Національна академія наук України
Інститут кібернетики імені В. М. Глушкова

Кваліфікаційна наукова
праця на правах рукопису

ПАНЧУК БОГДАН ОЛЕКСАНДРОВИЧ

УДК 004.05

ДИСЕРТАЦІЯ

ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК АЛГОРИТМАМИ ШТУЧНОГО ІНТЕЛЕКТУ

122 Комп’ютерні науки
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

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

Б.О. Панчук

Науковий керівник – Летичевський О.О., доктор фізико-математичних наук, завідувач відділу теорії цифрових автоматів Інституту кібернетики імені В.М. Глушкова Національної Академії Наук України

Київ – 2025

АНОТАЦІЯ

Панчук Б. О. Виявлення мережевих атак алгоритмами штучного інтелекту. - Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки. – Інститут кібернетики імені В.М. Глушкова НАН України, Київ, 2025.

У дисертаційній роботі розв’язана актуальна дослідницька та практична задача – створення системи виявлення зразків трафіку зловмисних програм з підвищеною стійкістю до спроб навмисного ухилення від виявлення та формально верифікованими оцінками показників стійкості.

Метою роботи є оцінювання, верифікація та підвищення стійкості систем на основі штучного інтелекту для виявлення шкідливого мережевого трафіку, зокрема ботнет-атак, до змагальних впливів з урахуванням семантики можливих дій зловмисника.

Для досягнення зазначеної мети було поставлено ряд завдань в рамках цього дослідження:

- проаналізувати існуючі підходи до виявлення мережевої активності зловмисних програм засоби штучного інтелекту, розглянути відкриті набори трафіку ботнетів та реалізувати метод їх представлення у формі придатній для обробки алгоритмами штучного інтелекту;
- розробити систему виявлення зловмисного трафіку на основі методу виділення мережевих потоків та їх класифікації засобами штучного інтелекту, експериментально виділити моделі з найбільш оптимальними показниками ефективності на доступних наборах даних;
- реалізувати метод генерації штучних змагальних прикладів мережевих потоків та здійснити доповнення ними навчальної вибірки даних для підвищення стійкості системи до можливостей ухилення від виявлення;
- формалізувати критерій локальної стійкості до збурень у вхідних даних класифікатора мережевого трафіка, через накладання обмежень на

параметри нейронної мережі та розробити метод верифікації виконуваності цих обмежень за допомогою SMT-розв'язувача;

У *першому розділі* дисертації зроблено огляд різних методик запропонованих іншими дослідниками в області аналізу мережевих даних та виявлення мережевих загроз методами штучного інтелекту. У розділі проведена систематизація існуючих знань та зроблено порівняння існуючих підходів, а також описано їх історичний та практичний контекст та висвітлено їх переваги та недоліки.

У *другому розділі* було описано узагальнюючий підхід для побудови повноцінних систем для виявлення мережевої активності ботнетів методами штучного інтелекту (ШІ). Розглядається алгоритм реконструкції потоків мережевого трафіку як основний метод виділення числових характеристик, а також різні типи класифікаторів ШІ з метою досягнення найкращої якості виявлення. Було застосовано і описано різні методи попередньої обробки даних, що допомогло ще більше покращити результати. Також пропонуються деякі варіанти для майбутніх удосконалень підходу до виділення мережевих характеристик. На основі цих досліджень була реалізована програма реконструкції потоків та виявлення трафіку ботнетів та інших загроз у режимі онлайн. В рамках розробки даного застосунку, проводилось навчання кількох моделей класифікації мережевих даних, які були породжені різними типами програмного забезпечення. Для навчання та оцінки якості моделей було використано два відкриті набори даних ботнетів. Для покращення ефективності роботи класифікаторів був створений окремий розширений набір на основі комбінування даних з двох попередніх, в який також включались додаткові приклади трафіку новіших типів вірусних програм, не присутні в жодному з оригінальних наборів. Для кожного класифікатора була проведена оцінка хибно позитивних рівнів, а також показників чутливості до різних типів зловмисного ПЗ. На базі отриманих даних для кожної з моделей були підібрані найбільш вигідні вихідні порогові значення.

У *третьому розділі* описується метод оцінювання стійкості систем виявлення ботнетів на основі нейронних мереж з точки зору їх вразливості до змагальних атак, і підвищення стійкості таких систем через розширення

навчального набору штучними даними. У ході роботи реалізовано метод генерації змагальних прикладів для системи класифікації трафіку на основі нейронної мережі шляхом адаптації «методу швидкого знаку градієнту» для роботи з мережевими даними. Метод показав свою ефективність не лише для області виявлення ботнетів, а й для мережевих атак іншого роду. Визначними рисами підходу є обчислювальна простота та правдоподібність отриманих прикладів трафіку.

У *четвертому розділі* представлено метод формальної верифікації властивостей нейронних мереж за допомогою “Satisfiability Modulo Theories” (SMT) розв’язувача. Запропоновано спосіб спрощення обчислювального графа нейромережі над певними регіонами вхідного простору, що значно прискорило алгоритм верифікації. Описаний метод можна використовувати для верифікації довільних повнозв’язних нейромереж глибокого навчання з кусково-лінійними функціями активації. Для демонстрації результатів, метод верифікації був застосований для перевірки роботи нейромережі-аналізатора мережевого трафіку з метою виявлення активності ботнетів, що дозволило показати стійкість класифікатора до «змагальних атак» на основі варіювання розмірів пакетів та міжпакетних інтервалів. Отримані результати роблять важливий внесок у розвиток застосування штучного інтелекту в області кібербезпеки, де можливість інтерпретації та передбачуваність роботи систем виявлення загроз є критично необхідними.

Під час проведення цього дослідження було отримано ряд важливих результатів:

1. Набув подальшого розвитку метод генерації штучних прикладів та доповнення наборів даних шляхом адаптації швидкого методу знаку градієнту до простору ознак мережевих потоків, з метою оцінки та підвищення стійкості систем виявлення шкідливого трафіку до змагальних атак з урахуванням семантики можливих дій зловмисника.
2. Вперше розроблено універсальний метод формальної верифікації властивостей глибоких нейронних мереж з кусково-лінійними

функціями активації, який базується на представленні обчислювального графу нейромережі у формі спрощеної SMT-формули, побудованої інкрементально, шляхом розв’язуванням локальних SMT-задач сформованих для функцій активації нейронів та перевірки можливості їх тотожної заміни на лінійні функції.

3. Вперше формалізовано критерій локальної стійкості класифікатора мережевого трафіка та здійснено його верифікацію шляхом автоматичного доведення (чи спростування) виконуваності накладених на модель обмежень за допомогою SMT-розв'язувача, що дозволило достовірно оцінити стійкість створеної системи виявлення загроз до можливих збурень у вхідних даних.

Практичне значення роботи полягає у розв’язанні актуальної дослідницької та прикладної задачі розробки системи виявлення зразків трафіку ботнетів з підвищеною стійкістю до спроб навмисного ухилення від виявлення, а також формальним доведенням критеріїв її стійкості.

В ході роботи проводилось навчання класифікаторів мережевого трафіку на базі алгоритмів машинного навчання та нейронних мереж з використанням різних наборів мережевих даних. Для навчання та тестування було створено розширену вибірку навчальних даних на основі комбінації різних відкритих наборів, що дозволило підняти показник повноти виявлення TCP-трафіку зловмисного ПЗ з 46% до 80.7% без погіршення значення хибнопозитивного рівня. Зокрема, отримані моделі здатні виявляти трафік таких зловмисних програм як TrickBot, Emotet та Wannacry з повнотою >80%. Крім того, навчання на наборі доповненому штучними прикладами дозволило знизити показник вразливості нейронної мережі до змагальних атак з 8.5% до 2% за умови варіювання розмірів пакетів в межах до 30% від оригінальних значень прикладів з тестового набору.

На основі отриманих результатів був створений прототип багатоцільового аналізатора мережевого трафіку, екземпляр якого було впроваджено в експлуатацію у ролі системи виявлення мережевих загроз у компанії ТОВ «НВП

«Радікс», що спеціалізується на розробці апаратного та програмного забезпечення для АЕС.

Ключові слова: *кібербезпека, штучний інтелект, системи виявлення вторгнень, мережа, мережеві потоки, машинне навчання, ботнети, розширення даних, прогнозування даних, змагальні атаки, змагальне навчання, формальна верифікація, стійкість, надійність, оптимізація, якість систем, критерії якості систем, вимоги якості програм, оцінка якості програм, класифікація, аугментація, нейронні мережі, функція втрат, порогове значення, інтернет-рішення, мережева інфраструктура, гетерогенні дані*

ABSTRACT

Panchuk B. O. Detection of Network Attacks Using Artificial Intelligence Algorithms – Qualification scientific work presented as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 122 Computer Science. – V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine, Kyiv, 2025.

In this study, an important scientific and applied problem has been addressed — the creation of a system for malicious traffic samples detection with enhanced resilience against deliberate evasion attempts and formally verified robustness metrics.

The aim of this work is to assess, verify, and improve the robustness of systems based on artificial intelligence for detecting malicious network traffic, botnet attacks in particular, against adversarial influence while considering the semantics of potential attacker actions.

In order to achieve the goal, several tasks were established within this study:

- To analyze existing approaches for malicious network activity detection using artificial intelligence, review public botnet traffic datasets, and implement a method for representing the network data in a format suitable for AI processing;

- To develop a malicious traffic detection system based on network flows using AI techniques, and to experimentally identify the models with the highest performance metrics on available datasets;
- To implement a method for generating artificial adversarial examples of network flows and use them to augment the training dataset to enhance the system's robustness against possible evasion attempts;
- To formalize the criterion of local robustness to input data perturbations of a traffic classifier by imposing constraints on the parameters of a neural network, and develop a method for verifying the satisfiability of these constraints using an SMT solver.

In the first chapter of the dissertation provides a review of various techniques proposed by other researchers in the field of network data analysis and threat detection using AI. It systematizes existing knowledge, compares current approaches, and discusses their historical and practical context, highlighting strengths and weaknesses.

The second chapter describes describes a generalized approach to building full-fledged botnet detection systems using AI. The method of network flow reconstruction is considered as the primary technique for extracting numerical features, alongside various AI classifier types aimed at achieving optimal detection quality. Several preprocessing techniques are discussed and applied to further improve results. Suggestions for enhancing feature extraction methods are also provided. Based on these studies, a program for real-time flow reconstruction and detection of botnet and other malicious traffic was implemented. During its development, multiple network classification models were trained using different software-generated traffic. Two public botnet datasets were used for training and evaluation. To boost classifier performance, a separate extended dataset was created by combining the data with additional samples of newer malware types not present in the original datasets. For each classifier, false positive rates and sensitivity to various malware types were assessed, and optimal output thresholds were selected based on the results.

The third chapter presents a method for evaluating the robustness of botnet detection systems based on neural networks in terms of their vulnerability to adversarial attacks, and for increasing their resilience by augmenting training sets with synthetic data. An adversarial example generation method was implemented using an adapted version of the Fast Gradient Sign Method (FGSM) for network data. This method proved being effective not only for botnet detection but for other types of network threats as well. The benefits of the approach are the computational efficiency and the credibility of the generated network flow samples.

The fourth chapter introduces a method for formal verification of neural network properties using a Satisfiability Modulo Theories (SMT) solver. A method for simplification of the computational graph of a neural network over specified input regions is proposed, which allowed to significantly accelerate the verification algorithm. The described method is applicable to arbitrary fully connected deep neural networks with piecewise-linear activation functions. To demonstrate its utility, the verification method was applied to a neural network traffic analyzer tasked with botnet activity detection, showing classifier robustness to adversarial attacks involving variation in packet sizes and inter-packet intervals. The obtained results contribute significantly to the application of AI in cybersecurity, where interpretability and predictability of threat detection systems are critically important.

Several important results were obtained during the course of this research:

1. The method of artificial example generation and dataset augmentation was further developed by adapting the fast gradient sign method to the feature space of network flows, aiming to assess and improve system robustness against adversarial attacks while accounting for possible attacker actions.
2. A new universal method for formal verification of deep neural networks with piecewise-linear activation functions was developed. It is based on representing the network's computational graph as a simplified SMT formula built incrementally by solving local SMT tasks for activation functions and verifying whether they can be identically replaced with linear functions.

3. A formalization of the local robustness criterion for network traffic classifiers was proposed and verified through automatic proof (or refutation) of the imposed model constraints using an SMT solver. This allowed for a reliable assessment of the detection system's robustness to potential perturbations in the input data.

The practical significance of this lies in solving a relevant research and applied problem—designing a botnet traffic detection system with enhanced resistance to deliberate evasion and formally proven robustness criteria.

During the research, classifiers based on machine learning and neural networks were trained using various network datasets. A combined, expanded training dataset was created, resulting in an increase in detection recall for TCP-based malware traffic from 46% to 80.7%, without increasing false positive rate. Notably, the resulting models are capable of detecting malware such as TrickBot, Emotet, and Wannacry with recall rates above 80%. Moreover, training with artificially augmented data reduced the neural network’s vulnerability to adversarial attacks from 8.5% to 2% under conditions of up to 30% variation in packet sizes relative to original values in the test set.

Based on the findings, a prototype of a multipurpose network traffic analyzer was developed, and the program was deployed as a threat detection system at Radics LLC, which specializes in hardware and software development for nuclear power plants.

Keywords: *cybersecurity, artificial intelligence, intrusion detection systems, network, network flows, machine learning, botnets, data augmentation, data prediction, adversarial attacks, adversarial training, formal verification, robustness, reliability, optimization, system quality, system quality criteria, software quality requirements, software quality assessment, classification, augmentation, neural networks, loss function, threshold value, internet solutions, network infrastructure, heterogeneous data*

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Основні наукові результати дисертації опубліковані у фахових виданнях України

1. Панчук Б. Виявлення ботнет-трафіку на основі потоків, використовуючи ШІ // Проблеми програмування. – 2022. – № 3–4 (Спеціальний випуск). – С. 376–386. – DOI: 10.15407/pp2022.03-04.376.
2. Панчук Б. О. Генерація та використання змагальної вибірки для протидії ухиленню ботнетів від виявлення нейронними мережами (до 100-річчя з дня народження академіка В.М. Глушкова) // Проблеми кібербезпеки та інформаційних технологій. – 2023. – Т. 68, № 5. – С. 71–85. – DOI: 10.34229/1028-0979-2023-5-6.
3. Панчук Б. Формальна верифікація нейронних мереж глибокого навчання // Проблеми програмування. – 2024. – № 2–3. – С. 253–262. – DOI: 10.15407/pp2024.02-03.253.
4. Летичевський О., Панчук Б. Проблема точності в системах протидії кібератакам та верифікація нейронних мереж на прикладі задачі виявлення ботнетів // Кібербезпека та інформаційні технології. – 2025. – С. 3–12. – DOI: 10.34229/KCA2522-9664.25.2.1.

Наукові результати дисертації апробовано та опубліковано у матеріалах міжнародної конференції

1. Panchuk B. Flow-Based Botnet Detection with AI Models // Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG 2022. – Kyiv, Ukraine: CEUR Workshop Proceedings (CEUR-WS.org), 2022. – С. 315–328.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ – Штучний Інтелект

IDS – Intrusion Detection System (система виявлення вторгнень)

NIDS – Network Intrusion Detection System (мережева система виявлення вторгнень)

C&C – Command and Control (центр команд та контролю)

DoS – Denial-of-Service (відмова в обслуговуванні)

DDoS – Distributed Denial-of-Service (розподілена відмова в обслуговуванні)

FNR – False-Negative Rate (хибнонегативний рівень)

FPR – False-Positive Rate (хибнопозитивний рівень)

TPR – True-Positive Rate / Sensitivity (чутливість / повнота)

HTTP – Hypertext Transfer Protocol

UDP – User Datagram Protocol (протокол датаграм користувача)

TCP – Transmission Control Protocol (протокол управління передачею)

ICMP – Internet Control Message Protocol (міжмережевий протокол керуючих повідомлень)

DNS – Domain Name System (система доменних імен)

DGA – Domain Generation Algorithm

IRC – Internet Relay Chat

TLS – Transport Layer Security (захист на транспортному рівні)

SMTP – Simple Mail Transfer Protocol (простий протокол пересилання пошти)

P2P – Peer-to-Peer ("рівний до рівного")

IoT – Internet of Things (інтернет речей)

LAN – Local Area Network (локальна мережа)

WAN – Wide Area Network (глобальна мережа)

NAT – Network Address Translation ("перетворення мережевих адрес")

IPFIX – IP Flow Information Export

IANA – Internet Assigned Numbers Authority

IAT – Inter-arrival Time (час міжпакетного прибуття)

DNN – Deep Neural Network (глибока нейронна мережа)

LSTM – Long Short-Term Memory (довга короткочасна пам'ять)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

PCA – Principal Component Analysis (метод головних компонент)

ROC – Receiver Operating Characteristic (робоча характеристика приймача)

AUROC – Area Under ROC (площа під ROC-кривою)

RF – Random Forest (випадковий ліс)

SVM – Support Vector Machine (машина опорних векторів)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

LIME – Local Interpretable Model-agnostic Explanations

SMOTE – Synthetic Minority Oversampling Technique

ANOVA – Analysis of Variance (дисперсійний аналі)

SMT – Satisfiability Modulo Theories

ВСТУП

Актуальність теми.

На ранніх етапах розвитку мережі Інтернет вже було відомо про потенційний масштаб проблеми мережевих загроз для під'єднаних до неї пристроїв. Однією з перших задокументованих масштабних мережевих атак, що відбулася в 1988 р., став «хробак Морріса». Програма, що пізніше отримала таку назву, була створена аспірантом Корнельського університету Робертом Моррісом, і взагалі не задумувався як злоякісна, а лише мала на меті «виміряти розмір інтернету». Однак через допущену при кодуванні помилку, хробак став безконтрольно поширюватись і по оцінкам істориків заразив біля 10% всіх комп'ютерів, під'єднаних до мережі на той час, завдавши значних збитків.

З плином часу та все більшим розростанням мережі Інтернет, мережеві загрози стали звичайним явищем. Звичайні засоби захисту, як-от ретельно налаштовані правила брандмауера та антивірусні програми на основі сигнатур, хоч і корисні, однак не охоплюють весь простір можливих атак. Безліч атак проводяться через загальновідомі відкриті порти і маскуються під звичайну мережеву активність. Таку активність неможливо виявити, використовуючи лише статично орієнтовані механізми безпеки, які працюють на основі наперед визначених шаблонів чи алгоритмічних правил.

Яскравим прикладом загрози, для якої проблема виявлення стоїть особливо гостро й на якій сфокусована увага в даній роботі, є ботнети. Ботнет – це координована група під'єднаних до мережі заражених пристроїв, яка керується зловмисником і використовується для проведення масивних атак різного роду, як-от DDoS (distributed denial of service), крадіжка даних та розсилання спаму. За оцінками експертів ще на 2007 рік від 16 до 25% всіх комп'ютерів в мережі були інфіковано ботнетами, а на момент 2014 року ботнети щорічно інфікували близько півмільйона нових пристроїв завдаючи збитків порядку 110 мільярдів доларів згідно з даними ФБР. У звіті дослідницької компанії Imperva «Bad Bot Report 2024» зазначається, що середній об'єм трафіку, який створюють шкідливі боти, досяг позначки в 32% від глобального, що є рекордним показником за всю історію.

Однак, незважаючи на масштаб загрози, усталеного методу виявлення ботнетів наразі немає.

Фактично, задача зводиться до завчасного розпізнавання трафіку окремих заражених ботом мережевих пристроїв, також відомих як «зомбі» хости. Під «хостом» далі матиметься на увазі будь-який комп'ютер чи пристрій, який під'єднаний до мережі та має унікальну мережеву адресу. Життєвий цикл «зомбі» хоста можна розділити на три фази: зараження, прихована комунікація з командним центром та активна фаза атаки. Командний центр (Command and Control, скорочено C&C) це один або група мережевих вузлів, через які зловмисник здійснює координацію та контроль окремих ботів. На практиці, розпізнати бот у активній фазі, на приклад, під час DDoS-атаки, значно простіше ніж, скажім, на фазі встановлення з'єднання з C&C чи спробі зараження іншого хоста. При цьому, критично необхідно виявляти боти саме на ранніх стадіях зараження для запобігання крадіжки чутливої інформації та проведення зловмисником подальших атак.

При застосуванні конвенційних підходів до виявлення, більшість з яких базуються на методі співставлення сигнатур спостережуваних даних чи подій, інженери зіткнулися з рядом труднощів обумовлених специфікою феномену ботнетів. Оператори ботнетів застосовують різноманітні техніки ухилення від виявлення змінюючи поведінку ботів та шаблони комунікації з C&C. Розробники ботнетів мають повний доступ до заражених машин і можуть віддалено оновлювати зловмисне програмне забезпечення (ПЗ), змінюючи структуру коду та принципи його функціонування, що робить співставлення сигнатур малоефективним. Відомо також, що хакери використовують «поліморфне ПЗ», яке автономно змінює власну сигнатуру тим самим стаючи невидимим для багатьох антивірусних програм, при цьому зберігаючи свій базовий функціонал. Прикладом такого ПЗ є хробак Storm, який на момент 2007 року заразив близько 1 мільйону пристроїв і мав здатність модифікувати свою сигнатуру кожні 10-30 хвилин. Крім того, ботнети постійно еволюціонують і зловмисники регулярно створюються нові варіанти ботів на основі старих. Наприклад, бот-троян Zeus (Zbot), який вперше був

виявлений у 2007 році, з тих пір породив чимало удосконалених варіацій таких як SpyEye, Ice IX, та найбільш небезпечний – Gameover Zeus, який по різним оцінкам заразив від 500 000 до 1 мільйону пристроїв на піку своєї активності.

Все згадане обумовило потребу в нових методах виявлення шкідливих мережевих програм. Водночас, стрімке розповсюдження кіберзагроз співпало з розквітом застосування прикладних методів штучного інтелекту (ШІ). Моделі класифікації ШІ мають кілька особливостей, які роблять їх використання в даній області доцільним в порівнянні з системами, що базуються на правилах створених вручну. Останні вимагають вивчення поведінки нових зразків зловмисного ПЗ з подальшою формалізацією висновків у формі правил та алгоритмів їх співставлення. Зазвичай експерт не має доступу до вихідного коду зловмисних програм, а виконувані файли доступні лише в обфускованому вигляді. Відповідно, для вивчення їх роботи необхідно вести спостереження за подіями породженими ними, що в свою чергу, обумовлює потребу в ретельному аналізі великої кількості даних. Методи ШІ дозволяють автоматизувати та суттєво пришвидшити цей процес. Класифікатори на основі машинного навчання та нейромереж здатні самостійно виявляти комплексні залежності та шаблони в даних під час навчання. Особливо багатообіцяючим є застосування цих підходів для виявлення складних та варіативних загроз, на кшталт зараження пристроїв ботнетами.

Слід зауважити, що в комерційному секторі засобів мережевої кібербезпеки штучний інтелект активно застосовується для виявлення широкого спектру атак і загроз. Зокрема, компанія Cisco пропонує цілий портфель рішень, що інтегрують алгоритми машинного навчання. Так, Cisco Secure Network Analytics використовує телеметрію мережевих потоків NetFlow/IPFIX, а також журнали подій хмарних платформ для поведінкового аналізу трафіку. ШІ-модулі цієї системи здатні виявляти аномальні дії, спроби ексфільтрації даних та з’єднання ботнетів з C&C навіть у зашифрованому трафіку. Іншим прикладом є Cisco Umbrella, яка аналізує DNS-запити та HTTP-трафік за допомогою моделей машинного навчання для блокування доступу до шкідливих доменів, у тому числі тих, що згенеровані алгоритмами динамічного створення доменних імен (DGA), які широко

використовуються ботнетами для уникнення блокування. На жаль, такі програмні рішення є повністю пропрієтарними і мають закритий вихідний код, тому їхні архітектурні деталі не доступні для незалежного аналізу. Це також дещо ускладнює можливості їх тестування на наборах історичних мережевих даних, зібраних без їх участі.

На противагу, в системах мережевої безпеки з відкритим вихідним кодом наразі ШІ має здебільшого лише експериментальне застосування. Для прикладу, найпопулярніша система виявлення мережевих вторгнень Snort, яка фактично є індустріальним стандартом, лише в березні 2024 року вперше додала механізм виявлення на основі машинного навчання під назвою «SnortML». На момент написання цієї роботи, SnortML має лише одну модель класифікації, яка працює тільки з параметрами HTTP запитів і не здатна виявляти активність ботнетів.

В академічній сфері вже було проведено низку досліджень на тему використання ШІ для класифікації мережевого трафіку. Більшість з них однак зауважують проблему гострої нестачі навчальних даних. Крім того, жодна з відомих робіт по застосуванню нейронних мереж для виявлення мережевих загроз не адресує важливу проблему можливості цілеспрямованого ухилення від виявлення і не надає жодних формальних гарантій для даних не присутніх у навчальній вибірці.

Саме тому дана тема зберігає свою актуальність і потребує подальшого дослідження, особливо в напрямках розширення навчальних наборів даних, підвищення стійкості моделей класифікації та формальної верифікації їх властивостей.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана у відповідності з планами науково-дослідних робіт Інституту кібернетики імені В.М. Глушкова:

- В.П.100.16 «Розробити формальні методи виявлення вразливостей програмних систем»;

- В.П.100.17 «Розробити формальні методи виявлення зловмисної поведінки в мережі та хмарному оточенні на основі комбінації алгебраїчних методів та машинного навчання».

Роль автора під час виконання науково-дослідних робіт полягає в проектуванні системи виявлення ботнетів на основі аналізу мережевого трафіку моделями машинного навчання і нейромереж з застосуванням синтетичного розширення обсягу набору навчальних даних, а також в розробці та демонстрації підходу аналізу вразливості такої системи до технік ухилення від виявлення використовуючи формальні методи автоматизованого доведення.

Мета і завдання дослідження. Метою роботи є оцінювання, верифікація та підвищення стійкості систем на основі штучного інтелекту для виявлення шкідливого мережевого трафіку, зокрема ботнет-атак, до змагальних впливів з урахуванням семантики можливих дій зловмисника.

Для досягнення зазначеної мети було поставлено ряд завдань в рамках цього дослідження:

- проаналізувати існуючі підходи до виявлення мережевої активності зловмисних програм засоби штучного інтелекту, розглянути відкриті набори трафіку ботнетів та реалізувати метод їх представлення у формі придатній для обробки алгоритмами штучного інтелекту;
- розробити систему виявлення зловмисного трафіку на основі методу виділення мережевих потоків та їх класифікації засобами штучного інтелекту, експериментально виділити моделі з найбільш оптимальними показниками ефективності на доступних наборах даних;
- реалізувати метод генерації штучних змагальних прикладів мережевих потоків та здійснити доповнення ними навчальної вибірки даних для підвищення стійкості системи до можливостей ухилення від виявлення;
- формалізувати критерій локальної стійкості до збурень у вхідних даних класифікатора мережевого трафіка, через накладання обмежень на параметри нейронної мережі та розробити метод верифікації виконуваності цих обмежень за допомогою SMT-розв'язувача;

Об'єктом дослідження є процес виявлення мережевих загроз засобами штучного інтелекту.

Предметом дослідження є способи побудови систем виявлення мережевих загроз, алгоритми генерації штучних мережевих даних, методи доповнення наборів мережевих даних та методи формальної верифікації властивостей систем виявлення загроз.

Методи досліджень. В ході дослідження було використано набір теоретичних та прикладних методів, а саме: методи збору та попередньої обробки даних, статистичний аналіз даних, машинне навчання, навчання нейронних мереж, методи оцінки якості класифікаторів, генерація штучних даних, методи моделювання та формальної верифікації властивостей нейромереж.

Наукова новизна отриманих результатів. У дисертаційній роботі розв'язана актуальна дослідницька та практична задача – ***створення системи виявлення зразків трафіку ботнетів з підвищеною стійкістю до спроб навмисного ухилення від виявлення та формально верифікованими оцінками показників стійкості.***

На відміну від споріднених робіт в області аналізу мережевих даних моделями ШІ, де увага зосереджується лише на проблемі підвищення точності класифікації, в цьому досліджені вирішується актуальна науково-прикладна задача - підвищення стійкості систем виявлення до можливих «змагальних атак» направлених супроти моделі з ціллю приховування зловмисної мережевої активності. Також надаються оцінки стійкості класифікатора потоків мережевих даних до атак такого роду, перевірені за допомогою формальних методів верифікації.

Під час проведення цього дослідження було отримано ряд важливих результатів:

1. Набув подальшого розвитку метод генерації штучних прикладів та доповнення наборів даних шляхом адаптації швидкого методу знаку градієнту до простору ознак мережевих потоків, з метою оцінки та підвищення стійкості систем

виявлення шкідливого трафіку до змагальних атак з урахуванням семантики можливих дій зловмисника.

2. Вперше розроблено універсальний метод формальної верифікації властивостей глибоких нейронних мереж з кусково-лінійними функціями активації, який базується на представленні обчислювального графу нейромережі у формі спрощеної SMT-формули, побудованої інкрементально шляхом розв’язуванням локальних SMT-задач сформованих для функцій активації нейронів та перевірки можливості їх тотожної заміни на лінійні функції.

3. Вперше формалізовано критерій локальної стійкості класифікатора мережевого трафіка та здійснено його верифікацію шляхом автоматичного доведення (чи спростування) виконуваності накладених на модель обмежень за допомогою SMT-розв'язувача, що дозволило достовірно оцінити стійкість створеної системи виявлення загроз до можливих збурень у вхідних даних.

В ході проведення досліджень автором дисертації була розроблена програма-прототип, здатна в режимі онлайн перехоплювати дані на мережевому інтерфейсі хоста, виділяти з них окремі потоки трафіку та проводити їх аналіз моделями ШІ з метою виявлення слідів мережевої активності ботів та іншого зловмисного програмного забезпечення.

Практичне значення отриманих результатів продемонстровано у формі прототипу багатоцільового аналізатора мережевого трафіку. Його вихідний код та детальні інструкцій до використання опубліковані на ресурсі Github. Також було зібрано та опубліковано виконувані файли програми готові для запуску на Windows та Linux. Екземпляр даної програми було впроваджено в експлуатацію у ролі системи виявлення мережевих загроз у компанії ТОВ «НВП «Радікс», що спеціалізується на розробці апаратного та програмного забезпечення для АЕС (див. Додаток Б).

Окрім засобу виявлення загроз, створена програма також може бути використана як інструмент агрегації мережевих даних для більш високорівневого їх представлення у формі окремих мережевих потоків. Виділені числові та категоріальні атрибути потоків мають значно менший відбиток у пам’яті і є більш

придатними для подальшого аналізу статистичними алгоритмами та моделями ШІ незалежно від цільової задачі. Такий функціонал може мати практичну цінність для інших дослідників в споріднених областях, адже дозволяє:

- Здійснювати агрегацію власних наборів мережевих даних у режимах офлайн та онлайн.
- Будувати нові набори даних на основі трафіку перехопленого у лабораторних умовах без необхідності збереження та транспортування самих мережевих пакетів за межі хоста.

В ході роботи проводилось навчання класифікаторів мережевого трафіку на базі алгоритмів машинного навчання та нейронних мереж з використанням різних наборів мережевих даних. Для навчання та тестування було створено розширену вибірку навчальних даних на основі комбінації різних відкритих наборів, що дозволило підняти показник повноти виявлення TCP-трафіку зловмисного ПЗ з 46% до 80.7% без погіршення значення хибнопозитивного рівня. Зокрема, отримані моделі здатні виявляти трафік таких зловмисних програм як TrickBot, Emotet та Wannacry з повнотою >80%. Крім того, навчання на наборі доповненому штучними прикладами дозволило знизити показник вразливості нейронної мережі до змагальних атак з 8.5% до 2% за умови варіювання розмірів пакетів в межах до 30% від оригінальних значень прикладів з тестового набору.

Також, в даному дослідженні був запропонований метод автоматизованого доведення локальних властивостей повнозв’язних нейромереж з кусково-лінійними функціями активації. Цей метод є універсальний і може мати практичне застосування для задач формальної верифікації за межами кібербезпеки.

Особистий внесок здобувача. Основні наукові результати дисертаційної роботи отримані автором самостійно, а у публікаціях внесок здобувача є наступним:

[1] – здобувачем описано метод виділення потоків з мережевих даних та проведено ряд експериментів по їх класифікації різними моделями ШІ на предмет виявлення слідів активності ботнетів. На основі цього здобувачем проведена

порівняльна характеристика та сформовані висновки щодо ефективності застосування такої методики для виявлення мережевих загроз.

[2] – здобувачем запропоновано та реалізовано підхід по оцінці стійкості до змагальних атак систем виявлення мережевих загроз на основі нейромереж. Здобувачем описано метод підвищення стійкості таких систем шляхом штучної генерації навчальних даних та експериментально показано його ефективність в контексті можливих спроб ухилення від виявлення.

[3] – здобувачем формалізовані критерії надійності системи виявлення мережевих загроз на основі потоків. Здобувачем було запропоновано метод формальної верифікації повнозв'язних нейромереж і продемонстровано можливість його застосування для перевірки сформованих критеріїв стійкості класифікаторів мережевих даних.

[4] – здобувачем розроблена методологія по перевірці критеріїв стійкості нейромереж-класифікаторів мережевого трафіку до змагальних атак, і проведено ряд експериментів по підвищенню їх надійності. Співавтором *О. Летичевським* сформовані формальні критерії та описана методика підвищення надійності нейромереж на основі нейросимвольного підходу.

[5] – здобувачем експериментально показана можливість підвищення ефективності виявлення трафіку ботнетів шляхом організації історичних мережевих даних у часові ряди і їх подальшої класифікації рекурентними та згортковими нейромережами.

Апробація результатів дисертації. Результати дослідження презентувалися на науковій конференції 13th International Scientific and Practical Programming Conference - UkrPROG'2022.

Публікації. Основні положення дисертаційної роботи опубліковано в 4 статтях у наукових фахових виданнях, 2 з яких належать до категорії А. Також презентовані результати включені до збірника матеріалів 1 міжнародної конференції.

Структура і обсяг дисертації. Дисертаційна робота складається із анотації, вступу, чотирьох розділів, висновків, списку використаних джерел із 106

найменування. Основний текст роботи викладено на 128 сторінках та 5 додатках на 10 сторінках. Повний обсяг дисертації становить 167 сторінок, в тому числі 24 таблиці та 24 рисунка.

РОЗДІЛ 1. МЕТОДИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК ТА ЗАГРОЗИ БОТНЕТІВ

1.1. Вступ

1.1.1. Мережеві загрози та системи протидії

У середовищі загально розповсюдженої інтеграції систем в мережу Інтернет мережеві атаки стали звичайним явищем. Звичайні засоби захисту, як-от ретельно налаштовані правила брандмауера та антивірусні програми на основі сигнатур, хоч і корисні, однак не охоплюють весь простір можливих атак. Безліч атак проводяться через загально-відомі відкриті порти і маскуються під звичайну мережеву активність. Таку активність неможливо виявити, використовуючи лише статично орієнтовані механізми безпеки, такі як брандмауери та методи співставлення сигнатур. Системи виявлення вторгнень (IDS, Intrusion Detection Systems) — це динамічна лінія захисту, яка призначена для виявлення шкідливої діяльності програм під час їх виконання. У рамках цього дослідження розглянуто способи створення та методи покращення ефективності та стійкості систем виявлення мережевих загроз на основі алгоритмів штучного інтелекту. Щоб краще зрозуміти переваги використання ШІ в даній області, варто розглянути інші поширені альтернативи.

Ерве Дебар у 2009 році [1] дав огляд різних типів IDS та їх таксономію, розділивши їх за способами виявлення на базі знань і на базі поведінки.

Системи з виявленням на базі знань (також відомі як «засновані на правилах») вважаються класичним підходом. Вони використовують набір заздалегідь визначених правил, створених людиною-експертом, що описують відомі моделі атак і порівнюють їх із зафіксованими подіями. Події можуть мати різні представлення: рядки журналів роботи застосунку («логи»), аудиту системи, звичайні мережеві пакети та інші. Прикладом такого правила може бути *«сталося більше ніж N невдалих спроб підключення з IP-адреси X»*. Після спрацювання якогось з правил система генерує сповіщення і потенційно виконує автоматичну відповідь (наприклад, блокує шкідливу IP-адресу). У такого підходу є кілька

недоліків. По-перше, для створення правил і подальшого оновлення їх списку потрібна людина-експерт. По-друге, побудова комплексних правил може бути надзвичайно складною, оскільки моделі атаки не завжди можна легко описати. Зловмисники часто навмисно розтягують втручання в часі, надсилаючи шкідливі пакети нерегулярно, а часто вперемішку з доброякісними даними. Врешті решт, як випливає з назви, системи, що базуються на знаннях, вимагають спеціальних попередніх відомостей про атаку для її виявлення.

Системи, засновані на поведінці, натомість, є більш гнучкими, оскільки призначені для «вивчення» поведінки користувача та створення так-званого «профілю нормальної активності». Відповідно, події, що вибиваються з «нормального» шаблону активності (аномалії), класифікуються як вторгнення. На жаль, незважаючи на вищу гнучкість, складність цього підходу полягає у складності самого процесу визначення «нормальної поведінки». Звичайні користувачі можуть з часом змінювати шаблони своєї звичайної поведінки, і важко визначити, в якій мірі ця зміна прийнятна, перш ніж вважати її аномалією.

1.1.2. Застосування ШІ для виявлення вторгнень

Системи, керовані штучним інтелектом, є альтернативою, яка виглядає найбільш перспективно, враховуючи швидкий розвиток ШІ за останнє десятиліття. Перевага штучного інтелекту полягає в тому, що він містить сильні сторони як систем заснованих на знаннях, так і систем на базі вивчення поведінки, при цьому потребує значно менше зусиль від інженера-розробника для опису та налаштування. Моделям ШІ не потрібні люди-експерти, оскільки вони самі вивчають правила та профілі з існуючих даних. Наприклад, дерева рішень та нейронні мережі можуть в ході тренування виводити значно складніші набори правил у порівнянні з системами виявлення на базі знань. Моделі машинного навчання «з учителем» вивчають нормальний профіль активності користувача одночасно з вивченням шаблонів самих атак, а моделі «без учителя» можуть самостійно навчитися відрізняти аномальний трафік, що робить їх досконалішими за системи виявлення на базі поведінки.

Однак, системи, керовані ШІ, мають два основних недоліки: по-перше, виведені ними правила часто не піддаються інтерпретації людиною; по-друге, потрібен достатній набір навчальних даних – у випадку ж навчання моделей «з учителем» (supervised learning) набір даних має бути ще й розміченим (тобто, там мають міститися не лише вхідні дані, а й вірний очікуваний вихідний результат).

Дослідження показали, що якість IDS на основі штучного інтелекту значно залежить від методу виділення ознак, а також від вибору базової архітектури класифікатора. У роботі під редакцією Аль-Сакіба Хана Патана у 2014 році [2] коротко описані як переваги, так і недоліки використання різних методів ШІ для виявлення. Там згадуються: глибокі нейронні мережі (DNN), самоорганізаційні карти Кохонена, моделі Маркова, байєсівські системи та машини опорних векторів (SVM). Використання більш складних архітектур нейромереж описано в роботі Arnaldo та ін. 2017 [3] де окрім випадкових лісів (RF) та звичайних мереж із прямим зв’язком (FFNN) вдало застосовані також згорткові (CNN) та рекурентні нейронні мережі (RNN) для аналізу даних, представлених у вигляді часових рядів. Безліч інших робіт також демонструють перспективи застосування машинного навчання та нейронних мереж в даній області.

