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

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____ 20__ р.

На правах рукопису
УДК 004.8

ДИПЛОМНА РОБОТА
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Нейромережевий модуль захисту інформаційних систем від генерації шкідливої інформації

Виконавець: Ю.В. Капустянська
Керівник: професор кафедри БІТ С.В. Казмірчук
Нормоконтролер: професор кафедри БІТ С.В. Казмірчук

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Капустянської Юлії Володимирівни

1. Тема: *Нейромережевий модуль захисту інформаційних систем від генерації шкідливої інформації* затверджена наказом ректора від «26» квітня 2021 р. № 652/ст.
2. Термін виконання: з 10.05.2021 р. по 20.06.2021 р.
3. Вихідні дані: проаналізувати існуючі методи захисту та методики аналізу інформаційних систем у галузі інформаційної безпеки; на основі аналізу виділити вхідні і вихідні параметри, завдяки яким можливо провести порівняння існуючих нейромережевих модулів, виявлення їх переваг і недоліків; розробити методику, алгоритм та нейромережевий модуль аналізу і захисту інформаційної системи.
4. Зміст пояснювальної записки: аналіз існуючих методів захисту та методик аналізу у галузі інформаційної безпеки; розробка методики аналізу та захисту інформаційних систем на основі нейронної мережі; розробка нейромережевого модуля інформаційної системи, верифікація отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН
виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	01.04.2021	*Виконано*
2.	Аналіз літературних джерел	2.04.2021 – 13.04.2021	*Виконано*
3.	Обґрунтування вибору рішення	13.04.2021 – 15.04.2021	*Виконано*
4.	Збір інформації	15.04.2021	*Виконано*
5.	Дослідження сучасних методик та нейромережевих модулів для захисту інформаційних систем	15.04.2021 – 21.04.2021	*Виконано*
6.	Розробка методики та структури нейромережевого модуля для захисту інформаційних систем від генерації шкідливої інформації	22.04.2021 – 30.04.2021	*Виконано*
7.	Розробка алгоритму для захисту інформаційних систем	01.05.2021 - 15.05.2021	*Виконано*
8.	Проведення тестування розробленого алгоритму	16.05.2021 – 19.05.2021	*Виконано*
9.	Перевірка на антиплагіат	04.06.2021	*Виконано*
10.	Оформлення і друк пояснювальної записки	30.05.2021- 02.06.2021	*Виконано*
11.	Оформлення презентації	03.06.2021	*Виконано*
12.	Отримання рецензій від рецензента	09.06.2021	*Виконано*

Здобувач вищої освіти Ю. Капустянська
(підпис, дата)

Керівник дипломної роботи С. Казмірчук
(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 50 сторінок основного тексту, 11 рисунків, 4 таблиці, 2 сторінки додатку. Список використаних джерел містить найменування і займає 6 сторінок. Загальний обсяг роботи 52 сторінки.

Метою роботи є розробка нейромережевого модуля для захисту інформаційних систем від генерації шкідливої інформації.

У роботі вирішено задачу побудови нейромережевого модуля захисту інформаційних систем від генерації шкідливої інформації.

У роботі розроблено алгоритм та нейромережевий модуль для захисту, аналізу та оцінки загроз інформаційним системам.

Розроблений метод та нейромережевий модуль відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності інформаційних систем.

Можливі напрямки розвитку цієї роботи пов’язані із розширенням моделі і алгоритму нейромережевого модуля відповідно до вимог міжнародних стандартів, для більш повного аналізу та оцінки ризиків.

Ключові слова: згорткова нейронна мережа. довга короткочасна пам'ять шкідливе програмне забезпечення, штучна нейронна мережа, нейромережевий застосунок, карусель постійних помилок.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ У СФЕРІ ЗАХИСТУ ІНФОРМАЦІЙНИХ СИСТЕМ ВІД ГЕНЕРАЦІЇ ШКІДЛИВОЇ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ НЕЙРОННОЇ МЕРЕЖІ 9
1.1. Поняття шкідливої інформації 9
1.2. Аналіз класичних методів виявлення шкідливої інформації 10
1.2.1. Статичний аналіз 10
1.2.2. Динамічний аналіз 11
1.2.3. Машинне навчання 12
РОЗДІЛ 2. ЗАХИСТ ІНФОРМАЦІЙНИХ СИСТЕМ ВІД ГЕНЕРАЦІЇ ШКІДЛИВОЇ ІНФОРМАЦІЇ ВИКОРИСТОВУЮЧИ НЕЙРОННУ МЕРЕЖУ .. 16
2.1. Опис запропонованого методу захисту від шкідливої інформації 16
2.2. Виявлення особливостей шкідливої інформації на основі зображень у відтінках сірого за допомогою ЗНМ 17
2.2.1. Генерування зображень у відтінках сірого 17
2.2.2. Нейронна мережа згортки 18
2.2.3. Попередня обробка даних для зображення у відтінках сірого 19
2.3. Витягування операційного коду за допомогою ДКЧП 20
2.3.1. Перетворення та аналіз послідовностей операційного коду 20
2.3.2. Навчання ДКЧП 21
2.3.3. Вибір послідовностей та злиття послідовностей 23
2.3.4. Стратегія збільшення даних на основі методу “розсувного вікна” 26
2.4. Стекове узагальнення 28
РОЗДІЛ 3. ЕКСПЕРИМЕНТИ ТА ОЦІНКИ 30
3.1. Впровадження системи 30
3.1.1. ЗНМ нейромережевого модуля 31
3.1.2. ДКЧП нейромережевого модуля 33
3.2. Результати тестування ЗНМ 34
3.3. Дослідження продуктивності ДКЧП 36
3.4. Результат стекового узагальнення 41

ВИСНОВКИ 43
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 44
Додаток А 50
Додаток Б 51

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ЗНМ - згорткова нейронна мережа.

ДКЧП - довга короткочасна пам'ять (різновид архітектури рекурентних нейронних мереж)

ШПЗ - шкідливе програмне забезпечення

ШНМ - штучна нейронна мережа

НМЗ - нейромережевий застосунок

КПП - карусель постійних помилок

ВСТУП

Актуальність. Незважаючи на активні дії зі сторони виробників антивірусного програмного забезпечення, комп'ютерні віруси продовжують успішно проникати в комп'ютерні системи користувачів по всьому світу і виконувати шкідливі дії зі знищення або крадіжки інформації. Інформаційні системи змінюються з кожним роком, змінюється як апаратна, так і програмна частина. Але й разом з цим змінюються і способи потрапляння комп’ютерних вірусів та й самі зловмисні програми підлягають модифікації. Традиційні методи виявлення шкідливих програм, що застосовуються сьогодні, не здатні забезпечити надійний захист комп'ютерних систем від проникнення комп'ютерних вірусів.

Методи штучного інтелекту дозволяють створити принципово нові алгоритми виявлення шкідливих програм, що дозволить значно підвищити рівень захищеності комп'ютерних систем. Нейронна мережа дозволяє автоматизувати процес детектування вірусів. А також добре підходить для задач класифікації, для аналізу коду шкідливих файлів. Це значно знижує ресурсозатратність, але ефективність детектування шкідливої інформації збільшується. Первагами такого методу є те, що нейронні мережі можуть давати хороші результати, при доволі низьких показниках помилок.

Метою дипломної роботи є розробка нейромережевого модуля для захисту інформаційних систем від генерації шкідливої інформації.

Для досягнення мети необхідно розв’язати наступні задачі:

- аналіз класичних методів захисту інформаційних систем;
- розробка нейромережевого модуля для захисту інформаційних систем;
- проведення експерименту для дослідження продуктивності та ефективності розробленого модуля.

Об’єкт дослідження: процес захисту інформаційних систем.

Предмет дослідження: методи та засоби класифікації шкідливої інформації, методи виявлення та протидії шкідливій інформації.

Практичне значення результатів полягає у тому, що розроблений нейромережевий модуль дозволяє класифікувати та виявляти шкідливу інформацію. Даний модуль може бути використаний у реальних практичних системах.

Результат класифікації було досягнуто за рахунок використання нейронних мереж ЗНМ та ДКЧП, а також алгоритму стекового узагальнення.

Оцінка сучасного стану проблеми на основі вітчизняної та зарубіжної літератури. На думку багатьох авторитетних досліджень, вже довгий час шкідливе програмне забезпечення є однією з найбільш небезпечних загроз сучасних комп'ютерних систем, як загального, так і спеціального призначення. Підтвердженням цього факту можуть служити відомі випадки кібератак, реалізованих з використанням різних шкідливих програм. Хоча розробкою відповідних антивірусних програм багато висококваліфікованих фахівців, але проблема ще не вирішена. Основні труднощі виникають у процесі детектування ШПЗ на стадії його проникнення в комп'ютерну систему. Перспективність вказаного напряму підтверджується вдалим застосуванням ШНМ як засобу розпізнавання комп'ютерних вірусів (антивірус з відкритим програмним кодом ClamAV, стартап Deep Instinct) та великою кількістю відповідних теоретико-практичних робіт. Разом з тим, недостатня точність розпізнавання та недостатня адаптованість до умов експлуатації, закритість використаних рішень значно обмежують сферу їх застосування. При цьому постійний прогрес в області теорії нейронних мереж вказує на можливість значного вдосконалення використаних нейромережевих засобів розпізнавання ШПЗ. Цим пояснюється актуальність досліджень в області вдосконалення існуючих НМЗ.

РОЗДІЛ 1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ У СФЕРІ ЗАХИСТУ ІНФОРМАЦІЙНИХ СИСТЕМ ВІД ГЕНЕРАЦІЇ ШКІДЛИВОЇ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ НЕЙРОННОЇ МЕРЕЖІ

1.1. Поняття шкідливої інформації

Поняття шкідливої інформації не має точних границь та визначень. Даний термін не має конкретного визначення у кібербезпеці. Однак це поняття чітко визначено та сформульовано в правовій сфері.

Шкідлива інформація - інформація, обіг якої заборонений через шкоду та небезпеку для суспільства та людини [51, с. 85]. Позитивно оцінивши таке визначення, можна вказати на такий недолік, як «заборона». Не вся інформація, яка має негативний вплив, заборонена, оскільки остання може мати місце в країні, що регулюється верховенством права, лише на підставі нормативних актів. Як показує реальне життя, законодавство часто відстає. Шкідлива інформація визначається як інформація, яка посягає на життєво важливі інтереси особистості, суспільства та держави, характер та ступінь суспільної небезпеки яких вимагають від держави вжиття таких заходів, які забезпечували б належний правовий захист від збитків, заподіяних цим суб'єктам, як результат його розподілу [52, с.7]. Останні два визначення є не зовсім добре розкритими. По-перше, інформація визначається словом "інформація". По-друге, посилання на "необхідність вжиття державою заходів для забезпечення належного правового захисту від шкоди, заподіяної цим суб'єктам в результаті її поширення", не розкриває значення досліджуваного терміна.

Шкідлива інформація ототожнюється з інформацією обмеженого використання - вона обмежена законом у доступі та / або використанні інформації, яка може мати негативний інформаційний вплив на громадян, організації, суспільство, державу та зашкодити їх правам, свободам та законним інтересам [53]. Однак запропоноване позначення не здатне чітко відображати суть досліджуваної концепції. А обмеження використання інформації можуть бути лише однією із захисних умов для обігу шкідливої інформації. Як видно з аналізу юридичної літератури, автори по-різному визначають поняття "шкідлива

інформація", що свідчить про відсутність єдиного підходу та поглядів. Крім того, існують різні назви одного і того ж поняття - шкідливий, негативний, небезпечний, неякісний, незаконний або інформація обмеженого використання.

