Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» травня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Модель виявлення шахрайських доменних імен»

Виконавець: студентка ІІ курсу, групи КБм-21

_____ **Вікторія ШМАТКО**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій БУЧИК	
Нормоконтроль	Інна МИХАЛЬЧУК	

Київ 2024

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» листопада 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Шматко Вікторії Олександрівни (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Модель виявлення шахрайських доменних імен

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 15.11.2023 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес виявлення зловмисних доменних імен, які згенеровані алгоритмами.
Предмет досліджень	Архітектури машинного навчання, застосовні для встановлення легітимності доменних імен.
Мета	Розроблення гібридної моделі машинного навчання для виявлення доменів, що згенеровані алгоритмами, ціллю яких є скомпрометувати інформаційну систему або ввести в оману користувача.

Вихідні дані для проведення роботи	Статистичні дані. Архітектури Long Short-Term Memory, Convolutional Neural Network та механізму самоуваги. Можливості мови програмування Python. Легітимні і шахрайські домени

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	Отримано подальший розвиток у застосуванні машинного навчання для виявлення шахрайських доменних імен шляхом інтеграції існуючих архітектур, який гарантуватиме надійні результати під час ідентифікації зловмисних доменів, згенерованих алгоритмами.
Практична цінність	Здобуті результати є рекомендованими для впровадження до систем безпеки в компаніях та організаціях різних масштабів, оскільки розроблена модель посилює захист та забезпечує високий рівень кібербезпеки у відповідному середовищі.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	17.11.2023 – 30.11.2023
Аналіз літературних джерел й існуючих розробок	01.12.2023 – 23.12.2023
Роз’яснення типів шахрайських доменів і загроз, які вони несуть	24.12.2023 – 10.01.2024
Деталізація небезпеки доменів, які згенеровані алгоритмами, на основі статистики за попередні роки	11.01.2024 – 25.01.2024
Обґрунтування потенціалу машинного навчання, вивчення архітектур глибокого навчання на предмет їхньої інтеграції	26.01.2024 – 02.03.2024
Розробка моделі машинного навчання, опис її архітектури	03.03.2024 – 19.04.2024
Оцінка отриманих результатів	20.04.2024 – 03.05.2024
Оформлення пояснювальної записки згідно з методичними рекомендаціями	04.05.2024 – 12.05.2024
Подача пакету документів на розгляд ЕК	13.05.2024 – 17.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через реалізацію кібератак та усунення витрат на дороговартісні системи захисту.

Соціальний ефект Підвищення захищеності користувачів від кібератак.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Сергій БУЧИК
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Вікторія ШМАТКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.

Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка: кваліфікаційна робота складається із вступу, 3 розділів, висновків, списку використаних джерел і 7 додатків. Основний текст займає 100 сторінок і містить 31 рисунок, 16 формул, 3 таблиці та 52 літературних джерела.

Метою роботи є розроблення гібридної моделі машинного навчання для виявлення доменів, що згенеровані алгоритмами, ціллю яких є скомпрометувати інформаційну систему або ввести в оману користувача.

Для втілення вказаної мети поставлені такі завдання:

1) охарактеризувати кібератаки з використанням шахрайських доменів;

2) дослідити підходи до запобігання реалізації кіберзагроз із застосуванням доменних імен;

3) з’ясувати перспективи впровадження машинного навчання для детектування відповідних кібератак;

4) описати метрики оцінки моделей та їхню залежність від наборів даних;

5) визначити архітектури машинного навчання, які застосовні для ідентифікації шахрайських доменів;

6) інтегрувати архітектури машинного навчання для створення гібридної моделі, провести аналіз отриманих результатів.

Об’єктом дослідження є процес виявлення зловмисних доменних імен, які згенеровані алгоритмами.

Предметом дослідження є архітектури машинного навчання, застосовні для встановлення легітимності доменних імен.

Методи дослідження: системний аналіз, ізолювання, узагальнення, абстрагування, моделювання, інтеграція.

Актуальність роботи пояснюється зростаючою загрозою ботнетів, які використовують алгоритми генерації доменних імен для уникнення виявлення та блокування, ускладнюючи їхню ліквідацію. Сучасні методи ідентифікації даних ресурсів вимагають глибоких знань та високої експертизи через необхідність ручного

визначення характеристик доменів, що зменшує їхню ефективність при змінах в алгоритмах зловмисників. Це підкреслює потребу у розробці нової гібридної моделі машинного навчання, здатної ефективно виявляти домени даного типу, адаптуючись до їхніх постійних змін.

Наукова новизна роботи – отримано подальший розвиток у застосуванні машинного навчання для виявлення шахрайських доменних імен шляхом інтеграції існуючих архітектур, який гарантуватиме надійні результати під час ідентифікації зловмисних доменів, згенерованих алгоритмами. Даний підхід сприятиме підвищенню рівня кібербезпеки завдяки оперативному розпізнаванню потенційно небезпечних доменів і попереджатиме кібератаки на користувачів.

Здобуті теоретичні та практичні результати розкривають *практичну цінність* кваліфікаційної роботи і є рекомендованими для впровадження до систем безпеки в компаніях та організаціях різних масштабів, оскільки розроблена модель посилює захист та забезпечує високий рівень кібербезпеки у відповідному середовищі.

Ключові слова: машинне навчання, Domain Generation Algorithm, Convolution Neural Network, Long Short-Term Memory, механізм уваги, ансамбль, гібридна модель, метрики оцінки.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ASCII	–	American Standard Code for Information Interchange
BiLSTM	–	Bidirectional Long Short-Term Memory
C&C	–	Command and Control
CNN	–	Convolutional Neural Network
DDoS	–	Distributed Denial-of-Service
DGA	–	Domain Generation Algorithms
DL	–	Damerau-Levenshtein
DNS	–	Domain Name System
ELU	–	Exponential Linear Unit
FN	–	False Negative
FP	–	False Positive
GPU	–	Graphics Processing Unit
ICANN	–	Internet Corporation for Assigned Names and Numbers
IDN	–	Internationalized Domain Names
IDNA	–	Internationalized Domain Names in Applications
IDS	–	Intrusion Detection System
LSTM	–	Long Short-Term Memory
ML	–	Machine Learning
ReLU	–	Rectified Linear Unit
RNN	–	Recurrent Neural Networks
SLD	–	Second Level Domain
TLD	–	Top Level Domains
TN	–	True Negative
TP	–	True Positive
ШІ	–	штучний інтелект
ШПЗ	–	шкідливе програмне забезпечення

ЗМІСТ

ВСТУП

Актуальність роботи обумовлена наявними та потенційними ризиками, що виникають внаслідок розповсюдження та зростання кількості ботнетів у кіберпросторі. Спочатку віддалений доступ від ботмайстра до інфраструктури ботнету залежав від заздалегідь встановлених IP-адрес або доменів. Такий зв'язок було легко блокувати або перешкоджати комунікації між ботами та сервером команд та керування. З метою уникнення виявлення наразі ботнети використовують алгоритми генерації доменних імен, Domain Generation Algorithms (DGA), для формування шкідливих доменних імен та забезпечення комунікації між зараженими ботами та сервером команд та керування.

Впроваджуючи DGA, місцезнаходження ботнету та його серверів є важкими для виявлення та ліквідації. Сучасні традиційні методи детектування на основі машинного навчання вимагають попередньо розроблених ознак доменних імен, що потребує більше зусиль, високої експертизи та знань. Якщо алгоритм, що міститься у зловмисному ботнеті, змінюється, то складання рядка доменного імені також варіюється. Відповідно моделі, які використовують вручну створені ознаки, втрачають високу точність. Тому постає необхідність у розробці гібридної моделі машинного навчання, яка буде ефективною у виявленні доменів DGA.

Таким чином, *наукова новизна* роботи – отримано подальший розвиток у застосуванні машинного навчання для виявлення шахрайських доменних імен шляхом інтеграції існуючих архітектур, який гарантуватиме надійні результати під час ідентифікації зловмисних доменів, згенерованих алгоритмами. Даний підхід сприятиме підвищенню рівня кібербезпеки завдяки оперативному розпізнаванню потенційно небезпечних доменів і попереджатиме кібератаки на користувачів.

Об'єктом дослідження є процес виявлення зловмисних доменних імен, які згенеровані алгоритмами.

Предметом дослідження є архітектури машинного навчання, застосовні для встановлення легітимності доменних імен.

Метою роботи є розроблення гібридної моделі машинного навчання для виявлення доменів, що згенеровані алгоритмами, ціллю яких є скомпрометувати інформаційну систему або ввести в оману користувача.

Для втілення вказаної мети поставлені такі завдання:

1) охарактеризувати кібератаки з використанням шахрайських доменів;

2) дослідити підходи до запобігання реалізації кіберзагроз із застосуванням доменних імен;

3) з’ясувати перспективи впровадження машинного навчання для детектування відповідних кібератак;

4) описати метрики оцінки моделей та їхню залежність від наборів даних;

5) визначити архітектури машинного навчання, які застосовні для ідентифікації шахрайських доменів;

6) інтегрувати архітектури машинного навчання для створення гібридної моделі, провести аналіз отриманих результатів.

Методи дослідження: системний аналіз, ізолювання, узагальнення, абстрагування, моделювання, інтеграція.

Здобуті теоретичні та практичні результати розкривають *практичну цінність* кваліфікаційної роботи і є рекомендованими для впровадження до систем безпеки в компаніях та організаціях різних масштабів, оскільки розроблена модель посилює захист та забезпечує високий рівень кібербезпеки у відповідному середовищі.

Апробація результатів роботи: відомості, внесені до кваліфікаційної роботи, були опрацьовані протягом VII Міжнародної науково-практичної конференції «Прикладні системи та технології в інформаційному суспільстві» з темою «Виклики та перспективи використання машинного навчання у запобіганні доменним атакам»; X International Conference «Information Technology and Implementation» (IT&I-2023) Satellite з темою «Strategies for dataset collection in domain attack prevention with machine learning»; VII Міжнародної науково-практичної конференції «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (PCSITS) з темою «Використання метрик оцінки алгоритмів для моделей виявлення шахрайських доменів».

РОЗДІЛ 1
ДОМЕННІ ІМЕНА ЯК КРИТИЧНА СКЛАДОВА БЕЗПЕКИ ЕЛЕКТРОННОЇ ВЗАЄМОДІЇ

1.1 Функціональність доменних імен в онлайн-середовищі

Шахрайські доменні імена є одними з основних ресурсів для здійснення атак в Інтернеті. Вони як невід'ємна складова шкідливих URL-адрес є поширеною та катастрофічною загрозою для кібербезпеки. Такі домени можуть використовуватися з метою заманювання користувачів, аби зробити їх жертвами фішингу, «drive-by» завантаження, розсилки спаму та іншого контенту. Дані ситуації призводять до порушення приватності користувачів, фінансових втрат або установки шкідливого програмного забезпечення на їхні пристрої [1].

Як представлено на рис. 1.1, домен та піддоменне ім'я є важливими частинами URL, оскільки їхня комбінація разом формує назву хоста та визначає фізичну машину, на якій розташований ресурс.

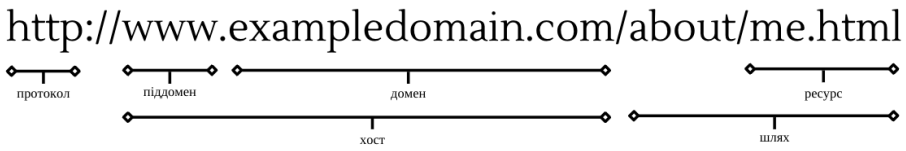

Рисунок 1.1 – Структура URL

Система доменних імен, Domain Name System (DNS) – це один з основних наборів протоколів Інтернету, який бере на себе відповідальність направляти запити до ресурсів Інтернету до конкретної хост-машини, а ці ресурси ідентифікуються за допомогою URL, що включають доменні імена. DNS є децентралізованою, розподіленою, ієрархічною системою іменування для ресурсів, підключених до Інтернету, яка перетворює читабельні для людей та прийнятних для запам'ятовування доменні імена на їхні відповідні IP-адреси, і іноді навпаки. Дані домену в DNS

організовані у структурі оберненого дерева з коренем у верхній частині, представленого «.», крапкою. Міжнародна корпорація з призначення імен та номерів в Інтернеті, Internet Corporation for Assigned Names and Numbers (ICANN) є інтернаціональним органом, якому надана влада щодо адміністрування кореня. Під ним розташовані домени верхнього рівня, Top Level Domains (TLD), до яких входять коди верхнього рівня, country code Top-Level Domains, неспонсоровані домени верхнього рівня, загальні домени верхнього рівня, generic Top-Level Domains [2].

Розподілена система DNS складається з 13 кореневих серверів із позначенням від A до M, що розподілені по всьому світу з реплікованою кореневою зоною. Дані DNS розділені на зони, при цьому кожну зону обслуговує набір авторитетних серверів імен, які несуть відповідальність за надання авторитетних відповідей для даних, що присутні в їхніх відповідних зонах. Ще один тип сервера імен, відомий як резольвер, надає неавторитетні відповіді клієнтам [3].

Резольвери починають свою роботу на кореневих серверах і слідкують за делегаціями зон, обробляючи авторитетні відповіді, отримані на кожному рівні, до того часу, поки вони не досягнуть призначеного авторитетного сервера імен для конкретної зони, як це представлено на рис. 1.2.

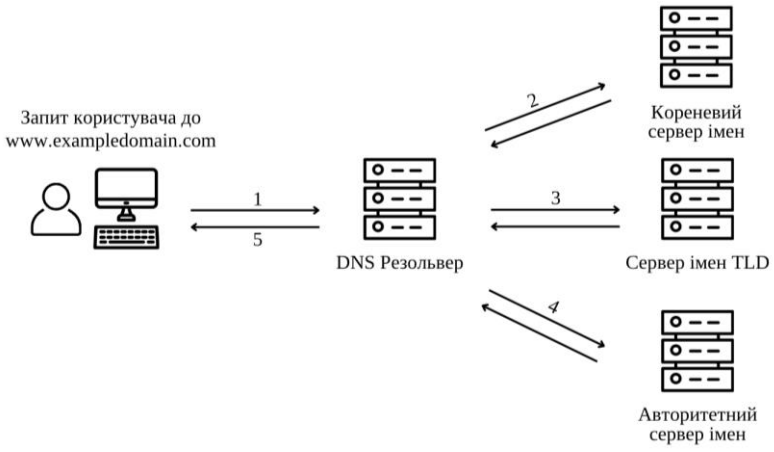

Рисунок 1.2 – Робота DNS

Дані в системі DNS мають високий рівень динамічності, а небажані записи систематично проникають в інфраструктуру доменних імен через низку технічних та

адміністративних вразливостей. Це відкриває можливості для експлуатації системи в нелегітимних цілях.

1.2 Характеристика кібератак з використанням шахрайських доменів

У сфері кібербезпеки відбувається постійне ускладнення аспектів формування шахрайських доменів, віддзеркалюючи невпинний технологічний прогрес зловмисників. Таке вдосконалення навичок є важливим аспектом дослідження, оскільки демонструє високий рівень адаптивності та інженерної винахідливості кіберзагроз. Скрупульозне розуміння даних технічних тенденцій стає необхідним для розробки високоефективних стратегій виявлення шахрайських ресурсів в умовах постійної еволюції кіберзагроз. У зв’язку з цим розглянемо найбільш високорівневі техніки створення доменів, що стають початковою ланкою для запуску кібератаки.

1.2.1 Інтернаціоналізовані доменні імена

Інтернаціоналізовані доменні імена, Internationalized Domain Names (IDN), розширили границі доступу до Інтернету, дозволяючи використовувати символи різних мов у доменних іменах. Проте ця технологія також відкрила можливості для хакерів та кіберзлочинців у проведенні атак, реалізовуючи витончені методи, що базуються на міжнародних символах та їхньої подібності.

На момент розповсюдження онлайн-взаємодії доменні імена могли включати лише літери латинського алфавіту, цифри та дефіси, оскільки Інтернет був загалом розроблений для англійської мови. Однак з поширеним його використанням в країнах, де розмовляють не тільки англійською, і через те, що для веб-сайтів та електронних листів можна використовувати кілька мов, зросли вимоги до використання декількох мов у доменних іменах. Щоб вирішити цю проблему, були створені IDN для можливості використання в доменних іменах символів, які не є частиною Американського стандартного коду для інформаційного обміну American Standard Code for Information Interchange (ASCII) [4].

На початку 2000-х стало можливим також використовувати не-ASCII символи у доменних іменах верхнього рівня, відомих як IDN TLDs. IDN реалізовані таким чином, що вони залишаються сумісними із існуючими доменними іменами та їхніми серверами. Зокрема, існує механізм під назвою Міжнародні доменні імена у застосунках, Internationalized Domain Names in Applications (IDNA), для перетворення IDN в ASCII-рядки. Цей процес визначено в стандарті RFC 3490 [4]. На рис. 1.3 представлено приклад перетворення японського IDN в ASCII-рядок за допомогою IDNA.

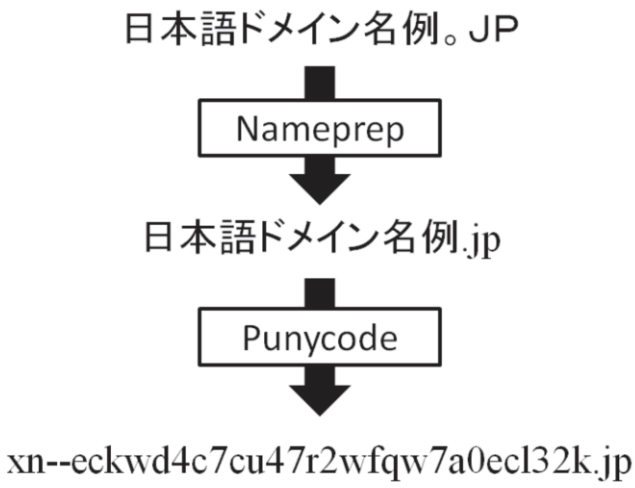

Рисунок 1.3 – Приклад перетворення IDNA

Загалом IDNA включає два процеси – Nameprep та Punycode. Nameprep, який визначено в документі RFC 3491, проводить попередню обробку доменних імен для усунення варіацій запису і ґрунтується на профілі Stringprep, описаному в RFC 3454. Він передбачає три етапи, а саме: відображення, нормалізація та обробка заборонених символів. Спочатку перший етап уніфікує типи символів та перетворює всі прописні літери у строкові. Потім нормалізація обробляє будь-які спеціальні символи, перетворюючи символи половинної ширини у, відповідно, символи повної ширини, роз’єднуючи композитні символи та перетворюючи верхні і нижні індекси у звичайні символи. Завершальний етап заборонених символів повертає помилку, якщо доменне ім’я містить заборонені символи (наприклад, пробіл чи управляючі символи).

Punycode – це синтаксис кодування, затверджений у стандартні RFC 3492. Кодування є оборотним, і перекладені рядки передуються префіксом «xn–» для відрізнення їх від звичайних доменів. Даний процес виглядає наступним чином [4]:

1) видалити всі символи не з ASCII з імені домену і додати «-» у кінці залишеного рядка;

2) розрахувати числовий код символу та вихідне положення для кожного видаленого символу не з ASCII і замінити їх відповідними рядками;

3) додати конвертовані рядки в кінець вихідного рядка.

Сьогодні більшість сучасних веб-браузерів виявляють IDN в URL-адресах, які вводять користувачі, і автоматично перетворюють їх на Punycode. Наприклад, IDN «facebook[.]com», де «а» та «с» взято з кирилиці, перетворюється на його представлення Punycode «xn--febook-3nf0l[.]com» у веб-браузері.

Розглянувши фактичний механізм утворення назви доменного імені, обґрунтуємо, яку має роль описаний клас в здійсненні кібератаки. Безперечно, в Інтернеті доменні імена використовуються багатьма алфавітами – від загальновживаних латинських літер до грецьких, китайських і кириличних. Хоча ця різноманітність є фантастичною для глобальної доступності, вона, на жаль, також створює можливості для шахраїв. Поняття «омограф» відноситься до візуально схожих символів з різних сценаріїв. Наприклад, латинський символ «а» (код U+0061) виглядає ідентично кириличному символу «а» (код U+0430). Це різні символи в обчислювальній термінології (U+0061 проти U+0430) [5].

Саме в змішаній омографічній атаці IDN полягає експлойт. Зловмисники використовують ці схожі на вигляд символи з різних сценаріїв, щоб створити оманливі доменні імена. Такі ресурси імітують зовнішній вигляд легітимних, знайомих доменних імен. Вони потенційно змушують користувачів повірити, що вони відвідують перевірений сайт. У процесі вони найчастіше потрапляють на шкідливий сайт, призначений для фішингу. Загалом омографічні атаки на IDN виходять за рамки орфографічних помилок і можуть здійснюватися з доменами, які візуально майже неможливо відрізнити від імітованих веб-сайтів або брендів [5].

26 серпня 2022 року Консорціум Unicode оприлюднив Технічні стандарти Unicode №39 (UTS №39), у яких вичерпно описані механізми, використовувані для виявлення та уникнення проблем безпеки, пов’язаних із застосуванням Unicode. Разом із UTS №39 було випущено «confusables.txt» – документ, що містить відображення візуальної плутанини для використання при виявленні можливих проблем безпеки [6]. Документ містить всі символи Unicode, які візуально схожі на латинські літери, а також інші символи, дозволені для використання в доменах IDN.

1.2.2 Тайпо-домени

На відміну від омографічних доменів, тайпосквотинг – це процес реєстрації доменних імен, які містять орфографічну помилку відносно популярних чи розповсюджених доменів. У більшості випадків зловмисники конфігурують ресурси, які нагадує добре відому фразу, захищену торговою маркою чи авторським правом. Приклад можливих перетворень представлено на рис. 1.4.

y0ordomein.com
yourd0rnain.org
yourdomaln.net
yourdomain.it
yourdoma1n.gov

Рисунок 1.4 – Приклади тайпо-доменів

Було виявлено [7], що люди частіше допускають помилки, які включають відстань в один символ, яку також називають одиничною відстанню Дамерау-Левенштейна, Damerau-Levenshtein (DL). Попередньо в 1966 році було введено поняття відстань Левенштейна [8], що визначає схожість двох рядків як кількість операцій, необхідних для перетворення одного рядка в інший. Допустимі операції передбачають вставки, видалення та заміни. Наприклад, для рядків «care» і «car»

відстань Левенштейна буде рівна одиниці. Шляхом видалення або вставки літери «e» можна перетворити «care» на «car» за один крок операції, і навпаки. Алгоритм для обчислення відстані Левенштейна може бути описаний за допомогою наступної рекурсії:

$$d_{a,b}(i,j) = \begin{cases} 0 & \text{якщо } i = j = 0, \\ d_{a,b}(i-1,j) + 1 & \text{якщо } i > 0, \\ d_{a,b}(i,j-1) + 1 & \text{якщо } j > 0, \\ d_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} & \text{якщо } i,j > 0, \end{cases} \quad (1.1)$$

Відстань Левенштейна реалізується через матрицю розміром $n \times m$, де n та m позначають довжину рядка a та b відповідно. Таким чином, i та j вказують на рядки та стовпці матриці. Значення обчислюються ітерацією по кожному рядку та стовпцю, застосовуючи вищезазначене рівняння [9]. На рис. 1.5 представлено приклад такої матриці.

		c	**a**	**r**	**e**
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	0	1	2
r	3	2	1	0	1

Рисунок 1.5 – Приклад матриці Левенштейна

Аналогічно розглянутому поняттю, відстань Дамерау-Левенштейна, запропонована Фредом Дамерау в 1964 році, є також метрикою редагування з відмінністю від Левенштейна: транспозиції розглядаються як допустима операція. Еквівалент транспозиції відстані Левенштейна включає дві операційні дії: вилучення та вставку. На відміну від цього, транспозиція відстані Дамерау-Левенштейна розглядається як один крок. Один із можливих прикладів даного порівняння представлено на рис. 1.6-1.7.

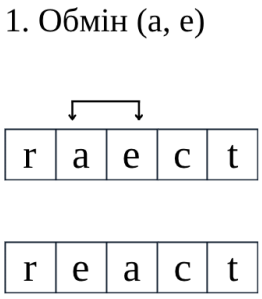

Рисунок 1.6 – Приклад транспозиції Дамерау-Левенштейна

Рисунок 1.7 – Відповідність транспозиції у відстані Левенштейна

Наступне рівняння нагадує рекурсію Левенштейна з додаванням включеної транспозиції:

$$d_{a,b}(i,j) = \{0\ d_{a,b}(i-1,j) + 1\ d_{a,b}(i,j-1) + 1\ d_{a,b}(i-1,j-1) + \quad (1.2)$$
$$+\ 1_{(a_i \neq b_j)}\ d_{a,b}(i-2,j-2) + 1,$$

якщо $i = j = 0$, якщо $i > 0$, якщо $j > 0$, якщо $i, j > 0$, якщо $i, j > 1$ та

$$a_i = b_{j-1} \text{ та } a_{i-1} = b_j$$

Таким чином, друкарські помилки DL-1 можна розділити на чотири загальні типи: додавання, пропуск, перестановка і заміна символів. Перші два види є

очевидними в той час, як перестановка вказує на обмін місцями двох суміжних символів, а заміна передбачає введення хибного символу в рядку замість необхідного.

Неправильна інтерпретація представляє потенційні випадки, коли користувачі помилково ідентифікують два символи, які схожі один на одного. Наприклад, користувачі можуть хибно сприймати «l» (ел) у «alice» як «1» (один) і, відповідно, використовувати «a1ice» під час процесу завантаження [10]. Загалом, навіть «fat-finger» або неправильна клавіша (два символи розташовані близько один до одного на клавіатурі) також викликають подібні ситуації.

Розглянемо детальніше техніку «typosquatting» разом з додатковими технічними відомостями. Кіберзлочинці реєструють доменні імена, які націлено містять помилки у написанні популярних веб-сайтів. Їм може навіть вдалося придбати кілька варіантів такого помилкового домену, щоб збільшити ймовірність привернення довірливих відвідувачів. Наприклад, маючи на меті завдати атаку клієнтам або працівникам компанії Microsoft через її офіційний ресурс «microsoft.com», зловмисник буде купувати «microsft.com», «microsoftt.com» або «micosoft.com».

Далі ініціюється етап взаємодії з користувачем, тобто в даній ситуації тайпо-домени стають небезпечними. Це може статися двома способами:

1) хибне введення: користувачі можуть помилково вводити домен у свій веб-браузер через типографські помилки або поспіх під час набору;

2) фішинг: кіберзлочинці надсилають оманливі електронні листи із посиланнями на нелегітимні домени.

Для більшого обману користувачів, веб-сайти, що використовують техніку «typosquatting», розробляються так, щоб вони імітували вигляд і функціональність легітимних аналогів. Це включає додавання логотипів, елементів дизайну та контенту, що належить реальним організаціям.

Опишемо наступні сценарії, де задіяна техніка «typosquatting» в онлайн-середовищі для обману користувачів [11]:

1) помилки при введенні, які роблять користувачі через брак часу або велику залежність від автокорекції. Зловмисники заздалегідь реєструють домени з поширеними помилковими варіантами;

2) помилки в написанні: користувачі можуть не знати правильного написання назви бренду, і кіберзлочинці це добре усвідомлюють. Вони реєструють помилкові варіанти для перенаправлення користувачів на фейкову сторінку, забезпечуючи захоплення потенційних жертв;

3) альтернативні написання насамперед стосуються Сполучених Штатів Америки та Великої Британії, оскільки вони мають багато слів з однаковим значенням, але з відмінністю в написанні. Наприклад, розглянемо слово «favourite» – британський варіант, тоді як в американській англійській його пишуть «favorite»;

4) додавання чи вилучення дефісів у доменних іменах викликає плутанину з метою обдурити користувачів. Наприклад, шахраї можуть використовувати «best-online-shop.com.ua» замість «best-onlineshop.com.ua», щоб нажитися на неуважних відвідувачах;

5) маніпуляція структурою URL: зловмисники здатні додавати піддомени аби надати URL подібного вигляду, як у легітимного. Наприклад, конфігурація online.cyber.security.com замість online.cybersecurity.com, де реалізовані піддомени, вводить в оману клієнтів і направляє їх на шахрайський сайт;

5) комбінація пов’язаних слів: кіберзлочинці створюють тайпо-домени, поєднуючи схожі слова, що стосуються тематики цільового домену. Наприклад, «online-cybersecurity-tutorial.com» замість «online-cybersecurity.com», де комбінація підвищує ймовірність того, що користувачі потраплять у пастку;

6) схожі доменні розширення: наявність різних TLD для різних країн і організацій надає додаткові можливості для атак. Зловмисники спрямовують свої зусилля на схожі TLD, такі як використання «.co» замість «.com», щоб обдурити відвідувачів і направити їх на свої підставні веб-сайти.

Варто наголосити, що вплив техніки «typosquatting» виходить за межі окремих користувачів і також несе загрозу власникам бізнесу. Кожен відвідувач,

перехоплений цими зловмисними сайтами, представляє потенційно втраченого клієнта для легітимних компаній.

1.2.3 Домени, згенеровані через алгоритми

Останнім класом у даній роботі буде представлено доменні імена, створені за допомогою алгоритмів генерації доменів. DGA генерує випадковий рядок, вводячи заздалегідь визначене початкове значення (SEED), і комбінує домен другого рівня, Second Level Domain (SLD) та домен верхнього рівня, щоб створити доменну адресу. Хоча DGA генерує мільйони доменів, зловмисник може передбачити, який домен DGA сформує, використовуючи дані часового ряду, такі як час або курси валют, які він може знати одночасно зі значенням SEED. Зловмисник розраховує доменну адресу, яку створить DGA, завчасно реєструє та використовує адресу домену сервера Command and Control (C&C) через легітимні процедури. На рис. 1.8 детально висвітлено процес з'єднання шкідливого програмного забезпечення (ШПЗ) із сервером C&C шляхом застосування DGA [12].

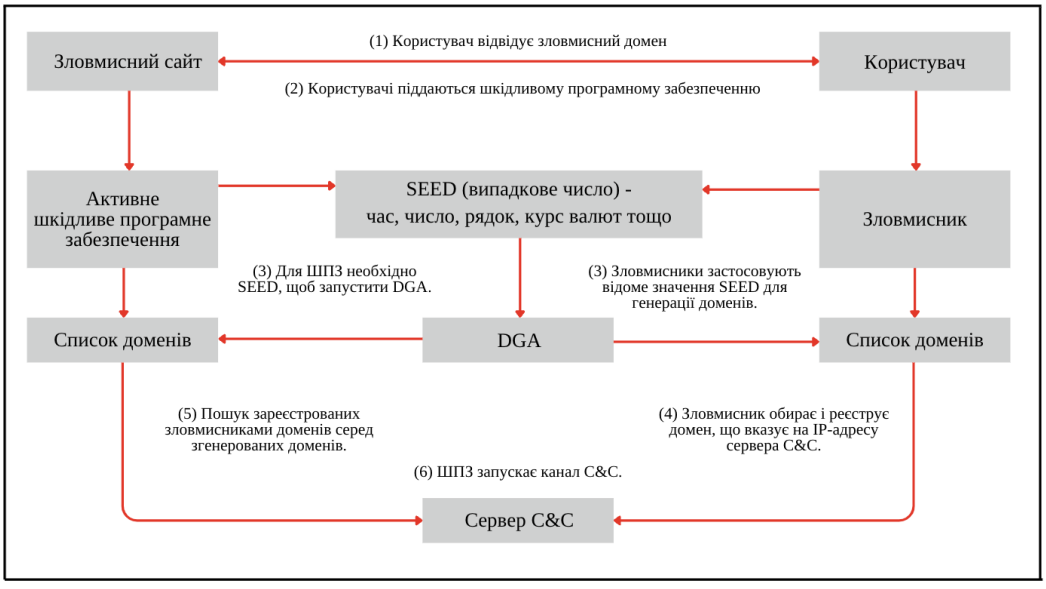

Рисунок 1.8 – Процес роботи загального DGA

На основі виявленого доменного імені, що використовує DGA, типи алгоритмів можна описати наступним чином [13]:

1) на основі арифметики: доменні імена генеруються шляхом розрахунку значень за визначеною формулою, які потім перетворюються в літери, числа або символи на основі ASCII або випадкового кодування. Цей алгоритм є найбільш широко використовуваним. Прикладом є домен fgavropgu.com – згенеровано алгоритмом DGA вірусу Conficker;

2) на основі хешу: алгоритми використовують методи хешування для створення доменів. Техніки хешування включають такі, як MD5 та SHA256. Прикладом є домен 47faeb4f1b75a48499ba14e9b1cd895a.org – створено алгоритмом DGA вірусу Bamital;

3) на основі словника: доменні імена утворюються шляхом поєднання двох чи більше слів із списків англійської або інших мов. Такі згенеровані зловмисні домени важко виявити, оскільки на перший погляд вони нагадують звичайні, легітимні домени, що не вказують на сервер C&C. Прикладом є домен catpeakfearinterview.com – створено алгоритмом DGA вірусу Matsnu.