1.1.3. Предметна область та рамки цієї роботи

В рамках цього дослідження головна увага зосереджується на виявленні саме трафіку ботнетів. Ботнети є одними з найбільш руйнівних загроз кібербезпеки, і їх кількість щороку активно зростає. На відміну від інших атак, ботнети створюються так, щоб залишатися прихованими протягом тривалого періоду часу – чим довше інфекція залишається невиявленою, тим більше шкоди завдається у вигляді витоку даних, зловмисного спостереження та мимовільної участі зараженої машини в інших мережевих атаки. Саме тому завчасне виявлення програм ботів є критично важливим для будь-якої системи інформаційної безпеки.

Слід зазначити, що, хоча фокус дослідження і зроблений на ботнетах, однак спеціалізація загальної схеми проводиться лише на етапах виділення числових характеристик з мережевого трафіку, а також вибору наборів даних, що

використовується для навчання моделей ШІ. Загальну ж систему виявлення можна застосувати і для інших видів мережевих атак, а відмінність лиш буде у навчальній вибірці та обраній множині мережевих характеристик. Деякі методи, які застосовують для виявлення мережевих атак іншого роду можуть бути використані для виявлення ботнетів і навпаки.

1.2. Огляд області, існуючих методів та споріднених досліджень

1.2.1. Класифікація мережевого трафіку з метою виявлення загроз

Ранні дослідження по класифікації спостережуваного мережевого трафіку проводилися ще на початку 2000-х років [4], з метою отримання необхідної інформації для налаштування мережі та покращення якості обслуговування. Серед проведених досліджень були спроби класифікувати мережеві дані шляхом аналізу корисного навантаження пакетів, однак швидке розповсюдження TLS (Transport Layer Security) шифрування часто зводило нанівець такий підхід. Саме тому замість глибокого аналізу пакетів, дослідники все частіше схилялись до аналізу більш високорівневої, узагальненої інформації про потоки даних [5], використовуючи мережеву статистику, а також не зашифровану інформацію доступну із заголовків пакетів мережевого та транспортного рівнів. Класифікація часто проводилась на основі сигнатур - наборів заздалегідь відомих правил, або статистичних моделей, які якомога точніше описували би шаблони поведінки даних, що спостерігаються [6][7].

Хоча методи штучного інтелекту тоді і не були настільки широко розповсюдженими як зараз, в ті роки все ж були спроби застосування підходів машинного навчання та видобутку даних (data mining), для класифікації мережевого трафіку, зокрема техніками Байєсівського аналізу. [8].

Безпосередньо у сфері кібербезпеки, однак, ШІ в ті часи застосовувався нечасто, і обмежувався задачами на зразок аналізу поштових повідомлень на предмет спаму. Для задачі протидії мережевим атакам здебільшого використовувались системи виявлення вторгнень на основі сигнатур, правил та знаходження аномалій без використання ШІ. Як приклади можна згадати такі

системи як "Bro" (сучасна назва "Zeek" [9]) розроблену в 1995, а також Snort [10] створену в 1998. Обидві широко використовуються до сьогодні, а Snort фактично стала індустріальним стандартом в області NIDS (Network Intrusion Detection Systems).

З технічним прогресом в інформатиці та все більшим розповсюдженням інтернету, інформаційні загрози також росли та розвивалися. Мережеві атаки ставали все більш складними і невдовзі стало зрозуміло, що конвенційні методи виявлення не здатні в повній мірі протистояти їм. Однією із таких нових на той час загроз стали ботнети (бот-мережі). Особливістю ботнетів стало те, що їх поведінка, а відповідно і мережевий трафік, який вони створюють, часто були дуже схожі на поведінку звичайних користувачів чи програм. Концепція ботентів виникла ще в кінці 80-х років зі створенням протоколу IRC (Internet Relay Chat), який найперші мережи ботів використовували для комунікації. Як і слідує з назви, IRC початково був призначений для ведення розмов між людьми: безліч користувачів під’єднуються до єдиного IRC-серверу, де підписуються на IRC-канали, куди можуть писати свої повідомлення і бачити повідомлення всіх інших підписаних на канал користувачів. Розробники ботнетів використовували описаний функціонал для координації мереж ботів. На кожному віддаленому комп'ютері запускалася програма, яка підписувалась на канали приватного (а інколи й публічного) IRC-серверу, звідки отримували інструкції від власника ботнету, чи передавала йому інформацію з підконтрольної машини. IRC-боти як правило мають щонайменше 2 канали: один командний, а другий канал статусу ботів. Коли бот під’єднується до мережі, він оповіщає через канал статусу "центр команд та контролю" (C&C - Command and Control), про доступність нової виконавчої одиниці. В подальшому, бот час від часу оновлює свій статус, таким чином центральний контрольний вузол тримається в курсі всіх активних ботів і може надсилати їм спільні чи індивідуальні команди через командний канал. Ідея розподілу ролей на "виконавчі вузли" (боти) та "центр команд та контролю" зберіглася і до сьогодні, хоча структура C&C значно ускладнилась через використання кластерів серверів та децентралізованого керування.

На відміну від сьогодення, перші ботнети не були шкідливими, а використовувались здебільшого для того, щоб тримати сервери "зайнятими" для запобігання відключенню, а також для віддаленого автоматизованого запуску завдань.

Найпершим виявленим злоякісним ботнетом вважається Pretty Park [11], масоване зараження яким мало місце в 1999 та 2000 роках. Цей ботнет розповсюджувався як поштовий хробак, а для координації та виводу інформації з системи-жертви використовував IRC-канали. IRC ботнети і далі набирали популярність і в середині початку 2000-х років стали значною проблемою. На момент 2007 року, експерти оцінювали, що від 16 до 25% комп'ютерів, під’єднаних до мережі, вже були частиною якогось ботнету [12].

У випадку більш класичних мережевих загроз, на зразок DoS (Denial of Service), часто вже є заздалегідь відомі сценарії атаки. Відповідно можна передбачити очікувану структуру мережевого трафіку та формалізувати набір правил для виявлення таких випадків. Саме такий підхід здебільшого використовувався в класичних NIDS (Network Intrusion Detection Systems). У випадку ботнетів однак, як виявилось, задача відрізнення зашифрованої комунікації ботів від легітимного користувацького IRC трафіку є зовсім нетривіальною. Методи ШІ є ідеальним кандидатом для рішення задач класифікації, в яких складно формалізувати алгоритми чи конкретні правила, за умови, якщо є достатня кількість даних. Саме тому, дослідження застосування, зокрема, машинного навчання стало наступним логічним кроком.

Хоча ботнети і є найбільш наглядним прикладом, чому застосування ШІ в цій області стало перспективним, однак потенціал такого підходу звісно не обмежився лише ботнетами. Згодом, було проведено чимало досліджень по виявленню й інших різноманітних мережевих атак.

1.2.2. Історія ранніх досліджень по виявленню ботнетів з використанням штучного інтелекту.

Однією із найперших робіт по застосуванню машинного навчання для аналізу мережевого трафіку на предмет виявлення ботнетів є стаття Lividas та ін. 2006 року [13]. Ця робота є досить фундаментальною в цій області, адже автори зіткнулися з багатьма проблемами, з якими продовжили стикатися їх наступники два десятиліття потому. По-перше, для машинного навчання необхідна достатня кількість навчальних та тестувальних даних. Так як на той час зовсім не було відкритих навчальних наборів, автори були змушені формувати вибірку самостійно. У ролі звичайних (доброякісних) даних були використаний анонімізований мережевий трафік зібраний з університетського кампусу, значний відсоток якого складався з IRC трафіку. Дані розмічались на основі портів за замовчуванням (6667 та 6697), які використовується за замовчуванням для даного протоколу. Для генерації тестової вибірки автори в лабораторних запустили знешкоджену версію реального ботнету Kaiten. В роботі фокус робиться саме на ботнетах на базі IRC, відповідно спроба виявлення робиться для трафіку саме цього протоколу. Також вважається, що усі пакети зашифровано, а для підключення ботнет використовує довільний порт. Відповідно, задача виявлення розділена на 2 стадії. На першій стадії, класифікатор намагається відрізнити потоки IRC від трафіку інших протоколів. На другій стадії система намагається виділити ті IRC потоки, які є частиною комунікації ботів з іншими операційними компонентами ботнету. Для виокремлення IRC потоків автори застосовують три моделі класифікації, а саме наївний Баєс, Баєсівські мережі та дерево прийняття рішень J48. Розглянутий в роботі [13] набір ознак потоків включає ознак, які згодом зустрічатимуться і в інших роботах по класифікації мережевих потоків, а саме тривалість потоку, середню кількість пакетів та бітів, середнє значення та дисперсію байтів на пакет, дисперсію довжини міжпакетних інтервалів, кількість пакетів з керуючим бітом PSH, а також гістограму розмірів пакетів. Також в роботі робляться експерименти з більш розумним вибором ознак, зокрема шляхом оцінки важливості ознаки на основі її висоти в дереві прийняття рішень. Цікаво зауважити, що видаливши лише одну ознаку, яка відображала розмір TCP вікна, авторам вдалося зменшити відсоток хибно-негативних результатів (FNR) з 68.4% до 7.9%.

Це є наглядним прикладом того, як різниця між умовами циркуляції трафіку в різних середовищах може звести нанівець коректність роботи класифікатора, особливо якщо модель надає перевагу ознакам, які повністю залежать від умов мережевого середовища. В згаданій роботі наприклад, дані навчального середовища збирались переважно з машини на основі Windows, тоді як в тестовому середовищі був Linux, що спричиняло відхилення в значеннях згаданого параметра і негативно впливало на модель. Після успішного виявлення IRC трафіку в першій стадії, в другій стадії необхідно безпосередня відрізнити ботнети від нормального трафіку, однак, цей крок автори залишили до розгляду у своїй наступній роботі.

Робота Strayer et al. 2007 [14] є логічним продовженням роботи [13]. В ній розвивається ідея виявлення комунікаційних потоків IRC ботнетів шляхом класифікації мережевого трафіку. Як вже зазначалось, виявлення ботнету розділено на 2 стадії. Ціллю першої стадії є максимально скоротити об'єм трафіку, який необхідно аналізувати. На цій стадії за допомогою машинного навчання виділялися ті мережеві потоки, які ймовірніше за все використовують протокол IRC. Також застосовувались різноманітні логічно-обґрунтовані фільтри на зразок відсіювання потоків без вдалого встановлення TCP з'єднання, потоків в яких здійснювалось завантаження великих об'ємів даних і так далі. Таким чином, автори скоротили початкову вибірку в 37 разів. На другій стадії, над залишком множини потоків проводиться пошук "корелюючих пар". Автори висувають гіпотезу, що якщо в мережі, за якою здійснюється спостереження, наявні кілька машин заражених ботнетом, то локалізований у часі трафік породжуваний цими машинами буде корелювати. Кореляцію визначається через міру подібності двох потоків, яка в свою чергу обраховується як нормована різниця у відстані між двома точками в n-вимірному просторі ознак потоків. Ознаки при цьому представлені як вектори першого та другого моментів випадкової величини на основі розподілів значень характеристик потоків, які вже згадувались у минулий роботі. Для ефективної обробки пакетів у режимі реального часу при оцінці моментів автори пропонують використовувати зважене плаваюче середнє та зважену дисперсію - тобто з кожним новим пакетом їх значення оновляються враховуючи минуле. Як і

очікувалось, в результаті було виявлена кореляція (схожість моментів розподілів) потоків всіх ботів у локальній нейромережі, що є показовим результатом. Метод запропонований в описані роботі однак має кілька суттєвих слабкостей та недоліків. По-перше, такий підхід може працювати, лише у випадку якщо в спостережуваній мережі є кілька заражених машин одночасно і обидві отримують та виконують однакові команди від центру командування та контролю. Наприклад, було продемонстровано, що системі не вдалося виявити бот, який знаходиться за межами локальної мережі, адже його трафік суттєво відрізняється. По-друге, даний підхід є суто евристичним, і засновується на припущенні, що серед множини усіх потоків корелювати будуть лише потоки ботнету. Це очевидно не завжди є вірним, адже на приклад в системі, де присутні, скажім, проксі-сервери, які реплікують запити, або ж віддалені застосунку налаштовані на синхронну роботу (наприклад, задачі, що виконуються за часовим розпорядком), очікувано також будуть з'являтися корелюючі мережеві потоки локалізовані в часі. Тим не менше, роботи [13] та [14] стали гарним фундаментом для подальших досліджень у області, адже вони вдало обходять проблему нестачі навчальних та тестових даних, а також застосовують концепцію "мережевого потоку" для аналізу трафіку, при цьому висвітлюючи логіку та проблеми вибору характеристичних ознак корисних для виявлення ботнетів.

1.2.3. Сучасні дослідження по виявленню ботнетів моделями ШІ.

Наразі вже написано чимало робіт оглядового характеру, на тему сучасних стратегій та підходів до виявлення ботнетів моделями штучного інтелекту, які запропоновані різними дослідниками. Наприклад, в роботах [15] та [16] неведено таксономію та порівняння різних споріднених досліджень в даній області. Автори розділяють методи аналізу трафіку зловмисного ПЗ за напрямками як-от: методи на основі статистичного аналізу, методи комплексного аналізу мережевих графів, методи на основі ройового інтелекту та їх комбінації. При цьому в переважній більшості з них також використовуються класифікатори машинного навчання, як-от машини опорних векторів, алгоритми кластеризації та навчання з підкріпленням.

Одним з найбільш визначних досліджень є робота Zhao et al. 2013 [17], де автори розвивають ідею виявлення ботнетів на основі аналізу потоків мережевого трафіку моделями штучного інтелекту. В більшості попередніх робіт шкідливі потоки виділялись на основі міжпотокових кореляції та кластеризації взаємопов'язаних подій. В цій роботі, однак, класифікатор ШІ використовується безпосередньо для визначення кінцевого класу потоку: шкідливий чи звичайний. Також змінюється базова одиниця класифікації на більш атомарну. До цього при аналізі трафіку зазвичай розглядалися вся сукупність пакетів, що циркулювали між парою віддалених IP адрес та портів за весь час спостереження. В цій роботі потоки розділяються на обмежені часові інтервали і різноманітні статистики ознак обчислюються в межах кожного інтервалу. Фактично це дозволяє збільшити "розподільну здатність" деталізації спостережень, адже на різних інтервалах поведінка трафіку потоку може суттєво змінюватись. Зроблено спостереження, що ботнети зі схожим механізмом комунікації демонструють схожі шаблони мережевого трафіку. Наприклад, при встановлені початкового з'єднання з центром команд та контролю, розміри першої групи пакетів та міжпакетні інтервали можуть бути суттєво схожі як в межах одного ботнету так і різних. Інформація такого роду релевантна на весь час спостереження за потоком, а тому при класифікації наступних інтервалів автори пропонують додавати й ті ознаки, які обраховані для першого. Із використаних авторами ознак потоків варто відзначити евристично-обумовлені як-от: розмір першого пакету, кількість перевстановлення TCP з'єднання, та відношення кількості потоків з даної IP адреси до кількості всіх потоків за годину. У якості класифікатора використовується дерево прийняття рішень. У ролі навчальної вибірки використовується об'єднання кількох наборів даних, зокрема французький розділ «The Honeypot Project» [18] в якому присутній трафік 2 ботнетів: Storm та Waledac. Обидва ботнети використовують HTTP трафік для комунікації та операцій. Авторам вдалося досягнути точності виявлення 98.3%, тоді як позитивно-негативні результати склали лише 0.01%. Також в роботі досліджується можливість виявлення новітніх ботнетів. Автори показали, що така система може виявляти з дуже високою точністю деякі ботнети, які також

використовують протокол HTTP, а саме Weasel та BlackEnergy, навіть якщо модель не була знайома з ними при навчанні. Водночас, однак, автори помічають, що при додаванні у тестовий набір нормального (нешкідливого) трафіку який з якихось причин нагадує шкідливий, відсоток позитивно-негативних результатів катастрофічно зростає. Як правило, ця проблема адресується через додавання в навчальну вибірку звичайного трафіку перехопленого безпосередньо з оточення у якому в подальшому буде функціонувати система виявлення, що однак негативно позначається на портативності такого рішення.

Особливо значний внесок у розвиток методології виявлення ботнетів привносить робота Beigi et al. 2014 [19]. Автори поставили за ціль виокремити універсальний набір ознак мережевих потоків який би підходив для виявлення більш широкого спектру можливих ботнетів, а не лише для кількох екземплярів із навчальної вибірки. Для цього, автори зібрали та систематизували різні ознаки мережевих потоків, які використовувались до цього в різних роботах по виявленню ботнетів. 16 початково обраних ознак було розділено на 4 категорії:

1. характеристики розмірів пакетів (в байтах), як-от сумарна кількість байт, середня величина корисного навантаження і так далі.
2. статистики пакетів у потоці: наприклад, відношення кількості вхідних до вихідних пакетів, кількість та відсоток малих пакетів та пакетів без корисного навантаження.
3. часові характеристики, такі як кількість пакетів та байтів за секунду, статистики міжпакетних інтервалів і тому подібне.
4. поведінкові ознаки, як-от кількість спроб повторного встановлення з’єднання чи розмір першого пакету в потоці.

Після цього автори застосували метод ітеративного покращення точності класифікатора через поступове вилучення ознак. На кожному кроці, прибиралась та група ознак, вилучення якої сприяло приросту точності класифікації, а далі за тим самим принципом, одна з вилучених ознак групи повертається назад у набір. В результаті, автори показали, що комбінація з 4 ознак, а саме "тривалість потоку", "середній розмір корисного навантаження", "відношення вхідних до вихідних

пакетів" та "кількість біт на секунду", дають найкращий результат, який склав 75% точності класифікації. У ролі класифікатора використовувалося дерево прийняття рішень на основі алгоритму C4.5. Варто зауважити, що показник точність отриманий у роботі Beigi et al. 2014 [19] виглядає значно нижчим за показники точності наведені авторами у попередніх роботах. Це пояснюється тим, що на відміну від попередньо згаданих робіт [13][14][17], у яких модель навчалася та тестувалася супроти 1-2 типів ботнетів, в даній роботі модель навчалася на наборі даних зі слідами активності 7 різних ботнетів, а тестова вибірка містила аж 16. Для отримання такої репрезентативної вибірки, автори скомбінували дані з кількох доступних наборів, за допомогою методу накладання. Результуючий набір даних став відомий як ISCX Botnet 2014 [20] і у подальшому використовувався іншими авторами ще в багатьох роботах на дану тематику.

Наступною роботою, як значно вплинула на розвиток даного напрямку є праця Arnaldo et al. 2017 [3], що вже була згадана. Автори ставлять за мету створення системи виявлення активності ботнетів шляхом представлення потоків як часових рядів. В роботі проводяться два окремих експерименти: аналіз послідовностей записів в журналі подій системи та аналіз послідовностей потоків мережевого трафіку. Для експерименту з мережевими потоками, автори використали вищезгаданий відкритий набір даних ISCX Botnet 2014[20], з якого виділили 23 ознаки за допомогою (на сьогодні вже застарілої) версії інструменту FlowMeter[21]. Дана робота розвиває метод аналізу мережевих потоків, шляхом їх організації у впорядковані за часом серії і демонструючи спосіб інтерпретації ознак таких часових рядів класифікаторами різних архітектур, як-от випадковий ліс, повнозв'язна глибока, рекурентна чи згорткова нейромережа. У випадку перших двох, автори пропонують конкатенацію ознак фіксованої кількості потоків ряду в єдиний вектор ознак. Після чого застосовуються методи зменшення розмірності, зокрема метод головних компонент (PCA - principal component analysis). У випадку згорткових нейромереж, в роботі використовуються темпоральні (одновимірні) згортки. В залежності від довжини ряду, показники AUROC класифікаторів

наведених для ISCX Botnet 2014 варіюються від 0.449 до 0.808 (для ряду довжиною в 28 потоків).

На основі описаних фундаментальних досліджень почав формуватися більш усталений підхід до побудови систем виявлення ботнетів на основі методі мережевих потоків та моделей штучного інтелекту. Зокрема його варіації зустрічаються у схожих роботах [22][23][24][25], які демонструють застосування стандартних моделей машинного навчання, таких як дерево прийняття рішень, машини опорних векторів та логістичної регресії, для класифікації окремих мережевих потоків виділених з добре відомих відкритих наборів трафіку ботнетів. Робота [81] є цікава тим, що робить фокус на продуктивності використання такого підходу в системах реального часу.

В роботі Pektaş et al. 2018 [26] автори застосовували методи глибокого навчання для класифікації статистичних характеристик комунікацій віддалених процесів. Схеми таких комунікацій були змодельовані через мережеві графи, на основі чого підраховувались статистики потоків протоколів UDP, TCP та ICMP між різними хостами. Автори стверджують, що оцінювана точність виявлення трафіку ботнетів складає майже 99% для наборів ISOT[27] та CTU-13[18].

Окремої та особливої уваги потребує питання захисту систем «Інтернету Речей» (IoT, Internet Of Things), які набули стрімкого розповсюдження за останнє десятиліття і сформували окрему технологічну парадигму [28]. Системи IoT – представляють з себе розподілені комплекси датчиків та інших вузько спеціалізованих апаратних засобів тісно пов’язані через мережу Інтернет. Вони дозволяють інтегрувати різні віддалені пристрої, підвищувати ефективність управління розподіленими ресурсами в реальному часі та створювати інтелектуальні рішення в різних сферах, таких як розумні будинки, міста, промисловість і охорона здоров’я. Цілком очікувано, що такі системи постійно стають ціллю різноманітних кібератак. Найбільш відомим прикладом став спалах масового зараження IoT-пристроїв ботнетом Mirai в 2016 [29] варіації якого широко розповсюджені до сьогодні. Для побудови систем виявлення вторгнень для інтернету речей необхідно враховувати специфіку роботи пристроїв даної області.

Наприклад, протоколи комунікації, такі як 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) та RPL (Routing Protocol for Low-Power and Lossy Networks), створені для енергоефективної передачі даних по бездротових каналах зв'язку і широко використовуються системами IoT є вразливим до атак IPv6 маршрутизації [30]. Крім того IoT пристрої, мають обмежений обчислювальний ресурс, що не дозволяє запускати на них ресурсоємкі програми безпеки.

В роботах [31] та [32] було проведено огляд та порівняння існуючих підходів побудови IDS для IoT, в тому числі в них розглянуто приклади характерних атак та застосування моделей машинного навчання для їх виявлення. Більш глибоке дослідження проведено в роботі [33], де автори пропонують використовувати парадигму «кордонних обчислень» (edge computing), а також рекурентні та згорткові нейромережі. В роботі Sriram et al. [34] автори показали, що можливо використовувати вже раніше загаданий підхід аналізу мережевих потоків для виявлення ботнетів в середовищі IoT методами машинного навчання. В роботі [35] демонструється можливість «інкрементального навчання» моделей класифікації трафіку IoT з використанням дерев прийняття рішень, а також алгоритму відбору ознак за допомогою критерію Фішер. В роботі [36] використовується алгоритм розумного відбору ознак за допомогою «оптимізуючого алгоритму зебри» (Zebra Optimization Algorithm [37]) в комбінації з двоканальною графовою мережею (Dual-channel Graph Attention Network) для вивчення особливостей комунікації пристроїв інтернету речей та виявлення аномалій, таких як ботнети.

Для навчання моделей ШІ систем виявлення мережевих загроз необхідні набори даних з прикладами відповідного зловмисного трафіку. Одним зі способів отримання мережевих даних такого роду є «пастки» (honeypots), які виконують роль приманки для зловмисного ПЗ з ціллю вивчення особливостей його роботи для можливості розробки методів протидії. В роботах [38] та [39] згаданий метод використовувався у контексті розробки систем безпеки IoT на базі машинного навчання.

Наразі дослідниками вже сформовано кілька відкритих навчальних та оцінювальних вибірок специфічних для сфери інтернету речей. Зокрема IoT-23 [40]

трафік якого був перехоплений в рамках Stratosphere Laboratory Університету Чехії, та CIC IoT-DIAD 2024 [41] отриманий Канадським Інститутом Кібербезпеки. В наборах мережевих даних часто зустрічається проблема наявності «шумів», – несуттєвих даних, специфічних для конкретної мережі, які негативно впливають на процес навчання моделей ШІ. В роботі [42] автори відфільтровують такі шуми шляхом застосування таких алгоритмів як RENN (Repeated Edited Nearest Neighbor) для обробки прикладів трафіку IoT ботнетів. В роботі [43] пропонується альтернативний підхід, де фокус робиться на виявленні доменних імен керуючих центрів ботнетів на основі зразків трафіку перехоплених в реальному мережевому середовищі.

В підсумку зауважимо, що виявлення ботнетів та інших загроз в середовищі інтернету речей є досить новими напрямком і дана галузь потребує подальших досліджень.

1.2.4. Розвиток методів класифікації потоків

З розвитком технологій обробки мережевих даних зростала й кількість різних варіантів наборів числових характеристик, но основі яких проводиться аналіз потоків. Відповідно, з’явилась необхідність у методах зменшення об’ємів даних та скороченні розмірності простору ознак з якими працюють класифікатори при цьому без втрати їх якості.

В роботі Ullah Khan та ін. 2019 [44] пропонується техніка багаторівневого виявлення P2P (peer-to-peer) ботнетів. Автори пропонують окремо розділяти P2P та не P2P трафік. При цьому з останнього виділяється лише контролюючі TCP пакети, що відповідають за встановлення та завершення з’єднання, що допомагає суттєво знизити об’єм даних, які підлягають аналізу. В роботі застосовується фільтрація пакетів на основі добре відомих мережевих портів та DNS запитів. Також у роботі пропонується звужувати набір ознак класифікації, які менш суттєво впливають на можливість виявлення трафіку ботнетів.

Farhan та ін. 2019 [45] зробили спробу підвищити ефективність стандартного методу класифікації окремих мережевих потоків шляхом застосування різних

методів розумного вибору ознак потоків виділених з набору даних ISCX Botnet 2014[20]. В роботі порівнюються показники точності різних алгоритмів машинного навчання при виборі найкращих 5, 10 та 15 ознак із 80 початкових. Відбір ознак здійснюється за допомогою методу дисперсійного аналізу, методу мінімізації міжознакової взаємної інформації та методу головних компонент (PCA). Також, проводиться порівняння з набором ознак рекомендованих найбільшою кількістю літературних джерел. Для кожного набору ознак автори підрахували показники точності для різних алгоритмів класифікації, а саме: метод k-найближчих сусідів (KNN), логістична регресія, дерево прийняття рішень, випадковий ліс, наївний Баєсівський класифікатор та машина опорних векторів. Авторам вдалося досягти 81% точності на тестовому наборі даних для випадкових лісів при обранні 15 ознак на базі критерію взаємної інформації. На жаль, автори не наводять інших показників якості (як-от F1 чи AUROC).

Ще одну спробу застосування алгоритмів розумного вибору ознак мережевого трафіку у контексті виявлення атак штучним інтелектом роблять в статті 2023 р. Чичкарьов та ін. [46]. Автори застосовують нечіткі логіки, для ранжування значущості ознак та вибору найбільш суттєвих на основі F-статистики в рамках дисперсійного аналізу, методу спільної інформації, та MDI (mean decrease in impurity). Аналіз проводиться на наборах даних KDDCup99, NSL-KDD, UNSW-NB15 та IDS 2018, а у ролі класифікаторів використовуються різні моделі машинного навчання, а також їх ансамбль, зокрема з вибором відповіді на основі голосування. Автори стверджують що навіть при базовому найширшому наборі ознак, випадкові ліси досягають максимальної точності (майже 100%) на згаданих наборах. При скорочення набору ознак на основі зазначених критеріїв більше ніж вдвічі точність більшості моделей фактично не падає. При цьому єдина суттєва перевага від меншої множини ознак показана лише у вищій швидкості навчання, та швидкості висновування (хоча остання, авторами не вимірювалась). Крім того, в роботі стверджується, що ансамбль моделей навчається значно швидше ніж окремі базові моделі, що є дуже незвичним. На жаль в роботі для різних наборів даних

використовувались суттєво різні набори ознак, тому важко виділити якийсь тренд, що був би спільний для всіх розглянутих наборів.

В роботі Patil et. al 2022 [47] автори скомбінували вже відомі методи виявлення мережевих атака за допомогою машинного навчання з технікою так-званого "пояснювального штучного інтелекту" (XAI - explainable artificial intelligence) [48], з метою зробити результати роботи класифікатора більш придатними для інтерпретації людиною. В роботі для навчання та тестування використовувався набір даних CIC-IDS-2017, а ціллю виявлення були DDoS (distributed denial of service) атаки. Мережевий трафік представлявся у вигляді мережевих потоків з 78 числовими ознаками. В роботі проводилось попереднє вилучення високо-корелюючих ознак, а також ознак в яких містяться аномалії. В результаті було виділено 10 найбільш значущих ознак, однак конкретний алгоритм вибору ознак не уточнюється. Також була застосована стандартизація даних. Для збалансування навчальної вибірки автори використали алгоритм SMOTE (Synthetic Minority Oversampling Technique). В ході роботи, авторами було використано класифікатори трьох архітектур: дерево прийняття рішень, випадковий ліс та машина опорних векторів (SVM - Support Vector Machines). Всі 3 архітектури показали схожу ефективність (точність ~95.5-96%), однак у SVM суттєво менший F1-рахунок (0.8 супроти 0.88). Додатково застосовується техніка ансамблю цих трьох класифікаторів із голосуванням, що однак не принесло суттєвого підвищення точності. На жаль в роботі не вказано чи наведені показники стосуються навчальної чи тестової вибірки. Для аналізу результатів роботи отриманих моделей був застосований алгоритм пояснювального ШІ під назвою LIME (Local Interpretable Model-agnostic Explanations). Це дозволило аналізувати величину впливу різних ознак на рішення класифікатора для конкретного прикладу з набору даних.

В роботі [49] також проводиться дослідженням по створенню моделей аналізу мережевих даних з більш прозорою для користувача поведінкою. Автори використовують класифікатори згорткових нейронних мереж у комбінації з методами пояснювального штучного інтелекту для можливості кращої інтерпретації отриманих результатів.

В роботі Balise et al. 2020 [50] було запропоновано спосіб виявлення ботнетів шляхом обчислення сигнатури мережевої поведінки окремих хостів, що є дещо відмінним від класифікації "сигнатур" окремих мережевих потоків. "Сигнатура" в даній роботі представляється частотами розподілу пекетів за деякими атрибутами. У якості таких атрибутів беруться в тому числі і характеристики мережевих пакетів, які в інших роботах зазвичай навмисно опускаються при класифікації, а саме: IP адреса призначення, порт джерела та порт призначення. Такий хід обумовлений деякими спостереженнями про особливості роботи ботнетів. Зокрема, в роботі стверджується, що ботнети використовуються менш звичні діапазони ефемерних портів при створенні нових з'єднань. Також трафік ботів часто містить більш широкий набір портів призначення нехарактерних для звичайних застосунків, що можна пояснити регулярним скануванням портів та процесом пошуку вразливостей для подальшого розповсюдження. Також для деяких ботнетів є характерним використання портів за замовчуванням таких протоколів як Telnet для комунікації з C&C та SMTP для розсилки спаму. Крім того, автори стверджують, що тоді як для здорових хостів є більш звичним обмінюватись даними з хостами з тієї самої підмережі, то боти частіше комунікують із машинами за її межами. Взявши до уваги дані евристики, автори зробили припущення, що обчисливши ймовірність розподілу кількості пакетів за цими та деякими іншими атрибутами можна отримати сигнатуру хоста достатню для відрізнення бота від звичайного незараженого хоста. В ролі навчальної та тестової вибірки автори використали відкритий набір даних CTU-13 [18]. Для створення моделі класифікації в тій роботі застосовується 2 підходи. В першому використовується кластеризація за групами хостів з маркуванням всього кластеру як "ботнети", якщо принаймні 1 із хостів у кластері заражений ботом. У якості алгоритму кластеризації застосовувався DBSCAN (Density-Based Spatial Clustering of Applications with Noise), який корисний у випадках, коли кількість кластерів заздалегідь невідома. Якість такого підходу дуже сильно залежить від параметру максимально дозволеної відстані (радіусу околу) між точками одного кластеру. Для обчислення ознак потоків хостів використовується квантування значень атрибутів пакетів

надісланих у певних діапазонах IP-адрес чи мережевих портів з конкретного хоста. Квантування здійснюється через побудову гістограми частот розподілу з "кошами" (інтервалами) як фіксованої так і адаптивної довжини. Найкращих результатів авторам вдалося досягти використовуючи 512 кошів адаптивного розміру і отримавши значення влучності 74% при чутливості 100% та рахунку F1 = 80%. При збільшенні радіусу околу, точність зростає до 100%, однак влучність падає до 85%. Як альтернативу для порівняння автори також застосували деякі класифікатори ШІ, зокрема машину опорних векторів (SVM), випадковий ліс та багаторівневий перцептрон. Найкращі результати отримані для класифікатора SVM, для якого влучність, чутливість та F1 рахунок одночасно досягли значення в 93%.

Ще одним дослідженням в області виявлення ботнетів, де для класифікації трафіку використовувались алгоритми кластеризації є робота Kirubavathi et al. 2021 [51]. Автори проводять порівняння якості алгоритмів K-Means, X-Means, а також підходу на основі «Моделі суміші Гауса» (Gaussian Mixture Model, GMM). Кластеризація здійснювалась на базі статистик кількості пакетів та байтів в TCP та UPD потоках, а також їх тривалості. В результаті була показана ефективність кластеризації слідів трафіку ботів, зокрема для Zeus [52], та SpyEye [53].

Також, уваги заслуговує дослідження Lopes et al. 2022 [54], де автори використовують обернені статистики потоків для виявлення аномалій серед потоків звичайного трафіку.

Висновки до Розділу 1.

За результатами проведеного огляду та аналізу літературних джерел показано, що дослідження по виявленню мережевих загроз ведуться з часів початку розростання мережі інтернет. Спроби застосування методів штучного інтелекту в цій області стало відповіддю на стрімке зростання інтенсивності та варіативності мережевих атак, а також на збільшення складності зловмисних програмних, що їх проводять. Використання методів ШІ доцільне для виявлення більш витончених та хитрих атак, принципи роботи та виявлення яких складно описати класичними алгоритмічними методами. При цьому, і досі немає єдиного усталеного підходу до

створення систем аналізу мережевих даних на основі ШІ. Однак можна виділити деякі схожі методи та алгоритми, які застосовуються в даній області. Зокрема, більшість дослідників схиляються до створення високорівневої інтерпретації трафіку через застосування методу виділення окремих потоків з множини мережевих даних. На противагу аналізу окремих пакетів, виділення потоків дозволяє передавати класифікаторам семантично агреговану інформацію, що суттєво зменшує об’єм оброблюваних даних та надає моделі більше контекстуальної інформації необхідної для розумного прийняття рішень. Крім того, особливої уваги вимагає задача вибору важливих ознак потоків, на основі яких проводиться класифікація, а також архітектура самих моделей ШІ.

Водночас, всі дослідники стикаються з рядом проблем характерних для цієї області, а саме:

- Гостра нестача обсягу розмічених навчальних та тестувальних даних.
- Швидке старіння існуючих навчальних наборів даних та їх низька репрезентативність через постійну еволюцію ботнетів.
- Незбалансованість прикладів різних класів у існуючих вибірках даних.
- Задача виділення числових ознак семантики поведінки зловмисного ПЗ із зашифрованого мережевого трафіку.
- Необхідність виділення найбільш значущого набору ознак із мережевих даних для якісної та швидкої їх класифікації.
- Питання можливості перенесення результатів навчання моделей з однієї вибірки мережевих даних на інші.

Крім того, жодна зі згаданий в цьому розділі робіт не розглядає дві важливі проблеми притаманні й для інших областей застосування штучного інтелекту, які в контексті побудови системи виявлення загроз стоять особливо гостро. Першою такою проблемою є можливості зловмисного ПЗ ухилятися від виявлення шляхом внесення незначних змін у свою поведінку, що відобразиться на породжених ними мережевих даних і може спричинити помилку про їх класифікації моделями ШІ. Другою важливою проблемою є надання більш формальних гарантій щодо надійності та точності систем виявлення заснованих на нейромережах, що

обумовлено відносною непередбачуваністю та низькою придатністю до інтерпретації результатів їх роботи у порівнянні з іншими методами.

РОЗДІЛ 2. СИСТЕМА ВИЯВЛЕННЯ ЗАГРОЗ МЕТОДАМИ ШІ НА ОСНОВІ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

У розділі детально описано розроблену архітектуру та функціонування системи виявлення загроз на основі аналізу мережевого трафіку, яка була реалізована в ході цього дослідження. Подано передумови та попередні припущення в контексті яких робляться всі подальші твердження та проводиться побудова системи. Завдання виявлення загроз сформульовано в рамках задачі класифікації мережевих даних. Подано опис процесів обробки мережевих даних та способи представлення їх у формі наборів числових ознак прийнятних для обробки моделями класифікації на основі машинного навчання та нейронних мереж. Описано набори даних, які використовуються для навчання та тестування моделей. Подано детальний опис моделей ШІ, які використовувалися для класифікації, процес їх навчання, а також порівняння їх ефективності.

2.1. Побудова структурного підходу до проектування мережевих систем виявлення загроз на основі ШІ

На основі аналізу матеріалів з Розділу 1, можна сформувати узагальнюючу послідовність кроків для побудови систем аналізу мережевих даних з метою виявлення загроз, заснованих на моделях класифікації ШІ.

1. Збір достатнього об’єму прикладів трафіку цільового класу мережевих атак необхідних для навчання та тестування моделі. Згенерована вибірка має бути достатньо репрезентативною, і включати різні варіації однієї й тієї ж атаки. При цьому спостережувані дані бажано отримувати в різних мережевих середовищах. Зокрема, має сенс використовувати рекомбінацію існуючих відкритих навчальних наборів. Також корисним є схрещення таких наборів з власними даними отриманих з «пасток» для зловмисного трафіку (англ. honeypot) або з даними перехопленими в локальних дослідницьких мереж в лабораторних умовах.

2. Семантична агрегація пакетів та виділення числових ознак, які характеризують сформовані групи і здатні відображати якомога більше аспектів поведінки характерних для відповідних типів мережевих загроз.
3. Попередня обробка даних та ознак: відкидання агрегатів з недостатнім об’ємом чи якістю їх характеристик необхідних для прийняття рішення класифікатором, щодо їх природи. Також на цьому етапі проводиться стандартизація даних, зокрема їх масштабування у відповідності з діапазоном практично досяжних значень.
4. Проведення балансування навчальної вибірками генерацією штучних прикладів, зокрема методами на зразок SMOTE.
5. Побудова та навчання моделей штучного інтелекту на основі сформованих вибірок даних.
6. Оцінка ефективності виявлення загроз, зокрема для прикладів мережевих даних, що суттєво відрізняються від наявних у навчальній вибірці.
7. Підвищення точності класифікації за допомогою методів розумного відбору ознак, наприклад, шляхом аналізу взаємної інформації, ранжування за коефіцієнтом кореляції Пірсона та оцінка критеріїв значущості для різних комбінацій чи класів ознак. Також корисним може бути перехід до простору іншої розмірності, на приклад, за допомогою методу головних компонент (PCA).
8. Оцінка та підвищення стійкості моделей класифікації до спроб навмисного ухиляння від виявлення.
9. Формалізація та верифікація математичними методами критеріїв надійності системи.

Очевидно, що не всі окреслені етапи є обов’язковими, однак якісне виконання кожного з них може суттєво підвищити ефективності системи, що розробляється. В даному розділі будуть описані та експериментально продемонстровані кроки 1-3, а також 5-7. В наступних розділах особлива увага буде приділена пунктам 8 та 9.

