Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«СИСТЕМА ЗБОРУ ДОМЕННОЇ ІНФОРМАЦІЇ НА ОСНОВІ МУЛЬТИАГЕНТНОГО ПІДХОДУ ДЛЯ ЗАДАЧ КІБЕРБЕЗПЕКИ»

Виконав: студент 2 курсу, групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____ Олександр ЗАЛЕПА

Керівник: к. т. н., доцент каф. ЗІ
_____ Леонід КУПЕРШТЕЙН
«16» грудня 2025 р.

Опонент: к. т. н., доцент каф. ПЗ
_____ Наталя БАБЮК
«16» грудня 2025 р.

Допущено до захисту
В. о. зав. каф. ЗІ д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» грудня 2025 р.

Вінниця ВНТУ – 2025 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ

В. о. зав. каф. ЗІ д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«29» вересня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Залепі Олександру Вячеславовичу

1. Тема роботи: «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки»
   керівник роботи: Куперштейн Леонід Михайлович, к. т. н., доцент кафедри ЗІ, затверджені наказом №313 ректора ВНТУ від «24» вересня 2025 р.
2. Строк подання студентом роботи «16» грудня 2025 р.
3. Вихідні дані до роботи:
   – формат додатку – веб-застосунок;
   – архітектура – мультиагентна система;
   – мова програмування – Python;
4. Зміст текстової частини: Вступ. 1. Аналіз предметної області. 2. Розробка архітектурних рішень. 3. Програмна реалізація системи. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Архітектура мультиагентної системи. Загальна структурна схема ШІ-агента. Блок-схема алгоритму роботи агента координатора мультиагентної системи. Блок-схема алгоритму розширеного DNS-аналізу. Блок-схема алгоритму аналізу веб-контенту. Фрагменти тестування. Блок-схема алгоритму роботи WHOIS-агента.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	25.09.25	6.10.25
2	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	25.09.25	03.11.25
3	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	29.09.25	17.11.25
4	Олександр ЛЕСЬКО, к. е. н., доц., зав. каф. ЕПВМ	18.11.25	22.11.25

7. Дата видачі завдання «24» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.09.2025	
3	Науково-технічне обгрунтування	11.10.2025 – 22.10.2025	
4	Проєктування архітектури мультиагентної системи та специфікація компонентів	23.10.2025 – 26.10.2025	
5	Розроблення програмної структури спеціалізованих ШІ-агентів та агента координатора системи	27.10.2025 – 02.11.2025	
6	Інтеграція модулів, розширення функціоналу та реалізація вебінтерфейсу	03.11.2025 – 10.11.2025	
7	Тестування системи та підготовка результатів експериментів	10.11.2025 – 17.11.2025	
8	Розробка економічного розділу	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студент _____ Олександр ЗАЛЕПА

Керівник роботи _____ Леонід КУПЕРШТЕЙН

(підпис)

АНОТАЦІЯ

УДК 004.056.5

Залепа О. Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 106 с.

Укр. мовою. Бібліогр.: 62 назв; рис.: 38; табл.: 20.

Магістерська робота присвячена розробленню мультиагентної системи автоматизованого збору та аналізу доменної інформації з відкритих джерел для задач кібербезпеки. Система об’єднує спеціалізовані агенти WHOIS, DNS, SSL/TLS, Shodan, VirusTotal, crt.sh та агент аналізу вебвмісту, які виконують паралельне збирання технічних артефактів, забезпечуючи високу швидкість і повноту обробки даних. Розроблено архітектуру взаємодії спеціалізованих агентів та агента координатора, формалізовано математичну модель системи й алгоритми функціонування всіх модулів. Реалізовано програмний прототип із вебінтерфейсом та візуалізацією графу доменної інфраструктури. Проведено тестування роботи системи на доменах різної категорії та виконано порівняння з існуючими інструментами. У економічному розділі оцінено витрати та потенційну ефективність розробленого рішення.

Ключові слова: OSINT, доменна розвідка, мультиагентна система, DNS аналіз, WHOIS дані, SSL/TLS, Shodan, VirusTotal, аналіз вебвмісту, кібербезпека.

ABSTRACT

Zalepa O. System for collecting domain information based on a multi-agent approach for cybersecurity tasks. Master's thesis in the specialty 125 – Cybersecurity and Information Protection, educational program – Security of Information and Communication Systems. Vinnytsia: VNTU, 2025. 106 p.

In Ukrainian language. Bibliography: 62 titles; figures: 38; tables: 20

The master's thesis is devoted to the development of a multi-agent system for the automated collection and analysis of domain information from open sources for cybersecurity tasks. The system combines specialized WHOIS, DNS, SSL/TLS, Shodan, VirusTotal, crt.sh agents and a web content analysis agent, which perform parallel collection of technical artifacts, ensuring high speed and completeness of data processing. The architecture of interaction between agents and the coordinator has been developed, and the mathematical model of the system and the algorithms for the functioning of all modules have been formalized. A software prototype with a web interface and visualization of the domain infrastructure graph has been implemented. The system has been tested on domains of various categories and compared with existing tools. The economic section evaluates the costs and potential effectiveness of the developed solution.

Keywords: OSINT, domain intelligence, multi-agent system, DNS analysis, WHOIS data, SSL/TLS, Shodan, VirusTotal, web content analysis, cybersecurity.

ЗМІСТ

ВСТУП

У сучасному світі кількість публічної інформації в Інтернеті стрімко зростає, а отже обробка таких обсягів вручну стає практично неможливою. Відповідно, збір даних із відкритих джерел та аналіз доменних даних стають ключовими для своєчасного виявлення кіберзагроз. Доменна розвідка охоплює збір, аналіз та застосування інформації про доменні імена та їхню інфраструктуру. Такий підхід передбачає отримання відомостей про реєстрантів доменів, DNS-записи та IP-адреси, пов’язані з доменом, що дозволяє виявляти зловмисні ресурси та попереджувати потенційні атаки. Для ефективного опрацювання великих масивів відкритих даних перспективним є застосування мультиагентних систем. Мультиагентні системи – це розподілені системи, що складаються з кількох автономних агентів, які взаємодіють для досягнення спільної мети. Вони відзначаються високою гнучкістю, адаптивністю, здатністю до самовідновлення та стійкістю до збоїв, що робить їх привабливим рішенням для динамічних завдань кібербезпеки.

Ця проблема особливо актуальна в умовах гібридної війни в Україні, коли кібератаки є частиною загроз національній безпеці. Під час повномасштабного вторгнення РФ розвідка на основі відкритих даних відіграла вирішальну роль у своєчасному отриманні актуальних відомостей про воєнні ризики. Зокрема, зазначається, що опубліковані в Інтернеті доменні імена разом з відповідними IP-адресами є загальнодоступними через стандартні DNS-запити. Це свідчить про наявність великих обсягів відкритої доменної інформації, яку необхідно автоматизовано збирати та аналізувати для забезпечення кібербезпеки українських систем.

Метою роботи є розширення функціональних можливостей інструментів для збору доменної інформації та її аналізу для потреб кібербезпеки на основі мультиагентного підходу із використанням генеративного штучного інтелекту.

Завдання:

1) Проаналізувати існуючі підходи і засоби збору доменної інформації з відкритих джерел і їх обмеження.
2) Розробити архітектуру мультиагентної системи, здатної автоматизувати процес збору даних про домени.
3) Спроектувати взаємодію автономних агентів у системі та розробити алгоритми їхньої координації.
4) Реалізувати прототип системи, що складається з декількох спеціалізованих агентів для отримання WHOIS-даних, DNS-записів та іншої доменної інформації.
5) Провести експериментальне тестування розробленої системи на прикладах доменних даних та оцінити її продуктивність і ефективність порівняно з традиційними методами.

Об’єктом дослідження є процес збору доменної інформації для задач кібербезпеки.

Предметом дослідження є методи та засоби збору доменної інформації.

У роботі застосовано такі методи дослідження:

- системний аналіз і узагальнення наявних підходів до збору та аналізу доменної інформації;
- методи моделювання та проєктування архітектури розподіленої мультиагентної системи;
- теоретико-множинний підхід і формалізація для опису процесів збору та структури даних;

– розробка програмного прототипу системи та емпіричне тестування на фактичних прикладах доменних даних.

Наукова новизна роботи полягає у формуванні концепції мультиагентної системи збору доменної інформації, у якій інтелектуальні агенти різної спеціалізації взаємодіють у єдиному обчислювальному середовищі, координуючи свої дії через центрального керуючого агента, що дозволяє забезпечити адаптивний, масштабований та формалізований процес збору доменних даних з відкритих джерел у режимі, близькому до реального часу.

Практичне значення роботи полягає у тому, що запропонований мультиагентний підхід і розроблена на його основі система дозволяють істотно скоротити час комплексного OSINT-аналізу доменних імен. На основі експериментальних вимірювань встановлено, що тривалість повного циклу аналізу одного домену зменшується з 10–12 хв до 1–2 хв, що відповідає прискоренню у 5–6,5 разів порівняно з ручним або напівавтоматизованим підходом.

Автоматизація рутинних операцій збору та первинного аналізу доменної інформації дозволяє фахівцям з інформаційної безпеки зосередитись на інтерпретації результатів і прийнятті управлінських рішень. Система формує структурований технічний звіт із інтегральною оцінкою ризику, що робить результати аналізу доступними як для технічних спеціалістів, так і для керівників.

Результати роботи можуть бути використані в службах кіберзахисту підприємств, центрах реагування на інциденти, навчальних лабораторіях і дослідницьких середовищах.

Отримані результати створюють методичну основу для подальших досліджень мультиагентних моделей у доменній розвідці, зокрема з використанням генеративних агентів і механізмів самонавчання.

Основні результати магістерської кваліфікаційної роботи обговорювались міжнародній науково-практичній Інтернет-конференції «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2026)» (Україна, Вінниця) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття доменної інформації та її роль у кібербезпеці

Доменна інформація – це сукупність технічних і реєстраційних даних, що описують доменне ім’я, його власника, налаштування системи доменних імен (DNS), параметри SSL/TLS-сертифікатів та інші характеристики, які забезпечують коректне функціонування ресурсу в мережі Інтернет [2]. Вона є невід’ємним елементом інтернет-інфраструктури, адже поєднує зрозумілі для користувачів адреси з технічними параметрами мережі, забезпечуючи маршрутизацію трафіку та ідентифікацію ресурсів.

Отримання доменної інформації дає змогу дослідити структуру веб-ресурсу, визначити власника, інфраструктуру розміщення, термін делегування, типи DNS-записів і наявність захисту трафіку. Такі дані мають вирішальне значення для кібербезпеки, оскільки доменні імена часто виступають ключовими індикаторами компрометації (Indicators of Compromise, IoC) – їх використовують у фішингових кампаніях, мережах командно-керувальних серверів, розповсюдженні шкідливого ПЗ чи викраденні даних [3].

За статистичними даними, щодня у світі реєструється близько 200 000 нових доменних імен, при цьому до 70 % із них можуть бути потенційно зловмисними або щонайменше підозрілими. Значна частина таких доменів створюється на короткий термін і використовується для проведення швидких атак, після чого вони припиняють існування або змінюють призначення. Такий масштаб і динамічність загроз зумовлюють необхідність застосування автоматизованих засобів моніторингу новостворених доменів, аналізу їхніх характеристик та оцінювання репутації з використанням алгоритмів машинного навчання й аналітичних платформ (рис. 1.1) [4].

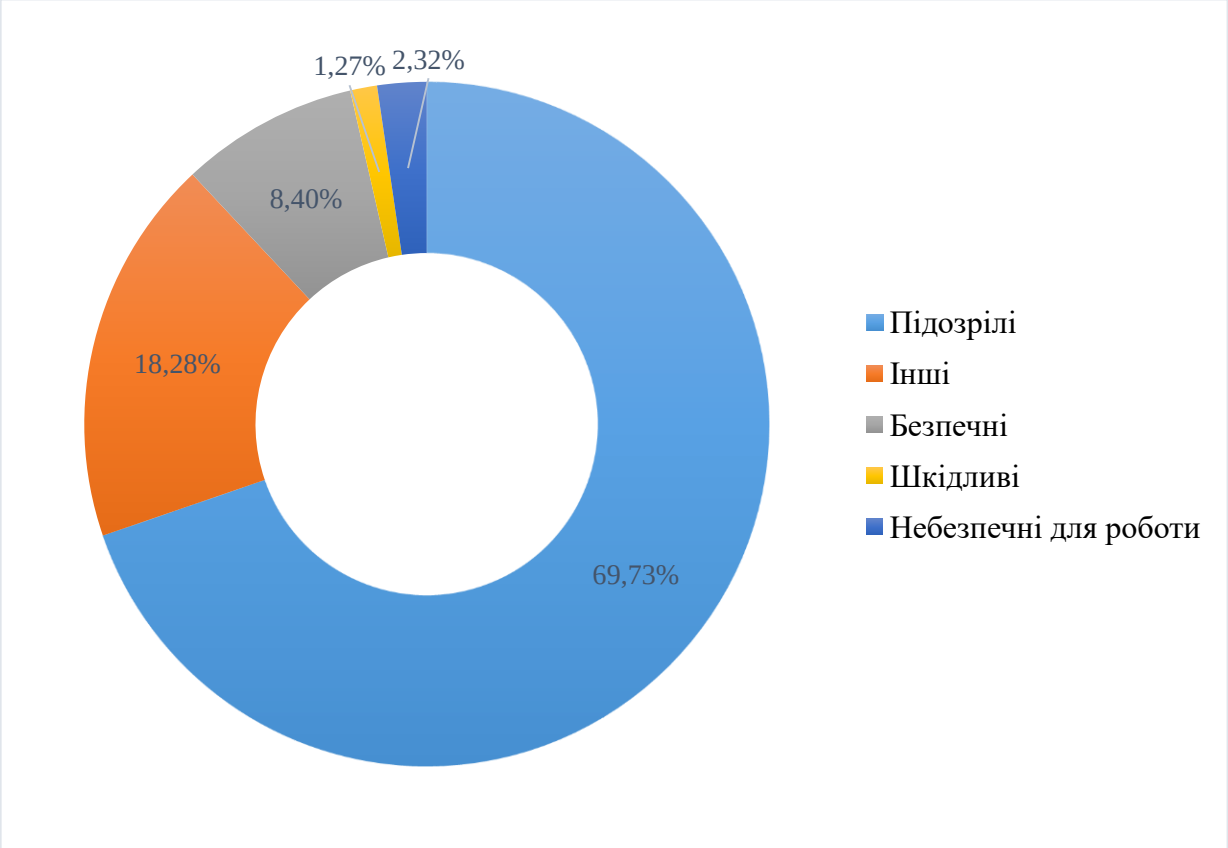


Рисунок 1.1 – Статистика розподілу нових доменів за зонами (TLD) та категоріями безпечності доменів у 2019 році

У корпоративному середовищі поширеною практикою є тимчасове блокування доступу до невідомих доменів, доки вони не пройдуть перевірку через системи аналізу WHOIS-даних, DNS-записів і сертифікатів SSL/TLS.

Одним із найпоширеніших видів зловживань у кіберпросторі залишаються фішингові домени, що імітують легітимні ресурси компаній та державних установ. Згідно зі звітом Interisle «Phishing Landscape 2024», у 2023 році зафіксовано приблизно 1,9 млн фішингових інцидентів у всьому світі, що на близько 50 тис. більше, ніж роком раніше. Дослідження також відзначає зростання на 51 % кількості атак, розміщених на субдомен-провайдерах, а також активне використання нових доменів верхнього рівня (gTLD) для маскування зловмисних ресурсів [5].

Зловмисники використовують техніки транспозиції символів (наприклад, paypaI.com замість paypal.com), гомографічні атаки з Unicode-символами (apple.com), а також субдоменне маскування (login.google.com.fake-secure.net). Такі ресурси спрямовані на крадіжку облікових даних, поширення шкідливого ПЗ або збір персональної інформації.

Виявлення таких доменів ускладнюється їх коротким життєвим циклом – у середньому від кількох годин до кількох діб. Для приховування активності зловмисники застосовують динамічні DNS-служби та автоматизовані платформи реєстрації; саме тому аналіз WHOIS-записів, DNS-структури та SSL/TLS-сертифікатів залишається одним із небагатьох способів виявлення та кореляції підозрілих доменів. Співпадіння дати реєстрації, реєстратора або використання однакових DNS-серверів часто вказує на існування спільної інфраструктури або автоматизованої фішингової мережі.

Окрім фішингу, доменні імена активно використовуються в інших типах кіберзагроз:

- для розміщення командно-керувальних (Command-and-Control, C2) серверів ботнетів [6];
- поширення шкідливого програмного забезпечення [7];
- реалізації домен-генераційних алгоритмів (DGA) [8];
- перехоплення поштового трафіку через зміну MX-записів [9].

Моніторинг таких активностей дозволяє проводити проактивний захист – виявляти фішингові кампанії, DGA-активність чи підготовку до атак на ранніх етапах.

У межах кіберрозвідки (Cyber Threat Intelligence, CTI) доменна інформація використовується для виявлення зв’язків між елементами атакуючої інфраструктури. Аналіз WHOIS-записів дозволяє виявити спільні атрибути – електронну адресу реєстранта, DNS-сервери чи IP-адреси, що об’єднують кілька

доменів. Метод pivoting дає змогу встановити групи доменів, пов’язані з однією кіберзлочинною кампанією, та ідентифікувати нові елементи загроз [10].

У процесі реагування на інциденти (Digital Forensics and Incident Response, DFIR) доменна інформація також є важливим джерелом цифрових слідів. Дані WHOIS і DNS допомагають з’ясувати час і місце реєстрації домену, пов’язати інциденти за повторним використанням IP-адрес, реєстраторів або DNS-серверів. Додаткову кореляцію забезпечує аналіз TLS-сертифікатів – збіг серійних номерів, імен емітентів чи параметрів шифрування дозволяє визначати домени, що належать до однієї зловмисної інфраструктури [11].

Окрім аналітичних і форензічних задач, доменна інформація відіграє важливу роль і в процесах тестування на проникнення (penetration testing). Під час пентесту дослідники безпеки імітують дії потенційного зловмисника, виконуючи розвідку доменів (reconnaissance) для виявлення піддоменів, служб, відкритих портів і конфігураційних слабких місць. Аналіз WHOIS, DNS та SSL/TLS-даних дозволяє ідентифікувати додаткові вектори атаки, пов’язані з неправильними записами, вразливими хостами або некоректно налаштованими сертифікатами. Таким чином, доменна розвідка є першим етапом пентесту, який забезпечує побудову карти цілей та виявлення поверхні атаки, що підлягає подальшому аналізу під час експлуатаційних тестів.

Таким чином, доменна інформація є базовим аналітичним ресурсом для систем кібербезпеки, кіберрозвідки та цифрової криміналістики. Її систематичний збір і аналіз створює передумови для проактивного виявлення загроз, підвищення надійності інформаційних систем і зміцнення загального рівня кіберзахисту.

1.2 Види доменної інформації та джерела її отримання

Доменна інформація охоплює сукупність даних, що характеризують доменне ім’я та пов’язану з ним інфраструктуру.

Для формування повного профілю домену використовуються кілька основних типів таких даних:

- WHOIS-записи [12];
- DNS-записи [13];
- дані цифрових сертифікатів (SSL/TLS) [14];
- відомості з відкритих джерел (OSINT) [15].

Кожен із цих типів відображає окремий аспект функціонування домену, а їх комплексний аналіз дозволяє отримати цілісне уявлення про власника, архітектуру та технічне середовище ресурсу.

WHOIS-інформація містить реєстраційні дані домену, а саме ім’я та контакти власника (реєстранта), реєстратора, дати створення і закінчення делегування, статус домену, а також використовувані DNS-сервери [16]. Вона дає змогу встановити, хто стоїть за доменом, коли він був зареєстрований і наскільки давно існує. Ці дані отримують за допомогою консольних утиліт (наприклад, whois, nslookup) або веб-сервісів (ICANN Lookup, Whois.com тощо).

Після впровадження законодавства про захист персональних даних (зокрема GDPR) частина відомостей може бути прихована або замінена записами типу Redacted for privacy. Проте навіть за умов обмежень WHOIS залишається важливим джерелом технічних метаданих (реєстратор, статус, NS-сервери, дата створення), які допомагають ідентифікувати юрисдикцію домену та оцінити ризики. Історичні WHOIS-дані, доступні через комерційні сервіси (наприклад, WhoisXML API, DomainTools), дозволяють відстежувати зміни власників і використовуються під час розслідувань або побудови графів доменних зв’язків.

DNS-записи визначають технічну конфігурацію домену й описують, як маршрутизується трафік [13].

Основні типи записів включають:

– A/AAAA – прив’язка домену до IP-адреси (IPv4/IPv6);

– MX – поштові сервери;

– NS – авторитетні DNS-сервери;

– CNAME – псевдоніми (перенаправлення на інші домени);

– TXT – додаткові дані, зокрема SPF/DKIM-записи для захисту пошти;

– CAA – дозволені центри сертифікації.

Отримання DNS-даних здійснюється через утиліти dig, nslookup, host або спеціалізовані онлайн-сервіси (Google DNS Dig, MXToolbox). Аналіз DNS дає змогу виявити IP-адреси серверів, постачальників послуг (наприклад, CDN чи поштових провайдерів), а також побічно визначити географічне розташування хостингу. Крім того, DNS-аналіз допомагає знайти відомі субдомени або виявити приховані зв’язки між компонентами інфраструктури. Попри окремі обмеження, DNS залишається базовим джерелом технічних відомостей, необхідних для побудови карти зв’язків домену.

Дані SSL/TLS-сертифікатів доповнюють профіль домену інформацією про його автентичність і безпеку [14]. Кожен сертифікат містить відомості про доменне ім’я (Common Name), перелік альтернативних імен (Subject Alternative Names, SAN), емітента (центр сертифікації), терміни дії та криптографічні параметри. Для сертифікатів типу EV або OV може бути зазначена юридична особа-власник. Завдяки системі Certificate Transparency усі публічні сертифікати зберігаються у відкритих журналах, що дозволяє виконувати пошук за доменами. Сервіс crt.sh забезпечує доступ до таких даних і дозволяє знайти всі сертифікати, у яких згадується певний домен або його субдомени.

Це корисно для виявлення прихованих частин інфраструктури, якщо для example.com коли-небудь видавався сертифікат на mail.example.com чи api.example.com, ці імена відображатимуться у базі.

Аналіз SSL-сертифікатів допомагає не лише підтвердити автентичність ресурсу, а й виявити технічні або організаційні зв’язки між доменами, що використовують спільні сертифікати або центри сертифікації.

Відкриті джерела інформації (Open Source Intelligence, OSINT) становлять фундамент сучасних підходів до пасивної кіберрозвідки та доменного аналізу, оскільки дозволяють отримувати технічні, поведінкові, репутаційні та інфраструктурні дані про мережеві ресурси без виконання активних втручань або дій, що можуть розцінюватися як агресивні. Концепція OSINT ґрунтується на принципі використання загальнодоступних ресурсів, у тому числі вебсервісів, публічних баз даних, реєстрів сертифікатів, пошукових механізмів інтернет-сканерів та репутаційних платформ. Така модель забезпечує високу масштабованість і водночас мінімізує ризики виявлення під час аналізу, що особливо важливо в умовах дослідження потенційно небезпечних або зловмисних доменів.

Важливою особливістю OSINT є його багатовимірність, тобто різні джерела відображають різні частини інфраструктури домену, а їх сукупність створює інтегроване уявлення про технічний профіль ресурсу. Пошукові системи, зокрема Google та Bing, дозволяють оцінити ступінь індексації домену, виявити історичні сліди змін контенту або згадок у сторонніх ресурсах, що може бути корисним у разі аналізу фішингових доменів, створених для імітації легітимних служб. Проте їхній внесок у технічний аспект обмежений, оскільки пошукові індекси не відображають внутрішню мережеву структуру чи конфігурацію сервісів.

Значно глибший рівень технічного аналізу забезпечують платформи, спеціально призначені для систематичного сканування Інтернету. Наприклад, Shodan використовує механізми постійного активного зондування відкритих портів, банерів сервісів, протоколів та конфігурацій серверів, створюючи глобальну карту пристроїв, підключених до мережі [16]. Інформація з Shodan дозволяє оцінити

рівень відкритості доменної інфраструктури, визначити потенційні вектори атак, знайти некоректно налаштовані або застарілі служби. Censys, у свою чергу, доповнює цей підхід більш глибоким аналізом отриманих даних та автоматичним зіставленням знайдених сервісів з відомими вразливостями (CVE) [17]. Ці платформи формують критичний шар технічного контексту, який неможливо отримати з класичних DNS- або WHOIS-запитів.

Репутаційні джерела, такі як VirusTotal, відіграють ключову роль у виявленні ознак шкідливої активності [18]. Завдяки агрегуванню даних від численних антивірусних рішень та систем виявлення загроз, VirusTotal дає можливість визначити, чи асоціюється домен з фішингом, ботнетами, поширенням шкідливих файлів або підозрілою поведінкою в Інтернеті. Аналіз категорій, голосів спільноти та репутаційних тегів дозволяє формувати більш точну оцінку ризику, особливо для нових або нещодавно зареєстрованих доменів, які можуть бути створені зловмисниками.

Окремий напрям OSINT-розвідки стосується роботи з історичними та пасивними базами сертифікатів. Платформа crt.sh агрегує записи з Transparency Logs, дозволяючи виявляти всі TLS-сертифікати, видані для певного доменного імені [18]. Це відкриває можливості для пасивного пошуку субдоменів, відстеження історії SSL-конфігурацій, виявлення підозрілих або дубльованих сертифікатів, а також визначення змін в інфраструктурі – наприклад, переходу ресурсу до іншого хостинг-провайдера. Такі дані є особливо важливими для мультиагентних систем, що виконують автоматичний аналіз структури домену з метою побудови графів зв’язків.

Історичні DNS-дані, доступні через платформи на кшталт SecurityTrails, доповнюють OSINT-картину, дозволяючи досліджувати, як змінювалися IP-адреси домену, які субдомени з’являлися або зникали, а також які службові записи (наприклад, MX або TXT) зазнавали модифікацій [20]. Це дає можливість визначати

аномальні шаблони поведінки, характерні для підготовчих етапів фішингових кампаній або перемикання інфраструктури на нові сервери після інцидентів безпеки.

Таким чином, відкриті джерела інформації не є однорідним масивом даних, а представляють комплекс взаємодоповнюючих ресурсів, кожен з яких виконує окрему роль у формуванні цілісного технічного портрету домену. Поєднання адміністративних (WHOIS), технічних (DNS), криптографічних (SSL/TLS), мережевих (Shodan, Censys) та репутаційних (VirusTotal) даних створює основу для розроблення систем автоматизованого доменного аналізу.

1.3 Мультиагентні системи в кібербезпеці

Мультиагентна система (МАС) – це програмна система, що складається з кількох взаємодіючих інтелектуальних агентів. Кожен агент є автономним програмним компонентом, наділеним власними цілями, знаннями та можливостями. На відміну від монолітних рішень, МАС дозволяє розподілити складні завдання між спеціалізованими агентами, які спільно досягають глобальної мети системи. Важливою властивістю є автономність – агенти приймають рішення незалежно, маючи лише локальне уявлення про середовище. Це робить мультиагентні системи гнучкими, масштабованими та стійкими до збоїв окремих компонентів [21].

Архітектура МАС може бути централізованою або децентралізованою. У централізованій моделі існує агент координатор, що розподіляє завдання та об’єднує результати, спрощуючи керування, проте створюючи залежність від єдиного вузла. Децентралізовані системи функціонують без глобального контролю: агенти обмінюються повідомленнями або взаємодіють через спільне середовище, узгоджуючи дії шляхом комунікації. Така архітектура забезпечує відмовостійкість, але вимагає розв’язання задач узгодження планів і спільного прийняття рішень [21].

У кібербезпеці мультиагентний підхід широко використовується для моніторингу, виявлення загроз і реагування на інциденти. Наприклад, у системах виявлення вторгнень (IDS) кілька агентів одночасно аналізують різні сегменти мережі, обмінюються даними про підозрілу активність і колективно формують висновок про наявність атаки. Це знижує ризик хибних спрацювань і усуває єдину точку відмови.

Мультиагентні системи застосовуються не лише для виявлення загроз, а й для автоматизованого реагування на інциденти. У такій системі окремі агенти можуть бути відповідальними за виявлення, класифікацію атаки, оцінку її впливу та формування плану реагування. Координацію дій виконує оркеструючий агент, який синхронізує роботу інших. Такий підхід узгоджується з сучасними підходами до SOAR/гіперавтоматизації, де агентні (agentic) рішення підвищують гнучкість сценаріїв автоматизації та підтримують адаптивне формування/виконання дій реагування [22].

Особливу ефективність MAC демонструють у сфері захисту критичної інфраструктури, зокрема енергетичних систем (smart grid). У таких мережах агенти розподіляються між вузлами інфраструктури й безперервно контролюють параметри роботи системи. Взаємодіючи між собою, вони здатні колективно виявляти аномалії, наприклад атаки типу False Data Injection, які змінюють показники сенсорів. Завдяки алгоритмам консенсусу результати аналізу окремих агентів узгоджуються, що дозволяє виявляти підозрілі впливи в реальному часі з високою точністю – у деяких експериментах точність досягала понад 99%. Такий підхід підвищує надійність системи, адже кожен агент може локально ініціювати дії для ізоляції скомпрометованого вузла, зберігаючи стабільність мережі.

Загалом мультиагентні системи дають змогу реалізувати принципи децентралізованого, адаптивного та інтелектуального захисту. Вони поєднують переваги розподіленої обробки даних із можливістю колективного прийняття

рішень, що особливо важливо у складних динамічних середовищах кібербезпеки. Їх використання сприяє підвищенню ефективності моніторингу, швидкості реагування та стійкості систем до нових типів атак. Завдяки автономності, комунікаційним можливостям і спеціалізації агентів МАС стають ключовим інструментом побудови сучасних інтелектуальних систем кіберзахисту.

