МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль аналізу зловмисного програмного забезпечення

Виконавець: Данило ХАНІН
Керівник: к.т.н., доцент Лілія ГАЛАТА
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: к.т.н., доцент Лілія ГАЛАТА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Ханіна Данила Андрійовича

1. Тема: Програмний модуль аналізу зловмисного програмного забезпечення.
2. затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
3. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
4. Вихідні дані: проаналізувати наявні системи та методики аналізу шкідливого коду; на основі аналізу розробити програмний модуль аналізу дампа пам'яті для виявлення кіберзагроз; провести порівняння з наявними методами.
5. Зміст пояснювальної записки: аналіз наявних систем і методик аналізу шкідливого програмного коду; розробка моделі машинного навчання на основі дампів пам'яті; верифікація отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2022	*Виконано*
2.	Аналіз літературних джерел	20.10.2022	*Виконано*
3.	Обґрунтування вибору рішення	22.10.2022	*Виконано*
4.	Збір інформації	24.10.2022	*Виконано*
5.	Дослідження сучасних систем і методик аналізу шкідливого програмного коду	27.10.2022	*Виконано*
6.	Дослідження сучасних методик в аналізі шкідливого програмного коду з використанням машинного навчання та нейронних мереж	05.11.2022	*Виконано*
7.	Розробка програмного модуля аналізу зловмисного програмного забезпечення	12.11.2022	*Виконано*
8.	Перевірка на антиплагіат	13.11.2022	*Виконано*
9.	Оформлення і друк пояснювальної записки	27.11.2022	*Виконано*
10.	Оформлення презентації	16.12.2022	*Виконано*
11.	Отримання рецензій від рецензента	22.12.2022	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Данило ХАНІН

(підпис, дата)

Керівник кваліфікаційної роботи Лілія ГАЛАТА

(підпис, дата)

РЕФЕРАТ

Дипломна робота на тему: «Програмний модуль аналізу зловмисного програмного забезпечення» складається зі вступу, основної частини, що містить 4 розділи, загальний висновок, 3 додатки та список використаної літератури. Загальний обсяг роботи –94сторінки. Робота містить 14 рисунків та 10 таблиць. Список використаних джерел включає 25 джерел.

Метою кваліфікаційної роботи є розробка програмного модулю аналізу зловмисного програмного забезпечення.

У дипломній роботі розглянуті питання щодо сучасних методів виявлення зловмисного програмного забезпечення.

Проведені дослідження побудовані на сучасних підходах виявлення шкідливого коду з використанням моделей машинного навчання.

Реалізація власного програмного модуля, побудованого на базі моделі машинного навчання 'випадковий ліс', демонструє високі показники ефективності. Цей метод ефективніший, як порівняно з класичними методами, так і з аналогічними методами, побудованими на основі моделей машинного навчання.

Запропоновані методи дозволяють забезпечити швидкий та надійний захист системи.

Ключові слова: МОДЕЛЬ, ВИПАДКОВИЙ ЛІС, ЗЛОВМИСНЕ ЗАБЕЗПЕЧЕННЯ, НАВЧАННЯ

ЗМІСТ

ВСТУП

Актуальність. В умовах стрімкого розвитку інформаційних технологій і загальної цифровізації, питання кібербезпеки та аналізу шкідливого програмного забезпечення стають особливо актуальними і важливими. З кожним роком обсяг цифрових даних збільшується, так само як і кількість загроз, спрямованих на їх захист і збереження. Шкідливе програмне забезпечення постійно еволюціонує, стаючи дедалі складнішим і витонченішим, що вимагає розроблення нових методів і підходів до його аналізу та запобігання.

Кібератаки можуть призвести до катастрофічних наслідків, включно з витоком конфіденційної інформації, фінансовими втратами і шкодою репутації для організацій та індивідів. Тому, розробка нових методів та інструментів для ефективного аналізу та виявлення шкідливого програмного забезпечення є ключовим аспектом забезпечення кібербезпеки.

З урахуванням вищеописаного, розробка програмного модуля для аналізу шкідливого програмного забезпечення, використовуючи методи машинного навчання та аналізу даних з дампа пам'яті, є вельми актуальним і важливим завданням. Застосування машинного навчання може підвищити ефективність і швидкість виявлення шкідливих програм, а аналіз дампа пам'яті дасть змогу виявляти навіть ті загрози, які здатні ховатися від традиційних методів виявлення.

Дослідження і розробка в цій галузі сприяє не тільки підвищенню загального рівня кібербезпеки, а й розширенню наукових знань і практичних навичок у сфері аналізу шкідливого програмного забезпечення, що, безумовно, є актуальною і значущою проблемою сучасності.

Мета роботи - розробка програмного модулю аналізу зловмисного програмного забезпечення.

Виходячи з поставленої мети завданнями кваліфікаційної роботи є:

*провести дослідження сучасних підходів до класифікації шкідливого програмного забезпечення та рішень щодо аналізу зловмисного програмного забезпечення;

*розробити авторський програмний модулю аналізу зловмисного програмного забезпечення з використанням моделі "Випадковий Ліс"

*провести дослідження розробленого власного програмного модуля.

Об'єкт дослідження — процедура та підходи щодо аналізу зловмисного програмного забезпечення..

Предмет дослідження — інформаціна мережа, шкідливе програмне забзпечення, машинне навчання

Методи — проведені дослідження базуються на сучасних методах побудови захищених інформаційних мереж, підходах щодо класифікації шкідливого програмного забезпечення, методах аналізу зловмисного програмного забезпечення.

НАУКОВА НОВИЗНА полягає у тому, що запропоновано авторський програмний застосунок аналізу зловмисного програмного забезпечення, на основі застосування теорії машинного навчання, а саме моделі "Випадковий Ліс", що забезпечує точніть та здатніть правильно класифікувати позитивні випадки та шкідливі додатки.

ПРАКТИЧНА ЦІННІСТЬ — полягає у створенні власного програмного модуля аналізу зловмисного програмного забезпечення, на на основі даних із дампів пам'яті, з використанням мови програмування Python, що дозволяє забезпечити більш ефективне запобігання та реагування на кіберзагрози за рахунок викристання динамічного типу аналізу шкідливого програмного забезпечення і аналізатору дампів пам'яті.

Апробація.

1.Галата Л. П., Мазур Я. С., Ханін Д. А., Сучасні рішення щодо аналізу зловмисного програмного забезпечення // Modern problems of science, education and society: VIII Міжнародна науково-практична конференція, 9-11 жовтня 2023 р.: тези доповіді. – К., 2023. – С.348-353.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОНЯТТЯ ЩОДО АНАЛІЗУ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Проблема забезпечення безпеки інформації в інформаційних мережах

В епоху цифровізації суспільства проблема забезпечення безпеки інформації стає дедалі актуальнішою. Інформаційні технології проникають в усі сфери людської діяльності, забезпечуючи високий ступінь взаємозв'язку та доступу до даних. Однак, цей процес відкриває нові вектори для атак і експлуатації вразливостей, що висуває на перший план питання інформаційної безпеки.

Історичний контекст та актуальність проблеми

Зі зростанням залежності суспільства від інформаційних систем зростає і ступінь потенційного збитку від їхньої недоступності або компрометації. В останні десятиліття світ став свідком низки масштабних кібератак на державні та корпоративні інформаційні системи, що підкреслює важливість проблеми.

Один із яскравих прикладів кібератак, що демонструють глобальні наслідки та високий ступінь шкоди, - атака WannaCry, що сталася в травні 2017 року. Цей шкідливий криптовірус швидко поширився по всьому світу, заразивши сотні тисяч комп'ютерів у більш ніж 150 країнах. WannaCry блокував файли користувачів і вимагав викуп у криптовалюті Bitcoin за їх відновлення. Атака особливо сильно вплинула на організації з критично важливою інфраструктурою, такі як лікарні, банки та виробничі підприємства.

Випадок WannaCry ілюструє, як масштабні та складні кібератаки можуть паралізувати роботу важливих організацій і структур, а також як важливо мати ефективні методи аналізу та боротьби зі шкідливим програмним забезпеченням.

Основні загрози та виклики інформаційної безпеки.

У контексті інформаційної безпеки існує безліч загроз, які можна класифікувати за різними параметрами, як-от джерело, метод поширення, мета атаки тощо. Розглянемо деякі з найбільш поширених і небезпечних загроз, з якими стикаються сучасні інформаційні системи.

Таблиця 1 представляє загальний огляд різних типів загроз у сфері інформаційної безпеки, їхніх характеристик і потенційного збитку.

Таблиця 1.1.

Тип загрози	Характеристика	Приклади впливу	Методи захисту
Фішинг	Шахрайство через електронну пошту або веб-сайти, спрямоване на отримання конфіденційних даних	Витік персональних даних	Навчання користувачів, антифішингові фільтри
Зловмисне програмне забезпечення	Програми, створені для завдання шкоди або несанкціонованого доступу до систем	Знищення даних, вимагання викупу	Антивірусне програмне забезпечення, фільтри вмісту
DDoS-атаки	Перевантаження ресурсів сервера кількістю запитів	Недоступність сервісу	Анти-DDoS системи.
Внутрішні загрози	Дії або бездіяльність співробітників, що призводить до шкоди	Витік конфіденційної інформації	Політики доступу, моніторинг поведінки

Продовження таблиці 1.1.

Тип загрози	Характеристика	Приклади впливу	Методи захисту
Ін’єкції	Введення шкідливих даних, що призводить до несанкціонованого виконання коду	Порушення цілісності даних, несанкціонований доступ	Валідація вводу, параметризовані запити
Крос-сайтовий скриптинг (XSS)	Введення шкідливих скриптів у веб-сторінки, що переглядаються іншими користувачами	Витік даних користувача, перехоплення сеансів	Валідація вводу, санітація виводу

Ці категорії загроз відображають різноманітні та складні виклики, з якими стикаються фахівці з інформаційної безпеки в сучасному цифровому світі. Важливо підкреслити, що кожна загроза вимагає індивідуалізованого підходу і, в деяких випадках, комплексу заходів із забезпечення безпеки.

Статистика і приклади кібератак

З кожним роком кількість кібератак зростає, а їхні масштаби та наслідки стають дедалі серйознішими, що підтверджують статистичні дані та конкретні приклади.

В останні кілька років світ спостерігав низку масштабних кібератак, які охоплювали різні галузі та географічні регіони. Наприклад, атака на компанію Equifax у 2017 році призвела до витоку персональної інформації понад 147 мільйонів осіб, включно з іменами, соціальними страховими номерами, адресами та іншими даними. Атака обійшлася компанії в $4 мільярди у вигляді штрафів, відшкодування збитків і поліпшення інфраструктури безпеки[2].

Ще одним прикладом може слугувати атака WannaCry, про яку ми вже згадували раніше. Вона призвела до масштабного паралічу систем охорони

здоров'я у Великій Британії та впливала на організації по всьому світу, включно із залізничними станціями в Німеччині та автомобільними заводами в Японії.

Згідно з доповіддю компанії Verizon "Доповідь про дослідження витоків даних" (Data Breach Investigations Report) за 2021 рік:

85% усіх кібератак мали фінансову мотивацію;

У 61% випадків кібератак брали участь зовнішні актори;

Фішинг був причиною 36% витоків даних;

28% інцидентів були пов'язані з внутрішніми акторами.

Ці статистичні дані та приклади підкреслюють масштаб і серйозність проблеми кіберзагроз. Важливість розроблення та впровадження ефективних механізмів захисту інформації є ключовим аспектом для забезпечення стійкості та безпеки організацій різних масштабів і напрямів діяльності. У світлі зростаючої складності та хитромудрості кібератак, розробка нових і вдосконалення наявних методів захисту від кіберзагроз набуває особливої актуальності та значущості.

Важливість аналізу шкідливих програм.

Аналіз шкідливих програм стоїть у центрі кібербезпеки, оскільки надає ключові дані для розуміння і, отже, запобігання кібератакам. Зломщики та кіберзлочинці постійно розробляють нові та вдосконалені методи атаки, а аналіз шкідливих програм дає змогу фахівцям з кібербезпеки тримати руку на пульсі та прогнозувати потенційні загрози.

Визначення та розпізнавання загроз

Детальний аналіз шкідливих програм допомагає не тільки визначити сутність загрози, а й розробити методи її розпізнавання. Це включає в себе виявлення специфічних характеристик шкідливого коду, таких як сигнатури, поведінкові характеристики та інше, що може бути використане для детектування та блокування подібних загроз у майбутньому.

Розробка стратегій захисту та реагування на інциденти

Розуміння того, як працює шкідливе програмне забезпечення, дозволяє розробити ефективні стратегії захисту та плани реагування на інциденти. Це

може включати в себе створення патчів для вразливостей, розробку противірусних сигнатур і стратегій для мінімізації збитків у разі порушення.

Покращення системи кібербезпеки

Аналіз шкідливого програмного забезпечення забезпечує цінні уроки, які можуть бути використані для поліпшення загальної стійкості системи кібербезпеки. Відгуки та аналіз інцидентів є ключовими компонентами для постійного поліпшення й адаптації до мінливого ландшафту загроз[1].

Запобігання майбутнім атакам

Здатність аналізувати та розуміти шкідливі програми прямо пов'язана зі здатністю запобігати майбутнім атакам. Через вивчення та аналіз минулих загроз фахівці з кібербезпеки можуть прогнозувати та запобігати майбутнім атакам, створюючи перепони та захисти проти відомих векторів атак[2].

Аналіз шкідливого програмного забезпечення є фундаментальною частиною захисту інформаційних систем і даних від кіберзагроз. Поглиблене розуміння загроз дає змогу професіоналам галузі кібербезпеки розробляти ефективні стратегії та рішення для захисту критично важливої інформації та запобігання кібератакам.

У першому підрозділі було розглянуто ключові аспекти, пов'язані зі шкідливим програмним забезпеченням і його аналізом у контексті кібербезпеки:

1. Актуальність Проблеми:

- Історичний контекст і актуальні приклади кібератак, як-от WannaCry, підкреслили глибокий і широкий вплив, який шкідливе програмне забезпечення може мати на глобальні системи та інфраструктури.

2. Різноманітність Загроз:

- Було представлено огляд різних видів загроз кібербезпеки, виявляючи складність і різноманіття методів атаки, використовуваних зловмисниками.

3. Статистичні Дані:

- Наведені статистичні дані та приклади підкреслили зростаючу загрозу, яку становлять кібератаки, і підтвердили необхідність розроблення більш

ефективних методів виявлення та реагування на шкідливе програмне забезпечення.

4. Значимість Аналізу:

- Було наголошено на важливості аналізу шкідливого програмного забезпечення для виявлення та прогнозування загроз, розроблення стратегій захисту та поліпшення загальних систем кібербезпеки.

З вищезазначених пунктів стає очевидною критична необхідність розробки ефективних стратегій та інструментів для аналізу шкідливого програмного забезпечення. Таким чином, цей підрозділ встановлює основу для подальшого дослідження та обговорення способів оптимізації та вдосконалення методів аналізу шкідливого програмного забезпечення, які будуть розглянуті в наступних розділах цієї дипломної роботи[3].

Додатково, підкреслюється важливість поглибленого розуміння механізмів роботи шкідливих програм та їхнього впливу на системи для створення більш ефективних і адаптивних засобів захисту, що буде основою практичної частини роботи.

1.2 Сучасні підходи до класифікації шкідливого програмного забезпечення

Класифікація шкідливого програмного забезпечення відіграє важливу роль у сфері кібербезпеки, полегшуючи процес ідентифікації, аналізу та пом'якшення загроз. Сучасні підходи до класифікації варіюються і залежать від різних чинників і характеристик, таких як вектор атаки, механізм поширення, поведінка після зараження тощо[9].

Визначення та характеристики шкідливого програмного забезпечення

Шкідливе програмне забезпечення є програмами або кодом, розробленими з метою нанесення шкоди або заподіяння незручностей

користувачам, пристроям, мережам та/або організаціям. Це може включати в себе різні форми загроз, такі як віруси, черв'яки, троянські коні і рансомвар, кожен з яких має свої унікальні характеристики і методи поширення.

Основні характеристики шкідливого програмного забезпечення включають в себе:

- Мета: Шкідливе програмне забезпечення завжди створюється з певною метою, чи то крадіжка даних, чи то пошкодження файлів, чи навіть просто створення хаосу в мережевих системах. Найчастіше, ці цілі спрямовані на отримання фінансової вигоди, але можуть також бути мотивовані бажанням влаштувати деструкцію або навіть просто інтересом[11].
- Скритність: Шкідливе програмне забезпечення часто розробляється так, щоб уникнути виявлення. Це може включати в себе механізми, що дають змогу обходити антивірусне програмне забезпечення, приховувати мережеву активність або маскувати шкідливу активність під виглядом легітимних процесів.
- Поширення: Здатність до поширення - важлива характеристика багатьох видів шкідливого програмного забезпечення. Деякі форми, як-от комп'ютерні віруси та черв'яки, спеціально розробляються для автономного розповсюдження, тоді як інші, наприклад, троянські коні, можуть поширюватися за допомогою соціальної інженерії або інших методів.
- Поведінка: Різні види шкідливого програмного забезпечення діють по-різному, коли вони потрапляють у систему. Деякі активно взаємодіють з операційною системою і користувацькими файлами, тоді як інші працюють у фоновому режимі, щоб уникнути виявлення[4].
- Збиток: Ступінь і характер шкоди, яку може заподіяти шкідливе ПЗ, також є важливою характеристикою. Деякі форми можуть просто слугувати подразником (наприклад, adware), тоді як інші, такі як рансомвар, можуть завдати серйозної та довгострокової шкоди.

- Механізм обходу: Шкідливе програмне забезпечення також часто містить механізми, що дають змогу обходити заходи безпеки, як-от фаєрволи, антивірусні програми та системи виявлення вторгнень.

Кожна з цих характеристик є критичною для розуміння динаміки та потенційної загрози, яку становить шкідливе програмне забезпечення. Вони також служать важливими параметрами для аналізу і класифікації шкідливого програмного забезпечення з метою забезпечення кібербезпеки[7].

У таблиці 2 наведені основні характеристики різних видів шкідливого програмного забезпечення:

Таблиця 1.2.

Тип	Характеристика	Метод поширення	Потенційна шкода	Приклади
Вірус	Інфекційний код	Інфіковані файли	Пошкодження файлів	Zeus
Черв'як	Самопоширення	Мережі	Перевантаження мережі	Conficker
Троянський кінь	Маскування	Підроблені додатки	Несанкціонований доступ	Emotet
Рансомвар	Шифрування	Інфіковані вкладення	Втрата даних	WannaCry
Шпигунське ПЗ	Збір даних	Вбудоване ПЗ	Витік конфіденційної інформації	Zbot
Логічна бомба	Умовний запуск	Вбудоване ПЗ	Втрата даних, пошкодження системи	July 4th Bomb

Кожен тип шкідливого програмного забезпечення в таблиці має свої унікальні характеристики і являє собою різні рівні загрози для цифрових систем і мереж. Сучасні методи захисту та виявлення шкідливого програмного

забезпечення повинні враховувати це розмаїття, забезпечуючи комплексний і багаторівневий підхід до кібербезпеки.

У наступному підрозділі розглядаються критерії та методи класифікації шкідливого програмного забезпечення, що дає змогу глибше зрозуміти та категоризувати різні види та методи атак[8].

