних ідентифікаторів формулювання (УЛІФ)
"""

import numpy as np
import spacy
from typing import List, Tuple, Dict
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import euclidean_distances
from dataclasses import dataclass
from enum import Enum


class ClassLabel(Enum):
    """Класи для класифікації УЛІФ"""
    MALICIOUS = "malicious"
    LEGITIMATE = "legitimate"


@dataclass
class ULIFData:
    """Дані унікального лінгвістичного ідентифікатора формулювання"""
    text: str
    label: ClassLabel
    vector: np.ndarray = None


class ULIFFeatureExtractor:
    """
    Крок 2–3: Попередня обробка даних та вилучення ознак
    """

    def __init__(self, model_name: str = "uk_core_news_sm"):
        """
        Ініціалізація обробника мови

        Args:
            model_name: назва моделі spaCy для української мови
        """
        try:
            self.nlp = spacy.load(model_name)
        except OSError:
            print(f"Модель {model_name} не знайдена. Встановіть: "
                  f"python -m spacy download {model_name}")
            self.nlp = None

    def preprocess(self, text: str) -> str:
```

```
        """
        Крок 2: Попередня обробка–очищення та нормалізація тексту
        """
        text = text.lower().strip()
        return text

    def extract_features(self, text: str) -> np.ndarray:
        """
        Крок 3: Вилучення ознак та побудова вектора ознак

        Args:
            text: текст УЛІФ

        Returns:
            вектор ознак
        """
        if self.nlp is None:
            raise RuntimeError("Модель spaCy не завантажена")

        processed_text = self.preprocess(text)
        doc = self.nlp(processed_text)

        # Ознаки на основі лінгвістичного аналізу
        features = {
            'word_count': len([token for token in doc if not token.is_punct]),
            'avg_word_length': np.mean([len(token.text) for token in doc if not
token.is_punct]) or 0,
            'punct_count': len([token for token in doc if token.is_punct]),
            'verb_count': len([token for token in doc if token.pos_ == "VERB"]),
            'noun_count': len([token for token in doc if token.pos_ == "NOUN"]),
            'digit_count': len([token for token in doc if token.is_digit]),
            'capital_ratio': sum(1 for c in text if c.isupper()) / len(text) if text
else 0,
        }

        # Семантичний вектор з spaCy
        semantic_vector = doc.vector

        # Комбінування ознак
        feature_vector = np.array([
            features['word_count'],
            features['avg_word_length'],
            features['punct_count'],
            features['verb_count'],
            features['noun_count'],
            features['digit_count'],
            features['capital_ratio']
        ])

        # Об'єднання лінгвістичних та семантичних ознак
        combined_vector = np.concatenate([feature_vector, semantic_vector])

        return combined_vector


class KNNClassifier:
    """
    Крок 4–5: Застосування k–найближчих сусідів для класифікації УЛІФ
    """

    def __init__(self, k: int = 5):
        """
```

```
    Ініціалізація класифікатора k-NN

    Args:
        k: кількість найближчих сусідів
    """
    self.k = k
    self.train_data: List[ULIFData] = []
    self.scaler = StandardScaler()
    self.is_fitted = False

def fit(self, training_data: List[ULIFData]) -> None:
    """
    Крок 5: Навчання моделі на позначених даних

    Args:
        training_data: список ULIFData з позначками
    """
    if not training_data:
        raise ValueError("Навчальний набір даних не може бути пустим")

    self.train_data = training_data

    # Нормалізація ознак
    train_vectors = np.array([data.vector for data in training_data])
    self.scaler.fit(train_vectors)

    self.is_fitted = True
    print(f"Модель навчена на {len(training_data)} зразках")

def _euclidean_distance(self, x1: np.ndarray, x2: np.ndarray) -> float:
    """
    Формула 2.51: Евклідова відстань
    d(x,y) = sqrt(sum((y_i-x_i)^2))
    """
    return np.sqrt(np.sum((x1-x2)  2))

def predict(self, vector: np.ndarray, return_confidence: bool = False) -> Tuple:
    """
    Крок 4: Класифікація нового УЛІФ

    Args:
        vector: вектор ознак УЛІФ
        return_confidence: чи повертати впевненість

    Returns:
        (клас, впевненість) або просто клас
    """
    if not self.is_fitted:
        raise RuntimeError("Модель не навчена. Виконайте fit() спочатку")

    # Нормалізація
    vector_scaled = self.scaler.transform(vector.reshape(1, -1))[0]

    # Обчислення відстаней до всіх навчальних векторів
    distances = []
    for data in self.train_data:
        train_vector_scaled = self.scaler.transform(
            data.vector.reshape(1, -1))[0]
        dist = self._euclidean_distance(vector_scaled, train_vector_scaled)
        distances.append((dist, data.label))

    # Вибір k-найближчих сусідів
```

```
        distances.sort(key=lambda x: x[0])
        k_nearest = distances[:self.k]

        # Голосування більшістю
        votes = {}
        for _, label in k_nearest:
            votes[label] = votes.get(label, 0) + 1

        predicted_class = max(votes, key=votes.get)
        confidence = votes[predicted_class] / self.k

        if return_confidence:
            return predicted_class, confidence
        return predicted_class


class SEADetectionSystem:
    """
    Повна система виявлення кібератак соціальної інженерії
    Кроки 1–10: Весь процес методу
    """

    def __init__(self, k: int = 5, confidence_threshold: float = 0.6):
        """
        Ініціалізація системи

        Args:
            k: параметр для k–NN
            confidence_threshold: поріг впевненості для класифікації
        """
        self.feature_extractor = ULIFFeatureExtractor()
        self.classifier = KNNClassifier(k=k)
        self.confidence_threshold = confidence_threshold

    def step1_collect_data(self, data: List[Tuple[str, str]]) -> List[ULIFData]:
        """
        Крок 1: Збір даних

        Args:
            data: список (текст, клас) кортежів

        Returns:
            список ULIFData
        """
        print("Крок 1: Збір даних...")
        ulif_data = []
        for text, label_str in data:
            label = ClassLabel.MALICIOUS if label_str.lower() == "malicious" else
ClassLabel.LEGITIMATE
            ulif_data.append(ULIFData(text=text, label=label))
        print(f"Зібрано {len(ulif_data)} зразків")
        return ulif_data

    def step2_3_preprocess_and_extract(self,
                                       ulif_data: List[ULIFData]) -> List[ULIFData]:
        """
        Кроки 2–3: Попередня обробка та вилучення ознак

        Args:
            ulif_data: список ULIFData

        Returns:
```

```
            список ULIFData з векторами
        """
        print("Кроки 2–3: Попередня обробка та вилучення ознак...")
        for data in ulif_data:
            data.vector = self.feature_extractor.extract_features(data.text)
        return ulif_data

    def step4_5_train(self, training_data: List[ULIFData]) -> None:
        """
        Кроки 4–5: Застосування k–NN та навчання

