ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Дослідження та розробка методів виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

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

_____ *Єгор КАЗНАДЄЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КІД-41
Єгор КАЗНАДЄЙ
Ім'я, ПРІЗВИЩЕ
Керівник: Юлія БОРОВА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Казнадєй Єгор Вадимович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Дослідження та розробка методів виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання
керівник кваліфікаційної роботи Юлія БОРОВА,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «_____» _____2024р. № _____
2. Строк подання кваліфікаційної роботи «_____» _____ 2024р.
3. Вихідні дані до кваліфікаційної роботи: _____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. _____
2. _____
3. _____
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «_____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	26.02-22.03.2024	Виконано
2	Обробка матеріалу	23.03-05.04.2024	Виконано
3	Розробка теоретичної частини	06.04-10.04.2024	Виконано
4		11.04-30.04.2024	Виконано
5		01.05-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-11.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	25.05-1.06.2024	Виконано

Здобувач(ка) вищої освіти _____ Єгор КАЗНАДЄЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Юлія БОРОВА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 стор., 9 рис., 1 табл., 26 джерел.

Мета роботи – дослідження та розробка методів виявлення та захисту від шкідливого програмного забезпечення.

Об'єкт дослідження – технології машинного навчання.

Предмет дослідження – методи виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання.

Короткий зміст роботи: Досліджено та розроблено методи виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання. Використані різні методи аналізу, такі як аналіз на основі сигнатур, поведінковий аналіз та евристичні методи, які можуть бути використані окремо або в поєднанні для ефективного виявлення та захисту. Застосовано методи машинного навчання для класифікації та виявлення шкідливого програмного забезпечення, де показано переваги і обмеження існуючих статичних та динамічних методів. Розроблено систему з використанням кватерніонних нейронних мереж, що має високу точність виявлення та класифікації шкідливого програмного забезпечення.

КЛЮЧОВІ СЛОВА: ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАШИННЕ НАВЧАННЯ, АЛГОРИТМ, KERAS, TENSORFLOW.

ЗМІСТ

ВСТУП..... 8

РОЗДІЛ 1. АНАЛІЗ СУТНОСТІ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 12

1.1 Поняття та види шкідливого програмного забезпечення 12

1.2 Методи аналізу шкідливого програмного забезпечення 19

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ 28

2.1 Автоматична класифікація та виявлення шкідливих програм за допомогою машинного навчання та нейронних мереж..... 28

2.2 Основи машинного навчання та нейронних мереж..... 30

2.3 Попередня робота з застосування машинного навчання у класифікації шкідливого програмного забезпечення 36

РОЗДІЛ 3. МЕТОД НАВЧАННЯ МОДЕЛІ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 42

3.1 Аналіз методів навчання моделі виявлення та захисту від шкідливого програмного забезпечення 42

3.2 Реалізація алгоритмів виявлення шкідливого програмного забезпечення за допомогою бібліотек TensorFlow та Keras 44

ВИСНОВКИ 52

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... **Ошибка! Закладка не определена.**

ВСТУП

Актуальність теми. З розвитком інтернет-технологій і збільшенням кількості різноманітних смарт-пристроїв зростає і перелік потенційних небезпек для їх користувачів. Різні комп'ютерні віруси та шпигунські програми можуть завдати значної шкоди, викликати збої у роботі комп'ютера, спричинити втрату конфіденційної інформації та інші негативні наслідки. Крім того, існують програми, які здатні виконувати шкідливі дії без згоди користувачів, наприклад, підписувати їх на небажані розсилки, здійснювати покупки в інтернет-магазинах, показувати небажану рекламу. Зловмисники постійно вдосконалюють свої методи атак і розробляють нові види вірусів та шкідливих програм, що робить традиційні методи захисту застарілими та неефективними. Отже, актуальність теми дослідження та розробки методів виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання є надзвичайно високою. У сучасному світі, де інтернет-технології розвиваються стрімкими темпами, а кількість підключених до мережі пристроїв постійно зростає, кіберзагрози стають все більш серйозними та різноманітними. Шкідливе програмне забезпечення, таке як віруси, трояни, програми-шпигуни, здатне завдати значної шкоди як приватним користувачам, так і компаніям, спричиняючи втрату конфіденційної інформації, фінансові збитки та порушення роботи систем.

Традиційні методи захисту, такі як антивірусні програми, стають менш ефективними у боротьбі з новими видами шкідливого програмного забезпечення, оскільки зловмисники постійно вдосконалюють свої методи атак. У цьому контексті, використання машинного навчання для виявлення та захисту від шкідливого програмного забезпечення відкриває нові перспективи. Машинне навчання дозволяє створювати системи, які здатні самостійно вчитися на основі великого обсягу даних, виявляти нові загрози та адаптуватися до них у режимі реального часу.

Методи машинного навчання можуть аналізувати поведінкові дані програм,

виявляти аномалії та передбачати потенційно шкідливі дії до того, як вони завдадуть шкоди. Це значно підвищує рівень безпеки та дозволяє швидше реагувати на нові види атак. Таким чином, дослідження та розробка таких методів є критично важливими для забезпечення кібербезпеки в сучасному цифровому світі, що робить цю тему надзвичайно актуальною.

Вітчизняні та зарубіжні дослідники, такі як Anderson B., BooJoong Kang, Islam R., Kolter J., Liu L., Makandar A., Muhammet Sahin, Rafiqul I., Schultz M., Sung A., Xu J., Zhou D., І.А. Білан, О.О. Волков, Д.С. Волошко, М.С. Діденко, І. Жульковська, С.М. Лисенко, О.Ю. Новотарський, В. Педяш, А. Плужник, Д.В. Прочухан, С.Г. Семенов, І.А. Терейковський, Р.В. Щука та ін. досліджували різні методи виявлення шкідливого програмного забезпечення. Але тема дослідження та розробки методів виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання залишається вельми актуальною внаслідок швидкої еволюції шкідливого програмного забезпечення, можливості автоматизації процесів виявлення та захисту за допомогою алгоритмів машинного навчання та зростання кількості підключених до Інтернету пристроїв, що створює нові можливості для атак.

Мета і завдання дослідження. Метою кваліфікаційної роботи є дослідження та розробка методів виявлення та захисту від шкідливого програмного забезпечення. Для досягнення поставленої мети необхідно вирішити наступні завдання:

- навести поняття шкідливого програмного забезпечення;
- розглянути види шкідливого програмного забезпечення;
- проаналізувати методи аналізу шкідливого програмного забезпечення;
- розглянути автоматичну класифікацію та виявлення шкідливих програм за допомогою машинного навчання та нейронних мереж;
- охарактеризувати основи машинного навчання та нейронних мереж ;
- аргументувати попередню роботу з застосуванням машинного навчання у класифікації шкідливого програмного забезпечення;

- описати метод навчання моделі виявлення та захисту від шкідливого програмного забезпечення (ініціалізація ваг, градієнтний спуск, функції втрат та активації);
- навести реалізацію алгоритмів виявлення шкідливого програмного забезпечення за допомогою бібліотек TensorFlow та Keras.

Об’єкт дослідження – технології машинного навчання.

Предмет дослідження – методи виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання.

Методи дослідження. При написанні цієї роботи використовувалися різноманітні методи наукового дослідження для досягнення поставлених цілей. Було застосовано методи аналізу та синтезу для детального вивчення характеристик шкідливого програмного забезпечення та методів машинного навчання. Використано абстрактно-логічні методи, такі як дедукція, індукція, аналогія, перехід від конкретного до абстрактного та навпаки, що дозволило побудувати чіткі дефініції понять та класифікації.

Для розробки моделі виявлення та захисту від шкідливого програмного забезпечення застосовано структурно-функціональний підхід, який сприяв аналізу компонентів системи та їх взаємодії. Також використовувався порівняльний метод для оцінки ефективності різних алгоритмів машинного навчання та нейронних мереж. В роботі застосовувалися загальнонаукові методи, такі як аналіз літературних джерел, систематизація інформації, а також спеціальні методи, зокрема, реалізація та тестування алгоритмів на основі бібліотек TensorFlow та Keras. Ці методи дозволили глибше зрозуміти сутність проблеми, розробити ефективні підходи до її вирішення та обґрунтувати доцільність використання машинного навчання для виявлення та захисту від шкідливого програмного забезпечення.

Теоретична значущість отриманих результатів полягає в узагальненні та систематизації знань про шкідливе програмне забезпечення та застосування методів машинного навчання для його виявлення та захисту від нього. Розглянуті в роботі теоретичні засади захисту інформації та розвиток технологій машинного

навчання дозволяють глибше зрозуміти принципи функціонування цих технологій та їхній потенційний вплив на кібербезпеку. Описані ключові характеристики, алгоритми та методи класифікації шкідливого програмного забезпечення допомагають уточнити розуміння процесів, що відбуваються в системах кібербезпеки, та виявити можливості й обмеження використання машинного навчання для захисту інформації. Це сприяє розвитку теоретичних уявлень про ефективні способи боротьби зі шкідливими програмами та вдосконаленню методів їх виявлення та нейтралізації.

Методична значущість полягає в розробці методів застосування машинного навчання для виявлення та захисту від шкідливого програмного забезпечення у різних сферах, таких як інформаційна безпека, фінансовий сектор та охорона здоров'я. Аналіз пропозицій щодо застосування машинного навчання дозволяє виявити переваги та можливості використання цієї технології для підвищення рівня кібербезпеки в різних галузях діяльності.

Практична значущість результатів полягає в можливості їх використання для розробки та впровадження систем захисту від шкідливого програмного забезпечення на основі машинного навчання. Розглянуті у роботі практичні приклади застосування машинного навчання дозволяють ідентифікувати конкретні сценарії використання технології для забезпечення кібербезпеки і розробляти стратегії їх впровадження. Це сприяє створенню більш ефективних систем захисту, що можуть бути інтегровані у реальні умови роботи різних організацій та підприємств.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. АНАЛІЗ СУТНОСТІ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття та види шкідливого програмного забезпечення

Шкідливе програмне забезпечення (англ. Malware, утворене поєднанням слів «malicious» («зловмисний») і «software» («програмне забезпечення»)) – це програми, призначені для порушення нормального функціонування комп'ютера, мережі або інших пристроїв, а також для компрометації конфіденційної інформації, що зберігається в системі [21, с. 389]. Воно може бути у вигляді файлу, коду, скрипту, байтової послідовності тощо і зазвичай поширюється через Інтернет.

Комп'ютерний вірус – це тип шкідливого програмного забезпечення, який може приєднуватися до інших програм, розмножуватися та поширюватися без згоди користувача з одного комп'ютера на інший. Після зараження комп'ютера вірус створює свої копії, приєднується до інших файлів або документів, змінюючи їх і продовжуючи своє поширення [23, с. 104].

Віруси непомітно заражають пристрої, зазвичай призначені для знищення особистих файлів або отримання контролю над пристроями. Створюючи власні копії, комп'ютерні віруси поширюються між пристроями та мережами подібно до біологічних вірусів, що передаються від однієї людини до іншої. Як і біологічні віруси, деякі комп'ютерні віруси просто дратують користувачів і не становлять реальної загрози, тоді як інші можуть завдати значної шкоди як особистим пристроям, так і великим компаніям, спричиняючи великі збитки [26].

Часто люди ототожнюють поняття «шкідливе програмне забезпечення» і «комп'ютерний вірус», але технічно вони відрізняються [26]. Вірус є одним із типів шкідливого ПЗ, ключовою особливістю якого є здатність приєднуватися до файлів і самовідтворюватися. Він заражає файли та поширюється через пристрій під час кожного запуску зараженого файлу чи програми.

Як зазначає І.А. Білан до загальновідомого шкідливого програмного

забезпечення (malware) експерти відносять будь-які програми, які несанкціоновано проникають у комп'ютерну техніку або периферійні пристрої. Шкідливе програмне забезпечення створене з метою завдання шкоди або виведення з ладу певного пристрою або мережі. Кіберзлочинці зазвичай використовують його для доступу до даних, які можуть бути використані для отримання фінансової або іншої вигоди [14, с. 141]. Основною метою поширення шкідливих програм є викрадення інформації, включаючи персональні дані користувачів, зараження комп'ютерів вірусами, примусове взяття контролю над комп'ютерами для запуску DDoS-атак проти інших мереж тощо. За понад 30 років існування шкідливих програм було розроблено багато способів атак, серед яких вкладення в електронну пошту, шкідлива реклама на популярних сайтах, встановлення підробленого програмного забезпечення, інфіковані USB-накопичувачі та додатки, фішингові електронні листи і навіть SMS-повідомлення.

Шкідливе програмне забезпечення (ШПЗ) – це програма, яка при запуску може завдати шкоди пристрою різними способами, зокрема: блокувати пристрій, роблячи його непридатним для використання; красти, видаляти або шифрувати дані; використовувати пристрій для атак на інші пристрої; отримувати інформацію про облікові дані, що дозволяють доступ до систем або служб; незаконно майнити криптовалюту на вашому пристрої; використовувати платні послуги на основі ваших даних (наприклад, телефонні дзвінки на платні номери) тощо [15, с. 217].

