НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ВИЯВЛЕННЯ ЗЛОВМИСНОЇ АКТИВНОСТІ В КОРПОРАТИВНИХ ЛОКАЛЬНИХ МЕРЕЖАХ ЗА ДОПОМОГОЮ АНАЛІЗУ DNS-ЗАПИТІВ»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології»,
спеціальність 125 «Кібербезпека»
Гуренко Марк Анатолійович

Керівник: к.т.н., доцент
Бойко Віктор Дмитрович

Рецензент: к.т.н., доцент
Ахмаметьєва Ганна Валеріївна

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Гуренку Марку Анатолійовичу

1. Тема роботи: «Виявлення зловмисної активності в корпоративних локальних мережах за допомогою аналізу DNS-запитів»
Керівник роботи: к.т.н, доцент Віктор Дмитрович Бойко
затверджені наказом НУ ОЮА від «__» _____ 20___ року № _____
2. Строк подання здобувачем роботи «__» _____ 20___ рік
3. Дата видачі завдання«__» _____ 20___ рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Виявлення зловмисної активності в корпоративних локальних мережах за допомогою аналізу DNS-запитів» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 13 рисунків, 2 таблиці, 1 додаток, 38 літературних джерел за переліком посилань. Робота виконана на 46 сторінках загального тексту і 33 сторінках основного тексту.

У процесі роботи було проаналізовано методи та підходи для виявлення трафіку від алгоритмів генерації доменів (DGA), вивчено особливості та принципи роботи DGA. Для реалізації було вибрано засоби та технології, сформовано початковий вибірку даних для тренування та описано його характеристики. Обрано моделі для навчання, зокрема архітектуру Long Short-Term Memory (LSTM) та техніку ансамблевого навчання на основі логістичної регресії, які з високою прогнозтичною спроможністю виявляють домени, створені алгоритмом DGA. Реалізація проводилась з використанням мови програмування Python, веб-середовища Jupyter Notebook та бібліотек TensorFlow, Keras та SciKit-Learn.

Результати дослідження показують високу ефективність оцінювання домених імен у DNS-запитах на допомогою алгоритмів глибокого навчання, так вихідна точність досягає понад 99,2%.

Результати роботи можуть бути використані при проектуванні превентивних мір як частини систем виявлення вторгнень (IDS), а також можуть слугувати для академічних досліджень у сфері імплементації алгоритмів глибокого навчання в технології аналізу мережевого трафіку глобальної мережі Інтернет.

ABSTRACT

Qualification work on the topic "Detection of malicious activity in corporate local networks by analysing DNS queries" for the first (bachelor's) level of higher education in the speciality 125 Cybersecurity, educational programme: Cybersecurity, contains 13 figures, 2 tables, 1 appendix, 38 references. The work is performed on 46 pages of general text and 33 pages of the main text.

In the course of the work, there were analysed methods and approaches for detecting traffic from domain generation algorithms (DGA), and the features and principles of DGA operation were studied. Tools and technologies were selected for implementation, an initial data set for training was formed, and its characteristics were described. The models for training were selected, in particular, the Long Short-Term Memory (LSTM) architecture and the ensemble learning technique based on logistic regression, which detect domains created by the DGA algorithm with high predictive capability. The implementation was carried out using the Python programming language, the Jupyter Notebook web environment, and the TensorFlow, Keras, and SciKit-Learn libraries.

The results of the study show the high efficiency of domain name evaluation in DNS queries using deep learning algorithms, with the output accuracy reaching over 99,2%.

The results of the work can be used in the design of preventive measures as part of intrusion detection systems (IDS), and can also serve for academic research in the field of implementing deep learning algorithms in the technology of analysing network traffic on the Internet.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

C2, C&C – Command and Control (управління та керування)

DNS – Domain Name System (система доменних імен)

DGA – Domain Generation Algorithm (алгоритм генерації доменів)

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

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

TLD – Top-Level Domain (верхній рівень домену)

SLD – Second-Level Domain (домен другого рівня)

ISP – Internet Service Provider (постачальник інтернет-послуг)

ML – Machine Learning (машинне навчання)

DL – Deep Learning (глибоке навчання)

GRU – Gated Recurrent Unit (вентильний рекурентний вузол)

PNN – Probabilistic Neural Network (імовірнісна нейронна мережа)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

KNN – k-Nearest Neighbors (k-найближчих сусідів)

TP – True Positive (істинно позитивні випадки)

TN – True Negative (істинно негативні випадки)

FP – False Positive (хибно позитивні випадки)

FN – False Negative (хибно негативні випадки)

TPR – True Positive Rate (частота істинно позитивних випадків)

FPR – False Positive Rate (частота хибно позитивних випадків)

ROC – Receiver Operating Characteristic (робоча характеристика приймача)

AUC – Area Under the Curve (площа під кривою)

ВСТУП

У сучасному світі мережевої безпеки виявлення зловмисної давно є викликом будь-якій мережевій системі. Зокрема, шкідливе програмне забезпечення часто взаємодіє з серверами управління (Command and Control Server, C2 server) через доменні імена. Аналіз запитів до системи доменних імен (DNS) є практикуючим методом виявлення зловмисної активності. Ця робота зосереджена на створенні системи машинного навчання для виявлення та усунення цієї загрози.

Важливість цього дослідження підкреслюється зростаючою складністю і частотою атак шкідливих програм, що використовують DGA. Традиційні методи виявлення на основі сигнатур часто виявляються неефективними проти таких загроз, що підкреслює необхідність більш просунутих підходів. Використовуючи методи машинного навчання, рекурентні нейронні мережі з довгою та короткою пам'яттю (LSTM-RNN), а також у поєднанні з технікою ансамблевого навчання, це дослідження має на меті покращити можливості розпізнавання злочинних доменів, підвищуючи таким чином безпеку конкретної мережі.

Велика кількість робіт, у цій області зосереджені на використанні методів машинного навчання для класифікації доменів. Це дослідження доповнює цю низку робіт, узагальнюючи та спрощуючи підхід для широкого використання.

Мета роботи – розробка системи виявлення зловмисних дій в корпоративних локальних мережах на основі глибокого навчання для шляхом вивчення DNS-запитів, зокрема доменних імен у запитах.

Завдання роботи включають:

- Огляд досліджень і методів, що стосуються аналізу DNS-запитів.
- Складання чіткого формулювання проблеми та дослідницьких запитів, які спрямовуватимуть дослідження.
- Побудова моделі LSTM RNN для ідентифікації потенційно шкідливих доменів.
- Побудова ансамблевої моделі, використовуючи результати прогнозування LSTM моделі.

- Оцінка ефективності запропонованої моделі в порівнянні з існуючими рішеннями, щоб продемонструвати її дієвість.

Об'єктом дослідження є мережевий трафік корпоративних локальних мереж.

Предметом дослідження є зловмисна активність у межах DNS-запитів.

Результати цього дослідження мають важливе практичне значення для фахівців та організацій, що займаються мережевою безпекою, зокрема у корпоративних локальних мережах. Успішна реалізація моделі LSTM-RNN, поєднаної з технікою ансамблевого навчання, для виявлення шкідливих доменів може значно підвищити здатність запобігати зараженню шкідливим програмним забезпеченням та пом'якшити потенційні порушення безпеки завдяки відносній простоті реалізації та низьких вимог до обчислювальних ресурсів.

Основні положення кваліфікаційної роботи були викладені на науковій конференціях [1, 2]. У доповідях, зокрема, обговорювалося використання рекурентних нейронних мереж з довгою короткочасною пам'яттю (LSTM) в реальному часі без необхідності контекстної інформації або створених вручну ознак для виявлення доменів, створених алгоритмом генерації доменів (DGA), які є загрозою для кібербезпеки. LSTM виявилися ефективними у багатьох завданнях машинного навчання, особливо в обробці послідовних даних, і їх можна успішно використовувати для виявлення складних закономірностей у DNS-запитах, що вказують на DGA-домени.

1 ЗАГАЛЬНІ ВІДОМОСТІ ТА МЕТОДИ ВИЯВЛЕННЯ ШКІДЛИВИХ DNS-ЗАПИТІВ

1.1 Початкова інформація

Ботнет – це мережа скомпрометованих комп'ютерів (ботів), з'єднаних і керованих віддалено за допомогою C&C-серверу. Ці машини, які можуть бути комп'ютерами, мобільними пристроями або загальнодоступними системами, розподілені географічно і заражені шкідливим кодом (шкідливим програмним забезпеченням). Це шкідливе програмне забезпечення змушує інфіковані пристрої підкорятися командам бот-майстра через сервер управління (C&C-сервер) [3]. Ботнети розвиваються разом з ботом-майстром і відрізняються за розміром й структурою, але проходять однакові етапи життєвого циклу [4]. Заражені хости, які також називають «зомбі», налаштовані на виконання шкідливих завдань, таких як крадіжка інформації або запуск атак на інші пристрої чи мережі.

Зомбі-мережа та її контролери можуть складатися з систем з різних мереж і локацій. Сила бот-мереж полягає в розподіленій обчислювальній потужності, доступній серверу управління та контролю (C&C-сервер). Цей сервер має вирішальне значення для бот-мереж, і хакери постійно працюють над тим, щоб приховати його ідентичність і місцезнаходження, роблячи зв'язок між хостами ботів і C&C-сервером не відстежуваним. Система доменних імен (DNS) відіграє життєво важливу роль в архітектурі бот-мережі, маскуючи ідентифікацію C&C-сервера за доменним ім'ям, що дозволяє бот-майстрам легко і швидко змінювати місцезнаходження сервера [5].

Моніторинг DNS-трафіку може допомогти виявити зловмисні доменні імена, оскільки зловмисники постійно вдосконалюють тактику уникнення виявлення, наприклад, динамічно змінюючи IP-адреси C&C-серверів. Поширеним методом приховування ідентичності C&C-сервера є алгоритм генерації доменів (Domain Generation Algorithm, DGA). DGA використовується різними ботнетами, генеруючи численні доменні імена, з яких лише деякі є активними та зареєстрованими [6]. Бот-хости регулярно використовують DGA, щоб створити список потенційних доменів для зловмисників, який варіюється на основі

випадкової вибірки, наприклад, будь-якими випадковими символами або поточної дати і часу.