У табл. 1.1 наведено приклади відомих типів DGA, їхні техніки та зразки згенерованих доменних імен відповідно. Техніка представляє SEED, який використовується DGA [12].

Таблиця 1.1

Стислий опис існуючих методів DGA

DGA	**Техніки DGA**	**Приклад домену**
Zeus	MD5 року, місяця, дня та порядкового номера від 0 до 999	krhafeobleyhiytrwuduzlbucutwt, vsmfabubenvibwolvgilhirvmz
Conficker	Greenwich Mean Time як початкове число генератора випадкових чисел	fabdpdri, sfqzqigzs, whakxpvb
Kraken	Випадковий рядок від 6 до 11 символів	rhxqccwdhwg, huwoyvagozu, gmkxtm
Srizbi	Перетворення даних за допомогою операцій XOR	wqpyygsq, tqdaourf, aqforugp
Torpig	Поточна дата та число 8 як початкове число генератора випадкових чисел	16ah4a9ax0apra, 12ah4a6abx5apra, 3ah0a16ax0apra
Kwyjibo	Марковський процес на англійські склади	overloadable, refillingman, multible

Значення SEED для генерації домену за алгоритмом Zeus використовує MD5-хеш року, місяця, дня та порядкового номера в діапазоні від 0 до 999. П’ять правих біт згенерованого MD5-хеша додаються до шістнадцяткового значення літери «a». Потім числа перетворюються у буквені символи. Усі згенеровані буквені символи конкатенуються для формування доменного імені.

Ботнет Srizbi інфікував приблизно 450 000 комп’ютерів для відправлення спаму. Він виконує операції виключного «або» (XOR), використовуючи конкретні дні, місяці й роки, та ділить, змінює та конкатенує. У результаті щоденно генерується чотири доменні імена протягом року.

Torpig вкрав близько 500 000 онлайн-банківських рахунків, а також фінансову інформацію, таку як кредитні картки. Так само цей алгоритм генерує щоденно доменне ім’я, використовуючи поточну дату.

Kwyjibo створює словник випадкових коротких слів, які легко вимовляються. Крім того, він використовує алгоритм переривання слів для розділення словника на слова з 2-4 склади.

Загалом існують різні алгоритми генерації доменних імен, відповідно до яких реалізовані окремі технології. DGA охоплюють клас алгоритмів, який періодично створює псевдовипадкові комбінації символів або слів для формування сотень чи навіть тисяч нових доменних імен. Ключова ідея полягає в тому, що DGA можуть генерувати той самий набір нових доменних імен при запуску на двох різних машинах, таких як бот-майстер і заражена машина. Бот-майстер реєструє одне створене доменне ім’я, тоді як заражені машини систематично запитують домени зі списку до того моменту, поки одне з них не буде з’ясоване. Домени зі списку, які не були зареєстровані бот-майстром, зазвичай викликають відсутню відповідь від домена після запиту і можуть бути відкинуті зараженою машиною [14].

Як тільки заражена машина запитує зареєстроване доменне ім’я, встановлюється зв’язок між ботом і центром управління. З цього моменту можуть виконуватися зловмисні дії, які визначив центр C&C. Постійні зміни в домені сервера центру управління роблять набагато складнішими виявлення та обмеження атак для

Intrusion Detection System (IDS) та брандмауерів. Виклик в подоланні атак, які використовують техніки DGA, полягає в ідентифікації зловмисних доменів [14].

1.3 Розповсюдження атаки C&C в онлайн-просторі

Успішна кібератака – це не лише проникнення у сторонню організацію, використовуючи непомічені методи. Щоб отримати реальну користь, нападник повинен забезпечити постійність у цільовому середовищі, взаємодіяти з інфікованими або скомпрометованими пристроями всередині мережі та виводити конфіденційні дані. Ключем до виконання всіх цих завдань є стійка інфраструктура керування та контролю – C&C.

Ботнети – це мережі комп'ютерів, які заражені шкідливим програмним забезпеченням (ботами) та віддалено керуються нападником (ботмайстром) через канал комунікації для команд та контролю, C&C. Ботнети стали основним засобом для кіберзлочинців у здійсненні їхніх зловмисних дій, таких як проведення атак з відмовою в обслуговуванні, відправлення спаму та крадіжки особистих даних [15].

Останні дослідження свідчать, що деякі ботнети складаються з більше ніж мільйона ботів [16], що яскраво ілюструє масштаб їхньої загрози. Представники правоохоронних органів та дослідники з безпеки часто намагаються завадити поширенню активним ботнетам, проводячи заходи з їхньої ліквідації. Тут основною метою є інфраструктура комунікації з C&C ботнету. Одним із визначених прикладів таких дій є затоплювання (sinkholing), при якому всі боти перенаправляються на машину, що контролюється нападником, і називається воронкою. У результаті цього боти не в змозі взаємодіяти з оригінальними серверами C&C. Відповідно до цих зусиль, ботмайстри розпочали вигадувати нові техніки для захисту інфраструктури своїх ботнетів. Важливим напрямком, який отримав широку популярність у останні роки, є використання алгоритмів генерації доменних імен.

Використання DGA створює вкрай асиметричну ситуацію між нападниками (ботмайстрами) та захисниками (дослідниками з безпеки та правоохоронцями). Ботмайстрам потрібен доступ до одного домену для керування або міграції своїх

ботів, тоді як захисникам потрібно контролювати всі домени для забезпечення успішного припинення діяльності. З більш ніж 1000 TLD для вибору, нападникові легко створити глобально розподілену відповідальність за доменами, що змушує захисників докладати екстремальних зусиль до координації та співпраці. Наприклад, DGA ботнету Conficker генерував 50 000 доменних імен щодня, розподілених по 113 TLD. Це вимагало глобальних спільних зусиль 30 різних організацій, включаючи ICANN, для стримування загрози [15].

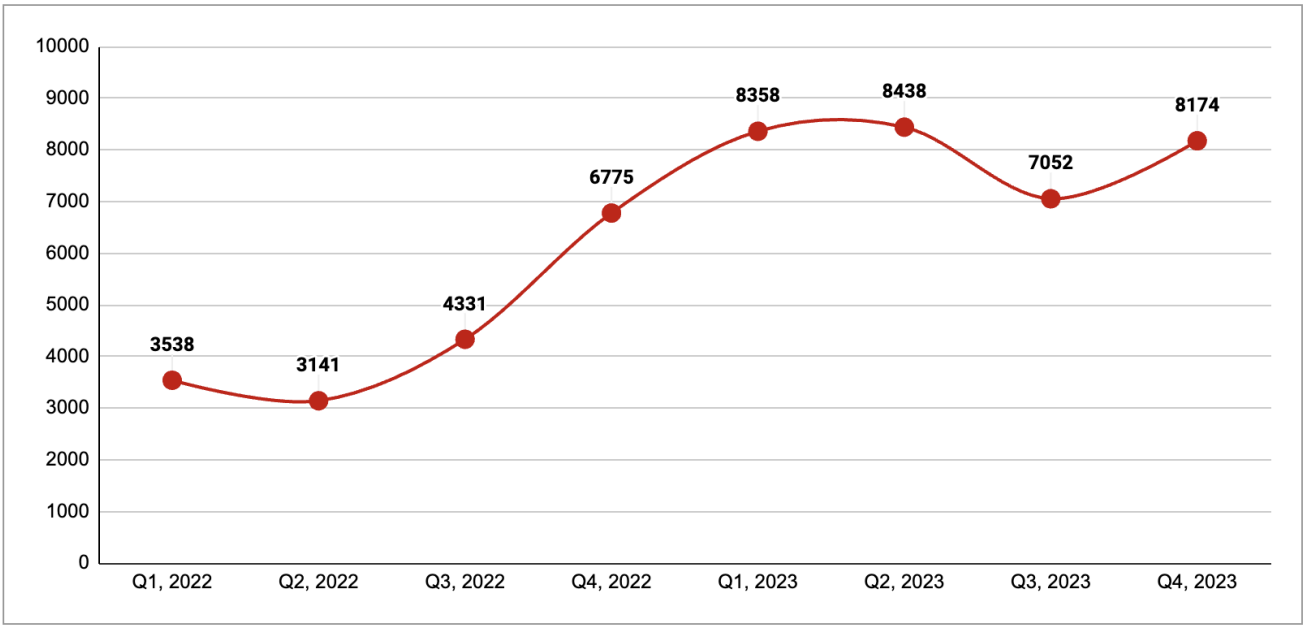

Рисунок 1.9 – Кількість атак C&C протягом 2022-2023 років

Як представлено на рис. 1.9, згідно з оновленим звітом про загрози ботнетів за четвертий квартал 2023 року, який був оприлюднений Spamhaus Project [17], було виявлено 8174 C&C серверів ботнетів, порівняно з 6775 в аналогічний період 2022. Це свідчить про значний ріст у кількості створених серверів ботнетів, а саме більше, ніж на 20%. На квартальній основі кількість виявлених серверів ботнетів зросла з 7052 в третьому кварталі 2023 до 8174 в четвертому 2023. У зв’язку з цим значним збільшенням вимагаються подальші дослідження для виявлення доменних імен, згенерованих алгоритмами, з метою зменшення ризику інфікування комп’ютерних пристроїв ботнетами.

Аналізуючи геолокацію ботнетів C&C, Китай утримує лідерську позицію, витіснивши США з першого місця в третьому кварталі. У четвертому кварталі кількість ботнетів, розміщених у цій країні, продовжила зростати на додаткові 35% і досягла 2125, не виявляючи ознак сповільнення.

Разом з Китаєм, США виходить на передову позицію зі значущими відсотковими збільшеннями у ботнет-центрах управління – на 27%. Проте найбільший приріст у четвертому кварталі відзначається в Болгарії із вражаючим збільшенням на 227%. Загалом на противагу третьому кварталу більшість геопозицій зафіксувала збільшення кількості ботнет-центрів управління, за винятком таких країн, як Індія (-44%), Канада (-41%), Мексика (-39%), Сполучене Королівство Великої Британії та Північної Ірландії (-21%) і Уругвай (-6%). Тим часом, Південна Африка та Швейцарія вийшли з двадцятки лідерів. Статистичні дані перших 20 країн за кількістю атак представлено на рис. 1.10.

Top 20 locations of botnet C&Cs

Rank	Country	Q3 2023	Q4 2023	% Change Q on Q
#1	China	1570	2125	35%
#2	United States	1267	1606	27%
#3	Russia	441	612	39%
#4	Netherlands	542	603	11%
#5	Germany	378	478	26%
#6	France	242	322	33%
#7	Bulgaria	64	209	227%
#8	Singapore	164	190	16%
#9	United Kingdom	221	175	-21%
#10	Uruguay	170	160	-6%

Rank	Country	Q3 2023	Q4 2023	% Change Q on Q
#11	Saudi Arabia	153	159	4%
#12	Mexico	232	141	-39%
#13	Japan	70	111	59%
#14	Finland	78	102	31%
#15	Canada	157	92	-41%
#16	Poland	-	90	New entry
#16	Sweden	69	80	16%
#18	Korea (Rep. of)	73	74	1%
#19	India	123	69	-44%
#20	Turkey	-	55	New entry

Рисунок 1.10 – Статистичні дані атак C&C за третій та четвертий квартали 2023 року

Щодо ШПЗ, пов’язане з ботнетами, то розглянемо детальні дані звіту, що представлені на рис. 1.11. Після входження у шостий квартал поспіль як продукт, пов’язаний з найбільшою кількістю центрів C&C, Cobalt Strike утримав звичну позицію в четвертому кварталі зі збільшенням на 26%. Цей інструмент для тестування

на проникнення пов'язаний із понад чотириразовою кількістю центрів управління ботнетами, ніж його найближчий конкурент, AsyncRat, що посідає друге місце.

Згадуючи про засоби віддаленого доступу, то QuasarRAT пережив значущий приріст (341%), піднявшись з 12-го місця на 5-те в четвертому кварталі. Це не є несподіванкою, враховуючи, що це відкритий інструмент; тому будь-хто може налаштувати сервер і використовувати його.

Malware families associated with botnet C&Cs

Rank	Q3 2023	Q4 2023	% Change	Malware Family	Description
#1	2491	3137	26%	Cobalt Strike	Pentest Framework
#2	373	710	90%	AsyncRAT	Remote Access Trojan (RAT)
#3	285	515	81%	Sliver	Pentest Framework
#4	646	513	-21%	Flubot	Android Backdoor
#5	75	331	341%	QuasarRAT	Remote Access Trojan (RAT)
#6	341	329	-4%	Remcos	Remote Access Trojan (RAT)
#7	273	286	5%	RedLineStealer	Remote Access Trojan (RAT)
#8	197	229	16%	DCRat	Remote Access Trojan (RAT)
#9	-	223	New entry	Hook	Android Backdoor
#10	-	198	New entry	Pikabot	Backdoor
#11	-	181	New entry	FakeUpdates	Loader/downloader
#12	797	145	-82%	Qakbot	Backdoor
#13	186	111	-40%	IcedID	Credential Stealer
#14	-	88	New entry	Meterpreter	Backdoor
#15	89	78	-12%	NjRAT	Remote Access Trojan (RAT)
#16	63	77	22%	Havoc	Backdoor
#17	-	66	New entry	BianLian	Ransomware
#18	-	49	New entry	Lumma	Credential Stealer
#19	53	45	-15%	Rhadamanthys	Credential Stealer
#20	269	42	-84%	RecordBreaker	Credential Stealer

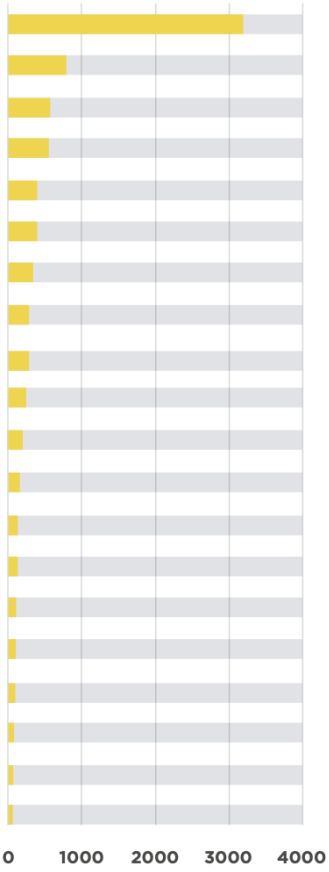

Рисунок 1.11 – Шкідливе програмне забезпечення, пов'язане з ботнетами, у третьому та четвертому кварталах 2023 року

У другому та третьому кварталах дослідники Spamhaus зафіксували збільшення кількості центрів управління ботнетами, пов'язаних із законними, але зловмисними інструментами для тестування на проникнення. У четвертому кварталі ця тенденція трималася із зростанням від 42,9% до 49,7%, що означає, що ці засоби становили

майже 50% від ШПЗ, пов’язаних із C&C у розглядуваному списку перших 20-ти інструментів. У четвертому кварталі також було зафіксовано шість нових вірусів, включаючи Pikabot на 10-му місці, пов’язаного із 198 ботнетами C&C. Виявлений вперше у 2023 році, цей backdoor-вірус містить кілька методів ухилення від пісочниць, віртуальних машин та інших типів відлагодження. Дослідники Spamhaus також зауважили, що Pikabot був вибором ШПЗ для відомого кіберзлочинця TA577.

1.4 Підходи до запобігання кіберзагроз із застосуванням доменів

У зв’язку із серйозними загрозами від зловмисних доменних імен, впроваджуються різні підходи для їхнього виявлення та запобігання експлуатації. Дані стратегії можна розділити на дві основні категорії – на основі чорних списків або евристичні методи та на базі на машинного навчання.

Blacklisting є найбільш поширеним і широко застосовним методом для детектування шахрайських доменів та обмеження їхнього поширення й впливу. У цьому підході ведеться список відомих зловмисних доменів, який використовується для перевірки. Якщо відвіданий ресурс знаходиться в списку, він вважається нелегітимним, і генерується відповідне сповіщення чи розміщення його в карантин. В іншому випадку, домен вважається безпечним. Перевагою чорних списків є швидкість виявлення відомих зловмисних доменів. Однак метод не здатний ідентифікувати нові ресурси, оскільки вони відсутні у базах. Крім того, неможливо підтримувати повний список усіх таких доменів через легкість реєстрації та конфігурації [18].

Евристичний підхід є розширенням методу blacklisting, у якому створюється та впроваджується база даних сигнатур атак або вторгнень замість списку зловмисних доменів. Сигнатура атаки може містити різні елементи, включаючи рядки та IP-адреси. Дані такого типу широко використовуються в системах виявлення вторгнень, IDS, для виявлення різних видів атак. Подібно до методу чорних списків, евристичний підхід не може детектувати нові атаки чи вторгнення, оскільки їхні

сигнатури не знаходяться в базі даних. Крім того, також дуже складно підтримувати в актуальному стані базу даних сигнатур атак і вторгнень [18].

Також часто використовується аналіз зворотної інженерії для розбору DGA, щоб ідентифікувати DGA через зловмисні зразки доменів, а потім їх заблокувати. Проте даний підхід залишається ефективним лише у відношенні невеликої частини DGA та вимагає значних зусиль і часу від експертів у цій галузі. Таким чином, метод зворотної інженерії є дуже повільним і непрактичним порівняно з іншими підходами.

У той же час, підхід, що базується на машинному навчанні, спочатку намагається проаналізувати інформацію про безпечні та зловмисні домени та/або їхні відповідні веб-сайти чи сторінки, щоб витягти ознаки для створення тренувальних даних. Далі ця віднайдена інформація вводиться для побудови класифікатора чи профілю за допомогою машинного навчання. Архітектури можуть бути традиційними керованими, такими як Naive Bayes, дерево рішень і випадковий ліс, або техніками глибокого навчання, такими як згорткова нейронна мережа, Convolutional Neural Network (CNN), рекурентні нейронні мережі, Recurrent Neural Networks (RNN) та довгострокова короткочасна пам’ять, Long Short-Term Memory (LSTM) [18].

Підхід на ML отримав велику увагу від наукової громадськості через його потенціал виявлення нових зловмисних доменів. Крім того, класифікатор чи профіль може бути автоматично створений за допомогою тренувальних даних, що зменшує потребу в ручній обробці для підтримки чорних списків або баз даних сигнатур.

Висновок до першого розділу

Як було досліджено, останні роки характеризуються широким впровадженням DGA сучасними ботнетами. Основна мета полягає в генерації великої кількості доменів і використанні невеликої підмножини для фактичної комунікації з сервером C&C. Такі можливості роблять DGA дуже привабливими для ботмайстрів, щоб зміцнити інфраструктуру їхніх ботнетів та зробити її стійкою до спроб ліквідації.

У розділі було розглянуто типи утворення шахрайських доменів. Опрацювавши специфіку атак із їхньою експлуатацією, було підтверджено необхідність сконцентруватися на виявленні доменів DGA. Швидке та точне детектування цих ресурсів є одним із ключових моментів у боротьбі з ботнетами. Існуючі методи виявлення мають багато недоліків, які не дозволяють повністю опрацювати характеристичні особливості доменних імен DGA.

РОЗДІЛ 2
ЗАХИСТ ВІД ДОМЕННИХ АТАК ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

2.1 Перспективи машинного навчання щодо попередження доменних атак

У 1955 році Джон Маккарті запропонував термін «штучний інтелект» (ШІ) та визначив його як «науку та інженерію створення розумних машин». Машинне навчання, Machine Learning (ML), підгалузь ШІ, також було представлене в те саме десятиріччя, але стало більш популярним в 1990-х роках завдяки розвитку обчислювальних технологій та експоненційному зростанню обсягу цифрових даних. У ML математичні та статистичні концепції утворюють основу для розвитку різноманітних складних моделей, призначених в основному для виявлення закономірностей, кореляцій та аномалій у даних. Результати їх роботи виражаються у вигляді ймовірностей та довірчих інтервалів. З огляду на обмеження у наявності експертів для аналізу великих обсягів даних, ML використовується для успішної автоматизації процесу навчання на основі ШІ [19].

Загалом машинне навчання охоплює навчання комп’ютерних систем на основі даних і прийняття прогнозів або рішень без явного програмування для цього. Моделі ML призначені для дослідження наборів даних, ідентифікації закономірностей та формування передбачень на їхній основі. Застосування ML включає в себе розпізнавання зображень та мовлення, обробку природної мови, генерацію систем рекомендацій та прогнозування для існуючих сфер. Безперечно, машинне навчання – це зростаюча галузь із значущим потенціалом для поліпшення багатьох аспектів нашого життя [20]. Існують три основні парадигми машинного навчання: кероване навчання (supervised learning), некероване навчання (unsupervised learning) та навчання з підкріпленням (reinforcement learning).

Кероване навчання – парадигма ML, у якій модель навчається на основі позначених даних. Модуль отримує набір вхідних-вихідних пар, відомих як навчальні

дані, і мета полягає в тому, щоб навчити функцію, яка точно може зіставити нові входи з їхніми відповідними виходами. Тут використовуються статистичні методи для визначення закономірностей та взаємозв'язків в даних та створення моделі, яка може узагальнити до нової, неявної інформації. Кероване навчання часто застосовується для завдань, таких як класифікація та регресія, де мета полягає в передбаченні категоріальної або числової вихідної змінної на основі однієї чи кількох вхідних змінних [20]. Прикладами використань даного підходу ML є фільтрація спаму, розпізнавання патернів атак, аналіз вразливостей систем та автоматизоване виявлення потенційних кіберзагроз.

Некероване навчання – наступна парадигма машинного навчання, у якій модель навчається з'ясувати закономірності та структури в непозначених даних. На відміну від supervised learning, у даному підході не надається позначена вихідна інформація. Замість цього модель отримує лише вхідні дані і повинна самостійно визначити закономірності чи взаємозв'язки в них. Моделі некерованого навчання використовують такі техніки, як кластеризація та аналіз головних компонентів, Principal Component Analysis, щоб об'єднати схожі точки даних або з'ясувати структуру інформацію [20]. Останній підхід є технікою зменшення розмірності, яка впроваджується в ML для перетворення набору даних великої розмірності в простір меншої розмірності зі збереженням найважливіших деталей. Застосування некерованого навчання дозволяє кластеризувати аномальну активність в мережі без заздалегідь визначених правил, що допоможе ідентифікувати нові загрози та вразливості без необхідності чіткого наперед заданого шаблону. Воно особливо корисне, коли позначені дані обмежені або їх важко отримати.

Навчання з підкріпленням – це ще одна парадигма машинного навчання, у якій машина (яку ще також називають «агентом») вчиться приймати рішення, взаємодіючи з оточенням і отримуючи зворотний зв'язок у вигляді винагород чи покарань. Метою агенту є максимізація накопиченої винагороди з часом. Моделі протягом навчання з підкріпленням використовують методи проб і помилок для вивчення оптимальних дій в конкретній ситуації на основі отриманих відгуків. Агент вчиться шляхом поєднання дослідження та експлуатації, випробовуючи різні ходи,

щоб визначити, які з них приводять до найвищих винагород. Даний підхід може включати розробку системи, яка автоматично навчається розпізнавати та вчасно реагувати на нові види шкідливого програмного забезпечення. Також це особливо корисно в ситуаціях, коли коректна стратегія наперед невідома, і агент повинен вчитися на основі власного досвіду.

Існує багато різних типів моделей машинного навчання в межах кожної парадигми ML, кожна з яких має свої переваги та недоліки. Вибір правильної моделі для конкретного завдання вимагає ретельного врахування даних, проблеми та бажаного результату. У сфері кібербезпеки машинне навчання виявляє великий потенціал у виявленні шахрайських доменних імен, що є важливою складовою боротьби з C&C, Distributed Denial-of-Service (DDoS) та фішинговими атаками, та іншими формами кіберзлочинності.

Моделі машинного навчання здатні враховувати такі параметри, як довжина домену, використання піддоменів, наявність чисел чи спецсимволів, які часто характерні для шахрайських схем. Моделі можуть також аналізувати структуру URL, наявність ключових слів, а також історію реєстрації домену. Використання існуючих даних дозволяє моделям розпізнавати шаблони та тенденції у створенні небезпечних доменів. Це робить можливим виявлення нових загроз, що ґрунтується на попередніх взаємодіях і патернах шахрайської діяльності. Узагальнюючи, ML сприяє створенню більш інтелектуальних та адаптивних систем виявлення зловмисних доменних імен, що забезпечує захист від онлайн-загроз.

2.2 Метрики оцінки моделей

В останнє десятиріччя моделі машинного навчання значно розширили своє застосування, перейшовши від простих експериментів до необхідної складової різноманітних галузей. Від науки про дані та аналітики до розробки продуктів та прийняття стратегічних рішень у сучасному світі, вони стали невід’ємною частиною технологічного ландшафту.

Проте із широким впровадженням моделей ML виникають питання про їхню точність та відповідність поставленим завданням. Під час розгортання та оптимізації моделей виникає потреба у кількісній оцінці їхньої продуктивності. Саме для цього використовуються метрики – набір інструментів, що дозволяють об’єктивно вимірювати, наскільки успішно модель впоралася з поставленим завданням та як вона працює з реальними даними.

Метрики стають ключовим індикатором в оцінці ефективності моделей, дозволяючи визначити їхню придатність для конкретних завдань машинного навчання. Враховуючи різноманіття цих завдань – від класифікації об’єктів до прогнозування числових значень – розмаїття метрик надає можливість глибше розуміти різні аспекти роботи моделей.

У процесі розробки для виявлення шахрайських доменних імен, метрики машинного навчання виступають як кількісні величини для вимірювання якості отриманої моделі. Вони надають об’єктивні критерії оцінки, дозволяючи оцінити її здатність правильно відрізняти небезпечні від легітимних доменів. Використання таких метрик не лише надає узагальнену картину ефективності моделі, але й допомагає в налаштуванні її параметрів для досягнення оптимального балансу.

Опишемо такі метрики продуктивності, як точність (accuracy), влучність (precision), повнота (recall) та F1-показник, для аналізу результативності моделі щодо виявлення небезпечних ресурсів. Ці індикатори є необхідними для розуміння сильних та слабких сторін моделі у відокремленні між позитивними (шахрайськими) та негативними (легітимними) класами.

Точність, влучність, повнота і F1-показник розраховуються на основі чотирьох можливих результатів прогнозів: істинно позитивний, true positive (TP); хибно позитивний, false positive (FP); істинно негативний, true negative (TN) та хибно негативний, false negative (FN). TP відповідають випадкам, коли модель правильно передбачає позитивний клас. Отже, у контексті виявлення шахрайських доменів істинно цей результат настає, коли правильно ідентифіковано ресурс як шахрайський, і він насправді є таким. FP відповідають випадкам, коли модель неправильно передбачає позитивний клас. Іншими словами, помилковий результат виникає, коли

помилково визначено легітимний домен як зловмисний. TN – це результати, коли модель правильно прогнозує негативний клас. Істинно негативний результат виникає, коли правильно детектовано домен як законний, і він справді є таким. FN – це результати, коли неправильно передбачено негативний клас. Тобто хибно негативний висновок з'являється, коли модель не може ідентифікувати шахрайський ресурс і неправильно класифікує його як законний [21].

Точність є фундаментальним показником, який вимірює загальну правильність моделі ML. Він кількісно визначає відношення правильно передбачених випадків (як істинно позитивних, так й істинно негативних) до загальної кількості примірників у наборі даних, як сформульовано у формулі (2.1). Висока точність вказує на здатність моделі робити правильні прогнози щодо позитивних (шахрайських) і негативних (легітимних) випадків.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.1}$$

Влучність вимірює коректність позитивних прогнозів, зроблених моделлю. Вона кількісно визначає відношення істинно позитивних прогнозів до загальної кількості випадків, прогнозованих як позитивні (TP, і FP), як сформульовано у формулі (2.2). Влучність є особливо важливою, оскільки вона оцінює здатність моделі уникати неправильної класифікації легітимних доменів як зловмисних. Значення високої влучності означає низьку частоту помилкових позитивних результатів.

$$Precision = \frac{TP}{TP+FP} \tag{2.2}$$

Повнота, також відома як чутливість або істинно позитивний показник, оцінює здатність моделі ідентифікувати всі позитивні випадки в наборі даних. Він вимірює відношення істинно позитивних прогнозів до загальної кількості фактичних позитивних випадків (істинно позитивних і хибно негативних), як сформульовано у формулі (2.3). Високий рівень повноти має вирішальне значення, оскільки він вказує

на здатність моделі щодо виявлення значної частини фактичних випадків онлайн-загроз, мінімізуючи хибно негативні результати.

$$Recall = \frac{TP}{TP+FN} \tag{2.3}$$

Показник F1 є середнім гармонійним значенням точності та повноти, що забезпечує збалансовану оцінку ефективності моделі, як сформульовано у формулі°(2.4).

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.4}$$

Параметр є пріоритетним для ситуацій, коли потрібна як висока точність, так і повнота. Показник F1 є особливо необхідним, коли досягнення балансу між точним визначенням шахрайських веб-сайтів і мінімізацією помилкових спрацьовувань має вирішальне значення [21].

Таким чином, використання описаних метрик сприяє об’єктивній оцінці продуктивності моделі та її здатності до точної ідентифікації класів вхідних даних.

2.3 Набори даних та їхній вплив на моделі ML

Моделі машинного навчання базуються на аналізі величезних обсягів відомостей – датасетів (datasets) – наборів даних, згрупованих у колекцію, з якою розробники можуть працювати для досягнення відповідних цілей. Тут рядки представляють кількість точок даних, а стовпці – їхні характеристики. Датасети можуть відрізнятися за розміром і складністю, і вони здебільшого потребують очищення та попередньої обробки, щоб забезпечити якість результату і придатність для аналізу чи моделювання. Найпоширенішими є файли CSV, Excel, JSON і zip для великих datasets. Ці складові моделі ML мають вирішальний вплив на результати,

забезпечуючи достатню кількість і варіативність інформації для тренування та удосконалення моделей.

Щоб виявити шахрайські домени, використовуються спеціально підготовлені датасети, які містять інформацію про характеристики доменів, що вказують на їхнє зловмисне призначення. Ці параметри охоплюють різноманітні аспекти, такі як структура доменного імені, часові параметри, попередні випадки обману, звіти про схожі домени. Технічно, моделі ML, навчені на таких датасетах, здатні автоматично з'ясовувати закономірності та особливості, що характеризують зловмисні домени. Це дозволяє побудувати ефективні моделі, які з високою точністю можуть класифікувати нові домени, визначаючи, наскільки вони ймовірно є об'єктом шахрайської діяльності. Використання датасетів робить процес виявлення нелегітимних ресурсів більш адаптованим до кіберпростору з постійними змінами й нововведеннями. Такий підхід стає гнучким елементом в сучасній боротьбі з кіберзлочинністю, надаючи можливість оперативно реагувати на нові загрози та захищати користувачів від потенційних атак [22].