Шкідливе програмне забезпечення – це зловмисна програма або код, що завдає шкоди кінцевим пристроям. У разі зараження пристрою шкідливим програмним забезпеченням (ПЗ) можливі несанкціонований доступ, пошкодження даних або блокування пристрою до сплати викупу. Шкідливе програмне забезпечення є однією з найпоширеніших форм кібератак. Воно може здійснювати декілька видів шкідливої діяльності: забороняти доступ до мережі, красти інформацію з жорсткого диска, порушувати роботу системи або виводити її з ладу. Шкідливе ПЗ може бути у формі шпигунських програм, які збирають інформацію для подальшого вимагання викупу, або програм-вимагачів, що шифрують дані користувача і вимагають викуп (зокрема, у криптовалюті) за їхнє розшифрування.

Майкрософт визначає шкідливе програмне забезпечення як зловмисну програму або код, який завдає шкоди кінцевим пристроям. Якщо пристрій заражений шкідливим програмним забезпеченням, може відбуватися несанкціонований доступ, пошкодження даних або блокування пристрою до моменту, коли ви не виплатите викуп [26].

Таким чином, з аналізу поняття шкідливого програмного забезпечення можна зробити декілька висновків. По-перше, його існування становить серйозну загрозу для безпеки інформації та нормального функціонування пристроїв. По-друге, розвиток цього виду програм стає все більш складним і винахідливим, що вимагає постійного вдосконалення засобів захисту. По-третє, розуміння природи та методів дії шкідливого програмного забезпечення допомагає розробникам програм та користувачам краще захищати свої системи та дані. У цілому, ця тема підкреслює важливість безпеки в інформаційному середовищі та необхідність постійного вдосконалення заходів захисту від кіберзагроз.

Одним з основних завдань, яке виконують аналітики шкідливого програмного забезпечення, є початкове сортування або класифікація зразків шкідливого ПЗ. Це може бути простим, наприклад, розпізнавання типу файлу, або складнішим, таким як визначення ступеня схожості з іншими зразками та виявлення спільних рис поведінки між варіантами того ж самого шкідливого програмного забезпечення.

Розглянемо основні види шкідливого ПЗ. Наприклад, програми-троянці (або «Троянські коні»). Це тип шкідливого програмного забезпечення, що приховується під звичайною корисною програмою, але насправді виконує додаткові шкідливі дії, які користувач не помічає. Одна з ключових особливостей цього типу шкідливого ПЗ – метод його поширення. Троянці не можуть самостійно копіювати себе в інші файли і не розповсюджуються через мережу, як віруси та черви. Наприклад, трояни можуть приховуватися в безкоштовних програмах для редагування мультимедіа, піратському програмному забезпеченні на недобросовісних веб-сайтах і т. д. [19, с. 102].

Інший тип – віруси, які заражають комп'ютери та інші файли шляхом

саморозмноження. Вони не можуть існувати самостійно, тому вони приєднуються до інших виконуваних файлів і програм. Інфікований код потім намагається заражати новий код. Саморозмноження є основною особливістю таких шкідливих програм.

«Мережеві черв'яки» – це тип шкідливого програмного забезпечення, яке має спільні риси з вірусами, проте їхньою основною відмінністю є методи поширення та вплив на систему. Віруси зазвичай потребують наявності хост-файлу для інфікування, у той час як черв'яки можуть самостійно поширюватися між комп'ютерами через мережу, використовуючи вразливості в операційній системі. Крім того, черв'яки можуть поширюватися іншими методами, такими як масова відправка себе електронною поштою на кожну адресу в адресній книзі інфікованого користувача. Зазвичай вони завдають шкоди своїм мережам, зменшуючи пропускну здатність і перевантажуючи веб-сервери. Черв'яки часто містять корисне навантаження, таке як «бекдори». Їхня здатність до самовідтворення та висока швидкість поширення дозволяють зловмисникам створювати мережі підконтрольних машин, які називають ботнетами. Ці мережі можуть використовуватися для проведення DDoS-атак та поширення спаму [26].

«Бекдори» або задні входи, представляють собою програмний код, який дозволяє зловмисникам отримувати несанкціонований доступ до комп'ютерної системи, програми або мережі, обходячи аутентифікацію та інші стандартні методи та технології безпеки. Після встановлення бекдор може отримати доступ до керування пристроєм, викрасти конфіденційну інформацію та використовувати комп'ютер для розсилки спаму або запуску DDoS-атак. Цей тип шкідливого програмного забезпечення часто потрапляє на пристрій жертви під час завантаження файлів користувачем. Наприклад, одна з наймасштабніших кібератак в Україні була здійснена бекдором NotPetya, який поширився через оновлення популярного програмного забезпечення «M.E.Doc». Загалом, збитки, спричинені цією вірусною атакою, сягнули 10 мільярдів доларів [12].

Щодо програм-шпигунів, вони приховано збирають конфіденційну інформацію про користувачів без їхньої згоди. Spyware може відстежувати

активність користувача в Інтернеті, збирати інформацію про веб-історію, паролі, банківські дані, адреси електронної пошти, а також контролювати дії користувачів, такі як запис натискань клавіш і знімки екрану. Інформація, зібрана шпигунською програмою, відправляється сторонньому суб'єкту чи організації і може використовуватися для реклами, злочинної діяльності, шахрайства, викрадення особистої інформації тощо. Такі програми зазвичай не мають здатності до саморозмноження і поширюються у вигляді безкоштовного або умовно-безкоштовного програмного забезпечення з мінімальними ліцензійними обмеженнями, щоб охопити якнайбільше користувачів [19, с. 103].

Рекламне програмне забезпечення (Adware) є програмою, яка відображає рекламу перед користувачами з метою заробітку для розробників. Зазвичай це безкоштовне програмне забезпечення, яке фінансується через показ реклами. Adware може використовувати різні методи для показу реклами, такі як випливаючі вікна, банери на веб-сторінках, реклама в програмах і т. д. Хоча саме за собою таке програмне забезпечення не є шкідливим, воно може бути нав'язливим, перериваючи роботу користувача та сповільнюючи пристрій. Крім того, деякі типи adware можуть збирати особисту інформацію користувача для персоналізованої реклами або продажу третім сторонам.

Ransomware – це програма, яка блокує доступ до файлів на комп'ютері користувача шляхом їх шифрування та вимагає викуп для повернення доступу до своїх даних. Розшифрування заблокованих даних можливе лише за допомогою ключа для розшифрування, який знаходиться у розпорядженні зловмисників. Поява криптовалют значно збільшила поширення програм-вимагачів, оскільки платежі у такій валюті здійснюються анонімно та їх майже неможливо відслідкувати [14, с. 140].

Потенційно небажані програми (Potentially Unwanted Programs, PUP) – це програми, які можуть бути встановлені на комп'ютер без згоди користувача або бути встановлені разом з іншими програмами, що завантажуються з Інтернету. Хоча вони не є шкідливими програмами у повному значенні цього слова, вони можуть містити рекламу, панелі інструментів і випливаючі вікна, які не мають

відношення до завантаженого програмного забезпечення [14, с. 140].

Комп'ютерні віруси можна класифікувати за методом зараження на резидентні і нерезидентні. Резидентний вірус, після того як проникає в комп'ютер, утримує свій код в оперативній пам'яті. Його небезпека полягає в тому, що він перехоплює всі звернення операційної системи до файлів та програм і може заражати будь-який з них. Такі віруси можуть блокувати певні функції системи, змінювати реєстр або вносити інші зміни, що дозволяють їм залишатися невидимими для антивірусного програмного забезпечення. Інфіковані дані в пам'яті продовжують свою діяльність на пристрої до того моменту, поки користувач не вимкне або перезавантажить систему. Нерезидентні віруси, навпаки, не зберігаються постійно в пам'яті комп'ютера. Вони можуть заражати файли безпосередньо при їх відкритті або виконанні.

Існує різновид класифікації комп'ютерних вірусів, яка визначається об'єктом, який вони заражають. Серед цих класифікацій виділяють такі (рис. 1.1):

1. Файлові віруси: вони прикріплюються до виконуваних файлів, документів або інших програм і поширюються через них. Після запуску зараженого файлу вірус поширюється на інші файли на комп'ютері і може призвести до їхнього пошкодження або видалення.

2. Завантажувальні віруси: ці віруси заражають boot-сектор завантажувального диска. Вони підміняють код програми, яка контролює запуск операційної системи, тому після завантаження системи контроль передається вірусу. На сьогоднішній день цей тип вірусів зустрічається дуже рідко.

3. Макро-віруси: вони впливають на програми обробки текстів, такі як Microsoft Word або Excel. Макро-віруси заражають макроси в документах та можуть поширюватися через електронну пошту або файли, що зберігаються на хостингах.

4. Комп'ютерні черви: ці віруси поширюються через мережу Інтернет і не потребують наявності файлу для інфікування. Їх поширення здійснюється шляхом використання вразливостей у мережі.

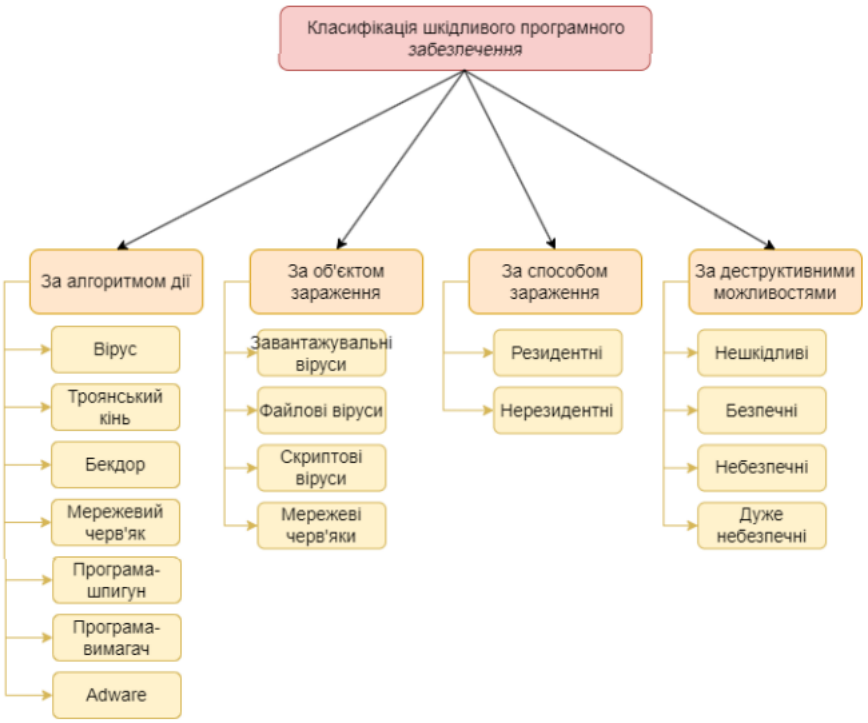

Рис. 1.1 – Узагальнена класифікація шкідливого ПЗ [15]

Комп’ютерні віруси можна класифікувати залежно від методу зараження на резидентні і нерезидентні. Резидентний вірус, після того як проникає в комп’ютер, утримує свій код в оперативній пам’яті. Його небезпечність полягає в тому, що він перехоплює усі звернення операційної системи до файлів та програм і може інфікувати будь-який з них. Резидентні віруси можуть блокувати деякі функції системи, вносити зміни у реєстр або здійснювати інші маніпуляції, що дозволяють їм залишатися невиявленими антивірусним програмним забезпеченням. Залишаючись у пам’яті, такі віруси продовжують свою діяльність на пристрої до тих пір, поки користувач не вимкне або перезавантажить систему. Нерезидентні віруси не залишаються в постійно в пам’яті комп’ютера; вони можуть заражати файли безпосередньо при їх відкритті або виконанні.

Таким чином, шкідливе ПЗ постійно еволюціонує та змінюється, адаптуючись до нових технологій та методів захисту. Шкідливе програмне забезпечення включає в себе різноманітні типи, такі як віруси, троянські коні, черв'яки, рекламне програмне забезпечення, програми-вимагачі, програми-шпигуни та потенційно небажані програми. Кожен з цих типів має свої унікальні

характеристики та методи зараження, а також може наносити різні види шкоди, включаючи крадіжку конфіденційних даних, блокування доступу до файлів, сповільнення роботи комп'ютера, підміну рекламних матеріалів та інше.

Однак, необхідно враховувати, що з виникненням нових видів шкідливого програмного забезпечення розвиваються також технології захисту. Антивірусні програми та інші засоби кібербезпеки стають все більш вдосконаленими в розпізнаванні та блокуванні нових загроз. Також зростає увага до освіти користувачів щодо потенційних ризиків та методів запобігання інфікування комп'ютерів шкідливим програмним забезпеченням. Враховуючи ці фактори, можна зробити висновок, що боротьба з шкідливим програмним забезпеченням є постійним процесом, в якому ключову роль відіграють технологічні рішення, освіта та свідомість користувачів.

1.2 Методи аналізу шкідливого програмного забезпечення

Виявлення шкідливого програмного забезпечення передбачає використання методів та інструментів для ідентифікації, блокування, запобігання та реагування на потенційні загрози. Основні методи виявлення шкідливого ПЗ можна класифікувати на три основні категорії: на основі сигнатур, на основі аналізу поведінки та на основі евристичного аналізу. Розглянемо кожну з цих категорій докладніше.

