КИЇВСЬКИЙ СТОЛИЧНИЙ УНІВЕРСИТЕТ ІМЕНІ БОРИСА ГРІНЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА МАТЕМАТИКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

«Допущено до захисту»
Завідувач кафедри
доц., к.т.н. Машкіна І.В.

_____
(підпис)

«___»_____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 «Комп'ютерні науки»
Тема:

Створення адаптивного ШІ агента для оптимізації оцінки звітів вразливостей у додатках Web 3.0

Виконав:
студент групи
ІНб-2-21-4.0д
Коротких Демид
Миколайович

(підпис)

Науковий керівник:
доктор філософії
Турукало А. В.

_____
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

«Затверджую»
Завідувач
кафедри комп’ютерних наук,
кандидат технічних наук, доцент
(науковий ступінь, вчене звання)
_____ Ірина МАШКІНА
(підпис)
« ___ » _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Створення адаптивного ШІ агента для оптимізації оцінки звітів вразливостей у додатках Web 3.0»

Виконавець – студент групи ІНб-2-21-4.0д,
спеціальності 122 Комп'ютерні науки,
освітньої програми 122.00.01 Інформатика
Коротких Демид Миколайович

1. Вихідні дані: тематична література з напрямку розробки ШІ технологій та побудови агентів на базі мовних моделей, теорія валідації та оцінення складності вразливостей в додатках Web3.0, повідомлення про потенційні вразливості агреговані компанією Hacken.
2. Основні завдання:
   - Ознайомитись з тематичною літературою для набуття відповідного теоретичного підґрунтя.
   - Створити систему для опрацювання наявних репортів у навчальний матеріал для ШІ агента.
   - Обробити відповідні дані.
   - Створити MVP версію ПЗ та провести тестування гіпотези.
     Імплементувати ШІ агента в платформу HackenProof.
3. Пояснювальна записка: Обсяг – 63 стор. формату А4 комп’ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: презентація, знімки екрана роботи програми, блок-схема процесів обробки інформації.
5. Строк подання роботи на кафедру: «____» _____ 2025 р.

Науковий керівник
доктор філософії
Турукало Андрій Валерійович

Виконавець:
Коротких Демид Миколайович
_____

_____ Дата:

Календарний план

№	**Етап виконання роботи**	**Термін виконання**	**Примітка**
1	Формування теми дипломної роботи бакалавра та її затвердження, планування робочого процесу.	04.10.2024-14.10.2024	Виконано
2	Збір та вивчення релевантної літератури.	16.10.2024-02.12.2024	Виконано
3	Аналіз та порівняння наявних аналогічних рішень в індустрії ШІ та кібербезпеки.	02.12.2024-23.12.2024	Виконано
4	Вибір технологій для реалізації проєкту дослідження.	06.01.2025-10.01.2025	Виконано
5	Розробка алгоритму обробки даних.	13.01.2025-31.01.2025	Виконано
6	Тестування навчання ШІ на даних — результатах розробленого алгоритму.	03.02.2025-07.02.2025	Виконано
7	Створення MVP версії ПЗ. Тестування з командою аналітиків безпеки, брейншторм.	10.02.2025-21.02.2025	Виконано
8	Корегування навчального процесу ШІ агента, та уніфікація алгоритму обробки даних.	03.03.2025-14.04.2025	Виконано
9	Створення копії ШІ агента та розміщення його у хмарному сховищі. Інтеграція ШІ агента на платформу HackenProof	17.04.2025-06.05.2025	Виконано
10	Тестування ШІ агента в реальних умовах.	06.05.2025-	Виконано

11	Формування пояснювальної записки та інших матеріалів для захисту кваліфікаційної роботи.	12.05.2025-16.05.2025	Виконано
12	Передзахист кваліфікаційної роботи.	21.05.2025	Виконано
13	Виконання роботи над помилками та доопрацювання звітних матеріалів	21.05.2025-05.06.2025	Виконано
14	Захист кваліфікаційної роботи.	19.06.2025	Виконано

Здобувач *Коротких Д.М.*

Керівник роботи *Турукало А.В.*

РЕФЕРАТ

Кваліфікаційна робота: 55 с., 10 рис., 36 посилань.

Актуальність: Чимраз більша популярність криптовалют привертають увагу до Web3.0 і збільшення користувачів пропорційно збільшує ризики атак на додатки блокчейну. Для запровадження більш ефективних безпекових заходів необхідно оптимізувати процес обробки повідомлень про потенційні вразливості та закцентувати роботу над найбільш вірогідними вразливостями. Роль застосування ШІ у цьому напрямку важко переоцінити.

Об’єкт дослідження: Процес обробки та оцінювання звітів про вразливості в системах кібербезпеки Web3.0.

Предмет дослідження: Можливості використання ШІ у якості інструменту попереднього аналізу звітів про потенційні вразливості у додатках Web3.0.

Мета роботи: Створення адаптивного ШІ агента для оптимізації оцінки звітів вразливостей у додатках Web 3.0

Завдання:

- Ознайомитись з тематичною літературою для набуття відповідного теоретичного підґрунтя.
- Створити систему для опрацювання наявних репортів у навчальний матеріал для ШІ агента.
- Обробити відповідні дані.
- Створити MVP версію ПЗ та провести тестування гіпотези.
  Імплементувати ШІ агента в платформу HackenProof.

Основні результати:

- Розроблено систему для опрацювання інформації та навчання ШІ агента.
- ШІ пройшов стадію MVP та було імплементовано на платформу HackenProof.

- Проведено порівняльний аналіз з наявними аналогами.

Практичне значення дослідження:

Створено програмне рішення для визначення пріоритетності звітів про потенційну вразливість в кореляції до вірогідності існування заявленої вразливості. Це ПЗ імплементовано та протестовано на платформі HackenProof.

Ключові слова: Кібербезпека, Штучний інтелект, Web3.0, Вразливості, LLM, пріоритизація, HackenProof, SCVSS/OWASP, Натренований агент, MVP.

ЗМІСТ

ВСТУП

Розвиток технологій є завжди асиметричним та потребує постійного докладання зусиль для розробки нових варіантів застосування чи поєднання наявних технологій. Часто новий крок розвитку технологій пов'язаний з застосуванням чинних концепцій в нестандартному поєднанні. Або покращення взаємодії між двома напрямками, використання інструментів в нових перспективах і таким чином розширювати можливості окремих галузей, завдяки технологічним інноваціям.

Метою цього дослідження є створення ШІ агента, що може підсилити команду аналітиків з безпеки у визначенні пріоритетності репортів про потенційні вразливості. Що своєю чергою пришвидшить процес обробки всі повідомлень від спільноти та зробить весь блокчейн простір відчутно безпечніше. В практичному просторі ця мета виражена трьома простими пунктами:

- Створення системи обробки інформації та навчання ШІ моделі
- Розробка першої (MVP) версії ПЗ для тестування гіпотези
- Інтеграція ШІ агента у платформу HackenProof

Я вже більше за три роки є штатним співробітником компанії Hacken, в позиції "Аналіти з безпеки" на платформі HackenProof. HackenProof є платформою для пошуку вразливостей за винагороди, себто наші клієнти публікують своє ПЗ для розгляду спільноти "white-hat hackers" (етичних хакерів) і у разі знаходження вразливості, клієнт винагороджує етичного хакера за його роботу. В цьому ланцюгу ми виконуємо роль посередника-регулятора! Наша команда визначає правила за якими мають працювати етичні хакери, та також є незаангажованою стороною для визначення серйозності знахідки та оцінки відповідної винагороди. Влітку 2023 року наша команда зіткнулась з потребою у скороченнях, через складну

ситуацію на рику та в індустрії, це було першим потужним поштовхом для вивчення можливостей оптимізації робочих процесів за допомогою ШІ. Саме тоді почалась робота над пошуком найкращих доступних варіантів застосування ШІ в повсякденній роботі, два роки тому ШІ був відчутно обмежений у своїх можливостях, і хоча зараз можливості нейромереж відчутно зросли — все одно є потреба в пильному контролі роботи мовної моделі професіоналом. Тому на меті мого дослідження є створення саме ШІ агенту який буде інструментом для команди аналітиків безпеки, а не самостійним актором.

Для розробки ШІ агента, як базу я обрав мовну модель Microsoft Phi-4. Ця невибаглива модель (лише 14 мільярдів параметрів) може ефективно виконувати завдання з невеликими технічними ресурсами, додатково якісні матеріали на який модель була натренована в процесі створення, робить ще легше доопрацювання-перенавчання моделі під задачі поставлені в цьому дослідженні. Також, для пришвидшення тестування гіпотези та створення MVP версії я використовував мовну модель з тої ж родини — Phi-4-mini, яку можливо використовувати локально на комп’ютері.

Для обробки даних та навчання ШІ було обрано такі інструменти як NLTK — обробка тексту до формату який сприймається мовними моделями, та spaCy — бібліотека для вилучення ключових слів, а також ембедінгу та подальшої роботи з текстом.

Результатом цього дослідження є створення системи обробки репортів про потенційні вразливості, тренування ШІ агента на базі обробленої інформації та інтеграція цього ШІ в платформу HackenProof. Очікуваним результатом є пришвидшення роботи команди аналітиків безпеки та потенційне покращення якості оцінювання повідомлень про потенційні вразливості.

РОЗДІЛ ПЕРШИЙ:
Огляд літератури та Підготовчий етап

1.1 Актуальність теми; Огляд літератури.

Ця дослідницька робота орієнтована на точку перетину сучасного ШІ та потреб в нових підходах до безпеки в просторі Web3.0. Перший квартал 2025 року відзначився рекордними втратами серед блокчейн протоколів і подібних продуктів.

Рис. 1.1 — Динаміка зростання ринку Web3.0 та втрат Web3.0 через злами

Покращення і пришвидшення процесу виявлення загроз та вразливостей має реальний вплив на всю індустрію пов’язану з криптовалютою. Я вже більш як 3 років працюю аналітиком з безпеки в провідній компанії з аудиту коду для смарт-контрактів та блокчейн протоколів. І з розумінням швидкості з якою оновлюються як програмна, так і виконавча компонента роботи з блокчейном та ШІ — кожен з цих напрямків зараз є дуже популярним та притягує щорічно дедалі більшу кількість

користувачів і розробників — тому я обрав спиратись на джерела оновлені не пізніше попередніх 12-18 місяців, аби не працювати з застарілою та не актуальною інформацією. Через мою дотичність до блокчейн галузі інформаційних наук, я намагався знаходити нових для себе авторів, що публікували роботи на цю тему, щоб отримати незаангажовані дані та нову перспективу на потреби цієї технології! Через швидкий розвиток не лише програмного забезпечення, але і безпосередньо обладнання, більшість авторів мають не лише теоретичний доробок. Саме доступність необхідних обчислювальних потужностей в західних навчальних та промислових центрах створює умови в яких більшість теоретичних концепцій на яких базується проведене дослідження, вже були перевірені в лабораторних умовах та й навіть в випадках комерційного застосування.

Найрелевантнішою для мене була робота за редакторства Hadj Ahmed Bouarara: "Advanced Machine Learning, AI, and Cybersecurity in Web3.0: Theoretical Knowledge and Practical Application". Hadj Ahmed Bouarara має досвід в розробці ШІ агентів орієнтованих на виявлення шахрайства в середовищі крипто-мереж. За дослідженням проведеним компанією Hacken OÜ — найпоширеніша причина зламу блокчейн систем є саме шахрайство як призводить до вразливостей у системі контролю доступу. Також Hadj Ahmed Bouarara має розробки для визначення та попередження зламів в децентралізованих мережах, за допомогою ШІ він створив інструменти для розпізнавання патернів аномальної активності, шахрайських операцій та маніпуляцій мережею.

Книга "Artificial Intelligence for Blockchain and Cybersecurity Powered IoT Applications" за редакції групи науковців (Mariya Ouaissa, Mariyam Ouaissa, Zakaria Boulouard, Abhishek Kumar, Vandana Sharma, Keshav Kaushik) є також доречним прикладом використання ШІ та технологій блокчейну в рамках безпекових заходів. Ця робота наводить практичні приклади

використання ШІ та блокчейн мережі для нових фронтирів безпеки. Одним з прикладів є створення біометричної системи аутентифікації яка використовує інфраструктуру в мережі блокчейну для збереження та верифікації даних та ШІ для попередження і нівелювання DDoS атак на пристрої IoT-мережі.

Книга Hadj Ahmed Bouarara мала вже знайомий для мене підхід до використання і комбінування ШІ з Web3.0 екосистемою, але робота "Artificial Intelligence for Blockchain and Cybersecurity Powered IoT Applications" дала мені новий погляд на потенціал використання цих технологій. Я навіть на деякий час повернувся до роботи над своїм проєктом з першого курсу університету метою якого була розробка IoT девайса для покращення досвіду використання 2FA без компромісів щодо безпеки такого захисту.