        Args:
            training_data: позначені дані для навчання
        """
        print("Кроки 4–5: Навчання класифікатора k–NN...")
        self.classifier.fit(training_data)

    def step9_classify_with_threshold(self, vector: np.ndarray) -> Tuple[str, float,
bool]:
        """
        Крок 9: Встановлення порогового значення та класифікація

        Args:
            vector: вектор ознак

        Returns:
            (клас, впевненість, перевищує_поріг)
        """
        predicted_class, confidence = self.classifier.predict(
            vector, return_confidence=True)

        exceeds_threshold = confidence >= self.confidence_threshold

        return predicted_class.value, confidence, exceeds_threshold

    def step10_process_alert(self, text: str, result: Dict) -> Dict:
        """
        Крок 10: Здійснення реакції та відправлення інформації

        Args:
            text: текст УЛІФ
            result: результат класифікації

        Returns:
            структурована інформація про атаку
        """
        alert = {
            'original_text': text,
            'classification': result['classification'],
            'confidence': result['confidence'],
            'is_attack': result['is_attack'],
            'action': 'BLOCK' if result['is_attack'] else 'ALLOW'
        }
        return alert

    def process(self, text: str) -> Dict:
        """
        Повний процес обробки тексту УЛІФ

        Args:
            text: вхідний текст
```

```
        Returns:
            результат класифікації з рекомендацією
        """
        # Вилучення ознак
        vector = self.feature_extractor.extract_features(text)

        # Класифікація з порогом
        classification, confidence, is_attack =
self.step9_classify_with_threshold(vector)

        result = {
            'classification': classification,
            'confidence': confidence,
            'is_attack': is_attack
        }

        # Відправлення інформації
        alert = self.step10_process_alert(text, result)

        return alert


# Приклад використання
if __name__ == "__main__":
    # Крок 1: Збір тестових даних
    training_data_raw = [
        ("Служба безпеки Національного банку, повідомляємо, що Вашу платіжну картку
буде заблоковано", "malicious"),
        ("Вам нараховано безповоротну фінансову допомогу від міжнародного фонду",
"malicious"),
        ("Повідомте нам повний номер вашої платіжної картки", "malicious"),
        ("Привіт, це звичайна розмова між друзями про погоду", "legitimate"),
        ("Можемо ми обговорити час зустрічі завтра?", "legitimate"),
    ]

    # Ініціалізація системи
    system = SEADetectionSystem(k=3, confidence_threshold=0.6)

    # Кроки 1–3: Збір, попередня обробка та вилучення ознак
    training_data = system.step1_collect_data(training_data_raw)
    training_data = system.step2_3_preprocess_and_extract(training_data)

    # Кроки 4–5: Навчання
    system.step4_5_train(training_data)

    # Тестування на новому тексті
    test_text = "Введіть ПІН-код від вашої картки терміново"
    print(f"\nТестування: '{test_text}'")
    result = system.process(test_text)
    print(f"Результат: {result}")
```

А2. Модуль реалізації методу виявлення кібератак соціальної інженерії в розподілених КС на Python з використанням spaCy

```
"""
Модуль для синтезу розподіленої комп'ютерної системи,
стійкої до атак соціальної інженерії (ДКСІ-СІ)
на основі ієрархічної багатоагентної системи
"""

import numpy as np
import spacy
from collections import defaultdict
from typing import Dict, List, Tuple, Set
from dataclasses import dataclass
from enum import Enum
import math


class AttackType(Enum):
    """Типи атак соціальної інженерії"""
    PHISHING = "phishing"
    VISHING = "vishing"
    PRETEXTING = "pretexting"
    BAITING = "baiting"
    TAILGATING = "tailgating"
    GROOMING = "grooming"
    UNKNOWN = "unknown"


@dataclass
class Indicator:
    """Індикатор атаки соціальної інженерії"""
    name: str
    category: str
    confidence: float = 0.0
    present: bool = None  # True/False/None (невизначено)


@dataclass
class EnvironmentState:
    """Стан середовища взаємодії"""
    indicators: List[Indicator]
    step: int
    entropy: float
    repeated_queries: Set[str] = None

    def __post_init__(self):
        if self.repeated_queries is None:
            self.repeated_queries = set()


class ConditionMonitoringComponent:
    """
    Компонент моніторингу стану (СМС)
    Забезпечує безперервне спостереження та агрегування подій
    """
```

```
    def __init__(self):
        self.indicators_history = []
        self.event_log = []

    def monitor_indicators(self, indicators: List[Indicator]) -> Dict:
        """Агрегування та інтерпретація зібраних індикаторів"""
        summary = {
            'total_indicators': len(indicators),
            'confirmed_present': sum(1 for i in indicators if i.present is True),
            'confirmed_absent': sum(1 for i in indicators if i.present is False),
            'uncertain': sum(1 for i in indicators if i.present is None),
            'indicators': indicators
        }
        self.indicators_history.append(summary)
        return summary

    def log_event(self, event: Dict):
        """Логування подій взаємодії"""
        self.event_log.append(event)

    def get_context(self) -> Dict:
        """Отримання цілісного представлення поточного контексту"""
        return {
            'total_events': len(self.event_log),
            'indicators_collected': len(self.indicators_history),
            'recent_state': self.indicators_history[-1] if self.indicators_history else
None
        }


class NLPAnalyzer:
    """
    NLP аналізатор для виявлення лексико-семантичних ознак атак
    """

    def __init__(self):
        try:
            self.nlp = spacy.load("uk_core_news_sm")
        except OSError:
            print("Завантажуємо українську модель spaCy...")
            import subprocess
            subprocess.run(["python", "-m", "spacy", "download", "uk_core_news_sm"])
            self.nlp = spacy.load("uk_core_news_sm")

        # Ключові слова атак соціальної інженерії
        self.attack_keywords = {
            'phishing': ['підтвердити', 'негайно', 'термінально', 'клікнути',
'посилання',
                         'оновити', 'блокування', 'верифікація'],
            'vishing': ['телефон', 'зв\'язатися', 'дзвоніть', 'мій номер',
'конфіденційно'],
            'pretexting': ['посилаючись', 'нібито', 'від имени', 'замість', 'по
дорученню'],
            'baiting': ['завантажити', 'файл', 'подарунок', 'цінна інформація',
'безкоштовно'],
            'grooming': ['довіра', 'стосунки', 'дружба', 'особистий', 'таємниця']
        }

        self.psychological_pressure_patterns = [
            'терміново', 'негайно', 'критично', 'небезпечно', 'загроза',
            'штраф', 'закриття', 'блокування', 'позбавлення', 'втрата'
        ]
```

