Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Загребельного Владислава Вікторовича

на здобуття ступеня вищої освіти Бакалавра

Система виявлення шкідливого програмного забезпечення на кінцевих пристроях

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

КРБКБ.2102146.21.02.25 ПЗ

Виконав студент 4 курсу, група КБ-21-2 30.05.25 Владислав ЗАГРЕБЕЛЬНИЙ
Підпис, дата Ініціали, прізвище

Керівник к.т.н, доцент Юрій КЛЬОЦ
Науковий ступінь, вчене звання Підпис, дата Ініціали, прізвище

Нормоконтролер старший викладач 02.06.25 Сергій МОСТОВИЙ
Науковий ступінь, вчене звання Підпис, дата Ініціали, прізвище

До захисту допускаю:

Зав. кафедри кібербезпеки Юрій КЛЬОЦ
Підпис, дата Ініціали, прізвище

2. 06 2025р.

Хмельницький, 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Загребельного Владислава Вікторовича

1 Тема роботи Система виявлення шкідливого програмного забезпечення на кінцевих пристроях

Керівник роботи к.т.н, доц. кафедри кібербезпеки Юрій Павлович Кльоц

Затверджено наказом ректора університету від 7 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 2.06.2025

3 Вихідні дані до роботи Проаналізувати існуючі рішення виявлення шкідливого програмного забезпечення. Підбір оптимального стеку технологій. Вибір мови програмування. Збір датасету для навчання. Розробка back-end та front-end частин. Тестування та налагодження. Представлення результатів. Порівняння з вже існуючими рішеннями.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Теоретична частина. Прототип антивірусного рішення. Розрахунок ефективності розробленого прототипу. Висновки.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Схема алгоритму роботи антивірусу. Графічний інтерфейс. Таблиці.

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 16 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Січень-лютий	+
Ознайомлення з предметною областю	Лютий	+
Дослідження існуючих рішень	Лютий	+
Постановка задачі	Лютий	+
Визначення загальних принципів рішення задачі	Березень	+
Деталізація принципів рішення задачі	Квітень	+
Розробка проєктних рішень	Квітень	+
Апробація проєктних рішень	Квітень	+
Оформлення пояснювальної записки згідно вимог	Травень	+
Оформлення графічної частини	Травень	+

Студент _____ Владислав ЗАГРЕБЕЛЬНИЙ

Керівник кваліфікаційної роботи _____ Юрій КЛЬОЦ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система виявлення шкідливого програмного забезпечення на кінцевих пристроях».

Автор роботи: студент групи КБ-21-2 Загребельний Владислав Вікторович.

Керівник роботи: канд. тех. наук, доцент кафедри КБ Кльоц Юрій Павлович.

Пояснювальна записка: 82 с, 2 додатки, 6 таблиць, 3 формули, 24 рис, 40 джерел.

Графічна частина: плакати 3., 10 презентаційних слайдів.

Ключові слова: антивірус, вірус, шкідливе програмне забезпечення, загроза, машинне навчання, штучний інтелект.

Кваліфікаційна робота бакалавра присвячена розробці системи виявлення ШПЗ на основі нейронних мереж для ефективного виявлення шкідливого програмного забезпечення, зокрема атак «нульового дня».

В роботі проаналізовано сучасні методи виявлення шкідливого програмного забезпечення, виявлено обмеження традиційних антивірусних систем та досліджено можливості використання машинного навчання для підвищення ефективності захисту. В результаті розроблено та протестовано систему, що використовує нейромережеві мовні моделі для аналізу файлів. Експериментально доведено високу точність виявлення загроз, включаючи атаки «нульового дня», що перевищує показники існуючих нейромережевих рішень. Здійснено підготовку до впровадження розробленого антивірусного програмного забезпечення.

30.05.25

ABSTRACT

Subject of qualification work: End-device malware detection system.

Author: student of CS-21-2 Zahrebelnyi Vladyslav Viktorovich.

Head of work: Ph. D. tech. Sciences. Associate Professor of the Department of Cybersecurity Klots Yurii Pavlovych.

Explanatory note: p 82, appendices 2, figures 24, tables 6, formules 3, sources 40

Graphic part: posters 3, presentation slides 10.

Key words: antivirus, virus, malware, threat, machine learning, artificial intelligence.

This Bachelor's qualification work is dedicated to the development of a malware detection system based on neural networks for the effective detection of malicious software, including zero-day attacks.

The work analyzes modern methods of malware detection, identifies the limitations of traditional antivirus systems, and explores the possibilities of using machine learning to improve protection efficiency. As a result, a system was developed and tested that uses neural network language models to analyze files. The experimental results demonstrate high accuracy in detecting threats, including zero-day attacks, exceeding the performance of existing neural network solutions. Preparations have been made for the implementation of the developed antivirus software.

30.05.25

ЗМІСТ

ВСТУП

У сучасному світі, де цифрові технології проникають у всі сфери життя — від особистого спілкування до бізнесу, медицини, державного управління та оборонної сфери, питання захисту конфіденційної інформації набуває особливої актуальності. Витоки даних, їх пошкодження чи компрометація можуть призвести до серйозних наслідків як для окремих людей, так і для цілих організацій. Щоб захиститися від таких загроз, використовуються різноманітні інструменти: мережеві фаєрволи здійснюють контроль та фільтрацію трафіку, антивіруси шукають шкідливий код у файлах, а системи виявлення та запобігання вторгненням (IDS/IPS) аналізують мережеву активність та реагують на підозрілі дії.

Проте навіть найсучасніші інструменти не завжди в змозі протистояти складним кіберзагрозам, наприклад, таким як атаки «нульового дня». Цей тип атак використовує уразливості програмного забезпечення, які ще не були виявлені розробниками та не були виправлені оновленнями безпеки. В таких випадках традиційні методи захисту, що базуються на сигнатурному аналізі або поведінковому моніторингу, стають менш ефективними, оскільки просто не мають відповідних шаблонів для виявлення нових загроз. Це створює потребу у впровадженні більш інноваційних підходів до захисту інформаційних систем.

Кіберзагрози не лише стають складнішими, а й постійно змінюють свої методи, змушуючи експертів у сфері кібербезпеки вдосконалювати захисні системи. У зв’язку з цим одним із перспективних напрямків є використання штучного інтелекту та машинного навчання. Ці технології дають змогу аналізувати великі обсяги даних у реальному часі, виявляючи аномалії та підозрілі шаблони, що можуть свідчити про потенційну загрозу. Особливо перспективним є застосування нейронних мереж, які здатні навчатися на величезних масивах різноманітних даних, розпізнавати складні закономірності та шаблони, що залишаються недоступними для класичних методів виявлення шкідливого програмного забезпечення.

Головна ціль цього дослідження полягає в припущенні, що використання нейронних мереж для аналізу файлових систем дозволить підвищити ефективність виявлення раніше невідомих загроз у порівнянні з традиційними антивірусними

методами. Відповідно, мета цієї кваліфікаційної роботи полягає у створенні системи, яка з використанням нейронних мереж зможе ефективно аналізувати елементи файлової системи кінцевих пристроїв та виявляти шкідливі компоненти, навіть ті, що раніше не фіксувалися в базах даних загроз.

Реалізація потребує комплексного підходу, який передбачає глибокий теоретичний аналіз існуючих методів і рішень у сфері виявлення шкідливого програмного забезпечення, детальне вивчення алгоритмів, які використовуються в сучасних нейронних мережах, а також практичний етап розробки прототипу системи та проведення його тестування. Важливим аспектом є формування якісного датасету, який би охоплював широкий спектр шкідливих програм, а також забезпечення його актуальності відповідно до сучасних тенденцій розвитку кіберзагроз. Лише за таких умов можна гарантувати універсальність і гнучкість розробленої системи, здатної ефективно адаптуватися до нових викликів.

Розроблене рішення має продемонструвати не тільки високий рівень точності у виявленні вже відомих атак, але й показати свою здатність оперативно реагувати на нові, раніше невідомі загрози. Адаптивність є ключовою характеристикою ефективної системи безпеки в умовах постійного розвитку кіберзлочинності, адже зловмисники активно вдосконалюють свої методи, шукаючи нові способи обійти традиційні засоби захисту. Крім того, інтеграція розробленої системи з існуючими інструментами забезпечення безпеки дозволить підвищити загальну стійкість інформаційних систем, створюючи багаторівневий захист від атак.

Таким чином, результати цього дослідження є важливим внеском у розвиток сучасних методів кібербезпеки, що мають потенціал для практичного застосування як у комерційному, так і в державному секторах. Вони створюють основу для подальшого розвитку інтелектуальних технологій, спрямованих на протидію сучасним і майбутнім кіберзагрозам, забезпечуючи надійний захист даних в умовах цифрової епохи. Отримані напрацювання можуть стати базою для створення адаптивних систем безпеки з елементами машинного навчання. Це відкриває нові перспективи для підвищення ефективності реагування на інциденти та прогнозування потенційних атак.

1 ТЕОРЕТИЧНІ ЗАСАДИ ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз класифікації та характеристики шкідливого програмного забезпечення

Класифікація та точне виявлення шкідливого програмного забезпечення (ПЗ) мають важливе значення для розуміння природи загроз і створення ефективних захисних рішень, які здатні розпізнавати характерні ознаки різних сімейств такого ПЗ. Залежно від цілей своїх розробників, шкідливе програмне забезпечення можна поділити на кілька основних категорій: деякі з них призначені для шпигунства, інші — для отримання фінансової вигоди, порушення роботи систем, крадіжки даних або інших шкідливих дій. Використання технологій машинного навчання для аналізу подібного ПЗ передбачає розуміння цих категорій, адже це дозволяє класифікувати загрози більш точно, орієнтуючись на їхню поведінку [1].

Одним з поширених типів є віруси. Віруси - це програми, що виконуються без відома користувача і можуть самостійно поширюватися на інші файли та програми в системі. Виявлення таких загроз може ґрунтуватися на аналізі поведінки, наприклад, виявленні незвичної активації процесів або змін у системних файлах. Черв’яки, які можна розглядати як вдосконалені віруси, поширюються вже не лише всередині однієї системи, а й через мережі, заражаючи інші підключені пристрої. Їхня присутність часто проявляється у вигляді підозрілої мережевої активності або підвищеного навантаження на ресурси [2].

Троянські програми маскуються під легітимне програмне забезпечення, вводячи користувача в оману, щоб він самостійно завантажив шкідливий код. Такі загрози можна розпізнати за допомогою аналізу сигнатур або поведінкових моделей, що вказують на нетипові дії для звичайного програмного забезпечення [3].

Рекламне ПЗ, яке створює нав’язливі рекламні вікна або активно використовує мережеві ресурси. Сучасні методи машинного навчання дають змогу виявляти таку активність за характерними шаблонами [4].

Шпигунське ПЗ, виходячи з назви, виконує приховане стеження за діями користувача, збираючи конфіденційні дані й передаючи їх зловмисникам. Його

можна виявити шляхом аналізу поведінки програм, які читають великі обсяги даних або виконують дії у фоновому режимі. Ще однією серйозною загрозою є бекдори, вони створюють прихований канал доступу до системи для зловмисника. Попри те, що бекдори не завжди завдають безпосередньої шкоди, вони відкривають можливості для подальших атак. Їхнє виявлення можливе через моніторинг підозрілих вхідних з'єднань і неочевидних системних процесів [5].

Особливо небезпечним типом шкідливого програмного забезпечення (ШПЗ) є програми-вимагачі. Вони шифрують дані користувача й вимагають викуп за їх розшифрування. Важливо виявляти такі загрози на ранніх етапах, це можна зробити завдяки відстеженню аномальної активності у файловій системі, зокрема різкого збільшення операцій шифрування або змін у великій кількості файлів за короткий час [6].

Таким чином, розуміння різновидів шкідливого програмного забезпечення й правильна класифікація загроз є ключем до побудови сучасних захисних систем. Технології машинного навчання надають нові можливості для розпізнавання як відомих, так і нових видів атак, дозволяючи підвищити ефективність кіберзахисту та зменшити ризики для користувачів і організацій.

Віруси часто потрапляють на пристрій у вигляді вкладень до електронних листів, що містять шкідливий код або частину програми. Зараження відбувається після відкриття такого файлу користувачем, що може призвести до різноманітних негативних наслідків. Зокрема, віруси можуть змінювати або видаляти дані, красти конфіденційну інформацію, створювати бекдори для подальшого доступу або навіть повністю виводити з ладу систему. Оскільки сучасне шкідливе ПЗ здатне маскувати свою присутність, традиційні антивірусні рішення часто не справляються з його виявленням. У цьому контексті значну роль відіграє поведінковий аналіз та моделі прогнозування, здатні виявити аномальні дії навіть у разі невідомого коду. У результаті, поєднання класичних методів захисту з інтелектуальними алгоритмами аналізу дозволяє створювати більш стійкі та адаптивні системи кібербезпеки.

Найпоширеніші приклади ШПЗ за 2023 рік представлені на рисунку 1.1.

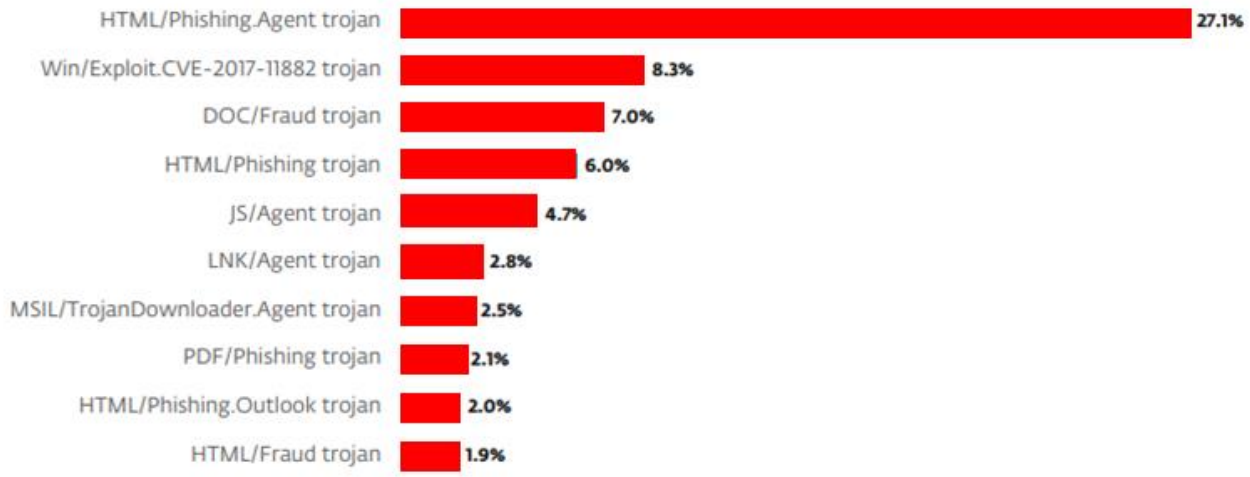


Рисунок 1.1 – Топ 10 видів ШПЗ за 2024 рік за версією McAfee [9]

З рисунку можна зробити висновок, що серед усіх видів шкідливого програмного забезпечення найбільше поширені трояни. Ознаками зараження можуть бути суттєве зниження продуктивності пристрою, переадресування на підозрілі веб-сторінки, хибні попередження про віруси з вимогою встановити підроблений антивірус, зашифровані файли без можливості доступу, а також часті спливаючі вікна з рекламою [7].

1.2 Методологічні основи функціонування антивірусних систем

Дія антивірусу базується на роботі низки технологій, які працюють за кадром і забезпечують його ефективність. Одна з найважливіших технологій, що використовується антивірусом, – це сканування файлів на наявність шкідливих об’єктів. Антивірусний движок аналізує код файлів та порівнює його зі списком відомих вірусів, що зберігається в базі даних. Окрім сигнатурного аналізу, сучасні антивіруси також застосовують евристичні методи та поведінкове виявлення, що дозволяє ідентифікувати потенційно небезпечні об’єкти навіть у разі відсутності їх у базі. У разі виявлення підозрілої активності система ізолює файл, блокує його виконання або переміщує до карантину для подальшого аналізу спеціалістами або хмарними сервісами безпеки. [8].

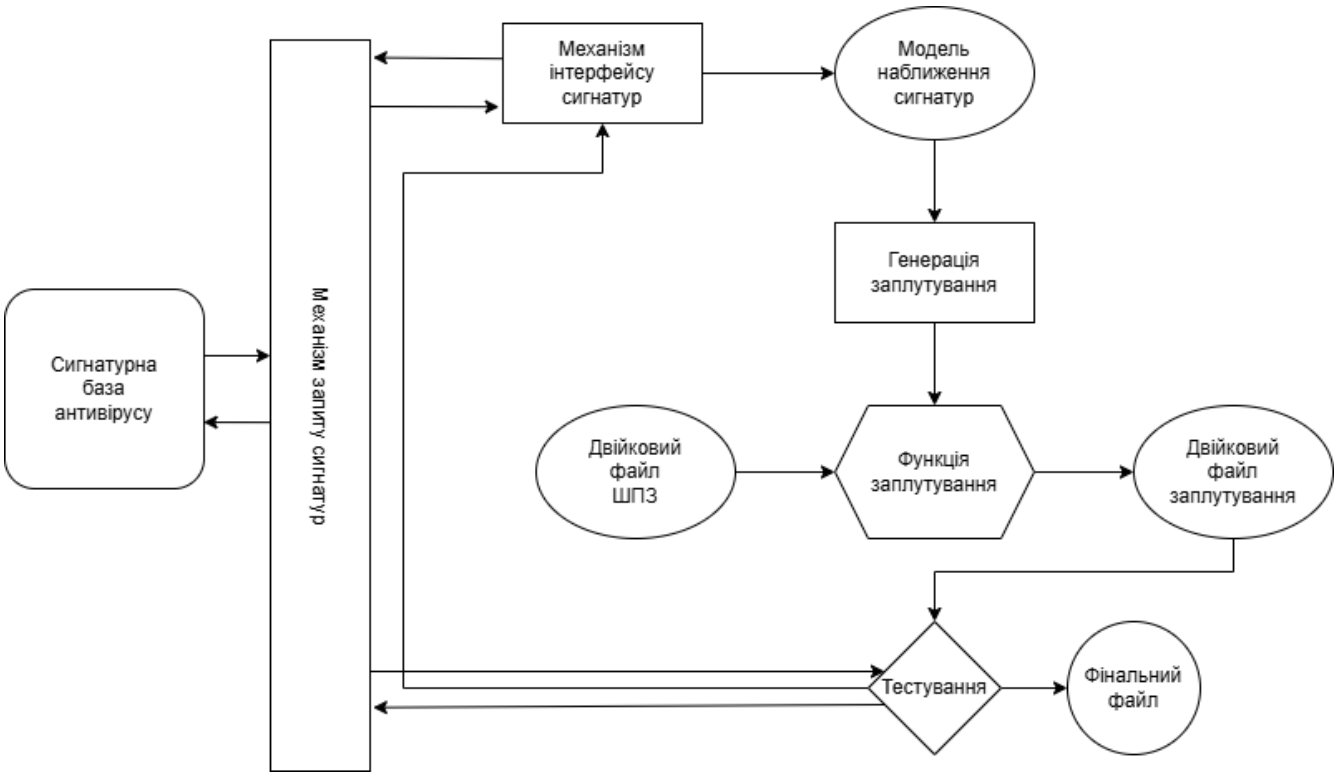


Рисунок 1.2 – Алгоритм роботи антивірусу

Також антивірус використовує евристичний аналіз, який дозволяє виявляти нові, раніше не відомі віруси. Він аналізує поведінку програм та виявляє підозрілі дії, які можуть свідчити про наявність шкідливого коду. Після виявлення потенційно небезпечної програми, антивірус блокує її запуск або надає користувачу вибір – дозволити або заборонити запуск програми.

Аналіз вихідного коду є важливим етапом в роботі антивірусів. Він дозволяє виявити потенційно шкідливі функції та забезпечити надійну захист від вірусів та інших шкідливих програм. Аналіз вихідного коду здійснюється за допомогою спеціальних технологій та механізмів, які перевіряють кожен кадр коду на наявність підозрілих дій. Антивірус аналізує дії, які виконує програма, та порівнює їх з відомими шаблонами шкідливого коду. Технологія аналізу вихідного коду використовується для виявлення різних типів загроз, таких як віруси, троянські програми, шпигунські програми та інші шкідливі програми. Механізми аналізу вихідного коду дозволяють забезпечити ефективну роботу антивірусного програмного забезпечення і забезпечити надійний захист комп’ютерної системи [10].

Аналіз вихідного коду відбувається у реальному часі, що дозволяє антивірусу реагувати на нові загрози миттєво. Завдяки цьому, антивірусна програма може швидко виявити та блокувати шкідливий код, що дозволяє запобігти поширенню вірусів та інших шкідливих програм.

Уважний аналіз вихідного коду є важливою складовою технології антивірусу. Він дозволяє забезпечити надійний захист комп'ютерної системи та зменшити ризик інфікування вірусами.

Евристичний аналіз антивірусу використовується для виявлення нових видів шкідливого коду, які ще не мають відомих сигнатур або інших ідентифікаційних ознак. Це дозволяє антивірусу виявити й зупинити нові шкідливі програми, які можуть бути небезпечними для системи користувача. Евристичний аналіз працює на основі аналізу поведінки програми або її коду. Антивірусний аналізатор використовує заздалегідь визначені правила та евристики, які дозволяють виявити підозрілі дії або фрагменти коду, що можуть вказувати на наявність шкідливої програми.

Наприклад, антивірус може виявити підозрілий кадр коду, який намагається змінити системні файли або виконати інші небезпечні дії. Це може бути ознакою наявності вірусу або іншої шкідливої програми. Антивірусний аналізатор може інтерпретувати цей підозрілий кадр коду, порівняти його з відомими шаблонами шкідливого коду та прийняти рішення щодо подальшої дії. Технологія евристичного аналізу дозволяє антивірусу бути більш ефективним у виявленні нових видів шкідливих програм та забезпечити захист системи користувача від небезпеки.

Механізм виявлення схожих вірусів є однією з основних технологій, яку використовує антивірус для своєї дії. Антивірус, за допомогою цієї технології, працює як пошукова система, яка аналізує вхідні файли та порівнює їх з відомими вірусами. Механізм виявлення схожих вірусів базується на порівнянні підписів вірусів, які раніше були відомі антивірусу. Кожен вірус має унікальний підпис, який складається з певних характеристик вірусу. Антивірус порівнює підписи з вхідними файлами і, якщо знаходить схожість, виявляє вірус.

Технологія виявлення схожих вірусів постійно вдосконалюється, оскільки

з’являються нові шкідливі програми та віруси. Антивіруси оновлюють свої бази даних, додаючи нові підписи вірусів, що дозволяє їм бути ефективними виявляти нові загрози.

Дія антивірусу за механізмом виявлення схожих вірусів полягає у скануванні файлів на наявність підозрілих характеристик. Антивірусний двигун, вбудований у програму, проводить аналіз файлу та знаходить схожість з відомими вірусами. Якщо виявляється схожість, антивірус сповіщає користувача про потенційну загрозу та пропонує видалити або помістити файл в карантин. Виявлення схожих вірусів є важливою складовою технології антивірусу, оскільки дозволяє ефективно виявляти загрози та захищати комп’ютери від шкідливих програм. Завдяки цьому механізму, антивіруси стають надійними інструментами для захисту від вірусів та інших загроз в Інтернеті [11].