Тип впливу шкідливого програмного забезпечення на систему

Різні форми шкідливого програмного забезпечення можуть чинити досить різноманітний вплив на комп'ютерні системи, мережі та дані, які вони інфікують. Важливим критерієм для класифікації шкідливого програмного забезпечення є тип впливу на систему, тому що це дає змогу аналітикам і фахівцям із кібербезпеки зрозуміти, які загрози становить конкретний вид ПЗ і яким чином можна мінімізувати або запобігти шкоді[5].

Нижче наведено деякі загальні типи впливу, які може чинити шкідливе програмне забезпечення на систему:

1. Пошкодження даних: Шкідливе ПЗ може модифікувати, видаляти або іншим чином пошкоджувати дані на зараженій системі[12].

2. Шифрування даних: Через використання рансомвара, шкідливе ПЗ може зашифрувати дані користувача і вимагати викуп за їх відновлення.

3. Викрадення системи: Шкідливе ПЗ може надати зловмиснику контроль над зараженою системою, дозволяючи йому виконувати довільні команди.

4. Збирання даних: Шпигунське ПЗ і кейлогери збирають і передають конфіденційні дані, такі як облікові дані, особисту інформацію тощо.

5. Відмова в обслуговуванні (DoS/DDoS): Шкідливе ПЗ може зловживати ресурсами системи або мережі, викликаючи відмову в обслуговуванні для легітимних користувачів[6].

6. Скритність і ухилення: Шкідливе ПЗ може активно ухилятися від виявлення, змінюючи свій код або поведінку.

7. Поширення: Шкідливе ПЗ може автоматично поширювати себе мережею або через знімні носії[10].

8. Використання ресурсів: Шкідливе ПЗ може використовувати ресурси системи для виконання шкідливої діяльності, такої як майнінг криптовалют.

9. Створення ботнетів: Шкідливе ПЗ може об'єднувати заражені системи в мережу (ботнет) і використовувати їх для додаткових атак.

Визначення та усвідомлення цих типів впливу шкідливого програмного забезпечення на систему забезпечують основу для розроблення ефективних заходів безпеки, спрямованих на виявлення, запобігання та реагування на різні загрози в середовищі інформаційної безпеки. У наступному розділі буде розглянуто різні механізми поширення шкідливого програмного забезпечення[12].

Механізми поширення та ухилення від виявлення

Шкідливе програмне забезпечення може використовувати різні стратегії та методи для розповсюдження та ухилення від виявлення, кожен з яких має свої унікальні характеристики та виклики для фахівців з кібербезпеки.

Механізми поширення

- Електронна пошта: Вкладення: Надсилання виконуваних файлів або документів, що містять шкідливі макроси.Фішинг: Використання підроблених електронних листів для крадіжки облікових даних або поширення шкідливого ПЗ.
- Мережеві вразливості: Експлуатація відомих вразливостей у ПЗ для несанкціонованого доступу або поширення шкідливого коду.
- Міжмережева взаємодія: Використання мережевих протоколів, таких як SMB або RDP, для переміщення мережею та інфікування інших систем[17].
- Соціальна інженерія: Використання різних психологічних трюків, щоб переконати користувача в тому, щоб той надав конфіденційну інформацію або запустив шкідливе ПЗ.
- Інфіковані носії даних:Поширення шкідливого ПЗ через знімні носії, такі як USB-драйви.

- Шкідливі веб-сайти:Використання експлойтів і шкідливих завантажень для зараження відвідувачів веб-сайту.

Механізми ухилення від виявлення

- Поліморфізм:Зміна шкідливого коду під час кожного запуску або передачі, щоб уникнути виявлення на основі сигнатур.
- Обфускація:Використання технік, щоб "заплутати" код, зробити його складнішим для аналізу та ідентифікації.
- Rootkits:Інтеграція в системні процеси або ядро ОС, щоб приховати шкідливу активність від антивірусів і адміністраторів.
- Тимчасові бомби:Увімкнення шкідливого коду тільки при виконанні певних умов (наприклад, коли минає певна кількість часу або на комп'ютері відкривається конкретна програма).
- Блокування безпеки:Виявлення та вимкнення антивірусних продуктів або інших заходів безпеки, щоб уникнути виявлення та видалення.
- Самознищення:Активація механізмів, які автоматично видаляють шкідливе ПЗ після виконання його завдання або якщо виявляється спроба аналізу.

Кожен із цих механізмів являє собою унікальні виклики в контексті захисту інформаційної системи та потребує спеціалізованих інструментів і стратегій для виявлення, аналізу та мітигації загрози. Розробка ефективних методів захисту та реагування на інциденти починається з розуміння та врахування цих механізмів у стратегії кібербезпеки[18] [22].

Приклади та аналіз основних класів шкідливого програмного забезпечення

Вивчення конкретних прикладів шкідливого програмного забезпечення допомагає виявити загальні стратегії й тактики, які використовують зловмисники, а також проблеми, з якими стикаються фахівці з кібербезпеки під час забезпечення захисту.

1. Conficker (Вірус)

Опис: Conficker, що з'явився 2008 року, був одним із найпоширеніших вірусів, які використовували вразливість у Windows для інфікування мільйонів комп'ютерів по всьому світу[13] [21].

Аналіз: Conficker став відомий завдяки своїй здатності заражати велику кількість комп'ютерів і створювати великі ботнети. Він використовував різні методи для обходу виявлення й ухилення від спроб усунення впливу, демонструючи важливість багатоаспектних підходів до захисту від вірусів і підкреслюючи роль оновлень і патчів безпеки в запобіганні поширенню шкідливого ПЗ.

2. ILOVEYOU (Хробак)

Опис: ILOVEYOU, що з'явився 2000 року, використовував соціальну інженерію, розповсюджуючись через електронні листи, які виглядали так, нібито їх надіслали друзі або колеги.

Аналіз: ILOVEYOU наголосив на небезпеках, пов'язаних із фішингом і соціальною інженерією, і показав, як важливим є навчання користувачів для розпізнавання та уникнення подібних загроз.

3. Emotet (Троянський кінь)

Опис: Emotet спочатку був троянським конем, спрямованим на крадіжку банківської інформації, але з часом еволюціонував, стаючи потужною загрозою, яка поширювала інші види шкідливого ПЗ.

Аналіз: Emotet показав, як шкідливе ПЗ може адаптуватися і змінюватися з часом, щоб задовольняти потреби зловмисників, підкреслюючи важливість постійного моніторингу та аналізу загроз.

4. WannaCry (Рансомвар)

Опис: WannaCry, сплеск якого стався у 2017 році, був глобальною загрозою, що вразила організації по всьому світу, шифруючи дані та вимагаючи викуп.

Аналіз: WannaCry продемонстрував руйнівну силу рансомвара і показав, наскільки важливим є регулярне створення резервних копій даних і забезпечення їхньої безпеки, а також важливість патчинг і оновлення систем.

5. Back Orifice (Бекдор)

Опис: Back Orifice, випущений 1998 року, був одним із перших прикладів шкідливого ПЗ, створеного з метою надання віддаленого доступу до Windows-комп'ютерів.

Аналіз: Back Orifice показав, як зловмисники можуть використовувати бекдори для обходу систем безпеки і непомітного управління пристроями користувачів, що підкреслює важливість використання комплексних заходів безпеки, щоб виявляти і блокувати всі види шкідливого ПЗ.

Кожен із цих прикладів демонструє унікальні виклики, що стоять перед професіоналами в галузі кібербезпеки, і показує різноманітність стратегій і тактик, які використовують зловмисники, щоб досягти своїх зловмисних цілей. Усі вони підкреслюють важливість багаторівневого підходу до кібербезпеки і необхідність постійного адаптування до нових загроз, що розвиваються[14].

Вивчення сучасних підходів до класифікації шкідливого програмного забезпечення та аналіз конкретних прикладів і механізмів його поширення й ухилення від виявлення дають змогу виокремити кілька ключових моментів:

1. Різноманіття та адаптивність Шкідливого ПЗ:

- Шкідливі програми можуть набувати безлічі форм і адаптуватися для виконання різних завдань, оминаючи стратегії виявлення та захисту.

2. Соціальна Інженерія:

- Людський фактор залишається однією з найслабших ланок у ланцюзі кібербезпеки, і соціальна інженерія продовжує бути ефективним методом поширення шкідливого ПЗ.

3. Технічні та організаційні заходи:

- Захист від шкідливого ПЗ вимагає комплексного підходу, включно з технічними засобами захисту, стратегіями управління ризиками та навчанням персоналу.

4. Постійна Загроза:

- Інциденти безпеки, пов'язані зі шкідливим ПЗ, продовжують становити загрозу для організацій усіх розмірів і масштабів, наголошуючи на необхідності постійного моніторингу та адаптації стратегій кібербезпеки.

5. Оновлення та патчі:

- Регулярні оновлення та патчі систем безпеки залишаються критично важливими для запобігання експлуатації вразливостей.

6. Превентивні Заходи:

- Запобіжні заходи, як-от резервне копіювання даних і багатофакторна автентифікація, можуть значно знизити ризики та наслідки інцидентів, пов'язаних зі шкідливим ПЗ.

7. Співпраця та Обмін Інформацією:

- Обмін даними про загрози і співробітництво в галузі кібербезпеки важливі для створення більш ефективних стратегій виявлення і відповіді на шкідливе ПЗ.

8. Дослідження і Розробка:

- Безперервні дослідження і розробки в галузі кібербезпеки необхідні для побудови більш ефективних і адаптивних механізмів виявлення і захисту від шкідливого ПЗ.

У контексті триваючого розвитку й адаптації шкідливого програмного забезпечення, стратегії кібербезпеки також повинні адаптуватися і розвиватися, щоб справлятися з новими і мінливими загрозами. Слід наголосити на важливості глибокого розуміння механізмів шкідливого ПЗ, а також застосування знань для створення і впровадження ефективних стратегій захисту та реагування на інциденти[15].

1.3. Дослідження сучасних рішень щодо аналізу зловмисного програмного забезпечення.

Огляд наявних методів аналізу

Аналіз шкідливого програмного забезпечення - це процес дослідження функціоналу та поведінки підозрілих файлів і програм з метою визначення їхніх намірів і потенційної загрози. Цей аналіз можна розділити на кілька основних типів:

Статичний аналіз

- Аналіз Сигнатур: Ідентифікація відомого шкідливого ПЗ на основі унікальних рядків або характеристик коду.
- Аналіз Метаданих: Вивчення атрибутів файлу, таких як розмір, хеш-суми і дата створення, для виявлення підозрілих характеристик.
- Реверс-інжиніринг: Декомпіляція та аналіз коду для розуміння його функціональності та механізмів роботи.

Динамічний аналіз

- Аналіз Поведінки: Спостереження за діями ПЗ під час його виконання, включно із взаємодією з файловою системою, реєстром і мережею.
- Ізоляція Середовища: Використання віртуального середовища або пісочниці для ізоляції шкідливого ПЗ і запобігання реальному впливу на систему.

Поведінковий аналіз

- Моніторинг Системних Викликів: Відстеження системних викликів, що виконуються шкідливим ПЗ, щоб зрозуміти, як воно взаємодіє з ОС.
- Аналіз Змін Системи: Вивчення змін, внесених у систему шкідливим ПЗ, включно зі створенням, видаленням і модифікацією файлів і ключів реєстру.

Кодовий аналіз

- Декомпіляція: Перетворення виконуваного коду назад у вихідний код для аналізу.
- Аналіз Управління Потоком: Вивчення того, як управління передається між різними частинами коду шкідливого ПЗ.

Машинне навчання для аналізу шкідливого ПЗ

- Витяг Ознак: Визначення ключових характеристик шкідливого ПЗ, які можуть бути використані для навчання моделі.
- Класифікація: Використання алгоритмів машинного навчання для автоматичного класифікування файлів як шкідливих або нешкідливих на основі витягнутих ознак.

Інтегративний підхід

Особливо ефективними можуть виявитися підходи, які об'єднують різні методи аналізу, щоб максимізувати шанси на успішне виявлення та аналіз шкідливого ПЗ. Наприклад, статичний аналіз може швидко ідентифікувати відомі загрози, динамічний аналіз може допомогти виявити загрози на етапі виконання, а машинне навчання може допомагати у виявленні нових, невідомих загроз. Комбінування цих підходів у рамках інтегрованої стратегії може надати рівень захисту, який набагато більший, ніж сума його частин[16] [19].

Технології та інструменти для аналізу шкідливого програмного забезпечення

Для ефективного аналізу шкідливого програмного забезпечення фахівці з кібербезпеки використовують різні технології та інструменти, які забезпечують глибокий аналіз і дають змогу отримувати цінну інформацію про шкідливі програми. Розглянемо деякі з ключових технологій та інструментів:

- Аналізатори Шкідливого ПЗ: Інструменти, такі як VirusTotal, надають швидкий спосіб перевірити файли на наявність відомого шкідливого ПЗ, використовуючи безліч антивірусних рушіїв.
- Пісочниці: Середовища, такі як Cuckoo Sandbox, дають змогу безпечно виконувати шкідливе ПЗ і спостерігати за його поведінкою в ізольованому середовищі.
- Інструменти Реверс-Інжинірингу: Програми, такі як IDA Pro і Ghidra, надають можливості для декомпіляції та аналізу коду шкідливого ПЗ.
- Системи Детекції Вторгнень (IDS): Інструменти, такі як Snort, моніторять мережевий трафік, щоб виявляти підозрілі дії та відомі атаки.

- Системи Управління Інцидентами та Подіями Безпеки (SIEM): Рішення, такі як Splunk, агрегують і аналізують дані про події безпеки з різних джерел.
- Фреймворки Автоматизації Аналізу: Інструменти, такі як MISP (Malware Information Sharing Platform & Threat Sharing), полегшують обмін і аналіз даних про загрози.
- Інструменти Аналізу Мережевого Трафіку: Wireshark та інші інструменти дають змогу аналізувати мережевий трафік і вивчати комунікацію шкідливого ПЗ.
- Інструменти Форензики: Інструменти, як-от Volatility, надають можливості для проведення форензичного аналізу та вивчення артефактів атаки.
- Моделі Машинного Навчання: Різноманітні інструменти та платформи машинного навчання, як-от TensorFlow, використовуються для розробки моделей, здатних ідентифікувати шкідливе ПЗ на основі його характеристик і поведінки.

Кожен із цих інструментів і технологій служить певним цілям і може бути використаний на різних етапах процесу аналізу шкідливого ПЗ. Важливо підходити до вибору інструментів стратегічно, з огляду на цілі та вимоги конкретного дослідження або проєкту[20].

Порівняльний аналіз інструментів і методів

Критерії порівняння

Під час порівняння інструментів і методів аналізу шкідливого програмного забезпечення, слід враховувати низку критеріїв:

- Ефективність виявлення: Імовірність успішного виявлення різних видів шкідливого ПЗ.
- Швидкість аналізу: Як швидко інструмент або метод може провести аналіз.
- Помилкові спрацьовування: Частота помилкових тривог або неправильної класифікації нешкідливих файлів як шкідливих.

- Здатність виявлення нових загроз: Ефективність у виявленні нових, раніше невідомих варіантів шкідливого ПЗ.
- Споживання ресурсів: Як багато системних ресурсів потрібно для роботи.
- Зручність використання: Простота налаштування, використання та інтерпретації результатів.
- Вартість: Включаючи як прямі витрати на купівлю/підписку, так і непрямі витрати на експлуатацію.
- Підтримка і співтовариство: Наявність підтримки та активності спільноти, яка може допомагати у вирішенні проблем і обміні досвідом.

Аналіз сильних і слабких сторін. Кожен інструмент і метод аналізу матиме свої сильні та слабкі сторони залежно від критеріїв порівняння. Наприклад, статичний аналіз може бути швидшим і менш ресурсномістким, але може не розпізнавати нові загрози так ефективно, як методи, що базуються на машинному навчанні[24].

Таблиця порівняння. Ми можемо створити таблицю порівняння для різних інструментів/методів, використовуючи згадані критерії. Для прикладу, давайте порівняємо кілька інструментів. Порівняльний аналіз інструментів і методів наведено в таблиці 3:

Таблиця 1.3.

Критерій	VirusTotal	Cuckoo Sandbox	IDA Pro	TensorFlow
Ефективність виявлення	Висока	Висока	Середня	Висока
Швидкість аналізу	Висока	Середня	Низька	Середня
Хибні спрацьовування	Низька	Середня	Середня	Середня

Продовження таблиці 1.3.

Критерій	VirusTotal	Cuckoo Sandbox	IDA Pro	TensorFlow
Виявлення нових загроз	Низька	Середня	Середня	Висока

Примітки:

VirusTotal надає швидкі результати і має високий ступінь зручності, але може бути менш ефективний для виявлення нових загроз порівняно з іншими інструментами.

Cuckoo Sandbox дає змогу проводити глибокий аналіз поведінки, але може споживати значні ресурси і вимагати складнішого налаштування та аналізу.

IDA Pro є потужним інструментом реверс-інжинірингу, але може бути не таким швидким або зручним для деяких типів аналізу.

TensorFlow надає потужні можливості для створення моделей машинного навчання, здатних виявляти шкідливе програмне забезпечення на основі різних характеристик і поведінкових патернів. Він може бути налаштований для використання різних алгоритмів і підходів машинного навчання, і має високу здатність до виявлення нових загроз. Однак, навчання моделей може споживати значні обчислювальні ресурси, і ефективність моделі буде залежати від якості даних, що використовуються для навчання, і правильності вибору та налаштування алгоритму[25].

Кожен інструмент має свої унікальні переваги і може бути оптимальним для різних сценаріїв використання і в різних стадіях процесу аналізу шкідливого програмного забезпечення.

Недоліки та проблеми наявних підходів

Аналіз шкідливого програмного забезпечення - складний і багатофакторний процес, який стикається з низкою проблем і викликів. Існуючі підходи та інструменти несуть у собі деякі недоліки й обмеження, які можуть включати в себе таке:

1. Здатність виявляти нові загрози

Сучасні методи виявлення шкідливого програмного забезпечення, особливо ті, що засновані на підписах і характеристиках відомого шкідливого програмного забезпечення, можуть зазнавати труднощів у розпізнаванні нових, невідомих варіантів загроз. Зловмисники активно використовують техніки обфускації і пакування для обходу традиційних методів виявлення.

2 Помилкові позитивні спрацьовування

Помилкові спрацьовування, або помилкове визначення нешкідливих файлів і процесів як шкідливих, можуть призвести до небажаних наслідків, як-от видалення або блокування важливих системних файлів і додатків.

3. Обчислювальні ресурси

Інтенсивні методи аналізу, як-от динамічний аналіз і машинне навчання, можуть споживати значні обчислювальні ресурси, що може бути проблематичним в умовах обмеженої обчислювальної потужності або в середовищах із високими вимогами до продуктивності.

4. Складність та експертні знання

Деякі інструменти та методи аналізу вимагають глибоких знань і досвіду в галузі кібербезпеки і можуть бути складними в налаштуванні та використанні, що створює бар'єр для широкого впровадження і використання.

5. Адаптація зловмисників

Атакуючі постійно розробляють нові методи і техніки для обходу наявних механізмів виявлення та аналізу, що являє собою "перегони озброєнь" між розробниками безпеки і зловмисниками.

6. Автоматизація

Незважаючи на те що автоматизація відіграє ключову роль в аналізі шкідливого програмного забезпечення, багато аспектів, особливо в галузі реверс-інжинірингу та глибокого аналізу поведінки, як і раніше, вимагають ручного втручання експерта.

