

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ
ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ КІБЕРБЕЗПЕКОЮ ТА ЗАХИСТОМ ІНФОРМАЦІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗАЦІЇ
ПРОЦЕСІВ ПЕНТЕСТУ”

на здобуття освітнього ступеня магістра
зі спеціальності 125 Кібербезпека та захист інформації
освітньо-професійної програми Управління інформаційною та кібернетичною
безпекою

*Кваліфікаційна робота містить результати власних досліджень. Використання
ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Лада СЛОБОДСЬКА

(підпис)

Ім'я, ПРИЗВИЩЕ здобувача

Виконала:

Здобувачка вищої освіти гр. УБДМ-61
Лада СЛОБОДСЬКА

Керівник:

Дмитро РАБЧУН

К.т.н.

Рецензент:

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

Навчально-науковий інститут кібербезпеки та захисту інформації

Кафедра Управління кібербезпекою та захистом інформації

Ступінь вищої освіти магістр

Спеціальність 125 Кібербезпека та захист інформації

Освітньо-професійна програма Управління інформаційною та кібернетичною безпекою

ЗАТВЕРДЖУЮ

Завідувач кафедри УКБЗІ

_____ Світлана ЛЕГОМІНОВА

“ _____ ” _____ 2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Студентці Слободській Ладі Олегівні

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: “Використання штучного інтелекту для автоматизації процесів пентесту”

керівник кваліфікаційної роботи Дмитро РАБЧУН, к.т.н.

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від “30” жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи “11” грудня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: тестування на проникнення, автоматизація процесів кібербезпеки, застосування штучного інтелекту для аналізу мобільних інформаційних систем.
4. Перелік питань, які потрібно розробити:
 1. дослідити теоретичні основи застосування моделей штучного інтелекту (зокрема LLM) у виявленні вразливостей та обробці артефактів мобільних застосунків;
 2. проаналізувати існуючі підходи та практичні інструменти, які використовуються для автоматизації пентесту за допомогою AI, визначити їхні можливості та обмеження;
 3. розробити та експериментально перевірити модулі автоматизації на основі штучного інтелекту для окремих етапів мобільного пентесту та порівняти їхню ефективність із традиційним підходом.
5. Перелік ілюстративного матеріалу: *презентація*.
6. Дата видачі завдання “02” жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення об'єкту, предмету, мети та завдань дослідження.	10.10.2025	
2.	Збір та аналіз літератури.	23.10.2025	
3.	Аналіз сучасних підходів до автоматизації тестування на проникнення мобільних застосунків із використанням штучного інтелекту.	27.10.2025	
4.	Дослідження принципів роботи та можливостей LLM-моделей у задачах семантичного аналізу коду, конфігурацій і безпекових артефактів мобільних застосунків.	10.11.2025	
5.	Визначення напрямів та методів інтеграції AI-модулів у процес пентесту, включно з автоматизацією статичного аналізу, аналізу локального сховища та генерації звітності.	15.11.2025	
6.	Формулювання висновків за результатами дослідження.	22.11.2025	
7.	Оформлення роботи.	09.12.2025	
8.	Оформлення презентації.	10.12.2025	
9.	Отримання рецензії на роботу.	11.12.2025	
10.	Захист в ЕК.	20.01.2026	

Здобувачка вищої освіти

_____ (підпис)

Лада СЛОБОДСЬКА

(Ім'я, ПРИЗВИЩЕ)

Керівник
кваліфікаційної роботи

_____ (підпис)

Дмитро РАБЧУН

(Ім'я, ПРИЗВИЩЕ)

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ
ІНФОРМАЦІЇ**

**ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра**

Направляється здобувачка Слободська Л. О. до захисту кваліфікаційної роботи
(*прізвище та ініціали*)

за спеціальністю 125 Кібербезпека та захист інформації
(*код, найменування спеціальності*)

Освітньо-професійної програми Управління інформаційною та кібернетичною безпекою
(*назва*)

на тему: “Використання штучного інтелекту для автоматизації процесів пентесту”
Кваліфікаційна робота і рецензія додаються.

Директор ННІКБЗІ

(*підпис*)

Євгенія ІВАНЧЕНКО

(*Ім'я, ПРІЗВИЩЕ*)

Висновок керівника кваліфікаційної роботи

Здобувачка **СЛОБОДСЬКА Лада** у кваліфікаційній роботі дослідила теоретичні та практичні аспекти застосування штучного інтелекту для автоматизації процесів тестування на проникнення мобільних застосунків. У роботі проаналізовано сучасні AI-підходи, розроблено та експериментально перевірено три модулі автоматизації пентесту, спрямовані на семантичний аналіз конфігурацій, оцінку безпечності локального сховища та формування звітності.

СЛОБОДСЬКА Лада показала високу теоретичну і практичну підготовку, володіння науково-дослідницькими методами, вміння самостійно знаходити шляхи вирішення проблеми дослідження. Результати дослідження апробовані на конференції “Стратегії кіберстійкості: управління ризиками та безперервність бізнесу” 27 лютого 2025 року.

Все це дозволяє оцінити кваліфікаційну роботу здобувача **СЛОБОДСЬКОЇ Лади** на оцінку “добре” та присвоїти йому кваліфікацію “Магістр з кібербезпеки та захисту інформації за освітньо-професійною програмою Управління інформаційною та кібернетичною безпекою”.

Керівник кваліфікаційної роботи _____

(*підпис*)

Дмитро РАБЧУН

(*Ім'я, ПРІЗВИЩЕ*)

“ ____ ” _____ 2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувачка Слободська Л. О. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою

Управління кібербезпекою та захистом
інформації

(*підпис*)

Світлана ЛЕГОМІНОВА

(*Ім'я, ПРІЗВИЩЕ*)

ВІДГУК РЕЦЕНЗЕНТА на кваліфікаційну магістерську роботу

здобувачки вищої освіти Слободської Лади Олегівни
на тему “Використання штучного інтелекту для автоматизації процесів пентесту”

Актуальність З огляду на стрімке зростання складності мобільних застосунків та обсягів даних, що підлягають аналізу під час тестування на проникнення, виникає потреба у впровадженні інтелектуальних інструментів, здатних автоматизувати рутинні та трудомісткі етапи пентесту. Сучасні моделі штучного інтелекту відкривають можливості для підвищення точності виявлення вразливостей, скорочення часу аналізу та стандартизації результатів. У зв'язку з цим тема дослідження є надзвичайно актуальною та відповідає сучасним тенденціям розвитку кібербезпеки.

Позитивні сторони

1. У роботі ґрунтовно проаналізовано сучасні підходи до застосування штучного інтелекту в автоматизації пентесту та досліджено можливості LLM-моделей у семантичному аналізі конфігурацій, обробці локального сховища та формуванні звітів. Представлено три розроблені та експериментально перевірені AI-модулі, що демонструють практичну ефективність обраних рішень.

2. Кваліфікаційна робота оформлена відповідно до вимог. Виклад матеріалу здійснено відповідно до плану, зроблено логічні висновки. Ключові положення роботи представлено у вигляді рисунків. Автор опрацювала значну джерельну базу: близько 21 публікацій та електронних джерел, в тому числі англійських.

3. За результатами дослідження сформульовано практичні рекомендації щодо інтеграції штучного інтелекту в процес тестування на проникнення, а також окреслено перспективи розвитку інтелектуальних інструментів у сфері мобільної безпеки.

Недоліки

1. Доцільно було б приділити більше практичному дослідженню питань оцінки ризиків використання AI у безпекових процесах, зокрема аспектам надійності моделей, можливим "галюцинаціям" LLM та механізмам валідації AI-висновків.

Однак, вищезгадані зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи.

Висновок: Кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує позитивної оцінки, а здобувачка Слободська Лада Олегівна заслуговує присвоєння кваліфікації “Магістр кібербезпеки за освітньо-професійною програмою “Управління інформаційною та кібернетичною безпекою”.

Рецензент:

підпис

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 85 стор., 12 рис., 21 джерел.

Метою роботи є встановлення ефективних підходів до застосування штучного інтелекту для автоматизації ключових етапів тестування на проникнення з метою підвищення точності, швидкості та результативності виявлення вразливостей.

Об'єктом дослідження є процес тестування на проникнення інформаційних систем.

Предметом дослідження є методи, моделі та інструменти штучного інтелекту, що застосовуються для автоматизації окремих етапів тестування на проникнення, зокрема збору даних, аналізу вразливостей, обробки результатів та написання звіту за результатами проведеного пентесту.

Методи дослідження. У роботі застосовуються методи системного аналізу для формалізації процесу тестування на проникнення та виділення етапів, які можуть бути автоматизовані засобами штучного інтелекту. Для аналізу структури мобільних застосунків та їхніх компонентів використовуються положення теорії графів, зокрема при побудові семантичних зв'язків між методами, даними та мережевими викликами. Методи машинного навчання та теорії штучного інтелекту застосовано для розробки та оцінювання AI-модулів, зокрема LLM-моделей, що виконують класифікацію артефактів, аналіз конфігурацій та генерацію структурованих звітів.

Короткий зміст роботи. Робота присвячена дослідженню можливостей застосування штучного інтелекту для автоматизації ключових етапів тестування на проникнення мобільних застосунків. У теоретичній частині проаналізовано сучасні методики пентесту, підходи до побудови та використання моделей ШІ, а також принципи автоматизації процесів інформаційної безпеки. Практична частина включає розробку й експериментальну перевірку трьох AI-модулів: для семантичного аналізу конфігурацій і статичних артефактів, оцінки безпечності локального зберігання даних та автоматичного формування звітів пентесту. Проведено порівняльний аналіз ефективності традиційного та AI-підсиленого підходів, визначено їхні переваги, обмеження та умови застосування. На основі отриманих результатів сформульовано рекомендації щодо інтеграції інтелектуальних систем у процес тестування на

проникнення та окреслено перспективи подальшого розвитку таких рішень.

Галузь застосування. Розроблені у роботі підходи можуть бути використані під час проведення тестування на проникнення мобільних застосунків, а також при побудові систем моніторингу та оцінки безпеки в організаціях, де необхідна часткова або повна автоматизація аналізу вразливостей. Запропоновані AI-модулі здатні доповнювати існуючі інструменти інформаційної безпеки, підвищуючи швидкість, точність та відтворюваність результатів, що робить їх корисними для команд пентесту, аналітиків безпеки, розробників та фахівців, відповідальних за впровадження практик безпечної розробки ПЗ.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, АВТОМАТИЗОВАНИЙ ПЕНТЕСТ, АВТОНОМНІ АГЕНТИ, АНАЛІЗ ВРАЗЛИВОСТЕЙ, КІБЕРБЕЗПЕКА.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 85 pages, 12 figures, 21 sources.

The purpose of the work is to identify effective approaches to applying artificial intelligence for automating key stages of penetration testing in order to improve the accuracy, speed, and overall effectiveness of vulnerability discovery.

The object of the research is the penetration testing process of information systems.

Subject of research includes methods, models, and artificial intelligence tools used to automate individual stages of penetration testing, including data collection, vulnerability analysis, processing of test results, and automated generation of penetration testing reports.

Research methods. This work employs methods of system analysis to formalize the penetration testing process and identify stages that can be automated using artificial intelligence. Graph theory principles are applied to analyze the structure of mobile applications and their components, particularly for constructing semantic relationships between methods, data flows, and network calls. Machine learning techniques and artificial intelligence theory are used to develop and evaluate AI modules, including LLM-based models responsible for artifact classification, configuration analysis, and structured report generation.

Brief content of research. The study explores the potential of artificial intelligence to automate key stages of penetration testing for mobile applications. The theoretical section analyzes modern pentesting methodologies, approaches to building and applying AI models, and principles of automation in information security processes. The practical section includes the development and experimental evaluation of three AI modules: one for semantic analysis of configurations and static artifacts, one for assessing the security of local data storage, and one for automated generation of penetration testing reports. A comparative evaluation of traditional and AI-enhanced approaches was conducted, identifying their advantages, limitations,

and conditions for effective use. Based on the results, recommendations were formulated regarding the integration of intelligent systems into the penetration testing workflow and the prospects for further development of such solutions.

Field of research. The approaches developed in this work can be applied during penetration testing of mobile applications, as well as in the design of security monitoring and assessment systems in organizations where partial or full automation of vulnerability analysis is required. The proposed AI modules can complement existing information security tools by improving the speed, accuracy, and reproducibility of results, making them valuable for penetration testing teams, security analysts, developers, and professionals responsible for secure software development practices.

KEYWORDS: ARTIFICIAL INTELLIGENCE, AUTOMATED PENETRATION TESTING, AUTONOMOUS AGENTS, VULNERABILITY ANALYSIS, CYBERSECURITY.

ЗМІСТ

ВСТУП.....	11
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ТЕСТУВАННІ НА ПРОНИКНЕННЯ.....	13
1.1. Сутність і етапи процесу тестування на проникнення.....	13
1.2. Технології штучного інтелекту: підходи, моделі та сфери застосування.....	19
1.3. Теоретичні засади автоматизації інформаційної безпеки засобами ШІ.....	32
Висновки до розділу 1.....	38
РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В АВТОМАТИЗАЦІЇ ПЕНТЕСТУ	40
2.1. Огляд сучасних AI-інструментів для автоматизації етапів пентесту.....	40
2.2. Практичне тестування AI-інструментів на прикладі вибраного середовища....	47
2.2.1 Реалізація та експериментальна перевірка AI-модуля №1 для семантичного аналізу конфігурацій і статичних артефактів мобільного застосунку.....	51
2.2.2 Реалізація та експериментальна перевірка AI-модуля №2 для аналізу локального сховища мобільного застосунку та виявлення ризиків зберігання даних.....	60
2.2.3 Реалізація та експериментальна перевірка AI-модуля №3 для автоматизації формування звітності пентесту.....	69
2.3. Порівняльний аналіз ефективності пентесту з використанням та без використання AI.....	74
Висновки до розділу 2.....	78
РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ У ПРОЦЕС ТЕСТУВАННЯ НА ПРОНИКНЕННЯ.....	81
3.1. Умови та критерії доцільності впровадження AI в пентест.....	82
3.2. Архітектурна модель інтеграції AI-інструментів у процес пентесту.....	86
3.3. Проблеми та перспективи застосування штучного інтелекту в кібербезпеці....	91
Висновки до розділу 3.....	96
ВИСНОВКИ.....	98
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	100
ДОДАТОК А.....	102
ДОДАТОК Б.....	105
ДОДАТОК В.....	106
ДОДАТОК Г.....	109

ВСТУП

Актуальність теми. Сучасні мобільні застосунки є складними програмними системами, що активно взаємодіють із локальним середовищем, хмарною інфраструктурою та сторонніми сервісами. Це робить їх одним із головних об'єктів атак у сфері кібербезпеки. Тестування на проникнення залишається ключовим методом виявлення вразливостей, однак традиційні підходи часто вимагають значних часових ресурсів, глибокої залученості фахівця та ретельного аналізу великої кількості артефактів.

На цьому тлі стрімкий розвиток моделей штучного інтелекту, зокрема великих мовних моделей (LLM), відкриває нові можливості для автоматизації складних і рутинних етапів пентесту. Доцільність використання таких моделей визначається потребою підвищити точність, швидкість і відтворюваність аналізу, а також зменшити когнітивне навантаження на фахівців з кібербезпеки. Саме тому дослідження застосування штучного інтелекту в автоматизації мобільного пентесту є своєчасним, актуальним та науково обґрунтованим

Метою роботи є встановлення ефективних підходів до застосування штучного інтелекту для автоматизації ключових етапів тестування на проникнення з метою підвищення точності, швидкості та результативності виявлення вразливостей.

Об'єктом дослідження є процес тестування на проникнення інформаційних систем.

Предметом дослідження є методи, моделі та інструменти штучного інтелекту, що застосовуються для автоматизації етапів тестування на проникнення, включно зі збором даних, аналізом вразливостей, обробкою результатів та формуванням звітності.

Для досягнення цієї мети в роботі необхідно виконати наступні завдання:

1. Здійснити аналіз наукової літератури, технічної документації та сучасних досліджень, що стосуються використання штучного інтелекту для автоматизації етапів тестування на проникнення.
2. Дослідити теоретичні основи застосування моделей штучного інтелекту у виявленні вразливостей та їх обробці.
3. Проаналізувати існуючі підходи та практичні інструменти, які

використовуються для автоматизації пентесту за допомогою AI, визначити їх можливості й обмеження.

4. Розробити та експериментально перевірити AI-модулі автоматизації для окремих етапів мобільного пентесту та порівняти їх ефективність з традиційним підходом.

Методи дослідження. У роботі застосовано методи системного аналізу для формалізації процесу пентесту та визначення етапів, які можуть бути автоматизовані засобами штучного інтелекту. Методи машинного навчання та теорії штучного інтелекту застосовано для створення й оцінювання AI-модулів, у тому числі LLM-моделей, що виконують класифікацію артефактів, аналіз конфігурацій і генерацію структурованих технічних звітів.

Наукова новизна одержаних результатів полягає у розробці та експериментальній апробації комплексного підходу до автоматизації мобільного пентесту за допомогою трьох взаємодоповнювальних AI-модулів, які реалізують семантичний аналіз конфігурацій, аналіз локального сховища та автоматичне формування звітності. Запропоновано архітектурну модель інтеграції LLM у процес пентесту, що дозволяє підвищити точність і швидкодію аналізу.

Практичне значення одержаних результатів полягає в можливості використання запропонованих AI-модулів у реальних процесах тестування на проникнення мобільних застосунків. Розроблені підходи можуть бути інтегровані в системи моніторингу безпеки, інструменти для DevSecOps-практик та методики внутрішнього аудиту. Отримані результати дають змогу значно зменшити обсяг ручної роботи пентестера, покращити структурованість аналітичних матеріалів та забезпечити стабільну якість звітності.

Апробація результатів кваліфікаційної роботи було оприлюднено на Всеукраїнській науковій конференції “Стратегії кіберстійкості: управління ризиками та безперервність бізнесу”.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ТЕСТУВАННІ НА ПРОНИКНЕННЯ

Сучасні системи кібербезпеки потребують комплексного підходу до виявлення та попередження загроз, серед яких важливе місце займає тестування на проникнення. Цей процес дозволяє оцінити реальний рівень захищеності інформаційних активів, моделюючи дії потенційного зловмисника та визначаючи вразливі місця до того, як ними скористаються на практиці. З огляду на стрімке розширення цифрових інфраструктур, збільшення кількості взаємопов'язаних сервісів та високий темп появи нових технологій, традиційні підходи до пентесту вже не завжди забезпечують необхідну швидкість і масштабованість.

У цих умовах дедалі більшої актуальності набуває застосування штучного інтелекту, який здатний автоматизувати рутинні етапи аналізу, покращити точність виявлення вразливостей та зменшити навантаження на фахівців. Інтеграція ШІ у процес тестування на проникнення відкриває можливості для глибшого аналізу даних, прогнозування потенційних шляхів атак та виконання складних технічних операцій у напівавтоматичному режимі.

У цьому розділі послідовно розглядаються теоретичні основи пентесту, сучасні технології штучного інтелекту та підходи до автоматизації інформаційної безпеки із використанням ШІ. Така структура дозволяє сформулювати цілісне уявлення про взаємозв'язок між традиційними методами тестування та інтелектуальними інструментами, що створюють підґрунтя для подальшого практичного аналізу у наступних розділах.

1.1. Сутність і етапи процесу тестування на проникнення

Тестування на проникнення це метод оцінювання рівня безпеки інформаційних систем шляхом моделювання реальних кібератак. Основною метою цього процесу є виявлення вразливостей, які можуть бути використані

зловмисниками, а також перевірка ефективності наявних засобів захисту. Пентест дає змогу не лише визначити технічні недоліки, але й оцінити можливі ризики для організації та спрогнозувати наслідки успішної атаки.

До ключових цілей тестування на проникнення належать: ідентифікація слабких місць у програмному забезпеченні, мережевій інфраструктурі чи конфігураціях; оцінювання рівня ризику, який створюють знайдені вразливості; перевірка того, наскільки ефективно працюють механізми захисту під час реальних атак; а також моделювання сценаріїв зловмисних дій з метою виявити потенційні шляхи компрометації системи.

Тестування на проникнення суттєво відрізняється від інших практик кібербезпеки. На відміну від автоматичного сканування вразливостей, пентест включає глибокий аналіз, ручні перевірки та експлуатацію недоліків. Red Team орієнтується на комплексну перевірку здатності організації протистояти атакам, фокусуючись на непомітності та обхідних методах. Bug Bounty передбачає відкриту модель пошуку вразливостей спільнотою дослідників. АРТ-симуляції моделюють дії добре організованих та тривалих атак, тоді як класичний пентест має обмежений час і чітко визначений об'єкт scope. Таким чином, penetration testing займає окреме місце серед методів оцінки кіберстійкості, забезпечуючи збалансоване поєднання практичної експлуатації та структурованої перевірки безпеки.

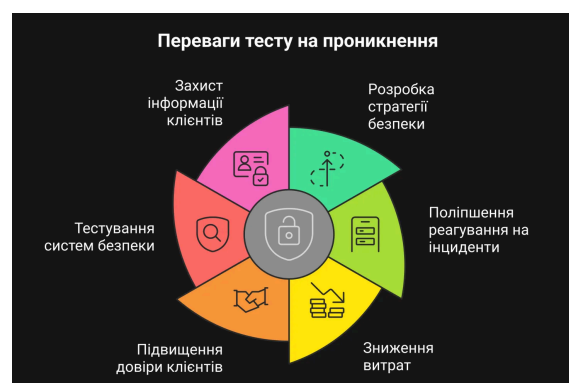


Рисунок 1.1 Переваги тестування на проникнення

Класифікація пентесту може здійснюватися за різними критеріями, залежно від підходу, умов тестування та характеру цільових активів. Одним із

ключових критеріїв є рівень доступу, який отримує тестувальник перед початком роботи.

1. Black-box тестування передбачає відсутність початкової інформації про систему. Фахівець працює так само, як зовнішній зловмисник, використовуючи відкриті джерела даних і методи активного аналізу. Такий підхід дозволяє оцінити захищеність системи від зовнішніх атак.

2. Grey-box тестування включає часткове надання інформації або обмежених облікових даних. Це дозволяє сфокусуватися на конкретних компонентах системи та перевірити безпеку з позиції внутрішнього користувача з невисоким рівнем привілеїв.

3. White-box тестування передбачає отримання повної технічної інформації, включно з документацією, архітектурними схемами, конфігураціями та вихідним кодом. Такий підхід забезпечує максимально глибоку перевірку безпеки та дозволяє виявити складні логічні або архітектурні вразливості.

Іншим важливим критерієм класифікації є об'єкт тестування, тобто компоненти інфраструктури, які підлягають оцінюванню. Серед найпоширеніших категорій виділяють:

1. веб-додатки, включно з бекенд-API та інтеграційними сервісами;
2. API, які часто мають розширену поверхню атаки через логіку запитів і авторизаційні механізми;
3. мережеві інфраструктури, що охоплюють маршрутизатори, сервери, сегментацію мережі та протоколи взаємодії;
4. мобільні додатки, де перевіряються механізми зберігання даних, взаємодія з сервером та безпека застосункових компонентів;
5. хмарні сервіси, що включають перевірку конфігурацій, політик доступу та взаємодії між ресурсами;
6. соціальна інженерія, спрямована на оцінку людського чинника, здатності співробітників розпізнавати фішинг чи інші методи маніпуляції.

Така класифікація дозволяє адаптувати процес тестування до конкретних потреб організації та забезпечити найбільш повне охоплення потенційних ризиків.

Проведення тестування на проникнення базується на загальновизнаних стандартах і методологіях, які визначають структуру, етапи, підходи та рекомендації щодо виконання перевірок безпеки. Використання таких документів забезпечує уніфікованість процесів, повторюваність результатів та відповідність кращим практикам галузі.

Одним із найпоширеніших ресурсів є OWASP Testing Guide, що містить детальні рекомендації з тестування веб-додатків. Документ охоплює класифікацію вразливостей, техніки аналізу, підходи до тестування автентифікації, сесій, авторизації, логіки застосунку та обробки введених даних. OWASP TG фактично є стандартом де-факто для перевірки веб-рішень.