Аналіз на основі сигнатур: в даний час більшість антивірусних систем для виявлення шкідливого програмного забезпечення використовують сигнатурний аналіз. Сигнатура – це унікальний фрагмент коду, який характеризується тим, що зустрічається лише у певному шкідливому файлі або вірусі, але ніколи не з'являється в інших програмах. Цей код може містити певні характеристики, які відрізняють вірус від інших файлів, а також інформацію про поведінку вірусу в системі. Основні характеристики сигнатури включають:

- Унікальна послідовність байтів, яка є характерною саме для цієї програми.
- Хеш-сума окремого файлу або всієї програми.
- Унікальний рядок у коді програми.
- М'ютекси – це унікальні маркери, що використовуються для перевірки, чи був вже скомпрометований даний пристрій.
- Ключі реєстру, які часто створюються шкідливим програмним забезпеченням для збереження власних даних.
- Шлях до системної бібліотеки, до якої програма звертається.
- Шляхи до програмної бази даних, що містить різноманітну службову інформацію для налагодження програм.
- Зашифровані рядки конфігурації, оскільки зловмисне програмне забезпечення часто шифрує свою конфігурацію, що містить важливі ідентифікатори, такі як протокол, домен, порт, на який програма виконує запити.

На рисунку 1.2 наведена загальна процедура виявлення шкідливого ПЗ на основі сигнатурного аналізу:

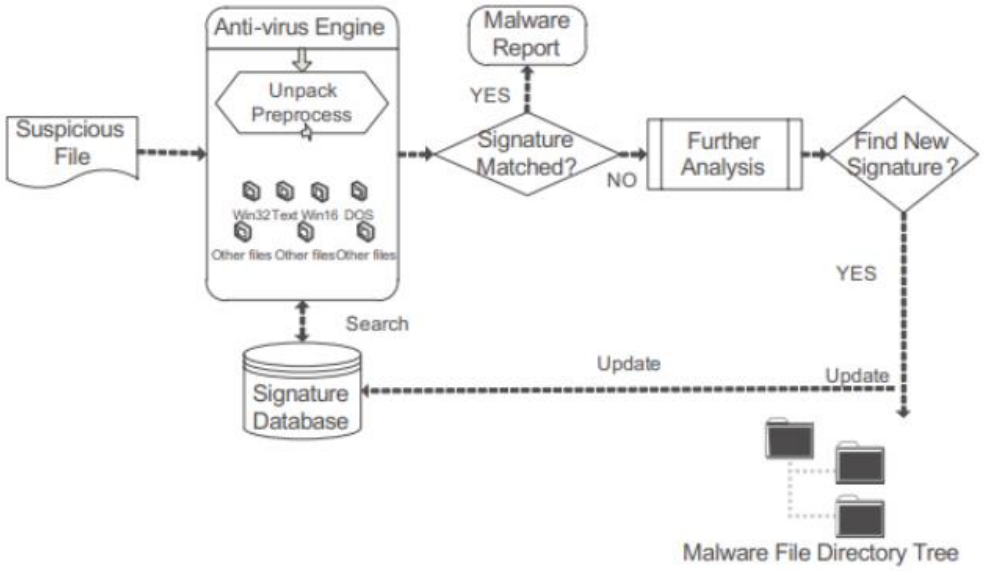

Рис. 1.2 – Загальна процедура виявлення шкідливого ПЗ на основі сигнатурного аналізу [16]

Після отримання підозрілого виконуваного файлу, його направляють на аналіз антивірусному механізму. Антивірус в першу чергу перевіряє, чи є цей

зразок упакованим. Якщо так, механізм розпаковує його та передає розпаковану версію на відповідний сканер ядра антивірусу. Сканер – це окрема частина механізму, яка працює з конкретним типом файлів. Розпакування виконуваного файлу – це вбудована функція в антивірусну систему, яка дозволяє видобути вихідний код програми для подальшого сканування. Далі антивірусний механізм аналізує асемблерний код файлу, порівнюючи конкретні байти коду з інформацією у своїй базі даних сигнатур. Якщо антивірусний продукт виявляє сигнатуру, яка відповідає відомому вірусу, то він класифікує цей файл як шкідливий та відповідно до налаштувань захисту може виконати одну з таких дій: видалити інфікований файл; помістити файл у карантин (тимчасове сховище файлів, де він шифрується для запобігання подальшого запуску та розповсюдження); спробувати «вилікувати файл», видаливши з нього шкідливий код, що дозволяє залишити корисний вміст [16, с. 42].

Якщо сигнатура не знайдена у файлі, тоді підозрілий файл передається антивірусним інженерам для подальшого аналізу і оновлення бази даних сигнатур шкідливого програмного забезпечення. На рисунку 1.3 зображено приклад частини сигнатури вірусу Casper описаний з використанням правил YARA («Yet Another Recursive Algorithm») [12]:

```
rule Casper_Included_Strings
{
    meta:
        description = "Casper French Espionage Malware - String Match in File - http://goo.gl/VRJNLo"
        author = "Florian Roth"
        reference = "http://goo.gl/VRJNLo"
        date = "2015/03/06"

    strings:
        $a0 = "cmd.exe /C FOR /L %%i IN (1,1,%d) DO IF EXIST"
        $a1 = "& SYSTEMINFO) ELSE EXIT"
        $mz = { 4d 5a }
        $c1 = "domcommon.exe" wide fullword                     // File Name
        $c2 = "jpic.gov.sy" fullword                            // C2 Server
        $c3 = "aiomgr.exe" wide fullword                        // File Name
        $c4 = "perfaudio.dat" fullword                          // Temp File Name
        $c5 = "Casper_DLL.dll" fullword                         // Name
        $c6 = { 7B 4B 59 DE 37 4A 42 26 59 98 63 C6 2D 0F 57 40 }   // Decryption Key
        $c7 = "{4216567A-4512-9825-7745F856}" fullword          // Mutex
    condition:
        all of ($a*) or ( $mz at 0 ) and ( 1 of ($c*) )
}
```

Рис. 1.3 – Приклад частини сигнатури вірусу Casper описаний з використанням правил YARA [12]

На сьогодні сканування на основі сигнатур є одним із ключових методів роботи антивірусного програмного забезпечення. Оновлення сигнатур становить

важливу складову захисту від шкідливих програм, оскільки це дозволяє антивірусному програмному забезпеченню підтримувати базу даних сигнатур вірусів у відповідності з новими загрозами, що з'являються в мережі Інтернет.

Недоліком системи виявлення на основі сигнатур є те, що для отримання сигнатури потрібно мати зразок вірусу, тому вона може виявити лише існуюче шкідливе ПЗ, яке вже є в базі сигнатур. Ще одним недоліком є розмір і обслуговування бази даних сигнатур. З появою нових зразків зловмисного програмного забезпечення база даних сигнатур зростає експоненційно, оскільки для кожного зразка потрібна окрема сигнатура. Частково ця проблема розв'язується за допомогою написання універсальних сигнатур, які описують цілі сімейства схожих за поведінкою вірусів. Однак це може призвести до помилкових спрацьовувань. Необхідність частого та регулярного оновлення бази даних сигнатур також є недоліком, тому вірусні аналітики потребують інструментів, які дозволять спростити та автоматизувати процес аналізу зразків шкідливого ПЗ [18, с. 47].

Серед основних переваг цього методу виявлення шкідливого ПЗ є те, що для аналізу зразка не потрібно запускати виконуваний файл. Це важливо, оскільки більшість вірусів розпочинають свій цикл життя після запуску виконуваного файлу, до якого вони прикріплені. Цей метод забезпечує високу точність виявлення вже відомих зразків при невеликих витратах ресурсів, а також найбільшу швидкодію порівняно з іншими методами.

II. Аналіз поведінки. На відміну від попереднього методу, процес виявлення небезпечного програмного забезпечення відбувається шляхом аналізу дій об’єкта. Для цього в реальному часі реєструються всі дії, які виконує програма або процес. Це включає: спроби взаємодії з системними файлами або процесами; спроби форматування розділів накопичувача або інші незворотні операції з диском; спроби відкрити або видалити файли; намагання модифікувати виконувані файли, скрипти або макроси; модифікація конфігураційних файлів або файлів реєстру операційної системи; мережева активність [19, с. 106].

Якщо детектор виявляє аномалії в поведінці досліджуваної програми, то він

позначає її як потенційно небезпечну, сповіщає про це та вживає заходів для запобігання пошкодження системи. За алгоритмом роботи аналізатори поведінки дуже схожі на системи виявлення вторгнень на базі хоста і можуть бути їх складовою.

Аналіз поведінки може використовуватися як самостійний метод виявлення шкідливого ПЗ, але найбільшу ефективність він має у поєднанні з іншими методами. Основною перевагою є те, що цей метод дозволяє виявляти нові зразки шкідливого програмного забезпечення, яких ще не має в базі антивірусних сигнатур. Тому цей метод підходить для виявлення невідомих і поліморфних вірусів.

Серед недоліків варто зазначити те, що аналізатори даного типу часто мають значний процент хибних спрацювань. Тому важливо враховувати, що програми або антивірусні модулі, які ґрунтуються на методі поведінкового аналізу, можуть видавати значну кількість попереджень. Це може призвести до того, що користувачі перестануть реагувати на всі попередження, що зменшить ефективність виявлення нових загроз. Використання технології поведінкового аналізу не передбачає «лікування» модифікованих вірусом файлів і програм. Тривалий час сканування та більше, в порівнянні з сигнатурним методом, споживання ресурсів є ще одним з недоліків систем, побудованих на основі аналізу поведінки [19, с. 107].

III. Евристичний метод – це спосіб виявлення шкідливого програмного забезпечення шляхом перевірки коду на наявність підозрілих ознак за допомогою правил прийняття рішень або методів зважування. Ця технологія базується на припущенні, що нові віруси можуть мати спільні риси з вже вивченими зразками. Процес виявлення включає три основні етапи: статичний аналіз, динамічний аналіз та багатокритеріальний аналіз [17].

Під час статичного аналізу коду зразка визначаються ознаки, характерні для шкідливого програмного забезпечення. Потім проводиться динамічний аналіз для виявлення підозрілих активностей. Кожна ознака або активність отримує вагу, і якщо загальний коефіцієнт шкідливості перевищує встановлений ліміт, зразок

вважається шкідливим. Хоча евристичні методи можуть виявляти раніше не відомі зразки шкідливого програмного забезпечення, вони також мають високий рівень помилкових сигналів. Використання цього методу може вимагати значних обчислювальних ресурсів для аналізу поведінки програм та виявлення можливих загроз, що може призвести до зниження продуктивності системи, на якій працює антивірусна програма.

Важливо відзначити, що «вилікувати» інфіковані вірусом файли стане можливо лише після вивчення зразка, додавання інформації до бази сигнатур і розробки способу лікування. Евристичне сканування може бути неефективним проти передових новаторських вірусних програм, які не схожі на інші комп'ютерні віруси (вразливості «нульового дня»). На основі проведеного аналізу сформуємо основні переваги та недоліки методів виявлення шкідливого ПЗ, занесемо їх у Таблицю 1.1:

Таблиця 1.1 – Переваги і недоліки методів виявлення шкідливого ПЗ

Метод	Переваги	Недоліки
Сигнатурний аналіз	легке виявлення відомих зразків шкідливого ПЗ, мінімальне використання ресурсів	не може впізнати невідомі шаблони, потребує періодичного оновлення антивірусних баз
Аналіз поведінки	виявлення невідомих зразків	значне споживання ресурсів, високий рівень хибних спрацювань
Евристичний аналіз	виявлення відомих і невідомих зразків	вимагає постійного оновлення даних про віруси, потребує додаткового часу та обчислювальних ресурсів для перевірки, підвищений ризик неправильного визначення

Джерело: складено автором

Аналіз шкідливого програмного забезпечення важливий як у відновленні після кібератак, так і у запобіганні таким атакам у майбутньому. Це допомагає розробникам антивірусного програмного забезпечення виявляти та блокувати нові форми шкідливого ПЗ та розробляти методи захисту від атак. На рис. 1.4 показаний типовий процес аналізу шкідливого ПЗ:

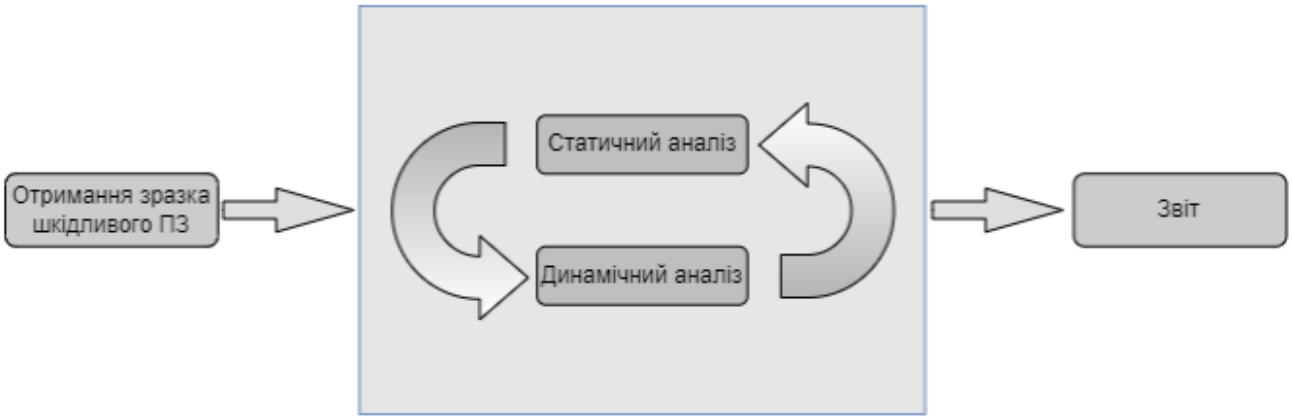

Рис. 1.4 – Діаграма процесу аналізу шкідливих програм [20]

Під час статичного аналізу проводиться перевірка шкідливих файлів або програм без їх активації. Це найбезпечніший метод аналізу, оскільки запуск коду може призвести до інфікування системи. Базовий підхід до статичного аналізу полягає у вивченні виконуваного файлу без розгляду його коду. Під час цього аналізу вивчаються такі параметри, як тип і розмір файлу, використані бібліотеки та функції, ресурси та рядки. Також обчислюється контрольна сума (хеш) підозрілого файлу, щоб визначити, чи був він раніше досліджений антивірусними лабораторіями.