Механізм, за допомогою якого антивірус працює, базується на технології створення сигнатур. Ця технологія є основною дією антивірусу. Сигнатура – це унікальна послідовність байтів, яка ідентифікує певний шкідливий код або вірус. Антивірус використовує ці сигнатури для пошуку і виявлення вірусів на комп’ютері користувача.

Створення сигнатур включає аналізування коду вірусу, його структури і особливостей. Антивірусний двигун аналізує кожен кадр вихідного коду і виявляє характерні ознаки, які вказують на наявність вірусу. За допомогою різних алгоритмів і евристичних методів, антивірусний двигун розпізнає вірусні сигнатури і створює базу даних з ними. Ця база даних оновлюється регулярно, щоб забезпечити ефективну роботу антивірусу [12].

Таким чином, за допомогою технології створення сигнатур антивірус здатний виявляти і нейтралізувати різноманітні віруси та інші шкідливі програми.

Одним із ключових механізмів дії антивірусу є використання бази даних сигнатур. Ця технологія дозволяє антивірусу виявляти шкідливі програми за їх унікальними характеристиками. База даних сигнатур – це спеціальний кадр, який містить інформацію про відомі шкідливі програми. Вона складається зі списку сигнатур, які представляють собою унікальні характеристики програм, такі як коди, ключові слова, маркери і т.д.

Коли антивірус сканує файл, він порівнює його з базою даних сигнатур. Якщо знайдено відповідність, то виконується певна дія, наприклад, файл блокується або видаляється. Такий механізм дозволяє антивірусу ефективно виявляти відомі шкідливі програми. Однак, він не здатен впоратися з невідомими вірусами або зміненими версіями відомих програм. Тому антивірусні компанії постійно оновлюють базу даних сигнатур, додавши в неї нові сигнатури.

Антивірус завантажує універсальні сигнатури з центрального сервера та перевіряє файлову систему користувача на наявність відповідних сигнатур. Якщо антивірус виявляє сигнатури, він розпізнає ці файли як потенційно небезпечні і вживає відповідних заходів для їхнього блокування або видалення. Універсальні сигнатури є дуже важливою складовою технології антивірусного захисту, оскільки вони дозволяють антивірусу ефективно боротися з новими видами шкідливих програм, які можуть з’явитися в мережі. Таким чином, антивірус забезпечує надійний захист комп’ютерної системи користувача від потенційних загроз.

Технологія антивірусу базується на скануванні вашого комп’ютера з метою виявлення потенційно небезпечних файлів. Цей процес складається з кадрів аналізу, де антивірусний двигун перевіряє кожен файл на наявність загроз. Якщо антивірус виявить підозрілий файл, він розміщує його в карантині, де він не може завдати шкоди вашій системі. Для виявлення вірусів антивірус використовує різні методи, такі як сигнатурний аналіз, аналіз поведінки та евристичний аналіз. Сигнатурний аналіз використовує базу даних, що містить відомі сигнатури вірусів. Антивірус порівнює сигнатури з файлами на вашому комп’ютері і виявляє, чи є вони відповідні.

Аналіз поведінки спостерігає за незвичайною діяльністю програми. Якщо програма здійснює підозрілі дії, антивірус може визначити її як шкідливу. Евристичний аналіз використовується для виявлення нових вірусів, які ще не мають відомих сигнатур. Після виявлення потенційно небезпечного файлу антивірус виконує дії для його блокування або видалення. Він може перекрити доступ до файлу, видалити його з системи або перемістити в карантин. Це залежить від налаштувань антивірусу та рівня загрози.

Таким чином, технологія антивірусу в дії робить ваш комп’ютер безпечним,

виявляючи та блокуючи шкідливі програми та віруси. Завдяки постійному оновленню бази даних антивірусу, ви можете бути впевнені, що ваша система завжди захищена від нових загроз.

Технологія антивірусів за лаштунками полягає у скануванні файлів, щоб виявити потенційно небезпечні програми або шкідливий код. Під час сканування, антивірус досліджує кожен кадр файлу, аналізуючи його структуру та вміст. Антивірус працює за принципом дії вірусу, він виявляє та аналізує підозрілі зміни в файлі, порівнює їх зі списком відомих вірусів або підписів і діє згідно з налаштованими правилами. Якщо антивірус виявляє потенційно шкідливий код, він може повідомити користувача, блокувати доступ до файлу або навіть видалити його з системи.

Сканування файлів є однією з основних функцій антивірусу. Під час сканування, антивірусний двигун аналізує кожен файл на наявність підозрілого коду або вірусів. Цей процес може включати виявлення вразливостей, перевірку цифрових підписів та порівняння з базою даних відомих загроз. За допомогою технології сканування файлів, антивіруси забезпечують захист комп’ютера від різних видів загроз, включаючи віруси, черв’яки, троянські програми та шпигунське програмне забезпечення. Ця технологія дозволяє виявити потенційно небезпечні файли, навіть якщо вони маскуються під надійний або нормальний вигляд.

1.3 Техніки та стратегії протидії шкідливому програмному забезпеченню

Існує доволі широкий перелік методів протидії зараженню системи ШПЗ, зокрема:

– регулярне оновлення системи та програмного забезпечення, яке дозволяє усувати вразливості, що можуть бути використані зловмисниками для впровадження та активації шкідливого коду;

– дотримання цифрової гігієни, наприклад, миттєве закриття підозрілих спливаючих вікон, ігнорування спам-листів та встановлення виключно відомих і

перевірених додатків;

– використання надійних паролів і двофакторної автентифікації, що значно підвищує рівень безпеки даних.

Одним із найефективніших способів захисту є встановлення спеціалізованого антивірусного рішення [13].

Дотримання чітких інструкцій та постійна обізнаність допоможе вберегти пристрої, мережу, файли, тощо. Також важливо регулярно створювати резервні копії даних, що дозволяє мінімізувати наслідки у разі успішної атаки. Особливу увагу слід приділяти навчанню користувачів основам кібербезпеки**.**

На рисунку 1.3 представлені найпопулярніші компанії-розробники антивірусних продуктів.

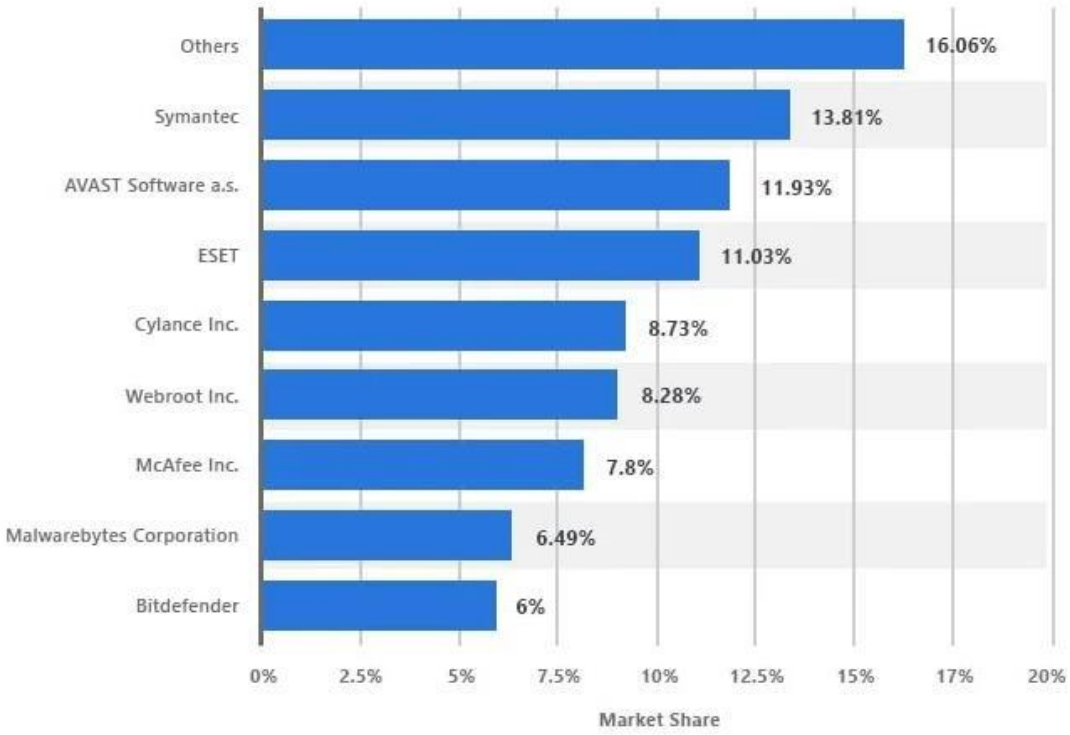


Рисунок 1.3 – Топ – 9 компаній які займаються розробкою антивірусів на ринку за 2024 рік за версією SafetyDetectives

Сучасні методи захисту від ШПЗ також включають активне використання хмарних технологій. Хмарні антивірусні платформи дозволяють централізовано зберігати оновлені бази даних та здійснювати аналіз підозрілих файлів у реальному

часі. Це забезпечує швидший відгук на нові загрози та мінімізує необхідність оновлення локальних антивірусних баз на пристроях користувачів. Крім того, хмарний підхід дозволяє антивірусним компаніям збирати інформацію про потенційно небезпечні файли від широкого кола користувачів, створюючи глобальну систему моніторингу загроз.

Однією з наявних технологій, яка також активно впроваджується, є пісочниці (sandboxing). Цей підхід полягає у створенні ізольованого середовища, у якому виконуються підозрілі файли. У пісочниці аналізується поведінка програм без ризику для основної операційної системи. Якщо файл демонструє ознаки шкідливої активності, його блокує антивірус, перш ніж він зможе завдати шкоди. Такий метод особливо ефективний для виявлення нового ШПЗ, яке використовує невідомі раніше методи обходу захисту [14].

Іншим важливим аспектом сучасних систем кібербезпеки є аналіз трафіку. ШПЗ часто взаємодіє з командно-контрольними серверами (C&C) для отримання інструкцій або передачі зібраних даних. Системи моніторингу мережевого трафіку дозволяють виявляти таку діяльність, аналізуючи аномалії у мережевих з'єднаннях. Наприклад, програми, що відправляють великі обсяги даних на невідомі сервери або здійснюють часті запити до нетипових IP-адрес, можуть бути ідентифіковані як загроза [15].

Крім того, важливим елементом сучасного захисту є використання штучного інтелекту (ШІ). Системи, що базуються на машинному навчанні, здатні самостійно адаптуватися до нових загроз, аналізуючи величезні обсяги даних. Завдяки цьому вони можуть передбачати ймовірність появи нових атак і розробляти методи їхнього запобігання ще до того, як ті стануть реальністю. Наприклад, алгоритми ШІ можуть аналізувати метадані файлів, їхню структуру та поведінкові характеристики, що дозволяє створити більш гнучкий і точний механізм захисту [16].

З огляду на стрімкий розвиток технологій кіберзлочинців, важливим фактором безпеки є не лише технічні засоби, але й людський фактор. Проведення тренінгів для співробітників компаній, розробка правил кібергігієни, а також формування культури відповідального користування цифровими пристроями є

головними для зниження ризиків. Переважна більшість атак, зокрема фішинг, спрямовані саме на людську неуважність або відсутність знань [17]. У майбутньому розвиток систем кіберзахисту, ймовірно, буде зосереджений на створенні інтегрованих платформ, що об'єднують методи аналізу трафіку, поведінковий аналіз, штучний інтелект і хмарні рішення. Лише поєднання різних підходів і технологій може забезпечити комплексний захист від сучасних кіберзагроз [18].

Крім зазначених методів, сучасні антивірусни все більше інтегрують концепцію багаторівневого захисту, яка охоплює не лише технічні, а й організаційні аспекти. Наприклад, важливим елементом є контроль доступу до системи, що включає використання багатофакторної автентифікації. Ця технологія забезпечує додатковий рівень захисту, вимагаючи від користувача підтвердження своєї особи за допомогою кількох різних методів, таких як паролі, SMS-коди або біометричні дані [19].

Додатково розробляються спеціалізовані системи для запобігання витоку даних (DLP). Вони дозволяють спостерігати й обмежувати передачу конфіденційної інформації, а також забезпечувати захист корпоративних даних навіть у разі компрометації кінцевих пристроїв. DLP-системи можуть аналізувати як зміст файлів, так і контекст, у якому вони передаються (наприклад, через електронну пошту, USB-накопичувачі або хмарні сервіси) [20].

Ще одним перспективним напрямком є розвиток технологій обману атакуючих – так звані пастки або honeypots. Це спеціально створені вразливі системи, які імітують роботу справжніх серверів або мереж. Їх завдання – привернути увагу кіберзлочинців, відволікаючи їх від реальних цілей, і паралельно збирати інформацію про їхні методи й інструменти. Такий підхід дозволяє не тільки захистити основні ресурси, але й вдосконалювати системи кіберзахисту, ґрунтуючись на отриманих даних [21].

Суттєвою інновацією є також інтеграція антивірусних систем із платформами аналізу великих даних (Big Data). Завдяки цьому стає можливим ефективніше відстежувати аномалії у поведінці користувачів і систем, аналізувати терабайти інформації з високою швидкістю та виявляти потенційні загрози на ранніх етапах. Такі системи, зокрема, застосовуються у великих корпораціях та урядових

структурах, де важлива здатність швидко реагувати на спроби проникнення в мережу [22].

Існує концепція Zero Trust, яка отримує все більше поширення у сфері кібербезпеки. Основний принцип Zero Trust полягає в тому, що жоден користувач або пристрій не може отримати доступ до ресурсів без підтвердження своєї автентичності й відповідності політикам безпеки. Ця модель дозволяє мінімізувати ризики внутрішніх і зовнішніх атак, оскільки доступ до даних надається виключно за необхідності й у рамках чітко визначених обмежень [23].

Також важливим аспектом є кіберзахист для пристроїв Інтернету речей (IoT). Кількість IoT-пристроїв, таких як розумні колонки, відеокамери чи термостати, стрімко зростає, але їхній рівень безпеки часто залишається низьким. Використання спеціалізованих рішень для захисту IoT є критично важливим, оскільки такі пристрої можуть стати точкою входу для атак, що ставлять під загрозу всю мережу [24].

Загалом, сучасні тенденції у сфері кібербезпеки спрямовані на проактивний підхід до захисту, коли головним завданням є не лише виявлення та усунення загроз, а й їхнє попередження. Це вимагає постійного вдосконалення технологій, взаємодії між антивірусними компаніями, урядовими організаціями та користувачами. Кіберзахист – це не лише про інструменти, а й про культуру безпеки, яка стає невід'ємною частиною сучасного цифрового суспільства [25].

1.4 Огляд сучасних наукових підходів та дослідницьких моделей для виявлення шкідливого програмного забезпечення

Згідно дослідженням Майка Нконголо та Махмута Токмака пропонує дослідження, у якому зв’язка моделей SAE (як енкодера) і LSTM (як класифікатора) демонструє досить високу точність передбачення (точність становить 98%), проте вона базується на класичному побудуванні баз даних ШПЗ, де розглядаються лише поверхневі ідентифікатори та малоінформативний поведінковий аналіз [26]. Нижче (рисунок 1.4) наведено датасет, використаний для навчання моделі авторами статті.

	Time	Protocol	Flag	Ransomware	Clusters	SeedAddress	ExpAddress	BTC	USD	Netflow_Bytes	IPaddress	Malware	Port	Prediction
0	40	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	504	8	A	Bonet	5061	SS
1	30	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	508	7	A	Bonet	5061	SS
2	20	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	512	15	A	Bonet	5061	SS
3	57	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	516	9	A	Bonet	5061	SS
4	41	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	520	17	A	Bonet	5061	SS
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
149037	33	UDP	AP	TowerWeb	3	1AEoiHYZ	1SYSTEMQ	1010	1590	3340	A	Scan	5062	A
149038	33	UDP	AP	TowerWeb	3	1AEoiHYZ	1SYSTEMQ	1014	1596	3351	A	Scan	5062	A
149039	33	UDP	AP	TowerWeb	3	1AEoiHYZ	1SYSTEMQ	1018	1602	3362	A	Scan	5062	A
149040	33	UDP	AP	TowerWeb	3	1AEoiHYZ	1SYSTEMQ	1022	1608	3373	A	Scan	5062	A
149041	33	UDP	AP	TowerWeb	3	1AEoiHYZ	1SYSTEMQ	1026	1614	3384	A	Scan	5062	A

149042 rows × 14 columns

Рисунок 1.4 – Фрагмент використаного датасету

Дослідження Тоні Керт'є та Грегуара Баррю розповідає про дещо глибший аналіз, у якому для кожного виконуваного шкідливого та безпечного файлу після їх запуску відбувається витягування даних, подальший збір, розділення та перетворення у формат чорно-білого зображення (метод grayscale).

Нижче (рисунки 1.5 та 1.6) наведено кінцеві результати конвертації вихідних даних.

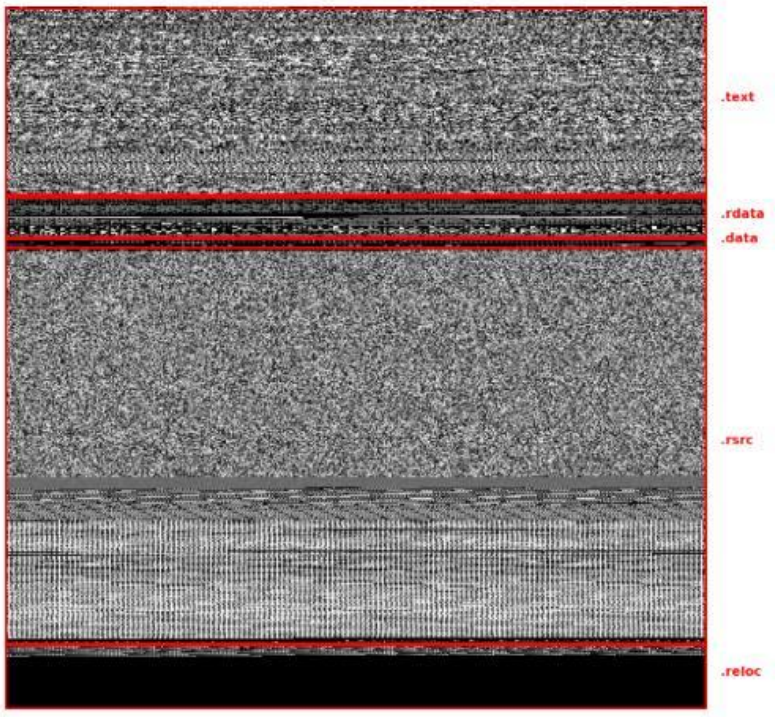


Рисунок 1.5 – Сегменти виконавчого файлу після конвертації [27]

Такий вид візуалізації може використовуватися для аналізу структури файлу, зокрема для виявлення шкідливих патернів (шаблонів) та інших аномалій, що порушують звичайний розподіл даних усередині файлу. Область із високим рівнем шуму свідчить про наявність стиснених або зашифрованих даних.

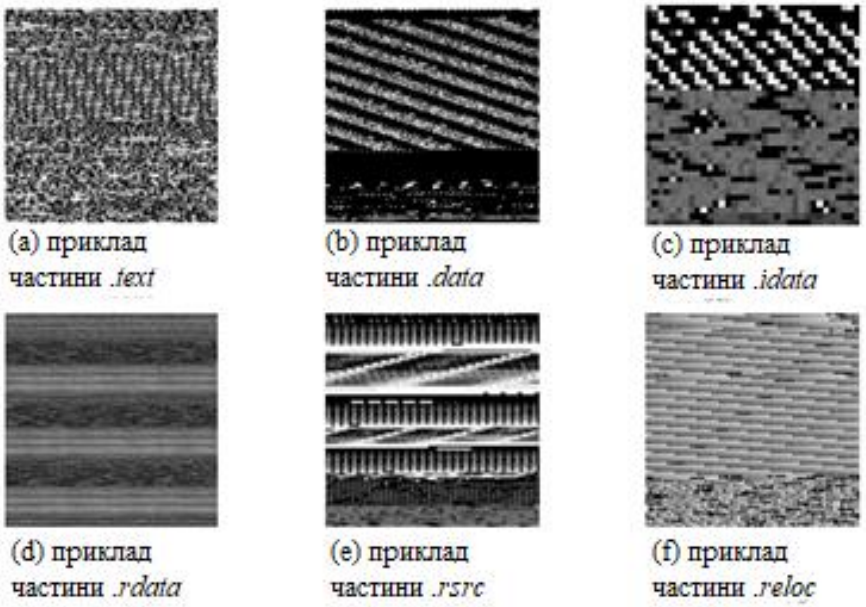


Рисунок 1.6 – Сепаратний аналіз кожної секції виконавчих файлів [28]

Такий підхід передбачає подальше навчання моделі згорткової нейронної мережі окремо для кожного розділу набору даних. Крім того, авторська розробка дозволяє аналізувати файли, виходячи з бінарного представлення, що дає змогу навчити модель для глибокого аналізу.

Однак ця розробка має також низку суттєвих недоліків, серед яких варто зазначити:

– наявність щонайменше шести різних класифікаторів (значне збільшення розміру кінцевої моделі та неминуча втрата контекстуального розуміння);

– необхідність роботи із зображеннями (велика кількість перетворень, які, зокрема, вимагають наявності графічного прискорювача у користувача для ефективної роботи моделі; на відміну від текстових даних – вкрай неочевидна інтерпретація виділених ознак).

Метрики моделі, навченої авторами статті, представлені в таблиці 1.1.

Таблиця 1.1 – Метрики розробленої моделі на базі CNN [27]

	Навчання	Дійсність	Тестування	
Частина	Точність	Точність	Точність	F_1-міра
Text	0.98	0.92	0.92	0.84
Data	0.98	0.92	0.93	0.87
Rdata	0.98	0.93	0.92	0.89
Rsrc	0.98	0.94	0.94	0.90
Reloc	0.98	0.93	0.94	0.80
iData	0.99	0.97	0.97	0.94
TLS	0.85	0.84	0.84	0.5

Ранжування підсумкових метрик у значеннях від 0.5 до 0.98 не гарантує безпечного функціонування моделі в реальних умовах, що може призвести або до втрати чутливих користувацьких і системних даних, або до хибнонегативного результату класифікації (шкідливий файл буде пропущено).

Дослідження Врінди Малхотри, Катерини Потіки та Марка Стампа показує що, життєздатним є метод застосування нейронних мереж, заснованих на графах (векторно-орієнтованих). Кодування вихідних даних у векторне представлення в багатьох випадках забезпечує усунення проблеми перенавчання моделі й, відповідно, вищі значення метрик. Проте такий підхід передбачає високий рівень абстракції від початкових даних. Першими, хто запропонував використання механізмів «self-attention» для навчання, були Дмитрійс Трізна, Лука Деметріо, Баттіста Біджіо та Фабіо Ролі, які у 2023 році опублікували відповідну статтю під назвою «Nebula: Self-Attention for Dynamic Malware Analysis» [29]. Автори звертають увагу на надмірну популярність і повторюваність досліджень моделей LSTM та CNN, і першими пропонують використання нейронних архітектур на основі трансформерів (transformer). Цей підхід відкрив нові перспективи в аналізі динамічної поведінки ШПЗ, забезпечуючи глибше розуміння взаємозв'язків між подіями в системі. Використання self-attention дозволяє моделі фокусуватись на ключових аспектах виконання шкідливого коду, що значно підвищує ефективність

виявлення навіть складних та замаскованих загроз.