З використанням датасетів у виявленні небезпечних доменів пов'язані деякі аспекти. По-перше, створення високоякісних та репрезентативних datasets вимагає систематичної обробки та аналізу великого обсягу інформації. Далі, важливо обрати та врахувати різноманітні етапи для навчання моделей. Розуміння того, які характеристики доменів є найбільш цінними для виявлення шахрайства, є значущим етапом. Також моделі ML, які передбачають класифікацію та кластеризацію, застосовуються для тренування моделей на підготовлених датасетах. Такі моделі здатні виявляти відмінності та аномалії в нових доменних іменах, класифікуючи їх за ймовірністю шахрайства [22].

Критичним елементом є постійне вдосконалення та оновлення моделей на основі нових даних, оскільки кіберзлочинці постійно адаптують свої методи. Це вимагає регулярного завантаження останніх відомостей та переоцінки моделей з метою забезпечення високої здатності систем щодо виявлення. Це запорука реалізації динамічної системи, що готова до викликів сучасних та майбутніх загроз кіберпростору.

Попри значний прогрес у використанні датасетів та моделей машинного навчання для виявлення шахрайських доменів, існують окремі виклики та питання, які потребують уваги та подальших досліджень. Одним із них є постійна еволюція методів кіберзлочинців. Вони завжди шукають нові способи ухилення від детектування, що може вимагати регулярного апгрейду моделей та постачання їх новими даними. Крім того, розуміння та аналіз контекстуальних змін в Інтернет-просторі також залишаються пріоритетними [22].

Іншим важливим аспектом є збалансованість між точністю та швидкодією моделей. Забезпечення високої точності виявлення є пріоритетним завданням, однак необхідно мати інструмент з майже миттєвим результатом, особливо в умовах великого потоку нових доменів. Це особливо актуально в момент потужних DDOS-атак. Крім того, питання конфіденційності та етичності у зборі та використанні даних є вельми вагомим в сучасному цифровому світі, де особиста та юридична інформація стає все більш вразливою перед загрозою її неправомірного застосування та порушення законодавства.

Усі ці виклики вимагають від дослідників та фахівців в галузі кібербезпеки постійного вдосконалення методів та технологій, щоб забезпечити надійний захист від шахрайських доменів у динамічному середовищі Інтернету.

Наразі немає загальнодоступних підтверджених датасетів для дослідження виявлення шкідливих доменів. Більшість опублікованих результатів щодо детектування зловмисних ресурсів базуються на приватних наборах даних при оцінці ефективності різних класичних моделей ML. Ці відомості збираються з різних джерел, таких як Alexa, DMOZ, Phishtank, OpenPhish, MalwareDomains, MalwareDomainList та інші. Однак такі підходи не можуть вважатися загальними через відсутність спільної інформації у методиках [23].

Загалом прийняття підходу ще досі перебуває у початковій стадії через відсутність стандартизації в моделях машинного навчання і датасетах. Розробка та використання бенчмарк датасетів рівноцінно постає невід’ємною складовою для встановлення здатності та порівняння різних методів виявлення та захисту від кіберзагроз. Тобто постає потреба, щоб дослідити наявні тактики у стандартизованих

умовах і визначити найкращі практики або найефективніші з них. У табл. 2.1 наведено перелік методів виявлення загроз, а також список бенчмарк датасетів, які застосовуються для їхньої оцінки в процесі забезпечення кібербезпеки.

Таблиця 2.1

Деталі бенчмарк датасетів

Task	**Benchmark Dataset**	URL
Intrusion detection	KDDCup-99	http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
	NSL-KDD	https://www.unb.ca/cic/datasets/nsl.html
	UNSW-NB15	https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
	ADFA-LD	https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
	UNM	https://www.cs.unm.edu/~immsec/systemcalls.htm
	ISCX-IDS-2012	https://www.unb.ca/cic/datasets/ids.html
	CICIDS2017	https://www.unb.ca/cic/datasets/ids-2017.html
	Kyoto	http://www.takakura.com/Kyoto data/
	UNIBS	http://netweb.ing.unibs.it/~ntw/tools/traces/
	CAIDA	https://www.caida.org/data/
	LBNL	https://commons.lbl.gov/display/cpp/100G+Intrusion+Detection
	CIC DoS	https://www.unb.ca/cic/datasets/dos-dataset.html
	CSE-CIC-IDS2018	https://www.unb.ca/cic/datasets/ids-2018.html
	AWID	http://icsdweb.aegean.gr/awid/index.html
	WSN-DS	https://www.hindawi.com/journals/js/2016/ 4731953/
Botnet and DGA Analysis	UNB Botnet	https://www.unb.ca/cic/datasets/botnet.html
	DGArchive	https://dgarchive.caad.fkie.fraunhofer.de/
	AmritaDGA	https://vinayakumarr.github.io/AmritaDGA/
URL Analysis	ISCX-URL-2016	https://www.unb.ca/cic/datasets/url-2016.html
	Sophos URL	https://github.com/Maddy12/SophosMachineLearningBuildingBlocksTutorial
Spam and phishing Email detection	CSDMC	http://csmining.org/spam-email-datasets-.html/
	Enron	https://www.cs.cmu.edu/~./enron/
	TREC	https://trec.nist.gov/data/spam.html
	SpamAssassin	https://spamassassin.apache.org/old/publiccorpus/
Malware Detection	EMBER	https://github.com/endgameinc/ember
	BIG 2015	https://www.kaggle.com/c/malware-classification
	Malrec	https://giantpanda.gtisc.gatech.edu/malrec/dataset/
	Microsoft Malware Prediction	https://www.kaggle.com/c/microsoft-malware-prediction
Binary Analysis	ByteWeight	http://security.ece.cmu.edu/byteweight/
Image spam detection	Image spam hunter	https://www.unb.ca/cic/datasets/android-adware.html
Android	AAGM	https://www.unb.ca/cic/datasets/android-adware.html
	CICAndMal2017	https:// www.unb.ca/cic/datasets/andmal2017.html
	Drebin	https://www.sec.cs.tu-bs.de/~danarp/drebin/
	Kharon	http://kharon.gforge.inria.fr/dataset/
Traffic Analysis	ISCXVPN2016	https://www.unb.ca/cic/datasets/vpn.html
	ISCXTor2016	https://www.unb.ca/cic/datasets/tor.html
Security in IoT	N-BaIoT	archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT
	Bot-IoT	https://ieee-dataport.org/documents/bot-iot-dataset
Side channel attacks	DPA contest	http://www.dpacontest.org/index.php
	ASCAD	https://www.data.gouv.fr/en/datasets/ascad/
Insider threat detection	CERT	https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099

Знаходження задовільного набору даних для використання в кібербезпеці часто ускладнюється чотирма основними причинами [19]:

1) значна більшість публічно доступних наборів даних є застарілими;

2) вони не є автентичними наборами даних;

3) більшість дослідників використовують різні методи розділення даних на категорії «навчання», «перевірка» та «тестування»;

4) вони не доступні широкому колу дослідницької спільноти через причини безпеки та конфіденційності.

Це призводить до того, що експериментальні результати не є відтворюваними. Через ці проблеми впровадження кібербезпеки не має стандартизованого підходу, і більшість підприємств уникають використання рішень машинного навчання для покращення своїх застосунків в галузі кібербезпеки.

Існує кілька конкуруючих топ-списків легітимних доменів, у кожному з яких існує унікальна методологія визначення популярності. У деяких випадках списки спеціально ранжують найпопулярніші веб-сайти, тоді як інші просто обчислюють найпопулярніші доменні імена, незалежно від протоколу прикладного рівня. Майже в усіх випадках списки намагаються надати приблизний порядок ранжування списку імен. Опишемо списки, які найчастіше використовуються в моделях ML як джерело правомірних доменів [24]:

1) Рейтинг Alexa – це сервіс, який Amazon надає громадськості для оцінки популярності доменних імен. Завдяки статистиці та аналізу кожного доменного імені щодо кількості доступу, посилань та інших аспектів рейтинг доменного імені генеруються відповідно до результатів перевірки. Тому, якщо доменне ім’я займає відносно високі позиції в Alexa, воно, швидше за все, буде безпечним. Роботу веб-сайту було закрито 1 травня 2022 року. Список досліджень найпопулярніших веб-сайтів Alexa базується на останньому оновленому файлі CSV від 1 лютого 2023 року.

2) Cisco Umbrella 1 Million – це список найпопулярніших доменних імен (наприклад, .com займає перше місце), які переглядаються за допомогою служби DNS Cisco Umbrella. Їхній рейтинг базується на алгоритмі відповідності, що використовує кількість унікальних IP-адрес клієнтів, які відвідують кожен домен, відносно суми всіх запитів до всіх доменів для розрахунку популярності. Такий підхід, заснований на DNS, фундаментально відрізняється від збору відвідувань веб-сайтів або посилань. Таким чином, список Umbrella містить повноцінні доменні імена, Fully Qualified Domain Name, для будь-якого типу Інтернет-сервісу, а не лише веб-сайтів, як у випадку Alexa чи Majestic.

3) CommonCrawl – це некомерційний проект, який має на меті забезпечити загальнодоступний доступ до масиву веб-даних для стимулювання досліджень та інновацій у галузі обробки природної мови, машинного навчання, аналізу даних та інших суміжних областей. CommonCrawl застосовує вебскрейпінг для створення індексу веб-сторінок, а потім надає ці дані для загального використання, щоб дослідники, розробники програмного забезпечення та інші зацікавлені сторони могли реалізовувати ці дані для своїх цілей. CommonCrawl надає доступ до великої кількості веб-сторінок та їхнього вмісту, що може бути використано для різних цілей, включаючи прогнозування трендів в Інтернеті, розробку нових технологій та інше.

4) Majestic Million – це список популярних веб-сайтів, який підтримує Majestic SEO, що розраховується на основі кількості зворотних посилань на кожному сайті. Список з’явився у жовтні 2012 року, і його створено на базі пошукового робота Majestic. Majestic Million має ліцензію Creative Commons, яка вважається типовим методом, який автори можуть застосовувати для надання іншим особам дозволу на використання свого контенту. Часто Majestic Million впроваджується для аналізу тенденцій в Інтернеті, вивчення конкуренції, а також для планування маркетингових стратегій.

Ресурси з деталями про шахрайські доменні імена постають цінним інструментом для виявлення нових шкідливих доменів через реалізацію методів машинного навчання. Такі джерела надають доступ до широкого спектру інформації про історичні дані та патерни кібератак, що дозволяє розробляти релевантні моделі для автоматичної ідентифікації нових потенційно небезпечних доменів [25]:

1) PhishTank є онлайн-спільнотою та базою даних, яка служить для виявлення та спільного розповсюдження інформації про фішингові атаки. Ресурс дозволяє користувачам надсилати відомості про підозрілі веб-сайти, які, на їхню думку, можуть бути фішинговими, і наводять перевірку цих звітів. Якщо виявлено, що веб-сайт дійсно є шкідливим, його додають до бази даних PhishTank, яка доступна для перегляду та впровадження для захисту від таких атак.

Щоб використовувати PhishTank, розробники можуть інтегрувати його API в свої системи безпеки або застосунки. API PhishTank дозволяє відправляти URL-

адреси для перевірки на фішингові сайти та отримувати зворотні результати. Він також надає доступ до бази даних фішингових URL-адрес, яка може бути завантажена для локального використання.

2) OpenPhish – повністю автоматизована та вичерпна платформа для збирання інформації про фішинг. Вона ідентифікує фішингові сайти та проводить розвідувальний аналіз в реальному часі без участі людини та без використання зовнішніх ресурсів, таких як чорні списки.

OpenPhish отримує мільйони невідфільтрованих URL-адрес з різних джерел у своїй глобальній партнерській мережі. Фішинговий двигун OpenPhish визначає живі фішингові URL-адреси та витягує їхні метадані, що включають цільові бренди (за потреби), мережеві та географічні місцезнаходження, фішингові набори та облікові записи.

OpenPhish повідомляє тільки про нові та активні фішингові URL-адреси. Ресурс веде облік виявлених фішингових URL-адрес і не повідомляє про той самий домен більше одного разу протягом будь-якого 14-денного періоду. Також сервіс не повідомляє про неактивні фішингові URL-адреси, оскільки вони не становлять загрози.

3) DGArchive – це веб-сервіс, який надає можливість зворотного перегляду доменів для аналізу шкідливих програм та генерації списків доменів для захисту мережі. Він забезпечує можливість перевірки того, чи походить запитаний домен від алгоритму генерації доменів, а також надає списки доменів, які можуть бути використані як списки блокування для захисту мережі. Сервіс підтримується Інститутом інтегральних схем товариства Фраунгофера в Німеччині під наглядом Даніеля Пломана.

DGArchive містить приблизно 93 мільйони унікальних алгоритмів генерації доменів, створених 88 різними сімействами DGA на момент створення. Даний ресурс містить регулярно оновлювані набори даних і технічні відомості для десятків сімейств зловмисного програмного забезпечення.

DGArchive підтримує базу даних заархівованих доменів, створених за допомогою зворотної інженерії шкідливих програм-DGA, і перераховує всі можливі

домени, які може створити певний DGA. Однак DGArchive не охоплює всі зловмисні алгоритми DGA, оскільки величезна кількість нових постійно генерується.

Включення як легітимних, так і зловмисних доменів у навчальний набір моделі машинного навчання забезпечує різноманітність даних і допомагає у збалансуванні класів. Це сприяє покращенню загальної генералізації моделі, її здатності розрізняти складні патерни та більш коректному виявленню шахрайських доменів в реальних умовах.

2.4 Типи класифікації в машинному навчанні

У машинному навчанні термін «класифікація» відноситься до задачі прогнозування, де для заданого зразку вхідних даних передбачається мітка класу.

Приклади задач класифікації включають:

1) для заданого домену з’ясуйте, чи є він легітимним чи ні;

2) для рукописного символу визначте, чи є він одним із раніше відомих;

3) для останньої активності клієнта спрогнозуйте, чи збирається він припинити використовувати послугу.

З точки зору моделювання, класифікація вимагає навчального набору даних з численними прикладами вхідних та вихідних даних для навчання. Модель буде використовувати навчальний набір даних, щоб розрахувати, як краще зіставити приклади вхідних даних з конкретними мітками класу. Таким чином, навчальний набір даних повинен бути достатньо репрезентативним для проблеми і мати багато прикладів кожної мітки класу [26].

Мітки класу часто є рядковими значеннями, наприклад «шахрайський», «легітимний», і перед подачею на моделювання повинні бути відображені в числові значення. Це часто називається кодуванням міток, де кожній мітці класу присвоюється унікальне ціле число, наприклад, при розпізнаванні доменів можна використовувати наступне: «шахрайський» = 1, «легітимний» = 0.

Існує багато різних типів алгоритмів для моделювання задач прогнозування класифікації, проте немає однозначної теорії про те, як накласти моделі на типи

проблем. Зазвичай досліднику рекомендується реалізовувати контрольні експерименти та дізнаватися, яка модель і її конфігурація надає найкращі результати для даної задачі класифікації [26].

Замість міток класу, деякі завдання можуть вимагати прогнозування ймовірності належності до класу для кожного прикладу. Це надає додаткову неоднозначність у прогнозі, яку програма або користувач може інтерпретувати. Популярний діагностичний засіб для оцінки передбачених ймовірностей – це крива похибок.

Розглянемо чотири основних типи завдань класифікації [26]:

1) бінарна класифікація, Binary Classification;

2) багатокласова класифікація, Multi-Class Classification;

3) багатоміткова класифікація, Multi-Label Classification;

3) незбалансована класифікація, Imbalanced Classification.

Бінарна класифікація відноситься до тих завдань класифікації, які мають дві мітки класу. Зазвичай, завдання такої класифікації передбачають один клас, який є нормальним станом, та інший клас, який є аномальним станом. Схематично бінарну класифікацію можна представити на рис. 2.1.

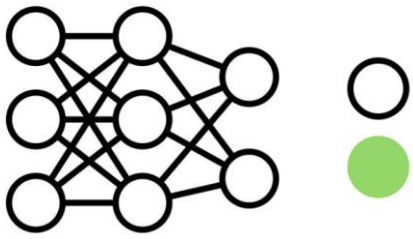

Рисунок 2.1 – Схематичне зображення бінарної класифікації

Наприклад, «легітимний домен» є нормальним станом, а «шахрайський домен», відповідно, – аномальним. Клас для нормального стану отримує класове позначення 0, а клас для аномального стану – класове позначення 1.

Зазвичай бінарне класифікаційне завдання формується за допомогою моделі, яка передбачає бінарний розподіл Бернуллі для кожного прикладу. Розподіл Бернуллі

– це дискретний розподіл ймовірностей, який охоплює випадок, коли подія матиме бінарний результат у вигляді 0 або 1 [26]. Для класифікації це означає, що модель передбачає ймовірність того, що приклад належить до того чи іншого класу.

Багатокласова класифікація відноситься до завдань розпізнавання, які мають більше, ніж дві мітки класу. Приклади включають: класифікація облич, класифікація видів рослин, оптичне виявлення символів. На відміну від бінарної класифікації, у багатокласовій відсутня концепція нормальних та аномальних результатів. Замість цього приклади класифікуються як ті, що належать до одного з численних відомих класів. Кількість міток класу може бути дуже великою для деяких завдань. Наприклад, модель розпізнає фотографію як ту, що належить до однієї з тисяч або десятків тисяч облич у системі [26].

Проблеми, які охоплюють послідовності слів, такі як моделі перекладу тексту, також можуть вважатися спеціальним типом класифікації з багатьма класами. Кожне слово у такій послідовності для передбачення включає класифікацію з багатьма класами, де розмір словника визначає кількість можливих класів, які можуть бути передбачені, і може бути десятки або сотні тисяч слів за розміром.

Зазвичай формується задача багатокласової класифікації з використанням моделі, яка містить багатономіальний розподіл ймовірностей для кожного прикладу. Багатономіальний розподіл – це дискретний розподіл ймовірностей, який охоплює випадок, коли подія матиме категоріальний результат, наприклад, K у {1, 2, 3, ..., K}. Для класифікації це означає, що модель передбачає ймовірність того, що приклад належить до кожної мітки класу [26].

Моделі, призначені для бінарної класифікації, можуть бути адаптовані для використання для багатокласових проблем. Це охоплює реалізацію стратегії встановлення кількох бінарних моделей класифікації для кожного класу проти всіх інших класів (називається «один проти всіх») або однієї моделі для кожної пари класів (називається «один проти одного») [26]. Таким чином, методика «один проти всіх» полягає у встановленні однієї бінарної моделі класифікації для кожного класу проти всіх інших класів. Підхід «один проти одного» включає встановлення однієї бінарної моделі класифікації для кожної пари класів.

Схематично багатокласову класифікацію представимо на рис. 2.2.

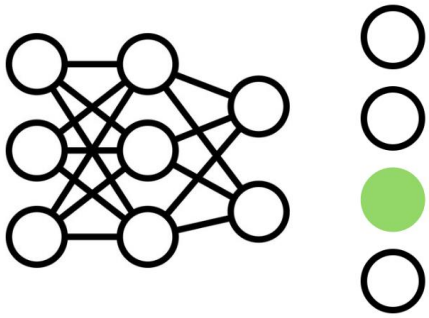

Рисунок 2.2 – Схематичне зображення багатокласової класифікації

Багатоміткова класифікація відноситься до тих завдань, які мають дві або більше міток класу, з яких одна або кілька можуть бути передбачені для кожного прикладу. Розглянемо приклад класифікації фотографій, яка може містити кілька об’єктів, і модель може передбачити наявність кількох відомих предметів, таких як «паркан», «автомобіль», «тварина». Це кардинально відрізняється від бінарної та багатокласової класифікації, де для кожного прикладу призначається одна мітка класу [26]. Зазвичай задачі багатоміткової класифікації формується за допомогою моделі, яка передбачає кілька виходів, кожен з яких вважається передбаченим як бінарний розподіл Бернуллі. Суттєвою є модель, яка генерує кілька бінарних класифікаційних прогнозів для кожного прикладу.

Моделі, що застосовуються для бінарної або багатокласової класифікації, не можуть бути безпосередньо використані для багатоміткової. Є можливість імплементувати спеціалізовані версії стандартних моделей класифікації, так звані багатоміткові версії, включаючи: багатоміткові дерева рішень (Multi-label Decision Trees), багатоміткові випадкові ліси (Multi-label Random Forests), багатоміткове градієнтне підсилення (Multi-label Gradient Boosting). Інший підхід полягає в тому, щоб використовувати окрему модель класифікації для передбачення міток для кожного класу [26].

Схематично багатоміткову класифікацію представимо на рис. 2.3.

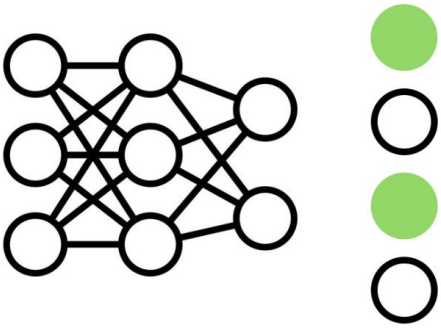

Рисунок 2.3 – Схематичне зображення багатоміткової класифікації

Незбалансована класифікація відноситься до тих завдань, де кількість прикладів у кожному класі розподілена нерівномірно. Зазвичай завдання нерівномірної класифікації відноситься до бінарної, де більшість прикладів у навчальному наборі даних належать до нормального класу, а меншість – до аномального. Приклади включають: виявлення шахрайства та відхилень, медичні діагностичні тести. Ці проблеми моделюються як завдання бінарної класифікації, хоча можуть вимагати спеціалізованих технік [26]. Схематично незбалансовану класифікацію представимо на рис. 2.3.

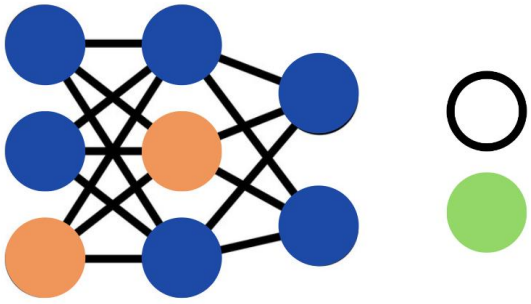

Рисунок 2.4 – Схематичне зображення незбалансованої класифікації

Для зміни композиції вибірок у навчальному наборі даних застосовуються підходи, такі як зменшення вибірок у більшості класу або збільшення вибірок у меншості класу. Такими прикладами є випадкове зменшення вибірок та SMOTE-збільшення вибірок. Може використовуватися спеціалізоване моделювання, яке більше уваги приділяє меншості класу під час підгонки моделі на навчальному наборі даних, наприклад, моделі машинного навчання з урахуванням витрат. Наприклад,

логістична регресія з урахуванням витрат, дерева рішень з урахуванням витрат, метод опорних векторів з урахуванням витрат.

Отже, класифікація є фундаментальним завданням у машинному навчанні, що включає категоризацію даних на попередньо визначені класи або категорії на основі їхніх ознак. При виявленні зловмисних доменів використання класифікації є важливим інструментом для аналізу широкого спектру факторів, таких як структура URL, історія та характеристики веб-сайту, що допомагає в ідентифікації потенційно небезпечних доменів та зменшенні ризиків для користувачів в Інтернеті.

2.5 Теоретичні аспекти архітектур машинного навчання

Опишемо теоретичні концепції моделей машинного навчання, що найчастіше застосовуються в області виявлення доменів DGA. Існуючі методи детектування даних ресурсів на основі ML, як правило, використовують лише доменні імена і можуть бути класифіковані як схеми CNN або RNN. Оскільки домен може бути розглянутий як послідовність символів, деякі дослідження впроваджують RNN, зокрема з архітектурами LSTM. Крім того, в останній час було зроблено кілька спроб додати механізм уваги до LSTM. Таким чином, у цьому розділі зосередимося на CNN, LSTM та механізмі уваги.

2.5.1 Згорткова нейронна мережа

У ML згорткова нейронна мережа передбачає тип глибокого навчання, який спеціально призначений для обробки та розпізнавання зображень. Порівняно з альтернативними архітектурами класифікації, CNN вимагає меншого попереднього оброблення, оскільки може автоматично вивчати ієрархічні характеристики з вихідних зображень. Цей клас відмінно справляється з призначенням важливості для різних об’єктів та характеристик у зображеннях за допомогою згорткових шарів, які застосовують фільтри для виявлення локальних патернів [27].

Зв’язковий патерн у CNN натхненний візуальною корою в людському мозку, де нейрони реагують на конкретні області або рецептивні поля у візуальному просторі. Ця структура дозволяє CNN ефективно захоплювати просторові зв’язки та патерни. Шляхом накладання декількох згорткових та агрегувальних шарів, CNN можуть вивчати все складніші характеристики, що приводить до високої точності у завданнях, таких як виявлення об’єктів та сегментація.

Архітектура CNN складається з трьох основних шарів: згорткових, агрегувальних та повнозв’язаного шарів. Структура шарів мережі CNN представлена на рис. 2.5. У будь-яких випадках може бути кілька згорткових та агрегувальних шарів. Чим більше шарів у мережі, тим складніша (теоретично) і точніша архітектура машинного навчання. Кожен додатковий шар, який обробляє вхідні дані, збільшує здатність розпізнавати об’єкти та патерни у даних [27].

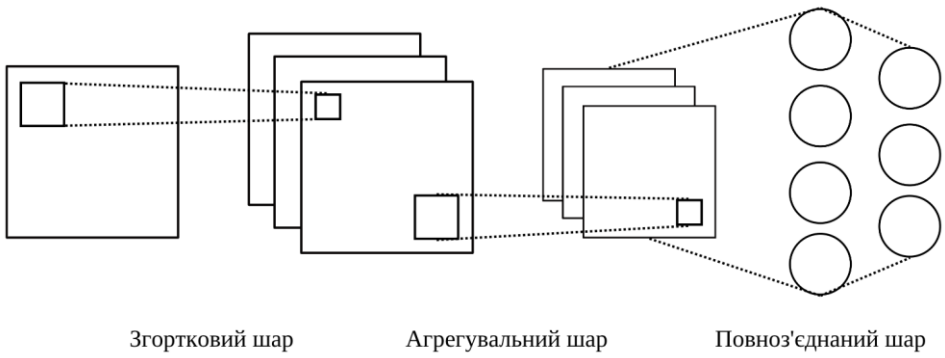

Рисунок 2.5 – Структура шарів мережі CNN

Згорткові шари є ключовим будівельним блоком мережі, де відбувається більшість обчислень. Вони працюють за допомогою застосування фільтра до вхідних даних для виявлення ознак. Фільтр, відомий як детектор ознак, перевіряє рецептивні поля вхідних відомостей на наявність певної ознаки. Ця операція відома як згортка.

Фільтр є двовимірним масивом ваг, який представляє частину двовимірного зображення. Фільтр зазвичай є матрицею розміром 3×3, хоча існують інші можливі розміри. Вік застосовується до області в межах вхідного зображення та розраховує скалярний добуток між пікселями, який подається на вихідний масив. Потім фільтр

зсувається і повторює процес, поки він не охопить всю картину. Кінцевий результат всіх операцій з фільтрами називається картою ознак [28]. Зазвичай CNN застосовує перетворення Rectified Linear Unit (ReLU) до кожної карти ознак після згортки, щоб внести нелінійність в модель машинного навчання. За згортковим шаром зазвичай слідує за агрегувальний шар. Разом згорткові та агрегувальні шари утворюють згортковий блок. Додаткові згорткові блоки будуть слідувати за першим блоком, створюючи ієрархічну структуру, при цьому наступні шари навчатимуться на основі раніше навчених. Наприклад, CNN може навчатися виявляти автомобілі на зображеннях. Автомобілі можна розглядати як суму їхніх складових, включаючи колеса, багажник та переднє скло. Кожна ознака автомобіля відповідає низькорівневому патерну, виявленому нейронною мережею, яка потім поєднує ці частини для створення високорівневого патерну [28].

Агрегувальний або зменшувальний шар зменшує розмірність введення. Подібно до операції згортки, агрегувальні операції використовують фільтр для проходження по всьому вхідних відомостях, але не для ваг. Замість цього фільтр застосовує функцію агрегації для заповнення вихідного масиву на основі значень рецептивного поля. Існують два основних типи агрегування [28]:

1) середнє: фільтр обчислює середнє значення рецептивного поля під час сканування вхідних даних;

2) максимальне: фільтр обирає піксель з максимальним значенням для заповнення вихідного масиву, тому цей підхід більш поширений.

Шари агрегування є необхідними, незважаючи на втрату деякої інформації, оскільки вони зменшують складність і підвищують ефективність CNN. Вони також зменшують ризик перенавчання.

Останній шар згорткової нейронної мережі – це повнозв’язаний шар, що виконує завдання класифікації за допомогою ознак, які витягли попередні шари та фільтри. Замість функцій ReLU, повнозв’язаний шар зазвичай використовує функцію softmax, яка класифікує вхідні дані більш адекватно та генерує ймовірнісний бал між значеннями 0 і 1.

2.5.2 Мережа довгострокової короткочасної пам’яті

Довгострокова короткочасна пам’ять – це різновид рекурентних нейронних мереж, спеціально розроблений для роботи з послідовними даними. Архітектура LSTM розв’язує проблему зникаючого градієнта, властиву традиційним RNN, за допомогою впровадження блоків пам’яті та вентилів для керування потоком інформації, а також завдяки унікальній структурі. LSTM широко використовується в глибокому навчанні, оскільки вона здатна захоплювати довгострокові залежності в послідовних даних. Це робить її придатними для таких завдань, як розпізнавання мовлення, переклад мов та прогнозування часових рядів, де контекст попередніх точок даних може впливати на наступні.

На рис. 2.6 представлена структура LSTM, де представлено такі параметри: X_t – вхідний крок часу, h_t – вихідний стан, C_t – стан комірки пам’яті, f_t – забувальний вентиль, i_t – вентиль входу, O_t– вентиль виходу, $\hat{C}_t$ – внутрішній стан комірки пам’яті. Операції всередині світло-червоного кола є поелементними.

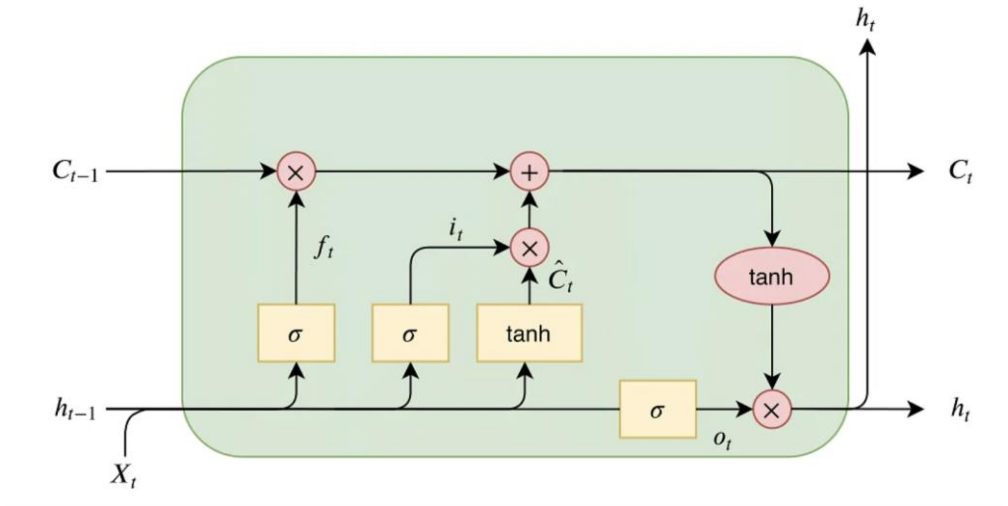

Рисунок 2.6 – Структура LSTM

У структурі LSTM на результат у конкретний момент часу впливає на три фактори [29]:

1) стан комірки, який представляє поточну довгострокову пам’ять мережі;

2) попередній прихований стан, який посилається на вихід попереднього кроку часу;

3) вхідні дані, які присутні на поточному кроці часу.