Penetration Testing Execution Standard (PTES) [1] пропонує структурований підхід до пентесту, включаючи передпроектну підготовку, збір інформації, моделювання загроз, експлуатацію вразливостей та звітування. Стандарт описує як технічні, так і організаційні аспекти, дозволяючи узгодити процес тестування з очікуваннями замовника.

NIST SP 800-115 є офіційним керівництвом Національного інституту стандартів і технологій США з технічних перевірок безпеки. Документ описує загальні принципи планування, виконання і документування тестів, а також підходи до сканування, соціальної інженерії та аналізу конфігурацій [2].

Ще одним важливим ресурсом є OSSTMM (Open Source Security Testing Methodology Manual) [3] комплексна методологія, яка охоплює різні аспекти безпеки: інформаційні системи, мережеві середовища, фізичну безпеку, процеси та взаємодію персоналу. OSSTMM вирізняється математичною моделлю оцінювання безпеки, що дозволяє кількісно визначати рівень ризику [4].

Застосування цих стандартів забезпечує системність, прозорість і високу якість процесу тестування, а також дозволяє адаптувати підхід до специфіки кожної організації.

Процес тестування на проникнення складається з низки послідовних етапів, кожен із яких має визначену мету, методи та результати. Дотримання структурованого підходу забезпечує повноту перевірки, повторюваність процесу та відповідність професійним стандартам.

1. Рекогносцировка (Reconnaissance) включає збір інформації про цільову систему. На пасивному етапі використовуються відкриті джерела, такі як реєстри доменів, соціальні мережі та загальнодоступні метадані. Активна рекогносцировка передбачає безпосередню взаємодію з об'єктом через сканування портів чи пошук доступних служб. Мета цього етапу сформувані початкове уявлення про поверхню атаки.

2. Сканування та аналіз вразливостей охоплює визначення відкритих портів, працюючих сервісів, версій програмного забезпечення та потенційно вразливих місць. На цьому етапі застосовуються автоматизовані інструменти та ручний аналіз для підтвердження отриманих результатів. Він дозволяє сформувані карту векторів атаки та визначити найімовірніші точки входу.

3. Експлуатація вразливостей полягає у практичному застосуванні знайдених недоліків для отримання доступу або виконання шкідливих дій. Це може включати обходи автентифікації, використання логічних помилок, атаки на конфігурації чи виконання коду. Етап демонструє реальну можливість компрометації системи.

4. Підвищення привілеїв і закріплення доступу, якщо потрібно, передбачає розширення отриманих можливостей до рівня адміністратора або іншого привілейованого облікового запису. Закріплення доступу виконується для збереження контролю над системою у випадку перезавантаження або зміни конфігурацій.

5. Постексплуатація включає оцінку наслідків доступу, аналіз чутливих даних, можливих шляхів горизонтального переміщення та моделювання впливу успішної атаки. Мета продемонструвати, до яких результатів може призвести компрометація.

6. Звітність і рекомендації є завершальним етапом, на якому оформлюються результати роботи. Він включає опис знайдених вразливостей, методи їх експлуатації, рівні ризику, а також рекомендації щодо усунення недоліків і покращення безпеки. Звіт слугує основою для підвищення кіберстійкості організації.

Послідовне виконання цих етапів дозволяє отримати комплексне уявлення про рівень захищеності системи та забезпечити обґрунтовані рекомендації щодо її вдосконалення.



Рисунок 1.2 Основні етапи пентесту

Традиційні підходи до тестування на проникнення мають низку обмежень, які помітно впливають на ефективність роботи у сучасних умовах. Передусім цей процес потребує значних часових та людських ресурсів: багато етапів виконуються вручну, що уповільнює перевірку й підвищує залежність від кваліфікації окремого спеціаліста. Це, у свою чергу, ускладнює регулярне

повторення пентестів та робить їх менш придатними для динамічних середовищ.

Ще одним викликом є обмежена масштабованість традиційних методів. Коли інфраструктура організації включає десятки чи сотні сервісів, ручне охоплення всіх компонентів стає практично неможливим, що збільшує ризик пропуску важливих векторів атаки.

Додаткову складність створює сучасна архітектура інформаційних систем. Перехід до хмар, контейнеризації, мікросервісів та розподілених платформ значно ускладнює аналіз. Такі середовища потребують глибокого технічного розуміння та часу, а ручні методи не завжди дозволяють дослідити їх достатньо детально.

Усе це підсилює потребу в автоматизації. Зростання кількості цифрових активів і швидкий розвиток загроз вимагають інструментів, здатних швидко обробляти великі обсяги даних і оперативно реагувати на зміни. Автоматизація окремих етапів пентесту, зокрема тих, що є рутинними або повторюваними, стає необхідною умовою для підтримання належного рівня безпеки.

1.2. Технології штучного інтелекту: підходи, моделі та сфери застосування

Сучасний штучний інтелект об'єднує набір методів, що дозволяють комп'ютерним системам аналізувати дані, робити висновки та приймати рішення. Підходи ШІ формуються навколо ідеї автоматичного виявлення закономірностей у даних, побудови моделей поведінки та оптимізації рішень. У центрі цих процесів стоїть машинне навчання (ML), фундаментальний інструмент, на основі якого побудовано більшість сучасних технологій ШІ.

1. Машинне навчання - це підхід, що дозволяє моделі самостійно вивчати залежності у даних і покращувати власні передбачення з досвідом. Усі методи ML поділяють на три ключові категорії залежно від того, які дані доступні під час навчання та яку мету має модель.

1. Supervised learning використовується тоді, коли кожному прикладу у даних відповідає правильна відповідь. Мета моделі навчитися зіставляти вхідні дані з відповідними результатами. Це підхід, на якому ґрунтується більшість задач передбачення: класифікація об'єктів, прогноз числових значень, автоматичне виявлення спаму тощо. Модель отримує приклади «питання - правильна відповідь» і оптимізує свої параметри так, щоб мінімізувати помилку між прогнозом і реальним значенням.

2. Unsupervised learning застосовується, коли дані не мають міток, а отже модель повинна самостійно знаходити приховану структуру. Метою є не конкретне передбачення, а виявлення закономірностей, подібностей або відмінностей між об'єктами. Найпоширенішими задачами є кластеризація (групування схожих об'єктів) та зниження розмірності (стиснення даних зі збереженням суттєвої інформації). Цей підхід дозволяє структурувати дані, відкривати нові закономірності та підсилювати інші моделі ШІ.

3. Reinforcement learning спрямоване на формування поведінки агента, який взаємодіє із середовищем та отримує винагороди за свої дії. Агент поступово вчиться приймати рішення, що максимізують довгострокову користь. Такий підхід моделює процеси, подібні до навчання живих істот: спроба → наслідок → корекція поведінки. RL застосовують для робототехніки, автономних систем, ігор та оптимізації складних процесів.

Усі три парадигми машинного навчання разом утворюють методологічну основу сучасного ШІ. Вони дозволяють будувати моделі, здатні сприймати інформацію, узагальнювати її та адаптуватися до нових умов. У наступних підрозділах буде розглянуто глибинне навчання як один із ключових напрямів розвитку ML, а також підходи, пов'язані з обробкою мови та генерацією нового контенту [6].

2. Глибинне навчання (Deep Learning) є піднапрямом машинного навчання, що базується на використанні багатoshарових штучних нейронних мереж. Його ключовою особливістю є здатність автоматично формувати багаторівневі представлення даних: від найпростіших ознак до високорівневих

абстракцій. Завдяки цьому глибинні моделі ефективно працюють із зображеннями, текстами, звуком, часовими рядами та іншими складними структурами, часто досягаючи результатів, недосяжних для класичних методів ML.

1. У центрі глибинного навчання знаходяться штучні нейронні мережі. Вони складаються з послідовності шарів, де кожен шар виконує перетворення даних: поєднання лінійного перетворення та нелінійної активації. Навчання відбувається шляхом поступової корекції ваг моделі через зворотне поширення помилки. Саме здатність створювати глибокі структури з десятків і сотень шарів дозволяє мережам виявляти складні залежності та узагальнювати інформацію.

2. Для обробки зображень особливо ефективними стали згорткові нейронні мережі (CNN). Вони використовують згорткові фільтри, які здатні виокремлювати локальні патерни країв, текстур, форми незалежно від їхнього положення на зображенні. Це забезпечує моделі інваріантність до зсувів та дозволяє поступово формувати уявлення про об'єкти та їхню структуру. CNN лежать в основі систем розпізнавання облич, медичної діагностики за зображеннями, аналізу відео та автономної навігації [7].

3. Для роботи з послідовними даними використовувалися рекурентні нейронні мережі (RNN), які зберігають внутрішній стан, що фіксує інформацію про попередні елементи. Це робить їх придатними для задач, де важливий порядок або контекст: розпізнавання мовлення, машинний переклад, моделювання часових рядів. Варіанти RNN такі як LSTM та GRU розв'язують проблему довгострокових залежностей і здатні зберігати корисну інформацію на великих відстанях у послідовності.

4. Найбільш революційним кроком у розвитку DL стали трансформери архітектура, що замінила послідовну обробку на паралельний аналіз усієї послідовності. Основою трансформерів є механізм self-attention, який дозволяє моделі визначати, які частини вхідних даних є найважливішими для поточного кроку. Це дає змогу ефективно працювати з дуже довгими контекстами, гнучко

моделювати залежності та масштабувати моделі до величезних розмірів. Саме трансформери лежать в основі сучасних великих мовних моделей, здатних генерувати текст, відповідати на запитання, писати код, виконувати аналіз і навіть створювати мультимодальні описи [8].

У сукупності всі ці архітектури базові нейронні мережі, CNN, RNN і трансформери формують основу глибинного навчання. Кожна з них оптимізована під певний тип даних і задач, а разом вони утворюють універсальний інструмент, який визначає розвиток сучасного штучного інтелекту.

3. Обробка природної мови (Natural Language Processing, NLP) - це напрям штучного інтелекту, що займається аналізом, інтерпретацією та генерацією людської мови. Його метою є навчити комп'ютер розуміти текст і мовлення на рівні, подібному до людського, враховуючи складність лінгвістичних структур, контексту, багатозначності, прихованих смислів та культурних залежностей. Сучасні методи NLP поєднують статистичні моделі, глибинне навчання та семантичні представлення, що дозволяє працювати з мовою значно гнучкіше, ніж класичні алгоритми.

1. Основою NLP є перехід від сирого тексту до форм представлення, які модель може обробляти. Раніше це здійснювалося за допомогою ручного створення ознак таких як частотні словникові моделі, n-грамні статистики чи синтаксичні дерева. Натомість сучасні підходи базуються на векторних представленнях слів і фраз (embeddings), які відображають їхній семантичний зміст. Такі моделі, як Word2Vec чи GloVe, стали важливим кроком уперед, оскільки дозволили відображати значення слів у безперервний простір, де близькі за сенсом слова мають подібні вектори.

2. Подальший розвиток NLP пов'язаний із глибинними нейронними мережами. Рекурентні моделі LSTM і GRU дали змогу вперше ефективно працювати з довгими реченнями, зберігаючи контекст. Але справжній прорив стався з появою трансформерів, які радикально змінили уявлення про те, як потрібно моделювати мовні залежності. Завдяки механізму self-attention

трансформери аналізують текст як цілісну структуру, визначаючи важливі фрагменти незалежно від їхнього положення у реченні. Це дозволило працювати з великими обсягами даних, охоплювати довгі контексти й вивчати складні смислові взаємозв'язки між словами.

3. Сучасні NLP-системи виконують широкий спектр задач. До фундаментальних належать розпізнавання та нормалізація тексту, визначення частин мови, побудова синтаксичних структур. Більш високорівневі задачі включають класифікацію текстів, аналіз тональності, видобування сутностей, пошук інформації, резюмування та машинний переклад. Окремий напрям генерація тексту, де моделі на кшталт великих мовних моделей (LLM) здатні створювати логічні, граматично коректні та стилістично узгоджені тексти.

4. Особливістю NLP є необхідність роботи з неоднозначністю людської мови. Одне слово може мати кілька значень, речення різні інтерпретації, а контекст вирішальний вплив на сенс. Саме тому сучасні моделі приділяють велику увагу контекстуальним представленням, де значення слова визначається оточенням, а не лише його «базовим» змістом. Технології BERT, GPT та інші контекстуальні моделі відкрили можливість аналізувати мову на значно глибшому рівні, фактично наближуючи машинне розуміння до людського.

NLP є ключовим елементом розвитку інтелектуальних систем: від чат-ботів та віртуальних асистентів до пошукових систем, автоматичного перекладу та інструментів аналізу великих текстових масивів. Воно забезпечує можливість природної взаємодії людини з машиною, що робить його одним із найважливіших напрямів у сучасному штучному інтелекті.

4. Генеративні моделі є одним із найдинамічніших напрямів сучасного штучного інтелекту, оскільки вони здатні створювати нові дані, які нагадують навчальний розподіл: текст, зображення, відео, аудіо, тривимірні об'єкти та навіть складні багатомодальні структури. На відміну від дискримінативних моделей, що лише класифікують або оцінюють вхідні дані, генеративні моделі формують новий вміст, поєднуючи статистичні закономірності, семантичні

структури та багаторівневі абстракції. Два ключові класи сучасних генеративних систем великі мовні моделі (LLM) та дифузійні моделі стали основою технологічного зсуву, від автоматизації тексту до генерації зображень та мультимодальних інтерфейсів.

1. Великі мовні моделі (Large Language Models, LLM) це глибинні нейронні мережі, які навчаються передбачати наступний елемент послідовності на основі попереднього контексту. Завдяки масштабуванню кількості параметрів, обсягів даних і глибині архітектури трансформерів LLM набули здатності узагальнювати мовні закономірності на високому рівні. Вони створюють зв'язний текст, відповідають на запитання, адаптуються до стилю, виконують логічні міркування та працюють зі складними інструкціями. Контекстуальні векторні представлення, які формує модель, дозволяють не лише відтворювати мовні патерни, а й здійснювати семантичний аналіз, інтегрувати знання та моделювати причинно-наслідкові зв'язки в тексті.

2. LLM функціонують через механізм attention, що дає змогу моделі оцінювати значення кожного слова щодо інших елементів послідовності. Саме це забезпечує гнучкість у роботі з різними мовами, доменами та типами завдань. Під час масштабування моделі виявляють нові здатності феномен, що отримав назву emergent abilities: від складних інструкцій до багатокрокового міркування. На їх основі створюються системи діалогових агентів, автоматичного написання коду, аналізу документів, навчальних та творчих інструментів.

3. Другий великий клас генеративних систем дифузійні моделі. Вони ґрунтуються на процесі поступового додавання шуму до даних і зворотнього навчання моделі відновлювати структуру з зашумлених прикладів. Цей процес можна розглядати як моделювання зворотної дифузії, де з хаотичного шуму модель відтворює структуру зображення або іншого типу даних. На кожному кроці модель передбачає «чистіший» варіант даних, доки не отримає повністю сформований приклад. Така поетапність дозволяє досягати високої якості та деталізації у згенерованому результаті.

4. Дифузійні моделі стали основою сучасних генеративних систем для створення зображень, оскільки здатні опановувати тонкі текстури, складні сцени та стилістичні особливості. Вони працюють у поєднанні з текстовими енкодерами, що дозволяє перетворювати текстові описи на візуальний контент. Завдяки цьому виникли системи, здатні генерувати фотореалістичні чи художні зображення, виконувати редагування, додавати деталі, змінювати стиль і структуру сцени.

5. Генеративні моделі змінюють спосіб створення інформації та взаємодії з нею. Вони дозволяють автоматизувати творчі процеси, підтримують наукові та інженерні дослідження, створюють нові інструменти для освіти, дизайну та програмування. Попри вражаючі можливості, вони також ставлять нові виклики: питання достовірності інформації, упередженості даних, етичних стандартів та контролю якості. Проте генеративні моделі вже стали одним із ключових стовпів сучасного ШІ, і їхній розвиток визначатиме подальший технічний і соціальний ландшафт.

Типи моделей у сучасному штучному інтелекті охоплюють широкий спектр методів, спрямованих на роботу з різними типами даних, задач та структур інформації. Кожен клас моделей розв'язує свій набір завдань і має власні методологічні особливості, які визначають спосіб навчання, інтерпретації та застосування. Нижче розглядаються ключові типи моделей, що формують основу сучасних систем машинного навчання та штучного інтелекту.

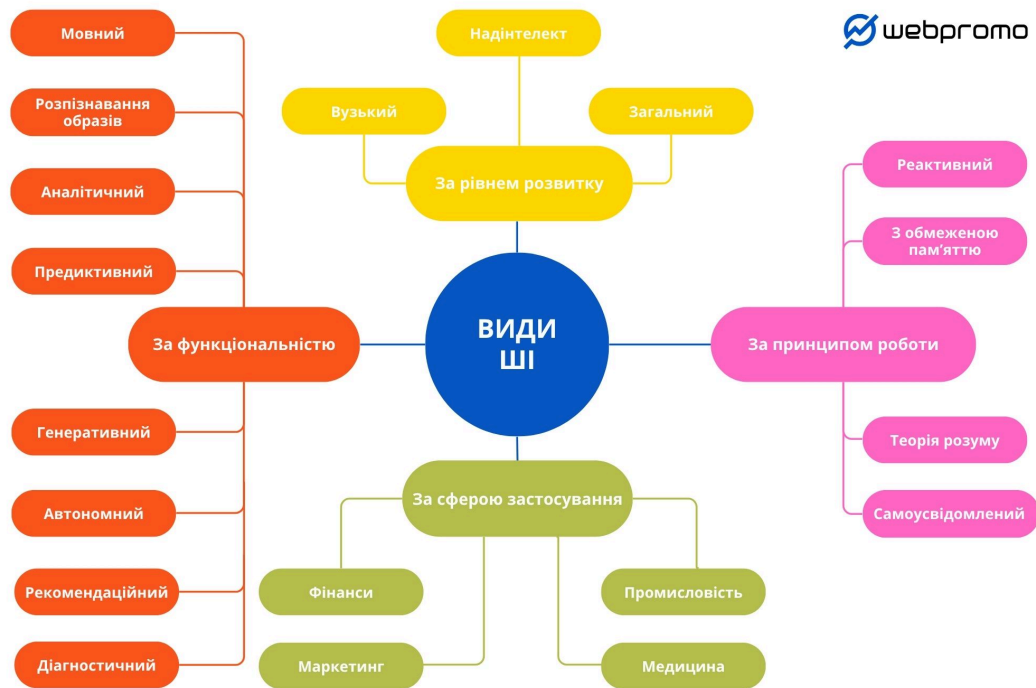


Рисунок 1.3 Види штучного інтелекту

Класифікаційні моделі спрямовані на визначення належності об'єкта до одного з наперед заданих класів. Вони працюють у випадках, коли доступні позначені дані, а мета полягає в передбаченні дискретної категорії. Такі моделі використовуються в розпізнаванні образів, аналізі медичних даних, фільтрації спаму, оцінці ризиків. Типові підходи включають логістичну регресію, методи на основі дерев рішень, ансамблеві моделі та нейронні мережі. Основним критерієм успішності є якість узагальнення, тобто здатність моделі робити точні передбачення на нових даних. Оцінка проводиться за допомогою метрик на кшталт accuracy, precision, recall чи F1-score.

Моделі кластеризації належать до навчання без учителя та призначені для виявлення структур у невпорядкованих даних. Вони розподіляють об'єкти на групи так, щоб елементи всередині групи були схожими, а між групами відмінними. Кластеризація використовується для сегментації клієнтів, групування текстів за темами, структурування великих наборів зображень, виявлення подібностей у поведінці користувачів. Найпоширенішими підходами є k-means, ієрархічні методи, Gaussian Mixture Models та алгоритми щільнісної

кластеризації. У кластеризації немає «правильної» відповіді, тому якість часто оцінюється непрямими метриками або експертною інтерпретацією.

Моделі аномалій зосереджені на виявленні даних, що відхиляються від загальної закономірності. Такі моделі особливо важливі у сферах безпеки, моніторингу, фінансового контролю, оскільки дозволяють виявляти шахрайські дії, технічні збої або нетипову поведінку систем. Підходи до виявлення аномалій включають моделі щільності, однокласові SVM, ізоляційні ліси, а також автоенкодери, які навчаються відновлювати нормальні патерни й сигналізують про аномалію, коли відновлення є поганим. Особливістю цієї категорії є складність доступу до якісних міток, тому оцінювання моделей часто відбувається у специфічних доменних умовах.

Нейронні мережі для прогнозування спеціалізуються на задачах регресії, модельованні часових рядів та передбаченні поведінки складних систем. Вони здатні враховувати нелінійні залежності, взаємодію між ознаками та складні патерни динаміки. Для часових рядів використовуються як класичні рекурентні моделі (LSTM, GRU), так і сучасні архітектури на основі трансформерів, здатні працювати з довгими контекстами та нерівномірними послідовностями. Такі мережі використовуються для фінансового прогнозування, аналізу попиту, прогнозування навантаження в інфраструктурі, медичних прогнозів. Основна складність полягає у правильній побудові вибірки, врахуванні сезонності та коректному масштабуванні даних.

Великі мовні моделі (GPT, Claude, Llama та інші) є одними з найпотужніших сучасних моделей, здатних генерувати текст, аналізувати інформацію, виконувати інструкції та здійснювати логічні міркування. Вони навчаються на величезних корпусах тексту, використовуючи архітектуру трансформерів. Завдяки механізму attention такі моделі формують контекстно залежні представлення слів і фраз, що дозволяє їм працювати з багатозначністю мови та фіксувати семантичні зв'язки на довгих відстанях. Великі мовні моделі можуть відповідати на запитання, створювати тексти різних стилів, перекладати, аналізувати документи, писати код, генерувати описи зображень та

інтегрувати знання з різних доменів. Вони стали основою сучасних інтелектуальних систем та визначають нову епоху розвитку штучного інтелекту.

Усі ці типи моделей разом формують багатогранну екосистему метода машинного навчання та ШІ, де кожен клас виконує свою роль: класифікація допомагає структурувати інформацію, кластеризація відкриває приховані закономірності, детектори аномалій забезпечують безпеку, прогнозні мережі підтримують стратегічні рішення, а великі мовні моделі надають універсальний мовний інтерфейс до знань та інформації.

Штучний інтелект відіграє дедалі важливішу роль у кібербезпеці, оскільки сучасні атаки стають складнішими, динамічнішими та краще прихованими. Традиційні інструменти вже не здатні ефективно реагувати на обсяг і швидкість подій у цифрових екосистемах, тому системи безпеки дедалі частіше використовують моделі машинного навчання, глибинного аналізу та автономного прийняття рішень. ШІ забезпечує адаптивний підхід до виявлення загроз, аналізу поведінки користувачів, автоматизації реагування та моделювання атак.

Виявлення загроз є однією з ключових сфер застосування ШІ у кібербезпеці. Моделі машинного навчання дозволяють виявляти підозрілу активність, що відрізняється від нормальної поведінки систем або користувачів. Моделі аналізують великі обсяги мережевих подій, процесів, запитів і взаємодій у реальному часі, виявляючи відхилення, які можуть бути ознакою зловмисної активності: несанкціоновані доступи, сканування портів, спроби експлуатації вразливостей. На відміну від класичних сигнатурних методів, ШІ може ідентифікувати раніше невідомі загрози, що робить його особливо цінним у сучасних умовах.

Аналіз мережевого трафіку та поведінки користувачів (UEBA User and Entity Behavior Analytics) є наступною важливою сферою. UEBA-системи використовують моделі для створення профілів нормальної поведінки кожного користувача, пристрою або сервісу. У разі відхилень наприклад, різке збільшення кількості запитів, підозрілі маршрути мережевого трафіку, незвичні

операції з даними система може автоматично визначити потенційні інсайдерські загрози або компрометацію облікового запису. UEBA дозволяє виявляти складні атаки, які не супроводжуються явними сигнатурами, але проявляються аномальною поведінкою.

