МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ ТА РАДІОЕЛЕКТРОННИХ СИСТЕМ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Віктор ГНАТЮК
“_____”_____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

Тема: «Метод автоматизованого тестування на проникнення телекомунікаційної мережі» .

Виконавець: _____ Алла ПІНЧУК
(підпис)

Керівник: _____ Роман ОДАРЧЕНКО
(підпис)

Консультанти з окремих розділів пояснювальної записки:

Консультант розділу «Охорона праці» _____ Катерина КАЖАН
(підпис)

Консультант розділу «Охорона навколишнього середовища»
_____ Лариса ЧЕРНЯК
(підпис)

Нормоконтролер: _____ Богдан ЧУМАЧЕНКО
(підпис)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра телекомунікаційних та радіоелектронних систем
Спеціальність 172 «Електронні комунікації та радіотехніка»
Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Віктор ГНАТЮК
“_____”_____2025 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Пінчук Алли Дмитрівни
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи: «Метод автоматизованого тестування на проникнення телекомунікаційної мережі»
затверджена наказом ректора від «02» вересня 2025 р. № 1672 /ст
2. Термін виконання роботи: з 29.09.2025 р. по 31.12.2025 р.
3. Вихідні дані до роботи:. архітектура та конфігурація тестового середовища телекомунікаційної мережі 5G
4. Зміст пояснювальної записки:. аналіз підходів до автоматизованого тестування на проникнення телекомунікаційних мереж і вимог до безпечної автоматизації; розробка методу; проєктування і реалізація ПЗ Montera та розробка методології тестування на тестовій мережі 5G
5. Перелік обов'язкового графічного (ілюстративного) матеріалу:. схеми, рисунки, таблиці

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів кваліфікаційної роботи	29.09.2025-30.09.2025	Виконано
2	Вступ	01.10.2025-03.10.2025	Виконано
3	Аналіз існуючих методів автоматизованого тестування на проникнення телекомунікаційних мереж	04.10.2025-10.10.2025	Виконано
4	Розробка та формалізація методу автоматизованого тестування на проникнення	11.10.2025-20.10.2025	Виконано
5	Дослідження алгоритмів машинного навчання для використання у методі	21.10.2025-02.11.2025	Виконано
6	Програмна реалізація запропонованого методу автоматизованого тестування на проникнення	03.11.2025-16.11.2025	Виконано

7	Охорона праці	17.11.2025-30.11.2025	Виконано
8	Охорона навколишнього середовища	01.12.2025-14.12.2025	Виконано
9	Усунення недоліків та захист кваліфікаційної роботи	15.12.2025-31.12.2025	Виконано

7. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона праці	к.т.н., доц. Катерина КАЖАН		
Охорона навколишнього середовища	д.т.н., доц. Лариса ЧЕРНЯК		

8. Дата видачі завдання: «01» вересня 2025 р.

Керівник кваліфікаційної роботи _____ Роман ОДАРЧЕНКО
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Алла ПІНЧУК
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Кваліфікаційна робота «Метод автоматизованого тестування на проникнення телекомунікаційної мережі» містить 202 сторінки, 32 рисунки, 9 таблиць, 101 використане джерело.

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, ТЕЛЕКОМУНІКАЦІЙНА МЕРЕЖА, КІБЕРБЕЗПЕКА, МЕРЕЖЕВА БЕЗПЕКА, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, LLM, REINFORCEMENT LEARNING, MARL

Об’єкт дослідження – процес забезпечення кібербезпеки телекомунікаційних мереж шляхом виконання автоматизованого тестування на проникнення.

Предмет дослідження – методи, моделі та процедурні підходи до автоматизації тестування на проникнення телекомунікаційної мережі.

Мета кваліфікаційної роботи – розробити новий метод автоматизованого тестування на проникнення телекомунікаційних мереж, також реалізувати програмний засіб на основі розробленого методу.

Метод дослідження – системний аналіз і узагальнення нормативно-методичних джерел; моделювання загроз і поверхні атаки; проєктування та реалізація програмного прототипу.

Матеріали кваліфікаційної роботи рекомендується використовувати при плануванні та стандартизації робіт з тестування на проникнення телекомунікаційної інфраструктури, підготовці регламентів/програм аудиту безпеки, а також у навчальному процесі за напрямами кібербезпеки та мережевих технологій.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

5G (Fifth Generation) – стільникова мережа 5-го покоління.

A2C/A3C (Advantage Actor-Critic/Asynchronous A2C) – алгоритми RL actor–critic.

AI (Artificial Intelligence) – штучний інтелект.

API (Application Programming Interface) – програмний інтерфейс застосунків.

ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) – таксономія MITRE технік атак.

CIA (Confidentiality, Integrity, Availability) – тріада властивостей ІБ.

CMDB (Configuration Management Database) – база даних керування конфігураціями.

COMA (Counterfactual Multi-Agent) – алгоритм кооперативного багатоагентного RL.

CTDE (Centralized Training, Decentralized Execution) – принцип MARL.

CVE (Common Vulnerabilities and Exposures) – реєстр публічно відомих вразливостей.

CVSS (Common Vulnerability Scoring System) – система оцінювання критичності вразливостей.

DDPG (Deep Deterministic Policy Gradient) – алгоритм RL для неперервних дій.

DQN (Deep Q-Network) – алгоритм RL на основі Q-навчання з нейромережею.

DRQN Deep Recurrent Q-Network) – рекурентне розширення DQN.

DTDE (Decentralized Training, Decentralized Execution) – режим MARL.

ETL (Extract, Transform, Load) – вилучення, перетворення та завантаження даних.

FastAPI – Python-фреймворк для розроблення REST API.

FiGHT (5G Hierarchy of Threats) – таксономія, онтологія загроз для мереж 5G.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

JSON (JavaScript Object Notation) – формат обміну структурованими даними.

LLM (Large Language Model) – велика мовна модель.

MAPPO (Multi-Agent Proximal Policy Optimization) – багатоагентний PPO.

MARL (Multi-Agent Reinforcement Learning) – багатоагентне навчання з підкріпленням.

MAL (Meta Attack Language) – мова моделювання атак (зокрема для securiCAD).

MDP (Markov Decision Process) – марковський процес прийняття рішень.

MITRE – організація/платформа таксономій.

ML (Machine Learning) – машинне навчання.

MulVAL (Multihost, Multistage Vulnerability Analysis) – фреймворк аналізу графів атак.

NetSPA (Network Security Planning Architecture) – підхід планування атак у мережі.

NFV (Network Functions Virtualization) – віртуалізація мережевих функцій.

NIST (National Institute of Standards and Technology) – інститут стандартів США.

NVD (National Vulnerability Database) – база даних вразливостей NIST.

OSINT (Open Source Intelligence) – розвідка з відкритих джерел.

OSSTMM (Open Source Security Testing Methodology Manual) – методологія тестування безпеки.

OWASP (Open Worldwide Application Security Project) – проєкт/спільнота з веббезпеки.

POMDP (Partially Observable Markov Decision Process) – частково спостережуваний MDP.

PPO (Proximal Policy Optimization) – алгоритм RL класу policy-gradient.

PTES (Penetration Testing Execution Standard) – стандарт виконання пентесту.

QoS (Quality of Service) – якість обслуговування.

RAN (Radio Access Network) – мережа радіодоступу.

REST (Representational State Transfer) – архітектурний стиль побудови веб-API.

RL (Reinforcement Learning) – навчання з підкріпленням.

SAC (Soft Actor-Critic) – алгоритм RL класу actor–critic.

SLA (Service Level Agreement) – угода про рівень сервісу.

NIST SP 800-115 – керівництво з технічного тестування безпеки.

TD (Temporal Difference) – методи RL на основі часової різниці.

TD3 (Twin Delayed DDPG) – удосконалення DDPG.

TRPO (Trust Region Policy Optimization) – алгоритм RL класу policy-gradient.

UE (User Equipment) – абонентське обладнання.

UI (User Interface) – інтерфейс користувача.

VDN (Value Decomposition Networks) – підхід MARL до декомпозиції цінності.

БД – база даних.

ІКТ – інформаційно-комунікаційні технології.

ОКІ – об’єкти критичної інфраструктури.

СУБД – система управління базами даних.

ВСТУП

Актуальність теми. Телекомунікаційні мережі характеризуються високою складністю архітектури, критичністю сервісів, наявністю доменно-специфічних протоколів та нетривіальних векторів атак, а також можливістю формування довгих ланцюжків компрометації, що мають прямі наслідки для безперервності та якості надання послуг. У цьому контексті тестування на проникнення виконує не лише функцію пошуку окремих вразливостей, а й слугує інструментом перевірки стійкості захисту до комбінованих сценаріїв атак і виявлення слабких місць у сигналізації, маршрутизації та сервісному контурі.

Разом з тим, аналіз сучасних підходів показує, що rule-based та model-based методи не забезпечують достатньої адаптивності, масштабованості й здатності працювати з неповними даними у складних телекомунікаційних середовищах (зокрема мережах із віртуалізованими функціями та багатодоменними взаємодіями). Відповідно, доцільним стає перехід до інтелектуальних гібридних підходів, які поєднують графи атак із багатоагентним навчанням з підкріпленням та використанням LLM для планування й інтерпретації результатів.

Зв’язок роботи з науковими програмами, планами, темами. Державний науково-дослідний проєкт молодих вчених МОН України «Методи побудови захищених багатошарових стільникових мереж 5G/6G на основі використання алгоритмів штучного інтелекту для моніторингу об'єктів критичної інфраструктури країни» (№ 0124U000197).

Мета і завдання дослідження. Розробити та обґрунтувати новий метод автоматизованого тестування на проникнення телекомунікаційних мереж, реалізувати прототип у вигляді програмного забезпечення.

Для досягнення поставленої мети вирішуються такі наукові завдання.

1. Проаналізувати існуючі методи автоматизованого тестування на проникнення телекомунікаційних мереж.

2. Розробити та виконати математичну формалізацію нового методу автоматизованого тестування на проникнення.
3. Дослідити існуючі алгоритми Reinforcement Learning на предмет використання у розробленому методі.
4. Розробити програмний прототип методу у вигляді веб-додатку.

Об'єктом дослідження – процес забезпечення кібербезпеки телекомунікаційних мереж шляхом виконання автоматизованого тестування на проникнення.

Предметом дослідження – методи, моделі та процедурні підходи до автоматизації тестування на проникнення телекомунікаційної мережі.

Методи досліджень. Системний аналіз і узагальнення нормативно-методичних джерел; моделювання загроз і поверхні атаки; проєктування та реалізація програмного прототипу.

Наукова новизна та практичне значення отриманих результатів.

Наукова новизна отриманих результатів:

1. Запропоновано гібридний підхід до автоматизованого тестування на проникнення телекомунікаційних середовищ, у якому стратегія тестових дій синтезується поєднанням графових моделей атак/знань із багатоагентним навчанням з підкріпленням та LLM-оркестрацією, що забезпечує адаптацію плану тестування до контексту стенду та результатів попередніх кроків.
2. Удосконалено модель представлення знань про цільове середовище шляхом інтеграції графа атак із динамічним графом знань (стан активів, зв'язків, виявлених фактів і артефактів виконання), що дозволяє формально описувати переходи між станами та підтримувати накопичення доказової бази під час тестування.
3. Сформульовано постановку задачі планування тестових дій як кооперативну багатоагентну задачу (MARL) у дискретному просторі «абстрактних дій» з відображенням на конкретні інструменти, що зменшує залежність методології від конкретного набору утиліт і підвищує переносимість підходу.

4. Запропоновано систему критеріїв та метрик якості інтеграційного тестування (коректність інтеграції/відтворюваність/покриття) для формального контролю працездатності пайплайна та оцінювання повноти досягнутих станів у графі атак.

Практичне значення отриманих результатів:

1. Спроєктовано та описано прототип програмного рішення Montera з багаторівневою, модульною архітектурою, придатною до розширення переліку інструментів і джерел даних, а також до відтворюваного розгортання у типових інфраструктурах.
2. Реалізовано інтеграційні механізми збереження артефактів запусків і єдиного «джерела істини» на рівні даних (граф знань, вразливості, журнали, метрики), що створює основу для подальшої аналітики, звітності та керованого удосконалення сценаріїв тестування.

Апробація отриманих результатів. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- 2025 IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Gliwice, Poland, 2025.
- Fourth International Conference on Cyber Hygiene Conflict Management in Global Information Networks, June 20–22, 2025, Kyiv, Ukraine.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ НА ПРОНИКНЕННЯ ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖ

1.1. Поняття та роль тестування на проникнення

Стрімкий розвиток інформаційно-телекомунікаційних технологій, зростаюча складність мережевої інфраструктури та поява окремого поняття «кіберпростір» спричинили нові виклики у сфері кібербезпеки та її забезпеченні.

Інформаційно-телекомунікаційні системи та мережі стали невід'ємною, критичною частиною функціонування бізнесу, державних органів, об'єктів критичної інфраструктури (ОКІ) тощо, адже вони забезпечують швидкий обмін даними, голосовий та відео- зв'язок, а також інтеграцію різноманітних цифрових сервісів та додатків. Саме тому з'явилася необхідність у забезпеченні їх безпеки.

Ключовими принципами забезпечення кібербезпеки є конфіденційність, цілісність та доступність (CIA) (рис. 1.1). Конфіденційність (confidentiality) гарантує збереження дозволених обмежень щодо доступу до інформації та її розкриття, включаючи засоби захисту особистої конфіденційності та інформації, що є власністю. Цілісність (integrity) – це захист від несанкціонованої модифікації або знищення інформації та забезпечення її незаперечності та автентичності. Доступність (availability), у свою чергу, має на меті забезпечення своєчасного та надійного доступу до інформації та її використання [1].

Рис. 1.1. Тріада CIA [2]

Порушення будь-якого з принципів тріади CIA може призвести до серйозних наслідків, тому, сьогодні існує ряд підходів та стратегій до забезпечення кібербезпеки, одним з яких є тестування на проникнення.

Тестування на проникнення[1] (пентест) – це підклас етичного хакінгу, включає в себе набір методів та процедур для імітації реальних атак для виявлення вразливостей, їх експлуатації та оцінки кіберризиків цільової системи чи мережі організації. Відповідно до стандарту NIST SP 800-115, пентест визначається як тестування безпеки, під час якого спеціалісти імітують реальні атаки з метою виявлення способів обходу засобів захисту програми, системи або мережі. Тестування на проникнення часто передбачає здійснення реальних атак на реальні системи та дані з використанням тих самих інструментів і методів, що й справжні зловмисники. Більшість пентестів передбачають пошук комбінацій вразливостей в одній або декількох системах, які можна використати для отримання більшого доступу, ніж це можливо за допомогою однієї вразливості [3].

Самі ж вразливості можуть існувати в операційних системах, сервісах і додатках, неправильних конфігураціях, неправильних налаштуваннях політик безпеки або ж в небезпечній поведінці кінцевих користувачів. Таким чином, пентести дозволяють виявити прогалини в безпеці, які вразливості присутні, IT-інфраструктури організації та перевірити чи зможе потенційний зловмисник отримати несанкціонований доступ до активів [4-5].

Важливим є підкреслити ключові відмінності тестування на проникнення від аудиту безпеки та оцінки вразливостей (vulnerability assessment). Аудит безпеки, в більшості своїй, є аналітичними процесом, спрямованими на систематичну та комплексну оцінку інфраструктури, політик та практик безпеки організації з метою виявлення вразливостей, оцінки відповідності нормативно-правовим вимогам та надання рекомендацій щодо поліпшення. Оцінка вразливостей – це процес систематичного виявлення, класифікації та визначення пріоритетності слабких місць безпеки (вразливостей) в системах, мережах та додатках організації. На відміну від

[1] Penetration testing (pentest, PT) – тестування на проникнення.

аудиту безпеки та оцінки вразливостей, пентест передбачає саме практичну перевірку та експлуатацію знайдених вразливостей, шляхом моделювання реальних дій зловмисника, включаючи отримання доступу до ресурсів, підвищення привілеїв тощо.

1.1.1. Цілі та завдання тестування на проникнення

Відповідно до міжнародних стандартів ISO/IEC та NIST, метою тестування на проникнення є оцінка вразливостей (ідентифікація слабких місць у IT-інфраструктурі, конфігураціях та додатках), зменшення ризиків (пріоритезація виправлення виявлених вразливостей відповідно до рівня ризику), дотримання вимог та регуляторів (пентест відповідає вимогам державних та міжнародних регуляторів, забезпечуючи високий юридичний статус), а також постійне вдосконалення (це є проактивним підходом до постійного вдосконалення наявних заходів безпеки та до ефективного реагування на швидкоплинний ландшафт кіберзагроз).

Тож, цілі тестування на проникнення можна сформулювати наступним чином:

- виявлення та ідентифікація вразливостей;
- моделювання реальних атак для оцінки ризиків;
- оцінка рівня захисту та ефективності заходів безпеки;
- надання практичної інформації для покращення безпеки.

Завдання тестування на проникнення можна розглянути через призму його процесу. Базово, пентест включає в себе наступні 6 етапів (рис. 1.2) [6-7].

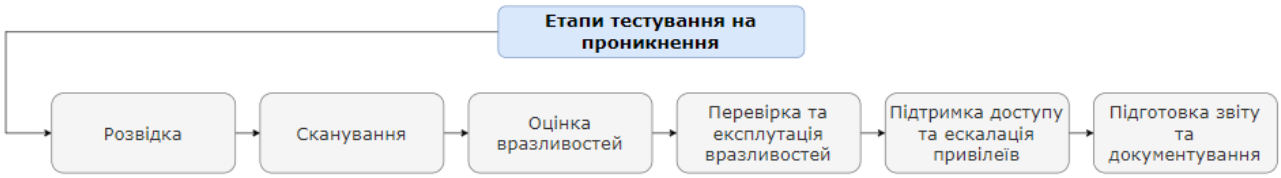

Рис. 1.2. Етапи тестування на проникнення

Розвідка (Reconnaissance). На цьому етапі пентестер збирає якомога більше інформації про цільову систему, включаючи інформацію про IP-адреси, DNS (Domain Name System), мережеві служби, поштові сервери, топологію мережі, операційні

системи та програми, облікові записи користувачів та іншу відповідно інформацію. Мета полягає в тому, щоб зібрати якомога більше даних, що стане основою всього подальшого процесу, щоб тестер міг спланувати ефективну стратегію атаки.

Сканування (Scanning). Під час етапу сканування проводиться ретельна технічна перевірка цільової системи. Для отримання інформації про цільові системи використовуються такі інструменти, як сканери портів, програмне забезпечення для сканування вразливостей та мережеві мапери. Сканування дозволяє тестувальникам оцінити поведінку цільової програми в різних сценаріях та виявити потенційні вразливості, які можна використати.

Оцінка вразливостей (vulnerability assessment). На цьому етапі пентестер використовує всі дані, зібрані на етапах розвідки та сканування, для виявлення потенційних вразливостей та визначення того, чи можна їх використати. Під час визначення ризику виявлених вразливостей на цьому етапі, фахівці з тестування на проникнення мають багато ресурсів, до яких можна звернутися. Одним із них є Національна база даних вразливостей (NVD), сховище даних про управління вразливостями, створене та підтримуване урядом США, яке аналізує вразливості програмного забезпечення, опубліковані в базі даних Загальних вразливостей та експозицій (CVE). NVD оцінює серйозність відомих вразливостей за допомогою Загальної системи оцінювання вразливостей (CVSS).

Перевірка та експлуатація вразливостей (Exploitation and vulnerability verification). На цьому етапі фахівці з тестування на проникнення намагаються отримати доступ до цільової системи та використати виявлені вразливості, розташовані в межах тесту на проникнення. Цей етап також дозволяє пентестеру та організації перевірити серйозність виявлених вразливостей у системах.

Підтримка доступу та ескалація привілеїв (Maintaining access and privilege escalation). Як тільки фахівці з пентесту отримують доступ до цільового середовища, вони мають на меті зберегти доступ та підвищити привілеї, намагаючись переміщатися по системі та отримувати доступ до конфіденційних даних і систем.

Підготовка звіту та документування (Reporting and documentation). На фінальному етапі тестування фахівці збирають свої спостереження, висновки та

кроки щодо усунення недоліків у звіт для цільової організації. Звіт містить відповідні рекомендації для організації щодо покращення її заходів безпеки.

Таким чином, до завдань тестування на проникнення можна віднести:

- створення моделі досліджуваного середовища;
- виявлення потенційних векторів атак;
- перевірка можливості практичної експлуатації виявлених вразливостей;
- оцінка наслідків компрометації цільового середовища;
- формування практичних рекомендацій та заходів з поліпшення кібербезпеки цільового середовища.

1.1.2. Значення тестування на проникнення для телекомунікаційних мереж

На відміну від класичних IT-середовищ, телекомунікаційні мережі мають підвищену критичність, велику масштабованість та розгалуженість, а також використовують спеціалізовані протоколи та сервіси. Через свою специфіку наслідки кібератак можуть бути особливо масштабними: масові збої зв'язку, витік персональних або білінгових даних, порушення роботи критичних служб, економічні та соціально-політичні наслідки.

Особливе значення тестування на проникнення телекомунікаційних мереж може бути обумовлене такими факторами як специфічні вектори атак і протоколи, довжина і складність ланцюгів довіри, висока критичність сервісів та широта атак через IoT/IIoT і масові пристрої.

Таким чином, тестування на проникнення виконує наступні важливі функції в контексті телекомунікаційних мереж:

- виявлення та пріоритезація вразливостей у специфічних елементах мережі – пентест дозволяє знайти не лише помилки конфігурації або вразливості ОС, а й слабкі місця на рівні сигналізації, маршрутизації та сервісів (наприклад, неправильно захищені інтерфейси управління, неналежна аутентифікація протоколів);
- оцінка ефективності засобів захисту в умовах реальних атак – перевірка роботи IDS/IPS, систем аналізу трафіку, фільтрації на межах мережі та

механізмів реагування на інциденти під навантаженням і в атаках, близьких до реальних;

- моделювання складних, комбінованих сценаріїв компрометації – враховуючи можливість послідовного використання кількох вразливостей для ескалації привілеїв і пересування мережею, пентести дозволяють оцінити кінцеву ризикованість таких ланцюжків та їх наслідки для бізнес-процесів і сервісів;
- підтримка вимог регуляторів і стандартів – для телеком-операторів важлива відповідність нормам і стандартам (включаючи вимоги до захисту ОКІ). Результати пентестів служать доказовою базою виконання регуляторних вимог, аудиту та політик безпеки;
- забезпечення безперервності вдосконалення заходів безпеки – регулярні або безперервні (continuous) пентести дозволяють відслідковувати появу нових ризиків, ефективність виправлень і вплив змін в інфраструктурі.

1.2. Класифікація та різновиди тестування на проникнення

Тестування на проникнення охоплює широкий спектр підходів та сценаріїв, адже сам процес може відрізнятися рівнем складності та доступності. Для того аби правильно визначити завдання пентесту і підібрати методи його проведення, важливо розуміти класифікацію таких тестувань. Вона базується на середовищі, у якому відбувається тестування безпеки, обсязі наданої інформації, а також напрямках доступу. Саме завдяки цим критеріям можна імітувати найбільш реалістичні сценарії атак, оцінити захищеність як зовнішніх, так і внутрішніх компонентів та вибудувати ефективну стратегію захисту.

1.2.1. Класифікація тестування на проникнення

Класифікація тестування на проникнення обумовлюється на різних аспектах тестування: цільове середовище, обсяг та вектори атак. Надаючи комплексне

уявлення про стан захищеності цільової системи, кожен аспект націлений на окремі вразливості.

Тестування на проникнення можна класифікувати за двома групами: за напрямком доступу та за середовищем. У свою чергу, вони поділяються на зовнішнє і внутрішнє тестування та локальне і у вебсередовищі відповідно (рис. 1.3) [8].

Рис. 1.3. Класифікація тестування на проникнення

Для того аби імітувати різні вектори атак, пентест може проводитися як ззовні, так і зсередини. Зовнішній пентест має на меті оцінку безпеки соціальних об’єктів та інфраструктури бізнесу (сервери, вебдодатки, вебсайти з доступом до Інтернету. Ключовою ідеєю є оцінка захисту з точки зору зовнішнього зловмисника. Внутрішній пентест, у свою чергу, оцінює захист внутрішніх систем і мереж організації, імітуючи зловмисника, який вже отримав доступ до мережі.

Залежно від середовища та об’єктів, пентест також може бути локальним або веб. Пентест вебсередовища зосереджується на оцінці захищеності вебсайтів, вебдодатків і вебсервісів, які доступні через мережу Інтернет. Тобто тут тестуються вразливості, які можуть бути використані через вебінтерфейс. На відміну від даного типу пентесту, локальне тестування включає перевірку об’єктів і додатків, які інстальовані та працюють на локальному пристрої користувача або в локальній мережі. Тут відбувається оцінка безпеки десктопних додатків, операційних систем і будь-якого іншого ПЗ (програмного забезпечення) або ж служб, що працюють на об’єктах, які перевіряються.

1.2.2. Види тестування на проникнення

Окрім цього є три різновиди (підходи) тестування на проникнення, які відрізняються залежно від того, що організація хоче перевірити і як вона хоче, щоб була перевірена парадигма безпеки: White Box (біла скринька), Gray Box (сіра скринька), Black Box (чорна скринька) (рис. 1.4).

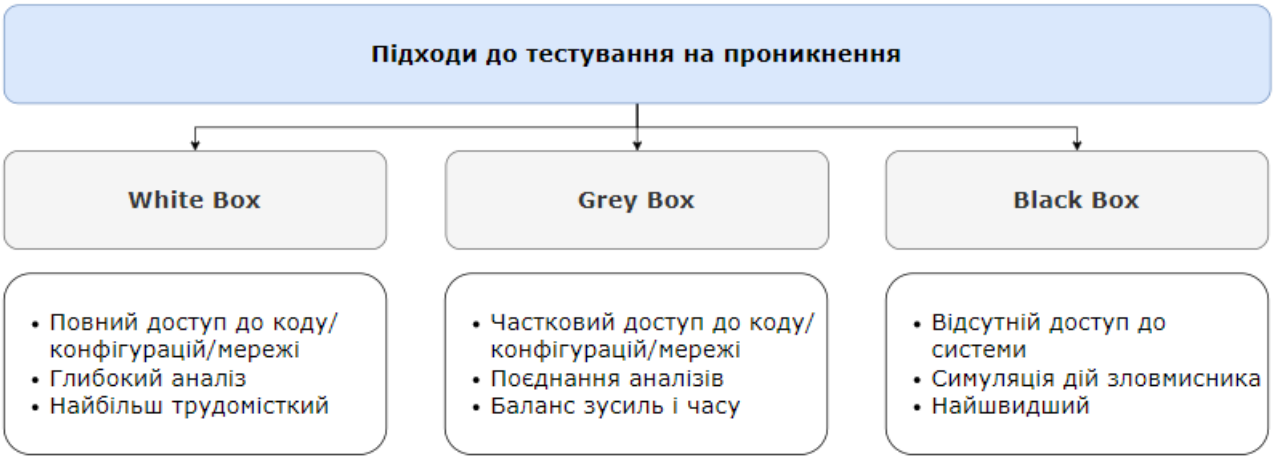

Рис. 1.4. Підходи до тестування на проникнення

White Box (біла скринька). Даний вид тестування на проникнення передбачає надання повної інформації про цільове середовище, включаючи повний доступ до вихідного коду, додатків та сервісів, відповідні версії, всіх конфігурацій та мережі. Тестування «білої скриньки» дуже поширене у внутрішніх/локальних пентестах. Метою цього виду пентесту є виявлення вразливостей, які не обов'язково були б очевидними без глибоких знань системи, та пропонування рекомендацій щодо посилення безпеки. Цей метод також особливо ефективний для перевірки якості коду. Крім того, він економить пентестерам величезну кількість часу та забезпечує більш вичерпне уявлення про стан безпеки [9].

Black Box (чорна скринька). Цей вид тестування не передбачає надання інформації про цільове середовище, або ж вона дуже обмежена. Тобто, така інформація як DMZ (Demilitarized Zone) середовища, операційні системи, версії серверів не надаватиметься, єдине, що може бути отриманим – діапазони IP-адрес, які будуть тестуватися. Будь-які вихідні коди, конфігурації тощо, також не надаються.

Метою буде визначення вразливої області, а потім використання технічних, логічних та/або людських вразливостей, які можуть поставити під загрозу конфіденційність, цілісність та доступність даних і систем. Це доволі поширений сценарій пентесту, який за своєю суттю, є зовнішнім та надає чітке уявлення як цільову систему бачить зловмисник ззовні, тому є досить прагматичним підходом і вважається найбільш реалістичним [10].

Gray Box (сіра скринька). Це проміжний варіант тестування на проникнення, адже тут частина інформації про цільову систему надається, а частина приховується. У випадку тесту на проникнення в мережу організація надає назви додатків, що працюють за IP-адресою, але не розкриває точну версію служб, що працюють. У певному сенсі, тест «сірої скриньки» – це поєднання сильних сторін методів тесту «чорної скриньки» та тесту «білої скриньки». Ця специфічна техніка робить тестування «сірого ящика» першим вибором у контрольованих середовищах, таких як військові та розвідувальні агентства. Тестування «сірого ящика» активно перевіряє як мережеву, так і фізичну безпеку, що робить його ідеальним для виявлення порушень периметра пристроїв, таких як фаєрволи. Ця техніка поєднує такі методи, як сканування мережі, сканування вразливостей, соціальна інженерія та ручна перевірка вихідного коду, щоб оцінити всі можливі наслідки дій хакерів або зловмисників [11].

За своєю суттю ці три види пентесту моделюють різних суб’єктів загроз. У випадку з Black Box – це зовнішній зловмисник, який не має жодного попереднього доступу або інформації про систему. White Box моделює навпаки внутрішнього (інсайдера) або добре проінформованого актора (наприклад, зловмисного працівника або аудитора), який має повний доступ до коду, конфігурацій і мережевої структури. В той час як Gray Box – це проміжний сценарій, що може моделювати поведінку компрометованого користувача або партнера, що дає компроміс між реалістичністю зовнішньої атаки і глибиною аналізу.

1.3. Аналіз методологій тестування на проникнення

Тестування на проникнення передбачає чіткий методологічний процес. Методології являють собою стандартизовані підходи та процедури проведення тестів на проникнення. Вони визначають етапи тестування, методи виявлення вразливостей, способи їх використання та рекомендації щодо зменшення ризиків та відповідають на наступні питання: «Що потрібно досягти?», «Що саме робити?», «В якій послідовності?», «Як робити?», «Чим робити?» (рис. 1.5).

Рис. 1.5. Сутність методологій в процесі тестування на проникнення

Таким чином, методології визначають цілі, окреслюють методи, що використовуються для оцінки безпеки, та визначають інструменти, необхідні для проведення ретельного та ефективного тестування на проникнення [12]. Вони виконуються, щоб забезпечити структуровану та достовірну перевірку. Це дає глибоке розуміння ландшафту безпеки для зміцнення позиції безпеки завдяки процесу пентесту.

Загалом, використання методологій дозволяє:

- дотримуватися покрокового, упорядкованого процесу;

- не пропускати ключові області;
- використовувати ті самі техніки, що і під час навчання;
- надавати чіткі звіти з вимірюваними результатами;
- відповідати галузевим стандартам і вимогам відповідності.

Оскільки сфера тестування на проникнення зовсім не є новою, вже існує багато добре перевірених методологій, таких як OSSTMM (Open Source Security Testing Methodology Manual) [14], ISSAF (Information Systems Security Assessment Framework) [15], PTES (Penetration Testing Execution Standard) [16], NIST SP800-115 (National Institute of Standards and Technology Special Publication 800-115) [17], OWASP (Open Web Application Security Project) [18] тощо (рис. 1.6), які добре зарекомендували себе при використанні для різних типів мереж і систем різного масштабу та призначення. Але телекомунікаційні мережі, зокрема системи стільникового зв'язку, мають свою специфіку, особливо з огляду на широке розмаїття технологій, що в них використовуються.

Рис. 1.6. Ключові методології тестування на проникнення

Сучасні телекомунікаційні системи та мережі, особливо у випадку п’ятого покоління стільникових мереж зв’язку, значною мірою орієнтовані на розподілені обчислювальні ресурси та віртуалізовані мережеві функції (NFV, CNF, SDN). Саме тому під час тестування на проникнення необхідно перевіряти не тільки традиційні

мережеві інтерфейси, а й вразливості гіпервізорів, контейнерів, систем оркестрації та міжмережевої ізоляції.

Окрім цього, телеком-мережі використовують спеціалізовані сигнальні та транспортні протоколи, такі як SS7, GTP, SIP та інші, які мають унікальні вектори атак (перехоплення повідомлень, підміна сигналів, розкриття локації (location disclosure)). Тому, такі методи як фаззинг та ін'єкції повідомлень, модулі для аналізу сигнального трафіку повинні бути включені до набору тестів.

Оскільки, для мереж ОКІ критичними є параметри якості обслуговування (затримка, джиттер та пропускна здатність), які можуть прямо впливати на безпеку і функціональність суміжних критичних застосунків (медицина, автономні транспортні системи, промислова автоматизація), методологія має передбачати експерименти з оцінки впливу атак на ці метрики, а на лише перевірку на дотримання CIA.

Масове підключення IoT та edge девайсів також створює додаткові вектори атак: пристрої можуть слугувати точками початкового доступу до мережі. Тестування має охоплювати захищеність firmware, процедур оновлення, механізмів аутентифікації й сегментацію мережі для запобігання lateral movement від пристроїв до ядра мережі.

Міжоператорська взаємодія та канали роумінгу збільшують ризик атак, що походять не з власної інфраструктури, а через партнерські мережі. Відповідні інтерфейси й сигнальні шлюзи повинні проходити окремі тестування на предмет некоректної фільтрації, підробки або витоку даних [13].

Використання beamforming і перехід на більш високі частити відкрили фізичні/радіочастотні вектори атак (спуфінг, перехоплення на рівні повітряного інтерфейсу, тощо). Саме тому, методологія має включати в тому числі й RF-тести.

Таким чином, тестування на проникнення телекомунікаційних мереж має поєднувати класичні елементи IT-пентесту, протокольно-специфічні процедури, оцінку віртуалізованих і хмарних компонентів, а також тести орієнтовані на продуктивність мережі. З огляду на це, здійснено мультикритеріальний аналіз

існуючих методологій тестування на проникнення щодо можливості їх використання для пентесту телекомунікаційних мереж (табл.1.1).

Таблиця 1.1

Порівняльний аналіз методологій тестування на проникнення

Методологія	**Короткий опис**	**Переваги**	**Обмеження**
OSSTMM (Open Source Security Testing Methodology Manual)	Відкрита методологія для тестування безпеки інформаційних систем, мереж, процесів та фізичних аспектів. Орієнтована на вимірюваність та об’єктивність результатів	Забезпечує ретельний багатовекторний аналіз (фізичний, мережевий, людські фактори). Корисний для оцінки великомасштабних телекомунікаційних інфраструктур.	Не стосується безпосередньо технологій, характерних для телекомунікацій (наприклад, NFV/SDN, slicing, супутникові канали зв'язку). Може вимагати значних ресурсів.
ISSAF (Information Systems Security Assessment Framework)	Розгорнутий фреймворк оцінки безпеки, що фокусується на широкому спектрі технічних і процесних контролів у IT-інфраструктурах.	Цінний для оцінки ризиків на рівні організації та процесів у телекомунікаційних операторів. Підтримує відображення відповідності вимогам.	Орієнтований на традиційні IT-системи; обмежена застосовність до динамічних розподілених телекомунікаційних середовищ.

Закінчення таблиці 1.1

PTES (Penetration Testing Execution Standard)	Стандартизована модель виконання тестування на проникнення (етапи, робочі процеси).	Чіткий, адаптивний процес координації масштабних телекомунікаційних випробувань. Підходить для повторюваних оцінок.	Надає обмежені рекомендації щодо галузевих питань (послуги, для яких критично важлива затримка, або гібридні телеком-архітектури).
NIST SP800-115	Технічний довідник із тестування інформаційної безпеки	Сильний акцент на структурованому управлінні ризиками та безпечному виконанні тестів, що є важливим для ОКІ.	Загальна IT-орієнтація; відсутність сценаріїв, специфічних для телекомунікацій
OWASP (Open Web Application Security Project)	Зосереджений на безпеці додатків, API та веб-безпеці.	Корисний для телекомунікаційних веб-порталів, платформ самообслуговування, OSS/BSS та API. Багатий набір інструментів та підтримка спільноти.	Вузька сфера застосування: не стосується основної телекомунікаційної інфраструктури, віртуалізованих мереж або аспектів, чутливих до продуктивності.

Аналіз методологій тестування на проникнення показав їх сильні сторони та обмеження при застосуванні до телекомунікаційних мереж. Кожна методологія має свою специфічну спрямованість, причому OSSTMM та PTES забезпечують комплексні та структуровані рамки, які можна адаптувати до складності

телекомунікаційних інфраструктур, включаючи розподілені та віртуалізовані середовища.

ISSAF пропонує цінні відомості для оцінки ризиків на організаційному та процесному рівнях, що є важливим для розуміння більш широкого операційного контексту безпеки телекомунікацій, тоді як NIST SP800-115 фокусується на структурованому управлінні ризиками та безпечному виконанні тестів, що є критично важливим аспектом, враховуючи значення безперебійності та доступності телекомунікаційних послуг.

OWASP є особливо корисним при оцінці веб-додатків, API та порталів для клієнтів, які є невід'ємною частиною надання телекомунікаційних послуг, але самостійно він не є достатнім для усунення унікальних вразливостей, що виникають у великих мережевих середовищах з високою доступністю та декількома операторами.

У цьому контексті жодна з методологій (OSSTMM, ISSAF, PTES, NIST SP800-115, OWASP) не задовольняє повністю всі потреби, характерні для телекомунікаційної галузі. Тому оптимальним є комбінований підхід: використання PTES/NIST для структури процесів і безпеки, OSSTMM для операційних показників і багатовекторного покриття, OWASP для веб-компонентів/API, а також доповнення їх модулями, специфічними для телекомунікацій, що стосуються NFV, периферійних обчислень, IoT та сценаріїв взаємозв'язку.

1.4. Аналіз існуючих методів автоматизованого тестування та визначення їх недоліків

Одним з недоліків ручного тестування на проникнення є необхідність у фахових навичках для його проведення, а також час для виконання, що обумовлює його дорогу вартість через ресурсозатратність. Сьогоднішня постійно зростаюча складність інформаційно-телекомунікаційних систем та мереж, складність кібератак та поява нових векторів проникнення в мережі досить ускладнюють процес ручного тестування на проникнення. Саме тому, стало актуальним завдання створення методів

автоматизованого тестування на проникнення, зокрема автоматизації всіх процесів за допомогою алгоритмів машинного навчання (ML). Алгоритми ML мають здатність ефективної адаптації до змін цільового середовища і більш точно визначати вразливості, нівелюючи людський фактор.

Ефективність застосування штучного інтелекту (ШІ) до тестування на проникнення можливо чітко побачити, порівнявши з традиційним підходом, взявши за критерії етапи пентесту (табл. 1.2).

Таблиця 1.2

Ефективність застосування ШІ

	Етап пентесту	**Традиційний підхід**	**Використання алгоритмів AI/ML**
1	Розвідка (Reconnaissance)	Довготривалий ручний збір інформації з відкритих джерел (OSINT), аналіз публічних записів WHOIS, Shodan, тощо.	Автоматизований збір і збагачення OSINT, класифікація релевантних даних, виявлення аномалій у великих масивах інформації; зниження часу на підготовку профайлу цілі.
2	Сканування (Scanning)	Запуск сканерів портів/сервісів, ручна інтерпретація результатів, багато шуму і хибнопозитивних результатів.	Інтелектуальна фільтрація результатів сканування, адаптивне налаштування сканерів, зменшення хибних спрацювань та пріоритизація релевантних знахідок.
3	Оцінка вразливостей (Vulnerability assessment)	Використання баз даних вразливостей і сигнатурних сканерів; ручна кореляція з	Автоматизована кореляція сигнатур з контекстом, прогнозування ризику експлуатації, автоматичне

Закінчення таблиці 1.2

		контекстом (версії, конфігурації).	групування схожих вразливостей і пріоритизація за їх впливом.
4	Перевірка та експлуатація вразливостей (Exploitation & Verification)	Ручна розробка експлойтів або використання готових з відкритих баз даних (зокрема, exploitdb), багато ручної роботи і ризиків для стабільності системи.	Автоматизоване створення та тестування експлойтів в контрольованому середовищі, симуляція атак з використанням навчання на попередніх кейсах; швидка верифікація при меншому ризику помилок.
5	Підтримка доступу та ескалація привілеїв (Maintaining access & Privilege escalation)	Ручні сценарії пост-експлуатації, створення бекдорів/скриптів, довгий час для дослідження можливих ланцюжків.	Автоматизація пошуку можливостей експлуатації, моделювання стійкості бекдорів, оптимізація послідовності дій для збереження доступу з мінімальним шумом, виявлення нетипових слідів.
6	Підготовка звіту та документування (Reporting & Documentation)	Ручне написання звітів, збір доказів, формування рекомендацій; довготривалий процес.	Генерація чернеток звітів на основі знайдених доказів, автоматична класифікація ризиків і рекомендацій, адаптація стилістики мови під технічну/не технічну аудиторію, швидше отримання фінальної версії звіту.

Таким чином, інтеграція AI/ML демонструє значний потенціал для можливості підвищення точності та ефективності процесу тестування на проникнення, що зменшить залежність від впливу людського фактору та прискорить загальний час всього процесу, дозволяючи суттєву економію часу. Окрім цього, автоматизоване тестування на проникнення буде більш бюджетним варіантом для організацій, оскільки буде не таким ресурсозатратним.

1.4.1. Види тестування на проникнення

Всі сучасні методи автоматизованого тестування на проникнення можна класифікувати за рівнем автоматизації та за підходом до планування атак.

Класифікація за рівнем автоматизації показує наскільки процес пентесту залежить від втручання людини та наскільки система здатна самостійно виконувати завдання, від простих автоматизованих скриптів до повністю автономних систем. Таким чином можна виділити три методи: rule-based, model-based та intelligent.

Rule-based automated penetration testing. Це є найстарішим підходом, де всі процеси тестування виконуються за попередньо визначеними правилами, наборами скриптів або політиками без можливості навчання.

Механізм роботи полягає у трьох етапах. Спочатку відбувається попередня конфігурація правил, де фахівці з пентесту визначають сигнатури атак, сценарії експлуатації та політики сканування. Далі йде автоматичне сканування, система виконує визначені дії, наприклад, сканування портів, пошук відомих вразливостей (за CVE, CWE) або ж запуск готових експлойтів. У кінці відбувається ручний аналіз результатів, тобто є необхідність участі експерта для інтерпретації звіту. Загалом, це всі доступні автоматизовані інструменти, такі як Nmap, OpenVAS, Nessus, Metasploit Framework.

Model-based automated penetration testing. У цьому підході створюється модель середовищ, де симулюється поведінка зловмисник, що дозволяє прогнозувати наслідки атак без втручання у реальну мережу.

Основними компонентами даного підходу є модель мережі (топологія хостів, підмереж, сервіси, конфігурації, відомі вразливості), модель зловмисника (доступні

інструменти та експлойти, рівень навичок, обмеження ресурсів) та модель невизначеностей (динамічні зміни у мережі, ймовірність успішної експлуатації, фактори неповного спостереження).

При такому підході зазвичай використовуються такі методи моделювання атак як:

- дерева атак (attack trees) – ієрархічне представлення цілей і шляхів досягнення;
- графи атак (attack graphs) – зв'язки між вузлами, сервісами та вразливостями.
- фреймворки для автоматичної побудови моделей безпеки (наприклад, MulVAL).

Intelligent automated penetration testing. Тут, відповідно, використовуються алгоритми штучного інтелекту та машинного навчання для прийняття рішень і самостійного планування атак.

До основних напрямків можна віднести:

- Reinforcement Learning (RL) – агент вчиться шляхом взаємодії з середовищем, отримуючи винагороду за успішні дії. MDP (Markov Decision Process) використовується у контрольованих середовищах, де стан мережі відомий, а POMDP (Partially Observable MDP) використовується у реальних умовах, коли дані неповні або застарілі.
- Large Language Models (LLM) – GPT-подібні моделі прогнозують можливі дії зловмисників, аналізують текстові звіти про вразливості, формують ймовірні сценарії атак та допомагають пояснювати результати тестування.
- Hybrid AI (RL+LLM) – поєднання RL для планування дій у динамічному середовищі з LLM для аналізу контексту та знань.

Інша класифікація – за підходом до планування атак. Ця класифікація описує алгоритмічні основи побудови шляхів атак (penetration paths). Тут можна виділити наступні підходи:

- *Rule-based:* виконання заздалегідь визначених статичних сценаріїв (скриптів);
- *Graph-based:* моделювання мережі у вигляді графа;

- *Search-based:* алгоритми пошуку оптимальних шляхів;
- *RL-based:* агент навчається шляхом взаємодії;
- *LLM-based:* використання LLM для прогнозу дій;
- *Hybrid AI:* поєднання RL і LLM.

Узагальнена класифікація існуючих автоматизованих методів тестування на проникнення наведена на рисунку 1.7.

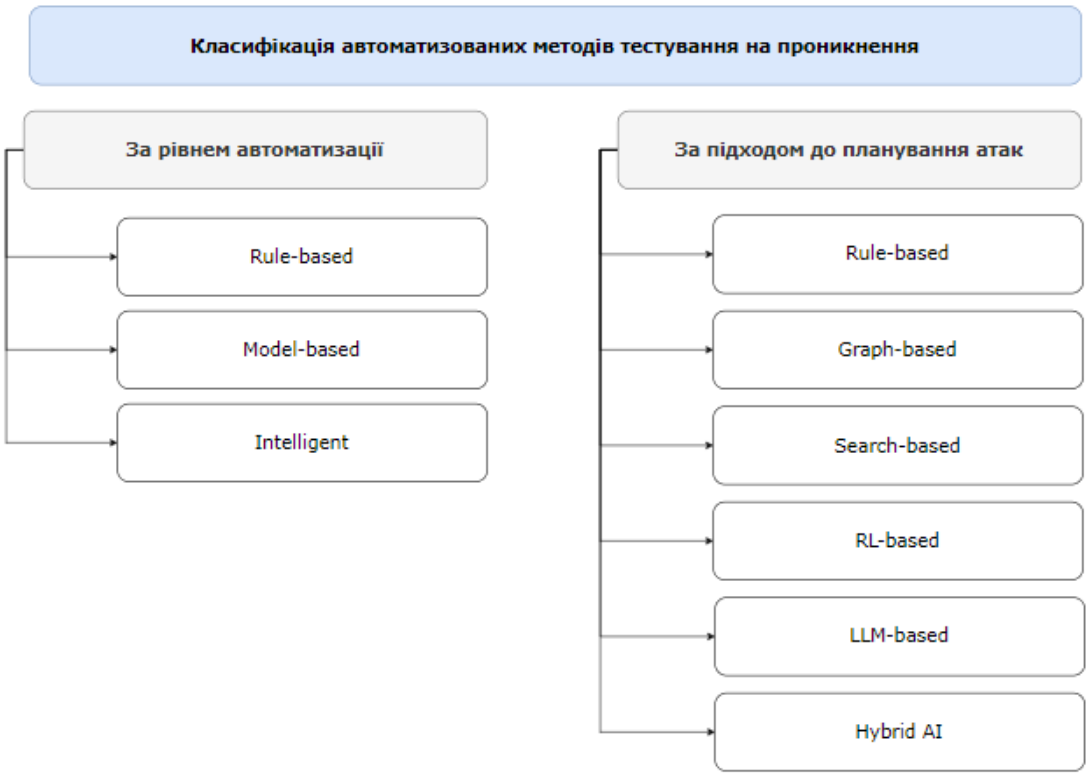

Рис. 1.7. Класифікація методів автоматизованого тестування на проникнення

1.4.2. Порівняльний аналіз автоматизованих методів

Рівень автоматизації є одним з ключових аспектів, що визначає ефективність процесу тестування на проникнення, зокрема у великих, складних середовищах, таких як сучасні телекомунікаційні мережі.