Нижче наведені типи даних, використані авторами статті для навчання моделі (рисунок 1.7).

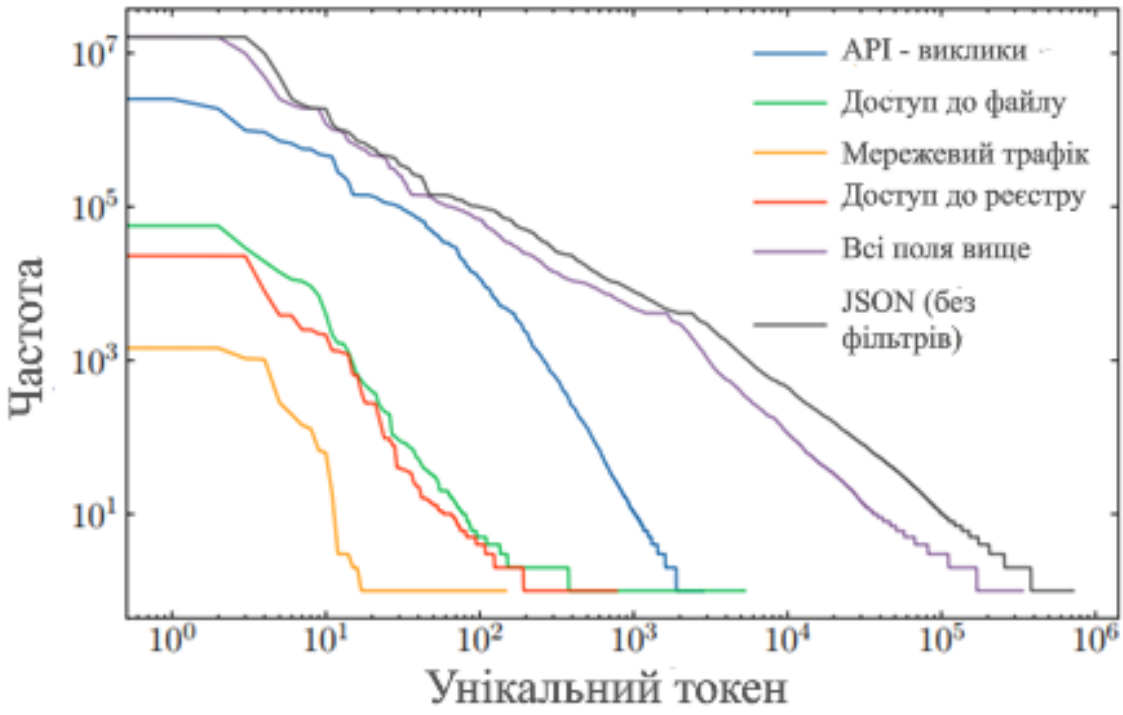


Рисунок 1.7 – Розподілення унікальних токенів

Дослідники використали лише поверхневі характеристики ШПЗ, торкнувшись незначної частини поведінкового аналізу, без глибшого структурного аналізу виконуваних файлів, тим самим не розкривши потенціал моделей сімейства трансформерів.

Загалом представлені дослідження та методи демонструють різноманіття та інноваційність нейромережевих підходів у сучасному аналізі й виявленні ШПЗ. Отримані висновки підкреслюють важливість індивідуального підходу до формування датасету та вибору програмного стеку для класифікації файлів на шкідливі та безпечні. Необхідність створення власних баз даних і персоналізованих аналітичних стеків стає критично важливою в умовах швидкої еволюції кіберзагроз та конкурентного наукового середовища, що динамічно розвивається.

У висновку дослідження варто відзначити переваги та перспективи використання методів глибокого навчання для аналізу та виявлення ШПЗ [30]. Глибоке навчання з використанням трансформерів та механізмів «self-attention»

дозволяє аналізувати великі обсяги даних, точно виявляти загрози та ефективно класифікувати ШПЗ завдяки фокусуванню на ключових аспектах даних.

1.5 Постановка задачі

Метою кваліфікаційної роботи є розробка прототипу антивірусу. А саме: «Системи виявлення шкідливого програмного забезпечення на кінцевих пристроях». Також в ході розробки буде використано ШІ та технології машинного навчання. Такий комплексний підхід дасть змогу проводити глибокий аналіз системи та її елементів на наявність зараження. Об’єктом дослідження є ШПЗ і методи його виявлення. Предметом дослідження є алгоритми машинного навчання, нейромережі та ШІ. Застосовувані для класифікації файлів на шкідливі та безпечні файли.

Завдання для досягнення цілі наступні:

– проаналізувати існуючі методи та визначити їхні недоліки;

– сформувати навчальний датасет, що включатиме загрози, забезпечуючи основу для тренування моделі;

– розробити та протестувати прототип системи;

– оцінити ефективність моделі за допомогою метрик точності, повноти та швидкості обробки, а також визначити можливі шляхи її вдосконалення.

Прототип розробленої системи виявлення ШПЗ, дозволяє ефективно виявляти як вже відомі так і не відомі загрози. Рішення демонструє не тільки високу точність у виявленні загроз, але й здібність швидко адаптуватися до нових типів атак та загроз, що робить прототип системи важливим внеском у сферу кібербезпеки. Також, під час реалізації було враховано питання безпеки взаємодії з файловою системою, а також інтегровано механізми підтвердження критичних дій користувача. Це не лише підвищує рівень захисту, але й дозволяє уникнути ненавмисних помилок під час використання системи.

2 РОЗРОБКА ПРОТОТИПУ АНТИВІРУСНОГО РІШЕННЯ

2.1 Вибір та обґрунтування технологічного стеку застосування мови Python

Python — це інтерпретована високорівнева мова програмування загального призначення, яка надає програмістам зручні та гнучкі можливості для створення різноманітних програмних продуктів, зокрема в галузях машинного навчання та штучного інтелекту. На відміну від мов, що потребують компіляції, Python виконує код послідовно, рядок за рядком, без попереднього перетворення у машинний код. Такий підхід дозволяє швидко налагоджувати програми та оперативно отримувати зворотний зв’язок під час розробки, що особливо корисно при побудові й тестуванні ML-моделей [31].

Завдяки простому та зрозумілому синтаксису, який наближений до англійської мови, Python спрощує процес програмування, дозволяючи зосередитися на розв’язанні задач, а не на деталях низькорівневої реалізації, таких як керування пам’яттю. Інтерпретатор Python забезпечує:

– гнучкість, можливість швидко перевіряти нові ідеї, що особливо важливо у сфері ML, де постійно змінюються налаштування й архітектури моделей;

– зручність налагодження, оскільки компіляція не потрібна, внесені зміни можна миттєво протестувати, скорочуючи час на виправлення помилок;

– кросплатформенність, Python працює на різних операційних системах без необхідності змінювати код, що є значною перевагою для командної розробки або роботи в різних середовищах [32].

Для різних перетворень вихідних файлів і щоб уникнути завдання шкоди персональному пристрою, було прийнято рішення скористатися програмним забезпеченням для створення віртуальних машин VirtualBox компанії Oracle. Для ітеративного підходу та закріплення результатів використовуються знімки станів віртуальної машини.

Вікно створених зрізів стану системи зображено нижче на рисунку 2.1.

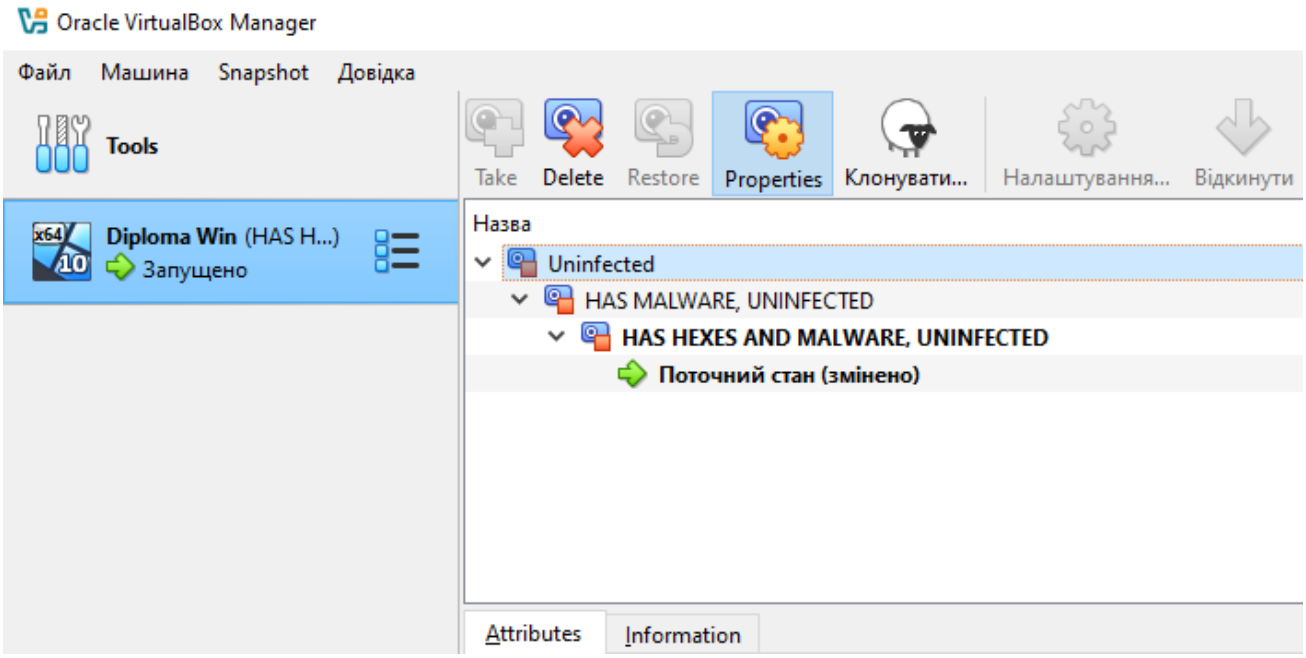


Рисунок 2.1 - Вікно створених зрізів стану системи

Крім ітеративного підходу, використання віртуальної машини з образом ОС Windows 10 та закріплення її станів через зрізи станів дозволяє аналізувати шкідливе ПЗ в контрольованому середовищі та проводити різні експерименти без ризику зараження основного робочого простору.

Після проведення огляду, найбільш перспективним напрямком для здійснення подальших досліджень є застосування механізмів «self-attention», зокрема – трансформерів. Їх особливість полягає у здатності вловлювати складні та далеко рознесені у вихідних даних поведінкові патерни, що виходить за рамки можливостей «традиційних» для наукових досліджень підходів LSTM та CNN.

Завдяки своїй здатності до паралельної обробки великих обсягів даних, трансформери ідеально підходять для роботи зі складними датасетами, характерними для кібербезпеки, та здатні швидко обробляти й аналізувати тисячі зразків ШПЗ, значно прискорюючи процес виявлення та реагування на загрози. Для проведення глибокого контекстуального аналізу кожного з виконуваних файлів (або інших файлів, що містять виконуваний код) було прийнято рішення конвертувати кожен об'єкт, що подається на вхід, у файл з шістнадцятковим (hexadecimal, скор. hex) представленням вихідного об'єкта, оскільки бінарний вигляд може зробити задачу навчання моделі невиправдано ресурсомісткою.

Отримання інших поверхневих даних про об'єкт недостатньо для навчання якісної моделі, щоб досягти поставленої в даній кваліфікаційній роботі мети.

Приклад шістнадцяткового представлення випадково відібраного виконуваного файлу зображено нижче на рисунку 2.2.

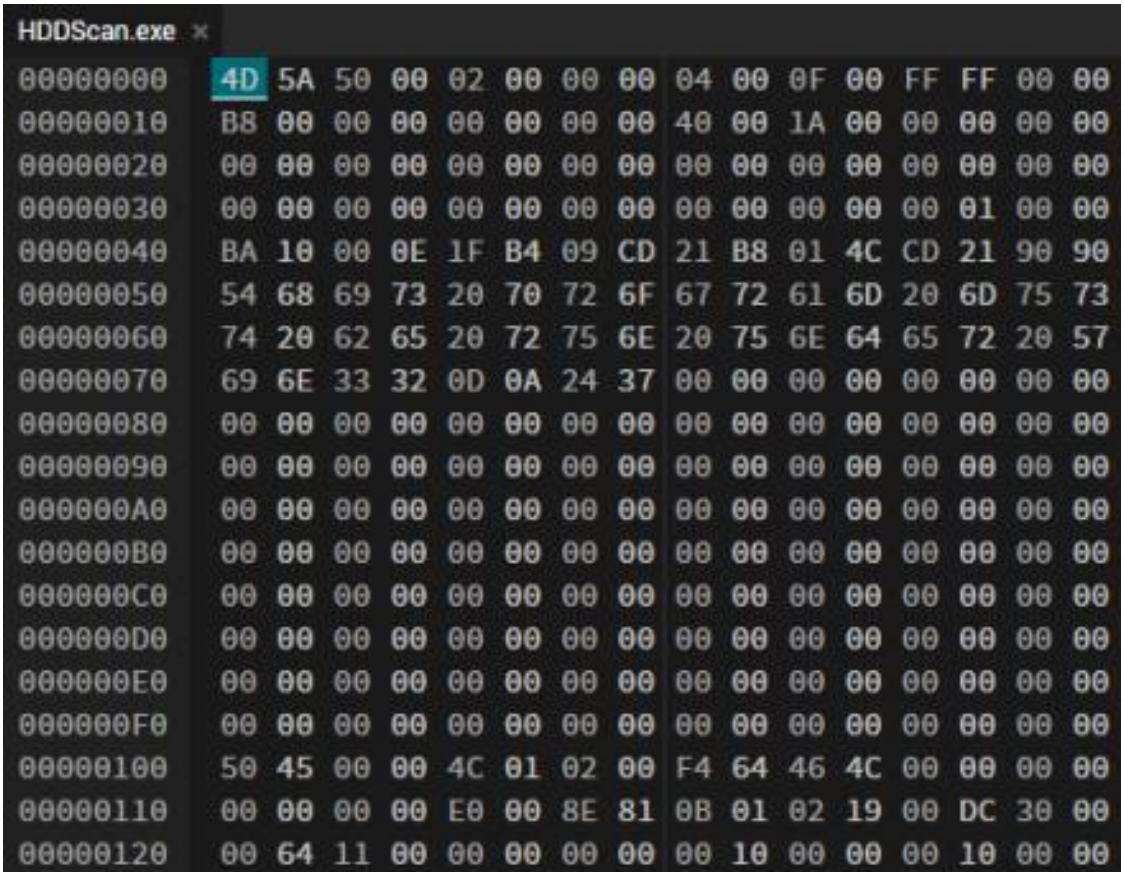


Рисунок 2.2 – Приклад шістнадцяткового представлення виконуваного файлу

Кожен символ у шістнадцятковому представленні несе в собі достатньо інформації для ідентифікації потенційно небезпечних патернів та аномалій у коді, що є основою при аналізі ШПЗ.

Попередній аналіз підходу для обробки «сирих» даних показав, що, зібравши, об'єднавши та розбивши на два класи шістнадцяткове представлення для кожного зі шкідливого (malware) та безпечного (benign) файлів у формат рядка (string), завдання набуває рис обробки природної мови, тобто порівнянне з пошуком смислових зв'язків у тексті, де доцільне використання мовних моделей як основи для навчання.

Один із способів попередньої обробки вихідних даних, коли зразок

конвертується у файл з його шістнадцятковим представленням зображено на рисунку 2.3.

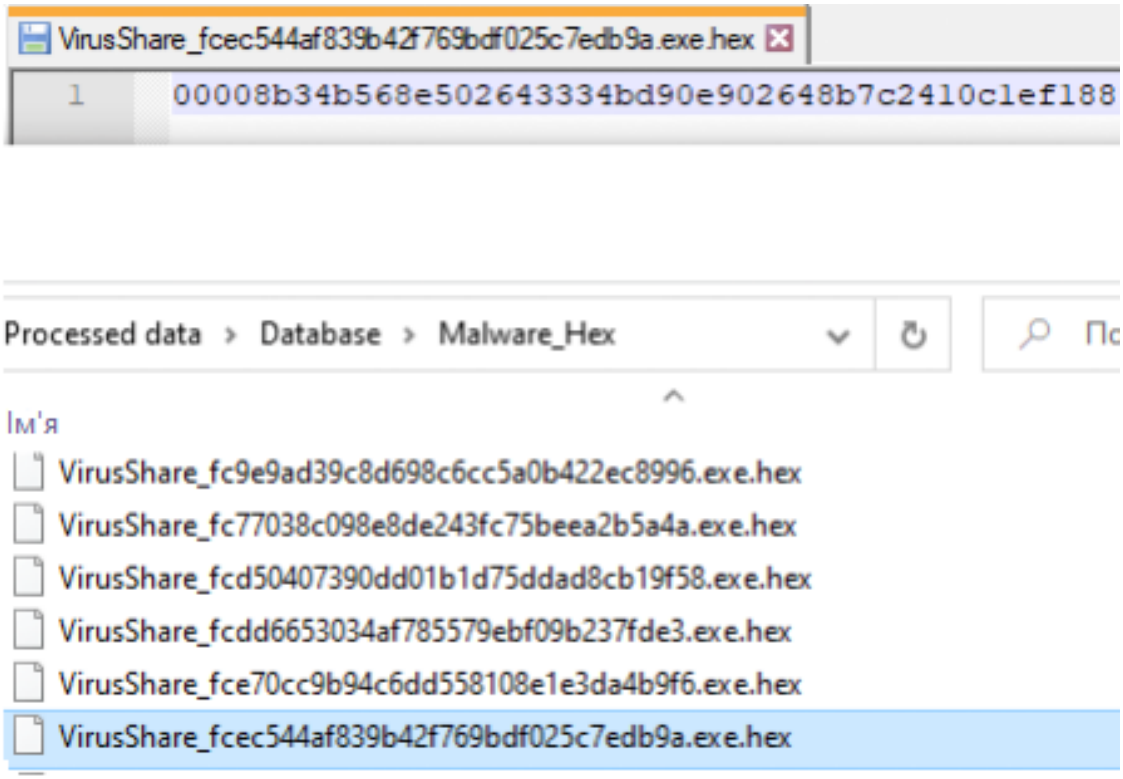


Рисунок 2.3 – Попередня обробка вихідних даних

Для самого процесу навчання будуть використані бібліотеки PyTorch, scikit-learn, NumPy та Transformers. Ці бібліотеки забезпечують надійну та гнучку платформу для побудови нейромережевих моделей, придатних як для задач класифікації, так і для передбачення загроз.

Важливо відзначити, що бібліотека Transformers надає можливість використання передових мовних моделей, таких як BERT та GPT, для створення потужного передбачувального програмного забезпечення. Інтеграція методів та інструментів, описаних вище, потенційно формує комплексне та сучасне антивірусне рішення, здатне ефективно протистояти загрозам у сфері кібербезпеки [33].

2.2 Аргументація ефективності використання Python

Гнучкість і ефективність, які пропонує Python, роблять цю мову одним із ключових інструментів у галузі машинного навчання та штучного інтелекту. Вона забезпечує доступ до широкого спектра спеціалізованих бібліотек, таких як Scikit-Learn, TensorFlow, PyTorch, Keras, LightGBM, CatBoost та інші, які значно спрощують процес побудови, навчання та валідації моделей. Це дозволяє розробникам уникнути рутинної реалізації алгоритмів "з нуля" і натомість зосередитися на логіці моделі, оптимізації гіперпараметрів та аналізі результатів. Такий підхід суттєво зменшує час на прототипування і підвищує продуктивність команди розробки.

Python заслужено вважається стандартом де-факто у сфері досліджень і розробки в галузі ШІ, завдяки не лише великій кількості готових інструментів, а й потужній технічній спільноті. Наявність величезної кількості відкритих прикладів, наукових публікацій із кодом, активних форумів (наприклад, Stack Overflow, GitHub, Medium, Towards Data Science) і професійної документації дає змогу швидко отримувати допомогу у випадку виникнення складнощів. Цей чинник особливо важливий на етапі експериментальної перевірки архітектур, коли кожна ітерація моделі потребує точного налаштування [33].

Крім того, Python вирізняється високим рівнем сумісності з іншими мовами програмування. Інструменти на кшталт Cython, SWIG або ctypes дають змогу інтегрувати компоненти, написані на C, C++ або навіть Fortran, без втрати продуктивності. Це критично важливо у випадках, коли обробка великих обсягів даних або виконання ресурсоємних обчислень вимагає оптимізації на низькому рівні.

Розвинена інфраструктура для обробки, візуалізації та аналізу даних. Бібліотеки на кшталт NumPy, Pandas, Matplotlib, Seaborn, а також інтерактивні середовища, як Jupyter Notebook, створюють зручне середовище для дослідження, передобробки та представлення даних у наочному вигляді. Це дозволяє не лише ефективно готувати дані до машинного навчання, а й виявляти приховані закономірності, аномалії або перекоси у вибірках, що особливо актуально при

роботі з нестабільними даними, як-от шкідливе програмне забезпечення [34].

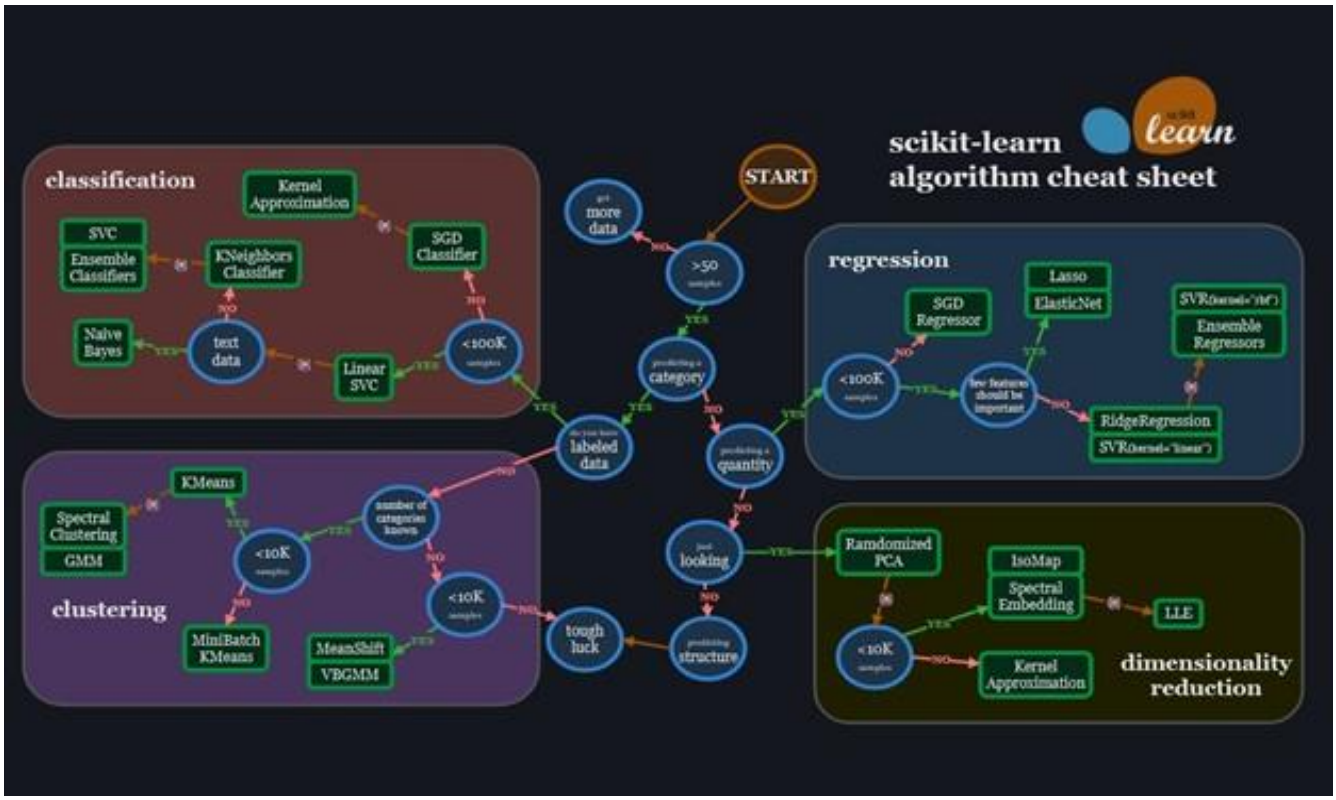


Рисунок 2.4 – Приклад навчання моделі

Завдяки гнучкості, масштабованості й широкому набору інструментів, Python є оптимальним вибором для реалізації систем на основі ШІ, орієнтованих на виявлення загроз, адаптацію до нових шкідливих зразків та автоматизацію аналітики у сфері кібербезпеки.