У кібербезпеці існує поняття - зловмисне програмне забезпечення. Тобто програмне забезпечення, яке заважає роботі комп'ютера, збирає конфіденційну інформацію або отримує доступ до приватних даних комп'ютерних систем.

А поняття "інформації" - це абстрактне поняття і його значення залежить від вживаного контексту. Беручи до уваги, те що "шкідлива інформація" має бути визначена у сфері кібербезпеки або ж інформаційної безпеки. Отже, можна припустити, що поняття шкідливої інформації охоплює не лише поняття шкідливого програмного забезпечення, а й соціально - шкідливу інформацію.

Тому можна зробити висновок: термін «шкідлива інформація» за своїм значенням досить широке, тому воно містить у собі всю інформацію, яка шкодить суспільству та державі. Тому шкідливе програмне забезпечення, яке поширює не тільки комп'ютерні віруси, але й іншу чи шкідливу інформацію, також підпадає під це визначення.

1.2. Аналіз класичних методів виявлення шкідливої інформації

Виявлення шкідливої інформації завжди було проблемою досліджень в останні роки. Було запропоновано декілька методів та прийомів для протидії зростаючій кількості шкідливої інформації.

1.2.1. Статичний аналіз

Статичний аналіз використовується для виявлення зловмисного програмного забезпечення. До його методів входить: лексичний аналіз, синтаксичний аналіз, контроль потоку та методи аналізу потоку даних [8] - це все використовується для дослідження коду програми. Одним із поширених методів статичного аналізу для виявлення шкідливого програмного забезпечення є використання сигнатурного методу. За допомогою такого методу можна

встановити чи невідомий виконуваний файл містить віруси, шукаючи, чи відповідна сигнатура, тобто сукупність властивостей, відповідає виявленим характерним ідентифікуючим властивостям вірусу у базі. Цей метод виявлення [9] створює унікальний ідентифікатор - сигнатуру, для шкідливого програмного забезпечення на основі деяких конкретних розроблених вручну функцій. Однак методи підпису обмежені для виявлення невідомих шкідливих програм, оскільки невідоме шкідливе програмне забезпечення може містити нові функції, які є невиявленими та невнесеними до сигнатур. Крім того, ці сигнатури мають ряд виправлених шкідливих характеристик. Отже, якщо шкідливе програмне забезпечення проходить через якусь операцію шифрування або обфускації, воно отримує велику ймовірність уникнути виявлення сигнатурним методом.

Така ситуація сприяє розвитку динамічного аналізу. Мозер та ін. [8] дослідили недоліки методів статичного аналізу та виявили схему обфускації коду, яка ускладнює процедуру виявлення, спираючись виключно на статичний аналіз. Оскільки динамічний аналіз не сприйнятливий до перетворення обфускації коду, він є важливим доповненням до статичного аналізу.

1.2.2. Динамічний аналіз

Динамічний аналіз для виявлення зловмисного програмного забезпечення. Динамічний аналіз використовується шляхом запуску шкідливого програмного забезпечення у контрольованому середовищі (віртуальна машина, симулятор, емулятор, пісочниця тощо) для аналізу поведінки шкідливого коду (взаємодія з системою) [10]. Перед виконанням зразка шкідливого програмного забезпечення потрібно спочатку відкрити відповідні інструменти моніторингу, такі як Process Monitor, Capture BAT (для моніторингу файлової системи та реєстру), Process Explorer і Process Hacker Replace (для моніторингу процесу), Wireshark (для моніторингу мережі), та Regshot (для виявлення змін у системі).

В процесі динамічного аналізу результат виявлення шкідливого програмного забезпечення базується на інформації про поведінку (включаючи сліди системних викликів, доступ до мережі та модифікації файлів та пам'яті [5,

11]) та збору й аналізу даних з операційної системи (середовища виконання програми) у період виконання програмного застосунку. Ці методи широко досліджувались як рішення для виявлення шкідливого програмного забезпечення, але також було виявлено, що вони менш надійні при здійсненні аналізу великих наборів даних [12]. Оскільки важко змоделювати кожну ситуацію та передбачити поведінку зловмисного забезпечення, визначити ефективний час для контролю за діяльністю шкідливого програмного забезпечення та коли зупинити його. Отже, моделювання поведінки шкідливого програмного забезпечення потребує постійного моніторингу, що призводить до колосального витрат ресурсів комп'ютера, і це є доволі важким завданням у наш час, зважаючи на кількість шкідливої інформації. Однак сьогодні антивірусні програми щодня стикаються з потоком нових зразків шкідливого програмного забезпечення, тому автоматичний підхід необхідний, щоб забезпечити швидкість виявлення та заощадити витрати на ручний аналіз. Для виявлення шкідливих програм було запропоновано та використано різноманітні методи, засновані на машинному навчанні.

1.2.3. Машинне навчання

Машинне навчання - звіт методів в області штучного інтелекту, набір алгоритмів, які застосовуються, щоб створити нейромережу, яка вчиться на власному досвіді. В якості навчання машина обробляє величезні масиви вхідних даних і знаходить у них відповідні закономірності. Нейромережі є одним із видів машинного навчання, а не окремим інструментом. Нещодавно методи машинного навчання (наприклад, Support Vector Machines (SVM), дерева рішень (DT)) використовуються для виявлення та класифікації невідомих зразків із сімейства шкідливих програм через його масштабованість, швидкість та гнучкість. Шульц та співавт. [13] вперше запропонував застосувати метод інтелектуального аналізу даних для виявлення шкідливої інформації та використовував три різні типи статичних функцій, відповідно, заголовки PE файлів, послідовність рядків та послідовність байтів. Потім заснований на

правилах алгоритм (ruled based algorithm) - Ripper [14] застосовується для отримання даних DLL і використовує “наївний байєсівський класифікатор”, тобто ймовірнісний класифікатор, що працює за теоремою Баєса для визначення ймовірності елемента вибірки до приналежності до певних класів змінних, а також як алгоритм навчання для пошуку відповідних символів та особливостей шаблонів послідовностей байтів. Алгоритм приймає дані шкідливого коду, як вхідні та класифікує їх з точністю 97,11%. Колтер та Малуф [15] досягли кращого результату, використовуючи N - gram, тобто послідовність із n елементів, замість послідовності байтів, що не перетинаються. З їхнього висновку випливає, що найкраще рішення можна отримати, використовуючи дерево ухвалення рішень, за допомогою якого вираховуються очікувані значення паралельних альтернатив.

Натомість Сакс і Берлін [16] запропонували інший метод - відрізняти шкідливе програмне забезпечення від доброякісного із використанням нейронної мережі. У своїх дослідженнях гістограма ентропії обчислюється на основі двійкових даних та кількості викликів контекстних байтових даних, а також витягуються метадані файлів виконання та імпорту DLL. Ці чотири типи функцій перетворюються у 256 векторних вимірів один за одним. Невідомі зразки класифікуються за допомогою векторів ознак, які навчаються у чотирьохшаровій нейронній мережі. Їх результат становить: інстиннопозитивний рівень - 95,2%, тоді як істиннонегативний рівень - 0,1%. Тобто істиннопозитивний рівень - це процент шкідливих програм, які правильно визначені, як злоякісні, і відповідно істиннонегативний - це процент, коли нейронна мережа шкідливу программу, визначила як доброякісну.

Є кілька нових ідей щодо виявлення шкідливого програмного забезпечення. Натарадж та ін. [17] запропонував візуалізований підхід до класифікації шкідливих програм за допомогою обробки зображень. Зокрема, двійкові дані шкідливого програмного забезпечення перетворюються на зображення, яке складається тільки з сірих відтінків, і класифікація проводилася за моделлю kNN (метод k-найближчих сусідів) з розрахунком відстані Евкліда,

тобто класифікація об’єктів вираховується за розрахунком відстаней у рамках простору. Експерименти показують, що це швидкий метод виявлення шкідливого програмного забезпечення, але цей метод використовує глобальні функції для перетворення зображень, так що зловмисник може використати локальні перетворення для шкідливих програм. Тому у своїй подальшій роботі [18] вони порівняли два методи обробки зображення з динамічним аналізом. Експериментальні результати показують, що метод, заснований на використанні функцій для перетворення зображень, є ефективним та зручним і може мати точність, що відповідає результату динамічного аналізу. Вони також виявили, що цей вдосконалений метод може чудово справлятися як з упакованими(заархівованими), так і з нерозпакованими зразками шкідливих програм. Конг та Ян [19] запропонували метод для автоматичної класифікації шкідливих програм, заснований на вивченні “спонтанної” кластеризації, якщо нейромережа була навчена за типом “навчання без вчителя” за допомогою структурованої інформації (графіків викликів функцій). Після вилучення функції для графіку виклику функцій шкідливого програмного забезпечення, він обчислить схожість шкідливого програмного забезпечення за допомогою методу виявлення дискримінації(розрізнення) на основі матриці різниць методом кластерної вибірки з тим самим сімейством шкідливих програм. Після цього ці пари відстаней(різниць) шкідливого програмного забезпечення використовуються для класифікації за допомогою ансамблю методів навчання нейромереж. Сантос та ін. [20] запропонував метод із використанням функцій *N*-gram для відрізнення шкідливих програм від доброякісних програм. Під час досліджень невідоме зловмисне програмне забезпечення виявляється за *k*-найближчими зразками з найбільш подібними функціями *N*-грам. І є більше підходів з подібною ідеєю з використанням *N*-gram на основі байтів, кодів операцій або частоти викликів API для ідентифікації шкідливих програм [15, 21].

Більше того, оскільки важко точно та ефективно виявити шкідливе програмне забезпечення або за допомогою статичного аналізу, або ж за допомогою динамічного аналізу, деякі дослідження почали інтегрувати як

динамічні, так і статичні функції. Сантос та ін. [22] запропонував гібридний засіб виявлення зловмисного програмного забезпечення на основі алгоритмів машинного навчання, який називається OPEM, даний метод використовує набір функцій, отриманих в результаті статичного та динамічного аналізу шкідливого коду. Статичні функції отримуються шляхом отримання операційних кодів із виконуваних файлів, а динамічні - за допомогою моніторингу системних викликів, операцій та винятків. Максимальна точність виявлення шкідливого програмного забезпечення становить 96,60% за допомогою методу опорних векторів, тобто це метод аналізу даних для використання регресійного аналізу та класифікації за допомогою моделей з навчених за методом «навчання з учителем», також всі моделі мають бути навчені за пов'язаними алгоритмами. Експерименти довели, що гібридний метод має кращі показники порівняно з проведенням статичного та динамічного аналізу окремо.

Перевагами використання нейронної мережі є те, що вона може порівнювати файли для визначення відповідних шаблонів та структур, додавати їх до бази, для майбутнього детектування шкідливої інформації. Також нейромережа гарно підходить для моніторингу поведінки системи, а автоматичний підхід дозволить своєчасно попереджувати користувача про несанкціоновані дії у системі. Автоматизація значно знижує ресурсозатратність на ручний аналіз і як показали вже проведені досліди у даній сфері, даний метод є ефективнішим.