Під час динамічного аналізу (також відомого як аналіз поведінки) відбувається запуск шкідливого програмного забезпечення для спостереження його активності. Оскільки цей процес завжди супроводжується певним ризиком, виконання динамічного аналізу рекомендується в безпечному середовищі. Зазвичай цю роль виконує середовище «пісочниці» – віртуальна система, ізольована від основної мережі, яка дозволяє запускати шкідливе ПЗ без наслідків для робочих систем. Щоб забезпечити об'єктивність дослідження, пісочниця повністю моделює характеристики хоста, включаючи процесор, системну пам'ять та всі пристрої. Пісочниці можуть здійснювати знімки свого стану, що дозволяє легко відновити їх до початкового стану після завершення аналізу [20].

Після запуску шкідливого ПЗ у віртуальному середовищі фіксуються різні важливі дані, такі як мережевий трафік, системні виклики, зміни файлів, нові ключі реєстру, IP-адреси, доменні імена і т. д. Ці дані використовуються для аналізу

поведінки шкідливої програми, виявлення ознак компрометації та оцінки потенційного впливу на реальну систему.

Метод аналізу у віртуальному середовищі виявляється особливо ефективним, оскільки він є проактивним і дозволяє виявляти загрози без ризику для головної системи. Проте деякі шкідливі програми можуть розпізнати, що вони працюють в пісочниці та змінити свою поведінку для унеможливлення аналізу. Це може ускладнити процес аналізу. Також серед недоліків можна відзначити те, що аналіз може бути повільнішим через необхідність підготовки віртуального середовища.

ВИСНОВОК ДО РОЗДІЛУ 1

Шкідливе програмне забезпечення – це зловмисна програма або код, що завдає шкоди кінцевим пристроям. У разі зараження пристрою шкідливим програмним забезпеченням (ПЗ) можливі несанкціонований доступ, пошкодження даних або блокування пристрою до сплати викупу. Шкідливе програмне забезпечення є однією з найпоширеніших форм кібератак. Воно може здійснювати декілька видів шкідливої діяльності: забороняти доступ до мережі, красти інформацію з жорсткого диска, порушувати роботу системи або виводити її з ладу.

Різні методи аналізу, такі як аналіз на основі сигнатур, поведінковий аналіз та евристичні методи, мають свої переваги та обмеження. Методи аналізу на основі сигнатур дозволяють ефективно виявляти відомі шкідливі програми, але потребують постійного оновлення баз даних для виявлення нових загроз. Поведінковий аналіз, з іншого боку, може виявляти невідомі шкідливі програми за їх характеристиками виконання, але може стикатися з великою кількістю хибних спрацювань та вимагати значних обчислювальних ресурсів. Евристичні методи можуть допомагати виявляти нові типи загроз, але також мають свої обмеження. Усі ці методи можуть бути використані окремо або в поєднанні для забезпечення більш ефективного виявлення та захисту від шкідливого програмного

забезпечення. Крім того, важливо враховувати, що зростання складності шкідливого програмного забезпечення вимагає постійного вдосконалення та розвитку методів аналізу для ефективного протистояння кіберзагрозам.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ

2.1 Автоматична класифікація та виявлення шкідливих програм за допомогою машинного навчання та нейронних мереж

Шкідливе програмне забезпечення – це будь-яке програмне забезпечення, яке завдає шкоди користувачам, комп'ютерам або мережам. Це можуть бути віруси, хробаки, бекдори, троянські коні та інші шкідливі програми. В наш час це стає серйозною проблемою в галузі інформаційної безпеки. Щоденно виявляються сотні тисяч нових шкідливих програм, більшість з яких походять від відомих сімей шкідливих програм.

Методи маскування шкідливого програмного забезпечення включають пакування, метаморфозу та використання віртуальних технологій. Вони широко використовуються для уникнення виявлення антивірусним програмним забезпеченням. Більшість систем виявлення шкідливих програм базуються на виявленні підписів та аномалій. Техніка підписів, хоча має високу точність, не може впоратися з новими шкідливими програмами і вимагає постійного оновлення своєї бібліотеки функцій. Нове шкідливе програмне забезпечення можна виявити за допомогою виявлення аномалій, але це часто призводить до високої кількості помилкових тривог [21, с. 390].

Залежно від характеристик шкідливого програмного забезпечення, аналіз його можна розглядати у двох аспектах: статичний та динамічний. Статичний аналіз означає аналіз файлів без їх виконання. Він має перевагу у виявленні авторського стилю та аналізі потоку коду. Але його легко обійти за допомогою методів замаскування. З іншого боку, динамічний аналіз дозволяє спостерігати роботу програми в безпечному середовищі, відображаючи її поведінку точно. Цей

метод не впливає на ефективність шифрування, стиснення чи метаморфозу. Проте він вимагає часу на налагодження та відстеження процесів програми, що робить його менш ефективним у порівнянні зі статичним аналізом. Крім того, динамічний аналіз обмежується робочим середовищем і може некоректно виявляти деякі зловмисні дії через умови активації [23, с. 105].

Завдяки успішному впровадженню машинного навчання в обробку зображень, розпізнавання мови та програмну інженерію, цей метод став важливим для аналізу шкідливих програм за останнє десятиліття. Зазвичай процес виявлення шкідливих програм за допомогою машинного навчання включає три етапи.

По-перше, функції виконуваного файлу, які отримані через статичний та динамічний аналіз, перетворюються на вхідні дані для машинного навчання. По-друге, на основі цих функцій випробується модель прогнозування, що дозволяє створити високорівневий підпис шкідливого програмного забезпечення. І, нарешті, проводиться прогнозування для невідомого програмного забезпечення.

Машинне навчання здатне автоматично аналізувати та виявляти характеристики програмного забезпечення, що дозволяє відрізняти доброякісне програмне забезпечення від шкідливого. Крім того, воно може класифікувати невідоме шкідливе програмне забезпечення до відомих сімейств. Однак, автори шкідливих програм можуть швидко створювати численні варіанти шкідливих програм за допомогою автоматичних інструментів. Тому необхідно вирішувати такі проблеми за допомогою машинного навчання:

- Як швидко та ефективно відносити варіанти до відповідних сімейств.
- Як виявляти нове шкідливе програмне забезпечення.

Таким чином, машинне навчання та нейронні мережі стали ефективними інструментами для автоматичної класифікації та виявлення шкідливих програм. Завдяки їх здатності аналізувати великі обсяги даних і виявляти складні патерни, вони перевершують традиційні методи виявлення, які часто базуються на сигнатурах та виявленні аномалій. Ці технології можуть швидко адаптуватися до нових загроз, класифікувати шкідливе програмне забезпечення та навіть відносити його до відомих сімейств.

Проте, виклики залишаються, зокрема у здатності протидіяти різноманітності та швидкому розповсюдженню нових варіантів шкідливих програм. Для ефективного використання машинного навчання та нейронних мереж необхідно продовжувати удосконалювати моделі, забезпечувати актуальність даних для тренування та розробляти нові підходи для зниження кількості хибних спрацьовувань. Інтеграція цих технологій в системи інформаційної безпеки забезпечує більш надійний захист проти сучасних загроз кібербезпеки.

2.2 Основи машинного навчання та нейронних мереж

Оскільки кількість шкідливих програм теоретично необмежена, цю задачу потрібно автоматизувати. Для цього нам необхідний класифікатор, який зможе виконувати таку ж функцію. Класифікатори можуть бути різних типів. Один із способів створення класифікатора — це використання евристики, тобто набору правил, розроблених на основі вивчення різних типів шкідливих програм. Наприклад, якщо у файлі виявляються певні типи заголовків, з високою ймовірністю можна припустити, що це шкідливе програмне забезпечення. Однак існує безліч способів створення зловмисних програм, і розробка евристики для всіх типів є складним завданням. Крім того, нам може знадобитися спершу дослідити нове шкідливе програмне забезпечення, щоб оновити наші евристичні правила, а на той час багато машин уже можуть бути заражені. Тому потрібні класифікатори, які автоматично вивчають структуру файлів і ключові характеристики шкідливого програмного забезпечення, та ефективно узагальнюють ці знання. Для вирішення цього питання у цій роботі використовуються штучні нейронні мережі [3].

Штучні нейронні мережі частково натхнені роботою біологічного мозку, моделюючи його за допомогою статистики та прикладної математики. Людський мозок складається з безлічі взаємопов'язаних клітин, званих нейронами, які передають інформацію у вигляді електричних сигналів. Мозок отримує значну

кількість сенсорної інформації, і нейронні мережі обробляють її у свідомій частині мозку. Кожен нейрон можна змоделювати як нелінійну функцію $f(x)$, яка приймає вхідні сигнали від інших нейронів у формі електричних імпульсів. Сила цих сигналів змінюється під час їхнього проходження від одного нейрона до іншого, залежно від хімічного складу з'єднання. Якщо сумарний вхідний сигнал перевищує певний поріг, нейрон генерує вихідний сигнал y, який потім надходить до інших нейронів, як показано на рис. 2.1. Цей процес триває у складній мережі нейронів, що створює дуже складну функцію. Вхід нейрона X можна представити як $w1x1 + w2x2 + \cdots + wnxn$, де wi є вагами, що модифікують силу сигналу. Потім нейрон використовує безперервну функцію активації $\varphi(X)$. Під час навчання і набуття нових знань старі зв'язки змінюються, а нові формуються, що, як вважається, і є основою процесу навчання [7].

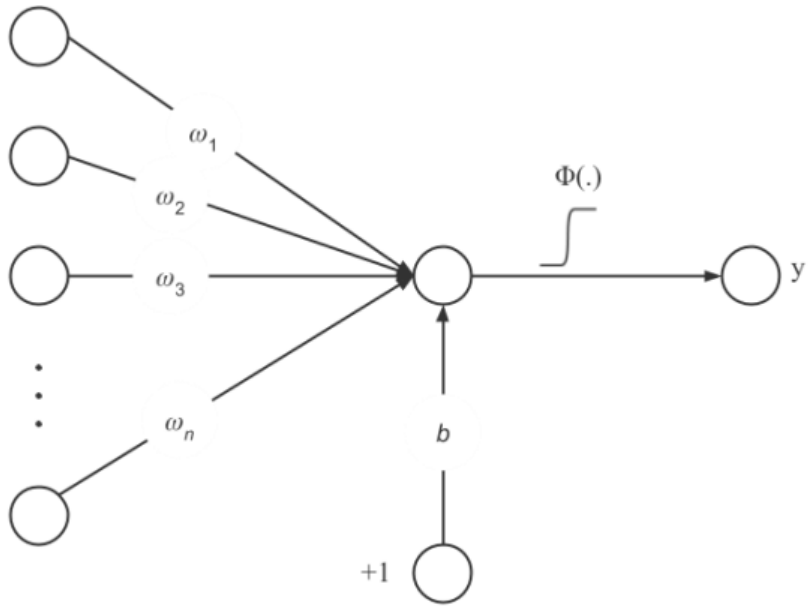

Рис. 2.1 – Графічне зображення того, як працює нейрон. На зображенні: y – вихідний сигнал, φ – функція активації, xi – входи інших сполучених нейронів, wi – відповідні ваги. b додає поріг активації [7]

Штучні нейронні мережі (ШНМ) складаються з взаємопов'язаних одиниць, що мають назву штучних нейронів, організованих у шари. Дані переміщуються від вхідного до вихідного шару, проходячи через численні проміжні шари. Кожне з'єднання між двома нейронами має вагу wi, зазвичай це реальне число в діапазоні від 0 до 1, яке визначає, як модифікується вхідний сигнал перед передачею

наступному нейрону. Крім того, на кожному вузлі зазвичай є порогова або активаційна функція. Ці шари можуть бути повністю з'єднаними (коли кожен нейрон з'єднаний з усіма нейронами в наступному шарі) або частково з'єднаними (вибрано випадковим чином). Таким чином, кожен нейрон можна представити як функцію [25]:

$$y = \varphi(\sum_{i=1}^{n} \omega_i x_i + b) \quad (2.1)$$

де y – вихід;

φ – функція активації;

n – кількість вхідних нейронів, що надходять у нейрон;

wi – величина ваги і-го з'єднання;

xi – вихід нейронів, що подаються в нейрон;

b – поріг.