2.3 Порівняльний аналіз Python з альтернативними мовами програмування

Python за кілька десятиліть здобув провідні позиції у світі програмування. Завдяки простому синтаксису та доступності, ця мова стала ідеальним вибором для тих, хто тільки починає кар'єру в IT. За даними Finextra, близько 40% користувачів Python молодь віком від 21 до 29 років, ще 10% у віковій категорії 18–20 років.

Станом на 2020 рік кожен третій Python-розробник мав менш ніж один рік досвіду програмування. Такий показник зумовлений як легкістю вивчення мови,

так і впливом глобальних факторів, зокрема пандемії, що пришвидшила попит на цифрові рішення. Python дозволяє швидко перейти до практичних результатів без потреби у тривалому навчанні [36].

Бюро статистики праці США (BLS) прогнозує, що у період 2020–2030 років попит на програмістів зросте більш ніж на 22%. Така тенденція сприяє притоку нових кадрів у сферу розробки, зокрема серед тих, хто шукає перехід у технічну галузь з інших професій. Python, завдяки простоті, гнучкості та широким можливостям, є оптимальним вибором для такої категорії фахівців. Також варто зазначити, що Python використовується як серверна мова для створення близько 1,3% усіх вебсайтів у світі. Серед компаній, які активно застосовують цю мову у своїх продуктах — Instagram, Netflix, Uber, Dropbox, Google та інші. Python вирізняється невисоким порогом входу, освоїти його базовий синтаксис можна за короткий час. Це забезпечує легшу співпрацю між технічними й нетехнічними фахівцями в команді. Мова має зручну структуру, яка не перевантажує розробника деталями на початкових етапах роботи.

Однією з головних переваг є багатий набір бібліотек і фреймворків, які охоплюють такі напрямки:

- машинне навчання та штучний інтелект;
- data science;
- візуалізація та аналітика даних;
- розробка веб т десктоп систем;
- обробка природної мови.

Серед найбільш популярних фреймворків Django, Flask, Pyramid, Twisted, Falcon. Вони суттєво прискорюють процес розробки, зменшуючи кількість коду і підвищуючи надійність продукту. Це робить Python ефективним інструментом для реалізації проєктів будь-якого масштабу. від стартапів до систем світового рівня, зокрема в таких організаціях, як YouTube, Spotify, IBM, NASA та Google.

На рисунку 2.5 зображено де найчастіше використовують Pyhon.

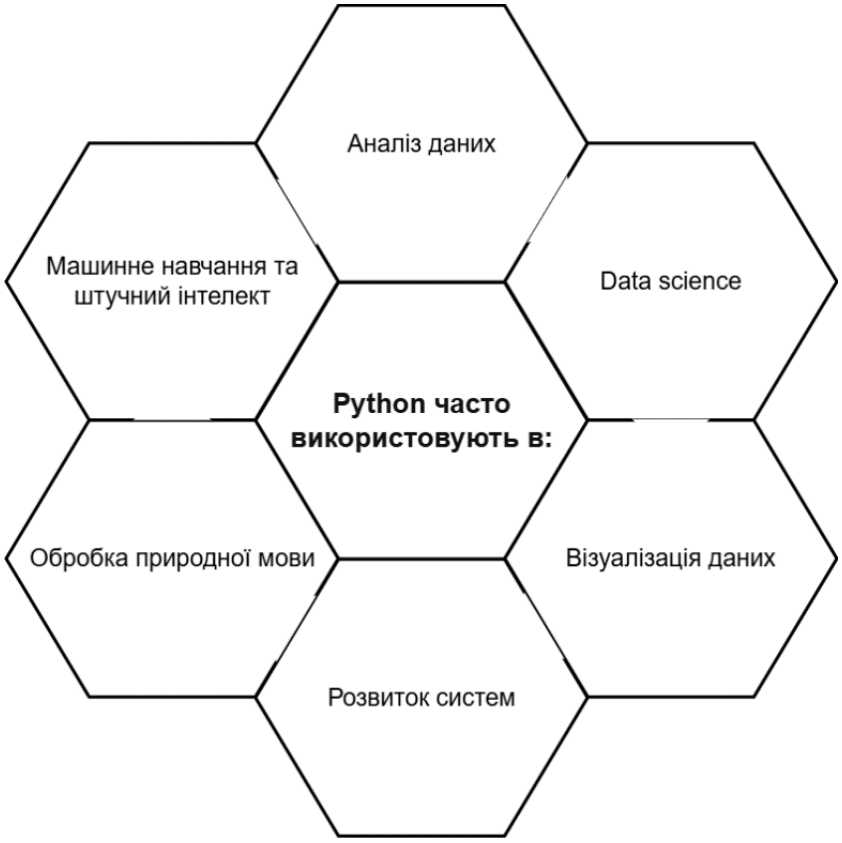


Рисунок 2.5 – Часте використання Python [37].

Щоб зрозуміти усі переваги або недоліки Python, потрібно порівняти його з іншими популярними мовами програмування.

Python забезпечує швидку розробку завдяки простому синтаксису та великій кількості бібліотек. Java, будучи компільованою мовою, забезпечує високу продуктивність та широко використовується в корпоративних системах. Python простіший у вивченні та використанні, але поступається C++ у продуктивності та контролі над ресурсами, що робить C++ кращим вибором для системного програмування та розробки ігор. Python переважає у сфері наукових обчислень, автоматизації та бекенд-розробки, тоді як JavaScript є незамінним для фронтенд-розробки та створення інтерактивних веб-додатків. Python забезпечує швидку розробку та гнучкість, тоді як C# краще підходить для розробки під Windows та створення ігор за допомогою Unity. Python має ширший спектр бібліотек та простіший синтаксис, але Go забезпечує вищу продуктивність та краще підходить для розробки мікросервісів та високонавантажених систем. Python є універсальнішою мовою з широким спектром застосувань, тоді як R

спеціалізується на статистичному аналізі та візуалізації даних.

Обидві мови мають простий синтаксис та використовуються у веб-розробці. Python має ширший спектр застосувань, тоді як Ruby відомий завдяки фреймворку Ruby on Rails. Python забезпечує чистіший синтаксис та ширший спектр застосувань, тоді як PHP залишається популярним вибором для серверної веб-розробки.

Таблиця 2.1 - Дані порівняння мов [39].

Мова	Навчання	Кросплатформеність	Сумісність з різними ОС	Типізація	Продуктивність	Тип мови	Наявність бібліотек
Python	Легко	Так	Так	Динамічна	Відносно повільна	Інтерпретована	Так
Ruby	Середнє	Так	Так	Динамічна	Швидка	Інтерпретована	Так
C++	Важко	Так	Так	Статична	Швидка	Компільована	Так
JavaScript	Легко	Так	Так	Динамічна	Відносно повільна	Інтерпретована	Так
Java	Важко	Так	Так	Статична	Швидка	Компільована	Так
Perl	Важко	Так	Так	Динамічна	Швидка	Інтерпретована	Так (модулі)
PHP	Легко	Ні	Так	Динамічна	Повільна	Компільована	Так
C#	Середнє	Так	Так	Статична	Швидка	Компільована	Так
Kotlin	Легко	Так	Так	Статична	Швидка	Компільована	Так
Swift	Легко	Так	Так	Статична	Швидка	Компільована	Так
R	Важко	Так	Так	Динамічна	Повільна	Інтерпретована	Так

Отже з таблиці порівняння можна зробити наступні висновки:

– python є однією з найлегших для вивчення мов, з гарною підтримкою кросплатформенності та багатими бібліотеками;

– java та C++ демонструють високу продуктивність, але мають складну

криву навчання;

– javaScript та Ruby також мають легкий старт, але менш багаті на бібліотеки;

– r ідеально підходить для статистики й аналізу даних, проте має високу криву навчання та не дуже високу продуктивність;

– php втрачає позиції через слабку підтримку кросплатформенності та повільну продуктивність;

– swift і Kotlin стають популярними для мобільної розробки, мають хорошу продуктивність, але обмеж0ені в екосистемі.

Python заслужено займає провідне місце серед сучасних мов програмування завдяки своїй простоті, гнучкості та універсальності. На відміну від більшості інших мов, таких як C++ чи Java, Python має надзвичайно низький поріг входу, що робить його ідеальним вибором для початківців. Простий синтаксис, який максимально наближений до людської мови, дозволяє швидко писати зрозумілий код без складної структури чи надмірної формальності.

Хоча Python поступається деяким компільованим мовам у швидкості виконання, його недоліки з лишком компенсуються наявністю великої кількості бібліотек, фреймворків та активною спільнотою. Це дає змогу розробляти складні проєкти, від аналізу даних і машинного навчання до веброзробки та автоматизації без потреби створювати інструменти «з нуля». Саме завдяки цьому Python став основною мовою для таких гігантів, як Google, NASA, Spotify, Instagram, Netflix. У порівнянні з мовами, такими як Java або C#, Python не вимагає суворої типізації, що знижує складність підтримки коду та прискорює його написання. Якщо Java та C++ часто використовуються для системного програмування, мобільних додатків або великих інфраструктурних рішень, то Python став стандартом у сферах наукових обчислень, штучного інтелекту, скриптування та роботи з даними.

З урахуванням тенденцій розвитку, Python не лише не втрачає актуальності, а й продовжує нарощувати вплив, особливо в галузях, що потребують гнучких, швидких і зрозумілих інструментів для вирішення нестандартних задач. Для тих, хто лише починає свій шлях у програмуванні або шукає універсальний інструмент для широкого спектру завдань, Python залишається беззаперечним фаворитом.

2.4 Формування та попередня обробка навчального датасету

Як вже було сказано раніше, жодні більш-менш поверхневі відомості про файли (наприклад, SHA256 і MD5 хеш-суми, сигнатури, поведінковий аналіз на основі викликів API) не дають повного уявлення про шкідливість або безпеку тих чи інших файлів, тому датасет буде розроблений на основі шістнадцяткового їх представлення з міркувань створення моделі глибокого аналізу.

Головними джерелами для створення датасету є як приватні колекції ШПЗ, розміщені в GitHub, так і колекції зразків за авторством великих видань і компаній, наприклад – «PE Malware Machine Learning Dataset» від «Practical Security Analytics LLC» і щоденно оновлюваний інтернет-ресурс «MalwareBazaar» від «abuse.ch». Оскільки більшість колекцій складаються виключно із зразків шкідливих файлів, для створення «безпечного» набору даних було прийнято рішення витягти всі потенційно важливі зразки безпосередньо з «чистого» образу операційної системи Windows 10, а саме – виконувані файли у форматі «.exe» і динамічно підключаються бібліотеки у форматі «.dll». На додаток до цього для створення датасету були задіяні практично всі наявні виконувані файли і динамічно підключаються бібліотеки самого різного ПЗ, розміщені на персональному комп'ютері спеціально орендованому для виконання роботи. [39. 40]

Процес перетворення вихідних даних полягає в конвертації заздалегідь розбитих на дві вибірки («malware» і «benign») виконуваних файлів у файл з шістнадцятковим представленням. Цей підхід дозволяє як зробити аналіз вмісту файлу нейромережевою архітектурою наочним (у плані відстеження патернів) і найбільш повним, так і створити алгоритм уніфікації даних перед їх подачею в розроблювану антивірусну систему для подальшої класифікації. Код для попередньої обробки вихідних даних доступний для ознайомлення в додатку А.

Варто відзначити, що шістнадцяткове представлення даних також дозволяє мінімізувати вплив різних алгоритмів шифрування та архівації, використовуваних авторами ШПЗ для приховування істинного функціоналу.

Процес підготовки даних для подачі в нейромережеву модель для навчання включає збір і систематизацію вже перетворених в шістнадцятковий формат даних.

Код, представлений в додатку Б, зчитує перетворені файли, розбиває їх на два класи (спираючись на заздалегідь відсортовані підвібірки) і зберігає в структурованому вигляді в набір даних формату бібліотеки HDF5, що забезпечує швидкий і надійний доступ до великих обсягів даних, їх ефективне зберігання.

Вага кожного образу ранжується від 1 КБ до 170 МБ. Кінцевий датасет був розроблений в декілька етапів:

– перша ітерація, 19811 зразків malware, 2082 зразка benign, загальний розмір 15 ГБ;

– друга ітерація, 25837 зразків malware, 8724 зразків benign, загальний розмір 50 ГБ;

– третя ітерація, 26682 зразки malware, 10251 зразкок «benign», загальний розмір 55 ГБ.

В процесі створення датасету було взято до уваги важливий аспект правильного балансу між класами даних.

Ще одним важливим кроком стало впорядкування структури самого датасету. Для зручності використання та гнучкості масштабування, дані було розділено не лише за класами, а й за категоріями. Такий підхід дозволяє як проводити загальний аналіз, так і тестувати модель у різних контекстах, оцінюючи її стійкість до зміни джерел даних. Після створення та структуризації датасету були проведені первинні тести на його придатність для використання в задачах машинного навчання. Зокрема, було реалізовано базову класифікаційну модель на основі згорткової нейронної мережі, яка приймає вхідні шістнадцяткові представлення файлів, перетворені у псевдозображення (матриці фіксованого розміру). Початкові результати показали, що навіть на простій архітектурі можливо досягти прийнятного рівня точності (близько 91% при F1 понад 0.88), що свідчить про потенціал застосованого підходу.

Надалі датасет буде використовуватись не лише для класифікації, але й для експериментів зі змішаними архітектурами (наприклад, поєднання CNN з рекурентними або трансформерними шарами), що дозволить глибше аналізувати послідовності байтів, виявляючи приховані закономірності в структурі як

шкідливих, так і легітимних файлів. Такий підхід сприятиме підвищенню точності виявлення нових, невідомих зразків, які не мають чітких сигнатур, але містять характерні аномалії на рівні байтових послідовностей.

Проведений аналіз підходу до формування датасету для задач виявлення ШПЗ спрямований на подолання обмежень традиційних методів, таких як хешування чи сигнатурний аналіз, які не забезпечують надійної ідентифікації шкідливих файлів.

Обґрунтованим виглядає рішення працювати з сирими байтовими послідовностями у шістнадцятковому форматі, що дозволяє побудувати модель, чутливу до глибинних закономірностей в структурі виконуваних файлів. Джерела даних охоплюють як відкриті шкідливі так і легітимні зразки, що забезпечує змістовне наповнення обох класів.

Важливо, що під час побудови датасету було дотримано принципу класового балансу, а також враховано структурну організацію даних для подальшої гнучкої інтеграції в моделі. Первинні тести з використанням згорткової нейронної мережі продемонстрували високу ефективність, що підтверджує потенціал обраного підходу. Така система не лише дозволяє моделювати поведінку на основі вмісту файлу без необхідності ручного аналізу, але й відкриває перспективи застосування більш складних гібридних архітектур. Це є передумовою до розробки високоточних засобів виявлення сучасних загроз, які обходять класичні механізми захисту. Додатково, застосування методів автоматизованого навчання дозволяє адаптувати систему до нових типів шкідливого коду у реальному часі. Таким чином, створюється динамічне середовище, здатне до самостійного вдосконалення. Подальше масштабування моделі та розширення навчальної вибірки сприятимуть підвищенню її стійкості до атак типу «zero-day» та інших нетипових загроз. Це, у свою чергу, дає змогу істотно скоротити час реакції на інциденти безпеки та мінімізувати їх наслідки для інформаційної інфраструктури. Завдяки цьому системи захисту набувають здатності до проактивного виявлення загроз, що значно випереджає традиційні реактивні підходи до кібербезпеки.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНЮВАННЯ ДОСТОВІРНОСТІ РОЗРОБЛЕНОГО РІШЕННЯ

3.1 Архітектура та реалізація серверної частини системи

Трансформери, як одні з найпотужніших моделей для обробки послідовних даних, були обрані як основа для backend-частини розроблюваного антивірусного рішення. Вони здатні ефективно виявляти складні залежності у великих обсягах даних завдяки механізму «self-attention». Перша ітерація навчання полягала в перевірці гіпотези про можливість застосування великих мовних моделей, а саме BERT від компанії Google, яка є двонаправленою трансформерною моделлю, що використовує архітектуру «Transformer Encoder».

Повний цикл навчання складається з:

– використання попередньо навчених ваг моделі «bert-base-uncased»;

– створення токенізованого набору даних, придатного для подачі на модель;

– розділення даних на навчальну, валідаційну та тестову вибірки;

– додавання повнозв'язного шару нейронної мережі для задачі класифікації на основі вихідних даних BERT;

– використання «AdamW» для оптимізації параметрів моделі;

– виконання навчання моделі з підрахунком функції втрат;

– оцінки моделі на валідаційній та тестовій вибірках;

– збереження моделі у форматі «.pth».

Середнє значення функції втрат в середині навчання виявилося занадто велике (дорівнює 0.3; в подальшому вийшло на плато і не показувало позитивних зрушень) для реалізації глобальної мети навчання антивірусного ПЗ, бо подібна похибка для задачі класифікації чутливих даних неприпустима.

Процес навчання був зупинений, але сама суть використання трансформерів концептуально залишала надії на більш високі узагальнюючі здібності моделі, необхідні для досягнення поставленої мети. Після виявлення неприпустимо високого значення функції втрат при використанні моделі, було прийнято рішення перейти до наступної ітерації експериментів, але вже з поліпшеною моделлю.

Таблиця 3.1 - Програмні та апаратні відмінності двох моделей.

	BERT	RoBERTa
Розмір (мільйони)	Base: 110 Large: 340	Base: 110 Large: 340
Час навчання	Base: 12 днів Large: 4 дні	Large: 4 дні
Продуктивність	Повільно	На 20% більше
Дані	16 Гб	160 Гб
Метод	Траесформер з MLM та NSP	BERT без NSP**

RoBERTa перевершує BERT, демонструючи поліпшення продуктивності на 2-20% (в залежності від поставленої задачі) завдяки навчанню на значно більшому обсязі даних, додатковому налаштуванню гіперпараметрів і опущення методу NSP (Next Sentence Prediction) при навчанні.

Використання «roberta-base» в якості основи для навчання ПЗ «CERBEROS» версії «1.0» принесло позитивний зсув по відношенню до всіх метрик. Крім зміни трансформера, під час експериментів для версії «2.0» було зменшено розмір батчу з 64 до 32, що дозволило знизити стартове значення функції втрат і збільшити стабільність навчання. Кількість епох збільшено з 3 до 5, а на вхід подається розширений датасет вагою 50 ГБ. Весь процес навчання зайняв, орієнтовно, 6 днів.

Такі зміни сприяли глибшому засвоєнню моделей контекстної інформації та дозволили досягти кращого узагальнення на нових даних. Завдяки цьому покращилась не лише точність, а й надійність розпізнавання різних типів загроз. Крім того, використання більшого датасету дало змогу моделі краще враховувати варіативність шкідливих зразків і підвищити її стійкість до нових атак.

На рисунку 3.1 зображено класову структуру алгоритму навчання.

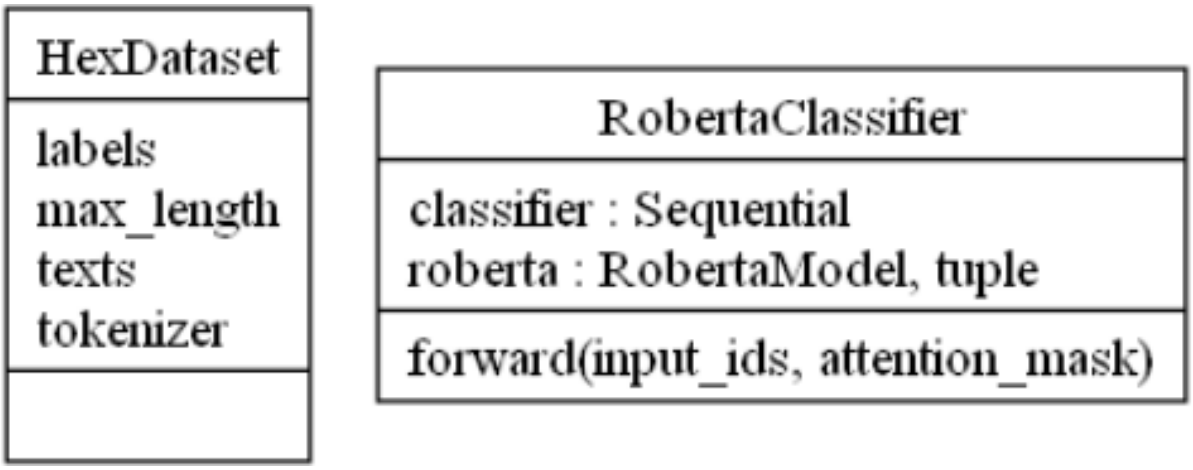


Рисунок 3.2 – Класова структура алгоритму навчання

Клас «HexDataset» містить наступні атрибути:

– labels, мітки класів для завдань класифікації;

– max_length, максимально допустима довжина символьної послідовності для обробки;

– texts, найдовші дані які підлягають класифікації;

– tokenizer, інструмент для перетворення «сирих» даних в послідовність токенів для подальшої подачі на модель.

Клас «RobertaClassifier» є класифікатором, заснованим на моделі RoBERTa, і включає в себе послідовність шарів нейронної мережі, кортеж параметрів і метод «forward» для обробки вхідних даних. Створена архітектура дозволяє проводити точну класифікацію, застосовуючи потужності трансформерної моделі. Нижче, на рисунку 3.2, представлено різницю значень функції втрат протягом п'яти епох навчання, який демонструє значне зниження втрат, особливо в перші кілька епох, що свідчить про швидку збіжність моделі до оптимального рішення. Цей результат підтверджує ефективність обраної архітектури та її здатність адаптуватися до складних патернів у даних. Завдяки гнучкості класу «RobertaClassifier», його можна легко інтегрувати в більш складні системи аналізу загроз або доопрацювати під специфічні задачі. Окрім цього, реалізація дозволяє зберігати модель після кожної епохи, що дає змогу відстежувати прогрес та повертатися до кращої версії у разі перенавчання.