Відповідно до аналізу сучасних досліджень [19-24], було виокремлено три основні підходи: rule-based automated penetration testing, model-based automated penetration testing, intelligent automated penetration testing. Ці підходи відображають еволюцію автоматизованих скриптів до повністю автономних платформ.

Rule-based методи є найбільш поширеними завдяки своїй простоті та швидкості впровадження, базуючись на заздалегідь створених сигнатурах атак та вразливостей, скриптах для автоматичного виконання тестів, статичних політиках безпеки та ручним налаштуванням. До їх переваг можна віднести:

- швидка інтеграція в існуючу інфраструктуру;
- мінімальні вимоги до обчислювальних ресурсів;
- ефективність для рутинних перевірок та комплаєнс-аудитів [19].

Проте, є ряд вагомих недоліків таких методів. У першу чергу, необхідність наявності експерта, який буде вручну перевіряти та формувати звіти. Окрім цього, відсутня будь-яка адаптивність. Усі сценарії статичні та потребують регулярного ручного оновлення. У швидкозмінних телекомунікаційних мережах це призводить до швидкого старіння бази сигнатур. Іншим недоліком є неможливість виявлення нових атак. Такі методи працюють лише з відомими вразливостями і не здатні виявляти zero-day атаки. Вагомим недоліком є високий рівень хибнопозитивних (false-positive) результатів. Також до недоліків слід віднести погану масштабованість. При збільшенні кількості вузлів у мережі час сканування та обсяг даних зростатимуть експоненційно [19].

Таким чином, rule-based метод не підходить для складних та динамічних середовищ, таких як телекомунікаційні мережі. Його статичність і залежність від людського фактору є головним обмеженням.

Model-based метод, за своєю суттю, є вдосконаленням попереднього, пропонуючи математичне моделювання мережі та сценаріїв атаки. Сама ж модель досить часто знаходила своє застосування у сучасних дослідженнях, зокрема у [20] вона продемонструвала ефективність побудови дерева атак за допомогою відкритого фреймворку MulVAL. У даному дослідженні використання алгоритму DQN (Deep Q-Network) дозволило скоротити кількість зайвих кроків та підвищити точність визначення оптимального шляху атаки до 86%. Іншим підходом використання даного методу стало інтегрування теорії ігор, врахувавши взаємодію між атакувальником і захисником, що підвищило реалістичність моделювання [21].

До переваг даного методу можна віднести:

- глибший аналіз у порівнянні з rule-based;
- можливість моделювати складні сценарії без реальної шкоди для інфраструктури;
- підвищена точність завдяки симуляціям.

Однак, до недоліків слід віднести високу ресурсомісткість. Адже створення та підтримка моделей потребує значних обчислювальних ресурсів і часу. Також присутня залежність від точності даних. Неточні або застарілі дані про топологію мережі призводять до помилок у прогнозуванні атак. Вагомим недоліком є складність масштабування. У великих операторських мережах кількість вузлів та зв'язків робить побудову графів атак надзвичайно складною та обчислювально затратною.

Незважаючи на ряд переваг даного методу, для великих телекомунікаційних мереж він залишається непрактичним через високу складність моделювання та масштабування.

Інтелектуальні методи автоматизації сьогодні є найбільш сучасним напрямком. Вони поєднують штучний інтелект (AI) та машинне навчання (ML) для автономного прийняття рішень і адаптивного планування атак [22-23]. До основних напрямків відносять Reinforcement Learning (RL), Large Language Models (LLM) [24], Hybrid AI (RL+LLM). У дослідженні [23] даний метод показав вищу точність і швидкість у порівнянні зі стандартним DQN, ефективно працюючи у black-box сценаріях без повної інформації про мережу.

До переваг даного методу слід віднести:

- повна автономність і здатність працювати у реальному часі;
- можливість виявляти zero-day вразливості;
- адаптивність до динамічних змін у мережі;
- робота з неповними даними завдяки POMDP-моделям.

У той же час, присутні наступні недоліки. В першу чергу, висока обчислювальна складність. Для навчання та роботи моделей потрібні великі обчислювальні ресурси. Також складність налаштування та пояснення результатів. Аналітикам важко зрозуміти логіку прийняття рішень RL-агентів [24]. Окрім цього,

залежність від великих обсягів даних. Для якісного навчання необхідні значні набори даних про атаки на мережі.

Інтелектуальні системи мають найбільший потенціал для тестування на проникнення сучасних телекомунікаційних мереж завдяки своїй автономності та здатності до навчання. Проте, їх практичне впровадження обмежене через високі вимоги до ресурсів та складність інтеграції.

Зведений порівняльний аналіз наведено у таблиці 1.3.

Таблиця 1.3

Порівняльний аналіз методів за рівнем автоматизації

Критерій	**Rule-based**	**Model-based**	**Intelligent**
Автономність	Низька	Середня	Висока
Адаптивність	Відсутня	Обмежена	Повна
Робота з неповними даними	Немає	Часткова	Так
Виявлення zero-day	Неможливе	Обмежене	Можливе
Ресурсомісткість	Низька	Середня	Висока
Масштабованість	Погана	Середня	Висока

Одним з ключових складових автоматизованого тестування на проникнення є стратегія планування атак (attack planning), яке визначає послідовність дій, необхідних для досягнення цілі атакувальника, наприклад, отримання доступу до критично важливих вузлів телекомунікаційної мережі або компрометації конфіденційних даних.

Ефективність планування атак безпосередньо впливає на результативність пентесту, зокрема на такі показники як швидкість, точність виявлення вразливостей та здатність системи працювати у динамічному середовищі.

На основі аналізу сучасних досліджень та наукових праць [21-24] можливо виділити шість основних підходів: rule-based, graph-based, search-based, RL-based, LLM-based та Hybrid AI.

Rule-based підхід аналогічний до підходу з попередньої класифікації. Тому серед недоліків відсутність адаптивності, неможливість роботи з невідомими вразливостями та zero-day атаками, хибнопозитивні результати та погана масштабованість.

Graph-based методи використовують графи атак для моделювання можливих шляхів компрометації. У цій моделі, вузли графа – це ресурси мережі або стани системи, а ребра – дії чи експлойти, що змінюють стан системи. Серед прикладів доступних інструментів є такі як MulVAL, Attack Tree Tools та інші платформи для візуалізації атак.

До переваг такого методу слід віднести:

- наглядне представлення шляхів проникнення;
- можливість визначати найкритичніші вузли для захисту;
- виконання симуляцій без впливу на реальну мережу.

До недоліків можна віднести значне зростання складності. У великих мережах кількість вузлів і ребер швидко зростає, що робить моделювання обчислювально затратним.

Search-based методи використовують алгоритми пошуку для визначення оптимального маршруту атаки: DFS/BFS (базовий перебір варіантів), A* (евристичний пошук із врахуванням вартості дій), GraphPlan (для складних систем з великою кількістю станів) [20].

До переваг такого методу слід віднести можливість знаходити оптимальний шлях атаки та формалізований процес, який добре описується алгоритмічно.

Серед недоліків такого методу є високі обчислювальні ресурси. Алгоритми стають неефективними при зростанні розміру мережі. Є потреба у повній інформації, адже без повних даних про мережу результати пошуку будуть некоректними. А також відсутність механізму навчання. Алгоритм не покращує роботу з досвідом.

При використанні RL-based методу, агент навчається шляхом взаємодії з середовищем та отриманням винагороди за успішні дії. Тут використовуються MDP (для повністю спостережуваних середовищ) та POMDP (для середовищ з неповною інформацією) [22].

Серед переваг слід відзначити адаптивність і здатність працювати у динамічних середовищах, можливість знаходити zero-day атаки та роботу з неповними даними.

До недоліків можна віднести високі обчислювальні ресурси, проблеми масштабування та складність пояснення результатів.

LLM-based підхід в більшості своїй використовується для аналізу звітів і баз вразливостей, прогнозування дій атакувальників та формування ймовірних сценаріїв атак [24].

Серед переваг такого методу є:

- робота з великими масивами текстових даних;
- можливість пояснення результатів тестування;
- генерація гіпотез щодо атак на основі контексту.

У той же час є ряд недоліків таких, як відсутність безпосереднього впливу на середовище (LLM лише прогнозує дії, а не виконує їх), залежність від навчальних даних (якість роботи напряму залежить від наявності актуальної інформації) та можливість некоректних гіпотез (моделі можуть створювати неправдиві сценарії).

Гібридні системи поєднуються RL для автономного виконання атак та LLM для аналітики й формування контексту. Такий підхід був використаний у дослідженні [21], де RL визначав оптимальні шляхи проникнення, а LLM аналізував дані та пояснював результати.

До переваг такого методу слід віднести вищу точність завдяки поєднанню двох підходів, адаптивність і здатність працювати у реальному часі та зниження хибнопозитивних результатів на понад 60% [21].

Серед недоліків слід відзначити високу складність впровадження (потрібні значні ресурси та знання для налаштування) та залежність від якості навчальних даних для RL та LLM.

Порівняння всіх шести методів здійснено за такими критеріями, як автономність, адаптивність, робота з неповними даними, виявлення zero-day вразливостей, ресурсомісткість, масштабованість, час проведення тестування, якість пояснення результатів та точність (табл. 1.4).

Таблиця 1.4

Порівняльний аналіз методів за підходами до планування атак

Критерій	**Rule-based**	**Graph-based**	**Search-based**	**RL-based**	**LLM-based**	**Hybrid AI**
Автономність	1	3	3	5	3	5
Адаптивність	0	2	2	5	3	5
Робота з неповними даними	0	3	0	5	5	5
Виявлення Zero-day	0	2	2	5	5	5
Ресурсомісткість	1	3	5	5	3	5
Масштабованість	1	3	1	3	3	5
Точність	3	5	5	5	3	5
Час проведення тестування	5	3	1	3	3	5
Якість пояснення результатів	5	5	3	1	5	5

У таблиці, у контексті підтримки (задоволення) критерію:

- повна/висока/можливе/так – «5»;
- середня/часткова – «3»;
- обмежена – «2»;
- низька/погана – «1»;
- відсутня/немає/неможливе – «0».

Порівняльний аналіз існуючих методів автоматизованого тестування на проникнення показав, що найбільш перспективним методом є інтелектуальний гібридний підхід RL+LLM, де RL-агент виконує планування та оптимізацію маршрутів атаки, а LLM використовується для аналізу, пояснення результатів та роботи з текстовими даними.

Однак, попри всі значні переваги, цей метод не є досконалим. RL-агент навчається на основі винагороди та попереднього досвіду, але не гарантує пошуку оптимального маршруту атаки, особливо у складних топологіях мережі з великою кількістю вузлів та станів. Це може призвести до неповного бо неефективного покриття вразливостей. Тобто, один RL-агент не здатний повністю охопити складу та розподілену телекомунікаційну мережу.

Щоб зменшити ці ризики та підвищити ефективність, можна інтегрувати такі підходи як Swarm Intelligence (SI), або ж Multi-Agent Reinforcement Learning (MARL) у загальну архітектуру методу. Це дозволить побудувати стійку та масштабовану систему для інтелектуального автоматизованого пентесту.

Окрім цього, телекомунікаційні мережі вимагають більш комплексного підходу. Новий метод має поєднати сильні сторони з кожного розглянутого, rule-based, model-based та intelligence.

Для підвищення ефективності всього методу, можливим є додавання динамічного графу знань, який буде охоплювати вузли мережі, сервіси, всі взаємозв’язки та вразливості. В такому випадку, LLM зможе опрацьовувати граф для формування сценаріїв, зокрема і нестандартних підходів, атак, а, наприклад, RL-агент/SI/MARL вже плануватимуть необхідні дії.

Такі підходи забезпечать баланс між автоматизацією, ефективністю, точністю та аналітичними можливостями, зменшуючи ризики, притаманні кожному окремому методу.

ВИСНОВКИ ДО РОЗДІЛУ 1

У розділі обґрунтовано, що телекомунікаційні мережі є критичною складовою функціонування бізнесу та об’єктів критичної інфраструктури, а тому вимоги до забезпечення конфіденційності, цілісності та доступності (CIA) набувають визначального значення для їх кіберстійкості.

Уточнено сутність тестування на проникнення як підходу, спрямованого на імітацію реальних атак для виявлення вразливостей, перевірки можливості їх практичної експлуатації, оцінювання наслідків компрометації та формування рекомендацій із підвищення захищеності цільового середовища.

Показано, що специфіка телекомунікаційних інфраструктур (зокрема 5G) зумовлює необхідність виходу за межі «класичного» IT-пентесту та врахування віртуалізованих мережевих функцій і хмарних компонентів (NFV/CNF/SDN), міжоператорських взаємодій (роумінг), а також фізичних/радіочастотних векторів атак і вимог до продуктивності мережі.

На підставі мультикритеріального аналізу методологій (OSSTMM, ISSAF, PTES, NIST SP 800-115, OWASP) встановлено, що жодна з них не покриває повною мірою всі потреби, характерні для телекомунікаційної галузі; обґрунтовано доцільність комбінованого застосування (PTES/NIST – для процесної структури й безпечного виконання, OSSTMM – для вимірюваності та багатовекторного покриття, OWASP – для веб/API-компонентів) із доповненням доменно-специфічними модулями.

Визначено ключові обмеження ручного пентесту (висока залежність від кваліфікації виконавця, значні часові витрати та ресурсозатратність), що в умовах зростаючої складності телекомунікаційних систем актуалізує перехід до автоматизованих підходів із залученням AI/ML для адаптивності та зниження впливу людського фактору.

Систематизовано автоматизовані методи тестування на проникнення та підходи до планування атак (rule-based, graph-based, search-based, RL-based, LLM-based, Hybrid AI). Порівняльний аналіз показав перспективність гібридного підходу

RL+LLM (оптимізація маршрутів атак + пояснюваність та робота з текстовими артефактами), водночас підкреслено, що один RL-агент не гарантує оптимальності й повного покриття у складних топологіях; обґрунтовано доцільність переходу до кооперативних схем (MARL/ін. колективні підходи) та введення динамічного графа знань як основи для масштабованої та контекстної інтелектуалізації процесу пентесту телекомунікаційних мереж.

Отримані в Розділі 1 результати формують вимоги та напрям подальшого дослідження: побудову доменно-орієнтованої моделі (графа атак/графа знань) і розроблення інтелектуального методу автоматизованого тестування на проникнення, здатного працювати в умовах динамічних і розподілених телекомунікаційних середовищ.

РОЗДІЛ 2
РОЗРОБКА ТА ФОРМАЛІЗАЦІЯ МЕТОДУ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

2.1. Запропонований метод

Аналіз проблеми створення нового методу автоматизованого тестування на проникнення телекомунікаційних мереж у першому розділі показав, що наявні rule-based та model-based підходи не забезпечують достатнього рівня адаптивності, масштабованості та здатності працювати з неповними даними у середовищі сучасних телекомунікаційних мереж. Особливо це стосується стільникових мереж зв'язку LTE/5G та, так званих, Next-G (NextGen), мереж із віртуалізованими мережевими функціями (NFV), механізмами поділу на зрізи (network slicing) та складною міжмережевою взаємодією. Для таких середовищ доцільним є використання інтелектуальних методів, що поєднують переваги підходів на основі графів атак, багатоагентного навчання з підкріпленням (Multi Agent Reinforcement Learning, MARL) та моделей штучного інтелекту типу LLM (Large Language Model).

З огляду на це, запропоновано новий гібридний метод автоматизованого тестування на проникнення телекомунікаційних мереж, який поєднує:

- підсистему моделювання мережі та загроз (мережева модель, множина вразливостей, граф атак, динамічний граф знань);
- інтелектуальних оркестратор, що складається з LLM-модуля планування й аналізу та Multi-Agent RL-модуля пошуку оптимальних маршрутів атаки;
- оркестратор інструментів тестування на проникнення, який забезпечує взаємодію із класичними та телеком-специфічними пентест-інструментами;
- підсистему політик та обмежень, а також підсистему формування результатів (звіт, метрики, рекомендації).

Структурна схема запропонованого методу наведена на рис. 2.1.

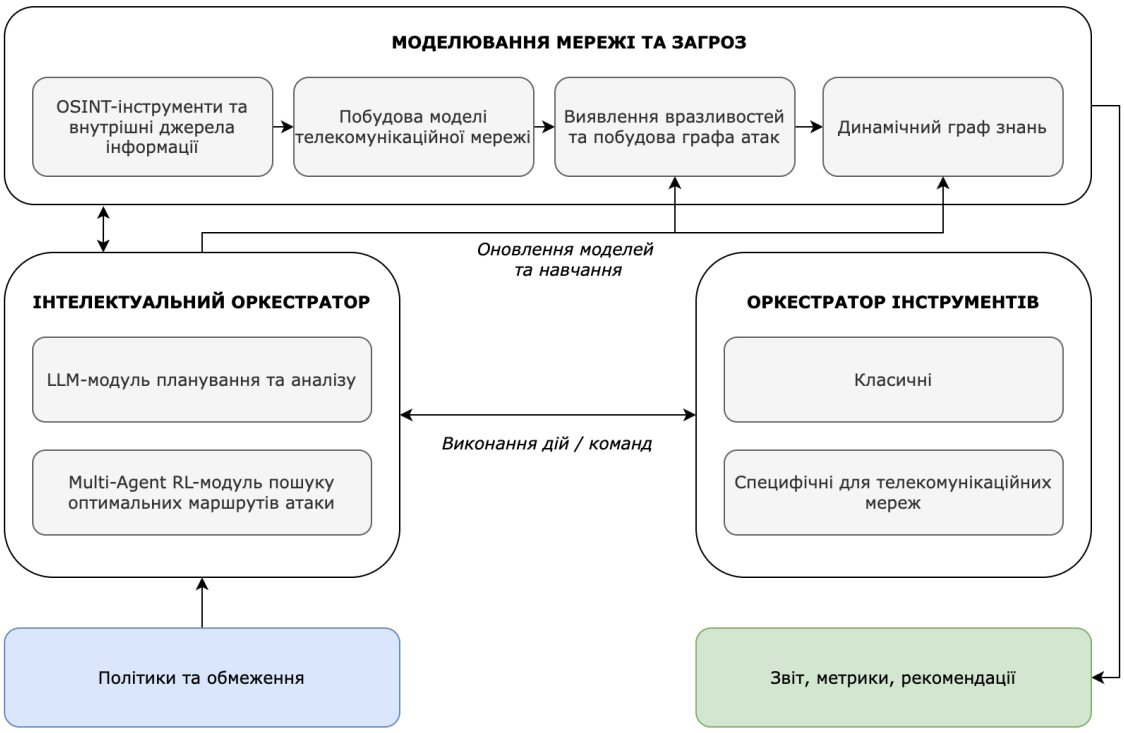

Рис. 2.1. Запропонований метод автоматизованого тестування на проникнення

2.1.1. Структурний опис запропонованого методу

Метод логічно поділений на три рівні. На верхньому рівні зображено підсистему моделювання мережі та загроз, що відповідає за формування формалізованого уявлення про цільову телекомунікаційну мережу та її вразливості. У центральній частині схеми розташований інтелектуальний оркестратор, який на основі побудованих моделей, графа атак та поточного стану графа знань планує та оптимізує сценарії тестування на проникнення. Праворуч показано оркестратор інструментів, що реалізовує фактичне виконання дій у мережі із використанням зовнішніх інструментів для пентесту. На останньому рівні зображені додаткові підсистеми. Підсистема політик та обмежень забезпечує врахування вимог безпеки, регламентів та SLA (Service Level Agreement) під час планування й виконання атак, а підсистему формування результатів агрегує та інтерпретує отримані дані у вигляді звітів, метрик ризику та рекомендацій.

Моделювання мережі та загроз. Першим етапом роботи методу є збір та агрегація даних про досліджувану телекомунікаційну мережу. Для цього використовуються як OSINT (Open Source Intelligence) інструменти та зовнішні

джерела інформації (сканери відкритих портів та сервісів, пошукові системи пристроїв, публічні бази вразливостей), так і внутрішні джерела оператора: системи моніторингу, CMDB (Configuration Management Database), журналі подій, результати попередніх аудитів та пентестів. На основі цих даних формується узагальнене подання про топологію, набір мережевих функцій, використовувані протоколи та сервіси.

Після чого виконується побудова моделі телекомунікаційної мережі, що включає множину вузлів (мережеві елементи, віртуалізовані функції, сервери додатків, абонентські пристрої), множину зв'язків між ними, а також атрибути, які описують конфігурацію, роль і критичність кожного елемента. Ця модель використовується як основа для подальшого аналізу вразливостей та побудови графа атак.

На основі створеної моделі мережі та результатів роботи сканерів виконується виявлення вразливостей та побудова графа атак. Вразливості відображаються у вигляді множини умов, за яких стає можливим перехід зловмисника (суб'єкта загрози) від одного стану мережі до іншого. Відповідно, будується орієнтований граф, вершини якого відповідають станам мережі або компонентам мережі, а дуги – потенційним крокам атаки. Для телекомунікаційних мереж додатково визначаються специфічні сценарії, пов'язані із сигналізаційними протоколами, міжмережевою взаємодією та механізмами network slicing.

Одержана інформація інтегрується у динамічний граф знань, який відображає не лише структуру мережі та вразливості, а й накопичений досвід попередніх тестувань, відомі шаблони атак, контекстні зв'язки між сервісами та політиками безпеки. Він стає центральним джерелом інформації для інтелектуального оркестратора та постійно оновлюється за результатами нових тестувань.

Інтелектуальний оркестратор. Даний оркестратор забезпечує перехід від статичних моделей до адаптивного планування та виконання тестування на проникнення. Загалом, включає два взаємопов'язані модулі.

Першим модулем є LLM-модуль планування та аналізу. На основі даних з графа знань та графа атак, а також заданих політик та обмежень LLM формує високорівневі

цілі тестування та набір кандидатських сценарії атак. Модель аналізує опис мережевих функцій, протоколів, виявлених вразливостей і, використовуючи вбудовані та зовнішні (з використанням механізмів RAG (Retrieval-Augmented Generation)) знання про типові атаки в телеком-середовищах, пропонує послідовності дій для досягнення критичних активів мережі. Крім того, LLM-модуль відповідає за інтерпретацію текстових результатів роботи пентест-інструментів та формулювання зрозумілих висновків для звітності (враховуючи й технічну частину).

Multi-Agent RL-модуль (MARL) пошуку оптимальних маршрутів атаки трактує процес тестування на проникнення як задачу навчання з підкріпленням у складному середовищі з частковою проінформованістю. Кожен агент відповідає за дослідження певної частини графа атак або певного зрізу мережі, а сумарна політика формується на основі взаємодії цих агентів. MARL одержує від LLM-модуля кандидати сценаріїв та обмеження, уточнює їх на основі реального зворотного зв'язку від інструментів, оптимізовує послідовність кроків з урахуванням функції винагороди (ризик, вплив на SLA, вартість кроку тощо) та повертає оркестратору інструментів конкретні дії/команди для виконання.

Обидва модулі працюють у тісному взаємозв'язку з динамічним графом знань. Результати нових експериментів, успішних та невдалих атак, спрацьовування засобів захисту тощо використовуються для подальшого вдосконалення політики RL-агентів і корекції генеративних стратегій LLM-модуля.

Оркестратор інструментів та результати роботи методу. Оркестратор інструментів реалізовує взаємодію із зовнішніми пентест-засобами. Він містить підмножину класичних інструментів та підмножину інструментів, специфічних для телекомунікаційних мереж, зокрема тестові фреймворки для LTE/5G, RAN, інструменти аналізу протоколів, емулятори абонентів, засоби перевірки безпеки мережевих слайсів. Оркестратор отримує від інтелектуального оркестратора конкретні дії та перетворює їх у послідовність команд до відповідних інструментів, а потім повертає результати у вигляді структурованих подій.

Окремим блоком виділено політики та обмеження. Вони задаються оператором або замовником тестування на проникнення й визначають допустимі типи атак, часові

вікна виконання, максимально припустиме навантаження на окремі сегменти мережі, заборонені до тестування вузли тощо. Ці політики враховуються як при генерації сценаріїв LLM-модулем, так і під час пошуку оптимальних маршрутів атаки MARL-модулем.

На основі результатів виконання метод формує звіт, метрики та рекомендації. До звіту включаються виявлені ланцюжки атак із вказанням використаних вразливостей, досягнутих цілей, ймовірних наслідків для бізнес-процесів та пропозиції щодо усунення або мінімізації ризиків. Метрики можуть включати ймовірність успішної атаки на ті чи інші сервіси, оцінку «вартості» атаки, ступінь покриття простору можливих маршрутів тощо. Ці дані, у свою чергу, повертаються до динамічного графа знань і використовуються для подальшого вдосконалення методу.

2.1.2. Математична формалізація запропонованого методу

Для математичної формалізації, запропонований метод автоматизованого тестування на проникнення телекомунікації мережі можна описати за допомогою теорії множин.

Таким чином, весь метод представляється як кортеж його основних структурних та процедурних компонентів:

$$M_{APT-TEL} = \langle D, N, V, G_A, K, П, O_{LLM}, O_{MARL}, O_{tool}, F_{report} \rangle. \quad (2.1)$$

Вхідні дані та модель цільової телекомунікаційної мережі описуються наступним чином як сукупність вхідних даних:

$$D = D_{ext} \cup D_{int}, \quad (2.2)$$

де D_{ext} – дані, отримані за допомогою OSINT-інструментів та публічних джерел інформації, а D_{int} – внутрішні дані оператора (моніторинг, CMDB, журнали подій, результати попередніх тестів/аудитів тощо).

На основі множини даних D формується модель телекомунікаційної мережі:

$$N = \langle H, L, \alpha \rangle, \tag{2.3}$$

де H – множина вузлів (мережеві сегменти, віртуалізовані функції, сервери додатків, абонентські пристрої тощо); $L \subseteq H \times H$ – множина логічних або фізичних зв'язків між вузлами; $\alpha: H \cup L \to A_\alpha$ – відображення, що задає атрибути вузлів та зв'язків (тип елемента, рівень критичності, підтримувані протколи, налаштування безпеки тощо).

Побудову моделі можна розглядати так:

$$f_N: D \to N. \tag{2.4}$$

Множину виявлених вразливостей позначимо через V:

$$V = \{v_1, v_2, \dots, v_n\}, \tag{2.5}$$

де кожна вразливість v_i характеризується щонайменше трьома складовими:

$$v_i = \langle h_i, c_i, \beta_i \rangle, \tag{2.6}$$

де $h_i \in H$ – вузол, до якого відноситься вразливість; c_i – умови її експлуатації (наявність відкритого порту, версія програмного забезпечення, конфігураційні параметри тощо), β_i – наслідок успішної експлуатації (підвищення привілеїв, розкриття інформації, відмова в обслуговуванні тощо).

На основі пари (N, V) будується граф атак, який представляється наступним чином:

$$G_A = \langle S, E_A \rangle, \tag{2.7}$$

де S – множина можливих станів мережі з точки зору зловмисника (суб'єкта загрози); $E_A \subseteq S \times S$ – множина орієнтованих ребер, кожне ребро $(s_i, s_j) \in E_A$ відповідає

можливому кроку атаки, який стає доступним за рахунок експлуатації однієї або декількох вразливостей з V.

Побудова множини вразливостей та графа атак формалізується як:

$$f_{V,G}: (N, D) \to (V, G_A), \tag{2.8}$$

Динамічний граф знань позначимо як:

$$K = \langle V_K, E_K, \psi \rangle, \tag{2.9}$$

де V_K – множина вершин графа знань (вузли мережі, сервіси, вразливості, типові сценарії атак, політики безпеки тощо); $E_K \subseteq V_K \times V_K$ – множина зв'язків між знаннями (причинно-наслідкові, ієрархічні, належності до одного сервісу тощо); $\psi: V_K \cup E_K \to A_\psi$ – набір атрибутів (ваги, ймовірності, часові мітки, джерела надходження інформації).

Граф знань ініціалізується на основі (N, V, G_A) та зовнішніх джерел доменнних знань, а в процесі роботи методу динамічно оновлюється:

$$f_K: (N, V, G_A, D_{ext}) \to K, \Delta K = f_{upd}(K, R_{raw}) \tag{2.10}$$

де R_{raw} – необроблені результати запуску інструментів.

Множину політик та обмежень представимо через П:

$$\text{П} = \{\pi_1, \pi_2, \dots, \pi_q\}, \tag{2.11}$$

де кожен елемент π_j описує допустимі/заборонені типи дій, часові обмеження, обмеження навантаження, винятки для окремих вузлів тощо. Формально політику можна представити у вигляді предиката:

$$\pi_j: (s, a) \to \{0, 1\}, \tag{2.12}$$

який для пари «стан-дія» повертає значення чи дозволено виконувати дію a у стані s.

Множину інструментів тестування на проникнення, що використовуються в методі, позначимо через T:

$$T = T_{cl} \cup T_{tel}, \tag{2.13}$$

де T_{cl} – множина класичних інструментів (сканери портів, сканери вразливостей, фреймворки експлуатації, веб-сканери тощо); T_{tel} – множина інструментів, специфічних для телекомунікаційних мереж (фреймворки для LTE/5G-core, RAN, аналізатори протоколів сигналізації, емулятори абонентів, засоби тестування мережевих зрізів та ін.).

Кожен інструмент $t \in T$ має множину підтримуваних елементарних дій $A_t = \{a_{t,1}, a_{t,2}, \dots\}$ та інтерфейс для отримання результатів.

LLM-модуль формалізується наступним чином:

$$O_{LLM}: (G_A, K, \Pi) \to \Sigma, \tag{2.14}$$

де Σ – множина кандидатних сценаріїв атак.

Кожен сценарій $\sigma \in \Sigma$ подається у вигляді послідовності абстрактних кроків:

$$\sigma = \langle a_1^{(h)}, a_2^{(h)}, \dots, a_n^{(h)} \rangle, \tag{2.15}$$

де $a_i^{(h)}$ – абстрактна високорівневна дія, пов’язана з переходами у графі атак G_A та вузлами графа знань K.

Також LLM-модуль виконує інтерпретацію текстових результатів інструментів, що можна подати як:

$$O_{LLM}^{res}: R_{raw} \to R_{sem}, \tag{2.16}$$

де R_{sem} – семантично інтерпретовані результати (нові вразливості, підтверджені шляхи атаки, текстові пояснення тощо).

Процес тестування на проникнення розглядається як багатоагентна задача навчання з підкріпленням (MARL) у середовищі:

$$\mathcal{E} = \langle S, A, P_{tr}, R, \gamma \rangle, \tag{2.17}$$

де S – множина станів (стани мережі з точки зору доступних атак); A – множина можливих дій; $P_{tr}(s'|s,a)$ – ймовірність переходу зі стану s у стан s' при виконанні дії a; $R: S \times A \to \mathbb{R}$ - функція винагороди (відображає досягнення цілей атаки, вартість дій, штрафи за порушення політик та SLA); $\gamma \in (0,1)$ – коефіцієнт дисконтування.

Нехай $\text{Å} = \{ag_1, \dots, ag_n)$ – множина агентів, кожен з яких має власну політику $\pi_i: S \to A$, що визначає вибір у кожному стані. Сукупний оператор MARL-модуля можна записати як:

$$O_{MARL}: (\mathcal{E}, \Sigma, \Pi) \to C, \tag{2.18}$$

де C – множина конкретизованих послідовностей дій/команд, які відповідають сценаріям Σ та враховують політики Π. Елементи $c \in C$ вже безпосередньо пов’язуються з інструментами T.

Оркестратор інструментів формалізується як:

$$O_{tool}: (C, T) \to R_{raw}, \tag{2.19}$$

який перетворює послідовності дій $c \in C$ у конкретні виклики інструментів із множини T та повертає необроблені результати R_{raw}.

Функція формування звіту задається наступним чином:

$$F_{rep}: (R_{sem}, K) \to R_{out}, \tag{2.20}$$

де R_{out} – кінцевий результат роботи методу (звіт про проведене тестування на проникнення, оцінка ризиків, рекомендації щодо усунення вразливостей та посилення захисту цільової телекомунікаційної мережі).

Узагальнено, роботу методу $M_{APT-TEL}$ можна записати у вигляді такої послідовності:

$$D \xrightarrow{f_N} N \xrightarrow{f_{V,G}} (V, G_A) \xrightarrow{f_K} K \xrightarrow{O_{LLM}} \Sigma \xrightarrow{O_{MARL}} D \xrightarrow{O_{tool}} R_{raw} \xrightarrow{O_{LLM}^{res}} R_{sem} \xrightarrow{F_{rep}} R_{out}, \quad (2.21)$$

при цьому результати R_{raw} та R_{sem} використовуються для оновлення графа знань K та параметрів RL-агентів, що забезпечує адаптивність і неперервне вдосконалення методу.

2.1.3. Алгоритм функціонування запропонованого методу

На основі формальної моделі $M_{APT-TEL}$ опишемо роботу запропонованого методу у вигляді послідовності етапів (рис. 2.2).

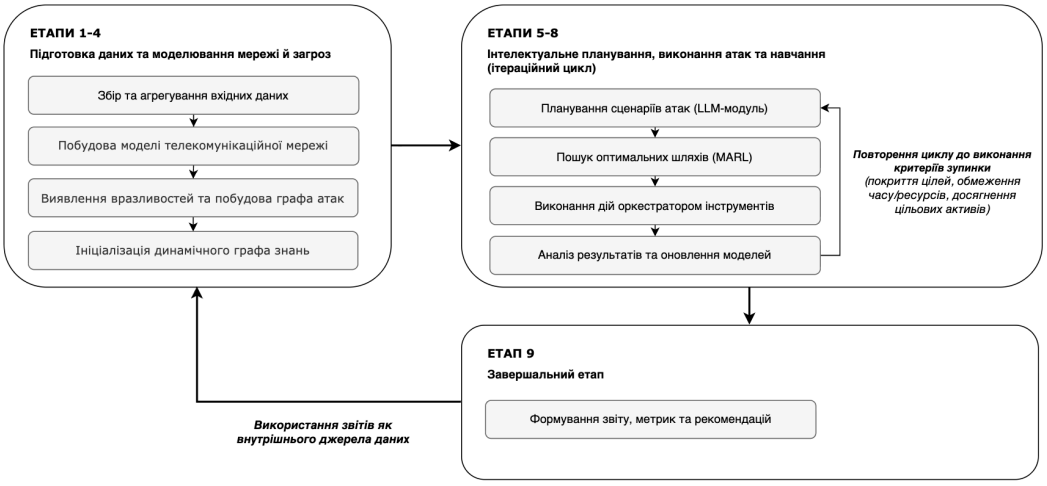

Рис. 2.2. Етапи функціонування запропонованого методу

Етап 1. На першому етапі відбувається збір та агрегування вхідних даних. Спершу, ініціалізуються джерела зовнішніх даних D_{ext} (OSINT-інструменти, публічні реєстри, бази вразливостей) та внутрішніх даних D_{int} (моніторинг, CMDB, журнали подій, попередні звіти.

Після чого виконується збір і попередня фільтрація даних, формується множина $D = D_{ext} \cup D_{int}$.

Етап 2. Другий етап передбачає побудову моделі телекомунікаційної мережі. На основі множини даних D формуються множини вузлів H та зв'язків L. Після чого для кожного елемента задаються атрибути α (тип, роль, критичність, протоколи, налаштування безпеки. У результаті отримуємо модель мережі $N = \langle H, L, \alpha \rangle$.

Етап 3. На даному етапі відбувається виявлення вразливостей та побудова графа атак. На основі моделі N та додаткових даних з D виконується сканування та кореляція з базами CVE/NVD (Common Vulnerabilities and Exposures / National Vulnerabilities Database) тощо. Після чого формується множина вразливостей V із прив'язкою до конкретних вузлів і конфігурацій. На основі пари (N, V) будується граф атак $G_A = \langle S, E_A \rangle$, де вершини – стани мережі, ребра – можливі кроки атаки.

Етап 4. На четвертому етапі – ініціалізація динамічного графа знань. На основі N, V, G_A та доменних знань формується початковий граф знань K. До нього інтегруються відомі шаблони атак, політики безпеки, історичні дані попередніх тестувань.

Етап 5. П'ятий етап передбачає планування сценаріїв атак LLM-модулем. Цей модуль отримує як вхід G_A, K та множину політик й обмежень П. На основі аналізу графа атак, знань та вимог П, LLM формує множину кандидатних сценаріїв $\Sigma = \{\sigma_1, \dots, \sigma_n\}$, де кожний сценарій – послідовність високорівневих кроків досягнення цільових активів. Сценарії фільтруються за релевантністю та пріоритизуються, наприклад, за очікуваним впливом на критичні сервіси.

Етап 6. На даному етапі відбувається пошук оптимальних маршрутів атаки Multi-Agent RL-модулем. Для вибраних сценаріїв Σ визначається RL середовище $\mathcal{E} = \langle S, A, P_{tr}, R, \gamma \rangle$ з урахуванням обмежень П. Після чого ініціалізується множина агентів $Å = \{ag_1, \dots, ag_n)$, кожен з яких відповідає за дослідження певної частини графа атак або певного сегмента мережі. Далі агентам передаються сценарії Σ як початкові гіпотези, вони досліджують простір можливих дій, отримуючи винагороду відповідно до цілей тестування на проникнення та штрафи за порушення політик П.

Після збіжності або досягнення критеріїв зупинки формується множина оптимізованих послідовностей низькорівневих дій/команд $\mathrm{C} = \{c_1, \dots, c_n\}$.

Етап 7. Сьомий етап передбачає виконання дій оркестратором інструментів. Він зіставляє кожну з послідовностей c_i з відповідними інструментами з множини $T = T_{cl} \cup T_{tel}$. Для кожного кроку генеруються конкретні виклики пентест-інструментів (параметри сканування, експлойти, телеком-специфічні тести тощо). Інструменти виконуються у визначених робочих вікнах з урахуванням політик П. Результати збираються у вигляді множини необроблених даних R_{raw}.

Етап 8. Восьмий етап – інтерпретація результатів та оновлення моделей. LLM-модуль аналізує R_{raw}, виділяє підтверджені вразливості, нові стани мережі, успішні та неуспішні ланцюжки атак, формує семантичні результати R_{sem}. На основі семантичних результатів оновлюється граф знань K та, за потреби, граф атак G_A. RL-агенти оновлюють свої політики на основі отриманого зворотного зв'язку (до навчання). Також перевіряються критерії зупинки: досягнення заданого рівня покриття, вичерпання бюджету часу/ресурсів, досягнення цільових активів мережі. У випадку, якщо критерії не виконані, то процес повертається до етапів 5-7 з оновленими моделями K, G_A.

Етап 9. На останньому етапі формуються звіт, метрики та рекомендації. На основі R_{sem} та актуального стану графа знань K функція F_{rep} формує вихідний набір результатів R_{out}: опис знайдених ланцюжків атак, вплив на цільові сервіси та бізнес-процеси, оцінку ризику та пріоритезацію вразливостей, рекомендації щодо усунення та підсилення кіберзахисту. Сам звіт подається у вигляді структурованого документа, а також може використовуватися як вхідні дані для наступних запусків методу (оновлення D_{int}).

2.2. Вибір фреймворку для побудови графа атак

Одним із базових інструментів для аналізу безпеки мережевих систем є графи атак (attack graphs). Під ним розуміють орієнтований граф, вершини якого відображають можливі стани системи або окремі проміжні факти (наприклад,

«суб'єкт загрози має доступ до вузла X із правами користувача»), а дуги – це окремі кроки атаки, які переводять систему з одного стану до іншого за рахунок експлуатації певної вразливості, помилки чи конфігурації. Такий підхід до моделювання атак був запропонований ще в одних із перших робіт з graph-based аналізу вразливостей мережі, а в подальших дослідженнях формалізований і розвинений для автоматизованої генерації та аналізу графів атак у мережах реалістичного масштабу [25-28].

Класично розрізняють два варіанти подання графів атак:

- стан-орієнтовані, де вершинами є глобальні або часткові стани системи;
- факт-орієнтовані (логічні), де вершини відображають окремі факти й умови, а дуги – правила виводу нових фактів [26-29].

У частині робіт мережа моделюється як кінцевий автомат і граф атак генерується засобами model-checking (state-based подання) [26], тоді як у логіко-орієнтованих підходах типу MulVAL використовується модель фактів і правил на основі Datalog, а сам граф атак будується як логічних граф виводу (exploit dependency graph) [27-29].

Візуально граф атак зазвичай подається у вигляді орієнтованого графа, де початкові вершини відповідають поточним можливостям суб'єкта загрози (доступ до певних сегментів мережі, наявні облікові дані), а цільові вершини – небажаним станам системи (компроментація критичних серверів, порушення правил тріади CIA ключових сервісів) [28-31]. Між ними можуть існувати десятки й сотні альтернативних шляхів, що відображають послідовності експлуатації вразливостей, переміщення між вузлами та підвищення привілеїв. Типові приклади таких графів для невеликих мереж наведено в роботах з мінімізації вартості «харденінгу» на основі графів атак та в оглядових публікаціях, присвячених поясненню методів аналізу графів атак [28, 30-31]. Саме така структура, у вигляді простору можливих шляхів атаки за допомогою графа, і є основою для застосування методів оптимізації та навчання з підкріпленням і надалі використовується як один з ключових компонентів у запропонованому методі автоматизованого тестування на проникнення телекомунікаційної мережі. Якість і придатність побудованого графа атак напряму

залежить від вибору фреймворку, який реалізовує генерацію графа на основі даних про топологію мережі, конфігурації та вразливості.

2.2.1. Огляд існуючих фреймворків побудови графу атак

Узагальнені огляди інструментів побудови графів атак [32-34] показують, що з практичного досвіду використовується обмежений набір фреймворків, які відрізняються своїми архітектурою, ступенем відкритості та підтримуваними можливостями.

Загалом, для середовища телекомунікаційних мереж можна висунути наступні вимоги до фреймворку побудови графа атак:

- здатність обробляти мережі операторського масштабу;
- підтримка багатокрокових і мультивузлових атак;
- наявність механізмів автоматизованого завантаження даних з існуючих джерел, таких як сканери вразливостей, CMDB, конфігураційні файли, бази типу CVE та інші;
- можливість розширення доменно-специфічними правилами, необхідних для опису протоколів та мережевих функцій (особливо важливо для LTE/5G, RAN, сигнальних шлюзів, мережевих слайсів тощо);
- відкритість програмного забезпечення, у контексті забезпечення прозорості роботи та можливості модифікації;
- підтримка та/або сумісність з ризик-орієнтованими підходами, зокрема ймовірнісні графи атак, інтеграція з графами знань тощо.

До найбільш використовуваних фреймворків можна віднести MultiVAL, TVA/Cauldron, NetSPA, CyGraph, MAL+securiCAD.

MultiVAL (Multihost, Multistage Vulnerability Analysis). Це один з найбільш відомих логіко-орієнтованих відкритих (open-source) фреймворків для побудови графів атак [27]. Для моделювання фактів про конфігурацію мережі, вразливостей та правил їх взаємодії використовується мова Datalog, після чого відбувається автоматична генерація логічного графу атак, де вершини – факти й проміжні стани, а ребрами є кроки атаки та наслідки їх виконання (рис. 2.3) [27, 33].

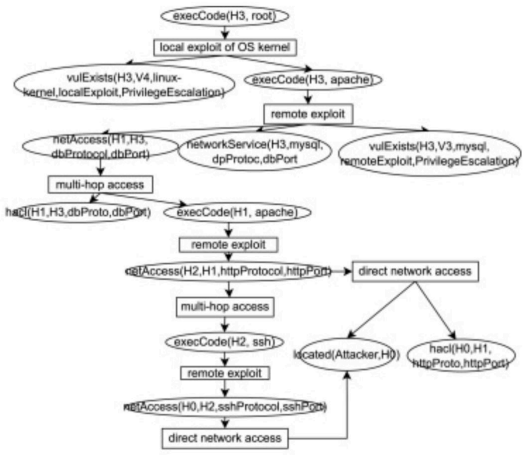

Рис. 2.3. Приклад графу атак MulVAL [35]

У [27] підкреслено, що даний фреймворк спроєктовано з урахування двох ключових вимог: можливості інтегрувати формалізовані специфікації вразливостей з відкритих баз і забезпечити масштабованість з багатохостовими мережами.

MulVAL і надалі широко використовується та активно розширюється. З'являються десятки нових розширень та правил, у тому числі маппінг з MITRE ATT&CK, побудова Баєсівських графів атак і підтримка різних сценаріїв.

Таким чином, до переваг можна віднести відкритість та прозору чітку модель, наявність великої кількості розширень, підтримку багатокрокових та мультивузлових атак, а також можливість формування основи для ризик-орієнтованого аналізу.

Проте, слід відзначити, що даний фреймворк початково орієнтований на класичні IT-мережі, а інтеграція з телеком-специфічними компонентами потребує додавання нових предикатів та фактів, що є його обмеженням.

TVA (Topological Vulnerability Analysis) / Cauldron. Рішення TVA та пов'язаний з ним інструментарій Cauldron являються підходом топологічного аналізу вразливостей. На основі конфігурацій мережі, даних сканерів та баз вразливостей автоматично генерується граф атак і виконується аналіз шляхів до критичних активів та метрик ризику [36-37]. Цей підхід дозволяє обробляти великі розгалужені мережі, інтегрувати різнорідні джерела даних і підтримує багатокрокові сценарії атак.

До переваг TVA/Cauldron можна віднести високий ступінь інтеграції з інфраструктурними даними та сканерами вразливостей, а також масштабованість і орієнтацію на практичний ризик-менеджмент.

Однак, серед недоліків слід відзначити те, що Cauldron є переважно попрієтарним та не доступний як повністю відкритий стек для подальшої модифікації та адаптації.

NetSPA (Network Security Planning Architecture). Це система розроблена MIT Lincoln Laboratory, яка генерує графи атак найгіршого випадку на основі мережевих конфігурацій та результатів сканування вразливостей [32, 38]. Рішення є масштабованим до великої кількості вузлів, підтримує багатокрокові графи атак та використовується для оцінювання ефективності контрзаходів і аналізу типу «what-if». На рисунку 2.4 наведено приклад графу атак NetSPA,

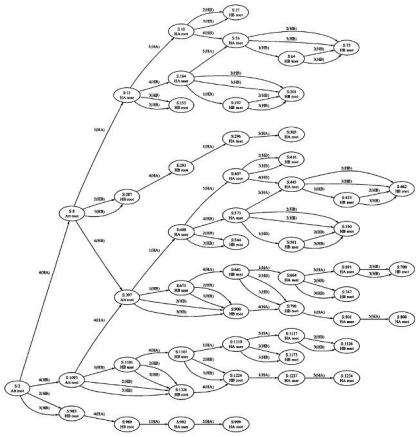

Рис. 2.4. Приклад графу атак згенерованого NetSPA

Але NetSPA не є повністю відкритим програмним забезпеченням, має в собі фіксовану модель вхідних даних і є менш зручним для глибокого розширення доменно-специфічним правилам. Саме тому це обмежує його застосування як базового фреймворку в прототипах з нестандартною архітектурою.