Неочікуваним викликом роботи з джерелами була кількість нерелевантної інформації. Через специфіку яку я вбачаю в обраному проєкті для дослідження — генеративний ШІ, до прикладу, як напрямок розвитку ШІ технології немає чіткого стосунку до напрямку який я обрав до дослідження. Мій орієнтир був на вивченні методів поєднання ШІ з Web3.0 екосистемою, пошук теперішніх прикладів відповідного ПЗ та оптимізація роботи ШІ з заданою базою даних. Для цього я заглибився у вивчення процеси з яких вибудовуються сучасні LLM та можливості NLP і як застосування propt-engennering впливатиме на результати роботи ШІ агента. Додатковою метою цього дослідження стала інтеграція ШІ агентів до застосунків, як додаткового функціоналу.

"Однак із великою силою приходить велика складність. Робота LLM, хоча й захоплююча, не є одразу доступною для всіх. Складна архітектура, базові алгоритми та етичні міркування, що супроводжують впровадження LLM, є предметами життєво важливого значення, які потребують ретельного вивчення. Тут стає очевидною потреба в комплексній книзі про LLM. Існує

нагальна потреба в ресурсі, який не лише демістифікує технічну роботу цих моделей, але й контекстуалізує їхній вплив, досліджує їхні застосування та вирішує етичні дилеми, які вони породжують.": Large Language Models: A Deep Dive — Bridging Bridging Theory and Practice авторства Uday Kamath, Kevin Keenan, Garrett Somers, Sarah Sorenson (2024) [7].

Ознайомившись з усім масивом джерел я окреслив першочерговою метою цієї роботи — тестування гіпотези щодо використання ШІ в галузі кібербезпеки. Я не буду висвітлювати всі дрібні деталі цього процесу, а заакцентую увагу цієї роботи на розв'язання питання щодо доцільності й загальної користі та етичності такого рішення.

1.2 Сучасний стан проблеми.

Завдяки роботі безпосередньо в цій сфері та підтримці керівництва в дослідженні цієї теми, мої ресурси є дещо більшими, але з цим приходить і відповідальність. Оскільки розробка такого рішення відбувається на базі компанії Hacken OÜ — інтелектуальний доробок належить також цій організації. Моє NDA не дозволяє розголосити всіх даних над якими я мав можливість працювати та тестувати описану вище гіпотезу, тому в цій роботі я викладу лише основні принципи які було сформульовано протягом розробки ШІ рішення для оптимізації роботи безпекового аналізу. Для демонстрації технологічних рішень буде використано MVP модель ШІ системи, але сама база даних підготовлена як частина цього дослідження — яка є левовою часткою практичного доробку, не підлягає публікації та розголошенню. Тому це дослідження вимушено обмежитись викладенням основної концепції яка була стала підґрунтям для створення відповідної бази даних. Унікальним в цьому випадку є саме об’єм та якість даних які були опрацьовані в результаті тестування цієї гіпотези, а сам fremework роботи з даними базується на галузевих стандартах та поєднанні багатьох перевірених методів для інтерпретації й валідації даних. За час роботи в компанії Hacken

ОЇ було зібрано понад 22 тисяч різноманітних повідомлень про потенційні вразливості, та всі вони були валідовані безпосередньо спеціалістами, це великий об’єм даних для тренування ШІ. З таким ресурсом я мав можливість не лише навчити нейронну мережу розпізнавати патерни, але й створити базу даних для перевірки й подвійного тренування — тюнінгу ШІ агента, як це на сленгу називають розробники ШІ.

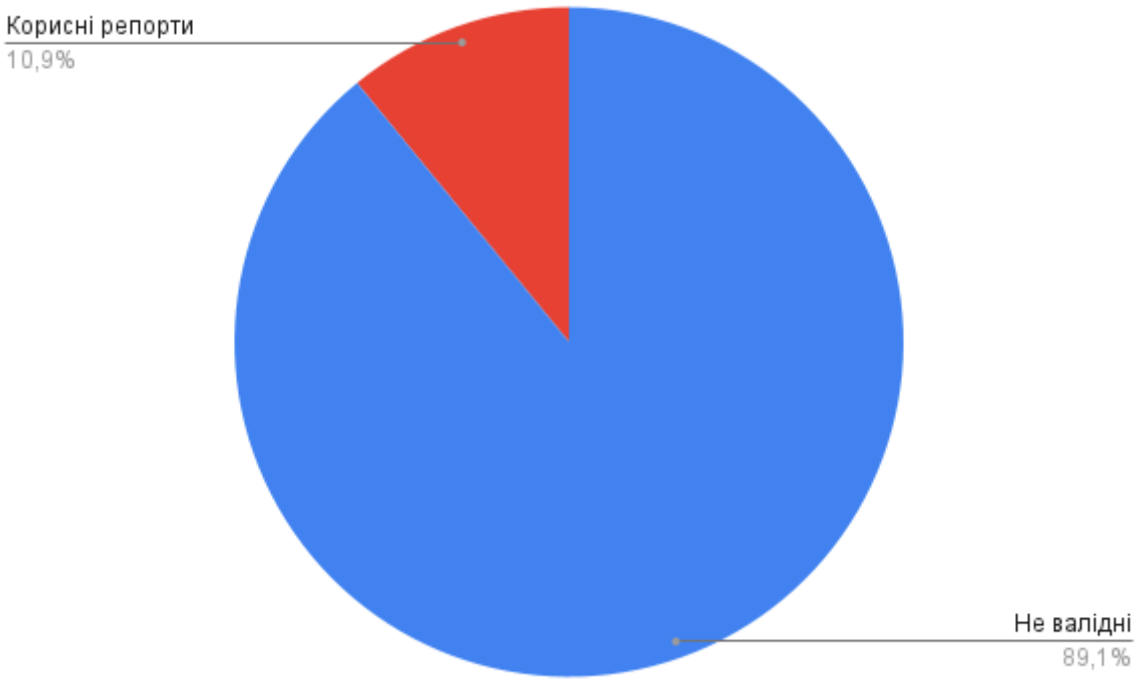

Рис. 1.2 — Відношення сигналу до шуму

Навіть на базі достатньо великої корпорації як Hacken ОЇ я стикнувся з відчутними викликами про розробку прототипу ШІ агента для аналізу повідомлень етичних хакерів про потенційні загрози. Одна з найбільших задач яку я мав на меті вирішити — створення простої структури на базі CVSS калькулятора та методологій прийнятих в середині компанії для оцінки репортів ШІ для додаткової точки порівняння на яку може зважати аналітик безпеки при перегляді повідомлення. Звичайно що ШІ агент в жодному випадку не може замінити кваліфікованого аналітика, але це і не є метою

гіпотези дослідження. Першочергова функція ШІ агента в моєму випадку використання — пришвидшення роботи аналітичної команди та швидший процес порівняння нових повідомлень від whitehathackers ком’юніті до попередніх. Черговим викликом було знайти рішення для оптимізації пошуку по масиву даних без втрат у "розуміння" ШІ контексту в якому використані терміни за якими проводиться пошук.

Детальніше про технології використані в розробці, оптимізації та імплементації розробленого агента ШІ розписано у другому розділі цієї роботи.

Галузь блокчейн технологій та Web3.0 має великий потенціал до інтеграції з нейромережами, лише перехід від принципу PoW до PoS вивільнив багато машинних потужностей, і таким чином частково вплинув на пришвидшення розвитку ШІ. Окремо взяті продукти побудовані на блокчейн технології мають велику популярність через свою відкритість, але це і створює виклики які можуть бути вирішені за допомогою кращої інтеграції ШІ в мережі блокчейну. Завдяки постійному доступу до переліку всіх операцій що відбуваються в мережі — блокчейн може вважатись більш безпечним за звичний користувачам WEB2, а також акцент на анонімності акторів мережі. З іншого боку, це є викликом до всієї галузі — як опрацьовувати цей нескінченний потік нової інформації. Оптимізації та перехід на L2 та RollUp/ZeroCode RollUp створюють додатковий шар інформаційного гамузу який дедалі важко організовано оглядати, послуговуючись попередньо прийнятими інструментами моніторингу мережі.

Саме через здатність ШІ безперервно і чітко опрацьовувати безперебійний потік інформації є найкращою відповіддю на сучасні виклики до блокчейн технологій. Створюючи системи перевірок, поєднуючи ШІ з роботою кваліфікованого персоналу вже має приклади покращення

загального безпекового рівня мережі. Звісно це не є кінцевою точкою розвитку — вже було багато прикладів в яких поєднання двох технологій народжувало нову галузь. Деякі діячі у сфері комп'ютерних наук вбачають, що в майбутньому дуже вірогідне створення децентралізованого ШІ який буде послуговуватись як фізичними, так і інформаційними ресурсами радше від розрізнених організацій та навіть окремих акторів, ніж централізовано, як зараз — принципово це є прикладом застосування технології блокчейн для роботи всієї моделі ШІ. ШІ та Web3.0 не просто добре співпрацюють, але від свого започаткування — технічних рішень що уможливили існування кожної з цих технологій — блокчейн та нейромережеві обчислення є зв'язаними, та навіть взаємопов'язаними. Я впевнений що багато екзистенційних питань (таких як оптимізація, прибутковість та організація процесів обробки запитів) що постають перед ШІ можуть бути вирішені з імплементацією блокчейн технологій, або ж навіть основних принципів на яких побудовано криптовалютні системи. Та навпаки — проблеми розширення та більш широкої імплементації в чинній екосистемі – основні виклики для Web3.0, можуть вирішуватись з застосуванням ШІ, це рішення може бути максимально простим — фактично спрощення самого процесу взаємодії користувача з блокчейн системою через ШІ агента, може привабити більш широкі маси користувачів, адже зараз для багатьох саме складність роботи з блокчейном є перешкодою до використання цієї технології в повсякденні.

Завдяки моєму досвіду у вразливостях які можливі в коді до програмного забезпечення яке утворює мережу блокчейн, або ж працює (відтворюється) в цій мережі, а також закумульованою базою повідомлень про виявлені вразливості/потенційні вразливості — я маю необхідні базові знання та базу даних для тренування ШІ агента який в майбутньому буде інтегровано в середовище роботи аналітиків безпеки для пришвидшення і підвищення якості виконання роботи! Опанування та імплементація ШІ в

робочих процесах зарекомендувало себе як надійний шлях до підвищення ефективності та прибутків бізнесу. Через економічні умови сьогодення, розробка відповідного ПЗ максимально схвалюється та підтримується керівництвом нашої організації. В недалекому майбутньому специфічні ШІ агенти будуть розцінюватись як інструменти та частина необхідного забезпечення і наявність таких розробок зробить компанії не лише успішніше, але й привабливіше для потенційних співробітників!

РОЗДІЛ ДРУГИЙ:

Методика та перебіг дослідження

2.1 Методи дослідження.

Для максимізації коректності та повноти виконання дослідження я використав багато різних методів дослідження, більшість з них полягала в вивченні наявних джерел тематичної інформації та застосування набутих знань в експериментальних умовах. Фактично експеримент був частиною моєї методології дослідження, адже було важливо зрозуміти чи запропоновані мною рішення можуть мати реальний вплив на стан речей в індустрії.

Серед методів дослідження я намагався дотриматись максимально різностороннього набору, тому крім вивчення відповідної літератури, моделювання та експериментів я проводив брейншторм сесії зі своїми колегами для того, щоб отримати не лише фідбек до власних ідей, але і разом знайти нові можливості для покращення і нових застосувань ШІ у сфері кібербезпеки в Web3.0.

Насиченість тестовими даними та різносторонність підходів до проведення дослідження допомогло мені створити максимально повну картину сучасного стану речей в індустрії інформаційних технологій, в розрізі інтеграції ШІ агентів в процеси забезпечення кібербезпеки для продуктів пов’язаних з блокчейном. Для кращого розкриття процесу

дослідження далі я наведу приклади використаних методів дослідження, опрацьовані дані та результати використання кожного конкретного методу. Важливо зазначити, що хоча не всі з використаних методів мали практичну компоненту, до фінального проєкту цієї роботи вони всі доклали відносно порівнянну кількість корисних даних. Безпосередньо підготовчі стадії з вивченням наявних проблем, рішень та ініціатив в галузі на розгляді — з самого початку відкоригувало деякі аспекти подальшого перебігу дослідження.

Багато традиційних методів досліджень не були мною використані, через інноватівність та специфічність теми дослідження — ще немає аналогічних технологічних процесів для дослідження їх в ретроспективі чи в призмі аналогій. Унікальність нашого часу в сфері прогресу технологій часто характеризують як зміну "раз на тисячу років". І я вважаю можливим порівняти винахід та популяризацію комп’ютерних технологій за довгостроковими наслідками з глобалізацією та індустріалізацією. Але це ставить перед дослідниками зараз великий виклик — як дослідити те що досі в стані розвитку і не є сталою в жодному з уявних вимірів. Знов таки, це обмежило літературні джерела, адже актуальність даних для вивчення точно зараз є однією з найважливіших метрик. За таким самим принципом я відмовився від методів спостереження, анкетування, історичного методу, формалізації (також, тому що моделювання комп'ютерних систем іноді навіть легше за їх формалізацію), та порівняльний аналіз з розглядом нормативів (бо існування нормативних практик в площині ШІ та блокчейну фактично ще не є прийнятою і загальновживаною доктриною, існують різні варіанти регуляцій для кожної з цих галузей, але жодна з них не була стандартизована). Однак за результатами мого дослідження я не вбачаю що обмеження в методах дослідження знизило чи в якомусь відчутному сенсі

повпливало на якість даних отриманих та опрацьованих. Частково завдяки моєму залученню і взаємодією в галузі через роботу.