Боти запрограмовані на зв'язок зі своїм C&C-сервером, що змушує їх надсилати DNS-запити для визначення (resolving) доменних імен, таких як ті, що генеруються DGA, доки не знайдуть живий домен. Ботнети на основі DGA мають високий рівень виживання, оскільки кожен бот може швидко переключитися на IP-адресу іншого живого C&C-сервера, якщо один з них заблокований або підозрюється в тому, що за ним ведеться спостереження. Це дозволяє бот-майстру динамічно змінювати свої C&C-сервера, не впливаючи на своїх ботів. Така поведінка є дуже складною для виявлення, оскільки традиційними методами важко ідентифікувати та нейтралізувати таку інфраструктуру. Незважаючи на високу ймовірність виживання завдяки використанню декількох доменних імен, ботнети на основі DGA все одно залишає сліди, які при ретельному вивченні можуть надати інформацію про їхню присутність. Наприклад, хости, заражені одним і тим же програмним забезпеченням для ботів на основі DGA, генеруватимуть споріднений набір доменних імен і, ймовірно, запитуватимуть одні й ті ж домени. У більшості випадків такий підхід призводить до численних запитів на неіснуючі домени (NXDomains) без справжніх IP-адрес.

1.2 Алгоритм генерації доменів (DGA)

Ботнети потребують прихованого зв'язку, щоб уникнути перехоплення захисним програмним забезпеченням. Вони часто використовують Domain Fluxing – комунікаційну технологію, що використовує систему доменних імен (DNS) для динамічного перемикання C&C-серверів [8]. Наприклад, шкідлива програма може заздалегідь налаштувати кілька доменних імен, а бот-майстер активує одне з них випадковим чином, щоб вказати на C&C-сервер, коли це необхідно.

Стратегія захисту від цього – «sinkhole», коли захисники легально перенаправляють всі шкідливі доменні імена на певний хост замість початкового C&C-сервера [9]. Цей хост не має функціональності C&C-сервера, подібно до чорної діри, яка блокує ботнет. Щоб підтримувати зв'язок між ботами та C&C-

серверами, зловмисники розробили алгоритм генерації доменів (DGA). DGA швидко генерує безліч доменних імен, з яких боти вибирають декілька для зв'язку з C&C-сервером.

Шкідливе програмне забезпечення отримує доступ до джерела випадкових даних і використовує його в алгоритмі для автоматичної генерації сотень або тисяч доменів (DGA). Потім шкідливе ПЗ намагається визначити (resolve) кожен з цих доменів за допомогою свого локального DNS-сервера (який перетворює доменні імена в IP-адреси, що є важливою частиною Інтернету). Бот-майстер реєструє кілька таких автоматично згенерованих доменів. Для зареєстрованих доменів шкідливе ПЗ отримує дійсну IP-адресу і може зв'язуватися з центром контролю та управління. Для незареєстрованих доменів шкідливе ПЗ отримує повідомлення про те, що вони не можуть бути вирішені (resolved), і ігнорує їх (NXDomains). DGA ускладнюють внесення доменів до чорного списку, оскільки зміна початкової випадкової вибірки (при використанні того самого алгоритму) може згенерувати абсолютно різні домени. Цей метод використовувався такими відомими шкідливими програмами, як Conficker, Stuxnet (націлений на ядерні об'єкти Ірану) і Flame [7]. Виявлення доменних імен, згенерованих алгоритмом DGA, стало ключовим напрямком у сфері інформаційної безпеки, що призвело до зростання інтересу до використання методів машинного навчання для виявлення доменів DGA.

DGA створюють вкрай асиметричну ситуацію між зловмисником і захисником. Зловмиснику достатньо вибрати одне або кілька згенерованих доменних імен, щоб здійснити атаку, тоді як захисник повинен заблокувати всі доменні імена. Наприклад, «Conficker C» може генерувати 50 000 доменних імен щодня і розподіляти їх у понад 113 доменах верхнього рівня (TLD), що робить пасивні методи захисту неефективними [10].

Кілька існуючих методів виявлення доменних імен, згенерованих DGA, показали багатообіцяючі результати, а деякі системи були перевірені в реальних мережах [11, 12]. Деякі підходи аналізують низькорівневий DNS-трафік або журнали, що вимагає задіяння дата-центру інтернет-провайдера (ISP). Ці методи

використовують видобуті вручну ознаки для кластеризації або класифікації. Низькорівневий DNS-трафік надає детальну інформацію про DNS-запити та відповіді, що дає змогу ретельніше шукати та класифікувати зловмисні доменні імена. Однак це обмежує їх застосування сценаріями, де для моніторингу трафіку DNS-серверів потрібен дозвіл, наприклад, у великих інтернет-провайдерів або мережевих центрах компаній. Доступ до DNS-серверів також може викликати занепокоєння щодо конфіденційності. Щоб подолати ці обмеження, деякі дослідники зосередилися на використанні характеристик самого доменного імені, використовуючи лише доменні імена як вхідні дані для методів виявлення [13, 14, 15].

1.3 Використання методів ML/DL для виявлення DGA

Найпростішим методом виявлення DGA-доменів є використання чорного списку. Однак цей підхід має обмежену ефективність та повноту через постійну генерацію нових доменів [16]. Іншим поширеним методом є зворотна інженерія, яка полягає в аналізі зразків шкідливих доменів для виявлення DGA і подальшого блокування згенерованих ним доменів. Однак зворотна інженерія займає багато часу і вимагає значних знань, що робить її повільним і непрактичним методом порівняно з іншими підходами.

1.3.1 Методи з використанням статистичних особливостей

Для подолання обмежень попередніх підходів чорних списків та зворотного інжинірингу в кількох дослідженнях вивчалося використання статистичних ознак у доменних іменах використовували біграми та розподіл символів/чисел у доменних іменах, щоб розрізнити кластери доменів, що належать та не належать до DGA [25]. Вони порівняли три загальновживані методи математичної статистики для кластеризації доменів: Розходження Коефіцієнт Жаккара, Кульбака — Лейблера та «Edit distance». Інше дослідження [26] зосередилося на виявленні DGA-доменів шляхом аналізу морфемних і лексичних особливостей доменних імен. Вони представили евристичний метод для ідентифікації морфемних токенів у доменних іменах і розглянули особливості цих токенів на додаток до існуючих

лексичних ознак. Однак ці ручні методи вилучення ознак займають багато часу і можуть бути використані зловмисниками.

У якості заходу захисту застосовуються підходи на основі машинного навчання, які враховують лексичні особливості. Один із підходів передбачає використання статистичних характеристик, таких як ентропія та біграми символів, а також довжина домену [17]. Інший підхід полягає в застосуванні лексичних шаблонів, витягнутих з не-DGA доменів, до алгоритмів машинного навчання, таких як Random Forest (Випадкові ліси) [18]. Однак вилучення цих особливостей також вимагає значного часу та досвіду, і зловмисники можуть використовувати їх, щоб уникнути виявлення.

Нещодавні дослідження вивчали використання додаткової інформації для покращення виявлення. Наприклад, інформація WHOIS, включаючи контактні дані власника домену, статус доступності та компанію, що зареєструвала домен, використовувалася як додаткові дані. Однак ці методи не дозволяють виявляти DGA-домени в режимі реального часу. Крім того, отримання інформації WHOIS стало проблематичним через впровадження Загального регламенту про захист даних (GDPR) [19].

Нещодавні досягнення в галузі глибокого навчання призвели до розробки декількох методів, спрямованих на підвищення ефективності виявлення DGA доменів, усуваючи недоліки в існуючих підходах [20, 21, 22, 23, 24]. Ці методи, засновані на глибокому навчанні, пропонують покращену продуктивність без необхідності додаткової інформації, усуваючи потребу в трудомісткому вилученні ознак. Крім того, їх непрозорий характер ускладнює зловмисникам завдання зворотного інжинірингу або обходу цих методів. Вони також здатні виявляти в реальному часі, використовуючи лише доменні імена, не вимагаючи додаткових даних.

Серед різних моделей глибокого навчання рекурентні нейронні мережі (RNN) виявилися особливо ефективними завдяки своїй здатності обробляти дані послідовностей з високою точністю. Оскільки доменні імена можна розглядати як послідовності символів, моделі RNN перевершують інші моделі нейронних мереж.

Зокрема, мережі з довгою короткочасною пам'яттю (LSTM) на основі RNN добре підходять для вирішення проблеми дисбалансу класів, яка часто зустрічається при виявленні DGA доменів [20].

1.3.2 Методи, засновані на N-грамах

У ряді досліджень також використовувалися ознаки, отримані на основі статистики n-грам. На додаток до n-грам, Selvi та ін. [27] використовували замасковані n-грами, в яких кожен символ замінюється символом, що представляє його тип (приголосний, голосний, цифра або інший). Alaeiyan та ін. [28] досліджували розподіл n-пропускних графів, де n центральних символів опускаються з послідовності сусідніх символів (для n=1,2).

Yang та ін. [29] і Patsakis та ін. [30] зосередилися на DGA на основі списків слів, використовуючи численні ознаки для розрізнення зловмисних і нешкідливих доменних імен на основі списків слів. Yang та ін. використовували 24 ознаки, засновані на частоті слів, частоті частин мови, кореляції між словами та міждоменній кореляції. Patsakis та ін. використовували 32 ознаки, які враховували алфавітно-цифрові послідовності, статистичні та лексичні характеристики, а також ентропію.

1.3.3 Методи глибокого навчання

LSTM-мережі були запропоновані Woodbridge та ін. [23] для прогнозування методів формування доменів. У їхньому дослідженні представлено класифікатор DGA, який прогнозує DGA без необхідності вручну генерувати ознаки або контекстні знання, використовуючи LSTM-мережі. Точна багатокласова категоризація є ще однією особливістю методу, яка дозволяє віднести домен, створений DGA, до певного сімейства DGA.

Woodbridge та ін. [23] класифікували домени DGA за допомогою порівняно простої мережі довготривалої короткочасної пам'яті (LSTM), що дозволило значно покращити результати порівняно з найсучаснішими дослідженнями. Хоча цей метод був визнаний ефективним, він виявився незадовільним у певних випадках, особливо коли мав справу зі складними родинами DGA, що імітують англійські слова.