CyGraph. Розробка MITRE, що поєднує графову базу знань (Neo4j) з атаково-графовою аналітикою [39]. У даному рішенні агрегуються дані про мережеву інфраструктуру, вразливості, події безпеки та місійні залежності в єдиний граф знань,

на основі якого формуються динамічно еволюційні графи атак, які використовуються для ситуаційної обізнаності та аналізу ризиків [39-40].

До сильних сторін можна віднести інтеграцію великої кількості джерел, використання графових баз даних та природну підтримку контекстної й місійної аналітики. В той же час, CyGraph є більше архітектурною платформою, ніж легковаговим фреймворком, а його програмні реалізації не позиціонуються як універсальні, повністю відкриті для вбудовуванням у сторонні прототипи.

MAL + securiCAD. Meta Attack Language (MAL) є відносно новим формальним підходом, що пропонує метамову для побудови доменно-специфічних мов моделювання атак, таких як coreLang, enterpriseLang, vehicleLang тощо [42-43]. На основі моделі інфраструктури, заданої у відповідній MAL-мові, автоматично генеруються великі ймовірнісні графи атак, що дозволяють проводити симуляції та оцінювати ризики. SecuriCAD є основним програмним втіленням, даного рішення, який інтегрує моделювання, генерацію графів та ймовірнісний аналіз.

Тож, до переваг можна віднести високу розширюваність за рахунок доменно-специфічних мов, орієнтацію на ймовірнісний аналіз та оцінку ризиків, а також сучасний розвиток.

Суттєвим недоліком є те, що на практиці робота з графами атак часто прив’язана до пропрієтарного securiCAD, а сама платформа задає власну модель життєвого циклу моделювання, що не завжди відповідає задачі інтеграції з LLM/MARL-орієнтованими підходами.

2.2.2. Порівняльний аналіз фреймворків

На основі проведеного огляду існуючих фреймворків для побудови графів атак можна виокремити наступні критерії для порівняння, які є принциповими для можливості використання в запропонованому методі автоматизованого тестування на проникнення телекомунікаційних мереж:

1. Відкритість та доступність реалізації (C1) – необхідна для прозорості алгоритмів, можливості модифікації, інтеграції з власними компонентами і відтворюваності результатів.

2. Масштабованість (С2) – здатність ефективно працювати на великих мережах з великою кількістю вузлів і вразливостей, що критично для операторських мереж.
3. Інтеграція з джерелами даних (С3) – підтримка завантаження результатів сканерів вразливостей, конфігураційних файлів, логів тощо.
4. Розширюваність доменно-специфічними моделями (С4) – можливість опису телеком-специфічних компонентів шляхом додавання нових фактів чи правил.
5. Підтримка багатокрокових і мультивузлових атак (С5) – необхідна для моделювання реалістичних ланцюгів атак у великих розподілених телекомунікаційних мережах.
6. Підтримка ризик-орієнтованої та контекстної аналітики (С6) – можливість використовувати отриманий граф для оцінки ризиків, пріоритезації вразливостей і врахування контексту місій/сервісів.

Узагальнені результати порівняння основних фреймворків за цими критеріями наведено у таблиці 2.1. У якості оцінок відповідності введено наступні значення: висока – 5; середня – 3; низька – 1.

Таблиця 2.1

Порівняльний аналіз фреймворків

Фреймворк	**С1**	**С2**	**С3**	**С4**	**С5**	**С6**
MulVAL	5	5	3	5	5	3
TVA/Cauldron	1	5	5	3	5	5
NetSPA	1	5	5	3	5	3
CyGraph	3	5	5	5	5	5
MAL + securiCAD	3	5	3	5	5	5

Проведений аналіз показав, що жоден з існуючих фреймворків побудови графів атак повною мірою не задовольняє сукупність всіх поставлених вимог, характерних

саме для задач автоматизованого тестування на проникнення телекомунікаційних мереж з використанням інтелектуальних інструментів (LLM, MARL). Логіко-орієнтований MulVAL відзначається своїми відкритістю та гнучкістю, але має недолік у контексті початково призначення для класичних IT-мереж. Такі платформи як CyGraph чи MAL/securiCAD хоч і надають розвинуту підтримку графів знань і ризик-орієнтованого аналізу, але є значною мірою попрієтарними рішеннями або комплексними архітектурами.

Таким чином, доцільним є обрання не тільки базового фреймворку, в даній ситуації найбільш оптимальним рішенням є MulVAL, а й запропонувати концепцію вдосконаленого фреймворку побудови графа атак для телекомунікаційних мереж, який поєднає в собі сильні сторони наявних рішень і використає доступні open-source компоненти там, де це можливо.

2.3. Концепція удосконаленого фреймворку для побудови графа атак

На основі проведеного аналізу можна виділити наступні ключові напрямки вдосконалення:

- доменне розширення логічної моделі MulVAL;
- інтеграція з графом знань (за аналогією до CyGraph);
- удосконалені механізми завантаження даних;
- підтримка ризик-орієнтованого аналізу та ймовірнісних оцінок;
- відкритий інтерфейс взаємодії з інтелектуальними компонентами.

У контексті доменного розширення, на основі механізму фактів і правил Datalog, пропонується розширити базову модель MulVAL спеціалізованими предикатами для опису елементів специфічних для телекомунікаційних мереж, особливо з урахуванням саме стільникових мереж зв’язку (таких як функцій 4G/5G core та RAN, сигнальних елементів, мережевих слайсів, специфічних протоколів та інтерфейсів. Це дозволить зберегти переваги логіко-орієнтованого підходу при одночасній адаптації до особливостей мереж.

Оскільки в запропонованому методі автоматизованого тестування на проникнення пропонується використання графової бази знань, є важливим забезпечення інтеграції з нею. Таким чином, пропонується використовувати ідею CyGraph як надбудову над логічним графом атак. Тобто результати виведення мають зберігатися в графовій базі даних (наприклад, Neo4j), де доповнюються контекстною інформацією. Це зможе забезпечити сумісність із динамічним графом знань і спрощує інтеграцію з LLM-модулем, який працює з графовими представленнями.

Для покращення механізму завантаження даних в частині інтеграції з результатами сканування і конфігураційними базами, пропонується використання ETL (Extract, Transform, Load) модуля, що зможе автоматизувати перетворення вихідних форматів (звітів Nmap, Nessus, CMDB тощо) у факти логічної моделі. Це зможе підвищити придатність фреймворку до використання в реальних телекомунікаційних мережах.

Щодо підтримки ризик орієнтованого аналізу та ймовірнісних оцінок, пропонується використання додаткового шару оцінювання ризиків поверх логічного графа. Вузлам і ребрам можуть призначатися ймовірності успішної експлуатації, ваги впливу на сервіси тощо. Це узгоджується з задачею формування кількісних метрик безпеки в рамках методу.

У контексті відкритого інтерфейсу взаємодії з інтелектуальними компонентами, запропонована модель не є монолітною, на відміну від існуючих. Пропонується спроєктувати її як модуль з чітко визначеним API для читання/запису графа та пов’язаних атрибутів. Це дозволить MARL-модулю використовувати побудований граф як середовище, а LLM-модулю – як джерело структурованих знань, що особливо важливо для реалізації запропонованого методу.

2.3.1. Модель нового фреймворку

Запропонований удосконалений фреймворк побудови графа атак для телекомунікаційних мереж матиме назву TelLAG (Telecom Logical Attack Graph) який поєднає логічний фреймворк побудови графів атак (на основі MulVAL) з динамічним графом знань та механізмами ризик-орієнтованого аналізу.

Даний фреймворк виконуватиме роль, так званого «ядра», яке прийматиме на вхід модель телекомунікаційної мережі та множину вразливостей, за допомогою логічних правил будуватиме логічний граф атак, синхронізуватиме його з динамічним графом знань, збагачуватиме вершини й ребра атрибутами ризику та контексту, а також надаватиме єдиний формалізований інтерфейс для LLM та MARL модулів.

Архітектурна схема запропонованого удосконаленого фреймворку наведена на рисунку 2.5. Загалом, даний фреймворк організований чотирма рівнями: джерела даних та ETL, логічне ядро, рівень генерації графа атак і ризик-аналізу, а також рівень інтеграції з динамічним графом знань та зовнішніми компонентами запропонованого методу автоматизованого тестування на проникнення.

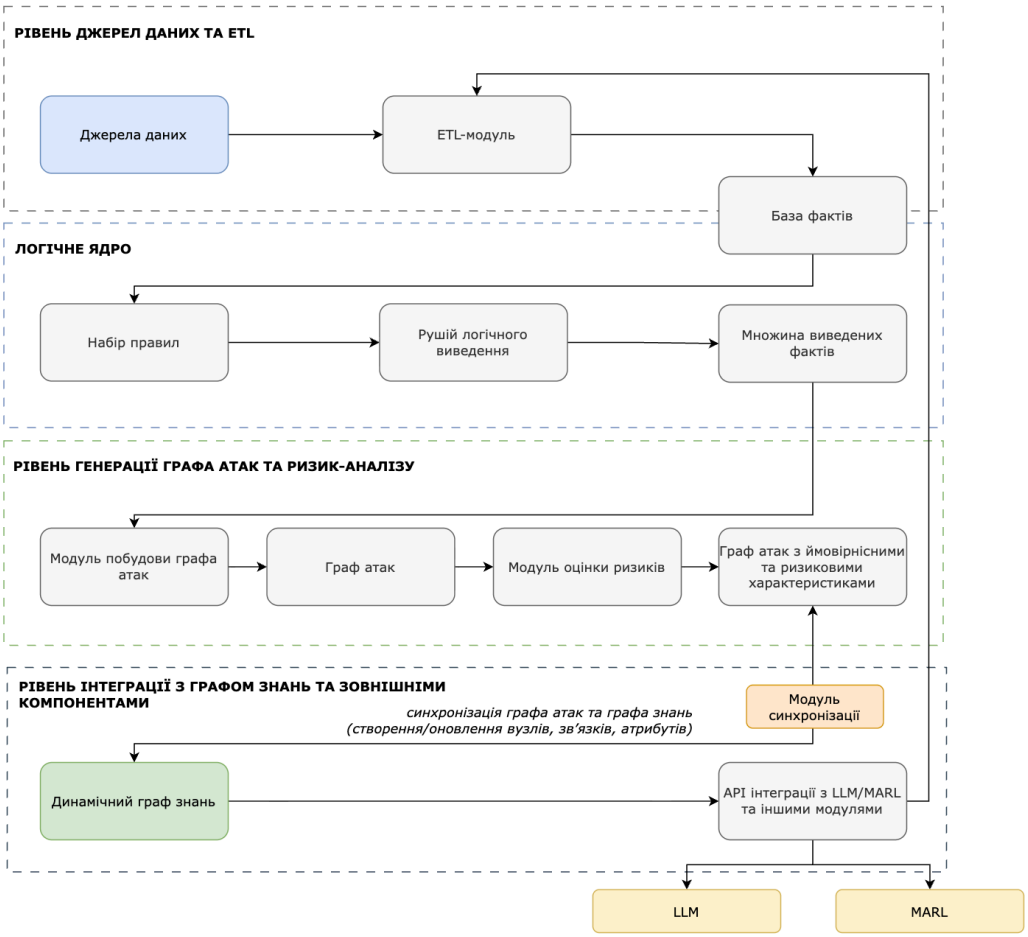

Рис. 2.5. Архітектура удосконаленого фреймворку

На рівні джерел даних та ETL агрегуються зовнішні та внутрішні джерела інформації про телекомунікаційну мереж. Модуль ETL виконує попередню обробку,

нормалізацію та перетворення цих даних у єдину базу фактів, яка надалі використовується логічним ядром фреймворку. Тобто, на першому рівні здійснюється перехід від «сирих» технічних даних до формалізованого представлення стану мережі.

На рівні логічного ядра включені набір правил та рушій логічного виведення, що формує множину виведених фактів після обробки. На основі зібраної бази фактів, сформованої ETL, рушій обчислює цю множину, що описує досяжні стані системи з урахуванням конфігурацій, вразливостей та доменно-специфічних залежностей телекомунікаційної мережі. Таким чином, даний рівень реалізовує логіко-орієнтовану складову від MulVAL, розширену правилами, специфічними для телекомунікаційних мереж.

На рівні генерації графа атак та ризик-аналізу, модуль побудови графа атак перетворює множину виведених фактів на граф атак, де вершини відповідають можливим станам або умовам, а ребра – крокам атак та їх причинно-наслідковим зв’язкам. Модуль оцінки ризиків збагачує цей граф атрибутами ймовірностей, впливу на сервіси та іншими показниками, формуючи ймовірнісний (або ризиковий) граф атак, що використовуватиметься для подальшої пріоритезації сценаріїв пентесту.

Рівень інтеграції з графом знань та зовнішніми компонентами передбачає синхронізацію отриманого графа з динамічним графом знань. Таким чином, створюються та оновлюються відповідні вузли, зв’язки та атрибути, що відображають як технічні характеристики мережі, так і контекст бізнес-сервісів, політик чи історичних подій безпеки. За допомогою API інтеграції, фреймворк надає доступ до цих структур LLM та MARL модулям, для планування та аналізу, та пошуку оптимальних маршрутів атаки й іншим підсистемам методу відповідно. А отже, так забезпечується їх робота на основі єдиної, узгодженої моделі мережі й загроз.

2.3.2. Формалізація запропонованого фреймворку

Запропонований фреймворк можна формально задати як наступний кортеж, який складається зі структурних множин та множин відображення та функції:

$$DTelLAG = \langle L, F_{tel}, R_{tel}, G_A, G'_A, E_{ETL}, I_{inf}, G_{gen}, \phi_K, \rho, API \rangle, \quad (2.22)$$

Структурні множини представлені $L, F_{tel}, R_{tel}, G_A, G'_A$. Тут, L – це логічна мова (наприклад, Datalog), що може задаватися як кортеж $L = \langle Pred, Const, Var \rangle$, де $Pred$ – множина предикатів, $Const$ – множина констант, а Var – множина змінних. Множина предикатів включає як базові предикати MulVAL, так і додаткові специфічні для телекомунікаційних мереж.

За допомогою F_{tel} відображається множина фактів телеком-домену, що є підмножиною множини всіх можливих фактів мови L: $F_{tel} \subseteq Facts(L)$. Факти описують конкретний стан мережі N, вразливості V та додаткову інформацію з джерел даних D.

Множина правил виведення позначена як R_{tel}, що включає в себе R_{base} – базові правила логічного фреймворку (досяжність, привілеї, експлуатація вразливостей) та доменно-специфічні правила для телекомунікаційних мереж R_{dom}. Загалом дана множина може бути представлена як $R_{tel} = R_{base} \cup R_{dom}$.

Оскільки G_A представлено як $G_A = \langle S, E_A \rangle$, то ймовірнісний граф, відповідно, позначаємо як G'_A, що може бути представлений наступним чином:

$$DG'_A = \langle S, E_A, \omega \rangle, \quad (2.23)$$

де $\omega: S \cup E_A \to \mathbb{R}_{\geq 0}$ – це функція ваг, що задає ймовірності, вплив, критичність або інші ризикові характеристики станів та переходів.

До відображення та функції відносяться $E_{ETL}, I_{inf}, G_{gen}, \phi_K, \rho, API$. Модулем, що перетворює вхідні дані на факти є E_{ETL}, який задається як: $E_{ETL}: D \to F_{tel}$. Для кожного окремного джерела s можна задати окреме відображення $E_s: D_s \to F_{tel}$.

Рушій логічного виведення позначений як I_{inf} та може бути формалізований наступним чином:

$$I_{inf}: (F_{tel}, R_{tel}) \to C, \quad (2.24)$$

де $C \subseteq Facts(L)$ – множина виведених фактів, тобто логічне замикання фактів відносно правил.

Модуль побудови графа атак описаний як $G_{gen}: C \to G_A$, тобто на основі множини всіх виведених фактів формується граф атак G_A з множиною станів S та ребер E_A.

За модуль ризик-орієнтованого оцінювання прийнято $\rho: (G_A, K) \to G'_A$, який призначає ваги ω вершинам і ребрам графа атак залежно від інформації в графі знань K (критичність сервісів, політики, історія атак тощо).

Модуль синхронізації з динамічним графом знань може бути формалізований як:

$$\phi_K: (G'_A, K_{old}) \to K_{new}, \tag{2.25}$$

який оновлює граф знань на основі актуального графа атак, створюючи/оновлюючи вузли, зв'язки та атрибути, що описують знайдені шляхи атак, підтверджені вразливості й інші артефакти.

Множиною інтерфейсів доступу до результатів роботи фреймворку для зовнішніх компонентів (LLM, MARL, системи візуалізації тощо) є множина API. Формально, можна виділити:

- API_{query} – набір функцій запиту, які повертають, наприклад, підграфи, шляхи атак, метрики ризику та описуються як:

$$API_{query} \subseteq \{f: (G'_A, K) \to requests\ results\}, \tag{2.26}$$

- API_{update} – набір функцій оновлення, що дозволяють зовнішнім модулям додавати нові факти та коригування знання, що описуються як:

$$API_{update} \subseteq \{f: tests\ results \to F_{tel} \cup K\}. \tag{2.27}$$

- API_{export} – функції експорту у формати, адаптовані для MARL, LLM або інших аналітичних інструментів.

Таким чином, роботу запропонованого удосконаленого фреймворку для побудови графів атак можна подати як композицію відображень:

$$D \xrightarrow{E_{ETL}} F_{tel} \xrightarrow{I_{inf}} C \xrightarrow{G_{gen}} G_A \xrightarrow{\rho} G_A' \xrightarrow{\phi_K} K_{new} \xrightarrow{API} LLM, MARL, etc. \quad (2.28)$$

Тобто фреймворк приймає множину вхідних даних D, перетворює її на множину фактів F_{tel}, на основі яких, за допомогою правил R_{tel} формує множину виведених фактів C. Далі з C будується логічний граф атак G_A, що збагачується ризиковими характеристиками, утворюючи G_A'. Отриманий граф синхронізується з динамічним графом знань K, а через інтерфейси API стає доступним для LLM та MARL модулів як формалізоване середовище для планування та оптимізації сценаріїв тестування на проникнення телекомунікаційної мережі.

2.4. Інструменти тестування на проникнення та їх оркестрація

Інструменти тестування на проникнення у запропонованому методі відіграють роль виконавчого шар, який реалізовує у телекомунікаційній мережі дії, сформовані інтелектуальним оркестратором на основі даних фреймворку побудови графу атак.

Враховуючи специфіку середовища, можна сформувати наступні вимоги до вибору пентест-інструментів: функціональне покриття етапів тестування, адаптованість до телекомунікаційних мереж, зокрема стільникових мереж зв’язку, автоматизованість та придатність до оркестрації, стандартизований формат вихідних даних, а також масштабованість та продуктивність. Окремо можна розглядати такі критерії як валідація в існуючих дослідженнях та практиці, а також поширюваність і підтримка спільнотою. Однак, ці критерії слугуватимуть виключно для підсилення вибору того чи іншого інструменту.

Критерій функціонального покриття етапів тестування передбачає, що сукупність інструментів повинна забезпечувати підтримку основних етапів пентесту,

від розвідки і сканування, до експлуатації та постексплуатації вразливостей, а також специфічні для телеком-мереж перевірки.

Окрім цього, інструменти мають коректно працювати в мережах з віртуалізованими функціями, SDN/NFV компонентами, сегментацією та підтримувати відповідні протоколи. Звідси й випливає критерій адаптованості до телекомунікаційних мереж.

Для інтеграції інструментів в оркестратор та подальшої взаємодії з фреймворком побудови графа атак, важливим критерієм є автоматизованість та безпосередня придатність до оркестрації. Тому, наявність CLI/API, можливість безінтерактивного запуску, параметризації та інтеграції у зовнішні сценарії є обов'язковою умовою.

Оскільки необхідним є трансформування результатів дій інструментів в факти, важливим аспектом є забезпечення виводу результатів у форматах, придатних для подальшої ETL-обробки. Це можуть бути структуровані логи, XML/JSON-звіти, експортування знайдених вразливостей з прив'язкою до CVE/NVD тощо.

Не менш важливим критерієм є масштабованість та продуктивність. Тому інструменти мають підтримувати роботу в середовищах із великою кількістю хостів та сервісів, характерних для операторських мереж, не створюючи надмірного навантаження на тестовану інфраструктуру.

2.4.1. Категоризація та вибір інструментів тестування на проникнення

З урахуванням сформульованих критерії, у межах запропонованого методу було визначено категорії інструментів тестування на проникнення та обрано найкращі варіанти, які зможуть забезпечити необхідне функціональне покриття. Серед визначених категорій:

- збір даних, мережева розвідка та топологічний аналіз;
- автоматизоване виявлення вразливостей (сканери вразливостей);
- аналіз трафіку та протоколів;
- експлуатація та емуляція вразливостей;
- тестування веб-інтерфейсів та API;

- тестування безпеки бездротового та радіодоступу.

Збір даних, мережева розвідка та топологічний аналіз. Зважаючи на те, що це в цілому є першим етапом тестування на проникнення, критично обрати саме ті інструменти, які дозволять створити повноцінне уявлення про цільове середовище. Таким чином, було обрано найбільш широко використовувані інструменти такі як Nmap, Masscan та Shodan (API).

Автоматизоване виявлення вразливостей. За своєю суттю, це є автоматизовані сканери вразливостей. У нашому випадку вони будуть використовувати в першу чергу для тестування веб-компонентів. До цієї категорії було віднесено наступні інструменти: OpenVAS/Greenbone, Nessus.

Аналіз трафіку та протоколів. Для тестування на проникнення телекомунікаційних мереж, особливо стільникових мереж зв'язку LTE/5G, це є основним етапом тестування, оскільки ряд вразливостей присутній саме на цьому рівні. Тут, в першу чергу, буде використовуватися Wireshark, оскільки він є універсальним аналізатором трафіку, який підтримує широкий набір протоколів, включно з тими, що використовуються в мобільних мережах. Також доцільним є використання спеціалізованого інструменту 5Greplay, що являється «фаззером» мережевого трафіку 5G. Він дозволяє відтворювати та модифікувати трафік, інжектувати сценарії атак у компоненти 5G Core та RAN. Окрім цього, також доцільним інструментом є U-Fuzz, що використовується для «фаззингу» мережевих/IoT протоколів.

Експлуатація та емуляція вразливостей. Після виявлення потенційних вразливостей необхідним є забезпечення можливості їх контрольованого експлуатування. До таких інструментів було віднесено Metasploit Framework, Cobalt Strike, SigPloit та UE-based 5G Pentesting Framework.

Metasploit дозволятиме реалізувати модулі атак на IT-компоненти мережі, такі як сервери, веб-інтерфейси, внутрішні служби. SigPloit призначений для тестування безпеки сигналізаційних протоколів (SS7, GTP, Diameter, SIP/IMS/VoLTE), спеціально розроблений для мережевих операторів. UE-based 5G Pentesting Framework призначений для функціонального та безпекового тестування мереж

стільникового зв’язку п’ятого покоління з точки зору кінцевого користувача (UE). Cobalt Strike буде застосований для сценаріїв, де необхідне моделювання APT (Advanced Persistent Threat) та постексплуатаційних дій.

Тестування веб-інтерфейсів та API. У сучасних телекомунікаційних мережах значна частина функцій управління, окрестрації та аналітики реалізовується через веб-інтерфейси й REST/HTTP-API, тому тестування їх безпеки є також критичною. До групи цих інструментів було обрано Burp Suite, OWASP ZAP, Nikto та SQLMap.

Burp Suite, OWASP ZAP, Nikto будуть використовуватися саме для тестування безпеки веб-додатків і API, які підтримують як інтерактивний режим роботи, так і автоматизовані скани. В межах методу вони відповідатимуть за виявлення вразливостей у веб-порталах управління, API оркестраторів, self-service порталах тощо.

SQLMap – спеціалізований інструмент для виявлення та експлуатації SQL-ін’єкцій, який також доцільно використовувати для веб-компонентів телекомунікаційних систем та мереж, що працюють із базами даних абонентів, білінгу, OSS/BSS тощо.

Результати роботи цих інструментів інтегруються через ETL як факти про вразливості та конфігураційні недоліки компонентів площини управління.

Тестування безпеки бездротового та радіодоступу. Для тестування безпеки, де використовується Wi-Fi або інші бездротові технології, доцільно використовувати спеціалізовані інструменти. Серед таких можна виділити Aircrack-ng та RAN Tester UE.

Aircrack-ng являється набором утиліт для аудиту безпеки Wi-Fi мереж (WEP, WPA, WPA2), який може бути використаний у сценаріях, де абонентський доступ здійснюється через Wi-Fi сегменти, інтегровані з 4G/5G-інфраструктурою. RAN Tester UE – це комплексна платформа для автоматизованого тестування безпеки 5G/O-RAN RAN через інтерфейс Uu.

Узагальнений набір використовуваних інструментів наведено в таблиці 2.2.

Таблиця 2.2

Інструменти тестування на проникнення

Категорія	**Інструменти**	**Застосування**	**Роль у методі**
Збір даних, мережева розвідки та топологічний аналіз	Nmap, Masscan, Shodan (API)	Виявлення активів і сервісів (порти); формування первинної карти топології та поверхні атаки	Формування початкової моделі цільового середовища; генерація фактів про вузли та сервіси для ETL і фреймворку побудови графу атак
Автоматизоване виявлення вразливостей	OpenVAS/Greenbone, Nessus	Автоматизоване виявлення CVE, CWE, перевірка конфігурацій	Формування множини вразливостей та їх атрибутів; передача результатів у ETL як факти про стан IT-/веб-компонентів
Аналіз трафіку та протоколів	Wireshark, 5Greplay, U-Fuzz	Аналіз трафіку та протоколів, з урахуванням LTE/5G технологій; модифікація трафіку, фаззинг протоколів	Виявлення вразливостей на рівні протоколів та трафіку; збагачення моделі для побудови графа атак і графа знань інформацією про поведінку мережі
Експлуатація та емуляція вразливостей	Metasploit Framework, Cobalt Strike, SigPloit, UE	Керована експлуатація та сценарне моделювання	Перевірка можливості здійснення шляхів атак з графа атак; уточнення ризиків;

Закінчення таблиці 2.2

	based 5G Pentesting Framework	атак (IT/сигналізація), UE-тестування 5G, APT і пост-експлуатація	отримання додаткових фактів про успішність/ неуспішність атак
Тестування веб-інтерфейсів та API	Burp Suite, OWASP ZAP, nikto, SQLMap	Тестування безпеки вебдодатків і REST/HTTP API; виявлення помилок конфігурації, ін’єкцій та вразливостей у площині керування	Виявлення вразливостей у веб-порталах управління, API оркестраторів, self-service порталах; інтеграція результатів у ETL як факти про вразливості та недоліки конфігурації
Тестування безпеки бездротового та радіодоступу	Aircrack-ng, RAN Tester UE	Аудит безпеки Wi-Fi; тестування безпеки 5G/O-RAN RAN через інтерфейс Uu з боку UE	Оцінка безпеки бездротових сегментів, інтегрованих, зокрема із LTE/5G-інфраструктурою; генерація фактів про стан RAN/UE для моделі графа знань

2.4.2. Архітектура оркестратора інструментів

У запропонованому методі використовується оркестратор інструментів тестування на проникнення, який забезпечуватиме перетворення високорівневих дій на конкретні виклики інструментів, зазначених у 2.4.1.

Архітектурно, оркестратор інструментів розташовується між інтелектуальним оркестратором (LLM, MARL) та фреймворком побудови графа. З одного боку, він отримуватиме від інтелектуального оркестратора опис абстрактних дій, наприклад, просканувати підмережу X, виконати фаззинг протоколу GTP для конкретного інтерфейсу, а з іншого – керує запуском доступних інструментів. У результаті, він повертає отримані результати до ETL.

Серед основних компонентів оркестратора інструментів можна виділити модуль прийому та валідації планів тестування, шар абстракції дій, модуль маппінгу дій на інструменти, модуль виконання та моніторингу, модуль інтеграції з ETL та графом атак, а також модуль застосування політик та аудиту. Архітектура зображена на рисунку 2.6.

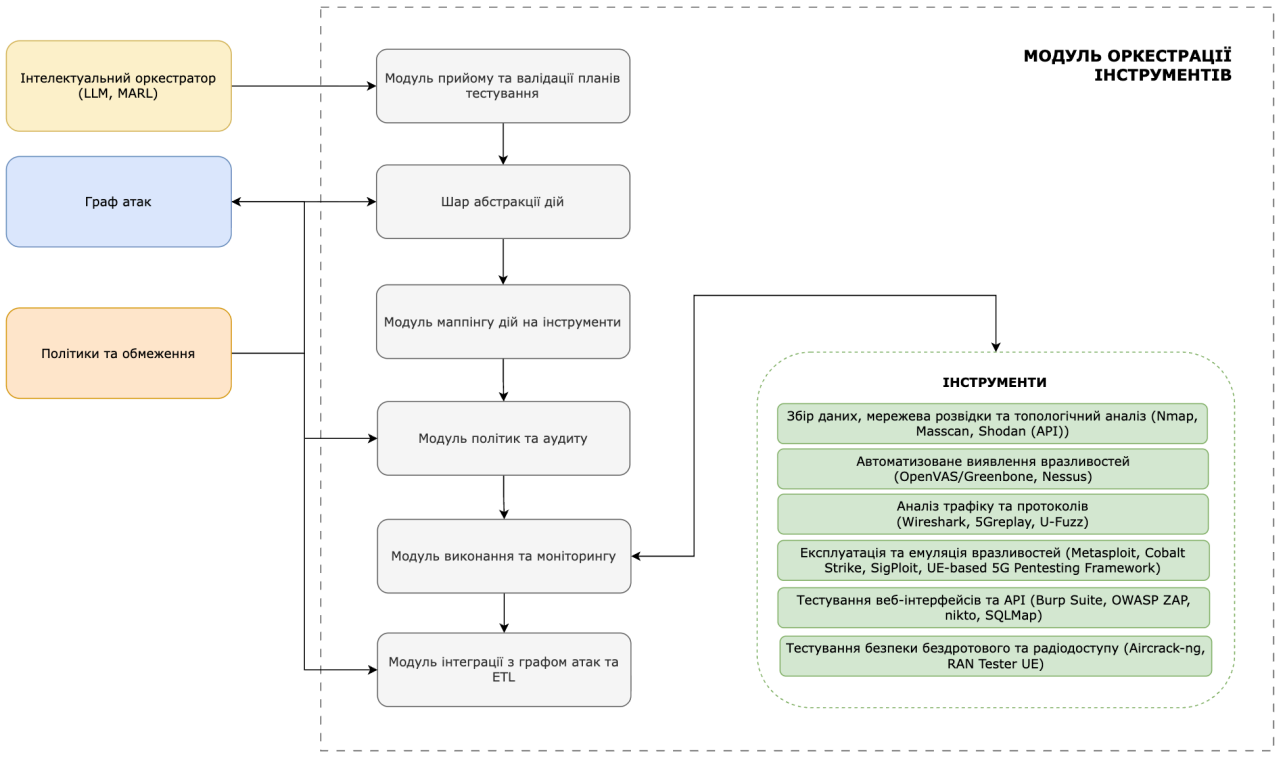

Рис. 2.6. Архітектура модуля оркестрації інструментів

Інтелектуальний оркестратор формує високорівневі плани тестування на проникнення у вигляді послідовностей дій, після чого вони надходять до модуля прийому та валідації планів тестування, де перевіряються на коректність, відповідність політкам і обмеженням, а також узгоджуються з поточним станом мережі, відображеним у графі атак. Нормалізовані плани передаються до шару абстракцій дій, який перетворює їх на уніфікований набір абстрактних дій, наприклад «сканувати підмережу» тощо, незалежних від конкретних інструментів.

Наступним етапом є робота модуля маппінгу дій на інструменти. Даний модуль на основі типу абстрактної дії, контексту графа атак та налаштованих політик обирає відповідну категорію інструментів і формує параметри для їх запуску. Узгоджені дії додатково проходять через модуль політик та аудиту, що накладає обмеження та фіксує всі рішення й запуски в журналі.

За допомогою модуля виконання та моніторингу здійснюється безпосередній запуск інструментів, який надсилає команди у відповідні категорії інструментів і відстежує хід їх виконання. Отримані від інструментів результати, звіти та логи передаються до модуля інтеграції з графом атак та ETL, де вони перетворюються на формалізовані факти й атрибути. Через рівень ETL факти потрапляють до графу атак, що призводить до його оновлення та, відповідно, оновлення динамічного графа знань. Таким чином, оркестратор інструментів тестування на проникнення реалізовує замкнений цикл: від планування дій інтелектуальним оркестратором до їх виконання конкретними інструментами й повернення результатів.

2.5. Використання таксономій MITRE ATT&CK та FiGHT у запропонованому методі

Ефективність автоматизованого тестування на проникнення значною мірою залежить від того, на скільки модель загроз і сценаріях атак узгоджені з усталеними таксономіями поведінки суб’єктів загроз та онтологіями типових атак. В даному випадку, в методі використовуються дві бази, які доповнюють одна одну: MITRE ATT&CK та FiGHT (5G Hierarchy of Threats).

MITRE ATT&CK є відкритою, постійно оновлюваною, базою TTP (techniques, tactics, procedures), що описують поведінку суб'єктів загроз, побудованою на основі спостережень у реальних інцидентах. Вона організована у вигляді матриць для різних доменів (enterprise, mobile, ICS тощо), де тактики відображають цілі суб'єкта загрози, а техніки – конкретні способи досягнення цих цілей [43].

FiGHT є спеціалізованою базою тактик і технік атак саме на інфраструктуру 5G, що також розроблена MITRE. Вона являє собою ієрархічну модель загроз для мереж стільникового зв'язку п'ятого покоління, їх застосунків та пристроїв та складається з технік трьох типів: теоретичні, лабораторні (PoC) та спостереження в реальних системах [44]. Таким чином, це дозволяє враховувати як уже підтверджені, так і потенційні вектори атак.

У запропонованому методі ці таксономії інтегруються на трьох рівнях: динамічний граф знань, фреймворк для побудови графу атак та інтелектуального оркестратора.

Інтеграція з динамічним графом знань. Динамічний граф знань K містить не лише інформацію про мережу, вразливості, стани системи, а також й онтологію загроз. У цій онтології, вузли типу Technique відповідають техніками MITRE ATT&CK та FiGHT, Tactic – описують тактики (цілі) суб'єкта загрози, а вузли, що описують стани/події безпеки мережі, пов'язуються з відповідними техніками ребрами типу uses_technique, implements_tactic тощо. Завдяки цій інтеграції, будь-який сценарій атаки, який генерується за допомогою графу атак, може бути інтерпретований у стандартизованих формах.

Інтеграція у фреймворку побудови графу атак. Тут таксономії використовуються у двох аспектах, множині правил R_{tel} та у модулі ризик-орієнтованої оцінки.

У множині правил R_{tel} чатсина доменно-специфічних правил задається з урахуванням відповідності певним технікам. Наприклад, правило, яке описує неавторизований доступ до інтерфейсу управління мережевою функцією в 5GC через слабку автентифікацію, може бути пов'язане з однією з технік FiGHT, що описує атаки на управлінські інтерфейси 5G.

У модулі ризик-орієнтованої оцінки ρ, ваговий коефіцієнт ω в графі атак визначатиметься з урахуванням критичності відповідної техніки в MITRE ATT&CK та FiGHT, типу техніки (актуально для FiGHT) та релевантності техніки до конкретного домену мережі.

Це дозволятиме задавати пріоритети тестування на рівні моделі. Наприклад, надавати більшу вагу сценаріям, що реалізовують реально спостережені критичні техніки FiGHT або часто використовувані для телеком-середовища техніки MITRE ATT&CK.

Інтеграція в інтелектуальний оркестратор. Завдяки цій інтеграції забезпечуватимуться семантичне планування LLM модуля, обмеження та збагачення простору дій MARL та інтерпретованість результатів.

LLM модуль зможе формувати цілі та сценарії тестування у термінах тактик і технік. MARL модуль зможе розглядати простір дій, як дії, згруповані за техніками MITRE ATT&CK та FiGHT. Це дозволить обмежувати навчання певними тактиками та техніками, будувати функції винагороди, які підвищать цінність шляхів, що реалізовують найбільш критичні техніки.

На процедурному рівні запропонований метод розглядає конкретні реалізації технік у вигляді сценаріїв застосування інструментів тестування на проникнення. Ці процедури будуть відображатися в оркестраторі інструментів як конкретні команди та параметри запуску обраних засобів, що забезпечує зв’язок між абстрактними техніками MITRE ATT&CK та FiGHT та практичними кроками застосування в цільовому середовищі.

Загалом, інтеграція MITRE ATT&CK та FiGHT у метод забезпечує ряд переваг, таких як стандартизована модель загроз, кількісне оцінювання покриття, підвищена релевантність для 5G-домену та краща інтерпретованість результатів.

Формально представити MITRE ATT&CK та FiGHT у запропонованому методі можна наступним чином. Нехай $Tech_A$ – множина технік MITRE ATT&CK, $Tech_F$ – множина технік FiGHT, Tac_A – множина тактик MITRE ATT&CK, а Tac_F – множина технік FiGHT.

Тоді повна множина технік та тактик, з якими працює запропонований метод, формально може бути представлена як:

$$Tech = Tech_A \cup Tech_F, Tac = Tac_A \cup Tac_F \quad (2.29)$$

Оскільки G_A, граф атак, представлено як $G_A = \langle S, E_A \rangle$, де S – множина станів, а E_A – множина переходів, то існує відображення $\tau: S \cup E_A \to 2^{Tech}$, яке кожному стану або кроку атаки ставить у відповідність підмножину технік $\tau(x) \subseteq Tech$, що реалізовуються або проявляються в цьому стані/кроці.

Аналогічно, для динамічного графа знань K існують такі підмножини як $V_{Tech} \subseteq V_K, V_{Tac} \subseteq V_K$, які містять вузли технік і тактик відповідно, а ребра задають відображення між станами/подіями та елементами множин $Tech, Tac$.

У модулі ризик-орієнтованої оцінки вводиться вагова функція $\omega: S \cup E_A \to \mathbb{R}_{\geq 0}$, яка може бути представлена у наступному вигляді:

$$T\omega(x) = f(\tau(x), crit(\tau(x)), type_F(\tau(x)), dom(\tau(x)), \quad (2.30)$$

де $crit(\tau(x))$ – оцінка критичності відповідних технік за MITRE ATT&CK/FiGHT, $type_F$ – тип техніки у FiGHT (теоретична, PoC, спостережена), dom – релевантність до конкретного домену мережі.

Таким чином, ваговий коефіцієнт ω для кожного стану або кроку атаки безпосередньо залежить від того, які техніки MITRE ATT&CK/FiGHT реалізовуються у відповідному фрагменті графа атак.

2.6. Узагальнений робочий процес розробленого методу

Узагальнений робочий процес запропонованого методу автоматизованого тестування на проникнення телекомунікаційної мережі наведено на рисунку 2.7.

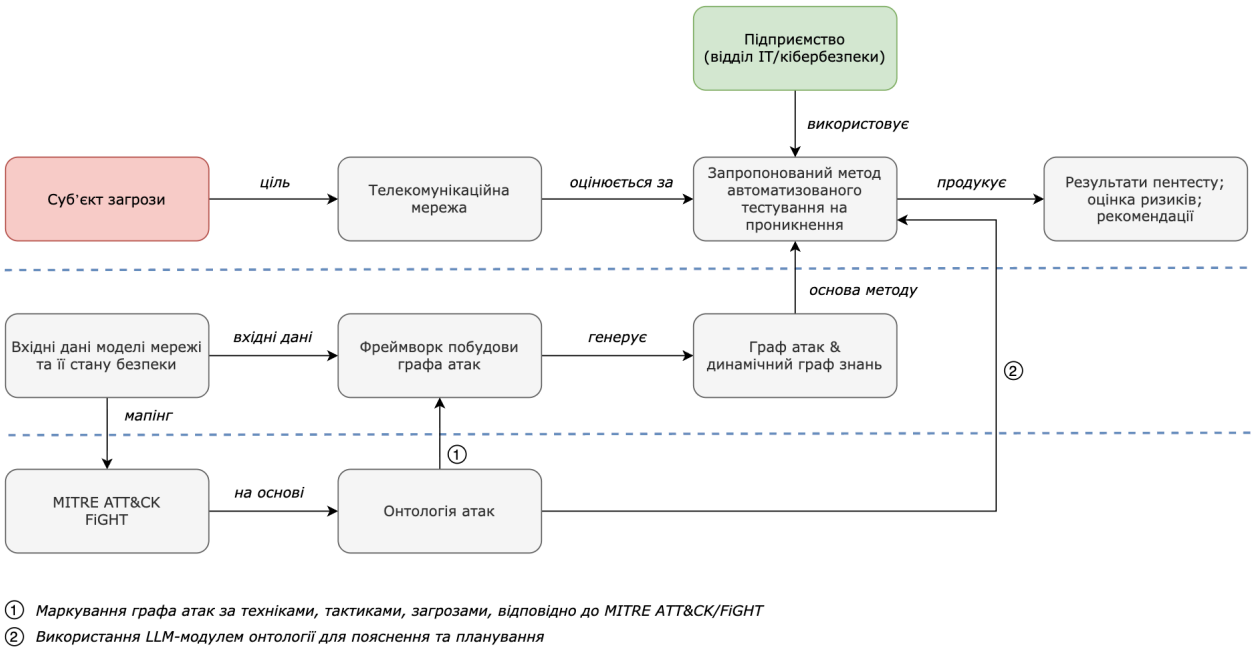

Рис.2.7. Узагальнений робочий процес методу

У верхній частині схеми показано контекст загроз. Суб'єкт загрози розглядає телекомунікаційну мережу як ціль для атаки. Щоб проактивно оцінити стійкість мережі, виправити всі недоліки в безпеці, підприємство (відділ IT/кібербезпеки) використовує запропонований метод автоматизованого тестування на проникнення. На основі внутрішньої моделі мережі та її стану безпеки метод формує результати пентесту, оцінку ризиків і рекомендації щодо покращення захисту.

В центральній частині відображено формальне «ядро» методу. Вхідні дані моделі включають модель телекомунікаційної мережі (інфраструктури), політики безпеки та SLA, відомі й виявлені вразливості, а також конфігураційні дані та журнал подій. Ці артефакти надсилаються як вхідні дані до фреймворку побудови графа атак, який генерує граф атак та динамічний граф знань. Отримані графові структури виступають основою методу, адже саме на них спираються інтелектуальні компоненти під час сценаріїв тестування, вибору цільових ділянок мережі та інтерпретації результатів.

Внизу схеми показано інтеграцію з базами MITRE. Вхідні дані моделі виконують маппінг на таксономії MITRE ATT&CK та FiGHT, на основі чого формується онтологія атак. Сама онтологія використовується для маркування графа

атак техніками, тактиками та категоріями загроз відповідно до MITRE ATT&CK/FiGHT. Також вона застосовується LLM-модулем для пояснення отриманих результатів і побудови зрозумілих, ризик-орієнтованих сценаріїв подальшого тестування.

ВИСНОВКИ ДО РОЗДІЛУ 2

У Розділі 2 обґрунтовано доцільність переходу від суто rule-based та класичних model-based підходів до гібридного інтелектуального методу, здатного працювати в умовах високої складності LTE/5G/Next-G середовищ (NFV, network slicing, міжмережева взаємодія) шляхом поєднання графів атак, LLM та багатоагентного навчання з підкріпленням.

Запропоновано структурно цілісний метод автоматизованого тестування на проникнення телекомунікаційних мереж, у якому поєднано підсистеми моделювання мережі та загроз (включно з графом атак і динамічним графом знань), інтелектуальний оркестратор (LLM+MARL), оркестратор інструментів, а також підсистеми політик/обмежень і формування результатів.

Виконано математичну формалізацію методу на основі теорії множин: визначено множини вхідних даних (OSINT та внутрішні дані оператора), модель мережі (вузли, зв’язки, атрибути), що створює підґрунтя для відтворюваного опису процедур та подальшої алгоритмізації.

Показано, що метод є адаптивним за рахунок замкненого циклу: результати виконання процедур тестування використовуються для оновлення графа знань та параметрів RL-агентів, а на прикінцевих етапах забезпечується інтерпретація результатів із подальшим поверненням до етапів планування/виконання до досягнення заданих критеріїв.

Проведено огляд і порівняльний аналіз фреймворків побудови графів атак із урахуванням телеком-вимог (масштабованість операторського рівня, доменна розширюваність, інтеграція з даними/ризик-аналізом тощо), встановлено, що жодне з наявних рішень не закриває весь спектр вимог, а як базове рішення найбільш доцільно

використовувати MulVAL із подальшим цільовим удосконаленням під телеком-домен.

Запропоновано концепцію удосконаленого фреймворку TelLAG (Telecom Logical Attack Graph), що поєднує логічну побудову графа атак на основі MulVAL із динамічним графом знань та ризик-орієнтованим шаром; визначено його роль як «ядра» з формалізованим інтерфейсом доступу для LLM/MARL.

Розроблено архітектуру оркестрації інструментів пентесту як проміжного виконавчого шару між інтелектуальним оркестратором та фреймворком графа атак; виокремлено ключові модулі (прийом/валідація планів, абстракція дій, маппінг на інструменти, виконання/моніторинг, інтеграція з ETL і графом атак, політики та аудит), що забезпечує керованість і трасованість виконання.

Інтегровано таксономії MITRE ATT&CK та FiGHT на рівнях графа знань, фреймворку графа атак і інтелектуального оркестратора, що забезпечує семантичне планування, інтерпретованість результатів і можливість кількісного оцінювання покриття та пріоритезації сценаріїв для 5G-домену.

Таким чином, у Розділі 2 сформовано завершену концептуально-формальну основу методу (компонентна модель, математична формалізація, алгоритміка, інструментальна оркестрація та стандартизована онтологія загроз), що створює безпосередній теоретико-методичний базис для подальшої реалізації прототипу та його експериментальної верифікації.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИКОРИСТАННЯ У МЕТОДІ

Ключовою особливістю запропонованого методу автоматизованого тестування на проникнення телекомунікаційних мереж є наявність інтелектуального компоненту оркестратора, що має автоматично обирати послідовність дій тестування з урахуванням поточного стану мережі, структури графа атак, обмежень за ресурсами та вимог до заданих політик проведення тестування.

Для реалізації такої інтелектуальної поведінки доцільно розглядати підходи машинного навчання, зокрема навчання з підкріпленням (reinforcement learning, RL) та його мультиагентні розширення. У такому випадку, процес тестування на проникнення інтерпретується як послідовна взаємодія одного або кількох агентів із середовищем, де агенти вчаться приймати рішення на основі отриманого досвіду, максимізуючи певну цільову функцію (винагороду), пов’язану з ефективністю та повнотою тестування. Саме такий підхід дозволить врахувати невизначеність, часткову спостережуваність стану системи та складну комбінаторну природу простору можливих дій.

3.1. Постановка задачі використання RL у методі

Основною метою запропонованого методу є автоматизація побудови та виконання сценаріїв тестування на проникнення на основі графів атак, таксономій та онтологій MITRE, а також даних про конфігурацію та вразливості мережі. У даному випадку, система має навчитись приймати послідовні рішення в умовах невизначеності, які дії виконувати, на яких вузлах, у якій послідовності та з якими параметрами, щоб досягти всіх поставлених цілей пентесту за обмеженого «бюджет» часу й ресурсів.

Для цього у методі пропонується використання алгоритмів RL, які розглядаються взаємодію агента з середовищем як послідовний процес прийняття рішень у вигляді марковського процесу прийняття рішень (MDP) або його багатоагентного узагальнення (рис. 3.1).