У контексті дослідження мультиагентних систем у кібербезпеці варто відзначити роботу Сухарева та Куперштейна [23], у якій представлено мультиагентну архітектуру для розвідки з відкритих джерел на базі фреймворку CrewAI. У ній п’ять ШІ-агентів із вбудованими великими мовними моделями виконують взаємодоповнюючі ролі – від збору технічних даних до формування аналітичного звіту.

Такий підхід продемонстрував високу гнучкість, масштабованість і модульність системи, оскільки сценарії взаємодії агентів можна змінювати без втручання в код. Дослідження підтверджує доцільність використання мультиагентних архітектур у сфері кібербезпеки, де швидкість обробки та узгодженість рішень мають критичне значення.

1.4 Огляд існуючих рішень та інструментів

Сучасна кіберрозвідка активно використовує доменну інформацію з відкритих джерел для виявлення прихованих зв’язків і ознак зловмисної діяльності. Відповідно, з’явився широкий спектр сервісів і інструментів, що автоматизують збір доменної інформації.

Багато веб-сервісів надають доступ до WHOIS-інформації – реєстраційних даних домену, таких як ім’я власника, реєстратор, дати створення і закінчення делегування, статус та список DNS-серверів. Після посилення захисту персональних даних відкриті WHOIS-записи часто маскують контактні дані, проте технічні метадані (реєстратор, дати, NS) все ще доступні та корисні.

Спеціалізовані провайдери, як-от WhoisXML API та DomainTools, агрегують величезні бази даних WHOIS та надають історичні записи змін для аналізу еволюції доменів. Такі платформи дозволяють відстежувати зміни власників або DNS-конфігурації та здійснювати розширене “pivoting”-аналізування – пошук пов’язаних доменів за спільними атрибутами [24, 25].

Глибина збору в них значна – включно з історичними даними реєстрації, зворотним пошуком WHOIS та оцінками ризику домену. Ліцензування таких сервісів зазвичай комерційне, хоча часто передбачені обмежені безкоштовні рівні доступу для дослідників.

Структура DNS-домену надає інформацію про технічне налаштування і зв’язки домену з мережевою інфраструктурою. Для автоматизованого збору DNS-записів існують як системні утиліти/бібліотеки (напр. dig, nslookup, пакет dnspython у Python), так і пасивні DNS-сервіси – великі бази даних історичних запитів DNS.

Прикладом є SecurityTrails, що зосереджується на збагаченні даних про домени і IP. Сервіс дозволяє отримувати поточні та історичні DNS-записи, перелік субдоменів, зворотні пошуки IP. SecurityTrails надає зручний веб-інтерфейс і розвинений API для швидкої інтеграції в зовнішні системи [20].

Подібну функціональність пасивного DNS пропонує некомерційний проект CIRCL Passive DNS, де накопичуються спостереження DNS-запитів з різних мереж. Він корисний для виявлення прихованих зв’язків між доменами та IP-адресами за історією їх резолюції. Глибина даних у пасивних DNS-системах залежить від охоплення джерел. Комерційні платформи (SecurityTrails, DomainTools Iris тощо) охоплюють мільйони доменів глобально, тоді як відкриті проекти типу CIRCL Passive DNS надають вибіркові дані спільноти.

Відкриті індекси Certificate Transparency (наприклад, сервіс crt.sh) дозволяють шукати всі сертифікати, видані для певного доменного імені або його піддоменів. Це дає змогу виявити додаткові субдомени та домени, пов’язані з ціллю, навіть

якщо вони не згадані відкрито. Для автоматизації можна використовувати публічний API crt.sh або комерційні рішення [19].

Сервіс Censys здійснює повнотекстовий пошук по базі TLS-сертифікатів, знаходячи будь-які згадки домену у полі Common Name чи Subject Alternative Name. Censys надає структуровані дані про кожен знайдений сертифікат (включно з терміном дії, емітентом, криптографічними параметрами) та пов’язані доменні імена. Крім того, Censys виконує глибоке сканування IPv4-простору і пов’язує результати із сертифікатами, що дозволяє бачити, на яких хостах використовуються певні доменні сертифікати. API Censys дає можливість інтегрувати ці пошуки у власні програми. Ліцензування Censys – комерційне, з обмеженим безкоштовним планом (кілька запитів на день) [17].

Окремо варто виділити сервіси активного сканування, такі як Shodan та згаданий Censys, котрі можна вважати “пошуковими системами” для пристроїв і серверів. Shodan індексує інформацію про інтернет-хости, збираючи банери сервісів на відкритих портах (наприклад, веб-серверів, FTP, SSH тощо) по всьому світу. На відміну від традиційних пошуковиків, які індексують лише веб-сторінки, Shodan орієнтований на пристрої та їх програмне забезпечення [16].

Для доменної розвідки Shodan корисний тим, що має окрему базу DNS-записів. Він може видати перелік субдоменів для заданого домену, знайдених під час його сканувань і OSINT-пошуків. Крім того, Shodan дозволяє дізнатися, на яких IP-адресах розгорнуто ресурси домену, які порти відкриті і які сервіси працюють – це розширює картину інфраструктури цілі. Сервіс має відкритий REST-API, що підтримує пошук за доменом, IP, портом, з можливістю фільтрації за країною, ASN, операційною системою тощо.

Окрім вузькоспеціалізованих API-сервісів, існують інтегровані платформи та фреймворки для автоматизованого збору OSINT-даних про домени. Ці рішення часто об’єднують декілька модулів або джерел інформації в єдиний інструмент,

спрощуючи роботу аналітика. Більшість із них орієнтовані на сценарії кібербезпеки (пентестинг, моніторинг загроз).

SpiderFoot – один з найвідоміших відкритих інструментів такого класу. Це платформа автоматизації OSINT, яка має понад 200 модулів збору даних з найрізноманітніших джерел. SpiderFoot може збирати інформацію про доменне ім’я, IP-адресу, субдомени, записи в чорних списках, згадки в витоках даних, навіть аналізувати згадки в даркнеті. Важливо, що значна частина модулів працює безкоштовно, використовуючи загальнодоступні джерела, хоча інтеграція з деякими комерційними API теж підтримується [26].

Інструмент надає як веб-інтерфейс, так і консольний режим, що зручно для скриптового запуску у складі більшої системи. Однак варто відзначити, що проект SpiderFoot останнім часом оновлюється повільніше – після його придбання компанією Intel 471 у 2022 році відкриту версію майже не підтримують. Ліцензія SpiderFoot – GPL (відкрите ПЗ), тому вона безкоштовна для використання, але у виробничих середовищах треба врахувати потенційні проблеми з актуальністю деяких модулів.

Recon-ng – альтернативний фреймворк OSINT-розвідки, написаний мовою Python. За своєю структурою він нагадує Metasploit. Recon-ng дозволяє керувати робочими просторами, зберігати результати у внутрішній базі даних і нарощувати функціонал шляхом підключення нових модулів [27].

Багато модулів призначено саме для збору доменної інформації – наприклад, пошук хостів за допомогою Google Dorks, вилучення субдоменів через API пошукових систем, отримання даних WHOIS тощо. Інтеграція Recon-ng у мультиагентну систему може бути здійснена через запуск конкретних модулів із скриптів. Перевагою фреймворку є об’єднання кількох інструментів в одному місці. Замість того, щоб викликати багато різних утиліт, Recon-ng надає єдину платформу для збору різнорідних даних. Це сприяє підвищенню швидкості та організованості

збору – результати з різних модулів відразу централізовано зберігаються і доступні для подальшого аналізу. Recon-ng – проект з відкритим кодом. Він поставляється у складі дистрибутивів типу Kali Linux, що свідчить про його популярність серед фахівців з безпеки.

TheHarvester – ще один класичний інструмент, орієнтований передусім на швидкий пошук субдоменів та email-адрес, пов'язаних з доменом [28].

На відміну від SpiderFoot і Recon-ng, TheHarvester не має модульної архітектури, а являє собою скрипт, який послідовно опитує кілька публічних джерел (пошукові системи Google, Bing, бази ключів PGP, Shodan та ін.) для знаходження всіх згадок цільового домену. Його плюс – простота та швидкість, запуск TheHarvester займає лічені хвилини і дає початкову вибірку субдоменів і потенційно пов'язаних об'єктів. У мультиагентній системі TheHarvester можна використовувати як легковаговий агент-процес для первинної розвідки. Він не потребує складної конфігурації і швидко повертає результат. Ліцензування TheHarvester відкрите, інструмент інтегровано у Kali Linux і широко доступний.

Maltego – на відміну від попередніх, є не стільки автономним сканером, скільки інтерактивною аналітичною платформою для OSINT. Maltego дозволяє через систему "трансформів" підключатися до десятків зовнішніх джерел (від WHOIS та DNS до соціальних мереж) і будувати граф знань – візуалізовані зв'язки між доменами, IP, людьми, документами тощо [29].

У контексті мультиагентних систем Maltego може бути корисним як інструмент для людино-машинного аналізу. Наприклад, після автоматизованого збору даних агентами, результати можуть імпортуватися в Maltego для ручного дослідження та побудови зв'язків. Пряма ж автоматизація Maltego у бекенд-системах ускладнена – хоча існує Maltego Transform API для написання власних модулів, сам Maltego клієнт призначений для взаємодії аналітика. Maltego пропонує

безкоштовну спільнотну версію з обмеженим функціоналом і комерційні версії для корпоративних користувачів.

Щодо готових фреймворків мультиагентної розвідки, ця галузь лише формується. SYNINT – приклад експериментального відкритого рішення, де акцент зроблено на самодостатності агентів (мінімум зовнішніх викликів, локальні алгоритми для аналізу) та уніфікованій структурі для додавання нових агентів. Хоча SYNINT не вимагає API-ключів (на відміну від SpiderFoot чи Recon-ng), його можливості обмежені відсутністю доступу до великих сторонніх баз даних [30].

Узагальнення ключових характеристик розглянутих сервісів та інструментів доменної кіберрозвідки наведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика сервісів і фреймворків доменної кіберрозвідки

Засіб	Тип / роль у розвідці	Ключові можливості	Ліцензування / використання
WhoisXML API	Комерційний провайдер доменних, WHOIS і DNS-даних	Поточні та історичні WHOIS, пасивний DNS, IP-дані, зворотний пошук	Комерційне ПЗ, є обмежений безкоштовний рівень доступу
DomainTools (Iris)	Платформа доменної розвідки та ризик-скорингу	Глибока WHOIS-історія, пасивний DNS, оцінка ризику, pivot-аналіз	Комерційна платформа, орієнтована на корпоративних клієнтів
SecurityTrails	Сервіс пасивного DNS та аналізу поверхні атаки	Поточні й історичні DNS-записи, субдомени, IP, технічний стек сайтів	Комерційне ПЗ з API, безкоштовний план із жорсткими квотами
CIRCL Passive DNS	Некомерційний пасивний DNS-сервіс	Історичні DNS-записи з мереж спільноти, виявлення прихованих зв’язків	Доступ за запитом для довірених організацій, безкоштовно
crt.sh	Відкритий індекс Certificate Transparency	Пошук усіх TLS-сертифікатів для домену та піддоменів	Відкрите безкоштовне використання, неофіційне API, без гарантій SLA

Продовження таблиці 1.1

Засіб	Тип / роль у розвідці	Ключові можливості	Ліцензування / використання
Censys	Платформа сканування Інтернету й CT-пошуку	Пошук за TLS-сертифікатами, глобальне сканування IPv4/IPv6, метадані сервісів	Комерційні тарифи, є безкоштовний tier з обмеженою кількістю запитів
Shodan	«Пошуковик» для інтернет-пристроїв і сервісів	Банери сервісів, відкриті порти, ОС, геолокація, DNS-дані по домену	Комерційний сервіс з REST-API, безкоштовний доступ сильно обмежений
SpiderFoot	OSINT-платформа автоматизації збору даних	200+ модулів: WHOIS, DNS, витоки, чорні списки, соцмережі, даркнет	Відкритий код (GPL) + комерційна версія HX; зручно як універсальний збирач
Recon-ng	Модульний OSINT-фреймворк (CLI)	Модулі для WHOIS, DNS, субдоменів, IP, пошукових систем; внутрішня БД	Open-source; запускається зі скриптів як «двигун» агента розвідки
TheHarvester	Легковаговий скрипт для первинного збору	Збір субдоменів і email через пошуковики, PGP, Shodan тощо	Відкрите ПЗ; зручно як швидкий інструмент первинного сканування
Maltego	Інтерактивна платформа OSINT та графів зв’язків	«Трансформи» до десятків джерел, візуалізація зв’язків домен–IP–особи	Community-edition (безкоштовна з лімітами) + комерційні редакції
SYNINT	Експериментальний мультиагентний OSINT-фреймворк	Самодостатні агенти, акцент на локальних алгоритмах без зовнішніх API	Open-source; можливості обмежені відсутністю великих комерційних баз

З таблиці видно, що жодне з існуючих рішень не поєднує повною мірою глибоку доменну розвідку з гнучкою мультиагентною архітектурою, що й обґрунтовує доцільність розробки власної системи.

На основі існуючих спеціалізованих сервісів та OSINT-фреймворків, доцільно запропонувати мультиагентну систему, яка забезпечує комплексний доменний аналіз у єдиному середовищі, поєднуючи паралельний збір, семантичну LLM-аналітику, ризик-скоринг та візуалізацію інфраструктури домену без ліцензійних обмежень і аналіз контенту розміщеного за доменом.

1.5 Формалізація вимог та постановка задачі

У процесі розробки програмного забезпечення вимоги системи визначаються через функціональні та нефункціональні аспекти, які відображають як очікувану поведінку системи, так і якість її роботи.

Функціональні вимоги описують операції, які система повинна виконувати автоматично й коректно, тоді як нефункціональні фіксують властивості, що забезпечують стабільність, продуктивність, безпеку та якість користувацького досвіду. Розроблена мультиагентна система призначена для комплексного аналізу доменів на основі відкритих джерел даних, автоматизованого збору технічних артефактів та формування зведених аналітичних висновків.

Функціональні вимоги базуються на архітектурі системи, яка включає спеціалізованих ШІ-агентів на основі великих мовних моделях для опрацювання WHOIS-даних, DNS-структури, SSL-сертифікатів, пасивних доменних даних із бази crt.sh, інформації з Shodan, репутаційних показників VirusTotal, а також змістового аналізу вебсторінки, що реалізується окремим контент-агентом. Система повинна автоматично виконувати запити до зовнішніх сервісів, аналізувати відповіді, структурувати отримані артефакти та передавати їх уніфікованому координаційному модулю.

Агент координатор зобов’язаний забезпечувати паралельне виконання агентів, синхронізувати результати, передавати знайдені субдомени між модулями та гарантувати незалежність і послідовність збору даних. Завдяки цьому процес збирання інформації відбувається швидко, без блокувань та з урахуванням того, що одні агенти використовують результати інших.

Вебінтерфейс повинен відображати зібрану інформацію в структурованому вигляді з окремими вкладками для кожного модуля, надавати користувачу інструменти для перегляду сирих JSON-даних і показувати ключові технічні характеристики у вигляді індикаторів.

Також наявність функції побудови ієрархічного графа доменної інфраструктури у стилі Maltego забезпечує візуалізацію субдоменів, IP-адрес, ASN, країн, CA та репутаційних атрибутів. Граф повинен мати спеціальні іконки для кожного типу вузла та підтримувати інтерактивні елементи.

Система також має забезпечувати можливість завантаження скріншота ресурсу через зовнішній сервіс та коректно обробляти недоступність вебсторінки або помилки під час отримання зображення.

Підтримка високорівневого аналітичного модуля, який реалізовано через CrewAI. Уся сукупність зібраних артефактів передається у вигляді єдиного JSON-об’єкта, на основі якого агент-аналітик формує внутрішній ризиковий бал, рівень ризику, класифікацію домену, структуровані технічні докази, рекомендації та повний markdown-звіт. Аналітичний результат повинен бути інтегрований у вебінтерфейс, а також доступний у форматах JSON та Markdown для завантаження.

Нефункціональні вимоги охоплюють ключові властивості системи, які забезпечують її стабільність, ефективність та якість роботи.

Архітектура системи повинна залишатися модульною, де кожен агент є незалежним компонентом з чітко визначеним інтерфейсом і можливістю розширення без зміни існуючого коду. Масштабованість передбачає здатність системи збільшувати продуктивність за рахунок паралельного виконання агентів та можливості використання більшої кількості потоків чи зовнішніх обчислювальних ресурсів. Продуктивність визначається швидкістю відповіді на запити користувача та загальним часом аналізу домену. Для цього система використовує сумісний паралелізм через ThreadPoolExecutor та оптимізовані процедури DNS-резолвінгу.

Відповідно до поставлених задач та вимог, у подальших розділах буде детально розглянуто процес розробки системи, описано її архітектуру, функціональні можливості та результати тестування в практичних умовах.

2 РОЗРОБКА АРХІТЕКТУРНИХ РІШЕНЬ

2.1 Обґрунтування архітектурних рішень і принципів побудови системи

Побудова системи збору доменної інформації базується на концепції мультиагентної архітектури, яка забезпечує гнучкість, масштабованість і стійкість до відмов. Вибір саме такого підходу є результатом аналізу особливостей задач OSINT-розвідки в кібербезпеці, де джерела даних численні, різнорідні й постійно змінюються, а оброблення запитів часто відбувається паралельно. Традиційні монолітні або навіть класичні мікросервісні рішення не завжди здатні забезпечити достатню автономність компонентів і ефективну взаємодію між спеціалізованими модулями.

Основна ідея полягає у тому, що кожен агент системи виконує лише одну чітко визначену функцію – наприклад, збір WHOIS-даних, опитування DNS-серверів чи аналіз SSL/TLS-сертифікатів. Такий підхід забезпечує принцип єдиної відповідальності (Single Responsibility Principle) і дозволяє легко модифікувати або розширювати систему шляхом додавання нових агентів без необхідності змінювати існуючі компоненти. Кожен агент має власний життєвий цикл, набір інструментів і формат результатів, що спрощує тестування та підвищує надійність загальної системи.

Вибір мультиагентної моделі також обґрунтовується її відповідністю принципу слабкої зв’язаності. Агенти взаємодіють між собою лише через агента координатора або спільний контекст, не маючи прямої залежності один від одного. Це дозволяє мінімізувати каскадні збої, коли вихід з ладу одного елемента не впливає на працездатність решти системи. За потреби система може виконувати декілька екземплярів окремих агентів паралельно, тим самим збільшуючи пропускну здатність під час масових перевірок доменів.

Ще однією перевагою є відповідність принципу відкритості/закритості – система відкрита до розширення, але закрита до змін. Новий функціональний агент можна додати як окремий модуль, зареєструвавши його в агенті координаторі, без необхідності змінювати код інших агентів або механізми взаємодії. Це особливо важливо у сфері OSINT, де з часом з'являються нові джерела даних (наприклад, нові API-платформи чи бази сертифікатів), які можна інтегрувати без порушення існуючої логіки роботи системи.

Ключовим архітектурним принципом при побудові системи є декомпозиція складної задачі на підзадачі. Завдяки цьому досягається ефективне паралельне виконання операцій. Кожен агент обробляє свою частину даних, після чого агент координатор виконує агрегацію та узагальнення результатів.

З точки зору якості програмного забезпечення, обрана архітектура задовольняє основні нефункціональні вимоги системи. Масштабованість досягається завдяки можливості горизонтального збільшення кількості агентів і асинхронній координації. Надійність забезпечується ізоляцією модулів: відмова окремого агента не призводить до критичної помилки всієї системи. Гнучкість проявляється у можливості швидко адаптувати систему до нових форматів даних або API-джерел, тоді як безпека – у чіткому контролі каналів обміну між агентами, обмеженні доступу до зовнішніх ресурсів і валідації вхідних даних.

Таким чином, вибір мультиагентної архітектури та принципів її реалізації є обґрунтованим із погляду як функціональних, так і нефункціональних вимог системи. Вона забезпечує необхідний баланс між гнучкістю, стабільністю та розширюваністю, що робить її оптимальним рішенням для задач автоматизованого збору та аналізу доменної інформації у сфері кібербезпеки. Обрана структура гарантує масштабування системи у майбутньому – шляхом додавання нових агентів-модулів без зміни основної логіки, а також дозволяє підтримувати високу продуктивність і точність навіть за значних обсягів вхідних даних.

2.2 Архітектура мультиагентної системи

Архітектура системи збору інформації про домени реалізована за принципом мультиагентної взаємодії, де кілька спеціалізованих агентів працюють одночасно над різними аспектами аналізу. Кожен агент – автономна одиниця, що виконує своєрідну підзадачу. Такий розподіл дозволяє розбивати складне завдання на спеціалізовані складові та опрацьовувати їх паралельно (рис. 2.1).

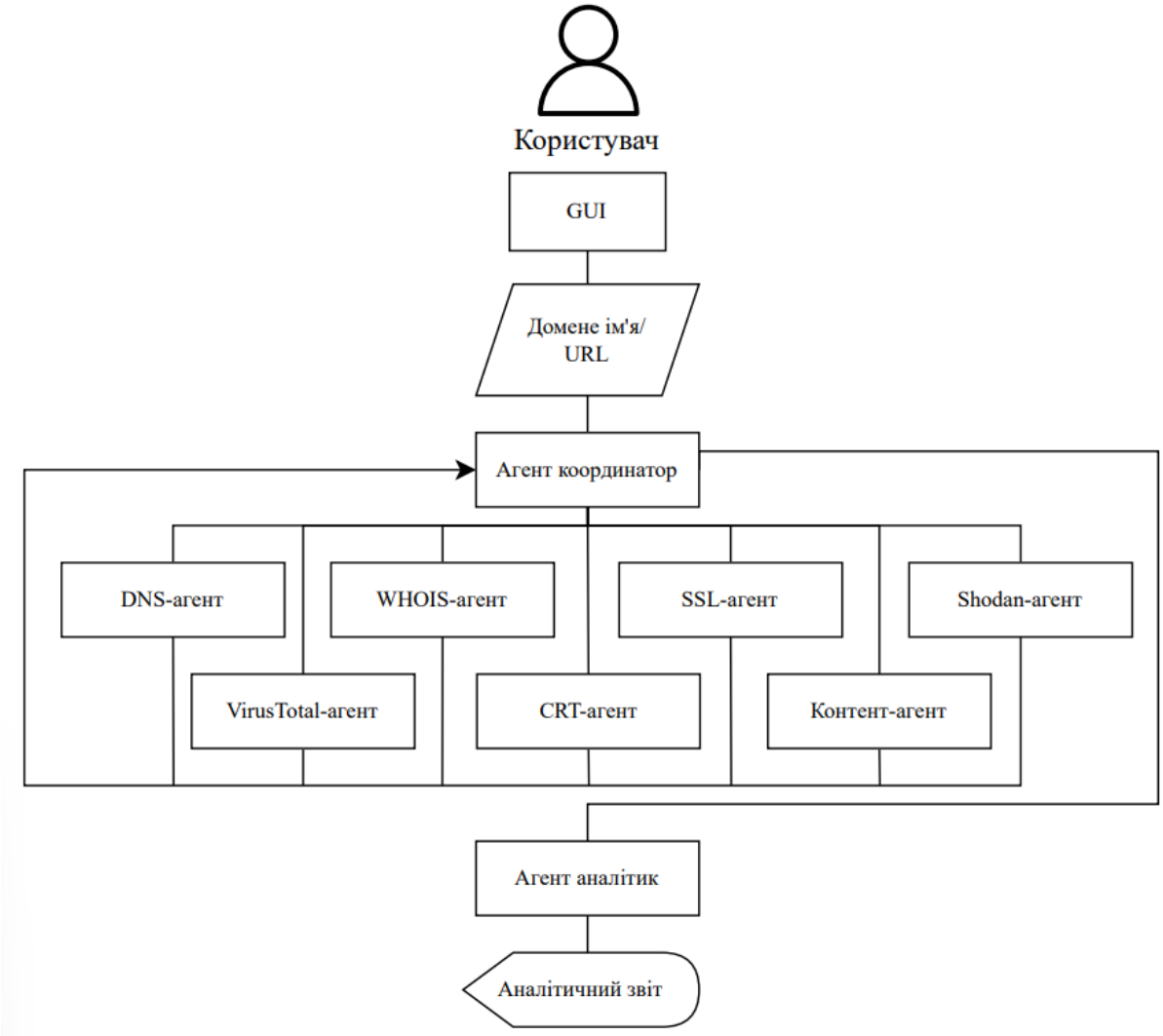


Рисунок 2.1 – Архітектура мультиагентної системи

Такий підхід забезпечує стандартизований спосіб ініціалізації, запуску та отримання результатів незалежно від конкретної спеціалізації агента. Агент

координатор взаємодіє з рештою агентами через цей інтерфейс, однаково викликаючи DNS-агент, WHOIS-агент, SSL-агент, Shodan-агент, VirusTotal-агент, CRT-агент чи Content-агент. Повернуті ними результати мають передбачувану структуру на основі єдиних моделей даних, що істотно спрощує подальшу агрегацію та аналіз.

Центральну роль у системі відіграє агент координатор, який реалізує логіку планування та керування роботою підлеглих агентів. Після отримання від користувача доменного імені або повного URL через графічний інтерфейс він формує початковий контекст завдання, визначає перелік необхідних агентів та конфігурує параметри їх запуску. На цьому етапі виконує декомпозицію загальної мети OSINT-аналізу на підзадачі, пов’язані з реєстраційною інформацією, DNS-інфраструктурою, станом SSL-сертифікатів, відкритими сервісами та репутаційними індикаторами, а також аналізом вмісту веб-сторінки.

Обробка інформації відбувається поетапно з використанням горизонтального розподілу завдань. Спочатку агент координатор ініціалізує збір базового контексту, а саме CRT-агент та інші джерела пасивного DNS отримують перелік відомих субдоменів і пов’язаних з ними записів. На основі цих даних оновлюється спільний контекст, який у подальшому використовують інші агенти. Після формування вихідного набору артефактів він паралельно запускає спеціалізовані агенти: WHOIS-агент проводить аналіз реєстраційних записів домену, DNS-агент опитує авторитетні та рекурсивні сервери для отримання записів різних типів, SSL-агент перевіряє ланцюжок сертифікатів та термін їх дії, Shodan-агент і VirusTotal-агент виконують технічний та репутаційний OSINT-аналіз, а Content-агент завантажує цільову веб-сторінку, виконує парсинг HTML та виділяє суттєвий текстовий і структурний вміст.

Паралельне виконання агентів дає змогу суттєво скоротити час аналізу та підвищити пропускну здатність системи. Кожен функціональний агент працює

незалежно, не покладаючись на внутрішню реалізацію інших компонентів, і взаємодіє з агентом координатором через строго визначений протокол обміну повідомленнями. Проміжні результати зберігаються у спільному контексті, який за потреби може повторно використовувати, ініціюючи додаткові перевірки або уточнення. Наприклад, нові субдомени, знайдені CRT-агентом, можуть бути повторно передані DNS-агенту, SSL-агенту та Content-агенту для поглибленого аналізу.

Після завершення виконання завдань спеціалізованими агентами центральний координувальний агент здійснює агрегування отриманих результатів. На цьому кроці відбувається нормалізація даних до єдиної схеми з виділенням логічних блоків WHOIS, DNS, SSL, OSINT та контент-аналізу, усунення дублікатів, уніфікація форматів дат і ідентифікаторів, а також формування консолідованого об’єкта звіту про домен. Узгоджений набір даних передається до вбудованого аналітичного агента, реалізованого на базі великої мовної моделі, який виконує високорівневу інтерпретацію технічних артефактів, виявляє нетривіальні зв’язки та формує зведений текстовий висновок з оцінкою ризиків.

2.3 Математична модель функціонування мультиагентної системи

Функціонування розроблюваної системи OSINT-аналізу доменів доцільно описати у вигляді формалізованої математичної моделі, яка відображає склад системи, характер вхідних і вихідних даних, а також логіку взаємодії між окремими програмними агентами. Така модель дозволяє однозначно інтерпретувати процес аналізу домену, забезпечує відтворюваність результатів і створює основу для подальшої оптимізації або масштабування системи.

У загальному вигляді систему можна подати як:

$$S = \langle D, A, C, R \rangle, \tag{2.1}$$

де D – вхідні дані системи;

A – множина спеціалізованих агентів;

C – агент, який виконує механізм координації та обміну даними між агентами;

R – результати аналізу домену.

Вхідні дані D у найпростішому випадку представлені одним доменним іменем:

$$\mathrm{D} = \{\mathrm{d}\}, \mathrm{d} \in \mathcal{D}, \tag{2.2}$$

де $\mathcal{D}$ – множина коректних доменних імен.

Кожний агент $a_i \in A$ є відображенням:

$$a_i : D \to R_i, \tag{2.3}$$

де $\mathrm{R_i}$ – структурований JSON-результат відповідного джерела збору інформації.

Згідно з реалізацією системи, множина агентів має вигляд:

$$A = \{a_{WHOIS}, a_{DNS}, a_{SSL}, a_{CRT}, a_{VT}, a_{SHODAN}, a_{CONTENT}\}, \tag{2.4}$$

Їх формальні функції:

1. WHOIS-агент:

$$a_{WHOIS}(d) = R_{whois}, \tag{2.5}$$

де R_{whois} – дата реєстрації, делегування, NS-сервери та інші атрибути домену.

2. DNS-агент:

$$a_{\mathrm{DNS}}(d) = R_{\mathrm{dns}} = (Z, S), \tag{2.6}$$

де Z – множина DNS-записів різних типів;

S – множина знайдених субдоменів.

3. SSL/TLS-агент:

$$a_{\mathrm{SSL}}(d) = R_{\mathrm{ssl}}, \tag{2.7}$$

де R_{ssl} – сертифікат, CN, Issuer, SAN-список, строки дії, fingerprint тощо.

4. CRT.sh-агент:

$$a_{\mathrm{CRT}}(d) = R_{\mathrm{crt}} = \{names\}, \tag{2.8}$$

де *names* – доменні імена, знайдені в історії сертифікаційних записів.