Методику поведінкової ідентифікації DGA в режимі реального часу на основі машинного навчання, що базується на моніторингу мережі, представила компанія Bisio [33]. Метод було протестовано на спеціальній мережі, до якої було додано кілька відомих DGA, а також на локальних мережах організації.

Підхід для ідентифікації DGA-доменів за допомогою RNN і побічної інформації запропонував Райан Р. Кертін (Ryan R. Curtin) [32]. Їхні результати показали, що модель перевершила попередні методи у виявленні доменів, згенерованих складними сім'ями DGA. Метод може бути використаний як окремий детектор DGA-доменів або як компонент більш широкої системи виявлення шкідливого програмного забезпечення. Він продемонстрував хороші результати, особливо для сімейств DGA, які нагадують англійські фрази.

Щоб скоротити час, необхідний для оцінки шкідливого програмного забезпечення, Shibahara та ін. [31] представили дещо модифіковану методику, яка використовує рекурентні нейронні мережі (RNN) для оцінки відмінностей у мережевому трафіку. Цей метод, хоча і не є специфічним для DGA, намагається охопити широкий спектр видів шкідливого програмного забезпечення, включаючи DGA-загрози, шляхом аналізу їхніх шаблонів комунікації. Тим не менш, він потребує додаткової інформації про час виконання, такої собі "пісочниці" шкідливого програмного забезпечення, якої не потребують багато інших методів. За словами авторів, цей підхід утримує рівень виявлення шкідливих URL-адрес на рівні 97,9%, скорочуючи час аналізу з приблизно 15 хвилин до 67%.

Із урахуванням вище сказаного, традиційні методи, такі як чорні списки та зворотна інженерія вичерпали себе та обмежені у роботі в режимі реального часу. Водночас різні методи машинного і глибокого навчання, в тому числі засновані на статистичних ознаках, n-грамах і рекурентних нейронних мережах (RNN), показали багатообіцяючі результати у виявленні доменів DGA.

2 ОГЛЯД ПРОГРАМНО-ТЕХНІЧНОЇ СКЛАДОВОЇ ТА ТЕОРЕТИЧНІ ОСНОВИ

2.1 Вибір інструментів розробки

Python – це основна мова програмування, яка використовується для вашого проекту. Її популярність у сфері машинного та глибокого навчання робить її ідеальним вибором для цього завдання. Простота, гнучкість та численні бібліотеки Python роблять його чудовим вибором для аналізу, обробки та візуалізації даних.

Pandas та Numpy – це дві основні бібліотеки Python, які використовуються для підготовки набору даних до навчання. Pandas надає структури даних та функції для ефективної обробки структурованих даних, включаючи табличні дані, такі як електронні таблиці та таблиці SQL. Numpy – бібліотека для мови Python, що додає підтримку великих багатовимірних масивів і матриць, а також велику колекцію високорівневих математичних функцій для роботи з цими масивами.

TensorFlow, Keras та Sklearn – використовуються для безпосереднього навчання моделі глибокого навчання та оцінки її продуктивності. TensorFlow – це програмна бібліотека з відкритим вихідним кодом для чисельних обчислень, яка особливо добре підходить для великомасштабних завдань машинного навчання (ML) та глибокого навчання (DL). Keras – це високорівневий API нейронних мереж, написаний на Python і здатний працювати поверх TensorFlow. Sklearn – бібліотека машинного навчання для Python, яка надає широкий спектр алгоритмів для класифікації, регресії, кластеризації та інших завдань.

Pickle – це модуль Python, який використовується для серіалізації та десеріалізації об'єктів Python. Він дозволяє зберегти навчену модель для подальшого використання без необхідності перенавчати модель з нуля. Це особливо корисно, коли потрібно розгорнути модель у виробничому середовищі або коли потрібно повторно використовувати модель для майбутніх завдань.

tldextract – це бібліотека Python, яка використовується для зручного розбиття доменного імені на основні компоненти, такі як домен верхнього рівня (TLD), домен другого рівня (SLD) та субдомени. Ця бібліотека корисна для попередньої обробки доменних імен перед подачею їх модель глибокого навчання.

Jupyter Notebook – це інтерактивне обчислювальне веб-середовище, яке дозволяє створювати та обмінюватися документами, що містять живий код, рівняння, візуалізації та описовий текст. Це чудовий інструмент для дослідження даних, створення прототипів та візуалізації, що робить його важливим компонентом вашого технологічного стеку.

Обраний стек технологій оптимально підходить для проекту, оскільки він широко використовується та підтримується у спільнотах машинного навчання та глибокого навчання. Таке широке розповсюдження забезпечує довгострокову життєздатність та продуктивність завдяки постійному потоку вдосконалень та виправлень від активної спільноти розробників. Постійна підтримка з боку творців гарантує сумісність на різних платформах та версіях, що має вирішальне значення для безперешкодної інтеграції з іншими інструментами. Обширна документація та ресурси ще більше полегшують процес навчання, сприяючи співпраці та впровадженню інноваційних рішень.

2.2 Алгоритм глибокого навчання для виявлення зловмисних DGA-доменів DNS-запитів

У цій роботі буде використано архітектуру нейронних мереж LSTM, яка має доведену ефективність у порівняні з GRU, PNN, CNN, KNN архітектурами згідно роботам [23, 24, 35, 36]. Архітектура LSTM була обрана тому, що вона краще розпізнає доменні імена на рівні символів завдяки своїй пам'яті та здатності відновлювати попередні стани, чого немає у згорткових нейронних мереж (CNN) [37].

«…мережі LSTM мають механізм вентилювання, який дозволяє їм вибірково запам'ятовувати або забувати інформацію, що дозволяє їм ефективно фіксувати довгострокові залежності в послідовних даних. Вони добре підходять для задач, де розуміння контексту довгих послідовностей має вирішальне значення. LSTM можуть обробляти вхідні послідовності довільної довжини, не страждаючи від проблеми зникаючого градієнта, яка характерна для традиційних RNN. їхня здатність зберігати релевантну інформацію протягом тривалих періодів часу

робить їх потужним інструментом для моделювання складних часових зв'язків.», як зазначено у статті [36].

LSTM, або мережі з довгою короткочасною пам'яттю, є унікальним класом RNN, які мають здатність навчатися довготривалим залежностям. Після того, як їх представили Хохрайтер і Шмідхубер (Hochreiter & Schmidhuber, 1997) [35]. Фелікс Герс, Фред Каммінс, Сантьяго Фернандес, Джастін Байєр, Даан Віерстра, Юліан Тогеліус, Фаустіно Гомес, Маттео Гальйоло та Алекс Грейвс вдосконалили та популяризували їх. Наразі їх часто використовують, оскільки вони неймовірно ефективно працюють у широкому діапазоні ситуацій.

LSTM цілеспрямовано створені для того, щоб обійти проблему довгострокової залежності. Запам'ятовування інформації на довгі періоди часу - це практично їхня поведінка за замовчуванням, а не те, чого вони намагаються навчитися [34].

Кожна RNN складається з ланцюжку нейромережевих модулів, які повторюються. Цей повторюваний модуль у традиційних RNN матиме структуру, наприклад, один тангенціальний шар (tanh layer) [34]. RNN можна виразити рівняннями (1) і (2):

$$h_t = \sigma\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right), \qquad (1)$$

$$y_t = W_{hy}h_t + b_y. \qquad (2)$$

У цих рівняннях h_t позначає прихований стан у момент часу t, де x та y – вхід та вихід відповідно. У рівнянні (1) h_t обчислюється як вхід x_t поточного стану і попереднього прихованого стану h_{t-1} . σ – функція активації, а W_{xh}, W_{hh} і b_h – параметри, що підлягають вивченню. У рівнянні (2) прогнозоване значення y_t обчислюється за допомогою h_t, а W_{hy} і b_y є параметрами навчання [38].

LSTM також мають подібну ланцюгову структуру, але модуль, що повторюється, має іншу структуру. Тут чотири шари нейронної мережі замість одного, і вони взаємодіють унікальним чином (рисунок 2.1).

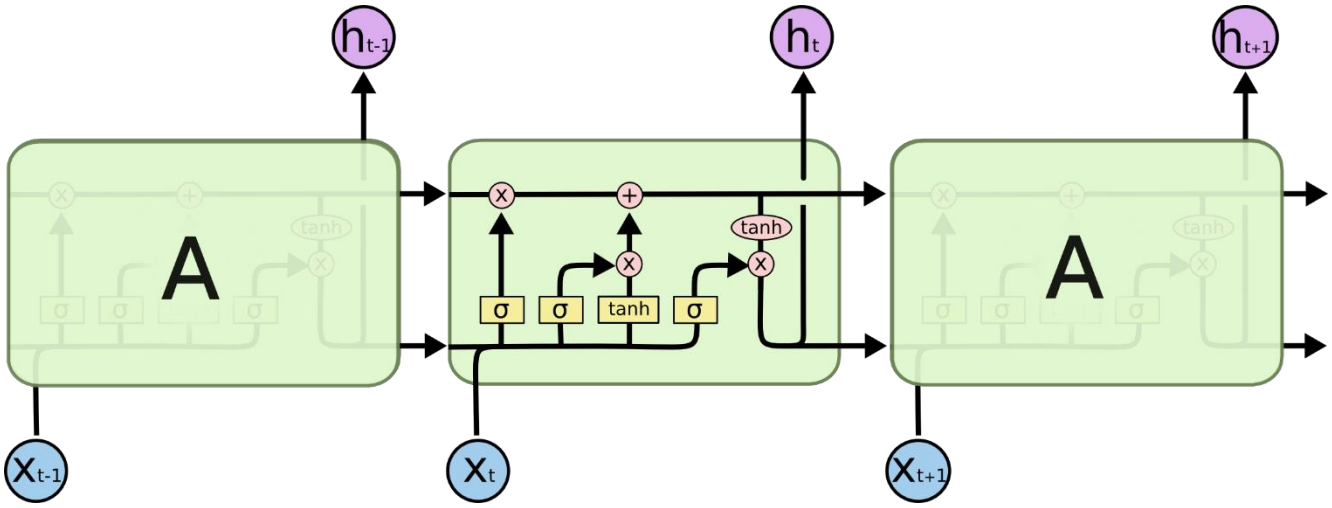


Рисунок 2.1 – Повторювальний модуль в LSTM, що складається з чотирьох взаємодіючих шарів [34].