2.2 Аналіз ринку.

Вже існують схожі рішення — більшість з них за архітектурою є незалежними додатками/інструментами/платформами. Для мого випадку я бачу користь проаналізувати їх досвід, хоча імплементація в моєму дослідженні матиме дещо іншу форму — а саме додати ШІ до вже наявної платформи та натренувати агента виконувати конкретні задачі.
Аналіз вже наявних рішень є для мене простою задачею, через те, що я вже більш як два роки тестую різні інструменти такого типу та маю досить великий досвід з кожним з них!

Один з гарних прикладів є інструмент AuditWizard:

Цей інструмент створено для виявлення причин вже знайдених проблем, застосування ШІ допомагає оптимізувати цей процес і пришвидшити розв'язування питань щодо виправлення помилок. Хоча сам застосунок пропонує багато варіантів як додати дані для роботи (за посиланням на GitHub репозиторій чи навіть напряму за адресою в блокчейні) — це не є автономним інструментом. Для застосування AuditWizard потрібно завчасно мати добре розуміння що саме є проблемою і тоді цей інструмент допомагає знайти першопричину виникнення помилки. Інтерфейс цієї платформи є дуже зрозумілим і зручним, пропонує не лише ШІ агента, але й мануальні тести — користувач може використати вже готові тестові програми для перевірки власного коду на знані вразливості, або ж створити самостійно унікальні тести для свого коду та розробити цілий робочий графік для тестування аби використовувати його в майбутньому при оновленні власного коду!

Недоліком в конкретному випадку, що я розглядаю, є відсутність автономності — лише персонал що користується AuditWizard може отримати від нього потрібну інформацію за запитом, немає можливості вбудувати й автоматизувати використання цього інструменту (фактично цей застосунок і не був запланований для такого використання). І також наразі ШІ агент не має відчутної різниці в результатах порівняно до ШІ від таких великих компаній як OpenAI(Chat-GPT) та Google (Gemini) — за моїми спостереженням, ШІ агент вбудований в AuditWizard не був окремо тренований, а лише системно заангажований для роботи в цьому специфічному напрямку.

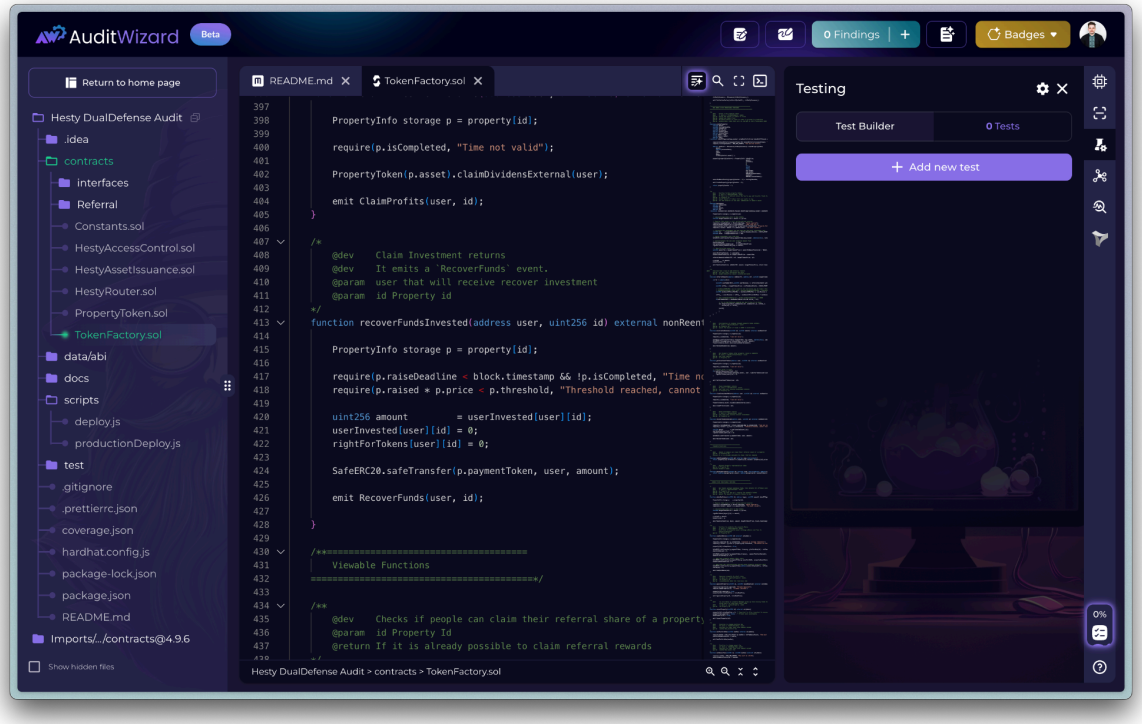

Рис. 2.1 — Інтерфейс AuditWizard

Також на вістрі застосування ШІ в інструментах для аудиту коду та перевірки безпеки є AuditBase. Через колаборацію Hacken з AuditBase я мав можливість застосовувати цей інструмент у найширшій вибірці сценаріїв. Сам інструмент AuditBase побудовано для сканування коду на наявність

потенційних вразливостей та структурні помилки, хибні чи застарілі залежності (взаємозв’язки); аналіз за допомогою LLM на наявність логічних та бізнес помилок і передбачення можливих похибок у зв’язку з зовнішніми змінними та створення звітів про актуальний стан системи/коду — остання функція дозволяє вбудувати цей інструмент в процес роботи команди з розробки та до певної міри автоматизувати процес сканування коду на наявність потенційних безпекових і бізнесових вразливостей!

З недоліків — обмеження AuditBase до лише однієї мови — Solidity та одного блокчейн протоколу — EVM. Хоча це дуже популярні рішення в розробці блокчейн технологій, паралельно з ними існує ще велика частка блокчейн протоколів та мов програмування які займають не малу відсоткову частину цього напрямку, галузі блокчейн технологій. З одного боку, такий фокус і акцент на одній технологічній гілці блокчейн галузі є важливою компонентою успіху AuditBase, але в перспективі це є великою проблемою яка може бути вирішена лише накопичуванням та обробкою даних що стосуються інших мов програмування та їх нативних мереж для навчання ШІ й розробки тестувальних програм під них. Це підсвітлює перевагу яку я маю як частина команди HackenProof — доступ до великої бази даних яка може бути перетворена на навчальні матеріали для ШІ. Фактично зараз фразеологізм "Інформація це нове золото" стала реальністю, адже всі шукають нові стеки даних для навчання власних нейромереж і “тюну”, себто підлаштування їх до нових сфер використання.

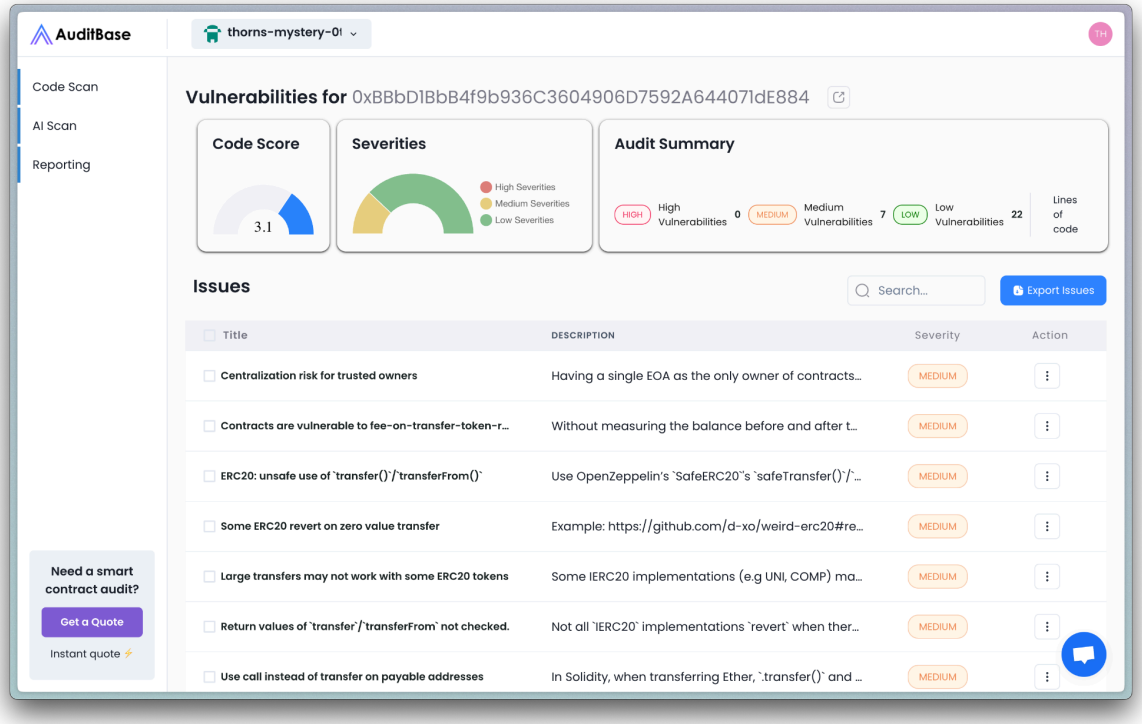

Рис. 2.2 — Інтерфейс AuditBase

З інших позицій, але теж з використанням ШІ підходить продукт компанії CredShields — SolidityScan. Цей інструмент також працює лише з Solidity кодом та не має інтеграцій в блокчейни не на базі EVM протоколу. Що суттєво вирізняє SolidityScan поміж іншими інструментами для аналізу коду та пошуку вразливостей — SolidityScan має можливість аналізувати код в реальному часі та оглядати весь процес виконання програм (on-chain) вже після публікації смартконтракту на блокчейн. Однією з найважливіших функцій є можливість миттєвих повідомлень про знайдені потенційні вразливості та одразу формулювати потенційні варіанти виправлення коду.

Але недоліком цього рішення є висока ціна на використання — при середньому розмірі проєкту у 20-50 тисяч рядків коду, часто ціна є порівнянний з аудитом коду від спеціалістів у галузі!

Тому зараз SolidityScan є найбільш поширеним саме серед невеликих проєктів, через свою відносну дешевизну, порівняно з послугами аудитора,

але це рішення стає менш привабливим при збільшенні код-бази та при роботі з іншими протоколами блокчейну.

Важливо зазначити, що через великий доробок CredShields у галузі аудитів коду — ШІ який вони використовують має відчуття переваги над результатами стандартних моделей та агентів з лише системною директивою.

Дивлячись на ці приклади, я бачу чіткий потенціал в розробці принципів тренування ШІ агента під цю специфічну галузь, база репортів накопичена Hacken має великий потенціал та створює перевагу.

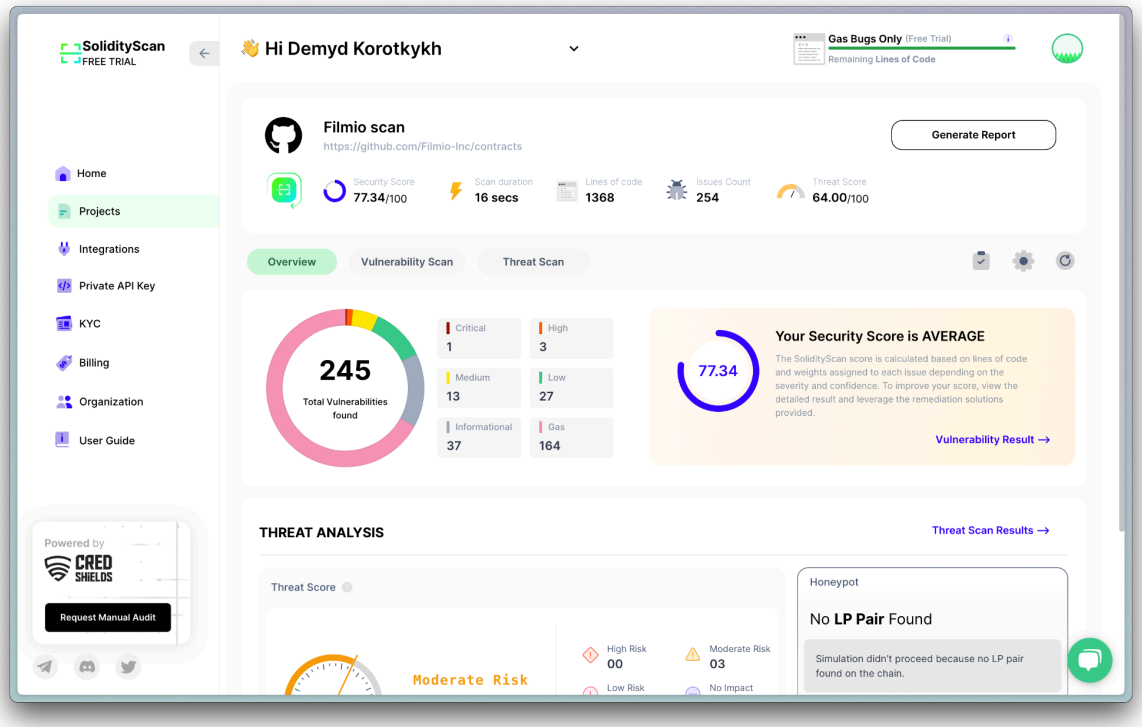

Рис. 2.3 — Інтерфейс SolidityScan

2.3 Обробка даних для навчання. Класифікації вразливостей.