7. Масштабованість

Здатність аналізувати великі обсяги даних і велику кількість зразків шкідливого програмного забезпечення є критично важливою в умовах постійно зростаючої кількості загроз, що ставить під загрозу ефективність і своєчасність аналізу.

Усвідомлення та розуміння цих проблем і недоліків є важливим кроком до розроблення покращених методів та інструментів для аналізу шкідливого ПЗ, що ми розглядатимемо в наступних підрозділах[25].

Розглгянемо основні інструменти для аналізу шкідливого програмного забезпечення, визначимо тип аналізу на якому вони працюють, основні функції, що лежать в їх основі, а також переваги та недоліки кожного інструменту та занесемо отримані результати дослідження в таблицю 4.

Таблиця 1.4.

Назва програми	Тип аналізу	Основні функції	Переваги	Недоліки та проблеми
VirusTotal	Статичний і динамічний	Агрегація даних антивірусних сканерів, для оцінювання файлів і URL.	Швидке сканування файлів і URL;	Обмежений в аналізі захищених або обфусцированих шкідливих зразків.
Cuckoo Sandbox	Динамічний	Автоматизований аналіз файлів і URL в ізольованому середовищі.	Детальні звіти аналізу; модульна структура; відкритий вихідний код і активна спільнота.	Потребує значних ресурсів і експертизи для налаштування та управління; може обходитись деякими шкідливими програмами.

Продовження таблиці 1.4.

Назва програми	Тип аналізу	Основні функції	Переваги	Недоліки та проблеми
IDA Pro	Статичний	Дизасемблер і відладчик для аналізу двійкових файлів.	Потужний аналізатор; велика спільнота та безліч плагінів.	Висока вартість (для комерційної версії); складний для нових користувачів.
Wireshark	Статичний	Аналізатор мережевих пакетів.	Підтримка великої кількості протоколів; можливість глибокого аналізу трафіку; відкритий вихідний код.	Може бути складним для непідготовлених користувачів; обмежений у функціональності аналізу шкідливих програм.

Можливі шляхи вдосконалення та інноваційні підходи

Адаптивні системи виявлення

Створення систем, здатних адаптуватися, може підвищити здатність виявляти нові та модифіковані варіанти шкідливого програмного забезпечення. Використання технологій штучного інтелекту і машинного навчання може допомогти у створенні динамічних моделей, здатних навчатися на основі нових даних і пристосовуватися до мінливих тактик атакувальників.

Автоматизація аналітичних процесів

Посилення ступеня автоматизації процесів аналізу для зменшення залежності від ручного аналізу експертів. Розроблення інструментів, здатних

автоматизувати складні аналітичні завдання, такі як реверс-інжиніринг і поведінковий аналіз, може значно підвищити ефективність процесів виявлення та реагування.

Інтеграція та кореляція даних

Спільне використання даних і висновків з різних джерел та інструментів аналізу може надати більш повне уявлення про шкідливе ПЗ і пов'язані з ним загрози. Кореляція даних між системами виявлення мережевого трафіку, статичного і динамічного аналізу може забезпечити глибше розуміння і можливість протистояти багатоступінчастим і складним атакам.

Хмарні технології та розподілений аналіз

Використання хмарних технологій і розподілених систем для проведення аналізу шкідливого ПЗ може забезпечити масштабованість і доступність ресурсів, необхідних для опрацювання великих обсягів даних і забезпечення швидкого часу реакції.

Колаборація та обмін даними

Формування платформ і стандартів для обміну даними про шкідливе ПЗ і загрози в спільноті забезпечить розподіл знань і даних, даючи змогу швидко адаптуватися до нових загроз. Створення загальних баз даних і платформ для обміну індикаторами компрометації (IoC), зразками шкідливого ПЗ і тактиками атакувальників може посилити колективні зусилля по боротьбі з кіберзагрозами.

Поліпшення користувацького досвіду та доступності інструментів

Розробка інструментів аналізу, які легше налаштовувати і використовувати, може розширити доступність і використання просунутих методів аналізу для ширшого кола фахівців і організацій.

Під час глибокого і багатоаспектного дослідження поточного стану шкідливого програмного забезпечення, був проведений огляд існуючих методологій, технологій та інструментів, які наразі активно застосовуються фахівцями в галузі кібербезпеки по всьому світу. Було встановлено, що, незважаючи на всю ефективність і просунутість наявних систем, існує низка

проблем і недоліків, які потребують уважного розгляду і, можливо, доопрацювання.

Виявлення нових загроз, мінімізація помилкових спрацьовувань, оптимізація використання обчислювальних ресурсів, спрощення інтерфейсів і процесів для користувачів, а також забезпечення адаптивності та масштабованості в умовах постійного розвитку цифрового простору, який постійно розвивається та ускладнюється, - всі ці аспекти було виділено як сфери, які потребують додаткових поліпшень та інновацій.

Було висунуто кілька можливих шляхів удосконалення, включно з, але не обмежуючись, активнішим впровадженням і використанням методів машинного навчання для підвищення рівня автоматизації та точності в процесі виявлення шкідливого ПЗ, а також для прогнозування і запобігання майбутнім загрозам. Наголошується на значущості колаборації та обміну інформацією між фахівцями та організаціями, а також забезпечення легкості доступу та використання інструментів аналізу шкідливого програмного забезпечення для розширення їх застосування та підвищення загального рівня кібербезпеки в мережі.

Усвідомлення вище зазначених проблем і потенційних шляхів їх вирішення забезпечує основу для розробки нового програмного модуля аналізу шкідливого програмного забезпечення, який буде розглянуто в наступних розділах дипломної роботи.

1.4 Висновки до першого розділу.

У рамках першого розділу цієї дипломної роботи було проведено дослідження, що висвітлює критичні аспекти і проблеми, пов'язані зі шкідливим програмним забезпеченням і методами його аналізу в сучасному цифровому світі. Шкідливі програми продовжують еволюціонувати, стаючи дедалі більш

витонченими та складними для виявлення й аналізу, що підкреслює абсолютну необхідність глибокого розуміння та дослідження цієї проблематики.

Детально розглянувши проблематику безпеки, мій огляд виокремив низку фундаментальних і водночас складних питань, що постають перед фахівцями з інформаційної безпеки. З урахуванням масштабів і рівня розвитку сучасних інформаційних технологій, ми стоїмо перед обличчям кіберзагроз, які стають дедалі численнішими та складнішими. Історичні приклади кібератак та аналіз поточної статистики підкреслюють важливість і актуальність проблеми, підсвічуючи різноманітні форми загроз та їхній негативний вплив на людей і організації.

Також ми розглянули класифікації шкідливого ПЗ, ми заглибилися в детальне вивчення різних типів, характеристик і класів шкідливих програм. Класифікація шкідливого програмного забезпечення дає змогу не тільки зрозуміти механізми роботи та наміри зловмисників, а й розробляти ефективні стратегії та методики для боротьби з конкретними типами загроз. Розглянуті приклади дали розуміння про різноманіття і складність шкідливих програм, а також підкреслили важливість продовження досліджень у цій галузі.

Аналіз наявних методів виявлення та аналізу шкідливого програмного забезпечення виявив низку сильних і слабких сторін поточних практик та інструментів. За всієї ефективності наявних систем, вони недосконалі та стикаються з низкою проблем, як-от нездатність виявляти нові та невідомі загрози, високий ризик хибних спрацьовувань тощо.

У пидрозділі про шляхи поліпшення було запропоновано і детально розглянуто кілька напрямків удосконалення існуючих методів та інструментів аналізу шкідливого програмного забезпечення. Зокрема, було підкреслено важливість і перспективність застосування методів машинного навчання для поліпшення процесів виявлення та аналізу, а також для прогнозування та запобігання майбутнім загрозам.

Загалом, цей розділ сформував основу для подальшої роботи, виявивши ключові проблеми та напрямки для дослідження й розроблення в наступних розділах дипломного проєкту.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ЩКІДЛИВОГО ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

2.1 Огляд наявних підходів

Історичний розвиток методів аналізу

Вступ до історії аналізу шкідливого ПЗ.

- Ранні роки (1980-ті - 1990-ті роки): Цей період характеризується появою перших комп'ютерних вірусів і троянських програм. У цей час методи виявлення були досить примітивні і в основному ґрунтувалися на сигнатурному аналізі, який полягає в пошуку специфічних послідовностей даних, характерних для відомих вірусів.

- Розвиток інтернету (2000-ті роки): З появою інтернету масштаб і складність кіберзагроз значно зросли. Це призвело до розвитку нових методів виявлення, включно з поведінковим та евристичним аналізом. Ці методи дають змогу визначати шкідливе ПЗ за його поведінкою в системі, а не тільки за сигнатурами.

- Сучасна ера (2010-ті роки - теперішній час): Сучасна епоха кібербезпеки характеризується появою високоцільових атак, поліморфних і метаморфних вірусів. Методи виявлення шкідливого ПЗ стали складнішими, включно з використанням машинного навчання та аналізу в глибоких мережах.

Порівняльний аналіз різних підходів

Статичний аналіз.

- Опис: Статичний аналіз полягає у вивченні коду шкідливої програми без її виконання. Це включає в себе аналіз виконуваних файлів, скриптів і документів на предмет підозрілих патернів і сигнатур.

- Переваги: Головною перевагою є безпека, оскільки шкідливе ПЗ не виконується. Це також дозволяє отримати детальне уявлення про структуру коду і потенційні загрози.

- Недоліки: Основним недоліком статичного аналізу є його неефективність проти поліморфного і метаморфного ПЗ, яке змінює свій код при кожному виконанні, ускладнюючи виявлення за сигнатурами.

Динамічний аналіз.

- Опис: Динамічний аналіз включає в себе спостереження за поведінкою шкідливої програми під час її виконання в контрольованому середовищі. Це дає змогу спостерігати за її взаємодією з операційною системою, мережею та іншими програмами.

- Переваги: Основна перевага динамічного аналізу полягає в його ефективності у виявленні поведінкових патернів і взаємодій шкідливого ПЗ із системою.

- Недоліки: Основним недоліком є потенційний ризик для системи безпеки, а також складність і трудомісткість аналізу.

Машинне навчання і Штучний Інтелект.

- Опис: Застосування алгоритмів машинного навчання і штучного інтелекту включає в себе використання статистичних методів для автоматичного виявлення шкідливого ПЗ. Це може включати як навчання з учителем, так і без учителя, а також глибоке навчання для розпізнавання складних патернів.

- Переваги: Головна перевага полягає в здатності систем машинного навчання адаптуватися до нових загроз і ефективності в обробці та аналізі великих обсягів даних.

- Недоліки: Основним недоліком є залежність від якості та обсягу навчальних даних, а також потенційні помилкові спрацьовування і складність інтерпретації результатів.

Цей порівняльний аналіз підкреслює відмінності між основними методами виявлення шкідливого ПЗ, їхні переваги та недоліки, а також їхню застосовність у різних сценаріях.

2.1 Основи машинного навчання в контексті аналізу шкідливого ПЗ

Огляд методів машинного навчання

Категорії машинного навчання та їх застосування в аналізі шкідливого ПЗ

Навчання з учителем (Supervised Learning)

Приклади: Логістична регресія, дерева рішень, метод опорних векторів (SVM), нейронні мережі.

Застосування: Класифікація файлів на шкідливі та безпечні, ґрунтуючись на відомих характеристиках і поведінці.

Приклад у контексті: Навчання моделі на базі даних з розміченими шкідливими і нешкідливими програмами для автоматичного виявлення нових загроз.

Навчання без вчителя (Unsupervised Learning)

Приклади: Кластеризація методом k-середніх, ієрархічна кластеризація, метод головних компонент (PCA).

Застосування: Групування шкідливого ПЗ на основі схожості їхньої поведінки для виявлення нових, невідомих типів загроз.

Приклад у контексті: Використання кластеризації для ідентифікації нових варіантів шкідливого ПЗ на основі їхніх поведінкових характеристик.

Навчання з частковим залученням вчителя (Semi-supervised Learning)

Приклади: Змішання підходів з учителем і без учителя, графові методи, методи, засновані на припущеннях про розподіл даних.

Застосування: Ефективне використання як розмічених, так і нерозмічених даних для навчання моделей.

Приклад у контексті: Навчання моделі на великому наборі нерозмічених даних з невеликою кількістю розмічених прикладів для поліпшення виявлення шкідливого ПЗ.

Навчання з підкріпленням (Reinforcement Learning)

Приклади: Q-навчання, глибоке навчання з підкріпленням (наприклад, алгоритми, що використовують глибокі нейронні мережі).

Застосування: Автоматичне прийняття рішень і оптимізація процесів на основі зворотного зв'язку від середовища.

Приклад у контексті: Розробка системи, яка адаптується до мінливих тактик атаки і покращує свої алгоритми виявлення в реальному часі.

Таблиця 2.1.

Категорія	Опис	Переваги	Недоліки	Приклади застосування
Навчання з учителем	Навчання на основі розмічених даних	Висока точність у відомих сценаріях	Вимагає великої кількості розмічених даних	Класифікація шкідливого ПЗ
Навчання без учителя	Дослідження прихованих структур у даних	Не потребує розмічених даних	Може бути менш точним	Кластеризація шкідливого ПЗ
Навчання з частковим залученням учителя	Поєднання обох попередніх категорій	Ефективно при обмежених розмічених даних	Може потребувати складного налаштування	Виявлення нових типів загроз
Навчання з підкріпленням	Навчання на основі зворотного зв'язку	Адаптивність	Вимагає складного середовища для навчання	Адаптивні системи виявлення загроз

Цей огляд і порівняльна таблиця показують різноманітність підходів машинного навчання, які можуть бути застосовані для аналізу шкідливого ПЗ.

Кожен метод має свої унікальні переваги і може бути ефективно використаний у різних аспектах виявлення та аналізу кіберзагроз.

Застосування машинного навчання в кібербезпеці

У сучасному світі, де кіберзагрози стають дедалі більш витонченими та різноманітними, машинне навчання (МО) відіграє ключову роль у забезпеченні кібербезпеки. Ця технологія дає змогу не тільки виявляти відомі типи шкідливого ПЗ, а й прогнозувати нові загрози, аналізуючи великі масиви даних і навчаючись на їхній основі.

Ключові аспекти застосування МО в кібербезпеці

- Виявлення та аналіз шкідливого ПЗ

Опис: МО здатне обробляти й аналізувати великі обсяги даних, що важливо для виявлення шкідливого ПЗ, особливо в умовах його постійної еволюції. Моделі можуть навчатися на основі характеристик відомих вірусів і шкідливих програм, виявляючи аналогічні загрози в нових файлах.

Приклад: Алгоритми класифікації можуть аналізувати різні аспекти програмного коду, мережевого трафіку або поведінки системи для виявлення потенційно шкідливої активності.

- Прогнозування та запобігання атакам

Опис: Застосування МО для передбачення майбутніх кібератак стає дедалі актуальнішим. Моделі можуть аналізувати зразки минулих атак, тенденції та поточні загрози для прогнозування майбутніх векторів атаки.

Приклад: МО може використовуватися для аналізу часових рядів і шаблонів поведінки в мережі, виявляючи аномалії, які можуть передвіщати плановану кібератаку.

- Реагування на інциденти

Опис: Автоматизація реагування на інциденти з використанням МО може значно прискорити і поліпшити процес реагування на загрози. МО може допомогти ідентифікувати та ізолювати атаку, мінімізуючи збитки.

Приклад: Системи, засновані на МО, можуть автоматично блокувати підозрілий трафік або процеси, запобігаючи поширенню шкідливого ПЗ у мережі.

- Розпізнавання аномалій і поведінковий аналіз
- Опис: МО здатне виявляти аномальну поведінку, яка може вказувати на наявність шкідливого ПЗ або неавторизовані спроби доступу. Це включає аналіз поведінки користувачів і системи загалом.
- Приклад: Алгоритми навчання без вчителя можуть аналізувати мережевий трафік, виявляючи незвичні зразки, які можуть свідчити про кібератаку.

Технологічні та оперативні виклики

- Якість даних: Одним із ключових аспектів ефективності МО є якість навчальних даних. Некоректно підібрані або недостатньо репрезентативні дані можуть призвести до помилкових висновків моделей.
- Адаптація до нових загроз: Важливо, щоб системи на основі МО могли оперативно адаптуватися до нових типів загроз, які постійно розвиваються і змінюються.
- Інтерпретація результатів: Багато моделей МО можуть бути складними для розуміння та інтерпретації, особливо коли йдеться про глибокі нейронні мережі. Це може ускладнити аналіз причин прийняття того чи іншого рішення моделлю.

Машинне навчання є потужним інструментом в арсеналі кібербезпеки, надаючи можливості для більш ефективного і швидкого виявлення, аналізу та реагування на кіберзагрози. Однак для повноцінної ефективності необхідно враховувати і вирішувати низку технічних і оперативних проблем, пов'язаних із якістю даних, адаптивністю та інтерпретованістю моделей.

Аналіз методів машинного навчання: Математичні Основи

- Логістична регресія: Логістична регресія

Основи: Логістична регресія - це статистичний метод для аналізу набору даних, у якому одна або кілька незалежних змінних визначають результат.

Математичний принцип: Працює на принципі логістичної функції, часто званої сигмоїдом. Функція сигмоїда перетворює будь-яке реальне значення на значення між 0 і 1, що робить її ідеальною для бінарної класифікації.

Формула:$P = (Y = 1) = \frac{1}{1+e-(\beta_0+\beta_1 X)}$, де $P = (Y = 1)$ - ймовірність того, що результат дорівнює 1, β^0, β_1 - параметри моделі, X - незалежна змінна, e - основа натурального логарифма.

- Випадковий Ліс

Основи: Випадковий ліс - це метод ансамблевого навчання, заснований на агрегуванні результатів безлічі дерев рішень.

Математичний принцип: Кожне дерево будується на випадковій вибірці з даних із заміною (bootstrap sample), і вибір ознак для поділу вузлів також відбувається випадковим чином. Рішення ансамблю базується на більшості голосів або середньому передбаченні окремих дерев.

Формула: Немає однієї уніфікованої формули, оскільки це ансамбль безлічі дерев. Прогнозований клас - це клас, який найчастіше вибирається деревами в ансамблі.

- Градієнтний бустинг

Основи: Градієнтний бустинг - це техніка машинного навчання для задач регресії та класифікації, яка будує модель у формі ансамблю слабких предиктивних моделей, зазвичай дерев рішень.

Математичний принцип: Метод полягає в послідовному додаванні до ансамблю нових моделей, які виправляють помилки попередніх моделей. Алгоритм мінімізує функцію втрат, використовуючи метод градієнтного спуску.

Формула: Ґрунтується на послідовному зменшенні значення функції втрат L(Y,F(x)), де Y - істинне значення, F(x) - передбачення моделі.

- Метод Опорних Векторів (SVM)

Основи: SVM - це метод, що використовується для класифікації та регресії, який прагне знайти гіперплощину в N-вимірному просторі (N - кількість ознак), яка явно класифікує точки даних.

Математичний принцип: Гіперплощину вибирають таким чином, щоб максимізувати зазор між двома класами точок даних. SVM використовує ядра для перетворення даних у високорозмірний простір, де можна знайти оптимальну межу між класами.

Формула: Основна ідея полягає в розв'язанні оптимізаційної задачі $\min w, b \frac{1}{2} \mid\mid w \mid\mid^2$, з умовами $y_i(w * x_i + b) \geq 1$ для всіх *i,* де *w* -ваговий вектор, b - зміщення, x_i - вхідні вектори, y_i - мітки класів.