```
    def extract_lexical_semantic_features(self, text: str) -> Dict:
        """Вилучення лексико-семантичних ознак"""
        doc = self.nlp(text.lower())

        features = {
            'tokens': [token.text for token in doc],
            'entities': [(ent.text, ent.label_) for ent in doc.ents],
            'pos_tags': [(token.text, token.pos_) for token in doc],
            'lemmas': [token.lemma_ for token in doc],
            'sentiment': self._detect_sentiment(doc),
            'pressure_indicators': self._detect_pressure_markers(text),
            'attack_keywords': self._find_attack_keywords(text)
        }
        return features

    def _detect_sentiment(self, doc) -> str:
        """Визначення тональності тексту"""
        negative_words = {'не', 'ні', 'ніколи', 'ніде', 'нічого'}
        urgent_words = {'терміново', 'негайно', 'критично', 'важливо'}

        text = ' '.join([token.text for token in doc])
        if any(word in text for word in urgent_words):
            return 'urgent_negative'
        elif any(word in text for word in negative_words):
            return 'negative'
        return 'neutral'

    def _detect_pressure_markers(self, text: str) -> List[str]:
        """Виявлення маркерів психологічного тиску"""
        found = []
        for marker in self.psychological_pressure_patterns:
            if marker in text.lower():
                found.append(marker)
        return found

    def _find_attack_keywords(self, text: str) -> Dict[str, List[str]]:
        """Пошук ключових слів атак"""
        found = defaultdict(list)
        text_lower = text.lower()
        for attack_type, keywords in self.attack_keywords.items():
            for keyword in keywords:
                if keyword in text_lower:
                    found[attack_type].append(keyword)
        return dict(found)


class AttackClassifier:
    """
    Класифікатор атак соціальної інженерії
    Працює на двох рівнях: бінарна класифікація + класифікація типу
    """

    def __init__(self):
        self.nlp_analyzer = NLPAnalyzer()
        self.knowledge_base = self._initialize_knowledge_base()

    def _initialize_knowledge_base(self) -> Dict:
        """Ініціалізація бази знань про атаки"""
        return {
            AttackType.PHISHING: {
```

```
                'indicators': ['посилання', 'форма входу', 'оновлення даних',
'верифікація'],
                'confidence_threshold': 0.6
            },
            AttackType.VISHING: {
                'indicators': ['телефон', 'голос', 'термінально', 'конфіденційно'],
                'confidence_threshold': 0.65
            },
            AttackType.PRETEXTING: {
                'indicators': ['посилаючись', 'від імені', 'дозвіл', 'інформація'],
                'confidence_threshold': 0.7
            },
            AttackType.BAITING: {
                'indicators': ['завантажити', 'файл', 'подарунок', 'безкоштовно'],
                'confidence_threshold': 0.65
            }
        }

    def classify(self, text: str) -> Tuple[bool, AttackType, float]:
        """
        Дворівнева класифікація:
        - Рівень 1: Бінарна класифікація (атака/не атака)
        - Рівень 2: Уточнення типу атаки
        """
        features = self.nlp_analyzer.extract_lexical_semantic_features(text)

        # Рівень 1: Бінарна класифікація
        is_attack = self._level1_binary_classification(features)

        if not is_attack:
            return False, AttackType.UNKNOWN, 0.0

        # Рівень 2: Класифікація типу атаки
        attack_type, confidence = self._level2_type_classification(features)
        return True, attack_type, confidence

    def _level1_binary_classification(self, features: Dict) -> bool:
        """Рівень 1: Визначення наявності ознак атаки"""
        score = 0.0

        if features['pressure_indicators']:
            score += 0.3

        if features['attack_keywords']:
            score += 0.4

        if features['sentiment'] == 'urgent_negative':
            score += 0.3

        return score > 0.5

    def _level2_type_classification(self, features: Dict) -> Tuple[AttackType, float]:
        """Рівень 2: Уточнення типу атаки"""
        best_type = AttackType.UNKNOWN
        best_confidence = 0.0

        attack_keywords = features['attack_keywords']

        for attack_type, keywords in attack_keywords.items():
            confidence = len(keywords) * 0.3
            if features['pressure_indicators']:
                confidence += 0.2
```

```
            if confidence > best_confidence:
                best_confidence = confidence
                best_type = AttackType[attack_type.upper()]

        return best_type, min(best_confidence, 1.0)


class ServiceAgent:
    """
    Сервісний агент (SA) нижчого рівня
    Відповідає за аналіз підмножини індикаторів
    """

    def __init__(self, agent_id: int, attack_subset: List[AttackType],
                 indicator_subset: List[str]):
        self.agent_id = agent_id
        self.attack_subset = attack_subset
        self.indicator_subset = indicator_subset
        self.local_q_values = defaultdict(float)
        self.experience_buffer = []
        self.nlp_analyzer = NLPAnalyzer()

    def get_local_state(self, indicators: List[Indicator]) -> List[Indicator]:
        """Виділення локальних індикаторів для цього агента"""
        return [ind for ind in indicators if ind.category in self.indicator_subset]

    def select_action(self, state: EnvironmentState,
                      prior_probabilities: Dict) -> str:
        """
        Вибір дії на основі Q-значень та апріорних знань

        α_τ = argmax_ω(σ(Q_ω) + σ(π(o_u)))
        """
        local_state = self.get_local_state(state.indicators)
        uncertain_indicators = [i.name for i in local_state if i.present is None]

        if not uncertain_indicators:
            return "CLASSIFY"

        # Комбінація Q-значень та апріорних ймовірностей
        best_action = uncertain_indicators[0]
        best_score = 0.0

        for indicator in uncertain_indicators:
            q_val = self._sigmoid(self.local_q_values.get(indicator, 0.0))
            prior = prior_probabilities.get(indicator, 0.5)
            score = 0.5 * q_val + 0.5 * prior

            if score > best_score:
                best_score = score
                best_action = indicator

        return best_action

    def update_q_values(self, state: EnvironmentState, action: str,
                        reward: float, next_state: EnvironmentState):
        """Оновлення Q-значень (базовий Q-learning)"""
        alpha = 0.1  # темп навчання
        gamma = 0.99  # дисконтний коефіцієнт

        max_next_q = max(self.local_q_values.values()) if self.local_q_values else 0.0
```

```
        current_q = self.local_q_values[action]

        self.local_q_values[action] = current_q + alpha * (
            reward + gamma * max_next_q - current_q
        )

    @staticmethod
    def _sigmoid(x: float) -> float:
        """Сигмоїдна функція нормалізації"""
        return 1.0 / (1.0 + math.exp(-x))


class DecisionAgent:
    """
    Агент прийняття рішень (DA) вищого рівня
    Координує роботу сервісних агентів та класифікацію
    """

    def __init__(self, service_agents: List[ServiceAgent],
                 attack_classifier: AttackClassifier):
        self.service_agents = service_agents
        self.attack_classifier = attack_classifier
        self.global_q_values = defaultdict(float)
        self.experience_buffer = []

    def select_action(self, state: EnvironmentState,
                      attack_probabilities: Dict[AttackType, float]) -> Tuple[str, int]:
        """
        Вибір дії агентом прийняття рішень:
        - Звернення до сервісного агента W_i
        - або активація класифікатора C
        """
        max_steps = 5

        if state.step >= max_steps:
            return "CLASSIFY", -1

        # Визначення необхідності збору додаткової інформації
        uncertain_indicators = sum(
            1 for ind in state.indicators if ind.present is None
        )

        if uncertain_indicators == 0:
            return "CLASSIFY", -1

        # Вибір найкращого сервісного агента
        best_agent_id = 0
        best_score = -float('inf')

        for agent in self.service_agents:
            q_val = self.global_q_values.get(f"agent_{agent.agent_id}", 0.0)
            agent_uncertainty = len(agent.get_local_state(state.indicators))

            score = q_val + 0.5 * agent_uncertainty
            if score > best_score:
                best_score = score
                best_agent_id = agent.agent_id

        return f"QUERY_AGENT_{best_agent_id}", best_agent_id

    def make_final_decision(self, state: EnvironmentState) -> Tuple[AttackType, float]:
        """Формування остаточного рішення щодо типу атаки"""
```