Одним з перших завдань є обробити дані які в наступних кроках реалізації можна буде використати для навчання ШІ агента, та перевірки його результатів.

Крім потреби в анонімізації даних та чистки її від "шумів", а також проведення ембедінгу та обробки векторів в матриці — потрібно чітко закріпити з самого початку яку класифікацію вразливостей для Web3.0 додатків та блокчейн протоколів буде використано для підґрунтя. Мій вибір зупинився на OWASP Smart Contract Security Testing Guide (SCSTG). OWASP є респектабельною фондацією, що вже протягом років створює єдиний реєстр регулярних вразливостей. З самого початку вони орієнтувались на WEB2 застосунки, але згодом доєднались ентузіасти блокчейну і в цій колаборації було створено OWASP SCSTG — перелік поширених вразливостей безпеки та конфіденційності в смарт-контрактах, це фактично компліментарний перелік вразливостей, частково аналогічний до стандартного з уточненнями саме специфіки для блокчейн коду і типових вразливостей галузі.

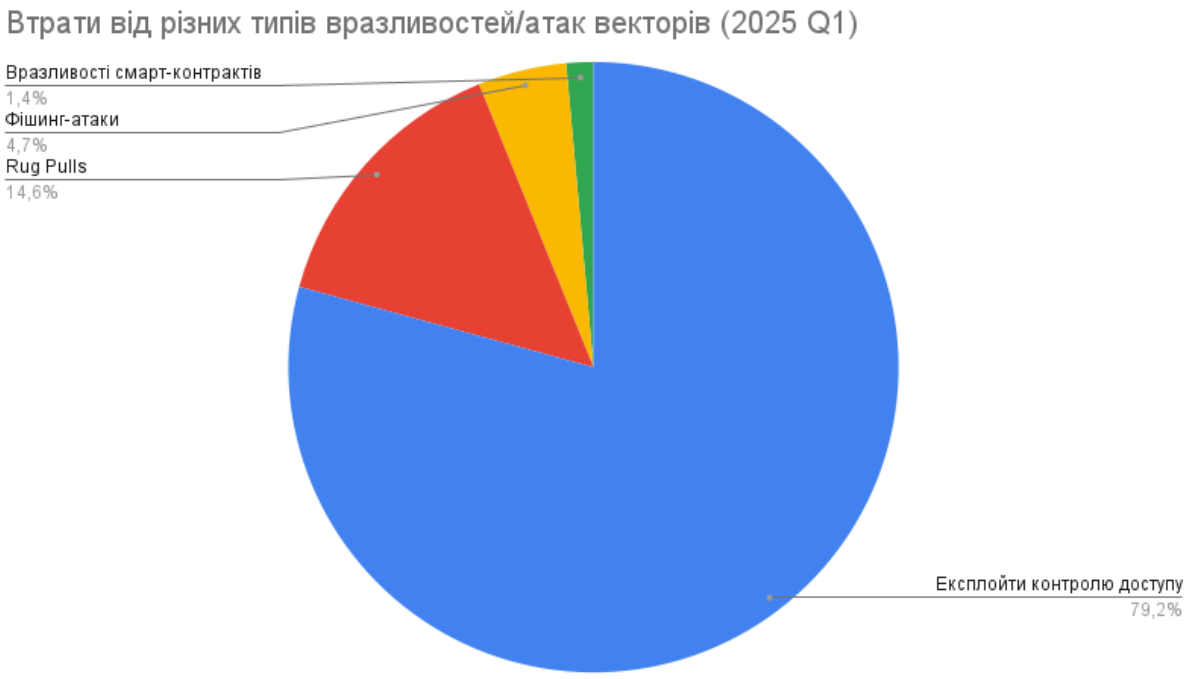

Рис. 2.4 — Втрати в Web3.0 додатках у першому кварталі 2025 від різних типів вразливостей

Загалом більшість вразливостей можуть бути розміщені в 11 категорій які є частиною Smart Contract Security Verification Standard (SCSVS) — стандарту перевірки безпеки смарт-контрактів.
Нижче наводжу кожну з категорій та приклади реальних вразливостей:

1. SCSVS-ARCH — Architecture — ці вразливості пов’язані з архітектурними вразливостями смарт-контрактів, блокчейн протоколів та інших програм зі сфери Web3.0. Ненадійна ініціалізація контракту або процес оновлення ПЗ чи неправильні межі доступу — одні з найпоширеніших типів вразливостей та можуть бути фатальними для всієї системи.

Прикладами таких вразливостей є SCWE-005: Insecure Upgradeable Proxy Design — вразливість яка дозволить зловмисникам перехопити оновлення ПЗ, підмінити його на своє чи модифікувати, що неодмінно призведе до збитків. А також SCWE-070: Incorrect Constructor Name — використання застарілих угод про імена конструкторів, що призводить до того, що функції стають загальнодоступними — тобто вразливими до будь-яких маніпуляцій без жодної авторизації.

2. SCSVS-CODE — Code Quality and Safety — через особливості блокчейн систем (всі програми є незмінними з моменту їх публікації в мережі та лише завдяки проксі можливо змінювати програму без потреби зміни всіх інших залежностей в мережі) — помилка на рівні неправильного синтаксису або ж неініціалізованої змінної — можуть призвести до відчутних втрат ресурсів та репутації.

Для демонстрації можемо розглянути: SCWE-046: Reentrancy Attacks — одна з доленосних вразливостей (The DAO hack) — перший випадок такої атаки спричинив невідворотні зміни у світі криптовалют, це вразливість яка полягає в можливості звернення до зовнішніх джерел (external call) до того як оновити внутрішній лог функції. Хоча зараз це є не дуже частою проблемою

для сучасного ПЗ — перевірка на захищеність від такого вектора атаки є базовою потребою будь-якого смарт-контракту та ПЗ в мережах блокчейн систем.

3. SCSVS-GOV — Governance and Administrative Controls — в кожній системі та програмі є привілейовані ролі в тій чи іншій формі, цей розділ сконцентрований на вразливостях управління блокчейн системами та Web3.0 застосунками та ресурсами. Часто через занадто закриту систему може бути надано неправомірну перевагу конкретному актору, що компрометує всю систему для централізованих атак — сценаріїв в який або конкретний актор починає діяти на шкоду системі, або ж він є жертвою атаки й через привілейований доступ цього актора злочинці можуть виконати власний вектор атаки.

4. SCSVS-AUTH — Authentication and Authorization — цей розділ в першу чергу зорієнтований на класифікації вразливостей пов’язаних з аутентифікацією та верифікацією користувачів. Частково він є споріднений з SCSVS-GOV, основною відмінністю є акцент на верифікацію звичайних користувачів яких звісно в рази більше, ніж акторів адміністративного рівня. Але саме можливість зі звичайного акаунту користувацького рівня виконувати адміністративні команди є основною причиною зламів блокчейн систем, смарт-контрактів та іншого ПЗ в мережі блокчейну.

Для кращого розуміння цієї категорії давайте розглянемо такі вразливості:

SCWE-016: Insufficient Authorization — в цьому випадку відсутність авторизації перед викликом функції може призводити до ескалації ролей та виклику функції будь-яким публічним акаунтом. Часто саме через таку вразливість стаються інциденти з перехопленням акаунтів та наступними маніпуляціями над Web3.0 застосунком.

SCWE-018: Use of tx.origin for Authorization — використання вразливих бібліотек для авторизації, або ненадійних варіантів авторизації (як в наведеному прикладі) є однією з варіацій вразливості цієї категорії. SCWE-018 має на увазі варіант авторизації який є вразливим для фішингових атак та тихого фішингу і front-runnong атак.

SCWE-085: Insecure Use of msg.sender — виявлення цієї вразливості свідчить про використання msg.sender у контексті який залишатиме можливість маніпуляції функцією (в поєднанні з, наприклад атаки подвійного доступу) це складна і дуже небезпечна вразливість яка призводить до несанкціонованих маніпуляцій над повідомленнями в мережі.

5. SCSVS-COMM — Communication and Messaging — в межах побудови WEB2 застосунків часто використовуються API чи HTTP/S виклики для комунікації між різними серверами, програмами та застосунками. У парадигмі Web3.0 більшість комунікації відбувається на рівні смарт-контрактів і подекуди завдяки Оракулам може бути отримана стандартна інформація з-за меж блокчейн мережі.

Зв'язок та обмін повідомленнями є важливою частиною сучасного блокчейну тому важливо приділяти увагу і вразливостям цього розділу, таких як:

SCWE-021: Unsecured Data Transmission — передача конфіденційних даних без відповідної верифікації, хешування або шифрування.

SCWE-022: Message Replay Vulnerabilities — вразливість що дозволяє повторне надсилання повідомлень та фактичне виконання однієї операції двічі чи більше разів без додаткової авторизації (Схоже до SCWE-046, але на практиці часто менш небезпечна вразливість).

SCWE-035: Insecure Delegatecall Usage — виклик сторонніх акторів (делегатів) без належної перевірки, що створює умови для маніпуляції блокчейн мережею.

6. SCSVS-CRYPTO — Cryptography — для забезпечення автентичності, цілісності та випадковості Web3.0 застосунки послуговуються криптографією. Якщо використана система для криптографії є вразливою — вся мережа блокчейну наражається на небезпеку, але в певних випадках (для генерації не впливових даних) — ці застереження можуть ігноруватись задля забезпечення оптимізації та покращення виконання програм. Але в жодному випадку є неприпустимим зберігання та використання криптографічних ключів у ненадійно (SCWE-025) та налаштування ввідних параметрів у такий спосіб який може створити ситуацію в якій буде згенеровано однаковий хеш для багатьох значень (SCWE-074).

7. SCSVS-ORACLE — Oracles and External Data — Оракули це ПЗ яке виконує функцію мосту між Web3.0 та WEB2, за потреби передачі інформації в обидва напрямки. Атака на оракула та загалом вразливість цього мосту впливає на загальний рівень довіри між внутрішнім та зовнішнім середовищем. Маніпуляція зовнішніми даними може вплинути на поведінку акторів всіх рівнів в мережі блокчейну. SCWE-047 є прикладом вразливості з переповнення або не доповнення цілочисельних змінних, що своєю чергою може призвести до хибних розрахунків — це дуже поширена вразливість в елементах імплементування Оракулів у Web3.0 застосунок.

8. SCSVS-BLOCK — Blockchain and Runtime Environment — для безпечного виконання програм (смарт-контрактів) та злагодженої роботи Web3.0 застосунків потрібно підтримувати певний порядок у виконання функцій базового рівня блокчейн мережі — запроваджувати детерміноване, але при тому конкурентне середовище в якому всі нюанси (послідовність ланцюжка блоків — себто той самий блокчейн та газу і міток часу) будуть враховані та можливі як для використання смарт-контрактами, так і бути

обмеженням для запобігання зловживанню і роботі злочинних акторів в мережі.

Прикладом таких вразливостей є SCWE-024 — Слабкі джерела випадковості — можлива маніпуляція ланцюжком блоків зі сторони адміністраторів нод. SCWE-031 — Незахищені часові мітки — на цей показник також можуть впливати адміністратори нод і важливо цей параметр убезпечити, адже він є частиною безпекового каркаса всієї мережі блокчейну.

9. SCSVS-BRIDGE — Cross-chain Bridges — для взаємодії між різними мережами блокчейну використовуються так звані "Bridges" — мости. Історично це один з найбільш атакованих елементів інфраструктури блокчейн системи. Міст є вартісною та складною програмою, вразливості на ній можуть включати блокування та втрату активів/повідомлень та змову адміністраторів нод чи порушення послідовності стану.

SCWE-028: Oracle Price Manipulation — як було висвітлено раніше, Оракули це важлива частина блокчейн системи та у випадку коли при трансфері активів Міст покладається на дані лише з одного джерела (одного Оракула) — трапляється можливість стороннього впливу на транзакцію та фінальну ціну активів після завершення переміщення в іншу мережу блокчейн.

SCWE-033: Chain Split Risks — проблема суто Мосту, яка полягає в розриві послідовності блоків при трансфері даних між мережами блокчейну. Своєю чергою це може створити невідповідність і дисбаланс в обох мережах.

SCWE-034: Insecure Cross-Chain Messaging — ця вразливість є прикладом відсутності послідовної верифікації або шифрування повідомлень під час переміщення між мережами.

10. SCSVS-DEFI — Decentralized Finance — протоколи роботи децентралізованих фінансів мають своєрідну й унікальну логіку та фінансові інструменти — тому і вразливості які впливають на DeFi системи мають

виразну специфіку. Маніпулювання цінами, зловживання кредитними інструментами (плече та пулл і так далі), а також недосконала логіка ліквідації позицій — все це є вразливостями притаманними конкретно для DeFi протоколів.