5. VirusTotal-агент:

$$a_{\mathrm{VT}}(d) = R_{\mathrm{vt}}, \tag{2.9}$$

де R_{vt} – репутація, голоси спільноти, категорії, пов’язані IP та субдомени.

6. Shodan-агент:

$$a_{\mathrm{SHODAN}}(\mathrm{d}, \mathrm{A\_records}) = R_{\mathrm{shodan}}, \tag{2.10}$$

де R_{shodan} – повна інформація яку отримав Shodan по вхідному домені.

7. Content-агент:

$$a_{\mathrm{CONTENT}}(d) = R_{\mathrm{content}}, \tag{2.11}$$

де $R_{content}$ – вміст веб-сайту домена (заголовки, посилання, контент).

Процес отримання результатів усіх спеціалізованих агентів можна представити таким чином:

$$R = C(d) = \|_{i=1}^{n} a_i(d), \tag{2.12}$$

де оператор ‖ позначає паралельне виконання у пулі потоків.

Далі ці результати об’єднуються в єдиний набір фактів:

$$F(d) = Aggregate(R), \tag{2.13}$$

Агент-аналітик виконує інтерпретацію цих фактів та формує підсумкову оцінку. Формально його роботу можна подати як відображення:

$$A_{analysis} : F(d) \to R_{final}(d), \tag{2.14}$$

де вихідний результат має вигляд:

$$R_{final}(d) \to (score, level, class, report), \tag{2.15}$$

де *score* ∈ [0,100] – числова оцінка ризику домену;

level ∈ {low, medium, high} – рівень ризику;

class ∈ {benign, suspicious, malicious} – узагальнена класифікація домену;

report – текстовий аналітичний звіт.

Зв’язок між числовою оцінкою та рівнем ризику визначається простою пороговою функцією:

$$level = \begin{cases} low, \ 0 \le score \le 30 \\ medium, \ 31 \le score \le 70 \\ high, \ 71 \le score \le 100 \end{cases} \tag{2.16}$$

Фінальний результат роботи системи визначається як:

$$\Psi(d) = A_{analysis}\ (F(d)), \tag{2.17}$$

де $\Psi(d)$ – фінальний структурований звіт за результатами комплексного аналізу домену, який представляється користувачу.

Таким чином, система формально є композиційним оператором перетворення доменного імені у багатовимірну структуровану оцінку ризику, отриману шляхом структурованої взаємодії множини агентів різної природи.

2.4 Обґрунтування вибору великої мовної моделі

У запропонованій мультиагентній системі роль великої мовної моделі є визначальною для якості узагальнення даних і коректності структурованих відповідей.

Для аналітика обрано модель класу GPT-4o як найсильнішу за контекстним міркуванням і стійкістю до «галюцинацій» у складних багатоджерельних кейсах. Вона краще тримає довгі контексти з неоднорідними фрагментами і надійніше дотримується стилю кінцевого звіту [31].

Натомість для технічних агентів, які виконують вузькі витягувальні та нормалізаційні задачі (перетворення даних у суворо визначені JSON-схеми, короткі відповіді з обмеженим контекстом), доцільно застосовувати GPT-4o mini: нижча вартість і менша латентність без помітної втрати якості на задачах типу «інструкція → структурований об’єкт» [32].

Такий поділ дозволяє зменшити загальну собівартість запитів при збереженні класу.

Порівняння з альтернативами на ринку демонструє, що різні LLM мають різні сфери застосування. Моделі на кшталт Claude 3.5 Sonnet відзначаються дуже доброю роботою з великими текстами, обережним стилем і сильним узагальненням, однак зазвичай мають співставну або вищу цінову категорію в класі «преміум» [33].

Gemini 1.5 Pro робить акцент на мультимодальності й довгому контексті, що корисно для оброблення змішаних джерел, проте на вузьких технічних

витягувальних задачах зазвичай не дає відчутної переваги порівняно з дешевшими моделями [34].

Відкриті моделі на кшталт Llama 3.1 70B Instruct чи Mistral Large цікаві можливістю локального або приватного розгортання для підвищених вимог до конфіденційності, але вимагають суттєвих інженерних зусиль для донавчання на форматах «tool-use/JSON», ретельного тюнінгу температури і penalty-параметрів та підтримки інфраструктури прискорення. На практичних задачах ці моделі добре справляються з короткими екстракціями і класифікаціями, але стабільність у «багатокроковому міркуванні по змішаних артефактах» усе ще більш прогнозована в закритих моделей верхнього класу [35, 36].

Важливою умовою інтеграції в дану систему є точність інтерфейсу. Моделі повинні надійно повертати валідний JSON без зайвого тексту для кожного агента. У цій площині моделі класу GPT дають дуже стабільну поведінку з JSON, що зменшує кількість повторних запитів і ручних виправлень.

Для технічних агентів ключовими критеріями є латентність першого байта, вартість за 1k токенів і коректність схемних відповідей. Для аналітика – якість багатокрокового міркування, узагальнення суперечливих даних і стійкість до фабрикацій фактів. Саме тому для аналітика доцільно утримуватися в класі «флагманських» моделей, тоді як для витягування DNS/WHOIS/SSL оптимально застосувати «compact-tier» із пріоритетом швидкості та ціни.

Агент Analyst використовує GPT-4o, оскільки його внесок у якість фінального висновку найбільший. Саме тут проявляється користь складних міркувань, кореляції артефактів і вироблення рекомендацій. Агенти WHOIS/DNS/SSL/OSINT за замовчуванням звертаються до GPT-4o mini як до економної для детермінованого заповнення схем. У разі довгих або хитких промптів, помилок у схемах чи потреби в тонкій інтерпретації вони можуть ескалувати запит до аналітичної моделі.

Порівняльний аналіз характеристик великих мовних моделей, релевантних для запропонованої мультиагентної системи, наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика великих мовних моделей

Модель	Input $/1k	Output $/1k	Підписки / плани	Плюси	Мінуси
GPT-5	0.00125	0.01000	ChatGPT Plus/Pro, API PAYG/Enterprise	флагманські міркування, великий контекст, кеш 90%	вищі вимоги до лімітів/квот у проді, дорожчий за «мікро» на масових потоках.
GPT-5 mini	0.00025	0.00200	ті самі	дешевий і швидкий для екстракцій/JSON, добре масштабується	не оптимальний для найскладнішої аналітики.
GPT-4o	0.00250	0.01000	ChatGPT Plus/Pro, API	стабільний function-calling/JSON, якісна мультимодальність	дорожчий за GPT-5 mini/DeepSeek/Qwen на масових запусках.
GPT-4o mini	0.00015	0.00060	ті самі	мінімальна ціна/латентність для технічних агентів	слабше міркування порівняно з флагманами.
Claude Sonnet 4.5	0.00300	0.01500	Claude Pro/Max, API	дуже сильний на довгих контекстах, акуратний стиль	«преміум»-клас за ціною; інакші ліміти/кеш-політика.
Gemini 2.5	~0.00030	~0.00250	Google AI/Vertex: AI Pro/Ultra, API	довгий контекст, мультимодальність, вигідні batch-тарифи	матриця тарифів складна; ціни залежать від режиму/моделі.
Llama 4	інфра-витрати	інфра-витрати	self-host/хостинг	приватність/контроль, гнучкість тюнінгу	потрібні GPU/MLOps; стабільність JSON залежить від тюнінгу.
DeepSeek	0.00028	0.00042	API PAYG	дуже низькі тарифи; «reasoning» доступний; кеш-знижки	інша семантика «thinking»/tools; різні обмеження за часом.
Qwen	~0.000022 –0.000173	~0.000216 –0.001721	Alibaba Cloud Model Studio	один із найдешевших PAYG; кеш контексту	тири залежать від розміру промпта; нюанси інтеграції

У підсумку система зберігає високу якість саме там, де від неї залежить довіра до висновків, і оптимізує витрати на масових операціях, де цінність вимірюється швидкістю та передбачуваністю JSON-виводу, а не стилістичною довершеністю тексту.

2.5 Проєктування агентів, їх ролей та зв’язків

У сучасних системах автоматизації, зокрема у сфері кібербезпеки та OSINT-аналізу, мультиагентний підхід забезпечує модульність, масштабованість і керовану взаємодію між складними компонентами. У такій архітектурі кожен агент розглядається як інтелектуальна сутність із чітко визначеною роллю, метою, параметрами поведінки та власним формалізованим текстовим промптом, який регламентує правила опрацювання даних.

Загальна структурна схема ШІ-агента подана на рисунку 2.2.

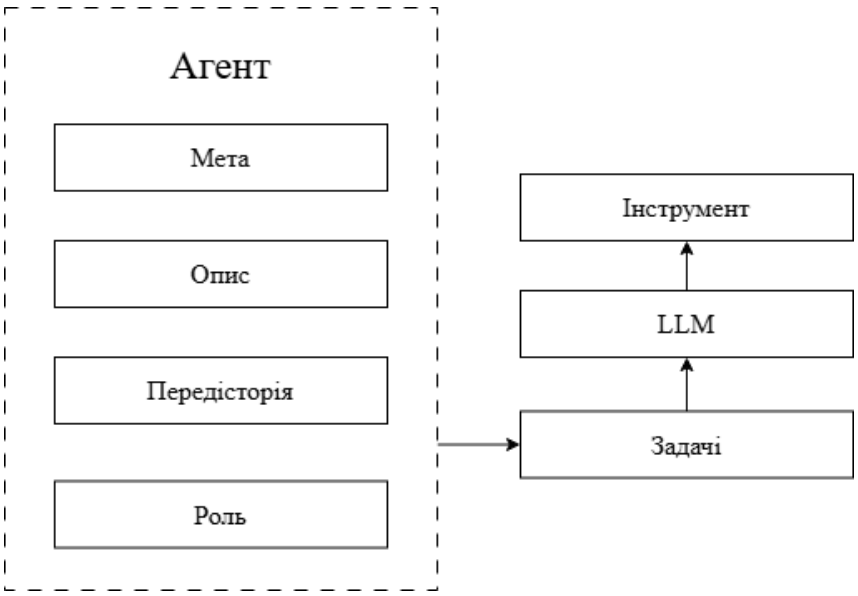


Рисунок 2.2 – Загальна структурна схема ШІ-агента

Процес створення агента передбачає визначення функціонального призначення, логіки поведінки та інструментів доступу до даних. Кожен агент формується з набору логічних компонентів, які визначають його поведінку,

спеціалізацію та спосіб взаємодії з зовнішнім середовищем і великими мовними моделями [37].

Призначення основних компонентів агента наведено в таблиці 2.2.

Таблиця 2.2 – Компоненти ШІ-агента та їх призначення

Компонент	Призначення
Назва агента	Унікальний ідентифікатор агента в межах мультиагентної системи, який використовується координатором для адресації та керування агентом.
Роль (role)	Визначає функціональну спеціалізацію агента (наприклад, WHOIS-аналітик, DNS-спеціаліст, аналітик безпеки) та окреслює його місце в загальній архітектурі.
Мета (goal)	Формалізує очікуваний результат діяльності агента у вигляді конкретного завдання або цілі, якої він має досягти під час виконання.
Опис (description)	Містить деталізований опис задач, обмежень та правил роботи агента, включаючи вимоги до формату вихідних даних.
Передісторія (backstory)	Формує експертний контекст агента, що моделює його досвід, знання та стиль мислення, необхідні для коректної інтерпретації даних.
Задачі (tasks)	Конкретні операції або інструкції, які агент повинен виконати у процесі аналізу, зазвичай прив’язані до одного домену або набору артефактів.
LLM (велика мовна модель)	Обчислювальне ядро агента, що здійснює інтерпретацію промптів, аналіз вхідних даних та генерацію структурованих відповідей.
Інструменти (tools)	Програмні модулі або API, до яких агент має доступ для отримання зовнішніх даних (WHOIS, DNS, SSL, OSINT-сервіси тощо).

Важливою особливістю реалізації системи є те, що всі логічні компоненти LLM-агентів задаються у вигляді текстових промптів. До таких компонентів належать роль, мета, передісторія, опис задач та вимоги до формату результату. Саме промпт визначає поведінкову модель агента та регламентує спосіб обробки інформації.

Агент у фреймворку CrewAI не містить жорстко закодованої логіки аналізу, а функціонує як параметризована інтелектуальна сутність, поведінка якої

формується через набір інструкцій природною мовою. Це забезпечує високу гнучкість архітектури, спрощує модифікацію агентів і дозволяє додавати нові спеціалізовані компоненти без зміни програмного коду системи.

Таким чином, описана структура ШІ-агента є універсальною концептуальною моделлю, яка визначає спосіб формування та поведінки інтелектуальних компонентів у мультиагентних системах. На її основі можливе проєктування конкретних агентів, орієнтованих на виконання прикладних задач у визначеній предметній області. У межах розроблюваної системи OSINT-аналізу доменів дана модель була адаптована до ієрархічної архітектури взаємодії агентів, де центральне місце займає агент-координатор, а спеціалізовані агенти реалізують окремі технічні підпроцеси збору та первинної обробки даних.

У межах системи OSINT-аналізу доменів застосовано ієрархічну модель взаємодії компонентів: агент-координатор виконує функції менеджера, а спеціалізовані агенти забезпечують технічний збір даних. Агент-координатор планує виконання підпроцесів WHOIS, DNS, SSL/TLS і OSINT-аналізу, передає параметри задач, контролює коректність відповідей і агрегує часткові результати у єдину структуру. Його основна місія полягає у забезпеченні точності, повноти й уніфікованості формату вихідних даних, необхідних для автоматизованого формування висновків та відображення результатів у користувацькому інтерфейсі.

Промпт для визначення мети агента-координатора:

«Організувати послідовне та узгоджене виконання підзадач WHOIS, DNS, SSL і OSINT-агентами. Забезпечити збір їх результатів, перевірку цілісності, формування єдиного інтегрованого JSON-звіту для подальшого аналізу кіберзагроз.»

Промпт для визначення передісторії агента-координатора:

«Координатор мультиагентної системи, який має навички технічного керівництва. Планує виконання завдань, контролює коректність повернених

результатів і об'єднує їх у єдину структуру. Його мета – забезпечити точність, повноту і єдність формату результату.»

Агент WHOIS відповідає за отримання й нормалізацію реєстраційної інформації домену. У межах своєї ролі він звертається до WHOIS-реєстрів, виділяє ключові атрибути – реєстратора, дати створення та закінчення делегування, контакти власника (за наявності), можливі статуси домену – і повертає їх у структурі WhoisResult.

Промпт для визначення мети WHOIS-агента:

«Отримати повну, валідну та структуровану інформацію WHOIS про домен у форматі JSON, включаючи дані про реєстратора, дати створення/закінчення, стани домену, контактну інформацію (якщо доступна) та технічні поля.»

Промпт для визначення передісторії WHOIS-агента:

«Досвідчений аналітик доменних реєстрів, який розуміє специфіку WHOIS-запитів для різних TLD та вміє інтерпретувати нетипові формати відповідей. Знає, що частина полів може бути прихована, а деякі – подані в іншому кодуванні або структурі. Його мета – побудувати максимально точний JSON-об'єкт без довільних пояснень чи тексту.»

Опис завдання, яке виконує WHOIS-агента:

"Виконай повний WHOIS-запит для домену {domain}. Використай інструмент whois_lookup, оброби всі отримані дані та поверни результат у вигляді строго структурованого JSON-об'єкта, що відповідає моделі WhoisResult. Не додавай текстових коментарів, описів або пояснень. Якщо певні поля відсутні – поверни null."

Промпт для визначення результату завдання WHOIS-агента:

"JSON-об'єкт типу WhoisResult, який містить повний набір доступних даних WHOIS (домен, реєстратор, creation_date, expiry_date, status, emails, name_servers тощо) без додаткових текстових вставок."

Така формулювання не лише дисциплінує вихідні дані, а й мінімізує похибки при подальшому серіалізованому обробленні, зокрема під час інтеграції в агрегований результат. Практичний сенс цього етапу – оцінка віку та легітимності домену, виявлення типових індикаторів ризику (свіжа реєстрація, приховані дані реєстранта, нетипові зміни реєстратора тощо).

Агент DNS формує інфраструктурний профіль домену. Його завдання – збирання широкого спектра DNS-записів (A/AAAA, NS, MX, TXT, CNAME, SOA) та помірковане виявлення субдоменів пасивними методами і легким брутфорсом. На виході – уніфікована структура, придатна для кореляції з іншими модулями (зіставлення MX із TLS конфігураціями поштових вузлів, аналіз SPF/DMARC/ DKIM-політик, картографія субдоменів як потенційних точок експозиції сервісів).

Промпт для визначення мети DNS-агента:

«Побудувати повну та достовірну картину DNS-структури досліджуваного домену, включаючи A, AAAA, NS, MX, TXT, CNAME-записи та знайдені субдомени. Усі результати повернути у форматі JSON.»

Промпт для визначення передісторії DNS-агента:

«Аналітик DNS-інфраструктури, який розуміє резолюцію, TTL, делегування та особливості кешування. Уміє відрізняти актуальні записи від застарілих, інтегрує результати активного й пасивного збору. Повертає тільки фактичні дані інструментів – без довільних інтерпретацій.»

Опис завдання, яке виконує DNS-агента:

«Для домену {domain} здійсни повне отримання DNS-записів основних типів (A, AAAA, MX, NS, TXT, CNAME, SOA) за допомогою інструмента dns_enumeration. За можливості виконай пасивне виявлення субдоменів (і, за потреби, легкий брутфорс у безпечних межах). Результат подай як строго структурований JSON, що відповідає моделі DNSResult. Не додавай текстових коментарів або пояснень. Відсутні поля – null, невиявлені колекції – порожні масиви.»

Промпт для визначення результату завдання DNS-агента:

«JSON-об'єкт типу DNSResult, що містить усі виявлені записи DNS для домену та знайдені субдомени (якщо доступні), без додаткових текстових вставок.»

Агент TLS/SSL зосереджується на безпечності транспортного шару та атрибутах сертифікатів. Його функція – встановити TLS-з'єднання із сервером на порту 443/TCP, зчитати поточний сертифікат (X.509), вилучити й нормалізувати такі поля: Issuer, Subject (Common Name), Subject Alternative Names (SAN), терміни чинності (not_before / not_after), серійний номер, криптографічні відбитки (SHA-1, SHA-256), алгоритми підпису та інші релевантні метадані.

Промпт для визначення мети SSL-агента:

«Отримати з сервера цільового домену дійсний SSL/TLS-сертифікат, розібрати його структуру та повернути усі релевантні поля у форматі JSON: subject, issuer, serial_number, fingerprints, SAN, not_before, not_after тощо.»

Промпт для визначення передісторії SSL-агента:

«Ти – криптоаналітик, який спеціалізується на сертифікатах безпеки. Знаєш структуру X.509, розумієш ланцюг довіри, вимоги сучасних браузерів і TLS-стеків, вмієш ідентифікувати ознаки прострочених, самопідписаних або неправильно виданих сертифікатів. Повертаєш тільки технічні дані без інтерпретацій у вільному тексті.»

Опис завдання, яке виконує SSL-агент:

«Встанови TLS-з'єднання з {domain}:443 (за можливості – перевірити SNI). Зчитай сертифікат (первинний сертифікат, не обов'язково повний ланцюг), розпізнай і поверни структуровані поля: subject (CN та інші RDN), issuer, serial_number, not_before, not_after, san (масив), signature_algorithm, public_key_algorithm, key_size, fingerprints (SHA1, SHA256), ocsp_urls, crl_urls (якщо є). Використай інструмент ssl_tool для отримання сирих даних, нормалізуй дати у

ISO-8601, всі невиявлені поля – null. Поверни тільки JSON, що відповідає моделі SSLResult, без додаткових пояснень чи текстових коментарів.»

Промпт для визначення результату завдання SSL-агента:

«JSON-об'єкт типу SSLResult, який містить усі ключові атрибути SSL/TLS-сертифіката (subject, issuer, SAN, not_before, not_after, fingerprints, serial_number тощо) у стандартизованому вигляді, без додаткових коментарів.»

Інші агенти мультиагентної системи реалізовані за аналогічним принципом. Кожен із них має чітко визначену роль, формалізовану мету, внутрішню передісторію, індивідуальний промпт із вимогами до поведінки та суворий контракт вихідних даних у форматі JSON. Відмінності між ними полягають виключно у спеціалізації й типах інформації, яку вони збирають.

Окремої уваги потребує агент аналітичного рівня, оскільки його промпт істотно відрізняється від промптів агентів збору даних.

Особливості промпта аналітичного агента зумовлені його ключовою функцією – перетворення структурованих технічних даних на узагальнену оцінку ризиків. На відміну від інших агентів, що повертають виключно JSON-об'єкти з фактами, аналітичний агент отримує агрегований результат роботи всієї системи та формує фінальний висновок з чітко визначеною структурою.

У промпті задаються правила оцінювання репутації домену, визначення ризикового рівня, формування класифікації та створення узгодженого Markdown-звіту українською мовою. Додатково передбачено оброблення блоку content_analysis, що дозволяє враховувати ознаки фішингових і шахрайських вебресурсів на основі аналізу наповнення сайту. Повний текст промпта містить формальні вимоги до полів результату (domain, risk_score, risk_level, classification, analysis, …), а також правила формування підсумку, аргументації та рекомендацій, які гарантують відтворюваність і однозначність отриманих висновків. З огляду на обсяг і деталізацію інструкції, повний варіант наведено в Додатку Б.

З точки зору архітектурних властивостей запропоноване проєктування забезпечує низький рівень зв’язності та високу взаємодію модулів. Додавання нового агента (наприклад, для репутаційних фідів, витоків облікових даних чи активного сканування за дозволом) потребує лише визначення ролі, чіткого промпта з очікуваним JSON-виходом і реєстрації цього агента в координатора. Водночас базові агенти залишаються незмінними, а наявна схема зв’язків продовжує працювати без порушення інваріантів.

2.6 Розробка алгоритмів роботи системи

Алгоритмічне функціонування мультиагентної системи базується на трьох ключових процесах, що визначають порядок обробки домену, структуру отримання технічних артефактів та інтеграцію даних у фінальний результат. В основі системи лежить алгоритм роботи координатора, який організовує взаємодію агентів, алгоритм розширеного DNS-аналізу, що забезпечує активний і пасивний збір даних про мережеву інфраструктуру, та алгоритм аналізу веб-вмісту, спрямований на семантичне дослідження сторінки домену.

Алгоритм роботи координатора визначає загальну логіку обробки запиту користувача. Після отримання доменного імені координатор створює базову структуру для зберігання результатів аналізу та переходить до пасивного пошуку субдоменів за допомогою сервісу crt.sh. Ця операція є попередньою та необхідною для того, щоб DNS-агент міг у подальшому порівнювати активні дані з уже відомими сертифікаційними іменами. Пасивні субдомени передаються у внутрішнє сховище, доступне іншим агентам.

Далі координатор ініціює паралельне виконання основних агентів системи: WHOIS, DNS, SSL і Content. Паралельність досягається використанням пулу потоків, завдяки чому незалежні операції виконуються одночасно, зменшуючи загальну затримку.

Після завершення їх роботи координатор аналізує отримані дані, зокрема витягує IP-адреси з A-записів DNS і використовує їх для виконання запиту до Shodan-агента. У такий спосіб забезпечується дослідження портів, мережевих сервісів, банерів і метаданих інфраструктури.

Наступним кроком викликається повноцінний аналіз сертифікаційних даних через CRT-агент, після чого координатор здійснює репутаційний запит до сервісу VirusTotal.

Усі отримані результати об’єднуються у структуру DomainResult, що включає як дані кожного агента, так і інформацію про можливі помилки (рис. 2.3).

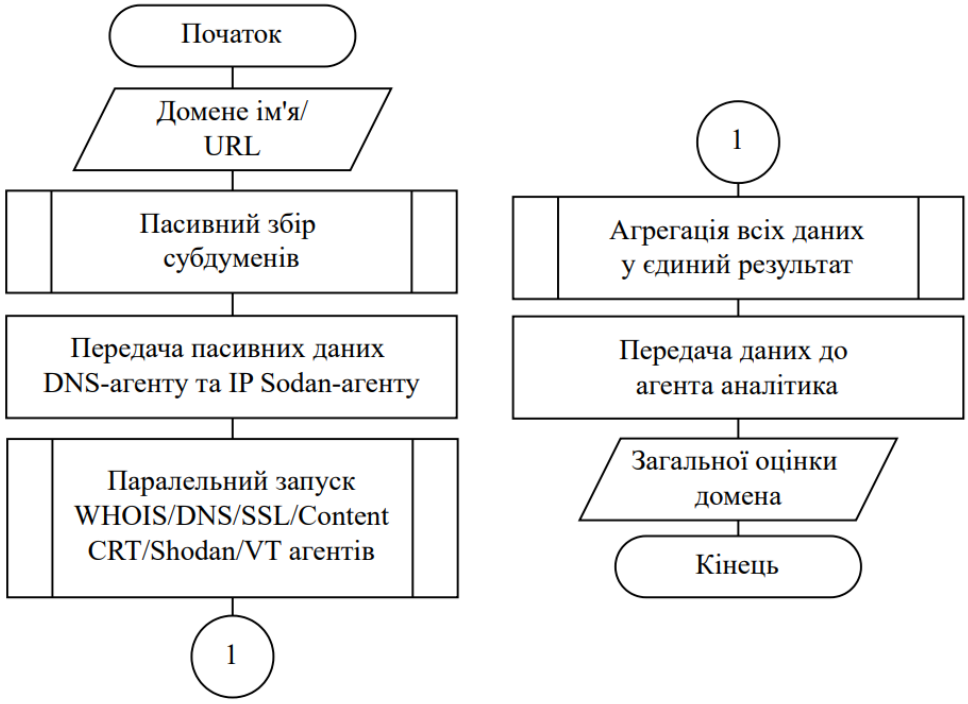


Рисунок 2.3 – Блок-схема алгоритму роботи координатора мультиагентної системи

Важливою складовою системи є модуль розширеного DNS-аналізу, що реалізується DNS-агентом. На початковому етапі агент налаштовує DNS-резолвер із використанням публічних серверів, що забезпечує незалежність аналізу від інфраструктури провайдера.

Далі виконується активне опитування домену щодо основних типів ресурсних записів – A, AAAA, NS, MX, TXT, CNAME, SOA та CAA. Це дозволяє сформувати загальне уявлення про адресу, поштову та службову конфігурацію домену.

Окрему роль відіграє інтерпретація TXT-записів, де можуть міститися політики SPF, а також визначення DMARC-записів, що дає змогу оцінити рівень захищеності електронної пошти та автентифікації відправників. Після цього агент перевіряє наявність DNSSEC шляхом спроби отримання ключів DNSKEY та записів DS.

Пасивні субдомени, отримані координатором із crt.sh, інтегруються у внутрішню базу потенційних назв. Після цього запускається активний пошук субдоменів шляхом багатопотокового перебору словникових префіксів.

У цьому процесі кожен варіант формується як окремий домен другого або третього рівня, після чого здійснюється DNS-запит і фіксація всіх позитивних відповідей. Для обмеження навантаження використовується механізм регулювання кількості потоків відповідно до конфігураційних параметрів системи.

Після завершення обох частин алгоритму – активної та пасивної – агент об’єднує результати, видаляє дублікати, сортує субдомени за алфавітом та формує структуру DNSResult.

Блок-схема алгоритму розширеного DNS-аналізу, яка відображає модель цього процесу, включаючи етапи формування резолвера, обробки DNSSEC, об’єднання активних і пасивних даних та паралельного виконання підпроцесів bruteforce зображено на рис. 2.3.

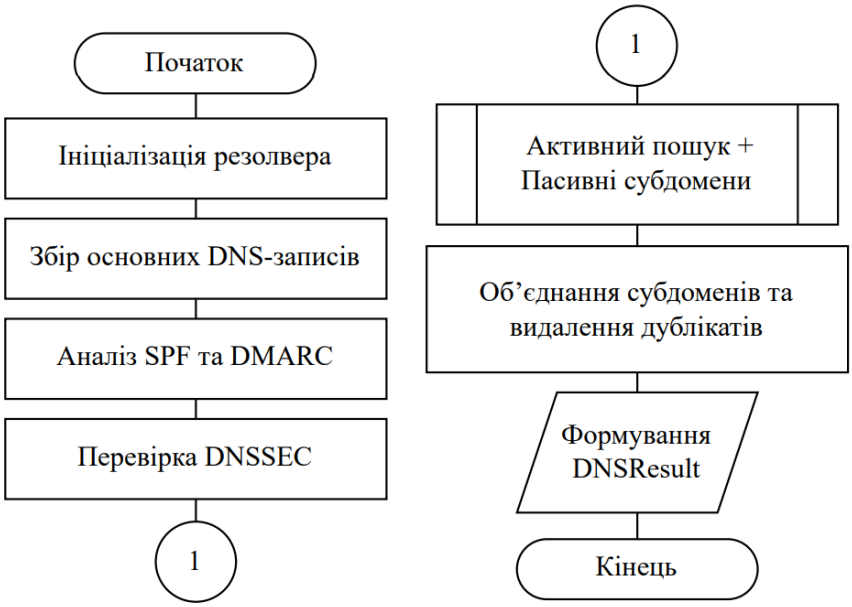


Рисунок 2.4 – Блок-схема алгоритму розширеного DNS-аналізу

Третім компонентом є модуль аналізу вмісту вебсайту, що виконується Content-агентом. Алгоритм його роботи спрямований на дослідження HTML-структури домену та визначення інформаційних характеристик сторінки. На першому етапі агент формує коректний URL, враховуючи, що користувач може вводити як повну адресу із шляхом, так і лише доменне ім'я. Система самостійно додає протокол HTTPS за замовчуванням або використовує протокол, що вже вказаний у введенні.

Далі відбувається спроба завантаження ресурсу з перевіркою SSL-сертифіката. Якщо перевірка завершується помилкою через недійсний або неправильно виданий сертифікат, алгоритм переходить на альтернативну гілку, у якій виконується запит без верифікації. Такий підхід дозволяє отримати контент навіть для неправильно налаштованих вебресурсів, що є суттєвим для OSINT-аналізу.