Нейронні мережі з довгостроковою короткочасною пам’яттю використовують серію воріт для регулювання потоку інформації в послідовності даних. Вентилі забуття, вводу та виводу служать фільтрами і діють як окремі нейронні мережі всередині мережі LSTM. Вони керують процесом того, як інформація вводиться в мережу, зберігається та в кінцевому підсумку виводиться [29].

Перший етап в архітектурі – Forget Gate, вентиль забуття. На цьому етапі нейронна мережа LSTM визначатиме, які елементи стану комірки (довгострокової пам’яті) є релевантними на основі попереднього прихованого стану та нових вхідних даних, рис. 2.7.

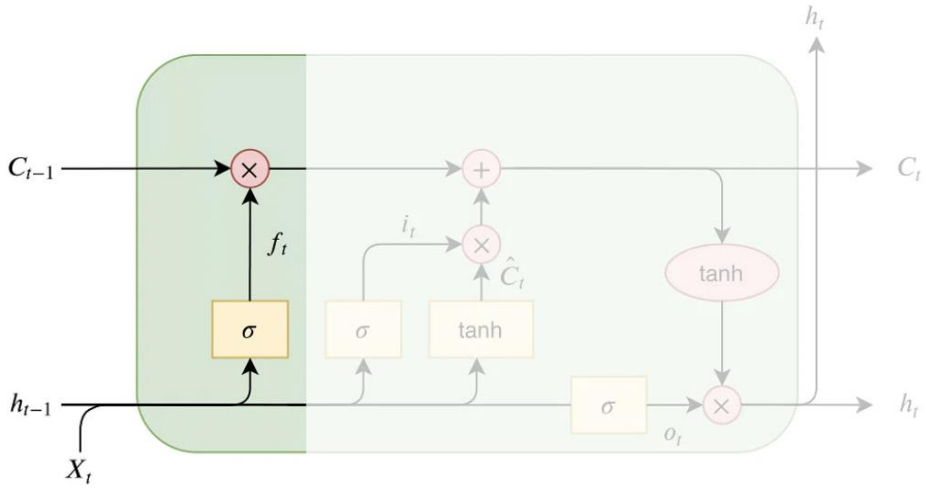

Рисунок 2.7 – Вентиль забуття архітектури LSTM

Попередній прихований стан (h_{t-1}) та нові вхідні дані (X_t) подаються на вхід нейромережі, яка виводить вектор, де кожен елемент являє собою значення від 0 до 1. Це досягається за допомогою сигмоїдної функції активації. Ця мережа всередині воріт забуття навчена виробляти значення близькі до 0 для інформації, яка вважається несуттєвою, і близькі до 1 – для важливої. Елементи цього вектора можна розглядати як фільтри, що дозволяють пропускати більше інформації, коли значення наближається до 1 [29]. Потім ці вихідні значення покомпонентно множаться на

попередній стан комірки (C_{t-1}). У результаті несуттєві частини стану комірки зменшують свою вагу завдяки множенню на значення близькі до 0, що знижує їхній вплив на наступні кроки. Загалом вентиль забуття визначають, які частини довгострокової пам'яті слід забути, враховуючи попередній прихований стан та нові вхідні дані в послідовності. Обчислення щодо частини інформації, яку потрібно видалити з комірки пам'яті, представлено через формулу 2.5.

$$f_t = \sigma(W_f \bullet [h_{t-1}, X_t] + b_f) \tag{2.5}$$

Input Gate, вхідний вентиль – це нейронна мережа, яка використовує сигмоїдну функцію активації та служить як фільтр для визначення цінних компонентів нового вектора пам'яті. Архітектура вхідного вентилю представлена на рис. 2.8.

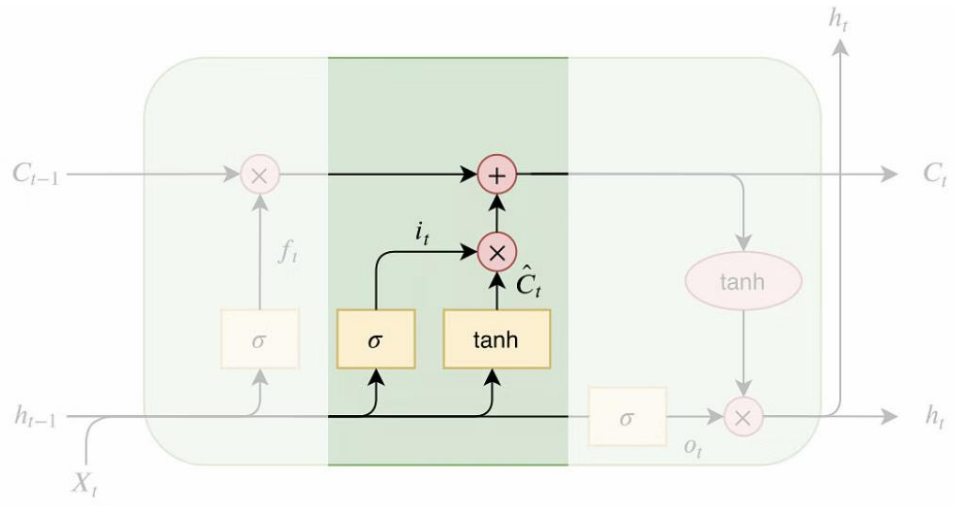

Рисунок 2.8 – Вхідний вентиль архітектури LSTM

Він виводить вектор значень у діапазоні [0, 1] в результаті сигмоїдної активації, що дозволяє йому функціонувати як фільтр через покомпонентне множення. Подібно до вентилю забуття, низьке вихідне значення з вхідного вентилю означає, що відповідний елемент стану комірки не повинен оновлюватися [29]. Обчислення щодо частини інформації, яка повинна бути оновлена в стані комірки, представлено через формулу 2.6.

$$i_t = \sigma(W_i \bullet [h_{t-1}, X_t] + b_i) \quad (2.6)$$

Нова мережа пам'яті – це нейромережа, яка використовує функцію активації *tanh* і навчена створювати «вектор оновлення пам'яті» шляхом комбінування попереднього прихованого стану та поточних вхідних даних. Цей вектор несе інформацію з вхідних даних та враховує контекст, наданий попереднім прихованим станом. Він визначає, наскільки кожен компонент довгострокової пам'яті (стану комірки) необхідно скоригувати на основі останніх даних. Процес створення кандидатського вектора для оновлення стану комірки в LSTM мережі представлено формулою (2.7).

$$\hat{C}_t = tanh(W_C \bullet [h_{t-1}, X_t] + b_C) \quad (2.7)$$

Функція активації tanh використовується, оскільки її значення знаходиться у діапазоні [-1, 1]. Здатність генерувати негативні значення є критичною для зменшення впливу певного компонента стану комірки.

Створений на цьому етапі вектор оновлення пам'яті не визначає, чи варто запам'ятовувати нові вхідні дані, саме тому також потрібен вхідний вентиль. Кінцевий результат комбінації оновлення пам'яті та фільтра вхідного вентиля використовується для оновлення стану комірки, який є довгостроковою пам'яттю мережі LSTM. Вихід оновлення пам'яті регулюється фільтром вхідного вентиля через покомпонентне множення, що означає, що до стану комірки додаються лише релевантні компоненти оновлення пам'яті [29].

Оновлений стан комірки представляє оновлену довгострокову пам'ять мережі. Внутрішній стан оновлюється за формулою (2.8).

$$C_t = i_t \bullet \hat{C}_t + f_c \bullet C_{t-1} \quad (2.8)$$

На завершальному етапі LSTM новий прихований стан визначається з використанням новооновленого стану комірки, попереднього прихованого стану та

нових вхідних даних. Даний процес прийняття рішень здійснює вихідний вентиль, структура якого представлена на рис. 2.9.

Цей вентиль використовується для визначення кінцевого прихованого стану мережі LSTM. На цьому етапі в ролі вхідних даних є оновлений стан комірки, попередній прихований стан та нові вхідні дані. Просте виведення одного оновленого стану комірки призвело б до розкриття занадто великої кількості інформації, тому реалізовано фільтр – вихідний вентиль.

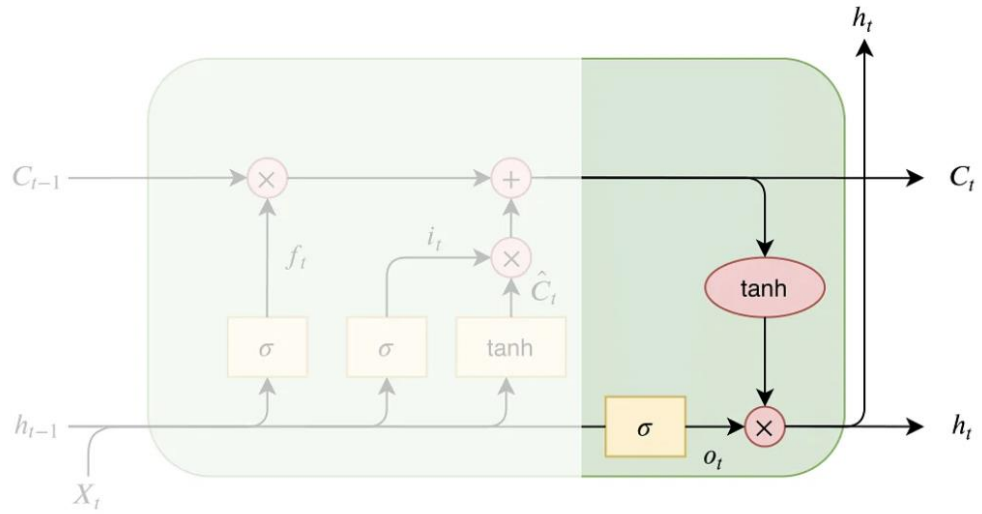

Рисунок 2.9 – Вихідний вентиль архітектури LSTM

Вихідний вентиль – це мережа з сигмоїдною активацією, яка діє як фільтр і вирішує, які елементи оновленого стану комірки є релевантними і повинні бути виведені як новий прихований стан. Вхідні дані для вихідного фільтру такі самі, як і для попереднього прихованого стану та нових даних, і застосовується сигмоїдна активація, щоб виробити вихідні значення в діапазоні [0,1].

Обчислення щодо частини стану комірки, яка буде використана для формування кінцевого прихованого стану, представлена у формулі (2.9).

$$o_t = \sigma(W_o \bullet [h_{t-1}, X_t] + b_o) \tag{2.9}$$

Потім оновлений стан комірки проходить через активацію tanh, щоб обмежити його значення до [-1,1] перед тим, як бути помноженим покомпонентно на вихід мережі вихідного вентилю, щоб створити кінцевий новий прихований стан. Оновлення прихованого стану у LSTM мережі визначено формулою (2.10).

$$h_t = o_t \times tanh\,(C_t) \tag{2.10}$$

Комірка LSTM використовує матриці ваг і зсуви в поєднанні з оптимізацією на основі градієнта для навчання своїх параметрів. Ці параметри пов’язані з кожним вентилем, як і в будь-якій іншій нейронній мережі. Матриці ваг можна ідентифікувати як W_f, b_f, W_i, b_i, W_o, b_o та W_C, b_C відповідно до вищенаведених рівнянь [29].

Підсумовуючи, останній крок визначення нового прихованого стану полягає в тому, щоб пропустити оновлений стан комірки через функцію активації tanh для отримання стиснутого стану комірки в діапазоні [-1, 1]. Потім попередній прихований стан та поточні вхідні дані пропускаються через мережу з сигмоподібною активацією для отримання вектора фільтра. Він потім покомпонентно множиться на стиснений стан комірки, щоб отримати новий прихований стан, який є виходом цього етапу. Вихідний вентиль є останнім етапом у комірці LSTM, і це лише одна частина повного процесу. Залежність комірок в архітектурі LSTM представлено на рис. 2.10.

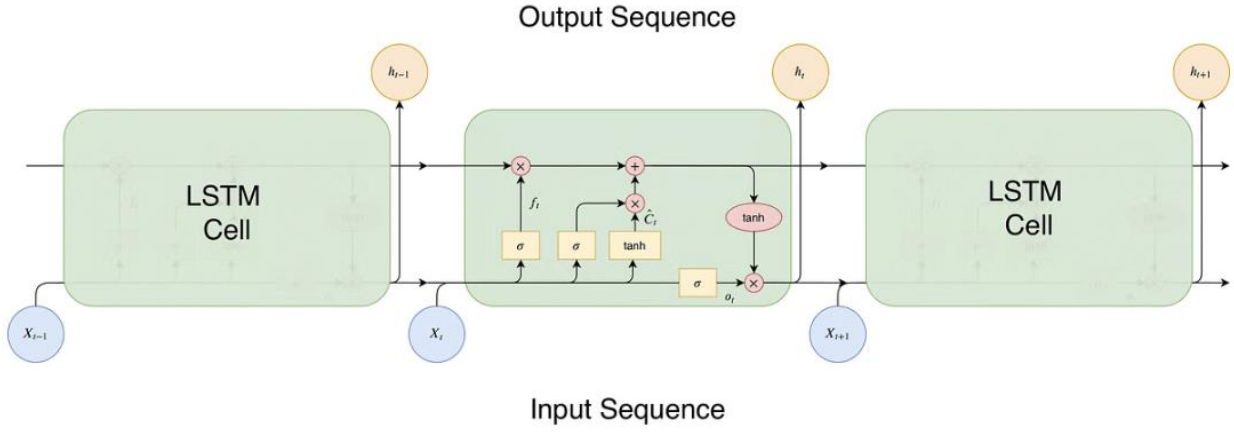

Рисунок 2.10 – Залежність комірок в архітектурі LSTM

Перед тим, як мережа LSTM може створити бажані прогнози, є ще кілька речей, які потрібно врахувати. У випадку ідентифікації шахрайських доменів на основі ознак зібраних з даних за останні 30 днів, то кроки в комірці LSTM повторюватимуться 30 разів. Це означає, що LSTM ітеративно створила б 30 прихованих станів, щоб ідентифікувати потенційно небезпечні домени.

Потік інформації в LSTM відбувається у рекурентний спосіб, формуючи структуру, схожу на ланцюг. Потік останнього виходу комірки до кінцевого стану додатково контролюється вихідним вентилем. Однак вихід комірки LSTM все ще є прихованим станом. Для перетворення прихованого стану в бажаний вихід, на останньому етапі процесу LSTM застосовується лінійний шар. Цей етап відбувається лише один раз, наприкінці, і він не включений в діаграми комірки LSTM, оскільки виконується після повторюваних кроків комірки LSTM [30].

Розгортання LSTM в часі означає процес розширення мережі LSTM на послідовність часових кроків. У цьому процесі мережа LSTM по суті дублюється для кожного такого кроку, а виходи з одного подаються в мережу як входи для наступного. Даний процес розгортання можна використовувати для навчання нейронних мереж LSTM на часових рядах, де метою є передбачення наступного значення в послідовності на основі попередніх. Розгортаючи мережу LSTM таким чином, мережа здатна вивчити довгострокові залежності та вловити патерни в часових рядах.

Двонаправлена довгострокова короткочасна пам’ять, Bidirectional Long Short-Term Memory (BiLSTM) – це розширення традиційної архітектури LSTM, яке включає двонаправлену обробку для покращення здатності захоплювати контекстуальну інформацію як з минулих, так і майбутніх вхідних даних [30]. Впроваджена як удосконалення однонаправлених LSTM, BiLSTM особливо ефективні в завданнях, де розуміння контексту послідовності в обох напрямках є критичним, таких як обробка природної мови та розпізнавання мови.

Структура BiLSTM, рис. 2.11, включає два окремі шари LSTM: один обробляє вхідну послідовність від початку до кінця (прямий LSTM), а інший – у зворотному порядку (зворотний LSTM). Виходи з обох напрямків конкатенуються на кожному

часовому кроці, забезпечуючи всеосяжне представлення, яке враховує інформацію як з попередніх, так і з наступних елементів послідовності.

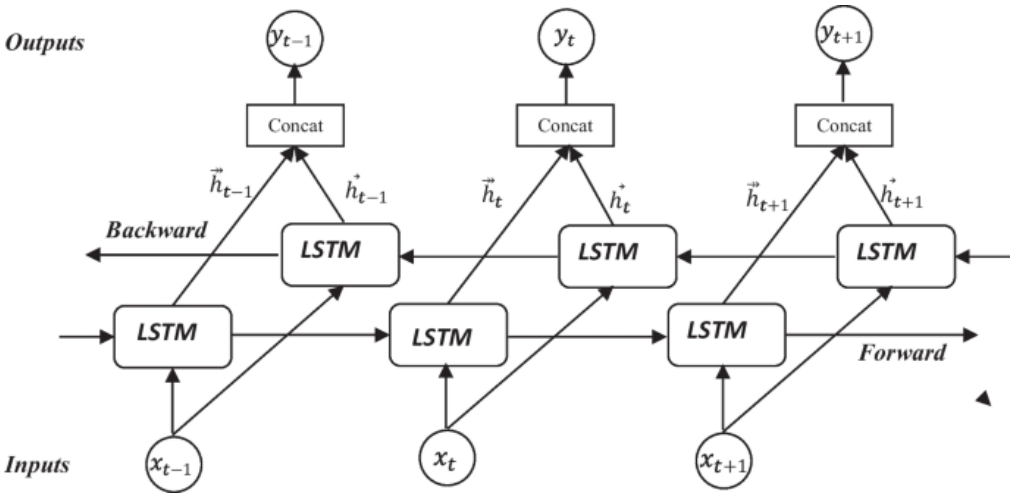

Рисунок 2.11 – Структура BiLSTM

Такий двонаправлений підхід дозволяє BiLSTM захоплювати більш багатий контекстуальний зв’язок, фіксувати довгострокові залежності й робити більш обґрунтовані прогнози [30].

Завдяки обробці послідовностей в обох напрямках, BiLSTM відмінно справляються з такими завданнями, як розпізнавання іменованих сутностей, аналіз емоційного забарвлення та машинний переклад, де для розуміння контексту слова або фрази необхідно враховувати як його минулий, так і майбутній контекст. Двонаправлена природа BiLSTM робить їх універсальними та добре придатними для дослідження послідовних даних.

BiLSTM широко використовуються в завданнях обробки природної мови, включаючи розмітку частин мови, розпізнавання іменованих сутностей та зчитування емоційного забарвлення. Їх також застосовують у розпізнаванні мовлення, де двонаправлена обробка допомагає захоплювати важливу фонетичну та контекстну інформацію. Крім того, BiLSTM впроваджуються у прогнозування часових рядів та аналізі біомедичних даних, де врахування інформації з минулого та майбутнього є критичним для точних прогнозів.

2.5.3 Механізм уваги

Механізм уваги – це архітектура нейромережі типу «кодер-декодер», яка дозволяє моделі зосереджуватися на певних сегментах вхідних даних під час виконання завдання. Він динамічно призначає ваги різним елементам, вказуючи на їхню відносну важливість або релевантність. Завдяки увазі модель може вибірково звертати увагу та обробляти найнеобхіднішу інформацію, фіксуючи залежності та зв'язки всередині даних [31]. Цей механізм є особливо цінним для задач, що включають послідовні або структуровані дані, такі як обробка природної мови чи комп'ютерний зір, оскільки він дозволяє моделі ефективно обробляти довгострокові залежності та покращувати продуктивність за рахунок вибіркового звернення уваги на визначальні ознаки або контексти.

Застосування механізму уваги до генерації підписів до зображень суттєво підвищило якість та точність створених описів. Завдяки цьому модель під час генерування кожного слова навчається фокусуватися на відповідних областях. Даний механізм дозволяє поетапно синхронізувати візуальні та текстові модальності, аналізуючи різні ділянки зображення [31]. Зосереджуючись на важливих об'єктах, модель може генерувати більш детальні та контекстуально доречні підписи. Моделі генерації підписів до зображень на основі уваги, як доведено, краще справляються із захопленням дрібних деталей, обробкою складних сцен та створенням зв'язних та інформативних підписів, що відповідають візуальному матеріалу.

Механізм уваги можна математично виразити рівняннями (2.11-2.14).

$$s_i = f(s_{i-1}, y_{i-1}, c_i)\,, \tag{2.11}$$

$$c_i = \sum_{j=1}^{T_x} a_{ij}\, h_y)\,, \tag{2.12}$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})})\,, \tag{2.13}$$

$$e_{ij} = a(s_{i-1}, h_j) = v_a^T \tanh(W_a[s_{i-1}, h_j])\,. \tag{2.14}$$

Спочатку кодер отримує вхідну послідовність x_1, x_2, ..., x_T та генерує відповідний вектор прихованого стану h_i. Потім, як представлено в формулі (2.11), поточний прихований стан s_i декодера обчислюється як попередній вектор прихованого стану s_{i-1}, попереднє вихідне значення декодера y_{i-1} та поточний контекстний вектор c_i У цей час контекстний вектор розраховується через зважену суму значень уваги a_{ij} кодера, як наведено в рівнянні (2.12). Тут Y_x – це довжина вхідного слова кодера. Це означає, що декодер враховує всі вхідні дані кодера, як відображено на рис. 2.12.

У формулі (2.13) ймовірність уваги обчислюється як нормалізована функція softmax від оцінки уваги e_{ij}, яка є скалярним значенням, що вказує на те, наскільки схожий попередній вектор прихованого стану s_{i-1} до j-го вектора h_j кодера. Нарешті формула (2.14) обчислює фактичну схожість за допомогою tanh, де v та W є параметрами навчання для застосування уваги.

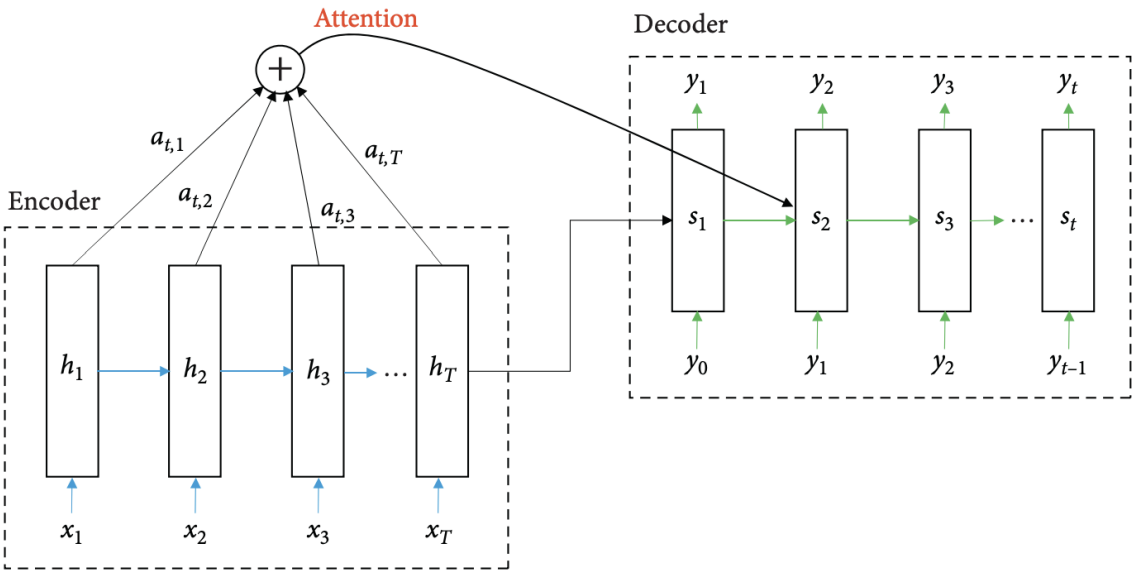

Рисунок 2.12 – Загальна архітектура механізму уваги

Даним способом механізм уваги зберігає контекст кожного слова в реченні, присвоюючи вагу уваги у відношенні до всіх інших слів. Таким чином, навіть якщо речення велике, модель може зберегти контекстну важливість кожного слова. Загалом, механізм уваги є потужним інструментом для покращення продуктивності

моделей послідовностей. Дозволяючи їм зосереджуватися на найбільш релевантній інформації, підхід підвищує точність прогнозів та робить модель більш ефективною, обробляючи лише найважливіші дані [31].

Самоувага – це тип механізму уваги, який став ключовим компонентом у машинному навчанні, зокрема, в обробці природної мови. Підхід дозволяє моделі аналізувати і інтерпретувати кожен елемент вхідної послідовності окремо, приділяючи увагу тим частинам, які найбільш значущі для виконання конкретного завдання [32.

Особливість самоуваги полягає в її здатності виявляти не тільки явні, але й приховані залежності між елементами послідовності, що робить її незамінною в таких задачах, як машинний переклад, де необхідно зрозуміти граматичні та контекстуальні взаємозв'язки в різних мовах. Вона також критично важлива для розпізнавання емоцій, де модель повинна точно ідентифікувати та інтерпретувати суб'єктивні тони і настрої в тексті, а також у системах відповідей на запитання, де потрібно витягувати відповідні відомості з великих обсягів даних [32].

Завдяки своїй здатності до паралельної обробки даних і збереження основної інформації на всіх етапах обробки, самоувага значно покращує ефективність і точність моделей машинного навчання. Це стало можливим завдяки зменшенню необхідності великих обсягів навчальних даних та забезпеченню високої міри гнучкості та адаптивності, тому самоувага позиціонується як провідний інструмент у сучасних дослідженнях та розробках.

2.6 Способи формування ансамблю

Оскільки сфера ML постійно розвивається, прагнення до підвищення точності прогнозування призвело до втілення методів ансамблювання. У контексті ML ансамбль – це група з декількох моделей, що працюють разом для здійснення прогнозування або класифікації. Ідея полягає в тому, щоб об'єднати сильні сторони окремих моделей для покращення загальної продуктивності та зменшення помилок.

Агрегуючи виходи декількох моделей, ансамбль часто досягає кращих показників прогнозування, ніж будь-яка окрема з них у його складі.

Концепцію навчання з використанням ансамблів можна представити за допомогою простої аналогії. Наприклад, для вирішення складної проблеми запросили декількох експертів, кожен з яких має свою область знань та підхід до вирішення проблем. Запитавши в кожного експерта рішення, а потім об'єднавши всі пропозиції, кінцеве розв'язання, швидше за все, було б більш точним і всебічним, ніж під час покладання на одного науковця. У машинному навчанні кожен «експерт» – це прогнозна модель, навчена на даних. Існує декілька методів створення ансамблів, які можна умовно поділити на три типи: бегінг (bagging), бустинг (boosting) та стекінг (stacking).

Бегінг, також відомий як агрегація бутстреп вибірок, – це техніка навчання ансамблю, що поєднує переваги бутстрепу та агрегування, аби створити стабільну модель і покращити її продуктивність прогнозування. Під час бегінгу спочатку обираються підмножини даних однакового розміру з датасету за допомогою бутстрепу, тобто вибірки з поверненням. Потім ці підмножини використовуються для незалежного навчання кількох слабких моделей [33]. Слабка модель – це та, що має низьку точність прогнозування. Відповідно, сильні є дуже точними. Для отримання сильної моделі агрегують прогнози з усіх слабких, як це представлено на рис. 2.13.

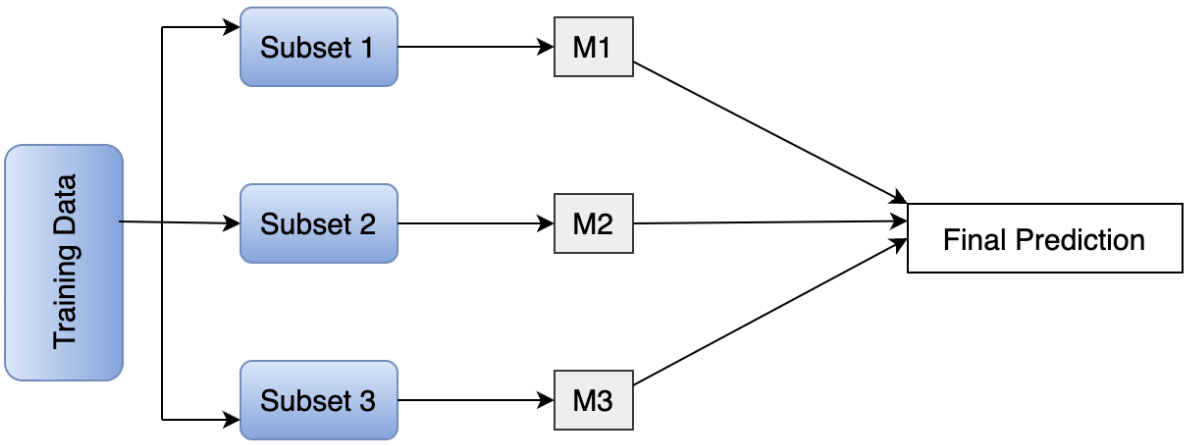

Рисунок 2.13 – Механізм методу ансамблювання «бегінг»

Отже, бегінг передбачає три кроки [33]:

1) відбір підмножин однакового розміру з поверненням;

2) незалежне та паралельне навчання слабких моделей на кожній з підмножин;

3) комбінування результатів з кожної зі слабких моделей шляхом усереднення або голосування для отримання кінцевого результату.

Результати агрегуються шляхом усереднення результатів для задач регресії або шляхом вибору класу з більшістю голосів у задачах класифікації.

Для бустинга відбувається навчання послідовності моделей, де кожна з них навчається на зваженій тренувальній вибірці. Ваги призначаються відповідно до помилок попередніх моделей у послідовності. Основна ідея послідовного навчання полягає в тому, щоб кожна модель виправляла помилки свого попередника. Це продовжується до тих пір, поки не буде навчено попередньо визначену кількість моделей або не будуть виконані інші критерії [33].

Під час навчання екземплярам, які класифіковано неправильно, призначаються вищі ваги, щоб надати їм деяку форму пріоритету при навчанні з наступною моделлю. Крім того, слабким моделям призначаються нижчі ваги, ніж сильним, коли їхні прогнози комбінуються в кінцевий результат. Схематично підхід бустинга представлено на рис. 2.14.

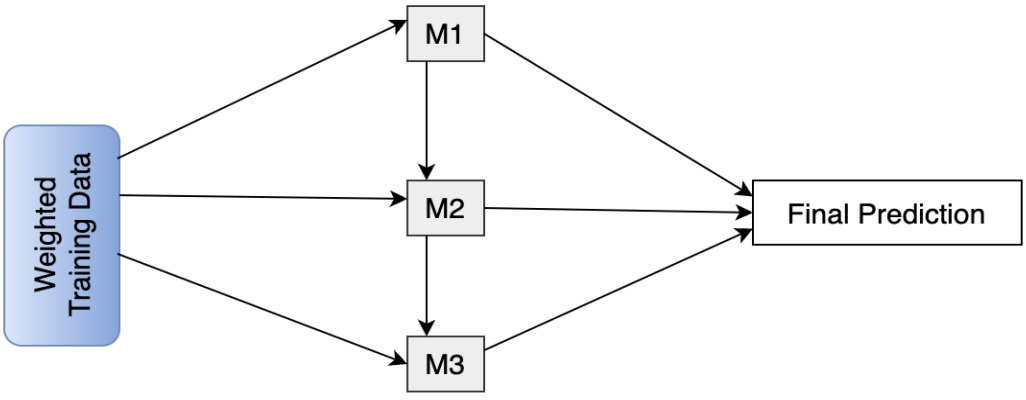

Рисунок 2.14 – Механізм методу ансамблювання «бустинг»

Отже, спочатку ініціалізуються ваги даних до однакового значення, а потім ітеративно виконуються наступні кроки [33]:

1) навчання моделі на всіх інстансах;

2) розрахунок помилок на виході моделі по всіх інстансах;

3) призначення ваги моделі (більша за високу продуктивність і навпаки);

4) оновлення ваг даних: вищі ваги зразкам з великими помилками;

5) повторення попередніх кроків, якщо продуктивність не є задовільною або виконані інші умови зупинки.

Нарешті, запускається комбінація моделі в одну, яку використовують для прогнозування.