Аналіз логів та інтеграція з SIEM-системами є важливою задачею для масштабних інфраструктур. SIEM (Security Information and Event Management) збирає та корелює журнали подій з різних джерел серверів, мережевих пристроїв, додатків, хмарних систем. ШІ надає можливість автоматизовано фільтрувати шум, виділяти важливі аномалії, знаходити приховані залежності між подіями та визначати потенційні ланцюги атаки. Використання моделей для аналізу логів зменшує кількість false positives і значно прискорює реакцію кіберзахисників.

Автоматизація SOC (Security Operations Center) стає можливою завдяки інтеграції ШІ в процеси моніторингу, аналізу та реагування. Моделі можуть автоматично класифікувати та пріоритезувати інциденти, запускати playbooks, генерувати рекомендації або навіть виконувати первинні дії: блокування підозрілих IP, ізоляція вузлів, обмеження доступу. Це дозволяє зменшити навантаження на аналітиків SOC та підвищити ефективність реагування на загрози.

Генерація та моделювання атак ще один напрям, де ШІ відкриває нові можливості. Генеративні моделі можуть створювати симуляції атак, які допомагають тестувати системи захисту та покращувати стійкість інфраструктури. Наприклад, моделі можуть моделювати фішингові листи, автоматизоване сканування вразливостей чи моделювання сценаріїв поведінки зловмисників. Це дає змогу підготувати системи до майбутніх загроз, які ще не існують у реальності.

Побудова автономних агентів у кібербезпеці представляє перспективу створення самонавчальних систем, що здатні без участі людини здійснювати моніторинг, виявлення та реагування на загрози. Такі агенти можуть адаптуватися до нових умов, оновлювати власні моделі поведінки,

координувати свої дії з іншими компонентами інфраструктури. У майбутньому автономні агенти можуть перетворитися на основу «самозахисних» систем, які забезпечуватимуть безпеку в режимі реального часу та без постійного людського контролю.

Загалом, застосування ШІ у кібербезпеці дозволяє переходити від реактивних методів до проактивних стратегій захисту. Інтеграція інтелектуальних моделей у процеси моніторингу та реагування забезпечує значно вищу швидкість виявлення загроз, точність аналізу та адаптивність систем безпеки до нових викликів.

Застосування штучного інтелекту у сфері аналізу даних, безпеки та автоматизації забезпечує значний прогрес у якості й швидкості прийняття рішень. Втім, разом із перевагами ШІ приносить і певні ризики, пов'язані з природою даних, методами навчання та можливістю цілеспрямованого впливу на моделі. Розуміння сильних і слабких сторін ШІ є важливо важливим для відповідального та безпечного впровадження.

Однією з ключових переваг є покращена точність та швидкість аналізу. Штучний інтелект здатний обробляти великі обсяги даних у реальному часі, виявляючи закономірності та залежності, які могли б залишитися непомітними для людини. Завдяки глибоким нейронним мережам моделі здатні аналізувати складні структури даних наприклад, взаємозв'язки між подіями в журналах безпеки або складні патерни трафіку. Це забезпечує швидшу і точнішу оцінку ризиків, а також дозволяє реагувати на загрози майже миттєво.

Ще однією суттєвою перевагою є зменшення кількості false positives. Традиційні правила та сигнатурні системи часто генерують велику кількість хибних спрацювань, що створює навантаження на аналітиків і уповільнює реакцію на реальні загрози. Моделі ШІ виявляють контекстні зв'язки, здатні враховувати історичну поведінку та адаптуватися до змін, що дозволяє точніше відрізняти легітимну активність від підозрілої. Завдяки цьому команди безпеки можуть зосередитися на важливо важливих інцидентах, а системи працюють ефективніше.

Особливо цінною є можливість автоматичного виявлення складних аномалій. Моделі машинного навчання можуть розпізнавати нестандартну поведінку, яка не вписується у традиційні шаблони атак. Це включає виявлення прихованих або нових технік зловмисників, міжмережових взаємодій, що відхиляються від нормальної динаміки, нехарактерних команд доступу чи зміну профілю користувача. Такий підхід дає змогу знаходити загрози, які раніше залишалися невидимими для сигнатурних систем.

Попри численні переваги, використання ШІ у критичних сферах супроводжується ризиками. Одним із них є упередженість даних: моделі навчаються на доступних вибірках, які можуть бути неповними, нерепрезентативними або містити систематичні перекоси. Це може призвести до хибних висновків або несправедливих рішень. Іншим ризиком є хибні рішення моделі, коли ШІ помиляється у класифікації чи оцінці загрози, особливо у ситуаціях, що виходять за межі навчального досвіду.

Окрему групу загроз становлять атаки на моделі, відомі як adversarial ML. Зловмисники можуть свідомо створювати спеціально модифіковані вхідні дані, які вводять модель в оману, змушуючи її робити неправильні висновки. Це може бути невидима на перший погляд модифікація зображення, пакету мережевого трафіку чи тексту, яка істотно змінює реакцію системи. Крім того, атаки можуть бути спрямовані на викрадення моделі, маніпуляцію результатами або отруєння даних (data poisoning) під час навчання.

Таким чином, застосування ШІ приносить значні переваги у точності, швидкості та ефективності оцінки загроз, але водночас потребує ретельного управління ризиками, прозорості алгоритмів та впровадження додаткових механізмів захисту моделей.

1.3. Теоретичні засади автоматизації інформаційної безпеки засобами ШІ

У сучасних умовах розвитку кіберпростору автоматизація процесів інформаційної безпеки стає важливою складовою ефективного захисту.

Швидкість появи нових загроз, зростання обсягів даних та складність інцидентів вимагають використання інструментів, здатних оперативно аналізувати інформацію, приймати рішення та виконувати рутинні дії без участі людини. Штучний інтелект, а особливо методи машинного навчання, пропонують системний підхід до автоматизації як стратегічних, так і тактичних аспектів безпеки. У контексті пентесту це означає можливість автоматизованого виявлення вразливостей, моделювання атак, генерації сценаріїв експлуатації та аналізу результатів тестування.

Теоретичні засади автоматизації в кібербезпеці охоплюють моделювання процесів реагування, аналізу ризиків, виявлення загроз та оптимізації рутинних операцій. ШІ у поєднанні з формальними методами дозволяє створювати адаптивні системи, які враховують контекст, історію поведінки, типові патерни атак і здатні підлаштовуватися під динаміку середовища. Такі системи забезпечують не лише автоматизацію аналізу, а й здатність прогнозувати можливі сценарії розвитку атаки що є особливо важливим для проникнення в системи (penetration testing).

Роль машинного навчання у автоматизації аналізу даних полягає в здатності обробляти великі обсяги різномірної інформації та виявляти приховані залежності. Моделі ML здатні аналізувати мережевий трафік, логи, поведінку користувачів, події на кінцевих точках та інші дані, визначаючи аномалії, класифікуючи події або прогнозуючи потенційні загрози. У контексті пентесту ML може автоматизувати процес виявлення вразливостей, оцінку поверхні атаки, класифікацію потенційних шляхів експлуатації та аналіз результатів сканування. Завдяки адаптивності ML-системи здатні постійно вдосконалюватися, підлаштовуючись під нові техніки атак.

Таким чином, теоретичні засади автоматизації інформаційної безпеки засобами ШІ становлять фундамент для побудови інтелектуальних систем, здатних масштабувати захист, швидко реагувати на інциденти та підтримувати процеси пентесту на новому рівні ефективності.

Використання штучного інтелекту в penetration testing відкриває можливості для глибокої автоматизації дій, які раніше виконувалися виключно вручну або з обмеженою підтримкою стандартних інструментів. ШІ не лише прискорює процеси пентесту, але й робить їх масштабованими, адаптивними та більш точними. Теоретичні підходи, що лежать в основі такого використання, охоплюють широкий спектр моделей від автоматизованого збору інформації до формування автономних агентів, здатних діяти як повноцінні пентестери.

Автоматичне виконання reconnaissance це фундаментальний етап penetration testing, на якому визначається поверхня атаки. Використання ШІ дозволяє системам аналізувати відкриті джерела (OSINT), сканувати мережу, будувати карту інфраструктури та виявляти потенційні точки входу значно ефективніше, ніж традиційні методи. Моделі можуть ранжувати активи за важливістю, аналізувати взаємозв'язки між сервісами, визначати приховані залежності та прогнозувати можливі маршрути атаки. На основі метрик схожості та поведінкових патернів ШІ здатний визначати, які вузли з більшою ймовірністю є вразливими або критичними для безпеки.

AI-асистоване generative fuzzing та fuzzing під управлінням ML представляють новий рівень розумного тестування на надійність. Традиційне fuzzing генерує випадкові або напіввипадкові вхідні дані, сподіваючись викликати збої. ML-кероване fuzzing робить цей процес значно ефективнішим: моделі навчаються на структурі вхідних даних, поведінці програм та попередніх результатах тестів. Генеративні моделі (наприклад, на основі трансформерів або варіаційних автоенкодерів) можуть створювати інтелектуальні тестові випадки, які спрямовані на найбільш проблемні ділянки коду або мережевих інтерфейсів. Це підвищує шанс знайти складні й нетривіальні вразливості, які класичне fuzzing не виявляє.

Автоматичний пошук вразливостей на основі моделей аномалій ґрунтується на припущенні, що уразливі або неправильно налаштовані компоненти поведуться інакше, ніж нормально функціонуючі. Моделі аномалій аналізують поведінку системних процесів, мережевих з'єднань, відповіді

сервісів та логічну структуру застосунків. Вони можуть визначати непослідовності, проблеми конфігурацій, незвичайні реагування драйверів чи бібліотек, патерни, що відрізняються від «здорової» поведінки. Такий підхід дозволяє виявляти zero-day-вразливості, неправильні ACL, непомічені точки доступу та інші приховані загрози.

Створення автономних агентів це найвищий рівень інтеграції ШІ в penetration testing. Такі агенти поєднують декілька моделей: для збору інформації, пошуку вразливостей, генерації експлоїтів, ухвалення рішень та планування дій. Вони здатні виконувати послідовність пентестових операцій, адаптуватися до змін у середовищі, навчатися на власних діях і коригувати стратегію. Агенти можуть діяти як симульовані зловмисники, тестуючи системи в режимі реального часу, або як інструменти для автоматизації офенсивних процесів у Red Team. У перспективі такі системи можуть працювати повністю автономно, виконуючи роль цифрових атакувальних аналітиків.

AI-додатки для аналізу вихідного коду, API й конфігурацій базуються на поєднанні статичного й динамічного аналізу з моделями машинного навчання та великими мовними моделями. Теоретично аналіз вихідного коду передбачає побудову абстрактних синтаксичних дерев, графів потоку даних і викликів, після чого моделі ШІ класифікують патерни як безпечні або потенційно вразливі. Для API додається семантичний рівень: аналіз структури запитів і відповідей, логіки авторизації, обмежень швидкості, обробки помилок. Конфігураційні файли (файли доступу, налаштування мереж, політики безпеки) перетворюються на набори правил і параметрів, де моделі шукають відхилення від кращих практик або від внутрішніх політик організації.

На концептуальному рівні ці AI-додатки працюють як моделі детекції вразливостей високого рівня: вони не просто шукають сигнатури, а аналізують логіку та контекст. Наприклад, LLM може інтерпретувати фрагмент коду, пояснити, чому той чи інший виклик небезпечний, запропонувати безпечну альтернативу й автоматично сформулювати пулл-реквест для виправлення. У контексті пентесту це дає змогу швидко оцінювати великі кодові бази,

інтерфейси мікросервісів, конфігурації хмарних ресурсів, виявляючи вразливості ще до етапу експлуатації.

Інтеграція ШІ з існуючими пентест-фреймворками (Metasploit, Burp Suite, Nmap) є важливою складовою створення цілісних автоматизованих систем. На теоретичному рівні така інтеграція реалізується через абстрактний шар оркестрації, де ШІ виступає як «мозковий центр», а класичні інструменти як виконавчі модулі. LLM або інша модель приймає високорівневий опис задачі («знайти всі доступні веб-сервіси та спробувати базові експлойти»), трансліює його у послідовність команд для Nmap, інтерпретує результати сканування, формує запити до Burp Suite для тестування веб-додатків, а при виявленні вразливостей генерує чи підбирає модулі Metasploit для експлуатації.

З точки зору архітектури, це можна описати як багаторівневу систему, де:

1. нижній рівень класичні пентест-інструменти з їхніми CLI, API та плагінами;
2. проміжний рівень адаптери й конектори, що перетворюють дані та команди у формат, зрозумілий для моделей ШІ та інструментів;
3. верхній рівень інтелектуальний шар, що планує, коригує та аналізує дії, використовуючи LLM, ML-моделі аномалій, RL-агентів тощо.

У результаті пентестер отримує не просто набір інструментів, а інтегроване середовище, де ШІ допомагає приймати рішення, автоматизує рутинні кроки, підказує оптимальні вектори атак і формує звітність. Теоретично це наближає концепцію «пентесту як сервісу», де значна частина офенсивної діяльності може бути делегована інтелектуальним системам із мінімальним втручанням людини, а роль експерта зміщується в бік контролю, валідації та прийняття стратегічних рішень.

Використання штучного інтелекту у сфері кібербезпеки та penetration testing відкриває нові можливості для автоматизації, швидкості та глибини аналізу. Проте цей напрям супроводжується значними обмеженнями та викликами, які мають фундаментальний характер і ґрунтуються як на

технологічних, так і на етичних та правових аспектах. Для відповідального застосування ШІ важливо розуміти ці обмеження та оцінювати ризики, пов'язані з роботою інтелектуальних систем.

Одним із центральних викликів є контроль та інтерпретованість роботи моделей. Багато сучасних методів, зокрема глибинні нейронні мережі та великі мовні моделі, мають «чорний ящик» як свою внутрішню природу. Моделі приймають рішення на основі високовимірних представлень та нелінійних взаємодій, які не піддаються прямому поясненню. У критичних системах безпеки це створює серйозну проблему: важливо не лише отримати результат, а й зрозуміти його. Із цим пов'язана потреба у розробці інтерпретованих моделей, інструментів пояснення рішень (explainable AI), аудитів моделей та методик трасування логіки прийняття рішень. Без таких механізмів складно оцінити, чи є рішення моделі обґрунтованим і чи не базується воно на хибному або випадковому патерні.

Другим суттєвим викликом є ризики, пов'язані з відсутністю гарантій коректності рішень ШІ. На відміну від детермінованих алгоритмів, моделі машинного навчання не забезпечують стовідсоткової точності та можуть помилятися у нових або непередбачуваних умовах. У контексті пентесту це означає можливість некоректного визначення вразливості, хибно позитивних чи хибно негативних результатів, неправильного вибору експлойту або генерації недопустимого сценарію атаки. З огляду на те, що автоматизовані дії можуть бути інвазивними, помилки моделей несуть потенційно серйозний ризик для цілісності системи або навіть для легального статусу виконуваних операцій. У таких умовах необхідно застосовувати механізми валідації, подвійної перевірки та обмеження автономних дій ШІ.

Етичні та правові аспекти автоматизованих атак становлять окрему категорію викликів. Пентест за своєю природою є контрольованою атакою, і використання автономних або напівавтономних ШІ-систем створює ризики виходу за рамки дозволених дій. Питання відповідальності за непередбачувану поведінку ШІ, юридичний статус автоматизованих експлойтів, можливість

порушення законодавства про доступ до інформації усе це вимагає нових підходів до регулювання. Крім того, існують етичні дилеми: використання генеративних моделей для створення фішингових сценаріїв, обхід засобів автентифікації чи виготовлення шкідливого ПЗ може як допомагати тестуванню, так і створювати технології, що опиняться в руках зловмисників. Баланс між дослідженнями, безпекою та етикою є особливо складним у сфері офенсивної автоматизації.

Останнім важливим обмеженням є потреба в якісних і безпечних даних для навчання моделей. Моделі ШІ безпосередньо успадковують властивості наборів даних, на яких вони тренуються. Якщо дані містять упередження, неповноту, застарілі патерни або некоректну розмітку моделі відтворюватимуть ті самі помилки. У кібербезпеці це особливо важливо: якісні еталонні дані для тренування систем виявлення атак або аномалій часто недоступні через конфіденційність, правові обмеження або їхню високу вартість. Крім того, існує ризик отруєння навчальних даних (data poisoning), коли зловмисник спеціально підсовує моделі помилкові приклади, змінюючи її поведінку. Це робить безпечну підготовку даних однією з ключових передумов впровадження ШІ.

Висновки до розділу 1

Підсумовуючи матеріал розділу, можна стверджувати, що поєднання штучного інтелекту та penetration testing формує нову парадигму офенсивної кібербезпеки. Penetration testing - це структурований метод оцінки безпеки, який моделює дії реального зловмисника з метою виявлення вразливостей та оцінки рівня стійкості системи. Штучний інтелект, зі свого боку, представляє сукупність методів від машинного та глибинного навчання до великих мовних моделей і reinforcement learning здатних автоматизувати прийняття рішень, інтерпретувати складні дані, виявляти аномалії та будувати адаптивні стратегії взаємодії зі середовищем.

Точки перетину між пентестом і ШІ виникають там, де традиційні методи вже не справляються з масштабом, складністю або динамікою сучасних інформаційних систем. Обсяги даних, швидкість атак, багат шарові архітектури та поява нових векторів експлуатації вимагають інструментів, здатних швидко аналізувати інформацію, самостійно приймати рішення та адаптуватися до нових сценаріїв. Саме тому інтеграція ШІ в процеси penetration testing є не лише інновацією, а й необхідністю для забезпечення актуальної, глибокої та ефективної оцінки безпеки.

Це особливо очевидно, якщо врахувати обмеження класичного пентесту: значну тривалість процедур, залежність від кваліфікації експерта, високу вартість комплексних перевірок, невміння масштабуватися на великі середовища та неможливість забезпечити постійний моніторинг. Ручні методи є ефективними в точкових сценаріях, але вони не здатні підтримувати швидкість сучасних загроз. Натомість автоматизовані ШІ-підходи дозволяють зменшити людське навантаження, пришвидшити пошук вразливостей, відтворювати складні сценарії атак і забезпечувати ширшу покритість інфраструктури.

Таким чином, перший розділ демонструє не лише теоретичні основи ШІ та його роль у кібербезпеці, але й логічно підводить до наступного етапу детального аналізу практичних підходів, інструментів та технік автоматизованого penetration testing. У розділі 2 буде розглянуто конкретні методика, архітектури та реалізації, які перетворюють описані принципи на реальні офенсивні інструменти, здатні працювати в автономному або напівавтономному режимі.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В АВТОМАТИЗАЦІЇ ПЕНТЕСТУ

2.1. Огляд сучасних AI-інструментів для автоматизації етапів пентесту

У сучасній кібербезпеці застосування штучного інтелекту стало закономірним наслідком різкого зростання складності цифрових систем і навантаження на фахівців з тестування на проникнення. Обсяг даних, який необхідно проаналізувати під час пентесту, включає десятки тисяч рядків журналів, сотні мережевих запитів, великі масиви декомпільованого коду та конфігурацій, що робить традиційні методи ручної перевірки дедалі менш ефективними. До цього додається зростання кількості логічних вразливостей, які не проявляються через очевидні технічні помилки та часто залежать від складних зв'язків між компонентами застосунку. У таких умовах фахівцю не завжди достатньо часу й уваги для всебічного дослідження кожного маршруту, послідовності запитів або гілки коду.

На цьому фоні великі мовні моделі (LLM) стали природним інструментом підсилення пентесту, оскільки здатні працювати з великими обсягами інформації, визначати приховані залежності й формувати узгоджені висновки. Їхня роль у кібербезпеці поступово розширюється: від автоматичної класифікації артефактів і попереднього відсіювання шуму до глибшого аналізу вихідного коду та конфігурацій. LLM добре справляються з ідентифікацією небезпечних патернів, які класичні статичні аналізатори або пропускають, або не можуть коректно інтерпретувати через надто широкий контекст. Окремим напрямом стало використання моделей для генерації PoC, де система на основі логів чи мережевого трафіку пропонує можливі сценарії експлуатації та перевірки гіпотез. Поступово моделі інтегруються і в документоорієнтовані етапи пентесту, автоматизуючи формування структурованих звітів, нормалізацію доказів і уніфікацію рекомендацій. У сукупності ці тенденції говорять, що ШІ перестає бути допоміжним інструментом і переходить до

статусу повноцінного компонента пентест-процесу, що здатен істотно підвищувати швидкість і точність роботи аудитора [10].

Статичний аналіз мобільних застосунків зазнав істотних змін із появою великих мовних моделей, які здатні обробляти складні структуровані й напівструктуровані дані, зокрема вихідний код, конфігураційні файли, метадані та декомпільовані артефакти. Традиційні підходи до статичного аналізу спираються переважно на фіксовані правила, сигнатури та евристики, що обмежує їхню здатність виявляти складні логічні взаємозв'язки між компонентами програми. У мобільному середовищі це особливо важливо, адже вразливості часто виникають не через окремі дефектні фрагменти коду, а через комбіновані сценарії взаємодії між компонентами наприклад, взаємозв'язок між надмірними дозволами в `AndroidManifest`, некоректною обробкою даних у сервісах, неправильним використанням криптографічних API чи неконтрольованим викликом `WebView`. Саме в цій зоні сучасні LLM-підходи демонструють найбільшу ефективність.

Застосування LLM у статичному аналізі Android та iOS полягає в тому, що модель може одночасно опрацьовувати великий обсяг гетерогенних даних структуру класів, вихідні коди методів, конфігурації, інструкції байткоду, опис дозволів і залежності між компонентами та формувати цілісне уявлення про поведінку застосунку. Це дає змогу виявляти патерни, які не зводяться до одного правила, наприклад доступ до небезпечного API лише за певної конфігурації дозволів або використання ключів шифрування, що витікають зі статичних ресурсів. Моделі здатні ідентифікувати `hardcoded secrets` у різних форматах, розпізнавати підозрілу логіку марширування даних (наприклад, у поточних викликах мережеских методів), а також аналізувати небезпечні конструкції, пов'язані з `WebView`, `reflection API`, некоректним `exception-handling` та ін [11].

Аналіз мобільних конфігурацій ще один ключовий напрям застосування III. Файли `AndroidManifest.xml` та `Info.plist` є критичними точками, які визначають дозволи, експортовані компоненти, точки входу, поведінку сервісів і

політики безпеки. LLM здатна не лише визначити окремі проблеми, як-от надлишкові дозволи або експортовані activity без відповідних обмежень, а й зіставити їх із логікою застосунку в декомпільованому коді. Класичний SAST, наприклад MobSF або QARK, виконує аналіз за правилами, що дозволяють ідентифікувати поширені конфігураційні помилки, але не можуть комплексно оцінити взаємодію між дозволами й реальним використанням компонентів. LLM, на відміну від сигнатурного підходу, здатна проаналізувати трансформацію даних, виявити надлишкові ADB-залежності, оцінити доцільність дозволів із погляду бізнес-логіки застосунку та визначити небезпечні сценарії, де конфігурація створює передумови для атак, наприклад для токен-підміни чи доступу до чутливих даних із суміжних застосунків.

Порівняння з класичними SAST-інструментами демонструє якісну різницю. MobSF і QARK залишаються потужними компонентами базового статичного аналізу, виконують швидке сканування й здатні забезпечити охоплення відомих шаблонів вразливостей. Однак вони працюють переважно у вузько визначених межах і не інтерпретують поведінку застосунку як цілісної системи. Їхні результати часто містять надлишкові сповіщення або пропускають нетривіальні логічні залежності, які не вкладаються у набір попередньо запрограмованих правил. Модель на основі ШІ натомість може обробляти контекст, розпізнавати абстракції високого рівня та будувати гіпотези щодо потенційних ризиків, ґрунтуючись на взаємозв'язках між компонентами, що значно наближує процес аналізу до логіки досвідченого аудитора. Таким чином, сучасні LLM-підходи не замінюють SAST, а виступають інтелектуальним шаром надбудови, що підвищує глибину та точність статичного аналізу в мобільному пентесті.