2.2. Необхідні передумови, вимоги та припущення при проектуванні системи

Для будь-якої системи безпеки необхідно чітко встановлювати умови в яких очікується, що система буде функціонувати коректно.

Перш за все, як і будь-якій NIDS, створеній системі виявлення вторгнень необхідно перехоплювати трафік, що циркулює через мережу, за якою ведеться спостереження. Якщо припускається, що першочерговий вектор атаки на локальну мережу буде направлений із-зовні усередину. Відповідно, мінімальною необхідною умовою для коректного функціонування системи, є перехоплення всього об'єму трафіку, що циркулює між конкретним хостом в локальній мережі (LAN) та безпосередньо інтернетом (WAN).

Додатковою бажаною умовою, є також перехоплення трафіку і між хостами самої локальної мережі, тобто трафіку який не покидає межі LAN. Це обумовлено тим, що якщо певний хост було скомпрометовано зсередини, або ж його зараження із-зовні залишилось невиявленим, то цілком можливо, що ботнет спробує заразити й інші хости локальної мережі через даний хост. При цьому, вимагається забезпечення відсутності "здвоювання" пакетів, яке виникає при перехопленні одного й того самого пакету двічі, в різних місцях. Різні варіанти налаштування перехоплення трафіку, які би задовольняли дані умови, будуть розглянуті в наступних розділах.

Ще однією деталлю вартою уваги є те, що ця система заснована на генерації та аналізі мережевих потоків, а не окремих пакетів. Під мережевим потоком мається на увазі логічно-пов'язана група пакетів, яка локалізована в часі та відображає «розмову» між двома віддаленим застосунками чи хостами. Повний алгоритм виділення мережевих потоків буде описаний у наступних розділах, та наразі зазначимо, що система передбачає агрегацію даних отриманих з множини атомарних одиниць (пакетів) в більш високорівневі об'єкти (потоки, чи групи потоків). При цьому, агрегація проводиться на самих ранніх етапах обробки, в ідеалі, безпосередньо в момент перехоплення трафіку. Заміна обробки «сирого» трафіку на можливість роботи з його агрегатами одразу вирішує, чи принаймні

пом’якшує, щонайменше кілька проблем, які виникають при проектуванні систем аналізу мережевих даних. По-перше, це значно скорочує кількість даних, які необхідно транспортувати. По-друге, не потрібно зважати на додаткові гарантії безпеки та санітизації зреплікованих чутливих даних, адже в агрегатах залишається лише мета-інформація про корисне навантаження пакетів оригінального трафіку. Третьою й можливо найбільш суттєвою перевагою є здатність такої системи працювати з зашифрованим трафіком на стільки-ж ефективно, як і з незашифрованим. Слід зазначити, що корисне навантаження пакетів в більшості випадків зашифроване (за допомогою SSL чи TLS), відповідно вимагається, щоб алгоритм виявлення міг працювати коректно без можливості глибокого аналізу окремих пакетів. Виключення зроблено для 2 протоколів, трафік яких зазвичай є незашифрованим, а саме DNS (domain resolution system) та ICMP (Internet Control Message Protocol). Зазначимо, що хоча трафік DNS резолюцій інколи і шифрується за допомогою технологій DoT (DNS поверх TLS) та DoH (DNS через HTTPS), однак таке шифрування використовується у звичайних умовах не так часто, і сам факт його наявності в підмережі, є підозрілим, якщо жоден відомий користувацький застосунок його не використовує.

Ще одним фундаментальним припущенням, на основі якого будується система виявлення загроз, є можливість ідентифікації зараженого чи скомпрометованого хосту за його IP адресою. Відповідно, перехоплення трафіку має бути налаштовано таким чином, щоб в точках перехоплення адреса джерела чи призначення в кожному з пакетів відповідала реальній IP адресі хоста з мережі, за якою ведеться спостереження. Відповідно, якщо в системі присутній NAT (network address translation) шлюз чи пристрій балансування навантаження, перехоплення має здійснюватися вже «за» ними, - тобто десь між цими мережевими елементами та хостами прихованими за ними. Додатковою, бажаною умовою є також і збереження мережевого порту цільових хостів. Інформація про порт може бути використана для ідентифікації конкретного процесу на скомпрометованому хості, якому належить шкідливий трафік.

2.3. Виявлення загроз як задача класифікації

Хоча IDS є комплексними системами з безліччю компонент, однак в ядрі будь-якої такої системи завжди лежить певний механізм дискримінації даних, які відображають характер поведінки системи, за якою ведеться спостереження, або ж природу подій, які відбуваються в ній. У випадку NIDS кінцевий механізм виявлення фактично можна звести до задачі класифікації мережевого трафіку, однак для цього треба визначити одиницю класифікації, результуючі класи, а також спосіб інтерпретації результатів в контексті виявлення загроз.

Як вже згадувалося, у якості одиниць класифікації, в цій роботі використовуються агрегати мережевих пакетів у формі потоків. При цьому застосовуються два різні підходи до агрегації:

1. Класифікація окремих мережевих потоків, де кожен окремий потік – це агрегат групи пакетів, які циркулювали між двома віддаленими процесами на двох різних хостах.
2. Класифікація груп мережевих потоків, впорядкованих за часом, де кожна група потоків представляє історію комунікації процесів одного хоста з іншими за певний проміжок часу.

В залежності від вибору одиниці класифікації буде змінюватися й спосіб інтерпретації результатів дискримінації. В першому випадку, при класифікації окремих мережевих потоків, система розділяє потоки на класи «звичайні» або «шкідливі». Якщо потік «шкідливий», це значить, що один або обидва із пари процесів, між якими відбувалась комунікація представлена цим потоком вірогідно є шкідливим програмним забезпеченням.

В другому випадку, під час класифікації груп потоків, суб’єкт класифікації визначається ідентифікаційною ознакою потоків, за якою здійснюється групування (наприклад мережевий порт чи IP адреса). Дискримінатор розділяє групи на два класи, а груп з лише звичайним трафіком, та групи з наявним шкідливий трафік. Слід зазначити, що в цьому випадку перевіряється саме наявність шкідливого трафіку, відповідно належність до другого класу не значить, що всі потоки у групі

– шкідливі. Припустимо, що групування потоків здійснювалось за IP адресою хоста, тобто в групі присутні всі мережеві потоки, в яких даний хост брав участь за певний проміжок часу. Тоді для кожної IP адреси з локальної мережі (LAN/VLAN) система виявлення зможе відповісти на запитання: «чи був даний хост заражений злоякісним програмним забезпеченням?» або ж принаймні «чи була зареєстрована спроба заразити даний хост?» З іншого боку, для всіх груп із IP адресами з зовнішньої мережі (WAN), система відповідатиме на запитання «чи чужорідний хост намагався вчинити несанкціоновані дії по відношенню до хостів локальної мережі?». В обох випадках, інформація отримана від системи виявлення є корисна і може бути використана для подальших мір протидії.

2.4. Мережеві потоки та алгоритм їх виділення

Концепція «мережевого потоку» доволі розповсюджена в сфері мережевого адміністрування та моніторингу. Однак, конкретна інтерпретація цього поняття може суттєво відрізнятися в різних роботах на тему виявлення мережевих загроз, тому в цьому розділі буде пояснено, що саме мається на увазі під даним терміном та буде проведена формалізація алгоритму виділення числових ознак мережевих потоків.

Існують різні способи виділення мережевих потоків, різної складності та якості результату. В основі концепції потоку лежить ідея можливості об’єднати окремі пакети трафіку, в контекстуально взаємопов’язані групи. Мережевий потік повинен об’єднувати у собі всі пакети, якими обмінювались два віддалені процеси («кінцеві точки») за певний часовий інтервал і має слугувати високорівневим відображенням сеансу комунікації між цими процесами. Прикладом може бути повна послідовність пакетів, обмін якими відбувається в межах одного TCP-з'єднання. Потоки можуть бути однонаправленими або двонаправленими, залежно від процедури виділення та вимог. Як випливає з назви, однонаправлений являє собою набір пов’язаних пакетів, що йшли в спільному напрямку від однієї кінцевої точки до іншої, тоді як двонаправлений включає пакети, що йдуть в обох

напрямках. В цій роботі виділяються саме двонаправлені потоки, адже класифікатору необхідна повна інформація про весь сеанс комунікації.

Кожен мережевий потік представляється двома наборами атрибутів. Перший набір – це «ідентифікаційний кортеж», яким потоки унікально відрізняються одне від одного. Другий набір атрибутів – це числові характеристики (ознаки) потоку. В подальшому, саме ці атрибути передаються як вектор ознак моделі-дискримінатору для класифікації.

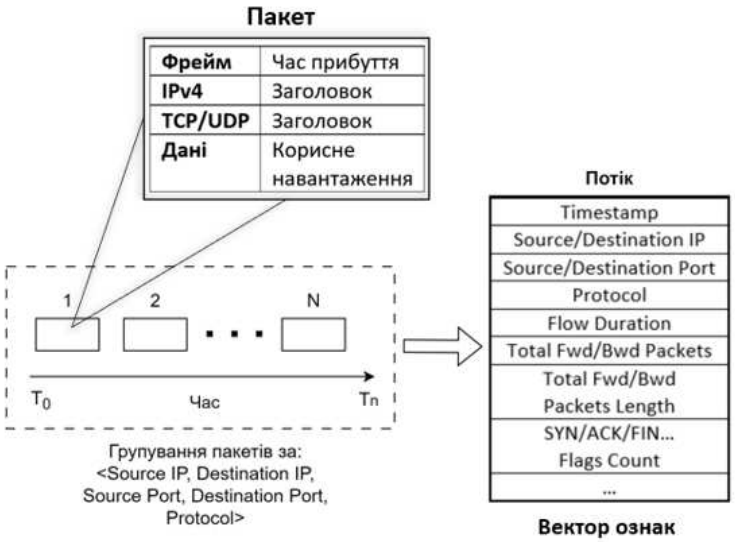

Рис. 2.1. Представлення мережевого потоку

Процес об’єднання груп пакетів в мережеві потоки та обчислення їх числових ознак будемо називати «реконструкцією мережевих потоків». В наступних розділах детально опишемо процес реконструкції потоку та формування кожного з набору його атрибутів.

2.4.1. Алгоритм реконструкції мережевих потоків

Як вже зазначалось, під «реконструкцією мережевих потоків» мається на увазі виокремлення із всього об’єму мережевого трафіку груп пакетів, які би представляли цільний сеанс комунікації між двома віддаленими процесами. Для ідентифікації такого сеансу комунікації необхідно 6 атрибутів, які називатимемо «ідентифікаційним кортежем» потоку:

<*IP-адреса джерела, IP-адреса пункту призначення,*

Порт джерела, Порт пункту призначення,

Протокол (Транспорт), Час початку>

Як і випливає з назви, ця група атрибутів завжди є унікальною і використовується для ідентифікації індивідуальних мережевих потоків. Реконструкція потоків здійснюється по мірі надходження нових пакетів. Алгоритм реконструкції можна розділити на 3 основні процеси: 1) реєстрація нових активних потоків 2) співставлення пакета з одним із поточно-активних потоків 3) маркування потоків, які досягли однієї з умов термінації, як завершені, та їх видалення зі списку активних потоків. Алгоритм постійно відстежує список вже активних потоків і для кожного нового пакету перевіряє чи є потік пов'язаний з ним. Для цього, алгоритм виділяє із заголовків мережевого та транспортного рівнів пакету IP-адреси та порти джерела і призначення, а також тип протоколу, після чого в списку активних шукає потік, ідентифікаційний кортеж якого містить дані атрибути. Якщо відповідність було знайдено, то пакет відноситься до даного потоку. Якщо ж не знайдено жодних активних потоків, з якими можна пов'язати даний пакет, то створюється новий потік, який додається до списку активних. Значення полів ідентифікаційного кортежа нового потоку береться з відповідних заголовків початкового пакету. «Час початку» отримується в процесі перехоплення трафіку і відповідає часовій мітці (timestamp) присвоєній початковому пакету в момент його захоплення. Процес співставлення пакетів з активними потоками та породження нових потоків у разі необхідності повторюється для кожного наступного пакета, що аналізується.

Будь-який сеанс комунікації є обмежений у часі. Відповідно, кожен потік має власні умови звершення, при досягненні яких алгоритм реконструкції видаляє його зі списку активних потоків. Універсальними умовами завершення, що актуальні для потоків будь-якого протоколу є «тайм-аут простою» та «тайм-аут активності»:

- «Тайм-аут активності потоку» – це максимальний інтервал часу, впродовж якого потік вважається активним. Якщо поточний час життя потоку перевищує це значення, він позначається як завершений і створюється новий потік. Ситуація перевищення дозволеного часу активності схематично показана на Рис. 2.2.

- «Тайм-аут простою потоку» – це максимальний проміжок часу, дозволений між двома послідовними пакетами в одному потоці. Якщо дозволений час простою минув, але новий пакет в потоці не зареєстровано, то потік вважається завершеним і видаляється зі списку активних потоків. На Рис. 2.3 продемонстровано випадок, коли дозволений час простою був вичерпаний.

Алгоритм реконструкції періодично перевіряє список активних потоків на наявність входжень, час життя яких закінчився за одним із згаданих тайм-аутів і видаляє їх зі списку.

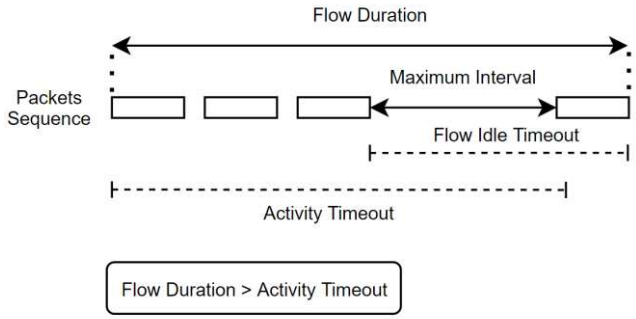

Рис. 2.2. Завершення потоку за тайм-аутом активності

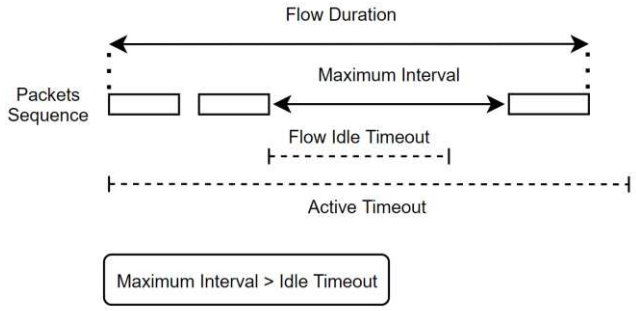

Рис. 2.3. Завершення потоку за тайм-аутом простою

Для UDP та ICMP потоків тайм-аути активності та простою - це єдині умовами термінації, оскільки інших механізмів завершення з'єднання вони не мають. TCP-потоки, однак, окрім універсальних мають також й інші умови завершення, регламентовані самим протоколом (див. Рис. 2.4). Таких умов є дві:

- Першою й найпоширенішою є умова «ввічливого завершення з'єднання», яка визначається послідовність бітових прапорців FIN-ACK-FIN-ACK. Якщо зустрічається пакет з прапорцем FIN, то реконструктор очікує на наступні пакети з прапорцями ACK-FIN-ACK, після чого позначає потік як завершений.
- Інший випадок завершення потоку TCP – коли зустрічається прапорець RST, що означає, що з'єднання було грубо перервано одним із учасників.

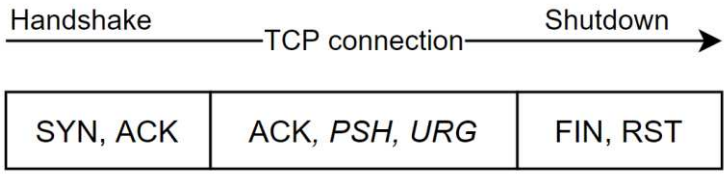

Рис. 2.4. життєвий цикл TCP-з'єднання та відповідні бітові прапорці

В результаті описаного процесу реконструкції, всю множину пакетів було розбито на логічно зв’язані групи, кожна з яких представляє один потік. В подальшому, під «аналізом потоку» матиметься на увазі, аналіз відповідної виділеної групи пакетів, які його представляють.

Варто зауважити, що процес реконструкції може працювати як режимі онлайн так і офлайн. В першому випадку, реконструктор отримує пакети від «сніфера» (програмного забезпечення, що перехоплює трафік) в режимі реального часу по мірі надходження нових пакетів. На практиці, в цій роботі це було реалізовано інтеграцією перехоплювача в сам реконструктор. В другому випадку, передбачається, що аналізуються історичні дані, як-от «дампи» трафіку збережені у форматі PCAP і реконструктор відкриває файл та читає пакети послідовно один за одним. В обох випадках описаний вище алгоритм реконструкції потоку не змінюється.

2.4.2. Обчислення ознак мережевих потоків

Після виділення окремого потоку зі всієї множини пакетів наступним кроком є представлення його у формі придатній для обробки моделлю класифікації. Цей процес також відомий як «вкладення» даних (англ. “embedding”) і є важливим етапом в обробці даних методами ШІ. Ціль процесу вкладання – привести гетерогенні дані до певної математично-стандартизованої форми, яку модель ШІ очікує отримати на вході. В цьому випадку процес вкладення зводиться до обчислення вектору числових ознак, який би в повній мірі характеризував шаблон поведінки конкретного мережевого потоку. В цьому розділі наведено перелік ознак, використаних в цій роботі, проведено їх систематизацію та описано механізм їх обчислення.

Зауваження: Для уникнення повторень при перерахуванні назв ознак введемо нотацію переліків через роздільник «/», що буде означати всі можливі комбінації перерахованих значень та базової назви. Наприклад, під «A/B 1/2 базова_назва» розумітимемо 4 ознаки, а саме «A 1 базова_назва», «A 2 базова_назва», «B 1 базова_назва» та «B 2 базова_назва».

Як вже зауважувалось при проведенні аналізу робіт інших авторів, існує безліч видів корисних ознак, які можна виділити з потоків трафіку. Для кращої систематизації, ознаки розділено на 2 основні класи: категоріальні та кількісні. Кількісні в свою чергу розділено на агреговані та статистичні. Розглянемо кожен з цих класів більш детально.

Під «категоріальними» ознаками маються на увазі ті, спектр яких є обмежений заздалегідь відомими наборами можливих значень, і які розбивають дані на певні підкласи. Прикладом ознаки такого типу є «тип транспортного протоколу». В даній роботі для цієї змінної допускаються лише три значення: «TCP», «UDP» та «ICMP, які отримуються з поля «Protocol» заголовку IPv4 пакету, згідно з RFC 791. До цього класу ознак також належать і бінарні атрибути, наприклад, «наявність DNS резолюції», про що аналізатор може дізнатися перевіривши корисне навантаження UPD пакету та спробувавши виокремити з нього секції «Header» та «Question», формат яких задекларований в стандарті RFC 1035. Іншим прикладом категоріального атрибуту може бути «тип завершення

з’єднання». Цей атрибут представляє за якою саме умовою завершився потік, що було описано в попередньому підрозділі. ICMP та UDP потоки можуть належати одній з двох категорій: завершені «за тайм-аутом активності» або «за тайм-аутом простою». Для TCP потоків додається ще 2 категорії: завершені «ввічливо» (послідовністю прапорців FIN-ACK), чи «грубо» (прапорцем RST). Також до категоріальних ознак будемо відносити й IP-адреси та порти. Хоча спектр можливих значень цих атрибутів значно ширший за всі попередні, однак, вони розбивають потоки на класи належності до певного хоста чи окремого процесу, а тому по суті вони також є категоріальними. Крім того, множина допустимих значень цих ознак є обмежена, а самі значення будуть регулярно повторюватись при тривалому спостереженні за мережею. В підсумку в даній роботі було виділено 6 категоріальних ознак для кожного потоку: *«IP-адреса джерела», «IP-адреса пункту призначення», «Порт джерела», «Порт пункту призначення», «Транспортний протокол» та «Умова завершення» потоку.* Слід зауважити, що більшість з цих ознак в подальшому буде використана для додаткового групування та агрегації потоків.

Тепер розглянемо кількісні ознаки потоків. Як зазначалось, має сенс їх розділити на агреговані та статистичні. Агреговані ознаки описують загальні властивості потоку, такі як загальна кількість виникнення певних подій чи загальна сума значень певних характеристик пакетів. Агреговані ознаки зазвичай мають кумулятивну природу, відповідно зручно їх обчислювати інкрементально. При розгляді нового пакету значення інкрементальної ознаки зростає шляхом додаванням певної числової характеристики, що описує даний пакет, до спільної суми, яка описуватиме весь потік в цілому. До такого роду можна віднести наступні 7 ознак, використані в даній роботі:

- «*Fwd/Bwd Total Header Length*» - сумарна довжина заголовків в байтах. Значення довжини заголовків обчислюється як сума значень поля «IHL» (Header Length) заголовку IPv4, яке задекларована в RFC 791, та поля «Data Offset» заголовку TCP, що визначається в RFC 9293. Згадані поля описують

довжини заголовків в 32-бітних словах, тому для конверсії в байти результат множиться на 4.

- *«Fwd/Bwd Total Packet Length»* - сумарна довжина корисного навантаження пакету, виміряна в байтах. Значення корисного навантаження пакет вираховується так: *«Payload Length» = «Total Length» (IPv4) – («Header Length» (IPv4) - «Data Offset» (TCP)) * 4*, де *Total Length* - поле IPv4 заголовку.
- *«Fwd/Bwd Total Inter-Arrival Time»* - сумарне значення часу міжпакетних інтервалів. Під «часом піжпакетного інтервалу» мається на увазі дельта між часом отримання поточного пакету та часом отримання попереднього пакету в цьому потоці. Час отримання визначається полем «Timestamp» («Часова мітка»), яке заповнює сніфер під час перехоплення пакету.
- *«Flow Duration»* - тривалість потоку в мікросекундах. Може кумулятивно оновлюватись з кожним новим пакетом, однак фактично, є значенням різниці часових міток першого та останнього пакету.

Зауваження: Нотація «*Fwd/Bwd» (*«Вперед/Назад») відображає двонаправлене відслідковування. Ознаки, які обчислені на основі пакетів, що йшли від джерела до пункту призначення позначаються через «Fwd» (англ. forward, тобто «вперед»), а ті що обчислені з пакетів у зворотному напрямку позначаються як «Bwd» (англ. backward, тобто «назад»). При цьому напрямок потоку задається напрямком першого пакету в ньому.

Окремим випадком відстежування кумулятивної суми є механізм лічильника, який інкрементується на одиницю якщо пакет, що розглядається, володіє певними атрибутами. Лічильник можна застосувати, наприклад, для обрахунку ознак «кількість TCP прапорців певного типу» у потоці.

В цій роботі використано 18 ознак на основі лічильника, а саме:

- «*Total Fwd/Bwd Packets»* - сумарна кількість пакетів надісланих в кожному з напрямків.
- «*Fwd/Bwd SYN/FIN/ACK/RST/CWR/PSH/URG/ECE flags count»* - кількість пакетів у потоці, в яких поле «Flags» TCP заголовку містить прапорець

відповідного типу. Розглядається кожен з 8 прапорців протоколу TCP, а саме SYN, FIN, ACK, RST, CWR, PSH, URG, ECE. Ці ознаки присутні лише в TCP потоках.

Із базових агрегованих ознак, отриманих на основі акумуляції послідовності значень характеристик пакетів, також можна отримати й більш складні, «композитні» ознаки, які обчислюються на основі кількох базових, і при цьому мають певну логічну інтерпретацію, корисну для аналізу мережевих даних. В багатьох роботах на цю тему, автори застосовують композитні ознаки, тому було прийнято рішення спробувати застосувати їх і в цій роботі для порівняння результатів. Нижче наведено перелік таких ознак:

- *Bytes/second* – середня кількість байтів переданих за секунду (впродовж всієї тривалості потоку). Обчислюється як *(Fwd+Bwd Packet Length Total + Fwd+Bwd Header Length) / Flow Duration*.
- *Packets/second* – середня кількість пакетів за секунду. Обчислюється як *Total Fwd+Bwd Packets / Flow Duration.*
- *Fwd/Bwd Bytes/Packet* – середній розмір пакету в потоці. Відношення суми розмірів пакетів їх кількості.
- *Subflow Fwd/Bwd Avg Packets* – середня кількість пакетів у під-потоках. Під «під-потоком» мається на увазі групи пакетів в межах одного потоку, в яких інтервали між окремими пакетами не перевищують певне значення (в нашому випадку – 1 секунда).
- *Subflow Fwd/Bwd Avg Bytes* – середня кількість байт у під-потоках.

Слід зауважити, що не всі з перерахованих ознак є корисними в контексті задачі класифікації моделями ШІ, особливо у випадку використання нейронних мереж. Мова йде про ті ознаки, які мають пряму кореляцію з іншими базовими ознаками, особливо якщо їх легко можна вирахувати. Так, наприклад, композитна ознака «Packets/second» фактично прямо залежить від базових «Total Fwd/Bwd Packets» та «Flow Duration», які також присутні в характеристичному векторі потоку. Модель класифікації швидко помічає такі прямі залежності, і ці ознаки

стають надлишковими. Тим не менше, композитні ознаки можуть бути корисними у випадку якщо частина (або всі) базові ознаки, від яких вони залежать, відсутні в наборі. Такий підхід допомагає зменшити розмірність вхідного простору моделі ціною деякої втрати деталізації. Також слід брати до уваги, що для обрахунку такого роду ознак, реконструктору потоків фактично необхідно зробити додаткову обчислювальну роботу, що в ситуаціях, коли ресурс системи обмежений, може негативно вплинути на продуктивність.

Останній і, можливо, найбільш значущий клас числових ознак потоків – це статистичні величини, які описують розподіл значень характеристик потоку. В цій роботі використовувались 4 найбільш розповсюджені статистики, а саме мінімальне (min) та максимальне (max), та середнє (mean) значення, а також стандартне відхилення (std). Нижче наведений перелік деяких з ознак такого роду, які використовувалися в даній роботі:

1. *Fwd/Bwd Inter-Arrival Time Min/Max/Mean/Std* – статистики розподілу довжин між-пакетних інтервалів потоку (по кожному з напрямків).
2. *Fwd/Bwd Packet Length Min/Max/Mean/Std* – статистики довжин корисного навантаження пакетів у потоці.
3. *Flow Active Duration Min/Max/Mean/Std* – статистики розподілів інтервалів активності потоку. Під «інтервалом активності» мається на увазі часовий проміжок, де прогалина між двома послідовними пакетами не перевищує певну часову константу. В цій роботі дана константа дорівнює 5 секундам.
4. *Fwd/Bwd Initial Window Bytes* – розмір початкового вікна TCP пакету, який задається при рукостисканні (в пакетах з прапорцем SYN).
5. *Fwd/Bwd Actual Data Packets* – кількість пакетів з принаймні 1 байтом корисного навантаження.
6. *Fwd/Bwd Segment Size Min/Max* – величина TCP сегменту пакету (довжина TCP заголовку + довжина корисного навантаження).

В підсумку, в результаті процесу реконструкції потоків та обчислення їх ознак, замість безлічі пакетів система отримує набір числових векторів, кожен з

яких уніфіковано описує завершений акт комунікації між двома віддаленими процесами. Кожному такому характеристичному вектору також поставлений у відповідність і масив категоріальних атрибутів, на основі яких здійснюється ідентифікація потоку, фільтрація та групування. В подальшому, під терміном «мережевий потік» без додаткових уточнень буде матися на увазі не початкова група пакетів, а вектор ознак, який його характеризує.

Рис. 2.5 демонструє приклад обчислення ознак з набору пакетів, що представляють потік.

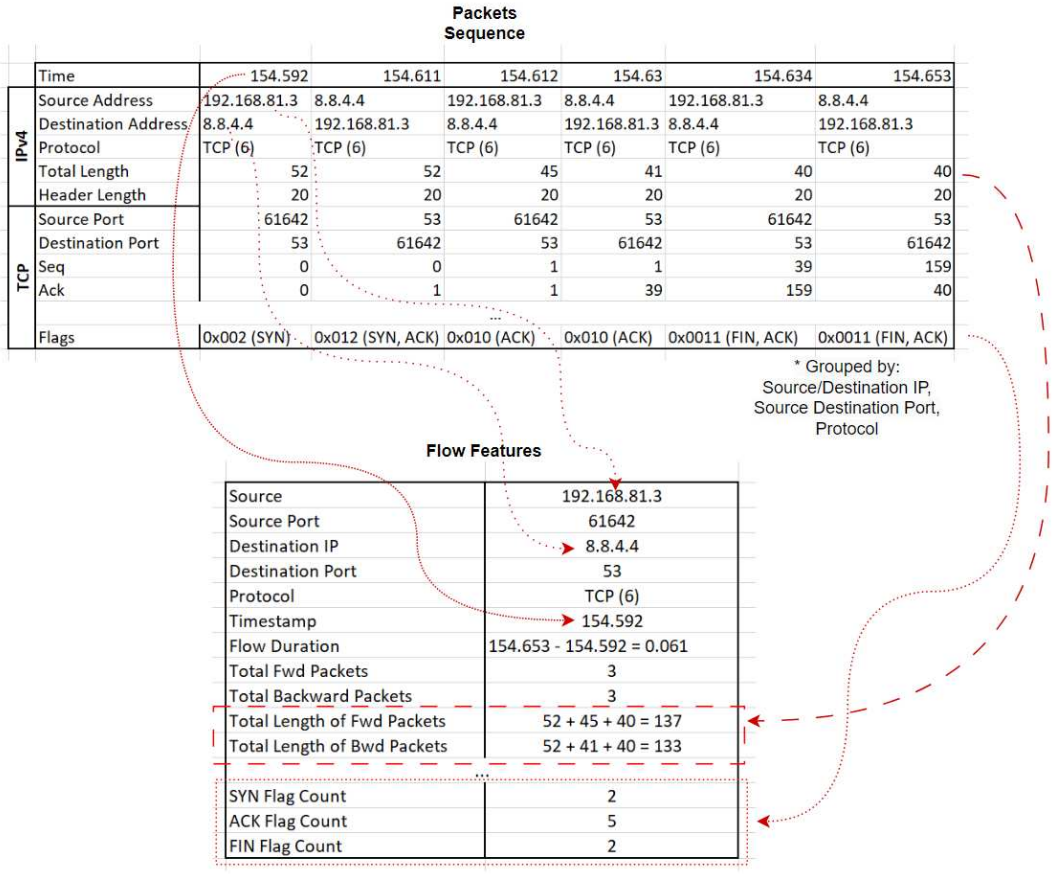

Рис. 2.5. Мережевий потік та його ознаки.

2.5. Інструменти виділення потоків та формати їх представлення

На практиці, щоб побудувати лінію аналізу мережевих потоків для виявлення втручання, важливо вибрати формат для представлення числових характеристик потоку та обрати відповідний інструмент для виділення цих значення. Більшість наборів мережевих даних надаються у вигляді PCAP-файлів (Wireshark-сумісний набір захоплених пакетів). Відповідно для їх аналізу також потрібен інструмент

перетворення пакетів із цих файлів у потоки формату прийнятному для подальшого аналізу моделями ШІ.

Одними з найбільш відомих технологій/протоколів виділення та представлення мережевих потоків є NetFlow та IPFIX (IP Flow Information eXport).

NetFlow — це технологія, розроблена та запатентована компанією Cisco у 1996 році та вбудована в більшість маршрутизаторів компанії. Перша реалізація (V1) обмежувалась трафіком IPv4 і містила лише кілька полів для опису потоку. Пізніше, після кількох внутрішніх ітерацій, була випущена версія V5, яка зараз є найбільш вживаною разом із V9. V5 має 18 статичних полів включаючи заголовки, що характеризують потік. Остаточною версією NetFlow є V9, в якій була розширена кількість статичних полів до 79 (104 для пристроїв Cisco) і було додано підтримку шаблонів, що дозволяє налаштовувати формат зібраної інформації про потоки відповідно до потреб: деякі поля можна додати, а інші можна виключити.

Існують дві основні проблеми з NetFlow в контексті IDS: 1) він є запатентованим 2) з самого початку він не розроблявся як інструмент моніторингу трафіку для виявлення вторгнень, а скоріше для спостереження за споживанням мережевих ресурсів та аналізу продуктивності або ж просто для керування мережею.

IPFIX (він же NetFlow V10, хоча окрім сумісної форми представлення, не має нічого спільного з оригінальним Cisco NetFlow) — це стандартизований протокол/технологія для захоплення мережевих потоків. На момент написання цієї роботи, згідно з документацією IANA (Internet Assigned Numbers Authority) [55] IPFIX має 529 інформаційних елементів, з яких 127 відповідають NetFlow версії 9 для зворотної сумісності. Як і NetFlow V9, IPFIX базується на шаблонах, тому зі згаданих 529 користувач може сам обрати множину полів для захоплення та відправки. По аналогії з V9, дана технологія підтримує IPv6, багатоадресну передачу та MPLS (Multiprotocol Label Switching). IPFIX не залежить від виробника і відповідає стандартам RFC 7011. Однією з основних відмінностей IPFIX від NetFlow є те, що IPFIX також підтримує аналіз протоколу прикладного рівня (L7) (наприклад, HTTP та IRC), що є дуже корисним для виявлення загроз. На приклад,

за словами Марка Грема, 2018 [56] IPFIX є кращим форматом для виявлення мережевих вторгнень.

Однак, після проведення аналізу атрибутів потоків зазначених в стандарті IPFIX в контексті цього дисертаційного дослідження, стало зрозуміло, що цей підхід може бути не достатньо гнучким для задачі виявлення складних мережевих загроз на зразок ботнетів. Як вже згадувалось, хоча IPFIX і містить великий набір атрибутів, однак в першу чергу ця технологія була спроектована саме для мережевого адміністрування та профілювання, а не для кібербезпеки. Звісно IPFIX все-ж містить деякі атрибути корисні для виявлення загроз, на зразок «часу початку та завершення» потоку, «кількість пакетів» у потоці та «кількість байтів». Водночас, в ньому відсутні більш складні статистики, на зразок стандартного відхилення розмірів пакетів чи інформація по розподілу значень за квантилями, які надзвичайно корисні для розпізнавання мережевих атак. Крім того, серед стандартних полів IPFIX повністю відсутня інформація про міжпакетні інтервали. Інформація про розмір TCP вікон присутня лише для початкових пакетів і не відображає можливість переузгодження їх розмірів. Також IPFIX не надає специфічної інформації на зразок «кількості пакетів принаймні з одним байтом корисного навантаження», яка є корисною для виявлення деяких мережевих атак.

IPFIX є відкритою технологією, тому існує кілька інструментів, здатних виділяти потоки в такому форматі. Зокрема, варто згадати ipfixprobe [57] та nfdump [58], вихідний код яких знаходиться у відкритому доступі або nProbe, який є пропрієтарною розробкою. Деякі з згаданих інструментів дозволяють розширювати базовий набір атрибутів, однак для цього необхідна розробка власних плагінів узгоджених з логікою роботи відповідного інструменту, що накладає ряд жорстких обмежень, а також йде врозріз з ідеєю використання вже готової стандартизованої технології.

В підсумку, хоча NetFlow та IPFIX забезпечуються якісне та широке представлення потоків, однак потенціал для їх використання в сфері кібербезпеки залишається обмеженим.

Після аналізу споріднених робіт інших дослідників в області, стає помітно, що в багатьох з них для виділення потоків з ціллю класифікації мережевого трафіку моделями ШІ використовується CICFlowMeter V4.0 [21]. CICFlowMeter — це інструмент з відкритим вихідним кодом, розроблений Канадським Інститутом Кібербезпеки. Він може реконструювати потоки з наборів пакетів в представлених у формі файлів PCAP. Вихідні потоки представляються у форматі CSV і кожен виділений потік містить 83 інформаційних поля. Вперше CICFlowMeter був створений для обробки набору даних ISCX Botnet 2014 [20], на основі якого було проведено чимало досліджень в сфері виявлення ботнетів штучним інтелектом. Популярність серед дослідників цей інструмент здобув саме через те, що оригінальні автори проектували його перш за все з метою аналізу мережевих даних на предмет виявлення загроз, а тому він містить набір атрибутів потоків, найбільш придатних для цього завдання. На жаль, цей інструмент спеціалізується лише на обробці дампів вже зібраних мережевих даних і не підходить для обробки трафіку онлайн, що необхідно для систем виявлення загроз у реальному часі.

Зважаючи на все сказане, в рамках цієї роботи був розроблений власний інструмент виділення потоків з мережевих даних, здатний функціонувати як в режимі офлайн так і онлайн. Набір атрибутів потоків, який він виділяє, є схожим на відповідний набір з CICFlowMeter, але вдвічі менший з міркувань ефективності. Також, в експериментальній частині цього дослідження CICFlowMeter використовується для отримання базових еталонних значень якості класифікації мережевих даних. Ці значення будуть корисні для порівняння ефективності роботи створеного інструменту з більш усталеним існуючим рішенням, використаним іншими дослідниками.

2.6. Можливі розширення множини ознак

Використання статистичних характеристик для узагальненого опису мережевого потоку дає досить високі показники виявлення, що буде продемонстровано в наступних розділах. Однак, виділення й інших ознак потоків, більш специфічних для виявлення ботнетів, може значно покращити результати.

Варто зауважити, що аналіз найбільш значущих ознак вже не один раз проводився різними авторами, але їх рекомендації з цього приводу суттєво відрізняються.

Використовуючи кластерний та кореляційний аналіз на даних зібраних з реальних ботнетів, Марк Грем, 2018 [56] дійшов висновку, що ознаки, наведені в Табл. 2.1, найбільше сприяють успішному виявленню їх активності. Особливо значною є інформація протоколів прикладного рівня L7 (DNS, HTTP, SMTP, IRC), яка тісно пов'язана зі способами комунікації ботів з центром команд та контролю (C&C). Яскравим прикладом є відомі методи «Fast flux» [59] і «Algorithm Generation Domain» (DGA) [60], що використовують DNS-запити для знаходження IP-адреси C&C при цьому уникаючи блокування. Вивчення особливостей функціонування ботнетів, шаблонів їх мережевого трафіку та притаманних їм евристик, може допомогти обирати набори ознак для найбільш ефективного виявлення. Це саме твердження дійсне і для виявлення інших типів мережевих атак.

Таблиця 2.1. Найбільш значущі ознаки для виявлення ботнетів згідно з результатами [56]

Назва поля	**Опис**
packetTotal	Загальна кількість переданих пакетів
flowEndMS	Часова мітка кінця потоку
flowStartMS	Часова мітка початку потоку
protocol	OSI протокол (TCP, UDP, ICMP і т.д.)
initTCPFlag	Прапорці, передані під час встановлення TCP-з'єднання
tcpSeqNos	Номери послідовності в TCP з'єднаннях
ircTextMessage	Текст та команди у випадку комунікації через IRC
httpGet	Наявність HTTP GET запиту
httpResponse	Код відповіді HTTP

dnsARecord	У потоці міститься резолюція домену записа типу «A»
dnsSOARecord	У потоці міститься резолюція домену записа типу «SOA»
smtpHello	В потоці містяться SMTP-команди «ELHO» чи «HELO»
sslName	Деталі центру SSL-сертифікації

На жаль, не всі з перерахованих ознак можна вирахувати якщо трафік зашифрований і колектор не знаходиться в точці термінації TLS з'єднання. Якщо з'єднання зашифроване, то фактично неможливо дізнатися інформацію прикладного рівня (L7), як-от HTTP метод чи код, як і текст IRC повідомлення. З іншого боку, такі ознаки як «початкова послідовність прапорців при встановленні з'єднання» та «номери TCP послідовностей» можна виділити без необхідності розшифрування корисного навантаження пакетів. Також не складно виділити і наявність резолюцій DNS-записів типу A та SOA в потоках на основі UDP, адже в більшості систем DNS запити є відкритими.