Вищезазначене виявлення шкідливої інформації при використанні нейронних мереж досягло досить хороших результатів; однак більшість із цих методів в значній мірі покладаються на знання експертів щодо проектування характеристик. У той же час, оскільки кількість зловмисного програмного забезпечення продовжує динамічно зростати, а розроблені людиною функції стикаються з багатьма проблемами, які вимагають значних витрат на оновлення функцій у відповідь на нові шкідливі програми.

РОЗДІЛ 2. ЗАХИСТ ІНФОРМАЦІЙНИХ СИСТЕМ ВІД ГЕНЕРАЦІЇ ШКІДЛИВОЇ ІНФОРМАЦІЇ ВИКОРИСТОВУЮЧИ НЕЙРОННУ МЕРЕЖУ

2.1. Опис запропонованого методу захисту від шкідливої інформації

У цьому розділі представлено запропонований метод виявлення шкідливої інформації, використовуючи мережі ЗНМ(згорткова нейронна мережа) та ДКЧП(довга короткочасна пам’ять). Для виявлення шкідливого програмного забезпечення запропонований програмний модуль насправді виконує завдання бінарної класифікації, отримуючи необроблені файлові дані як вхідні дані, і видає ймовірність розрізнення, вказуючи, наскільки ймовірно, що це шкідливе програмне забезпечення. Конкретно процес виявлення можна розділити на два етапи (див. рисунок 2.1). Першим етапом є попередня обробка зразків даних шкідливого програмного забезпечення, програма приймає виконуваний файл Windows у двійковому форматі, генерує з нього зображення, що складається тільки з сірих відтінків та витягує послідовність операційних кодів та функцію метаданих за допомогою інструменту декомпіляції. Отже, на цьому етапі формуються вхідні дані для подальших мереж ЗНМ та ДКЧП. На другому етапі застосовується основний процес, який приймає відповідно мережі ЗНМ та ДКЧП, навчаючись із зображення у відтінках сірого та послідовності коду операцій. Щоб оптимізувати ефективність виявлення, використовується один із ансамблів методів навчання нейромережі, а саме стекове узагальнення/стегування, щоб інтегрувати функції виводу та метаданих.

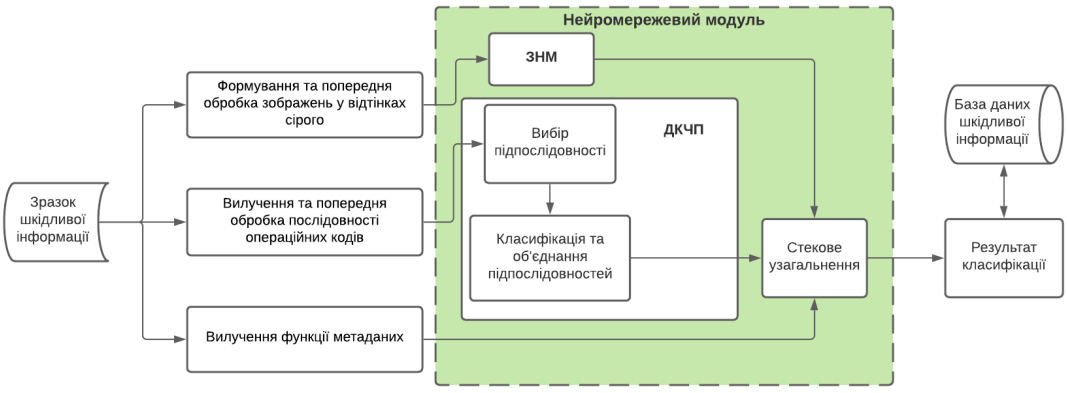

Рис. 2.1. Схема алгоритму виявлення шкідливої інформації

Програмний модуль розглядає три різні типи наборів функцій із вихідних даних; спочатку програмний модуль знаходить шкідливі особливості структури файлів із зображення у відтінках сірого від ЗНМ, а потім знаходить особливості шкідливого шаблону коду з послідовності коду операцій ДКЧП. Також додано деякі прості функції метаданих для опису загальної інформації. Конкретний дизайн нейромережевого модуля та процес виявлення описані в наступному розділі.

2.2. Виявлення особливостей шкідливої інформації на основі зображень у відтінках сірого за допомогою ЗНМ

У цьому підрозділі представлена мережа ЗНМ та описано, як виявити особливості структури шкідливого програмного забезпечення шляхом вивчення зображення, у відтінках сірого, шкідливого програмного забезпечення через ЗНМ. Даний метод візуалізує двійкові файли зловмисного програмного забезпечення як зображення у відтінках сірого, і ці зображення можуть чітко відображати структурні характеристики файлів шкідливих програм.

2.2.1. Генерування зображень у відтінках сірого

Для того, щоб створити зображення у відтінках сірого, необроблені дані потребують обробки та перетворення у формат зображення. Файл, що виконується є вхідними даними і розглядається як двійковий(бінарний) файл. Бінарний файл конвертується у шістнадцятковий файл, перетворюючи кожні 4 біти в шістнадцяткове число. Враховуючи, що діапазон шістнадцяткових чисел від 0 до 16, і ми поєднуємо кожні наступні два шістнадцяткові числа, що точно відповідають сірому значенню пікселя зображення на 256 рівнях. Тож необроблені дані можна перетворити на зображення у відтінках сірого за допомогою цього простого перетворення зображення. Весь набір послідовностей двійкового потоку сегментується по 8 бітів, що відповідає рівню сірого кожного пікселя, і послідовно розташовується для формування відповідного сірого зображення.

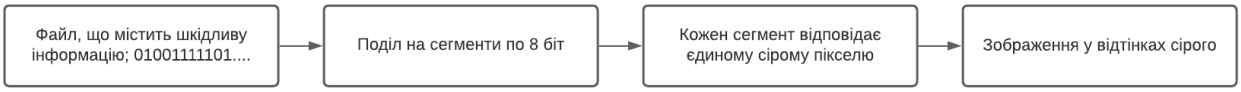

Рис. 2.2. Генерування зображень у відтінках сірого

Рис. 2.3. Генерування зображень у відтінках сірого з використання IDA Pro

Подібним методом можна скористатися для створення зображень у відтінках сірого з декомпільованого файлу з розширенням *.asm* , а потім також розглядати його як двійковий потік для перетворення у зображення. Однак було виявлено, що зображення у відтінках сірого, сформоване декомпільованим файлом, втратило багато структурних шаблонів (сформовані зображення у відтінках сірого видно на рисунку 2.3), різниця між двома методами помітна на зображеннях у відтінках сірого. Причина полягає в тому, що декомпільований файл має відносно фіксовану та організовану структуру. При використанні IDA Pro [23], який генерує декомпільований файл, початком кожного рядка якого є ім'я сегмента PE та адреса початкового файлу, а потім інструкція декомпіляції. Тому всі декомпільовані файли, як правило, мають схожі сформовані зображення у відтінках сірого. Таким чином, зображення у відтінках сірого, створене декомпільованим файлом, не підходить для подальшого процесу навчання.

2.2.2. Нейронна мережа згортки

Нейронну мережу згортки було обрано для навчання на зображеннях у відтінках сірого. Як типова глибока нейронна мережа, ЗНМ широко застосовується в області комп'ютерного зору та завдань, пов'язаних із зображеннями. Найбільш помітною характеристикою ЗНМ є те, що вона зменшує величезну кількість обчислень за допомогою ідеї розподілу ваг, рецептивного поля та пробної вибірки в просторі. Поділяється однакова вага між групами нейронів ЗНМ, де кожен згорткових нейрон обробляє дані лише для

свого рецептивного поля. ЗНМ приймає на вхід зображення у відтінках сірого та виводить результат класифікації або регресії із “end-to-end” структурою. А ваги нейронів ЗНМ навчаються за алгоритмом зворотного поширення помилки. Типова програма ЗНМ [24] використовується для розпізнавання рукописного вводу через безліч шарів згортки та об'єднання шарів для обробки вхідних даних. Кожен рівень згортки видає набір карт ознак, тоді як кожна карта ознаки представляє функції високого рівня, витягнуті за допомогою одного конкретного фільтра згортки. Агрегувальний шар в основному використовує принцип локальної кореляції для завершення зниження вибірки, так що наступний шар згортки може витягувати особливості з більш глобальної точки зору. Це значно зменшує кількість вагових параметрів і розрахунків для тренування глибинної мережі.

2.2.3. Попередня обробка даних для зображення у відтінках сірого

Для того, щоб задовольнити вимоги ЗНМ щодо вхідних даних, треба відредагувати зображення. Коли ЗНМ виконує таке завдання, як класифікація зображень, нейронна мережа отримує вхідні дані - зображення однакових розмірів. Як правило, дані зображення повинні мати однакову довжину та ширину (відношення довжини до ширини становить 1:1). Це для зручності у процесі подальшої операції згортки. Оскільки виконавчі файли мають різні розміри, різні розміри зображень у відтінках сірого також мають великі відмінності. Насправді велике зображення в сірих відтінках може досягати 1,04 МБ (2048 1036 пікселів), тоді як невеликий - лише 120 КБ (512 472 пікселів). Тому необхідно нормалізувати всі зображення у відтінках сірого.

Для цього використовується алгоритм білінійної інтерполяції, метод масштабування зображення для нормалізації. Він використовує значення чотирьох найближчих пікселів у вихідному зображенні для визначення значення віртуального пікселя цільового зображення, що забезпечує кращий ефект, ніж інтерполяція найближчого сусіда. Крім того, нормалізований розмір відтінку сірого - це гіперпараметр, який відображає різницю між точністю класифікації

та її прогнозом. Чим більший нормалізований розмір зображення, тим багатша інформація, що надходить на вхід ЗНМ; тоді із більш складною мережевою структурою буде отримано кращий результат виявлення, але такі параметри потребують більше часу на навчання мережі. З цією метою обрано 64 x 64 як нормалізований розмір відтінків сірого.

2.3. Витягування операційного коду за допомогою ДКЧП

У цьому розділі представлено ще одну важливу частину нейромережевого модуля, яка стосується послідовності операційних кодів із ДКЧП для вивчення шкідливих особливостей та шаблонів послідовностей. Послідовності операційного коду, тобто код операції(опкод) витягується з декомпільованих файлів. Ці послідовності фактично відображають логіку коду та логіку виконання програми виконавчих файлів. Отже, ДКЧП може знаходити у шкідливому коді особливості, що відповідають ознакам зловмисного програмного забезпечення.

2.3.1. Перетворення та аналіз послідовностей операційного коду

Щоб навчити нейромережу на послідовностях операційного коду, спочатку нам потрібно отримати послідовність опкоду з необроблених виконавчих файлів. Для цього декомпілюємо виконавчий файл через IDA Pro, який генерує декомпільований файл у форматі .asm. IDA Pro - це дебагер та інструмент декомпіляції, який аналізує шкідливий програмний код та перетворює бінарний код в асемблерний текст. Потім у файлі формату .asm кожне речення відділяється пробілом, щоб кожна фраза відповідала заздалегідь визначеному набору опкодів, що містить усі загальні асемблерні інструкції. Якщо збіг вдалий, код операції зберігається; в іншому випадку фраза видаляється. Під час цього процесу виявляється, що у декомпільованих файлах є велика кількість повторюваних послідовностей операційних кодів, таких як *dd, dd, ..., dd* або *db, db, db,..., db*. Тому потрібно відфільтрувати ці повторювані

підпослідовності, додавши деякі правила. Псевдокод алгоритму вилучення послідовності операцій наведено у додатку.