У сфері динамічного аналізу мобільних застосунків штучний інтелект поступово займає нішу, яка традиційно потребувала значних витрат часу та глибокого доменного досвіду. Динамічна поведінка застосунку зазвичай фіксується у вигляді мережевих трас, логів, runtime-артефактів, тимчасових файлів і локальних сховищ, що робить її багатовимірною й важкою для повного

ручного охоплення. Саме тому AI-орієнтовані інструменти дедалі частіше використовуються не тільки для попереднього фільтрування даних, але й для інтерпретації їхнього змісту у контексті потенційних вразливостей. Однією з ключових можливостей таких систем стала автоматична класифікація локальних артефактів SQLite-баз, SharedPreferences, Keychain-записів, кешованих файлів та інших елементів локальної моделі зберігання мобільного застосунку. У класичному підході аудитор доводиться вручну переглядати структури таблиць, аналізувати вміст ключів, ідентифікувати, які дані є чутливими, а які службовими. AI-моделі беруть на себе саме цей етап, визначаючи характер даних, відокремлюючи некритичні елементи та концентруючи увагу фахівця на тих даних, які можуть вказувати на потенційні проблеми: незахищені токени, ідентифікаційні дані користувача, недоречно кешовані параметри сесії або сліди внутрішніх механізмів авторизації [12].

Важливим напрямом застосування ШІ є також виявлення `hardcoded secrets`, що традиційно викликає труднощі через розпорошеність їх зберігання у різних частинах мобільного застосунку. Секрети можуть бути присутні у декомпільованому коді, у ресурсах, у частково стиснутих конфігураційних файлах, у тимчасових файлових системах чи навіть у нестандартних кешах, сформованих під час запуску додатка. Класичні регулярні вирази й евристики дають змогу виявляти лише частину ключів, тоді як LLM-моделі здатні розпізнавати зміст, формат та функцію підозрілих даних, оцінюючи їх у контексті взаємодії компонентів. Вони можуть визначити, що певний рядок у SQLite-базі, який зовні виглядає як ідентифікатор, насправді є токеном доступу; що поле в SharedPreferences, позначене як «`session`», містить невітікаючий JWT; або що рядок у ресурсах застосунку пов'язаний зі стороннім API і є ключем сервісу. Така контекстуальна інтерпретація значно підвищує точність перевірки й зменшує кількість хибнопозитивних результатів, які часто виникають під час класичного сканування.

Окрему увагу заслуговує автоматизація оцінки SQLite та Keychain критичних елементів локальної безпеки на Android та iOS відповідно. У

ручному режимі аудиторю потрібно аналізувати структуру кожної таблиці, перевіряти типи значень, ідентифікувати недоречно відкриті поля і співставляти їх із логікою застосунку. AI дозволяє значно прискорити цей процес. Модель може автоматично побудувати семантичний профіль бази даних, виділити поля, що потенційно містять конфіденційні дані, оцінити ризики, пов'язані з відсутністю шифрування, та запропонувати відповідні архітектурні альтернативи, зокрема використання Android Keystore або iOS Keychain як засобу захисту чутливих значень. Аналогічно, аналіз Keychain може виконуватися з урахуванням політик доступності, прив'язки до біометрії та обмежень між застосунками, що дозволяє визначити, чи дотримано рекомендацій Apple щодо захисту даних [13].

Таким чином, інструменти AI для аналізу динамічної поведінки та локального зберігання даних переходять від рівня простого фільтрування результатів до рівня глибокої контекстної інтерпретації. Моделі синтезують інформацію з різних джерел, визначають зв'язки між артефактами, виявляють небезпечні патерни та створюють більш повну картину безпеки мобільного застосунку, ніж це можливо в межах традиційних підходів статичного чи динамічного аналізу.

Автоматизація документування та формування звітів у пентесті стала одним із напрямів, де LLM демонструють найбільш виразну практичну цінність. У традиційному процесі значна частина часу витрачається не на пошук чи експлуатацію вразливостей, а на оформлення результатів: структурування артефактів, підбір доказів, формування описів, аргументація ризиків і рекомендацій. Цей етап потребує не лише технічної компетентності, але й уміння чітко викладати складні механізми у зрозумілому вигляді від кодових фрагментів до логічних сценаріїв атаки. ШІ-моделі стали ефективним інструментом саме тут, оскільки здатні перетворювати великі обсяги технічної інформації логи, PoC, фрагменти конфігурацій, SQL-запити, мережеві трасування у впорядковану, структуровану й уніфіковану форму, зводячи до мінімуму рутинну складову роботи аудитора [14].

Першим ключовим напрямом є нормалізація артефактів. Під час пентесту утворюється велика кількість неоднорідних даних, які не мають єдиного стандарту опису: записів локального сховища, HTTP-відповідей, скріншотів, коду, що виконується під час атаки, чи фрагментів журналів подій. LLM може виконувати роль семантичного агрегатора, що приводить такі дані до спільної моделі подання класифікує артефакти, вилучає суттєві елементи, відокремлює вхідні дані, параметри, контекст виконання і результат впливу. У підсумку аудитор отримує узагальнений опис, позбавлений технічного «шуму», але збережений у достатньо детальній формі для подальшого документування.

Другий напрям полягає в автоматичній генерації опису вразливостей. Модель здатна зіставити знайдені артефакти з відомими класами проблем і сформулювати первинний опис механізму експлуатації, включно з поясненням причини виникнення вразливості, умов її використання, можливих обмежень і сценаріїв впливу на систему. LLM досить ефективно ідентифікує структуру стандартних патернів SQL-ін'єкції, IDOR, неправильну валідацію токенів, небезпечні конфігураційні дозволи та формує опис у відповідності зі стандартними форматами (OWASP, PTES, чи внутрішні корпоративні стандарти). При цьому модель може доповнити висновок деталями, які інколи залишаються поза увагою: наприклад, взаємозв'язок між конфігурацією мобільного застосунку та поведінкою серверної частини [15].

Третій напрям формування технічних і бізнес-ризиків на основі зібраних даних. LLM здатна оцінити потенційний вплив знайденої вразливості не лише в технічному контексті, а й у межах бізнес-процесів, до яких залучений застосунок. Наприклад, модель може описати, як витік токенів доступу впливає на модель автентифікації, чи створює він ризик для сесій користувачів, чи може призвести до компрометації фінансових операцій. Вона здатна перетворювати технічний вияв у структуровану оцінку ризику, охоплюючи рівні впливу, можливі сценарії атаки, ймовірність її реалізації та відповідні рекомендації. Такий підхід дозволяє суттєво скоротити час, необхідний для підготовки

формального звіту, і водночас підвищує якість документації завдяки більш системному опису проблеми.

Отже, інструменти автоматизації документування, побудовані на основі LLM, фактично зміщують фокус пентестера з рутинного оформлення результатів на аналіз і валідацію висновків, зберігаючи його увагу для найскладніших аспектів аудиту та підвищуючи загальну продуктивність процесу.

Узагальнюючи огляд сучасних AI-інструментів для автоматизації окремих етапів тестування на проникнення, можна стверджувати, що вони займають усе помітніше місце у практиці мобільної безпеки. Їхні сильні сторони проявляються насамперед у здатності працювати з великими обсягами різнорідних даних, виявляти приховані зв'язки між компонентами застосунку та виконувати високорівневу інтерпретацію технічних артефактів, яку класичні інструменти побудовані на сигнатурах часто не здатні забезпечити. LLM демонструють ефективність у статичному аналізі коду та конфігурацій, де вони розпізнають небезпечні патерни і логічні залежності, що виходять за межі простих правил. Вони ж забезпечують значний прогрес у динамічному аналізі: класифікують локальні артефакти, виявляють секрети та інтерпретують поведінку застосунку в умовах, коли вручну це потребувало б суттєвих витрат часу. І нарешті, у сфері документування такі моделі автоматизують найбільш рутинні етапи нормалізацію доказів, опис механізмів експлуатації та формування ризиків, забезпечуючи стабільну структурну якість звітів.

Втім, попри явні переваги, існуючі рішення мають і низку слабких сторін. Комерційні та відкриті інструменти часто працюють у межах фіксованих шаблонів або обмежених фрагментів коду, не забезпечуючи повноцінного контекстного аналізу на рівні застосунку. Їхні можливості залежать від якості попередньої нормалізації даних, що покладає істотну відповідальність на пентестера. Крім того, рівень деталізації й точності висновків інколи суттєво варіюється, що створює ризики хибнопозитивних або хибнонегативних результатів. У багатьох випадках інструменти зосереджені лише на окремих

етапах аналізу і не дають цілісного підходу, який охоплював би повний цикл мобільного пентесту від статичного аналізу до формування звіту.

Саме ці обмеження формують обґрунтовану потребу у розробці власних автоматизованих рішень, адаптованих до специфіки конкретного середовища та завдань дослідження. Мобільні застосунки істотно відрізняються за архітектурою, підходами до зберігання даних і логікою взаємодії з бекендом, що робить універсальні інструменти недостатньо гнучкими. Використання ШІ у цільових автоматизаціях семантичному аналізу коду та конфігурацій, оцінці локального сховища та автоматизації формування звітів дає можливість створити інтелектуальний компонент пентест-процесу, який працює саме в межах реальних обмежень тестованого середовища та забезпечує глибину аналізу, недосяжну для типових рішень. У цьому контексті власні модулі автоматизації, описані у розділі 2.2, виступають логічним розвитком виявлених тенденцій і відповідають практичним потребам сучасної мобільної безпеки.

2.2. Практичне тестування AI-інструментів на прикладі вибраного середовища

У цьому дослідженні як тестове середовище обрано мобільний застосунок Damn Vulnerable Bank App (DVBA) спеціально створений приклад Android-додатка з навмисно закладеними вразливостями, орієнтований на навчання та практичне відпрацювання методик мобільного пентесту. На відміну від узагальнених демонстраційних проєктів, DVBA моделює реалістичну банківську логіку: авторизацію, роботу з балансом, транзакції, мережеву взаємодію з бекендом, використання токенів доступу та локальне зберігання даних. Це робить його надзвичайно цінним для тестування AI-автоматизацій, оскільки застосунок містить повний спектр типових проблем, що зустрічаються у реальних фінансових мобільних продуктах. Перевагою DVBA також є відкрита архітектура, наявність вихідного коду та можливість легально модифікувати або досліджувати його без будь-яких юридичних обмежень.

Для проведення статичного та динамічного аналізу використовувалася комбінація інструментів, що охоплює всі ключові етапи пентесту мобільних застосунків. Базовий інструментальний стек включав adb для взаємодії з емулятором Android, вилучення локальних файлів та журналів роботи додатка; jadx для декомпіляції APK у читабельний Java-код і витягнення конфігурацій, передусім AndroidManifest.xml; sqlite3 для аналізу локальних баз даних, що використовуються додатком; а також frida та objection для динамічної інструменталізації й дослідження поведінки застосунку під час виконання. Ця комбінація дає змогу отримати повний набір артефактів, необхідних для роботи AI-модулів: вихідний код, конфігураційні файли, локальні дані, проміжні значення під час виконання та мережеві взаємодії.

Методика отримання артефактів вибудовувалася так, щоб забезпечити чисте розділення між статичним і динамічним аналізом і водночас надати AI-модулям максимально структуровані вхідні дані. На статичному етапі виконувалася декомпіляція APK за допомогою jadx, після чого отримані файли сканувалися для виявлення підозрілих викликів API, hardcoded secrets, конфігураційних помилок у маніфесті та потенційних логічних вразливостей. Також вилучалися всі елементи локального сховища SQLite-бази, SharedPreferences, кешовані файли через adb і аналізувалися окремо. На динамічному етапі застосунок запускався в емуляторі, а frida/objection використовувалися для перехоплення викликів, спостереження за зміною стану локального сховища, перевірки поведінки WebView та мережевих операцій. Отримані статичні та динамічні артефакти потім передавалися в AI-модулі для семантичного аналізу, класифікації ризиків і формування висновків.

У сукупності DVBA забезпечує унікальне поєднання відкритості, реалістичної логіки, різноманітності помилок і структурованості коду, що робить його оптимальним вибором для демонстрації можливостей інтелектуальної автоматизації мобільного пентесту.

Вибір саме цих трьох AI-модулів ґрунтується на їх відповідності реальним етапам мобільного пентесту та на тому, що кожен із них вирішує

конкретні, добре відомі проблеми ручного аналізу, які залишаються трудомісткими навіть для досвідчених аудиторів. Мета полягала не в побудові універсальної автоматизації, а у виокремленні тих ділянок роботи, де використання LLM може дати найбільший практичний ефект підвищити точність, прискорити процес та виявити логічні залежності, які традиційні інструменти часто ігнорують.

Перший модуль семантичний аналіз коду та конфігурацій був обраний тому, що саме статичний аналіз мобільного застосунку є найбільш інформаційно насиченим, але водночас найскладнішим етапом ручного аудиту. `AndroidManifest.xml`, декомпільований код і ресурсні файли формують основу моделі загроз застосунку, однак їх обсяг і неоднорідність ускладнюють побудову зв'язків між дозволами, компонентами та внутрішньою логікою. Людина може прогледіти нетривіальні взаємодії, наприклад зв'язок між `exported Activity`, `hardcoded endpoint`-ом у ресурсах і небезпечним API-викликом у коді. LLM, на відміну від класичного SAST, здатна інтерпретувати ці фрагменти у спільному контексті та формувати семантичні залежності, що дає змогу виявляти логічні, а не лише структурні вразливості. Тому цей модуль обраний для автоматизації саме на етапі статичного аналізу та моделювання вразливостей.

Другий модуль аналіз локального сховища охоплює етап динамічного тестування, пов'язаний зі збором артефактів зі смартфона після встановлення та запуску застосунку. На практиці локальне сховище залишається одним із найбільш недооцінених векторів атак: розробники часто ненавмисно залишають у SQLite-базах, `SharedPreferences` або журналах чутливу інформацію, яка не піддається простому пошуку сигнатурами. Ручний аналіз таких даних потребує значного часу, адже аудитор повинен переглянути численні таблиці, файли й ключі, щоб зрозуміти їх призначення та оцінити можливі наслідки. LLM здатна автоматично класифікувати знайдені артефакти, визначати характер збережених даних, ідентифікувати токени, облікові записи, незашифровані конфіденційні значення та випадки, коли замість `Keystore` або `Keychain` використано

plaintext-зберігання. Завдяки цьому другий модуль логічно доповнює перший, охоплюючи вже не конфігураційні, а поведінкові артефакти застосунку.

Третій модуль автоматичне формування звіту був обраний як завершальний, оскільки саме етап документування є одним із найбільш трудозатратних і водночас формалізованих у всьому процесі пентесту. Перетворення артефактів, логів, знайдених вразливостей і PoC у структурований технічний звіт традиційно займає значно більше часу, ніж сам пошук вразливостей. LLM здатна систематизувати отримані дані, класифікувати вразливості, формувати технічні описи та рекомендації, а також генерувати узгоджені розділи звіту з урахуванням контексту тестованого застосунку. Такий модуль логічно завершує попередні два: перший забезпечує аналітичні висновки про структуру застосунку, другий про безпеку локального середовища, а третій перетворює ці результати у фінальний артефакт, який має практичну цінність для замовника пентесту.

У сукупності обрані модулі відображають три ключові етапи мобільного тестування на проникнення: статичний аналіз, динамічний аналіз артефактів та документування результатів. Кожен із них представляє ділянку, де можливості LLM забезпечують не лише автоматизацію, але й якісно вищий рівень інтерпретації даних, що робить обраний підхід обґрунтованим та релевантним у контексті дослідження.

2.2.1 Реалізація та експериментальна перевірка AI-модуля №1 для семантичного аналізу конфігурацій і статичних артефактів мобільного застосунку

Метою даного модуля є автоматизація первинного статичного аналізу мобільного застосунку шляхом використання великої мовної моделі (LLM), здатної інтерпретувати Android-конфігурації та структурні артефакти застосунку. На відміну від класичного ручного підходу, де аудитор поступово переглядає `AndroidManifest.xml`, рядкові ресурси та фрагменти

декомпільованого коду, запропонований модуль дозволяє делегувати LLM завдання виявлення надлишкових дозволів, неправильно експортованих компонентів, `hardcoded secrets` та потенційних логічних вразливостей.

Застосунок DVBA використовувався тут як тестове середовище; його характеристика та причини вибору були детально наведені в попередньому розділі, тому на цьому етапі достатньо зазначити, що DVBA має чітко визначену структуру, типові для банківських застосунків механізми автентифікації та свідомо вбудовані вразливості, що дозволяє коректно оцінити можливості AI-аналізу.

Підготовка артефактів для подальшого аналізу є необхідним етапом перед запуском AI-модуля, оскільки саме ці дані формують інформаційний контекст, на основі якого модель інтерпретує властивості застосунку та визначає потенційні вразливості. Передусім було отримано базові метадані APK за допомогою інструмента `arkid`. Цей крок дозволив встановити ім'я пакета, версію застосунку, а також значення `minSdk` і `targetSdk`. Рівні SDK є важливо важливими у статичному аналізі, оскільки визначають, які механізми безпеки доступні застосунку на платформі. Використання застарілих версій Android API обмежує наявні засоби захисту, а неправильно підібраний `targetSdk` може призвести до некоректної роботи `permission`-моделі або механізмів ізоляції компонентів. Таким чином, метадані забезпечують початкове розуміння того, у якому середовищі функціонує застосунок та які базові ризики можуть мати місце через його конфігураційні обмеження.

Після отримання метаданих ключовим завданням стало витягнення `AndroidManifest.xml` у придатному для аналізу вигляді. Для цього застосовано утиліту `Androguard`, яка дозволяє перетворити бінарний маніфест у читабельний XML-формат. Маніфест є центральним артефактом у безпековому аналізі мобільного застосунку, оскільки саме в ньому визначаються дозволи, доступні застосунку, а також компоненти `Activity`, `Service`, `Receiver` та `Provider` з їх атрибутами, зокрема рівнем доступності (`exported`), `requirement`-пермішенами та наборами `intent`-фільтрів. Ці елементи визначають, які частини застосунку

можуть бути доступні іншим процесам, які дії дозволені без автентифікації та як обробляються зовнішні взаємодії. Будь-які помилки або надлишкові конфігурації в маніфесті створюють критичні передумови для логічних вразливостей, тому якісний аналіз цього файлу є фундаментальним.



Рисунок 2.1 Приклад використання промпту та вивід результату аналізу LLM

Окремо було проведено витягнення надбудовних ресурсів, зокрема рядкових значень із файлу strings.xml, за допомогою декодування ресурсу argsc. Рядкові ресурси часто містять конфігураційні параметри, внутрішні URL, текстові маркери функціональних режимів та, у гіршому випадку, hardcoded secrets API-ключі, токени доступу, ідентифікатори середовища або порогові значення, що впливають на авторизацію чи логування. Аналіз рядкових ресурсів допомагає зрозуміти внутрішню логіку застосунку, його структуру взаємодії із зовнішніми сервісами та наявність залишкових конфігурацій від середовища розробки.

Завершальним етапом підготовки стало отримання інформації про підпис застосунку. Механізм підпису визначає цілісність APK і гарантує, що застосунок не було модифіковано після створення. Аналіз сертифіката дозволяє

встановити алгоритм підпису та перевірити, чи не використовується застарілий або криптографічно слабкий підхід. Якщо застосунок підписаний ключем з ненадійним алгоритмом, існує ризик, що зловмисник може створити модифіковану версію APK або обійти перевірку цілісності. Тому оцінка параметрів підпису є важливим елементом комплексного аналізу безпеки.

Таким чином, підготовка артефактів охопила збір усіх ключових статичних даних від системних метаданих до конфігурацій та ресурсів. Саме вони сформували інформаційну основу, на якій працює AI-модуль, забезпечуючи можливість виконувати семантичний аналіз та виявляти пов'язані між собою конфігураційні вразливості.

Проектування системи промптів стало центральним елементом побудови AI-модуля, оскільки саме промптинг визначає спосіб взаємодії великої мовної моделі з артефактами застосунку та глибину отриманого аналізу. Загальна логіка побудована на каскадному підході, де модель отримує дані поступово, у структурованому вигляді, що дає змогу уникнути змішування контекстів та забезпечує точність семантичних висновків. Такий підхід повністю відповідає природі статичного аналізу мобільних застосунків: спершу оцінюються загальні властивості, потім конфігураційні параметри, далі деталізовані ресурси та супутня метаінформація, а на завершення виконується побудова зв'язків між усіма знайденими елементами.

Першим елементом системи є системний промпт, який визначає роль моделі та рамки її роботи. На цьому етапі моделі задається контекст експерта з мобільної безпеки, обмежується спектр допустимих інтерпретацій та визначаються стандарти, на які вона має орієнтуватися, зокрема OWASP MASVS [5]. Таке попереднє формування ролі є принципово важливим, оскільки саме воно задає очікувану глибину аналізу, стиль інтерпретації та категоризацію виявлених проблем. Крім цього, системний промпт встановлює правила поведінки моделі: необхідність працювати виключно на основі наданих артефактів, уникати припущень, дотримуватися структурованого формату висновків і пояснювати ризики з урахуванням контексту мобільної платформи.

Після визначення системного рівня були сформовані окремі промпти для аналізу різних типів артефактів. Кожен із них орієнтований на конкретну частину структури застосунку та спрямований на отримання вузькоспеціалізованої інформації. Промпт, спрямований на аналіз даних із `apkid`, забезпечує інтерпретацію метаданих APK, включно з рівнями SDK та параметрами пакета, дозволяючи моделі визначити потенційні ризики, пов'язані із застарілими або некоректно налаштованими платформеними параметрами. Промпт для `AndroidManifest.xml` зосереджує увагу на дозволах, експортованих компонентах та логіці взаємодії між ними. Промпт для ресурсів `arsc` дає можливість моделі проаналізувати рядкові значення з точки зору наявності секретів, конфігураційних маркерів та внутрішніх URL. Окремий промпт для даних про підпис дає LLM змогу оцінити криптографічну стійкість схеми підпису та відповідність сучасним рекомендаціям Android. Розбиття аналізу на ці тематичні групи дозволяє уникнути надмірної концентрації різномірних даних у одному запиті та забезпечує чистоту й точність міркувань моделі на кожному етапі.

apkid.txt
Document

Проаналізуй наведені APK metadata (apkid output) і поверни структуровані висновки у форматі JSON . Визнач: Чи відповідає `minSdk` рівню безпеки сучасних Android-платформ. Чи створює `targetSdk` проблеми з безпековими механізмами (WebView, storage, permissions). Чи може `sharedUserId` або інші властивості створювати ризик. Які загальні висновки щодо безпеки можна зробити з цих метаданих. Формат відповіді: `"classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }`

```
json Copy code
{
  "classification": "APK Metadata Security Analysis",
  "security_issue": "minSdkVersion і targetSdkVersion не вказані у наведеному метаданих",
  "severity": "medium",
  "evidence": "Наявна інформація: packageName: com.app.dammvulnerablebank, versionCode: 1",
  "recommendation": "Перевірити AndroidManifest.xml на предмет явної вказівки minSdkVersion"
}
```

Пояснення:

- **minSdkVersion:** Якщо вона не вказана або занадто низька, застосунок може бути встановлений на пристрої зі старими, уразливими до експлойтів, версіями Android.
- **targetSdkVersion:** Старі значення (нижче 29) не активують сучасні захисні механізми (scoped storage, сучасний контроль WebView, runtime permissions).
- **sharedUserId:** Його відсутність — це добре, бо зменшує ризик ескалації привілеїв між

Рисунок 2.2 Приклад використання промпту та вивід результату аналізу LLM

Завершальним елементом каскадної структури є промпт для семантичного узагальнення. На цьому етапі модель отримує всі попередні висновки та має побудувати цілісну картину взаємозв'язків між конфігураційними, ресурсними та криптографічними властивостями застосунку. Основним завданням цього етапу є формування логічних ланцюгів, що пояснюють, яким чином окремі конфігураційні недоліки можуть поєднуватися у комплексні сценарії вразливостей. Наприклад, надлишковий дозвіл у маніфесті може компенсуватися або, навпаки, посилюватися наявністю відкритих URL у ресурсах, тоді як слабкий підпис APK створює ґрунт для модифікації застосунку з метою експлуатації цих ланцюжків. Промпт фінального узагальнення дає змогу моделі інтегрувати всі знайдені артефакти в єдиний семантичний простір і оцінити, які з виявлених взаємозв'язків становлять реальну загрозу.