2.7. Попереднє опрацювання даних

На цьому етапі вже можна провести навчання моделі класифікації на основі характеристичних атрибутів потоків, підрахованих на попередньому етапі. Однак для отримання прийнятних результатів необхідно провести додаткову попередню обробку даних.

По-перше, важливо провести очистку даних. На практиці часові мітки пакетів не завжди відповідають порядку їх надходження та обробки, що призводить до неправильного обчислення значень характеристик (у разі, якщо це не було передбачено інструментом реконструкції потоків). Наприклад, після обробки набору даних ISCX Botnet 2014 за допомогою CICFlowMeter для деяких потоків певні поля, як-от *Flow Duration (тривалість потоку)* та *Inter-Arrival Time (час між прибуттям)*, мали негативні значення. Такі потоки були відкинуті як недійсні.

По-друге, велику увагу необхідно приділяти нормалізації та стандартизації даних. Через природу обраних характеристик їх значення можуть дуже відрізнятися – як у порівнянні з самими собою (у різних потоках), так і один з одним (в рамках одного потоку). Наприклад, тривалість потоку може варіюватися від кількох мілісекунд до кількох годин. Така неоднорідність негативно впливає на здатність моделі навчатися. Щоб компенсувати це, дані потрібно нормалізувати. У цій роботі, для кожної числової характеристики було використано масштабування між мінімальним та максимальним (2.1), тому їх кінцеві значення знаходяться в діапазоні від 0 до 1.

$$x^i_{normized} = (x^i - x^i_{min})/(x^i_{max} - x^i_{min}), \quad (2.1)$$

Також була зроблена спроба застосування альтернативного методу нормалізації характеристик через стандартизацію значень, шляхом віднімання середнього та ділення результату на стандартне відхилення. Однак, у контексті поставленої задачі, такий підхід показав себе гірше і моделі навчались повільніше у порівнянні з масштабуванням ознак потоків між мінімальними та максимальними значеннями знайденими у навчальній вибірці. Тому у всіх подальших експериментах для нормалізації використовувалася саме формула (2.1).

2.8. Класифікація потоків мережевого трафіку

Найголовнішою та найбільш складною ланкою системи виявлення є дискримінатор. Його задача відрізнити звичайний трафік від шкідливого. Під «шкідливим» мається на увазі будь-який трафік, який асоційований з діями зловмисника. В контексті виявлення ботнетів та іншого зловмисного ПЗ, як вже було зауважено, задача зводиться до класифікації мережевих потоків. На виході класифікатор повертає значення вірогідності (також відома як «впевненість») того, що даний потік породжений зловмисною програмою.

2.8.1. Класифікація окремих потоків

Існує кілька різних способів для класифікації захоплених слідів мережевої активності. Самим простим і водночас доволі ефективним є класифікація окремих

мережевих потоків. Оскільки (під час навчання) кожен потік вже промаркований як шкідливий або доброякісний, вектори ознак потоків можна подавати в класифікатор один за одним, щоб на основі закодованих у них значень, дізнаватися, чи є потік частиною комунікацій ботнету.

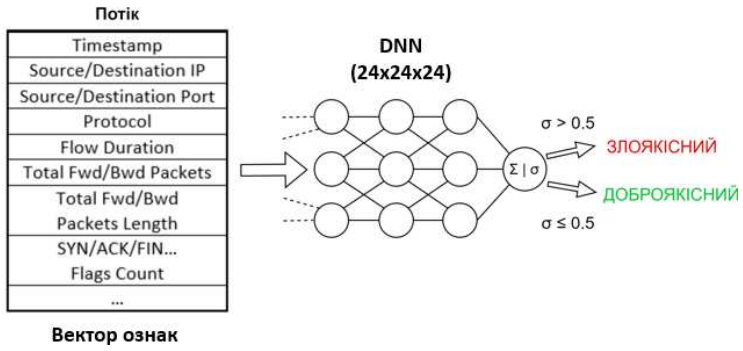

Рис. 2.6. Класифікація окремого потоку.

Якщо кожен вектор потоку містить F ознак, розмірність вхідних даних класифікатора та розмірність повної вхідної матриці під час навчання дорівнюватиме N×F, де N — кількість потоків у навчальному наборі. Цей метод буде використовуватися як базовий для подальших порівнянь. В Таблиці 2.2 наведено результати класифікації окремих потоків різними моделями. Також в таблиці наведено порівняння результатів отриманих в цій роботі з показниками ефективності наведеними авторами в інших роботах для даного набору даних для аналогічних моделей.

Найбільш продуктивними при такому підході виявились глибокі нейронні мережі (DNN) з 3 прихованими шарами 16x16x16 (AUROC - 0,866). Класифікатор "випадковий ліс" (Random Forest) посів друге місце (AUROC – 0,848), а логістична регресія – третє (AUROC – 0,778).

Таблиця 2.2. Ефективність моделей для ізольованих потоків (набір ISCX Botnet 2014)

Робота	**Вибір ознаки та зменшення розмірності**	**Класифікатор**	**Показники ефективності**

			(на незнайомих даних)
Ця робота.	PCA (16) (1 потік)	Логістична регресія	AUROC 0.804, Точність 0.78
	PCA (16) (1 потік)	Дерево прийняття рішень (глибина 9)	AUROC 0.844, Точність 0.833
	PCA (16) (1 потік)	Випадковий ліс (глибина 9)	AUROC 0.91, Точність 0.87
	(1 потік)	Логістична регресія	AUROC 0.778 (0.799), Точність 0.648 (0.722)
	(1 потік)	Дерево прийняття рішень (глибина 9)	AUROC 0.477, Точність 0.65
	(1 потік)	Випадковий ліс (глибина 9)	AUROC 0.848, Точність 0.798
	(1 потік)	DNN (16x16x16)	AUROC 0.866
Arnaldo et al. [3]	PCA (1 потік)	Випадковий ліс	AUROC 0.769
	(1 потік)	Випадковий ліс	AUROC 0.768
	PCA (28 з'єднаних потоків) + згенеровані ознаки	Випадковий ліс	AUROC 0.811
	(1 потік)	DNN	AUROC 0.724
Meshal Farhan et al. [45]	MUTUAL (1 потік)	Випадковий ліс	Точність 0.81
	MUTUAL (1 потік)	Логістична регресія	Точність 0.748
	ANOVA (1 потік)	Дерево прийняття рішень	Точність 0.73
	MUTUAL (1 потік)	DNN	Точність 0.736

Після отримання експериментальних результатів якості різних моделей ШІ у виявлені ботнетів, було б корисно узагальнити причини успіхів чи невдач, а також виділити можливі випадки їх використання.

У ролі базового класифікатора для порівняння можна взяти логістичну регресію. Завдяки своїй простоті ця модель швидко навчається і має найкращу продуктивність при роботі в режимі реального часу. Крім того, отримані при навчанні вагові коефіцієнти можна додатково інтерпретувати, щоб зрозуміти, які саме ознаки потоків роблять найбільший внесок у процес прийняття рішень. Результатом класифікації логістичною регресією є значення впевненості в достовірності гіпотези класифікації. Регуляцію значення порогу впевненості можна використовувати для зменшення кількості помилкових спрацьовувань у разі необхідності. Логістичну регресію можна використовувати для децентралізованих IDS, що працюють з обмеженою ємністю ресурсів. Під час експериментів для класифікатора логістичної регресії показник ефективності AUROC склав 0,778, а точність виявлення – 0,78. Ці значення були покращені шляхом застосування min-max масштабування за формулою (2.1), а також зменшенням розмірності вхідних даних за допомогою методу головних компонент (Principal Componet Analysis, PCA) (див. показники в Табл. 2.2).

Дерева рішень виявились одним з найкращих кандидатів для складної класифікації активності та виявлення загроз. Причина полягає в тому, що ознаки (характеристики) подій мережевої активності завжди мають певне семантичне значення і обираються базуючись на знанні про вже відомі методи та шаблони атак. Ось кілька прикладів таких ознак для виявлення ботнетів на основі мережевих потоків: «чи був у потоці використаний протокол IRC?», «чи було зроблено DNS-запити типу A і скільки разів?», «яка кількість відмов (тобто була відсутня відповідь на прапорець SYN) і чи грубого розриву (зустрівся прапорець RST) відбулася під час TCP-з'єднання?», «скільки даних було передано до розриву з'єднання?», тощо. Людина-експерт використала би цю інформацію для того, щоб винести судження, або принаймні, щоб позначити потік або хост як підозрілі, якщо якісь із цих та інших тверджень будуть правдиві (чи хибні). Дерева прийняття

рішень здатні автоматично будувати «ланцюжки міркувань» на основі вивчених абстрактних умов, які можуть віддалено відображати послідовність людських суджень (наприклад, *якщо X та Y істинно, то, за умови Z … а інакше…*). Тому очікувано, що цей тип класифікатора добре показав себе в ряді робіт [45][61][62] у галузі виявлення. Дерева прийняття рішень, однак, страждають від перенавчання, що є значним недоліком при виявленні вторгнень, оскільки навчальні набори для такого класу задач часто замалі, згенеровані штучно та містять обмежену кількість вибірок трафіку ботнетів чи атак. Це було продемонстровано в експериментальній частині цієї роботи, коли значення AUROC для класифікатора дерев прийняття рішень виявилось лише 0,477 (без використання методу головних компонентів).

Для вирішення згаданої проблеми використовується модель "випадкового лісу" (Random Forest). Випадкові ліси мають значно меншу дисперсію, ціною нижчої точності та більшої системної похибки. Цитата Hastie et al. [63]: *«так як [випадковий ліс] є інваріантним щодо масштабування та інших перетворень значень ознак, він стійкий до включення нерелевантних ознак і утворює моделі, які легко перевірятиВони [випадкові ліси], однак, менш точні»*. Випадкові ліси засновані на техніці класифікації дерев прийняття рішень, але замість використання єдиного дерева, яке вразливе до шумів та схильне до перенавчання, в них використовується група (ансамбль) дерев. Під час навчання для кожного дерева застосовуються вибірки із підмінами, отримані з рівномірного випадкового розподілу. Для рішення задачі регресії підраховується середній результат передбачення кожного окремого дерева. У випадку задачі класифікації використовується «голосування» (вибір класу більшості). Такий підхід надає класифікатору випадкового лісу значну перевагу над деревом прийняття рішень (AUROC = 0,848 та AUROC = 0,91 при застосуванні PCA). Це особливо важливо при виявленні мережевих атак, адже є дуже мало загальнодоступних розмічених наборів даних (датасетів) для навчання, а ті, які надаються, частіше за все є недостатньо репрезентативними, що ставить проблему перенавчання моделей ще більш гостро.

2.8.2. Класифікація груп потоків

Одного ізольованого потоку часто недостатньо для відтворення повної картини мережевої атаки. Тому методам класифікації єдиного потоку може бути складно виявити вторгнення, розтягнуте в часі, особливо якщо комунікація процесів розділена на кілька мережевих потоків. Крім того, як зауважувалось в сумісних роботах на цю тему, іноді, для виявлення зараження хоста ботнетом необхідно брати до уваги обмін даними з багатьма різними віддаленими хостами, тоді як індивідуальний потік навмисне ізолює комунікацію лише двох віддалених процесів. Виходячи з цих міркувань, доцільно класифікувати потоки у контексті їх оточення, іншими словами враховуючи їх відношення одне до одного.

Одним із варіантів такого контекстуального аналізу є логічне групування потоків за одним чи кількома категоріальними атрибутами та побудова нових векторів ознак, які описують всю групу. Іншим варіантом є представлення таких наборів потоків у формі часових рядів, і аналізу кожного нового потоку в контексті попередніх. Для досягнення цього необхідно враховувати часові рамки кожного потоку – всі потоки відсортовуються на базі часових міток їх початку, і таким чином із окремих потоків утворюються часові ряди (див. Рис. 2.6). Після цього можна використовувати різні методи для обробки таких рядів.

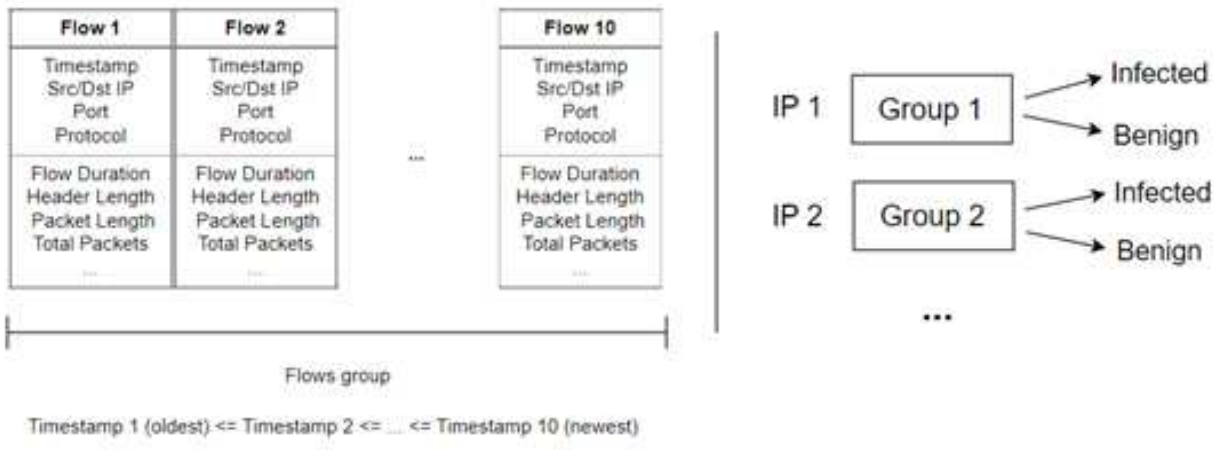

Рис. 2.6. Організація потоків в мережевий ряд з групуванням на основі IP-адреси

Потоки можна розділити на групи за фіксованими часовими інтервалами або ж на групи з однаковою кількістю потоків у кожній. Пізніше ці групи об’єднуються

в один вектор, який передається у класифікатор. Групування незв'язних між собою потоків, однак, може давати нечіткі результатів. Тому, окрім групування лише на основі їх послідовності у часі, доцільно також об’єднувати потоки на основі їхніх внутрішніх характеристик, таких як пара IP-адрес джерело-пункт призначення, портів та протоколів. Оскільки порт і протокол зв’язку зловмисник може змінювати досить регулярно, то щоб охопити повний процес комунікацій між ботом та C&C, мережеві потоки найкраще групувати за IP-адресами. Крім того, класифікація проводиться для всієї групи потоків, а не лише для окремого (ізольованого) потоку, при цьому група розглядається як часовий ряд. Об'єднавши ці два підходи, ми можемо робити висновки про характер всієї комунікації, оскільки буде враховано стан кожного потоку як у часі, так і в просторі подій. У цьому випадку класифікатор приймає вхідні дані як матрицю M×F, де F — кількість ознак, а M — розмір групи потоків. Вхідний навчальний тензор K×M×F, де K — це кількість груп.

Для класифікації часових рядів зазвичай використовуються згорткові нейронні мережі з темпоральними згортками та рекурентні нейронні мережі. Наприклад, такий підхід був використаний Arnaldo et al. у 2017 р. [3], де описано його застосування для даних отриманих з журналів подій (історії з’єднань), а також для даних мережевих потоків, отриманих з датасету ISCX Botnet 2014. У їхньому випадку, однак, це не принесло суттєвих результатів: якість класифікації була гіршою, ніж у випадку аналізу індивідуальних потоків. Хоча їм таки вдалося трохи покращити результат (на 5,6%) після доповнення початкового набору ознак додатково згенерованими.

У ході цієї роботи використано подібний підхід, але з деякими модифікаціями та покращеннями. По-перше, базовий набір ознак для кожного потоку був розширений з 23 до 72, що робить вхідні дані більш репрезентативними. По-друге, групування було проведено лише на основі IP-адреси джерела потоку. Це зроблено з метою визначення інфікованого хоста, який надсилає запити на різні віддалені IP-адреси, намагаючись знайти доступну з метою знаходження C&C або ж для подальшого розповсюдження зараження. Спроби пошуку та встановлення з'єднання з центром командування та контролю притаманні для всіх ботнетів, тому

таке рішення є виправданим. Остання відмінність у тому, що в методі використаному Arnaldo et al. [3] при розгляді груп з більш ніж одним потоком (N = 7, 14, 28) відкидалися групи, що містять менше ніж N потоків, а тому для різних випадків вони тренували та використовували різні моделі класифікаторів. У рамках цієї роботи, на противагу, було обрано розмір статичної групи (N = 10), і для всіх груп, з меншою кількістю елементів, вхідний тензор був доповнений нулями, що є загальноприйнятою практикою при використанні згорток на групах даних неоднорідної розмірності. Разом з методами попередньої обробки та фільтрації, описаними в минулих розділах, використання згорткової нейронної мережі (Convolutional Neural Network, CNN) для класифікації часових рядів дало значно кращі результати у порівнянні з базовою класифікацією окремих потоків звичайною нейронною мережею. Показники AUROC зросли на 6% (до 0,918), що свідчить про ефективність розгляду одразу кількох зв'язаних потоків з врахуванням їх порядку. Застосована CNN модель містила 2 згорткові шари і 2 одновимірних шари стягування. Перший згортковий шар мав розмір ядра 3 з 32 фільтрами, другий — розмір ядра 2 з 16 фільтрами. Цей підхід, однак, має серйозний недолік: його непрактично використовувати для класифікації в реальному часі, оскільки він, вимагає аналізу всіх потоків групи разом. Модель CNN виконує склейку (згортку) кількох значень ознак чи подій у єдине значення вищого рівня. Таку згортку можна зробити за значеннями однієї й тієї ж ознаки, яка змінюються в часі. Така одновимірна "склейка" називається темпоральною згорткою. Вона дозволяє описати часову мінливість ознаки з кількох пов’язаних мережевих потоків. Як було показано, це забезпечує більшу ефективність виявлення. Однак недоліком CNN моделі є те, що вона працює з групами подій фіксованого розміру. Таким чином, щоб отримати кінцевий результат прогнозу, необхідно мати весь ланцюжок потоків, що робить їх непридатними для виявлення вторгнень наживо, але дуже корисними для аналізу історичних даних.

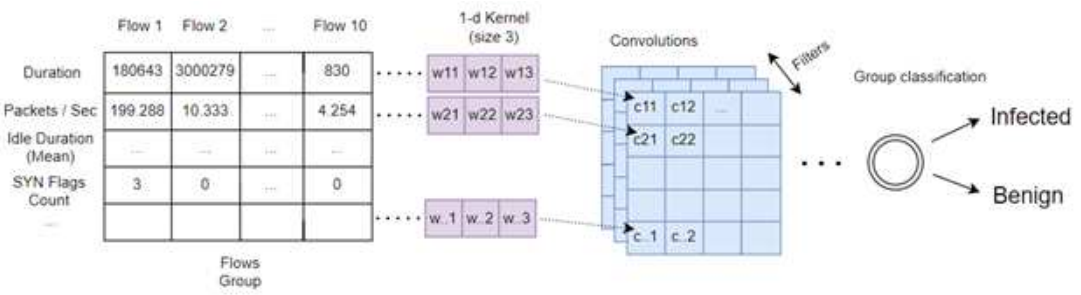

Рис. 2.7. Класифікація часового ряду потоків згортковою мережею

Інший широко відомий підхід до аналізу часових рядів заснований на обробці подій по черзі у порядку їх виникнення, але з урахуванням попередніх подій на кожному наступному кроці. Для цього спочатку генерується вектор «історичних значень ознак» через застосування моделі до події попереднього часового кроку (у нашому випадку через застосування до попереднього потоку в групі). Цей вектор представлятиме історичний контекст для потоку, що аналізується на даному кроці. Потім історичні ознаки комбінуються зі значеннями ознак поточного екземпляра і отриманий вектор передається класифікатору, який вже робить висновок на основі усіх ознак разом. Вхідні дані класифікатора можна описати як (2.2).

$$F_i = F_{historical} \cup F_{current} = (x_1^h, \dots, x_{|F_{historical}|}^h, x_1^c, \dots, x_{|F_{current}|}^c) \qquad (2.2)$$

де $F_{current}$ - вектор ознак поточного екземпляра, та $F_{historical} = E(\bigcup_{current-1}^{k} F_k)$ – вектор історичних ознак, який, для зменшення кількості підрахунків, можна подати як (3).

$$F_{historical} = E(\cup_{current-1}^{k} F_k) = E(F_{historical-1} \cup F_{current-1}) \qquad (2.3)$$

Такий підхід може бути реалізований за допомогою рекурентної нейронної мережі (Recurrent Neural Network, RNN), перевага якої полягає в тому, що серія векторів ознак оброблюється елемент за елементом. Групи потоків обробляються послідовно. Підхід відстежує залежності між подіями серії і рекурентно передає значення, отримані на попередньому кроці (історичні ознаки), на вхід класифікатора в даному кроці. Іншими словами, такі класифікатори можуть «запам’ятовувати» попередні стани та враховувати цей досвід під час виконання майбутніх класифікацій. У цій роботі в якості RNN була використана архітектура

«довгої короткочасної пам’яті» (Long Short-Term Memory, LSTM), яка є більш стійкою до проблеми зникаючих градієнтів порівняно зі звичайною RNN. Показники AUROC досягли значення 0,907, що на 2,37% краще за звичайний аналіз окремих потоків, але все ж гірше за CNN. Моделі LSTM можуть приймати послідовності довільної довжини, видаючи результат класифікації для кожного нового потоку. Точність виявлення спочатку може бути низькою, але поступово підвищується, оскільки модель отримує все більше і більше історичного контексту. Показники AUROC при класифікації груп потоків, а також їх порівняння з іншою роботою, де застосовувався схожий підхід, наведено в Таблиці 2.3.

Таблиця 2.3. Ефективність моделей для потоків представлених як часові ряди (набір ISCX Botnet 2014)

Робота	**Метод групування потоків**	**Класифікатор**	**Показники ефективності (на незнайомих даних)**
Ця робота.	Групування за IP джерела (10 потоків)	CNN (2 згортки + стяжка)	AUROC 0.918
	Групування за IP джерела (10 потоків)	LSTM (100 комірок)	AUROC 0.908
Arnaldo et al. [3]	Групування за IP джерела та призначення (7 потоків)	CNN	AUROC 0.644
	Групування за IP джерела та призначення (14 потоків)	CNN	AUROC 0.633
	Групування за IP джерела та призначення (7 потоків)	LSTM	AUROC 0.624
	Групування за IP джерела та	LSTM	AUROC 0.744

	призначення (14 потоків)		

2.9. Розробка системи аналізу мережевого трафіку «NetWatcher»

NetWatcher – це багатоцільовий аналізатор мережевого трафіку, розроблений в рамках цього дисертаційного дослідження. Першочерговою функцією NetWatcher є моніторинг мережевих даних та виявлення в них слідів трафіку породженого зловмисним ПЗ. Таким чином, NetWatcher виконує роль NIDS (Network Intrusion Detection System). Однією з особливостей цієї розробки є здатність функціонувати як в режимі онлайн, перехоплюючи дані з конкретного мережевого інтерфейсу хоста, на якому вона розташована, так і в режимі офлайн, аналізуючи PCAP дампи пакетів вже зібраних історичних даних.

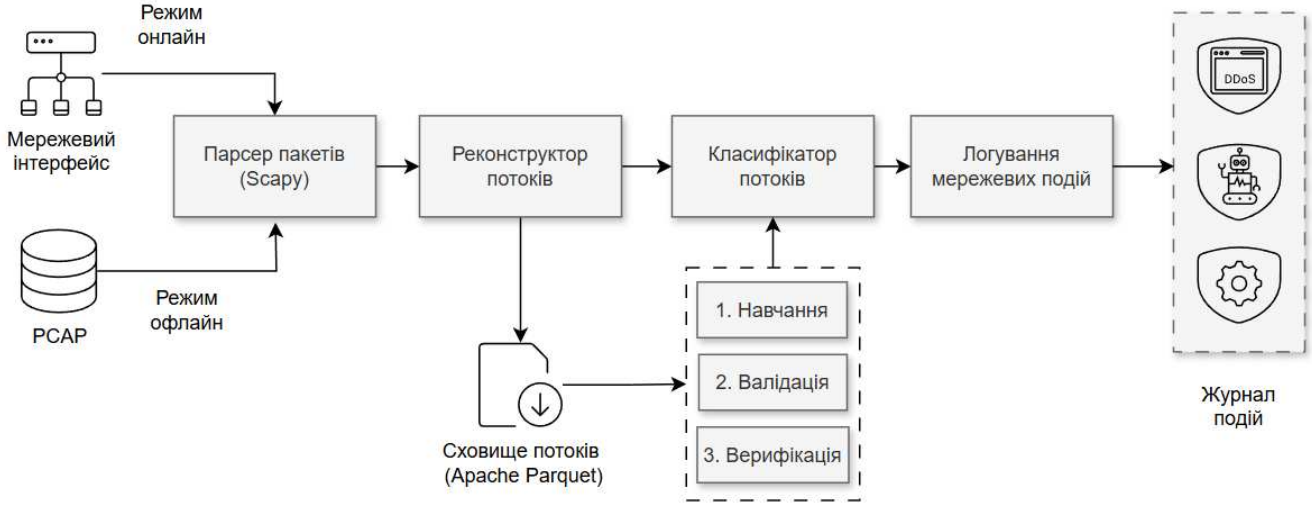

Рис. 2.8. Компоненти системи NetWatcher

Процес аналізу даних в NetWatcher є асинхронним, а сама програма багато-поточна. Розділення ресурсоємних операцій та операцій вводу/виводу на окремі виконавчі потоки є стандартною практикою для поліпшення оптимізації та уникнення можливої втрати даних, які надходять у реальному часі. В цьому контексті під словом «потік» мається на увазі «thread» (потік виконання), а не «flow» (мережевий потік). Для уникнення конфліктів термінів далі, виконавчі потоки будем називати «процесами».

В NetWatcher перший програмний процес займається «сніфінгом» (перехопленням) пакетів з мережевого інтерфейсу складаючи їх в чергу, з відповідними елементами забезпечення міжпотокової взаємодії. Другий процес забирає пакети з черги та проводить «парсинг» (розбір) за допомогою бібліотеки Scapy [64], виділяючи інформацію з заголовків рівні 2-7 OSI моделі та перетворюючи їх в Python-об'єкти. Також, даний процес проводить санітизацію пакету, видаляючи корисне навантаження і залишаючи лише інформацію про його розмір. В майбутньому при розширенні набору ознак, цю логіку можна змінити, й попередньо виділяти з корисного навантаження додаткову інформацію. Описане розділення на окремі виконавчі процеси, обумовлено тим, що процедура розбору пакету в Scapy виявилась досить ресурсоємкою і займає значну частину часу обробки трафіку.

Реконструкція потоків проводиться за логікою описаною в попередніх підрозділах, за що також відповідає окремий (третій) процес. Цей процес отримує пакети з черги від попереднього та знаходить активні мережеві потоки, які можна асоціювати з даним пакетом. Якщо такі потоки відсутні, то створюється новий потік. Якщо пакет містить TCP прапорці FIN або RST, то асоційований активний потік переводиться в стан завершення. Іноді буває так, що при завершенні TCP з'єднання можна спостерігати повторну передачу пакетів – в контексті виявлення зловмисного ПЗ така інформація також є корисною. Для порівняння, CIC-FlowMeter пропускає такі випадки і після отримання єдиного пакету з RST, або ж двох пакетів з FIN прапорцем одразу завершує потік. Це спричиняє появу «загублених» пакетів, не віднесених до потоку, якому вони насправді належать. Окрім втрати інформації, це також породжує додаткові «сміттєві» потоки для загублених пакетів, що негативно відображується на якості результуючих даних в цілому. Для уникнення таких ситуацій, NetWatcher дає кожному TCP потоку додатковий час на завершення (3 секунди за замовчуванням). Всі нові пакети асоційовані з потоком у стані завершення все одно додаються у потік поки не сплине заданий час. Виключення становлять пакети з SYN прапорцем, адже отримання такого пакету одразу після завершуючої послідовності прапорців

означає спробу перевстановлення з'єднання однією зі сторін. В цьому випадку старий потік завершується одразу і замість нього створюється новий. Ще однією відмінністю від CIC-FlowMeter є введення додаткової (четвертої) умови звершення потоку «за тайм-аутом простою», яка була описана в попередніх підрозділах. CIC-FlowMeter підтримує лише завершення потоку за послідовністю TCP прапорців та за часом сумарної тривалості потоку.

Після остаточного завершення потоку NetWatcher проводить підрахунок характеристичних атрибутів, на основі яких далі здійснюється класифікація. Повний перелік назв та описи всіх атрибутів, які виділяє цей інструмент, наведено в **Додатку Г**.

NetWatcher може працювати в двох режимах: «спостерігач» (observer) та «детектор» (detector). У режимі спостерігача процес аналізу мережевих даних завершується на етапі реконструкції потоків. Результуючий набір зберігається на диск у форматі Apache Parquet [65], що робить його легкодоступним для подальшого дослідження та аналізу іншими інструментами. В режимі детектора NetWatcher функціонує як система виявлення мережевих загроз і замість збереження потоків здійснює їх класифікацію заздалегідь навченою інтегрованою моделлю ШІ. Результати класифікації записуються в спеціальний журнал подій разом з додатковою корисною інформацією про кожен потік (IP-адреси, порти, протокол, час і т.д.).

```
2025-01-07 17:47:33 [ALERT] 192.168.1.103:1090 -> 192.168.5.122:143 (TCP): 9 packets, 0 bytes, 959s 948ms
2025-01-07 17:47:33 [OK] 192.168.2.113:4491 -> 192.168.5.122:22 (TCP): 20 packets, 1278 bytes, 5s 399ms
2025-01-07 17:47:33 [OK] 192.168.1.103:1705 -> 192.168.5.122:22 (TCP): 20 packets, 1278 bytes, 5s 478ms
2025-01-07 17:47:33 [OK] 192.168.2.113:4492 -> 192.168.5.122:22 (TCP): 22 packets, 1278 bytes, 5s 446ms
2025-01-07 17:47:33 [ALERT] 192.168.1.103:1706 -> 192.168.5.122:22 (TCP): 22 packets, 1278 bytes, 5s 395ms
```

Рис. 2.9. Приклад записів у журналі подій NetWatcher

В підсумку, стандартний цикл використання NetWatcher можна описати в кілька етапів. Спочатку, в лабораторних умовах проводиться виділення потоків та їх атрибутів з наявних мережевих даних отриманих з відомих джерел за допомогою режиму спостерігача. Дані можуть бути як історичним, так і перехоплюватись наживо. Після цього виконується розмітка груп потоків на класи і на їх основі навчається модель класифікації стандартними методами штучного інтелекту. Далі

модель інтегрується в NetWatcher після чого його можна використовувати в режимі детектора у ролі NIDS.

Код NetWatcher був опублікований у репозиторії [66] на ресурсі Github, де також містяться детальні інструкції та приклади використання. Також були опубліковані виконувані файли програми для OS Windows та Linux. Експериментальний прототип даного застосунку було впроваджено на підприємстві «ТОВ «НВП Радікс». Акт про впровадження надано у **Додатку Б**.

2.10. Застосування NetWatcher та ШІ для виявлення ботнет-трафіку

В цьому підрозділі описано результати навчання та оцінку якості різних моделей класифікації мережевих даних ШІ на основі потоків виділених за допомогою NetWatcher. Кожну з цих моделей можна інтегрувати в даний застосунок, для аналізу мережі з ціллю виявлення слідів активності ботів. Хоча використовувалися набори даних трафіку саме ботнетів, однак аналогічний підхід може бути використаним і для виявлення мережевих загроз інших типів. Крім того, зазвичай, трафік ботів також містить приклади різних атак проведення ними на інші інтернет-ресурси. Відповідно можна очікувати, що навчені моделі також будуть здатні вирізняти деякі з цих атак, незалежно від того чи породжені вони ботнетом, чи іншим зловмисним ПЗ.

При навчанні та оцінці якості моделей сумарно були використані 4 набори даних:

1. ISCX Botnet 2014 [20] (згадувався у попередніх розділах) використовувався для навчання і для тестування.
2. CTU-13 [67] (лише трафік зловмисного ПЗ) використовувався для тестування.
3. CTU-Custom – новий набір, створений спеціально для цієї роботи на основі даних зібраних Malware Capture Facility Project. Включає в себе як трафік зловмисного ПЗ [68] так і звичайний трафік [69]. Використовувався для тестування.

4. ISCX-CTU-Extended – новий набір, який є комбінацією ISCX Botnet 2014, CTU-13 та CTU-Custom. Використовувався для навчання та для тестування.

При виділенні потоків з усіх наборів даних за допомогою NetWatcher використовувався однаковий набір параметрів реконструктора, а саме:

- Тайм-аут простою потоку (idle_timeout) – 600 секунд.
- Тайм-аут активності потоку (activity_timeout) – 1000 секунд.
- Час дозволу повторних передач пакетів після завершення TCP з'єднання – 3 секунди.
- Фільтрація пакетів за протоколом – TCP.
- Відсіювались всі потоки, в яких менше 3 пакетів.

2.10.1. Вимоги до моделі класифікації мережевих потоків

До моделі виявлення ботнетів ставляться дві основні вимоги, які мають бути виконуватись одночасно, щоб систему було доцільно використовувати в реальних умовах. Перша вимога – це високий рівень виявлення трафіку ботнетів. Тобто модель має коректно класифікувати достатньо високий відсоток трафіку породженого ботом, щоб можна було зробити висновок про наявність проблеми та локалізувати її джерело. Друга, не менш важлива вимога, – це низький рівень хибно-позитивних результатів. Тобто звичайний трафік у мережі, який не належить зловмисному ПЗ, має бути класифікований відповідно. При цьому, чим більший об'єм звичайного трафіку, який класифікатор аналізує за звичайних умов, тим критичнішою стає ця вимога. Обумовлено це тим, що при великій кількості хибно-позитивних результатів реально зафіксовані події виявлення ботів будуть просто губитися серед всієї безлічі хибних спрацювань, що робить всю систему марною.

Кожній з зауважених вимог, можна поставити у відповідність класичні метрики оцінки ефективності класифікації. Для задовільнення першої вимоги необхідно максимізувати значення «повноти», яка також відома як «чутливість» (англ. recall) і вимірюється як відношення кількості істинно позитивних результатів до сумарного числа позитивних прикладів у вибірці. Для задовільнення другої

вимоги, необхідно мінімізувати значення показника «хибнопозитивного рівня», яке вимірюється як відношення кількості хибнонегативних результатів до сумарного числа позитивних прикладів. При цьому, на практиці для реального використання є сенс обирати лише ті моделі, в яких хибнопозитивний рівень не перевищує певної позначки. З цих самих міркувань також обирається порогове значення впевненості на виході класифікатора, при перевищенні якого гіпотеза класифікації приймається. В рамках цієї роботи, за бажане значення хибнопозитивного рівня бралося $\leq 0.1\%$, а за максимально допустиме – 0.2%. Тобто бажано, щоб з 1000 звичайних мережевих потоків модель хибно класифікувала не більше 1 потоку як «злоякісний».

Окремою, не менш важливою, вимогою є диверсифікація множини класів ботів, які модель здатна виявляти. Навіть відносно невисокий відсоток виявлення потоків певного класу атак може бути корисним, адже всього кількох виявлених екземплярів може бути достатньо для реєстрації факту наявності загрози.

Також варто зауважити про важливість підтримувати обчислювальну простоту моделей класифікації для збереження швидкості висновування, що критично в системах реального часу. Саме з цих міркувань у подальших експериментах використовувались досить прості архітектури моделей, робота яких не створює суттєвого навантаження на систему.

2.10.2. Набір даних ISCX Botnet 2014

Набір даних ISCX Botnet 2014 брався за основу для перших версій моделей, які були інтегровані у розроблену систему виявлення ботнетів. Набір вже заздалегідь розділений на навчальну та тестувальну вибірки. Після проведення реконструкції потоків за допомогою NetWatcher, було отримано набори потоків для кожної з вибірок. Статистика наведена у Таблиці 2.4.

Таблиця 2.4. Розподіл потоків в ISCX Botnet 2014

Клас	**Навчальна вибірка**	**Тестувальна вибірка**
Звичайні	93110 (66.16%)	27488 (25.49%)
Шкідливі	47633 (33.84%)	80345 (74.51%)

ISCX Botnet 2014 відомий тим, що поєднує в собі кілька наборів даних і містить достатньо репрезентативний набір прикладів трафіку різних типів ботнетів. Слід зауважити, що в навчальній вибірці міститься лише 4 типи, а саме: Neris, Rbot, Virut, та трафік невідомих IRC ботнетів. Тестувальна вибірка є значно більш варіативною, і крім зауважених типів містить Menti, Murlo, Sogou та Weasle. Також, в оригінальному тестовому наборі містилися приклади Black hole, TBot, Zero access та Zeus, однак число прикладів для них дуже невелике, і водночас жодна з навчених моделей не показала для них позитивних результатів. Тому для уникнення зашумлення підсумкових результатів, потоки цих класів були виключені з тестового набору. В Таблиці 2.5 наведена статистика розподілу за класами зловмисного ПЗ в навчальній та тестовій вибірках.

Таблиця 2.5. Розподіл потоків за класами шкідливого програмного забезпечення в ISCX Botnet 2014.

Клас ПЗ	**Навчальна вибірка**	**Тестувальна вибірка**
IRC bot [70]	2439 (5.12%)	250 (0.31%)
Neris [71]	12020 (25.23%)	18780 (23.37%)
Rbot [72]	32324 (67.86%)	154 (0.19%)
Virut [73]	850 (1.78%)	32740 (40.75%)
Menti (вірогідно DonBot [74])	–	2747 (3.42%)
Murlo [75]	–	2593 (3.23%)
Sogou [76]	–	40 (0.05%)
Weasel [17]	–	23041 (28.68%)

При навчанні використовувались 4 різні архітектури класифікаторів, а саме: логістична регресія (log_reg), випадковий ліс з глибиною 9 (rf_9) та дві 3-рівневі повнозв’язні нейромережі з 16 та 24 нейронами на кожному рівні (dnn_16_16_16 та dnn_24_24_24 відповідно). Також, окремо проводилось навчання цих моделей з попереднім застосуванням методу головних компонент (PCA, principal component

analysis) для переходу до нового простору ознак з розмірністю 12. Моделі з застосуванням PCA будуть позначатися префіксом “pca_12”. При навчанні нейромереж, випадкові 20% навчальної вибірки використовувались у ролі «притриманого набору» (validation split).

Таблиця 2.6. Показники AUROC для моделей навчених на ISCX Botnet 2014 (Тестувальна вибірка).

Модель	**36 ознак**	**12 ознак (PCA)**
iscx-botnet-2014/log_reg	0.7565	0.7532
iscx-botnet-2014/rf_9	0.9563	0.9165
iscx-botnet-2014/dnn_16_16_16	0.7239	0.7361
iscx-botnet-2014/dnn_24_24_24	0.7384	0.9006

Як видно з Таблиці 2.6, ефективність моделей класифікації на даній вибірці суттєво відрізняється, водночас. Варто звернути увагу, що на відміну від експерименту проведеного в підрозділі 2.8.1, в даному експерименті також включалися зразки ботнету Weasel, які складали 28.68% від тестувальної вибірки. Ботнет Weasle більшість з моделей не змогли виявити, що негативно позначилось на значеннях AUROC. Це однак, не завадило отримати показники ефективності порівнювані з отриманими в експерименті де потоки виділялись інструментом CIC-FlowMeter (Таблиця 2.2). При цьому для моделей випадкового лісу, а також комбінації нейромережі з методом головних компонент (pca_12_dnn_24_24_24) отримані значення виявились суттєво вищими, що свідчить про якість виділення потоків розробленим інструментом NetWatcher.