Після отримання HTML-коду виконується розбір документа. Агент вилучає заголовок сторінки, визначає мову документа, аналізує метадані, такі як опис сторінки, ключові слова та open-graph-дані. Далі з HTML-документа видаляються

скрипти, стилі та службові елементи, після чого формується основний текстовий зміст сторінки. Окремо обробляються заголовки H1–H3, що дозволяє оцінити структуру та основні тематичні блоки сторінки. Проводиться збір гіперпосилань і їхнє очищення від некоректних або неінформативних значень. З метою оптимізації подальшої роботи довгі HTML-документи та текстові блоки обрізаються до заданої довжини.

Заключним етапом є формування об’єкта ContentResult, який містить структурований опис веб-сторінки, включаючи її заголовок, мову, метадані, заголовки та основний текст (рис. 2.5).

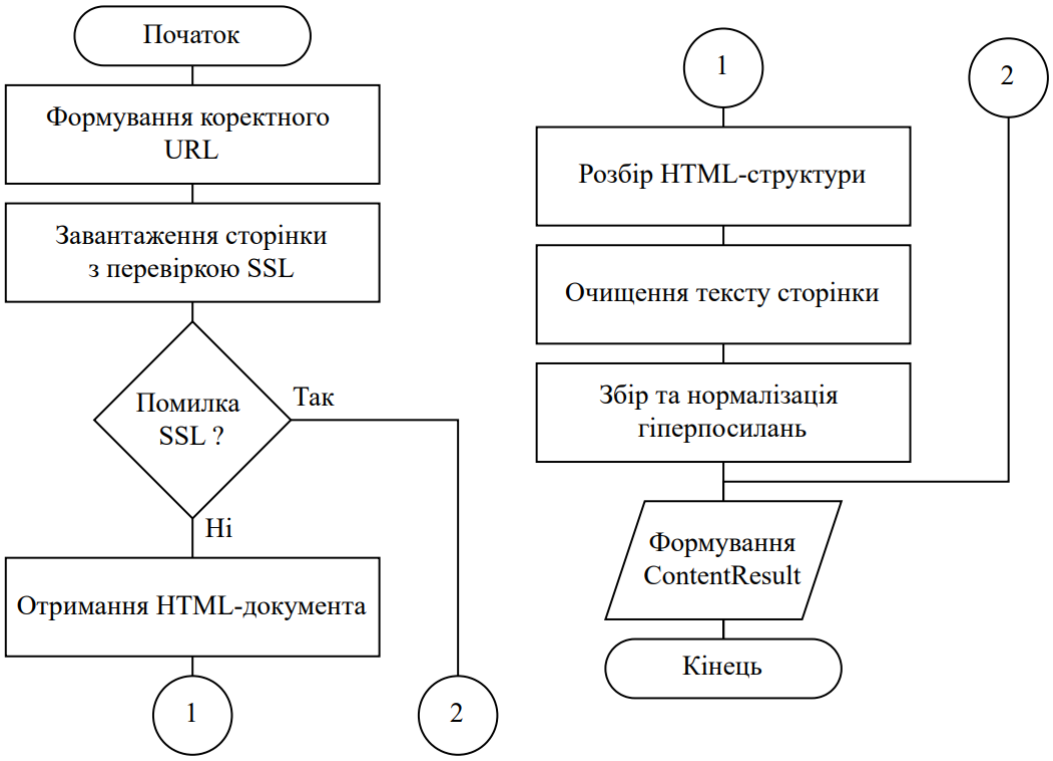


Рисунок 2.5 – Блок-схема алгоритму аналізу веб-контенту

Алгоритм роботи WHOIS-агента призначений для автоматизованого отримання та нормалізації реєстраційних даних доменного імені з урахуванням різноманіття форматів WHOIS-відповідей і регіональних особливостей доменних зон (рис. 2.6).

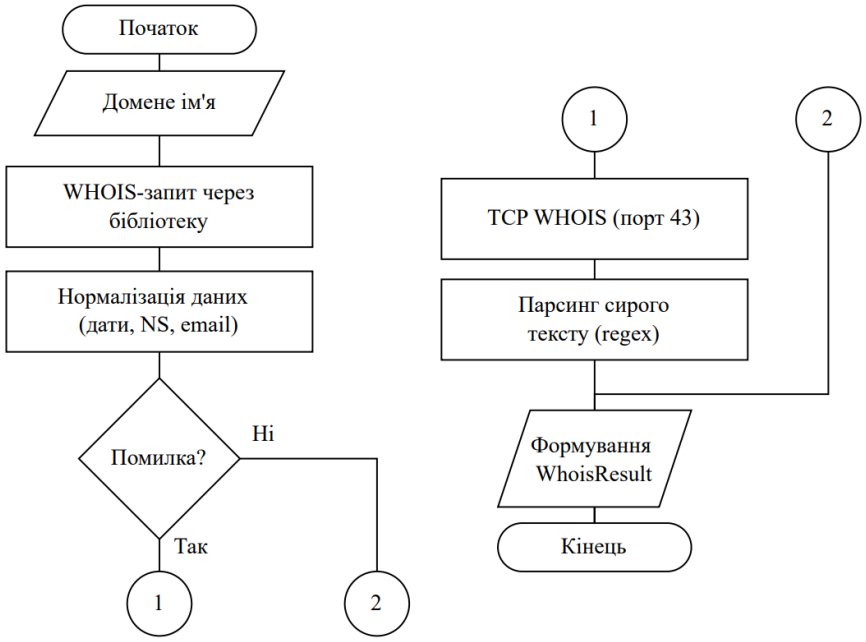


Рисунок 2.6 – Блок-схема алгоритму роботи WHOIS-агента

На першому етапі агент виконує стандартний WHOIS-запит із використанням спеціалізованої бібліотеки. Отримані дані піддаються попередній обробці, а саме дати створення, оновлення та завершення делегування домену приводяться до стандарту ISO 8601, а множинні поля, зокрема сервери імен, статуси та контактні електронні адреси, нормалізуються у вигляді списків.

У разі виникнення помилки або отримання некоректної відповіді алгоритм переходить до резервного механізму виконання. Подальший аналіз базується на регулярних виразах і дозволяє виділити ключові поля реєстратора, дат та серверів імен, зокрема для національних реєстрів.

Після успішного завершення одного з етапів формується об’єкт WhoisResult, який містить як структуровані значення, так і сирі дані для забезпечення трасованості аналізу.

Представлені блок-схеми дозволяють однозначно відтворити роботу кожного алгоритму та слугують базою для переходу від концептуальної моделі до практичної реалізації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Обґрунтування вибору технологічного стеку

Вибір технологічного стеку для розроблення мультиагентної системи OSINT-аналізу доменів був визначений вимогами до гнучкості, швидкої інтеграції з мережевими сервісами та стабільності роботи.

Основною мовою реалізації обрано Python. Це рішення базується на порівняльному аналізі з альтернативними мовами, зокрема Java, Go та C++ [38-41].

Мова Java забезпечує високу стабільність і масштабованість, проте її використання передбачає значно більший обсяг шаблонного коду та складнішу інтеграцію з низкою OSINT-інструментів, які орієнтовані саме на Python-екосистему . Go характеризується високою швидкодією та зручною моделлю конкурентності, однак має обмежену кількість готових бібліотек для OSINT-аналізу, зокрема для роботи з WHOIS, сертифікатами X.509 та спеціалізованими сервісами розвідки. C++, попри максимальну продуктивність, є недоцільним для систем, де час виконання здебільшого визначається затримками зовнішніх мережевих запитів, а не обчисленнями.

Python, навпаки, забезпечує високий рівень абстракції, велику кількість зрілих бібліотек для мережевої взаємодії та OSINT-аналізу, а також швидке прототипування. Це дозволило зосередитися на логіці мультиагентної взаємодії, не витрачаючи ресурси на розроблення низькорівневих компонентів. Таким чином, вибір Python є оптимальним компромісом між швидкістю розроблення, функціональністю та достатньою продуктивністю.

У процесі реалізації використано середовище PyCharm, яке надало розвинуті засоби навігації по коду, статичного аналізу, рефакторингу та зневадження. Альтернативами могли бути Visual Studio Code або Neovim, однак PyCharm

забезпечив глибшу інтеграцію з Python-типізацією та Pydantic-моделями, що є критичним для проєкту з великою кількістю агентів і структур даних [42-44].

Інтерфейс користувача реалізовано за допомогою Streamlit. Альтернативними підходами були використання класичних веб-фреймворків, таких як Django або Flask у поєднанні з окремим фронтендом (React, Vue). Проте такі рішення суттєво збільшили б складність проєкту та час розроблення. Streamlit дозволив реалізувати веб-застосунок без розділення на бекенд і фронтенд, зберігши єдину Python-кодову базу та забезпечивши швидке створення інтерактивних інструментів для аналітика. Обмеження дизайну Streamlit не є критичними, оскільки система орієнтована насамперед на функціональність, а не на візуальну складову [45-49].

Керування усіма агентами виконується за допомогою CrewAI. Вибір цього фреймворку здійснено шляхом порівняння з альтернативами, такими як LangChain, AutoGen [50-52].

LangChain орієнтований переважно на побудову ланцюжків викликів мовних моделей і менш придатний для чітко структурованих мультиагентних систем із фіксованими ролями. AutoGen забезпечує гнучку агентну взаємодію, однак є складнішим у налаштуванні та менш придатним для детермінованих сценаріїв збору OSINT-даних. Використання «чистого» asyncio вимагало б ручного керування потоками, контекстом і залежностями між агентами, що значно ускладнило б архітектуру системи [53].

CrewAI, у свою чергу, надав можливість формалізувати ролі агентів, визначити їхні цілі та інструменти, а також організувати паралельне виконання завдань у межах єдиного сценарію. Це дозволило інтегрувати спеціалізовані агенти WHOIS, DNS, SSL, Shodan, VirusTotal та аналізу контенту в уніфікований процес збору та агрегації даних без надлишкової складності керування потоками.

Для моделювання структур даних застосовано Pydantic, що забезпечує контроль типів та цілісність результатів, отриманих від різних агентів. Це полегшило агрегацію даних у координаторі та передавання їх у модуль фінального аналізу [54].

У межах мережевої взаємодії використано бібліотеку Requests для HTTP-запитів, що забезпечила простий і надійний спосіб взаємодії з API зовнішніх платформ [55]. Для DNS-резолвингу застосовано dnspython, який дозволив працювати з усіма основними типами DNS-записів та виконувати брутфорс субдоменів [56]. Аналіз TLS-сертифікатів виконано через модулі socket та ssl у поєднанні з cryptography, які забезпечили пряме підключення до сервера і коректний розбір сертифікатів X.509 [57-59].

Інтеграція зі сторонніми OSINT-платформами охоплювала Shodan SDK та REST API VirusTotal. Використання офіційної бібліотеки Shodan спростило доступ до інфраструктурних даних, тоді як прямі HTTP-запити до VirusTotal дозволили повністю контролювати структуру відповідей [60, 61].

Таким чином, обраний технологічний стек – Python у поєднанні зі Streamlit, CrewAI та спеціалізованими бібліотеками забезпечує баланс між швидкістю розроблення, архітектурною гнучкістю та функціональністю, що є критичним для експериментальної мультиагентної системи OSINT-аналізу доменів.

3.2 Реалізація програмних модулів та взаємодія компонентів

Практична реалізація прототипу системи базується на модульній архітектурі з головним координатором і набором спеціалізованих агентів. Усі компоненти реалізовано мовою Python, а для інтерфейсу користувача використано веб-фреймворк Streamlit. Кожен агент інкапсулює логіку роботи з окремим джерелом даних (WHOIS, DNS, SSL/TLS, crt.sh, Shodan, VirusTotal, контент сторінки), повертаючи результат у вигляді Pydantic-моделі. Узагальнення результатів

виконується у моделі DomainResult, яка містить поля для всіх типів агентів та список можливих помилок.

Центральним елементом є клас Coordinator, метод run_for_domain() якого створює екземпляри всіх агентів, організовує їх паралельний запуск та агрегує результати у об’єкт DomainResult.

На першому кроці координатор отримує пасивні субдомени через CrtAgent.passive_subdomains() і передає їх у контекст DNSAgent через механізм спільних даних (set_shared("passive_subs", …)). Далі, за допомогою ThreadPoolExecutor, паралельно запускаються агенти WHOIS, DNS, SSL та Content (рис. 3.1).

```
def run_for_domain(self, domain: str) -> DomainResult:  2 usage
    result = DomainResult(domain=domain)

    whois_agent = WhoisAgent()
    dns_agent = DNSAgent()
    ssl_agent = SSLAgent()
    crt_agent = CrtAgent()
    shodan_agent = ShodanAgent()
    virustotal_agent = VirusTotalAgent()
    content_agent = ContentAgent()

    try:
        passive_subs = crt_agent.passive_subdomains(domain)
    except Exception as e:
        result.add_error( agent: "crt.passive_subdomains", e)
        passive_subs = []

    dns_agent.set_shared("passive_subs", passive_subs)
```

Рисунок 3.1 – Вигляд фрагменту коду роботи координатора

Після завершення їх результати використовуються для формування списку IP-адрес, які передаються ShodanAgent як спільний параметр a_records. Окремо викликаються повноцінні методи run() для CrtAgent та VirusTotalAgent. Наприкінці координатор заповнює відповідні поля DomainResult (whois, dns, ssl, crt, shodan, virustotal, content) та, за потреби, додає записи про помилки.

Відповідний фрагмент коду подано на рисунку 3.2.

```
try:
    vt_res = virustotal_agent.run(domain)
except Exception as e:
    result.add_error( agent: "virustotal", e)
    vt_res = None

result.whois = intermediate.get("whois")
result.dns = intermediate.get("dns")
result.ssl = intermediate.get("ssl")
result.crt = crt_res
result.shodan = shodan_res
result.virustotal = vt_res
result.content = intermediate.get("content")

return result
```

Рисунок 3.2 – Вигляд фрагменту коду фінального формування результатів аналізу

Агент WHOIS реалізовано як клас WhoisAgent, який спочатку намагається отримати дані через стандартну бібліотеку python-whois. Оскільки WHOIS-відповіді можуть містити дати у різних форматах та колекції значень, у коді передбачено допоміжну функцію _to_iso(), яка уніфікує часові поля до ISO-формату або, у крайньому разі, повертає вихідний рядок.

Крім того, нормалізуються списки серверів імен, статусів та email-адрес (перетворення до списків, приведення до нижнього регістру, видалення дублікатів). Усі «сирі» дані зберігаються у полі raw моделі WhoisResult для подальшої діагностики. Якщо стандартний виклик WHOIS не вдається (наприклад, для доменів українських реєстрів), агент переходить до резервної гілки, а саме виконує сирі запити до серверів whois.ua, whois.net.ua, whois.uanic.net через функцію _raw_whois_query() і аналізує текстову відповідь спеціалізованим парсером _parse_uanic_text().

Результат у всіх випадках повертається у вигляді моделі WhoisResult.

Ця логіка демонструється на рисунку 3.3.

```
emails: List[str] = []
em = data.get("emails")
if isinstance(em, (list, set, tuple)):
    emails = sorted({str(x).lower() for x in em})
elif isinstance(em, str):
    emails = [em.lower()]

return WhoisResult(
    registrar=data.get("registrar"),
    creation_date=_to_iso(data.get("creation_date")),
    expiration_date=_to_iso(data.get("expiration_date")),
    updated_date=_to_iso(data.get("updated_date")),
    name_servers=name_servers,
    statuses=statuses,
    emails=emails,
    raw={k: (str(v) if not isinstance(v, (dict, list)) else v) for k,
)
```

Рисунок 3.3 – Вигляд фрагменту коду нормалізації та формування результату WHOIS

DNSAgent відповідає за побудову повної DNS-картини домену. У конструкторі створюється окремий екземпляр dns.resolver. Resolver(configure=False) з фіксованими публічними резолверами (Google, Cloudflare, Quad9), тайм-аутами та власним словником субдоменів, який або завантажується з файлу SUBDOMAIN_WORDLIST_PATH, або береться з вбудованого списку DEFAULT_SUBDOMAIN_WORDLIST.

Безпечна обгортка _q() виконує окремі DNS-запити (A, AAAA, NS, MX, TXT, CNAME, SOA, CAA, DNSKEY, DS), перетворюючи відповіді у текстове представлення та гарантовано повертаючи список рядків або порожній список у разі будь-яких помилок.

Окремі методи _check_spf() та _check_dmarc() витягують SPF/DMARC-записи з TXT-запитів, а _check_dnssec() намагається визначити наявність DNSSEC через DNSKEY та DS.

Для активного пошуку субдоменів використовується багатопотоковий метод _bruteforce_subdomains(), який формує список word.domain, паралельно перевіряє A-записи через ThreadPoolExecutor та повертає впорядкований перелік знайдених

піддоменів. Пасивні субдомени беруться з контексту (список passive_subs, попередньо зібраний CrtAgent) (рис. 3.4).

```
def _bruteforce_subdomains(self, domain: str, limit: int = MAX_SUBDOMAIN_BRUTE) -> List[str]:  1 usage  Oleksa
    out: List[str] = []
    words = self.wordlist[:limit]
    fqdns = [f"{w}.{domain}" for w in words]
    found: Set[str] = set()
    with concurrent.futures.ThreadPoolExecutor(max_workers=min(MAX_BRUTE_THREADS, len(fqdns) or 1)) as ex:
        future_to_fqdn = {ex.submit(self._probe_a,  *args: fqdn): fqdn for fqdn in fqdns}
        for fut in concurrent.futures.as_completed(future_to_fqdn):
            try:
                res = fut.result()
                if res:
                    found.add(res)
            except Exception:
                continue
    ordered = [fq for fq in fqdns if fq in found]
    return ordered
```

Рисунок 3.4 – Вигляд фрагменту коду брутфорс-пошуку субдоменів у DNSAgent

У фінальному методі run() усі записи та списки субдоменів об’єднуються й повертаються у вигляді DNSResult.

Основні фрагменти реалізації показано на рисунку 3.5.

```
    resolver = self.resolver
    records: Dict[str, List[str]] = {}
    rtypes = ["A", "AAAA", "NS", "MX", "TXT", "CNAME", "SOA", "CAA", "DNSKEY", "DS"]

    for rtype in rtypes:...

    txts = records.get("TXT", []) or []
    spf = self._check_spf(domain, txts)
    if spf:...
    dmarc = self._check_dmarc(domain)
    if dmarc:...

    dnssec = self._check_dnssec(domain)
    if dnssec.get("DNSKEY"):...
    if dnssec.get("DS"):...

    active_subs = self._bruteforce_subdomains(domain, limit=MAX_SUBDOMAIN_BRUTE)
    passive_subs = self.get_shared("passive_subs", []) or []

    combined = list(unique_keep_order(passive_subs + active_subs))
```

Рисунок 3.5 – Вигляд фрагменту коду фінальної обробки DNS-записів у DNSAgent

Агент SSLAgent реалізує низькорівневе зчитування X.509-сертифіката з цільового домену.

Метод _connect_and_get_der() встановлює TCP-з'єднання з хостом на типових TLS-портах (443, 8443, 9443), обгортає його в TLS-контекст з вимкненою перевіркою сертифіката та намагається отримати сертифікат у DER-форматі. За потреби виконується повторна спроба без SNI.

Допоміжна функція _parse_cert_der() розбирає DER-сертифікат за допомогою бібліотеки cryptography, а саме витягує Subject/Issuer CN, термін дії, альтернативні імена (SAN), серійний номер, SHA-256 відбиток, а також версію узгодженого TLS-протоколу.

Сукупність цих даних повертається у моделі SSLResult. Фрагмент реалізації і парсингу сертифіката наведено на рисунку 3.6.

```
host = domain
port_override: Optional[int] = None
if ":" in domain and not domain.count(":") > 1:
    try:
        h, p = domain.rsplit( sep: ":",  maxsplit: 1)
        port_override = int(p)
        host = h
    except Exception:
        host = domain
        port_override = None

ports_to_try = [port_override] if port_override else DEFAULT_TLS_PORTS
candidates: List[Tuple[str, int, Optional[str]]] = []

for p in ports_to_try:
    candidates.append((host, p, host))
```

Рисунок 3.6 – Вигляд фрагменту коду отримання та обробки SSL-сертифіката у SSLAgent

CrtAgent реалізує пасивну розвідку на основі журналів прозорості сертифікатів (Certificate Transparency).

Метод passive_subdomains() формує запит до https://crt.sh/ з параметрами q=%.{domain} та output=json, обробляє можливі варіації заголовків Content-Type та, у разі потреби, «вирізає» JSON-масив із відповіді.

Далі із поля name_value для кожного запису формується відсортований список унікальних доменних імен. Метод run() загортає цей список у модель CrtResult, яка надалі використовується як самостійно (у вкладці CRT.sh інтерфейсу), так і як джерело пасивних субдоменів для DNSAgent.

Відповідний код ілюструється на рисунку 3.7.

```
def passive_subdomains(self, domain: str) -> List[str]:  3 usages  Oleksandr Zalepa *
    try:
        params = {"q": f"%.{domain}", "output": "json"}
        headers = {"User-Agent": "Mozilla/5.0 (compatible; DomainIntellect/1.0)", "Accept": "application/json"}
        r = requests.get( url: "https://crt.sh/", params=params, headers=headers, timeout=60)
        ct = r.headers.get("Content-Type", "")

        text = r.text.strip()
        if "json" not in ct.lower():
            if "[" in text and "]" in text:
                import json, re
                m = re.search( pattern: r"(\[.*\])", text, re.S)
                if m:
                    data = json.loads(m.group(1))
                else:
                    print("crt.sh returned non-JSON response:", text[:200])
                    return []
            else:
                print("crt.sh returned non-JSON response:", text[:200])
                return []
        else:
            data = r.json()
```

Рисунок 3.7 – Вигляд фрагменту коду отримання пасивних субдоменів у CrtAgent

Однак для Shodan спеціально закладено керовану залежність від DNS-результату. Це зроблено через цикл wait(..., FIRST_COMPLETED). Координатор не просто “чекає всіх”, а реагує на завершення конкретних задач. Як тільки завершується dns_agent, координатор одразу витягує з результату DNS список IPv4 (переважно з A-записів кореневого домену), формує a_ips і тільки після цього динамічно додає нове паралельне завдання shodan_agent.run(...) у той самий пул потоків. Тобто Shodan не є “початковою” задачею в futures – він підключається пізніше, коли стає доступним мінімально потрібний контекст.

Передача a_records реалізується не через обмін між потоками “напряму”, а через координатора як диспетчера стану. Він викликає

shodan_agent.set_shared("a_records", a_ips) перед submit. Це означає, що ключова взаємодія між агентами є не peer-to-peer, а опосередкована: агенти не синхронізуються між собою і не знають про існування інших, натомість координатор збирає проміжні артефакти (intermediate) і на їх основі формує вхідні параметри для залежних кроків. Додатково передбачено “страхувальний” механізм: якщо A-записи відсутні, координатор намагається отримати IP через резолюцію піддоменів, використовуючи вже наявні дані (CRT/DNS), але фактично це також лишається логікою координатора, який вирішує, які джерела залучити і коли.

Сукупність полів описана у файлі моделей та використовується, зокрема, при побудові Maltego-подібного графа інфраструктури.

Основні фрагменти коду наведено на рисунку 3.8.

```python
    try:
        import shodan
    except Exception:
        return ShodanResult(domain=domain, hosts=[])

    api_key = os.getenv("SHODAN_API_KEY")
    if not api_key:
        return ShodanResult(domain=domain, hosts=[])

    a_records = self._get_a_records()
    if not a_records:
        return ShodanResult(domain=domain, hosts=[])

    api = shodan.Shodan(api_key)
    hosts = []
```

а)

```python
for ip in a_records:
    try:
        info = api.host(ip)
        host = ShodanHost(
            ip=ip,
            ports=[int(p) for p in info.get("ports", [])],
            org=info.get("org"),
            hostnames=info.get("hostnames", []),
            country=info.get("country_name"),
            city=info.get("city"),
            asn=info.get("asn"),
            isp=info.get("isp"),
            os=info.get("os"),
            tags=info.get("tags", []),
            vulns=info.get("vulns", []),
            cpes=info.get("cpes", []),
            services=info.get("data", []),
        )
        hosts.append(host)
```

б)

Рисунок 3.8 – Фрагменти коду роботи ShodanAgent: а) ініціалізація агента та отримання A-записів; б) обробка даних хостів Shodan

Окремий агент інтегрується з API VirusTotal та повертає результат у вигляді моделі VirusTotalResult, яка акумулює як агреговані показники безпечності домену, так і детальні технічні характеристики.

До ключових полів належать числова репутація домену (reputation), словник категорій, присвоєних різними аналітичними рушіями, статистика останнього аналізу (last_analysis_stats), що відображає кількість детекцій за класами harmless, suspicious, malicious тощо, а також дата останнього сканування.

Окрім цього, модель зберігає соціально орієнтовані метрики, такі як total_votes, що відображають оцінки користувачів спільноти, та набір tags, які характеризують функціональність або потенційний ризик домену. Додатково агент отримує пов'язані IP-адреси й субдомени, що дозволяє трасувати інфраструктурні зв'язки.

Усі «сирі» JSON-об'єкти – як основного ресурсу, так і рілейшнів – зберігаються у структурі результату для подальшої глибшої обробки або верифікації, що підвищує прозорість і гнучкість аналітичного процесу (рис. 3.9).

```
    ov = self._domain_overview(domain)
    vt = VirusTotalResult(domain=domain)
    vt.raw = ov

    if ov and "data" in ov and "attributes" in ov["data"]:
        at = ov["data"]["attributes"]
        stats = at.get("last_analysis_stats") or {}
        vt.last_analysis_stats = VTStats(
            harmless=stats.get("harmless", 0),
            malicious=stats.get("malicious", 0),
            suspicious=stats.get("suspicious", 0),
            undetected=stats.get("undetected", 0),
            timeout=stats.get("timeout", 0),
```

а)

```
vt.reputation = at.get("reputation")
vt.categories = at.get("categories") or {}
vt.last_analysis_date = at.get("last_analysis_date")
vt.total_votes = at.get("total_votes") or {}
vt.whois = at.get("whois")
vt.registrar = at.get("registrar")
vt.tags = at.get("tags") or []
```

б)

Рисунок 3.9 – Фрагменти коду обробки даних VirusTotal: а) формування статистики та атрибутів домену; б) отримання репутаційних полів і метаданих

Ці дані використовуються як у табличному представленні в інтерфейсі, так і як вхід для високорівневого AI-аналізу.

ContentAgent відповідає за автоматизоване завантаження та попередній аналіз головної вебсторінки.

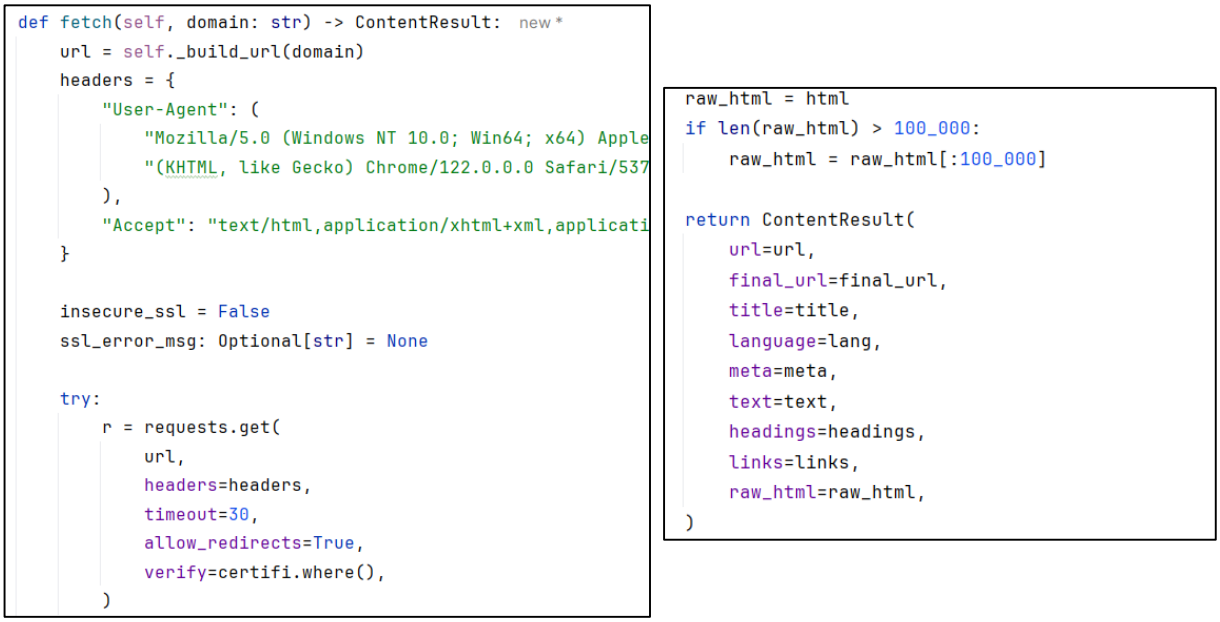
На вхід подається або домен, або повний URL. Допоміжний метод _build_url() формує кінцеву адресу.

Спочатку виконується HTTP-запит з повною перевіркою SSL-сертифіката, у разі виникнення SSLError – за прапорцем ALLOW_INSECURE_SSL агент може повторити запит без перевірки. HTML-відповідь парситься через BeautifulSoup, а саме витягуються заголовок сторінки, атрибут lang, основні meta-теги (description, keywords, og:*), заголовки H1–H3, основний текст та список посилань.

Результат повертається у вигляді ContentResult і відображається окремою вкладкою в інтерфейсі.

Зібрані та нормалізовані дані контенту надалі використовуються як важливе джерело контексту для виявлення фішингових ознак, аномалій у структурі сторінки та загального оцінювання репутаційних характеристик домену.