Нейрон b можна вважати нейроном із постійним виходом значенням -1. Однак, насправді значення b варіюється для різних нейронів, що дозволяє мережі вибирати різні пороги активації для кожного нейрона. На рис. 2.2 зображена повна нейронна мережа, яка складається з трьох типів шарів: вхідного, прихованого та вихідного. Вхідний шар обробляє необроблену інформацію, яка подається в мережу. Приховані шари отримують трансформоване зображення даних, визначене вагами та активаційними функціями попередніх шарів. Це важливо, оскільки приховані шари можуть самостійно навчитися представляти вхідні дані, полегшуючи задачу класифікації для наступних шарів [25].

Мережу можна уявити як набір шарів, де кожен шар витягує різне представлення вхідних даних, видаляючи зайву інформацію та зберігаючи лише корисну для виконання завдання. Такі штучні нейрони також називають перцептронами, а мережу з кількома шарами нейронів – багатошаровим перцептроном. Нейронні мережі, в яких виходи кожного шару подаються як входи до наступного шару, називаються FeedForward Networks (нейронні мережі прямого поширення).

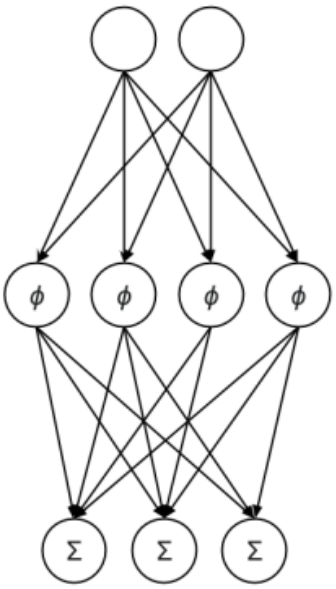

Рис. 2.2 – Графічне зображення штучної нейронної мережі із одним прихованим шаром [25]

Класифікатор повинен автоматично навчатися функціям. Штучні нейронні мережі досягають цього шляхом коригування своїх ваг так, щоб їхні виходи наближалися до бажаного результату для кожного наданого прикладу. Існують два основні типи навчання: контрольоване (supervised) і неконтрольоване (unsupervised). У контрольованому навчанні відомий очікуваний результат для кожного навчального прикладу, і ця інформація використовується під час навчання. В неконтрольованому навчанні правильні результати невідомі, і завдання полягає в тому, щоб об'єднати схожі дані або розділити різні дані, або вивчити розподіл вхідних даних.

Контрольоване навчання застосовується, коли є розмічені дані, як у цій дипломній роботі. Під час навчання приклади з набору даних подаються по черзі, і ваги змінюються на основі помилки, допущеної моделлю. Це можна розглядати як метод навчання через помилки. Процес триває доти, доки різниця між виходами мережі та бажаними виходами не стане дуже малою. Існує багато способів досягнення цього. Один із популярних методів – це випадкова ініціалізація ваг нейромережі за допомогою деякого розподілу. Вхідні дані X подаються в мережу, а вихідні дані $\hat{y}$ отримуються. Потім обчислюється помилка, яка визначає, наскільки $\hat{y}$ відрізняється від бажаного виходу y. Мета полягає в

тому, щоб мінімізувати цю помилку. Важливо, як саме обчислюється помилка, оскільки це визначає задачу оптимізації. Один із методів – обчислення середньої квадратичної помилки за допомогою методу найменших квадратів. На основі цієї помилки ваги оновлюються методом зворотного поширення помилки (backpropagation). Зворотне поширення ґрунтується на тому, що оптимальною зміною ваг є така, яка зменшує помилку на невеликий крок дельта. З лінійної алгебри відомо, що помилка зменшується найшвидше в напрямку, де негативний нахил функції помилки є максимальним. Щоб досягти мінімуму, необхідно робити багато маленьких кроків у цьому напрямку, поки значення помилки не стане мінімальним [24, с. 126].

Навчання відбувається пакетами (batches). Один пакет включає всі приклади з набору даних або певну фіксовану кількість прикладів, вибраних випадковим чином із набору даних. Пакети подаються доти, доки помилка не стане достатньо низькою або перестане змінюватися.

Розпізнавання зображень є одним із ключових застосувань, де нейронні мережі показують вражаючі результати. Штучні нейронні мережі вже активно використовуються в промисловості, охороні здоров'я, управлінні ризиками тощо. Як вже зазначалося, структура штучних нейронних мереж складається з вхідного шару, одного або більше прихованих шарів та вихідного шару. Багато завдань з розпізнавання зображень є надто складними для засвоєння за допомогою одного прихованого шару. Розглядаючи кожен шар як функцію fi на вході, можна використовувати багато шарів для навчання складніших функцій: $f4(f3(f2(f1)))$. Збільшення кількості шарів дозволяє навчатися складнішим функціям, тобто виконувати більш складні завдання. Кількість шарів у нейронній мережі визначає її глибину. Моделі з багатьма прихованими шарами називаються глибокими моделями, а процес навчання таких мереж – глибоким навчанням. Використання глибокого навчання в розпізнаванні зображень дозволяє досягти значно вищої точності, ніж попередні методи. Далі буде докладно розглянуто глибокі мережі [25].

Найпростіші класифікатори представлені лінійними моделями, які можуть

бути застосовані до ситуацій, де класифікаційну задачу можна вирішити за допомогою лінійного перетворення вхідних даних. Один з найпоширеніших прикладів цього – лінійна регресія. У лінійній регресії ми маємо скалярну залежну змінну y (вихід) та одну чи більше описових змінних, позначених X (вхід). Шляхом використання багатьох точок даних ми намагаємося знайти модель β, щоб [25]:

$$y_i = X_i^T \beta + \varepsilon_i \quad (2.2)$$

де $Xi\ T$ – і-тий елемент транспонованого вектора описових змінних;

β – лінійна модель;

εi – випадкова змінна, що моделює шум під час збору даних;

n – кількість описових змінних (розмір вектора описових змінних);

i – лічильник від 1 до n.

Звичайні штучні нейронні мережі добре справляються з розпізнаванням та класифікацією зображень, проте вони мають проблему з багаторозмірністю, тому можуть застосовуватися лише до зображень невеликого розміру. Наприклад, для зображення розміром 32x32 пікселів потрібно 3072 ваг на вході. Збільшення розміру призводить до значної кількості ваг і, отже, ускладнює процес навчання обчислювально. Крім того, через плоску структуру звичайні нейронні мережі не можуть врахувати взаємозв'язок між сусідніми пікселями і, відповідно, не можуть повністю використовувати просторову структуру зображень. Ці обмеження було подолано завдяки згортковим нейронним мережам (CNN – Convolutional Neural Network) [24].

Отже, машинне навчання та нейронні мережі – це потужні інструменти, що революціонізують різні галузі. Вони використовуються для розпізнавання зображень, класифікації даних, прогнозування та рекомендацій. Згорткові нейронні мережі особливо ефективні у роботі з зображеннями, оскільки враховують просторову структуру даних. Загалом, ці технології відкривають безліч можливостей для вирішення складних завдань у різних галузях.

2.3 Попередня робота з застосування машинного навчання у класифікації шкідливого програмного забезпечення

Зловмисне програмне забезпечення використовується для атак на критичну інфраструктуру, шпигунства проти держав, крадіжки приватної інформації або фінансових шахрайств. Майже всі ці атаки використовують мережу як засіб. При цьому, майже всі системи виявлення зловмисного програмного забезпечення в цій галузі застосовують або підхід на основі підписів, або підхід на основі виявлення аномалій. Підпис – це унікальна послідовність байтів, яка наявна у шкідливому бінарному файлі та в пошкоджених ним файлах. Методи на основі підписів використовують унікальні підписи, розроблені антивірусними компаніями на основі відомих зловмисних програм. Цей підхід є швидким і має високу точність, але він не здатен виявити раніше невідомі зловмисні програми. Зазвичай, лише після зараження новим зловмисним програмним забезпеченням багатьох систем аналітики можуть створити підпис і додати його в базу. Крім того, база підписів потребує ручної підготовки, що є трудомістким процесом. При підході на основі виявлення аномалій, антивірусні компанії створюють базу даних дій, які вважаються безпечними. Якщо процес порушує одне з цих заздалегідь визначених правил, він позначається як шкідливий. Хоча цей метод дозволяє виявити нові, раніше невідомі зразки зловмисного програмного забезпечення, він характеризується високою кількістю хибно позитивних результатів.

Ще один поширений метод – це евристичний підхід. У цьому підході аналітики використовують методи машинного навчання для створення класифікатора шкідливих програм. Статичні, динамічні, візуальні характеристики програм або їх комбінації застосовуються для навчання класифікатора на наборі даних, який включає злоякісні та доброякісні двійкові файли. Для класифікації та виявлення зразків зловмисного програмного забезпечення використовуються різні методи машинного навчання, такі як опорновекторні машини, випадкові ліси, дерева рішень, Naive Bayes, кластеризація за допомогою K-середнього, Gradient

Boost та Ada-Boost [24].

Статичний аналіз передбачає вилучення статичних характеристик із двійкового файлу за допомогою інструментів бінарного аналізу. Дослідники використовують такі деталі, як список DLL-функцій, викликаних виконуваним файлом, кількість унікальних системних викликів для кожної DLL, список DLL, які використовує виконуваний файл, символи для друку або рядки, закодовані у двійковому файлі, та послідовності байтів, отримані з hex-дампу, як ознаки для класифікації. Застосовують алгоритм Multinomial Naive Bayes для класифікації набору з 3265 шкідливих та 1001 доброякісних зразків програм і досягли точності 97,11%. Це було однією з перших спроб аналізу зловмисного програмного забезпечення з використанням методів майнінгу даних [9].

Колтер та інші досліджували точність різних методів машинного навчання, таких як Naive Bayes, опорновекторні машини, дерева рішень та їх розширені версії, для класифікації шкідливих програм у різні сімейства, використовуючи ознаки, запропоновані Шульцем та ін. Вони виявили, що покращені дерева рішень забезпечували найвищу точність класифікації. Zhang та Reeves запропонували автоматизований метод статичного аналізу для ідентифікації метаморфних зловмисних програм, які зазвичай уникають традиційних методів виявлення на основі підписів. Вони обчислювали ступінь схожості між двома виконуваними файлами, використовуючи список викликів бібліотечних функцій, зроблених цими файлами, і продемонстрували високу точність у виявленні метаморфних варіантів [4].

Контрольовані алгоритми навчання потребують великої кількості мічених двійкових файлів як злоякісних, так і доброякісних класів для ефективного виявлення зловмисних програм. Оскільки отримати такий мічений набір даних важко, був запропонований напівконтрольований метод для виявлення раніше невідомих зразків зловмисного програмного забезпечення. Напівконтрольоване навчання є дуже корисним, коли доступно мало мічених даних. Воно намагається навчити класифікатор, використовуючи наявні мічені дані, і передбачає мітки для немічених даних протягом кількох ітерацій. На кожній ітерації деякі немічені дані,

прогнози класів яких перевищують встановлений поріг впевненості, переміщуються до категорії мічених даних. Для представлення виконуваних файлів використовується метод розподілу байтових n-грам. Застосовують напівконтрольований алгоритм LLGC (Learning with Local and Global Consistency), де можна досягти точності 88,25% у виявленні зловмисних програм. Хоча точність такої моделі менша порівняно з іншими моделями, вдається зменшити кількість необхідних мічених зразків до 2 тисяч, зберігаючи досить високу точність класифікації [13].

Інші дослідники запропонували новий підхід для виявлення черв'яків серед доброякісних двійкових файлів, використовуючи послідовності інструкцій змінної довжини та методи машинного навчання. Вони застосували випадкові ліси та дерева рішень для класифікації набору даних, що складався з 1444 черв'яків і 1330 доброякісних файлів, і досягли точності 96%. Кан та інші використовували машинне навчання на послідовностях n-опкодів за допомогою класифікатора SVM (опорновекторна машина) і отримали точність 98% у виявленні зловмисних програм. Вони застосували оп-коди, мнемоніку машинного коду, а також методи косинусної схожості та критичні послідовності інструкцій у своєму процесі виявлення [2]. Сун та інші досліджували використання послідовностей викликів API для виявлення зловмисного програмного забезпечення і показали, що всі версії одного й того ж зловмисного програмного забезпечення мають спільний поведінковий профіль, який можна визначити за допомогою послідовностей викликів API [10].

Мозер та співавтори пропонували схему, що базується на техніці обфускації, для дослідження недоліків статичного аналізу. Їхні дослідження показали, що лише статичного аналізу недостатньо для ефективного аналізу зловмисного програмного забезпечення. Оскільки можна легко уникнути статичний аналіз, якщо зловмисне програмне забезпечення обфусковане або запаковане, для аналізу необхідні додаткові надійні поведінкові ознаки. Це розуміння спонукає до різних підходів до динамічного аналізу [11]. Один з ключових аспектів усіх підходів, що базуються на динамічному аналізі, полягає в тому, що бінарні зразки виконуються у

контрольованому середовищі для аналізу їх поведінкових особливостей всередині віртуальної машини.