Таблиця 2.2.

Метод	Математичний Опис
Логістична Регресія	Використовує логістичну функцію (сигмоїду) для моделювання ймовірності належності до певного класу. Заснована на принципі максимізації правдоподібності, де шукають такий набір ваг, за якого спостережувані дані найімовірніші.
Випадковий Ліс	Комбінація безлічі дерев рішень, де кожне дерево навчається на випадково обраній підвибірці даних. Рішення ухвалюється шляхом 'голосування' дерев для задач класифікації або середнього передбачення для регресії. Додавання випадковості зменшує кореляцію між деревами, підвищуючи надійність моделі.
Градієнтний Бустінг	Послідовна побудова набору слабких прогностичних моделей (найчастіше дерев рішень), де кожне наступне дерево прагне виправити помилки попереднього. Використовує техніку градієнтного спуску для мінімізації функції втрат. Відрізняється високою гнучкістю і здатністю до точного моделювання складних залежностей.

Продовження таблиці 2.2.

Метод	Математичний Опис
Метод Опорних Векторів (SVM)	Прагне знайти оптимальну роздільну гіперплощину між класами, максимізуючи зазор між ними. Застосовує трюк для перетворення даних у простір вищої розмірності, що дає змогу ефективніше розділяти класи, які не можна розділити лінійно у вихідному просторі.

2.3 Методи нормалізації та підготовки даних

Основи нормалізації даних

Нормалізація даних відіграє найважливішу роль у процесі машинного навчання, особливо в контексті аналізу шкідливого програмного забезпечення. Процес нормалізації охоплює перетворення сирих даних у формат, який спрощує оброблення й аналіз даних, забезпечуючи точніші та надійніші результати.

Нормалізація даних необхідна з кількох причин:

- Уніфікація масштабів: Різні ознаки можуть бути виміряні в різних масштабах, що може призвести до незбалансованості в процесі навчання моделей.
- Поліпшення конвергенції алгоритмів: Багато алгоритмів машинного навчання, особливо ті, які використовують градієнтний спуск, працюють ефективніше на нормалізованих даних.
- Зниження впливу викидів: Нормалізація може допомогти знизити вплив аномально високих або низьких значень у даних.
- Поліпшення інтерпретованості моделі: Нормалізовані дані полегшують розуміння внеску кожної ознаки в навчену модель.

Методи нормалізації даних:

- Мінімаксна нормалізація

Цей метод приводить усі значення до діапазону від 0 до 1. Він особливо корисний, коли значення даних сильно варіюються.

Застосування: Особливо підходить для даних, де необхідно зберегти розподіл даних, але зменшити масштаб.

- Z-оцінка (стандартизація)

Стандартизація приводить дані до розподілу з нульовим середнім і одиничним стандартним відхиленням.

Застосування: Ідеально підходить для сценаріїв, де важливо зберегти відносини між значеннями, наприклад, під час опрацювання функцій, які слідують за гаусовим розподілом.

- Нормалізація за L1 і L2 нормами

L1 і L2 нормалізація зменшують суму абсолютних і квадратичних значень ознак відповідно.

Застосування: Ці методи корисні в ситуаціях, де важливо керувати величиною внеску кожної ознаки, наприклад, для запобігання перенавчання.

У сфері аналізу шкідливого ПЗ, де дані можуть бути надзвичайно різноманітними (від бінарних значень до складних структурних характеристик), нормалізація відіграє ключову роль. Вона дає змогу моделям машинного навчання ефективніше виявляти та класифікувати потенційно шкідливу поведінку.

Вибір відповідного методу нормалізації залежить від характеру даних і вимог завдання. Наприклад, якщо дані мають багато викидів, мінімаксна нормалізація може бути не найкращим вибором, тоді як стандартизація буде більш стійкою до таких викидів.

Ретельна нормалізація даних - ключовий етап у підготовці даних для аналізу шкідливого ПЗ. Вона забезпечує уніфікацію різноманітних даних, покращує навчання і загальну продуктивність моделей машинного навчання, а також допомагає в інтерпретації результатів, що важливо для ухвалення рішень у сфері кібербезпеки.

Техніки попередньої обробки даних для машинного навчання

Попередня обробка даних - це критична фаза в машинному навчанні, особливо під час аналізу шкідливого програмного забезпечення, оскільки якість і відповідний формат даних безпосередньо впливають на ефективність і точність моделей.

Попередня обробка даних необхідна для:

- Усунення шуму і помилок: Шумові дані можуть спотворити результати моделі.
- Обробки пропущених значень: Пропуски можуть призвести до неточностей в аналізі та висновках.
- Перетворення типів даних: Узгодження різних типів даних покращує сумісність з алгоритмами машинного навчання.

Основні техніки попереднього оброблення даних:

- Очищення даних

Що включає: Видалення або корекція помилкових, неповних або несуттєвих даних.

Приклади: виправлення помилок і логічних помилок у даних, видалення дублікатів, які можуть спотворити результати аналізу.

- Обробка пропущених значень

Методи:використання середнього/медіани/моди для заповнення пропусків, застосування алгоритмів передбачення для оцінки відсутніх значень.

Важливість: Дозволяє зберегти цілісність і повноту набору даних.

- Перетворення категоріальних даних

Техніки: One-Hot Encoding - перетворення категоріальних даних у бінарні стовпці, label Encoding - Присвоєння кожній категорії унікального числового значення.

Значення: Забезпечує оброблення нечислових даних алгоритмами машинного навчання.

- Масштабування ознак

Мета: Приведення всіх ознак до одного масштабу для поліпшення роботи алгоритмів.

Методи: мінімаксна нормалізація для стиснення даних у діапазон [0, 1], Z-оцінка для стандартизації даних з нульовим середнім і одиничним стандартним відхиленням.

- Генерація нових ознак (Feature Engineering)

Процес: Створення нових, більш інформативних ознак з наявних даних.

Приклади: Створення комбінованих ознак, що відображають взаємозв'язки між наявними даними, використання поліноміальних ознак для підвищення складності моделі.

- Зменшення розмірності

Техніки: аналіз головних компонент (PCA) для зменшення кількості ознак, зберігаючи при цьому більшу частину інформації, t-SNE для візуалізації багатовимірних даних у дво- або тривимірному просторі.

Користь: Зменшує складність моделі, прискорює навчання і допомагає уникнути перенавчання.

У контексті аналізу шкідливого ПЗ, попереднє опрацювання даних допомагає створити чистий, структурований та інформативний набір даних, який покращує точність і надійність машинного навчання під час виявлення та класифікації шкідливої поведінки.

Ретельне попереднє опрацювання даних є фундаментом для побудови ефективних моделей машинного навчання. Воно відіграє вирішальну роль у забезпеченні якості вихідних даних, що особливо важливо в делікатній і складній галузі кібербезпеки.

Важливість якості даних в аналізі шкідливого ПЗ

У світі кібербезпеки, де кожна деталь має значення, якість даних, використовуваних для аналізу шкідливого програмного забезпечення, відіграє ключову роль. Це схоже на складання складного пазла, де кожна частина повинна ідеально підходити, щоб отримати цілісну картину.

Якість даних безпосередньо впливає на здатність системи виявляти й аналізувати загрози. Неякісні або неповні дані можуть призвести до низки проблем:

Неправильні висновки: Погані дані можуть вводити в оману, призводячи до помилкових висновків.

Пропущені загрози: Якщо дані не охоплюють усі можливі сценарії, існує ризик пропустити критичні загрози.

Низька ефективність: Неякісні дані можуть погіршити продуктивність системи, роблячи її менш ефективною.

Процес забезпечення якості даних

Поліпшення якості даних - це не разова дія, а безперервний процес. Він включає в себе:

Ретельне очищення і перевірку даних: Це допомагає переконатися, що дані точні та релевантні.

Постійне оновлення та збагачення даних: З урахуванням того, що загрози постійно еволюціонують, дані також повинні регулярно оновлюватися.

Значення даних у контексті шкідливого ПЗ

У сфері аналізу шкідливого ПЗ, де кожна дрібниця може бути ключовою для виявлення загрози, якість даних стає особливо важливою. Це не тільки про точність даних, а й про їхню повноту, актуальність і релевантність.

Насамкінець, можна сказати, що якість даних в аналізі шкідливого ПЗ - це як якість інгредієнтів для шеф-кухаря. Тільки з використанням найкращих інгредієнтів можна приготувати страву вищого класу. Аналогічно, тільки з якісними даними можна розробити ефективні та надійні системи кібербезпеки, здатні протистояти найвитонченішим загрозам.

Вибір відповідних методів нормалізації для різних типів даних

Під час роботи з даними в аналізі шкідливого програмного забезпечення, необхідно уважно вибирати методи нормалізації. Цей вибір залежить від характеристик даних і цілей аналізу.

В аналізі шкідливого ПЗ ми стикаємося з різноманітними типами даних, включно з числовими, категоріальними, текстовими та часовими рядами. Кожен тип вимагає свого підходу до нормалізації.

Вибір методу нормалізації

Числові дані: Тут важливо враховувати розподіл даних. Якщо він близький до нормального, краще використовувати Z-оцінку. Для даних з яскраво вираженими викидами або нестандартним розподілом краще підходить мінімаксна нормалізація або перетворення Box-Cox.

Категоріальні дані: Одним із найпопулярніших методів є One-Hot Encoding, який перетворює категорії на бінарні стовпці. Важливо враховувати розмірність даних, щоб уникнути проблеми "прокляття розмірності".

Текстові дані: Тут застосовуються методи векторизації, як-от TF-IDF, які допомагають перетворити текст на числові значення, зберігаючи при цьому інформативність.

Специфіка роботи з даними в кібербезпеці

У сфері кібербезпеки дані часто містять складні та різноманітні шаблони. Тому вибір методу нормалізації має враховувати специфіку даних і завдань, що стоять перед аналітиками.

Правильний вибір методів нормалізації в аналізі шкідливого ПЗ - це ключ до успішного навчання моделей машинного навчання. Він забезпечує необхідну стандартизацію та обробку даних, що дає змогу моделям ефективніше навчатися і виявляти приховані загрози. Це, своєю чергою, підвищує загальну ефективність систем кібербезпеки, роблячи їх надійнішими і стійкішими до нових і еволюціонуючих загроз.

2.4 Критерії оцінювання ефективності моделей машинного навчання

Огляд метрик ефективності

У рамках оцінювання ефективності моделей машинного навчання, що застосовуються для аналізу шкідливого програмного забезпечення, ключову роль відіграють різні метрики. Ці метрики допомагають визначити, наскільки успішно модель здатна виявляти і класифікувати загрози.

- Точність (Accuracy)

Детальний опис: Точність є однією з найосновніших метрик, що вимірює частку правильних прогнозів (включно з істинно позитивними та істинно негативними результатами) відносно всіх прогнозів. Формула точності: (Істинно позитивні+Істинно негативні)/(Усього передбачень).

Контекст застосування: Хоча точність може бути корисною для загального оцінювання продуктивності моделі, вона може вводити в оману у випадках сильно незбалансованих класів, де більшість прикладів належать до одного класу.

- Матриця помилок (Confusion Matrix)

Детальний опис: Матриця помилок являє собою таблицю, що показує відмінності між фактичними і прогнозованими класифікаціями. Вона складається з чотирьох частин: істинно позитивних, істинно негативних, хибно позитивних і хибно негативних результатів.

Значення в аналізі: Ця метрика надає детальне уявлення про продуктивність моделі, даючи змогу краще зрозуміти, як модель помиляється і як вона правильно класифікує приклади.

- Точність (Precision) і Повнота (Recall)

Точність: Описує, яка частка передбачених позитивних результатів дійсно позитивна. Формула: Істинно позитивні/(Істинно позитивні + Хибно позитивні).

Повнота: Описує, яка частка реальних позитивних випадків була виявлена моделлю. Формула: Істинно позитивні/(Істинно позитивні + Хибно негативні).

Застосування і значення: Вони важливі для розуміння балансу між уникненням помилкових спрацьовувань (точність) і виявленням усіх позитивних випадків (повнота).

- F1-мір (F1 Score)

Детальний опис: F1-міра об'єднує точність і повноту в одну метрику, яка обчислюється як гармонійне середнє між ними. Формула: 2×(Точність×Повнота)/(Точність+Повнота).

Значимість: Ця метрика особливо корисна в ситуаціях, де необхідно враховувати обидві характеристики: точність і повноту.

- ROC AUC Score

Детальний опис: Площа під кривою робочих характеристик приймача (ROC AUC) вимірює здатність моделі розрізняти класи. Що вище значення AUC, то краще модель розрізняє позитивні та негативні випадки.

Важливість: Особливо корисна у випадках, коли інтерес становлять різні пороги класифікації.

- Звіт про класифікацію (Classification Report)

Детальний опис: Звіт про класифікацію являє собою синтез різних метрик, включно з точністю, повнотою і F1-мірою для кожного класу.

Роль в аналізі: Надає всебічний огляд роботи моделі, включно з її здатністю правильно класифікувати кожен клас.

Таблиця 2.3.

Метрика	Опис	Додаток
Точність (Accuracy)	Частка правильних передбачень від усіх передбачень.	Загальна оцінка продуктивності моделі; може бути такою, що вводить в оману при незбалансованих даних.

Продовження таблиці 2.3.

Матриця помилок (Confusion Matrix)	Таблиця, що показує істинно позитивні, істинно негативні, хибно позитивні та хибно негативні результати.	Надає детальний погляд на здатність моделі розрізняти класи.
Точність (Precision)	Частка істинно позитивних результатів з усіх передбачених позитивних.	Важливо для зниження хибно позитивних результатів.
Повнота (Recall)	Частка істинно позитивних результатів з усіх реальних позитивних випадків.	Ключова метрика для оцінки всіх позитивних випадків.
F1-міра (F1 Score)	Гармонійне середнє між точністю і повнотою.	Надає баланс між точністю та повнотою.
ROC AUC Score	Міра, що показує здатність моделі розрізняти класи, виражена через площу під кривою ROC.	Оцінка здатності моделі розрізняти класи за різних порогів.
Звіт про класифікацію (Classification Report)	Детальний аналіз різних метрик для кожного класу.	Комплексна оцінка продуктивності моделі за різними класами.

Метрики ефективності є невід'ємною частиною оцінювання моделей машинного навчання у сфері кібербезпеки. Вони не тільки допомагають оцінити поточну ефективність моделі, а й слугують важливим інструментом для її поліпшення і налаштування. Правильне розуміння і застосування цих метрик

дають змогу досягти більш точного та ефективного виявлення шкідливих програм, що є критично важливим для забезпечення кібербезпеки.

Специфіка оцінювання моделей у контексті кібербезпеки

Оцінювання моделей машинного навчання у сфері кібербезпеки має свої особливості, які відрізняються від інших сфер застосування машинного навчання. Це зумовлено унікальними вимогами та викликами, характерними для завдань кібербезпеки.

Особливості оцінювання моделей.

- Незбалансованість даних

  Опис: У сфері кібербезпеки часто зустрічається ситуація, коли кількість нормальних (безпечних) подій значно перевищує кількість шкідливих дій. Це створює незбалансованість у наборі даних, що може призвести до високої точності моделі, але низької здатності виявляти реальні загрози.

  Вплив на оцінку: Необхідно приділяти особливу увагу метрикам, таким як повнота (Recall) і F1-мера, які допомагають краще розуміти здатність моделі виявляти загрози в умовах незбалансованих даних.

- Швидка зміна загроз

  Контекст: У сфері кібербезпеки загрози постійно еволюціонують, і шкідливе ПЗ регулярно оновлюється, щоб уникнути виявлення. Це вимагає від моделей здатності швидко адаптуватися до нових загроз.

  Вплив на оцінку: Важливо регулярно оновлювати і повторно оцінювати моделі на основі новітніх даних, щоб переконатися в їхній актуальності та ефективності.

- Складність помилкових спрацьовувань

  Опис: Помилкові спрацьовування (хибно позитивні результати) можуть бути особливо проблематичними в кібербезпеці, оскільки вони можуть призвести до непотрібних тривог і перевантаження системи безпеки.

  Вплив на оцінку: Крім загальної точності, важливо оцінювати точність (Precision) моделі для мінімізації помилкових спрацьовувань.

- Динамічність середовища

Контекст: Кібербезпека характеризується середовищем, що швидко змінюється, де нові типи атак з'являються регулярно.

Вплив на оцінку: Важливо оцінювати гнучкість і адаптивність моделі, а також її здатність до навчання на основі нових і мінливих даних.

У контексті кібербезпеки, оцінювання моделей машинного навчання вимагає особливого підходу, що враховує унікальні аспекти цієї сфери. Необхідно приділяти увагу не тільки загальній точності моделей, а й їхній здатності адаптуватися до нових загроз, мінімізувати хибні спрацьовування і працювати з незбалансованими даними.

Важливість різних метрик залежно від завдання

Вибір метрик для оцінювання моделей машинного навчання в кібербезпеці сильно залежить від конкретного завдання, яке ця модель має вирішувати. Різні завдання вимагають акцентування уваги на різних аспектах продуктивності моделі.

Виявлення шкідливого ПЗ

Повнота (Recall): Особливо важлива, оскільки пріоритетом є виявлення якомога більшої кількості загроз.

Точність (Precision): Також важлива для зменшення кількості помилкових спрацьовувань.

F1-міра: Надає баланс між точністю і повнотою.

Профілактика атак

Точність: Ключова метрика, оскільки важливо мінімізувати помилкові спрацьовування, які можуть викликати непотрібні тривоги.

ROC AUC Score: Важливий для оцінювання здатності моделі розрізняти нормальну поведінку й атаки за різних порогів.

Класифікація типів загроз

Звіт про класифікацію: Надає детальне уявлення про продуктивність моделі за окремими категоріями загроз.

Матриця помилок: Допомагає зрозуміти, які типи загроз найчастіше неправильно класифікуються.

Передбачення вразливостей

Точність і повнота: Обидві важливі для забезпечення точного і повного виявлення потенційних вразливостей.

F1-мір: Надає компроміс між точністю і повнотою.

Вибір метрик для оцінювання моделей машинного навчання в кібербезпеці залежить від специфічних цілей і завдань. Розуміння того, які метрики найбільш релевантні для конкретного завдання, є ключовим для ефективного розроблення та оцінювання моделей. Важливо враховувати, що в різних сценаріях акценти в оцінюванні можуть істотно відрізнятися, підкреслюючи необхідність гнучкого підходу до вибору та інтерпретації метрик ефективності.

2.5 Висновки до розділу

Узагальнення та ключові моменти

У другому розділі було розглянуто різноманітні аспекти та підходи, пов'язані із застосуванням машинного навчання для аналізу шкідливого програмного забезпечення. Зазначено важливість правильного вибору методів машинного навчання, підходів до нормалізації та попередньої обробки даних, а також критеріїв для оцінювання ефективності моделей.

Огляд методів машинного навчання показав, що різні алгоритми, включно з логістичною регресією, випадковим лісом, градієнтним бустингом і методом опорних векторів, мають свої унікальні переваги та недоліки, залежно від контексту їх застосування.

Важливість нормалізації та попередньої обробки даних підкреслено як критичний етап, що впливає на якість і ефективність моделей машинного навчання. Особливо в контексті кібербезпеки, де дані часто характеризуються великою різноманітністю та складністю.

Специфіка оцінювання моделей у контексті кібербезпеки зумовлена необхідністю справлятися з унікальними викликами, як-от незбалансовані дані, патерни загроз, що швидко змінюються, і важливість зниження помилкових спрацьовувань.