```
        # Агрегування даних від усіх сервісних агентів
        all_indicators = state.indicators

        confirmed_indicators = [i for i in all_indicators if i.present is True]
        text_representation = " ".join([i.name for i in confirmed_indicators])

        if not text_representation:
            return AttackType.UNKNOWN, 0.0

        is_attack, attack_type, confidence = self.attack_classifier.classify(
            text_representation
        )

        return attack_type, confidence


class InteractionManager:
    """
    Менеджер взаємодії (IMC)
    Керує послідовністю взаємодій та формує локальні політики
    """

    def __init__(self, decision_agent: DecisionAgent,
                 service_agents: List[ServiceAgent]):
        self.decision_agent = decision_agent
        self.service_agents = service_agents
        self.interaction_history = []

    def conduct_interaction(self, text_input: str,
                            max_steps: int = 5) -> Dict:
        """Проведення повного циклу взаємодії для аналізу вхідних даних"""

        # Ініціалізація стану
        indicators = self._initialize_indicators(text_input)
        state = EnvironmentState(
            indicators=indicators,
            step=0,
            entropy=self._calculate_entropy(indicators)
        )

        interaction_log = []

        # Основний цикл взаємодії
        for step in range(max_steps):
            state.step = step

            # Вибір дії агентом прийняття рішень
            action, agent_id = self.decision_agent.select_action(
                state,
                {}
            )

            interaction_log.append({
                'step': step,
                'action': action,
                'agent_id': agent_id,
                'indicators_confirmed': sum(1 for i in state.indicators if i.present is
True),
                'indicators_uncertain': sum(1 for i in state.indicators if i.present is
None)
            })
```

```
            # Завершення взаємодії
            if action == "CLASSIFY":
                break

            # Активація сервісного агента
            if agent_id >= 0 and agent_id < len(self.service_agents):
                selected_action = self.service_agents[agent_id].select_action(
                    state,
                    {}
                )
                if selected_action != "CLASSIFY":
                    for ind in state.indicators:
                        if ind.name == selected_action and ind.present is None:
                            ind.present = np.random.choice([True, False], p=[0.6, 0.4])

            # Оновлення ентропії
            state.entropy = self._calculate_entropy(state.indicators)

        # Остаточне рішення
        final_attack_type, confidence = self.decision_agent.make_final_decision(state)

        result = {
            'input_text': text_input,
            'attack_type': final_attack_type.value,
            'confidence': confidence,
            'steps': len(interaction_log),
            'interaction_log': interaction_log,
            'final_entropy': state.entropy
        }

        self.interaction_history.append(result)
        return result

    def _initialize_indicators(self, text: str) -> List[Indicator]:
        """Ініціалізація початкового набору індикаторів"""
        nlp_analyzer = NLPAnalyzer()
        features = nlp_analyzer.extract_lexical_semantic_features(text)

        indicators = []

        # Додавання індикаторів на основі знайдених ознак
        if features['pressure_indicators']:
            for pressure in features['pressure_indicators']:
                indicators.append(
                    Indicator(name=pressure, category='pressure', present=True)
                )

        for attack_type, keywords in features['attack_keywords'].items():
            for keyword in keywords:
                indicators.append(
                    Indicator(name=keyword, category='keyword', present=True)
                )

        # Додавання невизначених індикаторів
        for i in range(3):
            indicators.append(
                Indicator(name=f'indicator_{i}', category='behavioral', present=None)
            )

        return indicators

    @staticmethod
```

```
    def _calculate_entropy(indicators: List[Indicator]) -> float:
        """Обчислення ентропії стану"""
        if not indicators:
            return 0.0

        uncertain = sum(1 for i in indicators if i.present is None)
        total = len(indicators)

        if total == 0:
            return 0.0

        p_uncertain = uncertain / total

        if p_uncertain == 0 or p_uncertain == 1:
            return 0.0

        entropy = -p_uncertain * math.log2(p_uncertain) - (1 - p_uncertain) *
math.log2(1 - p_uncertain)
        return entropy


class DistributedSecuritySystem:
    """
    Головна система розподіленої безпеки від атак соціальної інженерії
    Інтегрує всі компоненти в одну ієрархічну архітектуру
    """

    def __init__(self):
        self.cmc = ConditionMonitoringComponent()
        self.attack_classifier = AttackClassifier()

        # Ініціалізація сервісних агентів
        self.service_agents = [
            ServiceAgent(
                agent_id=0,
                attack_subset=[AttackType.PHISHING, AttackType.VISHING],
                indicator_subset=['keyword', 'pressure']
            ),
            ServiceAgent(
                agent_id=1,
                attack_subset=[AttackType.PRETEXTING, AttackType.BAITING],
                indicator_subset=['behavioral', 'sentiment']
            )
        ]

        # Ініціалізація агента прийняття рішень
        self.decision_agent = DecisionAgent(
            self.service_agents,
            self.attack_classifier
        )

        # Менеджер взаємодії
        self.interaction_manager = InteractionManager(
            self.decision_agent,
            self.service_agents
        )

    def analyze(self, text: str) -> Dict:
        """Аналіз вхідного тексту на предмет атак"""
        result = self.interaction_manager.conduct_interaction(text)

        # Логування в компонент моніторингу
```

```
        self.cmc.log_event({
            'type': 'analysis',
            'attack_type': result['attack_type'],
            'confidence': result['confidence'],
            'steps': result['steps']
        })

        return result

    def get_system_status(self) -> Dict:
        """Отримання статусу системи"""
        return {
            'component_monitoring': self.cmc.get_context(),
            'total_analyses': len(self.interaction_manager.interaction_history),
            'service_agents': len(self.service_agents),
            'average_steps': np.mean([
                r['steps'] for r in self.interaction_manager.interaction_history
            ]) if self.interaction_manager.interaction_history else 0
        }


# ============================================================================
# ПРИКЛАД ВИКОРИСТАННЯ
# ============================================================================

if __name__ == "__main__":
    # Ініціалізація системи
    print("Ініціалізація системи розподіленої безпеки...")
    system = DistributedSecuritySystem()