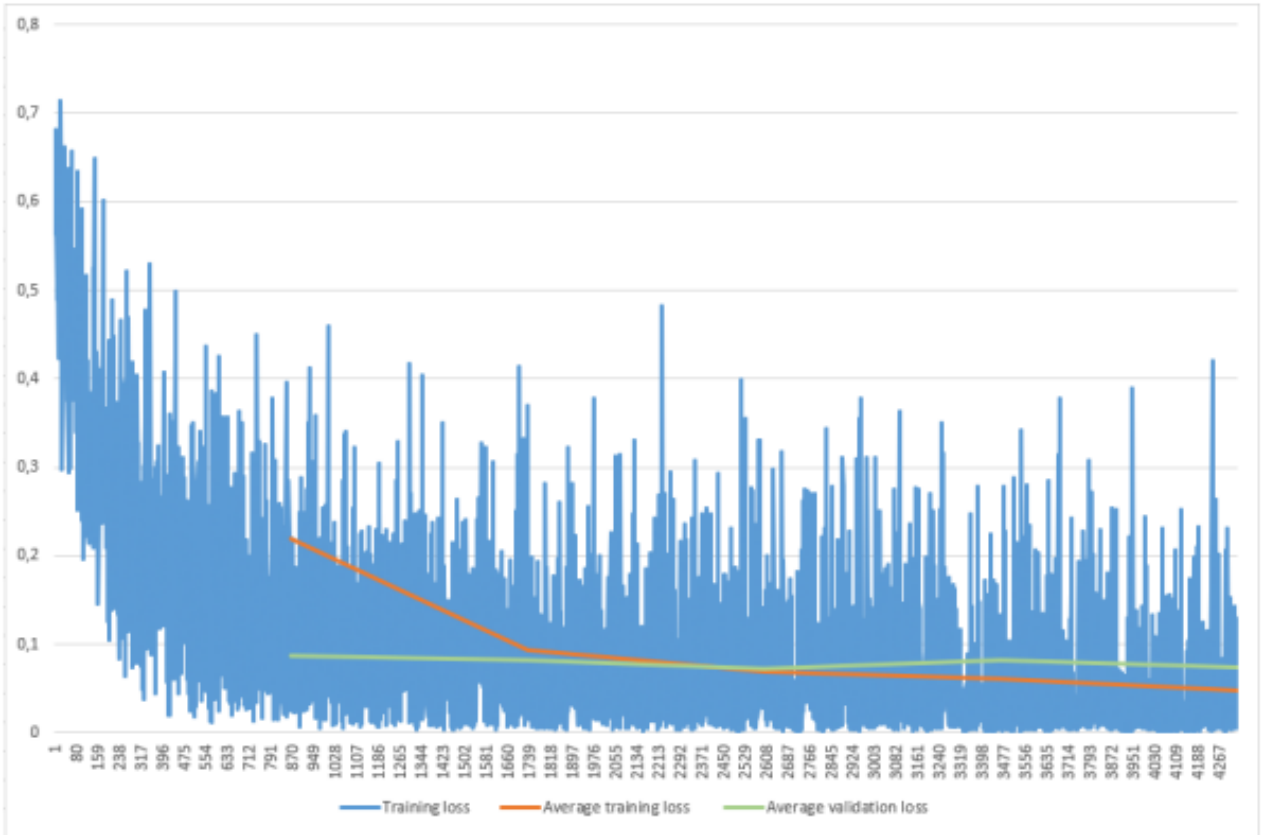


Рисунок 3.2 – Різниця значень функцій втрат

На рисунку представлено наступне:

– синім кольором наведено втрати для кожного кроку навчання на навчальні вибірці;

– помаранчевим кольором наведено середні значення функцій втрат кожної епохи для навчальної вибірки;

– зеленим кольором наведено середні значення функції втрат кожної епохи для валідаційної вибірки.

Також на рисунку виразно відстежується прогресуюча позитивна тенденція до зменшення втрат у міру навчання. Для підвищення ефективності моделі використовувався підхід перехресної ентропії «cross-entropy», що дозволило краще оцінити узагальнюючу здатність моделі на різних підвыбірках даних. В результаті були отримані метрики, що свідчать про більш високу стійкість до перенавчання в порівнянні з первісною моделлю на базі мовної моделі. Оскільки існують тимчасові та апаратні обмеження для реалізації нейромережевого антивірусного рішення, процес навчання був зупинений на кількості 5 епох без подальшого їх збільшення.

Метрики даної ітерації навчання зображено на рисунку 3.3.

```
Validation Precision: 0.994475138121547
Validation Recall: 0.975987606506584
Validation F1 Score: 0.9851446442533228
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.96       874
           1       0.99      0.98      0.99      2582

    accuracy                           0.98      3456
   macro avg       0.96      0.98      0.97      3456
weighted avg       0.98      0.98      0.98      3456

Test Precision: 0.996414342629482
Test Recall: 0.976953125
Test F1 Score: 0.9865877712031558
Test Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       897
           1       1.00      0.98      0.99      2560

    accuracy                           0.98      3457
   macro avg       0.97      0.98      0.97      3457
weighted avg       0.98      0.98      0.98      3457
```

Рисунок 3.3 – Метрики передфінальної моделі

Для оцінювання якості класифікації було використано стандартні метрики машинного навчання: Precision (точність), Recall (повнота) та F_1 Score (гармонійне середнє), що особливо важливі у задачах бінарної класифікації, пов'язаної з виявленням шкідливого програмного забезпечення.

Основні метрики обчислюються на основі формул описаних нижче.

Precision визначає частку коректно ідентифікованих позитивних зразків серед усіх зразків, які були класифіковані як позитивні за формулою (3.1).

$$Precision = \frac{TP}{TP+FP} \tag{3.1}$$

де TP (True Positive) — кількість правильно класифікованих шкідливих зразків; FP (False Positive) — кількість безпечних зразків, класифікованих як шкідливі.

Recall визначає частку коректно класифікованих позитивних зразків серед усіх справжніх позитивних зразків за формулою (3.2).

$$Recall = \frac{TP}{TP+FN} \quad (3.2)$$

де TP (True Positive) — кількість правильно класифікованих шкідливих зразків; FN (False Negative) — кількість шкідливих зразків, класифікованих як безпечні.

F_1 Score, гармонічне середнє між Precision та Recall за формулою (3.3).

$$F_1 Score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (3.3)$$

де Precision - частка коректно ідентифікованих позитивних зразків серед усіх зразків, які були класифіковані як позитивні; Recall - частка коректно класифікованих позитивних зразків серед усіх справжніх позитивних зразків.

Згідно з результатами, описаними вище, модель досягла наступних значень на валідаційній вибірці: Precision = 0.994475, Recall = 0.975987, F1 Score = 0.985145

Для перевірки правильності обчислень проводиться розрахунок за формулою (3.3).

$$F_1 = 2 * \frac{0.994475 * 0.975987}{0.994475 + 0.975987} \approx 0.9851$$

Це збігається з поданим у звіті значенням: 0.9851, що підтверджує баланс між точністю та повнотою при класифікації шкідливих файлів.

На тестовій вибірці спостерігаються аналогічно високі результати: Precision = 0.996414, Recall = 0.976953, F1 Score = 0.986588

Для перевірки правильності обчислень проводиться розрахунок за формулою (3.3).

$$F_1 = 2 * \frac{0.996414 * 0.976953}{0.996414 + 0.976953} \approx 0.9866$$

У деталізованому класифікаційному звіті представлено метрики окремо для кожного класу.

Таблиця 3.2 – Представлення метрик

Метрика	Безпечне ПЗ	ШПЗ
Precision	0.93-0.94	0.99-1.00
Recall	0.98 – 0.99	0.98
F_1 Score	0.96	0.99

Модель показує трохи гірші результати для класу безпечного ПЗ (зниження precision до 0.93–0.94), що вказує на наявність помилкових позитивних спрацювань. У контексті задачі виявлення шкідливого ПЗ це є прийнятним компромісом, оскільки головною метою є мінімізація False Negative, тобто невиявлених шкідливих об’єктів.

Отримані результати свідчать про високу здатність моделі точно і стабільно ідентифікувати зразки ШПЗ як на валідаційній, так і на тестовій вибірках. Високе значення Precision, демонструє низьку кількість помилкових спрацьовувань для шкідливих зразків, тоді як значення Recall, вказує на високу здатність знаходити більшість загроз. Незначне зниження точності для безпечних зразків є очікуваним і може бути оптимізоване шляхом додаткового підбору гіперпараметрів або балансування вибірки. Це свідчить про ефективність обраного підходу до навчання моделі та потенціал її подальшого вдосконалення.

3.2 Розробка інтерфейсу користувача та клієнтської логіки

Для розгортання, найбільш наочного використання кінцевим користувачем навченої моделі, а також більш зручного тестування та впровадження різного

функціоналу взаємодії з операційною системою необхідно створити графічний інтерфейс. Це забезпечить не тільки зручність у використанні, але й полегшить процес оновлення та додавання нових функцій до антивірусного рішення. Розробка графічного інтерфейсу є важливою складовою створення повноцінного програмного продукту.

Моменти побудови комфортного, зручного та функціонального інтерфейсу програми:

– інтуїтивність: інтерфейс має бути зрозумілим і очевидним для користувача, мінімізуючи необхідність навчання взаємодії;

– консистентність: всі елементи інтерфейсу повинні мати єдину стилізацію;

– простота: відсутність візуальної перевантаженості;

– чуйність: програмна оболонка повинна швидко і чітко реагувати на дії користувача;

– естетика: дизайн повинен бути приємним для очей.

Крім базових принципів побудови інтерфейсу, варто обмежити взаємодію користувача з операційною та файловою системами через розроблюване антивірусне рішення, оскільки навіть найдосвідченіший користувач може з необережності завдати шкоди операційній системі або іншим чутливим даним. Забезпечення безпеки даних користувача є пріоритетною задачею розробки. Для кожної важливої запитуваної дії (наприклад, видалення файлу) необхідно отримати додаткове підтвердження від користувача. Крім того, доцільно реалізувати систему журналювання дій користувача, що дозволить не лише забезпечити прозорість взаємодії, але й стане у пригоді при виявленні потенційних загроз або помилок у роботі програми. Журнали слід захищати від несанкціонованого доступу, а також передбачити можливість їхнього автоматичного аналізу з метою виявлення аномальної активності. Також доцільно інтегрувати механізми поведінкового аналізу, які дозволять виявляти підозрілі шаблони взаємодії користувача з системою.

Прототип графічного інтерфейсу зображено нижче на рисунку 3.4.

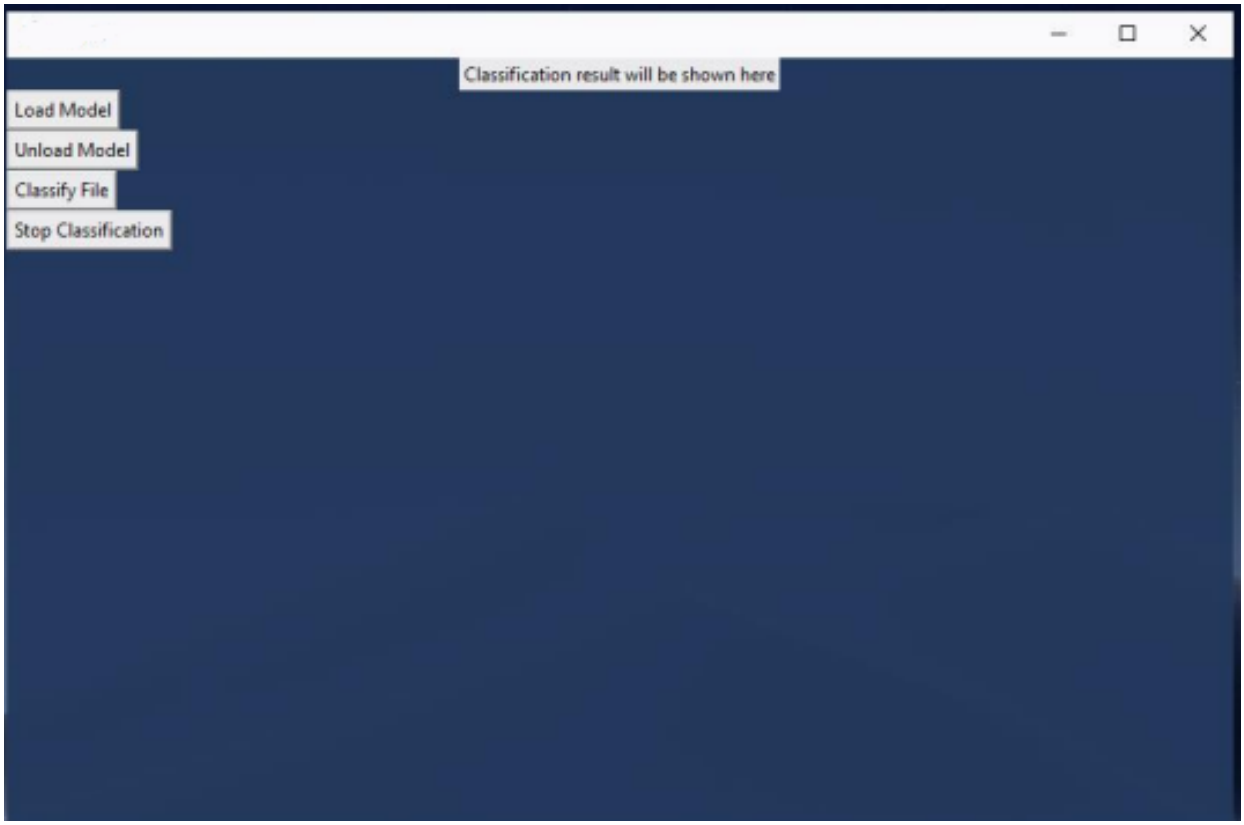


Рисунок 3.4 – Перший варіант інтерфейсу

Основним компонентом для відображення графічного інтерфейсу є бібліотека Tkinter. З її допомогою можна створювати віконні застосунки, застосовуючи інструменти кастомізації практично до будь-якого елементу графічного інтерфейсу, чи то іконка програми, шрифти або розмір і форма активних кнопок.

У першій ітерації можна побачити наявність функціональних елементів для завантаження та вивантаження з оперативної пам'яті навченої моделі, кнопки вибору файлу для класифікації та примусової зупинки запущеного процесу. Одним з важливих аспектів використання бібліотеки Tkinter є можливість створення модульних компонентів, що дозволяє легко тестувати та налагоджувати окремі частини інтерфейсу. Це значно спрощує процес розробки графічного інтерфейсу та забезпечує більшу гнучкість у реалізації логіки взаємодії з користувачем. Такий підхід також сприяє покращенню читабельності й підтримуваності коду.

На рисунку 3.5 зображено розвиток інтерфейсу головного вікна програми.

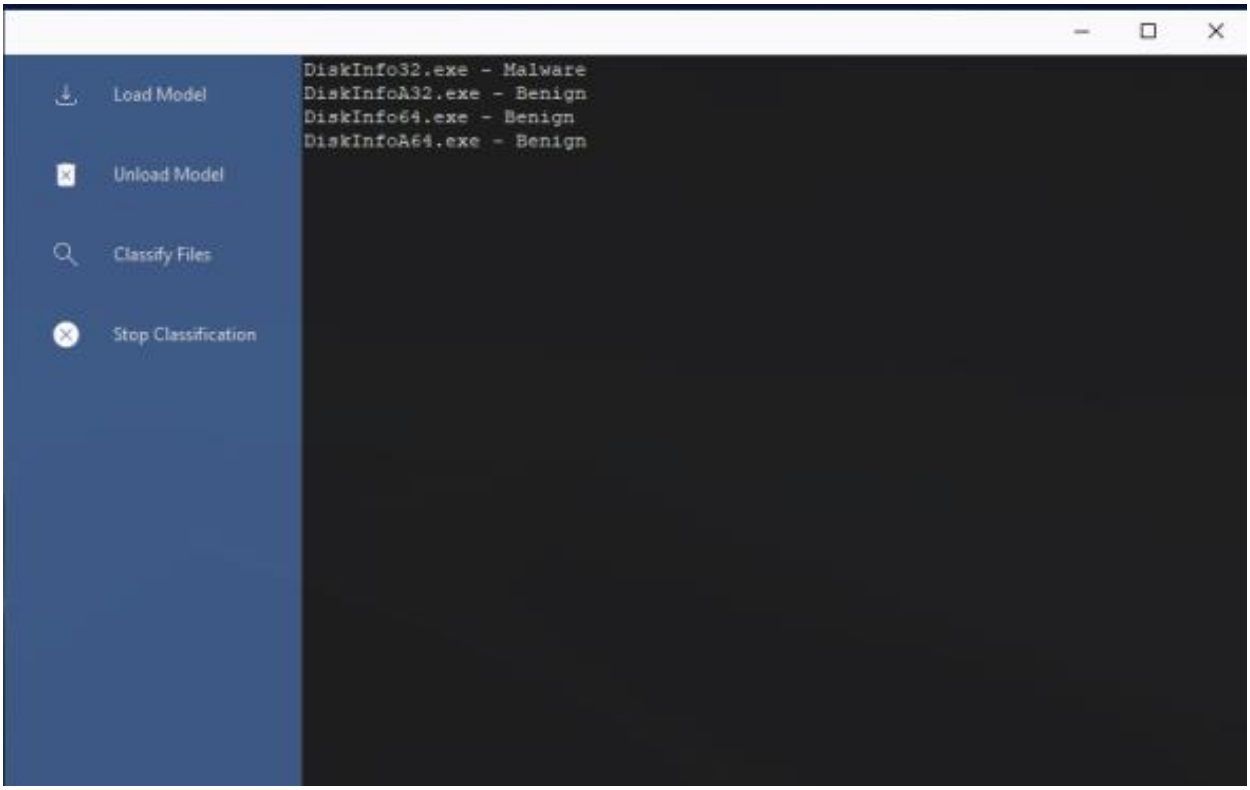


Рисунок 3.5 – Другий варіант інтерфейсу

В рамках розробки інтерфейсу було враховано не лише базові вимоги до функціональності, але й рекомендації з використання, зокрема принципи зворотного зв'язку, консистентності, передбачуваності та доступності. Це дозволило створити інтерфейс, що орієнтований на кінцевого користувача, незалежно від його попереднього досвіду у сфері комп’ютерної безпеки.

Застосовано іконки для кожної з кнопок, що значно покращило візуальне сприйняття та інтуїтивну навігацію між елементами GUI. Графічні елементи були тимчасово запозичені з документації «Introducing Material Symbols» з подальшим планом розробки повністю авторських наборів піктограм. Це рішення дозволило забезпечити сучасний вигляд інтерфейсу та адаптацію до різних типів екранів і DPI-параметрів. Головним удосконаленням стала реалізація функції мультивибору файлів, що відкриває можливість одночасної обробки декількох об’єктів без необхідності повторного завантаження інтерфейсу. Консоль, розташована праворуч, забезпечує оперативне інформування користувача про результати сканування кожного з обраних файлів у режимі реального часу. Такий підхід підвищує прозорість процесу аналізу й надає змогу швидко реагувати на загрози.

Друга ітерація інтерфейсу передбачала поглиблення взаємодії з користувачем: було додано низку візуальних підказок, зокрема, інформаційні банери, повідомлення про помилки, кольорову індикацію стану виконання задач тощо. Помилки та системні повідомлення виводяться в окремій частині консолі, що значно скорочує кількість невірних дій з боку користувача. Такий підхід підвищив загальну стабільність застосунку та зменшив поріг входу для нових користувачів.

У третій ітерації, яка й стала фінальною, інтерфейс було суттєво оптимізовано. З'явилася назва системи в заголовку програми, а всі іконки створено вручну в редакторі GIMP, що є вільним програмним забезпеченням. Завдяки цьому вдалося забезпечити унікальність графічного оформлення без порушення авторських прав. Кожна піктограма відповідає своїй функції, що дозволяє легко орієнтуватися навіть без додаткових текстових підказок.

Графічна частина інтерфейсу включає повний спектр необхідного функціоналу для гнучкої взаємодії з системою:

– завантаження моделі в оперативну пам'ять комп'ютера;

– запит оновленої версії моделі через API з хмарного сховища;

– вивантаження моделі з пам'яті ПК;

– можливість вибору одного або декількох файлів для сканування;

– сканування всієї папки з автоматичним рекурсивним обходом вкладених директорій;

– реалізація «швидкого сканування», яке фокусується на критичних директоріях, де найчастіше зустрічається шкідливе ПЗ (наприклад, %AppData%, %Temp%, C:\Windows\System32);

– опція повного сканування всіх наявних логічних дисків;

– кнопка зупинки процесу сканування у разі потреби;

– експорт результатів сканування у форматі таблиці .csv — ця можливість дає змогу інтегрувати результати з іншими аналітичними інструментами (див. рисунок 3.7);

– очищення вікна результатів, що забезпечує зручність при повторному використанні інтерфейсу;

– виведення короткої довідки з інструкцією щодо використання системи;

– аварійне завершення роботи програми за натисканням відповідної кнопки;

– можливість видалення знайденого шкідливого;

– модуль для виводу відсоткового навантаження на систему у реальному часі, що дозволяє оцінити вплив процесу сканування на ресурси комп’ютера.

Загалом, структура інтерфейсу розроблена таким чином, аби забезпечити ефективну взаємодію з користувачем на всіх етапах: від ініціалізації сканування — до експорту результатів або вжиття заходів щодо виявлених загроз. Таким чином, продукт може бути рекомендований як для персонального, так і для корпоративного використання.