Фрагмент методу fetch() подано на рисунку 3.10.



```python
def fetch(self, domain: str) -> ContentResult:  new *
    url = self._build_url(domain)
    headers = {
        "User-Agent": (
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
            "(KHTML, like Gecko) Chrome/122.0.0.0 Safari/537
        ),
        "Accept": "text/html,application/xhtml+xml,applicati
    }

    insecure_ssl = False
    ssl_error_msg: Optional[str] = None

    try:
        r = requests.get(
            url,
            headers=headers,
            timeout=30,
            allow_redirects=True,
            verify=certifi.where(),
        )
```

```python
raw_html = html
if len(raw_html) > 100_000:
    raw_html = raw_html[:100_000]

return ContentResult(
    url=url,
    final_url=final_url,
    title=title,
    language=lang,
    meta=meta,
    text=text,
    headings=headings,
    links=links,
    raw_html=raw_html,
)
```

а) б)

Рисунок 3.10 – Фрагменти коду роботи ContentAgent: а) виконання HTTP-запиту та обробка SSL-помилок; б) формування структурованого результату ContentResult

Функція build_full_facts() формує уніфікований JSON-об’єкт, який містить результати роботи всіх агентів системи у структурованому та узгодженому форматі. На цьому етапі дані з WHOIS, DNS, SSL, CRT.sh, Shodan, VirusTotal та ContentAgent нормалізуються, перетворюються на серіалізовані моделі та об’єднуються у єдину фактичну базу знань, придатну для подальшої машинної обробки.

Такий підхід дає змогу уникнути розрізненості вихідних джерел, забезпечуючи цілісне подання доменної інформації для аналітичних модулів (рис. 3.11).

```python
def build_full_facts(domain_result: DomainResult) -> str:  3 usages  new *
    payload = {
        "domain": domain_result.domain,
        "whois": domain_result.whois.model_dump() if domain_result.whois else None,
        "dns": domain_result.dns.model_dump() if domain_result.dns else None,
        "ssl": domain_result.ssl.model_dump() if domain_result.ssl else None,
        "crt": domain_result.crt.model_dump() if domain_result.crt else None,
        "shodan": domain_result.shodan.model_dump() if domain_result.shodan else None,
        "virustotal": domain_result.virustotal.model_dump() if domain_result.virustotal else None,
    }
```

Рисунок 3.11 – Фрагмент коду формування єдиного JSON-об’єкта результатів для LLM-аналізу

Для візуалізації інфраструктури використовується компонент pyvis з кастомним рендерером render_domain_graph(), який будує ієрархічний граф у стилі Maltego з іконками для домену, IP-адрес, ASN, провайдера, країни, CA та репутації VirusTotal.

Таким чином, кожен агент відповідає за свій шар OSINT-аналізу, DomainResult агрегує всі артефакти, інтеграція з CrewAI забезпечує опційний інтелектуальний шар інтерпретації, а інтерфейс Streamlit робить запуск аналізу та перегляд результатів зручним та наочним для користувача.

3.3 Функціональне тестування системи

Тестування розробленої мультиагентної системи OSINT-аналізу доменів проводилося з метою перевірки коректності роботи окремих агентів, узгодженості

їх результатів, а також зручності представлення зібраних даних у користувацькому інтерфейсі. Як тестовий об’єкт було обрано домен vntu.edu.ua та має розгалужену доменну інфраструктуру.

На першому етапі тестування здійснювався запуск повного циклу аналізу домену через вебінтерфейс системи. Після завершення роботи агентів у верхній частині результату відображаються зведені метрики, зокрема дата створення домену, центр сертифікації SSL, кількість A-записів та кількість знайдених субдоменів. Такий формат дозволяє отримати швидке уявлення про досліджуваний ресурс без детального перегляду вкладок.

Далі система формує короткий огляд домену з базовими характеристиками та автоматично отримує скріншот головної сторінки ресурсу за допомогою зовнішнього сервісу візуалізації (рис. 3.12).

Рисунок 3.12 – Короткий огляд домену та скріншот вебресурсу

Важливою частиною тестування стала перевірка модуля графічної візуалізації. Система автоматично будує граф доменної інфраструктури у стилі Maltego, де центральним вузлом є домен, а від нього відходять зв’язки до субдоменів, IP-адрес, автономної системи, центру сертифікації та репутаційних сервісів.

Для доменів із великою кількістю субдоменів використовується агрегований вузол зі списком, що забезпечує читабельність графа (рис. 3.13).

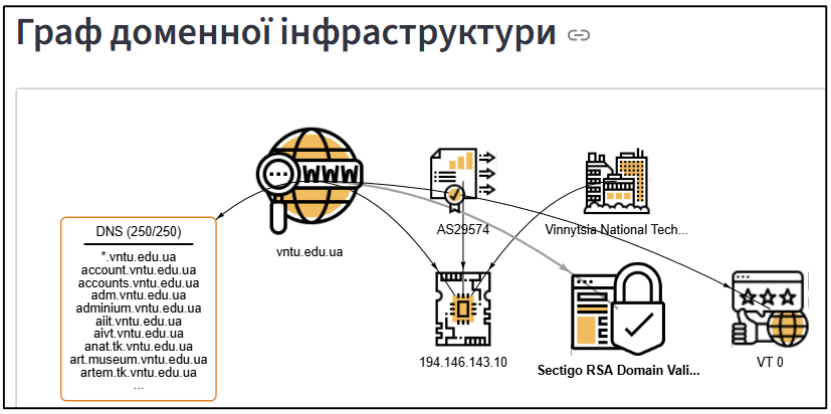


Рисунок 3.13 – Граф доменної інфраструктури досліджуваного домену

Наступним етапом тестування була перевірка відображення результатів переліку субдоменів. У відповідному вікні користувач може переглянути повний список виявлених імен, отриманих шляхом пасивної розвідки та DNS-аналізу. У ході тестування було підтверджено коректне відображення понад 250 субдоменів без втрати продуктивності інтерфейсу (рис. 3.14).

Рисунок 3.14 – Перелік знайдених субдоменів домену

Окремо тестувався агент аналізу контенту вебсторінки. У вкладці «Контент» система відображає URL сторінки, заголовок, мову ресурсу, а також основні meta-

теги Open Graph і SEO-параметри. Це дозволяє оцінити призначення сайту та його відповідність заявленій тематиці (рис. 3.15).

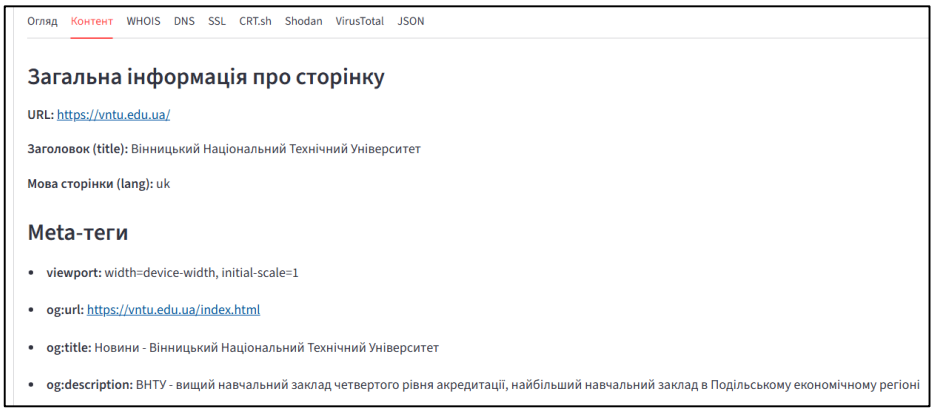


Рисунок 3.16 – Результати аналізу контенту вебсторінки

На наступному етапі перевірялася робота WHOIS-агента. Було встановлено, що система коректно обробляє обмежені WHOIS-дані для доменів, відображаючи доступні дати створення й оновлення, DNS-сервери та контактні електронні адреси адміністратора (рис. 3.16).

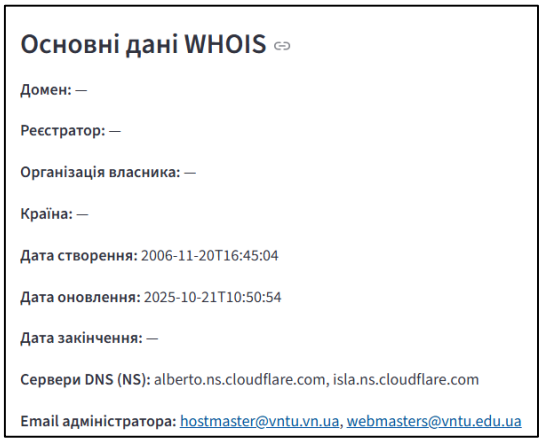
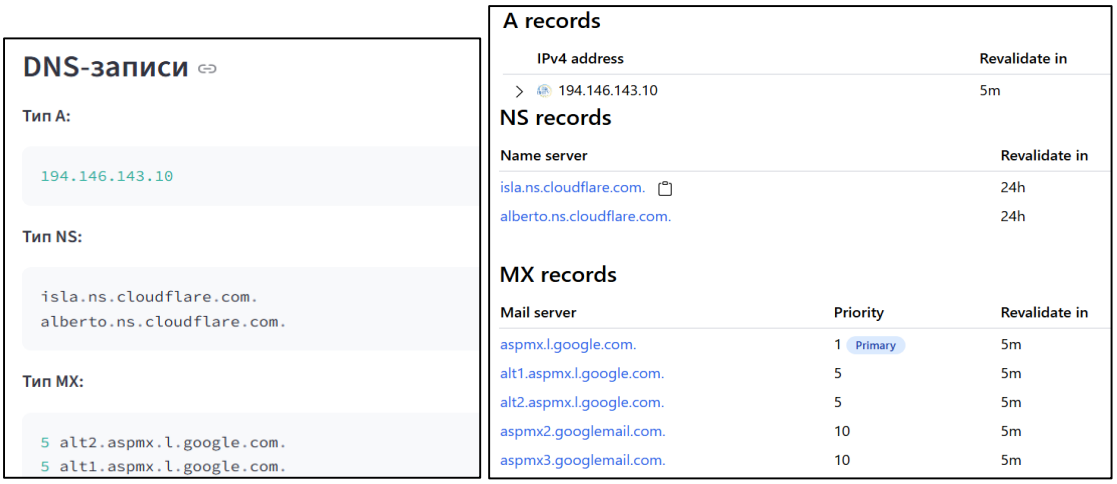


Рисунок 3.16 – Результати WHOIS-аналізу домену

Тестування DNS-агента показало стабільне отримання основних типів DNS-записів (A, NS, MX) для досліджуваного домену.

Коректність роботи агента була перевірена шляхом крос-перевірки результатів із незалежним веб-сервісом DNS-діагностики nslookup.io (рис. 3.17).



а) б)

Рисунок 3.17 – DNS-записи досліджуваного домену: а) фрагмент результату роботи DNS-агента розробленої системи; б) фрагмент результату перевірки DNS-записів за допомогою зовнішнього сервісу nslookup.io

Порівняльний аналіз засвідчив повний збіг IP-адреси в A-записі, переліку іменних серверів NS, а також хостів і пріоритетів MX-записів. На основі значень NS підтверджено використання сервісу Cloudflare для DNS-обслуговування, а за MX-записами – застосування Google Workspace (Google Mail) для обробки електронної пошти.

У межах перевірки SSL-агента було виконано зчитування та аналіз параметрів TLS-сертифіката досліджуваного вебресурсу.

Під час тестування система коректно визначила загальне ім’я сертифіката (Common Name), центр сертифікації та тип сертифіката, а також часові межі його дії, що дає змогу оцінювати актуальність і чинність криптографічного захисту (рис. 3.18).

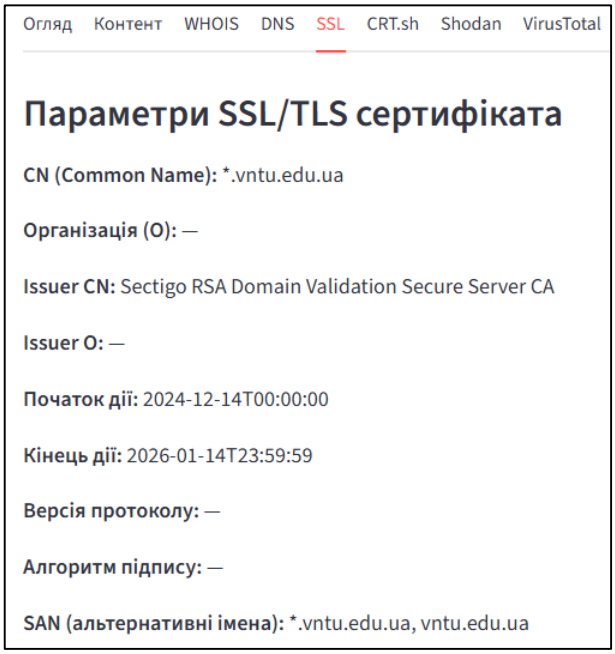


Рисунок 3.18 – Параметри SSL/TLS-сертифіката

Окрему увагу приділено аналізу списку альтернативних доменних імен (Subject Alternative Name, SAN), який використовується для перевірки відповідності сертифіката запитуваному домену.

Отримані результати підтверджують здатність SSL-агента автоматично вилучати ключові атрибути TLS-сертифікатів і використовувати їх як індикатори конфігурації безпеки та потенційних ризиків, що свідчить про коректність реалізації модуля криптографічного аналізу в розробленій системі.

Наступним кроком стало тестування пасивної доменної розвідки за допомогою сервісу crt.sh.

Система автоматично отримала та відобразила перелік доменних імен, пов’язаних із сертифікатами, що дозволяє виявляти приховані або історичні субдомени (рис. 3.19).

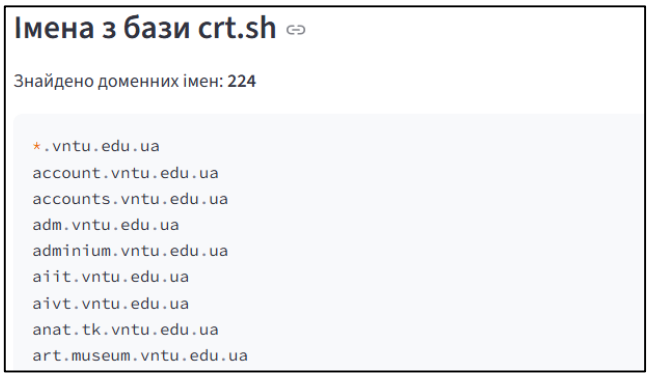


Рисунок 3.19 – Фрагмент результату пасивної розвідки з бази crt.sh

Тестування Shodan-агента показало, що система коректно визначає відкриті порти, організацію-власника IP-адреси, автономну систему та географічне розташування хоста. Отримані дані можуть бути використані для подальшого аналізу поверхні атаки (рис. 3.20).

Рисунок 3.20 – Результати аналізу інфраструктури за допомогою Shodan

Завершальним етапом тестування стала перевірка інтеграції з сервісом VirusTotal. Система коректно відобразила репутаційний бал домену, категорії за класифікацією антивірусних вендорів та оцінку спільноти, що підтвердило відсутність ознак шкідливої активності (рис. 3.21).

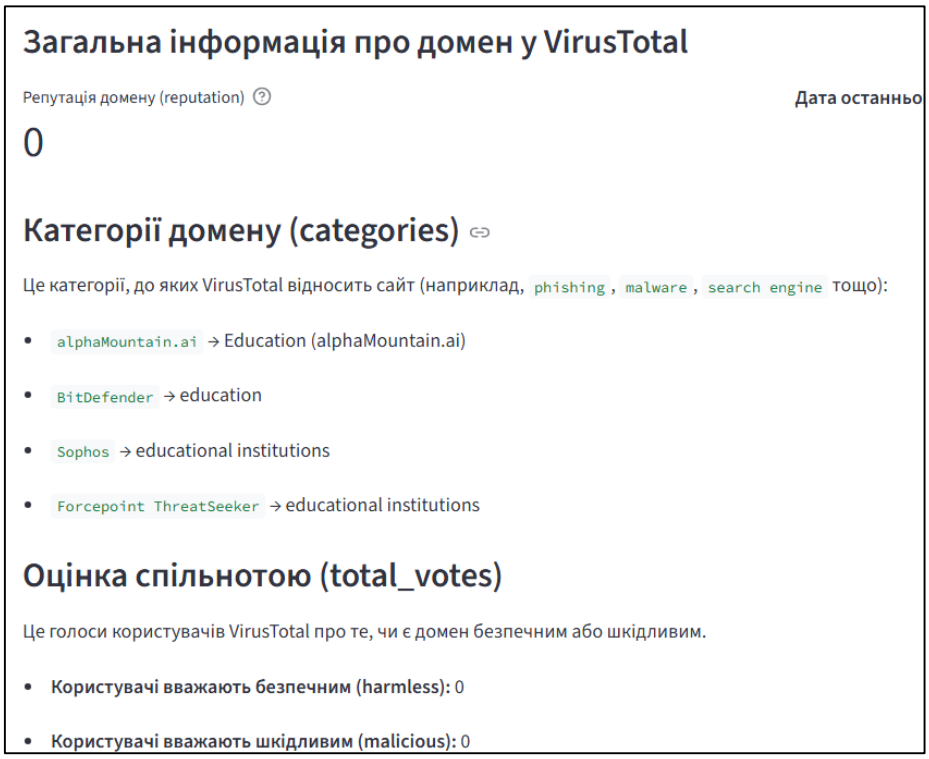


Рисунок 3.21 – Результати репутаційного аналізу домену у VirusTotal

Після детального тестування системи на прикладі легітимного освітнього домену, наступним етапом стала перевірка коректності роботи модуля узагальненого AI-аналізу ризиків.

Для домену vntu.edu.ua система автоматично сформувала інтегральну оцінку безпеки на основі агрегування результатів усіх спеціалізованих агентів. Під час аналізу враховувалися як технічні параметри (вік домену, конфігурація DNS, чинність SSL-сертифіката, мережеві характеристики), так і репутаційні показники з відкритих OSINT-джерел.

За результатами узагальнення було отримано низький ризиковий бал (15/100), рівень ризику Low та класифікацію benign, що узгоджується з фактичним статусом домену офіційного вебресурсу державного закладу вищої освіти. Відсутність ознак фішингу, шкідливого програмного забезпечення або аномальної мережевої

поведінки підтверджує коректність роботи модуля високорівневого AI-аналізу та адекватність сформованої оцінки (рис. 3.22).

Рисунок 3.22 – Загальний AI-аналіз безпеки домену vntu.edu.ua

Короткий висновок, згенерований системою, підтвердив стабільність домену, його тривалу історію існування, використання надійної DNS-інфраструктури Cloudflare та відсутність негативних сигналів у базах репутаційних сервісів. Таким чином, було підтверджено коректність агрегації даних та логіки зниження ризику за позитивними індикаторами.

Деталізований аналіз показав, що система коректно інтерпретує окремі технічні фактори. Зокрема, вік домену та наявність валідного SSL-сертифіката були віднесені до факторів зниження ризику, тоді як велика кількість субдоменів була ідентифікована як нейтрально-негативний фактор, що потенційно може бути використаний для атак, але без явних ознак зловмисності. Всі ці аспекти були прозоро відображені у блоці доказів з поясненням їх впливу на підсумкову оцінку ризику.

На наступному етапі тестування було виконано аналіз іншого ресурсу – phishtank.org, який свідомо обрано для перевірки роботи системи на

альтернативному, але також легітимному домені, щоб уникнути упередженості тестування на одному прикладі. Система коректно зібрала WHOIS-, DNS- та SSL-дані, відобразила інфраструктурний граф і сформувала низький ризиковий бал (10/100) із класифікацією benign (рис. 3.23).

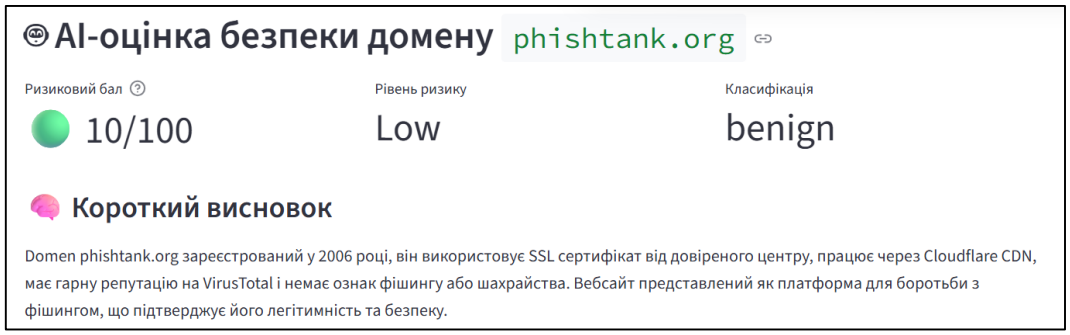


Рисунок 3.23 – Результати аналізу домену phishtank.org

Отримані результати відповідають реальному призначенню ресурсу, який використовується як платформа для боротьби з фішингом. Наявність SSL-сертифіката від довіреного центру сертифікації, використання CDN Cloudflare та відсутність негативних сигналів у VirusTotal були коректно інтерпретовані системою як фактори, що знижують ризик.

Завершальним і найбільш критичним етапом тестування стала перевірка системи на потенційно небезпечному домені, отриманому з відкритої бази підтверджених фішингових сайтів PhishTank.

Для цього було використано піддомен sso-metamask-secure-authd.webflow.io, який імітує інтерфейс популярного криптовалютного гаманця MetaMask.

У результаті аналізу система зафіксувала аномально велику кількість субдоменів (понад 1000), відсутність WHOIS-даних для конкретного піддомену, характерні ключові слова у назві (secure, auth, login), а також негативні сигнали з VirusTotal, де домен було позначено як фішинговий низкою антивірусних рушіїв.

На основі цих факторів система присвоїла високий ризиковий бал (85/100), рівень ризику High та класифікацію malicious_like (рис. 3.24).

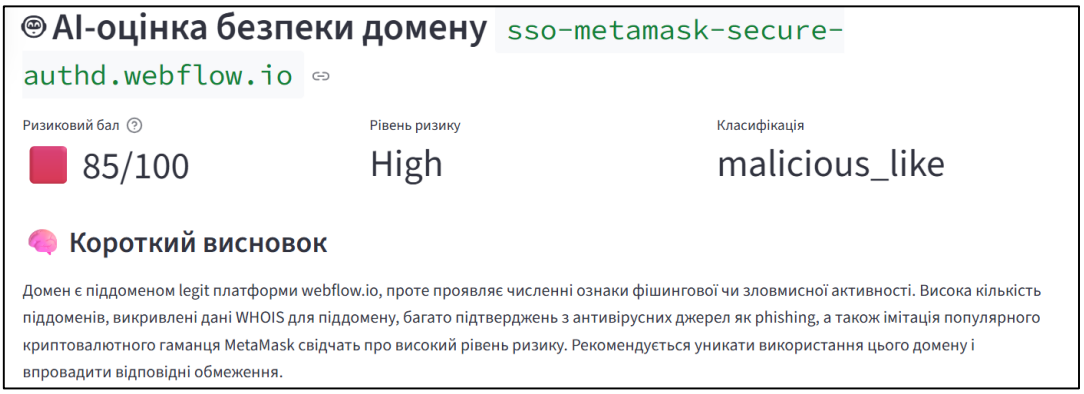


Рисунок 3.24 – AI-оцінка безпеки потенційно фішингового домену sso-metamask-secure-authd.webflow.io

Детальний аналіз чітко продемонстрував здатність системи відрізняти інфраструктурні ознаки легітимних доменів від поведінкових і контентних ознак фішингових ресурсів. Зокрема, навіть за наявності валідного SSL-сертифіката та використання CDN Cloudflare, система коректно визначила фішинговий характер ресурсу на основі контенту сторінки та репутаційних даних (рис. 3.25).

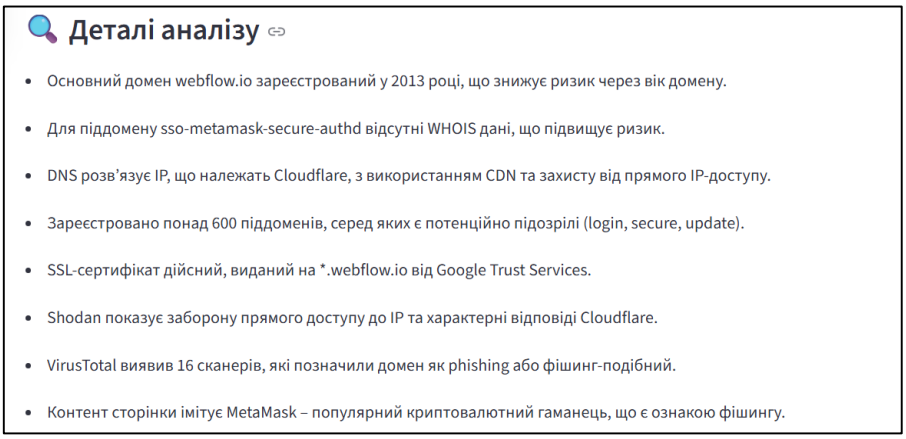


Рисунок 3.25 – Деталізація факторів ризику фішингового домену

Таким чином, проведене тестування підтвердило працездатність мультиагентної системи в різних сценаріях – від аналізу легітимних корпоративних ресурсів до виявлення фішингових доменів із прихованою інфраструктурою.

3.4 Оцінювання ефективності системи

Для оцінювання часової ефективності розробленої мультиагентної системи було проведено експериментальне вимірювання тривалості повного циклу аналізу одного доменного імені.

Під час експерименту час аналізу визначався як інтервал між моментом ініціації перевірки домену користувачем у вебінтерфейсі та моментом формування фінального зведеного аналітичного звіту, що містить агреговані результати роботи всіх спеціалізованих агентів (WHOIS, DNS, SSL/TLS, crt.sh, Shodan, VirusTotal та агент аналізу вебконтенту).

Вимірювання проводились у контрольованому середовищі на одному й тому ж пристрої за стабільного мережевого з'єднання. Для зменшення впливу випадкових затримок зовнішніх API кожен домен аналізувався не менше трьох разів, після чого обчислювалось середнє значення часу виконання.

У якості тестових об'єктів використовувались домени різних категорій, зокрема легітимні корпоративні ресурси та підозрілі домени, що дозволило врахувати різний обсяг отриманих даних та кількість знайдених субдоменів (табл. 3.1).

Таблиця 3.1 – Результати тестування мультиагентної системи OSINT-аналізу доменів

№	Тип домену	Кіль-кість тестів	Середній risk_score	Клас системи	Очікуваний клас	Помилки класифікації	Середній час формування звіту, с
1	Легітимні корпоративні ресурси	10	8–22	10/10 benign	benign	0%	65,4
2	Фішингові домени	10	72–95	8/10 malicious_like 2/10 benign	malicious_like	20%	78,1

У межах експериментального дослідження було протестовано 20 доменів, з яких 10 відповідали легітимним корпоративним ресурсам, а 10 – фішинговим або

потенційно зловмисним вебсайтам. Для кожного домену система виконувала паралельний збір WHOIS-, DNS-, SSL/TLS-, сертифікаційних, репутаційних та контентних ознак із подальшою агрегацією результатів у єдиний інтегральний показник ризику.

Результати тестування показали, що для легітимних доменів система стабільно формувала низькі значення risk_score та коректно класифікувала їх як benign. Для фішингових доменів значення інтегрального показника ризику суттєво зростали, що дозволяло віднести їх до класу malicious_like.

Таким чином, проведене тестування підтвердило працездатність мультиагентної системи в різних сценаріях – від аналізу легітимних корпоративних ресурсів до виявлення фішингових доменів із прихованою інфраструктурою. Експериментальні результати свідчать, що система здатна коректно агрегувати технічні, репутаційні та контентні ознаки, забезпечуючи формування обґрунтованої інтегральної оцінки безпеки доменів при прийнятному часі обробки.

Для оцінювання ефективності запропонованої системи було проведено порівняльний експеримент, у межах якого три умовних експерти з кібербезпеки виконували збір і первинний аналіз інформації про домени двома різними способами, а саме:

- з використанням розробленої мультиагентної системи;
- у ручному режимі із застосуванням сторонніх спеціалізованих OSINT-сервісів.

Кожен експерт працював з однаковим набором тестових доменів та виконував типові етапи доменного OSINT-аналізу, зокрема:

- отримання WHOIS-даних;
- аналіз DNS-записів;
- перевірку SSL/TLS-сертифіката;
- пошук пов’язаних доменів і піддоменів (crt.sh);

– оцінювання репутації за допомогою VirusTotal та Shodan;
– формування узагальненого аналітичного висновку.

У випадку ручного підходу експерти послідовно використовували окремі веб-сервіси, переходячи між ними вручну та самостійно агрегуючи результати для формування висновку. Час фіксувався окремо для кожного експерта від моменту виконання першого запиту до завершення підготовки узагальненого звіту.

Під час використання розробленої мультиагентної системи всі зазначені етапи виконувалися автоматично та паралельно спеціалізованими агентами, а роль експерта зводилася до ініціації аналізу та перегляду сформованого звіту. Час вимірювався від моменту запуску аналізу до отримання повного аналітичного результату.

Усереднені значення часу, отримані за результатами роботи трьох експертів, наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння середньої тривалості виконання етапів доменного OSINT-аналізу

Етап аналізу	Ручний OSINT-аналіз (сторонні сервіси), с	Розроблена мультиагентна система, с
Отримання WHOIS-даних	108	71
Аналіз DNS-записів	102	
Перевірка SSL/TLS-сертифіката	72	
Пошук піддоменів (crt.sh)	138	
Репутаційна перевірка (VirusTotal, Shodan)	168	
Формування узагальненого висновку	102	
Середній час аналізу одного домену	690	

Середній час формування повного аналітичного звіту за допомогою розробленої системи становив близько 1 хвилини, залежно від типу домену та обсягу зібраних даних.

Натомість ручний OSINT-аналіз із використанням сторонніх сервісів вимагав від 10 до 12 хв на один домен.

Таким чином, результати експерименту показали, що застосування запропонованої мультиагентної системи дозволяє:

- скоротити час аналізу одного домену в середньому на 8–10 хв порівняно з ручним підходом;
- зменшити тривалість аналізу у 5-6,5 разів у порівнянні з традиційним ручним збором даних;

Досягнутий виграш у часі обумовлений паралельним виконанням спеціалізованих агентів, відсутністю ручної агрегації результатів та автоматичним формуванням інтегральної оцінки ризику на основі технічних, репутаційних і контентних ознак.