Для стекінгу характерно те, що прогнози базових моделей подаються як вхідні дані для метамоделі (або мета-вчителя). Її завдання полягає в тому, щоб прийняти прогнози базових моделей і надати остаточний висновок.

Базові та метамодель не обов'язково повинні бути одного типу. Ключові кроки включають [33]:

1) побудову базових моделей на різних частинах тренувальних даних;

2) навчання метамоделі на прогнозах з базових моделей.

Схематично стекінг можна представити за допомогою рис. 2.15.

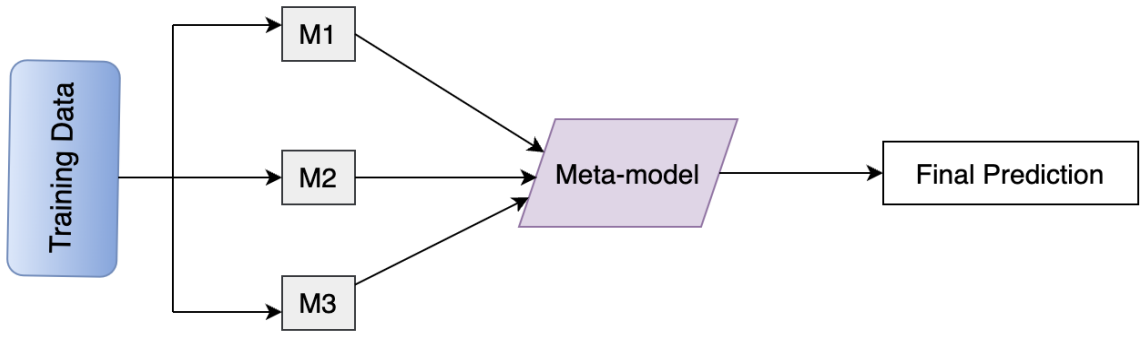

Рисунок 2.15 – Механізм методу ансамблювання «стекінг»

Основні відмінності між бегінгом, бустингом і стекінгом полягають у підході, базових моделях, виборі підмножини, цілях і їхніх комбінаціях. Охарактеризуємо їх детальніше у табл. 2.2, аби дослідити унікальні особливості, що впливають на використання в різних ситуаціях.

Таблиця 2.2

Порівняння методів формування ансамблю

Критерій	Бегінг	Бустинг	Стекінг
Підхід	Паралельне навчання слабких моделей	Послідовне навчання слабких моделей	Поєднання прогнозів моделей у метамодель
Базові моделі	Однорідні (розподіляє набір даних по різних вузлах)	Однорідні (розподіляє набір даних по різних вузлах)	Може бути гетерогенним (поєднує результат різних методів відбору ознак)
Вибір підмножин	Випадкова вибірка із заміною	Немає підмножин	Немає підмножин
Ціль	Зменшити дисперсію	Зменшити упередженість	Зменшити дисперсію та упередженість
Комбінація моделей	Мажоритарне голосування або усереднення	Голосування зваженою більшістю або усереднення	Використання моделі ML

Методи ансамблю, безумовно, пропонують переваги у сфері досліджень з машинного навчання, дозволяючи розробляти складні моделі та отримувати результати з високою точністю. Проте в галузях, де пріоритетною є інтерпретованість, їхнє застосування часто поступається більш зрозумілим підходам. Разом з тим, ефективність розглянутих підходів не підлягає сумніву, а їхні переваги за належного використання поліпшать результати.

Висновок до другого розділу

Машинне навчання не тільки є стратегічним інструментом у боротьбі з кіберзлочинністю, але й відкриває нові горизонти в автоматизації виявлення зловмисних доменних імен, створених за допомогою алгоритмів DGA. Архітектури

машинного навчання, такі як CNN, BiLSTM та механізм самоуваги, характеризуються сильними сторонами, що дозволять ідентифікувати шахрайські домени, значно підвищуючи рівень кібербезпеки в компаніях. У розділі було детально розглянуто кожну з них з метою їхньої подальшої інтеграції для побудови гібридної моделі, щоб детектувати зловмисні домени з вищою точністю.

Зокрема, було підкреслено, що CNN ефективні в розпізнаванні візуальних патернів в доменних іменах, тоді як BiLSTM оптимізовані для роботи з послідовностями даних, виявляючи складні залежності в структурі доменних імен. Механізм самоуваги, у свою чергу, забезпечує здатність моделі зосереджуватися на найбільш значущих частинах даних, що веде до кращого розуміння складної структури DGA доменів. Це дозволяє моделям не тільки точніше виявляти вже відомі зловмисні домени, але й адаптуватися до нових та еволюціонуючих патернів, що робить їх незамінними в сучасній системі кібербезпеки. Таким чином, інтеграція описаних підходів суттєво посилить здатність до ідентифікації шахрайських дій в інтернет-просторі, забезпечуючи більш надійний захист від кіберзагроз.

РОЗДІЛ 3
РОЗРОБКА ГІБРИДНОЇ МОДЕЛІ ДЛЯ ВИЯВЛЕННЯ ДОМЕНІВ DGA

3.1 Концептуальне обґрунтування гібридної моделі

У даній роботі представлено гібридний підхід до класифікації доменних імен на основі методів бегінгу та стекінгу. Моделями-кандидатами для ансамблювання слугують CNN та BiLSTM з механізмом самоуваги. Така стратегія дозволяє ефективно залучити переваги обох моделей: CNN навчається локальних закономірностей послідовностей символів вхідного доменного імені, тоді як BiLSTM здатний враховувати більш глобальні контекстні залежності [33].

Запропонований спосіб реалізації моделі передбачає агрегацію проміжних значень моделей-кандидатів за допомогою повнозв'язаного шару. Таким чином, цей шар буде виконувати подвійну функцію: з одного боку, він реалізує агрегацію, як це робиться в методі бегінгу, а з іншого – слугує наступною моделлю в методі стекінгу. Концептуальна структура запропонованої моделі ансамблювання детально представлена на рис. 3.1.

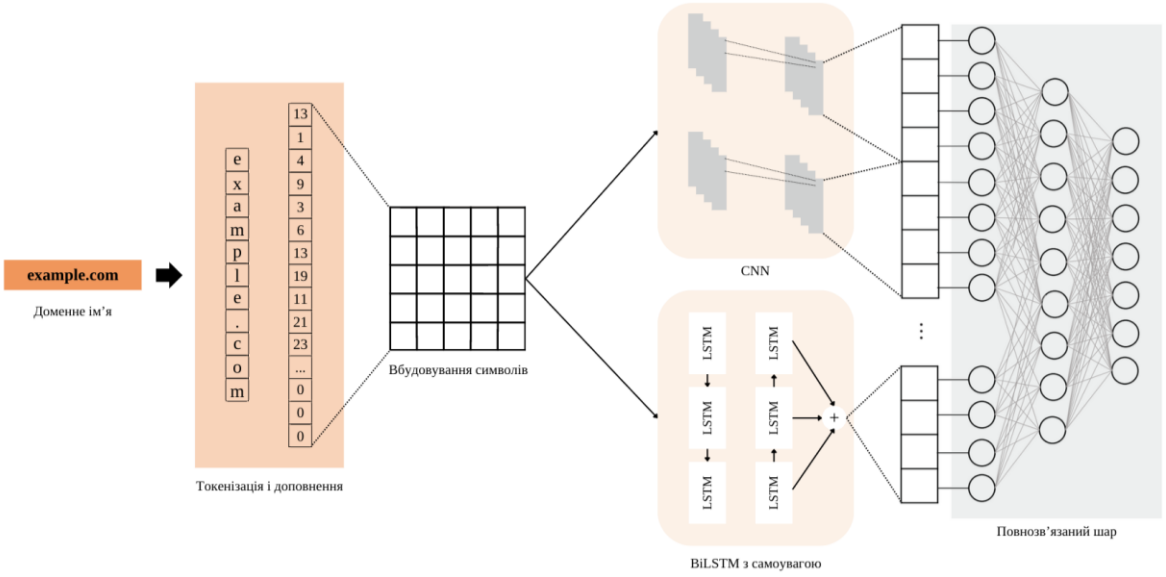

Рисунок 3.1 – Концептуальна структура гібридної моделі

На першому етапі здійснюється попередня обробка та отримання векторного представлення (embedding) вхідного доменного імені. Після цього застосовуються шар CNN та BiLSTM з шаром уваги. На стадії CNN виконуються чотири 1D-згорткові операції з різними розмірами фільтрів (від 2 до 5) та операція максимального об'єднання. Отримані на цьому етапі локальні ознаки доменного імені подаються на вхід повнозв'язаного шару, який виконує подальшу обробку та агрегацію даних згідно з обраним методом ансамблювання.

Додатково шар BiLSTM з шаром уваги складається з власне шару BiLSTM та шару уваги, до якого застосовується механізм самоуваги. Після конкатенації виходів шарів CNN та BiLSTM з увагою, клас DGA-домену визначається за допомогою повнозв'язаного шару.

Запропонована модель ансамблю навчається в наскрізному режимі, використовуючи проміжне значення кожної моделі-кандидата як вхід повнозв'язаного шару, а не навчаючи їх окремо, як у застарілих прикладах ансамблю. Таким чином, даний метод зможе краще враховувати особливості виходів шарів CNN та BiLSTM з увагою.

3.2 Вибір інструментів для реалізації моделі

Для реалізації зазначеної вище описаної концепції було прийнято рішення використовувати мову програмування Python, а сам процес розробки проходив у рамках інтегрованого середовища розробки PyCharm.

Python є ідеальним вибором для створення моделей машинного навчання, які виявляють шахрайські доменні імена, особливо коли впроваджуються складні архітектури, такі як CNN і BiLSTM з механізмом самоуваги. Перш за все, Python має багату екосистему наукових і технічних бібліотек, таких як TensorFlow, PyTorch, і Keras, які дозволяють гнучко та ефективно проектувати, тестувати та реалізовувати моделі ML. Ці інструменти вже включають передові функції для реалізації механізмів уваги, що є ключовим для завдань, які потребують врахування контекстуальної

важливості вхідних даних, як це потрібно при перевірці доменних імен на предмет шахрайства.

Крім того, Python дозволяє легко обробляти і маніпулювати даними завдяки бібліотекам Pandas та NumPy, що спрощує підготовку та передобробку великих наборів даних, типових для Інтернет-ресурсів. Ця здатність швидко обробляти та аналізувати дані дозволяє моделям краще навчатися і давати більш точні прогнози. Також Python підтримує паралельні обчислення і можливість використання GPU через бібліотеки як CUDA, що критично для тренування складних моделей з великими наборами даних. Це прискорює процес навчання, забезпечуючи більшу продуктивність та можливість вчасно вносити зміни. Завдяки цим особливостям, Python забезпечує не тільки потужні технічні можливості для створення і тестування машинно-навчальних моделей, але й гнучкість і швидкість у розробці складних систем, таких як виявлення шахрайських доменних імен.

PyCharm необхідний в розробці моделей машинного навчання завдяки своїй здатності спрощувати та оптимізувати процес програмування. Це IDE інтегрує провідні інструменти для розробки на Python і забезпечує багатий набір функціональностей, які роблять кодування більш інтуїтивним. Вбудовані засоби для управління бібліотеками та залежностями дозволяють легко налаштувати і підтримувати робоче середовище, значуще для роботи з такими складними моделями, як CNN і BiLSTM з механізмом уваги.

Крім того, вбудований дебагер і профайлер допомагають виявити та виправити помилки, а також сконфігурувати продуктивність моделі, що є фундаментальним при роботі з великими даними і складними моделями. Тому PyCharm значно полегшує процес розробки та тестування, забезпечуючи потужні засоби для створення ефективних моделей для виявлення шахрайства в онлайн-просторі.

Для реалізації моделі було застосовано TensorFlow – бібліотеку програмного забезпечення з відкритим кодом, створену Google, яка використовується для реалізації систем машинного навчання. В основі TensorFlow лежить бібліотека для програмування потоку даних. Вона містить різні техніки оптимізації, щоб спростити

та підвищити продуктивність обчислення математичних виразів. Основні особливості TensorFlow включають:

1) оптимальну роботу з математичними виразами з багатовимірними масивами;

2) якісну підтримку концепцій ML;

3) обчислення на GPU/CPU, де один і той самий код може виконуватися на обох архітектурах;

4) високу масштабованість обчислень між машинами та наборами даних.

API TensorFlow організовані за ієрархічним принципом, де базові, низькорівневі API слугують основою для побудови більш складних високорівневих. Така структура дозволяє користувачам обирати рівень абстракції, який найкраще відповідає їхнім потребам. Низькорівневі API забезпечують скрупульозний контроль над моделлю та обчисленнями, що є визначальним для розробників у галузі машинного навчання. Використовуючи ці інструменти, вони можуть експериментувати і розробляти інноваційні моделі ML, тестувати нові підходи та досягати проривів у своїх дослідженнях. Ця гнучкість у виборі між прямим доступом до низькорівневих функцій і впровадженням зручних високорівневих абстракцій робить TensorFlow однією з найпопулярніших платформ для розробки сучасних рішень у сфері штучного інтелекту.

Keras – це високорівневий API платформи TensorFlow. Він надає доступний та продуктивний інтерфейс для розв’язання задач машинного навчання. Keras охоплює кожен етап робочого процесу машинного навчання, від обробки даних до налаштування гіперпараметрів та розгортання. З Keras існує повний доступ до масштабованості та міжплатформної сумісності з TensorFlow.

Keras розроблений для зниження когнітивного навантаження, досягаючи наступних цілей:

1) пропонувати прості, послідовні інтерфейси;

2) мінімізувати кількість дій, необхідних для звичайних сценаріїв;

3) надавати чіткі й релевантні повідомлення про помилки;

4) дотримуватися принципу поступового розкриття складності: легко почати, а потім можна освоїти складніші робочі процеси, навчаючись на ходу;

5) допомагати писати стислий і зрозумілий код.

Для тренування і тестування моделі як джерело легітимних ресурсів було використано Majestic Million, оскільки сам список оновлюється щодня і відображає рейтинг веб-сайтів на основі їхньої важливості та впливу в Інтернеті [34].

Для того, щоб збалансувати законні ресурси шахрайськими доменами, було імплементовано приклади DGA із ресурсу DGArchive. Доступ до цієї платформи виключно персональний, тому для цього було надіслано запит з корпоративної пошти університету для отримання логіну та паролю, щоб отримати зразки доменів. Загалом у рівній кількості легітимних і шахрайських доменів виділено 4000 [34].

Таким чином, використання мови програмування Python у поєднанні з інтегрованим середовищем розробки PyCharm дозволило створити модель машинного навчання для виявлення зловмисних доменних імен. Платформи TensorFlow та Keras забезпечили необхідні інструменти для оптимізації, масштабування та впровадження складних архітектур машинного навчання. Завдяки їхній інтеграції, розробка стала більш гнучкою та продуктивною, що сприяло створенню надійних систем для боротьби з онлайн-шахрайством.

3.3 Опис розробленої моделі

Рішення моделі для виявлення шахрайських доменів організовано з трьох модулів: перший модуль, data_arrangement_module.py – додаток А, керує процесом попередньої обробки та підготовкою даних для тренування та тестування; другий, ml_model_pipeline.py – додаток Б, забезпечує завантаження даних та створення моделі машинного навчання; а третій – додаток В, results_estimation_module.py, відповідає за оцінку результатів моделі.

Перший модуль відповідає за підготовку даних, що є фундаментальною задачею для ефективного тренування та тестування моделі. Спочатку відбувається завантаження даних з CSV-файлу під назвою shuffled_domains_with_lables.csv, розташованого в директорії ../domains_source/, додаток Г. Цей файл містить основну

інформацію для моделі, включаючи URL-адреси та їхні класифікаційні мітки, де кожна мітка вказує на те, чи є домен шахрайським (мітка 1) або безпечним (мітка 0).

Після успішного завантаження даних у DataFrame, відбувається процес токенізації URL. Кожен символ у кожному URL трансформується в числовий індекс на основі його позиції у списку друкованих символів ASCII (printable). Ці індекси збільшуються на одиницю, оскільки нульовий індекс використовується для заповнення пропусків у даних (padding).

Далі для стандартизації довжини вхідних послідовностей токенізовані домени приводяться до єдиної довжини (80 символів) за допомогою функції pad_sequences() з Keras. Це забезпечує уніформність даних для мережі, де занадто короткі рядки доповнюються нулями, а занадто довгі – обрізаються до необхідної довжини.

Бінаризація міток класів здійснюється через проходження по кожному значенню у колонці class DataFrame і додавання до нового списку значення 0 або 1 залежно від його початкового значення. Завершальним етапом є імплементація train_test_split() з бібліотеки Scikit-learn для розділення даних на тренувальну і тестову вибірки, де 10% даних використовуються для тестування моделі. Такий підхід дозволяє оцінити реальну ефективність моделі на нових даних, забезпечуючи достовірність виявлення шахрайських доменів.

Другий модуль описує процес створення, тренування та валідації моделі. Він починається з імпорту бібліотек, включаючи ті, що потрібні для роботи з часом, відображення попереджень, обробки дат, моделювання в Keras та регуляризації. Особлива увага зосереджена на управлінні ресурсами Graphics Processing Unit (GPU). Після імпорту у коді налаштовані ресурси обчислення. Відбувається перевірка на наявність GPU, налаштовуючи TensorFlow на динамічне виділення пам’яті, що дозволяє оптимальніше використовувати доступні ресурси GPU.

Функція build_convolution_layer() призначена для побудови згорткового шару у моделі. Вона приймає чотири параметри: emb як вхідні дані з попереднього шару вбудовування, kernel_size для встановлення розміру ядра згортки, filter для встановлення кількості фільтрів у згортковому шарі, та padding, яке встановлено за замовчуванням на same для збереження розміру вхідних даних після застосування

згортки. Ключовим елементом є параметр kernel_size, який визначає розмір ядра згортки. Розмір впливає на кількість текстових елементів, що аналізуються одночасно, дозволяючи моделі вловлювати контекстуальні зв’язки в межах фіксованого вікна. Різні розміри ядер дозволяють їй краще адаптуватися до різних структур текстових даних.

Параметр filters вказує на кількість фільтрів, які будуть у згортковому шарі, та відповідно кількість вихідних каналів, які цей шар генерує. Більша кількість фільтрів дозволяє мережі краще вловлювати особливості вхідних даних, забезпечуючи більш деталізоване та глибоке розуміння тексту. Використання функції активації Exponential Linear Unit (ELU) після згорткового шару сприяє підвищенню нелінійності в обчисленнях та кращій здатності моделі розв’язувати завдання класифікації. ELU забезпечує плавний перехід негативних значень до нуля, що дозволяє уникнути проблем з затухаючими градієнтами під час навчання. Шар MaxPooling1D застосовується для зменшення розмірів вихідних даних, зменшуючи обсяги для обробки на наступних етапах та допомагаючи моделі фокусуватися на найважливіших ознаках у вхідних відомостях. Додавання шару Dropout з імовірністю 0.5 сприяє запобіганню перенавчанню моделі шляхом випадкового вимкнення деяких нейронів під час тренування, що змушує мережу не покладатися на будь-який один шлях в мережі та забезпечує узагальнені властивості моделі.

Функція define_hybrid_architecture() створює комплексну гібридну модель, комбінуючи згорткові нейронні мережі, двонаправлену довготривалу короткочасну пам’ять і механізм самоуваги. Ця функція приймає наступні параметри:

1) max_len – розмірність входу, а саме максимальна довжина послідовностей, які будуть оброблятися моделлю. Параметр необхідний для стандартизації довжин вхідних послідовностей, оскільки вимагаються вхідні дані фіксованого розміру.

2) emb_dim – розмір векторів, які генеруються шаром вбудовування. Параметр вказує на розмірність вихідного векторного простору, у якому кожне слово представляється як вектор. Величина цього параметра впливає на кількість інформації, яку вектор вбудовування може зберегти про слово, що, в свою чергу, впливає на точність та якість моделювання мовних зв’язків.

3) max_vocab_len – вказує на максимальну довжину словника, який використовується у шарі вбудовування. Параметр встановлює верхню межу кількості унікальних слов, які можуть бути представлені у моделі. Словник містить усі слова, які можуть зустрічатися в тренувальному наборі даних, і важливий для перетворення словесних токенів у числові індекси, які подаються на шар вбудовування.

4) w_reg – регуляризатор для шарів вбудовування. Регуляризатор, зазвичай L2, додає штраф до функції втрат за великі ваги в шарі вбудовування. Це запобігє перенавчанню шляхом зменшення складності моделі. Регуляризація в шарі вбудовування сприяє формуванню більш гладких та узагальнених векторів.

Функція define_hybrid_architecture() починається з визначення вхідного шару через функцію Input, яка задає форму вхідних даних для моделі за допомогою параметра shape установленого як кортеж із одним елементом max_len. Він відображає максимальну довжину послідовностей, які модель буде обробляти. Тип даних встановлено як int32, що є стандартом для індексів словників у задачах обробки природної мови, де кожне слово або токен представлено унікальним цілочисельним значенням. Шар вбудовування Embedding слідує відразу після вхідного шару та призначений для перетворення цих індексів у вектори щільних вбудовувань, що є представленням слова у багатовимірному просторі. Параметр input_dim встановлює максимальну кількість унікальних токенів, які шар може обробити, і визначає розмір словника, а output_dim задає розмірність кожного вектора вбудовування, що безпосередньо впливає на кількість інформації, яку вектор може нести про кожне слово. Регуляризатор w_reg додається до шару вбудовування для контролю за складністю моделі, запобігаючи надмірному пристосуванню до тренувальних даних шляхом накладення штрафів на великі значення ваг у векторах вбудовування. Після шару вбудовування впроваджено шар Dropout з параметром 0.2, що означає, що 20% нейронів випадково ігноруються під час кожної ітерації тренування, що зменшує ризик перенавчання та сприяє кращій загальній здатності моделі узагальнювати на невидимих даних.

Далі для ефективного вилучення ознак з вбудованих слів використовуються чотири згорткові шари з різними розмірами ядер: 2, 3, 4 і 5. Кожен з них обробляє

вбудовування слова, створене попереднім шаром. Застосування різних розмірів ядер дозволяє моделі захоплювати контекстуальні зв’язки в межах вікон вхідних даних, забезпечуючи ширший аналіз інформації. Фільтри в кількості 256 для кожного шару дозволяють генерувати багатий набір ознак для кожного з розглянутих розмірів ядер, забезпечуючи глибоке і різноманітне представлення вхідних текстових даних. Кожен шар застосовує нелінійну активацію ELU і вбудовування для зменшення розмірів вихідних даних, а також метод Dropout для уникнення перенавчання.

Після згорткових шарів в моделі використовується двонаправлений LSTM шар, який дозволяє обробляти текстові послідовності як в прямому, так і в зворотному напрямках. Застосування 128 одиниць в LSTM шарі забезпечує достатню складність моделі для аналізу тексту. На основі двонаправленого LSTM шару впроваджено шар самоуваги, де активація ReLU забезпечує високу ефективність при передачі критичної інформації, зменшуючи при цьому вплив менш значущих даних.

На етапі об’єднання виходів шарів використовується функція concatenate() для злиття даних з згорткових шарів та шару самоуваги, який є вихідним шаром після двонаправленого LSTM. Злиття відбувається вздовж осі 1, що означає горизонтальне об’єднання вихідних тензорів кожного шару в один тензор. Це дозволяє інтегрувати інформацію, яка була видобута з даних за допомогою різних підходів. Після об’єднання ці дані подаються в шар Flatten, який трансформує багатовимірний тензор у одновимірний вектор, придатний для подачі в повнозв’язані шари.

Далі інформація передається в два повнозв’язані (Dense) шари. Перший має 4128 одиниць, що значно збільшує кількість параметрів в моделі та здатність до узагальнення. Функція активації ELU застосовується для введення нелінійності в перетворення, а BatchNormalization нормалізує вихідні дані шару, що сприяє стабілізації та прискоренню навчання. Dropout з імовірністю 0.5 впроваджено для запобігання перенавчанню, випадково відключаючи частину нейронів під час тренування. Другий повнозв’язаний шар з 1024 одиницями продовжує обробку вихідних даних з першого, також застосовуючи ELU, BatchNormalization і Dropout для покращення навчального процесу.

Вихідний шар мережі має одну одиницю з сигмоїдною активацією, що типово для завдань бінарної класифікації, де модель вирішує, до якої з двох категорій належить вхід. Після встановлення архітектури модель компілюється, визначаючи, як саме модель буде оцінювати свої втрати і який оптимізатор буде використовуватися для мінімізації цих втрат під час навчання. У результаті створюється гібридна модель, яка інтегрує в собі кілька підходів до обробки тексту.

Далі в коді вказано для TensorFlow застосовувати GPU для обчислень. /GPU:0 вказує, що потрібно задіяти перший доступний графічний процесор. Це забезпечує оптимальне використання обчислювальних ресурсів, особливо коли в моделі є багато параметрів і великі об’єми даних.

Дані для тренування та тестування завантажуються за допомогою модулю Arrangement, що імпортує тренувальні та тестові набори, підготовлені спеціально для задачі класифікації. Після цього визначається структура гібридної моделі і назва моделі встановлюється як «Гібридна модель CNN-BiLSTM-Att».

Потім описуються два зворотні виклики, що є інструментами для оптимізації процесу навчання в Keras: EarlyStopping і ModelCheckpoint. Вони використовуються для того, щоб зробити навчання більш дієвим і запобігти перенавчанню.

EarlyStopping реалізовано для припинення навчання моделі, якщо вона не показує покращення протягом встановленої кількості епох. Ось параметри, які були встановлені в цьому випадку:

1) monitor='val_loss' – параметр вказує, що зворотній виклик слідкує за зміною значення втрат на валідаційних даних. Значення втрат є основним показником для визначення необхідності припинення тренування. Зменшення втрат свідчить про те, що модель вчиться ефективно.

2) mode='min' – режим вказує на те, що процес тренування припиниться, якщо відстежуване значення val_loss більше не буде знижуватися, тобто коли не спостерігається покращення мінімального значення.

3) verbose=1 – встановлення цього параметру дозволяє виводити додаткові повідомлення про припинення тренування, що допомагає відстежувати процес.

4) patience=3 – це кількість епох, протягом яких слід спостерігати за відсутністю покращення, перед тим як припинити тренування. У даному випадку, якщо після трьох епох не спостерігається зменшення втрат на валідаційних даних, навчання припиняється. Це дозволяє уникнути надмірного тренування моделі.

Для забезпечення надійного контролю за процесом тренування та збереженням досягнутих результатів у Keras використовується функція зворотного виклику ModelCheckpoint. Ця функція дозволяє зберегти копію моделі в момент, коли вона показує найкращі результати згідно із встановленими критеріями. Ось основні параметри, які задіяні у цій функції:

1) filepath='./trained_models/' + model_name + '.keras' – шлях і назва файлу, куди буде збережено модель.

2) monitor='val_loss' – зворотній виклик слідкує за значенням втрат на валідаційних даних, зберігаючи модель за умови її покращення.

3) mode='min' – збереження моделі відбувається тоді, коли відбувається зменшення зазначеного показника, val_loss.

4) save_best_only=True – параметр означає, що буде збережено лише найкращий варіант моделі, що відповідає мінімальному значенню втрат.

5) verbose=1 – виведення повідомлень про збереження моделі в файл.

Ці параметри управляють навчанням моделі таким чином, щоб оптимізувати його якість, уникнути перенавчання, а також зберегти найкращу конфігурацію моделі для подальшого використання.

Фаза тренування починається з визначення кількості епох для тренування, яка встановлена на рівні десяти. Епоха передбачає повний прохід через набір тренувальних даних. Також встановлюється розмір пакету для тренування, який становить 64. Розмір пакету впливає на кількість даних, які модель обробляє одночасно, що важливо для визначення швидкості та стабільності процесу навчання.

Далі відбувається налаштування оптимізатора Adam, який є одним із популярних методів оптимізації в машинному навчанні, особливо для великих наборів даних і складних архітектур. Головна особливість Adam полягає в адаптивному обчисленні коефіцієнтів навчання для кожного параметру на основі

оцінок перших і других моментів градієнтів. Градієнт – це вектор, що вказує напрямок найшвидшого зростання функції втрат, яку мінімізує модель. При оптимізації мета полягає в тому, щоб знайти такі параметри моделі, при яких функція втрат буде мінімальною. Оптимізатор оновлює кожен вагу, від'ємно коригуючи його на величину градієнта, помножену на швидкість навчання. Перший момент відповідає математичному очікуванню градієнтів (або їхньому середньому значенню), і це використовується для корекції кожної ваги відповідно до середньої величини його градієнта. Другий момент враховує дисперсію градієнтів, що допомагає адаптувати швидкість навчання відповідно до нестабільності змін вагів. Це стабілізує процес навчання, зокрема, коли градієнти мають різні дисперсії.

Впроваджені параметри оптимізатора Adam містять:

1) learning_rate=1e-4 – це основний параметр, який контролює крок оновлення вагів в мережі. Чим менше швидкість навчання, тим менший крок зміни вагів, що може призвести до більш гладкого, але повільнішого навчання.

2) beta_1=0.9 – цей параметр застосовується для оцінки першого моменту градієнта, тобто середнього значення. Значення 0.9 означає, що віддача від попередніх градієнтів становить 90%, і тільки 10% віддачі приходить від поточного.

3) beta_2=0.999 – цей параметр використовується для оцінки другого моменту градієнта, або дисперсії. Значення 0.999 вказує на те, що оновлення вагів відбуваються дуже плавно, що забезпечує стабільність в тренуванні.

4) epsilon=1e-08 – маленьке число, що додається в знаменник при обчисленні адаптивних швидкостей навчання для уникнення ділення на нуль.

Загалом параметри сприяють оптимальному поєднанню швидкості та стабільності тренування моделі, адаптуючи оновлення вагів до індивідуальних особливостей кожного з них в залежності від їхніх градієнтів.

Наступним кроком проводиться компіляція моделі, де визначається описаний вище оптимізатор, функція втрат і метрики для оцінювання. Функція втрат binary_crossentropy() є стандартним вибором для бінарної класифікації, оскільки вона вимірює розбіжності між фактичними мітками та прогнозованими ймовірностями, що є ключовим для точного передбачення. Метрики в процесі компіляції моделі надають

конкретні критерії для оцінки її продуктивності під час тренування та валідації. Клас Estimation включає реалізацію цих метрик, що дозволяє провести детальний аналіз моделі за окремими аспектами її роботи.

Після компіляції моделі фіксується час початку тренування, що дозволяє виміряти загальну тривалість даного процесу. Далі виконується метод fit(), який відповідає за тренування моделі. Він приймає наступні параметри:

1) x_train, y_train – тренувальні дані та відповідні мітки, які будуть використані для навчання моделі.

2) epochs=num_epochs – вказує кількість епох тренування.

3) batch_size=size_of_batch – розмір пакета, тобто кількість тренувальних прикладів, які обробляються за один раз під час тренування. Вибір розміру пакету впливає на швидкість тренування та на те, наскільки гладко модель оптимізується.

4) validation_split=0.11 – цей параметр вказує, що 11% даних з тренувального набору буде використано як валідаційний набір. Ці дані не беруть участь у безпосередньому тренуванні, але застосовуються для перевірки моделі після кожної епохи, щоб запобігти перенавчанню.

5) callbacks=[early_stop, checkpoint_monitor] – зворотні виклики реалізовано для додаткового контролю над процесом тренування. early_stop зупиняє тренування, якщо модель перестає покращуватися, тим самим запобігаючи витраті часу та ресурсів на надмірне тренування. checkpoint_monitor зберігає її найкращу версію під час тренування, що дозволяє відновити даний екземпляр за потреби.

Нарешті після завершення тренування знову фіксується час для визначення його тривалості та оцінки ефективності використання обчислювальних ресурсів.

У наступних рядках коду проводиться фаза передбачення за допомогою попередньо навченої моделі. Спочатку викликається функція define_hybrid_architecture(), яка повертає модель, що була визначена раніше з використанням гібридної архітектури. Далі завантажується ваги з файлу, у якому вони були збережені після тренування, через метод load_weights().