Звичайні RNN можуть зіткнутися з проблемою зникаючого градієнта, коли малі градієнти через крихітні вагові матриці можуть значно сповільнити навчання або навіть зупинити його взагалі. І навпаки, якщо ваги занадто великі, може виникнути проблема вибухоподібного градієнта (exploding gradient problem). LSTM, різновид RNN, вирішує ці проблеми. На рисунку 2.2 зображено архітектуру блоку в RNN і LSTM відповідно. LSTM – це ланцюгова структура на основі RNN, яка передає попередню інформацію через комірку і визначає стан комірки за допомогою трьох воріт: забуття, входу і виходу.

LSTM можна виразити рівняннями (3, 4, 5, 6, 7):

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \qquad (3)$$

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right), \qquad (4)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \qquad (5)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right), \qquad (6)$$

$$h_t = o_t \tanh\left(c_t\right). \qquad (7)$$

Спочатку вентиль забування f вирішує, яку інформацію видалити зі стану комірки. Потім вхідний вентиль i, відповідальний за зберігання поточної інформації, визначає, яку інформацію з нових входів слід оновити в стані комірки.

Він генерує нового кандидата на включення до стану комірки через tanh layer. Потім інформація від вентилів забування та входу об'єднується для оновлення стану комірки. Нарешті, вихідний вентиль *o* вирішує, яку частину стану комірки виводити, визначаючи остаточне значення результату h_t на основі виходу стану комірки та вихідного вентиля [38].

Модель LSTM складається з комірки пам'яті з чотирма основними елементами:

1. вхідний вентиль, який контролює вплив вхідного сигналу на стан комірки пам'яті;
2. рекурентне з'єднання, що повторюється, з вагою 1.0, яке зберігає стан комірки пам'яті;
3. вентиль забування, який налаштовує рекурентне з'єднання комірки пам'яті на забування або запам'ятовування її попереднього стану;
4. вихідний вентиль, який визначає, чи впливає стан пам'яті на інші нейрони.

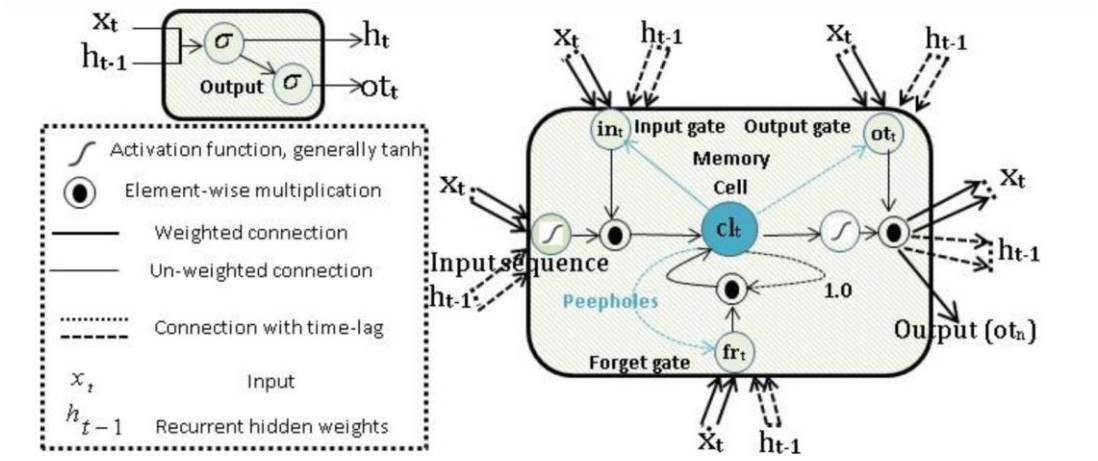


Рисунок 2.2 – Блок рекурентних нейронних мереж (ліворуч) та блок пам'яті LSTMs (праворуч) [36].

Слід зазначити, що в роботі Райана Р. Кертіна (Ryan R. Curtin) [32] приділено увагу залежності згенерованих DGA доменів від їх суфіксу домену верхнього рівня (TLD). Враховуючи цей фактор, належним буде покращити прогнозуючі спроможності LSTM моделі, додавши результати її тренування до ансамблевої

моделі. Техніка ансамблевого навчання використовується для підвищення точності моделі шляхом комбінування декількох моделей.

2.3 Метрики оцінювання

Для вимірювання ефективності запропонованої моделі у виявленні зловмисних DGA ми використовували матрицю невідповідностей, яка позначає ефективність класифікаційної моделі, а також точність (Accuracy), прецизійність (Precision), повнота (Recall) та F_1-міра (F_1-Score). Матриця невідповідностей містить наступні терміни:

- True Positive (TP, істинно позитивний): кількість доменних імен, які були правильно визначені як зловмисні.
- True Negative (TN, істинно негативний): кількість доменних імен, які були правильно визначені як безпечні.
- False Positive (FP, хибно позитивний): кількість доменних імен, які насправді є безпечними, але помилково визначені як зловмисні.
- False Negative (FN, хибно негативний): кількість доменних імен, які насправді є шкідливими, але помилково визначені як безпечні.

Accuracy (8) відображає здатність моделі правильно оцінювати вибірки в цілому, тобто здатність правильно класифікувати позитивні зразки як позитивні, а негативні - як негативні. Це міра здатності моделі правильно класифікувати та ідентифікувати шкідливі програми в цілому.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

Precision (9) – це міра здатності моделі давати якісні позитивні прогнози. Її можна інтерпретувати як ймовірність того, що позитивний прогноз, зроблений моделлю, є позитивним.

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

Recall (10), TPR (True Positive Rate) відображає здатність моделі правильно прогнозувати повний обсяг позитивних зразків, збільшуючи частку позитивних

зразків, що прогнозуються як позитивні до загальної кількості позитивних зразків. Це показник того, наскільки правильно буде ідентифіковано клас шкідливих програм.

$$Recall\ (TPR) = \frac{TP}{TP + FN} \quad (10)$$

Міра F_1 (11) є середнім гармонійним цих влучності та повноти. Відповідно, ця оцінка враховує як хибнонегативні, так і хибнопозитивні результати. Оцінка F1 зазвичай є більш цінною, ніж точність, особливо коли ми маємо нерівномірний розподіл класів.

$$F_1\ score\ = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

FPR (False Positive Rate) (12), частота хибно позитивних результатів, визначається як відношення доброякісних доменів, помилково визначених як домени DGA, до всіх доброякісних доменів:

$$FPR = \frac{FP}{FP + TN} \quad (12)$$

Компроміс між TPR та FPR вимірюється за допомогою ROC-кривої. AUC показує площу під ROC-кривою, і чим ближче AUC до 1, тим краще працює наш класифікатор. Метрики, які використовуються для оцінки нашої моделі, – це точність, TPR, FPR і AUC.

Таким чином, у цьому розділі представлено комплексний огляд програмних та апаратних компонентів, теоретичних основ та алгоритму глибокого навчання, що використовується для виявлення зловмисних DGA-доменів у DNS-запитах. Обраний стек технологій, що складається з Python, TensorFlow, Keras, Sklearn та Jupyter Notebook, є оптимальним для проекту завдяки його широкому розповсюдженню та постійній підтримці з боку спільноти розробників. Архітектура нейронної мережі LSTM була обрана за її ефективність у розпізнаванні доменних імен на рівні символів, зокрема, завдяки здатності вибірково запам'ятовувати або забувати інформацію та обробляти вхідні послідовності довільної довжини. Метод ансамблевого навчання використовується для

подальшого покращення прогностичних можливостей моделі LSTM шляхом поєднання її результатів з результатами інших моделей. Такий підхід підвищує загальну точність і надійність системи, що робить її корисним інструментом для виявлення і запобігання зловмисній діяльності в системі DNS. Моделі оцінюватимуться за допомогою різних метрик, включаючи влучність, повноту, прецизійність, оцінку F_1 та AUC, якими буде оцінена вихідна ефективність.

3 РЕАЛІЗАЦІЯ МОДЕЛІ ВИЯВЛЕННЯ ЗЛОВМИСНИХ DGA-ДОМЕНІВ

3.1 Формування вибірки даних

Дані для навчання були зібрані у форматі .csv файлів (тобто дані розділені комою). Вибірка даних складається із звичайних нешкідливих (умовно «legit») доменів (1,2 мільйона унікальних доменів) та DGA-доменів (756665 унікальних доменів).

Вибірка legit доменів була сформована з Tranco списку [39], у який входить топ один мільйон доменів (рисунок 3.1). Решта двісті тисяч було дібрано з відкритих джерел, зокрема веб-сервісу Github.

Вибірка DGA доменів була сформована з джерела консалтингової фірми Bambenek Consulting (рисунок 3.2), LTD. Bambenek Consulting – це консалтингова фірма, що займається розслідуваннями та розвідкою у сфері кібербезпеки, зосереджена на боротьбі з основними кримінальними загрозами. Відповідні дані були отримані із наданого доступу за безкоштовною ліцензією Bambenek Consulting, дивитись рисунок А.1.

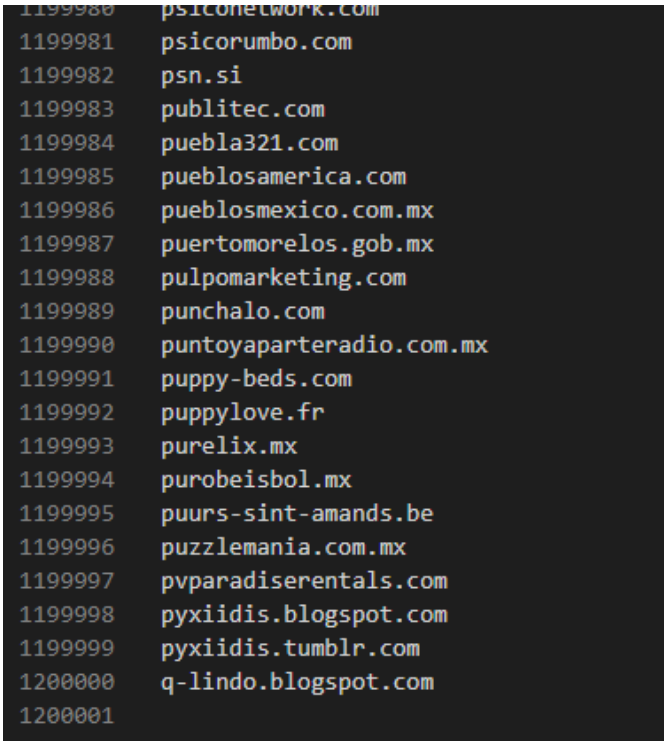


Рисунок 3.1 – Вибірка «legit» доменів.