Слід зауважити, що в контексті побудови системи виявлення вторгнень не можна покладатись лише на показники AUROC, адже як зауважувалось, до моделей ставляться вимоги про низьких хибнопозитивний рівень. Як виявилось, при встановлені порогового значення на рівні, необхідного для задовільнення даного критерію, повнота та точність виявлення суттєво падають для більшості моделей. Саме тому, високі значення AUROC можуть бути оманливими.

Наприклад, серед всіх моделей, звичайна логістична регресія у комбінації з методом головних компонент (pca_12_log_reg) показала найкращий результат повноти виявлення за умови не перевищення позначки 0.1% хибнопозитивного рівня. Як видно з Таблиці 2.7, модель досягає повноти виявлення 56.6% при цьому маючи лише 0.09% хибнопозитивних результатів.

Таблиця 2.7. Показники моделі iscx-botnet-2014/pca_12_log_reg на ISCX Botnet 2014 (Тестувальна вибірка)

Поріг	**Точність**	**Повнота**	**Хибнопозитивний рівень**
0.51	0.7193	0.6265	0.0092
0.60	0.7186	0.6245	0.0063
0.70	0.7163	0.6206	0.0041
0.80	0.6835	0.5759	0.0020
0.85	0.6796	0.5703	0.0011
0.90	**0.6765**	**0.5661**	**0.0009**
0.95	0.5380	0.3801	0.0004
0.97	0.5371	0.3788	0.0002

Для порівняння випадковий ліс (pca_12_rf_9) показує повноту 44.7% (Таблиця 2.8), а нейромережа (pca_12_dnn_16_16_16) – 43.5% (Таблиця 2.9) при хибнопозитивному рівні 0.08%.

Таблиця 2.8. Показники моделі iscx-botnet-2014/pca_12_rf_9 на ISCX Botnet 2014 (Тестувальна вибірка)

Поріг	**Точність**	**Повнота**	**Хибнопозитивний рівень**
0.51	0.5972	0.4603	0.0024
0.60	**0.5882**	**0.4476**	**0.0008**
0.70	0.5827	0.4400	0.0001
0.80	0.5744	0.4288	0.0001
0.85	0.5700	0.4229	0.0000

Таблиця 2.9. Показники моделі iscx-botnet-2014/pca_12_dnn_16_16_16 на ISCX Botnet 2014 (Тестувальна вибірка)

Поріг	**Точність**	**Повнота**	**Хибнопозитивний рівень**
0.51	0.6264	0.5020	0.0101
0.60	0.6215	0.4950	0.0088
0.70	0.6178	0.4892	0.0063
0.80	0.6083	0.4755	0.0034
0.85	0.6007	0.4649	0.0023
0.90	0.5946	0.4565	0.0018
0.95	0.5819	0.4394	0.0014
0.97	**0.5787**	**0.4348**	**0.0008**

В кожній таблиці виділено рядок, який містить мінімальне порогове значення при якому хибнопозитивний рівень не перевищує бажаний (0.01%) або ж критичний (0.02%). Фактично, поріг прийняття рішення для кожної з моделей обирається з міркувань знаходження компромісу між високою повнотою виявлення трафіку зловмисного ПЗ та низьким відсотком хибнопозитивних результатів для нормального трафіку у системі, за якою ведеться спостереження.

Іншим, не менш важливим, фактором у виборі архітектури моделі та вихідного порогу прийняття рішень є варіативність цільової множини класів зловмисного ПЗ. Наприклад, недоліком класифікатора на основі логістичної регресії є низька кількість типів ботів, які вона здатна виявляти. На Рис. 2.10 видно, що при встановлені порогового значення на рівні 0.9, дана модель може виявити лише 3 із 8 видів трафіку ботів, присутніх у тестовому наборі (а саме Virut, Menti, та Neris),.

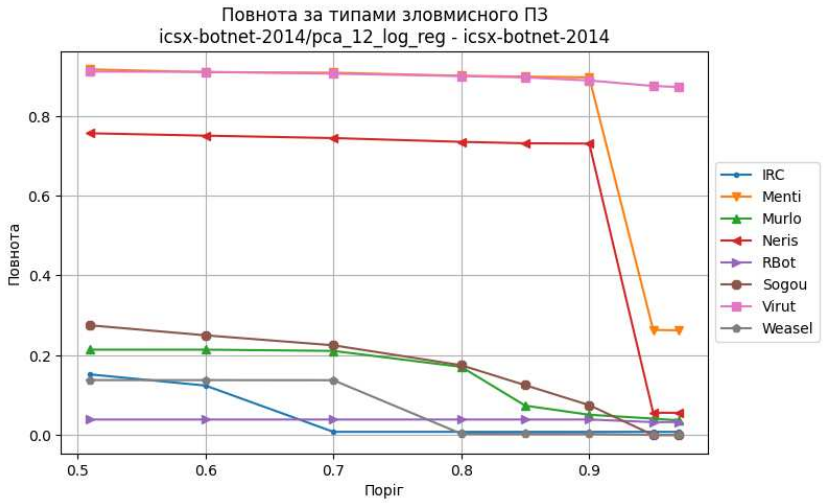

Рис. 2.10. Показники повноти класифікатора pca_12_log_reg для різних типів зловмисного ПЗ (ISCX Botnet 2014)

Для порівняння (див. Рис. 2.11), модель випадкового лісу може з високою точністю виявляти 4 типи ботнет (IRC, Menti, Virut та Rbot), і з дещо нижчою точністю виявляти п’ятий тип Murlo. Цікаво, що на відміну від логістичної регресії, випадковий ліс зміг виявити лише невеликий відсоток трафіку Neris, хоча й був навчений на тих самий даних. Звісно, така поведінка може суттєво залежати від початкової ініціалізації параметрів моделі, що є характерно для багатьох моделей ШІ.

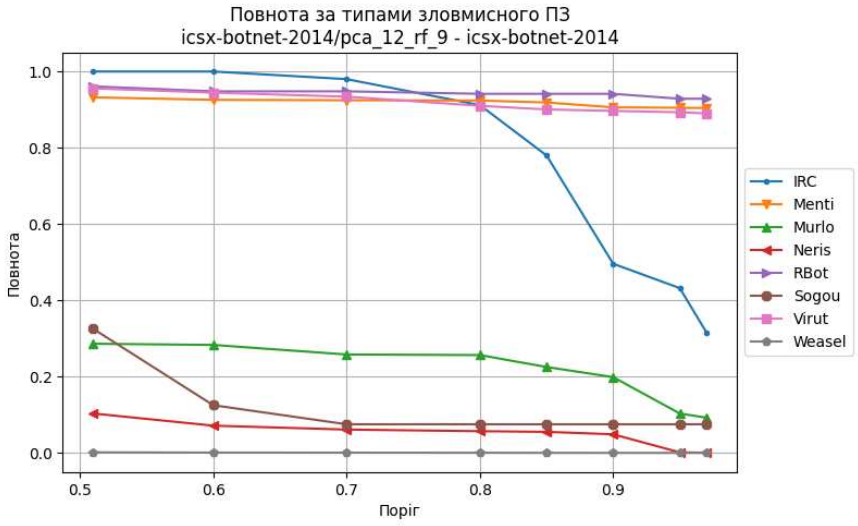

Рис. 2.11. Показники повноти класифікатора pca_12_rf_9 для різних типів зловмисного ПЗ (ISCX Botnet 2014)

Схожими характеристиками також володіє класифікатор на основі нейромережі (pca_12_dnn_16_16_16) як показано на Рис. 2.12. У порівнянні з

моделлю випадкового лісу, цей класифікатор здатний виявляти Murlo та Sogou з суттєво вищою точністю, але водночас, значення хибнопозитивного рівня дещо зростають.

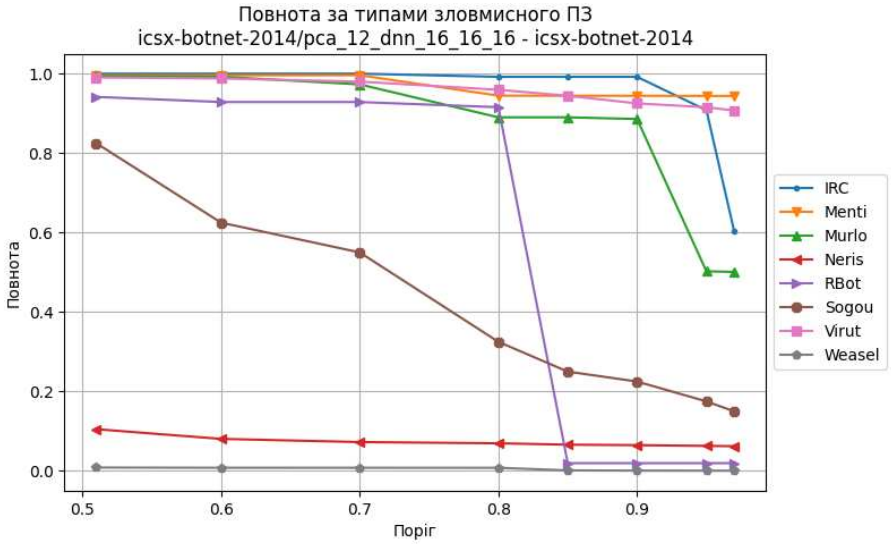

Рис. 2.12. Показники повноти класифікатора pca_12_dnn_16_16_16 для різних типів зловмисного ПЗ (ISCX Botnet 2014)

Цікаво зауважити, що жодна з навчених на даних ISCX Botnet 2014 моделей не змогла виявити потоки трафіку Weasle, що може свідчити про те, що шаблони поведінки цього ботнету фундаментально відрізняють від ботів наявних у навчальній вибірці. З іншого боку, деякі моделі здатні виявляти потоки Menti та Murlo, які в навчальній вибірці були повністю відсутні, що може свідчити про їх поведінкову спорідненість з Neris, RBot або Virut.

За результатами цього експерименту можна зробити висновок про важливість оцінки багатьох факторів при навчанні та виборі моделі класифікації для систем виявлення мережевих загроз. Не можна покладатися лише на одну метрику, але треба враховувати різні показники якості, в особливості залежність між повнотою виявлення та хибнопозитивним рівнем. Крім того, особливу увагу треба приділяти диверсифікованості множини класів зловмисного ПЗ, які модель здатна виявляти, а також побудові достатньо репрезентативної навчально вибірки.

2.10.3. Набір даних CTU-13

CTU-13 [18] – це ще один відомий набір мережевих даних ботнетів, який часто використовувався дослідниками в області побудови систем виявлення вторгнень. Частина набору, яка представляє звичайний (нешкідливий) трафік в цьому наборі доступна лише в санітизованому вигляді, де кожен з пакетів був обрізаний до певного фіксованого розміру (54 байти для TCP пакетів), що зроблено для анонімізації даних та уникнення витоку чутливої інформації. На жаль, моделі ШІ здатні дуже швидко призвичаюватись до неприродних деформацій в даних, що робить цю частину набору мало корисною для поставленого завдання. Відповідно, для цього експерименту будуть розглядатися лише дані ботнетів з цього набору.

CTU-13 не містить розділення на навчальну та тестувальну вибірки, тому було прийнято рішення використовувати весь набір лише для тестування. Це дозволить оцінити генералізуючі властивості моделей, навчених на ISCX Botnet 2014, адже мережеві пакети CTU-13 перехоплені в іншому мережевому середовищі і цей набір включає приклади трафіку ботів, відсутні в попередньому.

Також варто зауважити, що при обробці CTU-13 з нього було виключено частини 51 («сценарій 10») та 52 («сценарій 11»). В цих сценаріях більшу частину трафіку складають UDP-flood та ICMP атаки, і як було зазначено, в рамках експериментів NetWatcher розглядає лише TCP трафік.

Таблиця 2.10. Розподіл потоків за класами шкідливого програмного забезпечення в CTU-13.

Клас ПЗ	**Кількість потоків**	**%**
DonBot (Menti)	2747	1.75
Murlo	2592	1.65
Neris	85137	54.24
NSIS.ay	370	0.24
RBot	32489	20.70
Sogou	40	0.03
Virut	33590	21.40

В таблиці 2.10 наведено розподіл кількості потоків виділених за допомогою NetWatcher за типами ботів, якими вони були породжені. Основну частину складає Neris, Rbot та Virut, які також фігурували в ISCX Botnet 2014. Новими входженнями є DontBot та NSIS.ay. При більш близькому аналізі TCP пакетів DontBot, можна зробити висновок, про їх схожість з пакетами Menti присутніми у вибірці ISCX.

Як згадувалось раніше, в CTU-13 розглядалися лише приклади потоків, які належать ботам. Відповідно, повнота (чутливість) – це єдина метрика, яка має зміст в контексті задачі бінарної класифікації на даних з єдиним класом. В таблиці 2.11 наводяться показники повноти виявлення для моделей навчених на ISCX Botnet 2014 з попереднього підрозділу.

Таблиця 2.11. Показники повноти (чутливості) на наборі CTU-13 класифікаторів навчених на ISCX Botnet 2014.

Поріг	**rf_9**	**pca_12_log_reg**	**pca_12_rf_9**	**pca_12_dnn_16_16_16**
0.51	**0.8450**	0.7952	0.7915	0.8518
0.60	0.8181	0.7922	**0.7689**	0.8424
0.70	0.7880	0.7858	0.7524	0.8267
0.80	0.7607	0.7625	0.7231	0.7885
0.85	0.7342	0.7552	0.7090	0.7637
0.90	0.7084	**0.7435**	0.6990	0.7382
0.95	0.6898	0.5782	0.6860	0.7134
0.97	0.6855	0.5760	0.6816	**0.6972**

Як видно з таблиці, результати попередньо навчених моделей на наборі даних ISCX Botnet 2014 можуть бути перенесені і на CTU-13. У виділених клітинках наводяться показники для порогових значень при яких хибнопозитивний рівень класифікатора не перевищував 0.1% на попередній вибірці. При цьому, найкраще значення показав звичайний випадковий ліс, з повнотою виявлення 84.5% при пороговому значенні 0.51.

Класифікатори на основі алгоритму випадкового лісу здатні ефективно виявляти DonBot, Rbot та Virut, та виявляти Neris з середньою ефективністю. На жаль показник виявлення трафіку NSIS.ay, який не був присутній у ISCX Botnet 2014, для цих моделей дуже низький.

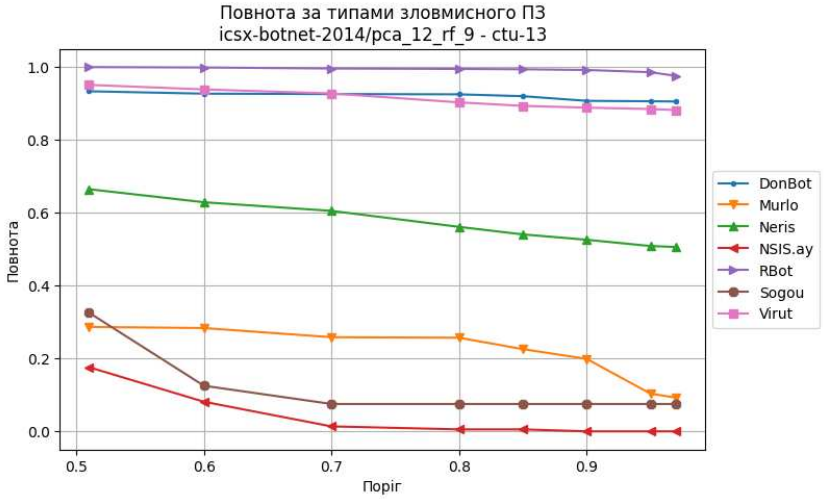

Рис. 2.14. Показники повноти класифікатора pca_12_rf _9 за типами зловмисного ПЗ (CTU-13)

Сумарні показники повноти виявлення для нейронних мережі дуже схожі, але нижчі. Як і очікувалось, з дещо вищою ефективністю нейромережі можуть виявити трафік Murlo (див. Рис. 2.15), з яким випадкові ліси мали проблеми і на попередній вибірці. Також нейромережі здатні розрізняти NSIS.ay, але для цього необхідно підвищувати порогове значення прийняття рішень класифікатора, що може бути неприйнятним.

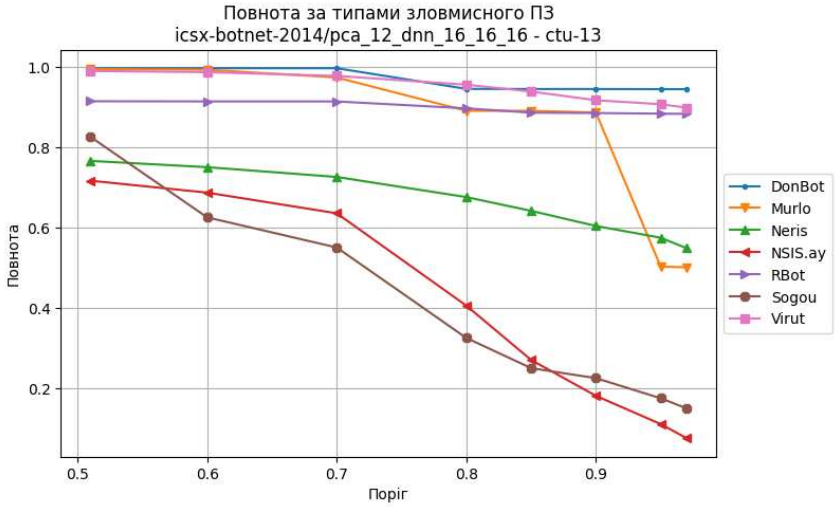

Рис. 2.15. Показники повноти класифікатора pca_12_dnn_16_16_16 за типами зловмисного ПЗ (CTU-13)

В підсумку, моделі навчені на попередньому наборі (ISCX Botnet 2014), зберігають свою ефективність навіть для даних перехоплених в іншому мережевому середовищі. Також моделі володіють достатніми узагальнюючими властивостями і можуть виявляти ботнети, приклади для яких не були присутні в початковому наборі.

2.10.4. Новий набір на основі проекту CTU

З часів публікації наборів CTU-13 та ISCX Botnet 2014 з'явилося безліч нових небезпечних ботнетів та інших типів зловмисного ПЗ, з суттєво більшими масштабами зараження в порівнянні з багатьма своїми попередниками. Існуючі набори швидко застарівають та потребують оновлення. З цією метою в рамках цього дисертаційного дослідження був сформований новий набір навчальних та тестувальних мережевих даних. Цей набір являється розширенням двох попередніх наборів новішими прикладами трафіку опублікованими Stratosphere Lab в рамках проекту «Malware Capture Facility Project». З даного проекту були взяті PCAP-дампи пакетів отримані з машин заражених ботами Zeus 2.1 [52], Kazy [77] (споріднений до Zeus), Emotet [78], TrickBot [79], а також хробаком WannaCry [80]. Приклади потоків перерахованих вірусних програм не були наявні у попередньо згаданих наборах і, як показали експерименти, моделі навчені на ISCX Botnet 2014 мають надзвичайно обмежені можливості їх виявлення. Наприклад, повнота виявлення потоків Zeus 2.0, Emotet та TrickBot не перевищила 1% для жодної з моделей. Потоки породжені WannaCry змогла виявити лише модель pca_12_dnn_24_24_24, з показником повноти в районі 22% незалежно від порогового значення. Виключенням був Kazy, повнота виявлення якого для всіх моделей становила від 15% до 22%. Це можна пояснити тим, що зразки трафіку Kazy зібрані ще в 2012, тому поведінка цього ботнету споріднена з більш старими, наявними в ISCX Botnet 2014.

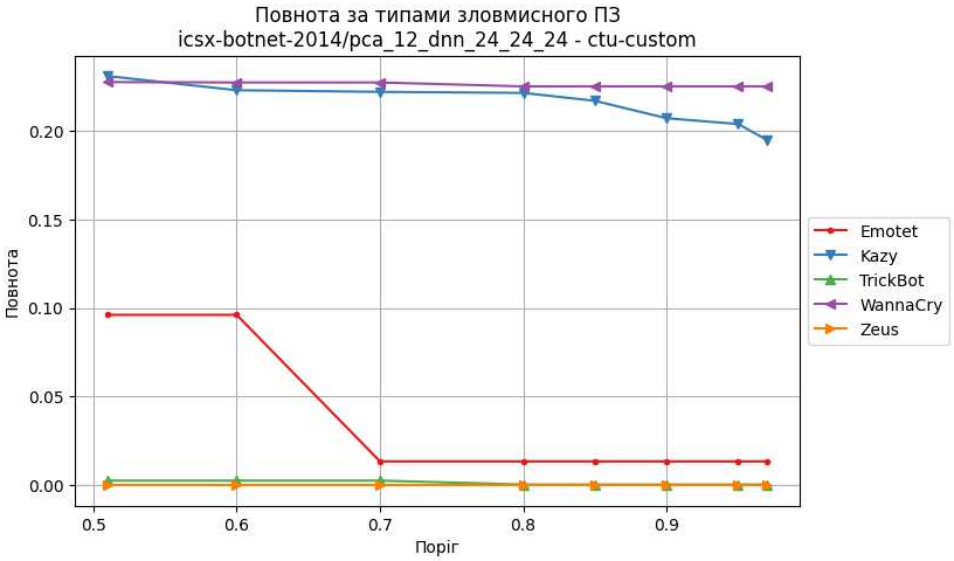

Рис. 2.16. Показники повноти класифікатора pca_12_dnn_24_24_24 за типами зловмисного ПЗ (Тестувальна вибірка CTU-Custom)

Для зрозумілості, далі під терміном «CTU-Custom» будемо мати на увазі безпосередньо згадані розширюючі дані взяті з Malware Capture Facility, а весь комбінований набір «ISCX Botnet 2014 + CTU-13 + CTU-Custom» будемо називати «ISCX-CTU-Extended». Крім того, в CTU-Custom було включено приклади звичайного P2P та HTTP трафіку, які також надані Malware Capture Facility. Це зроблено для підтримки збалансованості класів у навчальній вибірці, а також для оцінки хибнопозитивного рівня на нормальних даних перехоплених в мережевому оточенні відмінному від ISCX Botnet 2014.

Таблиця 2.16. Розподіл потоків в ISCX-CTU-Extended

Клас	**Навчальна вибірка**	**Тестувальна вибірка**
Звичайні	139178 (48.06%)	46477 (7.46%)
Шкідливі	150386 (51.94%)	576823 (92.54%)

Як видно з Таблиці 2.16 отримана навчальна вибірка залишається збалансованою, що є важливим фактором для отримання задовільних показників якості класифікаторів. Крім цього, акцент у цій роботі робився перш за все на репрезентативності вибірки даних. В Таблиці 2.17 наведено фінальний розподіл потоків за класами шкідливого ПЗ в ISCX-CTU-Extended.

Таблиця 2.17. Розподіл потоків за класами шкідливого програмного забезпечення в ISCX-CTU-Extended.

Клас ПЗ	**Навчальна вибірка**	**Тестувальна вибірка**
Emotet	32019 (21.3%)	148905 (25.9%)
IRC	2439 (1.6%)	250 (0. 0004%)
Kazy	4991 (3.3%)	11916 (2.1%)
Neris	12020 (8.0%)	103917 (18.1%)
RBot	32324 (21.5%)	32643 (5.7%)
TrickBot	19999 (13.3%)	153768 (26.8%)
Virut	850 (0.6%)	66330 (11.6%)
WannaCry	5744 (3.8%)	4531 (0.8%)
Zeus	40000 (26.6%)	20393 (3.6%)
DonBot (Menti)	–	2747 (0.5%)
Murlo	–	5185 (0.9%)
NSIS.ay	–	370 (0.0006%)
Sogou	–	80 (~0.%)
Weasel	–	23041 (4.0%)

Як видно з таблиці 2.17, суттєву частину навчальної вибірки складають TrickBot, Emotet, Rbot та Zeus, виявлення яких є найбільш критичним завданням. В тестувальній вибірці у невеликих кількостях фігурує трафік 5 ботів, що не був присутнім у навчальній.

2.10.5. Навчання та оцінка якості класифікаторів на наборі ISCX-CTU-Extended

Після навчання моделей класифікації на комбінованому розширеному наборі даних ISCX-CTU-Extended було виділено кілька моделей з найкращими сумарними показниками ефективності. На жаль, логістична регресія справляється з цим набором суттєво гірше за інші моделі, оскільки це лінійна модель. Тому далі розглядаються лише алгоритм випадкового лісу (rf_9) та штучні нейронні мережі

(dnn_16_16_16, та dnn_24_24_24). Як видно з Таблиці 2.18, використання нейронних мереж показало вищі показники AUROC ніж використання алгоритму випадкового лісу. При цьому у всіх сценаріях, досліджувані показники є суттєво вищими ніж для моделей, навчених на базовому ISCX Botnet 2014. Слід зауважити, що в цьому експерименті, отримані значення AUROC варто інтерпретувати дещо інакше, адже в розширеному наборі даних значну частину тестувальної вибірки складають приклади шкідливого ПЗ.

Таблиця 2.18. Показники AUROC для моделей навчених на ISCX-CTU-Extended (Тестувальна вибірка).

Модель	**36 ознак**	**12 ознак (PCA)**
iscx-ctu-extended/rf_9	0.9563	0.9545
iscx-ctu-extended/dnn_16_16_16	0.9714	0.9728
iscx-ctu-extended/dnn_24_24_24	0.9751	0.973

Як і в минулих експериментах, при виборі порогового значення класифікації будем перш за все зважати на хибнопозитивний рівень. Тестувальна вибірка також була розширена новими прикладами нормального трафіку, тому очікується, що отримані на цих даних показники будуть більш надійні ніж усі попередні. Для подальшого розгляду із всього набору навчених моделей було обрано дві, які показують високе значення повноти виявлення при достатньо низькому хибнопозитивному рівні, а саме pca_12_rf_9, dnn_24_24_24.

В таблиці 2.19 наведені показники для класифікатора на основі алгоритму випадкового лісу при застосуванні методу головних компонент. Як видно з таблиці, для порогового значення 0.7 кількість хибнопозитивних результатів не перевищує 0.7% при цьому, 73.4% трафіку зловмисних програм модель класифікує коректно. Якщо зменшити поріг до 0.6, то хибнопозитивний рівень зростає до 0.13%, і водночас суттєво зростає повнота виявлення до 79.87% всіх «шкідливих» потоків.

Таблиця 2.19. Показники моделі iscx-ctu-extended/pca_12_rf_9 на ISCX-CTU-Extended (Тестувальна вибірка)

Поріг	Точність	Повнота	Хибнопозитивний рівень
0.51	0.8207	0.8074	0.0140
0.6	**0.8136**	**0.7987**	**0.0013**
0.7	**0.7540**	**0.7342**	**0.0007**
0.8	0.6429	0.6142	0.0005
0.85	0.5549	0.5191	0.0004
0.95	0.4989	0.4585	0.0002

Повнозв'язна 3-рівнева нейромережа з 24 нейронами на рівень має вищий відсоток хибнопозитивних результатів, але й водночас вищу максимальну повноту виявлення. Як показано у Таблиці 2.2.0, при встановлені порогового значення впевеності 0.97, повнота виявлення становить 80.7%. Однак якщо допускається хибнопозитивний рівень 0.5%, то при пороговому значенні 0.7 повнота виявлення шкідливого трафіку складає 90.3%, що є суттєво вищим показником у порівнянні з усіма результатами, які показало дерево прийняття рішень.

Таблиця 2.20. Показники моделі iscx-ctu-extended/dnn_24_24_24 на ISCX-CTU-Extended (Тестувальна вибірка)

Поріг	Точність	Повнота	Хибнопозитивний рівень
0.51	0.9195	0.9137	0.0075
0.6	0.9152	0.9089	0.0064
0.7	0.9102	0.9034	0.005
0.8	0.8824	0.8733	0.0044
0.85	0.8778	0.8682	0.0032
0.9	0.8686	0.8583	0.0029
0.95	0.8212	0.807	0.0019
0.97	**0.8161**	**0.8014**	**0.0014**

Зауважимо, що нейромережі з 16 нейронами на рівень, показали схожі результати. Однак, для них показник повноти росте не настільки швидко з ростом

рівня хибнопозитивних результатів. Наприклад, для dnn_16_16_16 при хибнопозитивному рівні 0.5% повнота виявлення складала лише 82.7%. Таким чином, можна зробити припущення, що при збільшенні розміру навчальних даних також варто збільшувати складність нейромережі для отримання кращих результатів.

Для обох моделей pca_12_rf_9 та dnn_24_24_24 була проведена оцінка повноти виявлення для кожного з підкласів трафіку шкідливого ПЗ. Для початку, варто розглянути, на скільки ефективно моделі виявляють новіші типи зловмисних якими було розширено вибірку даних.

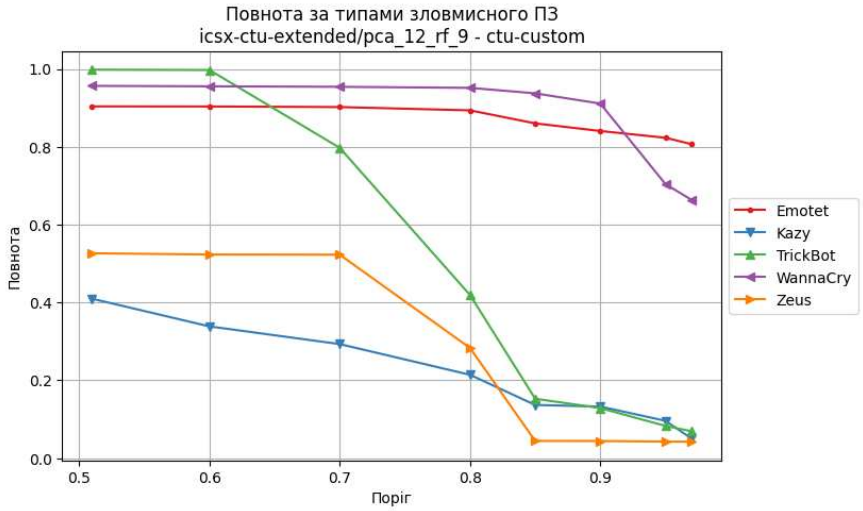

Рис. 2.17. Показники повноти класифікатора pca_12_rf_9 для різних типів зловмисного ПЗ (Тестувальна вибірка CTU-Custom)

Як видно з Рис. 2.17, отримана модель на основі випадкового лісу навчилася з високою точністю виявляти потоки Emotet, TrickBot та WannaCry при вказаних раніше порогових значеннях 0.6 та 0.7. Також з середньою точністю класифікатор здатен виявити Zeus та Kazy при цьому зберігаючи низький хибнопозитивний рівень.

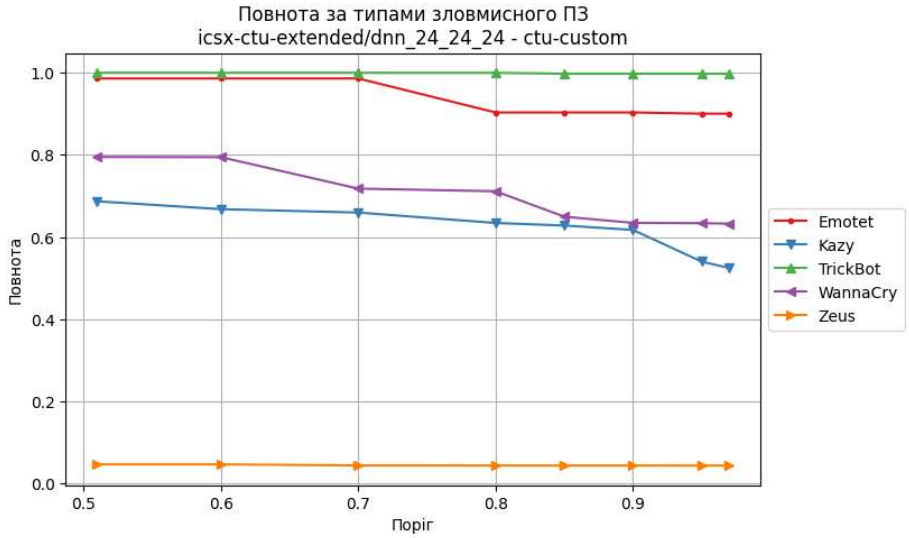

Рис. 2.18. Показники повноти класифікатора dnn_24_24_24 для різних типів зловмисного ПЗ (Тестувальна вибірка CTU-Custom)

Як і у випадку випадкового лісу, нейромережа dnn_24_24_24 навчилася з високою точністю виявляти новий трафік ботнетів Emotet та TrickBot. Відсоток виявлення WannaCry є дещо меншим, а трафік ботнету Zeus залишився повністю непоміченим. Водночас, якщо розглядати ефективність моделей для всієї множини типів шкідливого ПЗ, яке присутнє в тестовій вибірці ISCX-CTU-Extended, то стає зрозуміло, що нейромережа провела узагальнення не гірше за випадковий ліс.

На Рис. 2.19 показана залежність повноти виявлення трафіку 14 різних типів вірусного ПЗ алгоритмом випадкового лісу. При порогових значеннях в межах від 0.6 до 0.7 класифікатор з високою чутливістю (> 90%) здатен виявити трафік 7 типів шкідливих програм, а саме DonBot, TrickBot, Emotet, Rbot, Virut, Wannacry, а також IRC bot. З середньою чутливістю (> 30%, але < 60%) класифікатор може виявляти потоки породжені Kazy, Neris та Zeus. З низькою чутливістю (> 15%, але < 30%) модель здатна виявити Sogous, Weasle та Murlo. Трафік NSIS.ay залишається для випадкового лісу здебільшого непоміченим. Як загадувалось раніше, трафік позначений як «Virut» у наборі ISCX Botnet 2014 з високою ймовірністю це перейменований DonBot, і як видно з графіку, результати для нього абсолютно ідентичні. Таблиця з виміряними показниками повноти (чутливості) для моделі, на основі якої побудовано графік надається в **Додатку Д**.

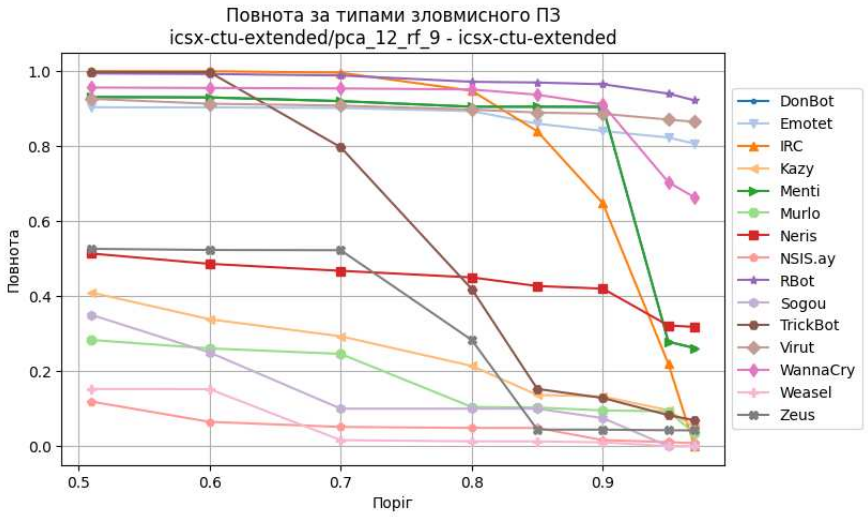

Рис. 2.19. Показники повноти класифікатора pca_12_rf_9
для різних типів зловмисного ПЗ (ISCX-CTU-Extended)

На Рис. 2.20 показана залежність повноти виявлення від порогу для кожного з типів трафіку для нейромережі dnn_24_24_24. Як видно, рівень повноти виявлення перевищує 60% для більшості типів потоків при використанні порогових, які не перевищують 0.9. Виключенням є Sogou та NSIS.ay, для яких модель показує невисокий відсоток виявлення, та Zeus, який навчена нейромережа зовсім не помічає. Також зауважимо, що при використанні значення порогу 0.97, яке забезпечує найменший хибнопозитивний рівень для цієї моделі, суттєво знижується повнота для підкласів Sogou, NSIS.ay, Murlo та Neris, що робить результати схожими з отриманими для випадкового лісу. Нейромережа з високою чутливістю (> 75%) виявляє потоки DonBot, Emotet, Virut, TrickBot, Rbot та IRC bot. З середньою чутливістю (> 30%, але < 60%) класифікатор здатен виявити потоки породжені Kazy, Murlo, Neris, WannaCry та Weasle. З низькою чутливістю (> 15%, але < 30%) модель може виявити NSIS.ay та Sogou. Таблиця з виміряними показниками повноти (чутливості) для моделі, на основі якої побудовано графік надається в **Додатку Д**.

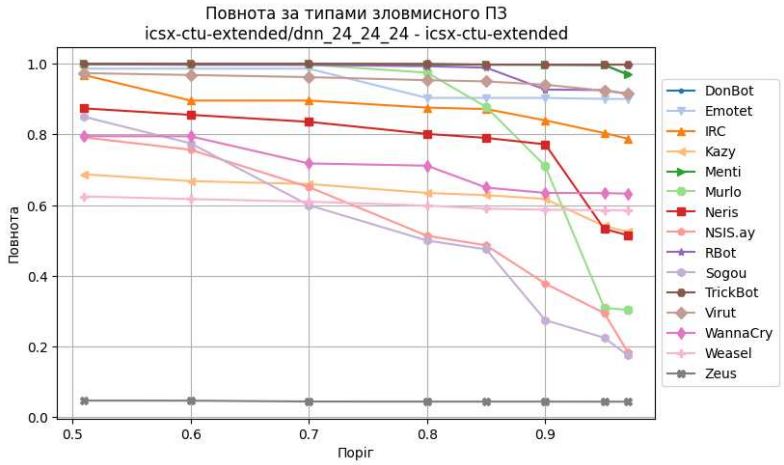

Рис. 2.20. Показники повноти класифікатора dnn_24_24_24 для різних типів зловмисного ПЗ (CTU-Extended)

В підсумку, створення розширеного набору даних дозволило значно підвищити ефективність та надійність виявлення слідів дій різноманітних шкідливих програм. При цьому, показано, що отримані моделі можуть бути ефективними не лише для ботів, а й для іншого роду вірусів, які взаємодіють з мережею, як-от WannaCry. Подальше розширення навчальних наборів даних та збільшення кількості ознак потоків може допомогти у створенні більш універсальних класифікаторів. Водночас, може мати сенс навчати окремі більш спеціалізовані моделі, для споріднених груп загроз, що може допомогти суттєво знизити відсоток хибнопозитивних результатів і підвищити довіру до системи в цілому.

Висновки до Розділу 2.

У розділі описано повний процес розв'язання задачі виявлення ботнетів засобами штучного інтелекту. Розглянута схема виявлення вторгнень не обмежена лише ботнетами і є гарним підґрунтям для майбутніх досліджень на тему виявлення будь-яких інших мережевих атак. Показники ефективності, отримані в ході навчання моделей класифікації трафіку, можуть бути використані як еталонні величини для порівняння у майбутніх дослідженнях, а удосконалення окремих етапів схеми має забезпечити подальші покращення ефективності виявлення в цілому.

Встановлено, що, метод об’єднання пакетів у семантично зв’язані групи дозволяє значно зменшити розмірність даних і дозволяє працювати навіть з тими пакетами, корисне навантаження яких зашифровано. При цьому, більш якісна реконструкція мережевих потоків та більш репрезентативна вибірка числових ознак, що їх характеризують дозволяються підвищити точність класифікації та допомагає виявляти та локалізувати сліди таких комплексних та варіативних загроз як ботнети серед усієї множини мережевих даних.