Після завантаження ваг модель компілюється знову, тому що потрібно встановити параметри компіляції, такі як оптимізатор, функція втрат та метрики. Така

підготовка дозволяє моделі коректно виконувати передбачення на нових даних з врахуванням попереднього навчання.

Наступним кроком знову за допомогою datetime.now() фіксується час початку передбачення. Потім викликається метод predict() для запуску процесу передбачення результатів на тестових даних x_test. У кінці datetime.now() фіксує час завершення, що дозволяє оцінити час виконання даного процесу.

У наступному фрагменті коду проводиться оцінка результатів моделі на тестових даних та їхня візуалізація. Починається з виклику методу Estimation.generate_validation_plots(), який генерує графіки для візуалізації результатів тренування. Параметр model_name вказує на назву моделі, а training_history – на її історію тренування, що містить інформацію про метрики та втрати під час тренування. Метод Estimation.display_validation_results() викликається наступним для відображення результатів валідації моделі.

Далі виконуються дії для оцінки та візуалізації результатів. Метод Estimation.compute_model_metrics() використовує переданий об’єкт моделі chief_model для проведення прогнозування на основі тестових даних x_test. Прогнози видаються у формі ймовірностей – чисел між 0 і 1, які показують, наскільки модель впевнена, що даний приклад належить до певного класу. Ці прогнози потім порівнюються з фактичними мітками y_test для визначення якості моделі. Метод виконує перетворення ймовірностей, отриманих від моделі, у бінарні класи шляхом встановлення порога 0.5: це означає, що ймовірності, які дорівнюють або перевищують 0.5, вважаються позитивним класом, а нижчі за 0.5 – негативним. Далі, за допомогою бібліотеки sklearn.metrics, він обчислює основні метрики якості класифікації, такі як точність, влучність, повнота та F1-показник, і виводить ці метрики разом із детальним звітом про класифікацію.

Функція Estimation.visualize_matrix() візуалізує матрицю невідповідностей для передбачених результатів моделі. Параметр normalize вказує, чи потрібно нормалізувати матрицю, щоб отримати пропорційні значення. Спочатку встановлено значення False, оскільки відображається матриця без нормалізації. Також дана функція візуалізує нормалізовану матрицю невідповідностей для передбачених

результатів моделі. І вже в цьому випадку параметр normalize встановлено на значення True, щоб нормалізувати матрицю та отримати пропорційні значення. Також виводяться час, який був витрачений на тренування, та час, який був витрачений на передбачення результатів за тестовими даними.

Третій модуль фокусується на оцінюванні та аналізі якості моделі, яка була розроблена та навчена за допомогою попередньо описаних частин. Модуль Estimation забезпечує інструментарій для детальної перевірки результатів, включаючи розрахунок ключових метрик ефективності та їхню візуалізацію [35].

Завдяки використанню TensorFlow та Keras, модуль Estimation здійснює вимірювання таких метрик, як влучність, повнота та F1-показник. Метрики впроваджено для оцінки, наскільки адекватно модель ідентифікує позитивні випадки (шахрайські домени) та відкидає негативні (легітимні сайти). Влучність вказує на частку правильно ідентифікованих позитивних випадків серед всіх позитивних, які модель класифікувала як такі. Повнота вимірює, скільки дійсно позитивних випадків було виявлено моделлю. F1-показник об’єднує обидві ці метрики, надаючи єдине число для оцінки загальної збалансованості між влучністю та повнотою.

Клас Estimation включає precision_metric та recall_metric, які ініціалізовані як екземпляри відповідних метрик точності та повноти з бібліотеки TensorFlow. Ці метрики оцінюють здатність моделі коректно ідентифікувати класи.

Метод update_metrics() приймає істинні мітки y_true і передбачені мітки y_pred, округляючи останні за допомогою tf.round для перетворення їх у бінарний формат. Цей метод викликає update_state() для кожної з метрик, щоб вони могли врахувати нові дані та оновити свій стан.

Метод fmeasure_metric() також використовує y_true та y_pred, спочатку оновлюючи метрики через update_metrics(). Після цього він зчитує поточні значення точності та повноти через result() для обох метрик, і обчислює F1-показник. Метод get_metrics() повертає список метрик, які використані під час тренування моделі. Цей список включає загальну точність, влучність (precision), повноту (recall) та F1-показник. Тут застосовано tf.keras.metrics.BinaryAccuracy, яка є оптимізованою для бінарних класифікаційних завдань, де моделі передбачають ймовірність того, що

зразок належить до одного з двох класів (0 або 1). Ця метрика вимірює частоту, з якою передбачення правильно відображають мітки класів, і є чутливою до порога класифікації. BinaryAccuracy враховує порогове значення, яке встановлюється на 0.5 за замовчуванням. Це означає, що якщо передбачена ймовірність моделі вища або дорівнює 0.5, зразок буде класифікований як позитивний, інакше як негативний.

Далі модуль забезпечує засоби для візуалізації результатів, такі як генерація графіків, які демонструють валідаційні криві, ілюструючи, як метрики моделі змінювалися протягом епох тренування. Функції generate_validation_plots() та visualize_matrix() відповідають за ці завдання, де generate_validation_plots() використовує дані з history.history для побудови графіків точності, влучності, повноти та F1-показника, а visualize_matrix() генерує зображення матриці через функції confusion_matrix() і classification_report() з sklearn.metrics. Це дозволяє зрозуміти, на якому етапі модель починає пере- або недонавчання. Користувачі можуть легко порівнювати результати різних сесій тренування та проводити детальний аналіз впливу змін в моделі чи даних на загальну продуктивність.

Однією з ключових функцій модуля є створення матриці невідповідностей, яка дозволяє візуально оцінити, як модель класифікує кожен із класів. Матриця представлена у нормалізованій та ненормалізованій формах. Нормалізація допомагає краще зрозуміти відсоткове співвідношення між класами, що особливо корисно при нерівномірному розподілі класів. Ці опції реалізовані у коді і можуть бути візуально порівняні вибір відповідних параметрів, що дозволяє користувачу зробити обґрунтований вибір щодо методу презентації результатів.

Крім того, модуль включає збереження всіх результатів у файл, що містить дату та час проведення аналізу, забезпечуючи зручність управління версіями і дозволяючи порівняти різні сесії тренування. Це виконується в методах compute_model_metrics(), який також виводить деталізований звіт про класифікацію та точність визначається через accuracy_score() з sklearn.metrics. Виведення текстових звітів із детальними метриками для кожного класу допомагає зрозуміти ефективність моделі в реальних умовах.

Таким чином, модуль Estimation здійснює валідацію та поліпшує модель з виявлення шахрайських доменів, забезпечуючи глибокий аналіз та інструменти для оцінки та візуалізації результатів навчання.

3.4 Валідація отриманих результатів

Після запуску модулю ml_model_pipeline.py спостерігається процес навчання гібридної моделі CNN-BiLSTM-Att протягом десяти епох. Для детального аналізу метрик, отриманих під час тренування та валідації моделі, розглянемо кожну з них окремо, включаючи зміни впродовж всіх епох та їхні взаємозв'язки.

Тренувальна точність збільшилася з 80.67% до 96.81% протягом десяти епох. Це показує загальне покращення здатності моделі правильно класифікувати домени. Валідаційна точність коливалася, але загальний тренд зростаючий, з 94.95% у першій епосі до 97.73% у десятій. Це вказує на хорошу узагальнюваність моделі.

Тренувальні втрати зменшилися від 58.42% до 9.11%, що свідчить про те, що модель стала кращою в адаптації до тренувальних даних і зменшенні помилок класифікації. Для валідаційних втрат відбувалося постійне зниження з 55.83% до 8.4%, що вказує на високу ефективність моделі при роботі з невидимими даними.

Зростання влучності вказує на зменшення помилково позитивних результатів. У тренувальній вибірці влучність зросла з 81.94% до 97.77%, а у валідаційній зазнала невеликих коливань, але залишалася високою, зі значенням 96.94% в останній епосі.

Тренувальна повнота постійно зростала з 80.02% до 95.91%. Валідаційна повнота, попри деякі коливання, загалом збереглася на високому рівні, з останнім значенням 98.45% у десятій епосі.

F1-показник у тренувальній вибірці збільшився з 76.47% до 97.06%, а у валідаційній вибірці – від 95.17% до 97.61%. Це покращення F1-показника в обох вибірках підтверджує, що модель не тільки точно ідентифікує позитивні випадки, але і робить це з мінімальною кількістю помилок.

Загальні тенденції вказують на високу продуктивність моделі, з якою вона є придатною до реальних застосувань, де вимагається висока точність та надійність класифікації.

Число 51, яке зустрічається у логах тренування кожної епохи, вказує на кількість кроків (батчів) в одній епосі. Тобто, під час кожної епохи тренування, датасет для тренування було поділено на 51 батч. Кожен батч обробляється моделлю окремо, і після кожного такого оброблення її параметри оновлюються. Під час тренування, метрики вимірюються окремо для кожного батчу, а потім їхнє середнє значення обчислюється по всіх батчах епохи для визначення загальних метрик епохи. Це дозволяє моделі відстежувати і виводити прогрес у навчанні після кожної епохи.

Тут валідаційні метрики використовуються для оцінки якості моделі на валідаційних даних під час тренування. Валідаційний набір даних – це частина даних, яка відокремлена від тренувального набору і не застосовується під час активного навчання моделі, щоб забезпечити неупереджену оцінку її загальної ефективності. Для відділення 11% даних для валідації впроваджено параметр validation_split=0.11 в методі model.fit(). Даний відсоток обирається випадково з тренувального набору даних на початку процесу тренування моделі. Цей валідаційний набір залишається незмінним протягом усіх епох тренування. Тобто для кожної епохи використовується один і той же набір валідаційних даних, а не новий набір для кожної епохи.

Модель зберігалася під час кожної епохи навчання відповідно до значення втрат, loss, на валідаційній вибірці. У коді імплементовано зворотний виклик ModelCheckpoint з TensorFlow/Keras, який налаштований на моніторинг val_loss – втрат на валідаційних даних. Опція save_best_only=True вказує, що потрібно зберегти лише ту модель, яка показала найкращий результат за мінімальним val_loss, тобто на кожній епосі перевіряється, чи є поточна модель кращою порівняно з попередньо збереженою, і зберігається лише у випадку поліпшення.

На рис. 3.2 містяться підсумкові метрики якості класифікації моделі на тестовому наборі даних, разом із детальним звітом по кожному класу, і тривалість тренування та передбачення.

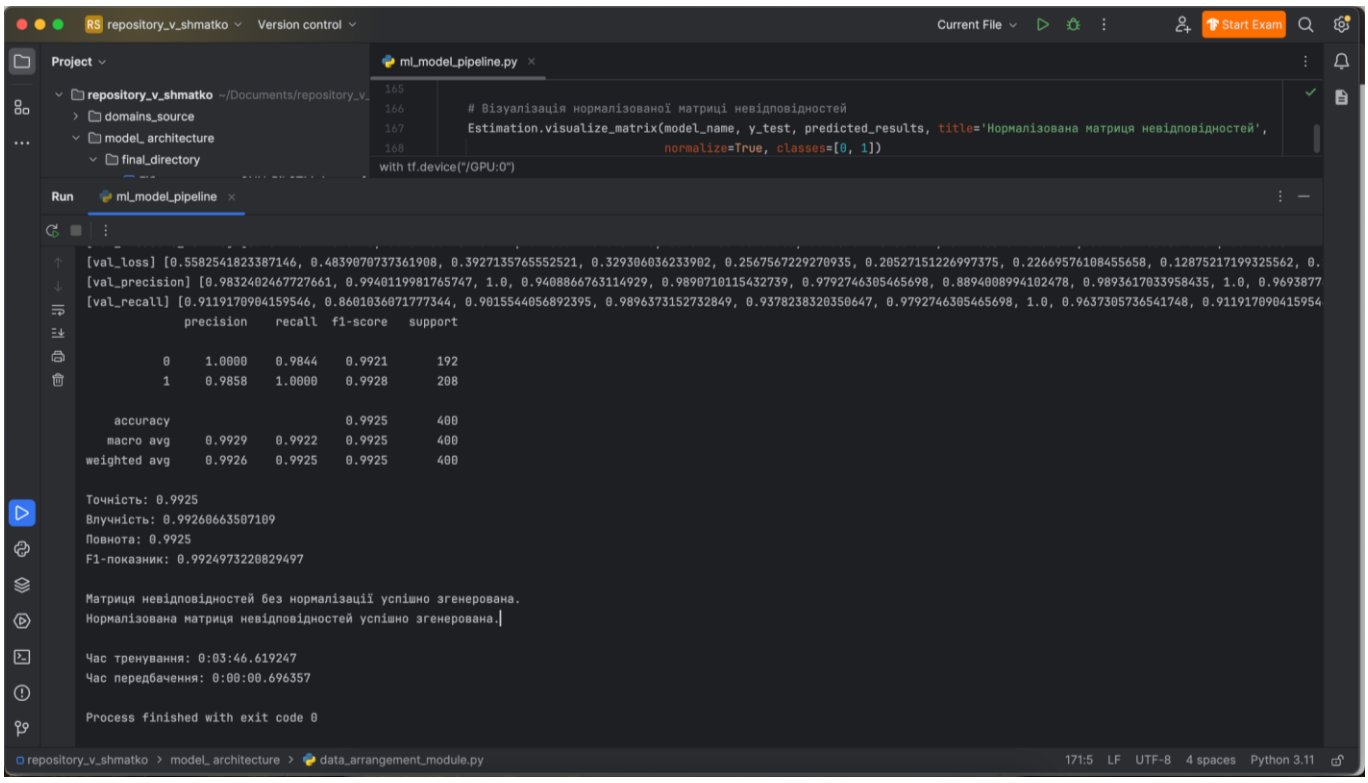

Рисунок 3.2 – Обчислені метрики на тестовому наборі даних

Загальна точність моделі становить 0.9925 (99.25%), що є високим показником. Це вказує на те, що вона правильно класифікувала приблизно 99.25% випадків у тестовому наборі даних.

Загальна влучність становить 0.9926 (99.26%), вказуючи на високу якість класифікації моделі з точки зору мінімізації помилково позитивних результатів. Влучність для класу 0 (легітимних доменів) становить 100%, а для класу 1 (зловмисних доменів) – 98.58%, що свідчить про високу здатність моделі точно ідентифікувати обидва класи.

Загальна повнота моделі становить 0.9925 (99.25%), що свідчить про те, що вона коректно ідентифікує позитивні випадки. Повнота для класу 0 дорівнює 98.44%, а для класу 1 – 100%, що означає високу здатність моделі виявляти клас 1.

Загальний F1-показник становить 0.9925 (99.25%) – це показує, що модель добре збалансована з точки зору обох метрик. F1-показники окремо для класів також високі (99.21% для класу 0 і 99.28% для класу 1).

Потім у виводі надано розбиття метрик за класами, надаючи детальний огляд ефективності моделі для кожного класу окремо, що вже було зазначено вище.

Значення support відноситься до кількості реальних випадків у кожному класі, які були використані для обчислення метрик. Це важливо також враховувати, бо дана характеристика дозволяє зрозуміти, наскільки збалансовані дані, і який вплив має кожен клас на загальні метрики моделі. Якщо один клас має значно більший support порівняно з іншими, він відповідно матиме більший вплив на середні значення метрик, що обчислюються на всьому наборі даних.

Далі надано індикацію про генерацію матриць невідповідностей з нормалізацією та без неї. На рис. 3.3 представлена матриця з нормалізацією, яка висвітлює помилки класифікації у відсотках, що допомагає зрозуміти пропорційність помилок у контексті загальної кількості екземплярів у кожному класі.

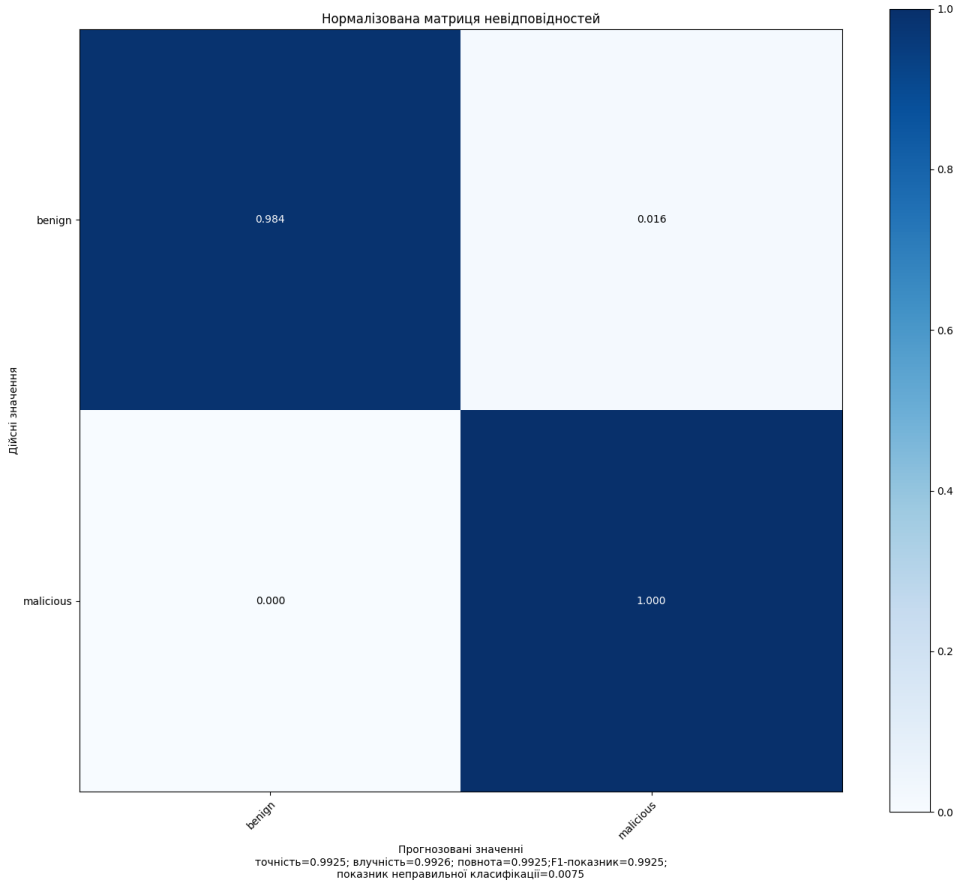

Рисунок 3.3 – Нормалізована матриця невідповідностей

Лівий верхній кут – True Negative: 0.984 означає, що 98.4% безпечних доменів були правильно класифіковані як безпечні. Це високий показник TN і вказує на те, що модель достатньо влучно виявляє домени, що не є шкідливими.

Правий верхній кут – False Positive: 0.016 свідчить, що 1.6% безпечних доменів були неправильно класифіковані як шкідливі. Це відносно малий відсоток FP, але в реальних ситуаціях навіть невелика кількість помилково позитивних результатів може мати серйозні наслідки.

Лівий нижній кут – False Negative: 0.000 показує, що модель не мала жодної помилки, де шахрайські домени були б помилково класифіковані як легітимні.

Правий нижній кут – True Positive: 1.000 означає, що 100% шкідливих доменів були правильно ідентифіковані як шкідливі. Це дуже високий показник TP, що свідчить про те, що модель ефективно виявляє шкідливі домени.

Матриця невідповідностей без нормалізації, представлена на рис. 3.4, дозволяє побачити абсолютні значення помилок класифікації між класами.

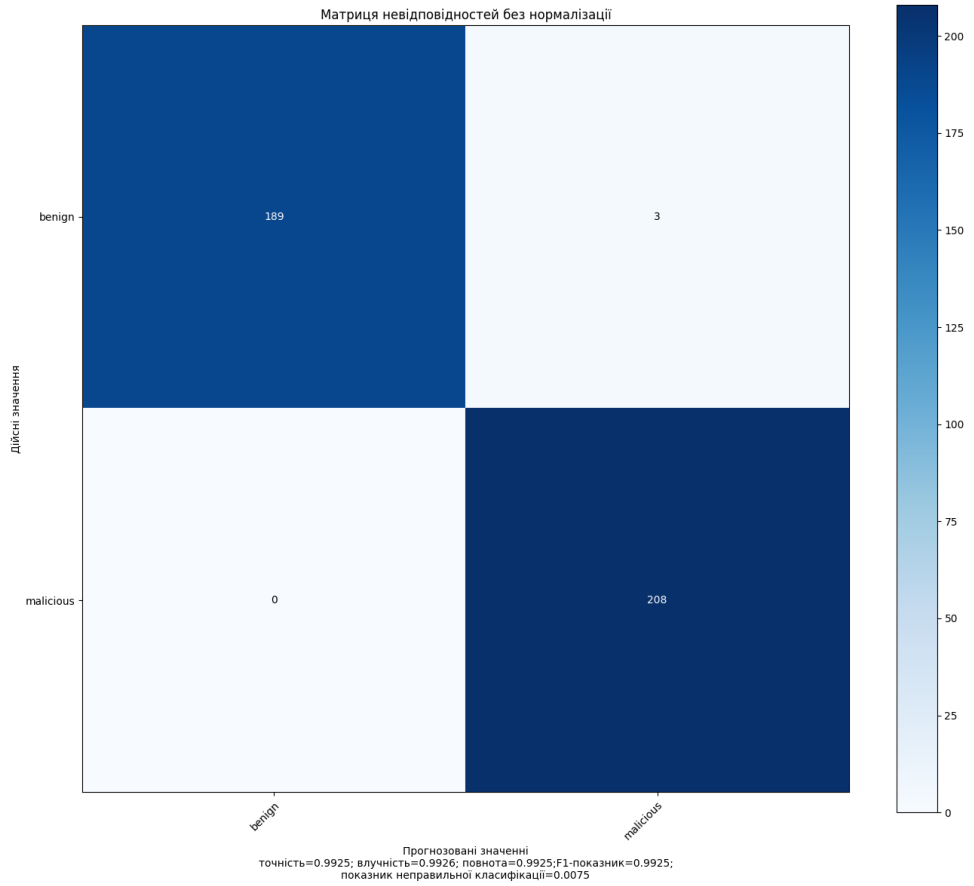

Рисунок 3.4 – Матриця невідповідностей без нормалізації

Лівий верхній кут – True Negative: 189 доменів правильно ідентифіковано як легітимні.

Правий верхній кут – False Positive: 3 домени помилково ідентифіковані як зловмисні (насправді є легітимні).

Лівий нижній кут – False Negative: немає доменів, які помилково ідентифіковані як легітимні, а насправді є шахрайськими.

Правий нижній кут – True Negative: 208 доменів правильно ідентифіковано як зловмисні.

Загалом, матриця невідповідностей без нормалізації та супутні метрики демонструють, що модель має високу ефективність з мінімальною кількістю помилок, що робить її дуже надійною для задачі класифікації доменів на легітимні та шкідливі.

Під матрицями, крім уже згаданих метрик, вказано значення 0.0125, 1.25% – показник неправильної класифікації, що вказує на частку помилок у загальній кількості передбачень, що є дуже низьким і позитивним показником.

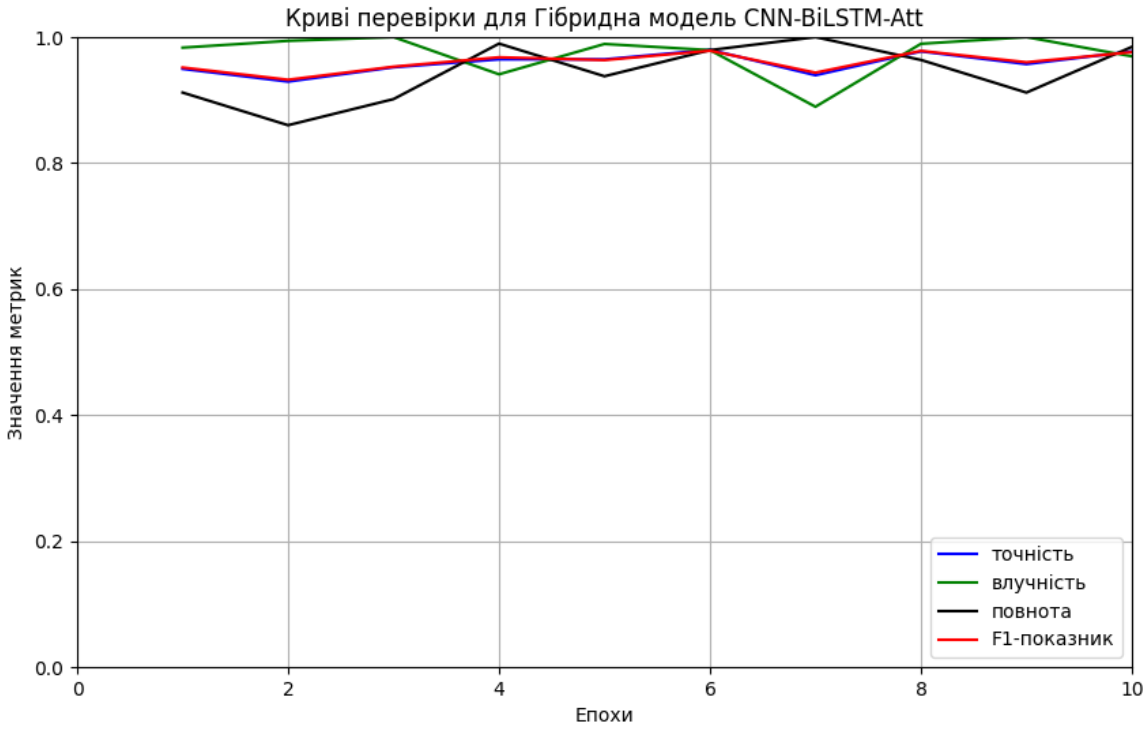

Рисунок 3.5 – Динаміка валідаційних метрик для моделі у процесі навчання

На рис. 3.5 представлено графік, де відображено динаміку валідаційних метрик протягом процесу навчання гібридної моделі. Кожна лінія представляє одну з метрик:

синя показує точність, зелена – влучність, чорна – повноту, а червона відповідає за F1-показник.

Із збільшенням числа епох від одного до десяти, можна спостерігати, що модель досягає стабілізації в показниках точності та F1-показника, що мають високі значення близькі до одиниці, забезпечуючи високу загальну продуктивність моделі.

Мінімальні варіації у влучності та повноті, які спостерігаються на графіку, можуть вказувати на чутливість моделі до різноманітності та складності валідаційного набору даних. Такі коливання є природними у процесі навчання, коли модель стикається з новими, менш однорідними даними. Проте загалом, стабільні показники влучності та повноти, які залишаються на високому рівні близькому до одиниці, підкреслюють надійність моделі у коректному виявленні та класифікації класів даних, що є особливо важливим для задач, де необхідно мінімізувати як помилково позитивні, так і помилково негативні результати. F1-показник, що залишається високим протягом всіх епох, вказує на стійкість і точність моделі у визначенні класів, свідчачи про її стабільність у різних умовах тестування. Цей графік є візуальним підтвердженням того, що модель успішно навчається та здатна ефективно класифікувати домени, що може бути використано для подальшого розвитку систем безпеки.

Час тренування моделі становив 3 хвилини 46 секунд і 619 мілісекунд. Цей період включає процес її навчання протягом всіх епох разом із передачею даних через мережу, оновленням ваг і валідацією після кожної епохи.

Час передбачення, з іншого боку, становив лише 696 мілісекунд. Цей період відображає швидкість, з якою модель змогла оцінити весь тестовий набір даних, після завершення тренування, що свідчить про час, який вимагається для прийняття рішень на основі навчених відомостей.

Модель показала високу ефективність у процесі навчання та валідації. Явне покращення задіяних метрик протягом десяти епох вказує на її здатність адаптуватися та оптимізувати свої параметри з метою мінімізації помилок. Значне зменшення втрати валідації свідчить про зростаючу здатність моделі узагальнювати навчене на невидимих даних, що є ключовим для використання у реальних сценаріях. Час

тренування та передбачення свідчить про її практичність у застосуванні, дозволяючи швидко обробляти великі обсяги даних.

У підсумку, модель продемонструвала високі показники продуктивності, здатність узагальнювати, тому є дієвим інструментом для класифікації доменних імен, як легітимних, так і зловмисних, з потенціалом для подальших покращень і реального застосування. Адаптація та інтеграція цієї моделі у виробничі середовища значно збільшить швидкість та точність виявлення шкідливих доменів, що, у свою чергу, підвищить загальний рівень кіберзахисту цільової компанії. Спроможність моделі працювати з різноманітними даними вказує на її гнучкість, що робить її придатною для використання в широкому спектрі сценаріїв виявлення загроз. Також здатність моделі до короткочасного навчання і адаптації до нових даних відкриває можливості для її застосування в умовах, де миттєве реагування на кіберзагрози є критично важливим.

Висновок до третього розділу

Розроблена модель заснована на гібридній основі, яка використовує комбінацію згорткових шарів, двонаправленого LSTM та механізму самоуваги для обробки текстових даних. Цей підхід дозволяє моделі ефективно вловлювати як локальні, так і глобальні залежності у доменних іменах, що є необхідним для розрізнення легітимних і зловмисних доменів.

Протягом тренування моделі спостерігається стабільне зниження втрати та поліпшення валідаційних метрик, що свідчить про виваженість навчального процесу та здатність моделі до узагальнення. Кінцеві результати показують високу точність, влучність, повноту та F1-показник, тому модель є потенційно вартісним інструментом для експлуатації в реальних умовах.

Враховуючи час тренування та передбачення, модель виявилася продуктивною з точки зору швидкодії, таким чином, вона є придатною для застосування в умовах, що вимагають вчасної обробки великих обсягів даних. Модель може бути інтегрована як модуль у системи безпеки для виконання безперервного сканування та аналізу

доменних імен, з використанням обчислювальних можливостей в хмарі або на локальних серверах для забезпечення мінімальної затримки та високої доступності. Це дозволить системам з кібербезпеки оперативно реагувати на зловмисні спроби та забезпечувати захист в реальному часі.

ВИСНОВОК

Протягом опрацювання першого розділу кваліфікаційної роботи було обґрунтовано, що ідентифікація DGA стає ключовим засобом у боротьбі проти сучасних ботнетів. Використання DGA дозволяє зловмисникам генерувати велику кількість доменів, значна частина яких ніколи не буде активована, але може бути задіяна у майбутньому для зв’язку з серверами C&C. Це створює серйозні виклики для систем захисту, які повинні ідентифікувати потенційні загрози серед безлічі легітимних доменів, розсіюючи увагу оборонних механізмів. Машинне навчання є одним із найефективніших інструментів для ідентифікації DGA-доменів, оскільки може автоматично адаптуватися до змінних шаблонів і нових методів, які експлуатуються ботнетами. Завдяки його здатності обробляти великі обсяги даних та вчасно виявляти складні взаємозв’язки між доменами, впровадження моделей ML значно підвищує шанси на успішне виявлення та нейтралізацію шкідливих доменів.

У другому розділі фокус роботи був направлений на дослідження найбільш придатних моделей для ідентифікації зловмисних доменних імен. Особлива увага була приділена аналізу потенціалу машинного навчання, зокрема таких моделей як згорткові нейронні мережі (CNN), двонаправленій довгостроковій короткочасній пам’яті (BiLSTM) та механізму самоуваги в контексті роботи з доменами.

CNN виявилися корисними в розпізнаванні візуальних та текстових патернів у доменних іменах, досліджуючи кожен символ як частину більшої структури. Це дозволяє моделі зосереджуватись на локальних особливостях імен, що є критичним для ідентифікації складних маскувань та мутацій у зловмисних доменах. BiLSTM, з іншого боку, відмінно справляється із задачами, де необхідне розуміння контексту та зв’язків між символами у послідовностях. Використання двонаправленої структури дозволяє моделі ефективно обробляти інформацію з обох кінців вхідної послідовності, підвищуючи точність виявлення аномалій. Механізм самоуваги, включений в архітектуру, спонує модель зосереджуватися на найбільш важливих аспектах даних, дозволяючи ігнорувати менш значущі деталі та спиратися на ключові

ознаки, які вказують на зловмисність доменів. Втілений гібридний підхід, який інтегрував CNN, BiLSTM та механізми самоуваги, став ідеальним прикладом надійної системи для виявлення нелегітимних доменів, адаптованої до постійно змінюваних тактик та методів кіберзлочинців.