При вилученні послідовностей операційних кодів, розмір цілого вхідного набору опкодів впливає на середню довжину послідовностей операційних кодів. Чим більше і довше набір опкоду, тим більше видів операційних кодів може він в собі містити. Оскільки заважати може шум, тобто дані з великою кількістю додаткової, непотрібної інформації, а занадто довга послідовність операційних кодів спричиняє складні проблеми з навчанням ДКЧП, тому потрібно обмежити розмір набору операційних кодів до такого розміру діапазону, щоб він містив лише найдостовірнішу інформацію. Отже, усі декомпільовані файли .asm розглядаються, як текст, а інструкції - як словники. Потім обраховується частота події та за цією статистикою фільтруються низькочастотні словники. Після цього частота відповідної лексики є ознакою та далі цей частотний словник класифікується за моделлю “випадкового лісу”, випадкові ліси надають рейтинг за всіма наданими особливостями. Обираються частотні словники, які мають найбільшу унікальність. Нарешті, отримуємо набір опкодів, що включає 185 елементів, і витягуємо з цим послідовності опкод. Наразі ці послідовності операційних кодів слід оцифрувати, перш ніж подавати на вхід нейронної мережі; використовуємо кодування унітарного коду, яке просто приймає перетворення відображення, щоб отримати розріджений вектор - [0, 0, 0, 1, 0,..., 0], такий що N бітів двійкового представлення представляють N станів, що містять лише один ненульовий елемент. І кожен опкод отримує унікальне зображення унітарного коду.

2.3.2. Навчання ДКЧП

ДКЧП - довга короткочасна пам`ять[25], штучна нейронна мережа, яка широко використовується для обробки часових рядів, що є вдосконаленою моделлю на основі рекурентної нейронної мережі (РНМ). РНМ використовує внутрішню пам’ять для представлення попередніх вхідних значень, що дозволяє йому фіксувати тимчасову інформацію. Виходячи з цього, ДКЧП використовує

“карусель постійних помилок” (КПП) та добре спроектовані структури «вентилів», щоб полегшити проблему зникнення градієнта під час зворотного розповсюдження помилок. Отже, втрати можуть проявлятись назад через більш тривалий крок часу, що дозволяє ДКЧП дізнатися довгострокову залежність та контекст. Коротше кажучи, ДКЧП містить три вентилі (“вхідний вентиль”, “забувальний вентиль” та “виходовий вентиль”) для прийняття рішення та контролю стану КПП. Тут програмний модуль використовує мережу ДКЧП для вивчення послідовності кодів операцій для виявлення зловмисного програмного забезпечення. Однак одна існуюча проблема для ДКЧП полягає в тому, що важко ефективно тренуватися, коли вхідна послідовність занадто довга, хоча ДКЧП може фіксувати більше інформації за більший проміжок часу, ніж РНМ. У даній роботі, якщо розмір виконавчого файлу дуже великий, то довжина витягнутої послідовності операційного коду теж виходить дуже великою. Наприклад, середня довжина послідовності операційних кодів зразків сімейства шкідливих програм Ramnit сягає 36000. Але продуктивність ДКЧП швидко знижується, коли довжина вхідної послідовності перевищує 200.

Однією простою стратегією обробки дуже довгих послідовностей за допомогою мережі ДКЧП є укорочування та заповнення (УТЗ). Зокрема, УТЗ спочатку встановлює фіксовану довжину, скорочує та відкидає частину довгих послідовностей, що перевищує довжину, і додає короткі послідовності до довжини із заздалегідь визначеним ідентифікатором. Це зручно, але велика кількість інформації втрачається завдяки операції зрізання. Зрізане зворотне поширення через час (ЗЗПЧ) - ще одне рішення [26], яке додає обмеження часового вікна для обмеження максимальної відстані для операції зворотного розповсюдження помилок. Зрізане зворотне розповсюдження через час (ЗЗПЧ) є широко розповсюдженим методом вивчення повторюваних обчислювальних графіків. Зрізане ЗПЧ зберігає обчислювальні переваги зворотного розповсюдження через час, одночасно полегшуючи необхідність повного зворотного відстеження по всій послідовності даних на кожному кроці. Отже, поширення помилок та розрахунок градієнта виконуються лише на екрані, а ваги

вузлів за екраном не оновлюються. Це підвищує ефективність обчислень, жертвуючи невеликою частиною точності порівняно зі стандартним ЗПЧ (або повним ЗПЧ), оскільки стандартний розрахунок ЗПЧ менш ефективний, коли відстань зворотного розповсюдження занадто велика. Крім того, усічений ЗПЧ більше підходить для навчання без учителя, оскільки він може швидко адаптуватись до новоствореної частини дуже довгої послідовності. Загалом, усічений ЗПЧ є кращим та ефективним, оскільки він засвоює всю інформацію про послідовність порівняно зі стратегією ТАР. У даному нейромережевому модулі пропонується практична реалізація на основі усіченого алгоритму ЗПЧ для мережі ДКЧП. Оскільки градієнти розповсюджуються лише у вікні, спочатку ми ділимо послідовність операційних кодів на кілька підпослідовностей, де довжина кожної підпослідовності дорівнює довжині вікна усіченого ЗПЧ. Тоді для кожної послідовності ми просто робимо повний ЗПЧ, який дорівнює виконанню усіченого ЗПЧ для всієї послідовності без поділу екрану. Найголовніше, що це дозволяє ДКЧП тренуватися паралельно на декількох послідовностях. Однією з основних проблем ДКЧП є те, що періодична структура обмежує його для послідовного навчання серій послідовностей, що неефективно. Однак з цими послідовностями тренувальний процес ДКЧП може бути в 3 рази швидшим.

2.3.3. Вибір послідовностей та злиття послідовностей

За допомогою вищезазначеної стратегії навчання підпослідовності, підпослідовність подається на вхід мережі ДКЧП і вихідні дані формуються для кожної підпослідовності окремо. Це може розцінюватися як результат дискримінації через те, що ДКЧП розподілила, наскільки ймовірно, що ця послідовність є зловмисною. Однак підпослідовності можуть містити багато шуму, тобто непотрібної інформації, особливо у зразках шкідливих програм. Оскільки деякі автори зловмисного програмного забезпечення просто впровадили шкідливий фрагмент коду в доброякісне програмне забезпечення, тому багато послідовностей такого зразка можуть бути виявлені, як нешкідливі.

З цієї причини надзвичайно важливо подбати про ці послідовності у зловмисному програмному забезпеченні. Тут метод вибору підпослідовностей для очищення цих підпослідовностей та надання набору даних вищої якості для ДКЧП.

В даному програмному модулі мережа ДКЧП використовується без додаткової моделі для завершення вибору послідовностей. У завданні двійкової класифікації (для виявлення шкідливого програмного забезпечення) вихідний рівень мережі ДКЧП використовує логістичну регресію, що надає значення ймовірності для ідентифікації негативного класу або позитивного класу. Це схоже на недавню роботу [27] з використанням помилки реконструкції як основи для виявлення аномалії підпослідовності за допомогою кодера-декодера на основі ДКЧП. Зокрема, рівень логістичної регресії приймає вихідні дані прихованого рівня і обчислює ймовірність $p\,(\,y \mid x_j)$ поточної вибірки підпослідовності - x_j . y - представляє негативний клас (доброякісна мітка), де $y = 0$ і $y = 1$ представляють позитивний клас (мітка шкідливого програмного забезпечення), а фактично логістичний результат можна представити як $p\,(y = 1 \mid x_j)$. Чим вище чи нижче інтелект $p\,(y = 1 \mid x_j)$, чим більше впевненість ДКЧП для поточного зразка. Оскільки доброякісна послідовність у зловмисному програмному забезпеченні позбавлена шкідливих функцій, але все ще має шкідливу мітку, що може заплутати ДКЧП, впевненість, надана ДКЧП, є відносно нижчою. Отже, використовуючи $p\,(y = 1 \mid x_j)$, можна допомогти у виборі послідовностей.

Для того, щоб використовувати вихідні дані $p\,(y = 1 \mid x_j)$ нейронною мережею ДКЧП для вибору послідовності для x_j, встановлюється порогове значення δ і порівнюється з максимальною ймовірністю правдоподібності. Формула показана наступним чином:

$$\delta \geq max\{P(y = 1 \mid x_j), 1 - P(y = 1|x_j)\}. \quad (1)$$

Для поточної підпослідовності x_j, якщо виконується наведена вище формула, це означає, що ДКЧП встановлює низький рівень впевненості, що

поточна підпослідовність належить до будь-якої категорії, вказуючи, що підпослідовність x_j, не може надати достатньо достовірної інформації для оцінки ДКЧП або просто з неправильною міткою (доброякісна послідовність шкідливого програмного забезпечення) . Тож це можна розглядати як шумну послідовність, тобто таку, що містить непотрібну та зайву інформацію, і тому буде відфільтрована.

На початку навчального процесу ДКЧП, оскільки параметри ДКЧП ініціалізовані рандомно, вихідним даним ДКЧП не можна довіряти, тому використовується інше порогове значення η щоб визначити, коли мережа має достатню потужність, щоб розпочати вибір послідовності після декількох ітерацій. Зокрема, обчислюється похибка навчання на серіях вхідних даних. Якщо помилки в навчанні безперервних вхідних даних M нижчі за порогове значення η, тоді ДКЧП ініціює вибір послідовності, де M - гіперпараметр зі значенням 5. Усі ці гіперпараметри обираються в результаті експерименту. Більше того, оскільки помилка навчання завжди зменшуватиметься шляхом вибору підпослідовності, набір валідацій поділяється та використовує помилку валідації, щоб відслідкувати правильний момент аби зупинити вибір підпослідовності. Як тільки помилка валідації не може більше зменшуватися, то це вважається моментом, для закінчення вибору послідовності (блок-схему даного алгоритму надано у додатку).

На даний момент ДКЧП виводить результати класифікації для підпослідовностей; потрібно використати ці результати для завершення класифікації вихідної послідовності операційного коду. Можна використати простий синтез для його вирішення. Однією із поширених стратегій синтезу є метод голосування. У даному сценарії використано метод відносної більшості голосів, щоб запобігти результату дискримінації послідовностей.

$$H(x) = C_{argmax_j} \sum_{i=1}^{T} w_i \, h_i^j(x), \quad (2)$$

де w_i представляє значення результату дискримінації $h_i^j(x)$. Зазвичай $w_i \geq 0$ та $\sum_{i=1}^{T} w_i = 1$.

Враховуючи, що вихідний результат класифікації кожної підпослідовності є ненормованою ймовірністю, заданою з рівня логістичної регресії ДКЧП, тут можна безпосередньо використовувати ці ненормалізовані ймовірності w_i, як коефіцієнти для кожної послідовності (див. рисунок 2.4).