Важливість різних метрик залежно від завдання дає змогу точніше оцінювати і налаштовувати моделі для конкретних цілей у сфері кібербезпеки, від виявлення шкідливого ПЗ до профілактики атак і класифікації загроз.

Висновки цього розділу підкреслюють значущість комплексного підходу до аналізу шкідливого програмного забезпечення з використанням машинного навчання. Кожен етап - від вибору алгоритму до оцінки його ефективності - відіграє ключову роль у створенні надійної та ефективної системи кібербезпеки. Ці висновки і знання стануть основою для практичної частини дипломної роботи, де буде розроблено і протестовано конкретний програмний модуль для аналізу шкідливого програмного забезпечення.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО МОДУЛЯ

3.1 Опис середовища розробки рішення

Мова програмування: Python

Вибір: Python був обраний за його чудову підтримку в галузі машинного навчання та аналізу даних, завдяки великому набору високоякісних бібліотек. Це робить його ідеальним для швидкого прототипування та ефективної розробки.

Приклади з коду: Використання Python дало змогу спростити безліч завдань, наприклад:

```
import pandas as pd
data = pd.read_csv('data.csv')
```

Інтегроване середовище розробки: Jupyter Notebook

Вибір: Jupyter Notebook пропонує інтерактивне середовище, яке спрощує експериментування та візуалізацію. Це ідеально підходить для розробки моделей машинного навчання, де необхідно швидко ітерувати та візуалізувати результати.

Приклади з коду: У Jupyter Notebook можна легко візуалізувати дані:

```
%matplotlib inline
import matplotlib.pyplot as plt
data.hist(bins=50, figsize=(20,15))
plt.show()
```

Бібліотеки машинного навчання: Scikit-learn

Вибір: Scikit-learn обрано за його широкий спектр алгоритмів машинного навчання, попередньої обробки даних та інструментів оцінювання. Це спрощує процес вибору та тестування різних моделей.

Приклади з коду: Приклад використання Scikit-learn для навчання моделі:

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X_train, y_train)

Бібліотеки для роботи з даними: Pandas і NumPy

Вибір Pandas: Pandas надає широкі можливості для роботи з табличними даними, включно із завантаженням, очищенням, трансформацією та аналізом даних.

Вибір NumPy: NumPy використовується для складніших математичних операцій і роботи з масивами, що критично для обробки даних у машинному навчанні.

Приклади з коду: Обробка даних за допомогою Pandas:

data['feature'] = data['feature'].fillna(data['feature'].mean())

Бібліотеки для візуалізації даних: Matplotlib і Seaborn

Вибір: Ці бібліотеки дають змогу створювати різні види графіків і діаграм, що необхідно для аналізу даних та інтерпретації результатів моделей.

Приклади з коду: З використанням Seaborn можна швидко створити інформативні візуалізації:

import seaborn as sns

sns.pairplot(data, hue='target')

Архітектура системи

Під час розроблення програмного модуля для аналізу шкідливого програмного забезпечення було обрано архітектуру, орієнтовану на гнучкість, продуктивність і зручність використання. Ключові компоненти архітектури включають:

Попередня обробка даних

Завантаження даних: Використовується Pandas для завантаження даних із різних джерел, включно з CSV-файлами та базами даних. Наприклад:

data = pd.read_csv('dataset.csv')

Очищення даних: Видалення або корекція некоректних, відсутніх або аномальних значень. Наприклад, виправлення помилок у даних або заповнення пропущених значень середніми значеннями:

data.fillna(data.mean(), inplace=True)

Трансформація даних: Перетворення даних у формат, придатний для аналізу. Це може включати кодування категоріальних змінних і нормалізацію числових даних.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data_scaled = scaler.fit_transform(data)

Поділ даних: Поділ набору даних на навчальну і тестову вибірки для перевірки ефективності моделей.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2)

Навчання моделей

Вибір моделей: Різні алгоритми машинного навчання зі Scikit-learn використовуються для навчання моделей. Кожна модель тестується для визначення її ефективності.

Оцінка моделей: Після навчання моделі оцінюються з використанням тестових даних. Використовуються різні метрики, як-от точність, повнота і F1-міра.

from sklearn.metrics import classification_report

y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))

Інтерфейс користувача (GUI)

Розробка інтерфейсу: Використання Tkinter для створення простого та інтуїтивно зрозумілого інтерфейсу, який дає змогу користувачам завантажувати дані, запускати аналіз і переглядати результати.

Взаємодія з моделлю: GUI інтегрований з основною логікою додатка, дозволяючи користувачеві взаємодіяти з моделями машинного навчання в реальному часі.

Інтеграція компонентів

Зв'язок між компонентами: Уся система спроектована таким чином, щоб забезпечити безперебійний потік даних між компонентами. Це охоплює передачу даних від попереднього оброблення до моделей і виведення результатів у GUI.

Модульність і розширюваність: Архітектура системи дає змогу легко додавати нові моделі або методи попередньої обробки даних, забезпечуючи гнучкість для майбутніх поліпшень.

Кожен із цих компонентів відіграє важливу роль у загальній архітектурі системи, забезпечуючи не тільки ефективне опрацювання та аналіз даних, а й зручність для користувача.

3.2 Основний функціонал запропонованого рішення

Обробка та аналіз вхідних даних

Опис: Система здатна автоматично обробляти й аналізувати завантажені дані, застосовуючи попередньо навчені моделі машинного навчання.

Приклад із коду:

```
def predict():
    input_str = text.get("1.0", "end-1c")
  feature_values = input_str.split(',')
```

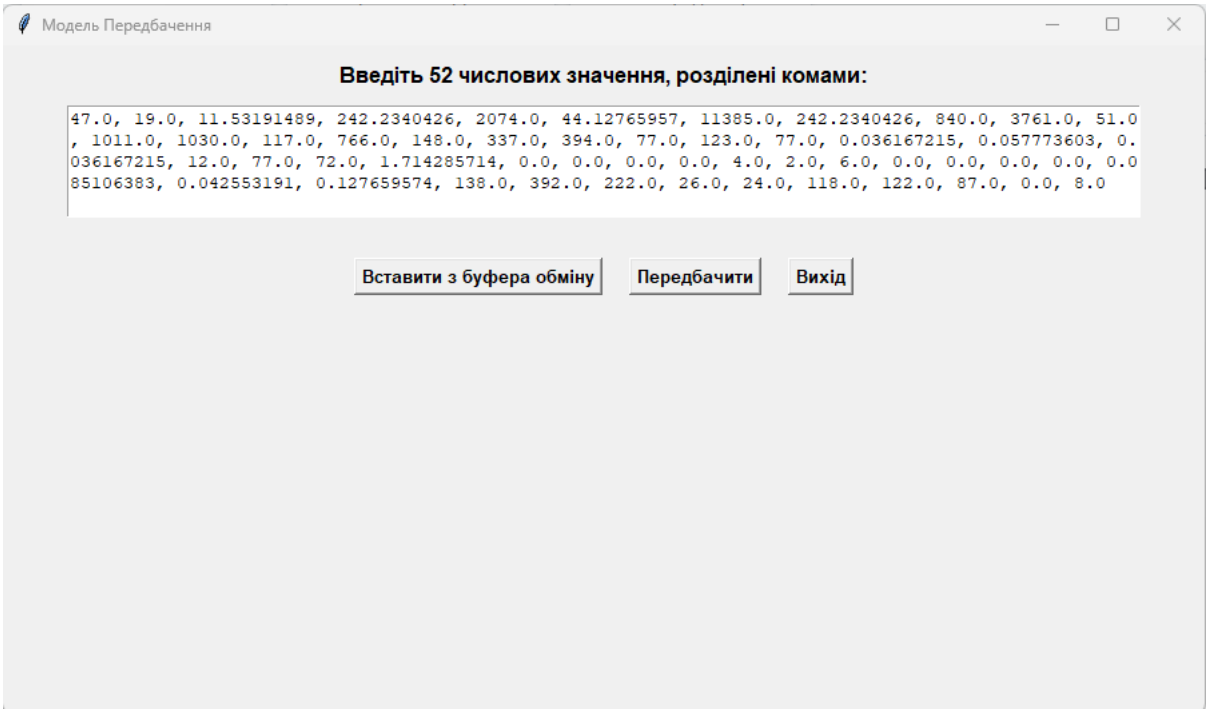


Рис.3.1. Вікно із введеним дампом пам'яті

output_label = tk.Label(bottom_frame, text="", font=label_font)

output_label.pack()

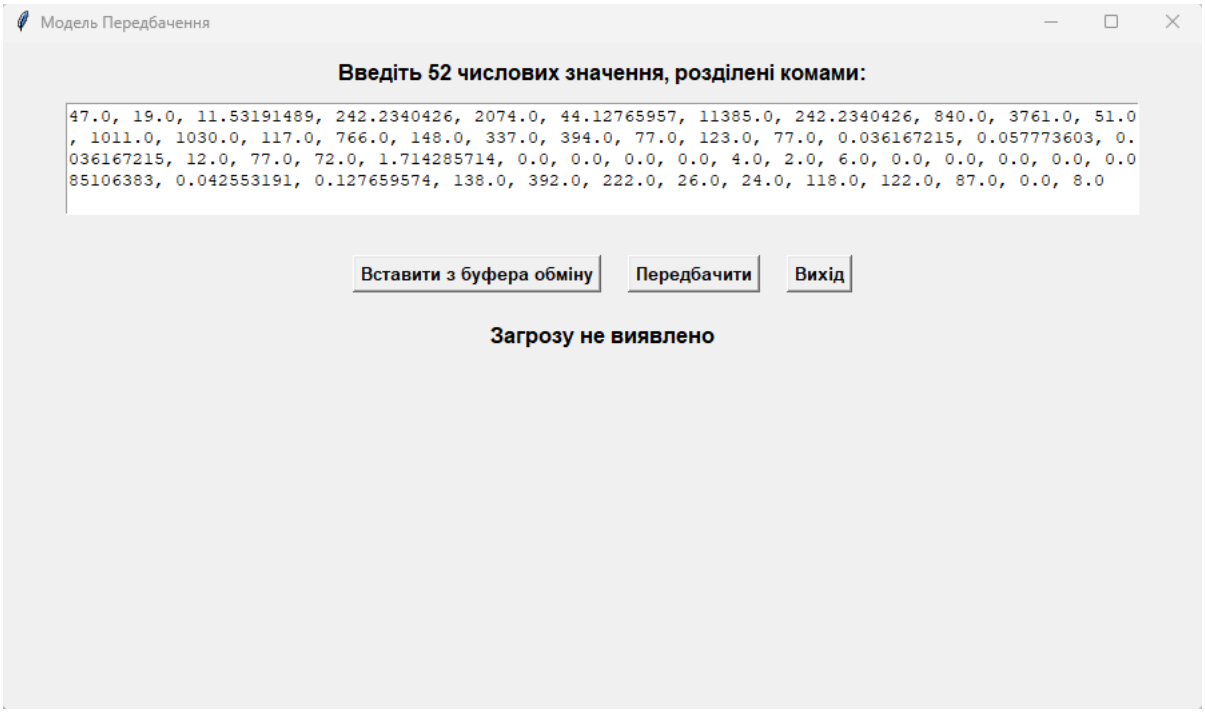


Рис.3.2. Вікно з результатом

Користь: Це дає змогу автоматизувати процес визначення потенційно шкідливих програм, прискорюючи реакцію на загрози і підвищуючи точність класифікації.

Інтерактивний користувальницький інтерфейс

Опис: Програма містить користувальницький інтерфейс, що дає змогу зручно керувати процесом аналізу та переглядати результати.

Приклад із коду:

```
# GUI для выбора файла и запуска анализа
root = tk.Tk()
root.filename = filedialog.askopenfilename()
analyze(root.filename)
```

Гнучкість і Масштабованість

Опис: Систему спроектовано таким чином, щоб забезпечити легку масштабованість і можливість інтеграції нових моделей і функцій.

Користь: Це забезпечує довгострокову перспективу розвитку й адаптації системи до мінливих умов і вимог.

Функціональні можливості забезпечують комплексний підхід до аналізу шкідливого ПЗ, роблячи програмний модуль потужним інструментом у сфері кібербезпеки.

3.3 Попередня обробка даних і тестування

Розбір попередньої обробки даних

1. Використання df.describe()

Крок перший в аналізі даних полягав у використанні функції df.describe() для отримання зведеної статистики за набором даних. Це включало середні значення, стандартні відхилення, мінімуми і максимуми для кожної ознаки.

Ця статистика допомогла в розумінні загального розподілу даних, виявленні аномальних значень і розумінні масштабу кожної ознаки.

Таблицю наведено в додатку С.

2. Побудова Heatmap

Для візуального аналізу кореляцій між різними ознаками було побудовано теплову карту (heatmap) кореляцій.

Використовуючи бібліотеку Seaborn, було створено графік, який показував, наскільки сильно кожна ознака корелює з іншими. Це допомогло у виявленні ознак, які могли дублювати інформацію або не мати значного впливу на цільову змінну.

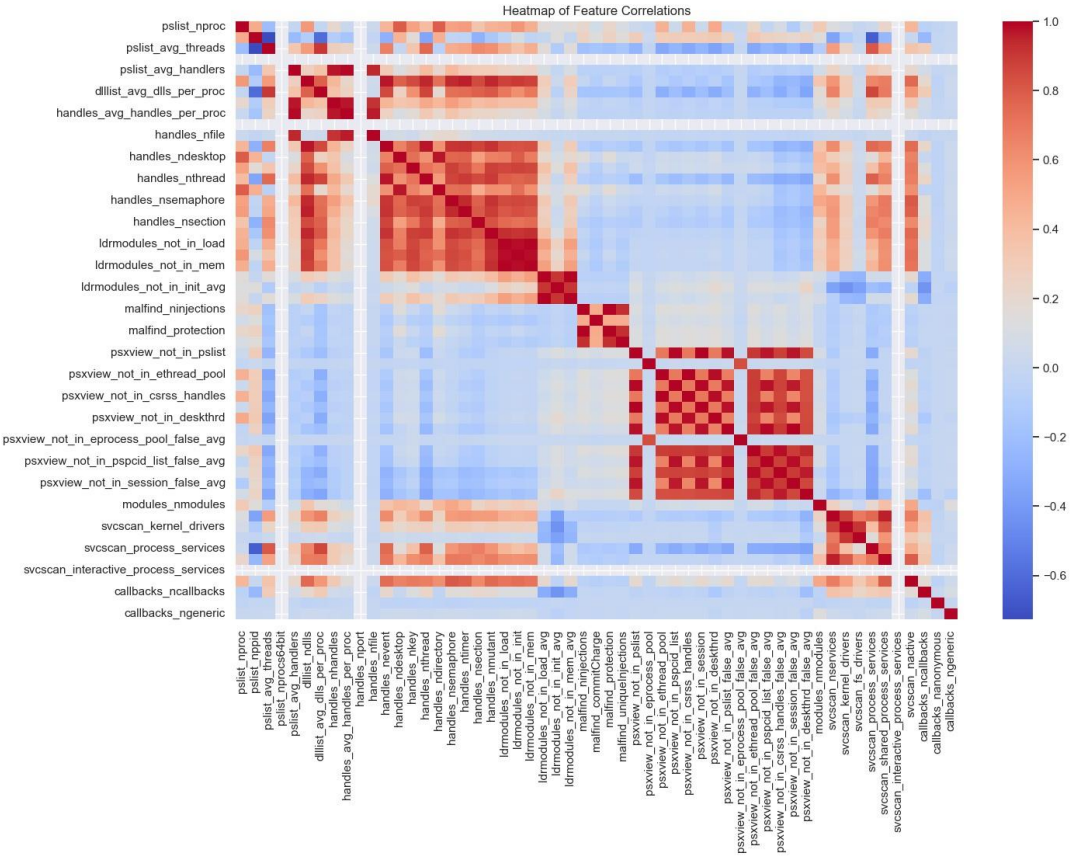


Рис.3.3. Теплова карта (heatmap)

3. Видалення нерелевантних ознак

Ґрунтуючись на аналізі heatmap, було видалено такі ознаки: 'pslist_nprocs64bit', 'svcscan_interactive_process_services', 'handles_nport', 'Raw_Type', 'SubType'.

Видалення цих ознак було здійснено, щоб зменшити розмірність даних і поліпшити здатність моделей до навчання, позбуваючись шуму та надлишкової інформації.

4. Перевірка балансу класів

Щоб переконатися, що класи в даних були збалансовані, використовувалася функція df.value_counts('Label').

```
Label
Benign     29298
Malware    29298
Name: count, dtype: int64
```

Рис.3.4. Перевірка балансу класів

Це було важливо для уникнення зсуву моделі в бік класів, які частіше зустрічаються, що могло призвести до неправильної класифікації рідкісних, але важливих випадків.

Перевірка показала, що класи були відносно збалансовані, що забезпечило більш ефективне і точне навчання моделей.

5. Нормалізації числових ознак

Для нормалізації числових ознак було обрано StandardScaler із бібліотеки Scikit-learn. Цей метод було обрано через його здатність стандартизувати ознаки, зменшуючи таким чином вплив різних масштабів ознак на навчання моделей.

Принцип роботи StandardScaler:

StandardScaler трансформує кожну ознаку, видаляючи середнє значення (центрування) і масштабуючи його до одиничної дисперсії.

Це робиться шляхом віднімання середнього значення кожної ознаки і ділення на стандартне відхилення.

Нормалізація важлива, оскільки багато алгоритмів машинного навчання чутливі до масштабу ознак. Наприклад, методи, засновані на вимірюванні відстаней між точками даних (як у SVM або k-NN), будуть некоректно працювати, якщо ознаки перебувають у різних масштабах.

Нормалізація також допомагає прискорити процес навчання, оскільки покращує числову стабільність алгоритмів.

Ці кроки передобробки даних забезпечили якісну підготовку набору даних до подальшого аналізу та навчання моделей машинного навчання, мінімізувавши ризики, пов'язані з перенавчанням, і підвищивши загальну точність передбачень.

Опис алгоритмів машинного навчання та їхніх результатів тестування

Модель: Логістична Регресія (Logistic Regression)

Опис моделі:

Логістична регресія - це статистичний метод для аналізу набору даних, у якому є одна або кілька незалежних змінних, що визначають результат. Особливо ефективна для бінарної класифікації.

У цьому дослідженні модель використовувалася для передбачення ймовірності належності об'єкта до класу шкідливого ПЗ.

```
Модель: LogisticRegression
Accuracy: 0.9988907849829352
Матриця помилок:
 [[5784    6]
 [   7 5923]]
Precision: 0.998988024962051
Recall: 0.9988195615514334
F1-score: 0.9989037861539759
ROC AUC Score: 0.9999918449848113
Звіт про класифікацію::
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5790
           1       1.00      1.00      1.00      5930

    accuracy                           1.00     11720
   macro avg       1.00      1.00      1.00     11720
weighted avg       1.00      1.00      1.00     11720
```

Рис.3.5. Результати тестування Logistic Regression

Модель: Випадковий Ліс (Random Forest Classifier)

Опис моделі:

Випадковий ліс - це ансамблевий метод машинного навчання, що використовує безліч дерев рішень для отримання більш надійного і точного результату.