Окремо варто виділити ефективність методу організації у послідовності мережевих потоків та аналіз їх як часових рядів. Такий підхід є мало дослідженим, однак було експериментально показано, що він здатний підвищити точність класифікації мережевого трафіку, адже надає класифікатору історичний контекст подій. Для такого підходу має сенс використовувати рекурентні нейромережі, які дозволяють без додаткових зусиль оброблювати послідовності довільної довжини, що є корисним при аналізі мережевих даних у режимі онлайн.

При аналізі історичних мережевих даних добре показали себе нейромережі з темпоральними згортками, які беруть до уваги зміни характеристичних величин даних у часі. При цьому, особливу увагу треба приділяти методу групування потоків. Як було показано, розгляд груп потоків, які належать єдиному хосту може бути більш ефективний ніж виділення менших груп на основі пар IP адрес. Це пояснюються тим, що для хостів заражених ботом більш характерні спроби встановлення контакту одразу з багатьма віддаленими машинами за короткий проміжок часу, з розрахунком на те, що успішними будуть лише деякі з них. Відповідно послідовності потоків локалізованих в часі в межах одного зараженого хоста дозволяють класифікатору виявляти шаблони поведінки такого роду.

РОЗДІЛ 3. ОЦІНКА НАДІЙНОСТІ ТА ПІДВИЩЕННЯ СТІЙКОСТІ СИСТЕМ ВИЯВЛЕННЯ МЕРЕЖЕВИХ ЗАГРОЗ ДО ЗМАГАЛЬНИХ АТАК

В попередньому розділі отримано модель класифікації мережевого трафіку основна мета якої – виявляти сліди ботнетів на основі інформації отриманої з попередньо розміченого набору даних. Модель володіє доволі високими показниками ефективності, які не поступаються наведеним в споріднених роботах. Водночас, в ній є принаймні одна суттєва слабкість, на яку дослідники ШІ звертають увагу доволі часто, однак саме в сфері кібербезпеки, з якихось причин вона постійно пропускається. В цьому випадку мова йде про вразливість класифікаторів до «змагальних атак» [81]. Змагальна атака – це специфічний феномен можливості штучного створення екземплярів вхідних даних, які модель буде систематично відносити до конкретного класу за бажанням зловмисника, незалежно від того, якому класу вони належать в дійсності. Хоча ця риса і притаманна моделям ШІ у будь-яких областях використання, але в області виявлення мережевих атак вона стоїть особливо гостро. Під час проектування будь-якої системи виявлення вторгнень, апріорі очікується, що і вона сама може стати ціллю атаки. Відповідно від неї вимагається значно вища стійкість до різних методів ухилення від виявлення. Проблема нестачі даних для реалізації процедур навчання засобами штучного інтелекту, може призводити до явища перенавчання («overfitting») класифікатора [82], а також може погіршити його екстраполюючу здатність через властивість кускової лінійності нейронних мереж по відношенню до вхідних даних [83]. Обидва згаданих фактори роблять моделі ШІ ще більш вразливими до змагальних атак.

Слід зазначити, що найбільш розповсюджені оцінки якості класифікації такі як «точність» та «площа під ROC-кривою», які зазвичай наводяться в роботах на цю тему [84][3][45], самі по собі не відображають стійкість класифікаторів до змагальних атак і отримані моделі потребують додаткових гарантій якості. Саме тому, необхідні додаткові кроки для покращення властивостей узагальнення для моделей, та підкріплення достовірності отриманих оцінок. Одним із напрямків

можливого покращення є аугментація, або збільшення обсягу початкової вибірки даних, штучно-згенерованими, зокрема змагальними екземплярами, і подальше тестування обраної моделі на стійкість до змагальних атак. Очікується, що якщо отримати достатньо якісний аугментований набір даних, то оцінки ефективності моделі на ньому будуть більш надійним.

Метою цього розділу, є розробка методу аугментації початкової навчальної вибірки даних штучними, але правдоподібними даними, для моделей виявлення ботнет-трафіку з метою підвищення їх стійкості до змагальних атак. Крім того ставилось кілька критеріїв до розробленого методу. По-перше, необхідно врахувати семантику мережевих потоків та забезпечити її збереження при генерації нових даних. По-друге, трансформації проведені над початковими даними для створення нових мають бути реалістичними та підлягати можливості інтерпретації з точки зору зловмисника. По-третє, необхідно переконатися, що отримана аугментована вибірка даних дійсно дозволить підвищити стійкість навчених на ній моделей виявлення до змагальних атак, і при цьому не спричинить регресію їх базових показників якості.

В порівнянні з попередніми результатами досліджень у цій області [85][86], які переймають «як є» методи генерації змагальних прикладів із області обробки зображень, в цьому розділі метод створення змагальних прикладів адаптовано до низьковимірного та гетерогенного простору ознак мережевих потоків. Крім того, хоча в деяких роботах [87][88] при генерації нових значень і беруться до уваги статистичні рамки окремих ознак, однак в них не вводяться жодні міжознакові логічні обмеження, що також враховано в цьому розділі.

Слід зазначити, що хоча в рамках цієї роботи, у ролі об’єкту дослідження взято проблему виявлення нейронними мережами даних ботнетів, однак описаний метод також можна узагальнити для виявлення мережевих атак й інших типів.

3.1. Метод доповнення даних

Під час проведення процедур доповнення або аугментації даних, початкова вибірка розширюється новими штучно-створеними елементами, які в подальшому

можна застосовувати для навчання чи тестування моделі. Для цього до початкових екземплярів реальних даних застосовується ряд модифікацій. Таким чином, з однієї точки в початковому просторі даних отримується кілька нових. При цьому нові екземпляри мають бути правдоподібними, тобто достатньо схожими на оригінали з яких вони породжені (має зберігатися семантика даних).

Є добре відомі методи доповнення даних в більш досліджених областях таких як обробка зображень, аудіо та тексту. Однак для обробки мережевих потоків, застосування класичних методів є обмежене через особливості представлення та семантику даних такого роду. На відміну від області обробки зображень, генерація нових мережевих даних через накладання шумів обмеженої інтенсивності виявилось малоефективним. Наприклад, під час проведення класифікації потоків із навчальної вибірки з накладанням шуму, амплітуда якого не перевищує 10% від оригінального значення, погіршення в точності показника класифікації склало менше 0.01% для повнозв'язної нейромережі, яка описана в попередньому розділі. Відповідно модель вже є досить стійкою до випадкових збурень.

Значно вищий відсоток помилок класифікації можна отримати використовуючи не випадкові шуми, а спеціальні модифікації, отримані методами на основі вирахування «змагальних прикладів» – термін, який вперше запропонували К. Сегеді та ін. в роботі 2013 року «Інтригуючі властивості нейронних мереж» [81].

3.2. Генерація змагальних прикладів

В цьому розділі описана адаптація «швидкого методу знаку градієнту» (FGSM) [6], для перетворення мережевих потоків представлених векторами ознак. Ціль перетворення: із початково вірно-класифікованих нейронною мережею екземплярів зробити семантично схожі, але такі, що модель би класифікувала їх невірно. В подальшому, доповнивши навчальну множину екземплярами такої природи, можна отримати більш репрезентативну вибірку. Очікується, що навчена на ній модель буде більш стійкою до схожих збурень у вхідних даних.

Для мережевих потоків алгоритм генерації буде наступний: обираються потоки, які належать заздалегідь відомому класу (наприклад, «злоякісні»). Нехай потік має N ознак та позначається як $x = (x_1, ..., x_N)$. Відносно кожного *x* знаходиться градієнт ∇_x функції втрат $J(\theta, x, y)$, яка використовувалась при навчанні моделі**.** Функція втрат *J* – це бінарна перехресна ентропія, обчислена для даного *x* та мітки його фундаментальної істини *y* (мітка “1” позначає потік як «злоякісний», “0” – як «доброякісний»). θ – це ваги та зсуви моделі, вивчені нею під час навчання.

Знаходження градієнту $\nabla_x J(\theta, x, y)$ відбувається через обчислення частинних похідних J по кожній з компонент (ознак) *x*.

$$\nabla_x J(\theta, x, y) = (\frac{\partial J(\theta, x, y)}{\partial x_1}, ..., \frac{\partial J(\theta, x, y)}{\partial x_N})$$

Частинні похідні нелінійної функції, якою являється *J*, обчислюються стандартним алгоритмом зворотного поширення [89] (окремий випадок автоматичної диференціації, який реалізується шляхом зворотної акумуляції в обчислювальному графі [90]):

1. Дано елемент вибірки *x***.** Спершу, проводиться прямий прохід по графу, з обчисленням результату кожної операції для даного *x*.
2. Далі, починаючи з вихідного шару і рухаючись у зворотному напрямку шар за шаром обчислюються частинні похідні функції втрат по параметрам та операціям представленими даним шаром. Для обчислення використовується класичне ланцюгове правило повної диференціації функції.
3. Нарешті, коли процедура зворотного розповсюдження доходить до першого “вхідного” шару, частинні похідні $\frac{\partial J(\theta, x, y)}{\partial x_i}$ обчислюються аналогічно похідним по всім іншим параметрах даного обчислювального графу.

Отриманий градієнт вказує напрямок найбільш стрімкого зростання значення функції втрат *J* у конкретній точці простору ознак. Чим більша величина *i*-тої

компоненти градієнта, тим більше модель класифікації є чутливою до *локальної* зміни *i*-тої ознаки (x_i).

Так як трансформації у просторі ознак не завжди мають зворотне відображення у простір подій оригінальної задачі, то може виникати необхідність накласти певні обмеження на припустимі зміни в ознаках екземпляра. Позначимо як $\overrightarrow{\nabla_x}$ вектор попередньо обчислених компонентів градієнту, де певні обрані ознаки залишаються незмінними, а всі інші компоненти виставляються в 0 (іншими словами ігноруються). Тоді для отримання потенційного змагального прикладу, ознаки оригінального екземпляру необхідно змасштабувати в напрямку градієнту, таким чином збільшуючи значення функції втрат. Отримання нового екземпляру x' з оригінального x показано у формулі (3.1).

$$x' = x \cdot (1 + \varepsilon \cdot sign\left(\overrightarrow{\nabla_x} J(\theta, x, y)\right)), \varepsilon \in (0,1) \tag{3.1}$$

Слід зазначити, що в класичному FGSM [83] використовується наступна формула:

$$x' = x + \varepsilon \cdot sign\left(\overrightarrow{\nabla_x} J(\theta, x, y)\right), \varepsilon \in (0,1), x_i \in [0,1], x'_i \in [0,1] \tag{3.2}$$

де x_i та x'_i – *i*-та ознака початкового й згенерованого прикладу відповідно.

Використання формули (3.2) передбачає, що ознаки x_i попередньо змасштабовано (як правило відносно мінімального та максимального значенням, які зустрічались у навчальному наборі), і її значення тепер лежить в межах від [0..1]. При цьому для кожного оброблюваного прикладу, значення відповідної ознаки збурюється на певну абсолютну величину. Якщо нове значення ознаки x'_i виходить за зазначені межі, то воно «обрізається», тобто встановлюється в 0 чи 1.

У ході виконання цієї роботи, було помічено, що спроба генерації нових мережевих потоків через формулу (3.2), призводить до 2 проблем:

1. Якщо діапазон масштабування ознаки, отриманий з навчального набору, є досить великим, то навіть при відносно невеликих ε, згенерований потік легко виходить за рамки семантичної подібності. Наприклад: припустимо значення «Розмір корисного навантаження» для конкретного потоку $x_i = 1010$

байт, але діапазон значень для цієї ознаку в межах якого проводилось масштабування, скажім – [0..10000], тоді при ε = 0.1 і знаку градієнту "-" значення ознаки отримане формулою (3.2) матиме вигляд $x_i' = \frac{1010}{10000} + 0.1 \cdot (-1) = 0.101 - 0.1 = 0.001$, що відповідає лише 10 байтам. Таке різке зменшення розміру корисного навантаження (в більше ніж 100 разів) порушує вимогу подібності згенерованого потоку до оригінального.

2. Крім того, значно зростає кількість випадків коли відбувається обрізка значення ознаки. Фактично при додаванні абсолютної величини ε, будь-яке значення ознаки, яке лежить в межах [0..ε] і [ε..1], буде обрізане при знаках градієнту "-" та "+" відповідно. При обробці мережевих потоків, представлених лише кількома десятками ознак, втрата інформації по одній з них є досить критичною.

Заміна додавання однакового абсолютного значення до ознаки для всіх прикладів (Формула 3.2), на відносне масштабування її величини для кожного окремого прикладу (Формула 3.1), повністю вирішує проблему 1 і також запобігає обрізкам значень по нижні границі, при ε < 1 і також зменшує частоту обрізок по верхній границі, що частково вирішує проблему 2.

3.3. Особливості простору ознак мережевих потоків

Обробка мережевих даних у формі статистик мережевого потоку має певну специфіку, без врахування якої, спроби генерації нових правдоподібних потоків будуть невдалими.

По-перше, простір ознак мережевих потоків є низьковимірним, на що вже зверталась увага в попередньому розділі. Тоді як, зображення складається з мільйонів пікселів, потік, як правило, представляється лише кількома десятками значень. В експериментальній частині цього розділу, потоки представлялися 50 ознаками (див. **Додаток В**).

По-друге, більшість ознак не мають чітко встановленої верхньої границі. На відміну від зображень, де кожен піксель закодований фіксованою кількістю біт і

кольори пікселів достатньо рівномірно розподілені по всьому спектру, значення характеристик потоку можуть необмежено зростати і значно варіюватися, адже їх кодування ніяк не регламентує їх верхню границю. Прикладом є ознака «кількість пакетів у потоці» та інші споріднені їй. Навіть для тих характеристик для яких існують певні технічні обмеження верхня межа варіюється від мережі до мережі, і її можна оцінити хіба що статистично. Наприклад, максимальне значення «тривалість потоку» може по-різному обмежуватись в залежності від конфігурації мережевого шлюзу чи веб-сервера.

Третьою особливістю, є те що, ознаки потоків зазвичай дуже гетерогенні. Вони значно відрізняються одна від одної своїми верхніми та нижніми границями, а також дисперсією. Наприклад, значення «кількість TCP ACK прапорців», як правило, суттєво менше за «максимальний розмір корисного навантаження» в одному й тому самому потоці. Це також є значною відмінністю від області обробки зображень, де ознаки є фактично гомогенними.

Крім того, не всі ознаки є доступними для довільної модифікації без спричинення порушення семантики потоку. Такі поля як «кількість прапорців SYN/ACK/FIN…» випливають з правил встановлених TCP-протоколом.

Також, через специфіку підрахунку та представлення ознак потоку (більшість із них – це статистики), деякі ознаки корелюють між собою та динамічно обмежують одна одну. Наприклад, коли величини описують мінімальне, максимально та сумарне значення певного параметру, то принаймні має виконуватись нерівність «мін.» ≤ «макс.» ≤ «сумарна».

З всього зазначеного стає зрозумілим, що генерація довільних варіацій мережевих потоків (не лише змагальних), як і взагалі будь-яка контрольована генерація мережевого трафіку, є нетривіальною задачею і потребує окремого дослідження.

При цьому, для генерації саме змагальних прикладів, до уваги варто взяти й те, що зловмисник має можливість змінювати значення ознак потоку лише опосередковано – величини ознак потоків обчислюються системою аналізу мережевого трафіку, до якої зовні немає прямого доступу, що ускладнює завдання

створення успішних атак проти класифікатора. Водночас завдання підвищення стійкості моделі до таких атак, може бути зведеним до створення надмножини, яка включає і певний відсоток реально недосяжних екземплярів, доти, поки ця множина максимально широко покриває всі реально можливі атаки. Саме такий метод був використаний в практичній частині роботи, яка буде описана в наступних підрозділах. Для мінімізації ризику порушення семантики потоку при модифікації та виходу за межі досяжності, збурення були здійснені лише над обмеженими групами ознак, в яких вони би мали конкретну семантичну інтерпретацію. При цьому було накладено ряд базових обмежень при порушенні яких, згенерований екземпляр відкидався.

3.4. Обробка вибірки та обрання груп вразливих ознак мережевих потоків

Як і в попередньому розділі, для отримання практичних результатів використовувався набір даних UNB CIC Botnet 2014 [19]. Самі потоки та їх ознаки із набору пакетів виділені модифікованою версією інструменту CICFlowMeter [21]. Крім стандартного ідентифікуючого кортежу, який складається з IP-адрес, портів, протоколу та часової мітки для кожного потоку було обрано та обчислено 50 ознак (див. Додаток В).

Зауважимо, що ботнети поділяються на 3 основних типи, за способом комунікації між різними вузлами їх робочої ієрархії [91]: IRC (Internet Relay Chat), HTTP (HyperText Transfer Protocol) та P2P (peer-to-peer). Зважаючи на те, що у навчальному наборі даних були повністю відсутні приклади P2P (peer-to-peer) трафіку, то в ході оцінки якості моделі на тестувальному наборі потоки P2P-ботнетів також виключались, для отримання більш чітких результатів.

Кількість потоків за класами в навчальному та тестувальному наборі наведено у Таблиці 3.1. Для навчальних даних, приклади отримані при різних значеннях є об’єднувались в спільну множину, якою доповнювався початковий навчальний набір, що також показано у таблиці. Для тестувальних даних показники класифікації підраховувались для кожного значення є індивідуально, тому в таблиці 1 їх кількість не наводиться. Під тестувальними даними мається на увазі

вибірка, яку модель не бачила при навчанні (ні самі елементи вибірки, ні похідні від них).

Таблиця 3.1

Дані	**Навчальні**		**Тестувальні (нові)**	
Доповнення	**До**	**Після**	**До**	**Після**
Нормальні	306056	1191153	126232	(без змін)
Шкідливі (ботнети)	109305	396711	87960	(без змін)
% шкідливих	35.7	33.3	69.7	(без змін)

Для отримання більш правдоподібних прикладів, використання збурень та подальша оцінка стійкості моделі проводилась лише за двома семантично виокремленими групами ознак:

- Група ознак 1 (9 ознак): Тривалість потоку, Мінімальний, максимальний міжпакетний інтервал в прямому та зворотному напрямках, Середньоквадратичне відхилення міжпакетних інтервалів у прямому та зворотному напрямках, Сума міжпакетних інтервалів в обох напрямках.
- Група ознак 2 (6 ознаки): Мінімальне, максимальне, та середньоквадратичне відхилення довжини пакетів в прямому та зворотному напрямках.

Збурення в ознаках «групи 1» можна інтерпретувати як маніпуляція з часовими параметрами потоку: віддалення чи зближення першого та остатнього пакету визначає значення «тривалості потоку», а додаткова затримка (чи навпаки її скорочення) між відправкою двох послідовних пакетів впливає на статистику міжпакетних інтервалів.

Зміни в ознаках «групи 2» відображають варіювання розмірів пакетів. Наприклад, додаючи надлишкові байтів в кінець корисного навантаження можна збільшити статистику «максимальної довжини пакету», а організувавши дані більш

компактно (чи взагалі передавши пустий пакет) – зменшити значення «мінімальної довжини».

При цьому, хоча й при внесенні довільних змін в ознаки кожної з груп не можна гарантувати повну семантичну тотожність нового отриманого потоку. Однак, як буде показано експериментально, генерація розширення даних з додаванням лише базових логічних обмежень є достатньою для якісного покращення стійкості моделі класифікації.

Для кожної з груп ознак були введені наступні обмеження (повний список назв ознак та їх опис наведено у Додатку В).

Група 1:

- *"Fwd IAT Min" ≤ "Fwd IAT Max"*
- *"Bwd IAT Min" ≤ "Bwd IAT Max"*

Група 2:

- *"Fwd Packet Length Min" ≤ "Fwd Packet Length Max"*
- *"Bwd Packet Length Min" ≤ "Bwd Packet Length Max"*
- *"Fwd Packet Length Max" ≤ "Total Length of Fwd Packet"*

  "Bwd Packet Length Max" ≤ "Total Length of Bwd Packet"

3.5. Базова модель та оцінка показників ефективності та стійкості

Так як навчання моделі і пошук змагальних прикладів є ресурсоємними задачами, а проведені експерименти по доповненню даних потребували багаторазових повторень, то однією з основних вимог до моделі була обчислювальна простота. Крім того, використання відносно простої архітектури нейронної мережі, дозволяє зменшити ризик отримання вдалих, але невідтворюваних результатів через випадкову успішну ініціалізацію початкових вагових коефіцієнтів класифікатора при навчанні.

В якості моделі класифікації було застосовано глибоку нейронну мережу з 3-ма прихованими повнозв’язними шарами по 24 нейрони, та одним вихідним. В

прихованих шарах у ролі функції активації використовується ReLU (зрізаний лінійний вузол), адже знаходження градієнту для неї тривіальне (частинна похідна завжди 0 або 1). В якості функції активації вихідного нейрону була використана сигмоїда, задача якої – звуження діапазону вихідних значень в межах між 0 та 1.

При навчанні функцією втрат слугувала бінарна перехресна ентропія. Для оптимізації параметрів моделі використовувався алгоритм Adam (адаптивна оцінка моменту).

Для більш достовірного порівняння отриманих моделей, класифікатор повторно навчався 10 разів до та після доповнення, після чого для кожного випадку було обрано модель з найвищими показником AUROC (площа під кривою робочої характеристики приймача) на тестувальних даних.

Після отримання базової моделі з достатньо високими показниками ефективності на тестовому наборі даних (AUROC = 0.932, точність = 0.88, при порозі впевненості = 0.51), було проведено аналіз її вразливості до змагальних атак. Для цього, для кожної точки в наборі тестувальних даних було підраховано вектори градієнтів моделі у просторі ознак. Далі, методом, описаним у минулих підрозділах, було отримано змагальні приклади для кожної точки базового набору через масштабування значень у напрямку градієнту, окремо для кожного з зазначених наборів ознак.

Для кожного значення ε було підраховано відсоток хибних класифікацій екземплярів отриманих описаним способом з оригінальних мережевих потоків, які належали класу «ботнетів». Результати наведені в Таблиці 3.2. Як було зауважено раніше, на етапі коли для конкретного ε трансформація стає недійсною (порушуються встановлені обмеження) для певного потоку, він та його оригінал виключається з експерименту. Відповідно, для більш коректної інтерпретації результатів, також необхідно слідкувати за загальною кількістю потоків, які розглядаються на даному етапі.

Нижче наведено псевдокод який показує порядок кроків генерації змагальних прикладів та оцінки вразливості до них моделі.

INPUT

F : ознаки

FG : група ознак, над якою проводяться збурення

C : обмеження

X : вектори ознак мережевих потоків

Y : мітки мережевих потоків (значення 0 – звичайний, 1 – злоякісний/ботнет)

E : величина збурення ({ 0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 })

t : поріг впевненості, при досягненні якого, потік вважається «злоякісним»

BEGIN

$f \leftarrow \{\ 1\ \ if\ \ f_i \in FG\ \ else\ \ 0\ \}_1^{|F|}$ // вектор-селектор групи ознак

$X_{\min}, X_{\max} \leftarrow \left\{\min_{\forall x \in X} x_i\right\}_1^{|F|}, \left\{\max_{\forall x \in X} x_i\right\}_1^{|F|}$ // визначити границі ознак

$\theta \leftarrow train\left(\theta, J\left(\theta, scale\left(X, X_{\min}, X_{\max}\right), Y\right)\right)$ // навчання моделі

// θ – параметри нейромережі, J – функція втрат (бінарна перехресна ентропія)

$X_{adversarial} \leftarrow \{\ \}$ // порожня ініціалізація

foreach ε **in** E:

foreach x^n **in** X:

$\overrightarrow{\nabla_{x^n}} \leftarrow f \cdot \nabla_{x^n} J\left(\theta, x^n, y^n\right)$ // підрахувати градієнт

$x^n_{adversarial} \leftarrow x^n \cdot \left(1 + \varepsilon \cdot sign\left(\overrightarrow{\nabla_{x^n}}\right)\right)$ // внести збурення

end

$X_{adversarial} \leftarrow \left\{x^n_{adversarial} \mid satisfies\left(x^n_{adversarial}, C\right)\right\}$ // застосувати обмеження

$X_{adv\_malicious} \leftarrow \left\{x^n_{adv} \mid y^n = 1, y^n \in Y, x^n_{adv} \in X_{adversarial}\right\}$ // виділити «злоякісні»

$P \leftarrow predict\left(\theta, scale\left(X_{adv\_malicious}, X_{\min}, X_{\max}\right)\right)$ // зробити передбачення

// обрати приклади, які модель помилково вважає за «звичайні»

$X_{adv\_misclassified} \leftarrow \left\{x^n_{adv} \mid p^n < t, p^n \in P, x^n_{adv} \in X_{adversarial}\right\}$

$missclassification\_rate[\varepsilon] \leftarrow \frac{\left|X_{adv\_misclassified}\right|}{\left|X_{adv\_malicious}\right|}$ // знайти частку помилок

end

END

Слід зазначити, що для більшої наглядності в таблиці 3.2 наведено не абсолютну величину відсотку хибних класифікацій, а її приріст відносно початкового набору даних без збурень.

Таблиця 3.2

	Група ознак 1		***Група ознак 2***	
ε	**Приріст % помилок класифікації**	**Кількість розглянутих екземплярів**	**Приріст % помилок класифікації**	**Кількість розглянутих екземплярів**
0	0	70024	0	70024
0.01	0.01	69663	0.07	67431
0.05	0.07	69658	0.80	67388
0.1	0.34	69654	2.6	67040
0.2	0.61	69645	4.52	66535
0.3	0.83	69635	9.12	66521
0.4	0.99	69450	11.35	66506
0.5	1.08	69286	12.44	66491
0.6	1.40	69263	13.00	66464
0.7	1.48	69240	13.21	66457

Як видно з Таблиці 3.2, модель виявилась маловразливою (хоча й не повністю стійкою) до змагальних атак описаним методом шляхом варіювання міжпакетних

інтервалів та тривалості потоку (група ознак 1). Водночас, маніпуляції статистиками розмірів пакетів (група ознак 2) вплинули на модель значно сильніше.

3.6. Побудова доповненого навчального набору даних

Після виявлення вразливості базової моделі до змагальних атак постає задача побудови нового навчального набору, доповненого змагальними прикладами і подальше навчання на ньому нової моделі класифікації. При цьому можна виділити 2-а основні критерії, які мають виконуватись для нової моделі:

1. Нова модель класифікації має бути більш стійка до змагальних атак (принаймні до класу атак представленого у навчальному наборі).
2. Загальні показники якості отриманої моделі, такі як точність та відсоток хибно-позитивних результатів мають бути не гірші ніж у базової.

Слід зауважити, що в ході виконання роботи було зроблено кілька спроб доповнення навчального набору змагальними прикладами створеними наведеним вище методом для обох вказаних груп ознак при цьому використовуючи різні значення ε. Доповнення навчальної вибірки прикладами отриманими для ознак групи 1 не принесло значних результатів: як видно з Таблиці 3.2, базова модель є слабко чутливою до варіацій в міжпакетних інтервалах. Однак, при доповненні екземплярами згенерованими для ознак групи 2 (статистики розмірів пакетів), стійкість до змагальних атак значно покращилась і модель задовільнила обидва із зазначених критеріїв.

Найкращий результат принесло доповнення навчального набору екземплярами отриманими з початкових масштабуванням в напрямку градієнту ознак 2 групи для ε = [0.05, 0.1, 0.2]. Утворена навчальна вибірка описується в таблиці 1, представленій раніше.

Результуюча модель, яка навчалась на доповненому наборі, показала себе значно більш стійкою до варіювання розмірів пакетів описаним методом, у порівнянні з оригінальною (див. Рис. 3.2). Додатковим стороннім ефектом стало й

те, що хоча доповнення і не включало в себе збурення ознак групи 1, однак на діапазоні ε від 0.1 до 0.3 вразливість до змагальних атак такого роду над цією групою також дещо покращилась (див. Рис. 3.3). Таким чином, навчена модель задовольняє першому критерію.

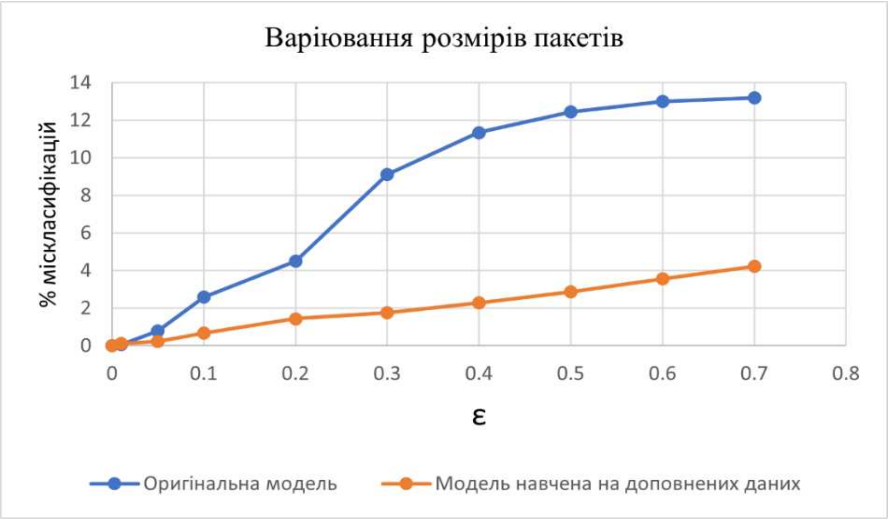

Рис. 3.2. Зростання частки помилок класифікації зі зростанням ε при варіюванні розмірів корисного навантаження пакетів.

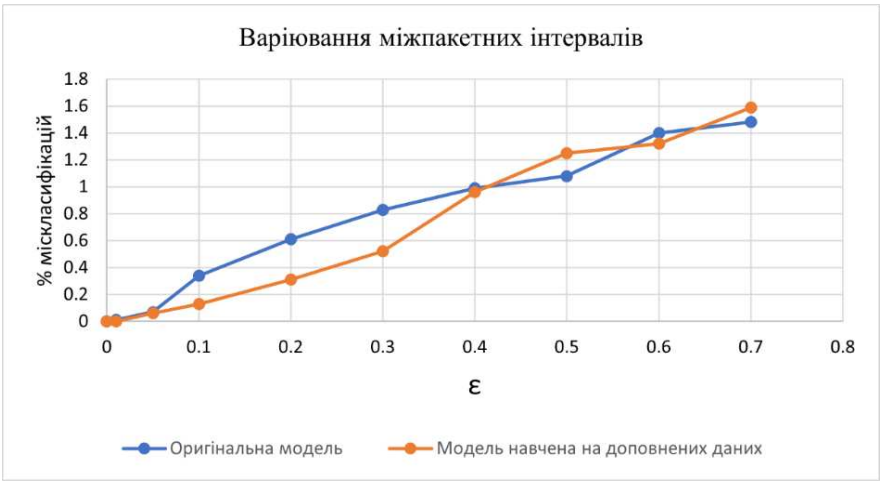

Рис. 3.3. Зростання частки помилок класифікації зі зростанням ε при варіюванні міжпакетних інтервалів пакетів.

Попри виконання критерію підвищеної стійкості до змагальних атак, необхідно пересвідчитись, що таке штучне доповнення навчального набору не призводить до погіршення якості класифікатора по іншим параметрам

ефективності. В Таблиці 3.3 наводиться порівняльна характеристика виміряних показників базової моделі та моделі навченої на доповнених даних. Для обох моделей обчислено такі показники ефективності:

1. AUROC (площа під ROC-кривою):

$$AUROC = \int_0^1 TPR(t) \cdot FPR'(t)\,dt \quad , \tag{3.5}$$

де *TPR(t)* – функція частки істинно-позитивних результатів, а *FPR(t)* – функція частки хибно-позитивних результатів, в залежності від порогу впевненості *t*.

2. Точність (Формула 3.3):

$$Accuracy = \frac{TP + TN}{N} \quad , \tag{3.3}$$

де *TP* – кількість істинно-позитивних результатів, *TN* – кількість істинно-негативних результатів, підрахованих для фіксованого порогу, а *N* – загальна кількість прикладів.

3. Частка помилок I та II роду (Формули 3.4 та 3.5 відповідно):

$$FPR = \frac{FP}{N} \quad , \tag{3.4}$$

$$FNR = \frac{FN}{N} \quad , \tag{3.5}$$

де *FP* – кількість хибно-позитивних результатів, *FN* – кількість хибно-негативних результатів, при обраному порозі впевненості.

Слід зауважити, що під «порогом впевненості» мається на увазі величина $t \in [0..1]$, з якою порівнюється значення впевненості передбачуване моделлю для кожного елементу вхідної вибірки. Гіпотеза класифікації приймається для даного елементу, якщо значення спрогнозоване моделлю перевищує поріг *t*, а інакше відкидається (Формула 3.6). В даній задачі, гіпотеза класифікації – це належність мережевого потоку до трафіку ботнетів.

$$\hat{y}_i = \begin{cases} +, & if\ \ p(x_i) > t \\ -, & else \end{cases} \quad , \tag{3.6}$$

де $p(x_i)$ – передбачувана нейронною мережею впевненість для екземпляра x_i.

Кожному елементу x_i класифікатор ставить у відповідність мітку $\hat{y}_i$. Після чого, вона порівнюється зі значенням апріорі відомим з початкової розмітки набору даних, і таким чином робиться висновок про істинність чи хибність зробленого моделлю припущення.

Таблиця 3.3

Дані	**Навчальні**		**Тестувальні (нові)**	
Модель	**До**	**Після**	**До**	**Після**
AUROC	0.976	0.981	0.932	0.932
Точність (поріг ймовірності = 0.51)	0.935	0.937	0.880	0.891
Частка помилок I роду (поріг ймовірності = 0.51)	0.032	0.041	0.061	0.053
Частка помилок II роду (поріг ймовірності = 0.51)	0.157	0.122	0.204	0.190

Висновки до розділу 3

В ході роботи над цим розділом було описано та реалізовано удосконалення методу генерації штучних прикладів мережевих потоків шляхом адаптації швидкого методу знаку градієнту з метою оцінки вразливості систем виявлення вторгнень до змагальних атак.

Особливістю використаного методу є врахування семантики ознак потоків: виділення логічних груп статистик, і побудова змагальних даних окремо супроти кожної групи, що робить результати перевірки вразливості моделі та підвищення її стійкості значно краще піддаються інтерпретації. На відміну від інших робіт, де ціллю є аналіз і покращення універсальної стійкості моделі до змагальних атак на всьому просторі ознак (не залежно від того як і чи взагалі можливо реалізувати таку атаку на практиці), запропонований метод дозволяє робити більш конкретні судження, як-от про вразливість/стійкість до «варіації міжпакетних інтервалів» чи

«маніпуляції з розміром корисного навантаження пакетів». Можливість такої семантичної інтерпретації має практичне значення в області кібербезпеки і у подальшому може бути використана для формулювання конкретних вимог і гарантій до системи виявлення загроз.

Також експериментально продемонстровано ефективність застосування методу доповнення вибірки навчальних даних змагальними прикладами мережевих потоків. Це забезпечило помітне підвищення стійкості системи до можливостей навмисного ухилення від виявлення при цьому не погіршуючи практичні узагальнюючі властивості класифікатора.

Описаний підхід є хорошою основою до подальших досліджень в цій області, зокрема в напрямках більш якісної генерації змагальних прикладів не лише однокроковими, а й ітеративними методами, а також отримання більш стійких моделей методами змагального навчання.

РОЗДІЛ 4. ФОРМАЛЬНА ВЕРИФІКАЦІЯ ВЛАСТИВОСТЕЙ СТІЙКОСТІ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ ЗАГРОЗ

В роботах, де зазвичай розглядається проблема підвищення стійкості моделей класифікації до змагальних атак, зазвичай використовуються класичні емпіричні методи статистичної оцінки стійкості моделей на штучно згенерованих даних, на кшталт техніки генерації змагальних прикладів, використаної у попередньому розділі. Недоліком такого підходу, є те, що в результаті не надається жодних абсолютних гарантій про надійність моделі, які б розповсюджувались за межі тестової вибірки. Відповідно, поведінка класифікаторів ШІ, в особливості заснованих на нейромережах, залишається досить непередбачуваною. Для класичних програмних систем вже є ряд добре сформованих стандартів для постановки та формулювання вимог якості [92]. Для систем на основі ШІ, однак, наразі не існує загальновизнаних стандартів верифікації та оцінювання їхньої якості, і наявні підходи переважно обмежуються статистичними гарантіями, здобутими на валідаційних вибірках. Це є вагомим аргументом проти використання ШІ в області кібербезпеки, де навіть незначна помилка може призвести до серйозних наслідків. Непередбачуваність моделей ШІ ставить під загрозу їхню ефективність у виявленні нових варіацій навіть відомих загроз, які не були враховані під час навчання. Це може призвести до пропущення критичних атак або, навпаки, до помилкових спрацьовувань, що знижує ефективність і довіру до систем кібербезпеки на основі ШІ. Важливо враховувати ці ризики при впровадженні таких технологій і розробляти додаткові методи контролю та валідації для підвищення їхньої надійності, а також створювати моделі та стандарти їх якості [93].

Не так давно, почав набувати розвитку підхід «формальної верифікації» нейронних мереж, при якому вимоги задаються формально у формі обмежень, що накладаються на входи та виходи моделі. Після цього відбувається автоматична математична перевірка виконання обмежень моделлю на всіх можливих вхідних даних. Такий підхід принципово відрізняється від перевірки результатів для

окремих штучно-згенерованих прикладів, адже отримані результати надійності розповсюджуються на всю формально визначену область в просторі вхідних даних.

В цьому розділі розроблено метод формальної верифікації властивостей нейронних мереж для перевірки їх стійкості в семантично визначених регіонах у просторі ознак за допомогою SMT-розв'язувача. І хоча в цій роботі предметною областю є системи виявлення трафіку зловмисного ПЗ, однак, реалізований метод верифікації можна застосовувати для довільних нейромереж прямого поширення з більше ніж одним прихованим шаром та кусково-лінійною функцією активації прихованих нейронів незалежно від їх цільової задачі.

Також формалізовано критерій стійкості нейромереж-класифікаторів мережевих потоків до збурень у вхідних даних. У експериментальній частині розділу продемонстровано застосування розробленого методу для верифікації локального задовільнення класифікатором сформованого критерію. Це дозволило з більшою достовірністю оцінити стійкість створеної системи виявлення загроз до змагальних впливів. Сам алгоритм оптимізовано, з метою уникнення комбінаторного вибуху складності при аналізі розв’язувачем графів нейромереж з більше ніж 2 прихованими шарами.

4.1. Методи верифікації властивостей моделей ШІ

Для того щоб краще зрозуміти переваги використання підходу до верифікації нейромережі, який описано в цьому розділі, варто згадати переваги та недоліки існуючих більш усталених методів перевірки властивостей ШІ.

Найбільш поширеним методом є емпірично-статистична оцінка характеристик моделі. Перевірка даних властивостей проводиться шляхом отримання результатів роботи моделі над фіксованою множиною точок з вхідного простору. Після чого проводиться порівняння отриманих результатів із заздалегідь відомими та підраховуються статистичні оцінки їх схожості. Класичними показниками при такій оцінці є, точність, влучність, частка позитивно-негативних результатів, відсоток помилок над множиною змагальних прикладів і т.д. Як і

випливає з назви, такі властивості не надають жодних гарантій якості моделі, а лише показують статистику роботи моделі на доступній вибірці.

Іншим методом верифікації є перевірка виконання глобальних обмежень (constraints satisfaction). Такі властивості визначаються у формі системи жорстких обмежень накладених на простір вхідних та вихідних даних моделі. При цьому верифікація проводиться не на фіксованому наборі вхідних даних, а на повному теоретично можливому підпросторі з простору входів. Прикладом такої властивості у випадку багато-класової класифікації може бути, така умова, що якщо модель класифікації вже віднесла певний об’єкт до якогось класу, то показник впевненості для всіх інших класів має буде нижчим за певний поріг.