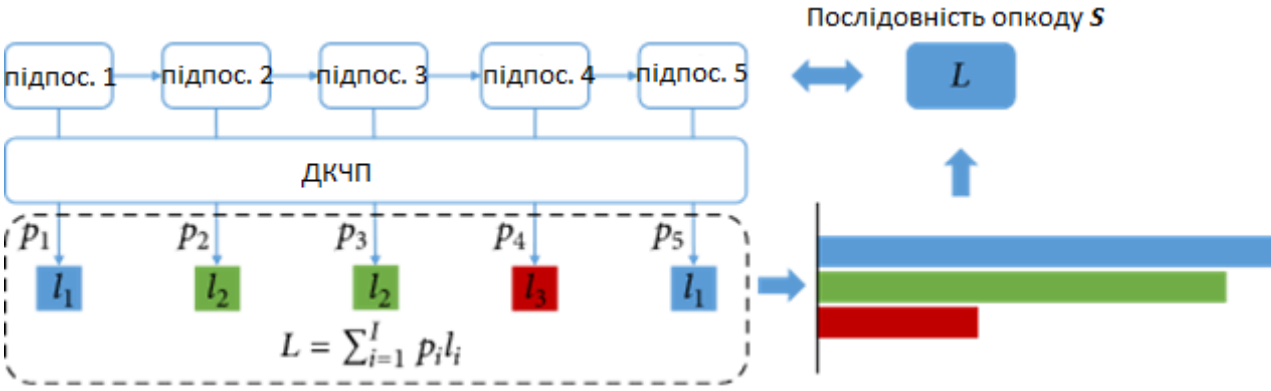

Рис. 2.4. Вибір підпослідовностей та їх злиття

На рисунку 2.4 показано, як метод зваженого голосування застосовується до синтезу підпослідовності. Кожна з вихідних послідовностей розділена на підпослідовності, які є вхідними даними для навченої мережі ДКЧП та отримують відповідні результати класифікації. Нарешті, ми використовуємо ці результати підпослідовності та відповідні ваги, щоб отримати прогноз для вихідної послідовності коду операцій.

2.3.4. Стратегія збільшення даних на основі методу “розсувного вікна”

При вирішенні завдань класифікації з використанням нейронних мереж часто в реальності виникає проблема дисбалансу категорій. Наприклад, кількість доброякісних програм насправді набагато більше, ніж шкідливих програм, і до них легко дістатись. Хоча розподіл різних класів шкідливих програм є більш нерівномірним, ті широко розповсюджені шкідливі програми мають більшу кількість доступних зразків, а непопулярний вид шкідливої програми має лише кілька зразків. Щоб уникнути ігнорування класифікатором цих непопулярних

категорій, що призводить до низької точності розпізнавання, стратегія збільшення даних часто використовується для вирішення проблеми дисбалансу категорій, де надмірні вибірки непопулярних категорій поєднуються з негативною вибіркою популярних категорій для досягнення більш збалансованого розподілу зразків різних категорій. Однак деякі методи збільшення даних, такі як перетворення відображення (широко використовується в даних зображень) та алгоритм SMOTE (заснований на інтерполяції), не підходять для даних послідовностей. Тому пропонується стратегію збільшення даних на основі розсувного вікна, яка використовується для вирішення проблеми дисбалансу категорій для класифікації сімейства шкідливих програм. У попередньому розділі спосіб сегментації підпослідовностей для дуже довгої послідовності не має перетинів, що означає, що між будь-якими двома підпослідовностями немає повторюваного елемента. Для розширення зразків даних ми розглянемо метод сегментації з перетином для дуже довгої послідовності за допомогою розсувного вікна. Довжина вікна точно дорівнює довжині вікна усіченого BPTT, а потім кількість генерованих послідовностей контролюється шляхом встановлення довжини кроку розсувного вікна. Отже, чим менша довжина кроку, тим більше кількість згенерованих послідовностей. Щоб гарантувати, що вихідна послідовність інформації не буде втрачена, ми додаємо обмеження, де довжина кроку повинна бути не більше довжини розсувного вікна. Припустимо, що кількість зразків поточної категорії l_i становить β_i; довжина вибірки α_{ij} , де $j \in (0,1,\ldots,\beta_i)$ і довжина розсувного вікна встановлюється як τ. Щоб розширити вибірку даних для поточного класу, спочатку обчислюємо загальну довжину послідовності категорії, як показано нижче:

$$len(l_i) = \sum_{j=0}^{\beta_i} \alpha_{ij}. \quad (3)$$

Для поточної категорії l_i довжина кроку d_i розсувного вікна обчислюється наступним чином:

$$d_i = \frac{\sum_{j=0}^{\beta_i} \alpha_{ij} - \tau}{\gamma},\ 1 \le d_i \le \tau\ . \quad (4)$$

Оскільки мінімальна довжина кроку становить 1, існує відповідна верхня межа кількості зразків γ, які потрібно розширити:

$$\gamma \le \sum_{j=0}^{\beta_i} \alpha_{ij} - \tau. \quad (5)$$

У наступному завданні класифікації сімейства шкідливих програм ця стратегія збільшення даних може досягти відносно збалансованого розподілу при вибірці даних різних сімейств шкідливих програм.

2.4. Стекове узагальнення

Програмний модуль також вилучає деякі функції метаданих шкідливого програмного забезпечення, крім ДКЧП та ЗНМ. Основна причина полягає в тому, що ДКЧП і ЗНМ фіксують локальну функцію та функцію метаданих, на відміну від них, можуть отримати загальний опис шкідливого програмного забезпечення. І ці функції метаданих легко отримати, такі як розмір вихідних файлів шкідливого програмного забезпечення, початкова адреса байтового файлу, розмір декомпільованого файлу, кількість рядків та довжина різних сегментів PE. Нейромережевий модуль має три частини тимчасових результатів, які є результатом дискримінації ДКЧП, результатом дискримінації ЗНМ та функціями метаданих. Для досягнення остаточного результату виявлення, інтегруємо ці три частини шляхом стогування. Загальна процедура методу стогування[28] включає тренування навчальної моделі для поєднання результатів інших навчальних моделей.

Окремі моделі це моделі першого рівня, що дають тимчасовий результат, тоді як об'єднувальна модель є моделлю другого рівня, для складання результатів моделей першого рівня для кращого прогнозування. Основна ідея полягає в тому, щоб моделі першого рівня були навчені за допомогою оригінального навчального набору, щоб потім використовувати їх результати для створення нового набору даних для навчання моделей другого рівня.

На даний момент вищезазначені три частини призначені для отримання функцій шкідливого програмного забезпечення, що знаходяться на локальному та глобальному рівнях. Використовуючи мережу ЗНМ, мережу ДКЧП та витягаючи особливості моделей першого рівня, а логістичну регресію як модель другого рівня для стекового узагальнення. Процес стогування можна побачити на рисунку 2.5.

Рис. 2.5. Опис процесу стегування/стекового узагальнення

І цільова функція логістичної регресії визначається як

$$P_\theta(x) = \frac{1}{1+\exp(-\theta^T x)}, \quad (6)$$

де θ - параметр. Діапазон значень $P_\theta(x)$становить *(0, 1)*. Для навчальних даних $(x^{(j)}, y^{(j)} = i)$, обчислюється максимальна ймовірність правдоподібності, а втрата використовується для оптимізації θза допомогою методу зворотного поширення помилки. Для навчання моделі логістичної регресії як моделі другого рівня використовується ітеративний метод оптимізації градієнтного спуску-стохастичний градієнтний спуск (СГС) .

РОЗДІЛ 3. ЕКСПЕРИМЕНТИ ТА ОЦІНКИ

У цьому розділі описується експериментальне середовище та конкретну реалізацію нейромережевого модуля. Щоб оцінити та оптимізувати даний модуль, увага була зосереджена на чотирьох частинах: продуктивність ЗНМ, продуктивність ДКЧП, результат стогування та порівняння з іншими роботами, відповідно.

3.1. Впровадження системи

Основна частина програмного модулю заснована на використанні глибоких нейронних мереж. Набір даних про шкідливе програмне забезпечення надає Microsoft [29], який містить 21 736 зразків шкідливого програмного забезпечення з 9 сімейств шкідливих програм в операційній системі Windows, що займає більше 500 ГБ місця. Для кожного зразка передбачено два формати файлів - вихідні файли шкідливого програмного забезпечення (файли двійкового потоку без головки PE) та відповідні декомпільовані файли IDA Pro відповідно. А вихідні файли доброякісного програмного забезпечення збираються за допомогою Cnet [30] та Baidu Software Center [31]. В експериментах з оцінки вимірюються такі показники продуктивності: точність, істиннопозитивний рівень (ІПР), хибнопозитивний рівень (ХПР), рівний коефіцієнт помилок (РКП) та робочу характеристику приймача (РХП). Рівний коефіцієнт помилок є таким же, як значення хибнопозитивного рівня, коли робочий поріг відрегульовано таким чином, що хибнопозитивний рівня і коефіцієнт помилкових відхилень (КПВ) стають однаковими. Основні показники визначаються наступним чином:

(і) істиннопозитивний рівень визначається як ймовірність того, що поточний зразок зловмисного ПО правильно визначений:

$$ІПР = \frac{ІП}{ІП + ХН}, \tag{7}$$

(ii) хибнопозитивний рівень визначається як ймовірність того, що доброякісний зразок був неправильно визначений як шкідливе програмне забезпечення:

$$ХПР = \frac{ХП}{ХП + ІН}, \quad (8)$$

ІП - істинно позитивний, кількість правильно визначених зразків зловмисного ПО, як шкідливі.

ХП - хибно негативний, кількість неправильно визначених зразків зловмисного ПО, як доброякісні програми.

Істиннопозитивний рівень відображає зручність використання програмного модуля, тоді як хибнопозитивний показує рівень безпеки. Тут, через упередженість з міркувань безпеки, оцінка значення ІПР, коли ХПР дорівнює 0,1%, що зберігає дуже низький ХПР, як необхідну умову. Крім того, для оцінки ефективності роботи модуля, потрібно мати на увазі витрати на навчання та на виявлення злоякісного ПО.

3.1.1. ЗНМ нейромережевого модуля

Було побудовано дві мережеві структури ЗНМ, які називаються ЗНМ №1 та ЗНМ №2. Для ЗНМ №1 матриця згортки 5 х 5 та розмір кроку 1. Крім того, для спрощення дизайну мережі ЗНМ, краї зображення заповнюються так, щоб розмір вихідної карти функції операції згортки відповідав вхідному зображенню. У той же час для ЗНМ №1 використовується матриця 2 х 2 для пулінгу та розмір кроку - 2, тому довжина та ширина вибіркових зображень зменшуються до половини оригіналу. ЗНМ №1 повністю використовує 2 шари згортки, 2 шари максимізаційного агрегування та 1 повноз’єднаний шар, а також виконується операція виключення [32] для кожного шару об’єднання та повноз’єднаного шару, щоб запобігти перенавчанню.