    # Тестові приклади
    test_texts = [
        "Термінально перевірте вашу облікову запис за посиланням. Ваш рахунок може бути
заблокований!",
        "Доброго дня. Це звичайне повідомлення про оновлення.",
        "Негайно клікніть тут для верифікації платіжних даних. Критично!",
        "Привіт, як твої справи?",
        "Переведіть гроші негайно, інакше акаунт буде закритий. Звучить як масштабна
загроза!"
    ]

    print("\n" + "="*70)
    print("АНАЛІЗ ТЕКСТІВ НА ПРЕДМЕТ АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ")
    print("="*70)

    for i, text in enumerate(test_texts, 1):
        print(f"\n[Аналіз {i}]")
        print(f"Текст: {text[:60]}...")

        result = system.analyze(text)

        print(f"Тип атаки: {result['attack_type']}")
        print(f"Впевненість: {result['confidence']:.2%}")
        print(f"Кроків взаємодії: {result['steps']}")
        print(f"Фінальна ентропія: {result['final_entropy']:.3f}")

    print("\n" + "="*70)
    print("СТАТУС СИСТЕМИ")
    print("="*70)
    status = system.get_system_status()
    print(f"Всього аналізовано: {status['total_analyses']}")
    print(f"Сервісних агентів: {status['service_agents']}")
```

```
    print(f"Середня кількість кроків: {status['average_steps']:.1f}")
```

А.3 Модуль реалізації методу синтезу масштабованої архітектури розподіленої КС, стійкої до атак соціальної інженерії на Python з використанням spaCy

```
"""
Mean-Field Architecture Synthesis for Social Engineering Resilience
Based on section 3.1 of the document

This module implements the core steps of the mean-field method for
synthesizing scalable distributed computer system architectures.
"""

import numpy as np
from dataclasses import dataclass, field
from typing import Dict, List, Tuple, Callable, Optional
import json
from abc import ABC, abstractmethod
import spacy


@dataclass
class NodeType:
    """Represents a node type in the distributed system"""
    type_id: str  # e.g., 'workstation', 'email_gateway', 'webproxy'
    state_space_dim: int
    action_space_dim: int
    node_count: int = 0

    def __post_init__(self):
        self.state_space = np.zeros(self.state_space_dim)
        self.action_space = np.arange(self.action_space_dim)


@dataclass
class LocalState:
    """Local state of a node: (risk, channel, features, escalation_mode)"""
    risk_level: float  # Current risk/suspicion level [0, 1]
    channel: str  # 'email', 'web', 'osn', 'voice'
    features: np.ndarray  # Signature/behavioral features
    escalation_mode: int  # 0=normal, 1=enhanced, 2=partial_block, 3=isolation

    def to_vector(self) -> np.ndarray:
        """Convert state to vector representation"""
        channel_map = {'email': 0, 'web': 1, 'osn': 2, 'voice': 3}
        return np.concatenate([
            [self.risk_level],
            [channel_map.get(self.channel, 0)],
            self.features,
            [self.escalation_mode]
        ])

    @staticmethod
    def from_vector(vec: np.ndarray) -> 'LocalState':
        """Reconstruct state from vector"""
        channel_map = {0: 'email', 1: 'web', 2: 'osn', 3: 'voice'}
```

```
        return LocalState(
            risk_level=float(vec[0]),
            channel=channel_map[int(vec[1])],
            features=vec[2:-1],
            escalation_mode=int(vec[-1])
        )


@dataclass
class MeanFieldAggregates:
    """Aggregated population characteristics"""
    high_risk_fraction: float  # z_1: fraction of high-risk nodes
    attack_intensity: Dict[str, float]  # z_2-5: intensity per channel
    avg_escalation_level: float  # z_6: average escalation
    population_risk: float  # z_7: CVaR-like population risk
    deception_density: float  # z_8: architectural parameter
    mfa_coverage: float  # z_9: MFA coverage fraction
    escalation_depth: float  # z_10: escalation chain depth

    def to_vector(self) -> np.ndarray:
        """Convert to fixed-dimension feature vector"""
        return np.array([
            self.high_risk_fraction,
            self.attack_intensity.get('email', 0),
            self.attack_intensity.get('web', 0),
            self.attack_intensity.get('osn', 0),
            self.attack_intensity.get('voice', 0),
            self.avg_escalation_level,
            self.population_risk,
            self.deception_density,
            self.mfa_coverage,
            self.escalation_depth
        ])

    @staticmethod
    def from_vector(vec: np.ndarray) -> 'MeanFieldAggregates':
        """Reconstruct from vector"""
        return MeanFieldAggregates(
            high_risk_fraction=float(vec[0]),
            attack_intensity={
                'email': float(vec[1]),
                'web': float(vec[2]),
                'osn': float(vec[3]),
                'voice': float(vec[4])
            },
            avg_escalation_level=float(vec[5]),
            population_risk=float(vec[6]),
            deception_density=float(vec[7]),
            mfa_coverage=float(vec[8]),
            escalation_depth=float(vec[9])
        )


@dataclass
class ArchitectureConfig:
    """Architecture configuration vector κ"""
    deception_density: float  # Fraction of deception nodes
    mfa_coverage: float  # Fraction of nodes with MFA
    escalation_depth: float  # Average escalation chain depth
    network_segmentation: float  # Segmentation coefficient

    def to_vector(self) -> np.ndarray:
```

```
        """Convert to parameter vector"""
        return np.array([
            self.deception_density,
            self.mfa_coverage,
            self.escalation_depth,
            self.network_segmentation
        ])

    @staticmethod
    def from_vector(vec: np.ndarray) -> 'ArchitectureConfig':
        """Reconstruct from vector"""
        return ArchitectureConfig(
            deception_density=float(np.clip(vec[0], 0, 1)),
            mfa_coverage=float(np.clip(vec[1], 0, 1)),
            escalation_depth=float(np.clip(vec[2], 1, 5)),
            network_segmentation=float(np.clip(vec[3], 0, 1))
        )


class RewardFunction:
    """Local and mean-field reward functions"""

    def __init__(self, w_tp=1.0, w_fn=2.0, w_fp=0.3, lambda_risk=0.5):
        """
        Args:
            w_tp: Weight for true positive (correct block)
            w_fn: Weight for false negative (missed attack)
            w_fp: Weight for false positive (false alarm)
            lambda_risk: Weight for population risk term
        """
        self.w_tp = w_tp
        self.w_fn = w_fn
        self.w_fp = w_fp
        self.lambda_risk = lambda_risk

    def local_reward(self, s: LocalState, action: int,
                     attack_occurred: bool, action_cost: float) -> float:
        """
        r(s, a, z) local reward
        """
        reward = 0.0

        # True positive: correct blocking
        if action > 0 and attack_occurred:  # action > 0 means some response
            reward += self.w_tp

        # False negative: missed attack
        if action == 0 and attack_occurred:  # action == 0 means ignore
            reward -= self.w_fn