Рисунок 3.2 – Вибірка DGA доменів.

3.2 Підготовка даних

Перейдемо до реалізації у коді програми. Першим кроком буде імпорт необхідних даних:

```
legit_1 = pd.read_csv('top-1m_0.csv', header=None, names=['domain'], usecols=[1])
legit_2 = pd.read_csv('top-1m_1.csv', header=None, names=['domain'])
legit = pd.concat([legit_1, legit_2]).drop_duplicates(subset='domain')

dga = pd.read_csv('dga-feed-high.csv', header=None, comment="#", dtype=str, names=['domain', 'description', 'date', 'url'])
dga = dga[['domain']]
```

Наступним, приведення даних у категорії: розділення рівнів доменних імен, для виокремлення первинних імен доменів (рисунок 3.3). Саме вони слугуватимуть основою тренування моделі відрізняти шкідливі домени від нешкідливих. Для цього був використаний модуль tldextract, що спрощує розподіл даних.

```
legit['primary'] = [tldextract.extract(d).domain for d in legit['domain']]
legit['tld'] = [tldextract.extract(d).suffix.split('.')[0] for d in legit['domain']]
legit.columns = ['domain', 'primary', 'tld']
```

```
    legit['label'] = 'legit'

    dga['primary'] = [tldextract.extract(d).domain for d in
dga['domain']]
    dga['tld'] = [tldextract.extract(d).suffix.split('.')[0] for d
in dga['domain']]
    dga.columns = ['domain', 'primary', 'tld']
    dga['label'] = 'dga'
```

	domain	primary	tld	label
0	tjxingyao.com	tjxingyao	com	legit
1	eikdxjdi.net	eikdxjdi	net	dga
2	ibobtoso.bazar	ibobtoso	bazar	dga
3	powercontrol.cloud	powercontrol	cloud	legit
4	teambodyproject.com	teambodyproject	com	legit

Рисунок 3.3 – Розподіл доменів на частини.

Далі отримані групи конкатинуємо:

```
    allDomains = pd.concat([legit, dga],
ignore_index=True).sample(frac=1).reset_index(drop=True)
```

3.3 Тренування LSTM моделі

Структура LSTM моделі виглядає так:

```
def build_model(max_features, maxlen):
    model = Sequential()
    model.add(Embedding(max_features, 128, input_length=maxlen))
    model.add(LSTM(128))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy',
                  optimizer='rmsprop')

    return model
```

Embedding Layer перетворює вхідні послідовності у щільні вектори фіксованого розміру (у даному випадку 128). Це допомагає моделі зрозуміти значення слів у вхідному тексті.

LSTM Layer обробляє послідовність вхідних векторів і зберігає контекстну інформацію. Він має 128 одиниць для вивчення складних патернів у послідовності.

Dropout Layer застосовується для запобігання перенавчання шляхом випадкового встановлення частки вхідних одиниць в `0` при кожному оновленні під час навчання (в даному випадку рівень відсіву становить `0.5`).

Dense Layer. Цей шар є повністю зв'язним шаром з одиницею `1`, який використовується для бінарної класифікації (виводить єдину ймовірнісну оцінку).

Activation Layer, функція активації (в даному випадку сигмоїдна) застосовується до виходу щільного шару для розподілу виходу між `0` та `1`, що представляє ймовірність належності входу до одного з класів.

Compilation. Компіляція з бінарною втратою перехресної ентропії (підходить для задач бінарної класифікації) та оптимізатором RMSprop.

Функція build_model приймає як аргументи max_features (максимальна кількість ознак у вхідних даних) та maxlen (максимальна довжина вхідних послідовностей) і повертає скомпільовану модель.

Наступним задаємо алгоритм тренування:

1. Кодування даних:

```
X = [x[1] for x in data]
labels = [x[0] for x in data]

valid_chars = {x:idx+1 for idx, x in enumerate(set(''.join(X)))}
max_features = len(valid_chars) + 1
maxlen = np.max([len(x) for x in X])

X = [[valid_chars[y] for y in x] for x in X]
X = sequence.pad_sequences(X, maxlen=maxlen)
```

Цей фрагмент видобуває доменні імена та мітки з вхідних даних, створює словник, що відображає символи в цілі числа, обчислює максимальну кількість

ознак і максимальну довжину доменного імені, кодує доменні імена в послідовності цілих чисел, використовуючи словник, і об'єднує послідовності так, щоб вони були однакової довжини.

2. Розмежування даних:

```
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
    X_train, X_holdout, y_train, y_holdout =
train_test_split(X_train, y_train, test_size=0.05, random_state=0)

    model = build_model(max_features, maxlen)
```

На виході маємо тренувальну вибірку з часткою 75%, вибірку для тестування – 20% та вибірку для валідації – 5%.

3. Тренувальний цикл:

```
for ep in range(max_epoch):
        model.fit(X_train, y_train, batch_size=batch_size, epochs=1)

        t_probs = model.predict(X_holdout)
        t_auc = sklearn.metrics.roc_auc_score(y_holdout, t_probs)

        if t_auc > best_auc:
            best_auc = t_auc
            probs = model.predict(X_test)
```

Даний цикл у кожній ітерації тренує модель і перевіряє її ROC AUC функціонал якості. Продовження ітерації тренування залежить його показнику. У разі якщо подальші ітерації не збільшують ефективність тренування, цикл завершується та повертає найкращу модель (рисунок 3.4).

```
11023/11023 [==============================] - 482s 44ms/step - loss: 0.1064
2321/2321 [==============================] - 16s 7ms/step
Epoch 0: auc = 0.997008 (best=0.000000)
11603/11603 [==============================] - 77s 7ms/step
confusion matrix on test data:
[[217819   1975]
 [  5426 146065]]
11023/11023 [==============================] - 511s 46ms/step - loss: 0.0556
2321/2321 [==============================] - 17s 7ms/step
Epoch 1: auc = 0.998236 (best=0.997008)
11603/11603 [==============================] - 81s 7ms/step
confusion matrix on test data:
[[217616   2178]
 [  3391 148100]]
11023/11023 [==============================] - 506s 46ms/step - loss: 0.0470
2321/2321 [==============================] - 16s 7ms/step
Epoch 2: auc = 0.998477 (best=0.998236)
11603/11603 [==============================] - 82s 7ms/step
confusion matrix on test data:
[[217863   1931]
 [  3079 148412]]
11023/11023 [==============================] - 514s 47ms/step - loss: 0.0425
2321/2321 [==============================] - 16s 7ms/step
Epoch 3: auc = 0.998593 (best=0.998477)
11603/11603 [==============================] - 81s 7ms/step
confusion matrix on test data:
[[216586   3208]
 [  2090 149401]]
11023/11023 [==============================] - 507s 46ms/step - loss: 0.0397
2321/2321 [==============================] - 17s 7ms/step
Epoch 4: auc = 0.998781 (best=0.998593)
11603/11603 [==============================] - 84s 7ms/step
confusion matrix on test data:
[[217878   1916]
 [  2681 148810]]
```

Рисунок 3.4 – Перші п’ять ітерацій тренування.

3.4 Тренування ансамблевої моделі

Алгоритми генерації доменів хоч і використовують псевдо-випадкові значення для формування нових доменних імен, вони все одно мають схильність до певних суфіксів [32], тобто TLD (Top Level Domain, домени верхнього рівня). Цю схильність можна виявити та використати для збільшення точності виявлення DGA-доменів. Для цього була використана класична модель логістичної регресії.

Виокремлення топ 250 найпопулярніших доменів:

```
top_tld = [tld[0] for tld in
domain_set['tld'].value_counts()[0:250].index.values]
```

Виокремлення 250 класів під кожен домен:

```
test.reset_index(drop=True, inplace=True)
domain_set.reset_index(drop=True, inplace=True)
domain_set = pd.concat([domain_set, test], axis=1)

domain_set.columns = [col[0] if isinstance(col, tuple) else col for
col in legit.columns]
```

Конкатинація із вірогідностями із LSTM моделі (рисунок 3.5):

```
lstm_probs = model_lstm.predict(X)
lstm_probs = pd.DataFrame(lstm_probs)
lstm_probs.columns = ['lstm_probs']
lstm_probs.reset_index(drop=True, inplace=True)
domain_set.reset_index(drop=True, inplace=True)
domain_set = pd.concat([domain_set, lstm_probs], axis=1)
```

	label	0	1	2	3	4	5	6	7	8	...	241	242	243	244	245	246	247	248	249	lstm_probs
0	legit	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0.000008
1	dga	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0.606417
2	dga	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1.000000
3	legit	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0.000020
4	legit	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0.000610

Рисунок 3.5 – Набір даних перед наступним тренуванням.