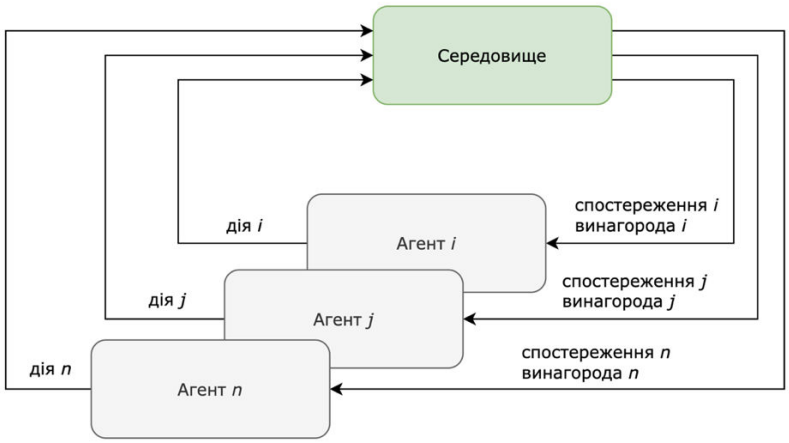

Рис. 3.1. Багатоагентне представлення

Агенти спостерігають поточний стан мережі, представлений через граф атак та знань, обирають дію (наприклад, запуск певного інструмента, спробу експлуатації або крок розвідки), отримують винагороду, що відображає корисність виконаної дії та переходять до нового стану.

Таким чином, задача використання навчання з підкріпленням у запропонованому методі може бути сформульована як задача синтезу політики, яка максимізує очікувану сумарну винагороду, пов’язану з досягненням цілей тестування на проникнення за дотримання обмежень політики безпеки. При цьому, політика має збалансовувати використання вже відомих ефективних дій з дослідженням нових шляхів атаки та невідомих ділянок графа, інтегруючи процес «дослідження» безпосередньо у загальний алгоритм роботи методу.

3.1.1. Модель середовища на основі графа атак

У запропонованому методі агенти RL взаємодіють з формалізованим середовищем, яке будується на основі графа атак та динамічного графа знань, що генеруються удосконаленим фреймворком TelLAG. Тому, модуль працює вже з

агрегованим та нормалізованим представленням стану мережі, її вразливостей та можливих шляхів атаки.

На вході фреймворку TelLAG знаходяться модель мережі, результати сканування вразливостей та інші артефакти, політики безпеки та бізнес-контекст, а також таксономії MITRE, якими позначаються відповідні вузли та ребра. На основі цих даних будується ймовірнісний граф атак:

$$TG^{att} = (V^{att}, E^{att}, \omega, \lambda), \tag{3.1}$$

де V^{att} – множина вузлів, що відповідають окремим станам безпеки або логічним умовам; $E^{att} \subseteq V^{att} \times V^{att}$ – множина орієнтованих ребер, що відображають потенційні кроки атаки; ω – це функція ваг, яка задає ймовірність успішної експлуатації, вплив на ризик або інші кількісні характеристики станів і переходів; λ – набір міток для вузлів та ребер (тактики/техніки/категорії MITRE, типи вразливостей тощо).

Середовище навчання з підкріпленням можна позначити як $\mathcal{E}_{RL} = \langle S, A, P, R, \gamma \rangle$.

Представлення стану середовища. Стан середовища описує не окремий вузол графа атак, а конфігурацію графа з точки зору прогресу атаки, доступних знань та ресурсів. Для кожного дискретного кроку *t* стан можна подати у вигляді кортежу $s_t = \langle V_t^{disc}, V_t^{comp}, C_t, B_t \rangle$, де $V_t^{disc} \subseteq V^{att}$ – множина відкритих (досліджених) вузлів графа атак, які вже відомі системі на момент часу *t*. Ця множина відповідає поточним знанням про можливі умови й кроки атаки, отриманих з даних фреймворку TelLAG та результатів попередніх дій агентів; $V_t^{comp} \subseteq V_t^{disc}$ – множина досягнутих (compromised) станів, тобто тих вузлів графа атак, передумови яких виконані і вважаються реалізованими; C_t – сукупність отриманих облікових даних, токенів, сесій, контексних атрибутів, які впливають на досяжність наступних вузлів; B_t – залишок доступного «бюджету» (час, кількість дозволених дій, обмеження на трафік і т.д.), що визначає ресурси, які ще можна витратити в межах поточного етапу пентесту.

Кожен стан $s_t \in S$ являється проєкцією поточного графа атак і графа знань на ті аспекти, що є релевантними для прийняття рішень у модулі MARL. Формально це може бути записано як:

$$s_t = \Phi(G_t^{att}, K_t), \tag{3.2}$$

де G_t^{att} – актуальний граф атак, з урахування нових виявлених вузлів і ребер), K_t – динамічний граф знань, а Ф – відображення, що виділяє з них компоненти $V_t^{disc}, V_t^{comp}, C_t, B_t$.

У загальному випадку середовище є частково-спостережуваним. Початково відома лише частина графа атак, а нові вузли та ребра з'являються в міру того, як агенти виконуються дії. Тобто, реальний стан мережі більш повний, ніж те, що бачить агент. Однак, враховуючи, що TelLAG використовується як «проміжний шар», можна вважати, що агент взаємодіє із поточним наближенням графа атак, який послідовно доповнюється результатами його ж дій.

Модель переходів у середовищі. Динаміка середовища задається функцією переходів $P(s_{t+1}|s_t, a_t)$, яка відображає, з якою ймовірністю виконання дії a_t у стані s_t призводить до нового стану s_{t+1}.

Для запропонованого методу функція індукується графом атак та логічним виведенням TelLAG наступним чином:

- кожна дія a_t пов'язана з одним або кількома ребрами $e = (u, v) \in E^{att}$ або з наборами ребер;
- виконання дії на практиці означає виклик відповідного інструмента, через оркестратор інструментів, з параметрами, що відповідають вихідному вузлу u та поточним атрибутам стану C_t;
- результати дії конвертуються в нові факти, які надходять у TelLAG і спричиняють оновлення множини вузлів та ребер графа атак.

Якщо дія a_t відповідає ребру (u, v), для якого в поточному стані виконані всі передумови, то з ймовірністю $p_{succ}(e)$, що задається функцією ваг ω, відбувається успішний перехід: вузол v додається до множини досягнутих станів V_{t+1}^{comp}, у

множину V_{t+1}^{disc} можуть бути додані нові вузли, що стали досяжними після виконання дій, множина C_{t+1} оновлюється, бюджет B_{t+1} зменшується з урахуванням вартості виконаної дії.

У протилежному випадку $(1 - p_{succ}(e))$, новий стан s_{t+1} може відрізнятися від s_t лише оновленим бюджетом, а також ознаками детекції або блокування (якщо задати такий функціонал моделювання в граф знань).

Як окремий випадок розглядаються дії, спрямовані на розвідку. Як правило, такі дії, не переводять агентів безпосередньо до нових «компрометованих» вузлів, але змінюють множину V_{t+1}^{disc}, додаючи нові вузли та потенційні ребра до графа атак. Відповідно до термінів RL, це означає, що перехід $s_t \to s_{t+1}$ відображає збільшення доступної інформації та розширення простору майбутніх дій.

Початкові та завершальні стани. Початковий стан епізоду s_0 формується на основі поточних вхідних даних, таких як відомі топології та конфігурації мережі, результати попередніх аудитів та сканувань, фіксовані обмеження політики (дозволені зони тестування, заборона на певні типи атак тощо), задані цілі сценарію пентесту (які активи треба перевірити, які типи впливу допустимі).

Множина термінальних станів $S_{term} \subseteq S$ включає, зокрема стани, у яких досягнуто поставлені цілі, у яких вичерпано бюджет B_t, у яких жодна подальша дія не задовольняє обмеження політик або не має шансів змінити стан мережі.

У кожному термінальному стані епізод завершується, а на основі накопиченої траєкторії станів, дій та винагород формується звіт та оновлюються політики агентів.

Таким чином, у запропонованому методі, середовище для MARL є дискретним, епізодичним та індукованим динамічним графом атак, що будується фреймворком TelLAG.

3.1.2. Багатоагентна постановка

Складність та масштабність сучасних телекомунікаційних мереж, велика кількість потенційних векторів атаки та необхідність паралельного дослідження ділянок топології роблять використання одного RL-агента малоефективним. Адже простір станів та дій стає надто великим, а процес дослідження, відповідно, повільним

і нестабільним. Тому задача автоматизованого тестування на проникнення у запропонованому методі формулюється у вигляді кооперативного багатоагентного навчання з підкріпленням, де кілька агентів взаємодіють із спільним середовищем та узгоджено навчаються досягати спільної мети [45].

Формальна модель MARL. Нехай $I = \{1, \dots, N\}$ – множина агентів. Багатоагентне середовище, індуковане графом атак, описується як:

$$\mathcal{E}_{MARL} = \langle I, S, \{A_i\}_{i \in I}, P, R, \{O_i\}_{i \in I}, \gamma \rangle, \tag{3.3}$$

де S – множина глобальних станів середовища, A_i – множина дій агента i, $P(s_{t+1}|s_t, a_t^1, \dots, a_t^N)$ – функція переходів, що описує еволюцію графа атак та мережевого стану під впливом спільної дії агентів, $R(s_t, a_t^1, \dots, a_t^N, s_{t+1})$ – функція винагороди (у нашому випадку, спільна для всіх агентів), O_i – множина локальних спостережень агента i, $\gamma \in (0,1)$- коефіцієнт дисконтування.

У кожний дискретний момент часу t, середовище знаходиться у глобальному стані $s_t \in S$, кожен агент i отримує локальне спостереження $o_t^i = \Omega_i(s_t) \in O_i$, яке є (можливо неповною) функцією від глобального стану; кожен агент обирає дію $a_t^i \in A_i$ згідно зі своєю політикою $\pi_i: O_i \to \Delta(A_i)$, де $\Delta(A_i)$ – множина розподілів ймовірностей на A_i; сукупність дій утворює спільну дію $a_t = (a_t^1, \dots, a_t^N) \in A_1 \times \dots \times A_N$; середовище переходить у новий стан s_{t+1} відповідно з P та повертає скалярну винагороду $r_t = R(s_t, a_t, s_{t+1})$, яка інтерпретується як якість поточного кроку сценарію пентесту.

Таким чином, агенти оптимізовують спільну цільову функцію, очікувано сумарну дисконтовану винагороду:

$$J(\{\pi_i\}_{i \in I}) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t], \tag{3.4}$$

де епізод триває до досягнення термінального процесу стану $s_T \in S_{term}$.

У реальному сценарії пентесту середовище є частково спостережуваним, тобто окремий агент бачить лише частину графа атак або лише локальні результати виконаних ним дій. Тому формально задача описується як кооперативний Dec-POMDP (decentralized partially observable Markov decision process) [46]. Практично, це відображається у використанні архітектур типу «централізоване навчання, децентралізоване виконання» (CDTE), коли під час навчання допускається доступ до глобальної інформації, але на етапі виконання кожен агент приймає рішення, спираючись лише на власні спостереження [47].

Відповідність агентів компонентам методу. У контексті запропонованого методу, агенти MARL працюють на рівні абстракцій, які надають TelLAG та оркестратор інструментів. Кожна дія агента інтерпретується як запит на виконання певного кроку сценарію атаки, а вже оркестратор інструментів підбирає конкретні технічні засоби.

Загалом, тут є два взаємодоповнювальні способи декомпозиції задачі на агентів: агенти за сегментами/підграфами мережі та агенти за класами дій/ролями атаки.

У першому способі, мережа радіодоступу, транспортна мережі, ядро EPC/5GC, IT-інфраструктура тощо, можуть моделюватися як окремі підграфи в рамках G^{att}, а кожному підграфу ставиться у відповідність агент, який спеціалізується на дослідженні шляхів атаки в межах відповідного домену та приймає рішення, пов’язані з локальними вузлами й ребрами. Взаємодія між агентами відбувається через спільний граф атак і спільну винагороду.

У другому способі, один агент може спеціалізуватися на розвідці, інший – на експлуатації, третій – на ескалації привілеїв тощо (тобто, відповідно до кожного загальновідомого етапу пентесту). Такий поділ дозволяє використовувати різні простори дій A_i та різні архітектури політик дій для агентів різних типів, зберігаючи при цьому спільну функцію винагороди.

Таким чином, взаємодія агентів є опосередкованою через середовище, кожен агент спостерігає оновлений граф атак, який відображає результати дій усіх агентів на попередніх кроках. Однак, можливо й розширення моделі засобами явної

комунікації між агентами, наприклад, обмін повідомленнями, однак базова постановка може бути реалізована і без цього, залишаючи CTDE підхід за основу.

Локальні спостереження та глобальний стан. Незважаючи на те, що глобальний стан задається через повний (наближений) граф атак і граф знань, кожен агент, фактично, працює з власним локальним представленням:

$$o_t^i = \Omega_i(s_t) = \Phi_i(G_t^{att}, K_t), \tag{3.5}$$

де Φ_i виділяє підмножину вузлів, ребер і атрибутів, релевантних для агента i.

Залежно від обраної схеми декомпозиції це можуть бути вузли підграфа, закріпленого за агентом, вузли і ребра, що відповідають певним тактикам/технікам чи інформація про поточні дані (credentials), активні сесії та бюджет дій, доступних конкретному агенту.

При цьому, функція переходів P та функція винагороди R залежать від повного набору дій усіх агентів a_t, що створює для кожного окремого агента нестаціонарне ефективне середовище. Така нестаціонарність є типовою проблемою MARL та компенсується використанням підходів по типу VDN, QMIX [48].

Синхронізація дій та епізодів. Особливістю поставленої задачі є те, що виконання інструментів тестування на проникнення займає певний проміжок часу, але окремі дії можуть виконуватися паралельно. У формальній моделі MARL це відображається через дискретні макрокроки, на кожному з яких усі агенти (або підмножина активних агентів) формують свої дії, оркестратор інструментів перетворює спільну дію a_t на набір конкретних технічних викликів, а після завершення виконання інструментів TelLAG оновлює граф атак та формується новий стан s_{t+1} і обчислюється винагорода r_t.

Формальна мультиагентна постановка узгоджується зі структурою запропонованого методу наступним чином: MARL-агенти приймають високорівневі рішення щодо сценаріїв атаки, TelLAG та динамічний граф знань забезпечують узгоджене подання стану та динаміки середовища, оркестратор інструментів реалізовує обрані дії на рівні конкретних інструментів, а функція винагороди інтегрує

досягнення цілей пентесту, покриття графа так, витрати ресурсу та дотримання політик безпеки.

3.1.3. Простір дій агентів

Ефективність модуля MARL напряму залежить від того, як саме буде декомпозовано задачу між агентами і як буде визначено їх простір дій. Варто зазначити, що з одного боку, дії мають бути достатньо виразними, щоб описувати різні кроки пентесту в цільовому середовищі, а з іншого – простір дій окремого агента не має бути надмірно великим, що може відображатися на якості навчання [49].

Для запропонованого методу буде доцільним обрати декомпозицію за класами дій/ролями, оскільки інструменти в оркестраторі інструментів вже розбиті за класами, тому оптимальним буде поставити цим класам у відповідність спеціалізованих агентів. Іншою вагомою причиною є краща узагальнюваність. Рольові агенти можуть переносити навчені політики між різними мережами й топологіями, тоді як жорстка прив’язка агентів до конкретних сегментів робить перенавчання необхідним при кожній значній зміні структури мережі. Також слід відзначити й співрозмірність простору дій. Для кожного класу дій можна задати відносно компактний та однорідний простір дій, уникаючи суттєвого зростання розмірності, яке виникає про спробі дати одному агенту всі етапи пентесту одразу. Окрім цього, тактики MITRE вже фактично задають природні «ролі» поведінки нападника, їх можна безпосередньо відображати на ролях агентів і множинах їх дій.

При цьому, інформація про сегменти мережі не втрачається, адже входить до стану середовища і параметризації дій. Таким чином, сегментація мережі відображається не у множині агентів, а в цілях і параметрах дій рольових агентів.

Оскільки, цільову середовище це телекомунікаційна мережа, то слід визначити специфічну деталізацію етапів пентесту. З урахуванням специфіки різних типів телекомунікаційних мереж, фіксованих (FTTx, xDSL тощо), мобільних (2G/3G/4G/5G), конвергентних IP/MPLS-мереж, а також мереж передачі даних і голосу наступних поколінь, доцільним буде виділити узагальнені етапи пентесту, які одночасно залишаються сумісними з класичною логікою пентесту, відображають

доменно-специфічні площини атаки оператора зв’язку та не прив’язані жорстко до технології доступу.

Таким чином, вважатимемо, що у даному випадку використовуються наступні етапи тестування на проникнення:

- *Eman 1 (Reconnaissance & Network scanning).* Виконується інвентаризація доменів мережі оператора (мережі доступу, транспортна мережа, ядро, платформи послуг, міжмережеві/міжоператорські інтерфейси), активне й пасивне сканування вузлів, сервісів, протоколів, версій програмного забезпечення, виконання мережевого й сервісного сканування, збору конфігураційних і діагностичних даних у дозволених зонах тестування.
- *Eman 2 (Vulnerabilities validation & Initial access).* Перевірка та експлуатація вразливостей, помилок конфігурації в різних доменах мережі, отримання перших «опорних точок» (initial foothold); фактично реалізовується «proof of exploitation» і підтвердження, що та чи інша знайдена вразливість може бути використана в середовищі тестування.
- *Eman 3 (Lateral movement & Privilege escalation).* Відображає класичний етап підтримки доступу та ескалації привілеїв у багатодоменному середовищі.
- *Eman 4 (Cross-network & Service-oriented attacks).* Концептуально являється підвидом класичних етапів, орієнтованими на особливості телекомунікаційних мереж. На цьому етапі аналізуються і реалізовуються вектори атак пов’язані з міжмережевими та міжоператорськими інтерфейсами, сервісами для абонентів, можливістю впливу на конфіденційність, цілісність та доступність послуг, маніпулюванням маршрутизацією трафіку, параметрами обслуговування, тарифікацією тощо.

Тому, у базовій конфігурації вважаємо, що існують щонайменше наступні агенти:

- агент розвідки та сканування мережі;
- агент експлуатації вразливостей та початкового доступу;
- агент «бічного» переміщення та ескалації привілеїв;
- агент міжмережевих та сервісно-орієнтованих атак.

Як зазначалося, для кожного агента існує i існує множина дій A_i, а у конкретному стані s_t доступним є лише піднабір $A_i(s_t) \subseteq A_i$, який задовольняє вимоги політик безпеки та логіки графа атак. У запропонованому методі, дії розглядаються як макродії, тобто узагальнені кроки тестування на проникнення, які можуть відповідати одному або кільком викликам інструментів.

У загальному вигляді, дію агента можна подати у формі кортежу $a_t^i = \langle \tau, v, \theta \rangle$, де τ – тип (клас) дії, який відповідає етапу пентесту, v – цільовий вузол або підграф графа атак, до якого застосовується дія, θ – набір параметрів, які конкретизується вже на рівні оркестратора інструментів.

Навчання з підкріпленням працює з дискретним простором дій. Агент обирає тип дії τ і клас цілі v із кінцевої множини, тоді як параметри θ визначаються заздалегідь сформованими профілями та евристиками. Така факторизація дозволяє зменшити розмірність простору дій і зосередити навчання на стратегічному виборі що і де робити, а не як саме викликати конкретний інструмент.

Множина доступних дій $A_i(s_t)$ формується динамічно на основі поточного стану графа атак, політик безпеки та обмежень бюджету.

Агент розвідки та сканування мережі. Основа роль даного агента є розширення відомих вузлів і ребер графа атак, збагачення вузлів атрибутами та ініціалізація моделі мережі. До його простору дій входять активне мережеве сканування, сканування портів та сервісів на конкретному вузлі v, ідентифікація сервісів і протоколів, пасивна розвідка та аналіз трафіку, а також збір конфігураційних та діагностичних даних. Враховуючи граф атак, дії агента розвідки та сканування мережі переважно збільшують множину V_t^{disc} і уточнюють атрибути.

Агент експлуатації вразливостей та початкового доступу. Дії агента спрямовані на підтвердження та експлуатацію вразливостей, виявлених на попередньому етапі, та отримання початкового доступу до різних доменів мережі. До основних класів дій можна віднести: експлуатація відомих вразливостей (CVE) на вузлі v із використанням відповідних експлоїтів, атаки на сервіси керування та адміністративні інтерфейси, експлуатація помилок конфігурації, атаки на облікові дані. Для кожної такої макродії, TelLAG надає оркестратору інформацію про відовідні

вузли та правила атаки, а оркестратор вже обирає конкретний інструмент чи ланцюжок інструментів, що дозволить реалізувати обрану дію.

Агент бічного переміщення та ескалація привілеїв. Працює переважно з уже скомпрометованим вузлами та наявними обліковими даними. До типових класів дій можна віднести переміщення між сегментами через наявні тунелі, проксі, або маршрути, відомі TelLAG; розгортання опорних точок для подальших кроків атаки; ескалація привілеїв на вже скомпрометованому хості та розширення контролю над критичними елементами, що впливають на маршрутизацію, обробку трафіку, керування абонентами та сервісами. У графі атак це відповідає активації нових ребр між доменами й вузлами з вищим рівнем привілеїв.

Агент міжмережевих та сервісно-орієнтованих атак. На даному етапі агентом аналізуються вектори атак, пов'язані з міжмережевими й міжоператорськими інтерфейсами та сервісами абонентів. До його простору дій належать тестові сигнальні запити через міжмережеві/міжоператорські інтерфейси, що дозволяють виявити потенційні маніпуляції з маршрутизацією, встановленням сесій, профілями абонентів; атаки на сервісні платформи, які можуть впливати на параметри обслуговування, тарифікацію, QoS (Quality of Service); маніпуляції маршрутами трафіку та налаштування політик. Такі дії є більш високорівневими з точки зору бізнес- та сервісного впливу і часто несуть більшу вагу у функції винагороди, оскільки демонструють максимального критичні сценарії для замовника.

3.1.4. Функція винагороди

У задачі RL функція винагороди визначає, яку поведінку система вважає успішною та чого намагається досягти політика агентів. Формально її можна відобразити як:

$$R: S \times A_1 \times \ldots \times A_N \times S \to \mathbb{R}, \tag{3.6}$$

де кожному переходу (s_t, a_t, s_{t+1}) у багатоагентному середовищі ставить у відповідність скалярну винагороду r_t. Відомо, що коректний дизайн функції

винагороди є критичним для збіжності й стабільності RL-алгоритмів, особливо в складних доменах [49].

У запропонованому методі ця функція має узгоджуватися з наступними цілями тестування на проникнення телекомунікаційної мережі:

- досягнення й демонстрація критичних сценаріїв атаки на важливі активи мережі;
- максимізація покриття графа атак, виявлення релевантних вразливостей, шляхів атаки та перевірка їх можливості експлуатації;
- мінімізація вартості сценарію;
- дотримання політик безпеки та обмежень середовища тестування;
- підтримка балансу між дослідженням і використанням, щоб уникнути як зациклення на відомих діях, так і хаотичного перебору.

Оскільки, у даному випадку розглядається кооперативна багатоагентна постановка, винагорода r_t є глобальною (спільною) для всіх агентів, тобто кожен агент отримує однаковий скалярний сигнал, який відображає якість спільної дії $a_t = (a_t^1, \dots, a_t^N)$.

Основні компоненти функції винагороди. Враховуючи модель графа атак TelLAG, є необхідним введення кількох допоміжних функцій. Нехай $V^{comp}(s) \subseteq V^{att}$ – множина скомпрометованих вузлів у стані s; $p(v) \geq 0$ – вага ризику вузла v, що заоежить від критичності активу, тактики/техніки/категорії загрози MITRE, оцінок CVSS тощо; $cov(s) \in [0,1]$ – нормалізований показник покриття графа атак у стані s; $viol(s_t, a_t, s_{t+1}) \in \{0,1\}$ – індикатор порушення політик безпеки або виходу за межі дозволених умов тестування; $dup(s_t, a_t) \in [0,1]$ – міра надлишковості дії (наприклад, повторне виконання дії, що не дає нової інформації й не змінює стан).

Тоді, зміну сукупного ризику скомпрометованих активів при переході $s_t \to s_{t+1}$ можна визначити як:

$$\Delta Risk_t = \sum_{v \in V^{comp}(s_{t+1})} p(v) - \sum_{v \in V^{comp}(s_t)} p(v), \tag{3.7}$$

а зміну покриття графа атак як:

$$\Delta Cov_t = cov(s_{t+1}) - cov(s_t). \tag{3.8}$$

На цій основі базову форму винагороди можна записати як:

$$r_t = \alpha_1 \Delta Risk_t + \alpha_2 \Delta Cov_t - \beta_1 cost(a_t) - \beta_2 viol(s_t, a_t, s_{t+1}) - \beta_3 dup(s_{t,}, a_t) + r_t^{term}, \tag{3.9}$$

де $\alpha_1, \alpha_2, \beta_1 \beta_2, \beta_3 \geq 0$ – коефіцієнти, що налаштовуються, а r_t^{term} – додатковий термін у випадку досягнення термінального стану.

У контексті семантики, приріст ризику скомпрометованих активів $\Delta Risk_t$ відображає «якість» досягнутої компрометації. Таким чином, компрометація малокритичного вузла отримує меншу винагороду, ніж коючового елементу ядра або сервісної платформи. Тому агенти заохочуються будувати шляхи до більш критичних цілей, а не лише накопичувати «дрібні» компрометації.

Покриття графа атак ΔCov_t заохочує дії, які розширюють знання про можливі шляхи атаки. Це стосується виявлення нових вузлів, ребер, підтвердження чи спростування гіпотез TelLAG. Це є важливим для процесу тестування на проникнення, адже цінним результатом є повнота аналізу поверхні атаки, а не тільки успішна атака.

Вартість дії $cost(a_t)$ накладає штраф за ресурсозатратні дії та дії в контексті операційних ризиків. Тому, у даному випадку, агенти вчаться шукати більш ефективні послідовності дій, досягаючи тих самих цілей за меншу кількість кроків і з меншим навантаженням.

Порушення політик безпеки $viol$ застосовується у випадку того, коли обрана дія формально дозволена в абстрактному графі атак, але суперечить, накладеним

замовником, обмеженням. У такому випадку, до винагороди додається значний негативний штраф, що дозволяє навчити агентів ігнорувати техічно можливі, але заборонені дії.

Надлишковість дії dup штрафує за повторювані або очевидно марні кроки, які не змінюють стан. Це додатково стимулюватиме агентів уникати локальних мінімумів, де вони постійно повторюють одну й ту саму дію.

Окремо визначається термінальний компонент r_t^{term}, що використовується, коли новий стан s_{t+1} належить до множини термінальних компонентів S_{term}. При цьому, якщо епізод завершується досягненням поставлених цілей пентесту, агентам нараховується додаткова позивтивна винагорода, пропорційна важливості досягнутої мети. Якщо ж епізод завершується через вичерпання бюджету без досягнення цілей, додається незначний негативний штраф. А у випадку, коли епізод припинено через критичне порушення політик або умов тестового середовища, буде застосовуатись суттєвий негативний штраф, щоб агенти навчились уникати таких сценаріїв.

Такий підхід дозволить поєднати локальні сигнали прогресу, через $\Delta Risk_t$ та ΔCov_t із глобальною оцінкою успіху сценарію на рівні всього епізоду.

Формування винагороди для різних етапів тестування на проникнення. Оскільки для цілей пентесту телекомунікаційних мереж було визначено чотири загальні етапи, їх також доцільно врахувати в структурі винагороди:

- Етап 1 – більшу вагу матиме приріст покриття графа атак ΔCov_t;
- Етап 2 – підтвердження можливості експлуатації вразливостей (позитивний $\Delta Risk_t$ для вузлів середньої та високої критичності);
- Етап 3 – досягнення більш привілейованих і стратегічно важливих позицій у мережі (зростання $p(v)$ для нових скомпрометованих вузлів);
- Етап 4 – демонстрація сценаріїв, що впливають на міжмережеві інтерфейси та сервісний рівень (тут, можна брати $p(v)$ з найбільшим значенням і додатковою термінальною винагородою).

Це може бути реалізовано через коефіцієнти, що залежать від етапів, $\alpha_1^{(k)}, \alpha_2^{(k)}$ для етапів $k = 1, \dots, 4$, що налаштовуються в TelLAG/оркестраторі в залежності від

поточного етапу сценарію тестування на проникнення. Таким чином, система буде заохочувати ті види прогресу, які є типовими для конкретної фази тестування.

Формування сигналу для мультиагентної системи. У рамках MARL, усі агенти отримують однаковий глобальний сигнал r_t, адже їх дії взаємопов'язані через спільне середовище та їх метою є спільний успішний сценарій пентесту, а не індивідуальний успіх окремого агента.

Окрім цього, сама структури винагороди може бути факторизована відповідно до ролей агентів:

- дії агента розвідки переважно впливатимуть на ΔCov_t;
- дії агента експлуатації спрямовані на $\Delta Risk_t$ та частково на $cost(a_t)$;
- дії агента lateral movement мають вплив на $\Delta Risk_t$ через доступ до нових доменів і рівнів привілеїв;
- дії агента міжмережевих/сервісних атак впливатимуть на термінальні винагороди та максимально критичні вузли.

Така факторизація можу використовуватися під час реалізації конкретних алгоритмів MARL, не змінюючи при цьому загальної форми функції винагороди.

Таким чином, запропонована структура функції винагороди забезпечуватиме узгодженість із цільовими метриками пентесту телекомунікаційної мережі, наявність більш щільного сигналу, ніж у випадку лише термінальної винагороди, а також можливість гнучкого налаштування під конкретну мережу, типи послуг та обмеження замовника.

3.2. Огляд основних алгоритмів RL

3.2.1. Класифікація алгоритмів навчання з підкріпленням

Всі алгоритми навчання з підкріпленням слід розглядати в багатовимірному просторі. Адже вони мають різні підходи за представлення функцій, різняться наявністю моделі середовища та його властивостями, об'єктом навчання та способом використання досвіду. Аналіз сучасних джерел [50-55] показав, що жодна

одновимірна класифікація не може бути достатньою для коректного вибору алгоритму.

Всі існуючі алгоритми можна поділити на наступні великі групи: за типом простору станів і дій, за наявністю моделі середовища, за політикою вибірки, за кількістю агентів.

Найбільшою групою є за типом простору станів і дій. У першу чергу, розглядаються *методи з обмеженим простором станів і дискретних дій*. Для скінченних MDP з відносно невеликою кількість станів і дискретними діями найбільш вдалим є використання табличних методів, у яких функції цінності $V(s), Q(s,a)$ зберігаються у вигляді таблиць [56]. Тут виділяються три основні класи алгоритмів:

- модельно-орієнтовані табличні методи, де за наявності відомої моделі переходів та, за потреби, функції винагороди, використовуються методи динамічного програмування (DP);
- методи Монте-Карло (MC) – алгоритми оцінюють функції цінності як середнє повернення, отримане з епізодів, повністю завершених під фіксованою політикою, і можуть використовуватися як для оцінки політик, так і для контролю;
- TD (Temporal-Difference) методи – до цього класу відносять TD(0), TD(λ), SARSA, Q-learning, Watkins Q(λ) та інші. Ці алгоритми поєднують ідеї MC та DP. У [52] зазначено, що дані методи є базовими для середовищ із дискретними станами/діями.

Головним недоліком цього класу алгоритмів, що у задачах з великою кількістю станів, табличне представлення стає непридатним.

Методи з великорозмірними станами та дискретними діями. У випадку великого або безперервного простору станів за збереження дискретного простору дій, табличні методи замінюються методами з функціональною апроксимацією, зокрема можна виділити Deep RL [51].

До типових алгоритмів даного класу відносять:

- Deep Q-Network (DQN) – нейронна мережа апроксимує $Q_\theta(s,a)$;

- вдосконалені алгоритми DQN, такі як Double DQN, Dueling DQN, distributional RL (C51, QR-DQN), а також інтегровані архітектури типу Rainbow, які поєднують кілька покращень;
- рекурентні варіанти (DRQN, R2D2) для частково спостережуваних середовищ.

Ці алгоритми класифікують як model-free value-based deep RL. Часто позначаються як алгоритми для дискретних просторів дій, що відповідає їх математичній природі.

Великорозмірні стани, неперервні дії. Для неперервних просторів дій дискретизація дій є неефективною, тому використовуються policy gradient та actor-critic підходи [51]. У цій категорії виділяють наступні алгоритми:

- policy gradient. Сюди належать REINFORCE – базовий стохастичний алгоритм із високою дисперсією градієнта, та Natural Policy Gradient, TRPO, PPO – вводять обмеження на «крок» в просторі політик (PPO став стандартом у багатьох практичних застосуваннях);
- actor-critic. Такі алгоритми поєднують параметризацію політики з оцінкою функції цінності. До стохастичних відносять A2C/A3C, ACER, ACKTR, IMPALA, а до детермінованих/ентропійних для неперервних дій – DDPG, TD3, SAC, D4PG.

З практичної точки зору, DDPG, TD3, SAC використовуються як основні алгоритми для просторів безперервних дій, в той час як PPO, A2C, A3C підтримують як дискретні, так і неперервні дії.

Наступною класифікацією алгоритмів є наявність моделі середовища. Тут виділяються model-based та model-free алгоритми.

Model-based RL. Алгоритми цього класу використовують явну модель переходів і винагороди для планування. Сюди відносять класичні DP-методи, а також сучасні підходи, такі як Dyna-Q, MBPO, Dreamer, AlphaZero/MuZero-подібні алгоритми. Вони можуть бути як табличними, так і глибинними, але спільною рисою для всіх є окреме навчання моделі середовища та/або використання моделі для планування [51, 56].

Model-free RL. У таких алгоритмах агент не будує явної моделі $P(s'|s,a)$, а навчає політику чи функцію цінності безпосередньо з досвіду. До цього класу належить більшість deep RL алгоритмів, що застосовуються на практиці (DQN, TRPO, PPO, A2C/A3C, DDPG, TD3, SAC), і саме вони розглядаються як основний інструментарій для складних середовищ промислового рівня [51, 55].

Іншим виміром класифікації є об'єкт навчання. За типом об'єкту навчання, що саме навчає алгоритм, виділяють такі алгоритми як value-based, policy-based, actor-critic.

Value-based алгоритми. Тут алгоритм навчає лише функцію цінності, тобто якість станів або пар стан-дія, а політика отримується як «жадібна» щодо цієї оцінки. Фактично, алгоритм будує карту, де для кожного стану й дії визначається наскільки це хороше рішення в середньому. Тобто, він вчиться оцінювати, а вже потім робить вибір на основі найвищої оцінки дії. Такий алгоритм використовується коли дії дискретні і коли важливо максимально видобути користь з накопиченого досвіду. Тут виділяють табличні методи (Q-learning, SARSA, Expected-SARSA, n-step Q-learning тощо) та deep-value-based (DQN та його розширення) [52, 56].

Policy-based алгоритми. У таких алгоритмах напряму параметризується політика (наприклад, нейромережа, що повертає розподіл ймовірностей по діях) і оптимізовується функціонал якості політики. Фактично, тут відразу навчається сама поведінка. Алгоритм дивиться, як політика поводиться, які винагороди отримує і параметризує так, щоб у середньому отримувати більше винагороди. Сюди відносять REINFORCE (Monte-Carlo policy gradient) та методи з обмеженням/регуляцією кроку по політиці (TRPO, PPO, PPG, SVPG) [52].

Actor-critic. Тут об'єктом навчання є одночасно і політика, і функція цінності. Актор – це частина, яка приймає рішення, критик – частина, що оцінює наскільки вдалим було рішення актора в тому чи іншому стані. Критик «підказує» актору, куди рухатися, а актор генерує поведінку, з якої критик навчається. Тобто, тут поєднуються ідеї двох попередніх алгоритмів. До представників цього алгоритму відносять A2C/A3C (Advantage Actor-Critic), ACER, ACKTR, IMPALA, deterministic actor-critic

(DPG, DDPG, TD3, D4PG), SAC (Soft Actor-Critic) [52]. Фактично, це є компромісом між стабільністю та ефективністю використання даних.

Наступним класом є за політикою вибірки (за використанням досвіду). Сюди відносяться on-policy/off-policy та online/offline RL.

On-policy. Політика, що навчається, збігається, або майже збігається з політикою, яка генерує досвід. Тут виділяють такі алгоритми як SARSA, REINFORCE, A2C/A3C, TRPO, PPO.

Off-policy. Тут цільова політика може відрізнятися від поведінкової. Алгоритм дозволяє повторно використовувати історичний досвід, що був зібраний іншими політиками. До прикладів належать Q-learning, DQN, DDPG, TD3, SAC, CQL, IQL [51, 54].

Online RL. Навчання в процесі безперервної взаємодії з середовищем. Типовий режим роботи класичних реалізацій DQN, PPO, SAC.

Offline (batch) RL. Навчання на фіксованому наборі траєкторії, без нових взаємодій. Сучасні offline RL алгоритми (CQL, IQL та інші) активно розвиваються у контексті безпечного використання RL у реальних системах [51, 54].

Ще одним класом є кількість агентів. Класична теорія RL фокусується на одноагентних MDP, однак для реальних систем (розподілені мережі, телекомунікаційна інфраструктура, складні кіберфізичні системи) релевантною є саме багатоагентна постановка.

Сучасні огляди MARL [50, 57] пропонують класифікувати алгоритми MARL, спираючись на вже розглянуті сімейства value-based, policy-based, actor-critic). Однак, пропонують також враховувати такі аспекти, як схеми навчання, структури функції цінності та структури винагороди.

Фактично, більшість багатоагентних методів будуються як узагальнення описаних вище одноагентних сімейств із додатковими механізмами координації, факторизації функції цінності та кредитного призначення.

3.2.2. Огляд класів мультиагентних алгоритмів

Перехід до багатоагеного навчання з підкріпленням означає супроводжується переходом від моделі марковського процесу прийняття рішень до марковських ігор, у яких кілька агентів взаємодіють у спільному середовищі та отримують індивідуальні або спільні винагороди. Це, в свою чергу, призводить до специфічних викликів, пов'язаних з нестаціонарністю середовища, з точки зору самого агента, експоненційним зростанням простору станів і дій, частковою спостережуваністю та вимогами до масштабованості обчислень і комунікації між агентами [58-59].

MARL розглядається у багатовимірному просторі ознак, де ключовими осями є:

- структура функцій винагород (кооперативні, конкурентні, змішані задачі);
- організація навчання та виконання (CTE, CTDE, DTDE);
- алгоритмічна парадигма (value-based, policy-based, actor-critic) та спосіб представлення взаємодій між агентами (наприклад, графові моделі) [58].

Структура функцій винагород. У [58] запропоновано класифікувати задачі MARL за типом структури винагород трьома способами: повністю кооперативні задачі, повністю конкурентні та задачі зі змішаними мотивами. У випадку повністю кооперативних задач, усі агенти розділяють спільну глобальну функцію винагороди (наприклад, командні задачі, де метою є максимізація сумарної продуктивності системи). У повністю конкурентних задачах виграш одного агента інтерпретується як програш іншого. Задачі зі змішаними мотивами передбачають, що агенти мають як узгоджені, так і конфлікт інтересів.

Організація навчання та виконання. З точки зору організації доступу до інформації та керування виділяють три основні схеми: Centralized Training and Execution (CTE), Centralized Training, Decentralized Execution (CTDE) та Decentralized Training and Execution (DTDE) [60].

У CTE як навчання, так і виконання здійснюється централізованим контролером, який спостерігає повний стан і обирає дії для всіх агентів. Ця схема має обмежену практичну придатність через погану масштабованість.

CTDE – під час навчання використовується глобальна інформація (спільний стан, дії всіх агентів), проте під час виконання кожен агент застосовує локальну політику на основі власних спостережень. Саме CTDE є домінуючою парадигмою для кооперативного MARL у сучасних дослідженнях.

У випадку DTDE, кожен агент навчається та виконує дії виключно на основі локальної інформації, без централізованих компонент. Такий підхід має мінімальні вимоги до координації, однак сильно страждає від нестаціонарності.

Для поставленої задачі автоматизованого тестування на проникнення кооперативною командою агентів найбільш релевантною є схема CTDE, де у просі навчання можливе використання агрегованого представлення стану мережі та графа атак, тоді як у режимі виконання агенти використовуватимуть локальні політики.

Алгоритмічна парадигма та представлення взаємодій між агентами. Як було згадано раніше, є три основні алгоритмічні парадигми: value-based, policy-based, actor-critic [58-59].

У value-based MARL алгоритми наближають спільну або фактиризовану функцію Q для всіх агентів. Найпростішим прикладом є незалежне Q-навчання (Independent Q-learning), у якому кожен агент навчає власну функцію $Q_i(o_i, a_i)$, розглядаючи інших агентів як частину середовища. У більш розвинених підходах на основі декомпозиції функції цінності (VDN, QMIX, QTRAN, QPLEX), взаємодія між агентами відображається структурою глобальної функції $Q_{tot}(s, \mathfrak{a})$: через адитивні схеми або параметризовані змішані мережі, які агрегують локальні оцінки Q_i у спільну оцінку команди. Таким чином, спосіб представлення взаємодій у даному підході задається саме вибором форми декомпозиціх та обмеженнь Q_{tot} [61-62].

У policy-based (strategy-based) MARL оптимізовується параметризована політика, яка може бути спільною для всієї системи або факторизованою на добуток локальних політик. До цього класу належать незалежні варіанти політики (independent policy gradient, independent PPO), а також централізовано спільні методи, такі як MAPPO, у яких спільний або централізований value-оцінювач використовується для оновлення набору локальних політик у парадигмі CTDE. Взаємодії агентів репрезентуються через спільну ціль оптимізації та через

архітектуру мереж, яка може включати спільні енкодери спостережень, механізми уваги чи комунікації між агентами [58-59].

У actor-based MARL алгоритми об’єднують переваги двох попередніх підходів, використовуючи окремі моделі для політики та функцій цінності. Для цього класу характерною є схема з централізованим або спільним критиком і децентралізованими акторами: COMA, MADDPG, MAAC, MAPPO та їх модифікації. У таких методах критик має доступ до глобальної інформації (стану, дій усіх агентів), що дозволяє явно моделювати взаємодію між агентами у просторі. Політики при цьому залишаються локальними й можуть використовувати лише часткові спостереження. В таких підходах взаємодія між агентами репрезентується через структуру централізованої ціннісної функції критика, яка виступає спільним об’єком навчання та координації [58-59].

У найпростіших випадках, структурне представлення взаємодій між агентами, залишається неявним. Проте, в сучасних дослідженнях вводяться явні структурні припущення: графові моделі, середньопольові (mean-field) наближення, архітектури з навчанням комунікації. У графових, агенти моделюються вершинами графа, а їх взаємодії – ребрами. Для агрегації інформації використовуються графові нейронні мережі й механіхми поширення повідомлень, що підходить для застосування у напрямку мереж та телекомунікацій [59]. У mean-field підходах вплив багатьох агентів апроксимується агрегованими характеристиками, що дозволяє масштабувати алгоритми на великі комплекси однотипних агентів. В алгоритмах з навчанням комунікації (CommNET, DIAL та їх модифікації) взаємодії явно представлені каналами передачі повідомлень між агентами, які самі є об’єктом оптимізації [58, 63].

3.3. Обґрунтування вибору підходу MARL

Виходячи з поставленої задачі, можна виділити наступні ключові характеристики цільового середовища та ролей агентів:

1. Повністю кооперативний характер задачі. Усі агенти працюють на досягнення спільної мети – побудова та виконання максимально

інформативного сценарію атаки для заданої мережі. Тому, в даному випадку, необхідна наявність єдиної глобальної функції винагороди, яка оцінює якість сценарію в цілому.

2. Висока розмірність та часткова спостережуваність. Повний стан системи включає топологію мережі, граф атак, конфігурацію вузлів, поточний прогрес сценарію та історію вже виконаних дій. Окремий агент має доступ лише до фрагмента цієї інформації (локальні спостереження, результати власних дій). Це типова постановка POMDP для кожного агента, а агрегований стан існує лише на рівні системи.
3. Гетерогенність агентів та їх просторів дій. Різні агенти оперують різними класами інструментів і діями різної природи. Простір дій має характер параметризованих дискретних дій: вибір типу інструмента доповнюється числовими або категоріальними параметрами.

Саме ці властивості доцільно використовувати в якості критеріїв для порівняння можливих підходів MARL: здатність працювати в кооперативних задачах з єдиною глобальною винагородою, толерантність до часткової спостережуваності та нестаціонарності, підтримка гетерогенних агентів і параметризованих дій, а також масштабованість до великих графових середовищ.

Незалежне навчання агентів є концептуально простим, однак у кооперативних сценаріях з сильною взаємозалежністю дій є недолік в контексті вираженої нестаціонарності середовища та відсутності механізмів кредитного призначення. Це суттєво ускладнює збіжність і робить результати навчання чутливими до початкових умова і гіперпараметрів, що є критичним недоліком до безпекових застосувань, де потрібна передбачуваність і повторюваність поведінки [64]. Тому такий клас не можна розглядати як основний для запропонованого методу.

Value-based підходи з декомпозицією функції цінності (VDN, QMIX, QTRAN, QPLEX) добре відповідають кооперативним задач, забезпечують явну декомпозицію глобальної функції Q_{tot} на локальні компоненти, та, відповідно, вбудований механізм кредитного призначення [61-62]. Вони демонструють високу ефективність у задачах

з однорідними агентами та дискретними діями і добре узгоджуються з парадигмою CTDE. Однак, для поставленої задачі є суттєві обмеження:

- більшість класичних схем декомпозиції припускають однакові структури політик та просторів дій для всіх агентів, тоді як у запропонованому методі агенти є гетерогенними за ролями та інструментами;
- природна модель дій у тестуванні на проникнення має форму параметризованих дискретних дій, що ускладнює застосування чисто дискретних value-based алгоритмів без додаткового шардування чи дискретизації параметрів;
- обмеження на форму декомпозиції можуть виявитися надто жорсткими для відображення складних взаємозалежностей між ролями агентів у багатокрокових сценаріях атаки [65-66].

Policy-based та actor-critic підходи з централізованим критиком у парадигмі CTDE (COMA, MADDPG, MAPPO та їх модифікації) забезпечують більш високий рівень виразної здатності моделей та природно підтримують гетерогенні агентні архітектури [58-60]. Центральний або спільний критик отримує доступ до глобальної інформації (агрегованого стану мережі, дій усіх агентів, додаткових контекстних ознак тощо), що дозволяє зменшити нестаціонарність, обумовлену зміною політик інших агентів, реалізувати гнучкі схеми кредитного призначення та поєднувати дискретні дії з неперервними або векторними параметрами у єдиній політиці, що важливо для параметризованих дій у пентесті [64, 67].

Також слід відзначити, що експериментальні дослідження показують, що MAPPO-подібні алгоритми з централізованим критиком забезпечують стабільну збіжність та перевершують як незалежні on-policy підходи, так і ряд off-policy методів у стандартних кооперативних бенчмарках при зростанні кількості агентів [67].

Графові та реляційні моделі взаємодії між агентами, які в сучасних роботах розглядаються як надбудова над основними MARL-парадигмами, є природно сумісними саме з actor-critic підходами. У випадку поставленої задачі, тестування на проникнення телекомунікаційних мереж, топологія мережі та граф атак вже задають графому структуру, де вузли відповідають об’єктам тестованої інфраструктури або

станами компрометації, а ребра – можливим крокам атаки. Вбудовування графових нейронних мереж (GNN) у архітектуру критика або спільних енкодерів спостережень дає змогу явно моделювати топологічні взаємозв'язки між компонентами мережі та агентами [68-69].

Таким чином, для реалізації запропонованого методу автоматизованого тестування на проникнення телекомунікаційної мережі доцільно використовувати підхід multi-agent actor-critic у парадигмі CTDE з централізованим критиком та децентралізованими акторами із використанням графового представлення стану мережі та графа атак у архітектурі критика.