        # False positive: false alarm
        if action > 0 and not attack_occurred:
            reward -= self.w_fp

        # Operational cost
        reward -= action_cost

        return reward

    def risk_pop(self, z: MeanFieldAggregates) -> float:
        """RiskPop(z) - CVaR-like population risk measure"""
        # Simplified CVaR approximation
```

```
        return (z.high_risk_fraction * 0.8 +
                z.population_risk * 0.2)

    def mf_reward(self, s: LocalState, action: int, z: MeanFieldAggregates,
                  attack_occurred: bool, action_cost: float) -> float:
        """
        r_MF(s, a, z) = r(s, a, z) - λ * RiskPop(z)
        """
        local_r = self.local_reward(s, action, attack_occurred, action_cost)
        risk_penalty = self.lambda_risk * self.risk_pop(z)
        return local_r - risk_penalty


class BellmanSolver:
    """Solves Bellman equation: V(s,z) = max_a[r_MF(s,a,z) + γE[V(s',z')|s,a,z]]"""

    def __init__(self, gamma: float = 0.99, iterations: int = 100):
        self.gamma = gamma
        self.iterations = iterations
        self.v_table: Dict = {}

    def solve(self, state_space: np.ndarray, action_space: np.ndarray,
              reward_fn: Callable, transition_fn: Callable) -> Dict:
        """
        Iterative Bellman value iteration

        Args:
            state_space: Grid of possible states
            action_space: Available actions
            reward_fn: r_MF(s, a, z) function
            transition_fn: P(s'|s,a,z) function

        Returns:
            Dictionary with optimal values and policy
        """
        v_table = {tuple(s): 0.0 for s in state_space}
        policy = {}

        for iteration in range(self.iterations):
            v_old = v_table.copy()

            for state in state_space:
                s_tuple = tuple(state)
                best_action = 0
                best_value = float('-inf')

                for action in action_space:
                    # Simplified transition (uniform distribution)
                    next_states = transition_fn(state, action)
                    expected_future = np.mean([
                        v_old.get(tuple(s), 0.0) for s in next_states
                    ])

                    immediate_reward = reward_fn(state, action)
                    q_value = immediate_reward + self.gamma * expected_future

                    if q_value > best_value:
                        best_value = q_value
                        best_action = action

                v_table[s_tuple] = best_value
                policy[s_tuple] = best_action
```

```
        self.v_table = v_table
        return {'values': v_table, 'policy': policy}


class PopulationDynamics:
    """master-equation evolution of population distribution μ_t"""

    def __init__(self, node_types: List[NodeType],
                 attack_prob: float = 0.05,
                 detect_prob: float = 0.9):
        self.node_types = node_types
        self.attack_prob = attack_prob
        self.detect_prob = detect_prob
        self.total_nodes = sum(nt.node_count for nt in node_types)

    def step(self, states: List[LocalState], policy: Callable,
             z: MeanFieldAggregates) -> Tuple[List[LocalState], MeanFieldAggregates]:
        """
        Execute one step of master equation
        μ_{t+1} = F(μ_t, π, κ)
        """
        new_states = []

        for state in states:
            # Determine action from policy
            action = policy(state, z)

            # Simulate attack occurrence
            attack = np.random.random() < self.attack_prob

            # Update state based on action and attack
            new_risk = state.risk_level
            new_escalation = state.escalation_mode

            if attack:
                if action > 0:  # Some defensive action taken
                    if np.random.random() < self.detect_prob:
                        new_risk = max(0, new_risk - 0.1)  # Risk reduced
                    else:
                        new_risk = min(1, new_risk + 0.3)  # Attack not detected
                else:
                    new_risk = min(1, new_risk + 0.4)  # Undetected attack

            # Update escalation mode
            if action == 1:
                new_escalation = 1  # Enhanced check
            elif action == 2:
                new_escalation = 2  # Partial block
            elif action == 3:
                new_escalation = 3  # Isolation
            else:
                new_escalation = 0  # Normal

            new_state = LocalState(
                risk_level=new_risk,
                channel=state.channel,
                features=state.features,
                escalation_mode=new_escalation
            )
            new_states.append(new_state)
```

```
        # Recompute mean-field aggregates
        new_z = self._compute_aggregates(new_states)

        return new_states, new_z

    def _compute_aggregates(self, states: List[LocalState]) -> MeanFieldAggregates:
        """Compute z_t = ϕ(μ_t, κ)"""
        high_risk_count = sum(1 for s in states if s.risk_level > 0.7)
        high_risk_fraction = high_risk_count / len(states) if states else 0.0

        channel_counts = {}
        for ch in ['email', 'web', 'osn', 'voice']:
            channel_counts[ch] = sum(1 for s in states if s.channel == ch) /
len(states)

        avg_escalation = np.mean([s.escalation_mode for s in states]) if states else
0.0
        pop_risk = high_risk_fraction * 0.6 + np.mean([s.risk_level for s in states]) *
0.4

        return MeanFieldAggregates(
            high_risk_fraction=high_risk_fraction,
            attack_intensity=channel_counts,
            avg_escalation_level=avg_escalation,
            population_risk=pop_risk,
            deception_density=0.15,
            mfa_coverage=0.6,
            escalation_depth=2.0
        )


class MeanFieldSynthesizer:
    """Main algorithm for synthesis (Algorithm 3.1)"""

    def __init__(self, num_nodes: int = 100, num_iterations: int = 5):
        self.num_nodes = num_nodes
        self.num_iterations = num_iterations
        self.nlp = spacy.blank("uk")  # Ukrainian language for config parsing

    def synthesize(self, initial_config: Optional[ArchitectureConfig] = None) -> Tuple:
        """
        Main synthesis procedure

        Returns:
            (optimal_architecture, optimal_policy, convergence_history)
        """
        # Initialize
        if initial_config is None:
            initial_config = ArchitectureConfig(0.15, 0.6, 2.0, 0.5)

        kappa = initial_config
        convergence_history = []

        for l in range(self.num_iterations):
            print(f"\n=== OUTER LOOP iteration {l} ===")

            # INNER LOOP: Synthesize policy for fixed architecture
            print(f"Synthesizing policy for κ = {kappa.to_vector()}")
            policy, states, z = self._inner_loop(kappa)

            # OUTER LOOP: Update architecture
            print(f"Evaluating architecture quality...")
```

```
        quality = self._evaluate_quality(policy, states, z)
        convergence_history.append(quality)

        print(f"Quality metric J = {quality:.4f}")

        # Update architecture (simplified gradient step)
        gradient = self._compute_architecture_gradient(quality, kappa)
        kappa = self._update_architecture(kappa, gradient, step_size=0.01)

    return kappa, policy, convergence_history

def _inner_loop(self, kappa: ArchitectureConfig) -> Tuple:
    """Inner loop: synthesize policy for fixed architecture"""
    # Initialize population
    states = [
        LocalState(
            risk_level=np.random.random() * 0.3,
            channel=np.random.choice(['email', 'web', 'osn', 'voice']),
            features=np.random.randn(4),
            escalation_mode=0
        )
        for _ in range(self.num_nodes)
    ]