11. SCSVS-COMP — Component and Third-party Risk — загальна категорія вразливостей пов’язаних з зовнішнім ПЗ та кодом, одна скомпрометована залежність чи імпортована бібліотека може стати передумовою для роботи та успіху атак націлених на всю систему.

Прикладами таких вразливостей можуть бути дуже загальні випадки як SCWE-083 — проблематична обробка граничних випадків, себто не гнучкість застосунку, а також SCWE-057 — погана організація пам’яті в процесах виконання функції, яка в такому випадку може призвести до перезапису важливої інформації.

Враховуючи цю категоризацію надалі можливо розробити більш детальну матрицю векторів, як буде вказувати на підґрунті контексту — яка є ступінь вразливости оперуючи як загальними описами вразливостей, так і розумінням наближеної критичності вразливостей в окремих структурних елементах системи блокчейну та інфраструктури що забезпечує підтримку мереж.

Через те, що ШІ агенти, нейромережі та LLM не сприймають текст так само як людська свідомість — важливо окреслити межі та розмітити орієнтири для вибудови логічних зв’язків та трансформації людської мови, тексту написаного людьми в дані зрозумілі ШІ агенту. З цим питанням я звернувся до моїх колег, аби разом краще визначити пріоритетність і загальні орієнтири — що в нашому досвіді більш вірогідно свідчить про потенційну валідну вразливість.

2.4 Результати брейншторм сесії команди HackenProof.

Разом з моїми колегами ми мали на меті зробити огляд процесу як ми особисто скануємо повідомлення о потенційній помилці при першому перегляді. Важливими аспектами є чіткість формулювань — навіть переглядаючи текст по діагоналі, можна одразу зрозуміти чи автор репорту має загальне розуміння з теми про яку пише, чи це є недосвідчений whitehat хакер, або ж він намагається створити лише вигляд розуміння, аби маніпулювати нашим рішенням. Точність першого враження звісно що є результатом великого доробку і є прямо пропорційною до досвіду спеціаліста, тому це важко транспонувати в програмне забезпечення, але можливо навчити ШІ оцінювати чистоту написаного повідомлення, визначити слова "паразити" — маркери які свідчать про недосвідченість автора, або його спроби заплутати аналітичну команду. В цьому рівнянні важливо розуміти, що обидві сторони зацікавлені в знахідці й попередженню вразливості. Але мотивація авторів здебільшого є монетарна, на відміну від аналітичної команди, яка є вмотивована на покращення безпеки екосистеми в цілому.

Додатковою метрикою на яку ми вирішили робити акцент при цьому дослідженні — визначення CVSS/SCVSS оцінки — наскільки серйозною є заявлена вразливість. Ця цифра має досить статичні показники, як зміна чи оригінальність цілі огляду — чи репорт стосується заявленої цілі для пошуку вразливостей. Так і більш абстрактні метрики — складність атаки до прикладу. Тому неможливо спиратись лише на CVSS рахунок при оцінці пріоритетності повідомлення про вразливість. Третьою й останньою позицією оцінки пріоритетності репорту є порядковий номер користувача в рейтингу. Це число може змінюватись лише у випадку якщо користувач був зареєстрований менше, ніж 2 місяці тому — в цьому випадку рейтинг буде проігноровано в фінальній оцінці пріоритетності репорту. Тому базуючись на цих трьох параметрах: чіткість репорту, CVSS рахунок та рейтинг автора —

ШІ агент базує свою оцінку пріоритетності кожного з повідомлень та поширює цю інформацію з аналітичною командою, для пришвидшення цього процесу.

2.5 Моделювання концептуальної моделі ШІ агента.

Для створення ШІ агента — себто навчання та перевірки LLM, я вирішив запровадити такий процес: (блок схема процесу тренування)

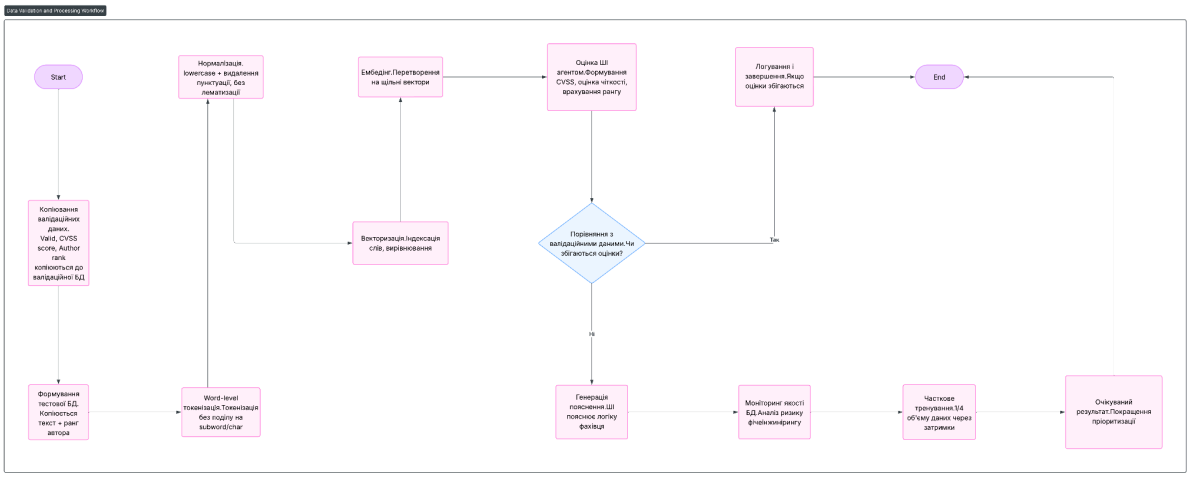

Рис. 2.5 — Флоучарт обробки та інформації та валідації результату

В першу чергу з наявної бази даних в нову (базу даних валідації) копіюються дані про результат оцінки повідомлення людиною-аналітиком: Valid: true/false; CVSS score: 0-10; Author rank: 1-...;

Наступним кроком основна частина повідомлення та ранг автора копіюється в другу базу даних (базу даних тестування), ця база даних проходить процес нормалізації, токенізації, векторизації та текстового ембедінгу. Опираючись на ці дані ШІ агент створює власну оцінку чіткості тексту репорту і можливий CVSS рахунок, та додає до цього ранг користувача який створив цей репорт.

Після оцінки — ШІ агент звіряє свої результати з даними з першої бази даних та при розбіжності має знайти аргументи чому саме таке рішення

прийняв спеціаліст при перевірці цього репорту. Це примірник моделі навчання для ШІ агента, найбільшим викликом цього процесу є потреба в моніторингу створення тестувальної бази даних, адже через потребу уникати завчання ШІ агентом конкретних стилів повідомлень (feature reverse-engineering) потрібно постійно перевіряти процесинг мови людини до матриці зрозумілої нейромережі, і сам процес перевірки з себе представляє просто використання більш загально орієнтованої моделі ШІ для оцінки якості обробки даних — це займає значні ресурси.

Затримки в процесі навчання ШІ (як я пояснив вище — збільшення часу потрібного на обробку даних) змусили мене проводити дослідження на моделях навчених лише на 1/4 від загального об’єму матеріалів. Я впевнений, що згодом, після повного опрацювання даних і тренування ШІ агента на них — точність результатів пріоритизації буде підвищена.

Через потребу обробити текст, але залиши його цілісним та зберегти можливість оцінити якість самого тексту переданого від автора — частина традиційних методів навчання для LLM не мала практичної користі для цього дослідження, але водночас зменшивши ніби обсяг роботи — це створило нові виклики. Як при цих вхідних даних та потребах до вихідних даних — якісно обробити інформацію, створити сприятливий контекст для навчання ШІ агента?

Першим кроком є токенізація слів, для оцінки якості тексту — збереження його первісної форми, на цьому етапі було обрано робити лише "Word-level" токенізацію, оминути узагальнення до "Subword-level", токенізація на рівні знаків взагалі не мала в цьому випадку сенсу.

Наступне рішення було застосування алгоритму для нормалізації тексту — переведення всіх знаків у нижній регістр та видалення знаків пунктуації для зменшення "шуму", на цьому кроці також ми оминули лематизацію —

повернення слів до їх початкових форм — за для можливості подальшої оцінки чіткості тексту.

Третім кроком цього процесу стала індексація слів за словником — перетворення їх на числові вектори, та вирівнювання всіх послідовних векторів.

І останній етап — ембедінг — перетворення попередніх векторів на "щільні" вектори — формати запису тексту який максимально ефективний для використання при навчанні LLM.

2.6 Експеримент: Перевірка точності ШІ агента після тренування.

В режимі тестування, ШІ агент після першого кроку навчання додали до нашої продакшн бази даних, яка оновлюється реальними репортами про можливі вразливості. Після отримання репорту в статусі "NEW" сервер автоматично надсилав дані до ШІ агента який був розташований на сервісі Hugging Face. Після обробки json файлу репорту — відповідь з оцінками від ШІ агента поверталась до нашого основного сервера (теж у форматі json) та додавалась до відповідного запису в базі даних, додаючи значення в поле пріоритетності повідомлення. Якщо при перших тестах на оброблених репортах ШІ показував правильність на рівні 70%, після навчання на ~5000 записах — в бета-тесті в умовах реальних повідомлень, точність рішень ШІ зросла на 26% до 96% правильної оцінки пріоритетності повідомлень о потенційних вразливостях. Це є задовільний показник, але як я зазначив вище — після навчання на більшому дата-сеті — я очікую підвищення цього показника до 99% відсотків.

Звісно що поки я не стикнувся з галюцинаціями ШІ агента, в першу чергу через суворі системні команди, які завдають чіткий формат відповіді очікуваний від ШІ.

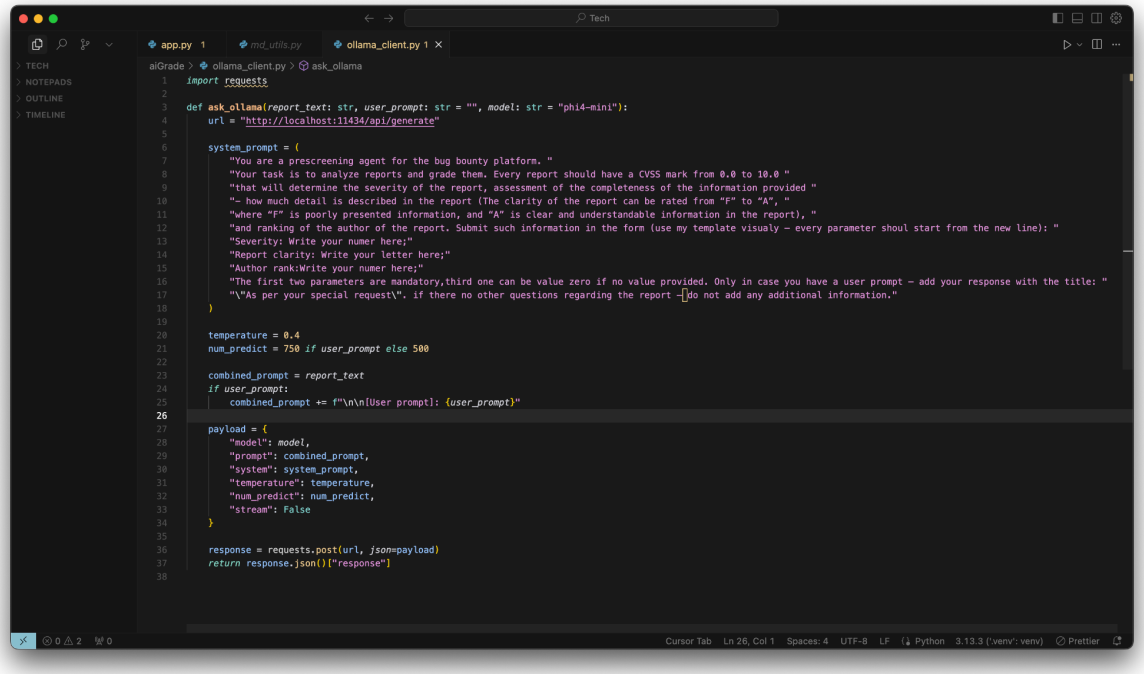

Рис. 2.6 — Одна з версій system_promp для модерації відповідей ШІ агента

2.7 Технології використані в процесі дослідження.

Для реалізації процесу навчання ШІ агента та створення середовища в якому можна буде протестувати гіпотезу я використовував різноманітні технології, як специфічні для роботи з нейромережами, так і звичні фреймворки для створення веб-інтерфейсів. За основу свого дослідження я взяв моделі Microsoft Phi-4 та Phi-4-mini — це оптимізовані мовні моделі для глибокого текстового аналізу, які мають великий потенціал у сфері розробки та читання коду. За час використання різних сторонніх ШІ агентів, наприклад в IDE Cursor я помітив що результати від моделі Phi-4 зазвичай є точніші та швидші в питаннях аналізу коду і генерації пропозицій щодо його покращення. Phhi-4-mini дуже легка модель LLM яка може бути розгорнута локально на одному комп’ютері, без потреби в сервері чи спеціалізованому обладнані, що також є перевагою на етапі тестування гіпотези.