Саме у такому випадку буде чітка відповідність кооперативній природі задачі, забезпечується підтримка гетерогенних та параметризованих дій, зменшується нестаціонарність та покращується кредитне призначення, а також забезпечується можливість інтеграції графових моделей.

3.4. Порівняльний аналіз алгоритмів

На основі вибраної парадигми кооперативного багатоагентного actor-ctiric у схемі CTDE з єдиними централізованим критиком постає необхідність визначення конкретного алгоритму для реалізації MARL-модуля.

Серед існуючих, що відповідають задачі, можна розглядати такі як COMA (Counterfactual Multi-Agent Policy Gradients), MADDPG (Multi-Agent Deep Deterministic Policy Gradient), MAPPO (Multi-Agent Proximal Policy Optimization), HAPPO/HATRPO (Heterogeneous-Agent PPO/TRPO).

COMA. Даний алгоритм реалізовує багатоагентний actor-ctiric підхід з централізованим критиком, який оцінює спільну функцію Q та використовує counterfactual baseline для розв'язання задачі кредитного призначення. Така конструкція забезпечує явний механізм оцінювання внеску окремих агентів у глобальну винагороду, що позитивно впливає на стабільність навчання в кооперативних задачах з дискретними діями. Водночас даний COMA є повністю on-

policy алгоритмом без trust-region обмежень, має низьку вибірковість за зразками і обмежену масштабованість за кількістю агентів та розмірністю простору дій [70].

MADDPG. Являється мультиагентним розширенням алгоритму DDPG, у якому кожен агент має власного актора та критика, а критик побудований у централізованій формі та отримує на вхід дії всіх агентів. Алгоритм орієнтований на неперервні або параметризовані простори дій та змішані (кооперативно-конкурентні) задачі. MADDPG має off-policy природу та experience replay, що забезпечує високу вибірковість за зразками, однак у кооперативних задача алгоритм демонструє підвищену чутливість до гіперпараметрів, проблеми стабільності і деградації політик у великих середовищах [71].

MAPPO. Цей алгоритм є багатоагентним варіантом Proximal Policy Optimization (PPO), у якому політики агентів навчаються on-policy, а централізований value-оцінювач використовується у схемі CDTE. У [72-74] демонструється ефективність використання даного алгоритму у різних прикладних доменах (транспортні системи, керування транспортними засобами, енергетичні системи).

MAPPO успадкував всі переваги PPO, стабільність навчання завдяки обмеження кроку оновлення політики (clipped objective, trust-region-подібна поведінка), природно підтримує CTDE та централізованого критика, а також стабільність оновлень та універсальність щодо типів дій (дискретні, неперервні, параметризовані). У той же час, обмежується одноагентним аналізом PPO та в кооперативному багатоагентному контексті розглядається переважно як емпірично ефективний.

HAPPO. Алгоритм відноситься до сімейства Heterogeneous-Agent Reinforcement Learning (HARL) і спеціально розроблений для кооперативних задач з гетерогенними агентами (різні простори спостережень, дій, архітектури політик) [75]. HAPPO підтримує парадигму CTDE з централізованим критиком та PPO-подібним кліпованим об’єктом, розширюючи MAPPO на випадок гетерогенних агентів.

На основі аналізу [75], відзначено, що HAPPO є коректним багатоагентним алгоритмом з доведеними властивостями монотонного покращення спільної політики і збіжності до рівноважних стратегій у кооперативних марковських іграх. При цьому

на практиці демонструючи кращу продуктивність і стабільність, ніж MAPPO та QMIX.

HATRPO. Алгоритм є більш «строгим» trust-region аналогом HAPPO, який безпосередньо адаптує TRPO до кооперативних гетерогенних мультиагентних середовищ. Тут послідовні оновлення політик агентів виконуються в рамках Heterogeneous-Agent Trust Region Learning (HATRL) із явними KL-обмеженнями для кожного агента, що забезпечує доведені теоретичні гарантії монотонного покращення спільної політики та збіжності [75].

Експериментальні результати досліджень показують, що HATRPO та HAPPO разом формують SOTA-клас кооперативних MARL-алгоритмів для гетерогенних агентів, суттєво перевищуючи MAPPO, QMIX та інші сильні базові методи на класичних наборах задач. Також слід відзначити, що нові модифікації HATRPO демонструють значний приріст продуктивності, на 20-25% за фінальною винагородою, за рахунок оптимізації розподілу KL-порогів між агентами, що підкреслює потенціал даного алгоритму як базової trust-region платформи для подальших удосконалень [76].

3.4.1. Критерії порівняння

Для оцінювання придатності наведених алгоритмів до застосування для запропонованого методу автоматизованого тестування на проникнення телекомунікаційної мережі можна виділити наступні критерії:

- відповідність кооперативні постановці із глобальною функцією винагороди (C1);
- підтримка CDTE та структурна критика (C2);
- типи просторів дій (C3);
- стабільність та вибірковість за зразками (C4);
- масштабованість за кількістю агентів і розміром стану (C5);
- механізми кредитного призначення (C6);
- сумісність зі структурними розширеннями (C7).

У контексті відповідності кооперативній постановці із глобальною функцією винагороди, важливим є те, щоб алгоритм був орієнтований на повністю кооперативні задачі, а не початково розроблений для змішаних чи конкурентних ігор.

Щодо підтримки CDTE та структурною критики, алгоритм має використовувати централізованого (спільного) критика з доступом до повного стану та дій усіх агентів, а не набори окремих централізованих критиків. Також тут варто враховувати чи можливою є інтеграція агрегованого представлення графа мережі та атак.

Не менш важливим критерієм є типи просторів дій. Алгоритм має підтримувати дискретні, неперервні і параметризовані дії, шо важливо для моделювання інструментів і їх параметрів у процесі тестування на проникнення.

Критерій стабільності та вибірковості (on-policy/off-policy, trust region властивості) передбачає з'ясування на скільки алгоритм забезпечує стабільне навчання при обмеженому обсязі взаємодії з середовищем та чи має гарантії типу trust region.

Також варто брати до уваги механізми кредитного призначення, чи присутні явні або неявні механізми оцінки внеску окремих агентів у глобальну винагороду.

Оскільки в методі використовуються графові моделі та є необхідною комунікація агентів, то доцільно виділити критерій сумісності зі структурними розширеннями. Наскільки природно алгоритм поєднується з графовими нейронними мережами та механізмами міжагентної комунікації, які є релевантними до цільового середовища телекомунікаційних мереж.

3.4.2. Аналіз MARL алгоритмів

Аналіз обраних алгоритмів, COMA, MADDPG, MAPPO, HAPPO, HATRPO, виконувався на основі зазначених вище критеріїв. Результати порівняльного аналізу наведені в таблиці 3.1. У таблиці:

- «++» – повна відповідність критерію;
- «+» – відповідність критерію з явними обмеженнями;
- «±» – часткова або опосередкована відповідність;

- «-» – невідповідність критерію.

Таблиця 3.1

Порівняльний аналіз MARL алгоритмів

Алгоритм	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**
COMA	++	++	-	±	±	++	+
MADDPG	±	+	++	±	±	±	+
MAPPO	++	++	+	+	+	±	+
HAPPO	++	++	++	++	+	++	++
HATRPO	++	++	++	++	+	++	++

За відповідністю кооперативній постановці з глобальною винагородою та підтримкою CTDE з централізованим структурним критиком, MAPPO, HAPPO та HATRPO задовольняють вимоги запропонованого методу тестування на проникнення, однак останні два додатково мають строго обґрунтовану схему спільного оновлення політик, що безпосередньо оптимізовує глобальний командний критерій.

За типами просторів дій HAPPO і HATRPO продемонстровані на поєднанні дискретних і неперервних середовищ (SMAC, MultiAgent MuJoCo), краще відповідають вимогам гібридного простору дій у поставленій задачі пентесту, ніж класичні реалізації MAPPO, які переважно орієнтуються або на дискретні, або на неперервні дії окремо.

За стабільністю та вибірковістю за зразками HAPPO та, особливо, HATRPO мають перевагу завдяки trust-region властивостям (монотонне покращення спільної політики), демонструючи вищу якість рішень при тому самому або меншому бюджеті взаємодії з середовищем порівняно з MAPPO.

За механізмами кредитного призначення HAPPO/HATRPO явно використовують multi-agetn advantage decomposition lemma, що дає більш коректний розподіл глобальної винагороди між агентами [77].

За сумісністю зі структурними розширеннями, завдяки trust-region у HATRPO, цей алгоритм більш привабливий для інтеграції з графовими нейромережами, які моделюють топологію телекомунікаційних мереж і граф атак, без втрати теоретичних гарантій [50, 77].

Таким чином, для поставленої задачі автоматизованого тестування на проникнення телекомунікаційних мереж з MARL модулем, найбільш оптимальним вибором є алгоритм HATRPO. Адже він найповніше задовольняє вимогам до кооперативності, CTDE, стабільності навчання, механізмів кредитного призначення та інтеграції зі структурними моделями цільового середовища тестування.

ВИСНОВКИ ДО РОЗДІЛУ 3

У Розділі 3 сформовано формальну постановку задачі як багатоагентного навчання з підкріпленням у середовищі, індукованому графом атак: задано множину агентів, простори станів/спостережень/дій, функцію переходів та глобальну функцію винагороди, яку оптимізує команда агентів у межах епізодів взаємодії.

Показано, що реалістичний сценарій пентесту є частково спостережуваним для кожного агента, тому задача природно описується як кооперативний Dec-POMDP і потребує організації «централізоване навчання – децентралізоване виконання» (CTDE), де під час навчання доступна агрегована глобальна інформація, а під час виконання рішення приймаються на основі локальних спостережень.

Запропоновано рольову декомпозицію процесу пентесту на чотири узагальнені етапи та, відповідно, виділено щонайменше чотири спеціалізовані агенти (розвідки/сканування, початкового доступу, бічного переміщення та міжмережевих/сервісно-орієнтованих атак), що відображає багатодоменність і специфіку телекомунікаційних мереж.

Обґрунтовано використання «макродій» як одиниць керування: дія агента подається у вигляді кортежу (тип дії, цільовий вузол/підграф, параметри), а множина допустимих дій формується динамічно з урахуванням поточного стану графа атак,

політик безпеки та бюджетних обмежень; на рівні реалізації дія інтерпретується як запит до оркестратора, який підбирає конкретні інструменти виконання.

Розроблено вимоги та структуру функції винагороди для кооперативної команди, що поєднує: (i) прогрес за «якістю» компрометації через приріст ризику скомпрометованих активів, (ii) стимулювання повноти аналізу через покриття графа атак, (iii) штрафування ресурсоємності дій, а також (iv) санкції за порушення політик, надлишкові/повторні кроки та умови завершення епізоду (термінальний компонент).

Систематизовано простір MARL-підходів за ключовими осями (структура винагороди, організація навчання/виконання, алгоритмічна парадигма та спосіб подання взаємодій), що дозволило узгодити властивості цільового середовища (кооперативність, гетерогенність агентів, часткова спостережуваність) із вимогами до алгоритму.

Обґрунтовано доцільність реалізації MARL-модуля у вигляді multi-agent actor-critic у парадигмі CTDE з централізованим критиком і графовим поданням стану мережі/графа атак у критикові як бази для координації гетерогенних агентів.

Виконано порівняльний аналіз придатних MARL-алгоритмів (COMA, MADDPG, MAPPO, HAPPO, HATRPO) за системою критеріїв та показано, що найбільш повну відповідність вимогам (CTDE, стабільність, коректне кредитне призначення, сумісність зі структурними моделями) забезпечує HATRPO, який і визначено як оптимальний вибір для подальшої реалізації.

Таким чином, у Розділі 3 сформовано теоретико-методичний базис MARL-складової запропонованого методу (постановка, декомпозиція на агентів, простір дій, функція винагороди та обґрунтований вибір алгоритму).

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО МЕТОДУ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

Практична реалізація запропонованого методу автоматизованого тестування на проникнення є ключовим етапом, що апробує теорію на реальних умовах функціонування телекомунікаційних мереж. Реалізація концепція у вигляді програмного забезпечення вимагає перетворення абстрактних моделей на конкретну архітектуру, програмні модулі та інженерні рішення, здатні працювати в умовах реальної мережевої інфраструктури оператора зв'язку.

Особливістю телекомунікаційного домену, зокрема стільникових мереж зв'язку п'ятого покоління, є поєднання високої складності топології, віртуалізованих мережевих функцій, протоколів сигналізації та суворих вимог до безпеки й доступності сервісів. Відповідно, прототип програмного забезпечення має не лише реалізувати логіку побудови графа атак, роботу динамічного графа знань, інтелектуального оркестратора та оркестратора інструментів, але й забезпечувати модульність, масштабованість та керованість.

4.1. Загальна архітектура програмного забезпечення

Проєктування архітектури програмного забезпечення є основним кроком від концептуальної моделі запропонованого методу до його практичної реалізації. На цьому рівні абстрактні компоненти методу набувають вигляду взаємопов'язаних програмних компонентів із чітко визначеними інтерфейсами, потоками даних і відповідністю.

Від розробленої архітектурної структури залежить не лише працездатність прототипу, але й можливість подальшої еволюції системи, її адаптації до різних конфігурацій телекомунікаційних мереж та інтеграції з новими засобами пентесту.

4.1.1. Вимоги до програмної реалізації

На основі концепції та формальної моделі методу автоматизованого тестування на проникнення, можливо сформулювати ряд вимог до програмного забезпечення, що реалізує даний метод. Вимоги згруповано на функціональні, що визначають необхідні можливості системи, та нефункціональні, що задають обмеження щодо якості, продуктивності й експлуатаційних характеристик (рис. 4.1).

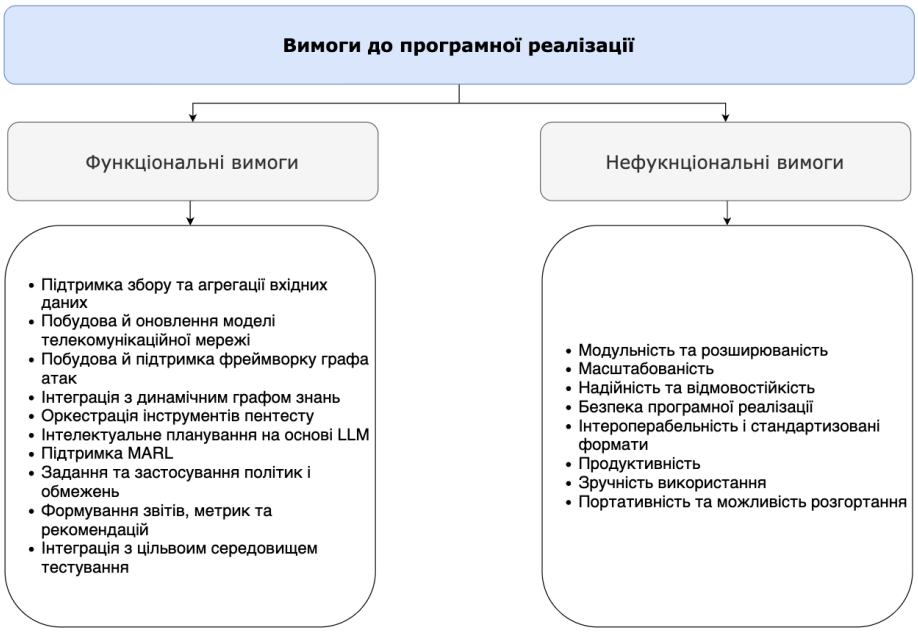

Рис. 4.1. Вимоги до реалізації

До функціональних вимог відносяться:

1. *Підтримка збору та агрегації вхідних даних.* Програмне забезпечення має забезпечувати автоматизований збір і попередню обробку даних про цільову телекомунікаційну мережу з різних джерел: результати сканерів вразливостей, мережевої розвідки та топологічного аналізу; конфігураційні файли мережевого обладнання та віртуалізованих функцій; журнали подій та систем моніторингу; база вразливостей (CVE/NVD тощо) та інші довідкові джерела. Для цього необхідний ETL-модуль, який трансформує «сирі» дані у внутрішнє формалізоване подання.
2. *Побудова й оновлення моделі телекомунікаційної мережі.* Рішення має будувати формальну модель мережі у вигляді множини вузлів, зв’язків та

атрибутів (тип елемента, критичність, протоколи, параметри безпеки) на основі агрегованих вхідних даних, а також підтримувати її актуальність при надходження нової інформації.

3. *Побудова й підтримка фреймворку графа атак.* Програмне забезпечення має реалізовувати удосконалений фреймворку TelLAG, який формуватиме логіко-орієнтований граф атак на основі фактів про конфігурацію й вразливості; підтримуватиме доменно-специфічні правила; забезпечуватиме оновлення графа атак при зміні моделі мережі та множини вразливостей; надаватиме API для доступу до вершин, ребер та їх атрибутів зовнішнім модулям.
4. *Інтеграція з динамічним графом знань.* Реалізація має забезпечувати синхронізацію TelLAG із динамічним графом знань, у якому відображатимуться мережеві елементи, сервіси, вразливості та стани системи; онтологія загроз з урахуванням MITRE ATT&CK, FiGHT; історичні дані попередніх тестувань і контексті бізнес-сервісів. Тут необхідне двостороннє відображення, від результатів аналізу до графа знань і навпаки.
5. *Оркестрація інструментів пентесту.* Рішення має забезпечувати реєстрацію та опис підтримуваних інструментів (категорія, параметри запуску, формати вихідних даних); перетворення абстрактних дій на конкретні інструменти (CLI/API); запуск, моніторинг виконання та отримання результатів у стандартизованому форматі; передачу результатів у ETL-модуль для подальшої інтерпретації й оновлення графа атак.
6. *Інтелектуальне планування на основі LLM.* Система повинна інтегрувати LLM-модуль, який формуватиме високорівневі сценарії атак та цілі тестування, спираючись на граф атак, граф знань і політики; трансформуватиме результати роботи інструментів у семантично інтерпретовані факти (нові вразливості, підтверджені шляхи атак); зможе обмежувати простір сценаріїв відповідно до заданих політик і операційних обмежень.
7. *Підтримка MARL.* Програмне забезпечення має реалізовувати MARL-модуль, який представлятиме процес пентесту як середовище MARL із визначеними просторами станів, дій та винагород; підтримуватиме множину ролей агентів і

їх кооперацію у постановці CTDE; інтегруватиметься з TelLAG як базовою моделлю середовища; забезпечуватиме цикл навчання, збереження й завантаження політик, а також виконання цих політик під час процесу тестування на проникнення.

8. *Задання та застосування політик і обмежень.* Це має бути модуль політик, що дозволить описувати допустимі й заборонені типи дій, часові вікна проведення тестів, обмеження за навантаженням; задавати винятки для окремих вузлів чи сегментів, а також автоматично перевіряти узгодженість планів тестування й дій агентів із цими політиками.
9. *Формування звітів, метрик та рекомендацій.* рішення має формувати структуровані звіти про проведене тестування з описом виявлених ланцюжків, використаних вразливостей та досягнутих цілей, обчислювати метрики (покриття графа атак, «вартість» атаки, ризик для критичних сервісів тощо); генерувати рекомендації щодо усунення вразливостей, прив’язуючи їх до елементів мережі та тактики/технік MITRE ATT&CK/FiGHT.
10. *Інтеграція з цільовим середовищем тестування.* Програмне забезпечення має повністю підтримувати взаємодію з середовищем телекомунікаційних систем і мереж, включаючи отримання даних про стан мережі, запуском інструментів, що оперують з трафіком в мережі, безпечним режимом роботи, який включає неконтрольований вплив на критичні компоненти мережі.

Доцільно також виділити наступні нефункціональні вимоги до програмного забезпечення:

1. *Модульність та розширюваність.* Архітектура програмного забезпечення має бути модульною, з чітко визначеними інтерфейсами між компонентами (ETL, TelLAG, граф знань, LLM, MARL, оркестратор інструментів і т.д.). Це повинно спрощувати додавання інструментів пентесту, розширення доменно-специфічних правил TelLAG, інтеграцію альтернативних LLM або RL фреймворків.
2. *Масштабованість.* Розроблена система має коректно працювати для мереж з великою кількістю вузлів, сервісів і вразливостей, у тому числі для

топологій, наближених до операторських. Це вимагає: ефективних алгоритмів побудови та зберігання графа атак; можливості асинхронного запуску інструментів та підтримки паралельного навчання й виконання MARL-агентів.

3. *Надійність та відмовостійкість.* Необхідно забезпечити збереження проміжних результатів (станів графа, політик агентів, журналів виконання), коректне завершення або відкат операцій при збої інструментів чи мережевих компонентів, а також можливість повторного запуску сценаріїв з фіксацією умов тестування.
4. *Безпека програмної реалізації.* Система має відповідати вимогам безпечного програмування: ізоляція процесів, що виконують потенційно шкідливі дії, безпечне зберігання облікових даних та конфіденційної інформації, обмеження прав доступу до цільової мережі відповідно до політик та шифрування каналів взаємодії між компонентами (за потреби).
5. *Інтероперабельність і стандартизовані формати.* Програмне забезпечення має використовувати відкриті формати даних (JSON, YAML, CSV, формати звітів сканерів і т.д.), а також надавати зовнішній API (наприклад, REST) для інтеграції з іншими системами моніторингу, управління інцидентами чи візуалізації.
6. *Продуктивність.* Час побудови графа атак, узгодження сценаріїв та запуску інструментів має бути прийнятним для використання в цільовому середовищі. Також має передбачатися можливість оптимізації, наприклад, кешування результатів, використання попередньо згенерованих графів тощо.
7. *Зручність використання.* Інтерфейс користувача повинен забезпечувати конфігурацію кампаній тестування; перегляд стану графа атак і ходу виконання сценаріїв, доступ до звітів та основних метрик без необхідності роботи безпосередньо з низькорівневими артефактами (логами інструментів, внутрішніми структурами даних тощо).

8. *Портативність та можливість розгортання.* Програмна реалізація має бути придатною до розгортання на типових програмно-апаратних платформах, бажано з використанням контейнеризації, що спростить загальну експлуатацію.

Сформульовані вимоги визначають рамки, в яких здійснюватиметься проєктування архітектури програмного рішення та реалізація його окремих модулів.

4.1.2. Архітектура програмного забезпечення

У загальному вигляді архітектура програмного забезпечення, Montera, побудована за класичним трирівневим клієнт-серверним принципом (рис. 4.2) і включає три взаємопов'язані складові:

- рівень користувача (User);
- клієнтський веб-інтерфейс (FRONTEND, Web UI);
- серверна частина (BACKEND).

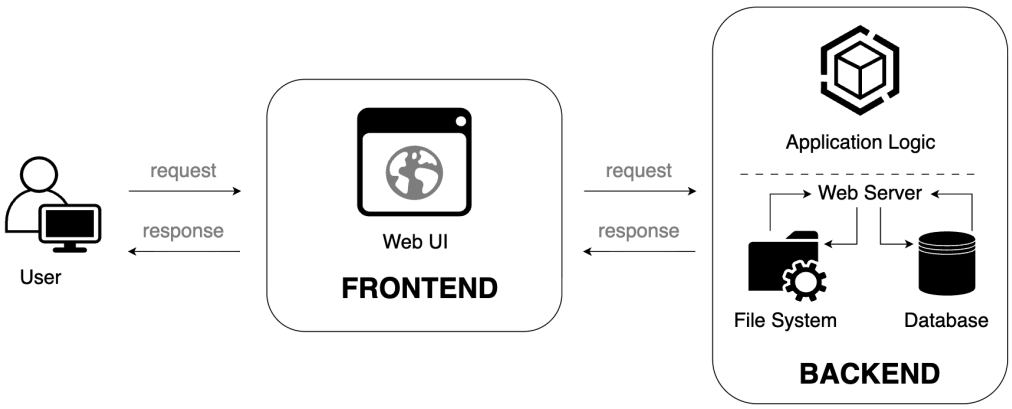

Рис. 4.2. Загальна архітектура

Рівень користувача показує робоче місце фахівця з кібербезпеки, який взаємодіє з системою за допомогою стандартного веб-браузера. На цьому рівні формуються бізнес-запити, що далі передаються до клієнтського застосунку у вигляді HTTP-запитів.

Frontend (Web UI) реалізовує презентаційний рівень застосунку. Це багатосторінковий веб-застосунок, відповідальний за візуалізацію дашбордів,

мережевих топологій, графів атак, звітів тощо. Фронтенд комунікує зі серверною частиною виключно через REST-інтерфейс, надсилаючи запити у форматі HTTPS/JSON та отримуючи структурні відповіді. Така декомпозиція забезпечує незалужену еволюцію інтерфейсу користувача й серверної логіки та полегшує інтеграцію альтернативних клієнтів.

Backend являється центральною частиною архітектури та інкапсулює всю прикладу логіку додатку. Серверна частина умовно поділена на веб-сервер, що обробляє HTTP-запити та підсистеми зберігання даних (файлова система та база даних). У файловій системі зберігаються всі програмні та конфігураційні компоненти додатку, а база даних забезпечує постійну пам’ять для результатів тестування, топологій і т.д.

Така трирівнева архітектура задає загальну схему розподілу відповідальностей між компонентами системи. Детальна архітектура розробленого веб-додатку показана на рисунку 4.3, що також демонструє всі внутрішні зв’язки між окремими його модулями.

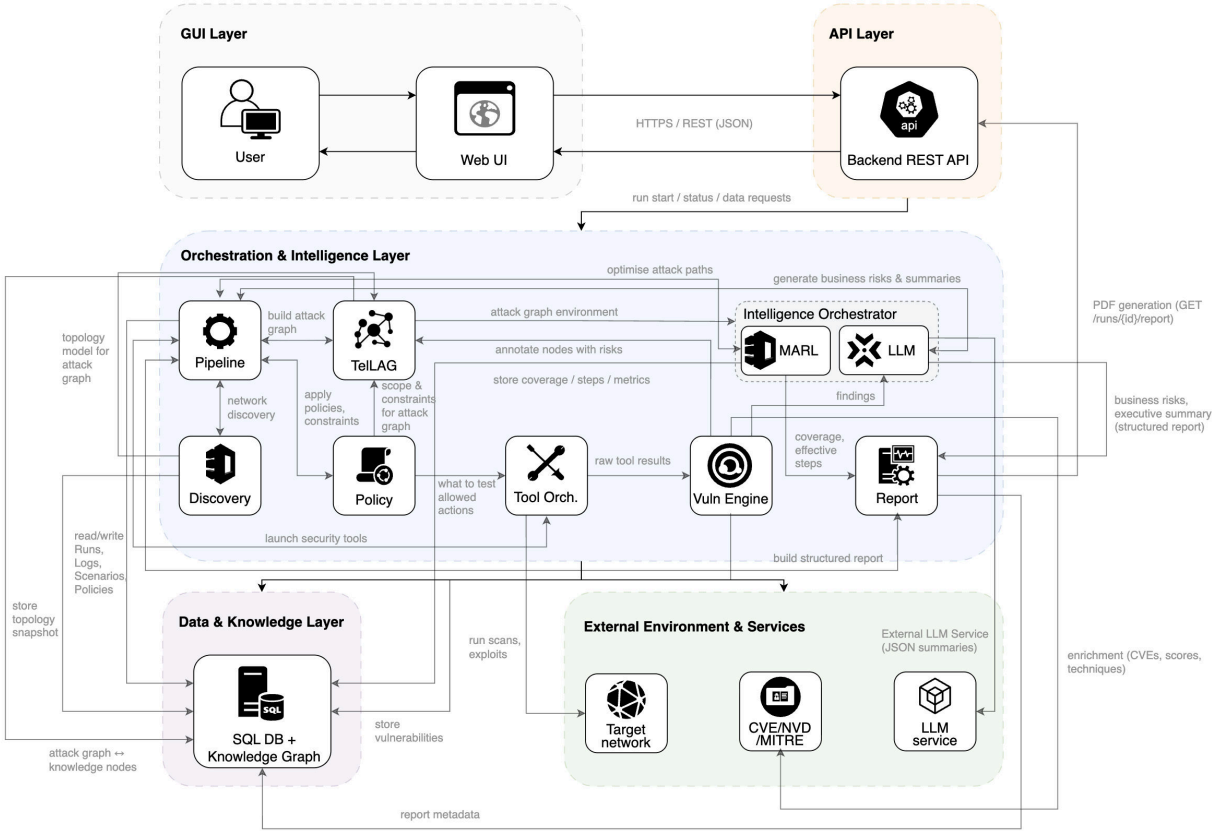

Рис. 4.3. Деталізована архітектура

Архітектура логічно поділяється на графічний інтерфейс користувача (GUI Layer), API-шар, шар оркестрації та інтелектуальної обробки (Orchestration & Intelligence Layer), шар даних та знань (Data & Knowledge Layer), а також контур зовнішнього середовища та сервісів (External Environment & Services).

GUI Layer. Графічний інтерфейс користувача реалізовано у вигляді веб-додатку, який надає доступ до основних сценаріїв використання: запуск автоматизованих тестувань на проникнення, перегляду топології, графа атак, метрик MARL-агентів, політик, журналів запусків і згенерованих звітів. Користувач взаємодіє виключно з Web UI, який формує HTTP(S)-запити до бекенда, отримує від нього JSON-відповіді та візуалізує їх у вигляді дашбордів, таблиць і графів. Таким чином, GUI шар виконує роль тонкого клієнта, а вся бізнес-логіка додатку зосереджена на серверній стороні.

API Layer. API-шар представлено блоком Backend REST API, який інкапсулює серверну логіку та надає фронтенду єдиний формат взаємодії. Через відповідні кінцеві точки REST реалізовуються операції створення та запуску сценаріїв, керування політиками тестування, доступу до топології, графа атак фреймворку TelLAG, вразливостей, MARL-метрик і PDF-звітів. Цей шар відповідає за автентифікацію, валідацію запитів, узгоджених форматів даних і перетворює зовнішні запити Web UI на виклики внутрішніх модулів оркестрації.

Orchestration & Intelligence Layer. Центральним елементом всієї архітектури є шар оркестрації та інтелектуальної обробки, що реалізовує формалізований метод тестування на проникнення, побудови графа атак тощо.

Модуль Pipeline виконує роль керуючого процесу (workflow engine) для одного запуску методу. Він послідовно ініціює мережеву сканування, побудову топології, формування графа атак TelLAG, запуск інструментів пентесту, кореляцію вразливостей, запуск MARL-оптимізації та формування звіту.

Модуль Discovery реалізовує етап мережевої розвідки. На основі конфігурації сценарії або політики виконується сканування цільової мережі, виявляються хости, відкриті порти та базові атрибути вузлів. Отримана множина хостів є вхідними даними для побудови топології та графа атак.

Модуль TelLAG відповідає за побудову логічного графа атак на основі мережевої моделі, зовнішніх таксономій та політик. На цьому етапі мережеві вузли та сервіси відображаються у множину станів графа атак, а потенційні дії зловмисника у множину ребер із відповідними атрибутами. Саме TelLAG формує середовище, у якому далі працюють MARL-агенти.

Модуль Policy реалізовує підсистему політик та обмежень. Політики задають дозволені дії в графі атак (правила доступу до сегментів мережі, заборонені операції, часові вікна тестування тощо) і застосовуються як під час побудови TelLAG, так і при запуску інструментів. Тим самим забезпечується відповідність тестування вимогам безпеки замовника.

Модуль Tool Orchestrator трансформує результати планування атак у конкретні виклики інструментів пентесту. На основі сценарію, політик і результатів роботи MARL-агентів він визначає, які інструменти запустити для конкретних вузлів/сервісів, передає їм параметри, а потім агрегує «сирі» результати сканування та експлуатації.

Модуль Vulnerability Engine виконує семантичну обробку результатів роботи інструментів: нормалізацію, кореляцію з зовнішніми базами CVE/NVD, таксономіями MITRE ATT&CK та FiGHT, обчислення й збереження показників CVSS, а також побудову вузлів/ребер графа знань. На цьому рівні формується однорідне уявлення про знайдені вразливості, яке згодом використовують MARL-агенти та підсистема звітності.

Intelligence Orchestrator (MARL+LLM) об’єднує модуль багатоагентного навчання з підкріпленням та модуль великої мовної моделі. Підсистема MARL працює в середовищі, породженому TelLAG, та на основі функції винагороди, яка враховує критичність вразливостей і політики, знаходить ефективні послідовності дій атакувальника та формує метрики покриття графа атак. Підсистема LLM, інтегрована через зовнішній сервіс, використовується для інтерпретації технічної інформації про проведене тестування у бізнес-орієнтовані ризики та для формування загального підсумку тестування у звіті. При цьому, LLM також використовується для генерації сценаріїв тестування та високорівнених задач для MARL.

Модуль Report формує структурований вихідний звіт про проведене тестування на проникнення телекомунікаційної мережі. Він об’єднує дані з бази (контекст запуску, топологія, вразливості, метрики MARL), семантичну інформацію з графа знань і текстові пояснення згенеровані LLM. На основі цих даних будується модель звіту, яка далі перетворюється у PDF-файл, що є доступним для завантаження.

У межах цього шару всі модулі взаємодіють як через безпосередній виклик функцій, так і через спільну схему даних у SQL-базі, що виступає інтеграційною шиною.

Data & Knowledge Layer. Шар даних і знань представлено єдиною SQL-базою, доповненою структурою графа знань. У ній зберігаються сутності запуску методу, сценарії, політики, журнали, топологічні знімки, граф атак TelLAG, знайдені вразливості та їх кореляція з таксономіями MITRE, а також вузли та ребра графа знань. Таким чином, цей шар реалізовує формальні множини D_{ext}, D_{int}, G, K, описані у формальній моделі, і забезпечує довготривале зберігання, відтворюваність і подальшу аналітику.

External Environment & Services. Даний контур зовнішнього середовища включає цільову мережу, по якій виконується тестування, зовнішні бази вразливостей та таксономії (CVE/NVD/MITRE, FiGHT), а також LLM-сервіс. Модуль Discovery взаємодіє з Target Network під час мережевої розвідки. Tool Orchestrator використовує LLM-сервіс для утворення бізнес-ризиків та управлінських резюме. Зв’язок із зовнішнім сервісами є асинхронним і конфігурованим, що дозволяє адаптувати платформу до інших джерел даних та моделей.

У сукупності описані компоненти забезпечують повну програмну реалізацію методу, від побудови моделі мережі та графа атак до оптимізації ланцюжків атак, кореляції з базами знань і генерації зрозумілих для бізнес-сторони звітів. Деталізована архітектура показує, як кожна теоретична підсистема методу відображається в окремому модулі або групі модулів програмного комплексу Montera та через які інтерфейси здійснюється їх взаємодія.

4.1.3. Вибір технологій та засобів реалізації

Реалізація програмного забезпечення Montera повинна забезпечувати не лише всі функціональні вимоги, формально описаного методу, а й відповідати нефункціональним вимогам. Таким як модульність, масштабованість, портативність та можливість подальшого розширення. З огляду на це, було обрано сучасний технологічний стек, з відкритим вихідним кодом (open-source), орієнтований на веб-платформи з контейнеризованим розгортанням (рис. 4.4).

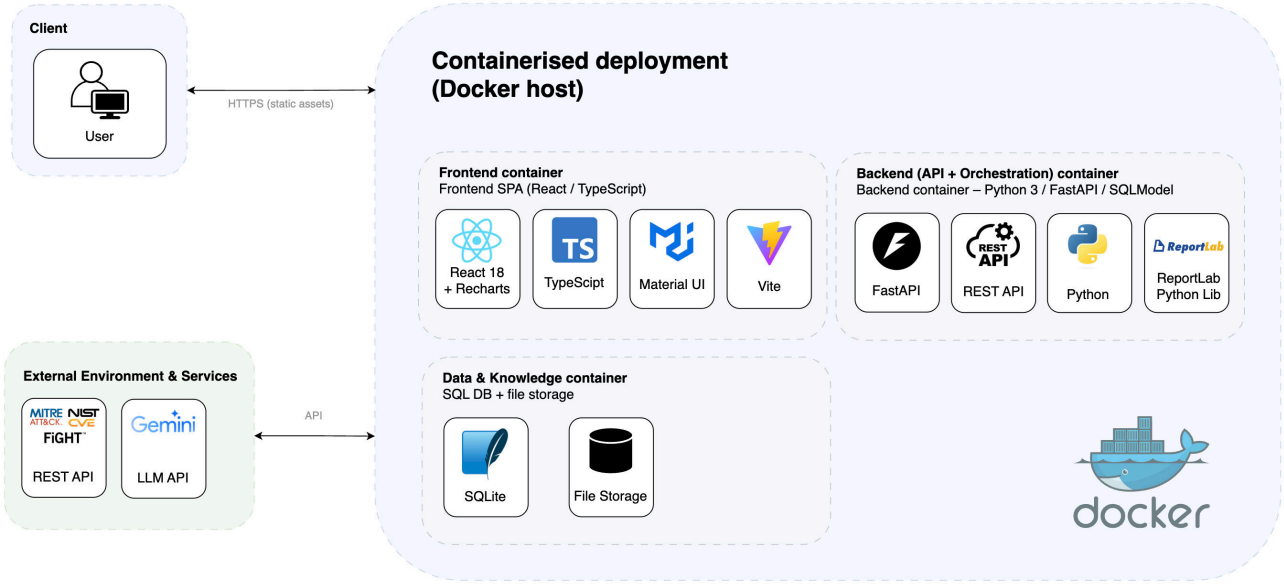

Рис. 4.4. Технологічний стек програмної реалізації

На рівні бекенда використано мову програмування Python з фреймворком FastAPI та ORM-бібліотекою SQLModel. Саме така комбінація забезпечить високий рівень виразності для реалізації складної логіки TelLAG, MARL та обробки звітів, а також дає змогу будувати типобезпечні моделі даних, автоматично формувати OpenAPI-специфікацію REST-інтерфейсів і працювати в асинхронному режимі. Для генерації PDF-документів використано бібліотеку ReportLab, що підтримує програмне формування структурованих звітів із динамічними секціями, графами та таблицями.

Фронтенд реалізовано як багатосторінковий веб-додаток на базі React з використанням TypeScript і збирача Vite. Такий вибір дає змогу відокремити

презентаційний шар від бізнес-логіки, реалізувати інтерактивні інтерфейси (дашборди, графи топології та атак, візуалізацію всіх метрик тощо) та забезпечити статичну типізацію на боці клієнта. Для побудови інтерфейсу використано бібліотеку компонентів MUI (Material UI), а для візуалізації часових рядів та метрик – бібліотеку Recharts. Це дозволяє швидко створювати консистентний інтерфейс у стилі інформаційної панелі без необхідності розробляти базові компоненти з нуля.

Підсистема зберігання даних реалізована на основі реляційної СУБД класу SQL, у MVP конфігурації SQLite, для рівня готового продукту передбачено можливість використання PostgreSQL. SQL-підхід забезпечує транзакційність, підтримку складних запитів для вибірки запусків, вразливостей, вузлів графів, а також є стандартом для інтеграції з іншими системами підприємства. Моделі програмного рішення описуються у вигляді типізованих класів SQLModel, що одночасно виконують роль ORM-сутностей та Pydantic-схем для API.

Окрему групу технологій становлять засоби інтеграції з зовнішніми сервісами. Для отримання бізнес-орієнтованих звітів, формування високорівневих задач/планів для MARL, використовується Gemini API (через клієнтську бібліотеку google.genai), що дозволяє вбудувати LLM-модуль у інтелектуальний оркестратор. Також за допомогою API, інтегровано з джерелами вразливостей типу CVE/NVD/MITRE, які використовуються модулем Vulnerability Engine для збагачення знайдених вразливостей показниками CVSS і таксономіями атак.

З точки зору розгортання, обрано підхід до контейнеризації на основі Docker, що дозволяє упаковувати фронтенд, бекенд та СУБД у відокремлені контейнери з чітко визначеними залежностями. Такий підхід спрощує перенесення системи між серверним середовищем тестування рішення та потенційним продуктивним середовищем, а також відповідає вимозі портативності та можливості швидкого відтворення тестувань.

Узагальнений перелік основних технологій та засобів реалізації наведено у таблиці 4.1.

Таблиця 4.1

Технологічний стек

Підсистема / завдання	**Технологія**	**Роль у системі та обґрунтування вибору**
Реалізація бекенд логіки, REST API	Python 3, FastAPI	Python забезпечує високу швидкість прототипування та багату екосистему для мережевих досліджень і ML. FastAPI надає типізований, асинхронний REST-фреймворк з автоматичною генерацією OpenAPI та високою продуктивністю, що важливо для оркестрації процесів.
Моделювання сутностей БД та API-схем	SQLModel (Pydantic + SQLAlchemy)	SQLModel поєднує декларативні моделі БД, що дозволяє єдиним чином описувати основні компоненти додатку. Забезпечує типобезпечність, валідацію та зменшує кількість дубльованого коду.
Зберігання структурованих даних	SQL-СУБД SQLite (PostgreSQL)	Реляційна модель даних дозволяє зручно зберігати журнали запусків, результати сканувань, вули графів. SQLite використано для розробки прототипу, для масштабованого варіанту в продуктивному середовищі заплановано використання PostgreSQL
Герерація PDF-звітів	Бібліотека ReportLab	Бібліотека надає низькорівневий контроль над розміткою сторінки, що дозволяє програмно формувати структуровані звіти з динамічними секціями, таблицями, діаграмами та брендингом Montera без залежності від зовнішніх офісних пакетів

Продовження таблиці 4.1

Реалізація MARL-середовища над TelLAG	Власна імплементація AttackGraphEnv на Python	Дозволяє гнучко задавати стан, простір дій та винагороди для агентів, адаптований до структури графа TelLAG. Обрана власна реалізація спрощує інтеграцію з доменними моделями та дає можливість подальшого підключення сторонніх MARL-бібліотек
Веб-інтерфейс, дашборди, інтерактивна візуалізація	React+TypeScript	React забезпечує компонентний підхід і ефективний рендеринг інтерфейсу, що є критичним для динамічних дашбордів. TypeScript додає статичну типізацію, зменшуючи кількість помилок при взаємодії з API та складними структурами даних.
Побудова та стилізація UI-компонентів	MUI (Material UI)	Надає набір готових, доступних компонентів (панелі, таблиці, діалоги), що прискорює розробку і забезпечує єдиний візуальний стиль. Легко налаштовується під власне брендування
Візуалізація часових рядів і метрик	Recharts	Декларативна бібліотека для побудов графіків у React. Підтримує респонсивний дизайн і інтегрується з існуючою компонентною системою
Збірка та розгортання фронтенду	Vite	Сучасний білдер з високою швидкістю для девсерверу та оптимізованими білдами. Забезпечує зручний DX при розробці складних SPA та інтеграцію з TypeScript/React без зайвої конфігурації

Закінчення таблиці 4.1

Інтеграція з LLM	Gemini API (google.genai)	Завдяки режиму JSON-відповідей легко інтегрується з іншими компонентами та забезпечує відтворювану структуру результатів
Збагачення вразливостей даними CVE/NVD/MITRE	Зовнішні REST-сервіси CVE/NVD/MITRE	Використовуються для отримання формалізованих описів вразливостей, CVSS-оцінок та таксономій атак. Це дозволяє Vulnerability Engine відповідати сучасним практикам керування вразливостями та забезпечувати сумісність зі стандартами галузі
Контейнеризація та розгортання	Docker	Ізолює компоненти у незалежні контейнери, що полегшує розгортання на власних серверах або у хмарній інфраструктурі. Відповідає вимогам портативності та відтворюваності

4.2. Реалізація модулів

Реалізація програмного забезпечення Montera спирається на класичний принцип багаторівневої архітектури, у якій чітко розмежовано відображення, прикладу логіку та доступ до даних. Такий підхід рекомендовано більшістю сучасних робіт з програмної інженерії, оскільки він підвищує керованість складністю, повторне використання компонентів та можливість їх незалежної еволюції.

Архітектура Montera логічно поділяється на фронтенд підсистему (каталог pt-frontend/src), серверну частину (каталог pt-backend/app) та шар зберігання даних. У такий конфігурації фронтенд відповідає за інтерфейс користувача та взаємодію з REST-інтерфейсами, бекенд інкапсулює доменну логіку пентесту, а шар даних забезпечується постійність запусків, топологій, графів атак тощо.

На рівні клієнтської частини використано компонентний підхід, типовий для веб-застосунків. Структура директорії src відображає розподіл відповідальностей:

- піддиректорія api містить типізовані клієнти доступу до бекенд-ресурів (runs.ts, stats.ts, tellag.ts тощо), що реалізовують єдиний спосіб виклику кінцевих точок REST;
- піддиректорія components акумулє повторно використовувані елементи інтерфейсу (StatCard, RunLiveLogs і т.д.), які інкапсулюють локальний стан та візуальне оформлення;
- піддиректорія layout (MainLayout, Sidebar, TopBar) задає каркас додатку та організовує навігацію;
- піддиректорія pages (наприклад, DashboardPage, NetworkTopologyPage, тощо) реалізовує окремі сторінки предметної області та пов’язує візуальні компоненти з відповідними викликами API.

Серверна частина реалізована у вигляді модульованого Python-додатку з чітким розділенням шарів API, доменної логіки та інфраструктури. Директорія app/api/routes містить декларативні описи REST-ендпоінтів (routes_runs.py, routes_tellag.py, routes_topology.py та інші), кожен з яких відповідає за один логічний ресурс. Ці модулі не містять безпосередньо алгоритмів тестування на проникнення, а лише делегують обробку у відповідні сервіси та оркестратори, що відповідає принципу розділення відповідальностей.

Доменні сутності представлені у пакеті app/models, де для кожної категорії даних (запуски, сценарії, політики, вузли топології тощо) визначено окремі модулі з ORM-моделями (RunDB, ScenarioDB та інші) та допоміжними об’єктами передачі даних. Такий підхід дозволяє явно фіксувати структурні обмеження предметної області, забезпечуючи цілісність даних на рівні схеми.

Найбільш складна функціональність зосереджена у пакеті app/orchestrator, який реалізовує інтелектуальний оркестратор та пов’язані з ним сервіси. Підпакет intel інкапсудює компоненти MARL середовища та агентів (marl_env.py, marl_agent.py), адаптер TelLAG та клієнт LLM-сервісу, тоді як підпакет tools відповідає за реєстр, маппінг і запуск зовнішніх інструментів тестування на проникнення. Оркестраційний

модуль pipeline.py описує послідовність кроків пентесту як єдину керовану процедру, що забезпечує узгодженість між підсистемами.

Окремий блок утворюють сервіси роботи зі знаннями, app/knowledge, модулі побудови та обробки TelLAG (app/tellag), компоненти поглибленого аналізу вразливостей (app/vuln) та генерації звітів (app/reports/builder.py, pdf_renderer.py). Вони використовують єдиний шар доступу до даних app/db/session.py, init_db.py, що спрощує керування транзакціями та конфігурацією підключення.

Таким чином, організація коду Montera реалізовує поєднання шарової та сервісно-орієнтованої архітектурних схем. Горизонтальний розподіл мінімізовує вертикальні залежності між підсистемами, тоді як внутрішня декомпозиція на сервіси дозволяє локалізувати зміни в межах окремих модулів без порушення загального каркасу системи. Це створює підґрунтя для подальшого масштабування платформи, розширення переліку підтримуваних інструментів та інтеграції з новими зовнішніми джерелами знань без фундаментальної перебудови ядра додатку.

4.2.1. Підсистема даних та графа знань

Підсистема даних та графа знань у Montera виконує роль опорного рівня всієї програмної архітектури. Вона забезпечує стале зберігання результатів пентестів, топологічної інформації, виявлених вразливостей та похідних знань, а також уніфікований інтерфейс доступу до цих даних для оркестратора, модулів аналізу та звітності. Фактично саме тут фіксується «цифровий слід» кожного запуску методу.