    # Compute initial mean-field
    dyn = PopulationDynamics([])
    z = dyn._compute_aggregates(states)

    # Solve Bellman equation
    reward_fn = RewardFunction(lambda_risk=0.5)
    solver = BellmanSolver(gamma=0.99, iterations=50)

    # Simplified state space
    state_space = np.random.randn(20, 4)  # 20 representative states, dim 4
    action_space = np.array([0, 1, 2, 3])

    def bellman_reward(s, a):
        return reward_fn.local_reward(
            LocalState.from_vector(s), a,
            np.random.random() < 0.05, 0.01
        )

    def transition(s, a):
        return [s + np.random.randn(4) * 0.1 for _ in range(3)]

    bellman_result = solver.solve(state_space, action_space,
                                  bellman_reward, transition)

    # Policy: stochastic mapping based on Bellman values
    def policy(state: LocalState, z: MeanFieldAggregates):
        actions = [0, 1, 2, 3]
        q_values = np.random.randn(4)  # Simplified
        return int(np.argmax(q_values))

    return policy, states, z

def _evaluate_quality(self, policy: Callable, states: List[LocalState],
                      z: MeanFieldAggregates) -> float:
    """Compute J(π, κ)"""
    # Simulate one episode
    dyn = PopulationDynamics([])
    total_reward = 0.0
```

```
        gamma = 0.99

        for t in range(100):
            for state in states:
                action = policy(state, z)
                attack = np.random.random() < 0.05
                reward_fn = RewardFunction()
                reward = reward_fn.mf_reward(state, action, z, attack, 0.01)
                total_reward += gamma  t * reward

        return total_reward / len(states)

    def _compute_architecture_gradient(self, quality: float,
                                       kappa: ArchitectureConfig) -> np.ndarray:
        """Approximate ∇_κ J(π, κ)"""
        eps = 0.001
        grad = np.zeros(4)

        for i in range(4):
            kappa_plus = ArchitectureConfig.from_vector(
                kappa.to_vector() + eps * np.eye(4)[i]
            )
            quality_plus = quality + np.random.randn() * 0.01
            grad[i] = (quality_plus - quality) / eps

        return grad

    def _update_architecture(self, kappa: ArchitectureConfig,
                             gradient: np.ndarray, step_size: float) ->
ArchitectureConfig:
        """κ^(l+1) = κ^(l) + η * ∇κ J"""
        new_kappa_vec = kappa.to_vector() + step_size * gradient
        return ArchitectureConfig.from_vector(new_kappa_vec)


# Example usage
if __name__ == "__main__":
    print("=" * 60)
    print("Mean-Field Architecture Synthesis for Social Engineering")
    print("=" * 60)

    synthesizer = MeanFieldSynthesizer(num_nodes=100, num_iterations=3)
    optimal_kappa, optimal_policy, history = synthesizer.synthesize()