У третьому розділі було надано роз'яснення щодо розробленої модель, що використовує гібридний підхід CNN-BiLSTM-Att для класифікації доменних імен на легітимні та зловмисні та демонструє комплексний підхід до обробки текстових даних в галузі кібербезпеки. Програмний код моделі складається з трьох модулів: перший націлений на підготовку та обробку даних, другий – відповідає за створення та тренування моделі, а третій – зосереджений на оцінці результатів.

Центральні аспекти моделі включають:

1) підготовку даних: модель коректно обробляє доменні імена, перетворюючи їх у токени на рівні символів;

2) структура: гібридна основа з CNN для вилучення особливостей з тексту, BiLSTM для аналізу контексту в обох напрямках і механізм самоуваги для акценту на значущих символах підвищує точність моделі;

3) оптимізацію: застосування оптимізатора Adam і механізмів ранньої зупинки та збереження кращої моделі допомагає уникнути перенавчання і забезпечує збалансоване навчання;

4) оцінку та візуалізацію результатів: використання набору метрик, таких як точність, влучність, повнота, F1-показник, і візуалізація даних через криві навчання та матриці невідповідності надає вичерпний звіт щодо справності моделі.

За результатами роботи модель демонструє потенціал для високоточної класифікації доменів, що є ключовим у боротьбі із цільовою кіберзагрозою. Також гнучка структура дозволяє легко адаптувати її під сучасні вимоги безпеки і типи даних. Розроблене рішення може бути впроваджене в реальні кібербезпекові додатки, забезпечуючи фундамент для подальших досліджень і розширення у цій області. Інтеграція розробленої моделі є пріоритетною і потенційно дуже корисною задачею з огляду на постійне зростання кіберзагроз. Враховуючи, що багато кібератак

починаються з маскування шахрайських доменів під законні, здатність швидко і точно визначати такі ресурси значно знизить ризик реалізації загроз.

Впровадження даної моделі у системи кібербезпеки, такі як фаєрволи, системи виявлення вторгнень та утиліти безпеки кінцевих точок, забезпечить більш ретельний та реактивний захист. Також розширення для браузерів з втіленою моделлю надасть додатковий шар перевірки посилань.

Для втілення поставленої мети були виконані такі завдання:

1) проведено специфіку реалізації кібератак, які використовують шахрайські доменні імена, з урахуванням їхньої природи та особливостей;

2) виконано дослідження підходів до запобігання кіберзагрозам, що використовують доменні імена, а також визначено їхню ефективність та обмеження;

3) досліджено перспективи використання методів машинного навчання для виявлення відповідних кібератак;

4) визначено ключові метрики для оцінки ефективності моделей машинного навчання та їхню залежність від характеристик наборів даних;

5) опрацьовано архітектури машинного навчання, які застосовні для ідентифікації та аналізу шахрайських доменів;

6) здійснено інтеграцію обраних архітектур для створення гібридної моделі та проведено аналіз отриманих результатів для визначення її ефективності.

Отже, мета роботи досягнута. Згідно з отриманими результатами тестування розроблена гібридна модель покращить захист інформаційних систем, зменшить шанси для успішних кібератак та допоможе реалізувати більш безпечне і контрольоване цифрове середовище.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yadav T., Rajendran B., and Rajani P. An Approach for Determining the Health of the DNS. International Journal of Computer Science and Mobile Computing, 2014.

2. Rajendran B., Shetty P. Domain Name System (DNS) Security: Attacks Identification and Protection Methods. International Conference of Security and management, 2018.

3. Weimer F. Passive DNS Replication. FIRST Conference on Computer Security Incident, 2005.

4. Sawabe Y., Chiba D., Akiyama M., Goto S. Detection Method of Homograph Internationalized Domain Names with OCR. Journal of Information Processing, 2019.

5. IDN Homograph Attacks: Guard Against Look-Alike Websites [Електронний ресурс]. Режим доступу: https://bluegrid.io/blog/idn-homograph-attacks-guard-against-look-alike-websites, 2023.

6. Unicode Security Mechanisms for UTS #39 [Електронний ресурс]. Режим доступу: https://www.unicode.org/Public/security/latest/confusables.txt, 2023.

7. Damerau. F. A Technique for Computer Detection and Correction of Spelling Errors. Communications of the ACM, 1964.

8. Haldar R., Mukhopadhyay, D. Levenshtein distance technique in dictionary lookup methods: An improved approach. arXiv preprint arXiv:1101.1232, 2011.

9. Truong M. T. Typosquatting Attacks and Mitigations. University of Applied Sciences, 2023.

10. Liu G. Investigating Security Threats of Resource Mismanagement in Networked Systems. Doctoral dissertation. Virginia Tech, 2023.

11. Typosquatting Unmasked: Exposing the Threats of Misplaced Keystrokes [Електронний ресурс]. Режим доступу: https://www.cyfirma.com/outofband/typosquatting-unmasked-exposing-the-threats-of-misplaced-keystrokes, 2023.

12. Hwang C., Kim H., Lee H., Lee, T. Effective DGA-Domain detection and classification with TextCNN and additional features. Electronics 9, 2020.

13. Hariaji A. S., Girsang A. S. Algorithmically Generated Malicious Domain Detection Using N-Grams Embedding and Attention-Based Bidirectional Gated Recurrent Unit. Journal of Theoretical and Applied Information Technology, 2023.

14. Maia R. J. M., Ray D., Pentyala S., Dowsley R., De Cock M., Nascimento A. C., & Jacobi R. An End-to-End Framework for Private DGA Detection as a Service. Cryptology ePrint Archive, 2023.

15. Plohmann D., Yakdan K., Klatt M., Bader J., Gelmar-Padilla E. A Comprehensive Measurement Study of Domain Generating Malware. The 25th USENIX Security Simposium, USENIX Association, 2016.

16. Rossow C., Andriesse D., Werner T., Dietrich C. J. Sok: P2PWNED – Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In Proceedings of the 34th IEEE Symposium on Security and Privacy, 2013.

17. Spamhaus Botnet Threat Update Q4 2023. Електронний ресурс. Режим доступу: https://info.spamhaus.com/hubfs/Botnet%20Reports/Q4%202023%20Botnet%20Threat%20Update.pdf?hsCtaTracking=52e45ca7-cb85-44b5-ac59-75bc2bbbf088%7C4a5be79a-8d82-4fb4-a2e2-16a0933e5693, 2024.

18. Hoang X. D., Le Minh D., Ninh T. T. T. A CNN-Based Model for Detecting Malicious URLs. RIVF International Conference on Computing and Communication, 2023.

19. Alazab M., Soman K. P., Srinivasan S., Venkatraman, S., Pham V. Q. Deep learning for cyber security applications: A comprehensive survey. Authorea Preprints, 2021.

20. Ruts D. Improved DGA-based botnet detection through context-related feature selection based on packet flow information, Master’s Thesis, 2023.

21. Kapan S., Gunal S. E. Improved Phishing Attack Detection with Machine Learning: A Comprehensive Evaluation of Classifiers and Features. Applied Sciences, 2023.

22. Zhou L., Pan S., Wang J., & Vasilakos, A. V. Machine learning on big data: Opportunities and challenges. Neurocomputing, 2017.

23. Sriram S., Soman K. P., Alazab M. Malicious URL detection using deep learning. Authorea Preprints, 2023.

24. Ruth K., Kumar D., Wang B., Valenta L., Durumeric, Z. Toppling top lists: Evaluating the accuracy of popular website lists. In Proceedings of the 22nd ACM Internet Measurement Conference, 2022.

25. Subashini K., Narmatha V. Phishing Website Detection Techniques: A Literature Survey. Journal of Data Acquisition and Processing, 2023.

26. 4 Types of Classification Tasks in Machine Learning [Електронний ресурс]. Режим доступу: https://machinelearningmastery.com/types-of-classification-in-machine-learning, 2020.

27. Convolutional Neural Network: Benefits, Types, and Applications [Електронний ресурс]. Режим доступу: https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network, 2017.

28. Shin H. C., Roth H. R., Gao M. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE transactions on medical imaging, 2016.

29. The Ultimate Guide to Building Your Own LSTM Models [Електронний ресурс]. Режим доступу: https://www.projectpro.io/article/lstm-model/832, 2024.

30. 5 Types of LSTM Recurrent Neural Networks [Електронний ресурс]. Режим доступу: https://www.exxactcorp.com/blog/Deep-Learning/5-types-of-lstm-recurrent-neural-networks-and-what-to-do-with-them, 2023.

31. Keles F. D., Wijewardena P. M., Hegde, C. On the computational complexity of self-attention. International Conference on Algorithmic Learning Theory, 2023.

32. Levine Y., Wies N., Sharir O., Bata H., Shashua A. Limits to depth efficiencies of self-attention. Advances in Neural Information Processing Systems, 2020.

32. Bagging, Boosting, and Stacking in Machine Learning [Електронний ресурс]. Режим доступу: https://www.baeldung.com/cs/bagging-boosting-stacking-ml-ensemble-models, 2024.

33. Бучик С.С., Шматко В.О. Виклики та перспективи використання машинного навчання у запобіганні доменним атакам: матеріали VII Міжнародної науково-практичної конференції «Прикладні системи та технології в інформаційному суспільстві», 2023.

34. Buchyk S.S., Shmatko V.O. Strategies for dataset collection in domain attack prevention with machine learning: матеріали X International Conference «Information Technology and Implementation» (IT&I-2023) Satellite, 2023.

35. Бучик С.С., Шматко В.О. Використання метрик оцінки алгоритмів для моделей виявлення шахрайських доменів: матеріали VII Міжнародної науково-практичної конференції «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (PCSITS), 2024.

36. Yu Y., Si X., Hu C., Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 2019.

37. Sharma T., Jatain A., Bhaskar S., Pabreja, K. Ensemble machine learning paradigms in software defect prediction. Procedia Computer Science, 2023.

38. Alshingiti Z., Alaqel R., Al-Muhtadi J., Haq Q. E. U. A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN. Electronics, 2023.

39. Shiri F. M., Perumal,T., Mustapha N., Mohamed, R. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv:2305.17473, 2023.

40. Hossain M. A., Islam M. S. Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. Array 19, 2023.

41. Al Odaini A. A., Zola F., Segurola-Gi, L., Gil-Lertxundi A., D'Andrea, C. Cybersecurity in Public Space: Leveraging CNN and LSTM for Proactive Multivariate Time Series Classification. In 2023 IEEE International Conference on Big Data, 2023.

42. Xu H., Wu L., Xiong S., Li W., Garg A., Gao, L. An improved CNN-LSTM model-based state-of-health estimation approach for lithium-ion batteries. Energy, 2023.

43. Alatawi M. N., Alsubaie N., Ullah Khan H., Sadad T. Cyber security against intrusion detection using ensemble-based approaches. Security and Communication Networks, 2023.

44. Torre D., Mesadieu F., Chennamaneni, A. Deep learning techniques to detect cybersecurity attacks: a systematic mapping study. Empirical Software Engineering, 20233.

45. Lin H., Zhang S., Li Q., Li Y., Li J., Yang, Y. A new method for heart rate prediction based on LSTM-BiLSTM-Att. Measurement, 207, 2023.

46. Andresini G., Appice, A. AI meets cybersecurity. Journal of Intelligent Information Systems, 2023 .

47. Das S., Gangwani P., Upadhyay H. Integration of machine learning with cybersecurity: applications and challenges. Artificial intelligence in cyber security: theories and applications, 2023.

48. Dini P., Elhanashi A., Begni A., Saponara S., Zheng Q., Gasmi, K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. Applied Sciences, 2023.

49. Saabith A. S., Vinothraj T., Fareez M. M. M., Marzook, M. M. A survey of machine learning techniques for anomaly detection in cybersecurity, International Journal of Research in Engineering and Science (IJRES), 2023.

50. Usoh M., Asuquo P., Ozuomba S., Stephen B., Inyang, U. A hybrid machine learning model for detecting cybersecurity threats in IoT applications. International Journal of Information Technology, 2023.

51. Thammareddi L., Agarwal S., Bhanushali A., Patel K., Venkata, S. Analysis On cybersecurity threats in modern banking and machine learning techniques for fraud detection, An International Multidisciplinary Online Journal, 2023.

52. Thapa N., Liu Z., Kc D. B., Gokaraj, B., Roy, K. Comparison of machine learning and deep learning models for network intrusion detection systems. Future Internet, 2020.

ДОДАТОК А

Код файлу data_arrangement_module.py

```
# Імпорт необхідних бібліотек
import pandas as pd  # Для обробки та аналізу даних
import numpy as np  # Для роботи з масивами
from sklearn import model_selection  # Для розділення даних на навчальні та тестові
from string import printable  # Всі друковані символи ASCII
from keras.preprocessing import sequence  # Для підготовки послідовностей даних

class Arrangement:
    def __init__(self):
        pass

    @staticmethod
    def import_domains():
        """
        Завантаження та попередня обробка даних.

        Функція виконує:
        1) Завантаження даних з директорії
        2) Токенізація на рівні символів
        3) Впорядкування послідовностей до однакової довжини

        Функція повертає:
        - x_train, x_test: навчальні та тестові датасети з доменів
        - y_train, y_test: навчальні та тестові мітки класів
        """
```

```
# Вказуємо шлях до даних
targeted_directory = '../domains_source/'
# Завантажуємо дані з файлу CSV, використовуючи pandas
df = pd.read_csv(targeted_directory + 'shuffled_domains_with_lables.csv', encoding='ISO-8859-1', sep=',')

# Токенізація URL на рівні символів. Кожен символ конвертується у відповідний індекс
# зі списку printable додаванням 1 (щоб уникнути індексу 0, який використовується для впорядкування).
domains_to_tokens = [[printable.index(x) + 1 for x in url if x in printable] for url in df.url]

# Встановлення максимальної довжини URL у 80 символів для впорядкування
max_len = 80

# Виконання впорядкування для послідовностей, щоб всі мали однакову довжину
x = sequence.pad_sequences(domains_to_tokens, maxlen=max_len)

# Ініціалізація порожнього списку для зберігання бінарних міток класів
binary_class_labels = []
# Проходження по кожному елементу колонки 'class' у датафреймі
for i in df['class']:
    # Якщо мітка класу дорівнює 0, додаємо 0 до списку міток
    if i == 0:
        binary_class_labels.append(0)
    # Якщо мітка класу не дорівнює 0, додаємо 1 до списку міток
```

```
            else:
                binary_class_labels.append(1)

        # Конвертація списку бінарних міток в масив NumPy для подальшої обробки
        y = np.array(binary_class_labels)

        # Розділення датасету на тренувальну та тестову вибірки, де 10% даних відводиться під тестування
        x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.1, random_state=33)

        # Повернення тренувальних та тестових датасетів
        return x_train, x_test, y_train, y_test
```

ДОДАТОК Б

Код файлу ml_model_pipeline.py

```
# Імпорт необхідних бібліотек
import time  # Для роботи з часом
import warnings  # Для керування повідомленнями про попередження
from datetime import datetime  # Для роботи з датами та часом
import tensorflow as tf  # Для роботи з моделями на основі TensorFlow
from keras import regularizers  # Для регуляризації параметрів моделі
from keras.callbacks import EarlyStopping  # Для зворотнього виклику для моделі Keras: рання зупинка
from keras.callbacks import ModelCheckpoint  # Для зворотнього виклику для моделі Keras: збереження найкращої моделі

# Шари для роботи з векторними представленнями слів (word embeddings) та текстовими даними
from keras.layers import Embedding, LSTM, Bidirectional  # Шар вбудовування, шар LSTM, шар BiLSTM

# Шари для обробки послідовних даних: вхідний шар, шар функції активації ELU, шар пакетної нормалізації,
# згортковий шар для обробки послідовних даних, шар максимального зведення для обробки послідовних даних,
# шар для об'єднання виходів різних шарів
from keras.layers import Input, ELU, BatchNormalization, Convolution1D, MaxPooling1D, concatenate

# Шари для побудови повнозв'язаних (dense) шарів нейронної мережі
```

```
from keras.layers import Dense, Dropout, Flatten  # Шар для побудови повнозв'язаних шарів, шар для випадкового
# відключення нейронів для регуляризації, шар для перетворення вхідних даних в одновимірний вектор

from keras.models import Model  # Для побудови моделі Keras
from keras.optimizers import Adam  # Для вибору та налаштування оптимізатора для навчання моделі
from keras_self_attention import SeqSelfAttention  # Для реалізації шару самоуваги в SeqSelfAttention плагіні
from results_estimation_module import Estimation  # Для оцінки та візуалізації результатів роботи моделі
from data_arrangement_module import Arrangement  # Для завантаження та підготовки даних перед поданням до моделі

# Ігнорування попереджень для чистоти виводу
warnings.filterwarnings("ignore")

# Налаштування TensorFlow для оптимального використання пам'яті на GPU, щоб уникнути переповнення
gpus = tf.config.experimental.list_physical_devices('GPU')

if gpus:
    try:
        # Увімкнення динамічного виділення пам'яті для кожного GPU
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
    except RuntimeError as e:
        # Виведення помилок, пов'язаних з конфігурацією GPU
        print(e)
```

```
with tf.device("/GPU:0"):

    # Функція для створення згорткового шару
    def build_convolution_layer(emb, kernel_size=5, filters=256):
        # Створення згорткового шару з використанням заданих параметрів
        conv_layer = Convolution1D(kernel_size=kernel_size, filters=filters, padding='same')(emb)
        conv_layer = ELU()(conv_layer)  # Функція активації
        conv_layer = MaxPooling1D(5)(conv_layer)  # Максимальне зведення
        conv_layer = Dropout(0.5)(conv_layer)  # Dropout для регуляризації
        return conv_layer

    # Функція для визначення гібридної архітектури моделі
    def define_hybrid_architecture(max_len=80, emb_dim=32, max_vocab_len=128, w_reg=regularizers.l2(1e-4)):

        # Визначення вхідного шару
        input_data = Input(shape=(max_len,), dtype='int32', name='main_input')

        # Шар вбудовування слів
        word_embeddings = Embedding(input_dim=max_vocab_len, output_dim=emb_dim, input_shape=max_len,
                                    embeddings_regularizer=w_reg)(input_data)
        word_embeddings = Dropout(0.2)(word_embeddings)  # Dropout для регуляризації

        # Побудова згорткових шарів з різними ядрами
        conv_layer2 = build_convolution_layer(word_embeddings, kernel_size=2, filters=256)
```

```
        conv_layer3 = build_convolution_layer(word_embeddings, kernel_size=3, filters=256)
        conv_layer4 = build_convolution_layer(word_embeddings, kernel_size=4, filters=256)
        conv_layer5 = build_convolution_layer(word_embeddings, kernel_size=5, filters=256)

        # Побудова двонаправленого LSTM шару
        bi_lstm_layer = Bidirectional(LSTM(units=128, return_sequences=True))(word_embeddings)

        # Шар самоуваги
        self_attention_layer = SeqSelfAttention(attention_activation='relu')(bi_lstm_layer)

        # Об'єднання виходів різних шарів
        cnn_lstm_combined = concatenate([conv_layer2, conv_layer3, conv_layer4, conv_layer5,
                                         self_attention_layer], axis=1)
        cnn_lstm_combined = Flatten()(cnn_lstm_combined)

        # Побудова повнозв'язаних шарів нейронної мережі
        fc_layer1 = Dense(4128)(cnn_lstm_combined)
        fc_layer1 = ELU()(fc_layer1)
        fc_layer1 = BatchNormalization()(fc_layer1)
        fc_layer1 = Dropout(0.5)(fc_layer1)

        fc_layer2 = Dense(1024)(fc_layer1)
        fc_layer2 = ELU()(fc_layer2)
        fc_layer2 = BatchNormalization()(fc_layer2)
```

```
        fc_layer2 = Dropout(0.5)(fc_layer2)

        # Вихідний шар (останній повнозв'язаний шар)
        output_data = Dense(1, activation='sigmoid', name='output')(fc_layer2)

        # Компіляція моделі та визначення оптимізатора
        hybrid_model = Model(inputs=[input_data], outputs=[output_data])

        return hybrid_model

    # Використання GPU для обчислень
    with tf.device("/GPU:0"):

        # Завантаження даних за допомогою модулю обробки даних
        x_train, x_test, y_train, y_test = Arrangement.import_domains()

        # Визначення гібридної моделі
        model_name = "Гібридна модель CNN-BiLSTM-Att"
        model = define_hybrid_architecture()

        # Визначення ранньої зупинки
        early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
        checkpoint_monitor = ModelCheckpoint(filepath='./trained_models/' + model_name + '.keras', monitor='val_loss', mode='min', save_best_only=True, verbose=1)

        ''' Фаза тренування '''
        num_epochs = 10  # Кількість епох тренування
        size_of_batch = 64  # Розмір пакету для тренування
```

```
        # Визначення оптимізатора Adam з параметрами
        adam_optimizer_instance = Adam(learning_rate=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)

        # Компіляція моделі з використанням оптимізатора Adam та метрик
        model.compile(optimizer=adam_optimizer_instance, loss='binary_crossentropy',
                metrics=Estimation.get_metrics())

        # Отримання поточного часу перед початком тренування моделі
        training_start_time = datetime.now()

        # Тренування моделі з використанням підготованих даних
        training_history = model.fit(x_train, y_train, epochs=num_epochs, batch_size=size_of_batch, validation_split=0.11, callbacks=[early_stop, checkpoint_monitor])

        # Отримання поточного часу після завершення тренування моделі
        training_end_time = datetime.now()

        # Фаза передбачення
        chief_model = define_hybrid_architecture()
        chief_model.load_weights('./trained_models/' + model_name + '.keras')
        chief_model.compile(optimizer=adam_optimizer_instance, loss='binary_crossentropy', metrics=['accuracy', Estimation.precision_metric, Estimation.recall_metric, Estimation.fmeasure_metric])

        # Отримання поточного часу перед початком передбачення результатів
        prediction_start_time = datetime.now()

        # Передбачення результатів на тестових даних
```

```
        predicted_results = chief_model.predict(x_test, batch_size=64)

        # Отримання поточного часу після завершення передбачення результатів
        prediction_end_time = datetime.now()

        # Візуалізація та оцінка результатів
        Estimation.generate_validation_plots(model_name, training_history)
        Estimation.display_validation_results(training_history)

        # Обчислення метрик моделі та їх вивід
        Estimation.compute_model_metrics(chief_model, x_test, y_test)
        print("\t")

        # Візуалізація матриці невідповідностей без нормалізації
        Estimation.visualize_matrix(model_name, y_test, predicted_results,
                                    title='Матриця невідповідностей без нормалізації',
normalize=False, classes=[0, 1])
        time.sleep(5)

        # Візуалізація нормалізованої матриці невідповідностей
        Estimation.visualize_matrix(model_name, y_test, predicted_results,
title='Нормалізована матриця невідповідностей',
                                    normalize=True, classes=[0, 1])
        print("\t")

        # Вивід часу тренування та передбачення
        print('Час тренування: ' + str((training_end_time - training_start_time)))
        print('Час передбачення: ' + str((prediction_end_time - prediction_start_time)))
```

ДОДАТОК В

Код файлу results_estimation_module.py

```
# Імпорт необхідних бібліотек
import tensorflow as tf
import matplotlib.pyplot as plt  # Для створення графіків та візуалізації даних
import numpy as np  # Для обробки масивів та виконання математичних операцій
from sklearn import metrics  # Для виконання оцінки моделі
from sklearn.metrics import classification_report  # Для генерації звіту з основними метриками класифікації
from sklearn.metrics import confusion_matrix  # Для створення матриці невідповідностей
from datetime import datetime  # Для роботи з датами і часом

class Estimation:
    precision_metric = tf.keras.metrics.Precision()
    recall_metric = tf.keras.metrics.Recall()

    # Статичні методи класу Estimation для обчислення метрик влучності, повноти та F1-показника
    @staticmethod
    def update_metrics(y_true, y_pred):
        Estimation.precision_metric.update_state(y_true, tf.round(y_pred))
        Estimation.recall_metric.update_state(y_true, tf.round(y_pred))

    @staticmethod
    def fmeasure_metric(y_true, y_pred):
        Estimation.update_metrics(y_true, y_pred)
        precision = Estimation.precision_metric.result()
```

```
        recall = Estimation.recall_metric.result()
        f1_score = 2 * (precision * recall) / (precision + recall + tf.keras.backend.epsilon())
        return f1_score

    @staticmethod
    def get_metrics():
        return [
            tf.keras.metrics.BinaryAccuracy(),
            Estimation.precision_metric,
            Estimation.recall_metric,
            Estimation.fmeasure_metric
        ]

    @staticmethod
    def generate_validation_plots(model_name, history):
        # Зберігаємо криві перевірки у форматі PNG

        history_dict = history.history  # Отримуємо історію тренування моделі
        print(history_dict.keys())  # Виводимо ключі доступних даних

        epochs = range(1, len(history_dict['loss']) + 1)  # Визначаємо кількість епох

        plt.figure(figsize=(10, 6))  # Встановлюємо розмір графіку у дюймах (ширина, висота)

        # Побудова графіків для різних метрик
        plt.plot(epochs, history_dict['val_binary_accuracy'], 'b', label='точність')
        plt.plot(epochs, history_dict['val_precision'], 'g', label='влучність')
        plt.plot(epochs, history_dict['val_recall'], 'k', label='повнота')
```

```
        plt.plot(epochs, history_dict['val_fmeasure_metric'], 'r', label='F1-показник')

        plt.title(f'Криві перевірки для {model_name}')  # Заголовок графіку
        plt.xlabel('Епохи')  # Підпис осі X
        plt.ylabel('Значення метрик')  # Підпис осі Y
        plt.grid()  # Включення сітки
        plt.legend(loc='lower right')  # Розміщення легенди

        now = datetime.now()
        now_datetime = now.strftime('%Y_%m_%d-%H%M%S')  # Форматування поточної дати і часу

        plt.axis([0, len(epochs), 0, 1])  # Встановлюємо масштаб по обом осях

        # Зберігаємо зображення графіку у файл PNG в папці 'final_directory' з унікальною міткою часу створення
        plt.savefig('./final_directory/' + model_name + '_val_curve_' + now_datetime + '.png')

    @staticmethod
    def display_validation_results(history):
        """ Вивід історії перевірки під час навчання моделі """
        history_dict = history.history  # Доступ до історії метрик навчання моделі

        # Проходження по всіх ключах в історії та вивід значень для ключів, що містять 'val' (валідаційні метрики)
        for key in history_dict:
            if "val" in key:
                print('[' + key + '] ' + str(history_dict[key]))
```

```
    @staticmethod
    def compute_model_metrics(model, x_test, y_test):
        """ Підрахунок основних метрик для оцінки моделі на тестових даних """

        y_pred_class_prob = model.predict(x_test, batch_size=64, verbose=0)  # Прогнозування ймовірностей класів
        y_pred_class = (y_pred_class_prob >= 0.5).astype(int)  # Конвертація ймовірностей у бінарні класи

        y_true_class = y_test  # Реальні класи для порівняння

        # Вивід звіту про класифікацію, який включає точність, влучність, повноту і F1-показник для кожного класу
        print(classification_report(y_true_class, y_pred_class, digits=4))

        # Вивід різних метрик моделі
        print('Точність:', metrics.accuracy_score(y_true_class, y_pred_class))
        print("Влучність:", metrics.precision_score(y_true_class, y_pred_class, average='weighted'))
        print("Повнота:", metrics.recall_score(y_true_class, y_pred_class, average='weighted'))
        print("F1-показник:", metrics.f1_score(y_true_class, y_pred_class, average='weighted'))

    @staticmethod
    def visualize_matrix(model_name, y_true, y_pred, classes=[0, 1], normalize=False, title=None, cmap=plt.cm.Blues):
        """ Збереження та візуалізація матриці невідповідностей """
```

```
    binary_labels_dict = {'benign': 0, 'malicious': 1}  # Словник для перетворення назв класів у числа

    # Створення списку строкових назв класів на основі їхніх числових значень
    classes_str = []
    for i in classes:
        for dga_str, dga_int in binary_labels_dict.items():
            if dga_int == i:
                classes_str.append(dga_str)

    # Створення списку строкових назв класів на основі їхніх числових значень
    y_pred_class = (y_pred >= 0.5).astype(int)
    y_true_class = y_true  # Реальні класи

    if not title:
        if normalize:
            title = 'Нормалізована матриця невідповідностей'
        else:
            title = 'Матриця невідповідностей без нормалізації'

    # Обчислення матриці невідповідностей
    cm = confusion_matrix(y_true_class, y_pred_class)
    accuracy = np.trace(cm) / float(np.sum(cm))  # Обчислення загальної точності
    misclass = 1 - accuracy   # Обчислення показника неправильної класифікації

    # Умова перевіряє чи потрібно нормалізувати матрицю невідповідностей
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Нормалізована матриця невідповідностей успішно згенерована.")
    else:
```

```
    print('Матриця невідповідностей без нормалізації успішно згенерована.')

    # Створення об'єкта для малюнка і осей графіка
    fig, ax = plt.subplots()

    # Відображення матриці невідповідностей з використанням кольорової шкали
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)

    # Додавання шкали кольорів до графіка
    ax.figure.colorbar(im, ax=ax)

    # Встановлення міток для кожної осі з відповідними назвами класів
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes_str, yticklabels=classes_str,
           title=title,
           ylabel='Дійсні значення',
           xlabel='Прогнозовані значення')

    # Поворот міток на осях для встановлення кращої видимості
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    # Додавання текстових анотацій для кожного елементу в матриці
    fmt = '.3f' if normalize else 'd'  # Вибір формату чисел в залежності від нормалізації
    thresh = cm.max() / 2.  # Визначення порогу для зміни кольору тексту для кращої читабельності
```

```
            for i in range(cm.shape[0]):
                for j in range(cm.shape[1]):
                    ax.text(j, i, format(cm[i, j], fmt),
                            ha="center", va="center",
                            color="white" if cm[i, j] > thresh else "black")  # Зміна кольору тексту залежно від значення
            fig.tight_layout()  # Автоматичне коригування розмірів компонентів графіка для оптимального відображення

            # Додавання нижнього підпису з основними метриками
            precision = metrics.precision_score(y_true_class, y_pred_class, average='weighted')
            recall = metrics.recall_score(y_true_class, y_pred_class, average='weighted')
            f1 = metrics.f1_score(y_true_class, y_pred_class, average='weighted')

            plt.xlabel('Прогнозовані значенні\nточність={:0.4f}; влучність={:0.4f}; повнота={:0.4f};'
                       'F1-показник={:0.4f};\nпоказник неправильної класифікації={:0.4f}'
                       .format(accuracy, precision, recall, f1, misclass))

            # Отримання поточного часу для створення унікального імені файлу
            now = datetime.now()
            now_datetime = now.strftime('%Y_%m_%d-%H%M%S')
            figure = plt.gcf()  # Отримання поточного графічного об'єкта
            figure.set_size_inches(15, 15)  # Встановлення розміру зображення

            # Збереження зображення графіка в файл
            plt.savefig('./final_directory/' + model_name + '_confusion_matrix_' + now_datetime + '.png', dpi=100)
```

ДОДАТОК Г

Фрагмент вмісту файлу shuffled_domains_with_lables.csv

```
url,label,class
66a08595d934d3cc1634d24239eae525.xyz,malware,1
sgjxestnessbiophysicalohax.com,malware,1
jzcmestnessbiophysicalohax.com,malware,1
jmstcbobrkcx.rent,malware,1
muucpgsjsstt.co.in,malware,1
irs.gov,benign,0
oynsorncdeeu.net,malware,1
rottentomatoes.com,benign,0
fjumestnessbiophysicalohax.com,malware,1
qmuexxgnsgoo.biz,malware,1
buffalo.edu,benign,0
sydney.edu.au,benign,0
xprcnjxqryyi.biz,malware,1
cnfkkkdeneyx.net,malware,1
scribd.com,benign,0
news-medical.net,benign,0
tunein.com,benign,0
towardsdatascience.com,benign,0
imperial.ac.uk,benign,0
fcderreiptcc.us,malware,1
pewinternet.org,benign,0
amazon.co.jp,benign,0
ycwgmen.com,malware,1
mviynnclyyny.me.uk,malware,1
```

ДОДАТОК Ґ

Копія наукової публікації «Виклики та перспективи використання машинного навчання у запобіганні доменним атакам», VII Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві»

Бучик С. С.
Київський національний університет
імені Тараса Шевченка, м. Київ
buchyk@knu.ua
Шматко В. О.
Київський національний університет
імені Тараса Шевченка, м. Київ
vika.shmatko.24@knu.ua

ВИКЛИКИ ТА ПЕРСПЕКТИВИ ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ У ЗАПОБІГАННІ ДОМЕННИМ АТАКАМ

Abstract: In an increasingly interconnected digital landscape, the security of domains has become paramount. The escalating threat of domain attacks has prompted cybersecurity researchers and professionals to explore innovative approaches to bolster defenses. Among these approaches, the strategic utilization of machine learning has emerged as a promising avenue for mitigating the risks posed by such attacks. By harnessing the power of machine learning algorithms, organizations aim to enhance their ability to detect and prevent malicious activities targeting domain names. However, as with any evolving technology, this endeavor is not without its challenges. Exploring the realm of leveraging machine learning for domain attack mitigation, this exploration sheds light on both the potential benefits and the complex problems that define this rapidly evolving landscape.