Оскільки ЗНМ №1 використовує велике вікно згортки, через це нейромережа не може підтримувати глибшу мережеву структуру, коли розмір вхідного зображення невеликий. З цією метою було побудовано ЗНМ №2 згідно з роботами Симоняна та Зіссермана [33], у якій використовується фільтр згортки

невеликих вікон для побудови більш глибокої мережі. Зокрема, ЗНМ №2 використовує матрицю згортки 3 х 3, розмір кроку 1 і відступ від краю 1 піксель. Крім того, використовується матриця 3 х 3 для операції максимізаційного агрегування, а розмір кроку встановлений на 2. Тож дана нейронна мережа може мати справу з вхідними зображеннями більшого розміру за однакового розміру карти об'єктів, вимагаючи лише невеликої кількості додаткових обчислень. ЗНМ №2 має більш глибоку мережеву структуру, в цілому 3 шари згортки, 2 агрегувальні шари та 2 повноз'єднані шари. Крім того, використовується функція активації Leaky ReLU [34], рівномірн ініціалізацію ваг розподілу та пакетної нормалізації [35], які покращують ефективність конвергенції мережі ЗНМ. Специфікація ЗНМ №1 та ЗНМ №2 наведена у таблиці 3.1.

Таблиця 3.1

Характеристики нейронних мереж ЗНМ №1 та ЗНМ №2

ЗНМ	***Мережева структура***	***Розмір***	***Вихідні дані***
ЗНМ №1	Шар входу	-	(1, 1, 64, 64)
	Шар згортки	32 5 х 5 ядер згортки	(1, 32, 64, 64)
	Агрегувальний шар	2 х 2, крок 1	(1, 32, 32, 32)
	Шар виключення	-	(1, 32, 32, 32)
	Шар згортки	64 5 х 5 ядер згортки	(1, 64, 32, 32)
	Агрегувальний шар	2 х 2, крок 1	(1, 64, 16, 16)
	Шар виключення	-	(1, 64, 16, 16)
	Повноз’єднаний шар	Логістична регресія	(1024, 1)
	Шар виключення	-	(1024, 1)
	Шар виходу	-	1

Продовження табл. 3.1

ЗНМ №2	Шар входу	-	(1, 1, 64, 64)
	Шар згортки	32 3 x 3 ядер згортки	(1, 32, 64, 64)
	Шар згортки	16 3 x 3 ядер згортки	(1, 16, 64, 64)
	Агрегувальний шар	2 x 2, крок 1	(1, 16, 32, 32)
	Шар виключення	-	(1, 16, 32, 32)
	Шар згортки	32 3 x 3 ядер згортки	(1, 32, 32, 32)
	Агрегувальний шар	2 x 2, крок 1	(1, 32, 16, 16)
	Шар виключення	-	(1, 32, 16, 16)
	Повноз’єднаний шар	512 maxout блоків	(32, 512)
	Шар виключення	-	(32, 512)
	Повноз’єднаний шар	Логістична регресія	(32, 1)
	Шар виключення	-	(32, 1)
	Шар виходу	-	1

3.1.2. ДКЧП нейромережевого модуля

Мережа ДКЧП складається з двох шарів, і кожен шар має 185 нейронних вузлів. Для ДКЧП використовується операція дропаут(виключення) як механізм регулювання для зменшення перенавчання. Шар дропаут додано для двох прихованих шарів ДКЧП і значення ймовірності ρ (вибір ρ визначає інтенсивність дропауту) 0.5. Крім того, механізм виключення набирає чинності лише на етапі навчання. Можна вважати, що дропаут допомагає навчатися багатьом підмережам під час фази тренування та фази прогнозування; він поєднує всі підмережі для прогнозування ансамблю.

Крім цього використовується стохастичний градієнтний спуск і встановлюється розмір партії (= 30) для навчання. А в якості оптимізатора використовується метод адаптивної оцінки моментів(Adam) [36]; даний метод поєднує коефіцієнт імпульсу з адаптивним градієнтним алгоритмом(AdaGrad), який забезпечує чіткий контроль за спадом швидкості навчання. Оптимізатору

методу Adam потрібна лише початкова швидкість навчання (=*2e − 3*) як гіперпараметр, і швидкість навчання під час тренувального процесу може бути скоригована адаптивно, не встановлюючи вручну зменшення ваги. Зведені налаштування та параметри мереж ДКЧП наведені у таблиці 3.2.

Таблиця 3.2

Параметри мережі ДКЧП

Параметри мережі	***ДКЧП***
Максимальна кількість ітерацій	**6.40E + 04**
Значення параметру ініціалізації	[-0.4, +0.4]
Усічений ЗПЧ	120
К-сть навчальних зразків	30
Початкова швидкість навчання	2.00E – 03
Ймовірність виключення	0.5
Коєфіцієнт нормалізації градієнту	10
Оптимізаційний алгоритм	Метод адаптивної оцінки моментів
Напрям поширення часових рядів	односторонній
Приховані шари	2
Приховані вузли	185

3.2. Результати тестування ЗНМ

Відповідні зображення у відтінках сірого з 21 736 зразків шкідливого програмного забезпечення та 20 650 зразків доброякісного програмного забезпечення спочатку генеруються та нормалізуються до розміру 64 x 64. Для оцінки двох мереж ЗНМ використовується 6-ступенева перевірка. Вихідний дискримінантний результат для зображень у відтінках сірого є відповідним результатом виявлення шкідливого програмного забезпечення. На рисунках 3.1 та 3.2 представлені ROC - криві (криві похибок) результату тестування для двох мереж та перелік результатів експерименту, що відповідають 6-кратній перехресній валідації, відповідно. Результати показують, що ЗНМ №2 має

проказник AUC 0,99952 та середню точність класифікування 98,14%, що є кращим, ніж середнє значення показника AUC 0,99896 і середня точність класифікації 96,82% для ЗНМ №1. Оскільки зображення у відтінках сірого містить безліч незначної інформації, тому можна вважати, що ЗНМ №2 може досягти кращої продуктивності завдяки своїй більш глибокій мережевій структурі, яка може виконувати більш кращу локалізацію зображень. І хоча глибша мережа часто вимагає більш тривалого навчання, яка забирає багато часу, ЗНМ №2 значно зменшує кількість вузлів, необхідних для повноз'єдналного шару, використовуючи механізм maxout, що не призводить до збільшення параметрів мережі та не зменшує ефективність навчання. Отже, ЗНМ №2 використовується як мережева частина ЗНМ для програмного модуля.

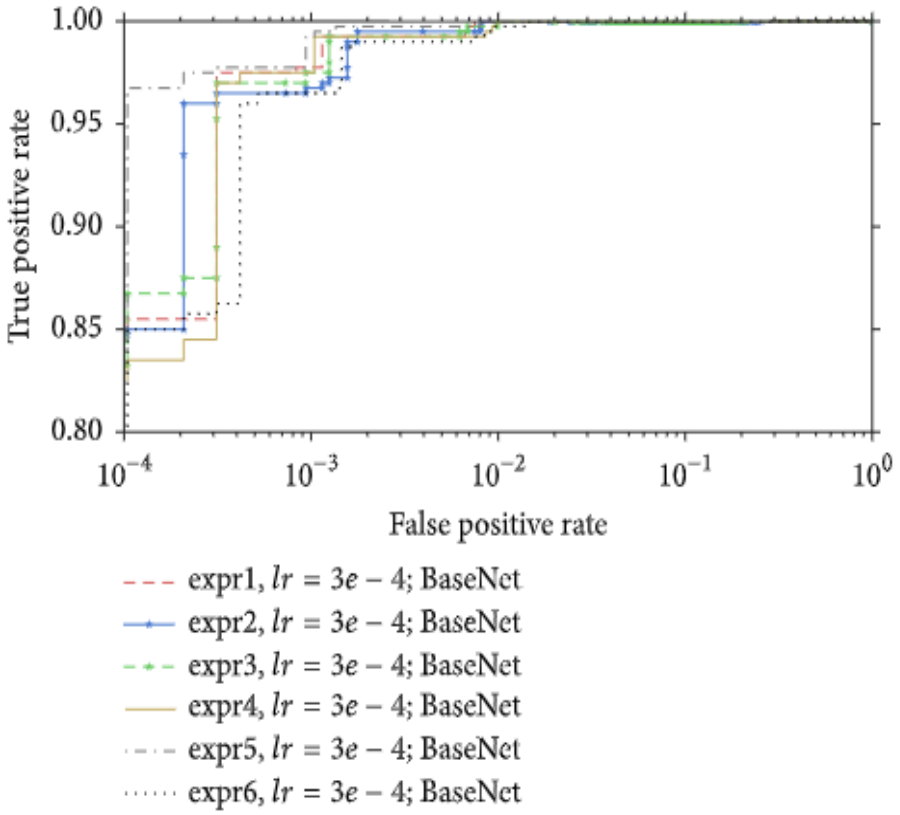

Рис. 3.1. ROC – крива результату класифікації мережі ЗНМ №1

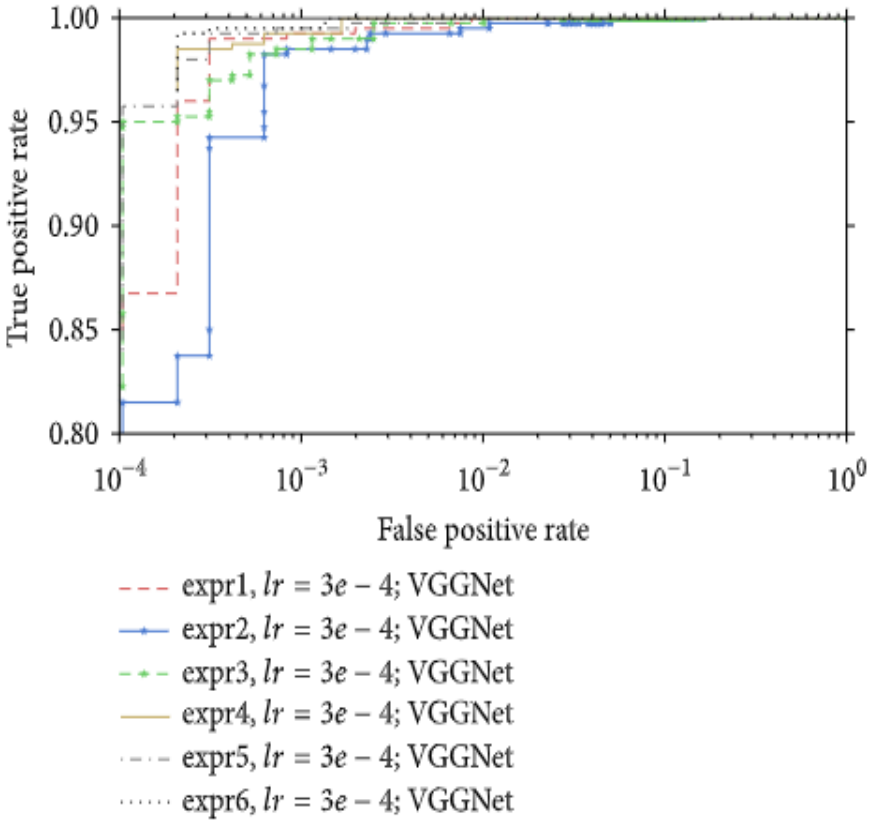

Рис. 3.2. ROC – крива результату класифікації мережі ЗНМ №2

3.3. Дослідження продуктивності ДКЧП

На етапі попередньої обробки даних спочатку визначаються набори кодів операцій(опкодів), які використовуються для вилучення послідовностей кодів операцій з декомпільованих файлів. А потім всі послідовності операційних кодів діляться на кілька підпослідовностей для подальшого навчання ДКЧП із усіченим ЗПЧ(зворотне поширення в часі).