Застосування в контексті нашого дослідження спрямоване на виявлення широкого спектра патернів шкідливого ПЗ.

```
Модель: RandomForestClassifier
Accuracy: 0.999914675767918
Матриця помилок:
 [[5789    1]
 [   0 5930]]
Precision: 0.9998313943685719
Recall: 1.0
F1-score: 0.9999156900767221
ROC AUC Score: 1.0
Звіт про класифікацію::
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5790
           1       1.00      1.00      1.00      5930

    accuracy                           1.00     11720
   macro avg       1.00      1.00      1.00     11720
weighted avg       1.00      1.00      1.00     11720
```

Рис.3.6. Результати тестування Random Forest Classifier

Модель: Градієнтний Бустинг (Gradient Boosting Classifier)

Опис моделі:

Градієнтний бустінг - це техніка машинного навчання для задач регресії та класифікації, яка будує модель у формі ансамблю слабких прогностичних моделей.

У цьому дослідженні її використовували для виявлення складних структур у даних про шкідливе ПЗ.

```
Модель: GradientBoostingClassifier
Accuracy: 0.9995733788395904
Матриця помилок:
 [[5790    0]
 [   5 5925]]
Precision: 1.0
Recall: 0.9991568296795953
F1-score: 0.9995782370307886
ROC AUC Score: 0.9999945244898019
Звіт про класифікацію::
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5790
           1       1.00      1.00      1.00      5930

    accuracy                           1.00     11720
   macro avg       1.00      1.00      1.00     11720
weighted avg       1.00      1.00      1.00     11720
```

Рис.3.7. Результати тестування Gradient Boosting Classifier

Модель: Метод Опорних Векторів (Support Vector Classifier, SVC)

Опис моделі:

Метод опорних векторів - це набір схожих алгоритмів машинного навчання, що використовуються для класифікації та регресії. У контексті цього дослідження, SVC використовується для поділу даних на категорії шкідливого і нешкідливого ПЗ.

```
Модель: SVC
Accuracy: 0.9992320819112628
Матриця помилок:
 [[5786    4]
 [   5 5925]]
Precision: 0.9993253499747006
Recall: 0.9991568296795953
F1-score: 0.9992410827219832
ROC AUC Score: 0.9999987476226674
Звіт про класифікацію::
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5790
           1       1.00      1.00      1.00      5930

    accuracy                           1.00     11720
   macro avg       1.00      1.00      1.00     11720
weighted avg       1.00      1.00      1.00     11720
```

Рис.3.8. Результати тестування Support Vector Classifier

Таблиця 3.1.

Модель	Accuracy	Precision	Recall	F1-міра	ROC AUC Score
Логістична Регресія	99.89%	99.90%	99.88%	99.89%	99.99%
Випадковий Ліс	99.99%	99.98%	100.00%	99.99%	100.00%
Градієнтний Бустинг	99.96%	100.00%	99.92%	99.96%	99.99%
Метод Опорних Векторів	99.92%	99.93%	99.92%	99.92%	100.00%

Ця таблиця являє собою порівняльний аналіз чотирьох різних моделей машинного навчання, які були протестовані для завдання класифікації шкідливого програмного забезпечення. Кожну модель оцінювали на основі п'яти ключових метрик ефективності.

З таблиці видно, що модель "Випадковий Ліс" показала найкращі результати за більшістю метрик, особливо в плані точності та здатності правильно класифікувати позитивні випадки (шкідливі додатки), що робить її найкращим вибором для подальшого використання в аналізі шкідливого ПЗ.

3.4 Приклади використання: Демонстрація роботи програмного модуля на практичних прикладах.

Програмний модуль, розроблений у межах цієї дипломної роботи, призначений для аналізу та класифікації потенційно шкідливого програмного забезпечення на основі даних із дампа пам'яті. Процес використання модуля містить такі етапи:

Завантаження моделі та нормалізатора: Під час запуску програма автоматично завантажує навчену модель "random_forest_model" і нормалізатор "scaler".

scaler = load('scaler.joblib')

model = load('random_forest_model.joblib')

Введення даних із дампа пам'яті користувачем: Користувачеві надається можливість ввести 52 параметри, витягнуті з дампа пам'яті. Ці параметри є ключовими для аналізу та класифікації ПЗ.

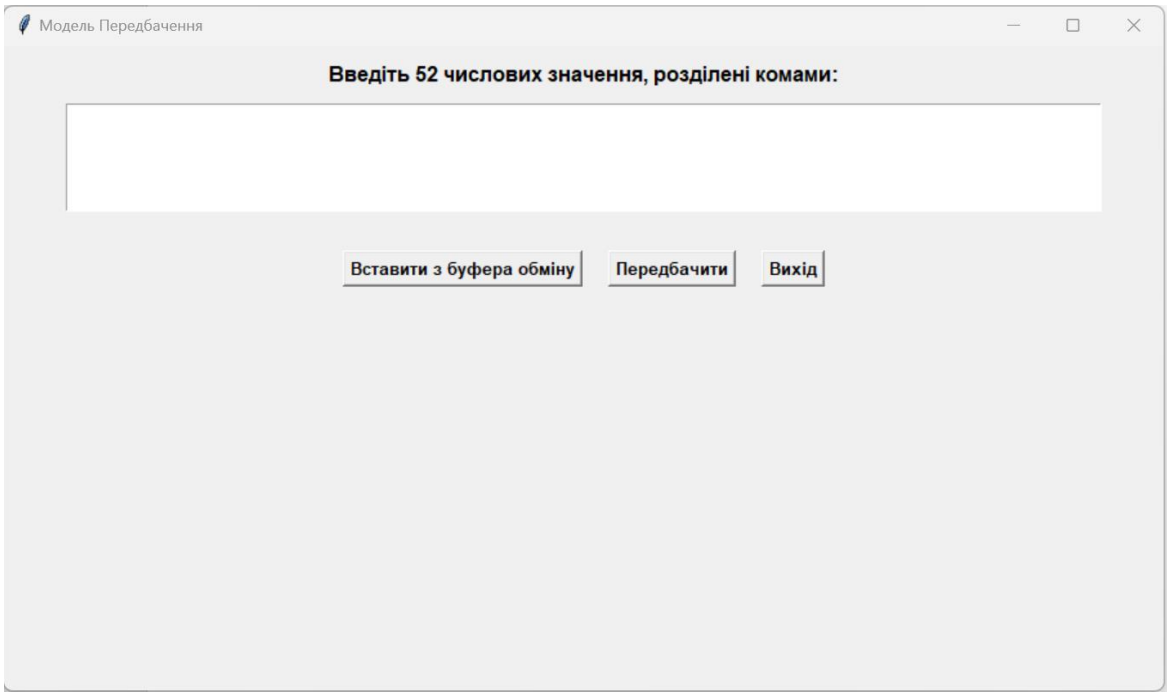


Рис.3.9. Стартовий екран програмного модуля

Перевірка коректності введених даних: Програма проводить валідацію введених даних. У разі виявлення помилок, користувачеві надається відповідне повідомлення:

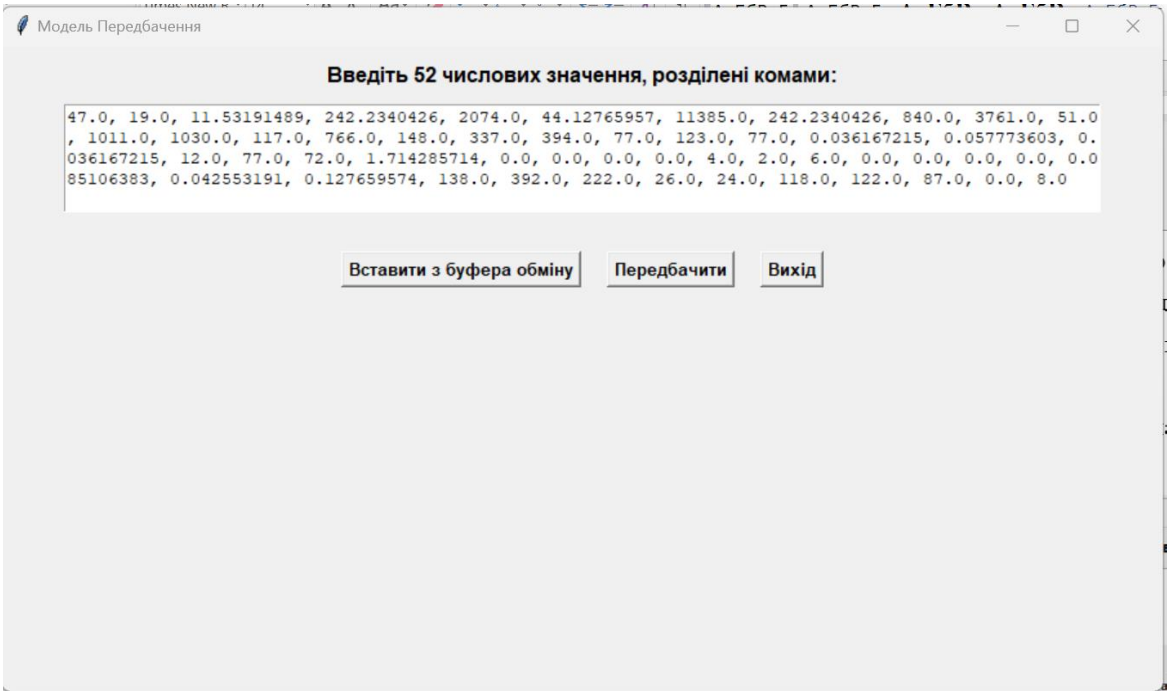


Рис.3.10. Введення коректних даних

Якщо кількість введених значень відрізняється від 52: "Помилка: введіть рівно 52 значення."

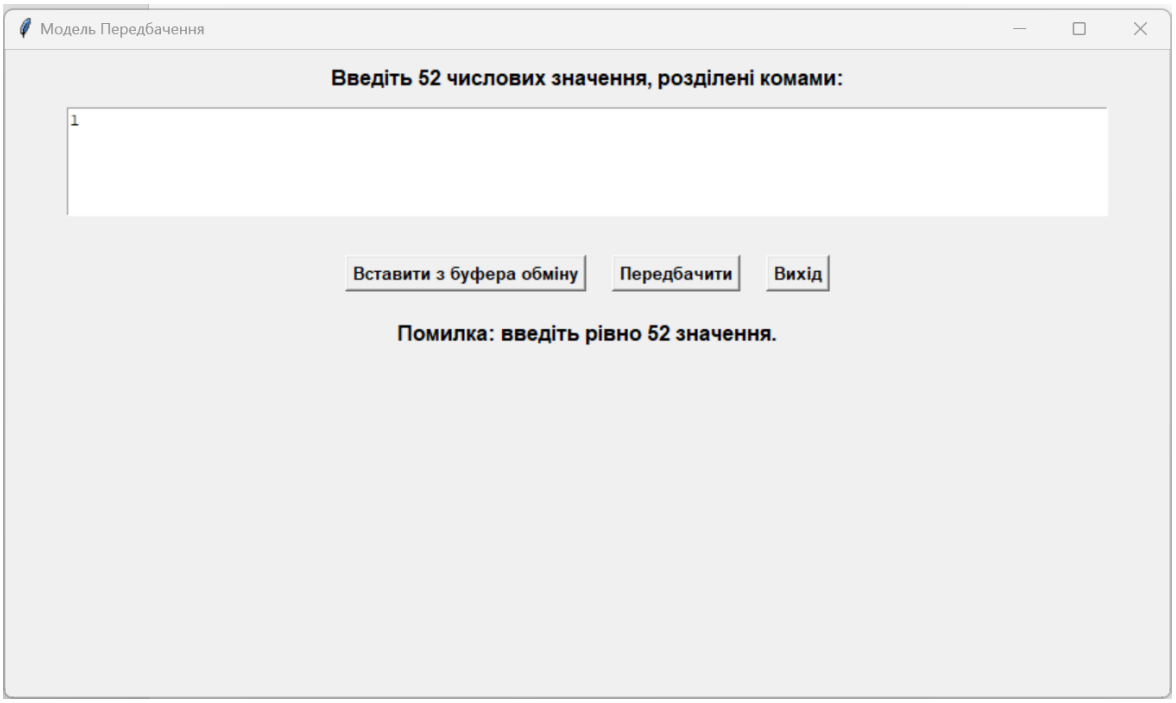


Рис.3.11. Помилка введення номер 1

Якщо введені нечислові значення: "Помилка введення: введіть числові значення."

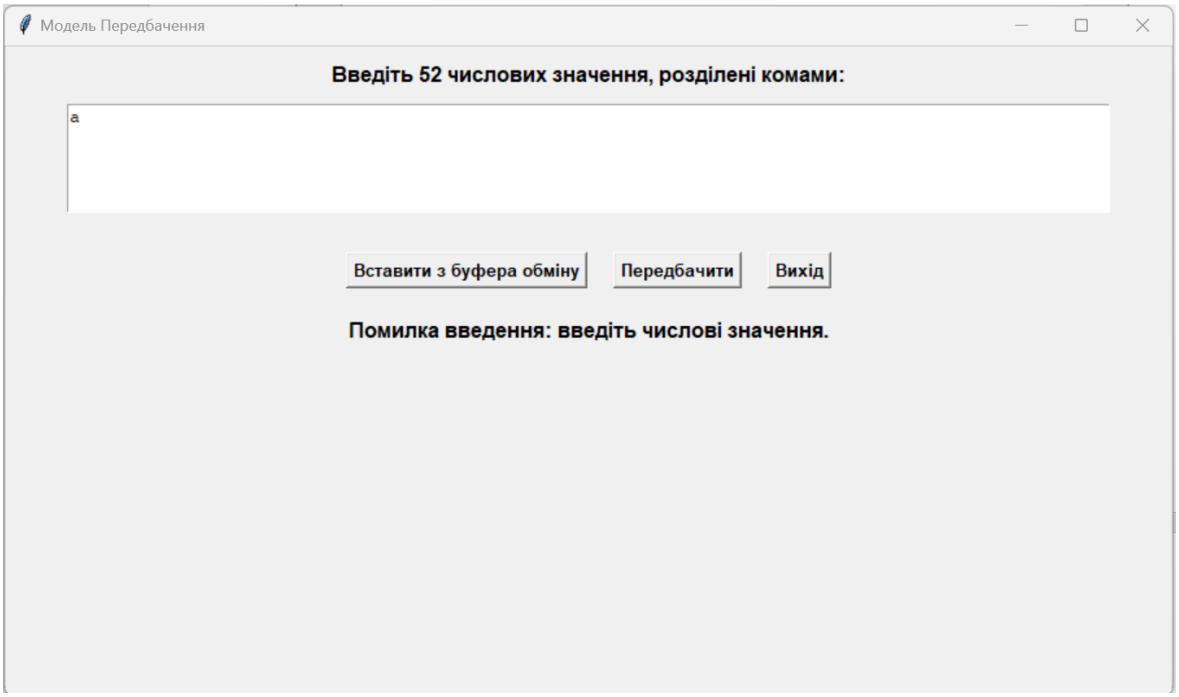


Рис.3.12. Помилка введення номер 2

Нормалізація та аналіз даних: Після перевірки коректності даних вони нормалізуються і подаються на вхід моделі для аналізу.

X_new = np.array([features])

X_new_scaled = scaler.transform(X_new)

prediction = model.predict(X_new_scaled)

Виведення результату: На основі аналізу даних моделлю програма виводить результат, що вказує на потенційну шкідливість аналізованого ПЗ.

Якщо програмний модуль знайшов загрозу, виходячи з аналізу дампа пам'яті, буде повідомлення : "Загрозу виявлено"

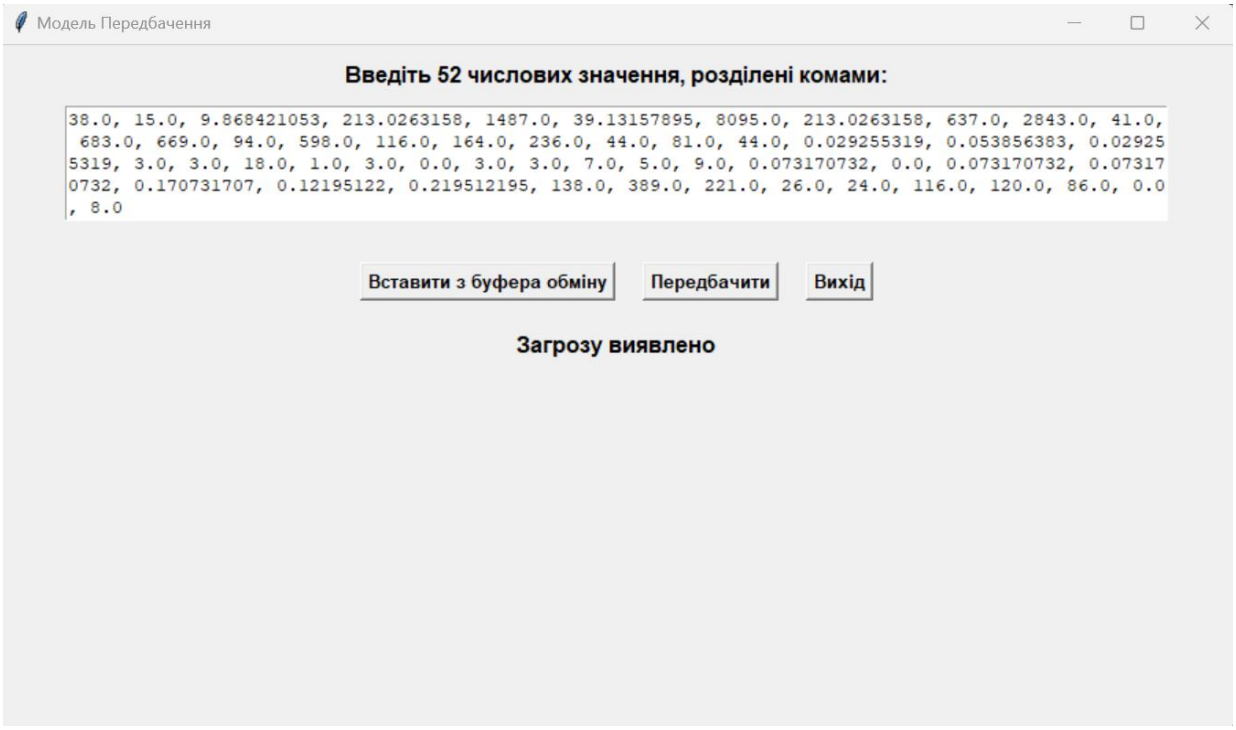


Рис.3.13. Виведення результату програмного модуля 1

Якщо програмний модуль не знайшов загрозу, виходячи з аналізу дампа пам'яті, буде повідомлення : "Загрозу не виявлено"

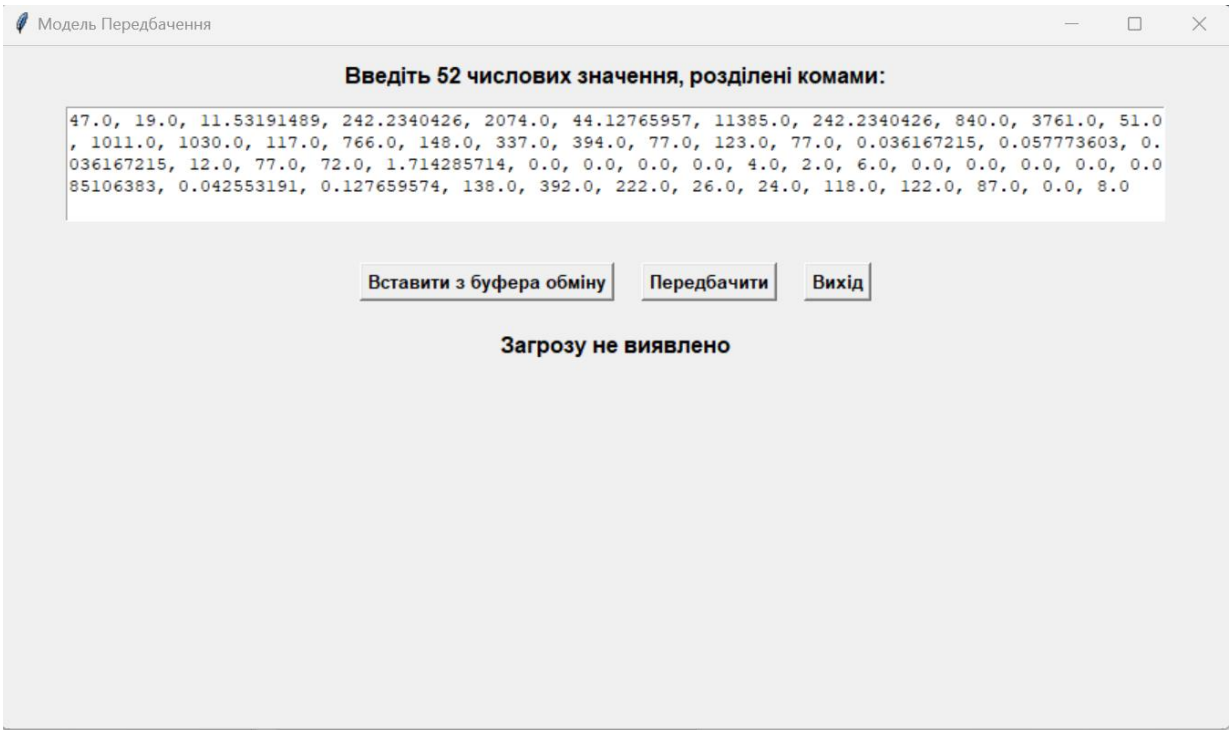


Рис.3.14. Виведення результату програмного модуля 2

3.5 Висновки до третього розділу

Під час цього дослідження було розроблено та реалізовано програмний модуль для аналізу шкідливого програмного забезпечення на основі даних із дампів пам'яті. Ключовою особливістю рішення є використання методів машинного навчання для класифікації програмного забезпечення як шкідливого або нешкідливого.