Метод який розглядається в цьому розділі – це верифікація локальної надійності (local robustness). Коли модель є локально надійною (стійкою), очікується, що якщо для певного елементу з області входу модель продукує коректний результат, то при внесенні відносно незначних пертурбацій у значення ознак даного вхідного елементу, результат змінюється лише в очікуваних межах. Наприклад, у випадку задачі класифікації, обидві – оригінальна та змінена точки мають бути віднесені моделлю до однакового класу. Ця властивість часто розглядається в контексті стійкості моделі до змагальних атак, однак може бути розширена до перевірки властивостей узагальнення моделі щодо довільних збурень вхідних даних в допустимих межах. В роботі [94] наводяться визначення 4 типів надійності в порядку посилення їх умов та обмежень:

1. **Надійність класифікації**, що передбачає збереження класу для всіх елементів з околу точки з простору вхідних даних.
2. **Стандартна надійність** полягає в тому, що для околу вхідної точки, вихідне значення моделі варіюватиметься також в певному околі.
3. **Надійність Ліпшица** визначається тим, що при внесенні змін у значення ознак вхідного елементу, вихід змінюється пропорційно.
4. **Жорстка надійність** класифікації передбачує, що в околі вхідної точки, значення на виході класифікатора для коректного класу буде вище певного порогу.

За реалізацією методи верифікації можна розділити на п’ять наступних класів.

Емпіричне тестування [95]. Найпростіший клас методів, суть якого - перевірка результатів виходів моделі, на множині заздалегідь відомих вхідних елементів. Тестова множина може бути отримана разом з навчальним набором даних, або згенерована через тривіальні методи розширення даних (як-от деформації, накладання випадкових шумів тощо). До цього класу належать й інші класичні методи як фазинг [96], чи тестування на основі покриття [97].

Евристичний пошук контрприкладів [98]. В даному класі різними евристичними методами проводиться пошук правдоподібних вхідних елементів, для яких модель порушує властивості, що перевіряються. Одним з класичних способів є перетворення початкових вірно-класифікованих елементів вхідної вибірки на “змагальні приклади” за допомогою градієнтних методів (наприклад, “методом швидкого градієнту” (FGSM) [83]). Цей клас методів однак, не обґрунтовує стійкість моделі, а лише може виявити її вразливість.

Рішення задачі оптимізації [99]. Якщо модель можна звести до системи лінійних рівнянь та нерівностей, то для верифікації можна застосувати методи лінійного чи змішаного цілочисельного програмування. Слід зауважити однак, що через наявність нелінійних функцій активації нейронів, як правило неможливо застосувати цей клас методів до нейронних мереж “як-є”. Відповідно, їх зазвичай використовують у комбінації з іншими підходами такими як застосування SMT (Satisfiability Modulo Theories) розв’язувача [100][101].

Формування абстрактної інтерпретації моделі [102]. Цей метод зазвичай доповнює інші. Його суть в перебудові структури нейронної мережі, що розглядається на семантично схожу, але більш просту [103]. Результуюча модель є під- або над- апроксимацією оригінальної. Мета такої операції – спрощення обчислювальної складності при подальшому доведенні виконання (чи порушення) накладених обмежень іншими методами. Очікується, що з виявлення порушення обмеження для під-апроксимованої моделі, випливає порушення даного обмеження і для оригінальної. Водночас доведення стійкості

під-апроксимованої моделі, не гарантує стійкість оригінальної. У випадку над-апроксимації все з точністю навпаки.

Доведення за допомогою SMT – розв'язувача [104]. В даному методі обчислювальний граф моделі представляється у вигляді SMT формули. Далі на входи та виходи моделі накладаються певні обмеження, після чого SMT-розв'язувач доводить або спростовує їх виконуваність. Зазвичай цей метод вимагає попередньої апроксимації всіх нелінійних функцій активації нейронів через лінійні сегменти.

В рамках цього розділу буде застосована комбінація методу абстрактної інтерпретації та доведення властивостей моделі за допомогою SMT розв'язувача (Microsoft Z3 [105]).

4.2. Постановка задачі верифікації в околі точки.

Для початку, необхідно формалізувати критерії за якими буде здійснюватися перевірка моделі. Продемонструємо автоматизацію перевірки локальної стійкості (надійності) класифікатора мережевого трафіку методом верифікації у відносному околі точки з вхідного простору. Класифікатор будемо вважати стійким в околі точки, якщо для будь-якої точка в межах даного околу зберігається вихідний клас.

Нехай x – n-вимірний вектор на вхідному просторі ознак, F – множина індексів ознак для яких здійснюється верифікація, ε – параметр, що задає величину околу ($0 \le \varepsilon \le 1$). Тоді позначимо **відносний** окіл точки x за ознаками F як $E(x, \varepsilon, F)$.

$$E\left(x,\ \varepsilon,\ F\right) \subset \mathbb{R}^{n}_{\ge 0}, \qquad (4.1)$$

Будемо вважати, що якщо точка із вхідного простору x' лежить в цьому околі: $x' \in E\left(x,\ \varepsilon,\ F\right)$, то

$$\begin{cases} (1-\varepsilon)\cdot x_i \le x'_i \le (1+\varepsilon)\cdot x_i, & \textit{якщо } i \in F \\ x'_i = x_i, & \textit{інакше} \end{cases} \qquad (4.2)$$

Нехай N – бінарний класифікатор, t – вихідний поріг класифікації. Тоді модель вважатимемо стійкою у відносному околі точки x якщо:

$$\forall x' \in E(x, \varepsilon, F), \begin{cases} N(x) > t \Rightarrow N(x') > t \\ N(x) \le t \Rightarrow N(x') \le t \end{cases} \quad (4.3)$$

4.3. Задача виконуваності обмежень.

Для доведення (або спростування) властивості стійкості моделі у відносному околі точки, зведемо задачу верифікації до задачі виконуваності системи обмежень. В систему включатимемо 3 класи обмежень:

Обмеження належності точки до відносного околу над підмножиною ознак визначають базову форму регіону вхідних даних над яким проводиться верифікація.

$$C_{eps} : \left(\bigwedge_{i \in F} \left((1-\varepsilon) \cdot x_i \le x_i' \le (1+\varepsilon) \cdot x \right) \right) \wedge \left(\bigwedge_{i \notin F} \left(x_i' = x_i \right) \right)$$

Семантичні обмеження над вхідними ознаками необхідні для забезпечення правдоподібності вхідних даних. Ці обмеження відсікають теоретично недосяжні регіони в просторі ознак шляхом встановлення міжознакових відношень.

$$C_{semantic} : \left(\bigwedge x_i' > x_j' \right) \wedge \left(\bigwedge x_k' < x_l' \right) \wedge \left(\bigwedge x_h' = x_m' \right)$$

де $i, j, k, l, h, m \in F$, і (i,j), (k,l) та (h,m) – це пари індексів ознак, які пов'язані відповідним міжознаковим обмеженням.

Вихідне обмеження – це обмеження, яке констатує, що передбачуваний моделлю клас для оригінальної точки та точок з її околу мають співпадати.

$$C_{output} : \begin{cases} N(x') > t, & \textit{якщо } N(x) > t \\ N(x') \le t, & \textit{якщо } N(x) \le t \end{cases} \quad (4.4)$$

Тоді враховуючи всі зазначені обмеження, сформуємо задачу виконуваності у вигляді:

$$\forall x', C_{eps}(x') \wedge C_{semantic}(x') \rightarrow C_{output}(x') \quad (4.5)$$

Якщо SMT-розв'язувачу вдається довести загальнозначимість формули вище, то модель є стійкою на заданому околі з урахуванням накладених семантичних обмежень.

Додатково зазначимо, що на практиці може бути корисним позбутися квантора загальності. В такому разі, задачу виконуваності можна переформулювати у тотожну задачу «невиконуваності» (тобто доведення, що не існує жодної інтерпретації, при якій формула є істиною). Для цього виразимо імплікацію через диз'юнкцію, після чого до отриманого виразу застосуємо заперечення і друге правило де Моргана. Тепер задача верифікації звелася до доведення невиконуваності оберненої формули:

$$\exists x', C_{eps}(x') \wedge C_{semantic}(x') \wedge \neg C_{output}(x') \quad (4.6)$$

Для кращої демонстрації описаного формулювання розглянемо спрощений приклад обмежень для класифікатора мережевих потоків. Нехай мережевий потік характеризується вектором із 5 ознак `["Duration", "IAT Std", "IAT Min", "IAT Max", "Bytes"]`. Маємо бінарний класифікатор потоків *N,* з пороговим значенням $t = 0.51$, при перевищенні якого гіпотеза класифікації приймається і потік відноситься до класу «шкідливі». Припустимо, необхідно довести стійкість моделі *N* у відносному околі «шкідливого» потоку $x = [0.3, 0.1, 0.01, 0.05, 0.5]$ при варіюванні міжпакетних інтервалів (IAT – "inter-arrival time") для $\varepsilon = 0.4$. Тоді індекси ознак, для яких проводиться верифікація: $F = \{1, 2, 3\}$ (нумерація починається з 0). Єдине семантичне обмеження над ознаками: `"IAT Min" ≤ "IAT Max"`. Сформулюємо всі три обмеження для цієї задачі:

1. Обмеження відносного околу:

$$\begin{aligned} &C_{eps}: \\ &\quad x'_0 = 0.3 \wedge \\ &\quad (1-0.4)\cdot 0.1 \le x'_1 \wedge x'_1 \le (1+0.4)\cdot 0.1 \wedge \\ &\quad (1-0.4)\cdot 0.01 \le x'_2 \wedge x'_2 \le (1+0.4)\cdot 0.01 \wedge \\ &\quad (1-0.4)\cdot 0.05 \le x'_3 \wedge x'_3 \le (1+0.4)\cdot 0.05 \wedge \\ &\quad x'_4 = 0.5 \end{aligned}$$

2. Семантичне обмеження над ознаками $C_{semantic}: x'_2 \le x'_3$.
3. Вихідне обмеження $C_{output}: N(x') > 0.51$.

Для автоматичної перевірки виконуваності даних обмежень необхідно створити символьне представлення повного обчислювального графу нейронної мережі *N(x)* у вхідній мові SMT-розв'язувача. В далі буде описано принцип побудови такого представлення для подальшої обробки розв'язувачем Microsoft Z3.

4.4. Застосування Z3 для верифікації нейронної мережі.

Як згадано у попередніх розділах, об'єктом перевірки є повнозв'язна нейромережа з 50 входами, 3 прихованими рівнями по 24 нейронами в кожному та одним вихідним нейроном. Функції активації прихованих рівнів - ReLU, а вихідного нейрону - Sigmoid.

Для аналізу нейронної мережі SMT-розв'язувачем Z3, необхідно побудувати її тотожну інтерпретацію через структурні елементи Z3. Для взаємодії із Z3 будемо використовувати інтерфейс Z3Py (Python Z3 API).

Побудова абстрактного синтаксичного дерева нейромережі відбувається порівнево, починаючи від вхідного рівня і закінчуючи вихідним. Це відбувається згідно базових правил обчислення активацій нейронів для повнозв'язних мереж: вектор-рядок активацій попереднього рівня множиться на вагову матрицю наступного з додаванням вектору-рядку зміщень. Після чого до отриманого вектору поелементно застосовується функція активації. Значення кожного параметру моделі «загортаються» в об'єкт `z3.RealVal`, який представляє дійсне число-константу. Значення вільних змінних (нейрони вхідного шару мережі) представляються через `z3.Real`. Так як в контексті алгебраїчних операцій над дійсними числами, API Z3 підтримує лише скалярні значення, а обчислення значень мережі легше представити у векторно-матричному вигляді ми використовуємо функцію векторизації з бібліотеки numpy. Це дозволяє створювати numpy-масиви з елементами типів розпізнаваємих Z3.

Далі наведено спрощений Python код для конверсії нейронної мережі у форматі TensorFlow в символьний граф Z3.

```
def _z3_relu(x):
    return z3.If(x > 0, x, 0)

z3_real = np.vectorize(z3.Real)
z3_real_val = np.vectorize(z3.RealVal)
z3_relu = np.vectorize(_z3_relu)

def to_z3(model):
    input_size = model.input_shape[1]
    input = z3_real([ "x_%s" % i for i in range(inputs_size) ])
    X = input
    for layer in model.layers:
        params = layer.get_weights()
        W = z3_real_val(params[0]) # Матриця ваг поточного рівня
        b = z3_real_val(params[1]) # Вектор зміщень
        X = X @ W + b
        # Не активувати останній шар
        if layer != model.layers[-1]:
            X = z3_relu(X)
    output = X
    return output, input
```

На виході отримуємо: 1) `input` - вектор вільних вхідних змінних 2) `X` - вектор абстрактних синтаксичних дерев, які відображають процес обрахунку значень кожного з вихідних нейронів (у випадку бінарної класифікації вихідний нейрон лише один). При цьому тут навмисне опускається активація вихідного нейрона (нейронів), адже нелінійні функції, такі як sigmoid, неможливо тотожно представити в Z3 лише через базові арифметичні операції та булеву логіку.

Тепер нехай x - точка з вхідної вибірки, і необхідно провести верифікацію моделі у **відносному** околі цієї точки. Тоді вхідні обмеженням задамо у формі:

```
constraints = [
    z3.And(x[i] * RealVal(1 - eps) <= input[i], input[i] <= x[i] * RealVal(1
+ eps))
    for i in range(len(x))
]
```

Для відсіювання практично недосяжних регіонів у вхідному просторі ознак, є зміст ввести додаткові обмеження, що випливають із семантики даних. Наприклад, для ознак мережевих потоків має виконуватись нерівність «Fwd IAT Min (f1) ≤ Fwd IAT Max (f2)», де f1 та f2 – індекси цих ознак у вхідному векторі. Тоді множина вхідних обмежень розширюється предикатом:

```
constraints += [input[f1] <= input[f2]]
```

У випадку бінарної класифікації, стоїть завдання перевірки чи всі елементи, які лежать в околі оригінального елементу в просторі ознак, належатимуть тому-ж класу. В нашому випадку, мережевий потік класифікується як злоякісний, якщо впевненість на виході моделі перевищує порогове значення 0.51, тобто необхідно перевірити чи для всіх точок із околу, що розглядається, значення активації вихідного нейрону моделі буде більше за 0.51.

Останнім кроком є зведення задачі перевірки обмежень над областю входів до задачі виконуваності. Виконуваність вихідних обмежень за умови виконання вхідних можна виразити додатковим обмеженням, через квантор загальності (∀) та оператор імплікації (⇒).

```
solver = SolverFor("NRA")
solver.add(constraints)
solver.add(
    z3.ForAll(input, z3.Implies(constraints, output[0] >
inv_sigmoid(threshold))))
solver.check() == sat
```

Такий підхід однак, вимагає від розв'язувача підтримку не лише лінійної арифметики дійсних чисел (LRA - Linear Real Arithmetic), та булевих виразів, але й підтримку кванторів, що негативно впливає на час доведення. Як зазначалось в попередньому розділі, квантора загальності можна позбудитися звівши задачу до проблеми доведення невиконуваності. Це дозволяє застосувати розв'язувач для безкванторної арифметики QF_LRA (Quantifier-Free Linear Arithmetic), що може прискорити процес доведення:

```
solver = SolverFor("QF_LRA")
solver.add(constraints)
solver.add(Not(output[0] > inv_sigmoid(threshold)))
solver.check() == unsat
```

В побудованому обчислювальному графі відсутня функція активації вихідного нейрону (яка у випадку нашої задачі - сигмоїда). Для представлення сигмоїди необхідна трансцендентна функція, а тому її не можна тотожно представити лише лінійною арифметикою та булевою алгеброю. Зазвичай, для представлення таких нелінійних функцій використовують апроксимацію

лінійними сегментами. В контексті поставленої задачі, однак, пропонується скористатися тим, що сигмоїда є монотонно зростаючою на всій множині визначення, а тому порівняння її значення з очікуваним порогом класифікації (`threshold`) можна замінити на порівняння її аргументу (лінійної комбінації активацій нейронів передостаннього рівня) зі значенням оберненої функції від порогу (`inv_sigmoid(threshold)`). Таким чином можливо уникнути необхідності апроксимації і таке представлення мережі залишається математично тотожним оригінальному.

Для демонстрації алгоритму описаного вище, на Рис. 4.1 показана тривіальна нейромережа з 2 входами, одним прихованим рівнем та одним вихідним нейроном.

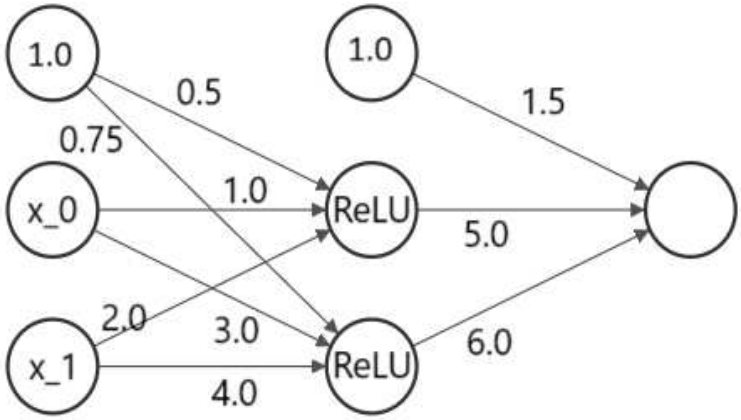

Рис. 4.1. Приклад графу нейромережі.

На Рис. 4.2 наведено структурну інтерпретацію обчислювального графу вихідного нейрону такої мережі, виражену елементами Z3Py (для простоти сприйняття, обгортки «z3.RealVal» над дійсними константами пропускаються).

```
5*z3.If(
    z3.Real("x_0")*1 + z3.Real("x_1")*2 + 0.5 > 0,
    z3.Real("x_0")*1 + z3.Real("x_1")*2 + 0.5,
    0) +
6*z3.If(
    z3.Real("x_0")*3 + z3.Real("x_1")*4 + 0.75 > 0,
    z3.Real("x_0")*3 + z3.Real("x_1")*4 + 0.75,
    0) + 1.5
```

Рис. 4.2. Представлення вихідного нейрону в Z3Py.

Слід зазначити, що ідея “наївної” конверсії повнозв’язної нейронної мережі у вираз Z3 вже застосовувалась (як-от в Sapphire [106]). Таке представлення є надзвичайно потужним, адже дозволяє задавати довільні обмеження над вхідними змінними та вихідними значеннями в рамках теорій відомих Z3 (лінійна, нелінійна, булева арифметика і т.д.). Цей підхід однак поки не набув значного розвитку через неприйнятну часозатратність при обробці SMT розв’язувачем такого обчислювального графу, що особливо помітно при зростанні кількості шарів в нейромережі. В наступному підрозділі запропоновано новий спосіб вирішення цієї проблеми для випадку локальної верифікації нейромережі в околах точок з вхідного простору.

4.5. Алгоритм спрощення структури представлення нейронної мережі.

Одним зі шляхів до скорочення часу необхідного для доведення чи спростування властивостей нейронної мережі SMT-розв’язувачем є спрощення представлення її обчислювального графу, а саме виявлення та вилучення нелінійностей.

Розглянемо ситуацію, коли всі функції активації нейронів мережі є глобально нелінійними, але при цьому кусково-лінійними (тобто лінійними на інтервалах). Для визначення рельєфу кордону прийняття рішень класифікатора над усім вхідним простором необхідна повна композиція нелінійних активацій усіх нейронів мережі. Однак, на відносно малому, локальному проміжку вхідних значень кількість нелінійних функцій необхідних для тотожного представлення нейронної мережі значно скорочується. В такому випадку обчислювальний граф нейромережі можна спростити. Для кожного нейрону розглядається можливість спрощення шляхом заміни його нелінійної функції активації, на лінійну, тотожну з оригінальною на заданому регіоні вхідного простору. Це стає можливим, якщо діапазон можливих значень функції активації лежить повністю в межах інтервалу її лінійності. Чим вужчий діапазон входів мережі, тим більше спрощень нейронних активацій на ньому можна провести. При цьому у випадку звуження вхідного

діапазону до єдиної точки, обчислювальний граф вироджується в цільну лінійну функцію.

На Рис. 4.3 продемонстрована ідея локальної апроксимації кусково-лінійної функції до нової, тотожної з оригінальною лише на певному інтервалі. Як видно, при звуженні інтервалу, глобальна складність апроксимованої функції зменшується, адже зменшується кількість точок зламу («кутових точок»).

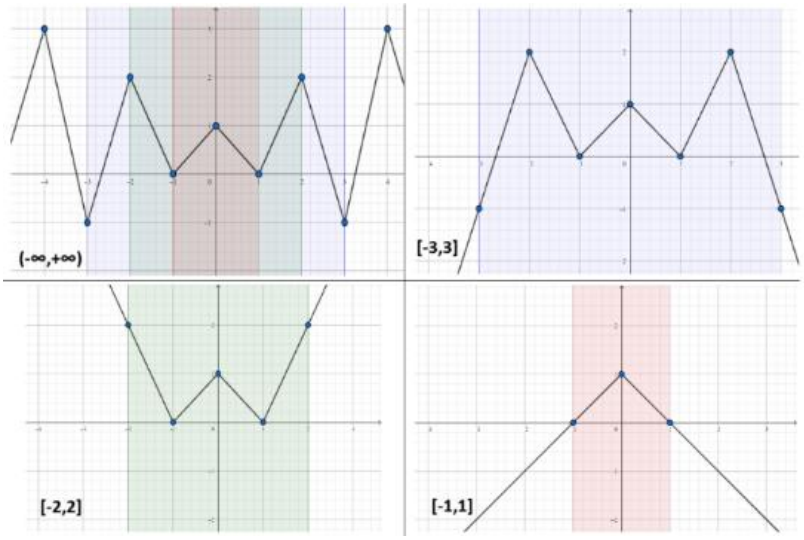

Рис. 4.3. Приклади локальних апроксимацій кусково-лінійної функції.

Для нейронних мереж властивість нелінійності забезпечується функціями активації нейронів - в даному випадку це ReLU. Водночас, саме функції активації є найбільшим фактором складності при аналізі графу нейромережі SMT-розв’язувачем, адже кількість можливих шляхів породжених умовними розгалуженнями («if-else», в які транслюється ReLU) у графі комбінаторно зростає разом зі зростанням глибини мережі.

ReLU складається лише з двох лінійних сегментів, які визначаються на проміжках (-inf, 0], та (0, +inf). Якщо на певній підмножині вхідних значень моделі діапазон значень, що подаються на вхід нейрона, лежить повністю в одному з цих двох проміжків, то функція активації ReLU для даного нейрону рівноцінна константі 0 (x = 0), або ж функції тотожності (x = x). Таким чином, при заміні ReLU на одну з тривіальних альтернатив в побудованому обчислювальному графі нейронної мережі, задача доведення властивостей SMT-розв’язувачем значно спрощується, адже зменшується кількість розгалужень. Можливість такого

спрощення згадується в [100], щоправда в контексті звуження можливих меж входів використовуючи симплекс-метод.

В цій дисертаційній роботі пропонується новий метод для знаходження можливостей спрощення функцій активації ReLU шляхом розв'язання локальної SMT-задачі для кожного окремого нейрону. Нехай необхідно провести верифікацію повнозв'язної нейромережі в околі точки x з вхідного набору даних. Як і раніше, окіл позначимо як $E(x,\varepsilon,F)$. Спрощення здійснюється порівнево в один прохід по графу мережі, починаючи з першого прихованого рівня. Для кожного рівня алгоритм виконує 2 фази.

Перша фаза – це спроба провести попередню «наївну» оптимізацію функцій активації нейронів без розв'язування SMT-задач. Для нейрона проводиться оцінка найширших теоретично можливих меж його вхідних значень виходячи з обрахованих меж активацій нейронів попереднього рівня. Очевидно, що отримані межі не обов'язково є практично досяжними і, зазвичай, будуть значно ширші ніж реально можливі. Якщо нижня межа вхідних значень нейрону > 0, то ReLU заміняється на тотожну функцію, а якщо верхня межа ≤ 0, то ReLU заміняється на константу 0. При цьому межі для вхідного рівня визначаються безпосередньо обраним околом $E(x,\varepsilon,F)$. Якщо для нейрону на першій фазі спрощення не відбулося, то для нього буде застосована друга фаза.

В **другій фазі**, SMT-розв'язувач проводить повний аналіз обчислювального графу кожного з неспрощених нейронів даного рівня над околом $E(x,\varepsilon,F)$. Нам вже відоме значення, що подається на вхід функції активації нейрона для вхідної точки x. Позначимо це значення як $n(x)$. Задача перевірки можливості спрощення ReLU зводиться до доведення відсутності такої точки $x' \in E(x,\varepsilon,F)$ для якої значення, що подається на вхід даного нейрону $n(x')$ лежатиме по протилежний бік від 0 відносно $n(x)$. Це можна виразити у формі обмеження:

$$C_{neuron}: \begin{cases} n(x') > 0, & якщо\ n(x) < 0 \\ n(x') \le 0, & якщо\ n(x) \ge 0 \end{cases} \quad (4.7)$$

Якщо SMT-розв'язувач доводить невиконуваність даного обмеження, то діапазон значень $n(x')$ повністю лежить в одному з інтервалів лінійності ReLU для $\forall x' \in E(x, \varepsilon, F)$, що дозволяє провести спрощення. ReLU замінюється на тотожну функцію, якщо $n(x) > 0$, або ж на константу 0, якщо $n(x) \leq 0$. В іншому випадку спрощення для даного нейрона не відбувається.

Нижче наведено лістинг коду алгоритму швидкої оцінки найширший теоретично-можливих меж, який використовується в першій оптимізаційній фазі. Параметри `weights` та `biases` – це матриця ваг та вектор зміщень, які характеризують параметри моделі між попереднім та поточним рівнем, Параметр `input_bounds` – це масив пар вже обрахованих меж активацій попереднього рівня. Для найпершого рівня межі визначаються безпосередньо обмеженнями накладеними на вхідні змінні нейромережі на етапі постановки задачі верифікації.

```
def _next_layer_bounds(weights, biases, input_bounds):
    next_min_bounds = []
    next_max_bounds = []
    prev_count, next_count = weights.shape
    min_bounds, max_bounds = input_bounds
    # for each neuron in the next layer compute the bounds
    for i in range(next_count):
        # compute the bounds for the i-th neuron
        next_min_bound = 0
        next_max_bound = 0
        for j in range(prev_count):
            if weights[j][i] > 0:
                next_min_bound += weights[j][i] * min_bounds[j]
                next_max_bound += weights[j][i] * max_bounds[j]
            else:
                next_min_bound += weights[j][i] * max_bounds[j]
                next_max_bound += weights[j][i] * min_bounds[j]
        next_min_bound += biases[i]
        next_max_bound += biases[i]
        next_min_bounds.append(next_min_bound)
        next_max_bounds.append(next_max_bound)

    return next_min_bounds, next_max_bounds
```

Після обчислення теоретичних меж, проводиться перша спроба спрощення поточного рівня (її код тривіальний).

Далі розпочинається друга фаза, лістинг коду якої наведений нижче. Параметр `layer_activation` - це матриця активацій, яка вже пройшла попередню оптимізацію в першій фазі, `constraints` - це предикатний вираз Z3, який представляє вхідні та семантичні обмеження, `neurons` - це масив обчислювальних графів Z3 нейронів поточного рівня, `x` - це масив числових значень, які поступають на вхід функцій активації кожного з нейронів поточного рівня конкретно для точки в околі якої здійснюється верифікація.

```
def _optimize_layer_activations(layer_activation, neurons, x, constraints):
    for i in range(len(neurons)):
        if layer_activation[i] == ReLU:
            s = Solver()
            s.add(constraints)
            if x[i] > 0:
                s.add(Not(neurons[i] >= 0))
                if s.check() == unsat:
                    layer_activation[i] = Linear
                else:
                    neurons[i] = z3_relu(neurons[i])
            elif x[i] < 0:
                s.add(Not(neurons[i] <= 0))
                if s.check() == unsat:
                    neurons[i] = RealVal(0.)
                    layer_activation[i] = Const0
                else:
                    neurons[i] = z3_relu(neurons[i])
            else:
                neurons[i] = z3_relu(neurons[i])
        if layer_activation[i] == Const0:
            neurons[i] = RealVal(0.)
```

Зауважимо, що ідея такого спрощення є евристичною. Очікується, що витрати на інкрементальне спрощення шляхом понейронного розв'язування SMT-задач на вже спрощених підграфах є менш часозатратними ніж одноразовий аналіз оригінального графу нейромережі. Крім того, очевидно, що зі збільшенням вхідного околу зменшується кількості можливостей спрощення функцій активації, і тому ефективність такого підходу буде падати. В найгіршому випадку, час виконання буде більшим ніж при аналізі неоптимізованого графу мережі. Незважаючи на це, в контексті задачі верифікації моделі класифікації мережевих

потоків, саме така оптимізація дозволила проаналізувати повну множину точок інтересу за прийнятний час, про що йдеться в наступному підрозділі.

4.6. Експериментальне застосування методу формальної верифікації нейронних мереж

Останнім кроком цього дослідження було застосування розробленого методу верифікації для перевірки властивостей системи виявлення трафіку зловмисного ПЗ на предмет її стійкості до можливих збурень у вхідних даних. Модель класифікації представлена нейронною мережею, навченою на мережевих потоках виділених з набору даних ISCX Botnet 2014 [19]. Як вже згадувалось, кожен мережевий потік представлений набором з 50 ознак, отриманих за допомогою інструменту CIC Flow Meter V4.0 [21]. Порогове значення класифікатора встановлено в 0.51 – якщо значення на вихідному нейроні мережі перевищує цей поріг, потік вважається «шкідливим», тобто таким, що містить активність ботнетів. Після проведення маркування всіх потоків з навчальної вибірки у відповідності з результатами класифікації, було виділено всі потоки, які модель класифікувала вірно, але при цьому вони лежать близько до межі прийняття рішень (тобто для них прогнозована впевненість > 0.51, але < 0.55) – всього 836 потоків (тобто 0.76% від повної вибірки).

Верифікація у відносному околі точок здійснювалась над семантично зв’язаною групою із 9 ознак, а саме, «загальна тривалість» та статистики «міжпакетних інтервалів» потоків (інші ознаки залишались фіксованими):

```
"Flow Duration", "Fwd IAT Total", "Fwd IAT Std", "Fwd IAT Max", "Fwd IAT Min", "Bwd IAT Total", "Bwd IAT Std", "Bwd IAT Max", "Bwd IAT Min"
```

При цьому накладались додаткові обмеження правдоподібності:

```
"Fwd IAT Max" >= "Fwd IAT Min",
"Bwd IAT Max" >= "Bwd IAT Min"
```

Верифікація проводилась для відносних околів з параметром ε: 0.05, 0.1 та 0.15. Характеристики системи на якій проводились обчислення наступні: процесор AMD Ryzen 7 5800H (3200 Mhz, 8 ядер), 16 GB ОЗУ. Обчислення проводились в однопоточному режимі.

Таблиця 4.1
Результати верифікації класифікатора мережевих потоків

ε	**# verified**	**# failed**	**Time**	**Avg. final ReLU count**
0.05	763	73	6m 32s	0.23
0.1	722	114	15m 8s	0.48
0.15	711	125	92m 22s	0.83

В Табл. 4.1, стовпець “# verified” показує кількість мережевих потоків, для яких доведено, що завжди зберігається клас при довільному варіюванні міжпакетних інтервалів в зазначених межах. Стовпець “# failed” відображає кількість випадків, для яких існує така варіація, при якій передбачуваний моделлю клас не співпадає з оригінальним. Як і очікувалось, при зростанні ε, також зростає і кількість можливостей помилки класифікації. В останньому стовпці наведено середню кількість функцій ReLU, які залишилися після проведення спрощення обчислювального графу мережі. Як видно, для більшості випадків достатньо щонайбільше однієї ReLU для математично-тотожного представлення нейромережі на вхідних діапазонах заданих обмеженнями верифікації. Це підтверджує гіпотезу, що на локалізованих вхідних інтервалах, рельєф межі прийняття рішень суттєво тривіалізується і містить лише кілька точок нелінійності, а в більшості випадків, взагалі не більше однієї. Таким чином, даний метод дозволяє проводити верифікацію локальних властивостей повнозв’язних нейромереж за значно менший час. Для порівняння, без використання описаного методу оптимізації, верифікація тієї самої нейромережі у відносного околі лише однієї точки з 836 займає від ~2 хвилин для $\varepsilon = 0.05$, за умови використання ідентичних обчислювальних ресурсів. При цьому для деяких точок SMT-розв’язувач взагалі не здатен знайти доведення чи спростувати гіпотезу верифікації

навіть через 10 хвилин після початку пошуку рішення, що робить використання Z3 просто непрактичним для даної задачі без застосування розробленого оптимізуючого рішення. Саме тому, розроблений метод оптимізації обчислювального графу нейромереж є критично необхідним для можливості доведення властивостей надійності класифікаторів такого роду. Зокрема, для формалізованої перевірки систем аналізу мережевих даних на основі нейромереж, щодо їх стійкості до змагальних атак, як і було експериментально показано.

Висновки до Розділу 4.

В цьому розділі було запропоновано та формалізовано критерій локальної стійкості системи виявлення мережевих загроз на основі нейронних мереж до можливих змагальних впливів. Було створено та реалізовано метод для верифікації сформованого критерію стійкості за допомогою SMT-розв'язувача. Було продемонстровано практичне застосування цього методу для верифікації системи аналізу мережевого трафіку на предмет активності ботнетів. Це є важливим результатом, адже застосування ШІ в області кібербезпеки суттєво обмежене об’ємом наявних наборів даних для їх емпіричного тестування. Крім того, цей метод дає можливість гарантувати локальну стійкість класифікаторів до змагальних атак, тобто навмисних спроб ухилення зловмисників від виявлення їх дій. Показано, що алгоритм здатен як довести коректність роботи моделі класифікації на певних регіонах вхідного простору даних, так і виявити проблемні зони, де модель може припуститися помилки.

Крім того, отримані експериментальні дані підтверджують ефективність запропонованої оптимізації обчислювального графу мережі через порівневе спрощення функцій активації шляхом рішення локальних SMT-задач.

Описаний підхід верифікації нейромереж може бути застосований не лише в задачах кібербезпеки, а й в будь-яких системах класифікації, де точність є критичною, зокрема в розпізнаванні образів або мови.

Одним із напрямків подальших досліджень, є використання запропонованого методу верифікації для мереж із довільними функціями активації. Наприклад, через їх апроксимацію лінійними сегментами.

Крім того, запропонований спосіб інкрементальної оптимізації графу мережі також має значний потенціал для подальшого розвитку. По-перше, саму процедуру спрощення обчислювального графу можливо значно прискорити, якщо послідовне рішення понейронних SMT-задач в межах одного рівня замінити на паралельне, адже такі задачі не потребують між-потокової синхронізації. По-друге, при вилученні нелінійностей можна брати до уваги межі можливих вхідних значень не якогось окремого нейрону, а комбінації меж для кількох нейронів даного рівня.

Нарешті, самим важливим напрямком подальших досліджень в безпосередньо області кібербезпеки є створення методів по підвищенню надійності та стійкості моделей виявлення мережевих загроз, як до змагальних атак, так і до випадкових флуктуацій в мережевих даних.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальну дослідницьку і практичну задачу – оцінювання, верифікації та підвищення стійкості систем виявлення шкідливого мережевого трафіку на основі методів штучного інтелекту, до можливих змагальних впливів з боку зловмисника.

В ході дослідження отримано ряд важливих науково-практичних результатів, серед яких слід виділити наступні:

1. На основі проведеного огляду та аналізу існуючих методів виявлення мережевих загроз, а саме трафіку зловмисних програм, засобами штучного інтелекту, показано, що штучні нейронні мережі дають змогу ефективно виявляти ботнети. Проте вони вимагають ретельного підбору характеристичних ознак, достатнього обсягу навчальних даних та вирішення проблем перенавчання і покращення здатності моделей класифікації до узагальнення. Водночас виявлено необхідність розв’язання двох, недостатньо досліджених, проте критичних задач у цій галузі. Перша полягає у необхідності розробки методів виявлення шкідливого програмного забезпечення, здатного обходити існуючі системи кібербезпеки шляхом мінімальних модифікацій ознак мережевої активності. Друга полягає у формалізації та забезпеченні математично-доведених гарантій надійності функціонування штучних нейронних мереж під час їх застосування для виявлення мережевих вторгнень у комп’ютерних системах.

2. В роботі експериментально показана важливість диверсифікації множини типів трафіку зловмисного ПЗ у навчальних та тестувальних вибірках. Сформовано новий розширений набір мережевих даних, шляхом комбінації відкритих наборів ICSX Botnet 2014, CTU-13, а також різних даних отриманих з Malware Capture Facility Project. На основі отриманого набору було проведено навчання класифікаторів мережевого трафіку з використанням алгоритмів машинного навчання та нейронних мереж. Найкращі результати показали класифікатори на основі алгоритму випадкового лісу у комбінації з методом головних компонент, а також

глибокі нейронні мережі. Для випадкового лісу значення площі під ROC-кривою на розширених тестових даних склало 0.95. Модель здатна виявляти приклади такого зловмисного ПЗ як WannaCry, TrickBot та Emotet з повнотою > 80%, при цьому зберігаючи хибнопозитивний рівень нижче 0.2%. При використанні глибокого навчання, найкращі результати були отримані для повнозв'язних нейромереж, для яких значення площі під ROC-кривою складає 0.975 на розширеному тестовому наборі. При хибнопозитивному рівні <0.2% модель показала повноту виявлення 80.7%. При збільшенні порогового значення класифікатора, показник повноти сягав 91.3% при цьому хибнопозитивний рівень не перевищував 0.75%. Також було продемонстровано можливість аналізу потоків мережевих даних як часових рядів, зокрема рекурентними та згортковими нейромережами із темпоральною складовою. Для доведення практичної цінності проведених досліджень, був реалізований багатоцільовий аналізатор мережевого трафіку. Прототип було впроваджено в експлуатацію у ролі системи виявлення мережевих загроз у компанії ТОВ «НВП «Радікс», що спеціалізується на розробці апаратного та програмного забезпечення для АЕС.

3. Розроблено метод оцінки вразливості та підвищення стійкості нейромереж-класифікаторів трафіку до змагальних атак, на основі вдосконалення «швидкого методу знаку градієнту» для генерації штучних даних, а також доповнення ними навчальної вибірки. На відміну від методів формування змагальних прикладів поширених у галузі обробки зображень, запропонований метод враховує низьковимірність та гетерогенність простору ознак мережевого трафіку, а також їх логічні та статистичні обмеження. Така адаптація дозволила створити правдоподібні змагальні приклади ботнет-трафіку та оцінити вразливість класифікаторів мережевих даних до атак такого роду. Запропонований метод також має перевагу надання семантичної інтерпретації результатів: він дозволяє окремо оцінити вразливість системи до таких змін, як варіація міжпакетних інтервалів чи

маніпуляція розмірами корисного навантаження пакетів. Також було експериментально продемонстровано користь доповнення навчальної вибірки змагальними прикладами — це дало змогу суттєво знизити вразливість нейромережі-класифікатора до атак на основі збурень у розмірах пакетів, без деградації інших показників ефективності виявлення.