Одним із важливих завдань є оцінка та оптимізація запропонованого механізму вибору підпослідовностей. Основною функцією вибору підпослідовностей є фільтрація підпослідовностей, які містять шум, особливо нешкідливої частини злоякісного програмного забезпечення, щоб якісні вхідні дані для мережі ДКЧП. Тому необхідний кількісний показник для вибору підпослідовності, щоб визначити ступінь допустимого відхилення на наявність шуму у послідовностях операційних кодів. Визначаючи гіперпараметр σ, який називається швидкістю відбору значень, які відносяться до тренувальних послідовностей, що проходять через вибір підпослідовностей. Вибір σ фактично

відображає компроміс між продуктивністю програмного модуля та якістю даних. При проведенні тестування використовуються п’ять різних показників відбору σ (= 100%, 99%, 97,5%, 95%, 90%).

На рисунку видно, що показник AUC є найнижчим, коли σ = 90%, і найвищим, коли σ = 97,5%. Показник AUC коефіцієнта відбору підпослідовностей є кращим за результати порівняння без відбору підпослідовностей (σ = 100%), що доводить, що стратегія відбору підпослідовностей ефективна. Однак ефективність виявлення швидко знижується, коли σ зменшується до 90%. Причиною цього може бути в тому, що мережа ДКЧП зазвичай має певну протишумову здатність, що означає, що вхідні дані з певним шумом (не надто великим) можуть здійснювати невеликий вплив на дискримінантний результат ДКЧП, і нейронна мережа може автоматично сприймати та протидіяти шуму в наборах даних. Фільтрування занадто великої кількості підпослідовностей призведе до того, що ДКЧП не вистачить даних для навчання. Отже, було обрано σ = 97,5% як оптимальний гіперпараметр для стратегії вибору підпослідовності.

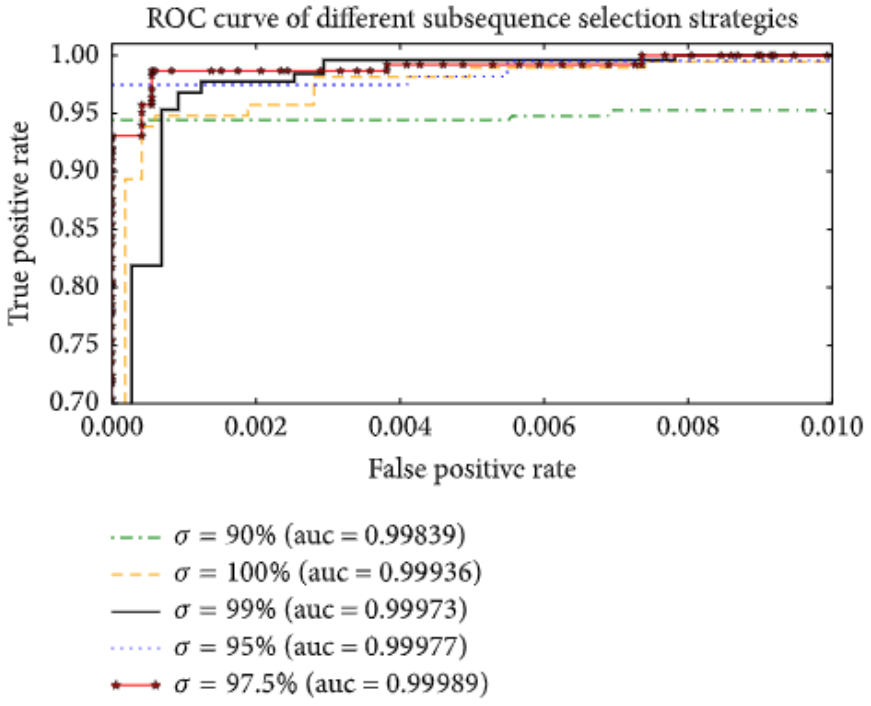

Рис. 3.3. ROC – крива результатів відбору підпослідовностей

Також у тестуванні порівнювалась ефективність роботи стратегії урізання та заповнення (Truncating And Padding) та усіченого ЗПЧ(зворотне поширення в часі) на різній довжині ЗПЧ α, де стратегія ТАР просто скорочує частину послідовності, що перевищує α перед тренуванням, а усічена ЗПЧ ділить послідовність операційних кодів на підпослідовність із довжиною α для навчання. Результат тестування (див. рис. 3.4) показує, що усічений ЗПЧ краще та ефективніше стратегії ТАР на всій тривалості α. При невеликій довжині α при ЗПЧ, стратегія ТАР втрачає велику кількість інформації про послідовність через операцію зрізання, і цей дефект може бути компенсований, коли α збільшується. Проте коли α більше 120, ефективність стратегії ТАР більше не може зростати через зникнення градієнта. На відміну від цього, усічений ЗПЧ не відкидає жодного сегмента послідовності, а розбиває послідовність на підпослідовності, тренуючи та аналізуючи їх, відповідно, а потім робить прогноз такого синтезу. Отже, навіть якщо α має невелике значення, все одно можна отримати значну ефективність виявлення. Але можна помітити, що ефективність лише незначно покращується, коли α збільшується і навіть починає знижуватися, коли α перевищує 180; це тому, що, коли занадто велике значення α, метод усіченого ЗПЧ на кожній підпослідовності зіткнеться з проблемою зникнення градієнта, і в цьому випадку підпослідовність слід додатково розділити та навчити окремо. Отже, нарешті було обрано α= 120 для ДКЧП у програмному модулі.

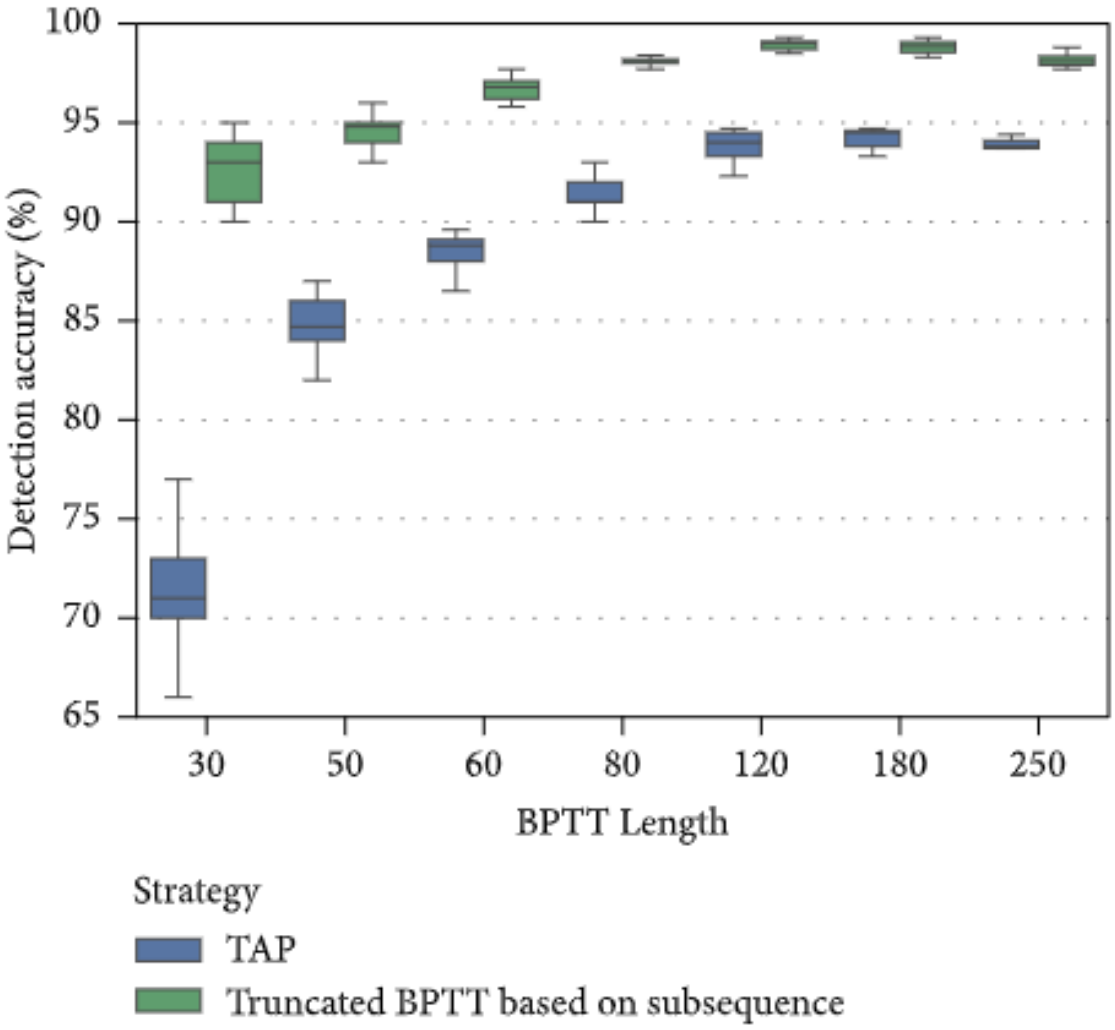

Рис. 3.4. Результат детектування при різних значеннях коефіцієнту ЗПЧ

Мережа ДКЧП містить 2 приховані шари з 185 нейронних вузлів у поєднанні із усіченими ЗНМ та стратегіями вибору послідовностей, де довжина α усіченого ЗНМ становить 120, а швидкість вибору підпослідовності σ становить 97,5%. Навчальний процес займає 1,34 години, а результат виявлення досягає точності 99,13%, а істиннопозитивний рівень досягає 98,69%, коли істиннонегативний становить 0,1%.

Таблиця 3.3

Результати тестування мережі ДКЧП

Алгоритм	***Модель***	***Точність (%)***	***AUC***	***ІПР (ІНР = 0, 1%)***	***РКП (%)***	***Швидкість навчання (год)***
Урізання та заповнення	ДКЧП (α = 30)	71.43	0.8863	52.03	-	0.41
	ДКЧП (α = 60)	87.13	0.9791	81.67	6.45	0.54
	ДКЧП (α = 80)	91.56	0.9854	85.39	4.88	0.89
	ДКЧП (α = 120)	94.86	0.9931	91.53	3.17	-
Усічений ЗПЧ	ДКЧП (α = 30)	94.37	0.9928	91.11	3.10	-
	ДКЧП (α = 60)	96.83	0.9950	93.37	-	-
	ДКЧП (α = 80)	98.08	0.9989	95.13	-	1.24
	ДКЧП (α = 120)	98.47	0.9993	96.81	-	1.36

Продовження табл. 3.3

	ДКЧП (α = 180)	97.82	0.9987	95.42	-	1.53
Усічений ЗПЧ + вибір підпослідовностей	ДКЧП (α = 30, σ = 90%)	97.98	0.9988	95.01	1.34	1.16
	ДКЧП (α = 60, σ = 95%)	98.83	0.9997	97.22	0.84	-
	ДКЧП (α = 80, σ = 99%)	98.66	0.9996	96.69	0.92	-
	ДКЧП (α = 120, σ = 97.5%)	**99.13**	**0.9999**	**98.69**	**0.54**	**1.34**

3.4. Результат стекового узагальнення

Нейромережевий модуль об'єднує результати мережі ЗНМ та мережі ДКЧП шляхом стогування(стекового узагальнення). По суті, стогування поєднує три частини (дискримінантний результат ЗНМ, дискримінантний результат ДКЧП та функції метаданих) для навчання учня другого рівня, який є класифікатором логістичної регресії.