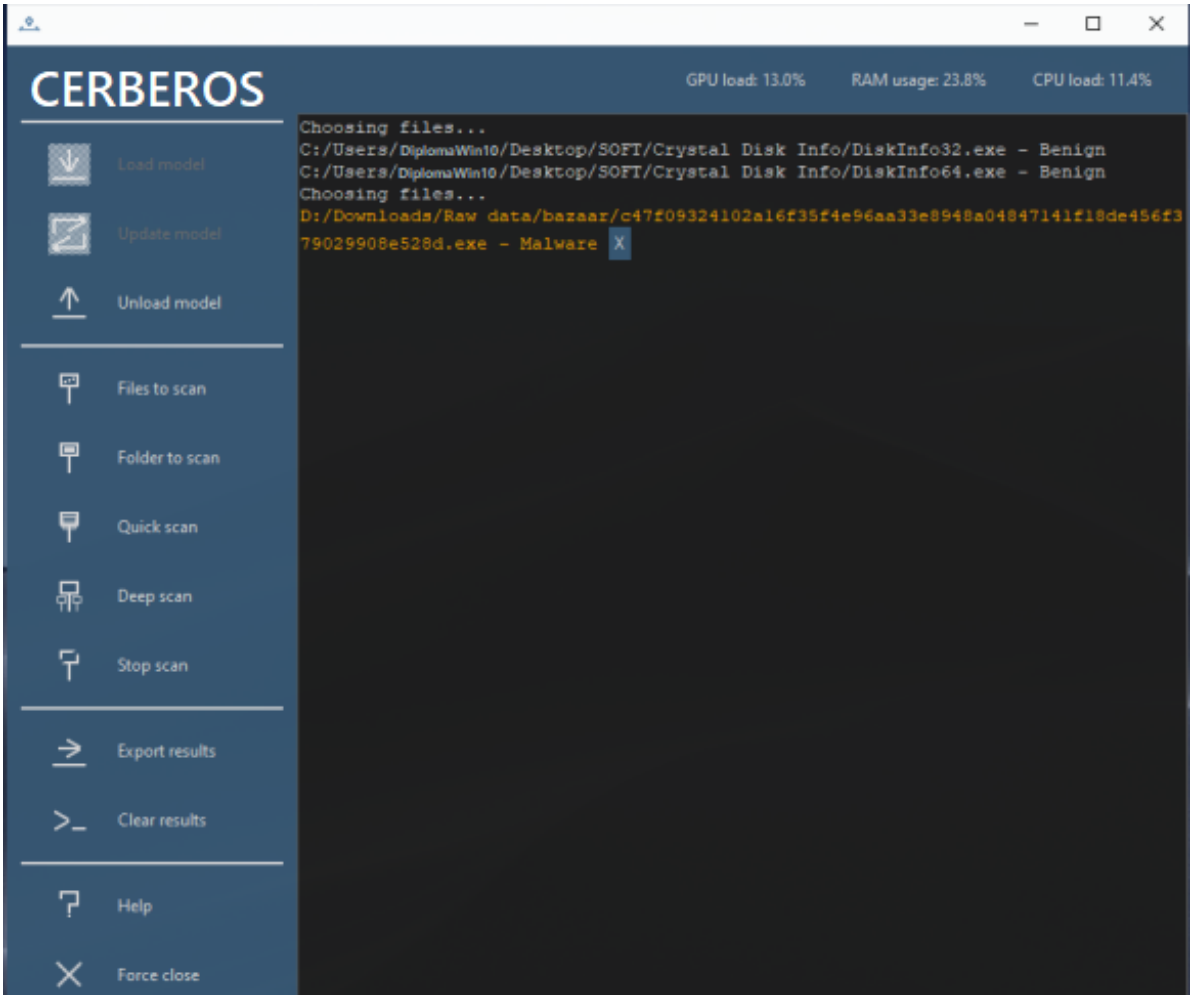


Рисунок 3.6 – Фінальний прототип інтерфейсу.

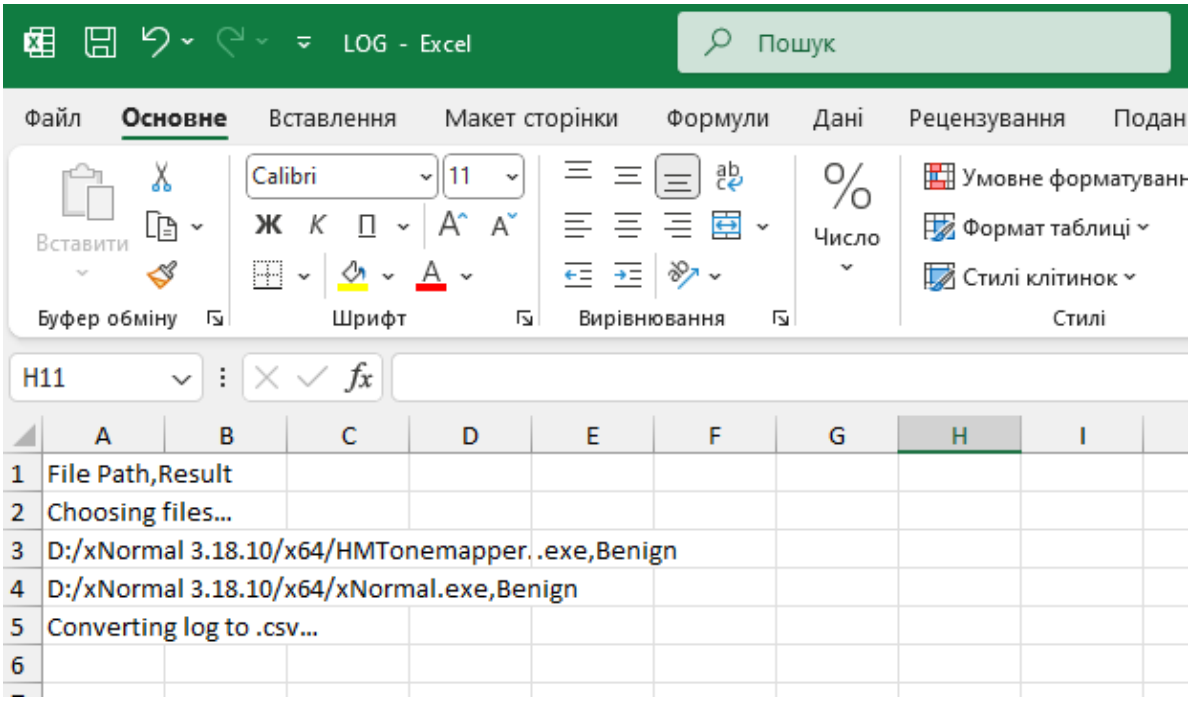


Рисунок 3.7 – Результати сканування в форматі .csv таблиці

Збережені в табличному форматі результати сканування корисні для ведення статистики хибнонегативних і хибнопозитивних результатів класифікації моделі при аналізі файлів на предмет наявності шкідливого коду.

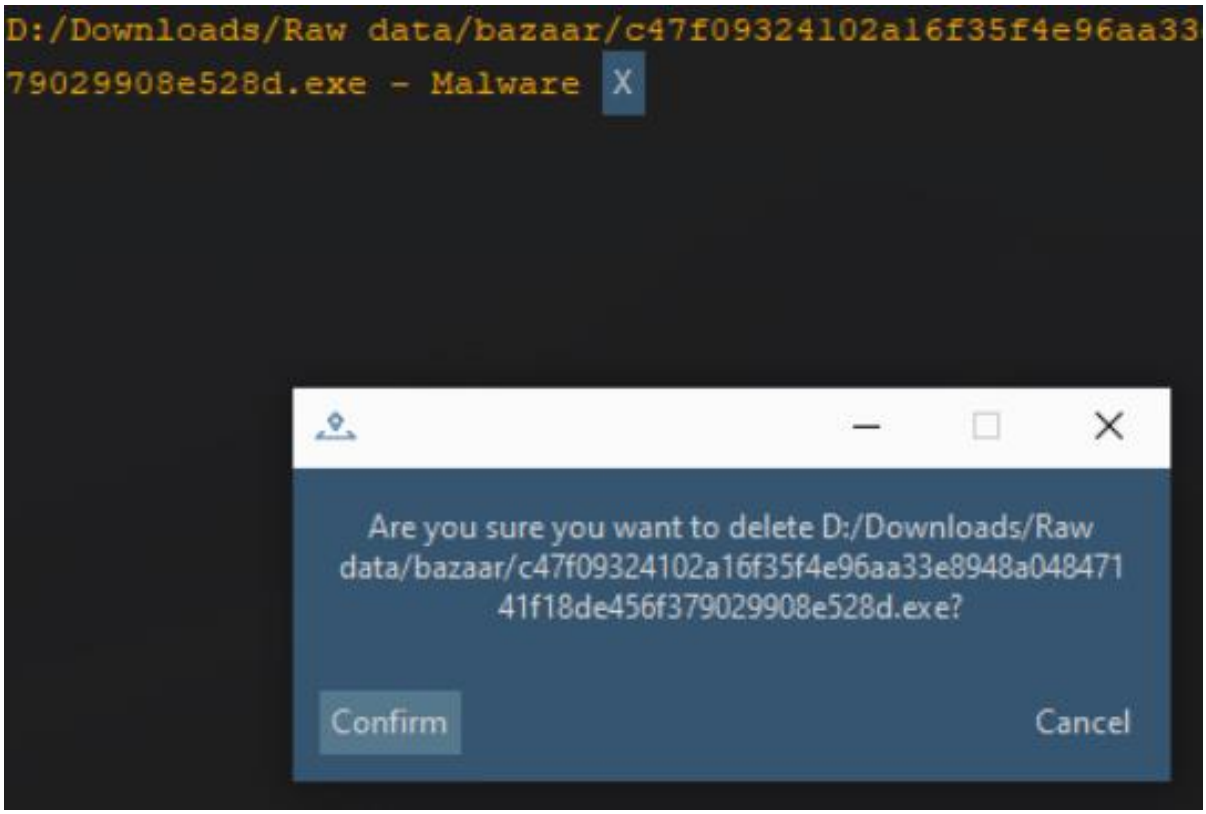


Рисунок 3.8 – Діалогове вікно з видаленням файлу

Діалогове вікно програми демонструє повний шлях (незалежно від кількості символів, розміри вікна автоматично підлаштовуються під інформацію, що виводиться), за яким знаходиться шкідливий файл. Якщо користувач підтверджує видалення – файл негайно видаляється з файлової системи, минаючи етап відправки в «кошик».

Як зображено на рисунку 3.9, розроблений графічний інтерфейс працює на двох мовах. Програма успішно обробляє латиницю та кирилицю, як на ввід так і на вивід.

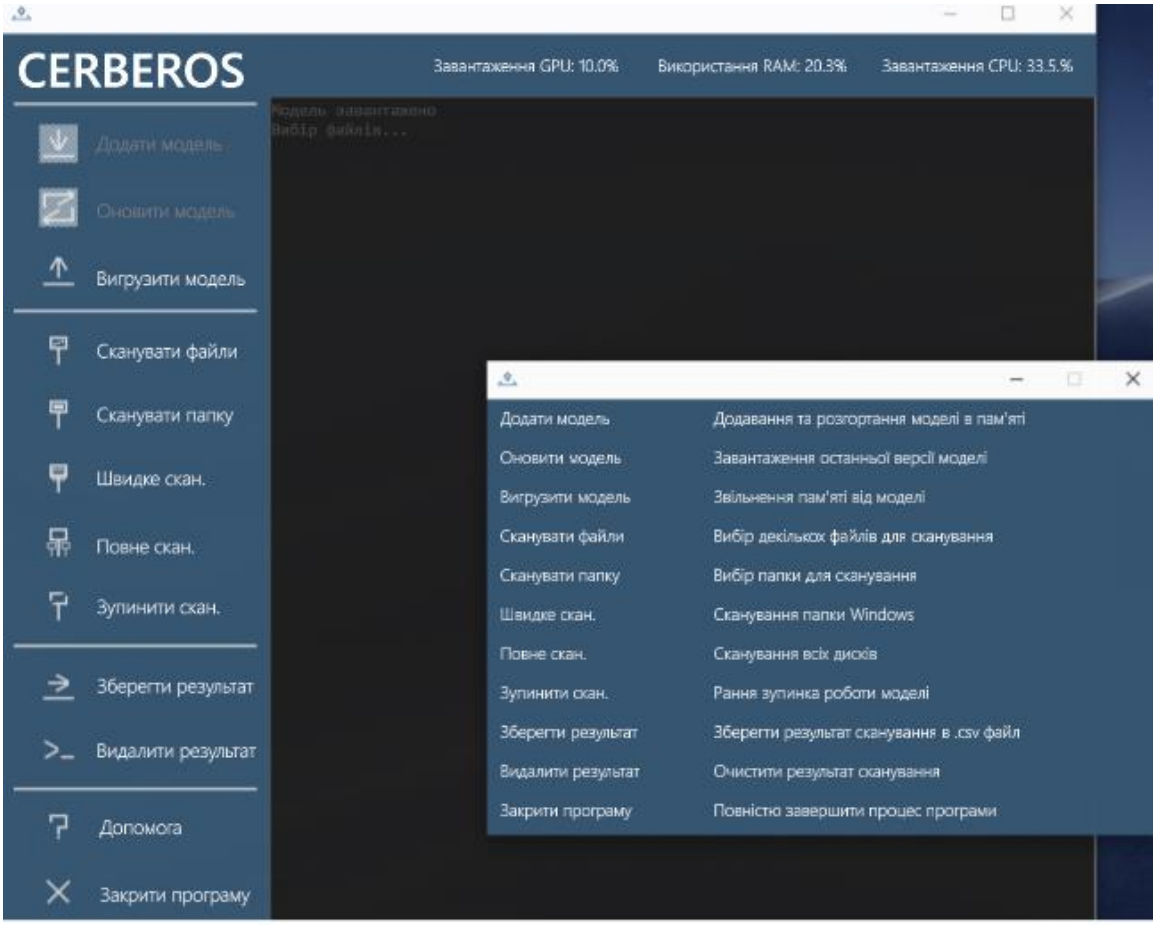


Рисунок 3.9 – Українська локалізація прототипу

Підтримувана розробленим прототипом багатозадачність і багатопотоковість, коректна обробка різних варіантів введення і виведення даних є головними аспектами для забезпечення якісного користувацького досвіду та комфорту під час роботи з програмою.

3.3 Тестування та налагодження програмного забезпечення

Як вже було зазначено раніше, метрики навченої моделі не є ідеальними, і лише подальше навчання дозволяє антивірусному рішенню на основі нейромережевої моделі робити менше хибнопозитивних і хибнонегативних висновків при здійсненні класифікації, практично будь-який програмний продукт без отримання оновлень нежиттєздатний.

Під час тестування було виявлено проблему, коли зразки однієї і тієї ж програми, написані для 32 і 64-бітних операційних систем, класифікувалися по-різному, приклад на рисунках 3.10 та 3.11

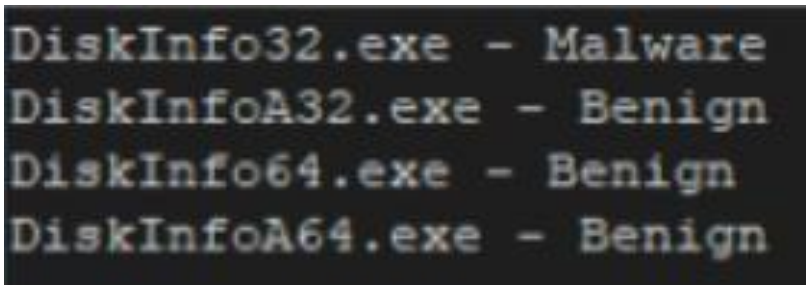


Рисунок 3.10 – Некоректний результат класифікації

Для вирішення даної проблеми було розроблено додатковий датасет, в якому серед класу «malware» переважали 64-бітові виконувані файли і динамічно підключаються бібліотеки, а для класу «benign» – 32-бітові. Модель була донавчена на додатковому датасеті вагою 5 ГБ, прототип «CERBEROS» отримало покращену версію.

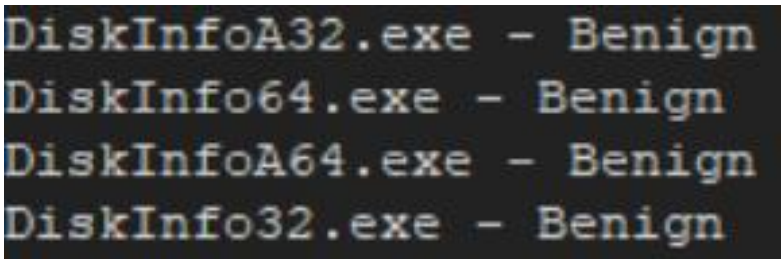


Рисунок 3.11 – Коректний результат класифікації

На момент написання цього розділу, критичних хибнопозитивних і хибнонегативних результатів класифікації не спостерігається, важливі елементи для коректного функціонування операційної системи антивірусним рішенням не зачіпаються.

Будь-який аналізований файл не приводиться до виконання, виключаючи можливість випадкового або навмисного зараження шкідливим кодом персонального пристрою користувача. Для останньої версії, щоб уникнути конфліктних моментів, автоматичне видалення знайдених шкідливих файлів було вимкнено, цей функціонал відтворюється в ручному режимі, користувач має повний контроль над процесом видалення, що повністю виключає ризик випадкового знищення важливої інформації.

Процес збору найбільш повного зворотного зв'язку в умовах різних версій операційної системи Windows і наповнення жорстких дисків здійснюється за допомогою залучення інших зацікавлених осіб. Необхідно провести подальші спостереження у форматі закритого тестування. Вищезгадане тестування не тільки дозволить зібрати цінні дані про поведінку моделі в реальних умовах, але й виявити потенційні вразливості, які можуть залишатися непоміченими за «лабораторних» умов.

3.4 Аналіз результатів функціонального тестування

Для перевірки ефективності розробленого рішення випадковим чином відібрано 100 раніше невідомих зразків програмного забезпечення, які були розподілені рівномірно між шкідливими та безпечними — по 50 екземплярів кожного типу. Такий підхід дозволив об’єктивно оцінити здатність моделі розпізнавати потенційні загрози у збалансованому середовищі.

Кожен піднабір містив по 25 зразків у форматі «.exe» та «.dll», що відповідає найбільш розповсюдженим форматам виконуваних файлів у Windows-середовищі. Це надало змогу перевірити поведінку моделі при взаємодії з різними типами вхідних даних. Кожна ітерація відрізнялася не лише обсягом знань, а й тривалістю

навчання, що дозволяє простежити взаємозв'язок між цими параметрами та показниками якості моделі. Результати тестування трьох ітерацій моделей на основі архітектури CERBEROS наведені в таблиці 3.3. Кожна ітерація відрізнялася не лише обсягом знань, а й тривалістю навчання, що дозволяє простежити взаємозв'язок між цими параметрами та показниками якості моделі.

Таблиця 3.3 – Результати тестування моделей

Версія прототипу	CERBEROS v1.0	CERBEROS v2.0	CERBEROS v2.5
Кількість хибнопозитивних рехультатів (з 50)	20	12	0
Кількість хибнопозитивних рехультатів	32	4	2
Точність класифікації	48%	84%	98%
Програмна основа	RoBERTa		
Апаратна основа	AMD Ryzen 5600G Nvidia GeForce RTX 2060 Super 16 Гб ОЗП		
Загальний об’єм знань	15 Гб	50 Гб	55 Гб
Час навчання	3 дні	5 днів	1 день

Перший прототип моделі, CERBEROS v1.0, було навчено на відносно невеликому наборі знань обсягом 15 ГБ. Тривалість навчання становила 3 дні. Показники точності виявилися найнижчими серед усіх трьох ітерацій, всього лише 48%, що свідчить про слабку здатність до класифікації. Кількість хибнопозитивних результатів становила 20 з 50 тестових об'єктів, що є значним показником помилкової ідентифікації безпечних файлів як шкідливих.

Ймовірною причиною таких низьких показників є обмежений обсяг

навчальних даних, недостатня кількість ітерацій оптимізації та, можливо, відсутність тонкого налаштування моделі.

У другій версії моделі (CERBEROS v2.0) було здійснено значне розширення навчальної вибірки – до 50 ГБ. Час навчання збільшився до 5 днів, що дало змогу здійснити глибше занурення у внутрішні зв'язки між характеристиками вхідних об'єктів. Це одразу відобразилось на якості: точність класифікації зросла до 84%, а кількість хибнопозитивних результатів зменшилася до 12 з 50. Такий результат демонструє позитивний ефект від розширення навчальної бази. Проте, все ще існує простір для вдосконалення, оскільки деякі безпечні зразки продовжують класифікуватися як потенційно шкідливі.

Останній прототип, CERBEROS v2.5, було навчено на найбільшій кількості даних що виконало навчання за найкоротший час, лише за 1 день. Така висока продуктивність стала можливою завдяки використанню оптимізованих методик попередньої обробки, прискореного навчання за допомогою апаратного прискорювача (Nvidia GeForce RTX 2060 Super), а також ретельному підбору гіперпараметрів.

Модель показала найвищу точність класифікації – 98%, а кількість хибнопозитивних результатів знизилась до 0 у першому підході з 50 об'єктів і лише до 2 у загальному тесті на 100 об'єктів. Це є безперечним свідченням того, що третя ітерація забезпечує найбільш надійні результати. Слід також відзначити, що програмною основою для всіх моделей є бібліотека RoBERTa, яка вже зарекомендувала себе в задачах обробки природної мови та текстової класифікації. Саме її було адаптовано під потреби системи виявлення загроз.

Загальний аналіз свідчить про те, що ефективність системи CERBEROS прямо залежить від обсягу знань, якості підготовлених даних та обчислювальних ресурсів, задіяних для навчання. Покращення апаратної складової, як і вдосконалення методів підготовки даних, відіграли ключову роль у підвищенні результативності моделі. Окрім того, важливим фактором стало джерело даних. Для формування шкідливої вибірки використовувався один з найактуальніших дампів з ресурсу MalwareBazaar, що дозволило досягти реалістичної та актуальної картини сучасних загроз.

В підсумку, версія CERBEROS v2.5 може бути рекомендована як стабільна і точна модель для реального розгортання у системах виявлення загроз, завдяки своїй високій продуктивності, мінімальній кількості хибнопозитивних результатів та короткому часу навчання.

3.5 Порівняльний аналіз із наявними системами виявлення шкідливого програмного забезпечення

Для об’єктивного оцінювання якості системи було проведено низку експериментів із застосуванням як відкритих, так і комерційних рішень у галузі виявлення шкідливого ПЗ. Усі рішення тестувалися на однаковому наборі з 100 зразків (50 шкідливих + 50 безпечних, рівномірно розподілених між .exe та .dll), що забезпечило коректне порівняння показників що й при порівнянні різних ітерацій навченої моделі, результати порівняння наведено в таблиці 3.4.

Таблиця 3.4 – Порівняння результатів класифікації в різних рішеннях

ПЗ	Malware Detection using machine	DeepLearning LSTM based malware	ESET Internet Security	CERBEROS
1	2	3	4	5
Кількість хибнопозитивних (з 50)	28	5	0	0
Кількість хибнонегативних (з 50)	2	8	4	2
Точність класифікації	70%	87%	96%	98%

Закінчення таблиці 3.4

1	2	3	4	5
Програмна основа	Random Forest	LSTM	Пропрієтарна	RoBERTa

Експериментальна перевірка порівнюваних рішень здійснювалася на уніфікованому масиві даних, що складався з рівних частин шкідливих і безпечних зразків у співвідношенні 50 на 50. Важливою передумовою було забезпечення репрезентативності вибірки: половину зразків становили файли з розширенням .exe, половину з розширенням .dll. Перед запуском класифікації всі файли пройшли стандартну стадію статичної обробки, під час якої з них витягувалися такі ознаки, як структури секцій, частотні розподіли n-грам інструкцій, виклики API та властивості заголовків PE.

Налаштування кожного тестового рішення виконувалося з урахуванням рекомендацій розробників і типових робочих параметрів. Для моделі на основі Random Forest було обрано ансамбль із ста дерев із максимальною глибиною тридцять рівнів; для LSTM-моделі реалізовано двошарову архітектуру з поступовим зменшенням кількості нейронів із 128 до 64 та регуляризацією через dropout. Комерційна система ESET Internet Security тестувалася у режимі «Максимального захисту» без змін у його внутрішніх сигнатурних базах чи евристичних правилах, як це зазвичай застосовують у реальному середовищі. Нарешті, для трансформерної моделі RoBERTa налаштування fine-tuning включало встановлення темпу навчання на рівні 2×10^{-5} і пакетний розмір у 16 зразків, що дозволило досягти оптимального балансу між швидкістю адаптації та стабільністю градієнтного спуску.