Keywords: Domain Name System (DNS), machine learning, malicious domains, DNS attacks.

В умовах цифрової взаємодії система доменних імен (Domain Name System, DNS) має невід’ємне значення для оптимальної роботи Інтернету. DNS передбачає

стандартну домовленість про найменування між зрозумілими для людини назвами доменів та машинно-маршрутизованими адресами Інтернет-протоколу [1].

У зв’язку з постійною еволюцією кіберзагроз хакери використовують все більш вдосконалені способи компрометації доменів, що призводить до витоку даних, фінансових збитків та підриву довіри користувачів. Захист від таких атак вимагає активного й адаптивного підходу, який здатний швидко ідентифікувати та нейтралізувати нові загрози. Саме тут набуває важливості застосування методів машинного навчання (Machine Learning, ML), типу штучного інтелекту, який дозволяє програмним рішенням точніше прогнозувати результати без явного програмування для цього [2]. Використовуючи переваги аналізу даних, розпізнавання закономірностей та прогнозного моделювання, ML постає ефективним способом щодо захисту доменів від шахрайських намірів.

Впровадження машинного навчання в аналіз трафіку DNS відкриває можливість передчасної протидії атакам, від фішингу до тунелювання DNS. Цей підхід передбачає навчання моделей на обширному наборі даних, який охоплює спектр легітимних та зловмисних доменних активностей [2]. Завдяки постійному аналізу DNS-трафіку моделі машинного навчання можуть виявляти витончені відхилення від встановлених шаблонів, що вдосконалюється завдяки вивченню історичних даних. Розпізнавши ознаки атак, такі як раптове збільшення кількості запитів чи швидкі зміни щодо власності домену, ці моделі таким чином набувають вміння відзначати підозрілі параметри для подальшого аналізу.

Шляхом впровадження навчання в реальному часі моделі можуть адаптуватися до нових векторів атак, забезпечуючи перевагу в цьому процесі. Однак людське втручання залишається незамінною складовою цього підходу. Офіцери з кібербезпеки, знаючи інсайти, що надаються моделями ML, можуть розглянути прапорці підозрілих активностей, розрізнити потенційні загрози від доброзичливих аномалій. Така співпраця між штучним інтелектом та людською експертизою приводить до більш точної ідентифікації атак, зменшуючи кількість false positives та false negatives [2].

Синергія між машинним навчанням та безпекою DNS виходить за межі окремих організацій. Колективні набори даних з різних джерел можуть бути анонімізовані та агреговані для підвищення точності та комплексності моделей. Цей підхід дозволяє кібербезпеці розвивати спільне розуміння шаблонів атак, які постійно змінюються.

У ландшафті кібербезпеки, де кожна секунда має значення, машинне навчання сприяє стрімкому усуненню загроз. Зменшуючи час між виявленням та реакцією, компанії будуть ефективно нейтралізувати атаки, запобігаючи несанкціонованим витоку даних та збою обслуговування [3]. Це забезпечить конфіденційність користувачів і неперервність бізнесу, а також буде підтримувати цілісність онлайн-середовища.

Розглянемо детальніше конкретні випадки застосування, виклики та потенційні реалізації ML у запобіганні доменним атакам. Конкретні випадки ілюструють, як машинне навчання може бути застосоване до різних аспектів безпеки доменних імен. Завдяки використанню даних для здобуття інсайтів та розпізнаванню шаблонів, ці методи підвищують здатність попередньо виявляти та зменшувати широкий спектр кіберзагроз, пов’язаних з доменами, що буде сприяти більш безпечному стану онлайн-середовища.

Фішинг (Phishing) – це тип атаки, під час якої відбувається перенаправлення користувачів для введення ними конфіденційної інформації на фіктивному сайті, що контролюється зловмисниками. Ці сайти переважно виглядають ідентично легітимним ресурсам [3]. Моделі ML можуть аналізувати атрибути доменних імен, вмісту електронної пошти, поведінки відправника та метаданих, щоб ідентифікувати патерни, які часто пов’язані з спробами фішингу. Завдяки навчанню на основі відомих випадків такої атаки, ці моделі надалі будуть відзначати підозрілі електронні листи або веб-сайти, допомагаючи користувачам уникнути взаємодії зі зловмисними доменами.

DNS-spoffing – це атака, під час якої зловмисники маніпулюють записами DNS, щоб перенаправити користувачів на шахрайські веб-сайти. Моделі ML можуть відстежувати патерни запитів та відповідей DNS, виявляючи аномалії, які свідчитимуть про спроби обману DNS-перенаправлення [3]. Аналізуючи історичний

трафік DNS та вивчаючи типову поведінку, ці моделі будуть ідентифікувати раптові відхилення та запускати сигнали або захисні дії.

Typosquatting передбачає створення доменних імен, які схожі на легітимні, але містять незначні відмінності або орфографічні неточності у своїй назві. Ці ресурси спрямовані на залучення користувачів, які допускають помилки при наборі тексту або поспішають [4]. Алгоритми ML можуть аналізувати доменні імена на наявність відмінностей та ідентифікувати патерни, що свідчать про спроби атаки. Машинне навчання надає також можливості щодо аналізу частоти вживання символів та порівняння відстаней Левенштейна між реальними іменами та їхніми варіаціями.

У сфері використання машинного навчання для зміцнення безпеки доменів виникають проблеми, які вимагають ретельного розгляду. Першими такими труднощами є якість і кількість даних, оскільки від них значною мірою залежить ефективність моделей ML. Отримання репрезентативних відомостей, які охоплюють різні типи доменних атак, може бути складним завданням [5]. Крім того, точне позначення даних як «зловмисних» або «доброякісних» потребує досвіду та всебічного розуміння динамічної тактики атак.

Наступною проблемою є змагальні атаки, які передбачають навмисне маніпулювання вхідними даними, щоб ввести в оману моделі машинного навчання [5]. Зловмисники можуть налаштовувати шаблони атак, щоб створювати дані, які виглядають легітимними для моделі, але все ще мають зловмисний намір. Це вимагає розробки надійних моделей, які будуть ідентифікувати спроби шахраїв та протистояти ним.

Третім викликом є встановлення балансу між мінімізацією false positives, тобто хибних спрацьовувань (позначення легітимних доменів як зловмисних), і false negatives, тобто хибно негативних результатів (нездатність виявити фактичні атаки) [6]. Надмірно обережні моделі можуть генерувати надлишкову кількість помилкових спрацьовувань, що призведе до нездатності оперативно обробити всі випадки, тоді як занадто ліберальні моделі будуть ставити під загрозу безпеку, дозволяючи реальним атакам залишатися непоміченими.

Труднощі можуть створювати й інтерпретовані моделі. У контексті доменних атак необхідно зрозуміти, чому модель машинного навчання класифікувала домен як зловмисний або легітимний. Така можливість інтерпретації допомагає офіцерам з безпеки приймати зважені рішення та виправляти false positives та false negatives [6]. Однак деякі складні моделі машинного навчання, такі як глибокі нейронні мережі, залишаться складними для розшифрування.

Майбутні реалізації натомість підкреслюють потенційний трансформаційний вплив машинного навчання на запобігання доменним атакам. Еволюція в бік автоматизованих реакцій, здатність захищатися від нових загроз, спільні зусилля щодо забезпечення безпеки, підходи до конфіденційності та важливість постійного навчання формують траєкторію використання ML для захисту доменів у цифровому середовищі.

Розширена поведінкова аналітика на основі машинного навчання зможе виявляти незначні аномалії у взаємодії щодо домену. Це дозволить ідентифікувати незвичайні шаблони поведінки користувачів або мережевого трафіку, сигналізуючи про потенційні атаки домену, такі як викрадання облікових даних. Поведінкова аналітика створює базову лінію очікуваної поведінки, яку можна порівняти з контрольованим експериментом у наукових дослідженнях [7]. Загалом підхід передбачає прискіпливе спостереження за цифровою діяльністю разом із виявленням будь-яких відхилень від встановленої норми.

Захист від атак нульового дня схожий на розширену систему раннього попередження для доменів. Це передбачатиме використання машинного навчання для швидкого виявлення та захисту від раніше невідомих кіберзагроз, які експлуатують вразливості, ще до того, як вони стануть загальновідомими або будуть виправлені [8]. Цей проактивний підхід гарантує, що домени залишаються в безпеці, навіть якщо зловмисники намагаються використати абсолютно нові недоліки, забезпечуючи надійний рівень захисту від ризиків.

Cybersecurity Orchestration, інтегроване з можливостями машинного навчання, втілить системний і науковий підхід до захисту доменів. Підхід може працювати як алгоритмічний провідник, ретельно інтерпретуючи та аналізуючи величезні набори

даних з метою організації добре скоординованої стратегії захисту від атак на домени [8]. Оркестровка, заснована на постійному аналізі мінливих ландшафтів загроз, ефективно керуватиме розгортанням заходів безпеки, щоб гарантувати, що домени залишаються стійкими перед новими загрозами.

Конкретні випадки застосування, виклики та потенційні реалізації ML у запобіганні доменним атакам в узагальненому вигляді представлені в табл.1.

Таблиця 1 – Випадки застосування, виклики та потенційні реалізації ML у запобіганні доменним атакам

№ з/п	**Випадки застосування ML**	**Виклики ML**	**Потенційні реалізації ML**
1	Виявлення фішингу: проведення аналізу даних, таких як доменні імена, вміст електронної пошти та поведінки адресанта.	Якість даних: існує вимога в точних й обширних відомостях, а також експертизі та розумінні стратегій атак.	Поведінкова аналітика на базі машинного навчання виявлятиме навіть невеликі аномалії в користувацькій чи мережевій активності, випадки яких сигналізуватимуть про можливі атаки.
2	Попередження DNS-Spoofing: виявлення спроб фальсифікації DNS-трафіку і перенаправлення на зловмисні сайти, аналізуючи патерни запитів та відповідей DNS.	Загроза змагальних атаки: над вхідними даними можуть виконуватися маніпуляції для обману моделей ML, тому існує потреба у створенні надійних моделей.	Захист від атак нульового дня передбачає раннє виявлення та захист від нових кіберзагроз завдяки ML. Такий проактивний підхід гарантує безпеку доменів навіть від абсолютно нових загроз.
3	Захист від Typosquatting: розкриття спроб атаки, яка полягає в створенні схожих на легітимні доменних імен з орфографічними помилками.	Баланс між false positives і false negatives: занадто обережні моделі можуть перевантажити систему, а ліберальні – загрожувати безпеці через упущення особливо важливої інформації.	Кібербезпека з оркестровкою та впровадженням машинного навчання представляє системний підхід до захисту доменів, який використовує аналіз даних для координування заходів безпеки.

Висновки. Отже, використання машинного навчання у запобіганні доменним атакам відкриває нові горизонти в забезпеченні кібербезпеки. ML дозволяє виявляти навіть найбільш витончені атаки та реагувати на них в реальному часі, що підвищує рівень безпеки доменних зон. Проте успішне впровадження цієї технології вимагає постійного моніторингу, оновлення моделей та уважного врахування етичних

аспектів щодо наданих відомостей моделям. Майбутні виклики вимагають подальшого розвитку заходів безпеки на основі машинного навчання, які стануть невід'ємною частиною стратегії протидії новим кіберзагрозам.

ЛІТЕРАТУРА:

1. Adiwal, Sanjay & Rajendran, Balaji & Shetty, Pushparaj: Domain Name System (DNS) Security: Attacks Identification and Protection Methods. International Conference on Security and Management, Las Vegas, USA, 2018.

2. Nguyen Q., Laborde R., Benzekri A. and Qu'hen B.: Detecting abnormal DNS traffic using unsupervised machine learning. 4th Cyber Security in Networking Conference (CSNet), 2020.

3. Bilge L., Sen S., Balzarotti D., Kirda E., Kruegel C.: EXPOSURE: A Passive DNS Analysis Service to Detect and Report Malicious Domains. ACM Transactions on Information and System Security, 2014.

4. Moubayed A., Injadat M., Shami A., Lutfiyya H.: DNS Typo-Squatting Somain Setection: A Data Analytics & Machine Learning Based Approach. IEEE Global Communications Conference (GLOBECOM), 2018.

5. Banadaki Yaser M.: Detecting malicious DNS over HTTPS traffic in Domain Name System Using Machine Learning Classifiers. Journal of Computer Sciences and Applications, 2020.

6. Yury Zhauniarovich, Issa Khalil, Ting Yu, and Marc Dacier: A Survey on Malicious Domains Detection through DNS Data Analysis. ACM Computing Surveys, 2018.

7. Houser R.: Investigations of the Security and Privacy of the Domain Name System, 2021.

8. Shaikh A., Pardeshi B., Dalvi F.: Overcoming Threats and Vulnerabilities in DNS. Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST), 2020.

ДОДАТОК Д

Копія наукової публікації «Strategies for dataset collection in domain attack prevention with machine learning», X International Conference «Information Technology and Implementation» (IT&I-2023) Satellite

[1] Serhii Buchyk

Doctor of Technical Sciences, Professor at the Department of Cyber Security and Information Protection, Faculty of Information Technologies

[2] Viktoriia Shmatko

Student at the Department of Cyber Security and Information Protection, Faculty of Information Technologies

[1,2] Taras Shevchenko National University of Kyiv

STRATEGIES FOR DATASET COLLECTION IN DOMAIN ATTACK PREVENTION WITH MACHINE LEARNING

Abstract. The emergence of machine learning (ML) as a powerful tool in the arsenal against cyberattacks has sparked intense interest in its application for domain attack prevention. However, the efficacy of machine learning models in thwarting these threats hinges crucially on the quality and suitability of the datasets used for training. In this context, the collection of datasets takes center stage as critical components in the development of robust defense strategies.

Keywords: machine learning, datasets, Domain Name System (DNS), domain name attacks.

In the contemporary digital landscape, the prevalence of cyberattacks targeting specific domains has reached alarming proportions. From domain spoofing and DNS hijacking to targeted phishing campaigns and malware delivery, malicious actors continually refine their techniques to infiltrate organizations, compromise sensitive data, and disrupt critical services. These domain-specific attacks often exploit vulnerabilities in an organization's web infrastructure, email systems, or network protocols, making them particularly challenging to detect and mitigate [1].

Amid this growing threat, machine learning has emerged as a potent ally in the fight against domain attacks. Its ability to analyze vast datasets, identify subtle patterns, and adapt to evolving threats positions it as a key component of proactive cybersecurity strategies [2]. However, the true efficacy of machine learning in domain attack prevention hinges decisively on the quality and appropriateness of the datasets used to train and fine-tune these models.

In the context of domain attack prevention, understanding the diverse landscape of attacks is paramount. For instance, domain spoofing involves creating fake websites or emails that mimic legitimate domains, making it imperative to include datasets that encompass a wide array of such spoofing techniques. Similarly, DNS hijacking attacks manipulate the domain name system to redirect traffic, necessitating datasets that can capture anomalies in DNS resolutions.

Phishing, one of the most common domain attacks, relies on luring users into revealing sensitive information through deceptive emails or websites. Datasets must, therefore, encompass a multitude of phishing attempts, capturing variations in content, tactics, and targets [3].

Moreover, malware delivery through compromised domains demands a rich dataset of malicious payloads, alongside benign software, to enable machine learning models to distinguish between the two effectively. Additionally, domain attacks can take on various forms, from drive-by downloads to credential theft attempts, each requiring tailored datasets for comprehensive model training.

Machine learning, a transformative field in the realm of artificial intelligence, empowers computers to learn from data and adapt their behavior without explicit programming. The journey from raw data to a sophisticated machine learning model involves a series of meticulously orchestrated steps, each playing a pivotal role in the creation and deployment of intelligent systems [4]. These steps constitute the machine learning pipeline, starting with data collection and preprocessing to ensure data quality. Each step, from model selection to deployment and monitoring, contributes to developing models for data-driven predictions. Proficiency in these steps is vital for unlocking machine learning's potential, solving real-world challenges, and driving innovation. A meticulous

approach begins with precise data collection and rigorous preprocessing, setting the foundation for the entire process.

The dataset collection step involves assembling a comprehensive dataset that includes diverse instances of benign and malicious activities within the target domain. This dataset must accurately reflect real-world scenarios, encompass various attack patterns, and be regularly updated to adapt to evolving threats. A high-quality and representative dataset is crucial for training machine learning models to effectively detect and prevent domain attacks [5].

To effectively combat threats, a meticulous dataset collection process is of utmost importance, forming the cornerstone of developing robust machine learning models. Within this context, the focus turns to domain names, which serve as a critical component of the digital ecosystem and are often a prime target for malicious activities.

The specific strategies are presented tailored for the collection of domain name-related data, each meticulously designed to enhance the efficacy of machine learning:

1) Attack Scenario Enumeration: initiate the process by systematically listing potential attack scenarios and threat vectors tailored to specific domains. This enumeration serves as a compass for the selection of data sources and the acquisition of pertinent data.

2) Data Collection Points: set up data collection points at critical junctures within your network or system architecture. For instance, capture data at network gateways, server logs, application endpoints, and user access points [6].

3) Network Traffic Data: collect raw network traffic data at the packet level. Analyzing packet data helps in identifying intrusion attempts, data exfiltration, and network anomalies.

4) Passive DNS Data: historical DNS resolutions, obtained through passive DNS data, offer insights into domain names associated with past attacks or malicious behavior. This retrospective view aids in identifying patterns and trends of malicious domain usage.

5) WHOIS Data: Acquiring WHOIS data for registered domain names is indispensable for understanding domain ownership and registration details. This information becomes an essential resource for tracking down malicious domain registrants, thus enhancing the dataset's forensic capabilities.

6) Domain Reputation Services: leveraging domain reputation services and threat intelligence feeds enriches the dataset with information on known malicious domain names. These services offer valuable insights into blacklisted domains and bolster the dataset's coverage [6].

7) Anomaly Detection Logs: Logs tracking anomalies in domain name resolutions, such as unusual spikes in traffic to a particular domain, play a crucial role in identifying ongoing attacks. Such insights enhance the dataset's capacity to spot irregular patterns.

8) Domain Generation Algorithm Data: Including data related to Domain Generation Algorithms used in malware and botnet command and control helps in identifying algorithmically generated domain names, which are often used in malicious activities.

9) Botnet Traffic Data: Capturing data related to botnet communication is essential since many domain attacks involve botnets. This data includes domains used by botnets for command and control purposes, aiding in their identification [6].

10) Data Cleansing: Regularly cleaning the dataset to remove expired or outdated domain entries and correcting inaccuracies in domain name records maintains data integrity and model performance.

11) GeoIP Data: Integration of GeoIP data allows mapping of IP addresses to geographical locations, which is instrumental in identifying suspicious or malicious domain activities originating from specific regions.

12) SSL/TLS Certificate Data: Including data on SSL/TLS certificates used with domain names helps in identifying secure connections and potential threats associated with specific domains.

Implemented thoughtfully, these domain-specific strategies create a dataset that empowers machine learning models to effectively prevent and combat domain attacks. With this rich and tailored data, these models become proactive guardians against malicious activities, enhancing digital security in the face of evolving cyber threats.

References:

1. Jin Y., Tomoishi M., Matsuura S. A detection method against DNS cache poisoning attacks using machine learning techniques: Work in progress, Proc. IEEE 18th Int. Symp. Netw. Comput. Appl., 2019.

2. Pacheco Y., Sun W. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets. ICISSP, 2021.

3. Alam M. N., Sarma D., Lima F. F., Saha I., Hossain, S. Phishing attacks detection using machine learning approach, Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT), 2020.

4. Ahmad T., Aziz M. N. Data preprocessing and feature selection for machine learning intrusion detection systems. ICIC Express Lett, 2019.

5. Roh Y., Heo G., Whang S. E. A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. IEEE Transactions on Knowledge and Data Engineering, 2019.

6. Gupta R., Tanwar S., Tyagi S., Kumar N. Machine learning models for secure data analytics: A taxonomy and threat model. Computer Communications, 2020.

ДОДАТОК Е

Копія наукової публікації «Використання метрик оцінки алгоритмів для моделей виявлення шахрайських доменів», VII Міжнародна науково-практична конференція «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (PCSITS)

Використання метрик оцінки алгоритмів для моделей виявлення шахрайських доменів

Viktoriia Shmatko[1], Serhii Buchyk[2]

1. Кафедра кібербезпеки та захисту інформації, Київський національний університет імені Тараса Шевченка, УКРАЇНА, м.Київ, вул.Володимирська, 60, Kyiv, Bohdan Gavrilishina Street, 24,

E-mail: vika.shmatko.24@gmail.com

2. Кафедра кібербезпеки та захисту інформації, Київський національний університет імені Тараса Шевченка, УКРАЇНА, м.Київ, вул.Володимирська, 60, Kyiv, Bohdan Gavrilishina Street, 24,

E-mail: buchyk@knu.ua

Коротка анотація – The digital age, characterized by an exponential increase in online activities, has also seen a parallel rise in cyber threats, among which domain fraud stands out due to its potential to cause significant harm to individuals and organizations alike. The sophistication of such fraudulent schemes necessitates equally advanced countermeasures, primarily in the form of detection algorithms that can identify and mitigate these threats before they inflict damage.

The development and refinement of these algorithms hinge critically on the use of robust evaluation metrics. These metrics serve as the cornerstone for assessing the performance and efficacy of domain fraud detection models, guiding researchers and cybersecurity practitioners in enhancing their detection capabilities. Through the examination of various metrics, their application, and impact, the aim is to illuminate the intricacies of effectively combating domain fraud, thereby contributing to a safer online environment for users worldwide.

Ключові слова – шахрайські доменні імена, машинне навчання, алгоритми, метрики оцінки, точність, влучність, повнота, F1-показник.

Вступ

Шахрайські домени є однією з найпоширеніших загроз у кіберпросторі, які несуть потенційні ризики викрадення особистих даних, фінансових втрат та поширення шкідливого програмного забезпечення. У зв’язку з цим розробка та вдосконалення моделей з виявлення таких доменів стає важливим завданням у кібербезпеці. Використання передових практик машинного навчання та аналізу даних дозволяє автоматизувати цей процес, покращуючи захист користувачів та попереджуючи потенційні кібератаки.

У вирішенні цього завдання важливе розуміння та застосування метрик оцінки алгоритмів. Точність, влучність, повнота та F1-показник є ключовими метриками, які дозволяють оцінити ефективність моделей для виявлення шахрайських доменів. Ці параметри надають можливість об’єктивно визначати якість класифікації моделі, враховуючи різноманітність шахрайських атак та їхні варіації [1].

Під час розробки та тестування моделей важливо аналізувати ці метрики для визначення оптимальних показників та покращення їхньої ефективності. Такий підхід дозволяє не лише створювати надійні інструменти виявлення шахрайських доменів, але й удосконалювати їх у відповідності до змін у царині кіберзлочинності. Крім того, постійний моніторинг та оновлення моделей є критичними складовими стратегії забезпечення кібербезпеки на постійно змінному ландшафті Інтернету.

Категорії результатів класифікації

В оцінці ефективності алгоритмів для виявлення шахрайських доменів ключову роль мають чотири основні терміни класифікації: True Positive (TP), True Negative (TN), False Positive (FP) та False Negative (FN). Ці терміни визначають результати класифікації і допомагають зрозуміти, наскільки ефективно алгоритм впорався з розпізнаванням шахрайських доменів.

True Positives – це випадки, коли модель правильно ідентифікувала злочинні домени як злочинні. TP вказує на те, що система правильно виявляє реальні загрози, забезпечуючи необхідний захист користувачів від потенційно шкідливих доменів.

Значна кількість TP свідчить про високу чутливість моделі до виявлення фактичних загроз [2].

True Negatives – це ситуації, коли модель правильно ідентифікує легітимні домени як нешахрайські. Висока кількість TN свідчить про здатність моделі дозволяти безпечний контент без надмірних обмежень, забезпечуючи при цьому захист від шахрайських сайтів. TN необхідні для підтримки користувацького досвіду, мінімізуючи кількість помилкових блокувань легітимних ресурсів. Також, TN вказують на те, що модель не перешкоджає доступу до перевірених доменів, що є важливим для забезпечення безперебійної роботи інтернет-сервісів.

False Positives – це результати, коли модель помилково ідентифікує легітимні домени як шахрайські. FP можуть призводити до невиправданих обмежень для користувачів, блокуючи доступ до безпечних сайтів. Високий рівень FP може негативно вплинути на користувацький досвід та довіру до системи безпеки [2].

False Negatives – це випадки, коли модель не виявляє шахрайські домени, помилково класифікуючи їх як легітимні. FN становлять серйозну загрозу, оскільки дозволяють шахрайським доменам залишатися невиявленими, що може призвести до фінансових втрат або витоку даних. Мінімізація FN є ключовим аспектом підвищення надійності моделей виявлення шахрайських доменів, впроваджуючи превентивний захист користувачів.

Застосування метрик

Такі метрики, як точність, влучність, повнота та F1-показник, дозволяють зрозуміти, наскільки правильно та повноцінно алгоритм класифікує дані перед прийняттям інформованих рішень щодо домену.

Точність (accuracy) вимірює відсоток випадків, коли модель правильно класифікує як шахрайські, так і легітимні домени. Це загальний показник ефективності моделі, що враховує як True Positives (TP), так і True Negatives (TN) відносно загальної кількості випадків. Висока точність свідчить про те, що модель надійно визначає і злочинні, і безпечні домени. Однак вона може бути оманливою в незбалансованих наборах даних, де один клас значно переважає інший. У таких випадках важливо також розглядати інші метрики для повної оцінки моделі [3].

Влучність (precision) визначає відсоток правильно ідентифікованих шахрайських доменів з усіх доменів, класифікованих моделлю як шахрайські. Цей показник важливий для оцінки якості роботи моделі, особливо в умовах, коли важливо мінімізувати кількість False Positives (FP). Висока влучність означає, що модель рідко помилково маркує легітимні домени як шахрайські, що критично для забезпечення безперебійного доступу користувачів до інтернет-ресурсів. Однак, фокусування лише на влучності може призвести до зниження чутливості моделі до виявлення реальних шахрайських доменів.

Повнота (recall) або чутливість, з'ясовує відсоток реальних шахрайських доменів, правильно ідентифікованих моделлю. Це показує, наскільки добре модель здатна виявити всі можливі шахрайські домени, мінімізуючи кількість False Negatives. Висока повнота є особливо важливою в умовах, де упущення шахрайських доменів може мати серйозні наслідки. Однак, прагнення до максимальної повноти може призвести до збільшення False Positives, що вимагає балансу з влучністю.

F1-показник (F1-Score) є гармонійною середньою між влучністю і повнотою, забезпечуючи баланс між ними. Цей показник є корисним у ситуаціях, де важливо одночасно мінімізувати як False Positives, так і False Negatives. Високий F1-показник свідчить про те, що модель має і високу влучність, і високу повноту, що робить її ефективною у виявленні шахрайських доменів з мінімальною кількістю помилок. F1-показник є особливо важливим у випадках, коли вартість False Positives та False Negatives є високою і приблизно еквівалентною [4].

Загалом використання метрик оцінки алгоритмів дозволяє ефективно калібрувати моделі для ідентифікації шахрайських доменів, забезпечуючи оптимальний баланс між точністю виявлення загроз і мінімізацією помилкових спрацювань.

Застосування метрик на практиці

Застосування описаних метрик оцінки алгоритмів є вирішальним для оптимізації та валідації моделей виявлення шахрайських доменів. Ці метрики використовуються для кількісної оцінки результатів класифікації, дозволяючи розробникам точно зрозуміти, як модель взаємодіє з даними та які недоліки вона має.

Технічно, оптимізація моделей з інтеграцією даних метрик здійснюється через процеси налаштування гіперпараметрів, інжинірингу та вибору моделі. Наприклад, модифікація порогових значень для класифікатора може змінити баланс між FP та FN, що вплине на повноту та влучність. Адаптація цих порогів дозволяє розробникам підлаштувати модель під конкретні вимоги задачі, зменшуючи вплив найбільш критичного типу помилок.

Для гібридних моделей, що використовують CNN, BiLSTM та механізми уваги для аналізу доменних імен, технічний аналіз цих метрик дозволяє оцінити, як кожен компонент впливає на загальну ефективність системи [5]. Це допомагає визначити, чи покращує комбінація різних архітектур загальну здатність моделі виявляти шахрайські домени, а також ідентифікувати можливості для покращень.

Застосування автоматизованих інструментів для моніторингу вказаних метрик в реальному часі є необхідним для підтримки високої продуктивності систем виявлення шахрайських доменів. Такий підхід забезпечує безперервне вдосконалення моделі, адаптацію до нових шахрайських схем та мінімізацію негативного впливу на легітимний трафік.

Висновок

Проаналізовані метрики оцінки алгоритмів є фундаментальними для розробки та оптимізації моделей з виявлення шахрайських доменів. Використання даних метрик на практиці сприяє балансу між зниженням помилкових спрацьовувань та забезпеченням комплексного захисту від загроз.

Література

[1] Logeswari G., Bose S., Anitha, T. An Intrusion Detection System for SDN Using Machine Learning. Intelligent Automation & Soft Computing, 2023. DOI: https://doi.org/10.32604/iasc.2023.026769.

[2] Naidu G., Zuva T., Sibanda E. M. A Review of Evaluation Metrics in Machine Learning Algorithms. In Computer Science On-line Conference 2023. DOI: https://doi.org/10.1007/978-3-031-35314-7_2.

[3] Bahaghighat M., Ghasemi M., Ozen, F. A high-accuracy phishing website detection method based on machine learning. Journal of Information Security and Applications, 2023. DOI:10.1016/j.jisa.2023.103553.

[4] Kapan S., Gunal E. S. Improved Phishing Attack Detection with Machine Learning: A Comprehensive Evaluation of Classifiers and Features. Applied Sciences, 2024. DOI: https://doi.org/10.3390/app132413269.

[5] Zhou J., Gandomi A. H., Chen F., Holzinger A. Evaluating the quality of machine learning explanations: A survey on methods and metrics. Electronics, 10(5), 2021. DOI: https://doi.org/10.3390/electronics10050593.

[6] Al-Juboori S. A. M., Hazzaa F., Jabbar Z. S., Salih S., Gheni H. M. Man-in-the-middle and denial of service attacks detection using machine learning algorithms. Bulletin of Electrical Engineering and Informatics, 2023. DOI: https://doi.org/10.11591/eei.v12i1.4555.