Для обробки даних і створення навчальних баз даних я використовував такі інструменти як NLTK — для базового аналізу тексту, токенізації, нормалізації й подібних процесів обробки тексту написаного людиною та перетворення його в вектори зрозумілі ШІ моделі. Також я застосовував spaCy для швидкісного аналізу тексту та вилучення ключової інформації та створення навчальної матриці — себто які специфічні терміни треба означити для ШІ агента як ключові, або додати в словник з чітким визначенням.

Основною мовою програмування яка зараз застосовується в розробці ШІ та роботі з аналогічними проєктами — це Python, багато допоміжних міні-програм та скриптів я писав саме на Python. Простий синтаксис та гнучкість мови Python робить її дуже ефективним інструментом в процесі прототипування, але вже на моменті роботи з великими масивами даних я помітив що швидкість виконання програм є незадовільною, тому на майбутнє потрібно пройти процес інкапсуляції та виокремити всі можливі частини коду в зовнішні бібліотеки та оптимізувати їх переписавши на C/C++.

Також для створення веб-інтерфейсу для MVP я використовував фреймворк Streamlit — розроблений для швидкого прототипування без залучення CSS/HTML цей Python фреймворк допоміг заощадити час на розробці UI та сконцентруватись на роботі з ШІ.

Для взаємодії з ШІ я використовував Ollama — для запуску мовних моделей локально, та Hugging Face — для взаємодії зі створеним ШІ агентом через Rest API.

Всі дані мого дослідження я зберігав в MongoDB, я вже мав досвід роботи з базами даних в цьому форматі — ми використовуємо саме MongoDB на робочих серверах і тому для уникнення потреби в міграції я також застосовував саме цей варіант системи керування базою даних.

РОЗДІЛ ТРЕТІЙ:

Реалізація та результати

3.1 Реалізація.

Результатом практичної частини цього дослідження є три розробки: формат обробки даних для навчання та саме навчання ШІ агента, MVP — веб-інтерфейс який дозволяє взаємодіяти з ШІ агентом напряму, та інтеграція створеного ШІ агента в наявну платформу HackenProof. На шляху досягнення мети досліду були як перепони, так і неочікувані варіанти для оптимізації чи спрощення процесів. Розроблений програмний код цих рішень був написаний мною, але також проходив перевірку спеціалістів з розробки команди HackenProof. Для інтеграції будь-якого нового функціоналу у масивну платформу потрібне не лише готове програмне забезпечення у форматі який буде сумісним, але також велика кількість координації з всіма розробниками та тестувальниками, аби пересвідчитись що нова функція буде працювати коректно та не вплине на інші робочі процеси платформи. Додатково, HackenProof за архітектурою є платформою типу моноліт, але зараз проходить процес переналаштування для роботи на мікросервісах. Через ці обставини — як імплементація ШІ агенту, так і загальна координація команди зайняла більше часу, ніж я розраховував. Чітка процедура перевірок і тестування кожної зміни на платформі є запорукою безпеки користувачів та клієнтів, а дотримання всіх необхідних процесів допомагає робити обслуговування платформи дещо спокійнішим, і не зважаючи на ці всі складнощі фінальна мета дослідження була досягнута.

Реалізація навчання ШІ складається з двох послідовних кроків — обробка інформації з людської мови до формату подання для ШІ та процес навчання, тестування і верифікації інформації. Структура обробки даних і навчання ШІ не має за собою жодного інноваційного контексту — я використовував сучасні та актуальні інструменти та методики для обробки

унікального масиву даних зібраних нашою командою за роки роботи в індустрії. Все починається з компіляції необхідних полів з бази даних в нові бази — для навчання, та тестування. Навчальна база даних проходить подальші трансформації:

- Токенізація (Tokenization) — розбиття всього тексту на менші частини. Від слів до окремих символів. В ході дослідження було визначено що має сенс проводити токенізацію до рівня слів (Word-level), для збереження загального контексту та автентичності тексту, що впливає на показник якості репорту, який є ключовою метрикою в нашому випадку.
- Очищення тексту (Cleaning / Normalization) — на цьому кроці всі знаки проходять процес перетворення на нижній регістр та видаляються знаки пунктуації, але оминаємо лематизацію та стемінг — ці процеси повертають слова до їх первісної форми, це гарна практика для оптимізації даних і пришвидшення навчання, але дослідження оцінювання репортів виявило погіршення результатів після застосування цих правил обробки даних. Для формування контекстного сприйняття в ШІ агента, важливо перетворити на навчальну програму текст зі збереженням максимуму оригінального повідомлення.
- Перетворення токенів у індекси (Encoding / Token IDs) — завдяки розповсюдженню ШІ, майже всі слова та їх форми мають сталі значення у цифровому форматі, а за потреби можуть бути обчислені. На цьому кроці токени перетворюються на цифрові значення — вектори у першій інстанції. На цьому етапі в даних зникає будь-яка інформація сприйнятлива для людини, тепер ці дані можуть інтерпретувати лише ШІ або великі мовні моделі.
- Вирівнювання (Padding & Truncation) — дані в цій інстанції вже представлені у вигляді числових послідовностей, для автоматизації їх обробки — вони вирівнюються. Padding відповідає за додатне

вирівнювання, фактично дописати необхідну кількість нульових значень до послідовності — це не впливає на передачу та зчитування інформації, але допомагає в організації обробки цієї інформації.

- Ембеддінг (Embedding) — фінальний крок перед початком навчання ШІ на свіжих даних. Цей процес перетворює попередньо утворені числові (векторні) послідовності в щільні вектори та в цьому форматі дані використовуються вже для неопосередкованого навчання ШІ.

Процес організовано з можливістю моніторингу кожного кроку обробки даних людиною. Поки дані можуть сприйматись як текст їх легко можна перевірити, але це не ефективний спосіб який додатково не пропонує рішення для перевірки числових та векторних даних. Додатково при збільшенні кількості даних що проходять через підготовчі процеси — важко продовжувати ретельне слідкування за якістю обробки даних. Основним викликом в цьому процесі були вхідні дані — якщо в повідомленні використані хибні терміни, або ж велика кількість граматичних помилок, такі дані потрібно або виправляти вручну, або ж виключати з навчального пайплайну, адже за своєю системною директивою ШІ агент завжди намагатиметься назначити найвищий з можливих рейтингів пріоритетності. Якість даних для навчання ШІ завжди має вирішальну роль, адже це дозволяє на меншій вибірці отримувати кращі результати, це оптимальний сценарій для досягнення мети дослідження. Тож протягом перший кроків створення системи для роботи з даними та навчанням ШІ — було зроблено багато корисних відкриттів стосовно тонкощів роботи з великими мовними моделями та сам процес пройшов декілька стадій розвитку. Обробка одного репорту про можливу вразливість у ранньому варіанті процесу навчання займала до 1 хв, залежно від об’єму тексту та складності термінології. Ми опрацьовували репорти в чергах по 100 задля оптимізації процесу перегляду результатів обробки тексту — для моніторингу вже оброблені дані

завантажувались в велику модель мовного процесингу — Chat-GPT і якщо результат аналізу LLM збігався з вхідними даними — обробка вважається успішною. Згодом ми залучили вже наявні рішення з платформи Hugging Face для оптимізації перевірки та моніторингу даних, а також аутсорсингу процесорного навантаження, бо вся ця робота не можливою для звичайних користувацьких комп'ютерів.

При проблематичній обробці, дані вертались в стадію нормалізації та мануально коригувались, або ж якщо примірник був максимально невдалий — видалялись.

Великою частиною підготовки до цього процесу була анонімізація даних. Через законодавчі нормативи та контрактні зобов'язання — потрібно було в навчальних даних замінити всі персональні дані та дані компаній стосовно якої ми отримали репорт на загальні, або ж умовні значення. Замінити конкретні посилання на домени заглушками по типу example.com і тому подібне. Для цих цілей я використовував різноманітні інструменти, з самого початку — стандартні функції "знайти та замінити" в текстовому редакторі Sublime text, згодом за допомогою Shell-скрипта і фінально аби пришвидшити цей процес — була розроблена міні програма для багато параметральної анонімізації репортів і додатково я оптимізував вивід даних з первісної бази даних, аби отримувати менше приватної інформації про автора повідомлення та компанію якої це повідомлення стосується.

На етапі обробки 10% означеного об'єму даних, я створив MVP версію — простий веб-застосунок з візуальним інтерфейсом який дозволяв завантажити репорти в форматі MD файлів та запитати в ШІ агента додаткову інформацію стосовно повідомлення про потенційну вразливість!

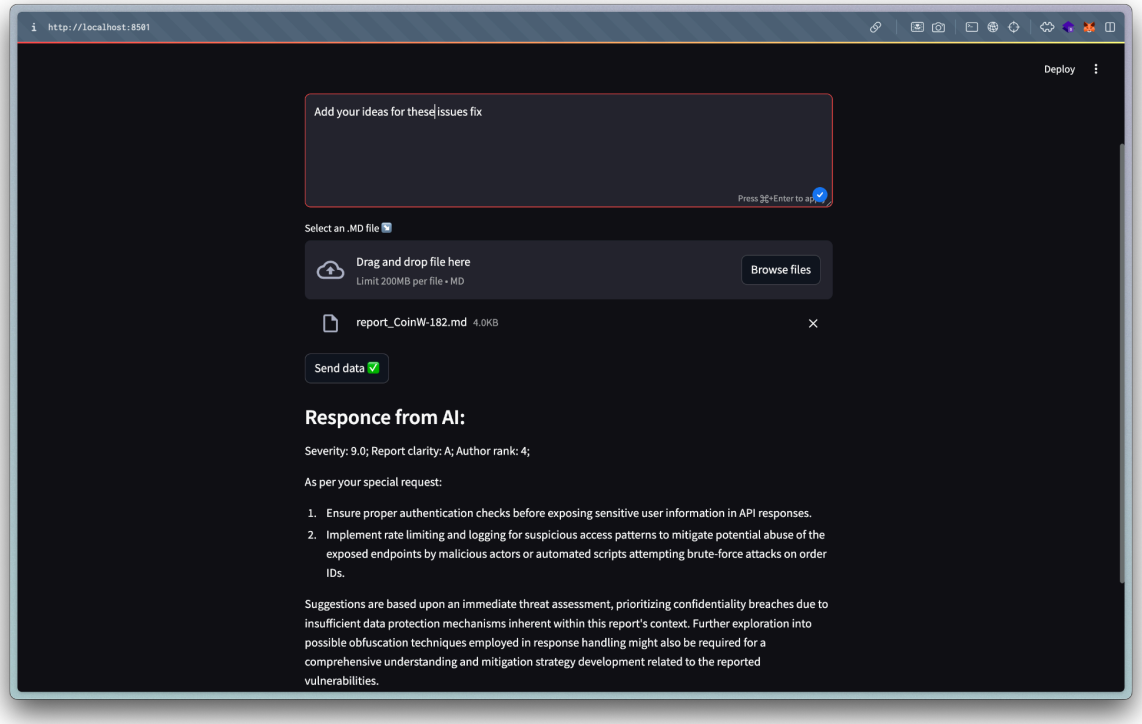

Рис. 3.1 — UI MVP версії та приклад роботи натренованого ШІ агента

Ця версія була використана не лише мною, але для розширення області дослідження і покращення зворотних даних про роботу ШІ я поділився цією версією зі своїми колегами та разом ми дійшли висновку, що немає дійсної потреби ставити одразу запитання до ШІ агента і цей функціонал на потрібно переносити на платформу. Адже цей ШІ агент має на меті в першу чергу бути інтегрованим в платформу HackenProof для створення конкурентної моделі пріоритетності повідомлень — які репорти потрібно переглянути в першу чергу. Теоретично лише 10% повідомлень які надходять до нашої команди є дійсно корисними даними про потенційні безпекові вразливості ПЗ, тому проблема яку вирішує цей ШІ агент це в першу чергу пріоритизація обробки даних — додаткові дані від ШІ можуть лише затримувати та плутати аналітичну команду в процесі роботи.

На етапі обробленні 25% від загальної кількості даних для навчання ШІ агента — у форматі тесту була проведена інтеграція ШІ в платформу HackenProof. Через відсутність пре детермінованих оцінок якості роботи важко сказати чи ШІ агент вплинув на цей показник, але точно пришвидшився процес обробки інформації командою. Адже більш пріоритетні репорти були оглянуті в першу чергу і репорти меншої пріоритетності не перехоплювали увагу аналітиків з безпеки. Додатково, оцінка пріоритетності виступила додатковим аргументом який допомагає в прийняті рішення стосовно валідності повідомлення — чи дійсно це є реальною проблемою безпеки, пошук подібних повідомлень завжди займав час і зараз знаючи що ШІ був навчений на нашій базі оброблених репортах — потреба в референсах зникла.

3.2 Результати.

Крім результатів практичних — побудови системи обробки інформації, створення MVP та інтеграції в режимі тестування в платформу HackenProof. Також були результати теоретичної значущості!