Усі випробування проводилися на ідентичному апаратному забезпеченні: процесор AMD Ryzen 5600G, відеокарта Nvidia GeForce RTX 2060 Super і 16 ГБ оперативної пам'яті, що виключало зміну обчислювальних ресурсів як фактор, що впливає на результат. Час навчання кожної моделі фіксувався автоматично, а для оцінки продуктивності класифікатора використовувалися лише три

фундаментальні метрики: кількість хибнопозитивних та хибнонегативних висновків, а також загальна точність класифікації. Такий підхід дозволив отримати максимально порівнянні показники, мінімізувавши вплив суб'єктивних налаштувань та апаратних обмежень.

Порівняння чотирьох моделей засвідчило різні стратегії роботи з ознаками шкідливості та їхню впливовість на кінцевий результат. Ансамблевий підхід Random Forest виявився досить оперативним у навчанні та класифікації, проте значна кількість хибнопозитивних результатів (28 із 50 безпечних файлів) вказує на надзвичайно консервативну поведінку моделі: вона прагне знизити ризик пропустити реальну загрозу за рахунок множинних помилкових спрацьовувань. ЛSTM-модель змогла набагато краще враховувати послідовні шаблони інструкцій, що зменшило хибнопозитивні випадки до 5, але водночас збільшило кількість хибнонегативних пропусків до 8 випадків, свідчачи про те, що рекурентна нейромережа іноді «не помічала» складно замасковані зразки.

Комерційне рішення ESET, яке спирається одночасно на обширну сигнатурну базу та розгалужені евристичні алгоритми, продемонструвало ідеальну відсутність хибнопозитивних висновків і лише чотири хибнонегативи, що забезпечило загальну точність 96 %. Це свідчить про високу якість довідкових баз та ефективність реалізації багаторівневого аналізу.

Найбільш збалансовані показники показав CERBEROS на основі RoBERTa: відсутність хибнопозитивних помилок і лише два пропущені шкідливі файли переклалися в рекордні 98 % точності. Використання трансформерної архітектури дозволяє одночасно оцінити як локальні залежності (через n-грамні ознаки), так і глобальні патерни у всьому файлі, що забезпечує стійкість навіть до поліморфних і рано нульового дня загроз.

До того ж, час класифікації одного файлу залишався на рівні десятків мілісекунд, що робить RoBERTa-рішення придатним для інтеграції в реальні системи захисту з високими вимогами до швидкодії.

Нижче в таблиці 3.5 наведено порівняння архітектур де вказано модель, підіхд та переваги і недоліки методу.

Таблиця 3.5 – Порівняння архітектур

Модель	Підхід	Переваги	Недоліки
Random Forest	Ансамбль рішень	Швидке навчання, інтерпретованість	Високі FP, обмежена гнучкість
LSTM	Рекурентна нейромережа	Здатність обробляти послідовності	Потребує багато даних, високий рівень FN
ESET Internet security	Сигнатурно-евристичний	Нуль FN, зріла технологія	Часте оновлення, висока вартість
CERBEROS	Трансформер	Баланс FN та FP, стійкість до поліморфізму	Потребує значних обчислювальних ресурсів

Проведене експериментальне порівняння різних програмних рішень для виявлення шкідливого програмного забезпечення дозволяє зробити низку глибоких висновків щодо ефективності використаних підходів, їхніх переваг і обмежень.

По-перше, стало очевидно, що традиційні методи машинного навчання, зокрема Random Forest, попри свою простоту, продовжують демонструвати досить конкурентоспроможні результати. Їхньою сильною стороною є здатність до швидкого навчання, стійкість до переобучення на великих наборах даних та інтерпретованість. Однак серйозною проблемою виявилася значна кількість хибнопозитивних спрацювань, що на практиці може призвести до блокування легітимного програмного забезпечення і, як наслідок, до зниження довіри користувачів до такої системи. У критичних інфраструктурах подібні помилки можуть мати серйозні наслідки, зокрема порушення стабільності роботи систем.

По-друге, рекурентні нейронні мережі типу LSTM продемонстрували значне зниження кількості хибнопозитивних випадків, що свідчить про їхню здатність аналізувати структури шкідливого коду в контексті. Проте їхній недолік — у порівняно високому рівні хибнонегативних помилок, що значно небезпечніше в

реальних умовах, де непомічене шкідливе ПЗ може завдати серйозної шкоди. Це пояснюється тим, що LSTM погано справляється з обробкою довгих залежностей у структурі виконуваних файлів, особливо коли ці структури штучно ускладнені або закамуфльовані.

Комерційне програмне забезпечення, таке як ESET Internet Security, підтвердило свою ефективність у широкому спектрі ситуацій. Використовуючи багаторівневу перевірку, велику сигнатурну базу та розвинену евристику, воно забезпечує надійний базовий рівень захисту. Водночас, такий тип рішень має обмеження: залежність від оновлень баз даних, повільне реагування на нові, невідомі типи шкідників (так звані загрози нульового дня), а також обмежена масштабованість у контексті нових архітектур чи середовищ виконання, наприклад, контейнерів або серверів без ОС.

У цьому контексті розроблена в межах даної кваліфікаційної роботи модель CERBEROS на основі RoBERTa постає як універсальне та адаптивне рішення. Застосування трансформерної архітектури дозволяє системі глибоко аналізувати як локальні, так і глобальні патерни у структурі виконуваних файлів. Її перевага — в здатності до генералізації, тобто розпізнавання нових типів загроз, з якими модель не зустрічалася раніше, на основі вивчених шаблонів поведінки.

Особливо варто підкреслити, що CERBEROS не потребує постійного ручного оновлення сигнатур або залежності від баз шкідливих зразків. Завдяки механізму донавчання (fine-tuning) на актуальних даних, система може оперативно адаптуватися до нових викликів кібербезпеки, зберігаючи високу точність. Практичні тести показали, що CERBEROS не лише досягає найвищої серед конкурентів точності (98 %), але й робить це при повній відсутності хибнопозитивних результатів — надзвичайно важливий аспект для зниження навантаження на ІТ-відділи та підтримання користувацького досвіду.

Підсумовуючи, можна стверджувати, що модель RoBERTa у контексті виявлення шкідливого ПЗ демонструє новий якісний рівень ефективності, що дає підстави для її подальшої інтеграції в реальні середовища, у тому числі в державному секторі, у хмарних платформах, корпоративних антивірусних рішеннях та навіть у розподілених системах безпеки. Успішна реалізація та

випробування прототипу CERBEROS підтверджують перспективність подальших досліджень у напрямі використання трансформерів у сфері кіберзахисту.

3.6 Напрям удосконалення системи та перспективи впровадження

Розроблений прототип антивірусного рішення, побудованого на основі нейромережевої враховуючи здатність шкідливого програмного забезпечення постійно змінюватися та пристосовуватись до нових умов, передбачається регулярне донавчання нейромережевої моделі. Для цього буде розроблено графік оновлень, що дозволить своєчасно адаптувати систему до нових загроз. Також буде доцільно продовжити роботу над оптимізацією графічного інтерфейсу користувача, зокрема в напрямку реалізації багатопотоковості при запуску та скануванні. Додавання більшої кількості мовних локалізацій для інтерфейсу, що зробить програму зручнішою для користувачів з різних країн та регіонів. Покращення інтерфейсу не лише підвищить користувацький досвід, а й сприятиме розширенню цільової аудиторії. Автоматичний запуск антивірусу при завантаженні операційної системи та режим постійного сканування. Проте перед впровадженням цих функцій необхідно провести дослідження впливу на загальну продуктивність системи, аби досягти оптимального балансу між рівнем захисту та навантаженням на ресурси комп’ютера. Навчання окремого модуля для аналізу інтернет-трафіку, який зможе блокувати шкідливі вебсайти. Цей модуль повинен підтримувати обробку зашифрованих з’єднань та мати гнучкі фільтри контенту, що дозволить підвищити безпеку користувачів під час вебсерфінгу. Окрім того, планується розширення переліку підтримуваних форматів файлів, адже різноманіття способів поширення ШПЗ постійно зростає. Ще одним важливим кроком стане впровадження функції карантину, де потенційно шкідливі файли можуть бути ізольовані до моменту прийняття остаточного рішення користувачем. Корисною також буде опція обмеження зони сканування за допомогою вказаних вручну шляхів до тек чи носіїв, користувач зможе сконцентрувати перевірку на конкретних ділянках файлової системи.

Інтеграція механізмів безперервного навчання моделі на основі нових прикладів як шкідливих, так і безпечних файлів, що дозволить підтримувати високу актуальність захисту у довгостроковій перспективі.

Ще одним важливим напрямом розвитку є розширення сумісності програмного забезпечення, зокрема адаптація антивірусу для інших платформ, що дасть змогу охопити ширше коло користувачів і пристроїв.

Для підвищення ефективності моделі та швидшої реакції на нові загрози передбачається налагодження співпраці з науковими установами, дослідницькими лабораторіями та комерційними компаніями. Обмін даними про загрози, а також участь у спільних експериментах дозволить забезпечити більш гнучке й актуальне оновлення моделей виявлення. Реалізація вищезазначених напрямів досліджень та розробки дозволить створити максимально надійне, адаптивне та сучасне антивірусне програмне забезпечення, здатне ефективно реагувати на виклики кіберпростору й забезпечувати високий рівень захисту в різних умовах. Особливу увагу необхідно приділяти стандартизації форматів обміну інформацією про загрози, що сприятиме підвищенню сумісності з існуючими системами захисту. Крім того, залучення експертного середовища дає змогу проводити поглиблену валідацію моделей і прискорює інтеграцію найновіших наукових досягнень у практичну реалізацію. Також важливим є створення спільної платформи для тестування нових рішень у реальних умовах, що дозволить виявляти слабкі місця ще на етапі розробки. У довгостроковій перспективі така кооперація сприятиме формуванню стійкої екосистеми кібербезпеки, здатної швидко адаптуватися до змін ландшафту загроз. У межах зазначених ініціатив доцільно також впроваджувати механізми безперервного навчання систем штучного інтелекту на основі актуальних даних, отриманих із реального середовища. Це дозволить системам виявлення загроз залишатися релевантними навіть у динамічно змінюваному інформаційному просторі. Особливої ваги набуває автоматизація процесів аналізу та класифікації потенційних атак із мінімальним залученням людських ресурсів, що значно скорочує час реагування та знижує ймовірність помилок.

ВИСНОВКИ

В цій кваліфікаційній роботі було успішно реалізовано розробку прототипу антивірусного програмного забезпечення, заснованого на сучасних методах машинного навчання, зокрема нейронних мережах трансформерного типу.

Результати експериментального дослідження підтвердили практичну доцільність використання глибинних мовних моделей у задачах виявлення шкідливого програмного забезпечення, що виводить запропоноване рішення на новий рівень ефективності.

На основі зібраного датасету, який включав як шкідливі, так і безпечні зразки у форматах .exe та .dll, було проведено навчання та тестування моделі, а також реалізовано зручний користувацький інтерфейс. Створена система показала високу точність класифікації та низький рівень хибнопозитивних і хибнонегативних спрацювань, що особливо важливо в умовах реального використання. Проведений порівняльний аналіз з альтернативними рішеннями на основі алгоритмів Random Forest, LSTM та комерційних пропрієтарних продуктів підтвердив перевагу запропонованої архітектури.

Завдяки комплексному підходу від збору даних до реалізації повноцінного прототипу, вдалося створити стабільно функціонуюче антивірусне рішення, здатне до масштабування та подальшого розвитку. Зокрема, було визначено перспективні напрями модернізації: оптимізація інтерфейсу, впровадження функцій автозапуску, безперервного сканування, розширення функціоналу аналізу мережевого трафіку та реалізація підтримки нових платформ.

Таким чином, виконана робота не лише засвідчила високий потенціал нейромереж у боротьбі з кіберзагрозами, а й стала міцним підґрунтям для подальших досліджень у цій галузі. Запропонований підхід може бути адаптований до різних середовищ та інтегрований у практичні системи інформаційної безпеки, сприяючи підвищенню стійкості цифрової інфраструктури до сучасних загроз.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Передові технологій ESET. ESET Progress. Protected. URL:: https://www.eset.com/ua/about/technology/ (дата звернення: 20.02.2025).

2. What is ransomware. McAfee. ULR:: https://www.mcafee.com/learn/ransomware/ (дата звернення: 20.02.2025).

3. Шнит Р. В. Троянські програми у сучасному інформаційному просторі. Розвиток сучасної науки та освіти: реалії, проблеми якості, інновації: матеріали III науково-практичної інтернет-конференції, м. Луцьк, 2023. С. 515-521.

4. Що таке шкідливе програмне забезпечення? Microsoft. ULR:: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-malware (дата звернення: 20.02.2025).

5. Ващук В. В. Шкідливе програмне забезпечення та основні його категорії. Розвиток сучасної науки та освіти: реалії, проблеми якості, інновації: матеріали III науково-практичної інтернет-конференції, м. Луцьк, 2023. С. 196-200.

6. Шамонін К. Є. Інформаційна технологія проєктування системи аналізу та реагування на програми-вимагачі : робота на здобуття кваліфікаційного ступеня магістра : спец. 122 – комп’ютерні науки / наук. кер. В. К. Ободяк. Суми : Сумський державний університет, 2022. 80 с.

7. Що таке шкідливе програмне забезпечення? Якої шкоди воно може завдати комп’ютеру чи смартфону? Державна служба спеціального зв’язку та захисту інформації України. ULR: https://cip.gov.ua/ua/faqs/sho-take-shkidlive-programne-zabezpechennya-yakoyi-shkodi-vono-mozhe-zavdati-komp-yuteru-chi-smartfonu (дата звернення: 21.02.2025).

8. Як працює антивірус – технологія за лаштунками. Mediacom світ новин. ULR: https://mediacom.com.ua/yak-pratsyue-antivirus-texnologiya-za-lashtunkami/ (дата звернення: 21.02.2025).

9. Діденко М. С. Евристичний аналіз як метод виявлення небажаного програмного забезпечення. Проблеми інформатизації: матеріали XI міжнародної науково-технічної конференції. Харків, 2023. С. 26.

10. Носов В. С. Методи і засоби автогенерації та аналізу вихідного коду

програмного забезпечення : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 123 Комп'ютерна інженерія / В. С. Носов ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2022. – 75 с.

11. Васильків Д. С. Алгоритми виявлення шкідливого програмного забезпечення на основі гешу чутливого до локалізації = Locale-Sensitive HashBased Malware Detection Algorithms : кваліфікаційна робота : спец. 125 – кібербезпека та захист інформації освітньо-професійна програма – кібербезпека / Дмитро Сергійович Васильків ; наук. керівник к.т.н., доц. Н. Г. Яцків. Тернопіль : ЗУНУ, 2024. 83 с.

12. Руденко А. С. Система по формуванню бази сигнатур комп’ютерних вірусів : магістерська дис. : 151 Автоматизація та комп’ютерно-інтегровані технології / Руденко Андрій Сергійович. – Київ, 2023. – 109 с.

13. Григор’єва, О. О. Технології протидії виявленню віртуалізації шкідливим ПЗ : дипломний проект … бакалавра : 125 Кібербезпека / Григор’єва Ольга Олександрівна. – Київ, 2022. – 72 с.

14. Anderson Jonathan. A comparison of Unix sandboxing techniques. FreeBSD Journal, 2017, 8-12.

15. Мєшков В. (2023). Аналіз систем інтелектуального моніторингу трафіку комп’ютерної мережі для систем виявлення атак. Information Technology: Computer Science, Software Engineering and Cyber Security, 1, 85–92, doi: https://doi.org/10.32782/IT/2023-1-11 (дата звернення: 04.03.2025).

16. Педяш В., Ледовський Є., Ткач В. Сучасні методи виявлення шкідливого програмного забезпечення. Вісник Хмельницького національного університету. Серія: Технічні науки. 2024. № 333(2). С. 389-392. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/164 (дата звернення: 05.03.2025).

17. Гончар С. В., Крапивний І. С. Кібергігієна: основи безпечної поведінки в цифровому просторі. Цивільний захист в умовах війни : збірник тез доповідей І Міжнародної науковопрактичної конференції, м. Львів, 17-18 квітня 2025 року. Львів: ЛДУ БЖД, 2025. 296 с.

18. Масліченко С. В. Розробка автоматизованої системи кіберзахисту на підприємстві : кваліф. робота магістра зі спеціальності 122 «Комп'ютерні науки» / С.В. Масліченко. – Полтава : Нац. ун-т ім. Юрія Кондратюка, 2025. – 65 с.

19. Романько В. В. (2024). Захист конфіденційної інформації на основі багаторівневого аналізу. У Т. О. Жирова (Ред.), Програмування та захист інформації: Збірник наукових статей студентів. Ч. 2. (с. 151–157). Державний торговельно-економічний університет.

20. Abid Noman. Advancements and Best Practices in Data Loss Prevention: A Comprehensive Review. Global Journal of Universal Studies, 1.1: 190-225.

21. Z. Aradi, A. Bánáti. The Role of Honeypots in Modern Cybersecurity Strategies. 2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI), Stará Lesná, Slovakia, 2025, pp. 000189-000196, doi: 10.1109/SAMI63904.2025.10883300 (дата звернення: 11.03.2025).

22. Селезько Р. Аналіз та Організація Big Data. Інформаційні технології та комп'ютерне моделювання: матеріали статей міжнародної науково-практичної конференції, м. Івано-Франківськ, 21-24 травня 2024 року. – Івано-Франківськ: п. Голіней О.М., 2024. – С 124 с.

23. LI Shan, IQBAL, Muddesar, SAXENA, Neetesh. Future industry internet of things with zero-trust security. Information Systems Frontiers, 2024, 26.5: 1653-1666.

24. Олійник Я., Платоненко А., Черевик В., Ворохоб М., Шевчук Ю. (2025). Методи захисту інформації в технологіях IoT. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(27), 100–108. https://doi.org/10.28925/2663-4023.2025.27.705 (дата звернення: 14.03.2025).

25. Шульга В. П., Іванченко Є. В., Вишневська Н. С., Бербер А. С. (2024). Дослідження методів та моделей оцінювання кіберзахисту критичної інфраструктури держави. Сучасний захист інформації, 3(59), 6–19. https://doi.org/10.31673/2409- 7292.2024.030001 (дата звернення: 15.03.2025).

26. Wa Nkongolo, Mike, Tokmak, Mahmut. (2024). Ransomware Detection Using Stacked Autoencoder for Feature Selection. Indonesian Journal of Electrical Engineering and Informatics (IJEEI). 12. 142-170. doi: 10.52549/ijeei.v12i1.5109 (дата звернення: 15.03.2025).

27. Nkongolo, Mike, Tokmak, Mahmut. Ransomware detection using stacked autoencoder for feature selection. arXiv preprint arXiv:2402.11342, 2024.

28. Malhotra Vrinda (2022). Graph Neural Networks for Malware Classification. Master's Projects. 1194. doi: https://doi.org/10.31979/etd.znr4-vz7n (дата звернення: 16.03.2025).

29. D. Trizna, L. Demetrio, B. Biggio, F. Roli. Nebula: Self-Attention for Dynamic Malware Analysis. IEEE Transactions on Information Forensics and Security, vol. 19, pp. 6155-6167, 2024, doi: 10.1109/TIFS.2024.3409083 (дата звернення: 16.03.2025).

30. Терейковський, І. А. Методи розпізнавання кібератак: розпізнавання комп'ютерних вірусів: навчальний посібник для здобувачів ступеня бакалавр за освітньою програмою «Системне програмування та спеціалізовані комп'ютерні системи» спеціальності 123 Комп'ютерна інженерія / І. А. Терейковський, О. Г. Корченко, В. В. Погорелов, Київ : КПІ ім. Ігоря Сікорського, 2022. – 127 с.

31. Основи роботи з Python. Python. URL: https://docs.python.org/uk/3.13/tutorial/index.html (дата звернення: 19.03.2025).

32. Використання інтерпретатора Python. Python. URL: https://docs.python.org/uk/3.13/tutorial/interpreter.html (дата звернення: 19.03.2025).

33. Haviv Adi, Berant Jonathan, Globerson Amir. BERTese: Learning to speak to BERT. arXiv preprint arXiv:2103.05327, 2021.

34. Коротка екскурсія Стандартною бібліотекою. Python. URL: https://docs.python.org/uk/3.13/tutorial/stdlib.html (дата звернення: 19.03.2025).

35. DikeDataset. GitHub. URL: https://github.com/iosifache/DikeDataset (дата звернення: 15.04.2025).

36. Python vs Other Programming Languages in 2024: Detailed Comparison. Uvik. URL: https://uvik.net/blog/how-python-is-different-from-other-languages/ (дата звернення: 16.04.2025).

37. What is Python used for? Exploring 7 practical Python uses. Vakoms. URL: https://vakoms.com/blog/what-is-python-used-for-exploring-7-practical-python-uses/ (дата звернення: 16.04.2025).

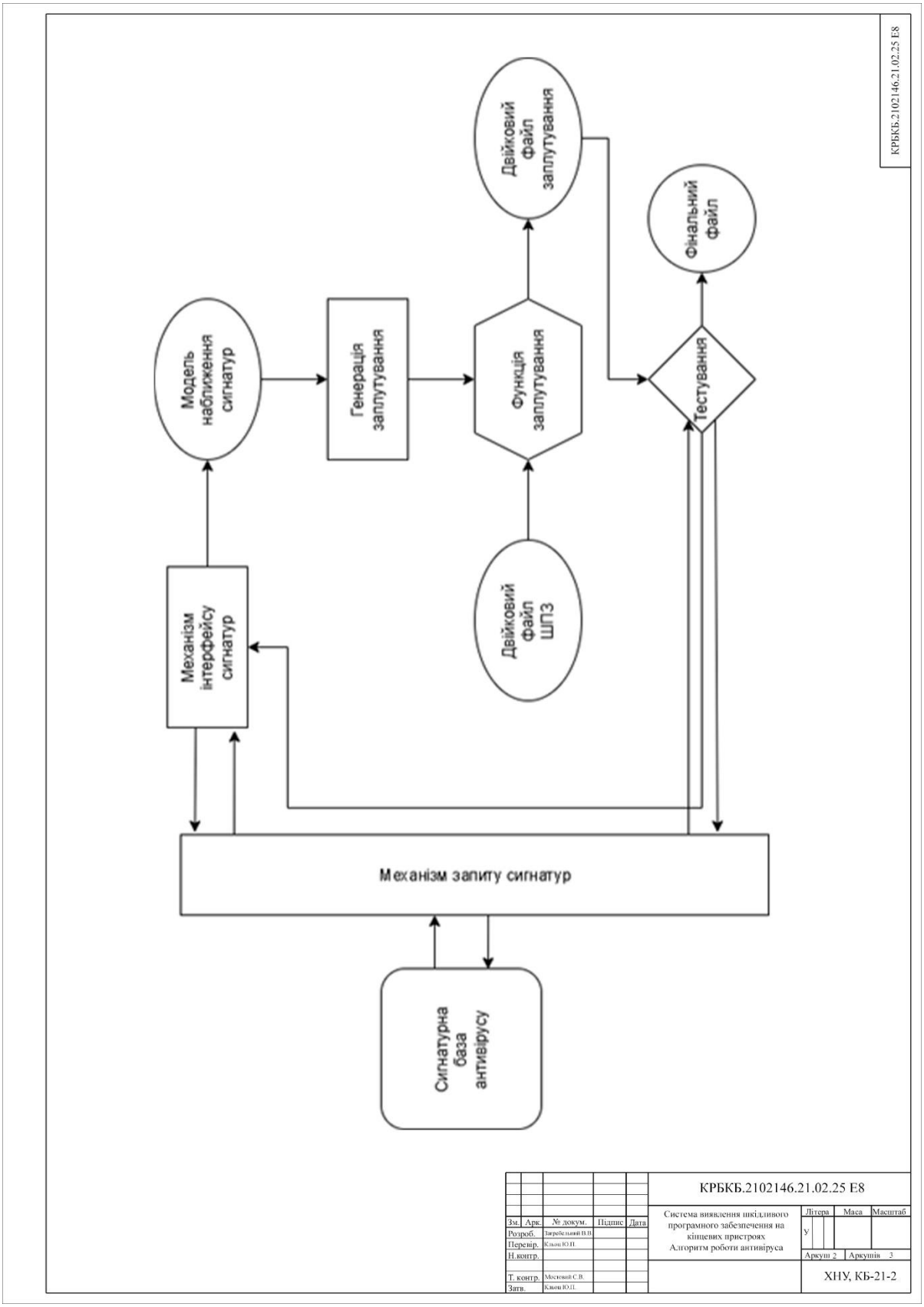
38. Comparing Python to Other Languages. Python URL: https://www.python.org/doc/essays/comparisons/ (дата звернення: 16.04.2025).