Розподіл тренувальної й тестової вибірок:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state = 0)
```

Тренування:

```
clf = LogisticRegression(random_state=0, multi_class='ovr', n_jobs =
-1).fit(X, y)
```

3.5 Результати тренування

Основною моделлю для тренування було обрано LSTM модель. Для покращення точності прогнозування, враховуючи схильності DGA до певних TLD суфіксів, імплементовано ансамблеву модель, побудовану на архітектурі логістичної регресії.

За результатами тренування першої моделі отримано відповіді (рисунок 3.6) та вираховувано метрики оцінювання прогнозування (таблиця 3.1). Частка хибно позитивних та хибно негативних відповідей становить ≈ 1% та ≈ 1,3% відповідно.

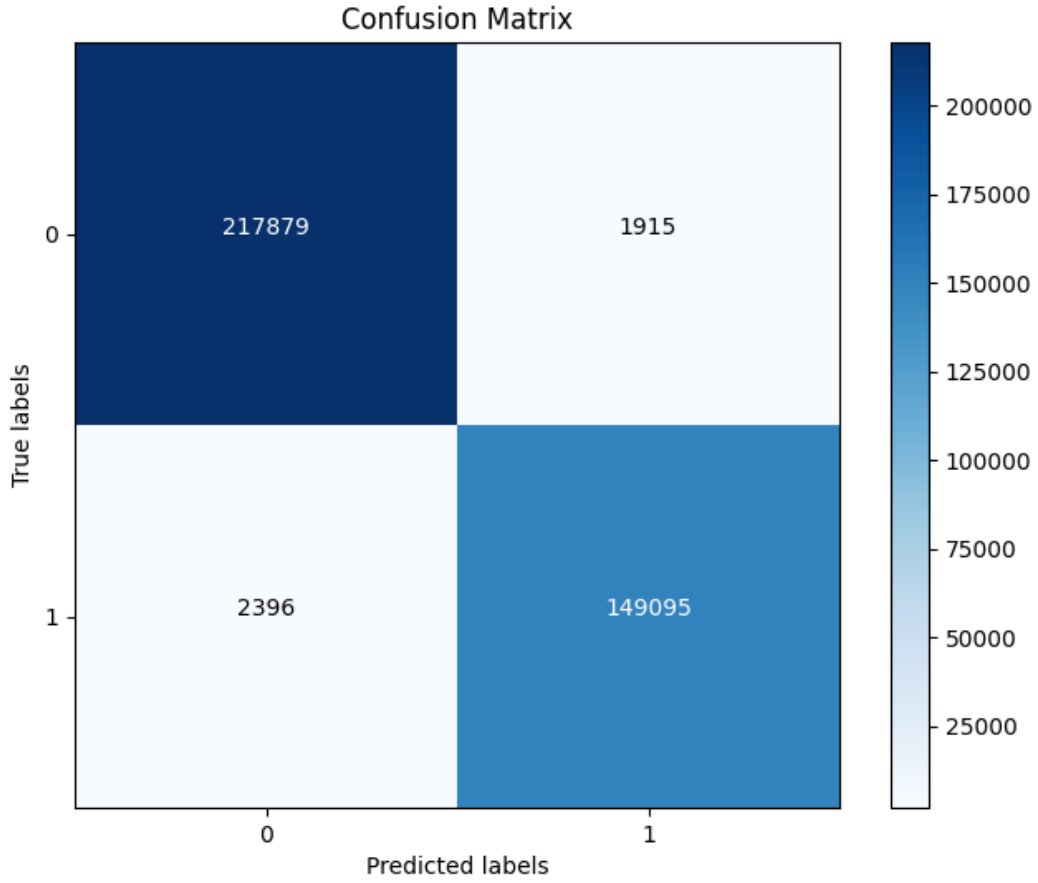


Рисунок 3.6 – Матриця невідповідностей LSTM моделі.

Таблиця 3.1 – Основні метрики LSTM моделі.

Accuracy	0,988388973
Precision	0,989122688
Recall	0,991287296
F_1 score	0,990203809
False Positive Rate	0,015816121
ROC AUC	0,998943

За результатами тренування другої моделі отримано відповіді (рисунок 3.7) та вираховувано метрики оцінювання прогнозування (таблиця 3.2). Частка хибно позитивних та хибно негативних відповідей становить $\approx 0{,}76\%$ та $\approx 0{,}78\%$ відповідно.

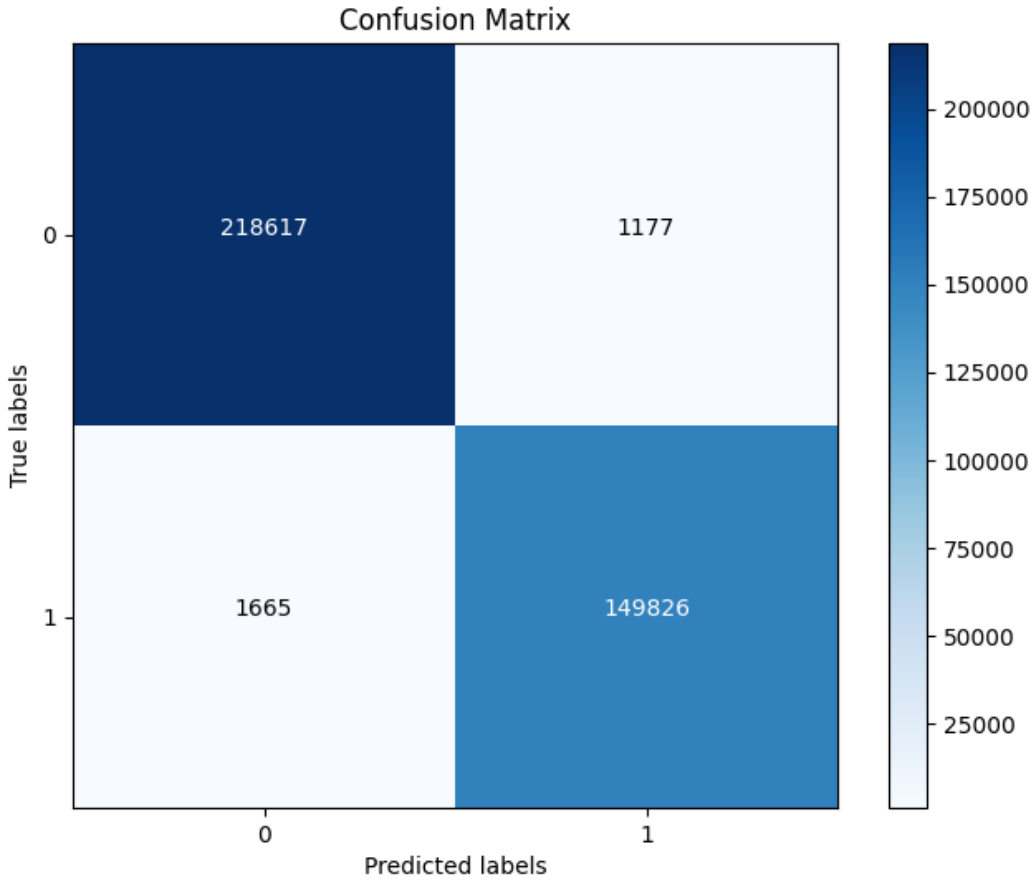


Рисунок 3.7 – Матриця невідповідностей ансамблевої моделі.

Таблиця 3.2 – Основні метрики ансамблевої моделі.

Accuracy	0,992345503
Precision	0,992441507
Recall	0,994644986
F_1 score	0,993542025
False Positive Rate	0,010990752
ROC AUC	0,998943

З вище наведених результатів тренування, випливає, що незважаючи на високу точність моделі з довгою короткочасною пам’яттю, врахування схильностей DGA доменів до певних суфіксів доменів верхнього рівня покращило передбачальну ефективність системи виявлення зловмисних DGA доменів. Наочно покращення зображено на рисунку 3.8. На графіку спостерігається покращення показників метрик оцінювання у порівнянні між LSTM та ансамблевою моделями.

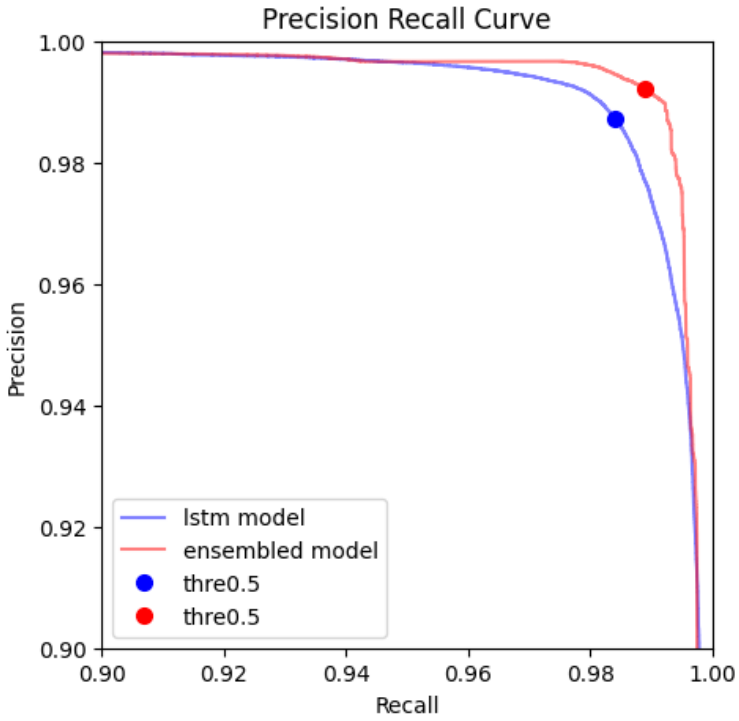


Рисунок 3.8 – Графік кривої Precision-Recall.

Наостанок, проведемо оцінку вихідної ансамблевої моделі за чотирьома показниками (рисунок 3.9 та 3.10):

1. Крива ROC (Receiver Operating Characteristic Curve - крива робочих характеристик приймача);
2. Крива Precision-Recall (Влучність-Повнота);
3. Крива Precision-Threshold (порогу влучності);
4. Крива Calibration (калібрування).

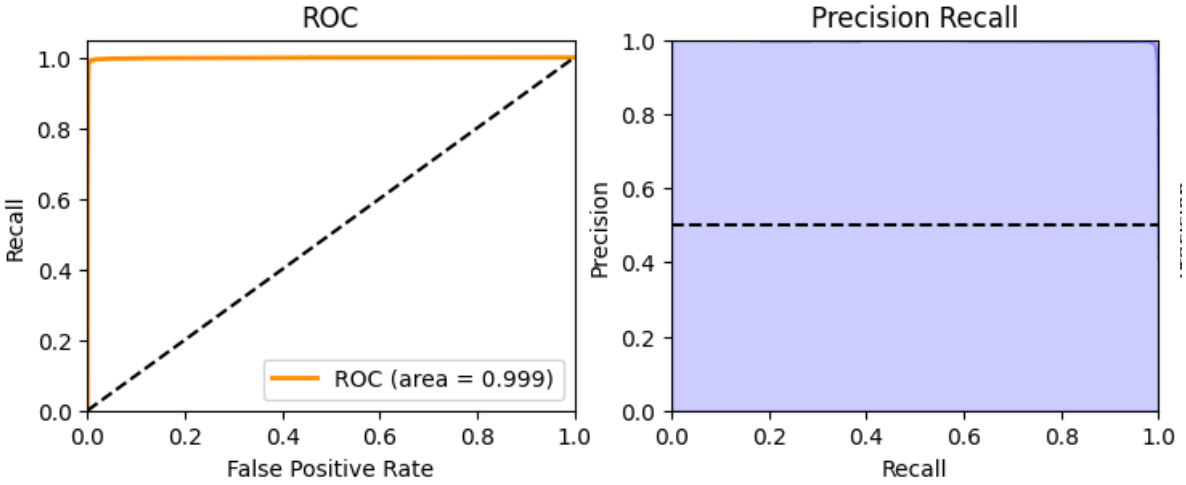


Рисунок 3.9 – Графіки кривих ROC та Precision-Recall.

ROC-крива – це графічне представлення продуктивності моделі класифікації при різних порогових значеннях. Вона ілюструє компроміс між частотою правильних спрацьовувань (Recall) і частотою хибних спрацьовувань (FPR). Частота правильних спрацьовувань вимірює частку фактично позитивних випадків, які модель правильно ідентифікує. З іншого боку, частота хибно позитивних результатів вимірює частку фактичних негативних випадків, які помилково класифікуються як позитивні. ROC-крива допомагає візуалізувати, як змінюється продуктивність моделі при зміні порогу дискримінації. Вища площа під кривою (AUC) вказує на кращу дискримінаційну здатність моделі, а значення 1 означає ідеальну класифікацію.

Крива Precision-Recall ілюструє баланс між влучністю та повнотою для різних порогових значень у класифікаційній моделі. Крива особливо корисна для оцінки моделей у незбалансованих наборах даних, де клас, що цікавить, зустрічається рідко, оскільки вона дає уявлення про роботу моделі на різних рівнях значення recall.

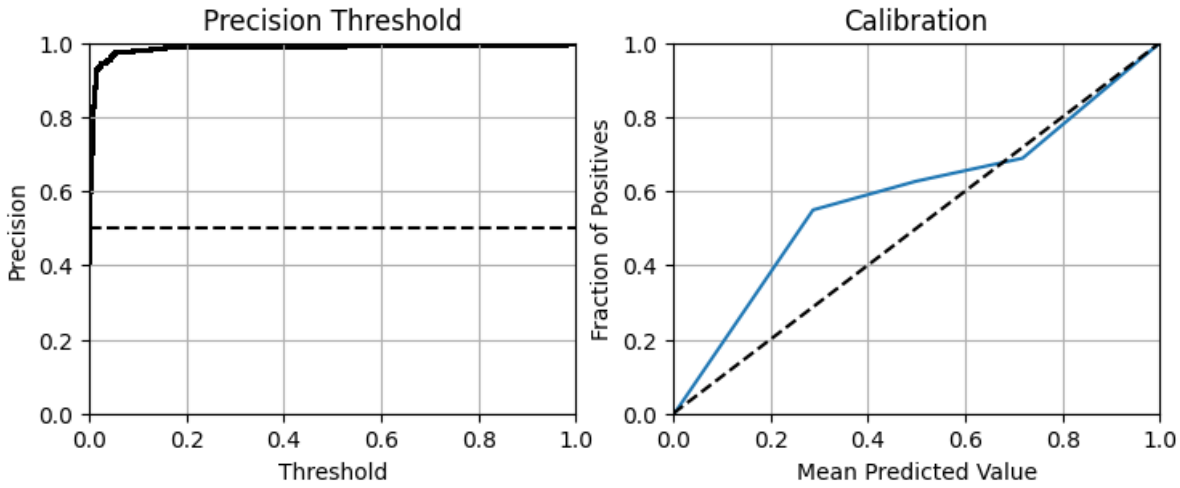


Рисунок 3.10 – Графіки кривих Precision-Threshold та Calibration.

Крива Precision-Threshold показує, як змінюється влучність моделі класифікації при зміні порогових значень для передбачення позитивного класу. Регулюючи поріг, який визначає поріг ймовірності для класифікації випадків як позитивних, можна спостерігати компроміс між точністю і кількістю істинно позитивних результатів. Ця крива дає детальне уявлення про те, як змінюється показник влучності моделі при різних порогах прийняття рішень, надаючи цінну інформацію для вибору відповідного порогу, виходячи з конкретних вимог або обмежень.

Ансамблева модель досягає як високої повноти (recall), так і високої влучності, що вказує на те, що ансамблева модель має високу здатність виявляти точки інтересу DGA у вибірці даних, а також високу здатність виявляти лише релевантні значення даних.

Але крива калібрації вказує на те, що модель потребує калібрування, оскільки вона має тенденцію передбачати менші значення. Крива лежить вище діагональної лінії, тобто модель є дещо самовпевненою. Це означає, що існує більше випадків з прогнозованою ймовірністю, близькою до 1, ніж повинно бути, і впевненість моделі у своїх прогнозах вища, ніж фактичний відсоток успішності.

Загалом, висвітлено підхід до виявлення зловмисних DGA доменів з використанням комбінації методів машинного та глибинного навчання. Продемонстровано ефективність підходу, застосовуючи модель довготривалої

короткочасної пам'яті (LSTM) та ансамблеву модель, яка включає логістичну регресію. Результати показують значне покращення точності та достовірності виявлення DGA доменів, причому ансамблева модель досягла точності більше 99,2%. Важливим факторм підкреслено необхідність врахування схильності DGA доменів до певних суфіксів доменів верхнього рівня (TLD) для покращення прогностичних характеристик системи. Надано детальні оцінки моделей з використанням різних метрик, включаючи ROC-криву, Precision-Recall криву, Precision-Treshold криву та Calibration криву, які дають вичерпну інформацію про продуктивність моделей.