На основі всіх попередніх відповідей виконай семантичний аналіз взаємодії компонентів застосунку. Сформує логічні ланцюги типу: конфігурація → дозволи → компоненти → ресурси → можливість атаки. Визнач: – комплексні вразливості, що виникають через поєднання кількох факторів; – можливі сценарії експлуатації; – технічні та бізнес-наслідки. Формат відповіді: – Vulnerability Name – Root Cause – Full attack chain – Affected components – Risk level – Recommended mitigation

json
Copy code

```

[
  {
    "Vulnerability Name": "Intent Hijacking та несанкціонований доступ до критичних ф",
    "Root Cause": "Недостатній контроль над exported Activity у поєднанні з відкритим",
    "Full attack chain": "Низький targetSdkVersion (29) не застосовує сучасних захист:",
    "Affected components": [
      "com.app.damnvulnerablebank.CurrencyRates (BROWSABLE intent-filter, exported by",
      "com.app.damnvulnerablebank.SendMoney (exported=true, без permission-guard)",
      "com.app.damnvulnerablebank.ViewBalance (exported=true, без permission-guard)",
      "androidx.biometric.DeviceCredentialHandlerActivity (exported=true)",
      "usesCleartextTraffic=true (application-level risk)"
    ],
    "Risk level": "high",
    "Recommended mitigation": [
      "Додати permission-guard або custom permission для всіх exported Activity з кри",
      "Ограничити usesCleartextTraffic=false і налаштувати networkSecurityConfig.",
      "Підняти targetSdkVersion до >=33, ↓ застосувати сучасні політики безпеки ком",
      "Переглянути й обмежити intent-filter-и, особливо BROWSABLE, лише для необхідни"
    ]
  }
]

```

Рисунок 2.3 Приклад використання промпту та вивід результату аналізу LLM

Таким чином, структура промптів формує логічну та технологічно обґрунтовану методику аналізу, у якій LLM рухається від інтерпретації базових властивостей до формування комплексних висновків про безпеку застосунку. Такий підхід забезпечує не лише точність аналізу, але й відтворюваність та прозорість процесу, що є важливим у контексті академічної роботи та практичного застосування штучного інтелекту в сфері безпеки.

У межах реалізації AI-модуля було обрано модель GPT-4.1 як основний механізм семантичного аналізу. Це рішення зумовлене тим, що GPT-4.1 демонструє стабільну здатність працювати з великими обсягами структурованих і напівструктурованих даних, а також упевнено інтерпретує конфігурації, логіку компонентів Android і зв'язки між ними. На відміну від спрощених моделей, GPT-4.1 дозволяє не лише виявляти окремі ознаки вразливостей, а й реконструювати причинно-наслідкові ланцюги, що виникають при взаємодії permissions, exported-компонентів, ресурсних значень та параметрів підпису. Висока точність у класифікації дозволів, здатність контекстно оцінювати роль компонентів та розпізнавати потенційно небезпечні патерни зробили модель оптимальним вибором для задачі статичного аналізу мобільного застосунку [9].

У процесі роботи модуля GPT-4.1 використовувалася на всіх етапах від аналізу конфігурацій до формування комплексних висновків. Під час первинного аналізу конфігурацій модель отримувала AndroidManifest.xml та інтерпретувала його з урахуванням контексту платформи Android і вимог OWASP MASVS. GPT-4.1 визначала надлишкові дозволи, некоректно експортовані компоненти та конфігурації, які потенційно змінювали модель загроз застосунку. Завдяки можливості глибокого контекстуального аналізу модель не обмежувалася фіксацією формальних відхилень, а враховувала їхню взаємодію з іншими елементами застосунку.

На наступному етапі GPT-4.1 здійснювала аналіз ресурсів, зокрема даних із файлу strings.xml. Модель ефективно ідентифікувала потенційні hardcoded

secrets, внутрішні URL, залишкові конфігурації розробницьких середовищ та інші елементи, які могли вказувати на неправильне управління конфіденційною інформацією. Важливо, що GPT-4.1 інтерпретувала ресурси не ізольовано, а з урахуванням того, які дозволи та компоненти присутні в маніфесті, що дозволяло визначити більш складні сценарії ризику.

Аналіз підпису застосунку також виконувався моделлю: GPT-4.1 оцінювала алгоритм підпису, ключові параметри сертифіката та можливі наслідки використання застарілих або слабких схем підпису. Модель коректно співвідносила ці параметри з особливостями сучасного Android App Signing, що дало змогу сформулювати точні висновки щодо криптографічної стійкості та цілісності застосунку.

Завершальним етапом стало семантичне узагальнення, у межах якого GPT-4.1 інтегрувала всі отримані раніше дані та будувала логічні ланцюги взаємодії між ними. Саме на цьому етапі модель демонструвала найбільшу цінність: окремі, на перший погляд, незначні проблеми поєднувалися у цілісні сценарії експлуатації. Наприклад, надлишковий дозвіл у маніфесті міг посилювати ризик, породжений відкритим компонентом, або взаємодіяти з hardcoded ресурсами, що спрощувало доступ до функцій, які мали бути обмежені. Зрештою GPT-4.1 формувала інтегральну модель ризику, що відображала комплексну картину вразливостей та їх потенційні наслідки.

Таким чином, вибір GPT-4.1 забезпечив можливість не лише автоматизувати окремі етапи аналізу, але й отримати висновки, які наближаються за точністю до експертної оцінки. Модель продемонструвала здатність працювати з неоднорідними артефактами, поєднувати їх у єдиний логічний контекст і формувати висновки, що значно виходять за межі стандартних сигнатурних перевірок.

Найважливішим результатом стала здатність моделі сформулювати комплексні логічні зв'язки між окремими артефактами. Наприклад, модель пов'язувала відкритий компонент із надлишковим дозволом і наявністю у ресурсах внутрішнього URL, що вказувало на потенційний сценарій атаки, у

межах якого зловмисник може ініціювати взаємодію зі слабо захищеним компонентом і отримати доступ до чутливих функцій застосунку. Таким чином, результати тестування продемонстрували, що LLM може виконувати не лише лінійний або сигнатурний аналіз, а й будувати причинно-наслідкові моделі, які дозволяють точніше оцінити реальний рівень ризику та потенційні шляхи експлуатації вразливостей.

Оцінювання ефективності запропонованого підходу показало, що застосування LLM суттєво змінює характер первинного статичного аналізу мобільного застосунку. Модуль продемонстрував здатність швидко інтерпретувати конфігураційні та ресурсні артефакти, які зазвичай потребують значних часових витрат при ручному перегляді. Завдяки структурованому промптингу модель не просто аналізувала окремі елементи, а будувала логічні зв'язки між ними, виявляючи комплексні залежності, що формують потенційні сценарії експлуатації. Саме ця властивість уміння інтегрувати дані різної природи та робити семантичні висновки стала ключовою перевагою підходу, оскільки вона виходить за межі традиційних сигнатурних або патерн-орієнтованих методів аналізу.

У практичному вимірі застосування LLM дозволило значно скоротити час, необхідний для попереднього аудиту структури застосунку, і одразу виокремити ті конфігураційні елементи, які потребують детальнішого дослідження. Водночас результати тестування підтвердили, що модель не може повністю замінити експерта: частина висновків потребує додаткової технічної перевірки або зіставлення з реальною поведінкою застосунку. Проте навіть за таких умов LLM виступає ефективним асистивним інструментом, який зменшує обсяг рутинної роботи, підвищує точність початкової оцінки й дає змогу зосередитися на більш складних аспектах безпеки.

Таким чином, ефективність підходу полягає в його здатності підсилювати процес аналізу, роблячи його швидшим, структурованішим і глибшим. Модель забезпечує значну економію часу та ресурсів, водночас збільшуючи повноту

виявлення логічних залежностей, що є важливою перевагою у контексті автоматизації процесів тестування на проникнення.

Важливо, що модель здатна працювати з неоднорідними джерелами інформації та інтегрувати їх у єдину семантичну картину, що підсилює роль автоматизації у тестуванні на проникнення. Хоча LLM не замінює експерта й потребує додаткової валідації своїх висновків, вона значно зменшує обсяг рутинної роботи та скорочує час, необхідний для первинної оцінки безпеки мобільного застосунку.

2.2.2 Реалізація та експериментальна перевірка AI-модуля №2 для аналізу локального сховища мобільного застосунку та виявлення ризиків зберігання даних

Аналіз локального сховища мобільного застосунку є одним із ключових етапів пентесту, оскільки саме в цьому шарі часто залишаються артефакти, які безпосередньо впливають на конфіденційність, цілісність та доступність даних користувача. Мобільні застосунки працюють у контексті обмежених прав, однак це не гарантує належного рівня захисту даних: розробники нерідко зберігають службову інформацію у незашифрованому вигляді, не використовують Keystore або Keychain для захисту секретів і залишають у локальних базах даних службові журнали, токени, маркери сесій чи інші чутливі значення. У реальній атаці компрометація смартфона, доступ до резервної копії або отримання файлової системи через інструменти root/adb надає зловмиснику прямий доступ до цих артефактів, що може призвести до повного обходу автентифікації чи відновлення доступу до облікового запису користувача. Саме тому локальне сховище потрібно розглядати як критичну частину поверхні атаки.

Водночас ручна перевірка локальних артефактів є вкрай трудомістким процесом. SQLite-бази можуть містити десятки таблиць і сотні записів, у яких чутливі дані не завжди очевидні – вони можуть зберігатися у вигляді хешів, кодувань або службових параметрів. SharedPreferences, будучи простими XML-файлами, часто розпорошують конфігурацію між різними файлами, де

кожен ключ має власний формат і призначення. Файлова система застосунку включає журнальні файли, кешовані відповіді API, службові логи та залишкові артефакти після попередніх версій застосунку. Аналітик змушений вручну проходити через кожен файл, читати таблиці, інтерпретувати значення та намагатися зрозуміти, які з них мають безпекове значення. Цей процес не лише займає багато часу, але й часто призводить до пропуску потенційних проблем через людський фактор складність структури даних, нестачу контексту або неповноту документації.

Використання LLM розв'язує одразу кілька суттєвих проблем. По-перше, модель здатна швидко класифікувати великі масиви даних, розуміючи семантичний зміст ключів, назв таблиць та значень. Це дозволяє автоматично виявляти токени, session identifiers, credentials, маркери конфігурації або внутрішні службові параметри, які могли би залишитися непоміченими під час ручного перегляду. По-друге, LLM може оцінити, чи є збережені дані результатом належного шифрування або чи виглядають вони як plaintext, і надати аргументоване пояснення, чи застосовано Keystore / Keychain або небезпечну альтернативу. По-третє, модель здатна формувати логічні висновки: співвіднести дані з локального сховища з іншими артефактами застосунку, визначити потенційні сценарії їх використання зловмисником і надати рекомендації щодо пом'якшення ризиків. У підсумку LLM істотно прискорює аналіз, мінімізує ймовірність пропуску критичних значень і перетворює великий обсяг розрізнених файлів у структуровані, зрозумілі та безпеково релевантні висновки.

Таким чином, автоматизація аналізу локального сховища за допомогою LLM не лише усуває рутинні та ресурсомісткі частини роботи, але й підвищує глибину та точність оцінки, дозволяючи аудитору зосередитися на інтерпретації результатів і моделюванні реалістичних сценаріїв атак. Це робить дослідження значно ефективнішим і наближає процес пентесту до рівня, де комплексний аналіз даних стає доступним навіть за обмежених часових рамок.

Для підготовки артефактів локального сховища до подальшого аналізу було виконано повноцінне вилучення файлової структури застосунку з пристрою за допомогою ADB. Оскільки всі дані мобільного застосунку зберігаються у директорії `/data/data/<package>/`, саме вона є основним джерелом як конфіденційних, так і службових артефактів, включно з базами даних, файлами налаштувань, журналами, кешами браузерних компонентів та внутрішніми структурами WebView. Для тестового застосунку DVBA команда `adb pull /data/data/com.app.damnulnerablebank ./damnulnerablebank_dump` дозволила отримати повний дамп робочої директорії застосунку у локальне середовище, не змінюючи структуру каталогів та зберігаючи всі артефакти у їх вихідному вигляді.

```

emu64a:/data/data/com.app.damnulnerablebank # ls -lR /data/data/com.app.damnulnerablebank
/data/data/com.app.damnulnerablebank:
total 40
drwxrwx--x 2 u0_a231 u0_a231      4096 2025-12-08 11:43 app_textures
drwx----- 3 u0_a231 u0_a231      4096 2025-12-08 11:43 app_webview
drwxrws--x 5 u0_a231 u0_a231_cache 4096 2025-12-08 11:43 cache
drwxrws--x 2 u0_a231 u0_a231_cache 4096 2025-12-08 11:16 code_cache
drwxrwx--x 2 u0_a231 u0_a231      4096 2025-12-08 11:43 shared_prefs

/data/data/com.app.damnulnerablebank/app_textures:
total 0

/data/data/com.app.damnulnerablebank/app_webview:
total 4140
-rw----- 1 u0_a231 u0_a231 4194304 2025-12-08 11:43 BrowserMetrics-spare.pma
drwx----- 8 u0_a231 u0_a231      4096 2025-12-08 11:43 Default
-rw----- 1 u0_a231 u0_a231         80 2025-12-08 11:43 last-exit-info
-rw----- 1 u0_a231 u0_a231       134 2025-12-08 11:43 pref_store
-rwx----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 variations_seed_new
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 variations_stamp
-rw----- 1 u0_a231 u0_a231        32 2025-12-08 11:43 webview_data.Lock

/data/data/com.app.damnulnerablebank/app_webview/Default:
total 176
-rw----- 1 u0_a231 u0_a231 24576 2025-12-08 11:43 Cookies
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 Cookies-journal
drwx----- 3 u0_a231 u0_a231      4096 2025-12-08 11:43 Local\ Storage
drwx----- 2 u0_a231 u0_a231      4096 2025-12-08 11:43 PersistentOriginTrials
drwx----- 2 u0_a231 u0_a231      4096 2025-12-08 11:43 Session\ Storage
drwx----- 3 u0_a231 u0_a231      4096 2025-12-08 11:43 Shared\ Dictionary
-rw----- 1 u0_a231 u0_a231 90112 2025-12-08 11:43 Web\ Data
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 Web\ Data-journal
drwx----- 3 u0_a231 u0_a231      4096 2025-12-08 11:43 blob_storage
drwx----- 3 u0_a231 u0_a231      4096 2025-12-08 11:43 shared_proto_db

/data/data/com.app.damnulnerablebank/app_webview/Default/Local Storage:
total 8
drwx----- 2 u0_a231 u0_a231 4096 2025-12-08 11:43 leveldb

/data/data/com.app.damnulnerablebank/app_webview/Default/Local Storage/leveldb:
total 32
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 000003.Log
-rw----- 1 u0_a231 u0_a231       16 2025-12-08 11:43 CURRENT
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 LOCK
-rw----- 1 u0_a231 u0_a231      289 2025-12-08 11:43 LOG
-rw----- 1 u0_a231 u0_a231       41 2025-12-08 11:43 MANIFEST-000001

/data/data/com.app.damnulnerablebank/app_webview/Default/PersistentOriginTrials:
total 8
-rw----- 1 u0_a231 u0_a231         0 2025-12-08 11:43 LOCK

```

Рисунок 2.4 Файли, що зберігаються додатком на файлової системі

Оскільки подальший аналіз передбачав роботу LLM з нормалізованими файлами, окремим кроком було виконано рекурсивне копіювання всіх файлів у плоску структуру для спрощення обробки. Команда `find damnvulnerablebank_dump -type f -exec cp {} flat_dump/\;` збрала всі артефакти у єдину директорію без вкладених папок, що дозволило зменшити кількість операцій у наступному етапі та забезпечити повний контроль над кожним окремим файлом. Такий підхід унеможлиблює пропуск уміщених у піддиректоріях ресурсів, які іноді містять критичні для аналізу значення.

Отриманий набір артефактів включав файли різної природи та призначення. Були виявлені журнальні файли RocksDB (000003.log, CURRENT, MANIFEST-000001, LOG), стандартні cookie-сховища WebView (Cookies, Cookies-journal), кешовані структури веб-переглядача (Web Data, Web Data-journal, webview_data.lock), а також конфігураційні файли, які потенційно містять чутливу інформацію наприклад, `jwt.xml`, `apiurl.xml`, `WebViewChromiumPrefs.xml`, `AwOriginVisitLoggerPrefs.xml`, `pref_store`. Серед інших артефактів були знайдені файли з інформацією про останній запуск (`last-exit-info`), платформи варіацій (`variations_seed_new`, `variations_stamp`), шрифтові метадані (`font_unique_name_table.pb`) та службові індекси. Окрему увагу привернули SQLite-бази (`db`, `Web Data`), які могли містити таблиці з ключами, історією запитів або службовими параметрами.

Важливо, що усі ці файли були отримані у «сирому» вигляді без попередньої зміни формату, оскільки модуль нормалізації передбачає перетворення різнотипних даних у формат, придатний для подальшого аналізу LLM. SQLite-файли мали бути перетворені у JSON-структури, XML-файли у пари ключ–значення, а журнальні файли розбиті на сегменти з виділенням релевантних фрагментів. На цьому етапі важливим було саме отримання повного набору артефактів, збереження їх початкової структури та забезпечення репрезентативності даних для наступної обробки.

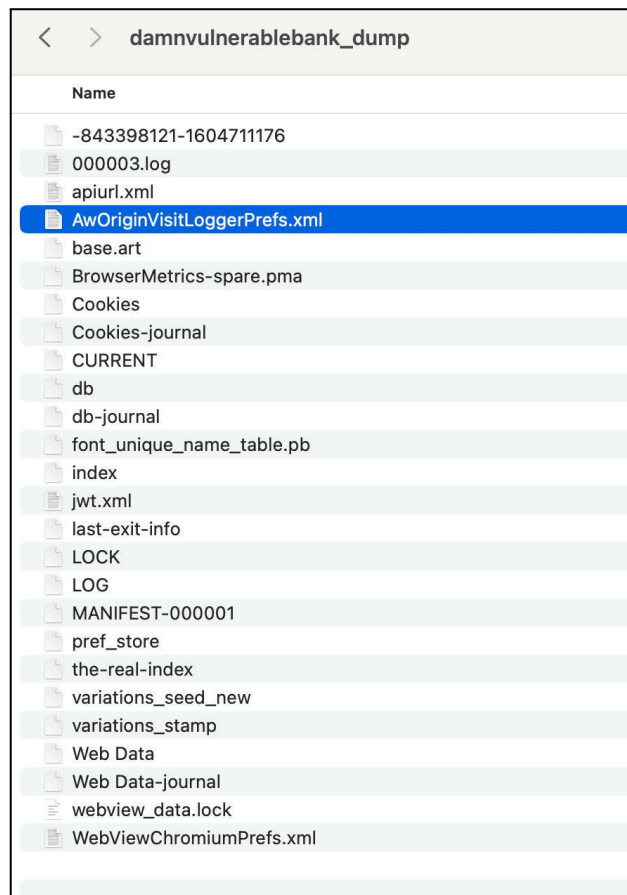


Рисунок 2.5 Файли додатку, які було завантажено для подальшого аналізу

Таким чином, завдання збору артефактів було виконано шляхом прямого вилучення файлової системи застосунку, формування плоскої структури для зручності обробки та первинного огляду отриманих ресурсів. Це створило основу для подальшої нормалізації та автоматизованого аналізу локального сховища за допомогою LLM.

Після збору файлових артефактів постала необхідність привести їх до уніфікованого формату, придатного для подальшого аналізу LLM. Сирі дані з локального сховища мобільного застосунку є надзвичайно різномірними: бази SQLite мають складну реляційну структуру, XML-файли SharedPreferences використовують ієрархічну модель ключів, журнальні файли складаються з фрагментованих технічних записів, а службові файли WebView і RocksDB мають власні внутрішні формати. У такому вигляді ці дані непридатні для прямого семантичного аналізу, оскільки LLM не може ефективно працювати з бінарними форматами, багаторівневими структурами або фрагментованими логами без попереднього перетворення їх у текстові й логічно зв'язані

представлення. Саме тому етап нормалізації є ключовим – він створює єдиний корпус даних, у якому різні типи артефактів зводяться до спільної структури, зрозумілої для моделі.

Першим кроком стала конвертація SQLite-баз у JSON-представлення. Це обґрунтовано тим, що JSON дозволяє передати як структуру таблиць, так і окремі записи у вигляді масивів об'єктів, зберігаючи семантику полів. У процесі перетворення кожна таблиця експортувалася у вигляді масиву словників, де ключ відповідає назві стовпця, а значення – вмісту конкретного запису. Такий формат суттєво спрощує наступний аналіз, оскільки модель отримує не сирі байти SQLite-файлів, а чітко структуровані дані, які можна класифікувати, аналізувати та порівнювати між собою.

Наступним етапом було перетворення файлів SharedPreferences, що зберігаються у форматі XML. Хоча цей формат є текстовим, він містить службові теги й метадані, які ускладнюють аналіз. Тому XML-файли були перетворені у структуру ключ–значення, де кожен параметр представлений у спрощеному вигляді, без зайвих тегів і атрибутів. Завдяки цьому LLM отримує чисті, лаконічні дані, які можна безпосередньо інтерпретувати: ідентифікувати токени, параметри авторизації, конфігурації середовища або службові значення.

Окрему складність становили журнальні файли та службові логи. Їхня структура зазвичай фрагментована, а дані нерідко перемішані з технічними повідомленнями, часовими мітками або двійковими сегментами. Щоб LLM могла обробити їх ефективно, логи були розбиті на окремі текстові сегменти. Кожен сегмент містив завершений фрагмент даних або логічно зв'язаний блок, з якого модель могла зробити висновки щодо потенційного витоку даних, внутрішніх URL, помилок авторизації або debug-інформації. Таке сегментування дозволяє уникнути інформаційного шуму та зменшує ризик втрати контексту.

Name	Date Modified	Size	Kind
base.art.txt	Yesterday at 13:27	127 KB	Plain
font_unique_name_table.pb.txt	Yesterday at 13:27	19 KB	Plain
Web Data.json	Yesterday at 13:27	896 bytes	JSON
-843398121-1604711176.txt	Yesterday at 13:27	757 bytes	Plain
AwOriginVisitLoggerPrefs.xml.json	Yesterday at 13:27	461 bytes	JSON
db.json	Yesterday at 13:27	318 bytes	JSON
jwt.xml.json	Yesterday at 13:27	316 bytes	JSON
LOG.txt	Yesterday at 13:27	289 bytes	Plain
Cookies.json	Yesterday at 13:27	226 bytes	JSON
pref_store.txt	Yesterday at 13:27	134 bytes	Plain
apiurl.xml.json	Yesterday at 13:27	103 bytes	JSON
WebViewChromiumPrefs.xml.json	Yesterday at 13:27	103 bytes	JSON
last-exit-info.txt	Yesterday at 13:27	80 bytes	Plain
the-real-index.txt	Yesterday at 13:27	42 bytes	Plain
webview_data.lock.txt	Yesterday at 13:27	31 bytes	Plain
MANIFEST-000001.txt	Yesterday at 13:27	27 bytes	Plain
index.txt	Yesterday at 13:27	21 bytes	Plain
CURRENT.txt	Yesterday at 13:27	16 bytes	Plain

Рисунок 2.6 Перетворені до JSON файли

D

last-exit-info.txt

Document

D

the-real-index.txt

Document

D

webview_data.lock.txt

Document

D

MANIFEST-000001.txt

Document

D

index.txt

Document

D

CURRENT.txt

Document

Ти — спеціалізований модуль аналізу безпеки локального сховища мобільного застосунку. Проаналізуй вміст кожного файлу і поверни структуровані висновки у форматі JSON: { "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" } Визнач для кожного файлу окремо: 1) чи містить файл токени, паролі, сесійні ідентифікатори; 2) чи є дані у plaintext і чи відповідає формат шифруванню; 3) чи присутні ознаки порушень (нешифровані токени, відсутність expiry, debug-логи тощо). 4) не пиши ніяких додаткових пропозицій по подальшій роботі

Рисунок 2.7 Приклад використання промпту та вивід результату аналізу LLM

У підсумку всі перетворені дані були інтегровані у проміжний корпус, що відповідає єдиному структурному формату, придатному для LLM-аналізу. Після нормалізації дані були проаналізовані LLM, яка класифікувала кожен файл як показує аналіз його призначення, оцінила наявність чутливої інформації та визначила ризики, пов'язані зі способом зберігання.