Андерсон та співавтори впровадили новий метод, який моделює графи ланцюгів Маркова з трасування, отриманого під час виконання двійкового файлу на платформі Ether, яка призначена для аналізу шкідливого програмного забезпечення. У цих графах вершини відображають одиниці трасування (інструкції), а ймовірність переходу оцінюється на основі даних, зібраних у звіті про трасування. Ядра графу відображають схожість між графами трасування на глобальному та локальному рівнях, створюючи таким чином матрицю схожості або ядерну матрицю між графами. Локальна схожість між графами оцінюється за допомогою ядра Гаусса, тоді як глобальна схожість вимірюється за допомогою спектрального ядра. Ці матриці схожості використовуються як вхідні дані для опорно-векторної машини з метою класифікації. У їхньому експерименті автори використали набір даних, який містив 615 доброякісних та 1615 злоякісних зразків, і повідомили про точність класифікації на рівні 96,41%. Незважаючи на те, що цей підхід демонструє високу точність, його обмежує значно великий час обчислень, що робить його непридатним для використання у реальних програмах [1].

Також була розроблена гібридна модель класифікації двійкових файлів на доброякісні та шкідливі, яка використовує як динамічні, так і статичні ознаки. Для статичних ознак використовувалися такі параметри, як друковані рядки та частота довжини функцій, а для динамічних – параметри API та назви функцій API. З метою тестування своєї моделі було використано 2939 зразків шкідливого програмного забезпечення та 541 доброякісних зразків. Автори використовували інтегровані метакласифікатори, такі як SVM, IB1, DT та RF, для класифікації зразків шкідливого програмного забезпечення і повідомили про точність на рівні 97.055% [3]. Гібридні підходи є дуже перспективними, оскільки значно підвищують ефективність класифікації порівняно з використанням тільки статичних або динамічних методів.

Макандар та його колеги перетворювали шкідливе програмне забезпечення у двовимірне зображення у відтінках сірого і класифікували зразки, використовуючи

текстурні ознаки. Вони витягували глобальні ознаки, базуючись на текстурі файлів, за допомогою вейвлет-перетворення Габора та GIST. Для проведення експериментів вони використали набір даних Mahenhur, який включав 3131 зразок двійкових файлів з 24 унікальних сімейств шкідливих програм. Для класифікації шкідливих програм вони використовували штучні нейронні мережі і повідомили про точність на рівні 96.35% [6].

Liu та його співавтори запропонували поетапний підхід до автоматичного визначення сімей зловмисних програм та виявлення нових шкідливих програм. Вони використовували комбінацію ознак, таких як сірий байтовий точковий графік, n-грами оп-коду та імпорт функцій. Модуль прийняття рішень використовував ці ознаки для класифікації зразків шкідливого програмного забезпечення до відповідних сімей та для виявлення нових невідомих шкідливих програм. Для кластеризації нових сімей шкідливих програм вони використовували алгоритм спільного найближчого сусіда (SNN). Їх модель була оцінена на наборі даних, який складався з 21740 зразків зловмисних програм з 9 різних сімей, і вони повідомили про точність класифікації на рівні 98.9% та точність виявлення на рівні 86.7% [6].

Таким чином, загальний висновок з аналізу різних досліджень застосування машинного навчання у класифікації шкідливого програмного забезпечення полягає в тому, що цей підхід є дієвим, але потребує постійного вдосконалення та розширення методів і моделей. Дослідження показали, що комбінація статичних і динамічних ознак може покращити ефективність класифікації, а також що гібридні підходи із поєднанням різних видів ознак можуть дати кращі результати, ніж використання лише одного типу ознак. Багато досліджень також вказують на важливість постійного аналізу та адаптації моделей до нових видів шкідливих програм та їх змінних характеристик. Врахування таких факторів, як обфускація, метаморфізм та інші техніки, що використовуються для уникнення виявлення, також є важливими для успішного застосування машинного навчання у цій області. Загалом, дослідження підтверджують, що машинне навчання має великий потенціал у виявленні та боротьбі з шкідливим програмним забезпеченням, але потребує подальшого розвитку і вдосконалення, включаючи застосування нових

методів, удосконалення алгоритмів та постійне оновлення даних та моделей.

ВИСНОВКИ ДО РОЗДІЛУ 2

У цьому розділі розглядалися теоретичні основи машинного навчання та нейронних мереж, включаючи сучасні архітектури. Було розглянуто застосування методів машинного навчання для класифікації зловмисного програмного забезпечення за сімействами, а також виявлення шкідливого ПЗ серед доброякісного. Показано, як машинне навчання сприяє у різних підходах до аналізу, включаючи статичний, динамічний та гібридний. Проте існуючі статичні та динамічні методи виявлення зловмисного програмного забезпечення, що застосовуються на практиці, виявляються недостатньо ефективними проти нових та мінімально змінених старих образців шкідливого коду. Динамічний аналіз нових, невідомих раніше шкідливих програм потребує значних обчислювальних ресурсів або часу, або має низьку ефективність через обмеженість перевірки лише одного шляху виконання коду. Статичний аналіз на основі підписів виявляється неефективним навіть проти мінімально модифікованих шкідливих програм.

Статичні методи аналізу, які використовують машинне навчання, можуть бути більш перспективними, оскільки вони здатні знаходити структурну близькість та працювати ефективно навіть для абсолютно нових одиниць шкідливого коду. Проте їх використання може бути обмеженим через високу складність та повільність роботи.

РОЗДІЛ 3. МЕТОД НАВЧАННЯ МОДЕЛІ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз методів навчання моделі виявлення та захисту від шкідливого програмного забезпечення

Правильна початкова настройка ваг моделі важлива для її успішного навчання. Параметр s можна розглядати як аналог ваги окремого ядра згорткової мережі, який «масштабує» трансформований вектор. Додатковий параметр θ визначає кут повороту цього вектора. При об'єднанні трансформованих векторів результативна норма залежить від значення параметра θ, хоча проекція на ось чорно-білого кольору залишається сталою. Тому належне початкове значення для ваг і кута повороту можна обрати з нормального розподілу, щоб забезпечити однакову дисперсію градієнтів протягом навчання. Таким чином, для кожного шару k кут повороту θ та вага s ініціалізуються випадковими значеннями з нормального розподілу [11]:

$$S_k = U\left[-\frac{\sqrt{6}}{\sqrt{n_k+n_{k+1}}}.\frac{\sqrt{6}}{\sqrt{n_k+n_{k+1}}}\right], \theta = U[-\frac{\pi}{2},\frac{\pi}{2}] \qquad (3/1)$$

де ⋃ позначає нормальний розподіл та nk — розмірність k-ї вхідної матриць кватерніонів.

Початкова ініціалізація ваг моделі та параметрів грає важливу роль у процесі навчання. Це може впливати на швидкість збіжності моделі та якість її результатів. Параметри моделі, такі як s і θ, що відповідають за масштабування та поворот трансформованих векторів, можуть бути ініціалізовані з нормального розподілу, щоб забезпечити сталу дисперсію градієнтів під час навчання. Цей підхід може бути особливо корисним для моделей, де точність та стабільність градієнтів важливі для досягнення оптимальних результатів. Отже, належна початкова ініціалізація параметрів може покращити ефективність навчання моделі та якість її

прогнозів.

Градієнтний спуск є основним методом навчання нейронних мереж, що ґрунтується на застосуванні ланцюгового правила диференціювання складних функцій. Цей процес призводить до оновлення параметрів моделі, яка навчається. Для навчання кватерніонних згорткових мереж, де $\hat{p} = rp + xp \cdot i + yp \cdot j + zp \cdot k$ та $\hat{q} = rq + xq \cdot i + yq \cdot j + zq \cdot k$ представляють собою кватерніонні числа, можна застосувати аналогічний підхід до градієнту кватерніону, використовуючи операцію повороту у вигляді множення матриць [11]:

$$\frac{\partial L}{\partial q} = \frac{\partial L \partial p}{\partial p \partial q}, \frac{\partial L}{\partial \theta} = \frac{\partial L \partial p}{\partial p \partial \theta}, \frac{\partial L}{\partial S} = \frac{\partial L \partial p}{\partial p \partial s}, \tag{3.2}$$

де $q = [q1\ q2\ q3]$ та $p = [p1\ p2\ p3]$ вектори, що відповідають $\hat{p}$та $\hat{q}$. У випадку коли q та p довільні елементи карт ознак $\hat{a}nn'$ та ядер $\hat{w}ll'$ часткові похідні градієнта виглядають наступним чином:

$$\frac{\partial p}{\partial q} = S \begin{bmatrix} f1 & f2 & f3 \\ f3 & f1 & f2 \\ f2 & f3 & f1 \end{bmatrix},$$

$$\frac{\partial p}{\partial \theta} = S \begin{bmatrix} f'1 & f'2 & f'3 \\ f'3 & f'1 & f'2 \\ f'2 & f'3 & f'1 \end{bmatrix} \begin{matrix} q_1 \\ , [q_2] \\ q_3 \end{matrix}$$

$$\frac{\partial p}{\partial \theta} = S \begin{bmatrix} f1 & f2 & f3 \\ f3 & f1 & f2 \\ f2 & f3 & f1 \end{bmatrix} \begin{matrix} q_1 \\ , [q_2] \\ q_3 \end{matrix}$$

де fi визначені як (8), матриця fi' має той самий вид що і вираз (7), змінюється лише порядок множення матриць, тобто градієнтний спуск можна інтерпретувати як поворот відносно осі на обернений кут θ.

Функції активації та втрат повинні бути диференційовані, щоб мати змогу обчислювати градієнти та передавати їх по мережі. У випадку кватерніонних мереж будь-яка функція, яка є диференційованою та застосовується до кожного компоненту кватерніону, дозволяє використовувати ланцюгове правило та може використовуватися як функція активації або втрат. Вибір конкретної функції залежить від завдань, які вирішуються за допомогою мережі. Наприклад, для класифікації об'єктів, де останнім шаром є повнозв'язний з дійсними значеннями,

та для перетворення вихідних кватерніонів на одновимірний вектор, функція втрат може бути сформована у вигляді категорійної крос-ентропії. У задачах регресії, коли виходи кватерніонів слід перевести у триканальні зображення, можуть використовуватися функції активації, такі як середньоквадратична помилка або подібні [25].

Отже, вибір функцій втрат та активацій грає критичну роль у навчанні нейронних мереж, зокрема кватерніонних мереж.

Диференційованість: функції втрат та активації повинні бути диференційовані, щоб мати можливість обчислити градієнти та використовувати метод градієнтного спуску для оптимізації параметрів мережі.

Вплив на результати навчання: вибір конкретних функцій впливає на результати навчання мережі. Наприклад, використання різних функцій втрат може підходити для різних типів завдань, таких як класифікація або регресія.

Залежність від завдання: вибір функцій також залежить від конкретного завдання, яке вирішується за допомогою мережі. Наприклад, для класифікації об'єктів може бути використана категорійна крос-ентропія, а для регресії – середньоквадратична помилка або подібні функції.

Отже, важливо вибрати функції втрат та активацій, які оптимізують результати для конкретного завдання та забезпечують ефективне навчання мережі.

3.2 Реалізація алгоритмів виявлення шкідливого програмного забезпечення за допомогою бібліотек TensorFlow та Keras

Файли зі шкідливим програмним забезпеченням можна перетворити в зображення, використовуючи восьмибітний цілочисельний вектор, який утворюється з їх бінарного вмісту. Кожне значення у цьому векторі відповідає яскравості пікселя у межах від 0 до 255. Отриманий вектор можна перетворити у двовимірну матрицю, представляючи чорно-біле зображення будь-яких розмірів. Наприклад, набір тестових даних, створений з оброблених таким чином файлів зі

шкідливим ПЗ, використовувався для навчання мережі. Цей набір містив 9339 зображень зразків шкідливого ПЗ, які поділені на 25 класів:

Adialer.C – це застарілий вірус, який розповсюджується через мережі Dial-up та автоматично здійснює дзвінки на випадкові номери, використовуючи інфікований комп'ютер.

Agent.FYI – це троянська програма, яка відключає антивірусні служби та завантажує програми backdoor для подальшого контролю над системою.

Allaple.A – це мережевий хробак, який поширюється шляхом численного копіювання в файловій системі, кожна копія якого є зашифрованою та відрізняється від інших. Він використовується для проведення DDoS-атак.

Allaple.L – це видозмінена версія мережевого хробака Allaple.A.

Alureon.gen!J – це троянська програма, яка здійснює зміну налаштувань DNS таблиць на мережевому обладнанні, такому як маршрутизатори та комутатори.

Autorun.K – це хробак, який поширюється шляхом копіювання на флеш-накопичувачі або інші переносні носії інформації.

C2LOP.gen!g – це троянська програма для зміни налаштувань браузера та відображення рекламних матеріалів.

C2LOP.P – це видозмінена версія троянської програми C2LOP.gen!g.