Базовим компонентом підсистеми є реляційна база даних, з якою взаємодія організована черех модуль app.db.session (об’єкти engine, SesscioLocal). Доступ до таблиці інкапсульовано у наборі ORM-моделей простору імен app.models.*, що реалізуються на основі SQLModel/SQLAlchemy.

Ключові сутності:

- RunDB – метадані запуску;
- TopologySnapshot, TopologyNodeStateDB – знімки топології мережі для конкретного запуску та стани окремих вузлів;

- VulnerabilityDB – нормалізовані вразливості з полями для CVE-ідентифікатора, рівня критичності, CVSS-оцінки, прив'язки до вузла та джерела інструмента;
- допоміжні моделі (PolicyDB, RunLogDB, StatsDB тощор) для збепігання політик, журнаів виконання та агрегованої статистики.

Усі записи зв'язуються через атрибут run_id, що забезпечує логічну ізоляцію окремих тестів та спрощує побудову порівняльних вибірок.

Поверх реляційної схеми будується доменно-орієнтований граф знань, який використовується для семантичного узагальнення результатів тестування на проникнення та подальшої інтелектуальної обробки. Граф зберігається у тих самих реляційних таблицях, але з окремими ORM-моделями (рис. 4.5) KnowledgeNodeDB, KnowledgeEdgeDB (app.models.knowledge_db):

- вузол графа (KnowledgeNodeDB) має атрибути id, type, label та гнучке поле metadata (JSON), де фіксуються деталі активу чи вразливості);
- ребро графа (KnowledgeEdgeDB) описує семантичні зв'язки між вузлами, також з додатковими метаданими.

```python
# ---- Моделі таблиць ----

class KnowledgeNodeDB(SQLModel, table=True):
    """
    Вузол динамічного графа знань K.

    type:
      - "host"       вузол мережі (AMF, gNB, UE тощо)
      - "vuln"       вразливість (CVE, внутрішній ID)
      - "technique"  техніка/тактика (MITRE ATT&CK, FiGHT)
      - "tool"       інструмент (nmap, zap, metasploit)
      - "run"        конкретний запуск методу
      - "policy"     політика / профіль обмежень
    """

    __tablename__ = "knowledge_nodes"
```

```python
class KnowledgeEdgeDB(SQLModel, table=True):
    """
    Орієнтоване ребро графа знань.

    type:
      - "includes_host"      run -> host
      - "has_vuln"           host -> vuln
      - "found_in_run"       run -> vuln
      - "allowed_by_policy"  policy -> run
      - "uses_tool"          run/technique -> tool
    """

    __tablename__ = "knowledge_edges"
```

Рис. 4.5. Моделі KnowledgeNodeDB, KnowledgeEdgeDB (лістинг коду)

Побудова графа знань відбувається у модулі обробки вразливостей app.vuln.scan_and_correlate та пов'язаних з ним компонентах. Після кореляції результатів зовнішніх сканерів з базами NVD/MITRE створюються записи VulnerabilityDB, а паралельно – відповідні вузли й ребра KnowledgeNodeDB/

KnowledgeEdgeDB. Це дає змогу узагальнювати інформацію: одна вразливість може бути пов’язана з кількома активами, техніками атаки та бізнес-процесами.

Для читання графа знань підсистема звітності використовує допоміжну функцію _load_knowledge у модулі app.reports.builder. Вона трансформує ORM-об’єкти у легковагову структуру {“nodes”: […], “edges”: […]}, яка потім застосовується для генерації текстового звіту, побудови статистики та візуалізації в інтерфейсі. Аналогічну структуру повертають кінцеві точки REST модуля app.api.routes_knowledge, до яких звертається фронтенд-сторінка графа знань.

Підсистема даних та графа знань інтегрована у всі ключові етапи пайплайну:

- функція run_full_pentest_for_run (app.orchestrator.pipeline) створює та оновлює записи RunDB, фіксує часові мітки та метрики MARL;
- модуль topology_builder зберігає топологічні знімки у KnowledgeNodeDB/ KnowledgeEdgeDB;
- модулі vuln.* створюють VulnerabilityDB та відповідні вузли/ребра графа знань;
- модуль reports.builder агрегує інформацію з RunDB, VulnerabilityDB та Knowledge*-таблицю для формування структурованого об’єкта звіту, який далі перетворюється на PDF у app.reports.pdf_renderer;
- REST-шар (routes_runs, routes_vulnd, routes_knowledge, routes_topology тощо) забезпечує читання цих даних з фронтенду, але всі запити йдуть через єдиний ORM-шар без прямого доступу UI до БД.

Таким чином, підсистема даних та графа знань створює єдине узгоджене джерело істини (single source of truth) для всіх інших компонентів Montera.

4.2.2. Підсистема мережевої розвідки та побудови TelLAG графа

Підсистема мережевої розвідки та побудови графа TelLAG реалізовує логічний місток між фізичною/логічною інфраструктурою мережі та інтелектуальною частиною методу. Саме вона перетворює «сирі» результати сканування у формалізовану топологічну модель та далі у граф атак, який використовується MARL агентами для пошуку оптимальних ланцюжків дій.

Мережева розвідка реалізована у просторі імен app.network.discovery у вигляді функції discovery_hosts, яка інкапсулює виклики низькорівневих інструментів та повертає результати у вигляді типізованих структур HostInfo, PortInfo. Таким чином, тут реалізоване уніфіковане представлення сканування. Тобто, для кожного виявленого вузла формується об’єкт HostInfo з IP-адресою, MAC-адресою (за наявності), ідентифікованим hostname, а також списком PortInfo, що містять номеру порту, протокол, виявлений сервіс та супровідні атрибути (рис. 4.6.).

```python
@dataclass
class PortInfo:
    port: int
    protocol: str
    service: Optional[str] = None
    product: Optional[str] = None
    version: Optional[str] = None
    cpe: Optional[str] = None


@dataclass
class HostInfo:
    ip: str
    hostname: Optional[str] = None
    os: Optional[str] = None
    mac: Optional[str] = None
    ports: List[PortInfo] = field(default_factory=list)
```

Рис. 4.6. Фрагмент коду для формування уніфікованої інформації

Отримані дані з мережевої розвідки трансформуються у топологічну модель у модулі app.network.topology_builder. Центральною функцією є build_topology_from_hosts (run_id, hosts, session), яка створює новий запис TopologySnapshotDB для заданого run_id, фіксуючи опис та час генерації знімка, для кожного HostInfo формує вузол TopologyNodeStateDB з такими атрибутами як node_id, name, type, status, а також ряд додаткових атрибутів для аналітичних модулів.

Паралельно реалізовується функція build_topology_graph_from_hosts, що повертає фронтенду структуру TopologyResponse із списками TopologyNode та TopologyLink. Вона виділяє шлюзовий вузол за IP-патернами та/або ім’я хоста, створює зв’язки типу шлюз-хост, кодує тип вузла у вигляді просторого рядка. Фрагмент лістингу програмного коду відображено у Додатку А.

Логічний граф TelLAG будується у модулі app.tellag.service функцією build_tellag_graph_for_run(session, run_id). Її завданням є перетворення топології, вразливостей та політик у структуру, придатну для подальшого використання MARL-агентами (фрагмент коду агентів наведено у Додатку Б). Алгоритм роботи наведено на рисунку 4.7.

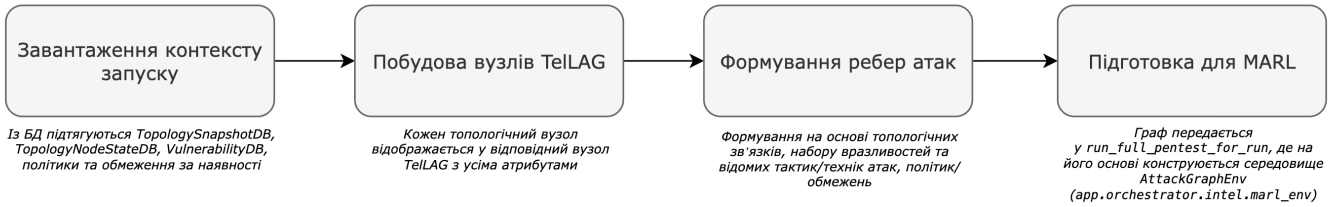

Рис. 4.7. Алгоритм побудови графа атак

Обидві складові мають своє графічне представлення на фронтенді, рисунок 4.8 та 4.9 відповідно.

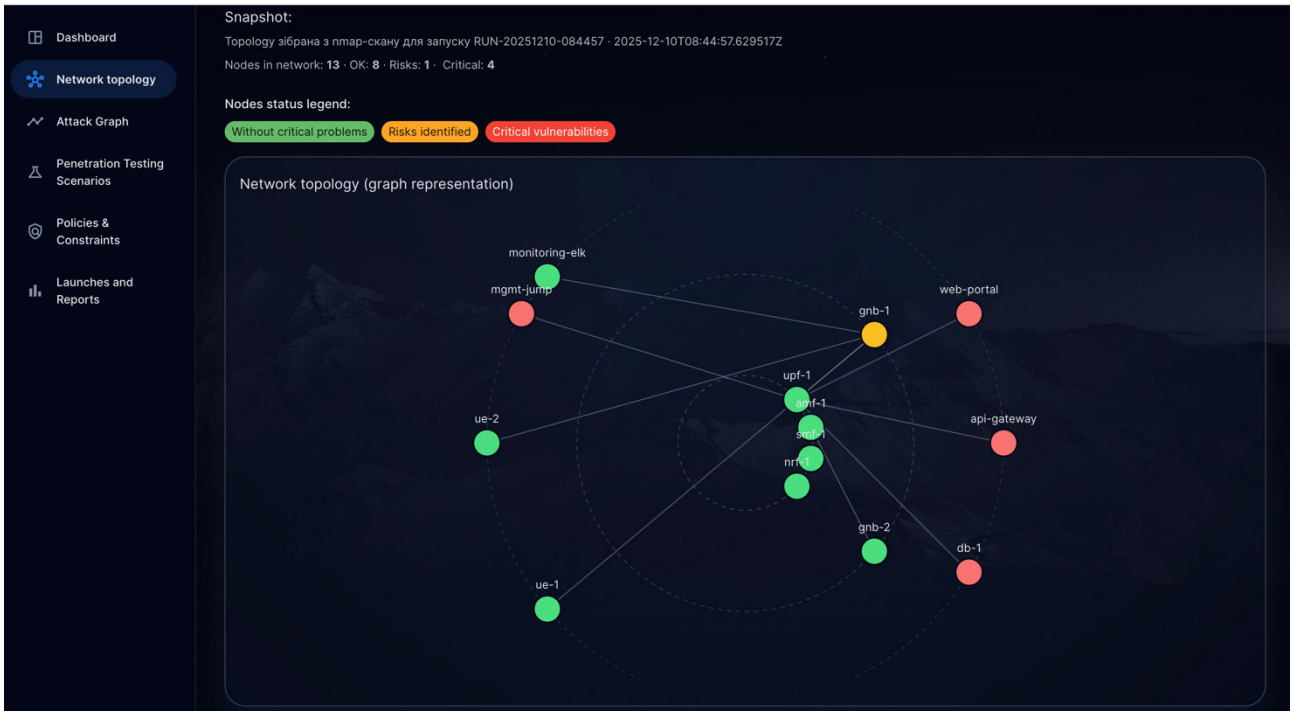

Рис. 4.8. Топологія мережі

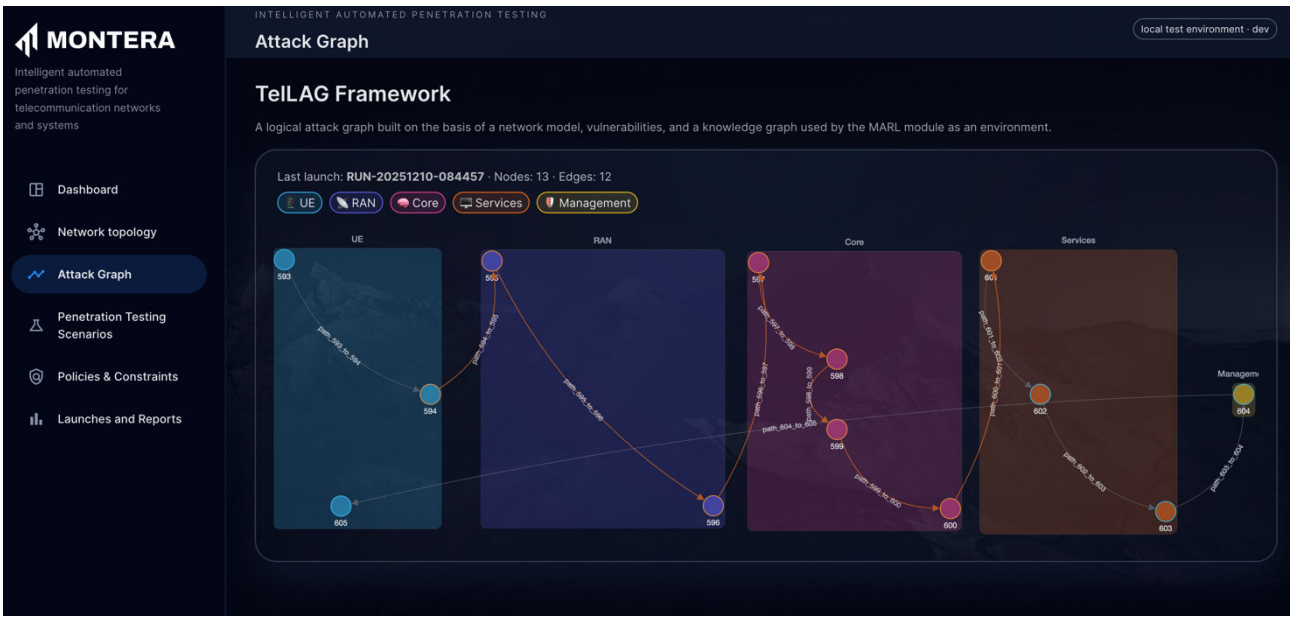

Рис. 4.9. Граф атак

4.2.3. Підсистема оркестрації сценаріїв тестування

Підсистема оркестрації сценаріїв тестування відповідає за перетворення абстрактного опису сценарію пентесту (множини цілей, обмежень, пріоритетів) у послідовність узгоджених технічних дій над цільовою телекомунікаційною мережею. На рівні архітектури вона поєднує модулі керування сценаріями, планування дій із залученням LLM, оркестрації зовнішніх інструментів та MARL компонентів, а також механізми моніторингу та журналювання перебігу запусків.

Логічна модель сценарію реалізована у шарі даних й інкапсулює цілі пентесту, набір активних агентів (Recon, Exploit, Lateral, Service), політики та обмеження (рис. 4.10). Далі це використовується як параметр середовища AttackGraphEnv та конфігурації інструментів.

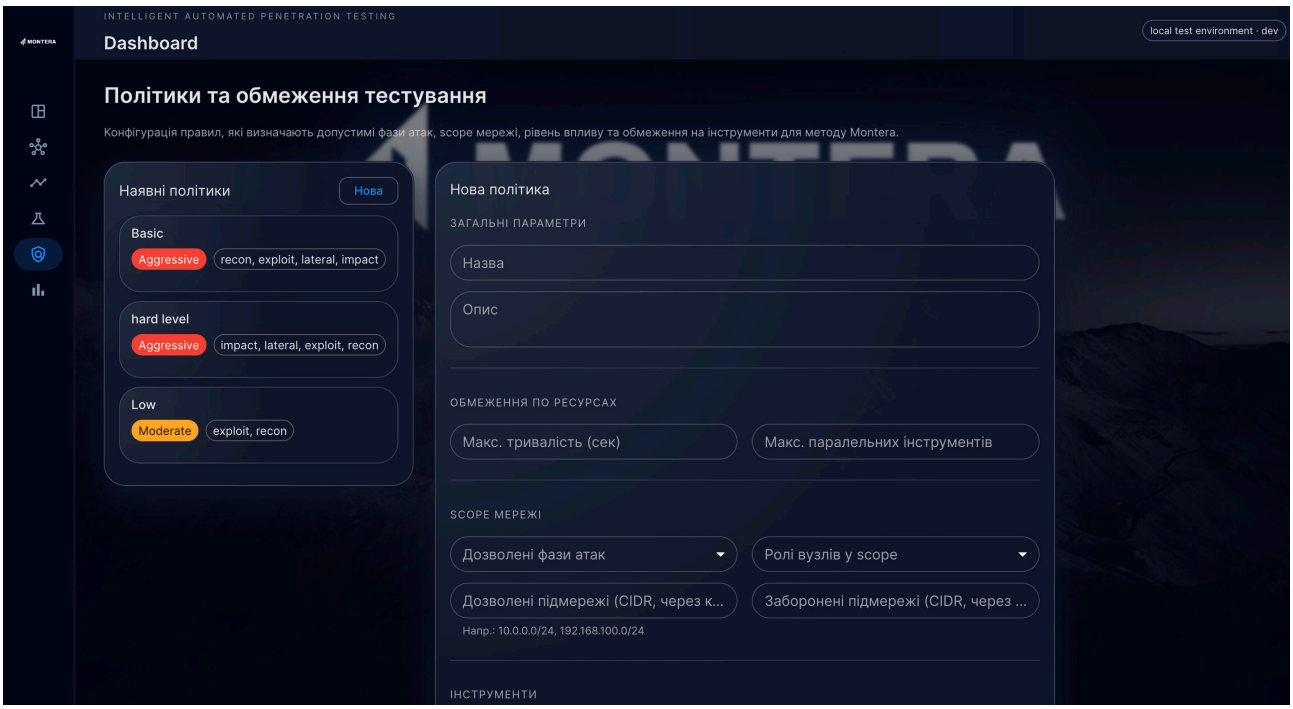

Рис. 4.10. Дашборд політик та обмежень

З боку бекенду ці сутності зберігаються у відповідних ORM-моделях та пов’язуються з конкретними запусками через ідентифікатори scenario_id, policy_id. Це дозволяє для кожного запуску відтворити початковий контекст тестування.

На рівні API робота зі сценаріями реалізована на кшталт routes-scenarios (CRUD-операції) та у сервісних модулях оркестратора. Створення сценарію може відбуватися:

- експліцитно – користувач задає параметри вручну;
- напівавтоматично – через інтегрований LLM-планувальник, що на основі опису середовища та вимог генерує рекомендовану конфігурацію сценарію

У типових інтегрованих запусках, шо відображаються на дашборді, фронтенд викликає бекенд-метод створення сценарію, який на сервері створює запис з дефолтним набором агентів [“Recon”, “Exploit”, “Lateral”, “Service”] і прив’язує його до нового RunDB. Таким чином, підсистема оркестрації формує єдиний «контракт» запуску, яким далі керує центральний пайплайн. Фрагмент коду наведений у Додатку В.

Приклад згенерованих сценаріїв за допомогою LLM наведено на рисунку 4.11.

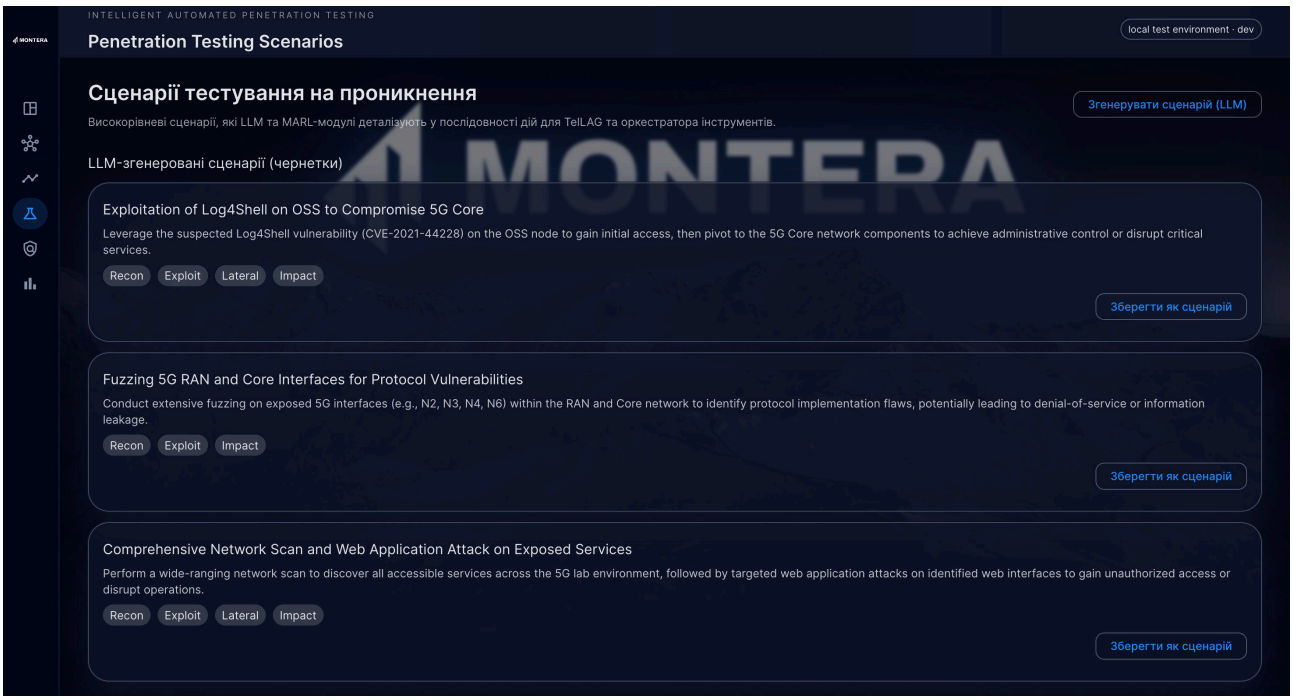

Рис. 4.11. Приклад згенерованих сценаріїв за допомогою LLM

Ключовим елементом підсистеми оркестрації є функція run_full_pentest_for_run у модулі app.orhestrator.pipeline. Вона реалізовує повний життєвий цикл сценарію як послідовність кроків: (1) інціалізація запуску, (2) організація режиму роботи (є можливість тестування на симуляції та робота з реальною мережею), (3) оркестрація етапів мережевої розвідки, топології, TelLAG, (4) оркестрація зовнішніх інструментів, (5) кореляція вразливостей та оновлення стану вузлів).

Після побудови TelLAG та збагачення його даними про вразливості, підсистема оркестрації ініціює MARL етап. Усередині run_full_pentest_for_run створюється середовище AttackGraphEnv (app.orchestrator.intel.marl_env) (Додаток Г), де вузлам TelLAG відповідають стани середовища, ребрам – дії з атрибутами action, agent_id, base_reward, recommended_tool, множини початкових і цільових станів задаються на основі entry_nodes та target_nodes.

Функція optimize_scenarios з модуля app.orchestrator.intel.marl_agent отримує список сценаріїв у вигляді графів і фабрику середовища _make_env, після чого повертає оптимізовані низькорівневі послідовності дій. Пайплайн зберігає метрики (кількість кроків, покриття графа, кількість критичних знахідок) у полях RunDB, які надалі використовуються дашбордом та підсистемою формування PDF-звітів.

Окремо оркестратор взаємодіє з LLM-компонентами у частині сценарного планування та семантичного узагальнення результатів. Це відбувається вже після завершення технічних етапів запуску, коли всі ключові артефакти тестування вже збережені у БД.

Підсистема оркестрації забезпечує наскрізне журналювання за допомогою сервісу log_to_db. Для кожної значущої події створюється запис із рівнем важливості (INFO/WARNING/ERROR), який відображається у UI як «живий» лог конкретного запуску (рис. 4.12).

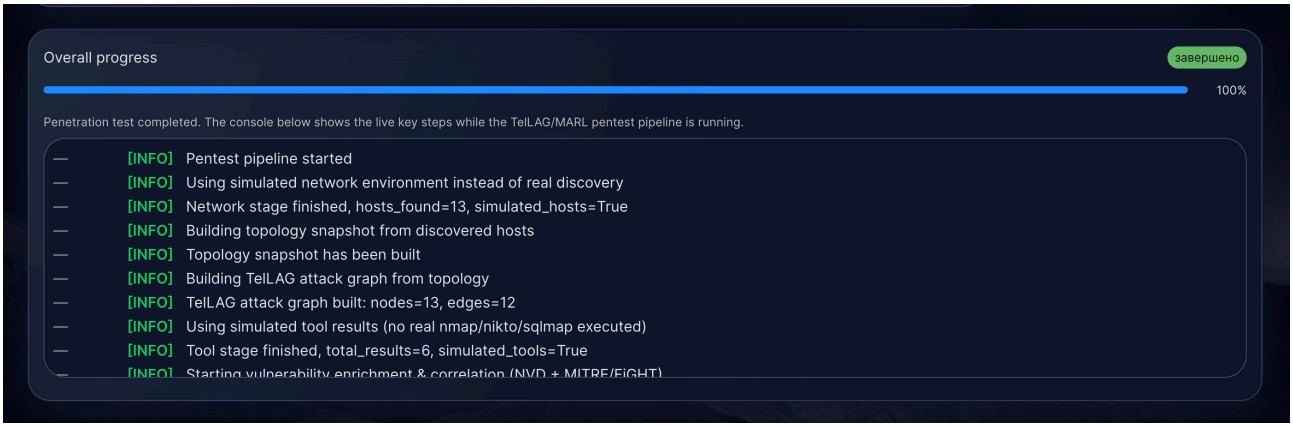

Рис. 4.12. Логи в UI

На завершенні всіх етапів пайплайн обчислює тривалість запуску, кількість кроків, оцінку покриття (run_db.coverage); оновлює статус запуску (completed або failed) та поле finished_at; фіксує підсумковий лог «Pentest pipeline completed successfully» або опис помилки.

Таким чином, підсистема оркестрації сценаріїв тестування реалізовує керований, відтворюваний та прозорий життєвий цикл запуску методу.

4.2.4. Підсистема аналізу вразливостей та семантичного збагачення

Підсистема аналізу вразливостей та семантичного збагачення реалізовує перехід від «сирих» технічних результатів роботи інструментів тестування на проникнення до узгодженого набору структурованих знахідок, бізнес-орієнтованих

ризиків та текстових узагальнень, придатних для використання у звітності та подальшому прийнятті рішень.

Вхідними даними для цієї підсистеми ж об’єкти ToolResult, сформовані підсистемою інструментів для кожного хоста й сервісу цільової тестованої мережі. Функціонально ключову роль відіграє модуль app.vuln.scan_and_correlate, який у контексті запуску використовується через функцію scan_and_correlate_for_run (рис. 4.13).

Рис. 4.13. Функція scan_and_correlate_for_run

На першому етапі результати сканування канонізуються. Для кожного знайденого потенційного дефекту виділяється базові атрибути (ідентифікатори CVE, рівень критичності, значення CVSS, опис, джерело інструменту, прив’язка до вузла топології). Ці дані зберігаються у таблиці VulnerabilityDB та стають основою для внутрішнього єдиного представлення вразливостей.

Для зручності подальшої обробки у полі extra_info зберігається JSON-структура, яка може включати фрагменти з офіційних баз даних, додаткові атрибути, проміжні маркери кореляції.

У підсистемі формування звіту (модуль app.reports.builder) ці записи завантажуються окремою функцією _load_vulnerabilities, яка перетворює ORM-об’єкти VulnerabilityDB у уніфікований список findings:List[Dict[str,Any]]. На цьому етапі узгоджується рівень критичності, уніфіковується значення CVSS; формується текстовий опис впливу (impact), який, залежно від наявних полів, складається з комбінації «business_impact”, “impact”, “summary” тощо; задається базова рекомендація, якщо вона не була надана інструментом чи зовнішнім джерело.

Таким чином, на виході формується структурований масив findings, що містить для кожної вразливості заголовок, рівень критичності, CVSS, прив’язку до вузла, потенційний CVE ID і назву інструмента-джерела (рис. 4.14).

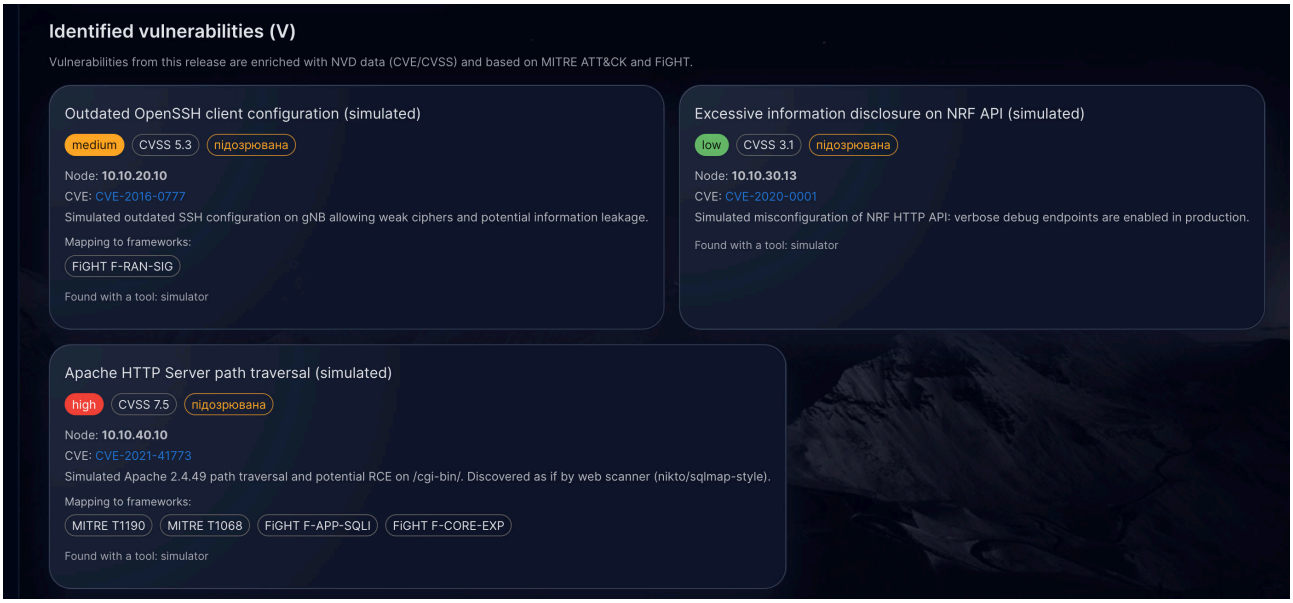

Рис. 4.14. Відображення в UI

Підсистема аналізу вразливостей тісно інтегрована з графом знань та семантичним підсумком. Дані графа знань завантажуються за допомогою функції _load_knowledge, яка працює з моделями KnowledgeNodeDB та KnowledgeEdgeDB. У поточні реалізації граф в основному використовується для агрегованого огляду і його структура дозволяє представлення вразливостей окремим типом вузлів, пов’язування цих вузлів з активами, елементами графа атак, тактиками/техніками та іншими сутностями, а також побудову складніших зв’язків для подальшого автоматизованого аналізу та оптимізації шляхів атак.

Семантичний підсумок RunSematicSummaryDB завантажується функцією _load_semantic_summary. Він розглядається як напівструкторований шар поверх технічних даних і може містити класифікацію знахідок за категоріями, узагальнення типових причин і рекомендованих напрямів удосконалення та агреговані оцінки ступеня ризику для різних сегментів мережі.

Обидві структури інтегруються у фінальний об’єкт звіту як окремі секції, який надалі передається у PDF-рендерер.

Отримані дані з VulnerabilityDB використовуються також для формування ключових технічних метрик запуску, що забезпечує консистентність між різними підсистемами. Адже топологічне відображення, дашборд, MARL метрики та звітність опираються на одну й ту ж саму базу фактів про вразливості.

Ключовою особливістю підсистеми є використання зовнішньої LLM. Gemini інтегрований через app.orchestrator.intel.llm_client.call_chat_json використовуються для автоматизованої побудови бізнес-орієнтованих інтерпретацій технічної інформації про знайдені вразливості. На рисунку 4.15 приклад отриманих результатів.

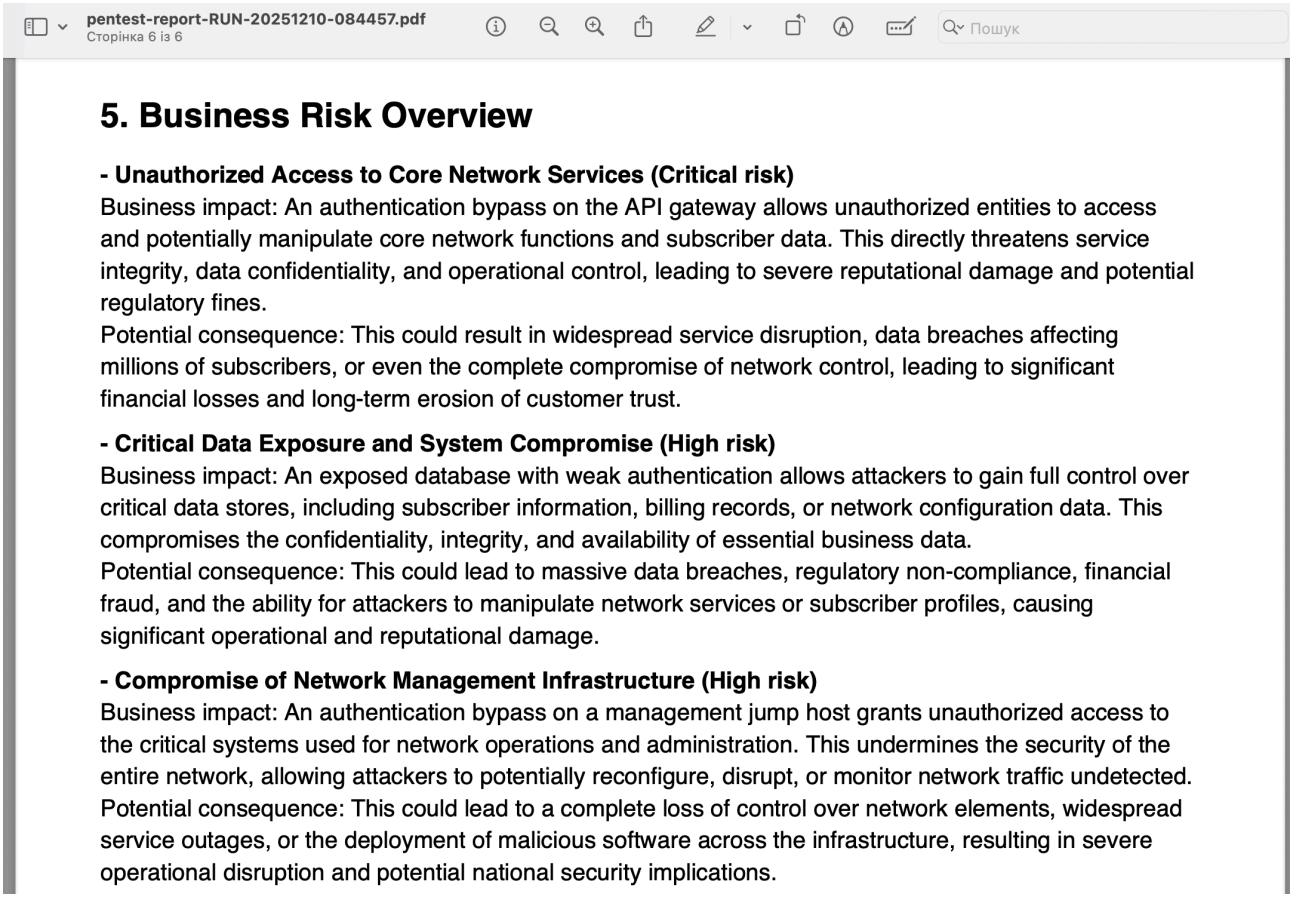

Рис. 4.15. Приклад отриманих бізнес-ризиків від Gemini

Одержана відповідь піддається валідації у контексті перевірки типів, заповнення обов’язкових полів, нормалізація всіх значень. У разі відсутності LLM або помилки виклику, підсистема застосовуватиме евристичні значення за замовчування.

Згенерується узагальнений бізнес-ризик для всієї мережі та формальний підсумок, що описує загальний характер тестування та потребу у пріоритезації виявлених вразливостей.

Завдяки чіткому поділу на технічний та семантичний рівні, підсистема легко розширюється. Можна додавати нові джерела даних, нові типи ризиків чи змінювати політику формування описів без модифікації інших частин системи.

4.2.5. Підсистема звітності та метрик

Підсистема звітності та метрики у додатку забезпечує формалізоване представлення результатів виконання методу у двох взаємодоповнювальних вимірах: (1) операційні метрики запуску для моніторингу прогресу, порівняння тестів і відображення на дашбордах; (2) аналітична звітність, яка агрегує технічні знахідки, узагальнення методології та бізнес-орієнтовану інтерпретацію ризиків. Архітектурно ця підсистема виступає «вихідним контуром» пайплайна. Вона спирається на дані, сформовані підсистемами оркестрації, мережевої розвідки, аналізу вразливостей та семантичного збагачення, трансформує їх у формат, придатний для експлуатаційного використання та документування результатів.

Первинні метрики формуються у межах виконання оркестраційного пайплайна app/orchestrator/pipeline.py після завершення ключових стадій. У структурі запуску моделі акумулюються як часові, так і процесні та якісні показники, зокрема: duration_seconds, steps_total, steps_effective, coverage, critical.

Важливою властивістю реалізації є те, що метрики узгоджуються з даними підсистеми вразливостей.

Для фронтенд інтерфейсу система надає агреговані показники на рівні набору запусків: загальна кількість запусків, кількість успішно завершених, сукупна кількість критичних знахідок, середнє покриття тощо (рис.4.16). Логічно ці величини формуються на підставі колекції RinDB і підвантаження деталей останнього запуску. Архітектурно важливо, що підсистема метрик розділяє метрики конкретного запуску та метрики на рівні системи. Саме таке розділення є необхідним для коректного масштабування, адже обчислення агрегатів не повинно залежати від дорогої

реконструкції всіх артефактів запуску, а має використовувати попередньо збережені значення та індексовані поля.

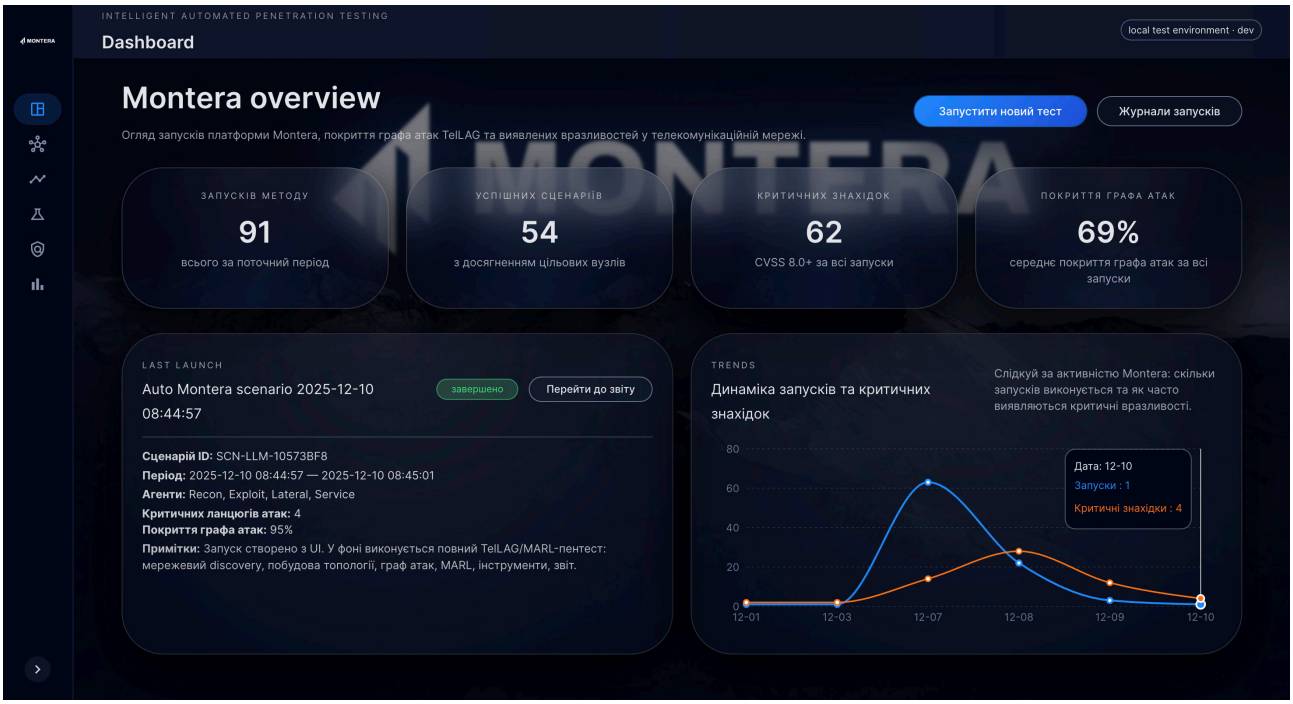

Рис. 4.16. Головний дашборд

Звітність у додатку реалізована через форматування високорівневої структури вихідних даних, яка виступає проміжним незалежним представлення між БД та PDF-рендером. Ця структура виконується у app/reports/builder.py функцією build_run_report(run_id, session) і включає:

- контекст запуску (summary);
- опис методології (methodology);
- технічні знахідки (findings);
- огляд графа знань (knowledge_overview) та семантичний підсумок (semantic_summary);
- бізнес-ризики та виконавчий звіт (business_risks, llm_summary).

Фізичне форматування PDF-документа виконується у модулі app/reports/pdf_renderer.py, який інтерпретує попередню структуру даних та формує документ засобами ReportLab (рис. 4.17). Це включає: керування пагінацією та

візуальною консистентністю; ієрархію секцій; базову типографіку. За своєю суттю, рендерер є суто «презентаційною» компонентою.

Рис. 4.17. Фрагмент документу звіту про пентест

4.2.6. Підсистема користувацького інтерфейсу та REST API

Дана підсистема виконує роль інтеграційного «контуру взаємодії» між користувачем і внутрішніми сервісами платформи. ЇЇ призначення полягає у (1) стандартизованому наданні доступу до функцій оркестрації та даних через HTTP-інтерфейси (REST); (2) інтерактивному поданні прогресу виконання та артефактів методу, а також (3) забезпеченні керованості тестувань. Архітектурно підсистема реалізовує принцип розділення відповідальностей: бекенд-API інкапсулює предметну логіку та доступ до БД, тоді як фронтенд зосереджений на візуалізації, навігації та коректній інтерпретації станів.

REST API виступає формальним контрактом, що узгоджує обмін даними між клієнтською частиною та серверними модулями. На концептуальному рівні API

організовано навколо доменних сутностей та процесів, зокрема: запуски (Runs), вразливості (Vulns), топологія (Topology), TelLAG/AttackGraph, сценарії та політики (Scenarios & Policies), звіти (Reports).

Важливо, що API не лише повертає дані, а й підтримує стани та життєвий цикл запуску. Це дозволяє інтерфейсу відображати прогресу у наближеному до реального часу режимі, спираючись на сервері записи логів та метрик.

Для забезпечення типобезпеки та стабільності контракту API використовується явне моделювання DTO-представлень. Даний підхід знижує ризик неявних залежностей між UI та ORM-структурами, оскільки:

1. ORM-моделі оптимізовані під збереження даних і зв’язки, а не під публічний інтерфейс.
2. API-схеми забезпечують контроль форматів.
3. Зміни внутрішньої реалізації меншою мірою впливають на клієнтські компоненти.

У контексті даного програмного забепечення і методу в цілому, це є критичним через гетерогенність джерел даних та потребу в стабільність інтеграції між фронт- та бекендом.

Однією з вимог до практично придатної платформи автоматизованого пентесту є наявність «пояснювального прогресу». Тобто, повідомлень та індикаторів стану, що дають користувачеві контекст, що саме виконується в даний проміжок часу, які стадії завершені, де виник збій/помилка. У додатку це реалізовано через DB live logs, узгодження логів із етапами пайплайна та відображення статусів.

Фронтенд Montera реалізовує візуально-аналітичний контур, який перетворює внутрішні артефакти методу в інтерфейс оператора. З позиції функціональної декомпозиції UI це включає: головний дашборд, мережеву топологію, граф атак, сценарії, політики та обмеження, а також запуски та звіти.

Оскільки Montera функціонує як веб-платформа з інтенсивною навігацією між модульними сторінками, важливим є узгоджений каркас інтерфейсу. На практиці це виражається наступний чином:

- централізоване компонування через MainLayout;

- інваріативне навігаційне меню (Sidebar) зі станами активної секції;
- адаптивність (responsive) та керованість робочого простору (зокрема, згортання сайдбару);
- єдина дизайн-система (Material UI), що забезпечує стилістичну консистентність та швидку ітерацію інтерфейсу.

4.3. Розробка методології тестування на тестовій мережі 5G

Валідація програмного забезпечення Montera, розглядається як інтеграційне тестування в контрольованому середовищі тестової мережі 5G, метою якого є підтвердження коректності наскрізного конвеєра: мережева розвідка, побудова топології та графа атак, оркестрація інструментів, кореляція/збагачення вразливостей, формування звітів і метрик. Такий підхід відповідає практиці безпечних досліджень у сфері кібербезпеки, коли оцінювання здійснюється в межах ізольованого стенді із чітко визначеною зоною відповідальності та правилами експлуатації, що забезпечує відтворюваність результатів і мінімізує ризики впливу на сторонні системи.

Практична потреба саме тестової мережі 5G пов’язана з тим, що сучасні мережі зв’язку п’ятого покоління використовуються віртуалізацію (NVF/CNF) і контейнеризацію мережевих функцій, а також мають специфічні протоколи й критичні метрики якості обслуговування (QoE, QoS), вплив на які також має враховуватись під час тестувань.

4.3.1. Огляд тестового середовища 5G

Тестовим середовищем 5G є розгорнута мережа п’ятого покоління на основі проєктів з відкритим вихідним кодом (OpenAirInterface), що включає в себе RAN, 5GC, UE, DN (рис. 4.18) у наступній конфігурації:

- 5G Core (Core VM): віртуальна машина, на якій розгорнуто ядро мережі 5G у Docker Compose як набір контейнеризованих мережевих функцій та допоміжних сервісів;

- RAN (Radio Access Netowork): окремий ПК (OS Ubuntu Server) із розгорнутим gNodeB, підключеним до SDR USRP B210, що забезпечує радіоінтерфейс доступу;
- UE (User Equipment): абонентський пристрій (смартфон), який виконує процедури реєстрації/підключення до мережі;
- DN: зовнішня мережа даних, що використовується в тестовому середовищі.