Крім того, метод виявлення зловмисного програмного забезпечення відтворюється на основі N - грам відповідно з дослідженнями [15, 20, 21] і використовується як базовий для набору даних. Він витягує 1-грамові, 2-грамові та 3-грамові функції як в байтах, так і в опкодах, відкидає низькочастотні характеристики, робить подальший вибір функцій відповідно до важливості особливостей із випадкового лісу і, нарешті, створює вектор ознак для кожного зразку і готує класифікатор методу опорних векторів для виявлення шкідливого програмного забезпечення.

Після навчання всіх моделей для виявлення, чи це шкідливе програмне забезпечення на тестовому наборі, що містить 2000 зразків шкідливого програмного забезпечення та 2000 зразків доброякісного програмного забезпечення, результати порівняння видно в таблиці 5 (ФМ представляє функцію метаданих).

Таблиця 3.4.

Результати виявлення шкідливого забезпечення різними методами

Алгоритм	***Точність (%)***	***AUC***	***IPP (%)***	***INP (%)***	***РКП (%)***	***Швидкість навчання (год)***	***Час виявлення (мілісек)***
N-грам	93.21	0.9864	89.22	0.1	3.94	4.18	2304
ДКЧП	99.13	**0.9999**	98.69	0.1	1.34	**1.34**	**13.62**
ЗНМ	98.14	0.9989	96.92	0.1	1.46	1.46	15.97
ДКЧП + ЗНМ + ФМ	**99.88**	**0.9999**	**99.14**	0.1	2.91	2.91	30.33

ВИСНОВКИ

Аналіз класичних методів захисту показав, що статичний та динамічний аналізи не є настільки ефективними та продуктивними, як використання нейронних мереж для захисту інформаційних систем.

У даній роботі пропонується метод виявлення шкідливої інформації, який використовує дві глибокі нейронні мережі ЗНМ і ДКЧП, щоб отримувати дані із зображень у відтінках сірого та послідовностей операційних кодів, вилучених з необроблених двійкових виконавчих файлів, а потім для їх злиття використовується стекове узагальнення. При тестування даний модуль досягає точності 99,88% та 99,14% - ІПР, а ІНР - 0,1%.

Даний метод виявлення шкідливої інформації виконує такі завдання як генерація зображень у відтінках сірого, вивчення дуже довгих послідовностей та проблема зникнення градієнта для ДКЧП, паралельні обчислення для ДКЧП та обробка даних із шумом.

У даній роботі наявні результати експериментів з оцінки нейромережевого модуля, включаючи виявлення шкідливого програмного забезпечення та класифікацію сімейства шкідливих програм.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kaspersky Security Bulletin 2016. Загальна статистика 2016," https://securelist.com/kaspersky-security-bulletin-2016-executive summary/76858/.

2. McAfee Labs Threats Report in June 2017, https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf.
View at: Google Scholar

3. D. K. S. Reddy and A. K. Pujari, "N-gram analysis for computer virus detection," *Journal of Computer Virology and Hacking Techniques*, vol. 2, no. 3, pp. 231–239, 2006.

4. M. Narouei, M. Ahmadi, G. Giacinto, H. Takabi, and A. Sami, "DLLMiner: Structural mining for malware detection," *Security and Communication Networks*, vol. 8, no. 18, pp. 3311–3322, 2015.

5. G. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using CWSandbox," *IEEE Security and Privacy*, vol. 5, no. 2, pp. 32–39, 2007.

6. K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Technical Report*, University of Mannheim, 2009.

7. M. Zakeri, F. Faraji Daneshgar, and M. Abbaspour, "A static heuristic approach to detecting malware targets," *Security and Communication Networks*, vol. 8, no. 17, pp. 3015–3027, 2015.

8. A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC '07)*, pp. 421–430, December 2007.

9. C. Wressnegger, K. Freeman, F. Yamaguchi, and K. Rieck, "Automatically inferring malware signatures for anti-virus assisted attacks," in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security, (ASIA CCS '17)*, pp. 587–598, UAE, April 2017.

10. L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. Mitchell, "A layered architecture for detecting malicious behaviors," in *Proceeding of the International Symposium on Recent Advances in Intrusion Detection (RAID '08)*, 2008.

11. W. Lee and S. J. Stolfo, “A framework for constructing features and models for intrusion detection systems,” *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 227–261, 2000.

12. P. Li, L. Liu, D. Gao, and M. K. Reiter, “On challenges in evaluating malware clustering,” in *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID '10)*, vol. 6307, 2010.

13. M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” in *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, pp. 38–49, May 2001.

14. W. Cohen, “Fast effective rule induction,” in *Proceeding of the 12th International Conference on Machine Learning*, 1995.

15. J. Z. Kolter and M. A. Maloof, “Learning to detect and classify malicious executables in the wild,” *Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2004.

16. J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in *Proceedings of the 10th International Conference on Malicious and Unwanted Software, (MALWARE '15)*, pp. 11–20, USA, October 2015.

17. L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in *Proceedings of the 8th International Symposium on Visualization for Cyber Security, (VizSec '11)*, USA, July 2011.

18. L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, “A comparative assessment of malware classification using binary texture analysis and dynamic analysis,” in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (AISec '11)*, pp. 21–30, 2011.

19. D. Kong and G. Yan, “Discriminant malware distance learning on structural information for automated malware classification,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (KDD '13)*, pp. 1357–1365, USA, August 2013.

20. I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," in *Proceedings of the ICEIS 2009 - 11th International Confeence on Enterprise Information Systems*, pp. 317–320, May 2009.

21. A. Shabtai, R. Moskovitch, C. Feher, and etal., "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Security Informatics*, vol. 1, no. 1, 2012.

22. I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "OPEM: A static-dynamic approach for machine-learning-based malware detection," in *Proceedings of the International Joint Conference CISIS'12-ICEUTE12-SOCO12 Special Sessions, 2013*, vol. 189, pp. 271–280.

23. IDAPro.,http://www.hexrays.com/products/ida/support/download_freeware.shtml.

24. A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

25. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

26. R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, vol. 2, no. 4, pp. 490–501, 1990.

27. P. Malhotra, A. Ramakrishnan, and G. Anand, "LSTM-based encoder-decoder for multi-sensor anomaly detection," https://arxiv.org/abs/1607.00148.

28. Z.-H. Zhou, *Ensemble methods foundations and algorithms. Machine Learning & Pattern Recognition*, Taylor & Francis, London, UK, 2012.

29. Microsoft Malware, https://www.kaggle.com/c/malware-classification.

30. PC. Windows Software, http://download.cnet.com/windows/.

31. Baidu Software Center, http://rj.baidu.com/.

32. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," https://arxiv.org/abs/1207.0580.

33. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proceedings of the 3rd International Conference for Learning Representations (ICLR '15)*, 2015.

34. A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities, improve neural network acoustic models,” in *In Proceeding of the 30th International Conference on Machine Learning (ICML '13)*, 2013.

35. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML '15)*, pp. 448–456, July 2015.

36. D. Kingma and J. B. Adam, “A method for stochastic optimization,” in *Proceedings of the 3rd International Conference for Learning Representations (ICLR '2015)*, 2015.

37. L. Wang, “Microsoft Malware Classification Challenge (BIG 2015) First Place Team: Say No To Overfitting,” https://github.com/xiaozhouwang/kaggle_Microsoft_Malware/blob/master/Saynotooverfitting.pdf.

38. M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, “Novel feature extraction, selection and fusion for effective malware family classification,” in *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy, (CODASPY '16)*, pp. 183–194, USA, March 2016.

39. B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, “Performance analysis of machine learning and pattern recognition algorithms for Malware classification,” in *Proceedings of the 2016 IEEE National Aerospace and Electronics Conference and Ohio Innovation Summit, (NAECON-OIS '16)*, pp. 338–342, USA, July 2016.

40. F. C. Garcia and F. P. Muga, “Random forest for malware classification,” https://arxiv.org/abs/1609.07770.

41. E. Burnaev and D. Smolyakov, “One-class SVM with privileged information and its application to malware detection,” https://arxiv.org/abs/1609.08039.

42. J. Kim, S. Bu, and S. Cho, “Malware detection using deep transferred generative adversarial networks,” in *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, 2017.

43. J. Drew, M. Hahsler, and T. Moore, “Polymorphic malware detection using sequence classification methods and ensembles: BioSTAR 2016 Recommended Submission - EURASIP Journal on Information Security,” *EURASIP Journal on Information Security*, vol. 2017, no. 1, article no. 2, 2017.

44. K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial perturbations against deep neural networks for malware classification,” https://arxiv.org/abs/1606.04435.

45. B. Biggio, I. Corona, D. Maiorca et al., “Evasion attacks against machine learning at test time,” in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2013.

46. A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” https://arxiv.org/abs/1412.1897.

47. N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security, (ASIA CCS '17)*, pp. 506–519, April 2017.

48. N. Carlini and D. Wagner, “Defensive distillation is not robust to adversarial examples,” https://arxiv.org/abs/1607.04311.

49. N. Carlini and D. Wagner, “Adversarial examples are not easily detected: Bypassing ten detection methods,” https://arxiv.org/abs/1705.07263.

50. F. Liao, M. Liang, and Y. Dong, “Defense against adversarial attacks using high-level representation guided denoiser,” https://arxiv.org/abs/1712.02976.

51. Бачило И.Л. Информационное право : учебник / И. Л. Бачило, В. Н. Лопатин, М. А. Федотов ; под ред. акад. РАН Б. Н. Топорнина. 2-е изд., с изм. и доп. СПб. : Изд-во Р. Асланова -Юридический центр Пресс, 2005. 725 с.

52. Лисенко О. О. Правовий захист суспільства від шкідливої інформації : автореф. дис. на здобуття наук. ступеня канд. юрид. наук : спец. 12.00.07 / О. О. Лисенко ; Харк. нац. ун-т внутр. справ. Х., 2011. 20 с.

53. Минбалеев А. В. Проблемы правового регулирования информации ограниченного использования (вредной информации), распространяемой в сети Интернет / А. В. Минбалеев // Междунар. конф. -Право и Интернет» (Москва, 2728 окт. 2005 г.).

Додаток А

Псевдокод алгоритму вилучення послідовностей опкодів

Input: Executive file
Output: Opcode sequence

```
(1)  files = get_files(); // Get all executive files;
(2) for i in files;
(3)    file = open(i.asm); // Open the corresponding IDA pro decompiled file;
(4)    for line in file; // Read in line;
(5)      words = line.split(" "); // Cut the line into phrases by space character;
(6)       for word in words;
   //To judge each phrase, it requires to meet the following two points at the same time:
   (1) The current word belongs to opcode set opcode_set;
   (2) The last three words are not duplicated opcodes.
(7)       if word in opcode_set and (word! = last_word and last_last_word! = last_word) :
(8)       last_last_word = last_word;
(9)       last_word = word;
(10)      filter_words.add(word);
(11)      end if
(12)      end for
(13)   end for
(14) end for
```

Додаток Б

Блок-схема вибору підпослідовностей ДКЧП