ВИСНОВОК

Кваліфікаційна робота містить дослідження методів виявлення зловмисної активності за допомогою аналізу DNS-запитів, та запропоновано підхід на основі алгоритмів глибокого навчання. Досліджено алгоритми генерації доменів (DGA), які використовуються у ботнетах для створення безлічі доменних імен, що ускладнює виявлення та блокування зловмисних доменів. Виявлено обмеження традиційних підходів, таких як чорні списки та зворотна інженерія, і пропонується використання методів глибокого навчання для підвищення ефективності виявлення доменів DGA. Різні методи, в тому числі засновані на статистичних ознаках, n-грамах і рекурентних нейронних мережах (RNN), розглядаються на предмет їхнього потенціалу у виявленні доменів DGA.

Ключовою вимогою для рішення проблеми є виявлення зловмисної активності в режимі реального часу, із можливістю аналізу без необхідності доступу до низькорівневих даних, що іноді потребують дозволу на обробку від третьої сторони.

У ході дослідження були вивчені теоретичнні основи та методи глибокого навчання, а саме архітектури довгої короткочасної пам’яті (LSTM) та техніки ансамблевого навчання у комбінації з програмними компонентами, а саме стек технологій: Python, TensorFlow, Keras, Sklearn та Jupyter Notebook.

У результаті було сформовано вибірку даних із різних відкритих та закритих джерел, нормалізовано до готовності для тренування, спроектовано дві моделі глибокого навчання на основі архітектури LSTM та логістичної регресії. Перша виконувала завдання ідентифікації DGA доменів, друга – слугувала допоміжною технікою покращення точності за допомогою врахування схильностей DGA доменів до певних суфіксів доменів верхнього рівня.

Об’єднані в одну систему та доповнюючи одна одну, моделі на виході досягли очікуваних результатів ефективності прогностичної спроможності виявлення зловмисних DGA доменів із точністю понад 99,2%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гуренко М. Використання рекурентних нейронних мереж з довгою короткочасною пам'яттю (LSTM) для надійного виявлення доменів алгоритму генерації доменів (DGA). Інформаційне суспільство: проблеми та перспективи : матеріали IX Всеукр. наук.-практич. конф. (м. Одеса, 24 травня 2024 р.) / відп. ред. Н.І.Логінова. Одеса, 2024, ДЕПОНОВАНО
2. Гуренко М. Виявлення зловмисної активності в корпоративних локальних мережах за допомогою аналізу DNS-запитів. Матеріали Міжнародної науково-практично конференції «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. с.49-51.
3. Sharifnya R., Abadi M. A novel reputation system to detect DGA-based botnets / ICCKE 2013. – IEEE, 2013. – С. 417-423.
4. Silva S. S. C. et al. Botnets: A survey / Computer Networks. – 2013. – Т. 57. – №. 2. – С. 378-403.
5. Bilge L. et al. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis / Ndss. – 2011. – С. 1-17.
6. Zhou Y. et al. DGA-Based Botnet Detection Using DNS Traffic / J. Internet Serv. Inf. Secur. – 2013. – Т. 3. – №. 3/4. – С. 116-123.
7. Yu B. et al. Character level based detection of DGA domain names / 2018 international joint conference on neural networks (IJCNN). – IEEE, 2018. – С. 1-8.
8. Yadav S. et al. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis / IEEE/Acm Transactions on Networking. – 2012. – Т. 20. – №. 5. – С. 1663-1677.
9. Plohmann D. et al. A comprehensive measurement study of domain generating malware / 25th USENIX Security Symposium (USENIX Security 16). – 2016. – С. 263-278.
10. Shin S. et al. A large-scale empirical study of conficker / IEEE Transactions on Information Forensics and Security. – 2011. – Т. 7. – №. 2. – С. 676-690.

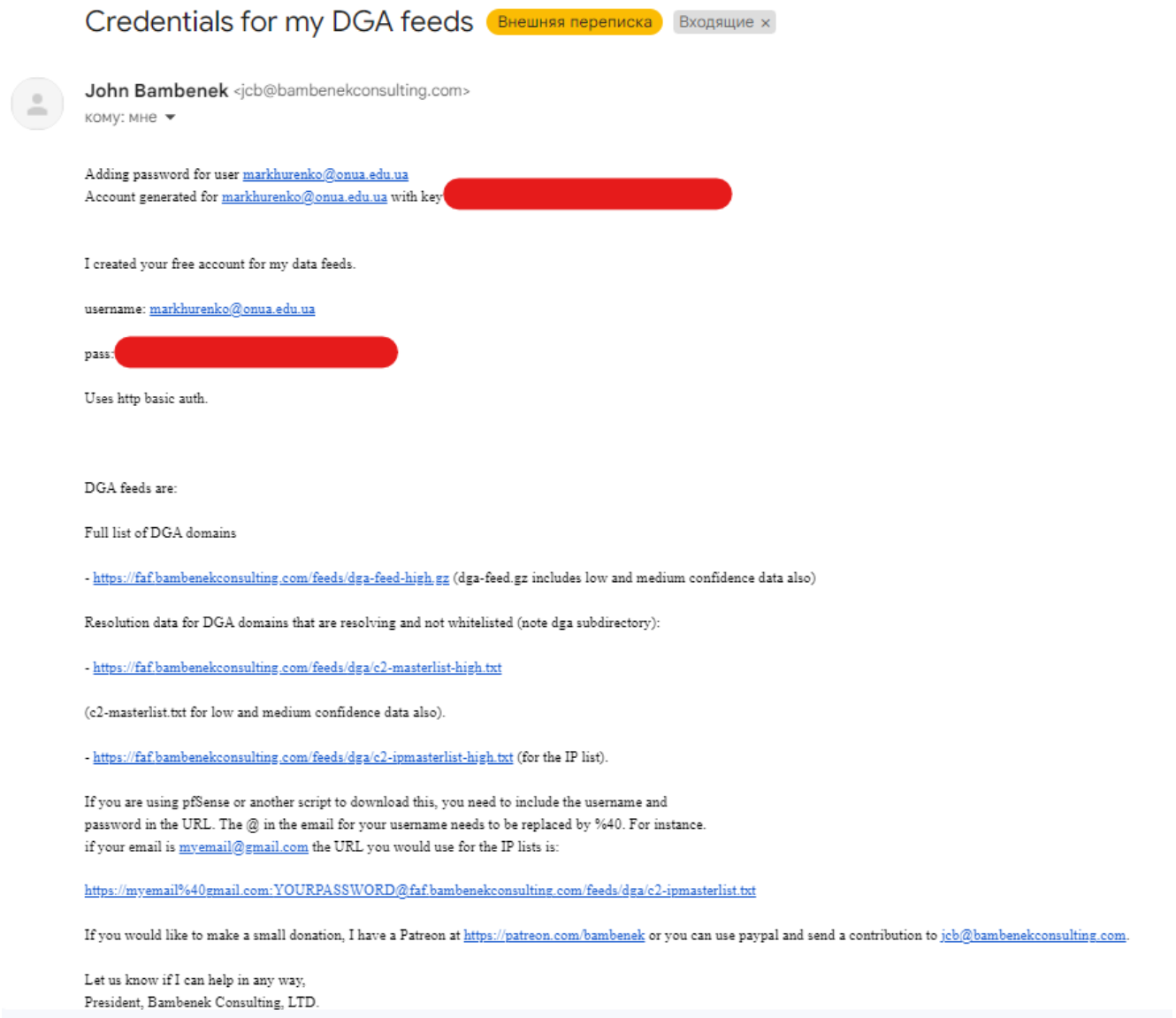
11. Schiavoni S. et al. Phoenix: DGA-based botnet tracking and intelligence / International Conference on detection of intrusions and malware, and vulnerability assessment. – Cham : Springer International Publishing, 2014. – С. 192-211.
12. Schiavoni S. et al. Phoenix: DGA-based botnet tracking and intelligence / International Conference on detection of intrusions and malware, and vulnerability assessment. – Cham : Springer International Publishing, 2014. – С. 192-211.
13. Fu Y. et al. Stealthy domain generation algorithms / IEEE Transactions on Information Forensics and Security. – 2017. – Т. 12. – №. 6. – С. 1430-1443.
14. Song W. J., Li B. A method to detect machine generated domain names based on random forest algorithm / 2016 International Conference on Information System and Artificial Intelligence (ISAI). – IEEE, 2016. – С. 509-513.
15. Tong V., Nguyen G. A method for detecting DGA botnet based on semantic and cluster analysis / Proceedings of the 7th Symposium on Information and Communication Technology. – 2016. – С. 272-277.
16. Kührer M., Rossow C., Holz T. Paint it black: Evaluating the effectiveness of malware blacklists / Research in Attacks, Intrusions and Defenses: 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings 17. – Springer International Publishing, 2014. – С. 1-21.
17. Zhang Y., Zhang Y., Xiao J. Detecting the DGA-based malicious domain names / Trustworthy Computing and Services: International Conference, ISCTCS 2013, Beijing, China, November 2013, Revised Selected Papers. – Springer Berlin Heidelberg, 2014. – С. 130-137.