3.5 Порівняльний аналіз розробленої системи з існуючими інструментами

Для обґрунтування практичної цінності розробленої мультиагентної системи було проведено порівняльний аналіз її результатів із одним із найпоширеніших комерційних інструментів доменної OSINT-розвідки – Maltego. Як тестовий об’єкт використано потенційно фішинговий домен sso-metamask-secure-authd.webflow.io, для якого в Maltego було виконано повний набір доступних трансформацій.

На рисунку 3.26 представлено фрагмент графа, згенерованого Maltego, який демонструє зв’язки між досліджуваним доменом, базовим доменом платформи Webflow, WWW-піддоменом та великою кількістю DNS-імен, отриманих у результаті автоматичних трансформацій.

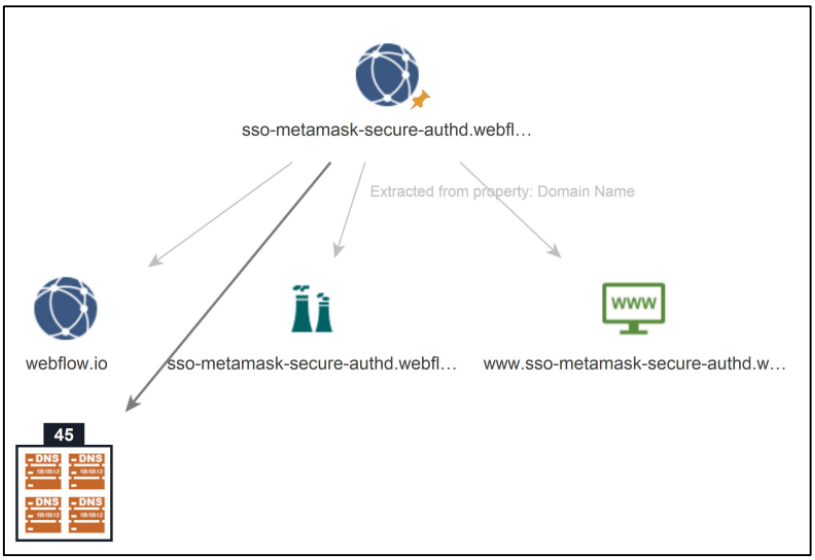


Рисунок 3.26 – Граф доменної інфраструктури в середовищі Maltego

Як видно з рисунка, Maltego ефективно виявляє структурні зв’язки між доменними сутностями та дозволяє швидко отримати великий обсяг сирих даних. Однак отримана інформація подається переважно у вигляді низькорівневих сутностей без автоматичної інтерпретації їхнього впливу на рівень ризику.

На рисунку 3.27 наведено детальні властивості доменної сутності в Maltego, зокрема значення Weight, інформацію WHOIS та контекстні атрибути.

Рисунок 3.27 – Детальна інформація про домен у Maltego

При цьому WHOIS-запит для піддомену має статус помилкового або некоректного, що потребує ручного аналізу з боку оператора.

Окрему увагу заслуговує перелік вихідних DNS-імен, отриманих у Maltego (рис. 3.28).

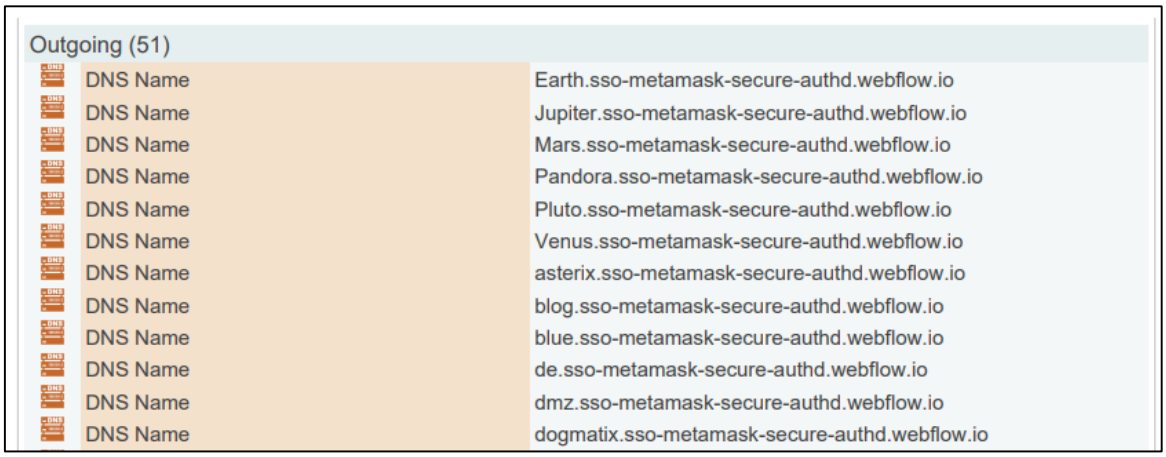


Рисунок 3.28 – Перелік DNS-імен, отриманих у Maltego

Інструмент виявляє десятки технічно коректних, але семантично схожих субдоменів (ftp, imap, forum, blog, gateway тощо), які можуть використовуватись у фішингових сценаріях. Водночас Maltego не здійснює автоматичної класифікації таких імен як підозрілих або небезпечних.

На відміну від цього, розроблена мультиагентна система не лише виявляє аналогічний перелік субдоменів, але й автоматично інтерпретує їх кількість, структуру та семантику в контексті оцінки ризику. Зокрема, велика кількість піддоменів із ключовими словами secure, auth, login була ідентифікована як суттєвий фактор підвищення ризику, що безпосередньо вплинуло на фінальну класифікацію домену як malicious_like.

Крім того, принциповою відмінністю розробленої системи є наявність інтегрованого AI-аналізу, який агрегує результати WHOIS-, DNS-, SSL-, crt.sh-, Shodan-, VirusTotal- та контент-аналізу в єдину інтерпретовану модель ризику. У той час як Maltego виконує роль потужного інструменту збору та візуалізації даних, остаточна оцінка безпеки в ньому повністю покладається на експерта.

Окрім Maltego, для порівняльного аналізу було використано хмарну OSINT-платформу SecurityTrails, яка широко застосовується для дослідження DNS-інфраструктури доменів.

Аналіз проводився для того самого потенційно фішингового домену sso-metamask-secure-authd.webflow.io з використанням безкоштовного тарифного плану платформи.

Платформа коректно визначає використання інфраструктури Cloudflare та відображає як IPv4-, так і IPv6-адреси, що підтверджує належність ресурсу до CDN-провайдера.

На рисунку 3.29 наведено результати, отримані у SecurityTrails, які включають перелік актуальних A- та AAAA-записів домену.

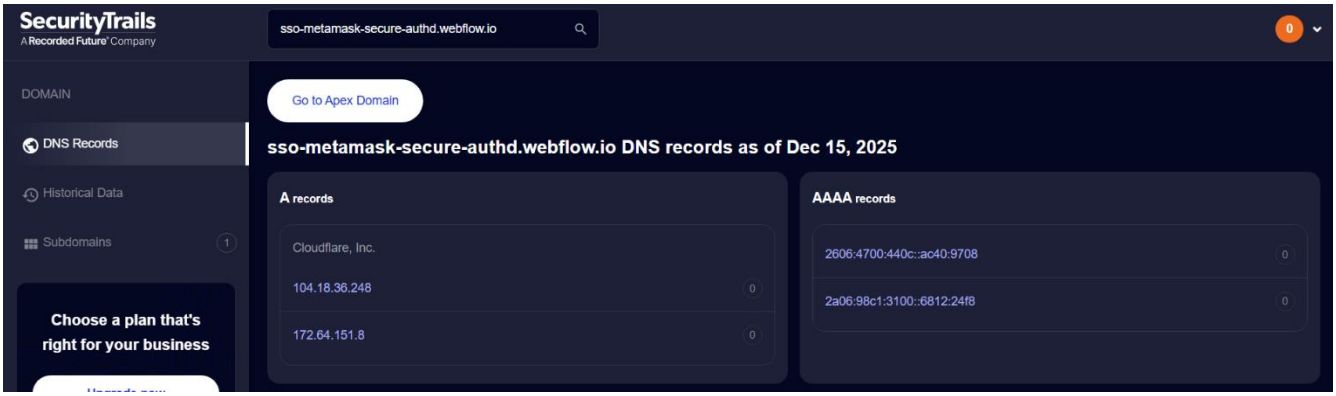


Рисунок 3.29 – Результати DNS-аналізу домену в SecurityTrails

Водночас аналіз показав, що функціональні можливості SecurityTrails у безкоштовній версії є суттєво обмеженими. Користувач отримує лише статичний перелік DNS-записів, без історії змін, без глибокого аналізу субдоменів та без будь-якої інтерпретації отриманих даних з точки зору кібербезпеки. Для доступу до розширених можливостей платформа вимагає платну підписку.

Для подальшого порівняльного аналізу можливостей розробленої мультиагентної системи було використано платформу Censys, яка спеціалізується на активному скануванні інтернет-інфраструктури та зборі інформації про відкриті

сервіси, хости й TLS-сертифікати. Аналіз проводився для двох доменів: потенційно фішингового ресурсу sso-metamask-secure-authd.webflow.io, підтвердженого базою PhishTank, та легітимного домену vntu.edu.ua, що використовувався в попередніх експериментах.

На рисунку 3.30 показано результати пошуку в Censys для фішингового домену sso-metamask-secure-authd.webflow.io.

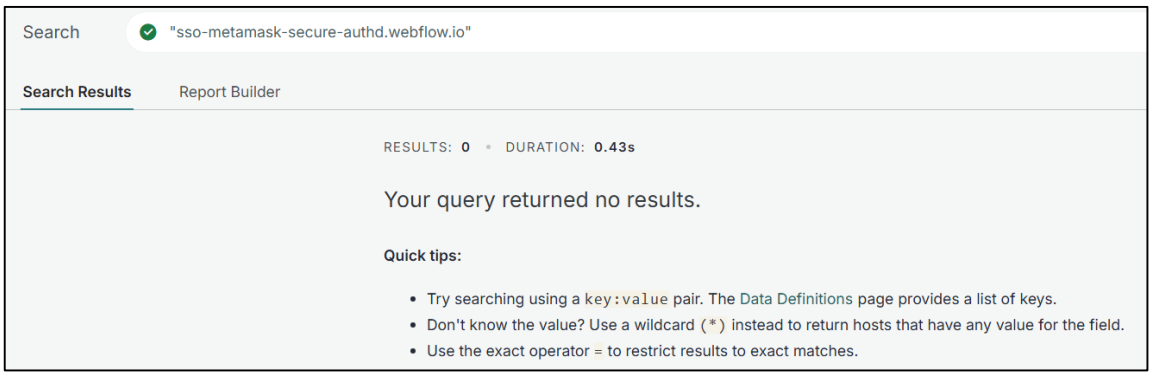


Рисунок 3.30 – Відсутність результатів пошуку в Censys для фішингового домену

Платформа не повернула жодних результатів, що свідчить про відсутність відкритих хостів або сервісів, безпосередньо асоційованих із цим доменом у базі Censys.

Такий результат пояснюється особливостями архітектури фішингових ресурсів, які часто розміщуються за CDN або хмарними платформами (у даному випадку Webflow та Cloudflare), що унеможливлює пряме сканування кінцевих IP-адрес.

Водночас це демонструє обмеження Censys у задачах доменно-орієнтованого OSINT-аналізу, особливо для виявлення фішингових кампаній, які навмисно приховують свою інфраструктуру.

На рисунку 3.31 наведено результати пошуку в Censys для легітимного домену vntu.edu.ua.

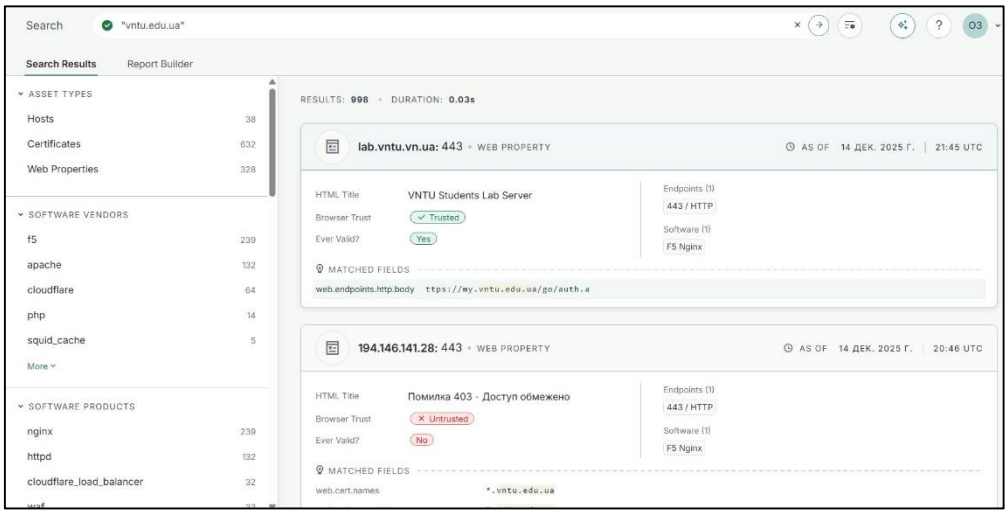


Рисунок 3.31 – Результати аналізу домену vntu.edu.ua в Censys

У цьому випадку платформа повернула значну кількість результатів, зокрема інформацію про вебресурси, TLS-сертифікати, програмне забезпечення серверів та відкриті порти.

Отримані дані підтверджують, що Censys є ефективним інструментом для аналізу відкритої серверної інфраструктури організацій із власними IP-адресами та сервісами. Проте навіть у цьому випадку платформа не надає інтегральної оцінки безпеки домену та не здійснює автоматичної класифікації його ризиковості, залишаючи інтерпретацію результатів на користувача.

З метою узагальнення результатів порівняння та забезпечення об’єктивної оцінки функціональних можливостей розробленої мультиагентної системи й існуючих OSINT-інструментів, подальший аналіз було зведено у вигляді систематизованої порівняльної таблиці за ключовими функціональними критеріями. Узагальнені результати такого порівняння наведено в таблиці 3.3, яка дозволяє наочно оцінити відмінності між підходами до збору, обробки та інтерпретації доменної інформації.

Таблиця 3.3 – Порівняльний аналіз функціональних можливостей OSINT-інструментів

Функціональний критерій	Розроблена система	Maltego	SecurityTrails	Censys
Орієнтація на доменний OSINT-аналіз	+	±	+	±
Підтримка аналізу піддоменів	+	+	±	±
Семантичний аналіз імен піддоменів	+	–	–	–
Виявлення фішингових патернів	+	–	–	–
Інтеграція WHOIS-аналізу	+	+	±	–
Коректна обробка WHOIS для піддоменів	+	–	–	–
DNS-аналіз (A, AAAA, NS, CNAME)	+	+	+	±
Аналіз SSL/TLS-сертифікатів	+	±	–	+
Аналіз crt.sh	+	–	–	±
Інтеграція з VirusTotal	+	–	–	–
Інтеграція з Shodan	+	±	–	±
Контент-аналіз вебсторінки	+	–	–	–
Працездатність для доменів за CDN	+	+	+	±
Низька залежність від активного сканування	+	+	+	–
Формування інтегральної оцінки ризику	+	–	–	–
Автоматична класифікація домену	+	–	–	–
Мінімальна потреба в експертній інтерпретації	+	–	–	–
Рівень автоматизації аналізу	+	±	–	±
Придатність для оперативного виявлення фішингу	+	±	–	–
Орієнтація на підтримку прийняття рішень	+	–	–	–
ШІ інтеграція	+	–	–	–

Для кількісної інтерпретації результатів функціонального порівняння було застосовано бальну модель оцінювання, що дозволяє узагальнити наявність або відсутність ключових можливостей у числовому вигляді.

Для кількісного порівняння введено бальну шкалу:

– “+” = 1 бал;

– “±” = 0,5 бала;

– “–” = 0 балів.

Результати підрахунку сумарної кількості балів для кожного інструменту наведено в таблиці 3.4.

Таблиця 3.4 – Узагальнена бальна оцінка функціональних можливостей OSINT-інструментів

Інструмент	Сума балів
Розроблена мультиагентна система	21
Maltego	7,5
SecurityTrails	6
Censys	6,5

Отримані значення підтверджують, що розроблена система суттєво перевищує існуючі рішення за сукупністю функціональних можливостей, особливо в частині автоматизованої інтерпретації результатів і підтримки прийняття рішень.

Проведений порівняльний аналіз показав, що існуючі інструменти доменної OSINT-розвідки (Maltego, SecurityTrails, Censys) орієнтовані переважно на отримання окремих класів технічних даних і потребують значної участі експерта для формування підсумкових висновків. На відміну від них, розроблена мультиагентна система реалізує інтегрований підхід, що поєднує пасивну й активну доменну розвідку, аналіз контенту вебресурсів та репутаційні дані зовнішніх сервісів з подальшою автоматизованою інтерпретацією результатів.

Результати, наведені в таблицях 3.2 та 3.3, підтверджують, що система забезпечує формування інтегральної оцінки ризику у вигляді числового показника, рівня ризику та семантичної класифікації домену менш ніж за одну хвилину, що є критично важливим для задач оперативного виявлення фішингових і зловмисних ресурсів, зокрема у середовищах із використанням CDN або хмарних платформ.

Таким чином, розроблена система не лише досягає функціонального паритету з існуючими засобами доменної розвідки, але й перевищує їх за рівнем автоматизації, аналітичної узгодженості та практичної орієнтації на підтримку рішень у сфері кібербезпеки.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на впровадження лише за умови відповідності вимогам науково-технічного прогресу та економічної доцільності. Саме тому результати дослідження повинні бути оцінені з позицій їх ефективності та потенціалу впровадження у практичну діяльність користувачів у сфері кібербезпеки.

Магістерська кваліфікаційна робота на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» належить до науково-технічних робіт прикладного характеру, орієнтованих на подальшу комерціалізацію та використання на ринку кіберзахисних рішень. Такий напрям є пріоритетним, адже впровадження інструменту дозволяє підвищити рівень інформаційної безпеки підприємств і державних установ, скоротити час на проведення доменної розвідки та автоматизувати процес аналізу потенційних кіберзагроз.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

- проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;
- розраховано витрати на здійснення науково-технічної розробки;
- розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором;
- проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту розробки є оцінювання її науково-технічного рівня, ступеня інноваційності та комерційного потенціалу на ринку кіберзахисних рішень. Такий аудит дозволяє визначити конкурентоспроможність продукту та доцільність подальшого інвестування у його вдосконалення та промислове впровадження.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [62].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші,	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі,
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 4.1

Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют ься у
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х

Продовження таблиці 4.1

Практична здійсненність					
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	4
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	2	2	3
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	4
11. Практична здійсненність (термін реалізації)	4	5	4

Продовження таблиці 4.2

Критерії	Бали:		
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	37	42	39
Середньоарифметична сума балів $СБ_с$	39,33		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [62].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» становить 39,33 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [62]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \,, \tag{4.2}$$

де $I_{нi}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{нi}} \,; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований програмний засіб	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість джерел OSINT-аналізу	од.	3	7	2,33	0,15
Швидкодія (час аналізу одного домену)	с	10	6	1,67	0,25
Глибина доменного аналізу (кількість типів артефактів)	од.	8	14	1,75	0,25
Автоматизація взаємодії між модулями (рівень інтелектуальності MAS)	бал	5	9	1,8	0,2
Виявлення підозрілих доменів за ризик-факторами	%	75	92	1,23	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2{,}33 \cdot 0{,}15 + 1{,}67 \cdot 0{,}25 + 1{,}75 \cdot 0{,}25 + 1{,}80 \cdot 0{,}20 + 1{,}23 \cdot 0{,}15 = 1{,}75.$$

Отже, за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,75 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, робітникам, студентам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [62]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-дослідної роботи (проєктний менеджер)	42 000,00	1 909,09	22	42 000,00
Інженер з кібербезпеки 1-ї категорії	38 000,00	1 727,27	18	31 090,86
Інженер-програміст	36 000,00	1 636,36	20	32 727,20
Інженер з тестування	30 000,00	1 363,64	10	13 636,40
Всього				119 454,46

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \tag{4.5}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М \cdot K_i \cdot K_c}{T_р \cdot t_{зм}} , \tag{4.6}$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [62];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 8000,00 · 1,20 · 1,15 / (22 · 8) = 62,73 *грн.*

$З_{р1}$ = 62,73 · 6,00 = 376,38 *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість, год	Розряд	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Витрати, грн
Підготовка робочого місця для побудови інфраструктури збору доменних даних	5,0	2	1,20	62,73	313,65
Інсталяція та налаштування сервісів для збору DNS/WHOIS/SSL-даних	7,0	3	1,35	70,60	494,20
Розгортання середовища аналізу кіберзагроз (Shodan, Censys, VT інтеграції)	10,0	4	1,50	78,41	784,10
Деплой сервісів мультиагентної взаємодії та брокера задач	14,0	3	1,35	70,60	988,40
Налаштування механізмів взаємодії агентів і тестування API	18,0	4	1,50	78,41	1 411,38
Збір та оновлення тестових доменів, підготовка OSINT-сетів	8,0	2	1,20	62,73	501,84
Розгортання та первинне налаштування UI-візуалізатора доменної інфраструктури	6,5	3	1,35	70,60	458,90
Проведення функціонального тестування мультиагентної системи	6,0	2	1,20	62,73	376,38
Контроль та супровід експериментального запуску системи	9,0	4	1,50	78,41	705,69
Всього					6 034,54

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (119454,46 + 6034,54) · 11 / 100% = 13803,79 *грн.*

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$З_н$ = (119454,46+ 6034,54 + 13803,79) · 22 / 100% = 30644,41 *грн.*

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = 1,1 ... 1,15);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 2{,}000 \cdot 185{,}00 \cdot 1{,}05 - 0 \cdot 0 = 388{,}50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, од.	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Багатофункціональний офісний папір А4 (80 г/м²)	185,00	2,000	0	0	388,50
Папір для нотаток А5 (100×150 мм)	95,00	3,000	0	0	299,25
Набір маркерів для дошки (перманентні, 4 кольори)	280,00	0,800	0	0	246,40
Комплект наклейок-стікерів для маркування доменів	150,00	1,200	0	0	198,00
Тонер-картридж для лазерного принтера	1 900,00	0,600	0	0	1 231,20
Всього					2 363,35

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1{,}1 \ldots 1{,}15$).

При цьому до комплектуючих віднесено елементи, необхідні для організації тестового стенду доменної розвідки: зовнішній накопичувач для збереження

зібраних доменних даних та логів, мережевий комутатор для виділеного лабораторного сегмента, USB-хаб і патч-корди для гнучкого підключення обладнання.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній SSD-накопичувач 1 ТБ (USB 3.2) для зберігання доменних даних та логів	1	3 200,00	3 360,00
8-портовий гігабітний мережевий комутатор для виділеного сегмента аналізу трафіку	1	1 100,00	1 155,00
USB 3.0 хаб (4 порти) для підключення зовнішніх пристроїв та діагностичних інструментів	1	450,00	472,50
Набір патч-кордів UTP Cat.6 (комплект 5 шт.) для підключення обладнання в лабораторному стенді	1	400,00	420,00
Всього			5 407,50

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» відносять витрати на придбання спеціального наукового обладнання, а також витрати на його розробку, транспортування, монтаж і налаштування. Таке обладнання повинне бути необхідним саме для проведення експериментальних досліджень за обраною тематикою.

Спецустаткування спеціального призначення не використовується, і витрати за даною статтею не передбачаються**.**

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 450,00 \cdot 1 \cdot 1,05 = 472,50$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт.	Ціна за одиницю, грн	Коеф. KiK_iKi	Вартість, грн
Доступ до мережі Internet (високошвидкісний), грн/місяць	1	450,00	1,05	472,50
PyCharm Professional (академічна ліцензія для розробки)	1	0,00	1,00	0,00
База даних PostgreSQL (open-source)	1	0,00	1,00	0,00
Пакет API-кредитів Shodan (для технічного доменного аналізу)	1	500,00	1,05	525,00
WhoisXML API Standard Lookup Credits (WHOIS/регістранти)	1	350,00	1,05	367,50
VirusTotal Public API (безкоштовно, не потребує оплати)	1	0,00	1,00	0,00
ОС Linux (Ubuntu Server LTS)	1	0,00	1,00	0,00
Всього				1 365,00

4.3.7 Амортизація обладнання, програмних засобів та приміщень

Амортизаційні відрахування по кожному виду обладнання, програмного забезпечення та приміщень, які використовувалися під час досліджень, розраховуємо за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.12)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = *(*52 000,00 · *1) / (4 · 12)* = 1 083,33 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання / ПЗ / інфраструктури	Балансова вартість, грн	Строк корисного використання, років	Термін використання під час НДР, міс	Амортизаційні відрахування, грн
1	2	3	4	5
Робоча станція інженера-розробника (Intel i7 / 32GB RAM / 1TB NVMe / RTX 4060)	52 000,00	4	1	1 083,33
Ноутбук для мобільного OSINT-аналізу (Ryzen 7 / 16GB / 512GB SSD)	36 500,00	3	1	1 013,89
Мережевий маршрутизатор з підтримкою IDS/IPS	7 200,00	4	1	150,00
Комутатор 1GbE для лабораторного сегмента	2 400,00	5	1	40,00

Продовження таблиці 4.10

1	2	3	4	5
Сертифіковане ОС Windows 11 Pro	8 800,00	3	1	244,44
Прикладний пакет Microsoft Office 2021	5 200,00	3	1	144,44
Приміщення лабораторії кібербезпеки (орендована площа)	480 000,00	30	1	1 333,33
Моніторингова периферія (монітори, клавіатури, UPS)	13 800,00	5	1	230,00
Всього				4 239,43

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.13)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Ц$_e$ = 12,56 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,38 \cdot 172,0 \cdot 12,56 \cdot 0,95 / 0,97 =$ 796,33 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість роботи, год	$K_{вп i}$	η_i	Сума, грн
Робоча станція інженера-розробника (Intel i7 / 32GB / RTX 4060)	0,38	172,0	0,95	0,97	796,33
Ноутбук для OSINT-аналізу (Ryzen 7 / 16GB)	0,10	172,0	0,90	0,97	199,82
Комутатор 1GbE для лабораторного сегмента	0,03	172,0	0,95	0,98	62,70
Маршрутизатор з IDS/IPS	0,02	172,0	0,95	0,98	41,80
Моніторингова периферія (монітор + UPS + клав/миша)	0,12	172,0	0,95	0,97	253,30
Лазерний ч/б принтер	0,25	4,0	0,90	0,95	11,90
ВСЬОГО					1 365,85

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, залучених до виконання дослідження, відрядження, пов’язані з проведенням випробувань програмно-апаратних засобів, а також поїздки на наукові з’їзди, конференції, семінари та наради, безпосередньо пов’язані з виконанням даної роботи.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.14)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження»,

приймемо $H_{св}$ = 0%.

$B_{св}$ = (119454,46 + 6034,54) · 0 / 100% = 0,00 грн.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.15)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (119454,46 + 6034,54) · 30 / 100% = 37646,70 грн.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (119454,46 + 6034,54) · 50 / 100% = 62744,50 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з

освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = (119454,46 + 6034,54) · 100 / 100% = 125489,00 грн.

Витрати на проведення науково-дослідної роботи на тему «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.18)$$

$B_{заг}$= 119454,46 + 6034,54 + 13803,79 + 30644,41 + 2363,35 + 5407,50 + 0,00 + 4 239,43 + 1 365,85 + 0,00 + 37 646,70 + 62 744,50 + 125 489,00 = 409 193,53 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 409 193,53 / 0,9 = 454659,48 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	300	700	900	600

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 200000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 6000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1200,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [63]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1200 \cdot 20000 + 7200 \cdot 300) \cdot 0,8333 \cdot 0,4 \cdot (1 - 0,18) \approx 7150113,98 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1200 \cdot 20000 + 7200 \cdot 700) \cdot 0,8333 \cdot 0,4 \cdot (1 - 0,18) \approx 7937282,50 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1200 \cdot 20000 + 7200 \cdot 900) \cdot 0,8333 \cdot 0,4 \cdot (1 - 0,18) \approx 8330866,75 \text{ грн.}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (1200 \cdot 20000 + 7200 \cdot 600) \cdot 0,8333 \cdot 0,4 \cdot (1 - 0,18) \approx 7740490,37 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

ПП ≈ 6384030,34 + 6327553,01 + 5929746,40 + 4919221,56 = 23560551,32 грн

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,5;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 454659,48 грн.

$PV = k_{інв} \cdot ЗВ$ = 1,5 · 454659,48 = 681989,22 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 23560551,32 грн;

PV – теперішня вартість початкових інвестицій, 681989,22грн.

$$E_{абс} = ПП - PV = 23560551{,}32 - 681989{,}22 = 22878562{,}10 \text{грн.}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 22878562,10 грн;

PV – теперішня вартість початкових інвестицій, 681989,22 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+22878562{,}10/681989{,}22)^{1/4} = 2{,}42.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.25}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін}$ = 0,11+0,3 = 0,41 < 2,42 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,42 = 0,41 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

На підставі проведених розрахунків встановлено, що рівень комерційного потенціалу науково-технічної розробки за темою «Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки» є високим,

що підтверджує доцільність її подальшого впровадження та виходу на ринок кібербезпекових рішень.

За результатами оцінювання за техніко-економічними параметрами вдосконалений інструмент збору та кореляції доменної інформації перевищує функціональні можливості наявних аналогів за рахунок:

- автоматизації OSINT-досліджень доменів;
- використання мультиагентного підходу для паралельної обробки артефактів;
- інтегрованої аналітики та формування ризик-профілю домену.