Dialplatform.B – це троянська програма, створена для мережі Dial-up, яка здійснює виклики на певні привілейовані номери.

Dontovo.A – це троянська програма для завантаження шкідливих файлів.

Fakerean – це вірус, який вбудовується у код антивірусних програм і змінює свою поведінку в залежності від операційної системи.

Instantaccess – це троянська програма, яка незаконно демонструє відео на інфікованому комп'ютері.

Lolyda.AA1, Lolyda.AA2, Lolyda.AA3 та Lolyda.AT – це різновиди троянських програм з сімейства Lolyda, які викрадають інформацію про ігрові аккаунти, а також логіни та паролі цих аккаунтів.

Malex.gen!J – це троянська програма, яка дозволяє здійснювати контроль над комп'ютером без відома користувача.

Obfuscator.AD – це троянська програма.

Rbot!gen – це backdoor для віддаленого керування інфікованим комп'ютером.

Skintrim.N – це троянська програма для завантаження шкідливого програмного забезпечення та його оновлень з певних веб-ресурсів.

Swizzor.gen!E та Swizzor,gen!I – це троянські програми для завантаження додаткового програмного забезпечення з веб-ресурсів.

VB.AT – це вірус для зараження окремих програм, написаних на мові VisualBasic.

Wintrim.BX – це троянська програма для завантаження додаткового шкідливого програмного забезпечення.

Yuner.A – це мережевий хробак для платформи Windows.

TensorFlow – це відома бібліотека для глибокого навчання, розроблена компанією Google і використовується для побудови та тренування нейронних мереж для різних задач. Основними компонентами нейронних мереж є три елементи: вхідний шар, приховані шари та вихідний шар. Вхідний та вихідний шари відповідають початковій і кінцевій частинам мережі відповідно. Вхідний шар містить початкові дані, які використовуються для розпізнавання, регресії тощо, а вихідний шар містить значення, отримані після обробки вхідної інформації прихованими шарами мережі [22].

Зазвичай в мережах глибокого навчання є велика кількість прихованих шарів між вхідним та вихідним. Ці шари містять ваги і виконують обчислення з використанням вхідних даних до отримання значень для вихідного шару. У прихованих шарах можна змінювати деякі параметри, такі як кількість нейронів, функції активації та параметри регуляризації. Кожен шар нейронної мережі виконує певні обчислення з використанням даних, що надходять до нього з попереднього шару, і обчислені значення слугують входами для наступного шару. Такий процес описується через тензори та операції у бібліотеці TensorFlow, які є базовими для опису нейронних мереж з її використанням.

Тензори представляють собою багатовимірні масиви, які містять дані конкретного типу (наприклад, цілі числа або числа з рухомою комою). У всій

нейромережі числові дані подаються у формі тензорів. Кожен тензор може мати різну кількість вимірів, але всередині мережі він має фіксований розмір. У деяких випадках тензор може бути порожнім і очікувати введення даних ззовні; таким тензором є вхідний тензор, який приймає початкові дані для подальшого навчання прихованих шарів нейромережі. Операції визначаються як певні обчислення, які дають певне значення. Отже, операції приймають на вхід тензори, виконують обчислення над ними і повертають результат у вигляді тензора. Таким чином, операції зв'язують всі тензори нейромережі у вигляді конвеєра обчислень від вхідного тензора до тензора вихідного шару [22].

У нейромережі, побудованій за допомогою бібліотеки TensorFlow, кожен шар містить тензор значень нейронів та тензор ваг, а операція обчислення значень нейронів приймає ці два тензори на вхід і повертає тензор значень нейронів для наступного шару. Операція включає в себе множення матриць ваг і значень нейронів та застосування функції активації до отриманого результату. Таким чином, тензор поточного шару є результатом операції попереднього шару і слугує входом для операції наступного шару. Цей «потік» тензорів через операції від вхідного шару мережі до вихідного шару виправдовує назву бібліотеки TensorFlow.

Бібліотека TensorFlow дозволяє представляти елементи мережі у вигляді тензорів та виконувати операції над ними, а бібліотека Keras виступає абстракцією над TensorFlow, що дозволяє побудувати нейронну мережу, використовуючи окремі шари. Кожен шар містить декілька об’єктів: вхідний тензор зі значеннями нейронів, тензор зі значеннями ваг зв’язків, внутрішню операцію, що обчислює результуючий тензор, який стає вхідним для наступного шару. При побудові нейронної мережі з використанням Keras необхідно описати окремі шари, а також зв’язки між ними. Для всіх шарів вхідною інформацією є вихід попереднього шару, за винятком вхідного. Keras надає дві моделі для побудови мереж: послідовну та функціональну. Послідовна модель побудована послідовно, з однією послідовністю від вхідного до вихідного шару, де окремі шари мають лише один вхід. Це можна зробити за допомогою методу add(), що робить процес побудови досить простим (рис. 3.1).

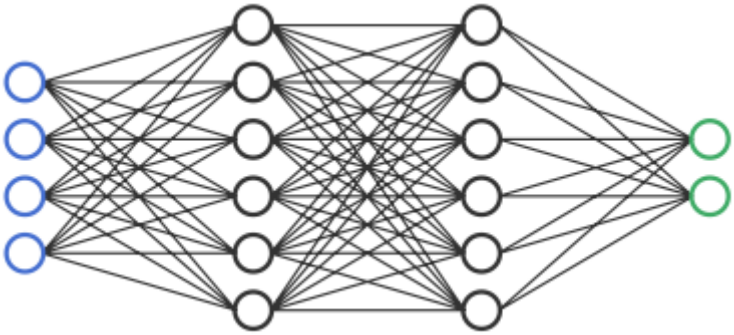

Рис. 3.1 – Приклад архітектури мережі для реалізації [22]

Для складніших моделей, де може бути кілька вхідних або вихідних шарів, або декілька зв’язків між шарами, доцільніше використовувати функціональний метод побудови мережі. У цьому випадку необхідно явно вказати зв’язок між окремими шарами мережі, і може бути кілька таких зв’язків.

Після побудови моделі за допомогою одного з вищезазначених методів необхідно провести компіляцію моделі для подальшого навчання. Процес компіляції включає в себе додавання до моделі кількох параметрів: функції втрат для вихідного шару, методу оптимізації для майбутньої нейронної мережі та метрик для оцінки якості моделі. На рис. 3.2. наведено приклад компіляції моделі з використанням логістичної функції втрат, методу оптимізації RMSProp та метрики якості – точності (відношення правильно спрогнозованих тестових об'єктів до всіх існуючих). У випадку, якщо модель має декілька вихідних шарів та потребує різних функцій втрат, слід вказати список функцій у правильному порядку, в якому розташовані вихідні шари.

```
model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])
```

Рис. 3.2 – Приклад компіляції моделі

Для створення нейронної мережі використовувалась послідовна модель побудови з бібліотеки Keras. Структура мережі включає наступні шари:

1. Початковий кватерніонний згортковий шар QConv2D із 32 фільтрами, розмірністю ядра 3 × 3, вхідною розмірністю зображення 64 × 64 × 3 та функцією активації ReLU.

2. Кватерніонний згортковий шар QConv2D із 32 фільтрами, розмірністю

ядра 3 × 3 та функцією активації ReLU.

3. Шар максимізаційного агрегування із розмірністю ядра 2 × 2.

4. Регуляризація виключенням (dropout) для 50% нейронів попереднього шару.

5. Два послідовні кватерніонні згорткові шари QConv2D із 64 фільтрами, розмірністю ядра 3 × 3 та функцією активації ReLU.

6. Ще один шар максимізаційного агрегування із розмірністю ядра 2 × 2.

7. Ще одна регуляризація виключенням для 50% нейронів попереднього шару.

8. Шар зменшення розмірності зображення до одновимірного вектору (Flatten).

9. Повнозв’язний шар QDense із 512 нейронами та функцією активації ReLU.

10. Регуляризація виключенням для 50% нейронів попереднього шару.

11. Вихідний повнозв’язний шар QDense з кількістю нейронів, що дорівнює кількості класів для розпізнавання, та функцією активації softmax, яка використовується для логістичної функції втрат.

Використовуючи тестову вибірку даних, що складалася з 9339 зразків шкідливого програмного забезпечення, як вхідних даних для навчання нейронної мережі з кватерніонними шарами, точність на навчальній вибірці даних змінювалась від 0.4465 до максимального значення 0.9419 на 25-й епосі навчання (рис. 3.3).

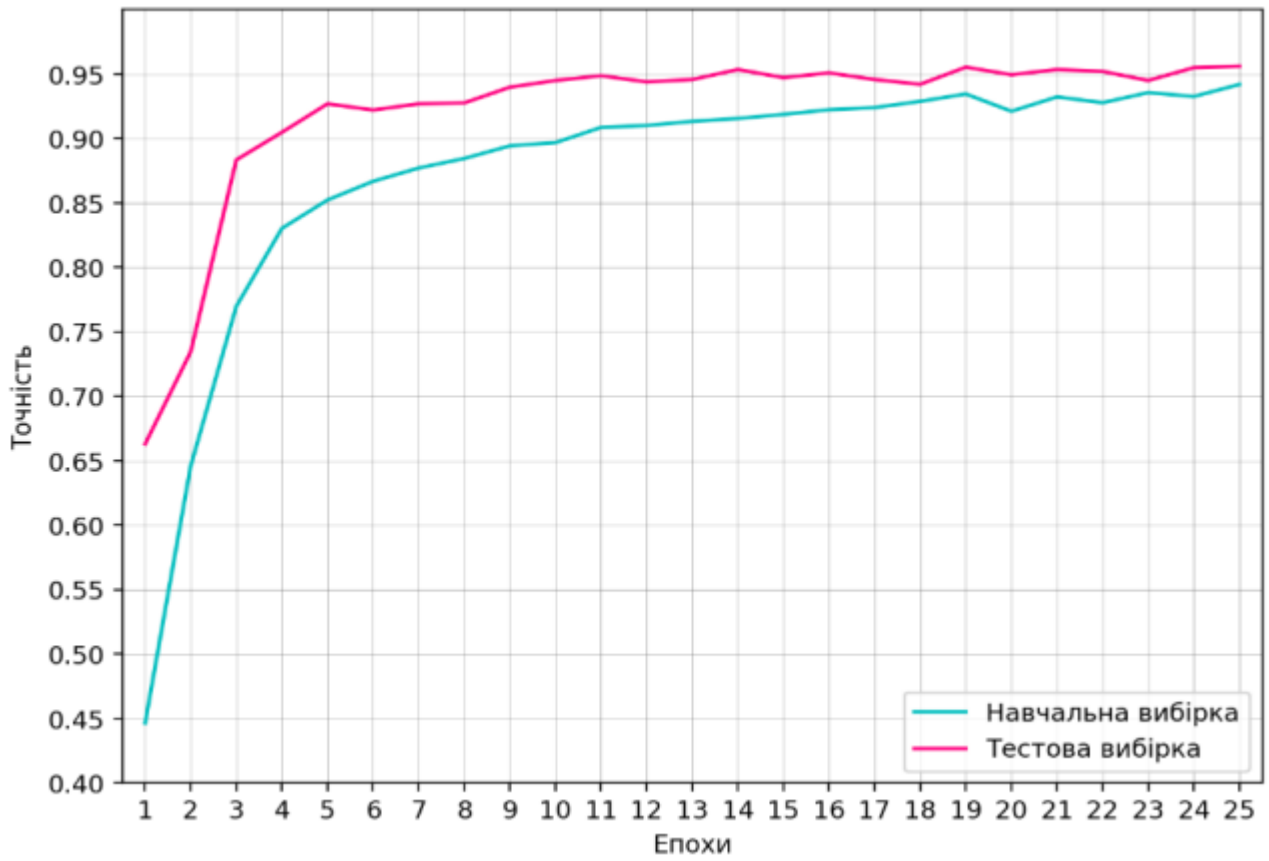

Рис. 3.3 – Залежність точності розпізнавання моделі від епох навчання

Для тестової вибірки, мінімальне та максимальне значення точності розпізнавання становили відповідно 0.6630 та 0.9561. Отримані значення свідчать про високу точність моделі розпізнавання. Значення функцій втрат для навчальної вибірки коливалися від максимального значення 1.8802 до мінімального 0.1719. Для тестової вибірки, максимальне та мінімальне значення функцій втрат складали відповідно 1.0375 та 0.1445.

Отже, з використанням бібліотек TensorFlow та Keras можна ефективно реалізувати алгоритми виявлення шкідливого програмного забезпечення. Ці бібліотеки надають зручний та потужний інструментарій для побудови та навчання нейронних мереж, які можуть розпізнавати шкідливе програмне забезпечення на основі навчальних даних. Використання послідовної моделі побудови та різноманітних шарів, таких як кватерніонні згорткові шари та регуляризація виключенням, дозволяє створювати потужні та ефективні моделі для виявлення шкідливого програмного забезпечення. Результати показують високу точність

розпізнавання та низькі значення функцій втрат, що свідчить про успішність побудованих моделей. Таким чином, використання бібліотек TensorFlow та Keras є ефективним підходом до реалізації алгоритмів виявлення шкідливого програмного забезпечення.