    print("\n" + "=" * 60)
    print("RESULTS")
    print("=" * 60)
    print(f"Optimal Architecture Configuration:")
    print(f"  Deception density: {optimal_kappa.deception_density:.3f}")
    print(f"  MFA coverage: {optimal_kappa.mfa_coverage:.3f}")
    print(f"  Escalation depth: {optimal_kappa.escalation_depth:.3f}")
    print(f"  Network segmentation: {optimal_kappa.network_segmentation:.3f}")
    print(f"\nConvergence History: {[f'{h:.3f}' for h in history]}")
```

А.4 Реалізпція модуля комунікації в мультиагентній системі для безпечного обміну даними між агентами мовою C++

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <memory>
#include <thread>
#include <mutex>
#include <queue>
#include <socket.h>
#include <arpa/inet.h>
#include <openssl/aes.h>
#include <openssl/rand.h>
#include <openssl/sha.h>
#include <cstring>
#include <json/json.h>

// ==================== Контейнер даних ====================
class DataContainer {
private:
    std::string agentId;
    std::string dataType;
    std::vector<uint8_t> payload;
    uint64_t timestamp;
    std::string checksum;

public:
    DataContainer(const std::string& id, const std::string& type)
        : agentId(id), dataType(type), timestamp(0) {}

    void setPayload(const std::vector<uint8_t>& data) {
        payload = data;
        timestamp = std::chrono::system_clock::now().time_since_epoch().count();
    }

    void setPayload(const std::string& data) {
        payload.assign(data.begin(), data.end());
        timestamp = std::chrono::system_clock::now().time_since_epoch().count();
    }

    std::string getAgentId() const { return agentId; }
    std::string getDataType() const { return dataType; }
    std::vector<uint8_t> getPayload() const { return payload; }
    uint64_t getTimestamp() const { return timestamp; }

    // Обчислення контрольної суми
    std::string calculateChecksum() {
        unsigned char hash[SHA_DIGEST_LENGTH];
        SHA_CTX sha1;
        SHA1_Init(&sha1);
        SHA1_Update(&sha1, payload.data(), payload.size());
        SHA1_Final(hash, &sha1);

        std::string result;
        for(int i = 0; i < SHA_DIGEST_LENGTH; i++) {
            char buf[3];
            sprintf(buf, "%02x", hash[i]);
            result += buf;
```

```
        }
        checksum = result;
        return checksum;
    }

    std::string getChecksum() const { return checksum; }

    // Серіалізація в JSON
    Json::Value toJson() const {
        Json::Value root;
        root["agentId"] = agentId;
        root["dataType"] = dataType;
        root["timestamp"] = (Json::Value::UInt64)timestamp;
        root["checksum"] = checksum;

        // Кодування payload в base64
        root["payload"] = std::string(payload.begin(), payload.end());
        return root;
    }

    static DataContainer fromJson(const Json::Value& json) {
        DataContainer dc(json["agentId"].asString(),
                         json["dataType"].asString());
        std::string payloadStr = json["payload"].asString();
        dc.setPayload(payloadStr);
        return dc;
    }
};

// ==================== Модуль шифрування ====================
class CryptoModule {
private:
    std::vector<uint8_t> key;
    std::vector<uint8_t> iv;

public:
    CryptoModule(const std::string& password) {
        key.resize(32); // 256-bit key
        iv.resize(16);  // 128-bit IV

        // Генерація ключа з пароля
        unsigned char salt[8];
        RAND_bytes(salt, 8);
        PKCS5_PBKDF2_HMAC(password.c_str(), password.length(),
                          salt, 8, 10000, EVP_sha256(), 32, key.data());
        RAND_bytes(iv.data(), 16);
    }

    // Шифрування AES-256-CBC
    std::vector<uint8_t> encrypt(const std::vector<uint8_t>& plaintext) {
        EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
        std::vector<uint8_t> ciphertext(plaintext.size() + EVP_MAX_BLOCK_LENGTH);
        int len = 0, ciphertext_len = 0;

        EVP_EncryptInit_ex(ctx, EVP_aes_256_cbc(), nullptr, key.data(), iv.data());
        EVP_EncryptUpdate(ctx, ciphertext.data(), &len,
                          plaintext.data(), plaintext.size());
        ciphertext_len = len;
        EVP_EncryptFinal_ex(ctx, ciphertext.data() + len, &len);
        ciphertext_len += len;

        EVP_CIPHER_CTX_free(ctx);
```

```
        ciphertext.resize(ciphertext_len);
        return ciphertext;
    }

    // Розшифрування
    std::vector<uint8_t> decrypt(const std::vector<uint8_t>& ciphertext) {
        EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
        std::vector<uint8_t> plaintext(ciphertext.size());
        int len = 0, plaintext_len = 0;

        EVP_DecryptInit_ex(ctx, EVP_aes_256_cbc(), nullptr, key.data(), iv.data());
        EVP_DecryptUpdate(ctx, plaintext.data(), &len,
                          ciphertext.data(), ciphertext.size());
        plaintext_len = len;
        EVP_DecryptFinal_ex(ctx, plaintext.data() + len, &len);
        plaintext_len += len;

        EVP_CIPHER_CTX_free(ctx);
        plaintext.resize(plaintext_len);
        return plaintext;
    }

    std::vector<uint8_t> getIV() const { return iv; }
};

// ==================== Мережевий транспорт ====================
class NetworkTransport {
private:
    int socket_fd;
    std::string hostAddress;
    int port;
    std::mutex socketMutex;

public:
    NetworkTransport(const std::string& host, int p)
        : hostAddress(host), port(p), socket_fd(-1) {}

    bool connect() {
        std::lock_guard<std::mutex> lock(socketMutex);
        socket_fd = socket(AF_INET, SOCK_STREAM, 0);
        if(socket_fd < 0) {
            std::cerr << "Помилка: не можна створити сокет\n";
            return false;
        }

        struct sockaddr_in serv_addr;
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_port = htons(port);

        if(inet_pton(AF_INET, hostAddress.c_str(), &serv_addr.sin_addr) <= 0) {
            std::cerr << "Помилка: невірна адреса\n";
            return false;
        }

        if(::connect(socket_fd, (struct sockaddr *)&serv_addr,
                     sizeof(serv_addr)) < 0) {
            std::cerr << "Помилка підключення\n";
            return false;
        }

        return true;
    }
```

```
    bool sendData(const std::vector<uint8_t>& data) {
        std::lock_guard<std::mutex> lock(socketMutex);
        if(socket_fd < 0) return false;

        uint32_t len = htonl(data.size());
        if(send(socket_fd, &len, sizeof(len), 0) < 0) return false;
        if(send(socket_fd, data.data(), data.size(), 0) < 0) return false;

        return true;
    }

    std::vector<uint8_t> receiveData() {
        std::lock_guard<std::mutex> lock(socketMutex);
        std::vector<uint8_t> result;
        if(socket_fd < 0) return result;

        uint32_t len;
        if(recv(socket_fd, &len, sizeof(len), 0) < 0) return result;

        len = ntohl(len);
        result.resize(len);
        if(recv(socket_fd, result.data(), len, 0) < 0) result.clear();

        return result;
    }

    void disconnect() {
        std::lock_guard<std::mutex> lock(socketMutex);
        if(socket_fd >= 0) {
            close(socket_fd);
            socket_fd = -1;
        }
    }

    ~NetworkTransport() { disconnect(); }
};

// ==================== Агент ====================
class Agent {
private:
    std::string agentId;
    std::vector<std::string> peers;
    std::unique_ptr<CryptoModule> crypto;
    std::map<std::string, std::unique_ptr<NetworkTransport>> connections;
    std::queue<DataContainer> messageQueue;
    std::mutex queueMutex;

public:
    Agent(const std::string& id, const std::string& password)
        : agentId(id), crypto(std::make_unique<CryptoModule>(password)) {}

    void registerPeer(const std::string& peerId, const std::string& host, int port) {
        peers.push_back(peerId);
        connections[peerId] = std::make_unique<NetworkTransport>(host, port);
    }

    bool sendContainer(const std::string& targetPeerId, DataContainer& container) {
        container.calculateChecksum();

        // Серіалізація
        Json::Value json = container.toJson();
```

```
        Json::StreamWriterBuilder writer;
        std::string jsonStr = Json::writeString(writer, json);
        std::vector<uint8_t> plaintext(jsonStr.begin(), jsonStr.end());

        // Шифрування
        auto encrypted = crypto->encrypt(plaintext);

        // Відправка
        if(connections.find(targetPeerId) == connections.end()) {
            std::cerr << "Агент " << targetPeerId << " не знайден\n";
            return false;
        }

        auto& transport = connections[targetPeerId];
        if(!transport->connect()) {
            std::cerr << "Помилка підключення до " << targetPeerId << "\n";
            return false;
        }

        bool success = transport->sendData(encrypted);
        transport->disconnect();

        std::cout << "[" << agentId << "] Відправлено контейнер для "
                  << targetPeerId << " (размер: " << encrypted.size() << ")\n";
        return success;
    }

    DataContainer receiveContainer(const std::vector<uint8_t>& encryptedData) {
        // Розшифрування
        auto decrypted = crypto->decrypt(encryptedData);
        std::string jsonStr(decrypted.begin(), decrypted.end());

        // Десеріалізація
        Json::Value json;
        Json::CharReaderBuilder reader;
        std::string errs;
        std::istringstream iss(jsonStr);
        if(!Json::parseFromStream(reader, iss, &json, &errs)) {
            std::cerr << "Помилка парсинга JSON: " << errs << "\n";
            return DataContainer("unknown", "unknown");
        }

        DataContainer container = DataContainer::fromJson(json);
        std::cout << "[" << agentId << "] Отримано контейнер від "
                  << container.getAgentId() << "\n";
        return container;
    }

    std::string getAgentId() const { return agentId; }
};

// ==================== Головна програма ====================
int main() {
    std::cout << "=== Мультиагентна система безпечного обміну даними ===\n\n";

    // Створення агентів
    Agent agentA("AgentA", "secure_password_123");
    Agent agentB("AgentB", "secure_password_123");

    // Реєстрація пірів
    agentA.registerPeer("AgentB", "127.0.0.1", 5000);
    agentB.registerPeer("AgentA", "127.0.0.1", 5001);
```

```
    // Створення та відправка контейнера
    DataContainer container("AgentA", "SensorData");
    container.setPayload("Temperature: 25.5C, Humidity: 60%");
    container.calculateChecksum();

    std::cout << "\nДані контейнера:\n";
    std::cout << "- Агент: " << container.getAgentId() << "\n";
    std::cout << "- Тип: " << container.getDataType() << "\n";
    std::cout << "- Контрольна сума: " << container.getChecksum() << "\n\n";

    // Симуляція передачі
    std::cout << "Відправлення безпечного контейнера...\n";
    agentA.sendContainer("AgentB", container);

    return 0;
}
```