Економічні показники комерціалізації проєкту підтверджують його інвестиційну привабливість:

- приведена вартість чистих прибутків інвестора за 4 роки може скласти ≈ 23,56 млн грн;
- період окупності інвестицій становить лише ≈ 0,41 року, що значно менше допустимого нормативу у 3 роки;
- внутрішня економічна дохідність істотно перевищує мінімально допустиму величину, що характеризує низькі інвестиційні ризики.

Отже, результати економічних розрахунків доводять, що розроблена система збору доменної інформації на основі мультиагентного підходу має високу економічну ефективність, значний комерційний потенціал та доцільна для впровадження і практичного застосування у сфері кібербезпеки.

ВИСНОВКИ

У роботі розв’язано поставлене завдання розширення можливостей доменної розвідки для потреб кібербезпеки шляхом розроблення мультиагентної системи, що автоматизує збір та первинний аналіз доменних артефактів із відкритих джерел. Досягнення мети підтверджується виконанням сформульованих у вступі етапів: від аналізу підходів до реалізації прототипу та експериментальної оцінки результатів.

Запропонована архітектура з центральним агентом-координатором та набором спеціалізованих агентів забезпечує паралельне отримання WHOIS, DNS, SSL/TLS, crt.sh, Shodan, VirusTotal і вебконтент-даних та подальшу агрегацію в уніфікований результат. Практична значущість рішення полягає в автоматизації рутинних операцій. Повний цикл комплексного аналізу одного домену займає менше хвилини, що суттєво скорочує час прийняття рішень порівняно з ручною перевіркою.

Достовірність отриманих результатів обґрунтовано експериментальним тестуванням на доменах різних категорій. Для легітимних ресурсів система коректно інтерпретує позитивні індикатори як фактори зниження ризику та формує низькі значення ризикового балу.

На прикладі підтвердженого фішингового ресурсу з бази PhishTank система продемонструвала здатність виявляти аномальні та репутаційні ознаки зловмисності: зафіксовано понад 1000 субдоменів, відсутність WHOIS для конкретного піддомену, наявність характерних маркерів у назві і негативні сигнали репутації, що узгоджується з реальним призначенням домену.

Економічні розрахунки свідчать про доцільність практичного впровадження розробки: рівень комерційного потенціалу оцінено у 39,33 бала, а узагальнений коефіцієнт якості становить 1,75, що відображає конкурентоспроможність рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Куперштейн Л. М., Залепа О. В. До проблеми збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки. *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)»*, Вінниця, 2026 р. URL:__.
2. Domain Names: Understanding the Basics. URL: https://www.rdj.ie/insights/domain-names-understanding-the-basics (accessed: 27.09.2025).
3. European Union Agency for Cybersecurity. ENISA Threat Landscape 2023: July 2022 to June 2023. 2023. URL: https://data.europa.eu/doi/10.2824/782573 (accessed: 28.09.2025).
4. Chen Z., Wang J. J., Kwan K. Newly Registered Domains: Malicious Abuse by Bad Actors. URL: https://unit42.paloaltonetworks.com/newly-registered-domains-malicious-abuse-by-bad-actors/ (accessed: 29.09.2025).
5. Phishing Landscape 2024: An Annual Study of the Scope and Distribution of Phishing. URL: https://interisle.net/insights/phishing-landscape-2024-an-annual-study-of-the-scope-and-distribution-of-phishing (accessed: 30.09.2025).
6. Alieyan K., et al. A survey of botnet detection based on DNS. Neural Computing and Applications. 2017. Vol. 28, no. 7. P. 1541–1558 (accessed: 01.10.2025).
7. Dube I., Wells G. An Analysis of the Use of DNS for Malicious Payload Distribution. 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC). Johannesburg, South Africa, 2020. DOI: https://doi.org/10.1109/IMITEC50163.2020.9334104 (accessed: 02.10.2025).
8. Threat Brief: Understanding Domain Generation Algorithms (DGA). URL: https://unit42.paloaltonetworks.com/threat-brief-understanding-domain-generation-algorithms-dga (accessed: 03.10.2025).

9. Pescatore J. SANS Top New Attacks and Threat Report. SANS Institute, 2019. URL: https://www.sans.org/media/vendor/SANS-Top-New-Attacks-and-Threat-Report.pdf (accessed: 04.10.2025).
10. Bilge L., et al. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS). San Diego, CA, USA, 2011. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/bilg.pdf (accessed: 05.10.2025).
11. Drichel A., et al. Finding Phish in a Haystack: A Pipeline for Phishing Classification on Certificate Transparency Logs. Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES). Vienna, Austria, 2021. URL: https://arxiv.org/pdf/2106.12343 (accessed: 06.10.2025).
12. ICANN. WHOIS and Registration Data Directory Services. URL: https://www.icann.org/resources/pages/whois-rdds-2023-11-02-en (accessed: 07.10.2025).
13. Cloudflare. DNS records. URL: https://www.cloudflare.com/learning/dns/dns-records/ (accessed: 08.10.2025).
14. Laurie B., Messeri E., Stradling R. Certificate Transparency Version 2.0. RFC Editor, 2021 (accessed: 09.10.2025).
15. Kureel V. K., et al. Osint Automation Application. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2023. P. 377–381. DOI: https://doi.org/10.32628/cseit232551 (accessed: 10.10.2025).
16. What is Shodan? URL: https://help.shodan.io/the-basics/what-is-shodan (accessed: 11.10.2025).
17. Censys Documentation. Internet Scanning. URL: https://docs.censys.com/docs/internet-scanning (accessed: 12.10.2025).
18. VirusTotal. How it works. URL: https://docs.virustotal.com/docs/how-it-works (accessed: 13.10.2025).

19. Exploring crt.sh: An essential resource for cybersecurity. URL: https://briacd.com/crt-sh-certificate-search-tool (accessed: 14.10.2025).
20. SecurityTrails. FAQ. URL: https://securitytrails.wpengine.com/knowledge-base/faq/ (accessed: 15.10.2025).
21. Кубрак Ю. О., Плечистий Д. Д., Романішин В. В. Принципи формування мультиагентної системи штучного інтелекту. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. Луцьк, 2022. Вип. 48. С. 76–82. DOI: https://doi.org/10.36910/6775-2524-0560-2022-48-12 (дата звернення: 16.10.2025).
22. Ismail, et al. Toward Robust Security Orchestration and Automated Response in SOCs with a Hyper-Automation Approach Using Agentic AI. Information (MDPI). 2025. Vol. 16(5). Art. 365 (accessed: 17.10.2025).
23. Сухарев Н. О., Куперштейн Л. М. Засіб для розвідки з відкритих джерел на основі ШІ-агентів. *Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи*, Вінниця, 15-16 червня 2025 р. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25724 (дата звернення: 17.10.2025).
24. WhoisXML API. WhoisXML API: Domain, WHOIS, IP, DNS & Threat Intelligence. URL: https://www.whoisxmlapi.com/ (accessed: 18.10.2025).
25. DomainTools. Iris API Documentation. URL: https://docs.domaintools.com/api/iris/ (accessed: 19.10.2025).
26. SpiderFoot. Intel 471 Acquires SpiderFoot. PR Newswire. 02.11.2022. URL: https://www.prnewswire.com/news-releases/intel-471-acquires-spiderfoot-301665787.html (accessed: 20.10.2025).
27. Recon-ng. recon-ng | Kali Linux Tools. URL: https://www.kali.org/tools/recon-ng/ (accessed: 21.10.2025).

28. theHarvester. Installation · theHarvester Wiki. URL: https://github.com/laramies/theHarvester/wiki/Installation (accessed: 22.10.2025).
29. Maltego. Writing Transforms. Maltego Docs. URL: https://docs.maltego.com/en/support/solutions/articles/15000015758-writing-transforms (accessed: 23.10.2025).
30. SYNINT. SYNINT: Agentic OSINT & Intelligence Framework. GitHub repository. URL: https://github.com/gs-ai/SYNINT (accessed: 24.10.2025).
31. OpenAI. Hello GPT-4o. 13.05.2024. URL: https://openai.com/index/hello-gpt-4o/ (accessed: 25.10.2025).
32. OpenAI. GPT-4o mini: advancing cost-efficient intelligence. 18.07.2024. URL: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (accessed: 26.10.2025).
33. Anthropic. Introducing Claude 3.5 Sonnet. 20–21.06.2024. URL: https://www.anthropic.com/news/claude-3-5-sonnet (accessed: 27.10.2025).
34. Google Developers Blog. Gemini 1.5 Pro Now Available in 180+ Countries… URL: https://developers.googleblog.com/gemini-15-pro-now-available-in-180-countries-with-native-audio-understanding-system-instructions-json-mode-and-more/ (accessed: 28.10.2025).
35. Meta AI. Introducing Llama 3.1: Our most capable models to date. 23.07.2024. URL: https://ai.meta.com/blog/meta-llama-3-1/ (accessed: 29.10.2025).
36. Mistral AI. Au Large. 26.02.2024. URL: https://mistral.ai/news/mistral-large (accessed: 30.10.2025).
37. Wang L., Ma C., Feng X., et al. A survey on large language model based autonomous agents. Frontiers of Computer Science. 2024. Vol. 18. Article 186345. DOI: 10.1007/s11704-024-40231-1 (accessed: 31.10.2025).
38. Python. Python Documentation. URL: https://docs.python.org/ (accessed: 01.11.2025).

39. Java. Java Documentation. URL: https://docs.oracle.com/en/java/ (accessed: 02.11.2025).
40. Go. Documentation – The Go Programming Language. URL: https://go.dev/doc/ (accessed: 03.11.2025).
41. C++. C and C++ reference. URL: https://cppreference.com/ (accessed: 04.11.2025).
42. PyCharm. JetBrains. PyCharm Features. URL: https://www.jetbrains.com/pycharm/features/ (accessed: 05.11.2025).
43. Visual Studio Code. Visual Studio Code documentation. URL: https://code.visualstudio.com/docs (accessed: 06.11.2025).
44. Neovim. Nvim documentation. URL: https://neovim.io/doc/user/ (accessed: 07.11.2025).
45. Streamlit. Streamlit documentation. URL: https://docs.streamlit.io/ (accessed: 08.11.2025).
46. Django. Django documentation. URL: https://docs.djangoproject.com/ (accessed: 09.11.2025).
47. Flask. Flask Documentation. URL: https://flask.palletsprojects.com/ (accessed: 10.11.2025).
48. React. React Documentation. URL: https://react.dev/ (accessed: 11.11.2025).
49. Vue. Introduction | Vue.js. URL: https://vuejs.org/guide/introduction (accessed: 12.11.2025).
50. CrewAI. Agents | CrewAI Documentation. URL: https://docs.crewai.com/en/concepts/agents (accessed: 13.11.2025).
51. LangChain. LangChain Agents (Python). URL: https://docs.langchain.com/oss/python/langchain/agents (accessed: 14.11.2025).
52. AutoGen. Microsoft. AutoGen Documentation. URL: https://microsoft.github.io/autogen/stable//index.html (accessed: 15.11.2025).

53. Asynchronous I/O. URL: https://docs.python.org/3/library/asyncio.html (accessed: 16.11.2025).
54. Pydantic. Models – Pydantic Validation. URL: https://docs.pydantic.dev/latest/concepts/models/ (accessed: 17.11.2025).
55. Requests. Requests: HTTP for Humans. URL: https://requests.readthedocs.io/ (accessed: 18.11.2025).
56. dnspython. dnspython documentation. URL: https://dnspython.readthedocs.io/ (accessed: 19.11.2025).
57. Low-level networking interface. URL: https://docs.python.org/3/library/socket.html (accessed: 20.11.2025).
58. TLS/SSL wrapper for socket objects. URL: https://docs.python.org/3/library/ssl.html (accessed: 21.11.2025).
59. pyca/cryptography. Documentation. URL: https://cryptography.io/ (accessed: 22.11.2025).
60. Shodan SDK. The official Python library for the Shodan search engine. URL: https://shodan.readthedocs.io/ (accessed: 23.11.2025).
61. VirusTotal API v3. Overview. URL: https://docs.virustotal.com/reference/overview (accessed: 24.11.2025).
62. Козловський В. О., Лесько О. Й., Кавецький В. В.Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця. ВНТУ. 2021. 42 с. (дата звернення: 25.11.2025).
63. Кавецький В. В., Козловський В. О., Причепа І. В. Економічне обґрунтування інноваційних рішень: практикум. Вінниця. ВНТУ. 2016. 113 с. (дата звернення: 26.11.2025).

ДОДАТКИ

Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Система збору доменної інформації на основі мультиагентного підходу для задач кібербезпеки

Автор роботи: Залепа Олександр Вячеславович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **3, 58** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к. т. н., доц. _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Леонід КУПЕРШТЕЙН

Здобувач _____ Олександр ЗАЛЕПА

Додаток Б

Формалізований опис логіки аналітичного агента

Роль агента аналітика:

Senior Cybersecurity Analyst

Ціль агента аналітика:

```
На основі OSINT-даних (WHOIS, DNS, SSL, CRT, Shodan, VirusTotal,
Content) оцінити безпечність домену, виявити ознаки фішингу або
шахрайства та надати практичні рекомендації.
```

Переісторія агента аналітика:

```
Досвідчений аналітик кібербезпеки з практикою SOC та Threat
Intelligence. Регулярно аналізує фішингові кампанії, молоді домени,
шкідливу інфраструктуру та підозрілі веб-ресурси. Вміє перетворювати
технічні артефакти на зрозумілий, аргументований висновок.
```

Опис завдання агента аналітика:

```
Ти — аналітик з кібербезпеки, який оцінює репутацію та безпечність
доменів на основі OSINT-даних.

Нижче ти отримаєш ОДИН JSON-об’єкт у полі facts — це повний результат
усіх підлеглих агентів (whois, dns, ssl, crt, shodan, virustotal,
content).

{{facts}}

На основі цього JSON побудуй РІВНО ОДИН валідний JSON-об’єкт такої
структури:
{
  "domain": "string",
  "risk_score": 0-100 (ціле число),
  "risk_level": "low" | "medium" | "high",
  "classification": "benign" | "suspicious" | "malicious_like",
  "markdown": "Повний markdown-звіт українською мовою з технічним
аналізом домену.",
  "analysis": {
    "summary": "1-2 абзаци українською з коротким висновком про
безпечність домену.",
    "details_ua": [
      "Короткі тези українською з поясненням ключових технічних
фактів.",
```

```
      "Кожен елемент — одне речення."
    ],
    "evidence": [
      {
        "source": "whois|dns|ssl|crt|shodan|virustotal|other",
        "field": "наприклад: whois.creation_date або
shodan.hosts[0].asn",
        "fact": "Короткий опис факту українською.",
        "impact": "Як цей факт впливає на ризик (підвищує, знижує,
нейтральний)."
      }
    ],
    "missing_data": [
      "список типів даних, яких бракує для повнішого аналізу,
наприклад: content_analysis, passive_dns"
    ],
    "recommendations_ua": [
      "Практичні рекомендації українською:
дозволити/моніторити/обмежити використання домену і т.д."
    ]
  }
}

====================
НОРМАЛІЗАЦІЯ CONTENT
====================
У facts може бути блок "content" або "content_analysis".
Якщо є "content", трактуй його як content_analysis з мапінгом:
- title <- content.title
- language <- content.language
- meta <- content.meta
- headings <- content.headings
- main_text_snippet <- перші 300–600 символів content.text
- links_sample <- перші 10 content.links

Обов’язково використовуй ці дані для визначення призначення сайту,
виявлення фішингових або шахрайських патернів.

====================
МАТРИЦЯ СИГНАЛІВ РИЗИКУ
====================
Використовуй ТІЛЬКИ факти з facts.
Вік домену <30 днів → +25
Вік домену 30–90 днів → +20
dns-parking у NS → +10
Порожній або шаблонний контент → +15
Title/meta типу 'Hostinger', 'Horizons', 'Parking' → +15
DMARC p=none → +5
```

```
Shodan: 'Checking your browser', JS challenge → +10
VT malicious >0 → risk_score >=80, classification=malicious_like

====================
SUBDOMAIN OVERFLOW
====================
Якщо кількість піддоменів:
- >50 → +15 до risk_score
- >200 → +25 до risk_score
Типові підозрілі назви піддоменів: login, secure, verify, update,
recovery.

У details_ua ОБОВ’ЯЗКОВО зазнач кількість піддоменів та потенційний
ризик.

====================
СТРОГІ ПРАВИЛА
====================
1) Використовуй ТІЛЬКИ дані з facts.
2) НІЧОГО не вигадуй.
3) Якщо блок даних відсутній — додай його назву у missing_data.
4) evidence: мінімум 8 елементів.
5) VT = 0 detections НЕ означає, що домен безпечний.
6) У відповіді ПОВИНЕН бути лише один валідний JSON.
7) Усі текстові поля — українською мовою.
```

Очікуваний результат після виконання завдання:

```
Рівно один валідний JSON з полями domain, risk_score, risk_level,
classification, markdown, analysis.
```

Додаток В

Текст програмного застосунку

Файл coordinator.py

```
from __future__ import annotations

import socket
import traceback

from agents.dns_agent import DNSAgent
from agents.ssl_agent import SSLAgent
from agents.whois_agent import WhoisAgent
from agents.crt_agent import CrtAgent
from agents.shodan_agent import ShodanAgent
from agents.virustotal_agent import VirusTotalAgent
from agents.content_agent import ContentAgent

from models import DomainResult
from settings import ENABLE_CREWAI
from concurrent.futures import ThreadPoolExecutor, wait, FIRST_COMPLETED

def _resolve_ipv4_bulk(hosts: list[str]) -> list[str]:
    ips = set()
    for h in hosts:
        h = (h or "").strip().lower().strip(".")
        if not h:
            continue
        try:
            for info in socket.getaddrinfo(h, None):
                ip = info[4][0]
                if ip and ip.count(".") == 3:
                    ips.add(ip)
        except Exception:
            continue
    return list(ips)


class Coordinator:
    def __init__(
        self,
        use_crewai: bool | None = None,
        max_workers: int = 8,
    ) -> None:
        self.use_crewai = ENABLE_CREWAI if use_crewai is None else use_crewai
        self.max_workers = max_workers

    def run_for_domain(self, domain: str) -> DomainResult:
        result = DomainResult(domain=domain)

        whois_agent = WhoisAgent()
        dns_agent = DNSAgent()
        ssl_agent = SSLAgent()
        crt_agent = CrtAgent()
        shodan_agent = ShodanAgent()
        virustotal_agent = VirusTotalAgent()
        content_agent = ContentAgent()

        intermediate: dict[str, object] = {}

        with ThreadPoolExecutor(max_workers=self.max_workers) as ex:
            futures: dict[object, str] = {
                ex.submit(whois_agent.run, domain): "whois",
                ex.submit(dns_agent.run, domain): "dns",
                ex.submit(ssl_agent.run, domain): "ssl",
                ex.submit(content_agent.run, domain): "content",
                ex.submit(crt_agent.run, domain): "crt",
                ex.submit(virustotal_agent.run, domain): "virustotal",
```

```
            }

            pending = set(futures.keys())
            shodan_submitted = False

            while pending:
                done, pending = wait(pending, return when=FIRST COMPLETED)

                for fut in done:
                    name = futures.get(fut, "unknown")
                    try:
                        intermediate[name] = fut.result()
                    except Exception:
                        result.add error(name, traceback.format exc())
                        intermediate[name] = None

                    # Запускаємо Shodan одразу після DNS (але 1 раз)
                    if name == "dns" and not shodan submitted:
                        dns_res = intermediate.get("dns")
                        a ips: list[str] = []

                        # A з кореневого домену
                        if dns_res and getattr(dns_res, "records", None):
                            a list = (dns res.records.get("A", []) or [])
                            a_ips = [str(ip).strip() for ip in a_list if str(ip).strip().count(".") ==
3]

                        # Якщо A порожні — підстрахуємось IP з піддоменів
                        if not a_ips:
                            hosts: list[str] = []

                            crt res = intermediate.get("crt")
                            if crt_res and getattr(crt_res, "crtsh_names", None):
                                hosts.extend(list(crt_res.crtsh_names)[:200])

                            if dns res and getattr(dns res, "subdomains found", None):
                                hosts.extend(list(dns res.subdomains found)[:200])

                            a_ips = _resolve_ipv4_bulk(hosts)

                        shodan agent.set shared("a records", a ips)
                        sh_fut = ex.submit(shodan_agent.run, domain)
                        futures[sh fut] = "shodan"
                        pending.add(sh_fut)
                        shodan_submitted = True

        result.whois = intermediate.get("whois")
        result.dns = intermediate.get("dns")
        result.ssl = intermediate.get("ssl")
        result.content = intermediate.get("content")
        result.crt = intermediate.get("crt")
        result.virustotal = intermediate.get("virustotal")
        result.shodan = intermediate.get("shodan")

        return result
```

Файл crew_setup.py

```
from __future__ import annotations
"""
Розширений приклад інтеграції з CrewAI для фінального аналітика.
"""
try:
    from crewai import Agent, Task, Crew
except Exception:
    Agent = Task = Crew = None

from typing import Optional
from models import DomainResult, ContentResult
```

```
import json


def build_full_facts(domain_result: DomainResult) -> str:
    payload = {
        "domain": domain_result.domain,
        "whois": domain_result.whois.model_dump() if domain_result.whois else None,
        "dns": domain_result.dns.model_dump() if domain_result.dns else None,
        "ssl": domain_result.ssl.model_dump() if domain_result.ssl else None,
        "crt": domain_result.crt.model_dump() if domain_result.crt else None,
        "shodan": domain_result.shodan.model_dump() if domain_result.shodan else None,
        "virustotal": domain_result.virustotal.model_dump() if domain_result.virustotal else None,
    }
    if domain_result.content:
        c: ContentResult = domain_result.content

        lines = (c.text or "").splitlines()
        snippet_lines = lines[:40]
        main_text_snippet = "\n".join(snippet_lines)

        payload["content_analysis"] = {
            "url": c.final_url or c.url,
            "title": c.title,
            "language": c.language,
            "meta": c.meta,
            "headings": c.headings[:30],
            "main_text_snippet": main_text_snippet,
            "links_sample": c.links[:50],
        }
    else:
        payload["content_analysis"] = None

    return json.dumps(payload, ensure_ascii=False, indent=2, default=str)


def make_analyst_crew() -> Optional[Crew]:
    if Crew is None:
        return None

    analyst = Agent(
        role="Senior Cybersecurity Analyst",
        goal=(
            "На основі OSINT-даних (WHOIS, DNS, SSL, CRT, Shodan, VirusTotal, Content) "
            "оцінити безпечність домену, виявити ознаки фішингу або шахрайства "
            "та надати практичні рекомендації."
        ),
        backstory=(
            "Досвідчений аналітик кібербезпеки з практикою SOC та Threat Intelligence. "
            "Регулярно аналізує фішингові кампанії, молоді домени, "
            "шкідливу інфраструктуру та підозрілі веб-ресурси. "
            "Вміє перетворювати технічні артефакти на зрозумілий, аргументований висновок."
        ),
        allow_delegation=False,
        verbose=True,
    )

    summarize_task = Task(
        description=(
            "Ти — аналітик з кібербезпеки, який оцінює репутацію та безпечність доменів "
            "на основі OSINT-даних.\n\n"

            "Нижче ти отримаєш ОДИН JSON-об’єкт у полі facts — це повний результат "
            "усіх підлеглих агентів (whois, dns, ssl, crt, shodan, virustotal, content).\n\n"
            "{{facts}}\n\n"

            "На основі цього JSON побудуй РІВНО ОДИН валідний JSON-об’єкт такої структури:\n"
            "{\n"
            "  \"domain\": \"string\",\n"
            "  \"risk_score\": 0-100 (ціле число),\n"
            "  \"risk_level\": \"low\" | \"medium\" | \"high\",\n"
            "  \"classification\": \"benign\" | \"suspicious\" | \"malicious_like\",\n"
            "  \"markdown\": \"Повний markdown-звіт українською мовою з технічним аналізом
домену.\",\n"
```

```
            "  \"analysis\": {\n"
            "    \"summary\": \"1-2 абзаци українською з коротким висновком про безпечність
домену.\",\n"
            "    \"details_ua\": [\n"
            "      \"Короткі тези українською з поясненням ключових технічних фактів.\",\n"
            "      \"Кожен елемент — одне речення.\"\n"
            "    ],\n"
            "    \"evidence\": [\n"
            "      {\n"
            "        \"source\": \"whois|dns|ssl|crt|shodan|virustotal|other\",\n"
            "        \"field\": \"наприклад: whois.creation_date або shodan.hosts[0].asn\",\n"
            "        \"fact\": \"Короткий опис факту українською.\",\n"
            "        \"impact\": \"Як цей факт впливає на ризик (підвищує, знижує, нейтральний).\"\n"
            "      }\n"
            "    ],\n"
            "    \"missing_data\": [\n"
            "      \"список типів даних, яких бракує для повнішого аналізу, наприклад:
content analysis, passive dns\"\n"
            "    ],\n"
            "    \"recommendations ua\": [\n"
            "      \"Практичні рекомендації українською: дозволити/моніторити/обмежити використання
домену і т.д.\"\n"
            "    ]\n"
            "  }\n"
            "}\n\n"

            "====================\n"
            "НОРМАЛІЗАЦІЯ CONTENT\n"
            "====================\n"
            "У facts може бути блок \"content\" або \"content analysis\".\n"
            "Якщо є \"content\", трактуй його як content_analysis з мапінгом:\n"
            "- title <- content.title\n"
            "- language <- content.language\n"
            "- meta <- content.meta\n"
            "- headings <- content.headings\n"
            "- main text snippet <- перші 300-600 символів content.text\n"
            "- links sample <- перші 10 content.links\n\n"

            "Обов’язково використовуй ці дані для визначення призначення сайту, "
            "виявлення фішингових або шахрайських патернів.\n\n"

            "====================\n"
            "МАТРИЦЯ СИГНАЛІВ РИЗИКУ\n"
            "====================\n"
            "Використовуй ТІЛЬКИ факти з facts.\n"
            "Вік домену <30 днів → +25\n"
            "Вік домену 30-90 днів → +20\n"
            "dns-parking у NS → +10\n"
            "Порожній або шаблонний контент → +15\n"
            "Title/meta типу 'Hostinger', 'Horizons', 'Parking' → +15\n"
            "DMARC p=none → +5\n"
            "Shodan: 'Checking your browser', JS challenge → +10\n"
            "VT malicious >0 → risk score >=80, classification=malicious like\n\n"

            "====================\n"
            "SUBDOMAIN OVERFLOW\n"
            "====================\n"
            "Якщо кількість піддоменів:\n"
            "- >50 → +15 до risk score\n"
            "- >200 → +25 до risk_score\n"
            "Типові підозрілі назви піддоменів: login, secure, verify, update, recovery.\n\n"

            "У details_ua ОБОВ’ЯЗКОВО зазнач кількість піддоменів та потенційний ризик.\n\n"

            "====================\n"
            "СТРОГІ ПРАВИЛА\n"
            "====================\n"
            "1) Використовуй ТІЛЬКИ дані з facts.\n"
            "2) НІЧОГО не вигадуй.\n"
            "3) Якщо блок даних відсутній — додай його назву у missing_data.\n"
            "4) evidence: мінімум 8 елементів.\n"
            "5) VT = 0 detections НЕ означає, що домен безпечний.\n"
            "6) У відповіді ПОВИНЕН бути лише один валідний JSON.\n"
```

```
            "7) Усі текстові поля — українською мовою.\n"
        ),
        agent=analyst,
        expected_output=(
            "Рівно один валідний JSON з полями domain, risk_score, risk_level, "
            "classification, markdown, analysis."
        )
    )

    crew = Crew(
        agents=[analyst],
        tasks=[summarize_task],
        verbose=True,
    )
    return crew
```