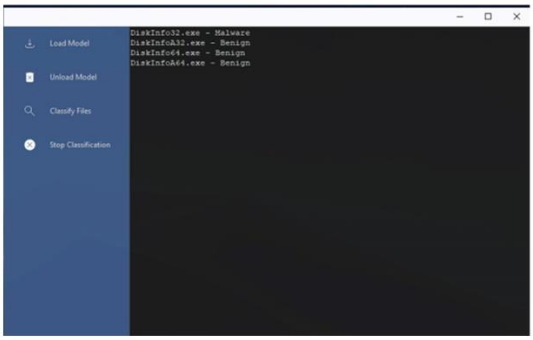
39. MalwareBazaar. MalwareBazaar from Abuse. URL: https://bazaar.abuse.ch/ (дата звернення: 13.05.2025).

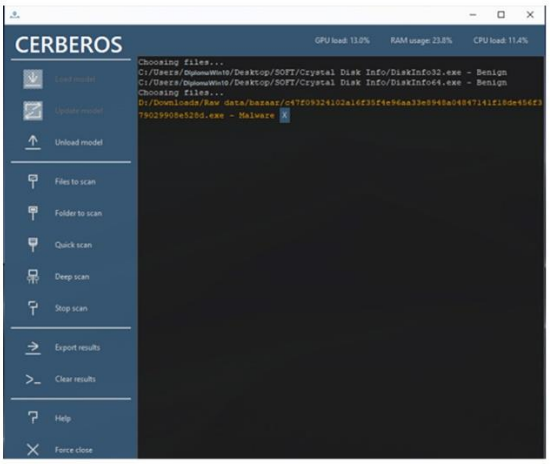
40. GitHub Advisory Database. GitHub. URL: https://github.com/advisories?query=type%3Amalware (дата звернення: 15.05.2025).

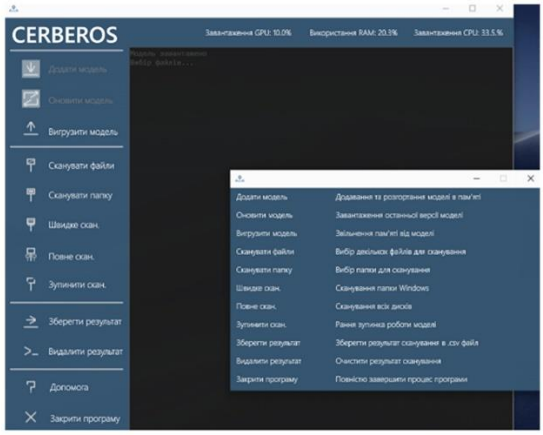
ДОДАТОК А

Копії графічної частини









					КРБКБ.2102146.21.02.25 Е8			
Зм.	Арк.	№ докум.	Підпис	Дата	Система виявлення шкідливого програмного забезпечення на кінцевих пристроях Графічний інтерфейс	Літера	Маса	Масштаб
Розроб.		Загребельний В.В.				У		
Перевір.		Ключ Ю.П.				Аркуш 2	Аркушів 3	
Н.контр.						ХНУ, КБ-21-2		
Т. контр.		Мостовий С.В.						
Затв.		Ключ Ю.П.						

ПЗ	Malware Detection using machine	DeepLearning LSTM based malware	ESET Internet Security	CERBEROS
Кількість хибнопозитивних (з 50)	28	5	0	0
Кількість хибнонегативних (з 50)	2	8	4	2
Точність класифікації	70%	87%	96%	98%
Програмна основа	Random Forest	LSTM	Пропрієтарна	RoBERTa

Версія прототипу	CERBEROS v1.0	CERBEROS v2.0	CERBEROS v2.5
Кількість хибнопозитивних рекультатів (з 50)	20	12	0
Кількість хибнопозитивних рекультатів	32	4	2
Точність класифікації	48%	84%	98%
Програмна основа	RoBERTa		
Апаратна основа	AMD Ryzen 5600G Nvidia GeForce RTX 2060 Super 16 Гб ОЗП		
Загальний об'єм знань	15 Гб	50 Гб	55 Гб
Час навчання	3 дні	5 днів	1 день

Модель	Підхід	Переваги	Недоліки
Random Forest	Ансамбль рішень	Швидке навчання, інтерпретованість	Високі FP, обмежена гнучкість
LSTM	Рекурентна нейромережа	Здатність обробляти послідовності, хороша адаптація до нових даних	Потребує багато даних, високий рівень FN
ESET Internet Security	Сигнатурно-евристичний	Нуль FP, зріла технологія	Пропрієтарна, необхідність частого оновлення, висока вартість
CERBEROS (RoBERTa)	Трансформер	Баланс FP/FN, гарний час класифікації, стійкість до поліморфізму	Потребує fine-tuning та значних обчислювальних ресурсів

Метрика	Безпечне ПЗ	ШПЗ
Precision	0.93-0.94	0.99 – 1.00
Recall	0.98 – 0.99	0.98
F_1 Score	0.96	0.99

Зм.	Арк.	№ докум.	Підпис	Дата	КРБКБ.2102146.21.02.25 Е8		
Розроб.		Загребельний В.В.			Система виявлення шкідливого програмного забезпечення на кінцевих пристроях Таблиці значень і порівнянь	Літера	Маса	Масштаб
Перевір.		Ключ Ю.І.				У		
Н.контр.						Аркуш 2	Аркушів 3	
Т. контр.		Мостовой С.В.				ХНУ, КБ-21-2		
Затв.		Ключ Ю.І.						

ДОДАТОК Б

Список публікацій

Міністерство освіти і науки України,
ДНУ «Інститут модернізації змісту освіти»,
Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського»,
Інститут кібернетики ім. В.М. Глушкова НАН України,
Інститут телекомунікацій і глобального інформаційного простору НАН України,
Інститут цифровізації освіти НАПН України,
Житомирський державний університет імені Івана Франка,
Житомирський військовий інститут імені С.П. Корольова,
Черкаський державний технологічний університет,
Вінницький національний технічний університет,
Опольська політехніка (Республіка Польща),
Варшавський технологічний університет (Республіка Польща),
Технологічний університет Лулео (Королівство Швеція),
Технічний університет (Чеська Республіка),
Технічний університет (Республіка Болгарія),
Університет країни Басків (Королівство Іспанія),
Віденський технічний університет (Республіка Австрія),
ADA University (Азербайджан)

ТЕЗИ ДОПОВІДЕЙ

XV Міжнародної науково-технічної конференції

Інформаційно-комп'ютерні технології

м. Житомир, 28-29 березня 2025 р.

Житомир
2025

УДК 004.49

Загребельний В.В., здобувач
Кльоц Ю.П., к.т.н., доцент
Хмельницький національний університет

ВИЯВЛЕННЯ МАЙНЕРІВ ЗА ДОПОМОГОЮ АНТИВІРУСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виявлення вірусів-майнерів є доволі складним завданням для сучасного антивірусного програмного забезпечення (далі – АВПЗ). Майнери використовують ресурси комп'ютера для видобутку криптовалюти, дуже часто діють приховано, мінімізуючи всі видимі ознаки своєї активності. Незважаючи на те що, АВПЗ постійно оновлюється, існують суттєві недоліки у виявленні та нейтралізації ними вірусів-майнерів. Недоліки часто зумовлені такими факторами як: постійна еволюція шкідливого коду, маскуванні та адаптації вірусів, обходом евристичного аналізу та постійне використання нових методів обходу систем захисту [1, 4].

Однією з проблем виявлення майнерів, є їх здатність до маскування. Майнери часто імітують системні процеси, що значно ускладнює їх виявлення та ідентифікацію. Крім того, такі віруси постійно еволюціонують адаптуючись до нових методів виявлення. Це означає, що антивірусні бази швидко стають застарілими [1].

Ще одним викликом є дуже низька помітність майнерів. На відміну від відомих вірусів, які діють активно, майнери можуть діяти тихо, мінімізуючи використання ресурсів комп'ютера. Це дозволяє майнерам залишатися непоміченими протягом доволі тривалого часу, особливо на потужних комп'ютерах, де невелике навантаження яке спричиняє майнер, може бути непомітним [2].

Окрему проблему також становить криптоджекінг. Криптоджекінг використовує JavaScript для майнингу в браузері. Антивіруси дуже часто не здатні відрізнити легітимний код від шкідливого, так як він використовується в межах веб-сайтів. Таким чином криптоджекінг несе особливу загрозу, вражаючи користувачів які відвідують навіть довірені та відомі ресурси [2].

Ефективність АВПЗ часто обмежена ресурсами комп'ютера. Постійне сканування на наявність нових загроз значно вповільнює роботу системи, тому АВПЗ постійно шукають баланс між безпекою та продуктивністю. В результаті, майнери, що застосовують технології маскування, можуть залишатися непоміченими, якщо АВПЗ не проводить глибокий аналіз [3, 4].

Ефективне вирішення проблеми потребує комплексного підходу до проблеми. Вдосконалення евристичного аналізу з використанням

штучного інтелекту та машинного навчання, дозволить більш точно виявляти аномальну поведінку. Посилення захисту браузерів з використанням спеціалізованих розширень та обмеження виконання ненадійних та не легітимних скриптів є критично важливим кроком в боротьбі з криптоджекінгом та мінімізують зараження вірусом-майнером. Постійний моніторинг мережевого трафіку для виявлення підозрілих пул-з'єднань майнінгу, та постійний моніторинг з контролем використання ресурсів комп'ютера, зокрема відеокарти та процесора. Системи моніторингу які надають інформацію в режимі реального часу та надсилають сповіщення в разі аномального збільшення використання ресурсу комп'ютера, дозволять швидко виявляти та реагувати на майнер [5, 6].

У підсумку, боротьба з вірусами-майнерами вимагає постійного вдосконалення АВПЗ та впровадження інноваційних рішень для протидії таким загрозам. Головними аспектами успішного виявлення та нейтралізації майнера є не лише вдосконалення АВПЗ, але й активний підхід, який включає в себе моніторинг, аналіз поведінки та різні превентивні заходи безпеки. Лише багатошаровий та комплексний захист, який враховує різноманітні вектори атаки та зараження вірусом, здатен забезпечити ефективний захист від прихованої загрози вірусів-майнерів.

Список використаних джерел:

1. Незаконний майнінг. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/nezakonnyy-mayning/. (дата звернення: 10.03.2025).

2. Прихований майнінг. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/skrytyy-mayning/. (дата звернення: 10.03.2025).

3. Захист від вірусів. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/zashchita-ot-virusov/. (дата звернення: 10.03.2025).

4. Шкідливі програми. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/vredonosnyye-programmy/. (дата звернення: 10.03.2025).

5. Як знайти та видалити вірус-майнер з комп'ютера. URL: https://www.binance.com/uk-UA/square/post/97665. (дата звернення: 10.03.2025).

6. Вас "майнять": як виявити і знешкодити прихований майнінг. URL: https://epravda.com.ua/publications/2018/04/20/636181/. (дата звернення: 10.03.2025).

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XVI Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2024»

15-16 листопада 2024

Хмельницький 2024

УДК 004.77

Загребельний В.В., Кльоц Ю.П.

Хмельницький національний університет

РОЛЬ OSINT У ПРОТИДІЇ КІБЕРЗЛОЧИННОСТІ ТА ВЕДЕННІ ІНФОРМАЦІЙНОЇ ВІЙНИ

Розглядається роль розвідки з відкритих джерел у боротьбі з кіберзлочинністю та веденні інформаційної війни в умовах збройного конфлікту. Аналізується, як технології OSINT дозволяють збирати, обробляти та інтерпретувати дані з відкритих джерел для ідентифікації кіберзлочинців, відстеження їхньої діяльності та виявлення загроз для інформаційної безпеки. Окрім переваг, розглянуто ризики, пов'язані з конфіденційністю та етичними аспектами використання відкритих даних.

The role of intelligence from open sources in the fight against cybercrime and conducting information warfare in the conditions of armed conflict is considered. Analyze how these technologies allow the collection, processing, and interpretation of data from open sources to identify cybercriminals, track their activities, and identify threats to information security. In addition to the benefits, the risks associated with privacy and ethical aspects of using open data are considered.

Open Source Intelligence (OSINT), або розвідка з відкритих джерел [1], стає все більш важливою частиною сучасного інформаційного суспільства. Її популярність і значення особливо зросли в контексті інформаційної війни та боротьби з кіберзлочинністю, яка набула актуальності у зв'язку з нинішньою війною в Україні. OSINT надає можливість отримувати і використовувати інформацію з відкритих джерел – таких як соціальні мережі, блоги, медіаплатформи, урядові та комерційні сайти, новинні ресурси. Окрім цього, OSINT є економічним засобом розвідки, оскільки дозволяє уникнути витрат на спеціалізоване обладнання та залучення спеціальних агентів. Це дозволяє застосовувати OSINT не лише військовими чи правоохоронними органами, а й компаніями, журналістами-розслідувачами, дослідниками та громадськими організаціями. Однак важливо зазначити, що ця доступність є одночасно перевагою і потенційною загрозою для конфіденційності, адже велика кількість персональної інформації стає доступною та вразливою до маніпуляцій. Сутність OSINT базується на тому, що інформація, доступна у відкритому доступі, є цінним джерелом знань, яке дозволяє створювати обґрунтовані висновки для використання в різних сферах.

Метою роботи є аналіз ролі та можливостей OSINT у протидії кіберзлочинності та веденні інформаційної війни в умовах збройного конфлікту, а також оцінка потенційних ризиків для конфіденційності та етичні аспекти використання відкритих даних.

OSINT широко застосовується у боротьбі з кіберзлочинністю. В сучасному цифровому світі значна частина злочинних дій здійснюється в онлайн-просторі, а тому відстеження та деанонімізація кіберзлочинців стає важливою для забезпечення безпеки [2]. Завдяки OSINT є можливість виявляти ідеологію та цілі кіберзлочинців, визначати їхні мережі та способи роботи, а також виявляти шляхи пересування незаконних коштів, що використовується у кіберзлочинній діяльності. Деанонімізація кіберзлочинців полягає в тому, що аналітики можуть зібрати інформацію з публічно доступних джерел, що включає не тільки особисті дані кіберзлочинців, але й їхню активність у мережі, геолокаційні дані, зв'язки, фінансові операції та інші важливі аспекти. Наприклад, дані з соціальних мереж, навіть якщо вони були призначені для обмеженого кола людей, можуть допомогти у створенні детального профілю кіберзлочинця, включаючи місце його роботи, навчання, коло спілкування та інші особисті деталі. Це може допомогти правоохоронним органам швидше ідентифікувати злочинців, їхні мережі та способи дій. Зокрема, використання OSINT дозволяє не лише відстежити дії злочинців, але й вчасно передбачити потенційні загрози кібербезпеці.

Алгоритм роботи OSINT має кілька послідовних етапів. Спочатку відбувається етап планування та управління, коли визначаються джерела інформації, що будуть використовуватися для аналізу, формуються критерії пошуку і плануються методи роботи з отриманими даними. Наступним кроком є збір інформації. Він включає систематичний пошук даних у відкритих джерелах, що може здійснюватися як вручну, так і автоматизовано за допомогою спеціальних програм. Збір даних відбувається з соціальних мереж, форумів, онлайн-карт, державних реєстрів та інших відкритих джерел. Після цього починається процес обробки та експлуатації даних. Це етап, на якому зібрані дані проходять обробку, структурування та перевірку на достовірність. Аналіз отриманої інформації дозволяє аналітикам виділити ключові аспекти, які можуть бути корисними для прийняття рішень або для підготовки звітів. Наприклад, аналіз геолокаційних даних може показати точне місцезнаходження підозрюваного в певний момент часу, що може бути використано для обґрунтування доказів у судових процесах. Завершальним етапом є поширення результатів дослідження, коли підготовлені аналітичні звіти передаються зацікавленим сторонам з конкретними рекомендаціями щодо подальших дій.

Попри свої численні переваги, OSINT також має певні обмеження, які вимагають вдосконалення як на технічному, так і на законодавчому рівнях. У першу

чергу, йдеться про питання конфіденційності та етичні аспекти. Використання персональної інформації, навіть якщо вона доступна у відкритих джерелах, вимагає дотримання певних правил та обмежень, щоб уникнути порушення прав людини та неправомірного втручання в особисте життя. Цифрова безпека та захист конфіденційності стають важливими в умовах постійного доступу до великого обсягу особистих даних. У цьому контексті важливо розробляти нові методи захисту інформації, що дозволяють уникнути зловживань, таких як шантаж чи маніпуляції. Крім того, використання OSINT має базуватися на принципах етики, які допомагають зберігати баланс між безпекою та приватністю. Це важливо в умовах сучасних інформаційних воєн, де рефлексивний контроль використовується для маніпулювання суспільною думкою та дезорієнтації противника.

Отже, OSINT виступає потужним інструментом, що має вирішальне значення для інформаційної війни та кібербезпеки, зокрема у контексті збройного конфлікту. Ця технологія дозволяє швидко й економічно отримувати значний обсяг інформації з відкритих джерел, що сприяє ефективній ідентифікації кіберзлочинців, виявленню та запобіганню потенційним загрозам. В умовах сучасної війни OSINT допомагає відстежувати переміщення противника, збирати докази воєнних злочинів та здійснювати моніторинг ситуації в реальному часі, що особливо актуально для оперативного прийняття рішень. Водночас використання OSINT потребує обережного підходу з точки зору конфіденційності та дотримання етичних стандартів, оскільки можливості розвідки з відкритих джерел можуть призводити до порушення приватності. Запровадження правових та етичних норм у сфері OSINT сприятиме розвитку цієї технології та дозволить ефективно використовувати її для забезпечення безпеки, зберігаючи при цьому права людини.

Перелік посилань

1. Степанюк, Р. Л. Сучасні криміналістичні засоби та методи протидії злочинності : навч. посіб. / Р. Л. Степанюк, В. О. Гусєва ; МВС України, Харків. нац. ун-т внутр. справ. – Харків : ХНУВС, 2024. – 232 с. – ISBN 978-966-610-278-5.
2. Yadav A., Kumar A., Singh V.. Open-source intelligence: a comprehensive review of the current state, applications and future perspectives in cyber security. Artif Intell. 2023. DOI: 10.1007/s10462-023-10454-y

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Загребельного Владислава Вікторовича
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

29.05.2025
дата


підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Загребельний Владислав Вікторович

Співавтор:

Назва: Система виявлення шкідливого програмного забезпечення на кінцевих пристроях

Науковий керівник:

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:1.1%

Коефіцієнт подібності 2:0%

Мікропробіли: 0

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-05-30 10:56:13.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

30.05.2025р.

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 9%**

ID: 242605 Title: Система виявлення шкідливого програмного забезпечення на кінцевих пристроях Added in a DB: 2025-05-30 Authors: Загребельний Владислав Вікторович Heads: Кльоц Ю.П. Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	81333	603	126 (0%)	3 (0%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система виявлення шкідливого програмного забезпечення на кінцевих пристроях

Автор: Загребельний Владислав Вікторович

Спеціальність: 125 – Кібербезпека

Освітня програма:Кібербезпека

Науковий керівник:Юрій КЛЬОЦ, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 99%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism складає 97,7%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 24.09.2024, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100%, визначається роботою з високою унікальністю тексту і допускається до захисту.

Керівник роботи _____ Юрій КЛЬОЦ

Гарант ОП _____ ВІктор ЧЕШУН

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Загребельний Владислав Вікторович

Тема Система виявлення шкідливого програмного забезпечення на кінцевих пристроях

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 82.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі, відповідно до поставленого завдання, проведене дослідження предметної області, Підібрано оптимальний стек технологій. Вибрано мову програмування. Зібрано датасет для навчання моделі. Розроблено back-end та front-end частини. Проведено тестування та налагодження. Представлено результати. Проведено порівняння з наявними рішеннями.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота повністю відповідає поставленому завданню, як у теоретичній, так і в практичній частинах.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У розділі 1 роглянуті існуючі рішення виявлення ШПЗ та систем виявлення ШПЗ. В другому розділі розглянуто та підібрано мову програмування для створення системи, зібрано датасет для навчання моделі. У третьому розділі розроблено front-end та back-end частину, відтестовано та налагоджено систему. Проведено порівняння системи з наявними популярними системами.

4. Позитивні сторони роботи Кваліфікаційна робота вирізняється високим рівнем актуальності, системним підходом до вирішення проблеми та практичною спрямованістю. Вона демонструє вміння студента застосовувати сучасні методи для вирішення реальних завдань кібербезпеки.

5. Негативні сторони роботи Окремі схеми потребують чіткішого оформлення й підписів.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення кваліфікаційної роботи відповідає темі роботи та виконане з дотриманням стандартів. В цілому, графічне оформлення є якісним, а пояснювальна записка відповідає нормам оформлення.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки весь матеріал роботи є структурованим, чітким та послідовним. Усі розділи роботи мають логічну послідовність, що сприяє зрозумінню викладеного матеріалу в рамках теми роботи. Графічний матеріал допомагає наочно продемонструвати доцільність та ефективність прийнятих рішень у створенні системи виявлення ШПЗ.

8. Інші зауваження Загалом робота виконана на високому рівні, однак доцільним було б більше уваги приділити деталізації графічних елементів та поясненням до них. Частину візуального матеріалу можна було б доповнити пояснювальними підписами для покращення сприйняття. Також рекомендується в подальшому дотримуватись єдиного стилю оформлення всіх елементів роботи.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні сторони кваліфікаційної роботи, а також негативні сторони, які не зменшують практичну цінність отриманих результатів і загальну якість роботи, рекомендованою оцінкою є «відмінно».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) _____
Підченко Сергій Костянтинович, завідувач кафедри телекомунікацій, медійних та інтелектуальних технологій, доктор технічних наук, професор, ХНУ

«28» травня 2025. _____ (підпис)