ВИСНОВКИ ДО РОЗДІЛУ 3

Ураховуючи прогрес у галузі інтелектуальних систем для розпізнавання зображень, можна використовувати згорткові нейронні мережі з кватерніонними елементами. Під час дослідження екстракції ознак був обраний метод, який перетворює бінарні файли програмного забезпечення у чорно-білі зображення, які використовуються як вхідні дані для згорткових нейронних мереж. Для реалізації алгоритмів використовувалися бібліотеки для машинного навчання, такі як TensorFlow та Keras. За допомогою послідовної моделі побудовано нейронну мережу, яка включала кватерніонні згорткові та повнозв’язні шари, а також шари максимізації та зменшення розмірності даних, з використанням виключення для регуляризації. Створена вибірка даних містила 9339 зразків зображень шкідливого програмного забезпечення, розділених на 25 класів розпізнавання. Процес тренування тривав 25 епох. В результаті отримана мережа мала точність 0.9561 на тестових даних, що складалися з 3113 зразків. Таким чином, мета досягнута, адже оптимальні показники точності виявлення та класифікації шкідливих зразків досліджуваного програмного забезпечення досягнуті. Тому побудова інтелектуальної системи для розпізнавання шкідливого програмного забезпечення може вважатись успішною.

ВИСНОВКИ

По закінченні дослідження відмітимо досягнення поставленої мети та вирішення завдань. Нами було проведене дослідження та розробка методів виявлення та захисту від шкідливого програмного забезпечення на основі машинного навчання.

Шкідливе програмне забезпечення – це зловмисна програма або код, що завдає шкоди кінцевим пристроям. У разі зараження пристрою шкідливим програмним забезпеченням (ПЗ) можливі несанкціонований доступ, пошкодження даних або блокування пристрою до сплати викупу. Шкідливе програмне забезпечення є однією з найпоширеніших форм кібератак. Воно може здійснювати декілька видів шкідливої діяльності: забороняти доступ до мережі, красти інформацію з жорсткого диска, порушувати роботу системи або виводити її з ладу.

Різні методи аналізу, такі як аналіз на основі сигнатур, поведінковий аналіз та евристичні методи, мають свої переваги та обмеження. Методи аналізу на основі сигнатур дозволяють ефективно виявляти відомі шкідливі програми, але потребують постійного оновлення баз даних для виявлення нових загроз. Поведінковий аналіз, з іншого боку, може виявляти невідомі шкідливі програми за їх характеристиками виконання, але може стикатися з великою кількістю хибних спрацювань та вимагати значних обчислювальних ресурсів. Евристичні методи можуть допомагати виявляти нові типи загроз, але також мають свої обмеження. Усі ці методи можуть бути використані окремо або в поєднанні для забезпечення більш ефективного виявлення та захисту від шкідливого програмного забезпечення.

У другому розділі розглядалися теоретичні основи машинного навчання та нейронних мереж, включаючи сучасні архітектури. Було розглянуто застосування методів машинного навчання для класифікації зловмисного програмного забезпечення за сімействами, а також виявлення шкідливого ПЗ серед доброякісного. Показано, як машинне навчання сприяє у різних підходах до

аналізу, включаючи статичний, динамічний та гібридний. Проте існуючі статичні та динамічні методи виявлення зловмисного програмного забезпечення, що застосовуються на практиці, виявляються недостатньо ефективними проти нових та мінімально змінених старих образців шкідливого коду. Динамічний аналіз нових, невідомих раніше шкідливих програм потребує значних обчислювальних ресурсів або часу, або має низьку ефективність через обмеженість перевірки лише одного шляху виконання коду. Статичний аналіз на основі підписів виявляється неефективним навіть проти мінімально модифікованих шкідливих програм.

Статичні методи аналізу, які використовують машинне навчання, можуть бути більш перспективними, оскільки вони здатні знаходити структурну близькість та працювати ефективно навіть для абсолютно нових одиниць шкідливого коду. Проте їх використання може бути обмеженим через високу складність та повільність роботи.

Ураховуючи прогрес у галузі інтелектуальних систем для розпізнавання зображень, можна використовувати згорткові нейронні мережі з кватерніонними елементами. Під час дослідження екстракції ознак був обраний метод, який перетворює бінарні файли програмного забезпечення у чорно-білі зображення, які використовуються як вхідні дані для згорткових нейронних мереж.

У третьому розділі дослідження для реалізації алгоритмів використовувалися бібліотеки для машинного навчання, такі як TensorFlow та Keras. За допомогою послідовної моделі побудовано нейронну мережу, яка включала кватерніонні згорткові та повнозв’язні шари, а також шари максимізації та зменшення розмірності даних, з використанням виключення для регуляризації. Створена вибірка даних містила 9339 зразків зображень шкідливого програмного забезпечення, розділених на 25 класів розпізнавання. Процес тренування тривав 25 епох. В результаті отримана мережа мала точність 0.9561 на тестових даних, що складалися з 3113 зразків.

Таким чином, мета досягнута, адже оптимальні показники точності виявлення та класифікації шкідливих зразків досліджуваного програмного забезпечення досягнуті. Тому побудова інтелектуальної системи для розпізнавання

шкідливого програмного забезпечення може вважатись успішною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anderson, B.and Quist, D., Neil, J., Storlie, C., and Lane, T. (2011). Graph Based Malware Detection Using Dynamic Analysis. In Journal in Computer Virology. URL: https://link.springer.com/article/10.1007/s11416-011-0152-x
2. BooJoong Kang, Suleiman Y. Yerima, K. M. S. S. (2016). N-opcode Analysis for Android Malware Classification and Categorization. URL: https://arxiv.org/pdf/1607.08149
3. Islam, R., Tian, R., Batten, L. M., and Versteeg, S. (2013). Classification of malware based on integrated static and dynamic features. In Journal of Network and Computer Applications. URL: https://www.sciencedirect.com/science/article/pii/S1084804512002214
4. Kolter, J. and Maloof, M. (2004). Learning to detect malicious executables in the wild. In In Proc. of the 10th ACM Int. Conf. on Knowledge Discovery and Data Mining. URL: https://www.jmlr.org/papers/volume7/kolter06a/kolter06a.pdf
5. Liu, L., Bao-sheng WANG, YU, B., and Qiu-xi ZHONG (2016). Automatic Malware Classification and New Malware Detection using Machine Learning. In Frontiers of Information Technology and Electronic Engineering. URL: https://link.springer.com/article/10.1631/FITEE.1601325
6. Makandar, A. and Patrot, A. (2015). Malware Analysis and Classification using Artificial Neural Network. In Trends in Automation Communications and Computing Technology. URL: https://ieeexplore.ieee.org/document/7492653
7. Muhammet Sahin and Serif Bahtiyar. 2021. A Survey on Malware Detection with Deep Learning. In 13th International Conference on Security of Information and Networks (SIN 2020). *Association for Computing Machinery, New York*, NY, USA, Article 34, 1–6. https://doi.org/10.1145/3433174.34336093.
8. Rafiqul I., Ronghua T., Batten L., Versteeg S. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications.* 2013. №36. С. 646–656. URL:

https://www.researchgate.net/publication/257489303_Classification_of_malware_based_on_integrated_static_and_dynamic_features

9. Schultz, M. G., Eskin, E., and Zadok, F. (2001). Data Mining Methods for Detection of New Malicious Executables. In In Proc. of the 22nd IEEE Symposium on Security and Privacy. URL: https://www.scirp.org/reference/referencespapers?referenceid=2132759
10. Seunghyun Yoon, Jin-Hee Cho, Dong Seong Kim, Terrence J. Moore, Frederica FreeNelson, Hyuk Lim, «Attack Graph-Based Moving Target Defense in Software-Defined Networks», IEEE Transactions on Network and Service Management, 2020, Vol.: 17, Issue: 3, p. 1653 – 1668. URL: https://ieeexplore.ieee.org/document/9063635
11. Sung A., Xu J., Mukkamala, S. (2004). Static Analyzer of Vicious Executables (SAVE). In Proceedings of the 20th Annual Computer Security Applications. URL: https://www.researchgate.net/profile/Abdulmunem-Khudhair-2/post/Can-anyone-suggest-references-for-Common-Malware-Attacks-and-Network-attacks/attachment/59d62926c49f478072e9c16d/AS%3A272467216535552%401441972654193/download/79.pdf
12. Willems, C., Holz, T., and Freiling, F. (2007). Toward Automated Dynamic Malware Analysis Using Cwsandbox. In IEEE Security and Privacy. URL: https://ieeexplore.ieee.org/document/4140988
13. YARA Rules Guide: Learning this Malware Research Tool. URL: https://www.varonis.com/blog/yara-rules (дата звернення 27.05.2024)
14. Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Scholkopf, B. (2003). Learning with local and global consistency. In Advances in Neural Information Processing Systems 16: Proceedings of the 2003. URL: https://proceedings.neurips.cc/paper/2506-learning-with-local-and-global-consistency.pdf
15. Білан І.А. Особливості застосування шкідливого програмного забезпечення спецслужбами країни-агресора. *Інформація і право.* 2023. №2(45). С. 139-152. URL: http://il.ippi.org.ua/article/view/282333

16. Волков О.О. Поняття шкідливого програмного засобу, призначеного для несанкціонованого втручання в роботу електронно-обчислювальної техніки. *Науковий вісник Національної академії внутрішні справ.* 2018. № 1 (106) С. 217-231. URL: https://elar.naiau.kiev.ua/bitstreams/5483d92c-58df-466e-8170-460835d1f702/download
17. Волошко Д.С., Гамза Д.Є., Смолєв Є.С. Технологія виявлення простих вірусів у програмному коді. *Сучасний захист інформації.* 2023. №2. С. 41-49. URL: https://journals.dut.edu.ua/index.php/dataprotect/article/view/2765/2662
18. Діденко М.С. Евристичний аналіз як метод виявлення небажаного програмного забезпечення. Problems of Informatization: the eleventh international scientific and technical conference. URL: https://repository.kpi.kharkov.ua/ (дата звернення 27.05.2024)
19. Єгоров С.В., Шкварницька Т.Ю. Розширений метод аналізу шкідливого програмного забезпечення з метою створення сигнатур. Вісник Унівеситету «Україна». 2020. №1(24). С. 161-170. URL: https://visn-it.uu.edu.ua/old_site/files/2020/16.pdf
20. Жульковська І., Плужник А., Жульковський О. Сучасні методи виявлення шкідливих програм. *Математичне моделювання.* 2021. №1(44). С. 46-54. URL: http://matmod.dstu.dp.ua/article/view/235922
21. Лисенко С.М., Щука Р.В. Аналіз методів виявлення шкідливого програмного забезпечення в комп'ютерних системах. *Вісник Хмельницького національного університету.* 2020. №2(283). С. 101-107. URL: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/01/17-4.pdf
22. Новотарський О.Ю. Метод виявлення шкідливого програмного забезпечення. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/viewFile/3118/2508 (дата звернення 27.05.2024)
23. Педяш В., Ледовський Є., Ткач В. Сучасні методи виявлення шкідливого програмного забезпечення. *Вісник Хмельницького національного університету. Серія: Технічні науки.* 2024. №2. Том 333. С. 389-392. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/164/148

24. Порівняння найкращих інструментів машинного навчання: TensorFlow, Keras, Scikit-Learn, Pytorch. URL: https://www.zfort.com.ua/blog/porivnyannya-naikrashikh-instrumentiv-mashinnogo-navchannya-tensorflow-keras (дата звернення 28.05.2024).
25. Прочухан Д.В., Руденко О.Г. Комп'ютерні інтелектуальні системи та мережі. *Матеріали XIV Всеукраїнської науково практичної WEB конференції аспірантів, студентів та молодих вчених* (23-25 березня 2021 р.). Кривий Ріг: Криворізький національний університе-тет, 2021. С.104-105. URL: https://www.researchgate.net/publication/353902723_Perevagi_vikoristanna_biblioteki_Keras_pri_pobuduvanni_zgortkovih_nejronnih_merez_v_zadacah_klasifikacii_zobrazen
26. Савенко О. С. Критерії класифікації методів виявлення шкідливого програмного забезпечення. *Вісник Хмельницького національного університету.* 2018. № 1. С. 23-27. URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=Vchnu_tekh_2018_1_6
27. Семенов С.Г., Гавриленко С.Ю., Глоба С.М., Бабенко О.С. Розробка системи виявлення комп'ютерних вірусів на основі нейронної мережі АРТ-1. Системи обробки інформації. 2015. Випуск 10 (135). С. 126-129. URL: http://nbuv.gov.ua/UJRN/soi_2015_10_29
28. Субач І., Фесьоха В., Фесьоха Н. Фесьоха Н. О. Аналіз існуючих рішень запобігання вторгненням в інформаційно-телекомунікаційні мережі. *Information Technology and Security*. 2017. Т. 5. № 1. С. 29-41. URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=inftech_2017_5_1_6
29. Терейковський І.А. Нейромережева методологія розпізнавання інтернет-орієнтованого шкідливого програмного забезпечення. *Безпека інформації.* 2013.

Т. 19. № 1. С. 24-28. URL: http://nbuv.gov.ua/UJRN/bezin_2013_19_1_6

30. Що таке шкідливе програмне забезпечення? URL: https://www.microsoft.com (дата звернення 22.05.2024).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