4. Розроблено та застосовано метод формальної верифікації нейронних мереж на предмет їх стійкості до збурень у вхідних даних. Формалізовано критерій локальної стійкості класифікатора та розроблено метод верифікації нейромереж за допомогою SMT-розв’язувача. Запропоновано представляти обчислювальний граф нейромережі у вигляді SMT-формули та проводити локалізовану перевірку коректності класифікації при внесенні обмежених збурень у вхідні параметри. Завдяки цьому вдалося знайти точки чи ділянки вхідного простору класифікатора, де нейромережа все ще здатна помилятися, або ж довести, що на формально заданих регіонах вхідних даних вона працює безпомилково. Також було розроблено новий метод спрощення представлення графу нейромережі на рівні окремих нейронів, що суттєво зменшує обчислювальну складність розв’язання поставленої SMT-задачі. Описані методи було використано для верифікації локальної стійкості системи виявлення трафіку зловмисних програм до можливості ухилення від виявлення шляхом варіювання міжпакетних інтервалів. Це дозволяє отримувати додаткові гарантії та збільшувати рівень довіри до систем автоматизованого виявлення загроз, що є надзвичайно важливим у сфері кібербезпеки.

Отримані результати та розроблені методи дозволяють створювати системи виявлення мережевих загроз на основі засобів штучного інтелекту та здійснювати оцінку, формальну верифікацію та підвищення стійкості таких систем до змагальних впливів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Debar H. An Introduction to Intrusion-Detection Systems / H. Debar. – 2009.

2. Pathan A.-S. K. New Unknown Attack Detection with the Neural Network–Based IDS / A.-S. K. Pathan. – 2014.

3. Arnaldo I. Learning Representations for Log Data in Cybersecurity / I. Arnaldo, A. Cuesta-Infante, A. Arun, M. Lam, C. Bassias, K. Veeramachaneni // Lecture Notes in Computer Science. – 2017. – Vol. 10423. – P. 250–268. – DOI: 10.1007/978-3-319-60080-2_19.

4. Dewes C. An analysis of Internet chat systems / C. Dewes, A. Wichmann, A. Feldmann // Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC '03). – New York: ACM, 2003. – P. 51–64. – DOI: 10.1145/948205.948214.

5. Umer M.F. Flow-based intrusion detection: Techniques and challenges / M.F. Umer, M. Sher, Y. Bi // Computers & Security. – 2017. – T. 70. – C. 238–254. – DOI: 10.1016/j.cose.2017.05.009.

6. Roughan M. Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification / M. Roughan, S. Sen, O. Spatscheck, N. Duffield // Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04). – New York: ACM, 2004. – P. 135–148. – DOI: 10.1145/1028788.1028805.

7. Sen S. Accurate, scalable in-network identification of p2p traffic using application signatures / S. Sen, O. Spatscheck, D. Wang // Proceedings of the 13th International Conference on World Wide Web (WWW '04). – New York: ACM, 2004. – P. 512–521. – DOI: 10.1145/988672.988742.

8. Moore A. W. Internet traffic classification using bayesian analysis techniques / A. W. Moore, D. Zuev // Proceedings of the 2005 ACM SIGMETRICS International

Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '05). – New York: ACM, 2005. – P. 50–60. – DOI: 10.1145/1064212.1064220.

9. Zeek. URL: https://zeek.org/ (дата звернення: 13.04.2025).

10. Snort. URL: https://www.snort.org/ (дата звернення: 13.04.2025).

11. Pretty Park. URL: http://virus.wikidot.com/prettypark (дата звернення: 13.04.2025).

12. Ahmad W. Why Botnets Persist: Designing Effective Technical and Policy Interventions / W. Ahmad // Internet Policy Research Initiative. – 2019. – IPRI(2019)02. – URL: https://internetpolicy.mit.edu/wp-content/uploads/2019/09/publications-ipri-2019-02.pdf (дата звернення: 13.04.2025).

13. Livadas C. Using Machine Learning Techniques to Identify Botnet Traffic / C. Livadas, R. Walsh, D. Lapsley, W. T. Strayer // Proceedings of the 31st IEEE Conference on Local Computer Networks. – Tampa, FL: IEEE, 2006. – P. 967–974. – DOI: 10.1109/LCN.2006.322210.

14. Strayer T. Botnet Detection Based on Network Behavior / T. Strayer, D. Lapsley, R. Walsh, C. Livadas // Advances in Information Security. – 2007. – Vol. 36. – P. 1–24. – DOI: 10.1007/978-0-387-68768-1_1.

15. Xing Y. Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation / Y. Xing, H. Shu, H. Zhao, D. Li, L. Guo // Mathematical Problems in Engineering. – 2021. – Article ID 6640499. – 24 c. – DOI: 10.1155/2021/6640499.

16. Alomari D. A Survey on Botnets Attack Detection Utilizing Machine and Deep Learning Models / D. Alomari, F. Anis, M. Alabdullatif, H. Aljamaan // Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering (EASE '23). – New York: ACM, 2023. – C. 493–498. – DOI: 10.1145/3593434.3593967.

17. Zhao D. Botnet detection based on traffic behavior analysis and flow intervals / D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant // Computers & Security. – 2013. – Vol. 39. – P. 2–16. – DOI: 10.1016/j.cose.2013.04.007.

18. French chapter of The Honeypot Project. URL: https://www.honeynet.org/2012/12/05/french-chapter-status-report-2012/ (дата звернення: 13.04.2025).

19. Beigi E. B. Towards effective feature selection in machine learning-based botnet detection approaches / E. B. Beigi, H. H. Jazi, N. Stakhanova, A. A. Ghorbani // 2014 IEEE Conference on Communications and Network Security. – 2014. – P. 247–255.

20. ISCX Botnet 2014 dataset. URL: https://www.unb.ca/cic/datasets/botnet.html (дата звернення: 13.04.2025).

21. Habibi Lashkari A. CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection / A. Habibi Lashkari. – 2018. – URL: https://github.com/ISCX/CICFlowMeter (дата звернення: 13.04.2025). – DOI: 10.13140/RG.2.2.13827.20003.

22. Velasco-Mata J. Real-time botnet detection on large network bandwidths using machine learning / J. Velasco-Mata, V. González-Castro, E. Fidalgo та ін. // Scientific Reports. – 2023. – Т. 13. – С. 4282. – DOI: 10.1038/s41598-023-31260-0.

23. Moorthy R.S.S. Botnet Detection Using Artificial Intelligence / R.S.S. Moorthy, N. Nathiya // Procedia Computer Science. – 2023. – Т. 218. – С. 1405–1413. – DOI: 10.1016/j.procs.2023.01.119.

24. Salih Y.T. Machine Learning Approaches for Botnet Detection in Network Traffic / Y.T. Salih, A. Fenjan, S.R. Ahmed, H. Ali, E.N. Abdulwahab, S. Algruri, N.A. Kurdi, M. Al-Sarem, J.F. Tawfeq // Proceedings of the Cognitive Models and

Artificial Intelligence Conference (AICCONF '24). – New York: ACM, 2024. – C. 310–315. – DOI: 10.1145/3660853.3660933.

25. Tikekar P. An Approach for Detection of Botnet Based on Machine Learning Classifier / P. Tikekar, S. Sherekar, J. Kumar // SN Computer Science. – 2024. – T. 5. – DOI: 10.1007/s42979-024-02636-4.

26. Pektaş A. Botnet detection based on network flow summary and deep learning / A. Pektaş, T. Acarman // International Journal of Network Management. – 2018. – T. 28. – DOI: 10.1002/nem.2039.

27. Saad S. Detecting P2P botnets through network behavior analysis and machine learning / S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, P. Hakimian // Proceedings of the 9th Annual International Conference on Privacy, Security and Trust (PST). – 2011. – C. 174–180.

28. Kumar S. Internet of Things is a revolutionary approach for future technology enhancement: a review / S. Kumar, P. Tiwari, M. Zymbler // Journal of Big Data. – 2019. – T. 6. – C. 111. – DOI: 10.1186/s40537-019-0268-2.

29. Griffioen H. Examining Mirai's Battle over the Internet of Things / H. Griffioen, C. Doerr // Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20). – New York: ACM, 2020. – C. 743–756. – DOI: 10.1145/3372297.3417277.

30. Lee H. Intrusion Detection Systems for IoT / H. Lee, A. Mudgerikar, N. Li, E. Bertino // In: IoT for Defense and National Security. – IEEE, 2023. – C. 237–258. – DOI: 10.1002/9781119892199.ch13.

31. Sicato J. A Comprehensive Analyses of Intrusion Detection System for IoT Environment / J. Sicato, S.K. Singh, S. Rathore, J. Park // Journal of Information Processing Systems. – 2020. – T. 16. – C. 975–990. – DOI: 10.3745/JIPS.03.0144.

32. Lefoane M. Internet of Things botnets: A survey on Artificial Intelligence based detection techniques / M. Lefoane, I. Ghafir, S. Kabir, I.-U. Awan // Journal of Network and Computer Applications. – 2025. – Т. 236. – С. 104110. – DOI: 10.1016/j.jnca.2025.104110.

33. Spadaccino P. Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing / P. Spadaccino, F. Cuomo // arXiv. – 2020. – arXiv:2012.01174. – DOI: 10.48550/arXiv.2012.01174.

34. Sriram S. Network Flow based IoT Botnet Attack Detection using Deep Learning / S. Sriram, R. Vinayakumar, M. Alazab, S. KP // IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). – Toronto, Canada, 2020. – С. 189–194. – DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162668.

35. Chen R. GBDT-IL: Incremental Learning of Gradient Boosting Decision Trees to Detect Botnets in Internet of Things / R. Chen, T. Dai, Y. Zhang, Y. Zhu, X. Liu, E. Zhao // Sensors (Basel). – 2024. – Т. 24.

36. Shareef S.K. Enhanced botnet detection in IoT networks using zebra optimization and dual-channel GAN classification / S.K. Shareef, R.K. Chaitanya, S. Chennupalli та ін. // Scientific Reports. – 2024. – Т. 14. – С. 17148. – DOI: 10.1038/s41598-024-67865-2.

37. Trojovská E. Zebra Optimization Algorithm: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm / E. Trojovská, M. Dehghani, P. Trojovský // IEEE Access. – 2022. – Т. 10. – С. 49445–49473. – DOI: 10.1109/ACCESS.2022.3172789.

38. Memos V.A. AI-Powered Honeypots for Enhanced IoT Botnet Detection / V.A. Memos, K.E. Psannis // 3rd World Symposium on Communication Engineering (WSCE). – 2020. – С. 64–68.

39. Lee S. Honeypot Coupled Machine Learning Model for Botnet Detection and Classification in IoT Smart Factory – An Investigation / S. Lee, A. Abdullah, N. Jhanjhi, S. Kok // MATEC Web of Conferences. – 2021. – Т. 335. – С. 04003. – DOI: 10.1051/matecconf/202133504003.

40. Garcia S. IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Dataset] / S. Garcia, A. Parmisano, M.J. Erquiaga // Zenodo. – 2020. – DOI: 10.5281/zenodo.4743746.

41. Rabbani M. Device Identification and Anomaly Detection in IoT Environments / M. Rabbani, J. Gui, F. Nejati, Z. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. Opushnyev, R. Lu, A.A. Ghorbani // IEEE Internet of Things Journal. – 2024. – Грудень.

42. AL-Akhras M. Botnet attacks detection in IoT environment using machine learning techniques / M. AL-Akhras, A. Alshunaybir, H. Omar, S. Alhazmi // International Journal of Data and Network Science. – 2023. – Т. 7, № 4. – С. 1683–1706.

43. Guo Z. Botnet Detection Method Based on Artificial Intelligence / Z. Guo, J. Peng, J. Fu, Y. Cheng, C. Chen // 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC). – Hangzhou, China, 2019. – С. 487–494. – DOI: 10.1109/DSC.2019.00080.

44. Khan R.U. An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers / R.U. Khan, X. Zhang, R. Kumar, A. Sharif, N. Amiri Golilarz, M. Alazab // Applied Sciences. – 2019. – Т. 9. – С. 2375. – DOI: 10.3390/app9112375.

45. Farhan M. Efficient Botnet Detection using Feature Ranking and Hyperparameter Tuning / M. Farhan, M. G. // International Journal of Computer Applications. – 2019.

46. Чичкарьов Є. Метод вибору ознак для системи виявлення вторгнень з використанням ансамблевого підходу та нечіткої логіки / Є. Чичкарьов, О. Зінченко, А. Бондарчук, Л. Асєєва // Кібербезпека: освіта, наука, техніка. – 2023. – № 1(21). – С. 234–251. – DOI: 10.28925/2663-4023.2023.21.234251.

47. Patil S. Explainable Artificial Intelligence for Intrusion Detection System / S. Patil, V. Varadarajan, S.M. Mazhar, A. Sahibzada, N. Ahmed, O. Sinha, S. Kumar, K. Shaw, K. Kotecha // Electronics. – 2022. – Т. 11, № 19. – С. 3079. – DOI: 10.3390/electronics11193079.

48. Ali S. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence / S. Ali, T. Abuhmed, Sh. El-Sappagh, K. Muhammad, J.M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, F. Herrera // Information Fusion. – 2023. – Т. 99. – С. 101805. – DOI: 10.1016/j.inffus.2023.101805.

49. Kundu P.P. Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation / P.P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, S.G. Teo // IEEE Transactions on Dependable and Secure Computing. – PrePrints 5555. – С. 1–15. – DOI: 10.1109/TDSC.2022.3183361.

50. Blaise A. Botnet Fingerprinting: a Frequency Distributions Scheme for Lightweight Bot Detection / A. Blaise, M. Bouet, V. Conan, S. Secci // IEEE Transactions on Network and Service Management. – 2020. – Т. 17, № 3. – С. 1701–1714. – DOI: 10.1109/TNSM.2020.2996502.

51. Kirubavathi G. Botnet Detection Based On Network Traffic Flow Statistical Features and Model Based Clustering / K. G, N. S. // Proceedings of the First International Conference on Combinatorial and Optimization (ICCAP). – 2021. – С. 1–6.

52. Ibrahim L. Analysis and Detection of the Zeus Botnet Crimeware / L. Ibrahim, K. Hatim // International Journal of Computer Science and Information Security. – 2015. – Т. 13. – С. 121–135.

53. Sood A.K. Dissecting SpyEye – Understanding the design of third generation botnets / A.K. Sood, R.J. Enbody, R. Bansal // Computer Networks. – 2013. – Т. 57, № 2. – С. 436–450. – DOI: 10.1016/j.comnet.2012.06.021.

54. Lopes D.A.G. Botnet detection based on network flow analysis using inverse statistics / D.A.G. Lopes, M.A. Marotta, M. Ladeira, J.J.C. Gondim // Proceedings of the 17th Iberian Conference on Information Systems and Technologies (CISTI). – Madrid, 2022. – С. 1–6. – DOI: 10.23919/CISTI54924.2022.9820318.

55. Internet Assigned Numbers Authority. [Електронний ресурс]. – Режим доступу: https://www.iana.org/assignments/ipfix/ipfix.xhtml (дата звернення: 02.03.2025).

56. Graham M. A Botnet Needle in a Virtual Haystack / M. Graham. – 2018.

57. ipfixprobe. [Електронний ресурс]. – Режим доступу: https://github.com/CESNET/ipfixprobe (дата звернення: 02.03.2025).

58. nfdump. [Електронний ресурс]. – Режим доступу: https://github.com/phaag/nfdump (дата звернення: 02.03.2025).

59. Hsu C.-H. Fast-Flux Bot Detection in Real Time / C.-H. Hsu, Ch.-Y. Huang, K.-T. Chen // Lecture Notes in Computer Science. – 2010. – Т. 6307. – С. 464–483. – DOI: 10.1007/978-3-642-15512-3_24.

60. Hwang C. Effective DGA-Domain Detection and Classification with TextCNN and Additional Features / C. Hwang, H. Kim, H. Lee, T. Lee // Electronics. – 2020. – Т. 9. – С. 1070. – DOI: 10.3390/electronics9071070.

61. Almutairi S. Hybrid Botnet Detection Based on Host and Network Analysis / S. Almutairi, S. Mahfoudh, S. Almutairi, J.S. Alowibdi. – 2019.

62. Alves P.A. HTTP and contact-based features for Botnet detection / P.A. Alves, A.C.D. Resende. – 2018.

63. Hastie T. The Elements of Statistical Learning / T. Hastie, R. Tibshirani, J. Friedman. – 2nd ed. – New York: Springer, 2008. – ISBN 0-387-95284-5.

64. Scapy. [Електронний ресурс]. – Режим доступу: https://scapy.net/ (дата звернення: 02.03.2025).

65. Apache Parquet. [Електронний ресурс]. – Режим доступу: https://parquet.apache.org/ (дата звернення: 02.03.2025).

66. NetWatcher. [Електронний ресурс]. – Режим доступу: https://github.com/NocturnalShadow/net-watcher (дата звернення: 02.03.2025).

67. Garcia S. An empirical comparison of botnet detection methods / S. Garcia, M. Grill, J. Stiborek, A. Zunino // Computers and Security. – 2014. – Т. 45. – С. 100–123. – DOI: 10.1016/j.cose.2014.05.011.

68. Malware Capture Facility Project. Malware Captures. [Електронний ресурс]. – Режим доступу: https://www.stratosphereips.org/datasets-malware (дата звернення: 02.03.2025).

69. Malware Capture Facility Project. Normal Captures. [Електронний ресурс]. – Режим доступу: https://www.stratosphereips.org/datasets-normal (дата звернення: 02.03.2025).

70. Canavan J. The evolution of malicious IRC bots / J. Canavan. – 2005.

71. Álvarez-Terribas F. A Deep Learning-Based Approach for Mimicking Network Topologies: The Neris Botnet as a Case of Study / F. Álvarez-Terribas, R. Magán-

Carrión, G. Maciá-Fernández, A.M. Mora García // Lecture Notes in Networks and Systems. – 2023. – Т. 532. – С. 218–229. – DOI: 10.1007/978-3-031-18409-3_19.

72. Microsoft Malware Encyclopedia. Win32/Rbot. [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32%2FRbot (дата звернення: 02.03.2025).

73. Polish Takedown Targets ‘Virut’ Botnet. [Електронний ресурс]. – 2013. – Режим доступу: https://krebsonsecurity.com/2013/01/polish-takedown-targets-virut-botnet/ (дата звернення: 02.03.2025).

74. Microsoft Malware Encyclopedia. Trojan:Win32/Donbot.A. [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan%3AWin32%2FDonbot.A (дата звернення: 02.03.2025).

75. Microsoft Malware Encyclopedia. TrojanDownloader:Win32/Murlo.S. [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Murlo.S (дата звернення: 02.03.2025).

76. DataDome. What is Sogou Spider? [Електронний ресурс]. – Режим доступу: https://datadome.co/bots/sogou-spider/ (дата звернення: 02.03.2025).

77. F-Secure. Trojan Kazy. – [Електронний ресурс]. – Режим доступу: https://www.f-secure.com/v-descs/trojan-downloader-w32-kazy17907.shtml (дата звернення: 02.03.2025).

78. Europol. World’s most dangerous malware EMOTET disrupted through global action. – 2021. – [Електронний ресурс]. – Режим доступу: https://www.europol.europa.eu/media-

press/newsroom/news/world%E2%80%99s-most-dangerous-malware-emotet-disrupted-through-global-action (дата звернення: 02.03.2025).

79. CISA. TrickBot Malware. – 2021. – [Електронний ресурс]. – Режим доступу: https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-076a (дата звернення: 02.03.2025).

80. Cloudflare. What was the WannaCry ransomware attack? – [Електронний ресурс]. – Режим доступу: https://www.cloudflare.com/learning/security/ransomware/wannacry-ransomware/ (дата звернення: 02.03.2025).

81. Szegedy C., Zaremba W., Sutskever I. et al. Intriguing properties of neural networks. – 2013. – Preprint. – DOI: https://doi.org/10.48550/arXiv.1312.6199.

82. Geman S., Bienenstock E., Doursat R. Neural Networks and the Bias/Variance Dilemma // Neural Computation. – 1992. – Vol. 4. – P. 1–58. – DOI: https://doi.org/10.1162/neco.1992.4.1.1.

83. Goodfellow I. J., Shlens J., Szegedy C. Explaining and Harnessing Adversarial Examples. – 2014. – Preprint. – DOI: https://doi.org/10.48550/arXiv.1412.6572.

84. Letteri I., Rosso M. D., Caianiello P., Cassioli D. Performance of Botnet Detection by Neural Networks in Software-Defined Networks // Italian Conference on Cybersecurity. – 2018. – URL: https://ceur-ws.org/Vol-2058/paper-03.pdf.

85. Zhang X., Zheng X., Wu D. Attacking DNN-based Intrusion Detection Models // IFAC-PapersOnLine. – 2020. – Vol. 53. – P. 415–419. – DOI: https://doi.org/10.1016/j.ifacol.2021.04.118.

86. Hu Y., Tian J., Ma J. A Novel Way to Generate Adversarial Network Traffic Samples against Network Traffic Classification // Wireless Communications and Mobile Computing. – 2021. – P. 1–12. – DOI: https://dx.doi.org/10.1155/2021/7367107.

87. Zolbayar B., Sheatsley R., McDaniel P. et al. Generating Practical Adversarial Network Traffic Flows Using NIDSGAN. – 2022. – Preprint. – DOI: https://doi.org/10.48550/arXiv.2203.06694.

88. Randhawa R., Aslam N., Alauthman M., Khalid M., Rafiq H. Deep Reinforcement Learning based Evasion Generative Adversarial Network for Botnet Detection. – 2022. – Preprint. – DOI: https://doi.org/10.48550/arXiv.2210.02840.

89. Rumelhart D. E., Hinton G. E., Williams R. J. Learning representations by back-propagating errors // Nature. – 1986. – Vol. 323. – P. 533–536. – DOI: https://doi.org/10.1038/323533a0.

90. Parr T., Howard J. The Matrix Calculus You Need For Deep Learning. – 2018. – Preprint. – DOI: https://doi.org/10.48550/arXiv.1802.01528.

91. AsSadhan B., Bashaiwth A., Al-Muhtadi J., Alshebeili S. Analysis of P2P, IRC and HTTP traffic for botnets detection // Peer-to-Peer Networking and Applications. – 2018. – DOI: https://link.springer.com/article/10.1007/s12083-017-0586-0.

92. Гордєєв О. О. Модель якості окремої вимоги програмного забезпечення / О. О. Гордєєв // Радіоелектронні і комп'ютерні системи. – 2020. – № 2(94). – С. 48–58. – DOI: 10.32620/reks.2020.2.04.

93. Gordieiev O. A Unified Approach to the Development of Technology-Based Software Quality Models on the Example of Blockchain Systems / O. Gordieiev, A. Rainer, V. Kharchenko, O. Pishchukhina, D. Gordieieva // IEEE Access. – 2024. – Vol. 12. – C. 118875–118889. – DOI: 10.1109/ACCESS.2024.3448271.

94. Casadio M., et al. Neural Network Robustness as a Verification Property: A Principled Case Study // У: Shoham S., Vizel Y. (ред.) Computer Aided Verification. CAV 2022. – Lecture Notes in Computer Science, Vol. 13371. – Cham: Springer, 2022. – DOI: https://doi.org/10.1007/978-3-031-13185-1_11.

95. Zhang J.M. Machine Learning Testing: Survey, Landscapes and Horizons / J.M. Zhang, M. Harman, L. Ma, Y. Liu // IEEE Transactions on Software Engineering. – 2022. – Т. 48, № 1. – С. 1–36. – DOI: 10.1109/TSE.2019.2962027.

96. Odena A., Olsson C., Andersen D., Goodfellow I. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing // Proceedings of the 36th International Conference on Machine Learning (ICML). – Proceedings of Machine Learning Research, Vol. 97. – 2019. – С. 4901–4911. – DOI: https://doi.org/10.48550/arXiv.1807.10875.

97. Yang Z., Shi J., Asyrofi M., Lo D. Revisiting Neuron Coverage Metrics and Quality of Deep Neural Networks // 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). – Honolulu, HI, USA: IEEE, 2022. – С. 408–419. – DOI: https://doi.org/10.1109/SANER53432.2022.00056.

98. Yuan X. Adversarial Examples: Attacks and Defenses for Deep Learning / X. Yuan, P. He, Q. Zhu, X. Li // IEEE Transactions on Neural Networks and Learning Systems. – 2019. – Т. 30, № 9. – С. 2805–2824. – DOI: 10.1109/TNNLS.2018.2886017.

99. Liu C. Algorithms for Verifying Deep Neural Networks / C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M.J. Kochenderfer // Foundations and Trends in Optimization. – 2021. – Т. 4, № 3–4. – С. 244–404. – DOI: 10.1561/2400000035.

100. Katz G., Barrett C., Dill D. L., Julian K., Kochenderfer M. J. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks // У: Majumdar R., Kunčak V. (ред.) Computer Aided Verification. CAV 2017. – Lecture Notes in Computer Science, Vol. 10426. – Cham: Springer, 2017. – DOI: https://doi.org/10.1007/978-3-319-63387-9_5.

101. Katz G., et al. The Marabou Framework for Verification and Analysis of Deep Neural Networks // У: Dillig I., Tasiran S. (ред.) Computer Aided

Verification. CAV 2019. – Lecture Notes in Computer Science, Vol. 11561. – Cham: Springer, 2019. – DOI: https://doi.org/10.1007/978-3-030-25540-4_26.

102. Singh G. An Abstract Domain for Certifying Neural Networks / G. Singh, T. Gehr, M. Püschel, M. Vechev // Proceedings of the ACM on Programming Languages. – 2019. – Т. 3, № POPL. – Стаття 41. – С. 1–30. – DOI: 10.1145/3290354.

103. Elboher Y. Y., Gottschlich J., Katz G. An Abstraction-Based Framework for Neural Network Verification // У: Lahiri S., Wang C. (ред.) Computer Aided Verification. CAV 2020. – Lecture Notes in Computer Science, Vol. 12224. – Cham: Springer, 2020. – DOI: https://doi.org/10.1007/978-3-030-53288-8_3.

104. Guidotti D. Verifying Neural Networks with Non-Linear SMT Solvers: a Short Status Report / D. Guidotti, L. Pandolfo, L. Pulina // IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI). – 2023. – С. 423–428. – DOI: 10.1109/ICTAI59109.2023.00068.

105. de Moura L., Bjørner N. Z3: An Efficient SMT Solver // У: Ramakrishnan C. R., Rehof J. (ред.) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2008. – Lecture Notes in Computer Science, Vol. 4963. – Berlin, Heidelberg: Springer, 2008. – DOI: https://doi.org/10.1007/978-3-540-78800-3_24.

106. Sapphire. [Електронний ресурс]. – Режим доступу: https://github.com/wenkokke/sapphire (дата звернення: 01.04.2024).

ДОДАТКИ

Додаток А

Список публікацій здобувача

1. Панчук Б. Виявлення ботнет-трафіку на основі потоків, використовуючи ШІ // Проблеми програмування. – 2022. – № 3–4 (Спеціальний випуск). – С. 376–386. – DOI: 10.15407/pp2022.03-04.376.
2. Панчук Б. О. Генерація та використання змагальної вибірки для протидії ухиленню ботнетів від виявлення нейронними мережами (до 100-річчя з дня народження академіка В.М. Глушкова) // Проблеми кібербезпеки та інформаційних технологій. – 2023. – Т. 68, № 5. – С. 71–85. – DOI: 10.34229/1028-0979-2023-5-6.
3. Панчук Б. Формальна верифікація нейронних мереж глибокого навчання // Проблеми програмування. – 2024. – № 2–3. – С. 253–262. – DOI: 10.15407/pp2024.02-03.253.
4. Летичевський О., Панчук Б. Проблема точності в системах протидії кібератакам та верифікація нейронних мереж на прикладі задачі виявлення ботнетів // Кібербезпека та інформаційні технології. – 2025. – С. 3–12. – DOI: 10.34229/KCA2522-9664.25.2.1.

Апробація матеріалів дослідження
на конференціях, семінарах, круглих столах тощо

Рівень заходу та його назва	**Місце і дата проведення**	**Форма участі**
13th International Scientific and Practical Programming Conference - UkrPROG'2022	Жовтень 2022р., Київ	Очна, 1 публікація в матеріалах конференції

Додаток Б

ЗАТВЕРДЖУЮ
Головний операційний директор
ТОВ «НВП «Радікс»
_____Олександр ІВАСЮК
«__»_____ 2025 р.

АКТ ВПРОВАДЖЕННЯ
наукових результатів дисертаційної роботи Богдана ПАНЧУКА, виконаної на здобуття наукового ступеню доктора філософії

Комісія у складі голови комісії – Головного операційного директора ТОВ «НВП «Радікс» к.т.н., доцента Олександра ІВАСЮКА, членів комісії – провідного наукового співробітника д.т.н., професора Олега ОДАРУЩЕНКА, старшого наукового співробітника к.т.н., доцента Олексія СТРЮКА встановила, що програмна система створена на основі досліджень, які є науковим результатом дисертаційної роботи Панчука Богдана Олександровича, виконаної на здобуття наукового ступеню доктора філософії, в рамках досліджень відділу теорії цифрових автоматів Інституту кібернетики ім. В.М.Глушкова НАН України за науковою темою ВП100.17 «Розробити формальні методи виявлення зловмисної поведінки в мережі та хмарному оточенні на основі комбінації алгебраїчних методів та машинного навчання» може бути використана з метою додаткового захисту від кібератак, а саме для виявлення активностей ботнетів, зловмисного програмного забезпечення, що є джерелом DDoS атак («Відмова від обслуговування»).

Програмну систему створено на основі нейронної мережі глибокого навчання та із застосування розширеного згенерованого набору даних.

Використання запропонованої програмної системи дозволить підвищити ступінь захисту локальної мережі від зловмисного програмного забезпечення та зменшити ризики вторгнення.

Голова комісії Олександр ІВАСЮК
Члени комісії Олег ОДАРУЩЕНКО
Олексій СТРЮК

14 01 2025 р.

Додаток В

Нижче наведено список з 50 ознак, отриманими інструментом CICFlowMeter [21], яким було представлено мережеві потоки, що використовувалися в експериментальних частинах даної роботи по розширенню даних та формальної верифікації.

Номери	**Оригінальні назва ознак**	**Тип даних**	**Опис**
1	Flow Duration	int	Тривалість потоку
2-3	Total Fwd/Bwd Packet	int	Кількість пакетів у потоці
4-5	Total Length of Fwd/Bwd Packet	int	Сумарна довжина пакетів у потоці
6-11	Fwd/Bwd Packet Length Min/Max/Std	float	Мінімальна/Максимальна/Середньоквадратичне відхилення довжина пакета
12-19	Fwd/Bwd IAT Total/Min/Max/Std	float	Сумарне/Мінімальний/Максимальний/Середньоквадратичне відхилення інтервалу між пакетами у потоці
20-23	Fwd/Bwd PSH/URG Flags	int	Кількість пакетів з встановленими TCP прапорцями PSH/URG
24-25	Fwd/Bwd Header Length	int	Сумарна довжина заголовків пакетів у потоці
26-33	FIN/SYN/RST/PSH/ACK/ URG/CWR/ECE Flag Count	int	Кількість пакетів з встановленими TCP прапорцями FIN/SYN/RST/PSH/ACK/URG/CWR/ECE
34	Down/Up Ratio	float	Відношення кількості завантажень до вивантажень
35-36	Fwd/Bwd Segment Size Avg	float	Середній розмір сегменту у потоці
37-40	Fwd/Bwd (Bytes/Packet)/Bulk Avg	int	Середня кількість байтів/пакетів в кластері пакетів

41-42	Fwd/Bwd Bulk Rate Avg	int	Середній кількість байт на секунду в кластерах пакетів потоку
43-46	Subflow Fwd/Bwd Packets/Bytes	int	Кількість пакетів/байтів у підпотоці
47-48	Fwd/Bwd Init Win Bytes	int	Кількість байтів передана в початковому вікні
49	Fwd Act Data Pkts	int	Кількість пакетів з хоча б 1 байтом даних преданих в напрямку від джерела до приймача
50	Fwd Seg Size Min	int	Мінімальна довжина сегменту у потоці

Додаток Г

Нижче наведено список атрибутів, які виділяє NetWatcher для кожного реконструйованого потоку.

1. **Метадані потоку.** Тип даних «string».

Назва атрибута	**Опис**
id	Ідентифікатор потоку у формі: `<src_ip>-<dst_ip>-<src_port>-<dst_port>-<protocol>`
timestamp	Часова мітка (у форматі Unix), що відображає час створення потоку
src_ip	IP-адреса джерела (ініціатора) потоку
dst_ip	IP-адреса призначення потоку
src_port	Номер порту джерела
dst_port	Номер порту призначення
protocol	Протокол транспортного рівня (наприклад, TCP чи UDP)
termination_reason	Причина завершення потоку (FIN, RST, idle_timeout, activity_timeout)

2. **Характеристичні ознаки потоку.** Всі атрибути мають тип даних «float».

   В полі «Напрямленість»:

   i. «→» символізує напрямок від ініціатора потоку до пункту призначення.

   ii. «←» символізує напрямок від пункту призначення до ініціатора.

   iii. «—» значить, що атрибут обчислюється сумарно для пакетів в обох напрямках.

Назва атрибута	**Опис**	**Напрям-леність**
duration_s	Тривалість потоку в секундах (час між першим і останнім пакетом)	—

packets_count	Загальна кількість пакетів у потоці	—
payload_bytes_seq	Послідовність байтів корисного навантаження для кожного пакета	—
payload_bytes_total	Загальна кількість байтів корисного навантаження (без заголовків) у потоці	—
payload_bytes_min	Мінімальна кількість байтів корисного навантаження у потоці	—
payload_bytes_max	Максимальна кількість байтів корисного навантаження у потоці	—
payload_bytes_std	Стандартне відхилення байтів корисного навантаження у потоці	—
payload_bytes_min_nonzero	Мінімальна кількість байтів корисного навантаження у потоці (серед пакетів з ненульовим корисним навантаженням)	—
payload_bytes_avg_nonzero	Середня кількість байтів корисного навантаження у потоці (серед пакетів з ненульовим корисним навантаженням)	—
payload_bytes_std_nonzero	Стандартне відхилення байтів корисного навантаження у потоці (серед пакетів з ненульовим корисним навантаженням)	—
interarrival_time_s_seq	Послідовність інтервалів між надходженням пакетів	—
interarrival_time_s_min	Мінімальний інтервал між надходженням пакетів	—
interarrival_time_s_max	Максимальний інтервал між надходженням пакетів	—
interarrival_time_s_std	Стандартне відхилення інтервалів між надходженням пакетів	—
syn_count	Кількість пакетів із встановленим прапорцем SYN	—
fin_count	Кількість пакетів із встановленим прапорцем FIN	—
rst_count	Кількість пакетів із встановленим прапорцем RST	—
ack_count	Кількість пакетів із встановленим прапорцем ACK	—

psh_count	Кількість пакетів із встановленим прапорцем PSH	—
urg_count	Кількість пакетів із встановленим прапорцем URG	—
ece_count	Кількість пакетів із встановленим прапорцем ECE	—
cwr_count	Кількість пакетів із встановленим прапорцем CWR	—
fwd_window_size_seq	Послідовність розмірів TCP вікна	→
fwd_window_size_min	Мінімальний розмір TCP вікна	→
fwd_window_size_max	Максимальний розмір TCP вікна	→
fwd_window_size_avg	Середній розмір TCP вікна	→
fwd_window_size_std	Стандартне відхилення розмірів TCP вікна	→
fwd_window_scaling _factor	Фактор масштабування TCP вікна	→
fwd_initial_window_size	Початковий розмір TCP вікна	→
fwd_zero_window_count	Кількість випадків нульового розміру TCP вікна	→
fwd_zero_window_update _count	Кількість оновлень нульового розміру TCP вікна	→
bwd_window_size_seq	Послідовність розмірів вікна	←
bwd_window_size_min	Мінімальний розмір вікна	←
bwd_window_size_max	Максимальний розмір вікна	←
bwd_window_size_avg	Середній розмір вікна	←
bwd_window_size_std	Стандартне відхилення розмірів вікна	←
bwd_window_scaling _factor	Фактор масштабування вікна	←
bwd_initial_window_size	Початковий розмір вікна	←
bwd_zero_window_count	Кількість випадків нульового розміру вікна	←
bwd_zero_window_update _count	Кількість оновлень нульового розміру вікна	←

Додаток Д

Нижче наведена таблиця показників повноти виявлення моделей класифікації навчених на ISCX-CTU-Extended для різних типів трафіку шкідливих прогам.

Модель: iscx-ctu-extended/pca_12_rf_9

Вибірка: iscx-ctu-extended (testing)

Threshold	**DonBot**	**Emotet**	**IRC**	**Kazy**	**Menti**	**Murlo**	**Neris**
0.51	0.931	0.904	1.000	0.410	0.931	0.283	0.514
0.60	0.930	0.904	1.000	0.339	0.930	0.261	0.486
0.70	0.920	0.902	0.996	0.293	0.920	0.246	0.468
0.80	0.905	0.894	0.948	0.214	0.905	0.105	0.450
0.85	0.905	0.860	0.840	0.137	0.905	0.103	0.427
0.90	0.905	0.841	0.648	0.133	0.905	0.095	0.420
0.95	0.278	0.823	0.220	0.096	0.278	0.093	0.322
0.97	0.262	0.807	0.000	0.050	0.262	0.034	0.318

Threshold	**NSIS .ay**	**RBot**	**Sogou**	**TrickBot**	**Virut**	**WannaCry**	**Weasel**	**Zeus**
0.51	0.119	0.994	0.350	0.998	0.926	0.957	0.153	0.526
0.60	0.065	0.993	0.250	0.997	0.913	0.955	0.152	0.523
0.70	0.051	0.989	0.100	0.797	0.908	0.954	0.016	0.523
0.80	0.049	0.972	0.100	0.419	0.898	0.951	0.013	0.283
0.85	0.049	0.970	0.100	0.153	0.890	0.937	0.013	0.044
0.90	0.016	0.965	0.075	0.129	0.887	0.911	0.010	0.044
0.95	0.011	0.940	0.000	0.083	0.871	0.704	0.000	0.043
0.97	0.008	0.922	0.000	0.069	0.865	0.664	0.000	0.042

Модель: iscx-ctu-extended/dnn_24_24_24

Вибірка: iscx-ctu-extended (testing)

Threshold	DonBot	Emotet	IRC	Kazy	Menti	Murlo	Neris
0.51	0.999	0.986	0.968	0.687	0.999	0.998	0.874
0.60	0.999	0.986	0.896	0.668	0.999	0.998	0.856
0.70	0.999	0.986	0.896	0.660	0.999	0.998	0.836
0.80	0.998	0.903	0.876	0.635	0.998	0.975	0.802
0.85	0.998	0.903	0.872	0.628	0.998	0.878	0.790
0.90	0.997	0.903	0.840	0.618	0.997	0.711	0.772
0.95	0.997	0.901	0.804	0.541	0.997	0.309	0.533
0.97	0.969	0.901	0.788	0.525	0.969	0.304	0.515

Threshold	NSIS.ay	RBot	Sogou	TrickBot	Virut	WannaCry	Weasel	Zeus
0.51	0.792	0.999	0.850	1.000	0.974	0.795	0.625	0.047
0.60	0.757	0.997	0.775	1.000	0.968	0.795	0.617	0.047
0.70	0.651	0.997	0.600	1.000	0.962	0.718	0.610	0.045
0.80	0.514	0.993	0.500	1.000	0.954	0.712	0.599	0.044
0.85	0.486	0.989	0.475	0.998	0.950	0.650	0.590	0.044
0.90	0.378	0.927	0.275	0.998	0.941	0.635	0.587	0.044
0.95	0.295	0.926	0.225	0.998	0.924	0.634	0.586	0.044
0.97	0.184	0.915	0.175	0.998	0.915	0.633	0.586	0.044