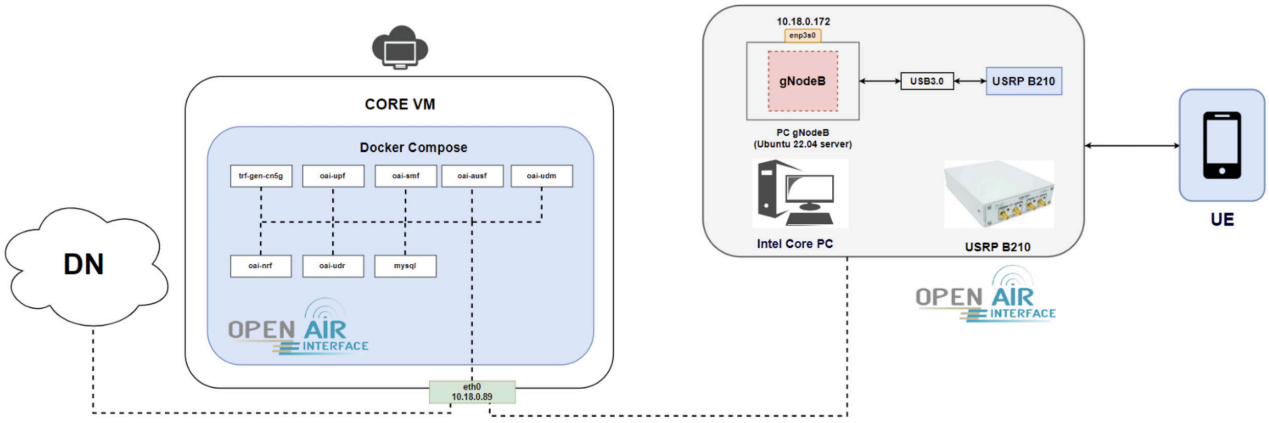

Рис. 4.18. Тестове середовище 5G [78]

Межі тестування (scope) задаються політиками та обмеженнями, які вузли/сегменти дозволено тестувати, які типи дій є допустимими/забороненими, які часові вікна застосовуються та які ліміти навантаження встановлюються. Відповідно до розробленого методу, такі політики мають враховуватися під час генерації сценаріїв та їх виконання.

4.3.2. Процедура інтеграційного тестування

Інтеграційне тестування додатку Montera організовано як керований ітеративний процес, у якому послідовно проходить повний цикл: ініціалізація середовища, побудова графових моделей, формування сценаріїв, виконання та збір артефактів, оцінювання, звітність, оновлення бази знань. Узагальнена логіка подана на рисунку 4.19.

Рис. 4.19. Методологія тестування розробленого додатку

Початком тестування є підготовлена, запущена тестова мережа 5G. Далі процедура тестування реалізовує чотири узгоджені фази.

На першій фазі, ініціалізація та фіксація базового стану мережі. Перед запуском фіксуються конфігурації та версії компонентів мережі, активуються необхідні механізми журналювання та визначаються межі тестування. Це створює контрольовану стартову точку та забезпечує порівнюваність результатів між прогонами.

Після чого відбувається інтеграція Montera та побудова графових представлень. Додаток підключається до джерел даних і результатів інструментів та ініціалізовує внутрішні моделі, граф знань та граф атак. На цій фазі також задаються політики виконання, які обмежують небажані дії та визначаються допустимі режими тестування.

Третя фраза передбачає формування сценаріїв тестування та їх виконання. На основі поточного стану графів знань та атак, формуватиметься набір тестових сценаріїв і передача їх на виконання оркестраторам.

На останній фазі відбувається оцінювання результатів, формування звіту, оновлення знань та ітерації. Отримані артефакти інтерпретуються як формалізовані факти, після чого актуалізовуються граф знань і за потреби граф атак. Далі можливі

зворотні переходи: (а) уточнення сценаріїв, якщо потрібне додаткове покриття або верифікація гіпотез, (б) уточнення політик чи інтеграції, якщо виявлено обмеження тестової мережі, некоректні припущення або потребу в зміні конфігурації збору даних. Цикл завершується після досягнення критеріїв зупинки.

Таким чином, тестування інтерпретується як відтворюваний протокол із чіткою послідовністю фаз, де кожна ітерація одночасно (1) перевіряє працездатність інтеграції програмного рішення з тестовим середовищем, (2) підтверджує коректність графових моделей на реальних даних та (3) надає вимірювані результати для оцінювання ефективності методу і якості його реалізації.

4.3.3. Метрики, критерії успіху та протокол відтворюваності

Оцінювання тестування виконується як багатокритеріальна перевірка, що охоплює: (i) коректність наскрізного конвеєра методу, (ii) результативність сценаріїв і графового аналізу, (iii) безпечність виконання в телекомунікаційному середовищі з огляду на політики SLA/QoS. Необхідність включення QoS-метрик обумовлена тим, що для мереж 5G критичними є затримка, джиттер і пропускна здатність, а методологія тестування має передбачати оцінювання тестових дій/атак на ці параметри.

Для уніфікації збору та інтерпретації результатів використовується набір метрик згрупованих за призначенням. Формально джерело даних для метрик є: (а) необроблені артефакти виконання інструментів, (б) семантичні результати інтерпретації (табл. 4.2).

Таблиця 4.2

Метрики тестування

Група	**Метрика**	**Визначення**	**Джерело**	**Інтерпретація**
Коректність інтеграції	SR_{run} – частка	Success_run /total_run	Журнали пайплайна, статуси	Базова працездатність

Продовження таблиці 4.2

	успішних ітерацій			
	SR_{act} – частка успішних ітерацій	Success_act/total_act у сценаріях	Логи окрестратора, коди завершення інструментів	Якість маппінгу абстрактних дій
Покриття	C_N – покриття вузлів	$C_N = \frac{\lvert N_{reached} \rvert}{N_0}$	Топологія мережі, граф атак	Якість абстрактних дій
	C_E – покриття ребер	$C_E = \frac{\lvert E_{activated} \rvert}{E_0}$	Топологія мережі, граф атак	Якість абстрактних дій
Результативність	V_{conf} – підтверджені вразливості	Кількість вразливостей, що отримали підтвердження виконанням/артефактами	Семантичні результати інтерпретування	Практична цінність
Якість інтерпрепретації	FP, FN	Хибнонегативні/хибнопозитивні спрацювання відносно контрольних істин	Контрольні мітки тестової мережі + результати Montera	Надійність результатів та висновків
Ризик-орієнтовані	R – агрегований ризик	Агрегування ваг/ймовірностей для активованих вузлів/ребер	Ваги графа, шар оцінювання ризику	Кількісна пріоритезація результатів

Закінчення таблиці 4.2

Дотримання політик	P_{viol} – порушення політик	Кількісна частка дій, що не проходять предикат дозволеності $\pi(s,a)$	Журнал рішень policy-check	Гарантія безпечності тестування
Вплив на сервіс	$\Delta L, \Delta J, \Delta B$	Відхилення затримки, джиттера, пропускної здатності від прийнятого значення	Вимірювання в тестовій мережі (iperf/ping)	Контроль та відповідність до SLA/QoS
Ефективність	T_{plan}, T_{exec}, T_{total}	Час планування сценаріїв / час виконання / загальний час	Таймстемпи	Придатність до практичного застосування
Стабільність	Var(•)	Дисперсія ключових метрик на повторних запусках	Повторні запуски	Ознака відтворюваності та керованості

Таким чином, після одного повного запуску буде отримано набір метрик:

$$m = \{SR_{run}, SR_{act}, C_N, C_E, V_{conf}, FP, FN, R, P_{viol}, \Delta L, \Delta J, \Delta B, T_{plan,} T_{exec}, T_{total}, Var(\cdot)\}. \quad (4.1)$$

Тоді інтегральний показник успішності тестування доцільно задавати як зважену скаляризацію нормалізованих метрик:

$$S = \sum_{i=1}^{k} \omega_i \tilde{m}_i, \quad \sum_{i=1}^{k} \omega_i = 1, \quad \omega_i \geq 0, \tag{4.2}$$

де $\tilde{m}_i \in [0,1]$ – нормалізовані показники (чим більше, тим краще).

Базові нормалізації C_N, C_E можна звести в один показник покриття наступним чином:

$$C = \lambda C_N + (1 - \lambda) C_E, \quad \lambda \in [0,1]. \tag{4.3}$$

Оскільки політика формується як предикат $\pi(s, a)$, який визначає чи дозволена дія a у стані s, то:

$$P_{comp} = \frac{1}{Total_{act}} \sum_{t=1}^{Total_{act}} \pi(s_t, a_t), \quad P_{viol} = 1 - P_{comp}. \tag{4.4}$$

У зведеному оцінюванні доцільно використовувати саме дотримання політик: $\tilde{P} = 1 - P_{viol} = P_{comp}$.

Враховуючи наявність вагової функції у методі автоматизованого тестування на проникнення, агрегований ризик можна визначити як:

$$R = \sum_{v \in N_{reached}} \omega(v) + \sum_{v \in N_{activated}} \omega(e) \tag{4.5}$$

Для зведення в [0,1], $\tilde{R} = \min(1, R/R_{max})$.

За наявності контрольних істин (міток), є можливим перехід від (FP, FN) до єдиного показника або до штрафу:

$$\tilde{Q}_{interp} = 1 - (\beta_{FP} FP_{rate} + \beta_{FN} FN_{rate}), \quad \beta_{FP} + \beta_{FN} = 1 \tag{4.6}$$

де $FP_{rate}, FN_{rate} \in [0,1]$ – частки, в залежності від того як ведуться контрольні істини.

Вплив на сервіс (SLA/QoS) можна визначити як:

$$\tilde{Q}_{svc} = 1 - \frac{1}{3}\left(\min\left(1, \frac{|\Delta L|}{\Delta L_{thr}}\right) + \min\left(1, \frac{|\Delta J|}{\Delta J_{thr}}\right) + \min\left(1, \frac{|\Delta B|}{\Delta B_{thr}}\right)\right) \tag{4.7}$$

Ефективність та стабільність визначається наступним чином:

$$\tilde{Q}_{time} = 1 - \min\left(1, \frac{T_{total}}{T_{thr}}\right), \quad \tilde{Q}_{stab} = 1 - \min\left(1, \frac{Var(S)}{Var_{thr}}\right). \tag{4.8}$$

Таким чином, загальна формула оцінювання успішності тестування може бути записана як:

$$\begin{aligned} = \omega_1 SR_{run} + \ \omega_2 SR_{act} + \omega_3 C + \omega_4 V_{conf} + \omega_5 \tilde{Q}_{interp} + \omega_6 \tilde{R} + \omega_7 (1 - P_{viol}) \\ + \omega_8 \tilde{Q}_{svc} + \omega_9 \tilde{Q}_{time} + \omega_{10} \tilde{Q}_{stab}. \end{aligned} \tag{4.9}$$

У результаті, зміст оптимізації/оцінювання узгоджується з цілями методу та відповідає логіці «ризик + покриття + штрафи», як в постановці винагороди.

Критерії успішності. Ці критерії тестування визначаються як необхідні умови коректної реалізації методології та її застосовності до тестової мережі 5G: (а) наскрізна працездатність конвеєра методу, якщо $SR_{run} \approx 1$, то відсутні аварійні завершення і розриви роботи модулів; (б) політико керована безпечність виконання, ключовим показником буде $P_{viol} = 1$; (в) результативність і прогрес за покритям, має бути або зростання C_N, C_E, або досягнення одного з критеріїв зупинки; (г) контрольований вплив на SLA/QoS, всі відхилення $\Delta L, \Delta J, \Delta B$ мають залишатися в визначених межах.

Протокол відтворюваності. Для забезпечення відтворюваності тестувань має застосовуватися протокол, який фіксуватиме конфігурацію, вхідні дані, політики, параметри запуску та артефакти:

- фіксація тестованого середовища (версії ОС, ПЗ, конфігурації контейнерів, параметри компонентів мережі, мережеві адреси/маршрути і т.д.);
- фіксація методичних параметрів (політики та обмеження, бюджети часу/ресурсів, визначені цілі/активи, часові вікна виконання);
- артефакти тестів (журнали оркестратора та інструментів, «сирі» результати виконання, семантичні результати інтерпретації, знімки графа знань і графа атак до та після, вимірювання QoS baseline тощо);
- параметри стохастичних компонентів: сіди генерації сценаріїв/планування, щоб мінімізувати недетермінізм та забезпечити порівнюваність між запусками);
- умови повторного тестування (інструкція запуску мережі, критерії зупинки).

Такий протокол забезпечуватиме (і) технічну відтворюваність (повторний запуск у тому самому тестовому середовищі) та (іі) наукову відтворюваність (можливість незалежно відтворити експериментальне дослідження на іншому середовищі за умови збереження еквівалентної конфігурації та вхідних артефактів).

ВИСНОВКИ ДО РОЗДІЛУ 4

У Розділі 4 сформовано повний набір вимог до програмної реалізації методу: визначено функціональні вимоги (збір/ETL даних, побудова моделі мережі та TelLAG, інтеграція з графом знань, оркестрація інструментів, інтеграція LLM і MARL, політики та звітність), а також нефункціональні вимоги, що задають критерії якості реалізації (модульність, масштабованість, надійність і безпечність, інтероперабельність, продуктивність і портативність).

Обґрунтовано архітектурну організацію Montera на основі трирівневого клієнт-серверного принципу з подальшим логічним розшаруванням на GUI, API, шар оркестрації та інтелектуальної обробки, шар даних і знань та контур зовнішніх сервісів, що забезпечує чіткий розподіл відповідальностей і масштабованість системи.

Визначено та реалізовано центральну роль шару Orchestration & Intelligence як «ядра» практичної імплементації методу: модуль Pipeline задає керований workflow одного запуску (від розвідки й побудови топології/графа атак до кореляції вразливостей, MARL-оптимізації та формування звіту), а модулі Discovery і TelLAG формують вхідне середовище для подальших інтелектуальних процедур.

Показано, що Data & Knowledge Layer є єдиним узгодженим контуром збереження станів і артефактів, який акумулює результати запусків, топологічні знімки, граф атак TelLAG, вразливості та їх кореляцію з таксономіями, а також структури графа знань; таким чином забезпечуються довготривале зберігання, відтворюваність та можливість подальшої аналітики.

Обґрунтовано вибір технологічного стеку як такого, що відповідає вимогам модульності, портативності та відтворюваного розгортання: застосовано сучасні web- і backend-технології, контейнеризацію, а також інтеграцію з LLM та зовнішніми джерелами вразливостей/таксономій для семантичного збагачення результатів.

Реалізацію модулів системи побудовано за принципами багаторівневої та сервісно-орієнтованої декомпозиції коду, що зменшує міжмодульні залежності та локалізує зміни в межах окремих компонентів (дані/знання, TelLAG, аналіз вразливостей, звітність тощо), створюючи підґрунтя для еволюції платформи без перебудови її ядра.

Розроблено та формалізовано методологію інтеграційного тестування в контрольованому 5G-стенді як ітеративний процес із чотирьох фаз (фіксація базового стану → побудова внутрішніх графових моделей → виконання сценаріїв → оцінювання, звітність та оновлення знань), що забезпечує керовану перевірку наскрізного конвеєра й коректності графових моделей на реальних даних.

Запропоновано багатокритеріальний апарат оцінювання (включно з метриками коректності інтеграції, покриття, результативності, дотримання політик, ризик-орієнтованими показниками та контролем впливу на SLA/QoS), а також визначено критерії успішності та протокол відтворюваності, що фіксує конфігурацію середовища, параметри запуску і ключові артефакти.

Таким чином, у Розділі 4 здійснено перехід від концептуально-формальної моделі до цілісного програмного прототипу Montera з визначеними інтерфейсами, модулями та протоколом експериментальної валідації, що забезпечує практичну перевірку працездатності методу на тестовій 5G-мережі та формує основу для подальшого експериментального дослідження.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ

Зважаючи на те, що під час тестування методу автоматизованого тестування на проникнення використовується тестовий стенд мережі 5G, виникає питання охорони праці та безпеки під час роботи з радіочастотним (РЧ) обладнанням у лабораторному середовищі 5G-стенду.

Суб'єктом захисту є працівник лабораторії – інженер з випробувань телекомунікаційного/радіочастотного обладнання, який виконує налаштування, підключення та вимірювання параметрів РЧ-тракту (SDR/передавачі, підсилювачі, антени або еквівалент навантаження, ВЧ-кабелі, вимірювальні прилади, блоки живлення, ПК).

Актуальність теми зумовлена тим, що експлуатація РЧ-джерел та високочастотних трактів у 5G мережах поєднує ризики впливу електромагнітних полів, електротравм, а також пожежної небезпеки внаслідок перенавантаження в електромережі, перегріву апаратури або коротких замикань.

5.1. Шкідливі та небезпечні виробничі фактори

Для аналізу обрано робоче місце інженера у лабораторії з мобільних мереж, де фізично знаходиться стенд тестової мережі 5G. Типові роботи включають: монтаж/перемикання ВЧ-тракту, увімкнення та налаштування SDR/передавачів і підсилювачів, вимірювання параметрів сигналу, налаштування програмного керування та ведення журналів випробувань. Відповідно до законодавчих вимог, роботодавець зобов'язаний забезпечити безпечні та нешкідливі умови праці, а працівник – виконувати вимоги нормативних актів та інструкцій з охорони праці [79].

З огляду на характер робіт у зазначеній лабораторії, на працівника можуть впливати наступні шкідливі та небезпечні виробничі фактори: фізичні фактори,

технічні та механічні фактори (небезпечні), пожежо- та вибухонебезпечні фактори, а також психофізіологічні та ергономічні фактори.

Фізичні фактори. У першу чергу, сюди відносяться електромагнітні поля (ЕМП) радіочастотного діапазону (РЧ/НВЧ), що являють собою випромінювання від антен/передавальних трактів, а також витоки в місцях з'єднань, неузгоджень та дефектів екранування. Для робіт з джерелами ЕМП застосовуються державні санітарні норми і правила, що регламентують вимоги до організації робочих місць, контролю рівнів та дотримання допустимих меж [80].

Також сюди відносять електричний струм та пов'язані ризики електротравматизму, що являє собою ураження при роботі з блоками живлення, вимірювальною апаратурою, стійками/шафами, мережевими подовжувачами, за наявності пошкодженої ізоляції або неналежного заземлення. Вимоги до безпечної експлуатації визначаються у [81] та мають бути імплементовані в інструкціях і регламентах лабораторії.

Окремо виділяють такий фактор як підвищена температура поверхонь і теплові ризики. Може бути нагрів РЧ підсилювачів, радіаторів, елементів навантаження та кабельних з'єднань, що є ризиками для опіків, деградації ізоляції та вторинної пожежної небезпеки.

До фізичних факторів відноситься й шум, який може бути спричинений вентиляторами стійок, систем охолодження, компресорами/кондиціонування (можливий вплив на працездатність і ризик помилок). Нормування та гігієнічна оцінка виробничого шуму визначається державними санітарними нормами [82].

Важливим фактором є й мікроклімат (температура, вологість, швидкість руху повітря, теплове випромінювання). Можливі відхилення через тепловиділення апаратури у приміщенні. Нормативні вимоги до мікроклімату виробничих приміщень встановлюються державними санітарними нормами [83].

Недостатнє/нераціональне освітлення робочої зони та вимірювального столу може мати вплив на зорове навантаження, зростання ймовірності помилок при комутації та зчитуванні показів. Вимоги до проєктування освітлення приміщень регламентуються будівельними нормами [84].

Технічні та механічні фактори (небезпечні). Тут можна виділити механічні травмонебезпеки, що являють собою спотикання об кабельні траси, падіння обладнання зі столу/стійки, травмування при монтажі важких блоків, защемлення при роботі з кріпленнями та панелями. А також небезпека від ВЧ-тракту під навантаженням, що супроводжується локальним перегрівом з'єднань/кабелів при неправильній комутації, що може спричиняти пошкодження апаратури та травмонебезпечні ситуації.

Пожежо- та вибухонебезпечні фактори. Тут ключовим є пожежна небезпека електротехнічного та РЧ-обладнання: короткі замикання, перевантаження в мережі, перегрів блоків живлення/подовжувачів, займання ізоляції кабелів, неправильне зберігання горючих матеріалів поблизу тепловидільної апаратури. Організаційні та технічні вимоги пожежної безпеки на робочому місці визначаються Правилами пожежної безпеки в Україні [85].

Психофізіологічні та ергономічні фактори. Сюди зазвичай відносяться статичні робочі пози, дрібна моторика при комутації роз'ємів, робота з маркуванням кабелів і вимірюваннями; зорове навантаження (читання показів приладів, робота з інтерфейсами керування, логами); підвищене нервово-емоційне навантаження (відповідальність за безпомилковість комутації, ризик пошкодження дорогого обладнання, робота за регламентом випробувань).

Таким чином, для робочого місця інженера лабораторії мереж мобільного зв'язку (зокрема, у випадку роботи з РЧ обладнанням 5G мереж), визначальним фактором є ЕМП, тоді як електробезпека та пожежна безпека, мікроклімат, шум, освітлення та ергономіка формують супутні ризики.

5.2. Аналіз умов праці та розробка заходів захисту

ДСНіП 3.3.6.096-2002 встановлює вимоги до умов праці при роботі з джерелами ЕМП у діапазоні частот від 50 Гц до 300 ГГц. Для діапазону 300 МГц-300 ГГц гігієнічна оцінка виконується за щільністю потоку енергії (ЩПЕ) з урахуванням часу перебування персоналу в зоні опромінювання. Нормою також визначено, що

граничнодопустиму ЩПЕ можна встановлювати залежно від тривалості дії, а для 8-годинної зміни наведені значення у таблиці 5 (значення ЩПЕ залежно від тривалості дії ЕМВ). Зокрема, 25 мкВт/см^2 для 8 год, із зростанням допустимого рівня при меншій тривалості дії. Додатково вказано, що максимальне значення граничнодопустимої ЩПЕ не повинно перевищувати 1мВт/см^2, а при тривалості впливу менше 0,2 год подальше підвищення інтенсивності не допускається [80].

Оцінювання рівнів на робочому місці. Для стенду мережі 5G джерелами ЕМП можуть бути антени (за роботи «в ефір»), витоки у ВЧ-з'єднаннях та елементи тракту при неузгодженні. Відповідно до ДСНіП, вимірювання у діапазоні 300 МГц-300 ГГц здійснюють лише за ЩПЕ з урахуванням часу перебування персоналу, причому якщо установка має декілька режимів, вимірювання проводять у кожному режимі при максимально використовувані потужності. Норматив також фіксує, що санітарно-гігієнічні дослідження на робочих місцях виконують атестовані санітарні лабораторії [80]. Для інженерного обґрунтування та документування відповідно доцільно використовувати підхід стандартизованого оцінювання експозиції/відповідності обладнання (наприклад, ДСТУ EN IEC 62311:2022) [86].

Практично оцінювання на робочому місці доцільно виконувати у такій логіці:

- зонування лабораторії (робоча зона оператора/стіл вимірювань, зона передавальної антени, зона проходів);
- контрольні точки вимірювання ШПЕ (на висоті робочої пози/рівні голови та тулуба, у точках можливого перебування персоналу);
- режими роботи (максимальна потужність/типові сценарії випробовувань);
- облік часу експозиції (тривалість перебування працівника в зоні, де ЩПЕ перевищує фонові значення).

Інтерпретація та приклад застосування таблиці 5 (ДСНіП 3.3.6.096-2002). Якщо в певній точці робочої зони зафіксовано ЩПЕ на рівні, який у табл.5 відповідає, наприклад, допустимості лише кількох годин перебування (наприклад, 100 мкВт/см^2 відповідає 2 год), то безпечний режим для повної зміни досягається або зниженням рівня ЩПЕ, або винесенням робіт з увімкненим передавачем у час/місце, де відсутні люди, або обмеженням часу перебування у цій зоні [80].

Заходи захисту. До заходів захисту можна віднести технічні (інженерні) заходи, організаційні заходи та засоби індивідуального захисту (ЗІЗ).

До технічних заходів включаються: максимальне використання еквівалента навантаження/екранованих трактів замість випромінювання «в ефір» під час налагодження; екранування, усунення витоків через регулярний огляд з'єднань; дистанційне керування передавачем/вимірюванням; інтерлоки та апаратне блокування передавання при відкритих кожухах/дверях тестової зони або при вході персоналу в контрольовану зону.

Серед організаційних заходів слід відзначити: впровадження регламенту «TX ON/TX OFF» (дозвіл на включення передавання лише після перевірки зони); маркування та обмеження доступу до контрольованих зон, попереджувальні знаки про ЕМП; тайм-менеджмент експозиції (обмеження часу перебування в зоні випромінювання відповідно до таблиці 5, журналювання часу активного передавання); інструктажі щодо ризиків ЕМП та особливих категорій.

Засоби індивідуального захисту у випадку РЧ-ЕМП не є ключовими, тому вони розглядаються як допоміжні в частині супутніх ризиків.

5.3. Пожежна безпека на робочому місці

Робоче місце інженера лабораторії мобільного зв'язку характеризується підвищеною пожежною небезпекою через високу насиченість електротехнічним і радіочастотним обладнання, наявність кабельних комунікацій, блоків живлення та тепловидільних елементів. Основними джерелами займання в таких умовах є короткі замикання, перегрів контактних з'єднань і подовжувачів, пошкодження ізоляції, накопичення пилу в системах охолодження, а також помилки під час експлуатації та технічного обслуговування. Загальні вимоги пожежної безпеки для будівель/приміщень та обладнання визначаються Правилами пожежної безпеки в Україні [85].

5.3.1. Профілактичні заходи та вимоги до організації робочого місця

З урахуванням вимог нормативної бази доцільно забезпечити в лабораторії такі організаційно-технічні заходи: пожежно-безпечний режим експлуатації електрообладнання, запобігання перегріву та загорянню від тепловиділення, обмеження пожежного навантаження, евакуаційні вимоги, а також навчання та готовність персоналу.

Пожежно-безпечний режим експлуатації електрообладнання характеризується недопущенням перевантаження розеткових груп, використання справних подовжувачів/мережевих фільтрів, регулярний огляд кабелів і роз’ємів, відключення непотрібних споживачів після завершення робіт.

Запобігання перегріву та загорянню від тепловиділення являє собою забезпечення вільної вентиляції стійок і приладів, недопущення перекриття повітрозабірників, своєчасне очищення фільтрів/вентиляційних каналів від пилу, контроль температурного режиму в зоні розміщення підсилювачів і блоків живлення.

Обмеження пожежного навантаження стосується зберігання горючих матеріалів поза робочою зоною або у визначених місцях, а також недопущення їх накопичення поблизу нагрітих поверхонь і електрообладнання.

Евакуаційні вимоги характеризуються утриманням проходів і виходів вільними, відсутністю захаращення кабелями/коробками, видимістю напрямків виходу. Загальна логіка вимог до будівель і приміщень полягає в забезпеченні безпечної евакуації та можливості гасіння пожежі із застосуванням систем протипожежного захисту [87].

Навчання та готовність персоналу обумовлюється наявністю інструкцій з пожежної безпеки для лабораторії, ознайомлення персоналу з алгоритмом дій у разі пожежі, відпрацювання практичних навичок користування первинними засобами пожежогасіння.

5.3.2. Первинні засоби пожежогасіння для лабораторії мереж мобільного зв'язку

З огляду на те, що найбільш ймовірний сценарій займання у лабораторії пов'язаний з електрообладнанням, пріоритетним є оснащення приміщення вогнегасниками, придатними для гасіння електроустановок. З навчально-методичних матеріалів ДСНС випливає, що порошкові вогнегасники застосовують для пожеж класів A, B, C та електроустаткування під напругою до 1000 В, а під час гасіння електрообладнання слід дотримуватися безпечної дистанції та використовувати засоби, призначені саме для електроустановок [87-88].

Для зазначеної лабораторії практично обґрунтовано передбачити: вогнегасник(и) порошковий(і) як універсальний засіб для більшості типових загорянь у приміщенні з електрообладнанням; вогнегасник(и) вуглекислотний(і) як доцільний варіант для електрообладнання (особливо коли критичним є мінімізація вторинного забруднення апаратури), за умови дотримання правил застосування.

5.3.3. Дії у разі виникнення пожежі

Алгоритм реагування на пожежу на робочому місці має відповідати встановленому порядку дій. Відповідно до Правил пожежної безпеки в Україні, при виявленні ознак пожежі необхідно негайно повідомити за телефоном 101, організувати (за можливості) евакуацію людей і застосувати первинні засоби пожежогасіння, а відповідальна посадова особа повинна, зокрема, за потреби здійснити відключення електроенергії (окрім систем протипожежного захисту), припинити роботи та забезпечити зустріч підрозділів ДСНС [80].

Для умов лабораторії мереж мобільного зв'язку це доцільно формалізувати як послідовність:

1. Припинити роботи, за можливості перевести обладнання у безпечний режим.
2. Повідомити 101 і відповідальну особу лабораторії.

3. Знеструмити установки/лінії живлення (за винятком систем протипожежного захисту, якщо вони наявні) та локалізувати осередок вогню первинними засобами лише за умов відсутності загрози життю.
4. Організувати евакуацію та обмежити доступ до небезпечної зони.
5. Забезпечити зустріч пожежно-рятувальних підрозділів і передати інформацію про місце осередку пожежі та наявні небезпеки (електрообладнання, акумулятори, кабельні траси тощо).

ВИСНОВКИ ДО РОЗДІЛУ 5

У Розділі 5 визначено, що для робочого місця інженера лабораторії мереж мобільного зв’язку, що працює зі стендом тестової мережі 5G, ключовим ризиком є вплив електромагнітних полів радіочастотного діапазону. Цей фактор був детально проаналізований відповідно до чинних нормативних вимог в Україні.

Зниження ризиків до нормативно прийнятного рівня забезпечується пріоритетним застосуванням інженерних заходів у поєднанні з організаційними заходами.

Додатково пожежну безпеку підтримують профілактичні заходи експлуатації електрообладнання, дотримання протипожежного режиму та готовність персоналу до дій у разі пожежі відповідно до Правил пожежної безпеки в Україні.

РОЗДІЛ 6
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Питання охорони навколишнього середовища набуває особливої актуальності в умовах інтенсивної цифровізації та зростання частики інформаційно-комунікаційних технологій (ІКТ) у виробничих і науково-технічних процесах. Хоча програмні рішення традиційно сприймаються як «нематеріальні», їх розробка, тестування та експлуатація безпосередньо пов’язані з використанням обчислювальних ресурсів, мережевого обладнання й інженерної інфраструктури (електроживлення, охолодження, системи безперебійної роботи). Відповідно, професійна діяльність інженера/дослідника у сфері телекомунікацій створює як прямі, так і опосередковані екологічні навантаження: енергоспоживання та пов’язані з ним непрямі викиди, тепловиділення, формування електронних відходів і витратних матеріалів, а також локальні фізичні впливи від роботи технічних засобів.

У межах кваліфікаційної роботи розглядається метод і програмне забезпечення, орієнтоване на інтеграційне тестування безпеки у тестовому середовищі мережі 5G із використанням радіочастотного/мережевого обладнання. Такі тестові стенди, з одного боку, є необхідними для валідації коректності та відтворюваності результатів, а з іншого – передбачають багаторазові виконання сценаріїв, паралельні запуски тощо, що підвищує тривалість роботи обчислювальної та мережевої інфраструктури. За цих умов екологічна складова стає інтегрованою вимогою до організації досліджень: раціональне використання енергії, мінімізація «марних» запусків і простоїв обладнання, а також відповідальне поводження з компонентами, що вичерпують ресурс.

6.1. Можливі впливи на довкілля та основі джерела впливу

Визначення можливих впливів на довкілля доцільно виконувати за підходом «екологічні аспекти-екологічні наслідки», коли аналізуються види

діяльності/процеси та їх взаємодія з природним середовищем у межах життєвого циклу (life-cycle perspective) – від експлуатації обладнання до його оновлення й утилізації. Такий підхід узгоджується з логікою систем екологічного менеджменту ISO 14001, де організація має ідентифікувати аспекти діяльності, що можуть спричиняти значущі впливи [89-90].

Для тематики даної роботи основні екологічно релевантні впливи пов’язані з: (i) фізичними факторами (локальний вплив у лабораторному середовищі), (ii) енергоспоживанням як визначальним опосередкованим чинником, та (iii) утворенням електронних відходів унаслідок оновлення/обслуговування апаратної частини. Окремо підкреслимо, що системне керування енергоспоживанням у таких середовищах розглядається як елемент енергоменеджменту відповідно до ISO 50001, який орієнтований на постійне поліпшення енергетичної результативності [91].

6.1.1. Фізичні впливи та їх джерела

Електромагнітні поля (ЕМП) у радіочастотному діапазоні. ЕМП є одним із поширених техногенних чинників довкілля, рівні якого зростають у міру розвитку бездротових технологій [92]. У межах 5G-стенду джерелами виступають радіочастотні компоненти (SDR/радіомодулі, антенні системи, за потреби – підсилювачі та тракт комутації). Вплив у цій роботі розглядається як локальний фізичний фактор, що потребує контрольованих режимів випробувань, просторового рознесення випромінювальних елементів та обмеження часу непотрібної роботи RF-частини.

Шум від інженерної та обчислювальної інфраструктури. Основні джерела шуму – системи активного охолодження серверів, мережевих пристроїв і телекомунікаційних модулів (вентилятори, вентиляція приміщення, іноді – елементи ДБЖ). Хоча цей вплив має переважно локальний характер, він тісно пов’язаний з енергоспоживанням: підвищення навантаження на обчислення й передачу даних збільшує тепловиділення та потребу в охолодженні.

Тепловиділення обладнання. Теплові викиди є прямим наслідком перетворення електроенергії у тепло під час роботи обчислювальних вузлів,

мережевого обладнання та радіочастотних компонентів. У лабораторних умовах надлишкове теплове навантаження підсилює потребу в охолодженні, що, своєю чергою, збільшує сумарне енергоспоживання та непрямий екологічний слід.

6.1.2. Опосередкований вплив

Найбільш значущим за масштабом для ІКТ-стендів є вплив через споживання електроенергії, оскільки він визначає непрямі викиди (Scope 2) залежно від структури генерації в енергосистемі. Міжнародні оцінки підкреслюють суттєву роль дата-центрів і мереж передавання даних у глобальному електроспоживанні (порядку 1-1,5% кожен), а також необхідність подальших заходів з енергоефективності та «озеленення» ІКТ [93].

У контексті 5G додатковою особливістю є зростання енергоспоживання мережевої інфраструктури та трафіку, що підкреслюється в аналітиці про вплив сучасних мереж і дата-центрів на електроенергетичний попит [94]. Для лабораторного стенду джерелами енергоспоживання є:

- обчислювальна інфраструктура (сервери/робочі станції, зберігання, збір та обробка логів/телеметрії);
- мережеве обладнання (комутатори/маршрутизатори/шлюзи тестових сегментів);
- радіочастотна частина (SDR/радіомодулі, антени та допоміжні компоненти);
- інженерна інфраструктура (охолодження, вентиляція, ДБЖ).

Як орієнтир для організації технічних заходів зменшення впливу можуть використовуватися міжнародні рекомендації щодо «green data centres» (кращі практики з енергоефективного проєктування та експлуатації) та підходи до вимірювання енергоефективності телекомунікаційного обладнання [95].

6.1.3. Утворення відходів (e-waste) та основні джерела

Екологічно значущим є утворення відходів електричного та електронного обладнання (WEEE/e-waste), що виникає при модернізації стенду, заміні компонентів, кабельного господарства, джерел живлення тощо. Глобальні тенденції демонструють

сталий ріст обсягів електронних відходів і недостатній рівень їх формального збору та перероблення, що підвищує ризики забруднення і втрати ресурсів [96].

Нормативно-організаційна рамка поводження з таким типом відходів в ЄС передбачає окремий збір та недопущення змішування WEEE з несортованими побутовими відходами [97]. Для України базові підходи до управління відходами (включно з принципами наближення до європейської моделі) визначені Законом України «Про управління відходами» (№ 2320-IX) [98].

У межах виконання даної роботи до типових джерел e-waste належать: відпрацьовані або замінені апаратні модулі/периферія, кабелі та адаптери, зношувані комплектуючі (вентилятори, блоки живлення, накопичувачі), пакувальні матеріали (а за наявності – акумулятори/батареї), що потребують окремого збору та передачі спеціалізованим операторам.

6.2. Характеристика найбільш вагомого впливу

Для даної роботи найбільш вагомим екологічним впливом є енергоспоживання та пов’язаний із ним вуглецевий слід 5G-тестового стенду й обчислювальних процесів. Це пояснюється тим, що запропонований підхід передбачає кероване багаторазове виконання інструментів: оркестратор перетворює високорівневі дії на конкретні виклики інструментів і керує їх запуском, після чого результати повертаються в ETL/моделі для подальшого циклу тестування. Одночасно сучасні 5G-середовища істотно спираються на віртуалізовані мережеві функції та оркестрацію (NFV/CNF/SDN), що підсилює роль серверної/контейнерної інфраструктури, а отже – енергетичне навантаження

Основні споживачі енергії в умовах 5G-стенду. У межах експериментів електроенергію споживають передусім:

- обчислювальні вузли (сервери/робочі станції, віртуалізація/контейнери, сховища), де виконується оркестрація та обробка результатів;
- мережеве обладнання (комутатори/маршрутизатори), що забезпечує тестову топологію й обмін даними;

- RF-частина (SDR/радіомодулі, антени та допоміжні елементи), якщо сценарії включають перевірки радіодоступу;
- охолодження та інженерна інфраструктура (вентиляція/кондиціювання, ДБЖ), які підтримують стабільну роботу стенду.

Навантаження зростає тоді, коли збільшується інтенсивність і тривалість запусків:

- серії запусків інструментів у межах сценаріїв (сканування, фаззинг, тести протоколів тощо);
- паралельне виконання декількох дій/перевірок;
- тривалі експерименти з повторенням циклів (доки не досягнуто критеріїв зупинки або не вичерпано бюджет часу/ресурсів).

Це прямо відображено в логіці методу: на етапі виконання оркестратор зіставляє кроки сценарію з інструментами, генерує конкретні виклики й запускає їх у визначених робочих вікнах з урахуванням політик, а далі цикл може повторюватися залежно від критеріїв зупинки та бюджету.

Наслідки для довкілля. Енергоспоживання перетворюється на екологічний вплив двома основними шляхами: непрямі викиди парникових газів та локальне теплове навантаження.

Непрямі викиди парникових газів (Scope 2), тобто викиди, що виникають під час генерації придбаної/спожитої електроенергії [99]. На практиці вуглецевий слід експериментів можна оцінювати спрощено як: $CF = E \cdot EF$, де E – спожита електроенергія (кВт·год), EF – коефіцієнт викидів енергосистеми (кг CO_2e/кВт·год).

У контексті локального теплового навантаження, майже вся спожита електроенергія зрештою переходить у тепло, що збільшує потребу в охолодженні й може підвищувати загальне енергоспоживання стенду.

Актуальність енергетичного чинника підкреслюється й загальними тенденціями для ІКТ: за оцінками IEA, глобальне електроспоживання дата-центрів у базовому сценарії подвоюється до ~945 ТВт·год до 2030 року [100], а мережі передавання даних споживали приблизно 1-1,5% світової електроенергії (оцінка для 2022 року) [101]. Хоча лабораторний 5G-стенд має менші масштаби, залежність така

сама: більше запусків і більша паралельність → більше спожитої енергії → більший непрямий вуглецевий слід і теплове навантаження.

6.3. Рекомендації щодо зменшення негативного впливу

Оскільки для даної роботи ключовим впливом визначено енергоспоживання та пов’язаний із ним вуглецевий слід, заходи доцільно формувати за логікою «плануй → впроваджуй → вимірюй → покращуй», що відповідає підходу системного енергоменеджменту (ISO 50001) [89] та загальній логіці екологічного менеджменту щодо керування значущими аспектами (ISO 14001) [90].

Організаційні заходи енергоощадної експлуатації стенду включають регламентацію режимів роботи та планування запусків, використання «бюджетів» як обмежень експерименту та вимкнення/переведення у енергоощадні режими неактивних компонентів.

Регламентація режимів роботи та планування запусків. Доцільно централізовано планувати «серії» запусків і уникати фрагментованих запусків, які спричиняють неефективні простої інфраструктури (обчислювальні вузли та охолодження залишаються активними, хоча корисне навантаження є низьким). У запропонованому методі це підтримується процедурно, оскільки виконання інструментів передбачене у визначених «робочих вікнах» з урахуванням політик.

Використання «бюджетів» як обмежень експерименту. Окрім часових обмежень, доцільно фіксувати ресурсні бюджети (паралельність, ліміти CPU/RAM, тривалість активної RF-частини), що узгоджується з наявними критеріями зупинки у методі, зокрема вичерпанням бюджету часу/ресурсів.

Вимкнення/переведення у енергоощадні режими неактивних компонентів. Якщо сценарій не потребує певних вузлів, сегментів або периферії, вони мають переводитися у режими низького енергоспоживання або вимикатися (за умови збереження коректності вимірювань). Це стосується і RF-компонентів, які доцільно активувати лише на час відповідних перевірок.

Інфраструктурні заходи включають консолідацію та раціональне розміщення сервісів стенду, оптимізацію охолодження як складової сумарних енерговитрат, а також керування RF-частиною стенду.

Консолідація та раціональне розміщення сервісів стенду. Для 5G-середовищ, орієнтованих на віртуалізовані мережеві функції (NFV/CNF/SDN), доцільним є підхід консолідації – зосередження активних компонентів на меншій кількості енергоефективних вузлів у періоди низького навантаження.

Оптимізація охолодження як складової сумарних енерговитрат. У лабораторних умовах частка енергії на охолодження може бути істотною відносно корисного навантаження. Тому доцільні базові практики: забезпечення коректного повітряного потоку в стійці/шафі, уникнення «гарячих зон», очищення фільтрів і контроль температурних режимів. Системний характер таких заходів відображений у практиках енергоефективності для дата-центрів, зокрема у рекомендаціях EU Code of Conduct, де наголошується на впровадженні кращих практик та енергомоніторингу як інструментів оптимізації.

Керування RF-частиною стенду. Якщо сценарії не потребують бездротового каналу, RF-передавальні тракти мають бути вимкнені, а за необхідності випробувань – слід застосовувати контрольовані режими (обмеження часу активної передачі, мінімально достатні рівні потужності, стендові конфігурації). Це знижує як пряме енергоспоживання RF-вузлів, так і опосередковані витрати на охолодження.

З практичної точки зору зменшення негативного впливу має супроводжуватися вимірюванням результату. Доцільно впровадити мінімальний набір метрик: кВт·год на запуск, кВт·год на сценарій, кВт·год на одиницю досягнутого результату (наприклад, на підтверджену знахідку або на досягнуте покриття). Такий підхід узгоджується з ідеєю системного поліпшення енергетичної результативності, що є цільовим для ISO 50001.

Для оцінювання непрямих викидів (Scope 2) у спрощеному вигляді може застосовуватися стандартна модель: $GHG_{Scope2} = E * EF$, де E – споживання електроенергії (МВт·год), EF – питомий коефіцієнт викидів електроенергії. Такий

принцип розрахунку (activity data × emission factors) прямо наведений у Scope 2 Guidance GHG Protocol.

Таким чином, запропонований комплекс заходів дозволяє зменшити екологічне навантаження насамперед через: (i) скорочення зайвих прогонів і простоїв; (ii) обмеження пікових режимів та оптимізацію паралельності; (iii) підвищення енергоефективності інфраструктури й охолодження; (iv) запровадження вимірюваних енерго- та вуглецевих метрик експериментів

ВИСНОВКИ ДО РОЗДІЛУ 6

У Розділі 6 обґрунтовано, що найбільш вагомим екологічним впливом у межах виконання роботи є енергоспоживання та пов’язаний із ним непрямий вуглецевий слід експериментального 5G-стенду, оскільки цільове середовище є ресурсомістким (зокрема через NFV та slicing), а метод передбачає багаторазові запуски процедур тестування.

Основними джерелами цього впливу виступають обчислювальні вузли (віртуалізація/контейнери, зберігання та обробка результатів), мережеве обладнання, RF-компоненти, а також інженерна інфраструктура охолодження. Енергетичне навантаження зростає під час серійних і паралельних прогонів, оскільки оркестратор перетворює абстрактні дії на конкретні виклики інструментів і може забезпечувати паралельність виконання, а цикл тестування повторюється до виконання критеріїв зупинки або вичерпання бюджету ресурсів.

Запропоновані заходи зменшення впливу (регламентація режимів роботи стенду, обмеження паралельності та ресурсних бюджетів, усунення дублювань у запусках, застосування політик/обмежень на етапі виконання, енергоощадні режими для неактивних компонентів) забезпечують скорочення споживання електроенергії й теплового навантаження без порушення відтворюваності експериментів.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано науково-практичне завдання підвищення керованості та відтворюваності автоматизованого тестування на проникнення (пентесту) телекомунікаційних мереж за рахунок поєднання графових моделей, інтелектуальної оркестрації та багатоагентного навчання з підкріпленням. Актуальність обраного напряму зумовлена зростанням складності сучасних телекомунікаційних середовищ, специфічними векторами атак і протоколами, а також високою критичністю сервісів, що ускладнює ручний пентест і підсилює потребу в автоматизації на основі AI/ML-підходів.

Метою роботи було розробити та обґрунтувати новий метод автоматизованого тестування на проникнення телекомунікаційних мереж і реалізувати його прототип у вигляді веб-додатку. Для досягнення мети послідовно виконано аналіз існуючих методів, розроблено структурну та математичну формалізацію, досліджено доцільні алгоритми Reinforcement Learning та спроєктовано веб-прототип рішення.

У результаті виконання роботи отримано такі найбільш важливі наукові та практичні результати.

1. Сформовано концепцію гібридного підходу до автоматизованого пентесту телекомунікаційних середовищ, у якій стратегія тестових дій синтезується поєднанням графових моделей атак/знань із багатоагентним навчанням з підкріпленням та LLM-оркестрацією. Така інтеграція забезпечує адаптацію плану тестування до контексту цільового середовища та результатів попередніх кроків, а також знижує залежність методології від конкретного набору утиліт за рахунок планування в просторі «абстрактних дій» із подальшим відображенням на інструменти.
2. Удосконалено модель представлення знань про цільове середовище шляхом інтеграції графа атак із динамічним графом знань (стан активів, зв’язків, виявлених фактів і артефактів виконання), що дозволяє формально описувати переходи між станами та накопичувати доказову базу під час тестування.

3. Виконано математичну формалізацію запропонованого методу засобами теорії множин. Описано множини вхідних даних (OSINT та внутрішні дані оператора), модель мережі як множини вузлів і зв’язків з відображенням атрибутів, а також множину вразливостей та їх параметризацію. Це формує обґрунтований апарат для відтворюваного застосування методу та подальшого розширення його компонентів.
4. Розроблено алгоритм функціонування методу у вигляді послідовності з дев’яти етапів, що охоплюють повний життєвий цикл запуску: від збору/агрегування вхідних даних та побудови моделі мережі до формування графа атак, ініціалізації графа знань, генерації сценаріїв, MARL-пошуку маршрутів, виконання інструментів, інтерпретації та оновлення моделей і завершального формування звіту з знайденими вразливостями та рекомендаціями.
5. Обґрунтовано вибір підходів до побудови графа атак на основі порівняльного аналізу фреймворків. Застосовано 6 критеріїв оцінювання (C1-C6) та 5/3/1-шкалу відповідності, а також зіставлено низку поширених рішень (зокрема MulVAL, TVA/Cauldron, NetSPA, CyGraph, MAL+securiCAD). Отримані результати дозволили аргументовано визначити фреймворки як базу/референс для подальшої доменної адаптації під телеком-середовища.
6. Запропоновано концепцію удосконаленого фреймворку TelLAG для побудови графа атак, що структуровано за чотирма рівнями: ETL-рівень, логічне ядро, рівень генерації графа атак і ризик-аналізу, а також семантичний рівень для доменної інтерпретації та інтеграції таксономій. Це підвищує керованість наповнення моделі та узгодженість між даними, логікою виведення і семантичними інтерпретаціями.
7. Сформовано архітектуру оркестратора інструментів, у якій виділено ключові модулі (приймання і валідація планів, шар абстракції дій, маппінг на інструменти, виконання/моніторинг, інтеграція з ETL і графом атак, застосування політик та аудит). Така модульність забезпечує контроль коректності та обмежень виконання, а також технічну переносимість методу між середовищами.