Я опанував роботу з багатьма технологіями які раніше не були мені знайомі — створення системи для парсингу інформації й обробки її в навчальні матеріали для ШІ, робота з ШІ агентом на рівні системного адміністрування, створення обмежень та робочих фреймворків для ШІ агента і звісно ж інтеграція свого проєкту у середовище великої платформи.

Аналізуючи результати виконання дослідження, чітко прослідковується потенціал ШІ у сфері запровадження безпекових рішень! Вже тестове використання ШІ агенту на нашій платформі довело свою ефективність та призвело до відчутного покращення організації роботи з лише незначною похибкою в 4% і зважаючи що всі репорти все одно переглядаються командою спеціалістів — така похибка є цілком прийнятно, та за умов

продовження навчання ШІ на якісних даних — можливо підвищити точність до максимуму!

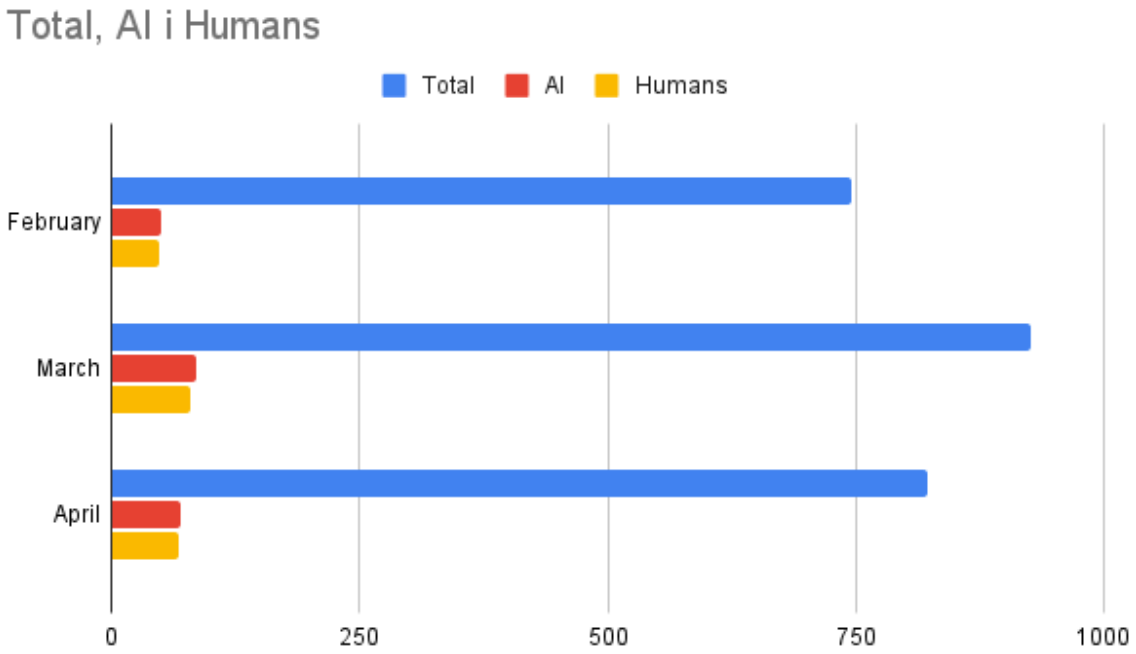

Рис. 3.2 — Загальна кількість репортів, ідентифіковані валідні репорти AI та працівників

Щодо результатів реалізації адаптивного ШІ агенту для оптимізації оцінки звітів вразливостей у додатках Web 3.0 — за час його застосування наша команда опрацювала більше ніж 200 репортів про дійсні вразливості з загальної кількості більше за 2500 повідомлено про потенційні вразливості. І це довело гіпотезу щодо використання ШІ агента для пришвидшення процесу обробки (а в першу чергу навіть ідентифікації) дійсно корисних репортів про вразливості.

Звісно не обійшлось без складностей вже в процесі імплементації ШІ агенту в робочі процеси команди. Для пришвидшення інтеграції ШІ в робочі процеси команда розробила рішення що дозволяло також передавати дані від ШІ агента в Jira робочий простір, для кращої організації перегляду

повідомлень з платформи. Додатково викликом стали репорти згенеровані ШІ, адже вони зазвичай є дуже логічно складені та відчутно довші за повідомлення написані реальним етичним хакером. Дослідження проблеми вказує що частина помилкових оцінок ШІ припадає саме не автентичні репорти. На жаль зараз немає чіткої пропозиції як можна працювати з цим та навчити ШІ розпізнавати саме ШІ, адже для цього потрібно використовувати неординарні мовні моделі та окремі дані й ресурси для навчання.

3.3 Аналіз.

Поєднання двох новітніх і стрімко розвиваних технологій — штучного інтелекту (ШІ) та Web 3.0 — дає унікальну можливість здійснити справжній прорив у сфері підвищення продуктивності, безпеки та ефективності обробки інформації. Застосування ШІ в середовищі Web3.0, особливо в контексті кібербезпеки, є не просто технічним експериментом, а логічним кроком еволюції цифрових систем. Досвід свідчить, що саме в тих ситуаціях, де присутній величезний обсяг рутинних завдань та обмежені часові ресурси, ШІ здатен виступити потужним інструментом оптимізації, не замінюючи людину, але значно покращуючи процеси прийняття рішень.

Працюючи з передовими інструментами й концепціями, ми часто стикаємося з явищем, яке в галузі розробки програмного забезпечення називають "overengineering" — надмірним ускладненням рішень, які насправді не потрібні або розв'язувати нереальні проблеми. Проте у випадку дослідження, описаного в цій роботі, мова йде про цілком виправдану, чітко сплановану і добре обґрунтовану інтеграцію ШІ агента в реальні робочі процеси команди аналітиків безпеки. Завдяки попередньому аналізу, тестуванню й перевіркам на основі реальних даних, було доведено, що створення системи для пріоритизації повідомлень про вразливості дало відчутний ефект: суттєво зменшився час на первинне сортування, а якість оцінки ймовірної серйозності вразливості зросла. Це є яскравим прикладом

ефективної взаємодії людини й машини, в якій ШІ не замінює спеціаліста, а підсилює його рішення і дозволяє зосередитися на більш складних аналітичних завданнях.

В той самий час, поява таких систем неминуче викликає питання — не лише технічні, а й етичні та юридичні. Одне з головних міркувань, що виникає при використанні ШІ, — це проблема джерела навчальних даних. Зокрема, наскільки морально та легально використовувати інформацію, створену людьми (наприклад, звіти white-hat хакерів), для навчання нейромереж? У контексті проєкту, реалізованого на платформі HackenProof, це питання було проаналізоване детально. З юридичного погляду, усі матеріали, що надходять від користувачів, передаються до інтелектуальної власності компанії відповідно до умов публічного договору користування платформою. Отже, використання цієї інформації з дотриманням політик конфіденційності є цілком правомірним.

З боку розгляду етичності цього рішення також важливо зазначити: розроблений ШІ агент не є повноцінним автономним учасником процесу, а виступає допоміжним інструментом на етапі первинного аналізу (triage). Він не приймає остаточних рішень, не впливає безпосередньо на розмір винагороди чи юридичні висновки — ці завдання залишаються за людським фахівцем. Така модель взаємодії мінімізує ризики некоректних рішень та одночасно гарантує прозорість у застосуванні результатів ШІ.

Крім того, розробка системи проходила із врахуванням принципів Responsible AI: враховано обмеження алгоритмів, дотримано чітке логування, а також забезпечено можливість людського втручання на кожному етапі. Саме така модель — Human-in-the-Loop — визнається в сучасній етиці ШІ найбільш безпечною і прийнятною для реального застосування.

Окремо варто звернути увагу на те, що ШІ агент, попри свою ефективність, має обмеження у контекстуальному розумінні, креативності та стратегічному аналізі. Він оперує на основі шаблонів, статистичних зв’язків і

ймовірнісних моделей, але не здатен розпізнати нові, неочевидні вектори атаки, які досвідчений аналітик може помітити інтуїтивно або на основі глибшого розуміння контексту й поведінки користувачів. Це чітко доводить, що роль ШІ — підтримка, а не заміна фахівців. У такому підході немає жодних компромісів із точки зору моралі — лише посилення безпеки завдяки технологіям, які допомагають команді реагувати швидше й точніше.

Таким чином, можна стверджувати, що у випадку створення ШІ агента для аналізу повідомлень про потенційні вразливості ми маємо справу з прикладом успішної етичної, правомірної та практично виправданої реалізації штучного інтелекту в складній галузі безпеки децентралізованих систем. Використання навчальної бази, створеної спільнотою, яка свідомо подає дані на платформу HackenProof, не порушує прав, не применшує значення людської роботи, а, навпаки, підносить її — даючи їй нове, ефективніше втілення.

ВИСНОВКИ

У результаті проведеного дослідження було успішно реалізовано комплексне технічне рішення, що складається з кількох взаємопов’язаних компонентів, кожен із яких виконує окрему, але критично важливу функцію. Передусім, була розроблена система обробки наявних звітів (репортів) про потенційні вразливості з метою їхньої трансформації у навчальний матеріал для ШІ агента. Цей компонент дозволив структурувати вхідні дані, провести нормалізацію тексту, анонімізацію чутливої інформації та сформувати датасет високої якості для подальшого тренування мовної моделі.

Наступним етапом стало створення MVP-версії — мінімально життєздатного продукту, який у даному контексті являє собою веб-інтерфейс, реалізований на основі Streamlit. Інтерфейс забезпечує зручний спосіб взаємодії людини зі штучним інтелектом, дає змогу надсилати репорти на обробку, переглядати

відповідь агента, а також аналізувати результат роботи в форматі, зручному для аналітиків з безпеки. Додатково було реалізовано інтеграцію із зовнішніми LLM-платформами через REST API (зокрема Hugging Face), що значно полегшує розгортання і тестування різних конфігурацій моделі.

Останнім блоком стало впровадження створеного ШІ агента у реальне середовище bug bounty-платформи HackenProof. Завдяки прямій інтеграції, модель була адаптована до робочих сценаріїв і змогла автоматично обробляти реальні вхідні повідомлення від white-hat хакерів, що надійшли у статусі "NEW". Це дає змогу автоматично визначати їхню чіткість, імовірну валідність, базовий CVSS-рейтинг, а також пріоритетність опрацювання. Система продемонструвала стійку працездатність, а також відповідність функціональним вимогам, сформульованим на початку дослідження. Її гнучка архітектура дозволяє легко масштабувати рішення, додавати нові модулі та перенавчати агента на оновлених датасетах, що є гарантією її довгострокової актуальності.

Ключовим технічним вибором стало використання мовної моделі Phi-4 від Microsoft як базової архітектури для створення агента. Ця модель продемонструвала високу ефективність у виконанні завдань із текстової класифікації, розуміння контексту і роботи з технічно-орієнтованими запитами. У ході розробки використовувалась модифікація Phi-4-mini для локального тестування, що дозволило зекономити обчислювальні ресурси та підвищити стабільність процесу навчання. У подальших ітераціях планується перехід на більш потужні варіанти — Phi-4-Reasoning або Phi-4-Reasoning-Plus, які, згідно з документацією, здатні краще опрацьовувати багаторівневу логіку, аргументацію та контекстні залежності, що критично важливо при оцінці повідомлень про складні логічні або архітектурні вразливості.

Застосування агента в умовах польового тестування — тобто в середовищі реальної роботи аналітиків HackenProof — показало помітний позитивний

вплив на швидкість розгляду репортів. У середньому, час первинної оцінки зменшився у 2–3 рази, а загальний рівень точності аналізу — згідно з внутрішніми порівняннями — досяг 96% збігів з думкою експертів. Це відповідає ключовій цілі дослідження: не створити повноцінну автоматизовану систему, а вбудувати інтелектуального асистента в існуючий робочий процес та підвищити ефективність команди без заміни людського фактора. Другорядною, але важливою метою було також підвищення якості аналізу — однак, через відсутність об’єктивних зовнішніх метрик оцінки якості (наприклад, точності аналітичного висновку), ця компонента потребує подальших розробок. У перспективі, за умови ширшого застосування та накопичення історичних результатів, стане можливим реалізувати відкладене порівняння — порівнювати передбачення ШІ з фінальними рішеннями щодо реального впливу вразливості на систему, що підвищить достовірність оцінки.

У цьому контексті слід окремо підкреслити, що мета проєкту не полягала в заміщенні аналітиків, а у створенні інструменту, що дозволяє автоматизувати рутинні етапи та вивільнити час спеціалістів для глибшого аналізу. В умовах сучасного IT-ринку, коли кількість інцидентів постійно зростає, а людські ресурси обмежені, такі підходи є не лише технічно доцільними, але й економічно вигідними. Крім того, це сприяє професійному розвитку фахівців, оскільки дозволяє їм зосереджуватись на складніших кейсах, а не витрачати час на тривіальні репорти низької якості.