Застосування Машинного Навчання: За основу аналізу було обрано кілька моделей машинного навчання, включно з Логістичною Регресією, Випадковим Лісом, Градієнтним Бустінгом і Методом Опорних Векторів. Ці моделі були навчені та протестовані на даних з дампів пам'яті, надаючи значну точність у визначенні шкідливого ПЗ.

Робота з даними: Важливою частиною роботи стала попередня обробка даних. Було проведено аналіз і нормалізацію ознак, видалення несуттєвих змінних на основі теплової карти кореляцій і забезпечення збалансованості класів. Ці кроки виявилися критично важливими для підвищення якості та надійності прогнозів моделі.

Вибір і Оцінка Моделей: Після тестування різних моделей було встановлено, що модель Випадкового Лісу демонструє найкращу продуктивність з погляду всіх ключових метрик: Accuracy, Precision, Recall, F1-score і ROC AUC. Це підкреслює її придатність для реальних застосувань у сфері кібербезпеки.

Реалізація та Тестування: Розроблений модуль було протестовано не тільки з погляду точності моделей, а й щодо користувацького інтерфейсу та взаємодії. Було створено систему, яка дає змогу користувачеві вводити дані та отримувати результати аналізу в зручній формі.

Практична значущість: Результати дослідження показують, що машинне навчання може відігравати ключову роль у поліпшенні методів виявлення шкідливого ПЗ. Застосування таких систем у реальних умовах може значно підвищити ефективність кібербезпеки.

Розглгянемо основні інструменти для аналізу шкідливого програмного забезпечення, визначимо тип аналізу на якому вони працюють, основні функції, що лежать в їх основі, а також переваги та недоліки кожного інструменту та занесемо отримані результати дослідження в таблицю.

Таблиця 3.2.

Назва програми	Тип аналізу	Основні функції	Переваги	Недоліки та проблеми
VirusTotal	Статичний і динамічний	Агрегація даних антивірусних сканерів, веб-сканерів і різних інструментів аналізу для швидкого оцінювання файлів і URL.	Можливість швидкого сканування файлів і URL; використовує безліч антивірусних движків; простий інтерфейс і API.	Обмежений в аналізі захищених або обфусцированих шкідливих зразків; не завжди здатний надати глибокий аналіз. анализ.
Cuckoo Sandbox	Динамічний	Автоматизований аналіз файлів і URL в ізольованому середовищі.	Детальні звіти аналізу; модульна структура; відкритий вихідний код і активна спільнота.	Потребує значних ресурсів і експертизи для налаштування та управління; може обходитись деякими шкідливими програмами.

Продовження таблиці 3.2.

Назва програми	Тип аналізу	Основні функції	Переваги	Недоліки та проблеми
IDA Pro	Статичний	Дизасемблер і відладчик для аналізу двійкових файлів.	Потужний аналізатор; велика спільнота та безліч плагінів.	Висока вартість (для комерційної версії); складний для нових користувачів.
Wireshark	Статичний	Аналізатор мережевих пакетів.	Підтримка великої кількості протоколів; можливість глибокого аналізу трафіку; відкритий вихідний код.	Може бути складним для непідготовлених користувачів; обмежений у функціональності аналізу шкідливих програм.
Запропонований програмний модуль	Динамічний	Аналізатор дампів пам'яті	Дуже висока точність передбачень, безкоштовний, простий інтерфейс, відкритий вихідний код	Потрібно мануально створювати дамп пам'яті після чого передавати його в програмний модуль

Оцінка ефективності розробленого програмного модуля базується на кількох ключових показниках, які підкреслюють його придатність для практичного використання у сфері кібербезпеки:

- Точність Моделей (Accuracy): Високі значення точності для всіх протестованих моделей, особливо для моделі Випадкового Лісу, вказують на те, що система здатна з високою ймовірністю коректно класифікувати дані, які входять, як шкідливі або нешкідливі.
- Метрики Precision і Recall: Ці метрики важливі для оцінки того, наскільки добре модель уникає хибнопозитивних і хибнонегативних результатів. Випадковий Ліс показав чудові результати, що свідчать про надійність моделі в різних сценаріях використання.
- F1-Score: Поєднання Precision і Recall в одній метриці показує, що модель Випадкового Лісу забезпечує збалансоване співвідношення між чутливістю і специфічністю, що є ключовим для застосунків у сфері кібербезпеки.
- ROC AUC Score: Цей показник демонструє, наскільки добре модель здатна розрізняти класи за різних порогів класифікації. Високі значення ROC AUC для моделі Випадкового Лісу підтверджують її ефективність у різних умовах.
- Універсальність і Масштабованість: Модуль показав свою застосовність для різних типів даних і здатний масштабуватися для обробки великих обсягів інформації, що робить його корисним для організацій різного масштабу.
- Інтерфейс користувача: Простота й інтуїтивна зрозумілість користувацького інтерфейсу робить модуль доступним для фахівців різного рівня підготовки, що розширює коло потенційних користувачів.

Загалом, оцінка ефективності показує, що розроблений програмний модуль є надійним, точним і зручним інструментом для аналізу шкідливого

програмного забезпечення, що робить його цінним внеском у сферу кібербезпеки.

Розроблений програмний модуль аналізу шкідливого програмного забезпечення має значний вплив на сферу кібербезпеки:

- Підвищення Рівня Захисту: Завдяки високій точності та надійності, модуль здатний ефективно виявляти та класифікувати шкідливі програми, що сприяє підвищенню рівня безпеки інформаційних систем.
- Профілактика Кібератак: Здатність модуля швидко аналізувати та визначати потенційні загрози дає змогу запобігти можливим кібератакам ще до їхньої реалізації, тим самим зменшуючи ризик витоку або пошкодження даних.
- Зниження хибних спрацьовувань: Оптимізовані алгоритми та використання машинного навчання допомагають зменшити кількість хибнопозитивних і хибнонегативних спрацьовувань, що важливо для підтримання балансу між безпекою та зручністю використання системи.
- Адаптивність і здатність до навчання: Модуль, заснований на машинному навчанні, може адаптуватися до нових видів загроз, навчаючись на актуальних даних. Це дає змогу системі залишатися ефективною навіть за появи нових видів шкідливого ПЗ.
- Розширення Зони Застосування: Універсальність програмного модуля дає змогу його використовувати в різних галузях і організаціях, що підвищує загальний рівень кібербезпеки в масштабах галузі або країни.
- Оптимізація Ресурсів: Автоматизація процесу аналізу та класифікації шкідливого ПЗ допомагає знизити навантаження на фахівців з кібербезпеки та оптимізувати використання людських і обчислювальних ресурсів.

Узагальнена блоксхема реалізації менеджера паролей з використанням блокчейн технологій представлена в додатку А.

Таким чином, впровадження та використання розробленого програмного модуля здатне мати значний позитивний вплив на рівень кібербезпеки, забезпечуючи більш ефективне запобігання та реагування на кіберзагрози.

Для забезпечення того, щоб система залишалася актуальною та ефективною в умовах постійно мінливих кіберзагроз, необхідно постійно працювати над її поліпшенням. Ось кілька ключових напрямків для майбутніх удосконалень:

- Розширення Навчального Набору Даних: Постійне оновлення і розширення навчального датасету з новими зразками шкідливого ПЗ дасть змогу моделі краще адаптуватися до нових загроз. Це може включати в себе як автоматичний збір даних, так і участь у загальних програмах обміну інформацією про кіберзагрози.
- Використання Ансамблевих Методів: Комбінування декількох моделей машинного навчання в ансамблеві методи може підвищити точність і надійність виявлення шкідливого ПЗ.
- Автоматизація фідбеку і навчання: розробка механізмів для автоматичного збору зворотного зв'язку від користувачів та інтеграція цієї інформації в процес навчання моделі.
- Використання Глибокого Навчання: Дослідження і застосування методів глибокого навчання, як-от згорткові нейронні мережі (CNN) або рекурентні нейронні мережі (RNN), для аналізу складніших шаблонів і поведінки шкідливого ПЗ.
- Розширення Функціоналу Аналізу: Впровадження додаткових функцій, як-от аналіз поведінки, аналіз мережевого трафіку і прогнозування нових загроз, для підвищення загальної ефективності системи.
- Співпраця з Дослідницькими Організаціями: Активна участь у наукових дослідженнях і співпраця з університетами та дослідницькими центрами для обміну знаннями та використання останніх досягнень у сфері кібербезпеки.
- Розвиток модульності та гнучкості Системи: Поліпшення архітектури системи для забезпечення її масштабованості та модульності, що дасть змогу легко додавати нові функції та інтегруватися з іншими системами.

Ці заходи допоможуть системі залишатися в авангарді боротьби з кіберзагрозами, забезпечуючи високий рівень захисту й адаптуючись до мінливого ландшафту кібербезпеки.

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Воєнні конфлікти призводять до ряду серйозних екологічних проблем, які можуть мати тривалі наслідки як для навколишнього середовища, так і для здоров'я людей.

До основних проблем можна віднести наступні:

1. Забруднення води та ґрунту. Військові дії, такі як бомбардування та використання вибухівки, можуть призводити до забруднення водних ресурсів та ґрунту. Це може включати розливи нафти, витік хімічних речовин або радіоактивних матеріалів.

2. Знищення екосистем. Військові операції часто призводять до руйнування лісів, боліт, річок та інших природних середовищ. Це може мати негативний вплив на біорізноманіття та призвести до зникнення рідкісних видів рослин та тварин.

3. Забруднення повітря. Використання важкої військової техніки та вибухівки призводить до збільшення рівнів вуглекислого газу та інших шкідливих газів в атмосфері.

4. Проблеми з утилізацією відходів: Війна генерує велику кількість відходів, включаючи військові уламки, нерозірвані боєприпаси та інші відходи, що створюють додаткові проблеми для навколишнього середовища та здоров'я людини.

5. Порушення аграрної діяльності. Військові конфлікти можуть серйозно порушити сільськогосподарську діяльність, внаслідок чого землеробство стає менш ефективним або неможливим, що веде до ерозії ґрунтів і втрати родючих земель.

6. Проблеми з водопостачанням. Війна може спричинити перебої у водопостачанні та санітарії, що негативно впливає на здоров'я населення і збільшує ризик захворювань.

7. Кліматичні зміни. Військові конфлікти можуть сприяти змінам

клімату через викиди парникових газів, руйнування лісів та інші фактори.

Неможливо не згадати про те що в нашій країні зараз війна, з самого початку у 2014 році вона несе не лише економічні, демографічні, психологічні, соціальні та культурні збитки, а й значні екологічні. Екологічні збитки мають як негайні, так і довгострокові наслідки для здоров’я людей та економіки.

Війна в Україні, спричинила значні екологічні наслідки. Однією з основних проблем є забруднення повітря, води та ґрунту внаслідок військових дій, таких як обстріли та бомбардування. Це включає в себе руйнування газопроводів, вибухи на промислових об'єктах, а також інциденти з викидом хімічних речовин, таких як аміак. Наприклад, відбувалися випадки пошкодження газопроводів і цистерн з аміаком, що призвело до створення зон з високим рівнем забруднення.

Другою серйозною проблемою стало знищення лісів і природних середовищ, важливих для збереження біорізноманіття. Воєнні дії завдали шкоди рідкісним і ендемічним видам, а також змінили поведінку та міграційні шляхи птахів. Знищення лісів також впливає на клімат та може призвести до ерозійних процесів, особливо на півдні України.

Крім того, війна призвела до значного збільшення викидів вуглецю в атмосферу, які були спричинені лісовими пожежами, бомбардуваннями нафтопереробних заводів та промислових об’єктів. Загальний обсяг викидів вуглецю в атмосферу за сім місяців війни становив щонайменше 31 мільйон тонн. Також було зазначено, що потенційно 79 мільйонів тонн парникових газів будуть вироблені під час повоєнної відбудови та відновлення інфраструктури.

При горінні торфовищ виділяються токсичні речовини, такі як оксид та діоксид вуглецю, дрібні частки пилу з розміром у 2,5 мікрони, типові для горіння, а також летючі органічні сполуки, включаючи акролеїн і формальдегід. Ці речовини можуть мати шкідливий вплив на здоров'я людини та навколишнє середовище.

Війна також мала вплив на природоохоронні території. Близько 20% природоохоронних територій України опинилися в зоні бойових дій, у тому числі

вісім природних заповідників і десять національних парків, які зазнали значних пошкоджень.

Загалом, війна в Україні призвела до серйозних екологічних викликів, які вимагатимуть значних зусиль для відновлення та реабілітації навколишнього середовища у майбутньому.

На відновлення екосистеми в Україні підуть десятиліття:

Заступник міністра захисту довкілля України Євгеній Федоренко повідомив про зафіксовані понад 2 тисячі 200 випадків екологічних злочинів, спричинених війною. Це включає забруднення атмосферного повітря, ґрунтів, збитки лісовим ресурсам, а також забруднення Чорного та Азовського морів. Федоренко зазначає, що збитки вплинули на екосистеми загалом і біорізноманіття, і відновлення довкілля може зайняти десятиліття.

Зміна міграційних шляхів птахів через військові дії не піддається вартісній оцінці. Птахи, які тисячі років слідували певним маршрутам і оселялися на цих територіях, через вибухи та загибель тварин змушені змінювати свої звичні місця проживання.

Вздовж узбережжя Чорного моря та вздовж кордонів країн, що межують з Чорним морем, було виявлено близько 1000 загиблих тварин, і очікується, що їхня кількість може зрости.

Євгеній Федоренко, також підкреслив серйозність екологічних викликів, які виникли внаслідок війни. Він зазначив, що багато земельних ділянок забруднено хімічними речовинами через бойові дії. Відновлення лісів і ґрунтів буде складним і тривалим процесом, що може зайняти десятиліття. Також війна загострила проблеми з доступом до питної води через пошкодження водних об'єктів. У міністерстві розробляється план відновлення довкілля після війни.

Окрім цього Міжнародні організації, зокрема Програма ООН з навколишнього середовища, допомагають в очищенні забруднених територій і відновленні екосистем. Український уряд також вживає заходів, включаючи прийняття нових законів та збільшення фінансування екологічних ініціатив.

Важливо, щоб міжнародна спільнота продовжувала підтримувати ці зусилля задля забезпечення сталого майбутнього для країни та її народу.

ВИСНОВКИ

У даній кваліфікаційній роботі були розглянуті ключові аспекти інформаційної безпеки, з акцентом на аналіз і класифікацію шкідливого програмного забезпечення. Особливу увагу було приділено використанню методів машинного навчання для виявлення та запобігання кібератак. Розроблене програмне рішення демонструє можливості сучасного підходу до аналізу шкідливого ПЗ з використанням алгоритмів машинного навчання.

Огляд наявних підходів: Проаналізовано різні методи та інструменти, що використовуються у сфері кібербезпеки, і виявлено їхні ключові недоліки.

Розробка і Тестування Рішення: Розроблено систему, засновану на машинному навчанні, яка успішно демонструє високу точність і ефективність у виявленні шкідливого ПЗ. Процес розробки включав вибір відповідних моделей, попередню обробку даних і ретельне тестування.

Аналіз Ефективності: Продемонстровано, що система має високу точність і здатна ефективно адаптуватися до нових загроз. Оцінка ефективності впровадження показала значне поліпшення у виявленні шкідливого ПЗ порівняно з традиційними методами.

Внесок у Кібербезпеку: Розроблене рішення здатне значно поліпшити поточні методи виявлення шкідливого ПЗ, пропонуючи більш гнучкий і адаптивний підхід до захисту інформаційних систем.

Перспективи Розвитку: Запропоновано шляхи подальшого вдосконалення системи, включно з інтеграцією з наявними системами кібербезпеки та використанням новітніх досягнень у галузі машинного навчання та штучного інтелекту.

Насамкінець, цей проєкт демонструє значний потенціал застосування машинного навчання в боротьбі з кіберзагрозами і може слугувати основою для розроблення просунутіших систем кібербезпеки в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. .Офіційний сайт: «Bloomberg, Coronavirus Could Bankrupt Most Airlines by End of May, Consultant Warns by Anurag Kotoky from 16.03.2020» [Електронний ресурс]. – Режим доступу: https://www.bloomberg.com/news/articles/2020-03- 16/virus-to-bankrupt-most-airlines-by-end-of-may-consultant-says?srnd=premiumeurope

2. Казмірчук С.В. Дослідження методик оцінки ризиків / С.В. Казмірчук, В.В. Волянська // Сучасні проблеми захисту інформації з обмеженим доступом: міжвідомча науково-практ. конф., тези доп. – К., 2008. − С.67-69.

3. Про Основні засади розвитку інформаційного суспільства [Електронний ресурс].— URL: http://zakon4.rada.gov.ua/laws/show/537-16

4. Брюс Шнаейр, Прикладна криптографія. Протоколи, алгоритми, вихідні тексти на мові Сі. Москва, 2002. 610 с.

5. Бабак В.П. Інформаційна безпека та сучасні мережеві технології : Англо-українсько-російський словник термінів / В.П. Бабак, О.Г. Корченко. – Київ : Издательство НАУ, 2003. – 670 с

6. Vigna G., Kemmerer R. A. NetSTAT: A Network-based Intrusion Detection Approach // Proceedings of the 14th Annual Computer Security Application Conference. – 2000. – P. 73-81.