18. Luo X. et al. Dgasensor: Fast detection for dga-based malwares / Proceedings of the 5th International Conference on Communications and Broadband Networking. – 2017. – С. 47-53.
19. Ferrante A. J. The impact of GDPR on WHOIS: Implications for businesses facing cybercrime / Cyber Security: A Peer-Reviewed Journal. – 2018. – Т. 2. – №. 2. – С. 143-148.
20. Tran D. et al. A LSTM based framework for handling multiclass imbalance in DGA botnet detection / Neurocomputing. – 2018. – Т. 275. – С. 2401-2413.

21. Curtin R. R. et al. Detecting DGA domains with recurrent neural networks and side information / Proceedings of the 14th international conference on availability, reliability and security. – 2019. – С. 1-10.
22. Yu B. et al. Character level based detection of DGA domain names / 2018 international joint conference on neural networks (IJCNN). – IEEE, 2018. – С. 1-8.
23. Woodbridge J. et al. Predicting domain generation algorithms with long short-term memory networks / arXiv preprint arXiv:1611.00791. – 2016.
24. Qiao Y. et al. DGA domain name classification method based on long short-term memory with attention mechanism / Applied sciences. – 2019. – Т. 9. – №. 20. – С. 4205.
25. Yadav S. et al. Detecting algorithmically generated malicious domain names / Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. – 2010. – С. 48-61.
26. Zhang W., Gong J., Liu Q. Detecting machine generated domain names based on morpheme features / 1st International Workshop on Cloud Computing and Information Security. – Atlantis Press, 2013. – С. 408-411.
27. Selvi J., Rodríguez R. J., Soria-Olivas E. Detection of algorithmically generated malicious domain names using masked N-grams / Expert systems with applications. – 2019. – Т. 124. – С. 156-163.
28. Alaeiyan M. et al. Detection of algorithmically-generated domains: An adversarial machine learning approach / Computer Communications. – 2020. – Т. 160. – С. 661-673.
29. Yang L. et al. Detecting word-based algorithmically generated domains using semantic analysis / Symmetry. – 2019. – Т. 11. – №. 2. – С. 176.
30. Patsakis C., Casino F. Exploiting statistical and structural features for the detection of domain generation algorithms / Journal of Information Security and Applications. – 2021. – Т. 58. – С. 102725.

31. Shibahara T. et al. Efficient dynamic malware analysis based on network behavior using deep learning / 2016 IEEE Global Communications Conference (GLOBECOM). – IEEE, 2016. – С. 1-7.
32. Curtin R. R. et al. Detecting DGA domains with recurrent neural networks and side information / Proceedings of the 14th international conference on availability, reliability and security. – 2019. – С. 1-10.
33. Bisio F. et al. Real-time behavioral DGA detection through machine learning / 2017 International Carnahan Conference on Security Technology (ICCST). – IEEE, 2017. – С. 1-6.
34. Understanding LSTM Networks. URL: https:/ colah.github.io/posts/2015-08-Understanding-LSTMs
35. Hochreiter S., Schmidhuber J. Long short-term memory / Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780.
36. Akarsh S. et al. Deep learning framework for domain generation algorithms prediction using long short-term memory / 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). – IEEE, 2019. – С. 666-671.
37. Vinayakumar R., Poornachandran P., Soman K. P. Scalable framework for cyber threat situational awareness based on domain name systems data analysis / Big data in engineering applications. – 2018. – С. 113-142.
38. Namgung J., Son S., Moon Y. S. Efficient deep learning models for DGA domain detection / Security and Communication Networks. – 2021. – Т. 2021. – С. 1-15.
39. Tranco. A Research-Oriented Top Sites Ranking Hardened Against Manipulation. URL: https:/ tranco-list.eu/

ДОДАТКИ

Додаток А. Дозвіл на використання даних Bambenek Consulting



Credentials for my DGA feeds Внешняя переписка Входящие ×

John Bambenek <jcb@bambenekconsulting.com>
кому: мне

Adding password for user markhurenko@onua.edu.ua
Account generated for markhurenko@onua.edu.ua with key [redacted]

I created your free account for my data feeds.

username: markhurenko@onua.edu.ua

pass: [redacted]

Uses http basic auth.

DGA feeds are:

Full list of DGA domains

- https://faf.bambenekconsulting.com/feeds/dga-feed-high.gz (dga-feed.gz includes low and medium confidence data also)

Resolution data for DGA domains that are resolving and not whitelisted (note dga subdirectory):

- https://faf.bambenekconsulting.com/feeds/dga/c2-masterlist-high.txt

(c2-masterlist.txt for low and medium confidence data also).

- https://faf.bambenekconsulting.com/feeds/dga/c2-ipmasterlist-high.txt (for the IP list).

If you are using pfSense or another script to download this, you need to include the username and password in the URL. The @ in the email for your username needs to be replaced by %40. For instance. if your email is myemail@gmail.com the URL you would use for the IP lists is:

https://myemail%40gmail.com:YOURPASSWORD@faf.bambenekconsulting.com/feeds/dga/c2-ipmasterlist.txt

If you would like to make a small donation, I have a Patreon at https://patreon.com/bambenek or you can use paypal and send a contribution to jcb@bambenekconsulting.com.

Let us know if I can help in any way,
President, Bambenek Consulting, LTD.

Рисунок А.1 – Електронний лист від президента фірми із погодженим запитом на доступ до даних Bambenek Consulting.