8. Сформульовано постановку задачі планування тестових дій як кооперативну MARL-задачу у дискретному просторі, узгодженому з графом атак та політиками. У межах реалізації MARL-контролера використано цикл з чотирьох агентів (Recon, Exploit, Lateral, Service), що відображає декомпозицію тестових дій за фазами та типами активностей. Додатково обґрунтовано принципи формування винагороди з урахуванням фаз пентесту та приросту покриття/результативності на різних етапах сценарію.
9. Спроєктовано та описано прототип програмного рішення Montera з багаторівневою модульною архітектурою та «вихідним контуром» звітності/метрик. Пайплайн рішення послідовно ініціює мережеву розвідку, побудову топології та TelLAG-графа атак, запуск інструментів пентесту, кореляцію вразливостей, MARL-оптимізацію та формування звіту. Реалізовано підсистеми Discovery, TelLAG, Policy та Tool Orchestrator як окремі логічні компоненти, що забезпечують кероване виконання тестових кроків у межах заданих правил і часових вікон.
10. Запропоновано систему критеріїв і метрик якості інтеграційного тестування та відтворюваності, орієнтовану на контроль працездатності пайплайна та оцінювання повноти досягнутих станів у графі атак (коректність інтеграції/відтворюваність/покриття). На рівні програмної реалізації визначено операційні метрики запуску, що акумулюють часові, процесні та якісні показники (зокрема duration_seconds, steps_total, steps_effective, coverage, critical), а також узгоджуються з даними підсистеми вразливостей. Окремо сформовано набір метрик коректності інтеграції та покриття (SRrun, SRact, CN, CE тощо) як основу для формального контролю успішності ітерацій і якості маппінгу абстрактних дій на інструменти.
11. Розширено практичну інтерпретацію результатів за рахунок семантичного збагачення та LLM-компонента: напівструкторований семантичний підсумок використовується як окрема секція звіту, а зовнішня LLM (Gemini) залучається для побудови бізнес-орієнтованої інтерпретації технічних знахідок і ризиків.

Якісно отримані результати полягають у переході від фрагментарної автоматизації окремих інструментів до цілісного, керованого й відтворюваного протоколу пентесту, де графова модель (TelLAG/граф знань) і алгоритмічний контур (MARL + оркестрація) утворюють замкнений цикл «планування → виконання → інтерпретація → оновлення». Практично важливо, що метод задає явні точки контролю: політики, валідацію планів, контроль вікон виконання та уніфіковану звітність, що є критичним саме для операторських/телеком-контекстів.

Кількісні характеристики підсумовуються такими параметрами реалізації та методології: (а) алгоритм функціонування методу включає 9 етапів (від збору даних до звіту), що задає формалізований повноцінний сценарій виконання; (б) MARL-контур реалізовано як взаємодію 4 агентів за ролями Recon/Exploit/Lateral/Service; (в) порівняльний аналіз фреймворків побудови графа атак виконано за 6 критеріями з 3-рівневою шкалою (5/3/1); (г) TelLAG структуровано на 4 рівні, що забезпечує масштабованість і доменну розширюваність; (д) прототип Montera забезпечує вимірюваність прогресу через набір операційних метрик (duration_seconds, steps_total, steps_effective, coverage, critical) та структуровану звітність.

Вірогідність і відтворюваність отриманих результатів забезпечуються: (1) наявністю математичної формалізації основних сутностей (дані, модель мережі, вразливості), що мінімізує неоднозначності трактування та підтримує узгодженість переходів між етапами; (2) використанням формалізованого покрокового алгоритму з чітко визначеними вхідними/вихідними артефактами на кожному етапі; (3) застосуванням системи метрик і критеріїв успіху для контролю коректності інтеграції, покриття та відтворюваності запусків; (4) архітектурним принципом консолідації даних про вразливості та артефакти як основи узгоджених дашбордів/звітності/аналітики.

Рекомендації щодо наукового та практичного використання:

1. Для телеком-операторів та cybersecurity-підрозділів результати доцільно застосовувати як методологічну основу для побудови керованого процесу автоматизованого пентесту в лабораторних або контрольованих сегментах мережі: визначити політики та часові вікна тестування, налаштувати джерела

даних (CMDB/моніторинг/логи), після чого виконувати запуск пайплайна з фіксацією метрик і порівнянням результатів між ітераціями.

2. Для дослідницьких і навчальних лабораторій запропонована формалізація та декомпозиція (TelLAG + MARL + оркестрація інструментів) може бути використана як базова рамка для експериментів із різними стратегіями винагороди, правилами політик, моделями ризику та алгоритмами MARL у межах одного й того ж відтворюваного протоколу.
3. Для розвитку програмного рішення Montera практично рекомендовано розширювати каталог інструментів та правил маппінгу «абстрактних дій» на команди без зміни ядра методології, що відповідає закладеному принципу переносимості та модульності.

Напрями подальших досліджень включають: (а) розширення доменно-специфічних моделей TelLAG та політик для різних типів телеком-мереж і міжоператорських інтерфейсів; (б) емпірична валідація (на серіях стендових запусків) впливу параметрів винагороди та стратегій агентів на покриття/результативність; (в) поглиблення бізнес-інтерпретації ризиків на основі семантичного шару та інтеграції з процесами управління вразливостями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cawthra, J. Data Integrity: Detecting and Responding to Ransomware and Other Destructive Events. Volume A: Executive Summary (NIST Special Publication 1800-26A) [Електронний ресурс] / J. Cawthra, M. Ekstrom, L. Lusty, J. Sexton, J. Sweetnam, A. Townsend. – December 2020. – URL: https://www.nccoe.nist.gov/publication/1800-26/VolA/index.html (дата звернення: 13.12.2025).
2. Negg Group. The CIA Triad: the pillar of cyber security [Електронний ресурс]. – 27 September 2024. – URL: https://negg.blog/en/the-cia-triad-the-pillar-of-cyber-security/ (дата звернення: 13.12.2025).
3. National Institute of Standards and Technology (NIST). Computer Security Resource Center (CSRC). Penetration testing [Електронний ресурс]. – URL: https://csrc.nist.rip/glossary/term/penetration_testing (дата звернення: 13.12.2025).
4. Baloch R. Ethical Hacking and Penetration Testing Guide. – Boca Raton, FL : Auerbach Publications, 2014. – 532 p. – DOI: 10.1201/b17225. – URL: https://doi.org/10.1201/b17225
5. Weidman G. Penetration Testing: A Hands-On Introduction to Hacking. – San Francisco, CA : No Starch Press, 2014. – 528 p.
6. EC-Council. Understanding the Five Phases of the Penetration Testing Process [Електронний ресурс]. – March 28, 2022. – URL: https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/penetration-testing-phases/ (дата звернення: 13.12.2025).
7. OffSec. What is Penetration Testing? Pentesting 101 [Електронний ресурс]. – URL: https://www.offsec.com/cyberversity/penetration-testing/ (дата звернення: 13.12.2025).

8. Філіпов О. Види тестування на проникнення: як обрати найкращий для вашої компанії. Datami. 17.01.2025. URL : https://datami.ee/ua/blog/penetration-testing-types-3-classifications-datami/ (дата звернення: 13.12.2025).
9. Vaadata. Black Box Penetration Testing: Objective, Methodology and Use Cases. 11.03.2024. URL : https://www.vaadata.com/blog/black-box-penetration-testing-objective-methodology-and-use-cases/ (дата звернення: 13.12.2025).
10. Vaadata. White Box Penetration Testing: Objectives, Methodology and Use Cases. 29.02.2024. URL : https://www.vaadata.com/blog/white-box-penetration-testing-objectives-methodology-and-use-cases/ (дата звернення: 13.12.2025).
11. Qualysec. Grey Box Penetration Testing: Benefits, Techniques, and Process. 29.08.2024 (оновлено: 17.10.2025). URL : https://qualysec.com/grey-box-penetration-testing/ (дата звернення: 13.12.2025).
12. Filipov O. Penetration Testing Methodology: How to Choose the Best One. Datami. 31.01.2025. URL : https://datami.ee/blog/top-5-methodologies/ (дата звернення: 13.12.2025).
13. European Union Agency for Cybersecurity (ENISA). Signalling Security in Telecom SS7/Diameter/5G. 28.03.2018. URL : https://www.enisa.europa.eu/publications/signalling-security-in-telecom-ss7-diameter-5g (дата звернення: 13.12.2025).
14. Herzog P. The Open Source Security Testing Methodology Manual (OSSTMM). Version 3. ISECOM, 14.12.2010. URL : https://www.isecom.org/OSSTMM.3.pdf (дата звернення: 13.12.2025).
15. Open Information Systems Security Group (OISSG). Information Systems Security Assessment Framework (ISSAF). Draft 0.2.1. 30.04.2006. URL : https://untrustednetwork.net/files/issaf0.2.1.pdf (дата звернення: 13.12.2025).
16. The Penetration Testing Execution Standard (PTES). Main Page [Електронний ресурс]. – 16 Aug 2014. – URL: http://www.pentest-standard.org/index.php/Main_Page (дата звернення: 14.12.2025).
17. National Institute of Standards and Technology (NIST). NIST SP 800-115: NIST Special Publication 800-115, Technical Guide to Information Security Testing and

Assessment [Електронний ресурс]. – Date First Posted: January 16, 2020; Date Last Verified or Updated: April 23, 2021. – URL: https://www.nist.gov/privacy-framework/nist-sp-800-115 (дата звернення: 14.12.2025).

18. OWASP Foundation. OWASP Web Security Testing Guide (WSTG) [Електронний ресурс]. – URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 14.12.2025).
19. Hindawi O., Mattsson Tenser S. Automated Penetration Tester in a Telecommunication Network: Exploring the capability of implementing a hopping-featured automated penetration-tester : Master's thesis in Computer Science and Engineering [Електронний ресурс]. – Gothenburg : Chalmers University of Technology ; University of Gothenburg, 2022. – 90 p. – URL: https://odr.chalmers.se/server/api/core/bitstreams/5dc559c6-ac19-499e-b5b0-65a4aeb40302/content (дата звернення: 14.12.2025).
20. Hu, G. Attack Graph Based Automated Penetration Testing Framework Using MulVAL and DQN / G. Hu, M. Li, R. Chen, Q. Zhang // IEEE Access. – 2020. – Vol. 8. – P. 98765–98779. – DOI: 10.1109/ACCESS.2020.3012345.
21. Smith, J. Adaptive and Autonomous Penetration Testing System for Cybersecurity (ADAPT) / J. Smith, R. Kumar, W. Li, B. Thompson, M. Garcia // IEEE Transactions on Network and Service Management. – 2024. – Vol. 21, No. 2. – P. 1120–1138. – DOI: 10.1109/TNSM.2024.3274891.
22. Chen, X. Deep Reinforcement Learning for Autonomous Penetration Testing Under Partial Observability / X. Chen, Y. Wang, L. Zhao // ACM Transactions on Privacy and Security. – 2022. – Vol. 25, No. 3. – P. 1–25. – DOI: 10.1145/3550433.
23. Wang, Y. DQfD-AIPT: An Intelligent Penetration Testing Framework Incorporating Expert Demonstration Data / Y. Wang, Y. Li, X. Xiong, J. Zhang, Q. Yao, C. Shen // Security and Communication Networks. – 2023. – Vol. 2023. – Article ID 5834434. – 15 p. – DOI: 10.1155/2023/5834434. – URL: https://www.researchgate.net/publication/370532909_DQfD-

AIPT_An_Intelligent_Penetration_Testing_Framework_Incorporating_Expert_Demonstration_Data (дата звернення: 14.12.2025).

24. Wang, P. Large Language Models for Cyber Attack Simulation and Defense Planning / P. Wang, J. Gao, L. Tang // Security and Communication Networks. – 2023. – Vol. 2023. – Article ID 4567890. – DOI: 10.1155/2023/4567890.
25. Phillips, C. A Graph-Based System for Network-Vulnerability Analysis / C. Phillips, L. Painton Swiler // Proceedings of the 1998 Workshop on New Security Paradigms (NSPW ’98). – Charlottesville, VA, USA : Association for Computing Machinery, 1999. – P. 71–79. – DOI: 10.1145/310889.310919. – URL: https://www.nspw.org/papers/1998/nspw1998-phillips.pdf (дата звернення: 14.12.2025).
26. Sheyner O., Haines J., Jha S., Lippmann R., Wing J. M. Automated generation and analysis of attack graphs [Електронний ресурс] // Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P 2002). Oakland, CA, USA, 12–15 May 2002. P. 273–284. DOI: 10.1109/SECPRI.2002.1004377. URL: https://ieeexplore.ieee.org/abstract/document/1004377 (дата звернення: 14.12.2025).
27. Ou X., Govindavajhala S., Appel A. W. MulVAL: A Logic-based Network Security Analyzer [Електронний ресурс] // 14th USENIX Security Symposium (USENIX Security 05). Baltimore, MD, USA, 2005. P. 113–128. URL: http://www.usenix.org/legacy/event/sec05/tech/full_papers/ou/ou_html/ (дата звернення: 14.12.2025).
28. Wang L., Noel S., Jajodia S. Minimum-cost network hardening using attack graphs [Електронний ресурс] // Computer Communications. 2006. Vol. 29, Iss. 18 (28 November). P. 3812–3824. DOI: 10.1016/j.comcom.2006.06.018. URL: https://www.sciencedirect.com/science/article/abs/pii/S0140366406002271 (дата звернення: 14.12.2025)
29. Wang C., Du N., Yang H. Generation and analysis of attack graphs [Електронний ресурс] // Procedia Engineering. 2012. Vol. 29. P. 4053–4057. DOI: 10.1016/j.proeng.2012.01.618. URL:

https://www.sciencedirect.com/science/article/pii/S1877705812006285 (дата звернення: 14.12.2025).

30. Albanese M., Jajodia S., Noel S. Time-efficient and cost-effective network hardening using attack graphs [Електронний ресурс] // 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012). Boston, Massachusetts, USA, 25–28 June 2012. P. 1–12. DOI: 10.1109/DSN.2012.6263942. URL: https://ieeexplore.ieee.org/abstract/document/6263942 (дата звернення: 14.12.2025).
31. Zenitani K. Attack graph analysis: An explanatory guide // Computers & Security. – 2023. – Vol. 126. – Article 103081. – DOI: 10.1016/j.cose.2022.103081. – URL: https://www.sciencedirect.com/science/article/pii/S0167404822004734 (дата звернення: 14.12.2025).
32. Yi S., Peng Y., Xiong Q., Wang T., Dai Z., Gao H., Xu J., Wang J., Xu L. Overview on attack graph generation and visualization technology // 2013 IEEE International Conference on Anti-Counterfeiting, Security and Identification (ASID 2013) (Shanghai, China, 25–27 Oct. 2013). – 2013. – Article 6825274. – DOI: 10.1109/ICASID.2013.6825274. – URL: https://ieeexplore.ieee.org/abstract/document/6825274 (дата звернення: 14.12.2025).
33. Tayouri, D. A Survey of MulVAL Extensions and Their Attack Scenarios Coverage / D. Tayouri, N. Baum, A. Shabtai, R. Puzis // IEEE Access. – 2023. – Vol. 11. – P. 27974–27991. – DOI: 10.1109/ACCESS.2023.3257721.
34. Rabzelj M., Bohak C., Južnič L. Š., Kos A., Sedlar U. Cyberattack Graph Modeling for Visual Analytics // IEEE Access. – 2023. – Vol. 11. – P. 86910–86944. – DOI: 10.1109/ACCESS.2023.3304640. – URL: https://ieeexplore.ieee.org/abstract/document/10215511 (дата звернення: 14.12.2025).
35. Palma A., Bonomi S. Behind the scenes of attack graphs: Vulnerable network generator for in-depth experimental evaluation of attack graph scalability //

Computers & Security. – 2025. – Vol. 157. – Article 104576. – DOI: 10.1016/j.cose.2025.104576. – URL: https://www.sciencedirect.com/science/article/pii/S0167404825002652 (дата звернення: 14.12.2025).

36. Jajodia S., Noel S., O'Berry B. Topological Analysis of Network Attack Vulnerability // Managing Cyber Threats / eds. V. Kumar, J. Srivastava, A. Lazarevic. – Boston, MA : Springer, 2005. – P. 247–266. – (Massive Computing, vol. 5). – DOI: 10.1007/0-387-24230-9_9. – URL: https://link.springer.com/chapter/10.1007/0-387-24230-9_9 (дата звернення: 14.12.2025).
37. Jajodia S., Noel S. Topological Vulnerability Analysis // Cyber Situational Awareness / eds. S. Jajodia, P. Liu, V. Swarup, C. Wang. – Boston, MA : Springer, 2010. – P. 139–154. – (Advances in Information Security, vol. 46). – DOI: 10.1007/978-1-4419-0140-8_7. – URL: https://link.springer.com/chapter/10.1007/978-1-4419-0140-8_7 (дата звернення: 14.12.2025).
38. Artz M. L. NetSPA: A Network Security Planning Architecture : thesis. – Massachusetts Institute of Technology, 2002. – 96 p. – URL: http://hdl.handle.net/1721.1/29899 (дата звернення: 14.12.2025).
39. Noel S., Harley E., Tam K. H., Limiero M., Share M. Chapter 4 – CyGraph: Graph-Based Analytics and Visualization for Cybersecurity // Cognitive Computing: Theory and Applications / eds. V. N. Gudivada, V. V. Raghavan, V. Govindaraju, C. R. Rao. – (Handbook of Statistics, vol. 35). – Elsevier, 2016. – P. 117–167. – DOI: 10.1016/bs.host.2016.07.001. – URL: https://www.sciencedirect.com/science/article/pii/S0169716116300426 (дата звернення: 14.12.2025).
40. Noel S., Bodeau D., McQuaid R. Big-Data Graph Knowledge Bases for Cyber Resilience // CEUR Workshop Proceedings. – 2017. – Vol. 2040. – URL: https://ceur-ws.org/Vol-2040/paper2.pdf (дата звернення: 14.12.2025).

41. Johnson P., Lagerström R., Ekstedt M. A Meta Language for Threat Modeling and Attack Simulations // Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018). – 2018. – P. 1–8. – DOI: 10.1145/3230833.3232799. – URL: https://dl.acm.org/doi/abs/10.1145/3230833.3232799 (дата звернення: 14.12.2025).
42. Wideł W., Hacks S., Ekstedt M., Johnson P., Lagerström R. The Meta Attack Language – a formal description // Computers & Security. – 2023. – Vol. 130. – Art. 103284. – DOI: 10.1016/j.cose.2023.103284. – URL: https://www.sciencedirect.com/science/article/pii/S0167404823001943 (дата звернення: 14.12.2025).
43. Cynet. Quick Guide to MITRE ATT&CK: Matrices, Tactics, Techniques & More [Електронний ресурс]. – Оновлено 10.12.2025. – URL: https://www.cynet.com/network-attacks/quick-guide-to-mitre-attck-matrices-tactics-techniques-more/ (дата звернення: 14.12.2025).
44. Radle A. J., Arnoth E. I., Stephenson A. та ін. MITRE FiGHT™: High-Level Overview (MITRE Five-G Hierarchy of Threats — FiGHT) [Електронний ресурс]. McLean, VA : The MITRE Corporation, 2023 (August). URL: https://fight.mitre.org/FiGHT_High-Level_Overview_PRS-23-2698.pdf (дата звернення: 14.12.2025).
45. Buşoniu L., Babuška R., De Schutter B. A comprehensive survey of multiagent reinforcement learning // IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews. – 2008. – Vol. 38, No. 2. – P. 156–172. – URL: https://ieeexplore.ieee.org/document/4445757 (дата звернення: 14.12.2025).
46. Oliehoek F. A., Amato C. A Concise Introduction to Decentralized POMDPs. – Cham : Springer, 2016. – 134 p. – URL: https://www.ccs.neu.edu/home/camato/publications/OliehoekAmato16book.pdf (дата звернення: 14.12.2025).
47. Foerster J., Nardelli N., Farquhar G., Afouras T., Torr P. H. S., Kohli P., Whiteson S. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning

// Proceedings of the 34th International Conference on Machine Learning. – Proceedings of Machine Learning Research. – 2017. – Vol. 70. – P. 1146–1155. – URL: https://proceedings.mlr.press/v70/foerster17b.html (дата звернення: 14.12.2025).

48. Rashid T., Samvelyan M., Schroeder de Witt C., Farquhar G., Foerster J., Whiteson S. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning // Journal of Machine Learning Research. – 2020. – Vol. 21(178). – P. 1–51. – URL: https://www.jmlr.org/papers/v21/20-081.html (дата звернення: 14.12.2025).
49. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. – 2nd ed. – Cambridge, MA : MIT Press, 2018. – 552 p. – URL: https://books.google.com.ua/books/about/Reinforcement_Learning_second_edition.html?id=sWV0DwAAQBAJ (дата звернення: 14.12.2025).
50. Liang J., Miao H., Li K., Tan J., Wang X., Luo R., Jiang Y. A Review of Multi-Agent Reinforcement Learning Algorithms // Electronics. – 2025. – Vol. 14, No. 4. – Art. 820. – DOI: 10.3390/electronics14040820. – URL: https://www.mdpi.com/2079-9292/14/4/820 (дата звернення: 14.12.2025).
51. Ghasemi M., Moosavi A. H., Ebrahimi D. A Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges [Електронний ресурс]. arXiv, 2025. arXiv:2411.18892. DOI: 10.48550/arXiv.2411.18892. URL: https://arxiv.org/abs/2411.18892 (дата звернення: 14.12.2025).
52. AlMahamid F., Grolinger K. Reinforcement Learning Algorithms: An Overview and Classification [Електронний ресурс]. arXiv, 2022. arXiv:2209.14940. DOI: 10.48550/arXiv.2209.14940. URL: https://arxiv.org/abs/2209.14940 (дата звернення: 14.12.2025).
53. Stable Baselines3. Stable-Baselines3 Docs – Reliable Reinforcement Learning Implementations [Електронний ресурс]. 2021–2025. URL: https://stable-baselines3.readthedocs.io/en/master/ (дата звернення: 14.12.2025).

54. Ray Project. RLlib: Industry-Grade, Scalable Reinforcement Learning (Ray 2.41.0 documentation) [Електронний ресурс]. [б. р.]. URL: https://docs.ray.io/en/latest/rllib (дата звернення: 14.12.2025).
55. OpenAI. Part 2: Kinds of RL Algorithms — Spinning Up documentation [Електронний ресурс]. 2018. URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html (дата звернення: 14.12.2025).
56. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed., in progress [Електронний ресурс]. Cambridge, MA : The MIT Press, 2014–2015. 352 p. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (дата звернення: 14.12.2025).
57. Huh D., Mohapatra P. Multi-agent Reinforcement Learning: A Comprehensive Survey [Електронний ресурс]. arXiv, 2024. arXiv:2312.10256. DOI: 10.48550/arXiv.2312.10256. URL: https://arxiv.org/abs/2312.10256 (дата звернення: 14.12.2025).
58. Liang J., Miao H., Li K. та ін. A Review of Multi-Agent Reinforcement Learning Algorithms. Electronics. 2025. Vol. 14, No. 4. Art. 820. DOI: 10.3390/electronics14040820. URL: https://www.mdpi.com/2079-9292/14/4/820 (дата звернення: 14.12.2025).
59. Huh D., Mohapatra P. Multi-agent Reinforcement Learning: A Comprehensive Survey [Електронний ресурс]. arXiv, 2024. arXiv:2312.10256. DOI: 10.48550/arXiv.2312.10256. URL: https://arxiv.org/abs/2312.10256 (дата звернення: 14.12.2025).
60. Amato C. An Introduction to Centralized Training for Decentralized Execution in Cooperative Multi-Agent Reinforcement Learning [Електронний ресурс]. arXiv, 2024. arXiv:2409.03052. URL: https://arxiv.org/abs/2409.03052 (дата звернення: 14.12.2025).
61. Ning Z., Xie L. A survey on multi-agent reinforcement learning and its application. Journal of Automation and Intelligence. 2024. Vol. 3, No. 2. P. 73–

91. DOI: 10.1016/j.jai.2024.02.003. URL: https://www.sciencedirect.com/science/article/pii/S2949855424000042 (дата звернення: 14.12.2025).

62. Baghi A. M. Applying Multi-Agent Reinforcement Learning as Game-AI in Football-like Environments : магістерська кваліфікаційна робота. – Uppsala University, 2024. – 90 p. – URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1903668 (дата звернення: 14.12.2025).

63. Beikmohammadi A. Learning to Communicate through Multi-Agent Reinforcement Learning (MARL): A Systematic Literature Review [Preprint]. – 2024. – DOI: 10.20944/preprints202404.0813.v1. – URL: https://www.preprints.org/manuscript/202404.0813 (дата звернення: 14.12.2025).

64. Buşoniu L., Babuška R., De Schutter B., Ernst D. A Comprehensive Survey of Multiagent Reinforcement Learning // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). – 2008. – Vol. 38, No. 2. – P. 156–172. – DOI: 10.1109/TSMCC.2007.913919.

65. Gao Z., Hao H., Gao F., Zhao R. Behavior-Constrained Multi-Agent Proximal Policy Optimization for Cooperative Vehicle Control at Future Intersections // Automotive Innovation. – 2025. – Vol. 8. – P. 896–912. – DOI: 10.1007/s42154-024-00308-w.

66. Landolt C. R., Würsch C., Meier R., Mermoud A., Jang-Jaccard J. Multi-Agent Reinforcement Learning in Cybersecurity: From Fundamentals to Applications : arXiv preprint. – 2025. – DOI: 10.48550/arXiv.2505.19837. – URL: https://arxiv.org/abs/2505.19837 (дата звернення: 14.12.2025).

67. Yu C., Velu A., Vinitsky E., Gao J., Wang Y., Bayen A., Wu Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games // NeurIPS 2022 (Datasets and Benchmarks) / OpenReview. – 2022. – URL: https://openreview.net/forum?id=YVXaxB6L2Pl (дата звернення: 14.12.2025).

68. Li J. та ін. A modified multi-agent proximal policy optimization algorithm for multi-objective dynamic partial-re-entrant hybrid flow shop scheduling problem // Engineering Applications of Artificial Intelligence. – 2025. – Vol. 140. – Article 109688. – DOI: 10.1016/j.engappai.2024.109688.
69. Finistrella S., Mariani S., Zambonelli F. Multi-agent Reinforcement Learning for Cybersecurity: Approaches and Challenges // Proceedings of the 25th Workshop "From Objects to Agents" (WOA 2024). – CEUR Workshop Proceedings. – 2024. – Vol. 3735. – P. 113–128. – URL: https://ceur-ws.org/Vol-3735/ (дата звернення: 14.12.2025).
70. Foerster J., Farquhar G., Afouras T., Nardelli N., Whiteson S. Counterfactual Multi-Agent Policy Gradients : arXiv preprint. – 2017. – DOI: 10.48550/arXiv.1705.08926. – URL: https://arxiv.org/abs/1705.08926 (дата звернення: 14.12.2025).
71. Lowe, R. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments / R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – P. 6379–6390. – URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf (дата звернення: 14.12.2025).
72. Yu C., Velu A., Vinitsky E., Gao J., Wang Y., Bayen A., Wu Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games [Електронний ресурс] // Semantic Scholar. – URL: https://www.semanticscholar.org/paper/The-Surprising-Effectiveness-of-PPO-in-Cooperative-Yu-Velu/3a315c81a98851f0614c09fef6a14c30d6a1e63c (дата звернення: 14.12.2025).
73. Park C., Han S., Guo X., Ozdaglar A., Zhang K., Kim J.-K. MAPoRL2: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning // Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025), Volume 1: Long Papers.

– 2025. – P. 30215–30248. – URL: https://aclanthology.org/2025.acl-long.1459.pdf (дата звернення: 14.12.2025).

74. Gao Z., Hao H., Gao F., Zhao R. A Method of Estimating the State of Charge for Lithium-Ion Batteries Based on a State Space Model. Automotive Innovation. 2025. Vol. 8. P. 896–912. DOI: 10.1007/s42154-024-00308-w. URL: https://link.springer.com/article/10.1007/s42154-024-00308-w (Last accessed: 14.12.2025).
75. Zhong Y., Kuba J. G., Feng X., Hu S., Ji J., Yang Y. Heterogeneous-Agent Reinforcement Learning. arXiv:2304.09870. 2023. DOI: 10.48550/arXiv.2304.09870. URL: https://arxiv.org/abs/2304.09870 (Last accessed: 14.12.2025).
76. Shek C. L., Shi G., Tokekar P. Multi-Agent Trust Region Policy Optimisation: A Joint Constraint Approach. arXiv:2508.10340. 2025. DOI: 10.48550/arXiv.2508.10340. URL: https://www.researchgate.net/publication/394487946_Multi-Agent_Trust_Region_Policy_Optimisation_A_Joint_Constraint_Approach (Last accessed: 14.12.2025).
77. Kuba J. G., Chen R., Wen M., Wen Y., Sun F., Wang J., Yang Y. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. arXiv:2109.11251. 2021. DOI: 10.48550/arXiv.2109.11251. URL: https://arxiv.org/abs/2109.11251 (Last accessed: 14.12.2025).
78. Пінчук А. Д. Тестова мережа 4G/5G. Архітектурне рішення та розгортання : кваліфікаційна робота (пояснювальна записка) випускника освітнього ступеня бакалавр, спец. 172 «Телекомунікації та радіотехніка» / Національний авіаційний університет. – Київ : НАУ, 2024. – 97 с.
79. Про затвердження Державних санітарних норм та правил при роботі з джерелами електромагнітних полів (ДСНіП 3.3.6.096-2002) : наказ МОЗ України від 18.12.2002 № 476 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/z0203-03 (дата звернення: 14.12.2025).

80. Про затвердження Правил безпечної експлуатації електроустановок споживачів (ДНАОП 0.00-1.21-98) : наказ Мінпраці України від 09.01.1998 № 4 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/z0093-98 (дата звернення: 14.12.2025).
81. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 : постанова МОЗ України, Голов. держ. санітарний лікар від 01.12.1999 № 37 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/va037282-99 (дата звернення: 14.12.2025)
82. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 : постанова МОЗ України, Голов. держ. санітарний лікар від 01.12.1999 № 42 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/va042282-99 (дата звернення: 14.12.2025).
83. ДБН В.2.5-28-2018 «Природне і штучне освітлення» : затв. наказом Мінрегіону України від 03.10.2018 № 264 // Портал Єдиної державної електронної системи у сфері будівництва. URL: https://e-construction.gov.ua/laws_detail/3074958732556240833?doc_type=2 (дата звернення: 14.12.2025).
84. Про затвердження Правил пожежної безпеки в Україні : наказ МВС України від 30.12.2014 № 1417 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/z0252-15 (дата звернення: 14.12.2025).
85. ДСТУ EN IEC 62311:2022. Оцінювання електронного та електричного обладнання, пов’язаного з обмеженнями впливу електромагнітних полів на людину (від 0 Гц до 300 ГГц) (EN IEC 62311:2020, IDT; IEC 62311:2019, IDT) : ДСТУ. Прийнято 28.12.2022; чинний від 31.12.2023. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=105931 (дата звернення: 14.12.2025).
86. ДБН В.1.1-7:2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги» : затв. наказом Мінрегіону України від 31.10.2016 № 287 // Портал Єдиної

державної електронної системи у сфері будівництва. URL: https://e-construction.gov.ua/laws_detail/3080743763845318619 (дата звернення: 14.12.2025).

87. Первинні засоби пожежогасіння. Вогнегасник. Будова та порядок використання : вебсторінка. НМЦ ДСНС України у м. Києві. 02.07.2024. URL: https://nmc.dsns.gov.ua/kyiv/news/ostanni-novini/pervinni-zasobi-pozezogasinnia-vognegasnik-budova-ta-poriadok-vikoristannia (дата звернення: 14.12.2025).
88. Вогнегасник : вебсторінка. Навчально-методичний центр ЦЗ та БЖД Сумської області. 15.07.2025. URL: https://nmc.dsns.gov.ua/sm/news/ostanni-novini/vognegasnik (дата звернення: 14.12.2025).
89. ISO. Introduction to ISO 14001:2015 (PUB100371). – Geneva : International Organization for Standardization, 2015. – [Електронний ресурс]. – URL: https://www.iso.org/publication/PUB100371.html (дата звернення: 14.12.2025).
90. Auditor Training Online. Understanding the Requirements of ISO 14001 Clause 6.1.2. – 16.04.2023. – [Електронний ресурс]. – URL: https://blog.auditortrainingonline.com/blog/understanding-the-requirements-of-iso-14001-clause-6.1.2 (дата звернення: 14.12.2025).
91. ISO. ISO 50001:2018. Energy management systems — Requirements with guidance for use. – Geneva : International Organization for Standardization, 2018. – [Електронний ресурс]. – URL: https://www.iso.org/standard/69426.html (дата звернення: 14.12.2025).
92. World Health Organization. Electromagnetic fields. – [Електронний ресурс]. – URL: https://www.who.int/health-topics/electromagnetic-fields (дата звернення: 14.12.2025).
93. International Energy Agency. Data Centres and Data Transmission Networks. – 11.07.2023. – [Електронний ресурс]. – URL: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks (дата звернення: 14.12.2025).

94. International Energy Agency. Electricity 2025. – Paris : IEA, 2025. – 200 p. – [Електронний ресурс]. – URL: https://iea.blob.core.windows.net/assets/0f028d5f-26b1-47ca-ad2a-5ca3103d070a/Electricity2025.pdf (дата звернення: 14.12.2025).
95. International Telecommunication Union. Recommendation ITU-T L.1300 (06/2014): Best practices for green data centres. – Geneva : ITU, 2014. – 68 p. – [Електронний ресурс]. – URL: https://www.itu.int/rec/dologin_pub.asp?id=T-REC-L.1300-201406-I!!PDF-E&lang=e&type=items (дата звернення: 14.12.2025).
96. International Telecommunication Union; UNITAR SCYCLE Programme; Fondation Carmignac. The Global E-waste Monitor 2024. – Geneva : ITU, 2024. – [Електронний ресурс]. – URL: https://www.itu.int/en/ITU-D/Environment/Pages/Publications/The-Global-E-waste-Monitor-2024.aspx (дата звернення: 14.12.2025).
97. Directive 2012/19/EU of the European Parliament and of the Council of 4 July 2012 on waste electrical and electronic equipment (WEEE) (recast) : consolidated version 04.07.2018. – [Електронний ресурс]. – URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?from=EN&uri=CELEX%3A02012L0019-20180704 (дата звернення: 14.12.2025).
98. Про управління відходами : Закон України від 20.06.2022 № 2320-IX // База даних «Законодавство України» / Верховна Рада України. – URL: https://zakon.rada.gov.ua/go/2320-20 (дата звернення: 14.12.2025).
99. World Resources Institute; World Business Council for Sustainable Development. GHG Protocol. Scope 2 Guidance: An amendment to the GHG Protocol Corporate Standard. – Washington, DC : WRI, 2015. – 120 p. – [Електронний ресурс]. – URL: https://ghgprotocol.org/sites/default/files/2023-03/Scope%202%20Guidance.pdf (дата звернення: 14.12.2025).
100. International Energy Agency. Energy demand from AI // Energy and AI. – Paris : IEA, 2025. – [Електронний ресурс]. – URL:

https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai (дата звернення: 14.12.2025).

101. International Energy Agency. Data Centres and Data Transmission Networks. – [Електронний ресурс]. – URL: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks (дата звернення: 14.12.2025).

Фрагмент коду побудови топології

```
# app/network/topology_builder.py
from __future__ import annotations
import logging
from datetime import datetime
from typing import List, Optional
from pydantic import BaseModel
from sqlmodel import Session, select
from app.models.topology_db import (
    TopologySnapshotDB,
    TopologyNodeStateDB,
    NodeType,
    NodeStatus,
)
from app.network.discovery import HostInfo, PortInfo

logger = logging.getLogger(__name__)

class TopologyNode(BaseModel):
    id: str
    label: str
    ip: str
    mac: Optional[str] = None
    type: str

class TopologyLink(BaseModel):
    source: str
    target: str
    type: str = "wifi"  # example

class TopologyResponse(BaseModel):
    nodes: List[TopologyNode]
    links: List[TopologyLink]

# ======================

def build_topology_from_hosts(
    run_id: str,
    hosts: List[HostInfo],
    session: Session,
) -> TopologySnapshotDB:
    existing = session.exec(
    select(TopologySnapshotDB).where(TopologySnapshotDB.run_id == run_id)
    ).first()
    if existing:
        logger.info("Topology snapshot для run_id=%s вже існує (id=%s)", run_id, existing.id)
        return existing

    snapshot = TopologySnapshotDB(
        run_id=run_id,
        description=f"Topology зібрана з nmap-скану для запуску {run_id}",
        generated_at=datetime.utcnow().isoformat() + "Z",
    )
    session.add(snapshot)
    session.flush()

    nodes_db: List[TopologyNodeStateDB] = []
    for host in hosts:
        node_type = _guess_node_type(host)
        name = host.hostname or host.ip
        node = TopologyNodeStateDB(
            snapshot_id=snapshot.id,
            node_id=host.ip,
            name=name,
            type=node_type,
            status=NodeStatus.ok,
            issues=[],
        )
        nodes_db.append(node)

    session.add_all(nodes_db)
```

```
    session.commit()
    session.refresh(snapshot)

    logger.info(
        "Побудовано topology snapshot для run_id=%s: %d вузлів",
        run_id,
        len(nodes_db),
    )

    return snapshot

# =====================

def build_topology_graph_from_hosts(
    hosts: List[HostInfo],
    gateway_ip: Optional[str] = None,
) -> TopologyResponse:
    nodes: List[TopologyNode] = []
    links: List[TopologyLink] = []

    if gateway_ip is None:
        gateway_ip = _guess_gateway_ip(hosts)

    gateway_id: Optional[str] = None
    if gateway_ip:
        gateway_id = f"gw:{gateway_ip}"
        nodes.append(
            TopologyNode(
                id=gateway_id,
                label="Wi-Fi Router",
                ip=gateway_ip,
                mac=None,
                type="gateway",
            )
        )

    for host in hosts:
        node_id = f"host:{host.ip}"
        node_type_enum = _guess_node_type(host)
        node_type_str = getattr(node_type_enum, "value", str(node_type_enum))

        nodes.append(
            TopologyNode(
                id=node_id,
                label=host.hostname or host.ip,
                ip=host.ip,
                mac=host.mac,
                type=node_type_str or "device",
            )
        )

        if gateway_id and host.ip != gateway_ip:
            links.append(
                TopologyLink(
                    source=gateway_id,
                    target=node_id,
                    type="wifi",
                )
            )

    topo = TopologyResponse(nodes=nodes, links=links)

    logger.info(
        "Побудовано граф топології: %d вузлів, %d ребер",
        len(nodes),
        len(links),
    )
    return topo
```

Фрагмент коду MARL агентів

```
class Transition(NamedTuple):
    state: int
    action: int
    reward: float
    agent_id: str

@dataclass
class MarlConfig:
    gamma: float = 0.99
    lr: float = 0.1
    baseline_lr: float = 0.1
    train_episodes: int = 50
    max_episode_steps: int = 20
    max_eval_steps: int = 20
    exploration_eps: float = 0.05

    max_kl: float = 0.01

LowLevelCommandStep = Dict[str, Any]
LowLevelCommandSequence = List[LowLevelCommandStep]

class SoftmaxPolicy:
    def __init__(
        self,
        n_states: int,
        n_actions: int,
        lr: float,
        baseline_lr: float,
        seed: Optional[int] = None,
    ) -> None:
        self.n_states = n_states
        self.n_actions = n_actions
        self.lr = lr
# ========================
    @staticmethod
    def _softmax(logits: List[float]) -> List[float]:
        m = max(logits)
        exps = [math.exp(x - m) for x in logits]
        s = sum(exps)
        if s == 0.0:
            return [1.0 / len(logits) for _ in logits]
        return [e / s for e in exps]

# ========================
    def greedy_action(self, state: int) -> int:
        probs = self.probabilities(state)
        return max(range(self.n_actions), key=lambda i: probs[i])

    def update(
        self,
        transitions: List[Tuple[int, int, float]],
        max_kl: float = 0.01,
    ) -> None:
        by_state: Dict[int, List[Tuple[int, float]]] = {}
        for s, a, G in transitions:
            by_state.setdefault(s, []).append((a, G))

# ========================

class MarlController:

    def __init__(
        self,
        agent_ids: Sequence[str],
        n_states: int,
        max_actions_per_state: int,
        config: MarlConfig,
        seed: Optional[int] = None,
# ========================
```

```
    def _run_episode(
        self,
        env: AttackGraphEnv,
        config: MarlConfig,
    ) -> Tuple[List[Transition], float]:
        state = env.reset()
        transitions: List[Transition] = []
        total_reward = 0.0

# ======================

    def _update_from_episode(
        self,
        transitions: List[Transition],
        gamma: float,
    ) -> None:
        returns: List[float] = [0.0] * len(transitions)
        G = 0.0
        for t in reversed(range(len(transitions))):
            G = transitions[t].reward + gamma * G
            returns[t] = G

        by_agent: Dict[str, List[Tuple[int, int, float]]] = {}
        for tr, G in zip(transitions, returns):
            by_agent.setdefault(tr.agent_id, []).append(
                (tr.state, tr.action, G)
            )

        for agent_id, traj in by_agent.items():
            self.policies[agent_id].update(traj, max_kl=self.config.max_kl)

    def train(self, env: AttackGraphEnv) -> None:
        for _ in range(self.config.train_episodes):
            transitions, _ = self._run_episode(env, self.config)
            if transitions:
                self._update_from_episode(transitions, self.config.gamma)

    def rollout_greedy(
        self,
        env: AttackGraphEnv,
        scenario_id: str,
    ) -> LowLevelCommandSequence:
        sequence: LowLevelCommandSequence = []

# ======================

def optimize_scenarios(
    scenarios: Sequence[Any],
    make_env: Callable[[Any], AttackGraphEnv],
    config: Optional[MarlConfig] = None,
) -> List[LowLevelCommandSequence]:
    if config is None:
        config = MarlConfig()

    all_results: List[LowLevelCommandSequence] = []

    for scn in scenarios:
        env = make_env(scn)

        agent_ids = ["Recon", "Exploit", "Lateral", "Service"]
        max_actions_per_state = max(
            (len(v) for v in env.adjacency.values()), default=1
```

Фрагмент коду сценаріїв

```
from datetime import datetime
from typing import Any, Dict, List, Optional
import json
import uuid
from fastapi import APIRouter, Depends, HTTPException, Request
from pydantic import BaseModel
from sqlmodel import Session, select
from sqlalchemy import text
from sqlalchemy.exc import IntegrityError
from app.db.session import get_session
from app.models.scenario_db import ScenarioDB
from app.orchestrator.intel.scenario_planner import generate_scenarios_with_llm

# ====

@router.get("/")
def list_scenarios(session: Session = Depends(get_session)) -> Dict[str, Any]:

    db_scenarios = session.exec(select(ScenarioDB)).all()

    scenarios: List[Dict[str, Any]] = []
    for s in db_scenarios:
        scenarios.append(
            {
                "id": s.id,
                "name": s.name,
                "description": s.description,
                "status": s.status,
                "agents": _normalize_agents(getattr(s, "agents", None)),
                "constraints": getattr(s, "constraints", None),
            }
        )

    return {"scenarios": scenarios}

@router.post("/")
async def create_scenario(
    request: Request,
    session: Session = Depends(get_session),
) -> Dict[str, Any]:
    try:
        data = await request.json()
    except Exception:
        raise HTTPException(status_code=400, detail="Invalid JSON")

    name = data.get("name")
    description = data.get("description") or ""
    agents_raw = data.get("agents") or []
    status = data.get("status") or "active"
    constraints_payload = data.get("constraints")

    if not name:
        raise HTTPException(status_code=400, detail="Field 'name' is required")

    if not isinstance(agents_raw, list):
        agents_list = [str(agents_raw)]
    else:
        agents_list = [str(a) for a in agents_raw]

    new_id = "SCN-LLM-" + uuid.uuid4().hex[:8].upper()

    pragma_rows = session.exec(text("PRAGMA table_info(scenarios)"))
    column_names = {row[1] for row in pragma_rows}
    has_constraints_column = "constraints" in column_names

    agents_json = json.dumps(agents_list)

    if has_constraints_column:
        if constraints_payload is None:
            constraints_json = None
```

```
        else:
            try:
                constraints_json = json.dumps(constraints_payload)
            except TypeError:
                constraints_json = json.dumps(str(constraints_payload))
    else:
        constraints_json = None

    try:
        if has_constraints_column:
            stmt = text(

            ).bindparams(
                id=new_id,
                name=name,
                description=description,
                status=status,
                agents=agents_json,
                constraints=constraints_json,
            )
# ====

        session.exec(stmt)
        session.commit()

# ====

    return {
        "id": new_id,
        "name": name,
        "description": description,
        "status": status,
        "agents": agents_list,
        "constraints": constraints_payload,
    }

# ====

class LLMGenerateRequest(BaseModel):
    policy_id: Optional[int] = None
    max_scenarios: int = 3

class LLMGenerateResponse(BaseModel):
    scenarios: List[Dict[str, Any]]

@router.post("/llm_generate", response_model=LLMGenerateResponse)
async def llm_generate_scenarios(
    payload: LLMGenerateRequest,
    session: Session = Depends(get_session),
) -> LLMGenerateResponse:
    scenarios = await generate_scenarios_with_llm(
        session=session,
        policy_id=payload.policy_id,
        max_scenarios=payload.max_scenarios,
    )
    return LLMGenerateResponse(scenarios=scenarios)
```

Фрагмент коду MARL середовища

```
# app/orchestrator/intel/marl_env.py
from __future__ import annotations
import logging
from dataclasses import dataclass
from typing import Any, Dict, List, Optional, Sequence, Tuple
logger = logging.getLogger(__name__)

@dataclass
class AttackEdge:
    src: int
    dst: int
    action: str
    agent_id: str
    base_reward: float = 1.0
    recommended_tool: Optional[str] = None

class AttackGraphEnv:
    def __init__(
        self,
        adjacency: Dict[int, List[AttackEdge]],
        start_state: int,
        target_states: Sequence[int],
        agent_cycle: Sequence[str],
        step_cost: float = -0.01,
        goal_reward: float = 5.0,
# ====
agent_cycle else ["Recon"]
        self.step_cost = step_cost
        self.goal_reward = goal_reward
        self.n_states = self._infer_n_states()
        self.current_state: int = start_state
        self.current_agent_idx: int = 0
        self.step_count: int = 0
        self.done: bool = False

        logger.debug(
            "[AttackGraphEnv] init: n_states=%d, start=%s, targets=%s",
            self.n_states,
            self.start_state,
            self.target_states,
        )
# ====
        try:
            max_id = max(int(s) for s in states)
            return max_id + 1
        except Exception:

            return len(states)

    def reset(self) -> int:
        self.current_state = self.start_state
        self.current_agent_idx = 0
        self.step_count = 0
        self.done = False
        return self.current_state

    @property
    def current_agent_id(self) -> str:
        if not self.agent_cycle:
            return "Recon"
        return self.agent_cycle[self.current_agent_idx %
len(self.agent_cycle)]

    def available_actions(self) -> List[AttackEdge]:
        return self.adjacency.get(self.current_state, [])
# ====
        edges = self.available_actions()
        if not edges:
            self.done = True
            reward = self.step_cost
            self.step_count += 1
            logger.debug(
                "[AttackGraphEnv] dead-end in state=%s, reward=%s",
                self.current_state,
                reward,
            )
            return self.current_state, reward, True, {"edge": None}

        if action_idx < 0 or action_idx >= len(edges):
            action_idx = len(edges) - 1

        edge = edges[action_idx]

        base = edge.base_reward if edge.base_reward is not None else 0.0
        reward = base + self.step_cost
        next_state = edge.dst
        self.current_state = next_state
        self.step_count += 1
```