Файл crewai_agents.py

```
from __future__ import annotations
import json
from typing import Tuple, Dict

from pydantic import BaseModel, Field
from crewai import Agent, Task, Crew, Process
from crewai.tools import BaseTool

from models import (
    WhoisResult,
    DNSResult,
    SSLResult,
    CrtResult,
    ShodanResult,
    DomainResult,
    VirusTotalResult,
    AnalysisResult,
    ContentResult,
)
from agents.whois_agent import WhoisAgent
from agents.dns_agent import DNSAgent
from agents.ssl_agent import SSLAgent
from agents.crt_agent import CrtAgent
from agents.shodan_agent import ShodanAgent
from agents.virustotal_agent import VirusTotalAgent
from agents.content_agent import ContentAgent  # NEW

class DomainInput(BaseModel):
    domain: str = Field(..., description="Цільовий домен, напр. example.com")


class WhoisTool(BaseTool):
    name: str = "whois_lookup"
    description: str = "Виконай WHOIS-запит і поверни структурований JSON (WhoisResult)."
    args_schema: type[BaseModel] = DomainInput

    def _run(self, domain: str) -> dict:
        return WhoisAgent().run(domain).model_dump()


class DNSTool(BaseTool):
    name: str = "dns_enumeration"
    description: str = "Збери DNS-записи (A/AAAA/NS/MX/TXT/CNAME/SOA) та легкий bruteforce субдоменів."
    args_schema: type[BaseModel] = DomainInput

    def _run(self, domain: str) -> dict:
        return DNSAgent().run(domain).model_dump()


class SSLTool(BaseTool):
    name: str = "ssl cert intel"
    description: str = "Зчитай TLS-сертифікат з 443 і поверни деталі
(CN/Issuer/SAN/validity/fingerprint)."
    args_schema: type[BaseModel] = DomainInput
```

```
    def _run(self, domain: str) -> dict:
        return SSLAgent().run(domain).model_dump()


class CrtTool(BaseTool):
    name: str = "crt passive"
    description: str = "Збери пасивні домени через crt.sh."
    args schema: type[BaseModel] = DomainInput

    def _run(self, domain: str) -> dict:
        return CrtAgent().run(domain).model_dump()


class ShodanTool(BaseTool):
    name: str = "shodan_scan"
    description: str = "Отримай інформацію про хости через Shodan (за API key)."
    args schema: type[BaseModel] = DomainInput

    def  run(self, domain: str) -> dict:
        return ShodanAgent().run(domain).model_dump()


class VirusTotalTool(BaseTool):
    name: str = "virustotal_lookup"
    description: str = "Отримай VirusTotal репутацію та артефакти для домену."
    args_schema: type[BaseModel] = DomainInput

    def _run(self, domain: str) -> dict:
        return VirusTotalAgent().run(domain).model dump()

class ContentTool(BaseTool):
    name: str = "content_fetch"
    description: str = (
        "Завантаж головну сторінку сайту і поверни структурований аналіз вмісту "
        "у форматі JSON (ContentResult): title, meta, текст, заголовки, посилання."
    )
    args_schema: type[BaseModel] = DomainInput

    def _run(self, domain: str) -> dict:
        return ContentAgent().run(domain).model dump()

# ---------- Побудова команди та задач ----------
def make_domain_crew() -> Tuple[Crew, Dict[str, Task]]:
    """Створює Crew з фахівців + фінальний агент-аналітик; повертає (crew, tasks)."""
    whois_tool = WhoisTool()
    dns tool = DNSTool()
    ssl_tool = SSLTool()
    crt tool = CrtTool()
    shodan_tool = ShodanTool()
    vt_tool = VirusTotalTool()
    content_tool = ContentTool()

    whois_specialist = Agent(
        role="WHOIS Specialist",
        goal=(
            "Отримати повну, валідну та структуровану інформацію WHOIS про домен "
            "у форматі JSON, включаючи дані про реєстратора, дати створення/закінчення, "
            "стани домену, контактну інформацію (якщо доступна) та технічні поля."
        ),
        backstory=(
            "Ти — досвідчений аналітик доменних реєстрів, який розуміє специфіку "
            "WHOIS-запитів для різних TLD та уміє інтерпретувати нетипові формати "
            "відповідей. Знаєш, що частина полів може бути прихована."
        ),
        tools=[whois tool],
        allow_delegation=False,
        verbose=True,
    )

    dns_specialist = Agent(
        role="DNS Specialist",
        goal=(
```

```
        "Побудувати повну та достовірну картину DNS-структури досліджуваного домену, "
        "включаючи A, AAAA, NS, MX, TXT, CNAME-записи та знайдені субдомени. "
        "Всі результати повинні бути повернуті у форматі JSON."
    ),
    backstory=(
        "Ти — аналітик DNS-інфраструктури, який чудово знає принципи роботи DNS "
        "і здатний інтегрувати результати активного та пасивного збору."
    ),
    tools=[dns tool],
    allow_delegation=False,
    verbose=True,
)

ssl specialist = Agent(
    role="SSL/TLS Specialist",
    goal=(
        "Отримати з сервера цільового домену дійсний SSL/TLS-сертифікат, розібрати його "
        "структуру та повернути усі релевантні поля у форматі JSON."
    ),
    backstory=(
        "Ти — криптоаналітик, який спеціалізується на сертифікатах безпеки."
    ),
    tools=[ssl_tool],
    allow delegation=False,
    verbose=True,
)

crt_specialist = Agent(
    role="crt.sh Specialist",
    goal=("Зібрати перелік доменів/піддоменів, пов'язаних з цільовим доменом, на основі crt.sh."),
    backstory=("Ти — аналітик сертифікатів, який вміє працювати з базою crt.sh."),
    tools=[crt tool],
    allow_delegation=False,
    verbose=True,
)

shodan specialist = Agent(
    role="Shodan Specialist",
    goal=("Отримати інформацію про публічні хости (по IP) через Shodan API."),
    backstory=("Ти — мережевий аналітик, що знає структуру Shodan результатів."),
    tools=[shodan tool],
    allow_delegation=False,
    verbose=True,
)

vt_specialist = Agent(
    role="VirusTotal Specialist",
    goal=(
        "Отримати репутаційні метрики домену у VirusTotal та пов’язані артефакти, "
        "повернути строго JSON за моделлю VirusTotalResult."
    ),
    backstory=("Ти знаєш обмеження тарифів VT і дбаєш про валідність JSON."),
    tools=[vt tool],
    allow_delegation=False,
    verbose=True,
)

analyst = Agent(
    role="Security Analyst",
    goal=(
        "На основі повних результатів WHOIS/DNS/SSL/CRT/Shodan/VirusTotal виконати "
        "комплексний аналіз домену та надати практичні рекомендації."
    ),
    backstory=(
        "Досвідчений фахівець з мережевої безпеки, який уміє консолідувати технічні "
        "артефакти в ясні висновки та план дій."
    ),
    allow_delegation=False,
    verbose=True,
)

content specialist = Agent(
    role="Content Analyst",
```

```
    goal=(
        "Отримати та проаналізувати вміст вебсайту (головної сторінки домену), "
        "щоб визначити призначення ресурсу, бренд, можливі ознаки фішингу/шахрайства "
        "та інші важливі характеристики."
    ),
    backstory=(
        "Ти — OSINT-аналітик, який спеціалізується на ручному та автоматизованому "
        "аналізі вебсторінок: звертаєш увагу на заголовки, текст, посилання, мову та стиль."
    ),
    tools=[content_tool],
    allow_delegation=False,
    verbose=True,
)

coordinator = Agent(
    role="Coordinator",
    goal=(
        "Організувати послідовне та узгоджене виконання підзадач і передати їхні повні "
        "результати аналітику для фінального висновку."
    ),
    backstory=(
        "Ти — координатор мультиагентної системи. Стежиш за якістю та цілісністю "
        "результатів."
    ),
    allow_delegation=True,
    verbose=True,
)

t_whois = Task(
    description=(
        "Виконай повний WHOIS-запит для домену {domain}. "
        "Використай інструмент whois lookup і поверни валідний JSON WhoisResult."
    ),
    agent=whois_specialist,
    expected_output="JSON-об’єкт типу WhoisResult без додаткового тексту.",
    output pydantic=WhoisResult,
    tools=[whois tool],
)

t_dns = Task(
    description=(
        "Отримай DNS-записи (A, AAAA, MX, NS, TXT, CNAME) і пасивні субдомени для {domain}. "
        "Поверни валідний JSON DNSResult."
    ),
    agent=dns_specialist,
    expected_output="JSON-об’єкт типу DNSResult без додаткового тексту.",
    output pydantic=DNSResult,
    tools=[dns_tool],
)

t_ssl = Task(
    description=(
        "Зніми X.509-сертифікат із 443/tcp для {domain} та поверни JSON SSLResult."
    ),
    agent=ssl specialist,
    expected_output="JSON-об’єкт типу SSLResult без додаткового тексту.",
    output_pydantic=SSLResult,
    tools=[ssl_tool],
)

t crt = Task(
    description=("Збери пасивні домени з crt.sh для {domain} та поверни JSON CrtResult."),
    agent=crt_specialist,
    expected output="JSON-об’єкт типу CrtResult без додаткового тексту.",
    output_pydantic=CrtResult,
    tools=[crt tool],
)

t_shodan = Task(
    description=("Отримай дані Shodan по IP для {domain} та поверни JSON ShodanResult."),
    agent=shodan_specialist,
    expected output="JSON-об’єкт типу ShodanResult без додаткового тексту.",
    output_pydantic=ShodanResult,
```

```
        tools=[shodan tool],
    )

    t_vt = Task(
        description=(
            "Отримай у VirusTotal репутацію та пов’язані дані для {domain}. "
            "Поверни строго валідний JSON VirusTotalResult."
        ),
        agent=vt specialist,
        expected_output="JSON-об’єкт типу VirusTotalResult без вільного тексту.",
        output_pydantic=VirusTotalResult,
        tools=[vt_tool],
    )

    t_content = Task(
        description=(
            "Завантаж головну сторінку для {domain} (за замовчуванням https://{domain}/). "
            "Витягни заголовок сторінки, основні meta (description, keywords, og:*), "
            "H1-H3 заголовки, основний текст та посилання. "
            "Поверни строго валідний JSON-об’єкт типу ContentResult без додаткового тексту."
        ),
        agent=content_specialist,
        expected_output="JSON-об’єкт типу ContentResult без додаткового тексту.",
        output pydantic=ContentResult,
        tools=[content_tool],
    )

    t_analysis = Task(
        description=(
            "Ти отримаєш ПОВНІ JSON-відповіді попередніх задач у контексті. "
            "Зроби комплексний аналіз ризиків (WHOIS/DNS/SSL/CRT/Shodan/VT) і сформуй результат "
            "в РІВНО ОДНОМУ JSON-об’єкті такого вигляду:\n\n"
            "{"
            "\"markdown\": \"<повний markdown-звіт>\", "
            "\"analysis\": {"
            "\"summary\": \"<1-2 абзаци>\", "
            "\"risks\": [\"\"- ...\"\"], "
            "\"recommendations\": [\"\"- ...\"\"], "
            "\"raw_markdown\": \"<копія markdown>\""
            "}"
            "}\n\n"
            "Без будь-якого зайвого тексту поза цим JSON."
        ),
        agent=analyst,
        expected_output=(
            "Рівно один JSON із полями {markdown,
analysis{summary,risks[],recommendations[],raw markdown}}."
        ),
        context=[t whois, t dns, t ssl, t crt, t shodan, t vt, t content],
    )



    crew = Crew(
        agents=[
            whois_specialist,
            dns_specialist,
            ssl_specialist,
            crt specialist,
            shodan_specialist,
            vt specialist,
            content_specialist,
            analyst,
        ],
        tasks=[t_whois, t_dns, t_ssl, t_crt, t_shodan, t_vt, t_content, t_analysis],
        process=Process.hierarchical,
        manager_agent=coordinator,
        verbose=True,
    )

    return crew, {
        "whois": t whois,
        "dns": t_dns,
```

```
        "ssl": t_ssl,
        "crt": t_crt,
        "shodan": t_shodan,
        "vt": t_vt,
        "content": t_content,  # NEW
        "analysis": t_analysis,
    }


def run_domain_with_crewai(domain: str) -> DomainResult:
    crew, tasks = make_domain_crew()
    _ = crew.kickoff(inputs={"domain": domain})

    def  load(task: Task, model):
        try:
            data = json.loads(task.output.json)
            return model(**data)
        except Exception:
            try:
                raw = getattr(task.output, "raw", None)
                if isinstance(raw, dict):
                    return model(**raw)
                if isinstance(raw, str):
                    return model(**json.loads(raw))
            except Exception:
                return None
        return None

    whois = _load(tasks["whois"], WhoisResult)
    dns =  load(tasks["dns"], DNSResult)
    ssl = _load(tasks["ssl"], SSLResult)
    crt =  load(tasks["crt"], CrtResult)
    shodan = _load(tasks["shodan"], ShodanResult)
    vt = _load(tasks["vt"], VirusTotalResult)
    content = _load(tasks["content"], ContentResult)

    analysis = None
    try:
        raw = getattr(tasks["analysis"].output, "raw", None) or getattr(tasks["analysis"].output,
"json", None)
        payload = raw if isinstance(raw, dict) else json.loads(raw)
        md = payload.get("markdown")
        analysis_obj = payload.get("analysis", {})
        if md and "raw_markdown" not in analysis_obj:
            analysis_obj["raw_markdown"] = md
        analysis = AnalysisResult(**analysis_obj)
    except Exception:
        analysis = None

    return DomainResult(
        domain=domain,
        whois=whois,
        dns=dns,
        ssl=ssl,
        crt=crt,
        shodan=shodan,
        virustotal=vt,
        content=content,
        analysis=analysis,
    )
```

Додаток Г

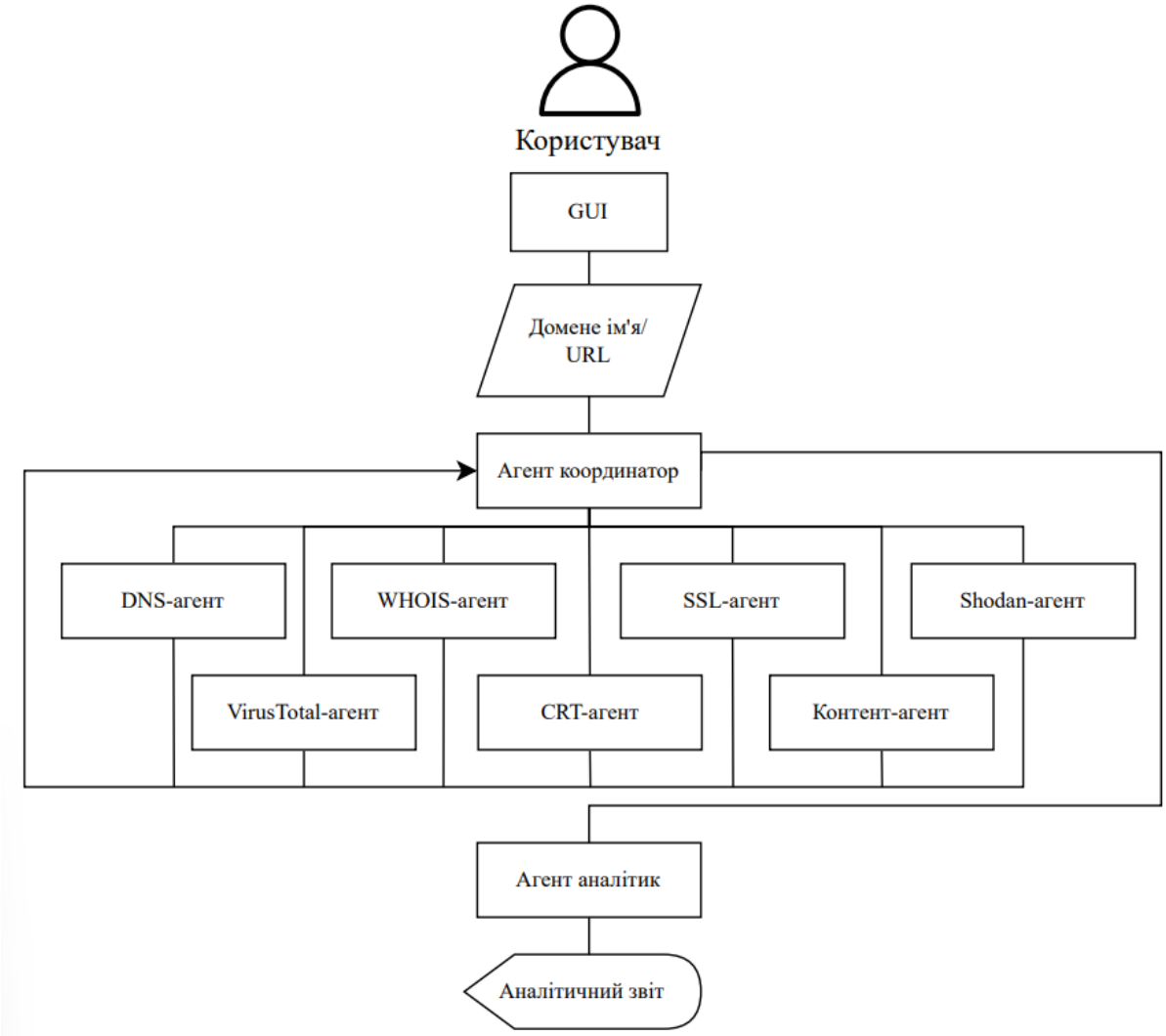
ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА ЗБОРУ ДОМЕННОЇ ІНФОРМАЦІЇ НА ОСНОВІ МУЛЬТИАГЕНТНОГО ПІДХОДУ ДЛЯ ЗАДАЧ КІБЕРБЕЗПЕКИ

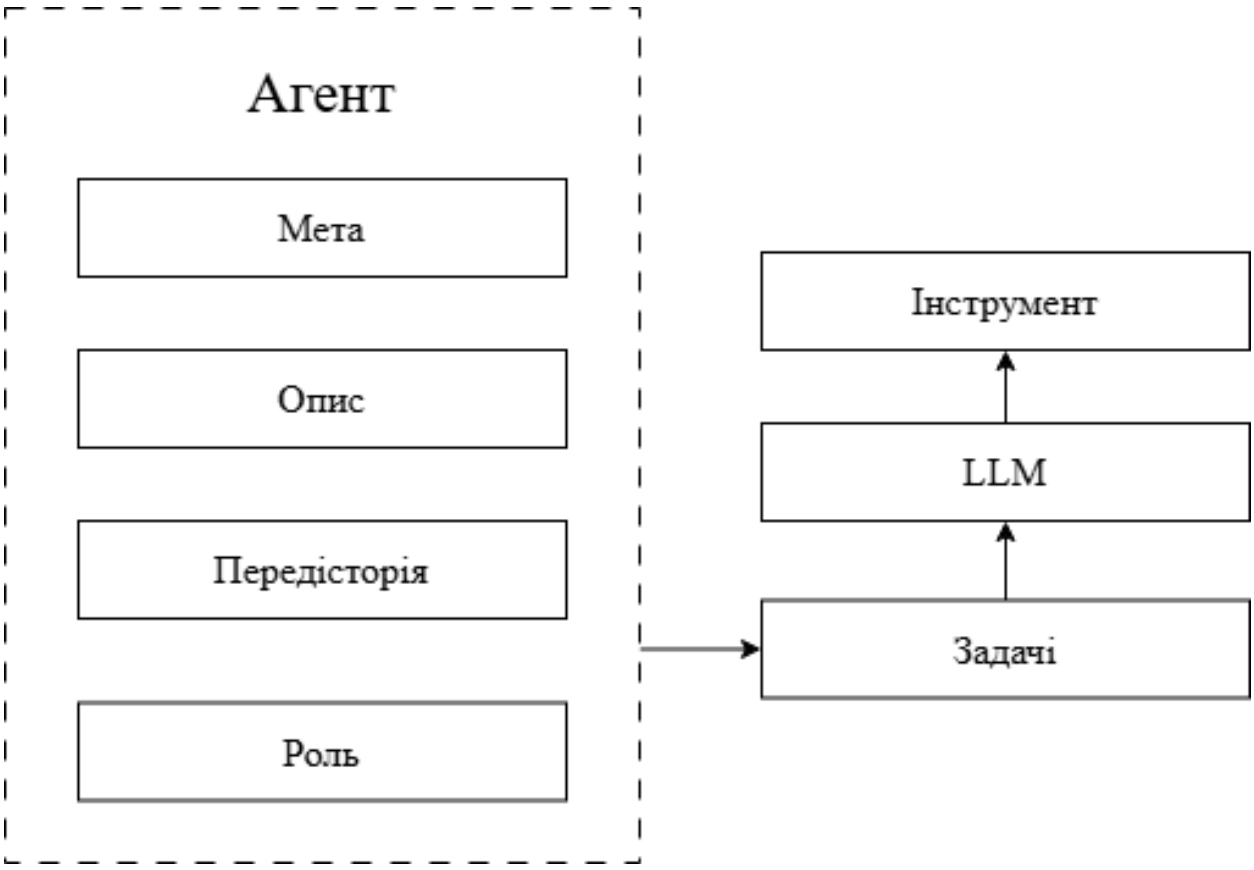
Виконав: студент 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____Олександр ЗАЛЕПА

Керівник: : к.,т. н., доцент каф. ЗІ.
_____Леонід КУПЕРШТЕЙН
«16» грудня 2025 р.

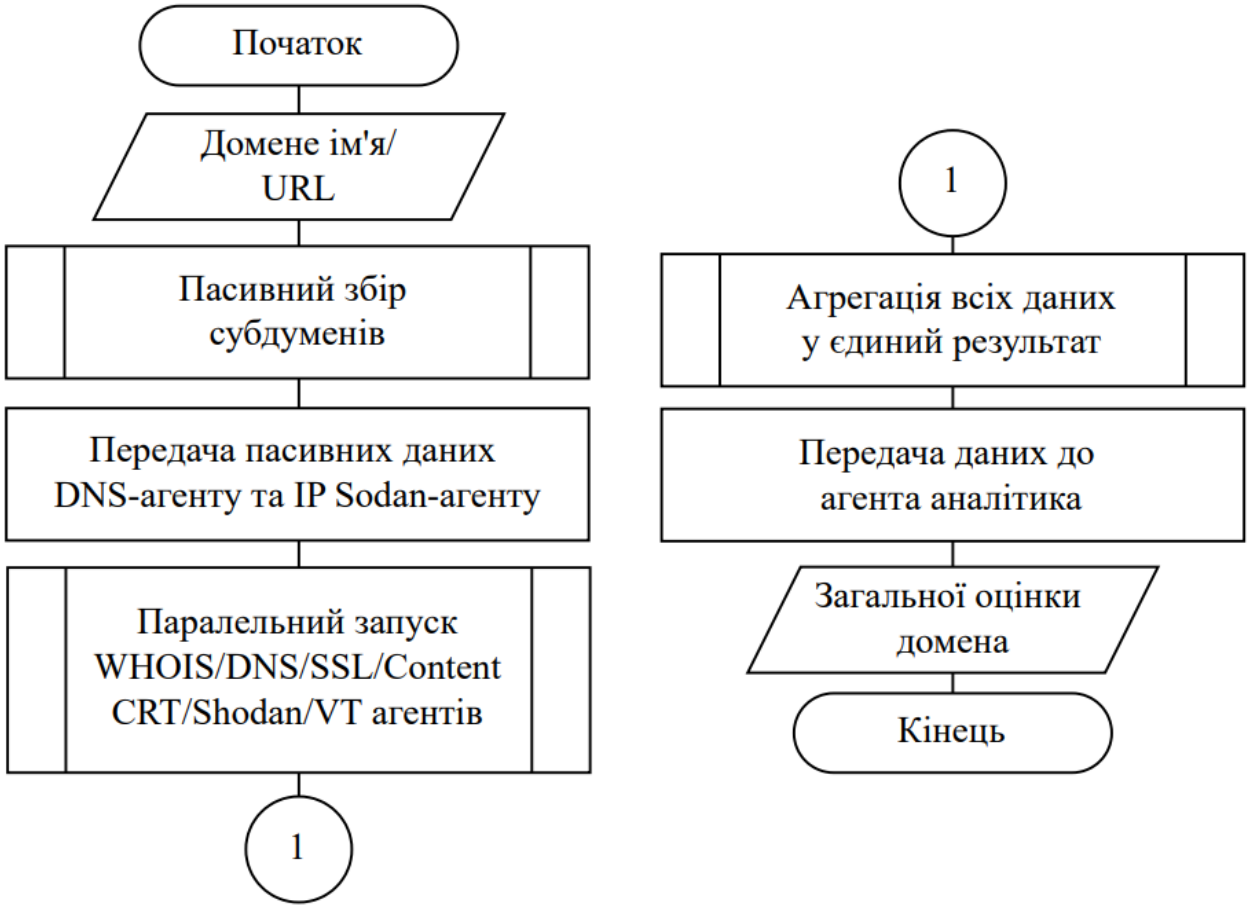
АРХІТЕКТУРА МУЛЬТИАГЕНТНОЇ СИСТЕМИ



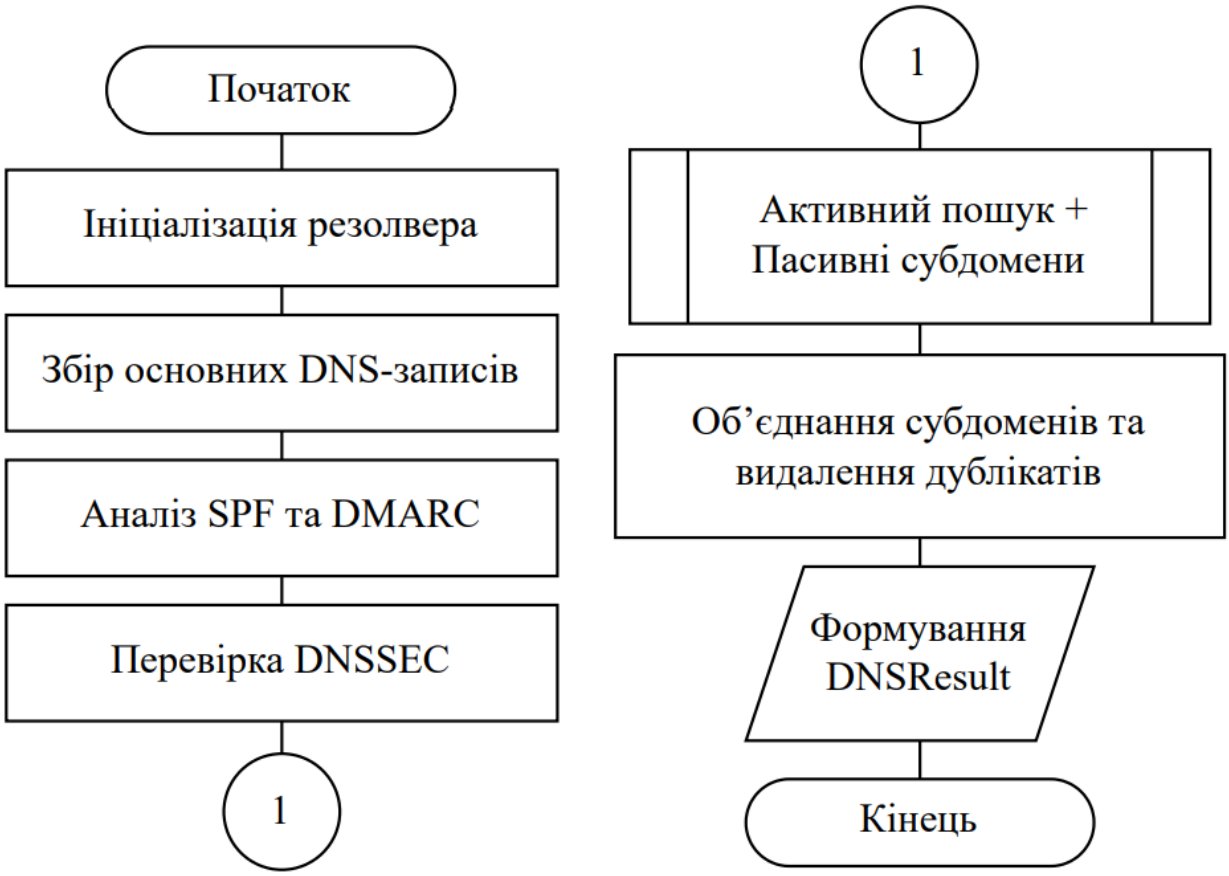
ЗАГАЛЬНА СТРУКТУРНА СХЕМА ШІ-АГЕНТА



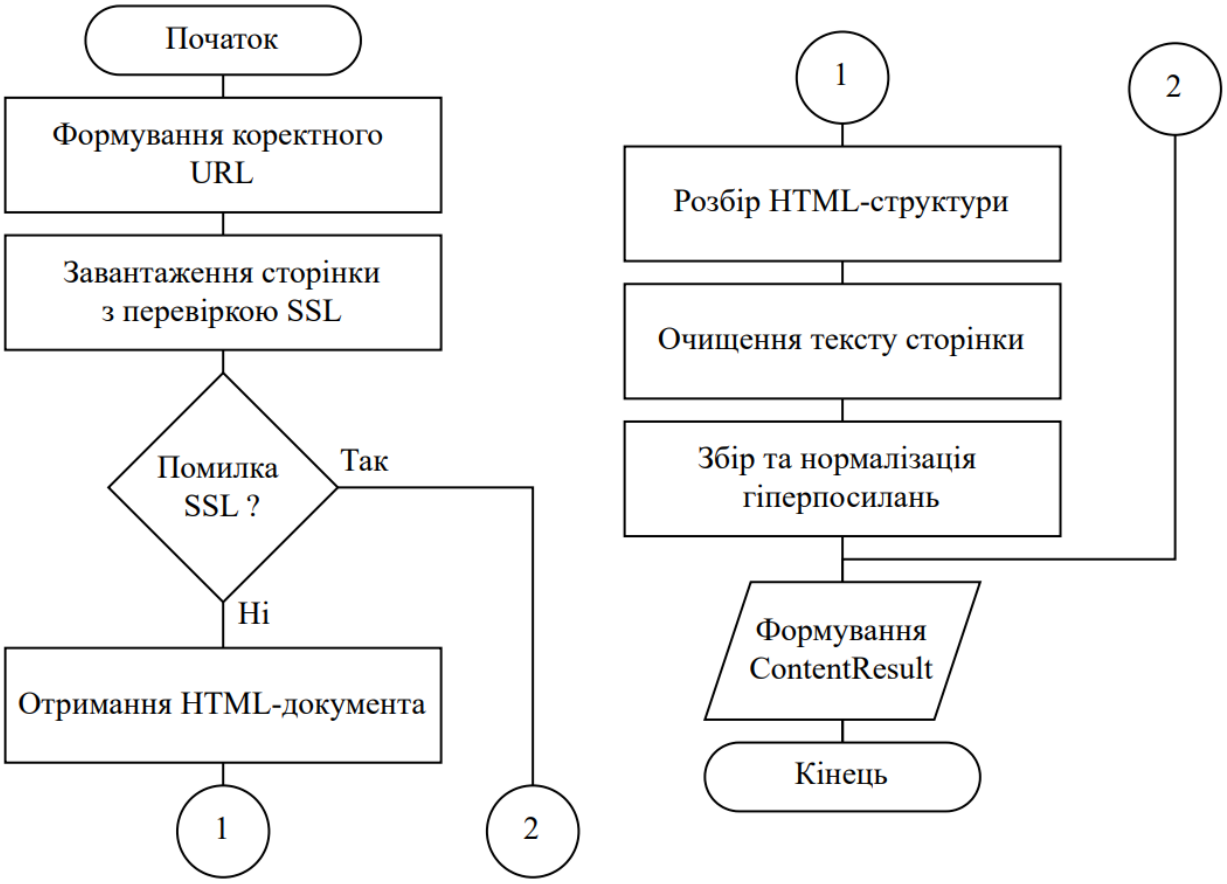
БЛОК-СХЕМА АЛГОРИТМУ РОБОТИ КООРДИНАТОРА МУЛЬТИАГЕНТНОЇ СИСТЕМИ



БЛОК-СХЕМА АЛГОРИТМУ РОЗШИРЕНОГО DNS-АНАЛІЗУ



БЛОК-СХЕМА АЛГОРИТМУ АНАЛІЗУ ВЕБ-КОНТЕНТУ



БЛОК-СХЕМА АЛГОРИТМУ РОБОТИ WHOIS-АГЕНТА