7. K. Bhargavan and A. Delignat-Lavaud. Web-based attacks on host-proof encrypted storage. In Proc. of WOOT, 2012.

8. D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song. Towards a formal foundation of web security. In Proceedings of the 23rd IEEE Computer Security Foundations Symposium, 2010.

9. Нові стандарти для безпарольної аутентифікації [Електронний ресурс]: - Режим доступу до ресурсу: https://habr.com/1cloud/blog/353966/.

10. Golovko V. Neural Networks approaches for Intrusion Detection and Recognition // Computing. – 2006. – Vol. 5. № 3. – P. 118-125.

11. Sh. Ataei et al. ‘Sensor fusion of a railway bridge load test using neural networks’. In: Expert Syst. Appl. 29 (3 2005), pp. 678–683.

12. Гребенніков В.В. - Комплексні системи захисту інформації: проектування,впровадження,супровід. Збірник лекцій. 2013. 161 с. - Режимдоступу:https://mybook.ru/author/vadimgrebennikov-3/kompleksni-sistemi-zahistuinformaciyi-proektuvann/

13. Критерії оцінки захищеності інформації в комп'ютернихсистемахвіднесанкціонованого доступу НД ТЗІ 2.5-004-99. Режим доступу: https://tzi.ua/assets/files/НД-ТЗІ-2.5-004-99.pdf?fbclid=IwAR16Qka92G63wtjvFfHcK5rALmf2z0iKjdO0rN6f005_fxovlJX3-RtIpqk

14. Офіційний сайт: «IT-enterprise. PdM (predictive maintenance, предиктивне обслуговування)» [Електронний ресурс]. – Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/pdm

15. Комп'ютерні системи штучного інтелекту. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 «Комп'ютерна інженерія», 122 «Комп'ютерні науки та інформаційні технології» / Укл.: Є.В. Мелешко – Кіровоград: КНТУ, 2016. – С.8-13.

16. Cyber Security requires strong UX [Електронний ресурс]: - Режим доступу до ресурсу: https://hackernoon.com/cyber-security-requires-an-important-ingredient-s trong-ux-d0727a0c076

17. Інтелектуальний агент // Вікіпедія – Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Інтелектуальний_агент (дата звернення: 06.05.2020).

18. Класифікація автоматизованих систем і стандартні функціональні профілізахищеності оброблюваної інформації від несанкціонованого доступу. Режимдоступу: https://tzi.ua/assets/files/НД-ТЗІ-2.5-005--99.pdf

19. Mike Gerdes and Dieter Scholz. 'Fuzzy Condition Monitoring of Recirculation Fans and Filters'. In: CEAS Aeronautical Journal 2 (1–4 2011), pp. 81–87.

20. І.В. Калініна, О.І. Лісовиченко Використання генетичних алгоритмів в задачах оптимізації / Міжвідомчий науково-технічний збірник. 2015. – № 1(26).

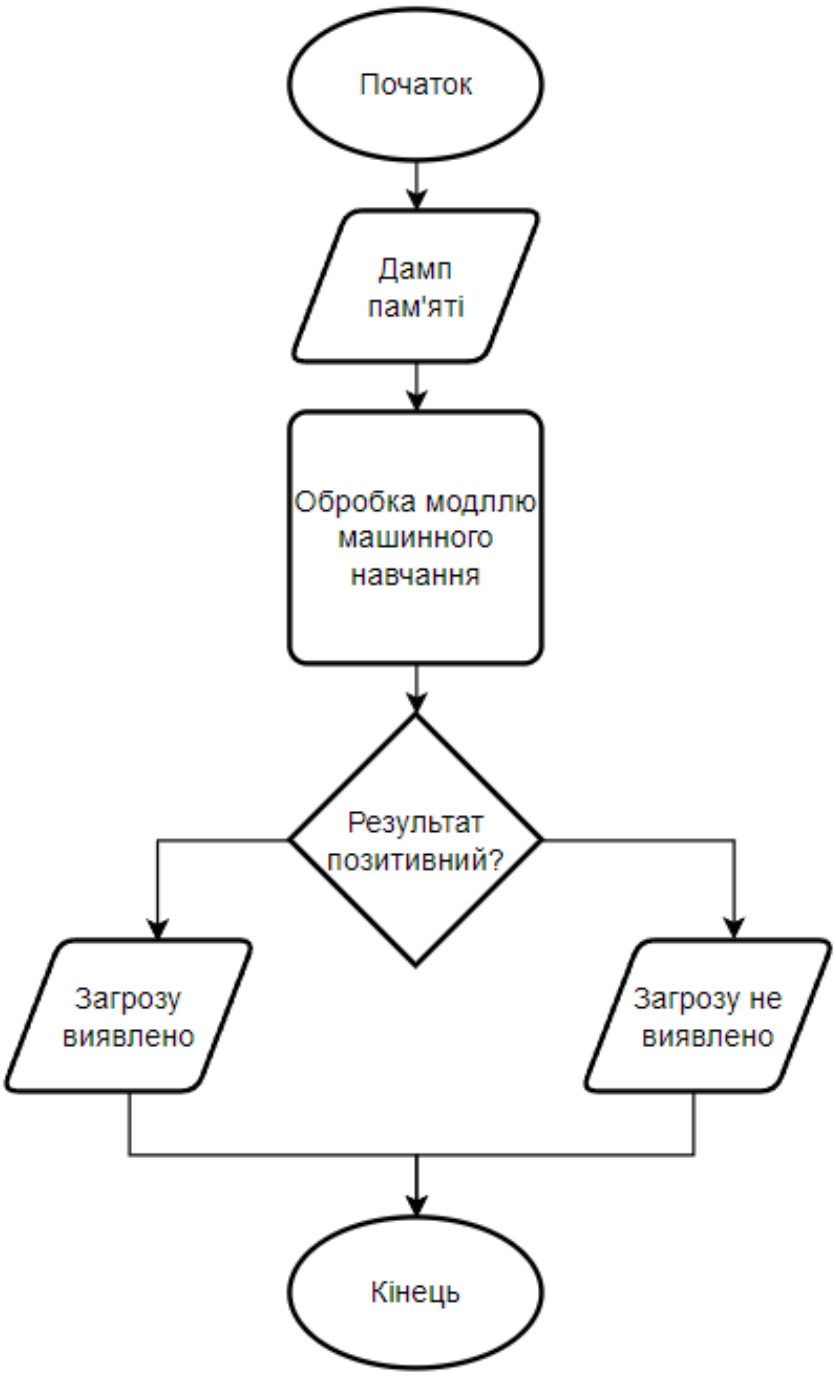
21. Як діє штучний інтелект і перспективи його розвитку[Електронний ресурс] – Режим доступу до ресурсу: https://aiconference.com.ua/uk/news/printsipi-raboti-iskusstvennosgo-intellekta-i-perspektiva-egoispolzovaniya-92238

22. Mike Gerdes and Dieter Scholz. ‘Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters’. In: Deutscher Luft- und Raumfahrtkongress 2009: Tagungsband - Manuskripte (DLRK, Aachen, 08.-10. September 2009). 2009.

23. Кононюк А. Ю. Нейроні мережі і генетичні алгоритми. Корнійчук, 2008. 446 с.

24. Zaenen A., van den Bosch A. Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics. Association for Computational Linguistics. 2007. P. 496–503.

25. Коцовський В. М. Методи та системи штучного інтелекту. Ужгород, 2016. 76 с.


Початок
Дамп пам'яті
Обробка модллю машинного навчання
Результат позитивний?
Загрозу виявлено
Загрозу не виявлено
Кінець

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Dropout
import warnings
warnings.filterwarnings('ignore')
from joblib import dump
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import roc_curve, auc
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
f1_score, roc_auc_score, classification_report
df=pd.read_csv('MalwareMemoryDump.csv')
pd.set_option('display.float_format', '{:.2f}'.format)
df.describe().T.to_excel("output.xlsx")
numeric_data = df.select_dtypes(include=[np.number])
sns.set(rc={'figure.figsize':(30,20)})
plt.figure(figsize=(15, 10))
sns.heatmap(numeric_data.corr(), annot=False, cmap='coolwarm')
plt.title('Heatmap of Feature Correlations')
plt.show()
df = df.drop([
        'pslist_nprocs64bit',
        'svcscan_interactive_process_services',
        'handles_nport',
        'Raw_Type',
        'SubType'
        ],axis=1)
df.head(5)
df.value_counts('Label')
X = df.drop('Label', axis=1)
y = df['Label']


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2,
random_state=42)
log_reg = LogisticRegression()

log_reg.fit(X_train, y_train)

log_reg_predictions = log_reg.predict(X_test)

log_reg_accuracy = accuracy_score(y_test, log_reg_predictions)
print("Точность Логистической Регрессии:", log_reg_accuracy)
random_forest = RandomForestClassifier()

random_forest.fit(X_train, y_train)

random_forest_predictions = random_forest.predict(X_test)

random_forest_accuracy = accuracy_score(y_test, random_forest_predictions)
print("Точность Случайного Леса:", random_forest_accuracy)

gradient_boosting = GradientBoostingClassifier()
gradient_boosting.fit(X_train, y_train)

gradient_boosting_predictions = gradient_boosting.predict(X_test)

gradient_boosting_accuracy = accuracy_score(y_test, gradient_boosting_predictions)
print("Точность Градиентного Бустинга:", gradient_boosting_accuracy)

svm_model = SVC(probability=True)

svm_model.fit(X_train, y_train)

svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("Точность SVM:", svm_accuracy)

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    matrix = confusion_matrix(y_test, predictions)
    precision = precision_score(y_test, predictions, average='binary')
    recall = recall_score(y_test, predictions, average='binary')
    f1 = f1_score(y_test, predictions, average='binary')

    # Использование вероятностей для положительного класса (обычно второй столбец) для
ROC AUC
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

    report = classification_report(y_test, predictions)
```

```
    print(f"Модель: {model.__class__.__name__}")
    print("Accuracy:", accuracy)
    print("Матриця помилок:\n", matrix)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)
    print("ROC AUC Score:", roc_auc)
    print("Звіт про класифікацію::\n", report)

def plot_roc_curve(model, X_test, y_test):
    # Получение вероятностей классов
    y_probs = model.predict_proba(X_test)

    # Используем вероятности второго класса (положительный класс) для бинарной
классификации
    y_probs = y_probs[:, 1]

    fpr, tpr, thresholds = roc_curve(y_test, y_probs)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()
evaluate_model(log_reg, X_test, y_test)
evaluate_model(random_forest, X_test, y_test)
evaluate_model(gradient_boosting, X_test, y_test)
evaluate_model(svm_model, X_test, y_test)
dump(scaler, 'scaler.joblib')
dump(random_forest, 'random_forest_model.joblib')
# Функция для запроса ввода пользователя и преобразования его в массив numpy
def get_user_input():
    # Предположим, что у вас 55 числовых признаков (исключая категориальные)
    input_str = input("Введіть значення ознак, розділені комами (52 числових значень): ")
    feature_values = input_str.split(',')

    # Преобразование строк в числа
    try:
        features = [float(value) for value in feature_values]
    except ValueError:
        print("Помилка введення: переконайтеся, що ви ввели 52 числових значень, розділених
комами.")
```

```
        return None

    # Проверка количества признаков
    if len(features) != 52:
        print("Помилка: необхідно ввести рівно 52 значень ознак.")
        return None
    else:
        print('Все добре')

    return np.array([features])

# Запрос ввода пользователя
X_new = get_user_input()
if X_new is not None:
    # Нормализация новых данных
    X_new_scaled = scaler.transform(X_new)

    # Предсказание с помощью модели
    predictions = model.predict(X_new_scaled)

    if predictions[0] == 0:
        print("Результати передбачення: Загрозу не виявлено")
    else:
        print("Результати передбачення: Загрозу виявлено")
from joblib import load

import warnings
warnings.filterwarnings('ignore')
scaler = load('scaler.joblib')
model = load('random_forest_model.joblib')
import tkinter as tk
from tkinter import font
import numpy as np
from joblib import load

scaler = load('scaler.joblib')
model = load('random_forest_model.joblib')

def predict():
    input_str = text.get("1.0", "end-1c")
    feature_values = input_str.split(',')

    try:
        features = [float(value) for value in feature_values]
        if len(features) == 52:
            X_new = np.array([features])
            X_new_scaled = scaler.transform(X_new)
            prediction = model.predict(X_new_scaled)
```

```
            result = "Загрозу не виявлено" if prediction[0] == 0 else "Загрозу виявлено"
            output_label.config(text=result)
        else:
            output_label.config(text="Помилка: введіть рівно 52 значення.")
    except ValueError:
        output_label.config(text="Помилка введення: введіть числові значення.")

def paste_from_clipboard():
    try:
        text.delete("1.0", tk.END)
        text.insert("1.0", root.clipboard_get())
    except:
        output_label.config(text="Помилка: не вдалося вставити текст з буфера обміну.")

# Створення головного вікна
root = tk.Tk()
root.title("Модель Передбачення")
root.geometry('900x500')  # Увеличение размера окна

label_font = font.Font(size=12, weight='bold')
button_font = font.Font(size=10, weight='bold')

# Створення фреймів
top_frame = tk.Frame(root)
top_frame.pack(pady=10)
middle_frame = tk.Frame(root)
middle_frame.pack(pady=10)
bottom_frame = tk.Frame(root)
bottom_frame.pack(pady=10)

# Створення віджетів
label = tk.Label(top_frame, text="Введіть 52 числових значення, розділені комами:",
font=label_font)
label.pack()

text = tk.Text(top_frame, height=5, width=100)
text.pack(pady=10)

paste_button = tk.Button(middle_frame, text="Вставити з буфера обміну",
command=paste_from_clipboard, font=button_font)
paste_button.pack(side=tk.LEFT, padx=10)

predict_button = tk.Button(middle_frame, text="Передбачити", command=predict,
font=button_font)
predict_button.pack(side=tk.LEFT, padx=10)

exit_button = tk.Button(middle_frame, text="Вихід", command=root.destroy, font=button_font)
exit_button.pack(side=tk.LEFT, padx=10)
```

```
output_label = tk.Label(bottom_frame, text="", font=label_font)
output_label.pack()
root.mainloop()
```

	count	mean	std	min	25%	50%	75%	max
pslist_nproc	58596	41,39477	5,777249	21	40	41	43	240
pslist_nppid	58596	14,71384	2,656748	8	12	15	16	72
pslist_avg_threads	58596	11,34166	1,588231	1,65	9,972973	11	12,86196	16,81818
pslist_nprocs64bit	58596	0	0	0	0	0	0	0
pslist_avg_handlers	58596	247,5098	111,8578	34,9625	208,725	243,9637	289,9743	24845,95
dlllist_ndlls	58596	1810,805	329,7826	670	1556	1735	2087	3443
dlllist_avg_dlls_per_proc	58596	43,70781	5,742023	7,333333	38,83333	42,78152	49,60528	53,17073
handles_nhandles	58596	10258,58	4866,864	3514	8393	9287,5	12193	1047310
handles_avg_handles_per_proc	58596	249,561	145,9999	71,13924	209,6482	247,209	291,3551	33784,19
handles_nport	58596	0	0	0	0	0	0	0
handles_nfile	58596	899,1195	3432,351	266	646	839	1080	807008
handles_nevent	58596	3572,41	805,4605	966	2923	3151	4321	7892
handles_ndesktop	58596	44,52917	5,161254	22	43	45	46	159
handles_nkey	58596	774,2807	150,4071	284	675	753	859	2668
handles_nthread	58596	928,5101	237,8176	388	708	848	1169	5637
handles_ndirectory	58596	102,3983	9,782695	57	99	103	107	498
handles_nsemaphore	58596	683,3393	94,53108	296	614	684	750	4268
handles_ntimer	58596	130,3279	14,96527	69	120	131	142	382
handles_nsection	58596	290,1275	144,2788	50	177	224	415	14687
handles_nmutant	58596	312,5888	73,17319	118	258	289	366	583
ldrmodules_not_in_load	58596	60,83035	18,76142	6	46	57	74	240
ldrmodules_not_in_init	58596	99,94641	21,43848	16	85	97	115	264
ldrmodules_not_in_mem	58596	60,8326	18,75995	6	46	57	74	240
ldrmodules_not_in_load_avg	58596	0,03317	0,009263	0,016176	0,028846	0,031361	0,03643	0,53112
ldrmodules_not_in_init_avg	58596	0,055223	0,010112	0,040526	0,052397	0,054036	0,05601	0,585062
ldrmodules_not_in_mem_avg	58596	0,033171	0,009266	0,016176	0,028846	0,031361	0,03643	0,53112
malfind_ninjections	58596	7,010274	15,39065	1	3	4	5	627
malfind_commitCharge	58596	969,1992	6041,621	1	3	4	6	220850
malfind_protection	58596	42,28241	92,33706	6	18	24	30	3762
malfind_uniqueInjections	58596	1,733699	2,741343	1	1	1,25	1,333333	90,66667
psxview_not_in_pslist	58596	1,875845	2,995955	0	0	1	3	43
psxview_not_in_eprocess_pool	58596	0,002082	0,045582	0	0	0	0	1
psxview_not_in_ethread_pool	58596	2,2735	4,621418	0	0	1	3	201
psxview_not_in_pspcid_list	58596	1,879463	3,017659	0	0	1	3	43
psxview_not_in_csrss_handles	58596	6,276572	4,622047	4	4	5	7	205
psxview_not_in_session	58596	3,875589	2,99597	2	2	3	5	45
psxview_not_in_deskthrd	58596	8,256605	4,736886	4	6	7	9	207
psxview_not_in_pslist_false_avg	58596	0,040991	0,057563	0	0	0,021739	0,068182	0,551282
psxview_not_in_eprocess_pool_false_avg	58596	7,95E-05	0,001269	0	0	0	0	0,043478
psxview_not_in_ethread_pool_false_avg	58596	0,047745	0,066907	0	0	0,023256	0,069767	0,8375
psxview_not_in_pspcid_list_false_avg	58596	0,041078	0,058216	0	0	0,021739	0,068182	1
psxview_not_in_csrss_handles_false_avg	58596	0,141784	0,061438	0,025806	0,1	0,119048	0,16618	0,854167
psxview_not_in_session_false_avg	58596	0,087964	0,05504	0,008333	0,04878	0,066667	0,113636	0,576923
psxview_not_in_deskthrd_false_avg	58596	0,187701	0,061534	0,04375	0,146341	0,166465	0,211691	0,8625
modules_nmodules	58596	137,9615	0,198251	126	138	138	138	138

svcscan_nservices	58596	391,3475	4,529704	94	389	389	395	395
svcscan_kernel_drivers	58596	221,4066	1,991087	55	221	221	222	222
svcscan_fs_drivers	58596	25,99625	0,17079	6	26	26	26	26
svcscan_process_services	58596	25,06342	1,529628	7	24	24	27	27
svcscan_shared_process_services	58596	116,8795	1,550401	26	116	116	118	118
svcscan_interactive_process_services	58596	0	0	0	0	0	0	0
svcscan_nactive	58596	121,9955	2,822858	30	121	122	123	129
callbacks_ncallbacks	58596	86,90566	3,134117	50	87	87	88	89
callbacks_nanonymous	58596	0,000853	0,029199	0	0	0	0	1
callbacks_ngeneric	58596	7,999881	0,010929	7	8	8	8	8