Водночас аналіз виявив низку артефактів, що містили потенційно небезпечні дані або такі, що потребують додаткової уваги. Файли типу `apiurl.xml.json`, попри відсутність секретів, зберігають відкритий у plaintext внутрішній API endpoint. У контексті тестового середовища це не становить суттєвої загрози, однак у реальному застосунку подібна практика спрощує реконструкцію структури бекенда та збільшує ризик спрямованих атак. Такі конфігурації модель віднесла до середнього рівня ризику з рекомендацією використовувати безпечне сховище або ізолювати production URL.

Найбільш критичні знахідки стосуються двох категорій даних. По-перше, це файли, що містили токени або інші маркери автентифікації у відкритому вигляді. Зокрема, у файлі `jwt.xml.json` модель виявила повноцінний JWT-токен без шифрування, без вказаного часу дії та з ознакою активного логіну. Така практика є грубим порушенням вимог MASVS-STORAGE та створює реальний ризик повного захоплення сесії користувача навіть за умов часткового компрометування пристрою. По-друге, у файлі `-843398121-1604711176.txt` було знайдено HTTP-запис із ETag-токеном та іншими службовими заголовками у plaintext. Подібні артефакти часто використовуються сервером для керування кешем і можуть містити ідентифікатори, корисні для атак повторного відтворення або несанкціонованого доступу. Модель класифікувала такі знахідки як високий рівень ризику та надала рекомендації щодо зашифрованого зберігання журналів або повної заборони збереження токенів у локальних логах.

Таким чином, результати роботи модуля показали, що LLM здатна системно класифікувати артефакти локального сховища, пріоритизувати ризики та виявляти як очевидні, так і приховані порушення політик безпечного зберігання даних. Найбільш критичні проблеми пов'язані зі зберіганням токенів у відкритому вигляді та збереженням службових даних HTTP-комунікацій без шифрування, що підтверджує доцільність використання LLM саме на цьому етапі пентесту.

Оцінювання ефективності другого AI-модуля показало, що використання LLM суттєво змінює характер роботи з локальним сховищем мобільного застосунку, насамперед за рахунок автоматизації аналізу великого масиву гетерогенних артефактів. Замість ручного перегляду десятків файлів різного формату від SQLite-баз і XML-конфігурацій до бінарних індексів і журнальних записів модель змогла систематизувати дані, класифікувати їх і одразу виділити ті елементи, що потенційно містять ризики. Такий підхід зменшує обсяг рутинної роботи практично до мінімуму: аналітик більше не витрачає час на пошук очевидних ознак вразливостей, а зосереджується на інтерпретації результатів та побудові сценаріїв експлуатації.

Узагальнюючи результати роботи другого AI-модуля, можна стверджувати, що застосування LLM у процесі аналізу локального сховища мобільного застосунку довело свою практичну ефективність і доцільність. Модель успішно впоралася з класифікацією великої кількості гетерогенних артефактів, виокремила серед них ті, що становлять реальну загрозу, та надала структуровані рекомендації щодо зменшення ризиків. Автоматизація цього етапу дозволила значно зменшити трудомісткість перевірки, яка у класичному пентесті потребує суттєвих часових витрат через складність і непередбачуваність структури локальних даних.

Найважливішим результатом стало виявлення критичних випадків зберігання чутливої інформації у відкритому вигляді, насамперед JWT-токена та службових HTTP-ідентифікаторів. Це підкреслює здатність моделі не лише ідентифікувати очевидні проблеми, але й виявляти логічні та контекстуальні вразливості, які виникають через неналежні практики зберігання даних. Водночас модуль продемонстрував високу точність у розмежуванні технічних і безпеково значущих артефактів, що дозволяє ефективно скоротити обсяг ручного аналізу та зосередитися на найбільш критичних аспектах.

Разом із тим результати підтверджують, що LLM не може повністю замінити експерта: окремі висновки потребують додаткової технічної валідації, особливо коли йдеться про складні бінарні структури або артефакти зі

специфічним форматом. Проте як інструмент попереднього аналізу та пріоритизації ризиків модель виявилася надзвичайно корисною. Вона формує основу для подальших етапів пентесту, дозволяючи швидко визначати напрямки поглибленої перевірки.

Таким чином, впроваджений AI-модуль продемонстрував значний потенціал у підвищенні ефективності, точності та структурованості аналізу локального сховища. Його використання є обґрунтованим кроком у напрямку широкої автоматизації мобільного пентесту та підтверджує доцільність інтеграції інтелектуальних систем у практику забезпечення інформаційної безпеки.

2.2.3 Реалізація та експериментальна перевірка AI-модуля №3 для автоматизації формування звітності пентесту

Автоматизація документування є ключем до підвищення ефективності пентесту, оскільки створення фінального звіту традиційно є найтривалішим і найбільш рутинним етапом. Хоча технічний аналіз і пошук вразливостей потребують високої кваліфікації, саме оформлення результатів часто займає ще більше часу, супроводжується ризиком помилок і втратою послідовності, особливо коли необхідно інтегрувати численні різноманітні артефакти логи, PoC, конфігураційні дані, результати статичного й динамічного аналізу. Це призводить до нерівномірної якості опису, дублювання або неповного відображення окремих аспектів.

Упровадження LLM як інструменту нормалізації та структурування результатів пентесту вирішує ці проблеми. Модель уніфікує формат подачі матеріалів, забезпечує послідовність термінології, логіку викладу та перетворює набір знайдених артефактів на цілісну аналітичну картину. Таким чином, автоматизація охоплює найменш інтелектуально цінний, але критично важливий етап формування якісного, структурованого та технічно обґрунтованого звіту. Це скорочує час роботи аудитора та підвищує професійний рівень документації.

у цьому процесі працює не як генератор вільного тексту, а як інструмент, що суворо дотримується правил, форматів і термінології.

Першим етапом є нормалізація артефактів: JSON-висновки, логи, PoC, результати аналізу й конфігураційні дані приводяться до єдиного формату назва, опис, рівень ризику, доказовість, контекст, джерело. Це забезпечує уніфіковану обробку даних та дає моделі змогу бачити зв'язки між фрагментами й уникати дублювання.

Другий етап автоматична класифікація вразливостей. LLM визначає тип проблеми (авторизація, конфігурація, криптографія, розкриття даних, логічні помилки тощо) та її ризик, враховуючи важливість компонентів, характер даних і можливість експлуатації. Завдяки цьому класифікація є не шаблонною, а контекстною та технічно обґрунтованою.

На третьому етапі модель формує звітні елементи за заданими шаблонами: опис контексту, сутність вразливості, технічну причину, сценарій експлуатації, PoC, докази й рекомендації. LLM узагальнює та структурує надані артефакти, перетворюючи їх на професійно оформлений, логічно завершений текст.

Четвертий етап побудова повного звіту. Модель інтегрує всі елементи у єдину структуру, створює вступ, огляд ризиків, групує вразливості за категоріями й критичністю, формує висновки та рекомендації. Вона забезпечує стильову однорідність і логічну цілісність документа.

Методика забезпечує майже повну автоматизацію документування пентесту від обробки сирих артефактів до формування професійного звіту. Тестування третього AI-модуля підтвердило, що LLM здатна створювати структуровані, технічно точні звітні елементи без ручних правок, дотримуючись визначених форматів і логіки викладу.

Під час перевірки було сформовано приклади записів для типових вразливостей зберігання JWT у відкритому вигляді, використання ETag-токенів у сервісних журналах, надлишкові дозволи в конфігурації застосунку. У всіх випадках модель коректно визначила суть проблеми, пояснила технічну

причину, сформувала PoC та відповідні рекомендації. Структура відповідала стандартним секціям (Title, Background, Description, Evidence, Recommendation, References), а стиль залишався стислим, нейтральним і професійним.

Перевірка технічної коректності показала, що LLM узгоджено інтерпретує вихідні артефакти, точно використовує реальні докази та не генерує необґрунтованих PoC. Якщо дані містили реальні токени чи конфігураційні значення, вони коректно включалися до доказової бази; якщо експлуатаційного сценарію не існувало, модель класифікувала проблему як конфігураційну та формувала релевантні рекомендації. Це свідчить, що LLM не лише дотримується структури, а й змістово узагальнює інформацію.

Практична користь автоматизації очевидна: тоді як ручне формування одного запису може займати десятки хвилин, модель виконує це за секунди, забезпечуючи стабільну якість і повну однорідність стилю. Усі вразливості оформлюються за стандартом, нормалізованою термінологією та уніфікованою глибиною опису. Для дипломного проєкту це підтверджує доцільність використання AI у документуванні одній із найбільш трудомістких фаз пентесту.

1. Title
User Preference Data Stored in Plaintext in Preference Store

2. Background
The `pref_store.txt` file is used for storing user preference data in the application. No authentication is required to access or modify this file locally. The data stored relates solely to user experience metrics and installation dates, with no sensitive authentication information present. The impact area is limited to local non-sensitive data storage.

3. Description
The application saves installation dates and user experience metrics in plaintext within the `pref_store.txt` file. This file does not contain any passwords, tokens, or unique user identifiers. The absence of encryption or access controls poses negligible risk due to the non-sensitive nature of the stored information. The root cause is the lack of data classification and secure storage practices for all configuration files, regardless of data sensitivity. No user input or network API endpoint is involved.

4. Evidence

makefile Copy code

```
# Sample content of pref_store.txt
installation_date2=2023-04-12T08:23:45Z
user_experience_metrics=low_entropy_source3

# No authentication or access control required to read the file
cat pref_store.txt
```

Figure 1. Example content of `pref_store.txt` displaying non-sensitive metrics
Figure 2. Terminal output showing successful local read access to the file

5. Recommendation

- Ensure critical or sensitive settings are always stored in a secure, encrypted location.

Рисунок 2.9 Генерація звіту за заданим промптом

Узагальнюючи роботу третього модуля, можна стверджувати, що автоматизація формування пентест-звіту стає ключовим елементом оптимізації всієї методології оцінювання безпеки. Модель успішно перетворює великий обсяг технічних даних на структуровані, узгоджені та якісно оформлені звітні елементи, усуваючи необхідність ручного оформлення після технічного аналізу. LLM не лише акумулює дані, а й правильно їх інтерпретує: визначає контекст, причини вразливостей, формує PoC, рекомендації та додає посилання на стандарти, перетворюючи звіт на логічно послідовний документ.

Стабільність стилю, узгоджена термінологія та передбачувана структура значно підвищують якість кінцевого звіту. Модель ефективно працює з різномірними артефактами, точно визначає рівень ризику та формує повноцінний інформаційний продукт. Таким чином, AI-модуль суттєво підсилює процес пентесту, автоматизуючи одну з найкритичніших фаз

створення звіту, забезпечуючи економію часу, підвищення точності та стандартизацію результатів.

2.3. Порівняльний аналіз ефективності пентесту з використанням та без використання AI

У цьому підрозділі порівнюється ефективність традиційного, повністю ручного пентесту та підходу, у якому ключові етапи автоматизовано трьома AI-модулями. Мета аналізу не лише визначити різницю у швидкості чи обсязі роботи, а комплексно оцінити вплив автоматизації на точність, повноту, повторюваність і структурованість результатів. Оскільки кожен модуль працює з різними типами даних і виконує окрему частину процесу, оцінювання проводиться індивідуально для кожного з них, що дозволяє точніше визначити сильні сторони AI та межі, де ключову роль зберігає експертний аналіз.

Для об'єктивного порівняння застосовується набір формалізованих метрик: час виконання аналізу, кількість знайдених проблем, точність їх класифікації, повторюваність результатів та ступінь зниження когнітивного навантаження. Часова метрика демонструє прямий ефект автоматизації на швидкість обробки великих масивів даних. Кількість виявлених проблем дозволяє оцінити вплив AI на повноту перевірки. Окрему роль відіграє якість класифікації адже саме вона визначає коректність рекомендацій і пріоритизації ризиків. Повторюваність важлива для стандартизації висновків, оскільки ручний підхід значною мірою залежить від індивідуального стилю та досвіду аудитора. Зменшення когнітивного навантаження показує, наскільки автоматизація усуває рутинні завдання та знижує ризик помилок, спричинених втомою.

Порівняння результатів ручного семантичного аналізу й аналізу, виконаного AI-модулем, демонструє значну перевагу автоматизованого підходу як у швидкості, так і в глибині опрацювання статичних артефактів мобільного застосунку. LLM здатна за секунди опрацювати `AndroidManifest.xml`,

конфігураційні файли та декомпільований код, у той час як ручний аналіз потребує десятків хвилин. Модель не лише швидко обробляє структуру дозволів, API-викликів і параметрів, а й точніше визначає надлишкові або небезпечні дозволи завдяки контекстному аналізу взаємозв'язків між компонентами.

Особливою перевагою AI стало виявлення логічних вразливостей, які традиційно є найскладнішими для ручного аналізу. Модель ідентифікує взаємопов'язані ланцюжки «дані → метод → мережевий виклик», що можуть спричинити витоки або неправильну обробку інформації. Ручний аналіз часто фокусується на очевидних порушеннях, пропускаючи такі приховані залежності через їх фрагментованість у коді. Наприклад, AI виявляє слабкі механізми оброблення даних у поєднанні з небезпечним API у подальшому виклику, що може створювати шлях для логічного обходу перевірок ситуацію, яку аналітик може пропустити під час ручного перегляду.

Таким чином, модуль семантичного аналізу довів, що автоматизація здатна не лише значно прискорити опрацювання статичних артефактів, але й підвищити точність і повноту виявлення логічних недоліків тих, що потребують найбільших когнітивних зусиль у традиційному підході. Це підкреслює технічну доцільність та практичну користь AI у початкових етапах пентесту.

Порівняння ефективності аналізу локального сховища вручну та за допомогою AI-модуля демонструє помітну перевагу автоматизації як у швидкості, так і в повноті виявлення проблем. Ручний перегляд SQLite-баз, SharedPreferences та службових файлів вимагає значних трудовитрат: аудитор повинен вилучити дані з пристрою, проаналізувати вміст кожного файлу й визначити, чи містить він чутливу інформацію. Це передбачає опрацювання десятків або сотень файлів, більшість з яких не становлять інтересу, але потребують перевірки. AI-модуль виконує ті ж завдання за хвилини, автоматично нормалізуючи структуру файлів, класифікуючи їх та миттєво

виділяючи потенційно небезпечні. Завдяки цьому пентестер фокусується лише на критичних артефактах.

AI також виявляє ширший спектр проблем, ніж ручний підхід. Модуль знаходить не тільки очевидні випадки зберігання чутливих даних у відкритому вигляді, а й менш очевидні службові ідентифікатори, сеансові маркери, конфігураційні параметри з ризиком або непрямі токени, які опосередковано впливають на безпеку. У тестовому середовищі AI виявив усі суттєві проблеми, зокрема незахищене зберігання JWT, тоді як ручний аналіз вимагав би набагато більше часу для повного перегляду всіх файлів.

Ще однією перевагою є зменшення кількості false positives. Під час ручного аналізу службові файли чи внутрішні структури SQLite можуть помилково розцінюватися як ризикові. AI-модуль точно визначає, що елементи на кшталт LevelDB-маніфестів, WebView-кешів чи lock-файлів не містять чутливих даних, зменшуючи потребу в повторних перевірках і роблячи процес аналізу більш ефективним.

Таким чином, AI-модуль для аналізу локального сховища забезпечує суттєве прискорення роботи, підвищує точність виявлення проблем і покращує структурованість результатів. У контексті мобільного пентесту, де файлові структури часто є складними й різномірними, це становить значну практичну цінність.

Найбільш вимірюваний ефект автоматизації простежується на етапі документування. Ручне створення звіту потребує значних зусиль: необхідно систематизувати артефакти, описати вразливості, підготувати PoC, визначити ризики та рекомендації. У середньому ця частина може займати третину або й половину всього часу пентесту. AI-модуль формує один структурований запис за секунди, а повний звіт за хвилини, зменшуючи часові витрати на порядок без втрати змістовної насиченості.

Окремою перевагою є якість структурування матеріалу. Ручні звіти часто відрізняються стилем, логікою викладу та рівнем деталізації. AI забезпечує повну уніфікацію: кожен елемент звіту має стало визначену структуру й чітко

розмежовані розділи (Background, Description, Evidence, Recommendation). Це підвищує читабельність та зручність використання як для технічних спеціалістів, так і для менеджменту.

Важливим є й дотримання стандартів, таких як OWASP MASVS чи PTES. У ручному режимі це залежить від підготовки аудитора, що може спричиняти нерівномірну якість документації. AI-модуль, навпаки, може бути чітко орієнтований на відповідність індустріальним вимогам, правильно класифікуючи вразливості та забезпечуючи повноту висвітлення необхідних елементів.

У підсумку модуль документування демонструє найбільший приріст ефективності серед усіх досліджених етапів. Він істотно скорочує час створення звіту, гарантує стабільну якість і повну відповідність сучасним стандартам безпеки, що робить його потужним інструментом для стандартизації та масштабування процесу пентесту в умовах великих команд і регулярних перевірок.

Узагальнюючи результати порівняльного аналізу, можна чітко визначити, які етапи пентесту отримують найбільше підсилення завдяки AI, а де вирішальна роль усе ще належить фахівцю. Найпомітніший ефект автоматизації спостерігається там, де потрібно обробляти великі обсяги структурованих даних: під час статичного аналізу конфігурацій та коду, класифікації артефактів локального сховища та формування фінальної документації. На цих етапах AI забезпечує значне прискорення роботи, підвищує точність, зменшує кількість помилок і гарантує повторюваність результатів. Модуль семантичного аналізу ефективно виявляє надлишкові дозволи, небезпечні API та логічні ланцюжки, тоді як модуль локального сховища точно і швидко ідентифікує чутливі дані. Найбільший приріст ефективності забезпечує модуль документування, який уніфікує структуру звіту, дотримується стандартів і суттєво скорочує час його підготовки.

Водночас окремі аспекти пентесту залишаються залежними від експертного ручного аналізу. Це передусім динамічне тестування, формування

гіпотез щодо складних сценаріїв атак, аналіз поведінки системи в нетипових умовах та прийняття рішень у ситуаціях, де значну роль відіграє контекст або інтуїція фахівця. Хоч AI може виявляти логічні невідповідності, він не здатен самостійно проводити експлуатаційні експерименти або достовірно моделювати поведінку складних систем за межами доступних даних. Те саме стосується вразливостей бізнес-логіки чи специфічних користувацьких сценаріїв вони нерідко вимагають саме людського досвіду.

Межі застосовності AI визначаються характером задачі та якістю вхідних артефактів. Модель працює найефективніше там, де доступні структуровані дані й правила аналізу можна формалізувати. Натомість задачі з високим рівнем неоднозначності або динамічності природно залишають за людиною роль ключового аналітика. Тому AI не замінює пентестера, а підсилює його, беручи на себе рутинні, повторювані та чітко формалізовані операції, тоді як критичні технічні оцінки та стратегічні рішення залишаються в зоні експертної компетентності.

Отже, ефективність AI у пентесті полягає у прискоренні роботи, підвищенні точності та стандартизації результатів, проте межі його застосування визначаються потребою в людському досвіді та аналітичному мисленні під час розв'язання складних і неоднозначних задач. Таке поєднання забезпечує оптимальний баланс між автоматизацією та експертністю.

Висновки до розділу 2

У підсумку проведеного дослідження роль штучного інтелекту в автоматизації мобільного пентесту проявилася як комплексне підсилення всіх його технічних етапів від первинного статичного аналізу до фінального формування звіту. AI дав змогу значною мірою зменшити обсяг ручної роботи, вилучити рутинні операції та зосередити увагу аудитора на тих аспектах, які потребують експертної інтерпретації. Розроблені три модулі продемонстрували різні, але взаємодоповнювальні можливості: семантичний аналіз коду й

конфігурацій забезпечив глибше й швидше виявлення небезпечних патернів, модуль аналізу локального сховища прискорив і структурував обробку гетерогенних даних, а модуль автоматичного документування стандартизував і значно пришвидшив підготовку звіту.

Результати тестування кожного модуля показали, що LLM здатні не лише повторювати окремі елементи ручного аналізу, а й покращувати їх за рахунок ширшого контекстного охоплення й здатності узагальнювати інформацію. Модуль семантичного аналізу виявив логічні вразливості й небезпечні API, які потребували б суттєвого часу для ідентифікації вручну. Модуль роботи з локальним сховищем визначив критичні проблеми зберігання даних і значно зменшив частку хибних спрацювань. Модуль документування, своєю чергою, показав найбільший приріст ефективності, забезпечивши майже повну автоматизацію створення структурованих записів про вразливості як показує аналіз професійних стандартів.

Порівняння автоматизованого та ручного підходів засвідчило, що AI забезпечує вимірюване скорочення часу, підвищення точності, повторюваність результатів і краще дотримання стандартів. Водночас аналіз підтвердив, що AI не усуває потреби в експертові, особливо коли йдеться про складні логічні вразливості, динамічну експлуатацію чи оцінку бізнесового контексту. Тому реальна цінність технології полягає у поєднанні автоматизації формалізованих етапів з експертним контролем критичних рішень.

Отримані висновки створюють основу для подальшого розділу, присвяченого рекомендаціям щодо впровадження інтелектуальних систем у процес пентесту. Вони дозволяють сформулювати чіткі критерії доцільності використання AI, визначити оптимальну архітектуру інтеграції та окреслити реалістичні межі застосовності таких технологій у практиці інформаційної безпеки.

РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ У ПРОЦЕС ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

У третьому розділі роботи розглядаються вже не окремі експерименти з використанням штучного інтелекту, а питання його практичного впровадження у процес тестування на проникнення. Якщо в попередньому розділі основна увага була зосереджена на перевірці працездатності окремих AI-модулів та порівнянні їх з ручним підходом, то далі важливо зробити крок від експерименту до рекомендацій. Метою цього розділу є визначити, за яких умов застосування AI у пентесті дійсно є доцільним і ефективним, які вимоги висуваються до організації процесу та технічної інфраструктури, а також як виглядає оптимальна архітектурна модель інтеграції таких інструментів у реальний робочий цикл безпеки.

Окремо розглядається питання того, як саме AI-інструменти, зокрема моделі класу LLM, можуть бути вбудовані у стандартні етапи пентесту: від збору і нормалізації артефактів до аналізу результатів і формування звіту. Важливо не просто показати, що це технічно можливо, а описати логіку взаємодії між модулями, точки інтеграції з існуючими інструментами та обмеження, які необхідно враховувати. У цьому контексті архітектурна модель використовується як спосіб узагальнити отримані в експерименті результати і показати, як вони можуть масштабуватися до реальних проєктів.

Третім важливим напрямом розділу є аналіз проблем і перспектив застосування штучного інтелекту в кібербезпеці. Попри очевидні переваги, використання LLM не позбавлене ризиків і обмежень, пов'язаних як із технічними аспектами (якість вхідних даних, "галюцинації" моделей), так і з питаннями конфіденційності та довіри до автоматичних висновків. Тому в розділі робиться спроба окреслити реалістичні межі застосовності таких систем: де AI дійсно підсилює пентестера, а де залишається лише допоміжним інструментом, який потребує жорсткого контролю з боку людини. Сукупність

цих трьох блоків умови, архітектура, проблеми та перспективи дозволяє перейти від окремих технічних експериментів до практичних рекомендацій щодо впровадження інтелектуальних систем у процес тестування на проникнення.

3.1. Умови та критерії доцільності впровадження AI в пентест

Організаційні передумови впровадження AI в пентест, на мою думку, пов'язані насамперед із тим, як побудований сам процес роботи команди безпеки. Якщо в організації більшість завдань під час тестування на проникнення мають повторюваний або рутинний характер, це створює сприятливий ґрунт для автоматизації. Йдеться, наприклад, про типові перевірки конфігурацій, аналіз схожих артефактів (маніфести, логи, бази даних), первинне структурування знайдених вразливостей, підготовку стандартних частин звітів. У реальних умовах значну частину часу пентестер витрачає не на пошук нових векторів атак, а на багаторазове виконання однакових кроків. Саме такі завдання є першими кандидатами для делегування AI-модулям.