Завдяки особистому досвіду роботи у сфері безпеки, тривалому залученню до практичних процесів компанії Hacken та прямій участі в проєкті зі створення ШІ агента, я переконався в тому, що мовні моделі — це потужний інструмент, який здатен трансформувати підхід до обробки інформації. Навички побудови, налаштування та адаптації таких систем у найближчому майбутньому стануть обов’язковими для кожного фахівця в галузі кібербезпеки. Успішна реалізація цього проєкту заклала міцний фундамент

для подальших досліджень і розвитку функціоналу, включно з глибшою інтеграцією в платформу HackenProof, додаванням аналітики якості звітів у динаміці, побудовою системи рекомендацій для аналітиків тощо. Робота над агентом триватиме — і я впевнений, що вона буде актуальною ще довгий час.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Huyen C. AI Engineering: Building Applications with Foundation Models. – 2025.
2. Raschka S. Build a Large Language Model (From Scratch). – 2024.
3. Alammar J., Grootendorst M. Hands-On Large Language Models: Language Understanding and Generation. – 2024.
4. Mastering Large Language Models: Advanced Techniques, Applications, Cutting-Edge Methods, and Top LLMs. – Harvard: Harvard University Press, 2024.
5. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
6. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
7. ОВ Бушма, АВ Турукало Evaluation of parameters in software implementation bar graph display devices - Cybersecurity: Education, Science, Technique, 2022,Vol 4 (16), p. 142-158

8. Alto V. Building LLM Powered Applications: Create Intelligent Apps and Agents with Large Language Models. – 2024.
9. Brousseau C., Sharp M. LLMs in Production: From Language Models to Successful Products. – 2025.
10. Kamath U., Keenan K., Somers G., Sorenson S. Large Language Models: A Deep Dive—Bridging Theory and Practice. – 2024.
11. Phoenix J., Taylor M. Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs. – 2024.
12. Iusztin P., Labonne M. LLM Engineer's Handbook: Master the Art of Engineering Large Language Models from Concept to Production. – 2024.
13. Wolfram S. What Is ChatGPT Doing...and Why Does It Work? – 2023.
14. Wolverhampton AI and cyber attack centre 'will be leading force' // BBC News. – 17.05.2025. – [Електронний ресурс]. – Режим доступу: https://www.bbc.com/news/articles/c4g2xrw5jl8o
15. AI-fueled cybercrime may outpace traditional defenses, Check Point report warns // Cybersecurity Dive. – 30.04.2025. – [Електронний ресурс]. – Режим доступу: https://www.cybersecuritydive.com/news/ai-security-cyber-crime-data-leak-check-point-report/746669/
16. Top 15 Real-Life Use Cases For AI In the Cybersecurity Industry // Redress Compliance. – 04.03.2025. – [Електронний ресурс]. – Режим доступу: https://redresscompliance.com/top-15-real-life-use-cases-for-ai-in-the-cybersecurity-industry/
17. The cyber threats to watch in 2025, and other cybersecurity news to know this week // World Economic Forum. – 19.02.2025. – [Електронний ресурс]. – Режим доступу: https://www.weforum.org/stories/2025/02/biggest-cybersecurity-threats-2025/
18. 2025 to be a Year of Reckoning for AI in Cybersecurity // Infosecurity Magazine. – 30.12.2024. – [Електронний ресурс]. – Режим доступу: https://www.infosecurity-magazine.com/opinions/2025-reckoning-ai-cybersecurity/
19. Major AI Trends Redefining Cybersecurity in 2024 // Deimos. – 24.10.2024. – [Електронний ресурс]. – Режим доступу:

https://www.deimos.io/blog-posts/major-ai-trends-redefining-cybersecurity-in-2024
20. Artificial Intelligence in Cybersecurity // Infosecurity Magazine. – [Електронний ресурс]. – Режим доступу: https://www.infosecurity-magazine.com/artificial-intelligence/
21. AI in Cybersecurity: How It's Used + 8 Latest Developments // Secureframe. – 24.02.2025. – [Електронний ресурс]. – Режим доступу: https://secureframe.com/blog/ai-in-cybersecurity
22. SylphAI-Inc/LLM-engineer-handbook // GitHub. – [Електронний ресурс]. – Режим доступу: https://github.com/SylphAI-Inc/LLM-engineer-handbook
23. Mastering Large Language Models: A Deep Dive into Sebastian Raschka's Build a Large Language Model // Medium. – [Електронний ресурс]. – Режим доступу: https://medium.com/@vimalkansal/mastering-large-language-model-471c2c20321e
24. Hands-On Large Language Models: A Comprehensive Review // Medium. – [Електронний ресурс]. – Режим доступу: https://medium.com/devreads/hands-on-large-language-models-review-2840c219902f
25. Mastering Large Language Models with Python (Paperback) // Harvard Bookstore. – [Електронний ресурс]. – Режим доступу: https://www.harvard.com/book/9788197081828
26. Building LLM Powered Applications // Amazon. – [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Building-LLM-Apps-Intelligent-Language/dp/1835462316
27. LLMs in Production // Manning Publications. – [Електронний ресурс]. – Режим доступу: https://www.manning.com/books/llms-in-production
28. Large Language Models: Bridging Theory and Practice // Amazon. – [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Large-Language-Models-Bridging-Practice/dp/3031656466
29. AI prompt engineering: learn how not to ask a chatbot a silly question // The Guardian. – [Електронний ресурс]. – Режим доступу: https://www.theguardian.com/technology/2023/jul/29/ai-prompt-engineering-chatbot-questions-art-writing-dalle-midjourney-chatgpt-bard

30. What Is ChatGPT Doing...and Why Does It Work? // Wolfram Media. – [Електронний ресурс]. – Режим доступу: https://www.wolfram-media.com/products/what-is-chatgpt-doing-and-why-does-it-work/
31. Q1-2025 Security Report // Hacken. – [Електронний ресурс]. – Режим доступу: https://wp.hacken.io/wp-content/uploads/2025/04/Q1-2025-Security-Report-1.pdf
32. Natural Language Toolkit (NLTK) – [Електронний ресурс]. – Режим доступу: https://www.nltk.org/
33. Phi-4 Model on Hugging Face – [Електронний ресурс]. – Режим доступу: https://huggingface.co/microsoft/phi-4
34. spaCy Usage Documentation – [Електронний ресурс]. – Режим доступу: https://spacy.io/usage
35. Streamlit Documentation – [Електронний ресурс]. – Режим доступу: https://docs.streamlit.io/
36. Ollama – Documentation – [Електронний ресурс]. – Режим доступу: https://github.com/ollama/ollama
37. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
38. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.

ОВ Бушма, АВ Турукало Evaluation of parameters in software implementation bar graph display devices - Cybersecurity: Education, Science, Technique, 2022,Vol 4 (16), p. 142-158

ДОДАТКИ

Додаток А

Повний код процесора ембедінга:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegressor
from sklearn.metrics import classification_report, mean_squared_error
from sentence_transformers import SentenceTransformer
import joblib
import faiss
import numpy as np
```

```
# === КОНФІГУРАЦІЯ ===
EMBEDDING_MODEL = "all-Phi-4-L6-v2"
CVSS_MODEL_PATH = "models/cvss_regressor.pkl"
VALID_MODEL_PATH = "models/valid_classifier.pkl"
VEC_DB_PATH = "vectors/index.faiss"

# === КРОК 1: ЗАВАНТАЖЕННЯ ДАНИХ ===
def load_data(path: str):
    df = pd.read_csv(path)
    return df

# === КРОК 2: ПЕРЕДОБРОБКА ТЕКСТУ ===
def merge_text_fields(df):
    df["merged_text"] = df["title"].fillna("") + " " + \
                        df["vector_expl"].fillna("") + " " + \
                        df["steps_to_reproduce"].fillna("") + " " + \
                        df["steps_to_fix"].fillna("")
    return df

# === КРОК 3: ВЕКТОРИЗАЦІЯ ===
def embed_texts(texts: list[str], model_name: str = EMBEDDING_MODEL):
    model = SentenceTransformer(model_name)
    embeddings = model.encode(texts, show_progress_bar=True)
    return embeddings

# === КРОК 4: НАВЧАННЯ МОДЕЛЕЙ ===
def train_models(embeddings, df):
    # CVSS (як регресія, припускаючи float 0-10)
    cvss_model = GradientBoostingRegressor()
```

```
    cvss_model.fit(embeddings, df["cvss_score"])

    # ВАЛІДНИЙ/ХИБНИЙ (бінарна класифікація)
    valid_model = RandomForestClassifier()
    valid_model.fit(embeddings, df["status"].map({"Valid": 1, "False": 0}))

    return cvss_model, valid_model

# === КРОК 5: ВАЛІДАЦІЯ ===
def evaluate_models(cvss_model, valid_model, embeddings, df):
    # Регресія
    cvss_preds = cvss_model.predict(embeddings)
        print("CVSS RMSE:", mean_squared_error(df["cvss_score"], cvss_preds,
squared=False))

    # Класифікація
    valid_preds = valid_model.predict(embeddings)
            print(classification_report(df["status"].map({"Valid": 1, "False": 0}),
valid_preds))

# === КРОК 6: ВЕКТОРНА БАЗА ДАНИХ ===
def save_vector_db(embeddings, ids):
    dim = embeddings.shape[1]
    index = faiss.IndexFlatL2(dim)
    index.add(embeddings)
    faiss.write_index(index, VEC_DB_PATH)
    np.save("vectors/ids.npy", ids)

# === КРОК 7: ЗБЕРЕЖЕННЯ МОДЕЛЕЙ ===
```

```
def save_models(cvss_model, valid_model):
    joblib.dump(cvss_model, CVSS_MODEL_PATH)
    joblib.dump(valid_model, VALID_MODEL_PATH)

# === ГОЛОВНИЙ ПРОЦЕС ===
def main():
    df = load_data("datasets/training_dataset.csv")
    df = merge_text_fields(df)

    # Розділення для валідації
    train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

    # Векторизація
    train_embeddings = embed_texts(train_df["merged_text"].tolist())
    test_embeddings = embed_texts(test_df["merged_text"].tolist())

    # Навчання моделей
    cvss_model, valid_model = train_models(train_embeddings, train_df)

    # Оцінка
    evaluate_models(cvss_model, valid_model, test_embeddings, test_df)

    # Збереження всього
    save_models(cvss_model, valid_model)
    save_vector_db(train_embeddings, train_df["id"].values)

if __name__ == "__main__":
    main()
```

Додаток Б

Повний код MVP версії продукту:

```
import streamlit as st
from md_utils import parse_md_file
from ollama_client import ask_ollama

st.set_page_config(page_title=" AIGrader")

# Сесійні змінні
if "md_text" not in st.session_state:
    st.session_state.md_text = ""
if "prompt" not in st.session_state:
    st.session_state.prompt = ""
if "response" not in st.session_state:
    st.session_state.response = ""
if "submitted" not in st.session_state:
```

```
    st.session_state.submitted = False

st.header("AI Reports Analyst", divider="gray")
st.subheader("Upload the .md file and enter the instructions for the AI")

# Інструкція
prompt = st.text_area("Instructions for the AI ⬇️", height=150, key="prompt")

# Завантаження файлу
uploaded_file = st.file_uploader("Select an .MD file ↘", type=["md"],
key="file")

# Кнопка подати дані
if st.button("Send data ✅"):
    if uploaded_file:
        st.session_state.md_text = parse_md_file(uploaded_file)
                        full_prompt = f"{prompt}\n\nText of the fileClick to
apply:\n{st.session_state.md_text}"
        st.session_state.response = ask_ollama(full_prompt)
        st.session_state.submitted = True
    else:
        st.warning("Please upload the file before submitting.")

# Відповідь
if st.session_state.submitted and st.session_state.response:
    st.subheader("Responce from AI:")
    st.write(st.session_state.response)
```

```
    # Кнопка копіювання (тільки коли є відповідь)
    st.download_button(
        label="Responce .MD 📄",
        data=st.session_state.response,
        file_name="ai_response.md",
        mime="text/markdown"
    )
```

Додаток В

Системна команда для ШІ агента:

```
import requests

def ask_ollama(report_text: str, user_prompt: str = "", model: str = "phi4-mini"):
    url = "http://localhost:11434/api/generate"

    system_prompt = (
        "You are a prescreening agent for the bug bounty platform. "
         "Your task is to analyze reports and grade them. Every report should have a CVSS mark from 0.0 to 10.0 "
        "that will determine the severity of the report, assessment of the completeness of the information provided "
         "- how much detail is described in the report (The clarity of the report can be rated from “F” to “A”, "
                "where “F” is poorly presented information, and “A” is clear and understandable information in the report), "
```

```
        "and ranking of the author of the report. Submit such information in the form (use my template visualy — every parameter shoul start from the new line): "
        "Severity: Write your numer here;"
        "Report clarity: Write your letter here;"
        "Author rank:Write your numer here;"
        "The first two parameters are mandatory,third one can be value zero if no value provided. Only in case you have a user prompt — add your response with the title: "
        "\"As per your special request\". if there no other questions regarding the report — do not add any additional information."
    )

    temperature = 0.4
    num_predict = 750 if user_prompt else 500

    combined_prompt = report_text
    if user_prompt:
        combined_prompt += f"\n\n[User prompt]: {user_prompt}"

    payload = {
        "model": model,
        "prompt": combined_prompt,
        "system": system_prompt,
        "temperature": temperature,
        "num_predict": num_predict,
        "stream": False
    }

    response = requests.post(url, json=payload)
```

```
return response.json()["response"]
```