Другий важливий момент це потреба у масштабуванні процесів пентесту. Якщо організація регулярно проводить перевірки великої кількості застосунків (наприклад, мобільних клієнтів для різних регіонів, декількох версій одного продукту чи внутрішніх сервісів), то ручний підхід починає "не встигати" за обсягом роботи. У такій ситуації застосування AI розглядається не як щось "експериментальне", а як практичний інструмент, який дозволяє обробляти більше об'єктів за той самий час без пропорційного збільшення числа спеціалістів. Це особливо актуально там, де є вимога до регулярного або безперервного тестування, наприклад, у контексті DevSecOps [16].

Третя організаційна передумова стосується вимог до стандартизації звітів і відтворюваності результатів. У багатьох компаніях звіти з пентесту є не лише технічним документом, а й формальним артефактом для аудиту, відповідності стандартам або взаємодії з регуляторами. У такому випадку важливо, щоб звіти були уніфіковані за структурою, стилем і глибиною опрацювання. Ручний підхід

завжди несе в собі ризик варіативності: різні спеціалісти можуть по-різному описувати схожі вразливості, по-різному деталізувати рекомендації або розставляти акценти. Використання AI-модулів дозволяє наблизитися до стандартизованого формату: одна й та сама логіка опису, чітке дотримання шаблонів, повторюваність структури та термінології. Це не лише підвищує якість документування, а й спрощує порівняння результатів між різними пентестами в часі.

Таким чином, можна сказати, що організаційні передумови для впровадження AI у пентест виникають тоді, коли: по-перше, є помітний пласт однотипних завдань; по-друге, з'являється потреба у масштабуванні без пропорційного збільшення ресурсів; по-третє, від команди безпеки вимагають стабільно структурованих і відтворюваних результатів. У таких умовах використання AI вже виглядає не як "опція на майбутнє", а як логічний крок розвитку процесу.

Технічні передумови впровадження AI в пентест, на пряму залежать від того, з якими саме даними ми працюємо і в якому вигляді вони доступні. Перш за все, потрібні артефакти, які придатні для більш-менш структурованого аналізу. У випадку мобільного пентесту це, як мінімум, APK-файли, AndroidManifest.xml, ресурси на кшталт strings.xml, журнальні файли роботи застосунку, бази даних SQLite, файли SharedPreferences та інші елементи локального сховища. Важливо, щоб ці артефакти або вже були текстовими (XML, JSON, логи), або могли бути перетворені у такий формат за допомогою стандартних інструментів (декомпіляція, дамп SQLite, конвертація XML у ключ–значення тощо). Якщо дані залишаються "закритими" у власному бінарному форматі й немає можливості їх коректно розпарсити, то LLM просто не матиме з чим працювати [17].

Другий момент це можливість збирати вхідні дані у більш-менш стандартизованому вигляді. Для того, щоб AI-модуль був корисним не один раз «для диплома», а в реальному процесі, потрібно продумати мінімальний pipeline: як саме з пристрою або з інфраструктури витягуються APK, як

знімаються дампи локального сховища (через adb, емулятор, тестове оточення), як саме логуються HTTP-запити, як результати SAST/DAST або попереднього аналізу зберігаються у форматі, який потім можна передати в LLM. Чим більш однаковим є цей формат між різними проєктами, тим легше побудувати повторюваний AI-аналіз. У протилежному випадку значна частина виграшу від автоматизації "з'їдається" одноразовою підготовкою даних під кожний окремий кейс.

Окремо потрібно враховувати вимоги до продуктивності та безпеки. З технічної точки зору AI-аналіз додає ще один шар обробки даних, тому важливо, щоб інструменти декомпіляції, парсингу логів та конвертації у JSON не створювали вузьких місць: інакше час, зекономлений на інтелектуальному аналізі, буде втрачений на попередню підготовку. З боку безпеки виникає питання, де саме виконується LLM-аналіз локально чи через хмарний API. Якщо йдеться про реальні мобільні застосунки з продакшен-даними, то передавання журналів, токенів, конфігурацій та інших чутливих артефактів у зовнішній сервіс може бути неприйнятним без додаткових заходів захисту (анонімізація, шифрування, ізольоване тестове середовище). Тому технічно коректне впровадження AI в пентест вимагає не тільки "моделі, яка все розуміє", а й продуманої інфраструктури для збирання, нормалізації й безпечної обробки даних [18].

Коли говоримо про доцільність застосування AI у пентесті, важливо не просто відмітити, що "штучний інтелект допомагає", а сформулювати конкретні критерії, за якими можна оцінити, чи справді його використання виправдане в конкретній команді й процесі. На мою думку, перший і найбільш очевидний критерій це скорочення часу виконання завдань без втрати якості. Тобто не достатньо того, що модуль працює швидко потрібно дивитися, чи результати, які він видає, не гірші за ручний аналіз. Якщо AI-модуль дозволяє, наприклад, замість кількох годин статичного аналізу чи перегляду локального сховища виконувати ту ж роботу за хвилини, при цьому виявляючи ті самі або навіть більше проблем, це вже суттєвий аргумент на користь його впровадження.

Другий важливий критерій це підвищення точності, а саме зменшення кількості false positives та false negatives. У пентесті це критично: якщо інструмент постійно "сипле" хибними спрацюваннями, аналітик витрачає час на перевірку неіснуючих проблем. Якщо ж навпаки реальні вразливості залишаються непоміченими, то така автоматизація більше шкодить, ніж допомагає. Тому AI має показувати відчутне покращення у якості класифікації знахідок: правильно відрізнити службові файли від чутливих даних, адекватно оцінювати ризики, не маркувати як "критичне" те, що по суті є інформаційною дрібницею.

Третій критерій збільшення повторюваності результатів. У ручному підході багато що залежить від конкретної людини: її досвіду, втоми, того, наскільки уважно вона вчиталася у логи або конфігурації. Як наслідок, два різні пентестери можуть дати різні за глибиною і деталізацією результати, навіть працюючи з однаковим застосунком. AI в цьому плані працює більш стабільно: якщо йому подати однакові вхідні дані й не змінювати промпти, результати будуть значно більш послідовними. Це особливо важливо для організацій, де потрібно не просто "знайти баг", а підтримувати певний стандарт якості й формату звітів.

У підсумку можна сказати, що доцільність застосування AI в пентесті має через набір конкретних критеріїв: чи реально скорочується час, чи росте точність, чи стають результати стабільнішими, чи стає легше працювати самому аналітику і чи виправдовує все це витрати на впровадження. Якщо відповідь на більшість із цих питань "так", тоді AI має сенс як невід'ємна частина процесу тестування на проникнення.

3.2. Архітектурна модель інтеграції AI-інструментів у процес пентесту

Якщо дивитися на процес пентесту не як на набір розрізнених кроків, а як на цілісний цикл, то загальна структура інтеграції AI-інструментів логічно

вибудовується навколо трьох модулів, які я описувала в попередньому розділі. Перший модуль відповідає за семантичний аналіз коду та конфігурацій застосунку, другий за аналіз локального сховища, а третій за автоматичне формування звіту. Разом вони утворюють ланцюжок: спочатку збір і осмислення технічних артефактів, потім поглиблений аналіз зберігання даних на пристрої, а вже потім перетворення всього цього в структурований звіт. Кожен із модулів закриває свій етап пентесту й одночасно готує дані для наступного.

Роль першого AI-модуля у цьому циклі дати максимально змістовну “картину з висоти” щодо самого застосунку: його дозволів, експортованих компонентів, використаних API та потенційних логічних вразливостей. На вхід він отримує артефакти статичного аналізу декомпільований код, AndroidManifest, ресурси і повертає уже не просто "сирі" списки, а напівготові висновки: які дозволи виглядають надлишковими, які компоненти небезпечно відкриті назовні, де можуть бути слабкі місця в логіці. Ці висновки далі використовуються як контекст для аналізу локального сховища: наприклад, якщо відомо, що застосунок працює з токенами або фінансовими даними, то в локальному сховищі їх наявність чи відсутність уже сприймається по-іншому.

Другий AI-модуль підключається після того, як із пристрою зібрані всі файлові артефакти (через adb pull тощо) і приведені до нормалізованого вигляду. Його задача системно пройтися по SQLite-базах, SharedPreferences, логах, технічних файлах і виявити саме ті місця, де відбувається зберігання чутливих даних у plaintext або з некоректним захистом. Фактично, він працює як “фільтр” над великим масивом файлів, виділяючи критичне і позначаючи те, що не несе ризику. Важливо, що результати цього модуля не живуть окремо: знайдені JWT-токени, ETag-и, конфігураційні URL, підозрілі журнали та інші артефакти потім стають частиною вхідних даних для модуля документування. Тобто другий модуль формує "сировину" не тільки для аналітика, а й для автоматичного формування звіту.

Третій модуль завершує цикл. Він отримує на вхід уже не хаотичні файли, а структурований набір артефактів: висновки семантичного аналізу (модуль 1), результати перевірки локального сховища (модуль 2), списки endpoint-ів, PoC, журнальні фрагменти та інші технічні дані. На основі цього він формує звітні блоки: опис вразливостей, їхній контекст, докази, оцінку ризиків і рекомендації. Тобто його роль “перекласти” технічну інформацію, яку зібрали й проаналізували попередні модулі, у формат, який можна відразу віддати замовнику пентесту. Виходить, що третій модуль залежить від перших двох, а перші два максимально розвантажують його, даючи вже підготовлені для опису об’єкти.

Щоб цей ланцюжок працював не як разова інтеграція "на колінці", потрібна централізована система збору артефактів. По суті, це проміжний шар між реальним середовищем (емулятором чи пристроєм) та AI-модулями. Він відповідає за те, щоб: зберегти APK і результати декомпіляції; зібрати й зафіксувати дамп локального сховища; упорядкувати логи; звести все це в єдиний набір структурованих даних (найчастіше у вигляді JSON). На практиці це може бути окрема папка проекту або навіть невеликий внутрішній інструмент, який за результатами пентесту "пакує" артефакти у стандартизований вигляд. Саме завдяки такій централізації AI-модулі бачать не уривки інформації, а повноцінний контекст, і їхні результати можна повторювати та масштабувати на інші застосунки.

У підсумку загальна структура інтеграції виглядає так: один раз налаштований збір артефактів → статичний AI-аналіз → AI-аналіз локального зберігання → AI-документування. Кожен модуль тут має свою чітку роль, а "клеєм" між ними виступає єдина система збирання й нормалізації даних. Це дозволяє не просто "додати AI до пентесту", а вбудувати його в процес так, щоб він реально економив час і додавав цінності, а не створював додатковий хаос.

У контексті інтеграції AI-модулів у процес пентесту одним із ключових аспектів є організація коректного збору даних та їх подальшої обробки. Саме від того, наскільки добре налагоджені канали передачі артефактів і процес їх

нормалізації, залежить якість роботи моделей та відтворюваність результатів. На практиці це означає, що весь потік технічної інформації від APK-файлів і декомпільованого коду до журналів роботи застосунку та дамів локального сховища має бути не просто зібраний, а структурований таким чином, щоб AI міг ефективно працювати з цими даними.

Першим елементом цієї системи є канали передачі артефактів. У мобільному пентесті основними джерелами даних виступають SAST-вивантаження (наприклад, результати декомпіляції кодової бази через JADX або Androguard), артефакти, отримані з пристрою командою adb pull, журнальні файли (логування HTTP-викликів, краш-логи, внутрішні логи застосунку) та ресурсні дані, такі як XML-файли або конфігурації. Важливо, щоб ці канали були стабільними й відтворюваними: тобто щоб дані завжди надходили в одному й тому ж форматі та з дотриманням однакової структури. Тоді AI-модуль може покладатися на передбачуваність вхідних даних, а аналітик отримує можливість масштабувати аналіз на інші версії або застосунки без додаткової підготовки.

Другим важливим компонентом є перетворення та нормалізація даних. Суть цього етапу полягає в тому, щоб привести всі отримані артефакти до єдиної структури, з якою зручно працювати моделі. Наприклад, SQLite-бази зазвичай необхідно конвертувати у JSON, SharedPreferences у текстові структури ключ–значення, журнальні файли розбити на окремі події або сегменти, а результати декомпіляції виділити у вигляді окремих модулів і класів. Така нормалізація виконує дві функції: по-перше, вона забезпечує стандартизований формат для AI-аналізу, а по-друге запобігає помилкам, пов'язаним із некоректним або неповним розбором даних. З мого досвіду, саме цей етап має найбільше значення, оскільки від нього залежить, чи модель зможе "побачити" логіку артефактів і правильно їх інтерпретувати.

Третій аспект це логування, трасування та забезпечення відтворюваності результатів. У пентестах важливо, щоб будь-який аналітичний висновок можна було перевірити, повторити або відтворити в майбутньому. Тому система збору

та обробки даних повинна фіксувати, які саме артефакти були передані в AI-модуль, якою була версія моделі, які промпти використовувалися та які результати вона повернула. Логування допомагає розуміти, чому AI зробив той чи інший висновок, а трасування перевіряти правильність даних у складних випадках, особливо коли йдеться про логічні вразливості. Відтворюваність важлива і з точки зору навчання: вона дозволяє порівнювати результати між різними запусками, оцінювати стабільність моделі та адаптувати промпти, якщо це необхідно.

Таким чином, дані та процесинг у контексті AI-підсиленого пентесту це не просто технічна деталь, а фундамент, на якому тримається весь аналіз. Без чіткої організації потоків артефактів, правильної нормалізації та якісного логування навіть найкраща модель буде працювати непослідовно. Натомість добре поставлений процес забезпечує високу якість, повторюваність і практичну цінність усіх AI-модулів, які інтегруються в реальний пентест.

Окремим важливим питанням при впровадженні AI в пентест є взаємодія із зовнішніми LLM-моделями, тобто тими, що викликаються через API або розгортаються локально в інфраструктурі організації. З точки зору безпеки тут першочергово постає вимога до ізоляції тестових даних: усе, що стосується конфігурацій продакшен-систем, токенів, внутрішніх URL, журналів із реальними користувацькими даними, не повинно “витікати” за межі контрольованого середовища. Якщо використовується хмарний API, потрібно чітко розуміти політику зберігання та обробки даних постачальником моделі, а також, за можливості, мінімізувати обсяг чутливої інформації, що передається назовні (наприклад, анонімізувати значення токенів, замінити реальні домени на псевдо, не надсилати повні логи з персональними даними). Додатково обов’язковим є використання шифрування на транспортному рівні (TLS) та, за потреби, власних механізмів шифрування або токенизації чутливих фрагментів перед відправленням до моделі. У випадку локально розгорнутих LLM ці ризики частково зменшуються, але натомість з’являються вимоги до

захищеності внутрішньої інфраструктури й контролю доступу до самого AI-сервісу.

Коли йдеться про вибір конкретної LLM, варто орієнтуватися не лише на «модність» моделі, а на практичні критерії: якість роботи з технічним текстом, стабільність дотримання структурованих форматів, вартість викликів та можливість інтеграції в існуючий інструментарій. Наприклад, моделі на рівні GPT-4.1 доцільно використовувати там, де критично важливі чітка структура звітів, правильне трактування коду, маніфестів і конфігурацій, а також здатність будувати логічні зв'язки між різними артефактами. Моделі типу GPT-o1 можуть бути корисними на етапах, де потрібен глибший “reasoning”, але, можливо, не настільки жорстке дотримання форматів. Локальні моделі мають сенс тоді, коли організація не готова передавати за межі периметра навіть частково знеособлені дані або коли є вимога до повного контролю над середовищем виконання. Водночас їх якість і зручність часто поступаються промисловим хмарним LLM, тому це завжди компроміс між безпекою, точністю та витратами на підтримку [19].

Окремо варто згадати про формування фінального звіту у стандартизованому форматі. На практиці часто вимагається, щоб результати пентесту були надані у вигляді PDF-документа для менеджменту, Markdown/HTML для внутрішніх wiki або JSON для подальшої автоматизованої обробки. AI-модуль може з одного й того ж джерела структурованих даних зібрати кілька варіантів представлення: більш технічний звіт для розробників (із деталізованими PoC і логами), більш узагальнений для керівництва, а також машинно-читаний варіант для інтеграції з іншими системами. Завдяки цьому результати пентесту перестають бути “разовим” документом і перетворюються на багатоформатний артефакт, який можна повторно використовувати, аналізувати в динаміці та підключати до інших процесів без необхідності щоразу все переписувати вручну.

3.3. Проблеми та перспективи застосування штучного інтелекту в кібербезпеці

Коли говоримо про застосування AI в кібербезпеці, важливо чесно визнати, що технологія має свої обмеження, і їх неможна ігнорувати, особливо якщо планується реальне впровадження в процес пентесту. Одне з перших і, на мою думку, найважливіших обмежень це сильна залежність якості результатів від коректності вхідних даних. LLM не "бачить" систему напряду, вона працює тільки з тим, що їй передали: фрагменти маніфестів, дампи локального сховища, логи, JSON-структури тощо. Якщо на етапі збору або нормалізації артефактів були допущені помилки щось не витягнули, неправильно розпарсили, загубили контекст, модель просто не має можливості це "надолжити". У результаті її висновки можуть виглядати логічно і навіть переконливо, але фактично ґрунтуватися на неповній картині. Це створює ризик хибного відчуття впевненості: звіт виглядає професійно, однак частина вразливостей так і залишилася непоміченою через брак даних.

Друге суттєве обмеження пов'язане з так званими "галюцинаціями" моделей. LLM іноді генерує інформацію, яка виглядає правдоподібно, але не має фактичного підґрунтя у вхідних даних. У контексті пентесту це особливо небезпечно: модель може "вигадати" зайві деталі в описі вразливості, некоректно інтерпретувати лог, неправильно реконструювати структуру запиту чи відповіді. Якщо аналітик беззастережно довіряє AI, є ризик, що у звіт потраплять неточні або навіть хибні висновки. Тому на практиці потрібен постійний критичний перегляд результатів: AI-генерацію потрібно сприймати як чернетку, а не як остаточну істину. Факти, PoC і технічні деталі все одно мають перевірятися людиною.

Ще одне обмеження стосується недостатньої здатності AI до динамічної експлуатації вразливостей. На теперішньому етапі моделі добре справляються з аналізом статичних артефактів і логіки, але вони не вміють самостійно "гратися" з реальною системою: змінювати параметри запитів, тримати стани

сесій, коригувати стратегію залежно від поведінки цілі в режимі реального часу. Навіть якщо AI може допомогти придумати PoC, саме виконання експлойтів, ланцюжки атак, обхід захисту, робота з нестандартними умовами усе це досі значною мірою залишається на стороні пентестера. Тобто модель непогано підсилює аналітичну частину, але не замінює живий експеримент і "живий" підхід до атаки [20].

Окремий пласт обмежень стосується юридичних та етичних питань. При використанні хмарних LLM виникає ризик витоку конфіденційної інформації: журнали, токени, внутрішні URL, частини коду все це може за замовчуванням передаватися сторонньому постачальнику. Це потрібно узгоджувати з політиками компанії, NDA, вимогами регуляторів, а іноді й із законодавством конкретної юрисдикції. Крім того, постає питання відповідальності: хто винен, якщо AI помилився, а на основі його звіту було прийняте неправильне рішення? Також не варто забувати про етичний аспект: використання потужних інструментів аналізу може бути як на користь захисту, так і на користь зловмисників, якщо потрапить у "не ті руки". Усе це означає, що впровадження AI в пентест це не тільки технічне, а й правове та етичне завдання, яке потребує продуманих обмежень, контролю й прозорих правил використання.

У сукупності ці чинники показують, що AI у кібербезпеці не можна сприймати як універсальне рішення. Технологія дає дуже сильний інструмент для аналізу та автоматизації, але працює ефективно тільки за умов якісних вхідних даних, критичного ставлення до результатів та відповідального ставлення до питань безпеки й конфіденційності. Без цього її обмеження можуть виявитися не менш значущими, ніж її переваги.

У процесі впровадження LLM у пентест одним із найважливіших аспектів, який неможливо ігнорувати, є безпекові виклики. Хоча моделі штучного інтелекту значно спрощують аналіз і пришвидшують роботу аналітика, їх використання створює додатковий шар ризиків, які не завжди очевидні на перший погляд. Перш за все, це ризик витоку даних під час використання хмарних моделей. Якщо модель працює у хмарі, будь-які передані

артефакти журнали, конфігурації, сесійні токени, URL внутрішніх API, фрагменти коду фактично покидають межі інфраструктури замовника пентесту. Навіть якщо постачальник моделі запевняє, що дані не зберігаються, не використовуються для тренування, проходять транзитом і захищені TLS, проблема довіри все одно залишається. У багатьох випадках такі дані підпадають під NDA або містять інформацію, яка прямо заборонена до передачі за межі організації. Для компаній у сфері фінансів або охорони здоров'я це взагалі може бути неприпустимим, і тоді використання хмарних LLM автоматично стає юридично або політично проблемним.

З цим пов'язана друга велика група викликів потреба у контролі конфіденційності тестових артефактів. Пентест за своєю природою працює з нерідко найбільш чутливими даними: дампами локальних сховищ, ключами, токенами, помилками сервера, збереженими паролями, гав-логами з РІІ, історіями авторизацій, даними платіжних сесій тощо. Навіть гіпотетичне потрапляння такої інформації в сторонні руки створює величезні ризики. Тому перед передаванням артефактів до LLM необхідно або повністю їх анонімізувати, або видалити критичні фрагменти, або використати локальну модель у межах контрольованої інфраструктури. Проблема в тому, що тотальна анонімізація часто знижує цінність аналізу, тому що модель втрачає контекст, а відтак і здатність зрозуміти залежності або знайти логічні помилки. Виходить компроміс між безпекою і точністю: чим ретельніше ми приховуємо дані, тим менше модель може зробити.

Окремим ризиком є можливість обходу моделей або маніпулювання їх відповідями. На відміну від класичних інструментів статичного чи динамічного аналізу, LLM працює за принципом генеративних моделей і має вразливість до так званих *prompt injection* атак. Теоретично злоумисник, який має можливість впливати на вхідні дані (наприклад, через інтегровані логи або API-відповіді), може навмисно вставити фрагменти тексту, які змінять поведінку моделі. Це може призвести до того, що AI проігнорує певні частини аналізу, неправильно класифікує вразливість, «замовчить» підтверджений ризик або згенерує хибні

рекомендації. У пентесті, де точність має критичне значення, навіть невелика маніпуляція може звести нанівець користь від моделі.

У підсумку можна сказати, що безпекові виклики, пов'язані з використанням LLM, є достатньо серйозними й не можуть бути вирішені тільки технічними засобами. Вони вимагають комплексної політики: чітких правил передачі даних, контролю артефактів, використання локальних моделей там, де це необхідно, а також механізмів моніторингу й виявлення нетипової поведінки моделей. Тільки враховуючи ці аспекти, впровадження AI в пентест може бути не лише ефективним, а й безпечним [21].

Перспективи розвитку AI в пентесті, як на мене, пов'язані не стільки з «ще більшим автоматичним пошуком вразливостей», скільки з поглибленням здібностей моделей до міркування та їх вбудовуванням у реальні інженерні процеси. Уже зараз видно, що простого генеративного підходу недостатньо: набагато ціннішими стають саме reasoning-здатності, тобто вміння моделі послідовно аналізувати ланцюжок подій, співставляти артефакти з різних джерел та робити обґрунтовані висновки. Для пентесту це означає рух від «підказок по коду» до побудови повноцінних сценаріїв атак: модель не просто бачить слабе місце у конфігурації або в коді, а здатна пояснити, як саме його можна використати в контексті всієї системи, які кроки треба виконати далі й які обмеження можуть завадити експлуатації.

Логічним продовженням цього є поява автономніших агентів, які можуть виконувати не лише аналітичну, а й операційну частину пентесту у напівавтоматичному режимі. Йдеться не про повну заміну пентестера, а про інструменти, які отримують завдання («оцінити безпеку мобільного застосунку»), самостійно збирають артефакти, запускають базові тести, генерують PoC для типових вразливостей і повертають фахівцю вже підготовлений набір знахідок. Такий агент може комбінувати декілька інструментів: емулятор, проксі на кшталт Burp Suite, SAST-аналізатор, сканер локального сховища і координувати їх роботу, зменшуючи кількість ручних дій.

Роль людини у такому сценарії зміщується в бік перевірки складних кейсів, роботи з бізнес-логікою та прийняття рішень, які AI поки що зробити не може.

Ще один перспективний напрям інтеграція AI безпосередньо у CI/CD-процеси для безперервної оцінки безпеки. Зараз безпека часто перевіряється «хвилями»: окремий проєкт, окремий пентест, окремий звіт. З появою AI-інструментів стає реалістичним сценарій, коли кожен новий білд мобільного застосунку автоматично проходить через набір перевірок: аналіз маніфестів і дозволів, швидкий перегляд змін у кодї з точки зору безпеки, аналіз оновленого локального сховища, а також базове моделювання загроз. Результати таких перевірок можуть потрапляти прямо у pipeline як «security quality gate»: якщо AI виявив критичні проблеми, білд не допускається до релізу, поки їх не буде усунуто. Це поступово наближає безпеку до стану, коли вона стає не епізодичною активністю, а частиною звичайної інженерної рутини.

Окремо варто згадати про розвиток спеціалізованих безпекових LLM, орієнтованих конкретно на задачі SAST, DAST та відповідність таким стандартам, як OWASP MASVS. На відміну від універсальних моделей, які «вміють усе потроху», такі системи можуть бути навчені на вузькому корпусі: кодових базах, технічній документації, звітах з пентестів, масивах типової вразливості та їх експлуатацій. Це дає шанс отримати інструмент, який не просто «розуміє текст», а добре орієнтується в патернах небезпечного коду, типових анти-патернах конфігурацій, структурах мобільних маніфестів і вимогах MASVS до сховища, автентифікації чи криптографії. У перспективі це можуть бути окремі «безпекові асистенти», які інтегруються прямо в IDE, мобільні тестові стенди чи платформи автоматизованого аналізу і працюють саме в контексті кібербезпеки, а не загального програмування.

У цілому, якщо підсумувати, розвиток AI у пентесті, скоріш за все, піде в бік глибшого розуміння контексту, більшої автономності на технічному рівні та тіснішої інтеграції з життєвим циклом розробки. Але при цьому роль людини навряд чи зникне: навпаки, фокус зміститься з рутинних задач на ті ділянки, де потрібні критичне мислення, досвід і розуміння бізнес-аспектів безпеки. AI тут

виглядає не як заміна пентестера, а як інструмент, який дозволяє перенести акценти з «ручної обробки даних» на по-справжньому складні й цікаві задачі.

Висновки до розділу 3

Узагальнюючи результати третього розділу, можна зробити висновок, що впровадження AI у процес тестування на проникнення є доцільним за умови правильного вибору сценаріїв застосування та належної організації роботи з даними. Штучний інтелект не вирішує всі проблеми пентесту автоматично, однак він суттєво підсилює ті етапи, де найбільше накопичується рутини: аналіз конфігурацій, обробка локального сховища, структурування результатів і підготовка звітів. Завдяки цьому аналітик може зосередитися на частинах пентесту, які потребують креативності, технічної інтуїції та глибокого розуміння бізнес-логіки.

Ключовими перевагами застосування AI є підвищення швидкості обробки даних, зменшення кількості помилок, покращення повторюваності результатів та можливість стандартизації як аналізу, так і звітності. Особливо помітним є вплив на якість структурування інформації: моделі можуть узагальнювати артефакти з різних джерел та формувати зв'язні висновки, які у ручному режимі потребували б значно більше часу. Водночас важливо пам'ятати про обмеження залежність від якості вхідних даних, ризики конфіденційності при використанні хмарних моделей, потенційні "галюцинації" LLM та неможливість повноцінної динамічної експлуатації вразливостей без участі людини. Ці аспекти роблять очевидним те, що AI не може повністю замінити експерта та повинен функціонувати як інструмент підтримки, а не як автономний засіб ухвалення рішень.

З огляду на проведені дослідження, стратегічними рекомендаціями для подальшого розвитку систем автоматизації є стандартизація pipeline збору артефактів, перехід до гібридних архітектур (поєднання локальних та хмарних моделей), впровадження механізмів перевірки достовірності AI-висновків, а

також розширення використання reasoning-орієнтованих моделей для аналізу складних логічних залежностей. У перспективі AI може бути інтегрований у CI/CD-процеси як додатковий рівень контролю безпеки, що зробить перевірку мобільних застосунків більш регулярною та менше залежною від людського ресурсу.

У підсумку, роль AI у пентесті варто розглядати саме як підсилення експертного аналізу. Модель ефективно закриває повторювані, технічно одноманітні задачі, але остаточна оцінка, інтерпретація ризиків та прийняття рішень залишаються за пентестером. Така взаємодія дає найбільшу користь: автоматизація пришвидшує й структурує процес, а фахівець забезпечує коректність, точність і глибину аналізу.

ВИСНОВКИ

У виконаній кваліфікаційній роботі проведено комплексне дослідження теоретичних та практичних аспектів застосування штучного інтелекту для автоматизації ключових етапів тестування на проникнення мобільних застосунків. На основі аналізу сучасних методик пентесту встановлено, що традиційний підхід, хоча й залишається ефективним, значною мірою залежить від ручної праці фахівця, що збільшує тривалість аналізу та підвищує ризик суб'єктивних помилок. Поглиблений огляд технологій штучного інтелекту засвідчив, що великі мовні моделі здатні виконувати не лише класифікаційні чи аналітичні задачі, але й забезпечувати семантичне узагальнення даних, виявляти логічні залежності в структурах коду та генерувати стандартизовані звіти, що робить їх перспективним інструментом у сфері кібербезпеки.

У межах роботи побудовано теоретичні засади автоматизації інформаційної безпеки засобами ШІ, що дозволило визначити ті елементи пентесту, які найбільше виграють від автоматизації. Це передусім аналіз конфігурацій мобільних застосунків, обробка даних локального сховища, семантичне групування отриманих артефактів та формування фінальної звітності. На основі цього обґрунтовано доцільність розробки трьох автономних AI-модулів, здатних взяти на себе частину рутинних і трудомістких операцій пентестера.

Практична частина роботи включала реалізацію зазначених модулів і їх експериментальну перевірку на тестовому мобільному застосунку. Модуль семантичного аналізу коду продемонстрував здатність швидко визначати надлишкові дозволи, небезпечні API-виклики та потенційні логічні вразливості, що забезпечує прискорення статичного аналізу та зменшує кількість пропущених проблем. Модуль аналізу локального сховища значно оптимізував обробку SQLite-баз, XML-конфігурацій і допоміжних файлів застосунку, автоматично класифікувавши артефакти та виявивши випадки зберігання чутливих даних у plaintext. Модуль автоматизованого формування звітності дав

змогу отримувати уніфіковані, чітко структуровані елементи звіту, що відповідають вимогам PTES та OWASP MASVS і зменшують навантаження на пентестера під час підготовки підсумкової документації.

Проведений порівняльний аналіз показав, що використання AI-модулів забезпечує суттєве скорочення часу аналізу, підвищує точність класифікації вразливостей, покращує повторюваність результатів і зменшує когнітивне навантаження фахівця. Водночас встановлено, що штучний інтелект не може повністю замінити експерта, особливо на етапах, де потрібна глибока доменна обізнаність, критичне мислення та інтерпретація складних контекстів.

У роботі також визначено умови доцільності впровадження ШІ у процес пентесту, зокрема наявність великих обсягів артефактів, необхідність швидкої підготовки звітів, потреба у масштабованості та повторюваності аналізу. Запропонована архітектурна модель інтеграції AI-інструментів демонструє, як такі модулі можуть гармонійно доповнювати традиційні методики, не порушуючи існуючих процесів безпеки. Окреслено перспективи подальшого розвитку напрямку, серед яких створення агентних систем для комплексного проведення пентесту, поглиблена автоматизація динамічного аналізу та використання спеціалізованих моделей, орієнтованих на безпеку мобільних застосунків.

Загалом результати дослідження підтверджують, що інтеграція штучного інтелекту в процес тестування на проникнення є ефективним та практично значущим шляхом підвищення якості, швидкодії та відтворюваності пентесту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Penetration Testing Execution Standard (PTES). Penetration Testing Execution Standard Project, 2010. URL: <https://www.pentest-standard.org> (дата звернення: 12.02.2025).
2. NIST SP 800-115. Technical Guide to Information Security Testing and Assessment. National Institute of Standards and Technology. Gaithersburg, 2008. 81 р.
3. OSSTMM 3.0. Open Source Security Testing Methodology Manual. ISECOM, 2010. 378 р.
4. OWASP. Mobile Security Testing Guide (MSTG). Open Web Application Security Project, 2023. URL: <https://owasp.org/www-project-mobile-security-testing-guide/> (дата звернення: 12.02.2025).
5. OWASP. Mobile Application Security Verification Standard (MASVS) v2.0. OWASP Foundation, 2023. URL: <https://owasp.org/www-project-mobile-appsec-verification-standard/> (дата звернення: 12.02.2025).
6. Anderson, R. Security Engineering. 3rd ed. Wiley, 2020. 1232 р.
7. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016. 800 р.
8. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2020. 1136 р.
9. OpenAI. GPT-4 Technical Report. 2023. arXiv:2303.08774. URL: <https://arxiv.org/abs/2303.08774> (дата звернення: 12.02.2025).
10. Microsoft. AI Security Risk Assessment Framework. Microsoft Security, 2023. URL: <https://learn.microsoft.com/security/> (дата звернення: 12.02.2025).
11. MITRE ATLAS. Adversarial Threat Landscape for Artificial-Intelligence Systems. MITRE Corporation, 2023. URL: <https://atlas.mitre.org> (дата звернення: 12.02.2025).

12. MITRE ATT&CK. Knowledge Base of Adversary Tactics and Techniques. MITRE, 2024. URL: <https://attack.mitre.org> (дата звернення: 12.02.2025).
13. IBM Security. Automating Cybersecurity with Artificial Intelligence and Machine Learning. IBM Corporation, 2021. 42 p.
14. Bishop, M. Computer Security: Art and Science. 2nd ed. Addison-Wesley, 2018. 1296 p.
15. Sharma, S. Practical Artificial Intelligence for Cybersecurity. O'Reilly Media, 2022. 312 p.
16. OWASP. Mobile Security Testing Guide (MSTG). Open Web Application Security Project, 2023. URL: <https://owasp.org/www-project-mobile-security-testing-guide/> (дата звернення: 13.02.2025).
17. OWASP. Mobile Application Security Verification Standard (MASVS) v2.0. OWASP Foundation, 2023. URL: <https://owasp.org/www-project-mobile-appsec-verification-standard/> (дата звернення: 13.02.2025).
18. Androguard Documentation. Androguard Project, 2023. URL: <https://androguard.readthedocs.io/> (дата звернення: 13.02.2025).
19. JADX – Dex to Java Decompiler. JADX Project, 2024. URL: <https://github.com/skylot/jadx> (дата звернення: 13.02.2025).
20. Frida. Dynamic Instrumentation Toolkit. Frida Project, 2024. URL: <https://frida.re> (дата звернення: 13.02.2025).
21. Burp Suite Documentation. PortSwigger Ltd., 2024. URL: <https://portswigger.net/burp> (дата звернення: 13.02.2025).

У цьому додатку наведено перелік інструкцій, які застосовувалися під час реалізації та експериментальної перевірки AI-модуля №1, призначеного для семантичного аналізу конфігурацій та статичних артефактів мобільного застосунку. Промпти були розроблені таким чином, щоб забезпечити максимальну структурованість та відтворюваність результатів аналізу. Кожен окремий промпт відповідав за певний етап семантичної обробки.

1. Ти експерт з мобільної безпеки, спеціалізуєшся на статичному аналізі Android-застосунків та стандартах OWASP MASVS. Завдання: аналізувати конфігураційні файли (AndroidManifest.xml), рядкові ресурси, метадані APK і дані про підпис, визначати їхні взаємозв'язки та формувати висновки щодо безпеки застосунку. Важливо: – працюй строго на основі наданих фрагментів, без домислів; – виявляй надлишкові permissions; – ідентифікуй небезпечні exported компоненти; – класифікуй hardcoded secrets; – аналізуй відповідність Android Security Best Practices; – пояснюй логіку ризику. Ти відповідатимеш на серію промπτів, кожен з яких аналізує окремий тип артефактів. Пізніше ти маєш сформувати інтегрований семантичний висновок.

2. Проаналізуй наведені APK metadata (apkid output) і поверни структуровані висновки у форматі JSON . Визнач: Чи відповідає minSdk рівню безпеки сучасних Android-платформ. Чи створює targetSdk проблеми з безпековими механізмами (WebView, storage, permissions). Чи може sharedUserId або інші властивості створювати ризик. Які загальні висновки щодо безпеки можна зробити з цих метаданих. Формат відповіді: "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }

а. Приклад виводу: { "classification": "APK Metadata Security Analysis", "security_issue": "minSdkVersion і targetSdkVersion не вказані у наведеному метаданих (apkid output), що не дозволяє прямо оцінити відповідність сучасним стандартам безпеки. Проте відсутність цієї інформації

або її занадто низьке значення є типовим індикатором низького рівня безпеки. SharedUserId відсутній, що знижує ризики міжзастосункової ескалації привілеїв.", "severity": "medium", "evidence": "Наявна інформація: packageName: com.app.damnvulnerablebank, versionCode: 1, versionName: 1.0. Вивід apkid не містить полів minSdkVersion, targetSdkVersion, sharedUserId.", "recommendation": "Перевірити AndroidManifest.xml на предмет явної вказівки minSdkVersion (рекомендовано ≥ 24 для мінімального захисту), targetSdkVersion (рекомендовано найновішу стабільну версію), а також відсутність sharedUserId без критичної необхідності. У випадку їх відсутності або занижених значень оновити політики безпеки згідно з актуальними Android Security Best Practices."}

3. Проаналізуй фрагмент AndroidManifest.xml і поверни структуровані висновки у форматі JSON. Мета: визначити проблеми безпеки, пов'язані з конфігурацією додатку. Проаналізуй: – надлишкові або небезпечні permissions; – exported Activity, Service, Receiver без захисних атрибутів; – неправильні intent-filters; – відсутність permission-guard там, де він потрібен; – ризики, пов'язані з мережевими або WebView компонентами; – потенційні конфлікти між дозволами та логікою програми. Формат відповіді: "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }

4. Це рядкові ресурси, витягнуті з resources.arsc (strings.xml) і поверни структуровані висновки у форматі JSON. Проаналізуй: – чи містяться тут hardcoded secrets або токени; – чи видно debug або test конфігурації; – чи вказують URL або шляхи на staging/debug середовища; – які з цих значень створюють ризики. Формат відповіді: { "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }

5. Це рядкові ресурси, витягнуті з resources.arsc (strings.xml) і поверни структуровані висновки у форматі JSON. Проаналізуй: – чи містяться тут hardcoded secrets або токени; – чи видно debug або test конфігурації; – чи вказують URL або шляхи на staging/debug середовища; – які з цих значень

створюють ризики. Формат відповіді: { "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }

6. Проаналізуй наданий сертифікат APK і параметри підпису і поверни структуровані висновки у форматі JSON. Визнач: – чи є алгоритм підпису застарілим або вразливим; – чи може це вплинути на цілісність застосунку; – чи відповідає схема підпису сучасним вимогам Android App Signing. Формат відповіді: { "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" }

7. На основі всіх попередніх відповідей виконай семантичний аналіз взаємодії компонентів застосунку. Сформууй логічні ланцюги типу: конфігурація → дозволи → компоненти → ресурси → можливість атаки. Визнач: – комплексні вразливості, що виникають через поєднання кількох факторів; – можливі сценарії експлуатації; – технічні та бізнес-наслідки. Формат відповіді: – Vulnerability Name – Root Cause – Full attack chain – Affected components – Risk level – Recommended mitigation

У цьому додатку наведено промпт, що застосовувалися під час реалізації та експериментальної перевірки AI-модуля №2, орієнтованого на аналіз локального сховища мобільного застосунку та виявлення можливих ризиків, пов'язаних із зберіганням даних.

- Ти спеціалізований модуль аналізу безпеки локального сховища мобільного застосунку. Проаналізуй вміст кожного файлу і поверни структуровані висновки у форматі JSON: { "classification": "", "security_issue": "", "severity": "low|medium|high", "evidence": "", "recommendation": "" } Визнач для кожного файлу окремо: 1) чи містить файл токени, паролі, сесійні ідентифікатори; 2) чи є дані у plaintext і чи відповідає формат шифруванню; 3) чи присутні ознаки порушень (нешифровані токени, відсутність exrīgu, debug-логи тощо). 4) не пиши ніяких додаткових пропозицій по подальшій роботі

У цьому додатку наведено скрипт, що використовувався для нормалізації артефактів, отриманих під час вилучення локального сховища мобільного застосунку. Цей етап був необхідною складовою підготовки даних для подальшого аналізу у межах роботи AI-модуля №2. Після отримання файлів із файлової системи (SQLite-бази, XML-конфігурації, текстові журнали, службові файли WebView та інші артефакти) виникла потреба привести їх до уніфікованого виду, який дозволив би моделі штучного інтелекту коректно обробляти та інтерпретувати їх зміст.

Скрипт забезпечує автоматичне перетворення різнорідних форматів у структури JSON, що включають ключові поля, вміст файлів, метадані та додаткову інформацію, корисну для подальшої класифікації. Така уніфікація значно спрощує аналіз великої кількості файлів, зменшує ризик втрати важливих даних та дозволяє застосовувати один узгоджений промпт для LLM-моделі.

```
import os
import json
import sqlite3
import xmltodict
import subprocess
from pathlib import Path
INPUT_DIR = "damnvulnerablebank_dump" # папка з файлами
OUTPUT_DIR = "normalized" # куди покласти результат
Path(OUTPUT_DIR).mkdir(exist_ok=True)
def is_sqlite(path):
    """Перевіряє 100% чи файл SQLite database."""
    try:
        with open(path, "rb") as f:
            header = f.read(100)
            return header.startswith(b"SQLite format")
    except:
        return False
def normalize_sqlite(db_path):
    print(f"[SQLite] Normalizing {db_path}")
    try:
        conn = sqlite3.connect(db_path)
        cursor = conn.cursor()

        cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
        tables = [t[0] for t in cursor.fetchall()]
        db_json = {}
        for table in tables:
            try:
                cursor.execute(f"SELECT * FROM {table}")
                rows = cursor.fetchall()
```

```

        columns = [d[0] for d in cursor.description]
        db_json[table] = [dict(zip(columns, r)) for r in rows]
    except:
        pass
    out = os.path.join(OUTPUT_DIR, Path(db_path).name + ".json")
    json.dump(db_json, open(out, "w"), indent=2)
except Exception as e:
    print(f" ERROR: {e}")
def normalize_xml(path):
    print(f"[XML] Normalizing {path}")
    try:
        with open(path) as f:
            data = xmldict.parse(f.read())
            out = os.path.join(OUTPUT_DIR, Path(path).name + ".json")
            json.dump(data, open(out, "w"), indent=2)
    except Exception as e:
        print(f" ERROR: {e}")
def normalize_journal(path):
    print(f"[JOURNAL] Extracting text from {path}")
    try:
        result = subprocess.run(
            ["strings", path], capture_output=True, text=True
        )
        content = result.stdout
        out = os.path.join(OUTPUT_DIR, Path(path).name + ".txt")
        open(out, "w").write(content)
    except Exception as e:
        print(f" ERROR: {e}")
def normalize_file(path):
    fname = Path(path).name.lower()
    # SQLite + Chrome WebView (Cookies, Web Data)
    if is_sqlite(path):
        normalize_sqlite(path)
        return
    # SharedPreferences XML
    if fname.endswith(".xml"):
        normalize_xml(path)
        return
    # WebView leveledb / журналні файли
    if (
        "journal" in fname
        or "manifest" in fname
        or "log" in fname
        or fname.startswith("000")
        or fname.endswith(".pma")
        or fname.endswith(".pb")
    ):
        normalize_journal(path)
        return
    # Інше просто зберігаємо як текст (LLM все одно читає)
    try:
        with open(path, "rb") as f:
            raw = f.read()
        try:
            text = raw.decode("utf-8", errors="ignore")
        except:
            text = str(raw)
        out = os.path.join(OUTPUT_DIR, Path(path).name + ".txt")
        open(out, "w").write(text)
        print(f"[RAW] Saved textual form of {path}")
    except Exception as e:
        print(f"[ERROR] Cannot process {path}: {e}")
# ---- MAIN ----

```

```
print("=== NORMALIZATION START ===")
for root, dirs, files in os.walk(INPUT_DIR):
    for file in files:
        full_path = os.path.join(root, file)
        print(f"\nProcessing: {full_path}")
        normalize_file(full_path)
print("\n=== NORMALIZATION COMPLETE ===")
print(f"Normalized data saved to: {OUTPUT_DIR}/")
```

У цьому додатку наведено інструкції, застосовані під час розробки та експериментальної перевірки AI-модуля №3, який відповідає за автоматизацію формування звітності пентесту на основі зібраних статичних і динамічних артефактів.

- Ти писатимеш опис однієї вразливості для звіту з тестування на проникнення. Напиши професійний і добре структурований запис, який строго дотримується наведеної структури та форматування. Використовуй нейтральний технічний стиль, без зайвої багатослівності. Включай перевірювані технічні докази та практичні рекомендації. Якщо доступні метадані цільової системи або часові мітки використовуй їх. Обов'язкові розділи (саме в такому порядку, саме з такими назвами): 1. Title Однорядкова назва англійською мовою (наприклад: Insecure direct object reference (IDOR) in transaction endpoint). 2. Background 2–4 короткі речення з описом контексту: який модуль/функція уражені, який рівень автентифікації потрібний, до якої області впливу це належить. Лише факти. 3. Description 3–6 речень, що описують суть вразливості, який компонент уражено, у чому полягає першопричина (некоректна валідація, логічна помилка, неправильна конфігурація тощо). Якщо це мережеве API вказати endpoint і параметри. 4. Evidence Надати компактний PoC (не більше 8 рядків), який розробник може відтворити. Використовувати code blocks для HTTP-запитів, відповідей або команд. Після PoC додати підписи до скріншотів (не вставляти самі зображення) у такому форматі: Figure 1. короткий опис зображення Figure 2. короткий опис зображення PoC повинен містити: • мінімальний відтворюваний запит (method, path, ключові параметри, payload); • ключові частини відповіді сервера (status code + чутливі поля); • команди (curl/httpie) або кроки браузера якщо доречно. 5. Recommendation 3–6 маркованих пунктів із чіткими, пріоритизованими та практичними рекомендаціями. Додати 1–2 речення про те, як перевірити виправлення. 6. References Один елемент на рядок у форматі:

CWE-XXX: <назва>, <URL> CVE-YYYY-NNNN: <назва>, <URL>
CAPEC-XXX: <назва>, <URL> OWASP: <сторінка>, <URL> (Якщо певний ідентифікатор недоречний пропустити його.) Правила стилю та обмеження:
Загальна довжина: приблизно 180–450 слів. Мова точна, нейтральна, технічна.
Уникай маркетингових фраз та надмірної емоційності. Чітко дотримуйся вимог до розділів і форматування. Виводь результат лише у вказаній структурі, без додаткових коментарів. Нижче описані дані вразливостей у форматі json: {
"filename": "pref_store.txt", "classification": "user preference storage",
"security_issue": "Паролі, токени чи ідентифікатори відсутні. Зберігає лише дані про дату встановлення, метрики користувача.", "severity": "low", "evidence":
"Дані: installation_date2, user_experience_metrics (low_entropy_source3) – не містять чутливої інформації, все у plaintext, але це не критичні дані:contentReference[oaicite:1]{index=1}.", "recommendation": "Ризик відсутній, проте для критичних налаштувань краще використовувати захищене сховище."
}