

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

кібербезпеки

(повна назва кафедри)

Дипломна робота

Застосування методів машинного навчання(ML) для розпізнавання DGA доменів.

Виконав: студент групи ПМК-41

спеціальності

125 «Кібербезпеки»

(шифр і назва спеціальності)

Фик М.Р.

(підпис)

(прізвище та ініціали)

Керівник

Венгершич П.С.

(підпис)

(прізвище та ініціали)

Рецензент

Шербина Ю.М.

(підпис)

(прізвище та ініціали)



2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики _____

Кафедра Кібербезпеки _____

Спеціальність 125 «Кібербезпека» _____

(шифр і назва)

«ЗАТВЕРДЖУЮ»

Завідувач кафедри _____

"31" серпня 2022 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Фик Максим Ростиславович

(прізвище, ім'я, по батькові)

1. Тема роботи Застосування методів машинного навчання(ML) для розпізнавання DGA доменів

керівник роботи Венгерський Петро Сергійович - завідувач кафедри кібербезпеки, проф. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від " 13" вересня 2022 року № 15

2. Строк подання студентом роботи 13.06.2023р.

3. Вихідні дані до роботи розробка ефективної та точної системи машинного навчання для класифікації доменів і виявлення адрес, створених за допомогою алгоритму генерації доменних імен (DGA), а також поліпшення розпізнавання та класифікації іноземних доменів.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

Робота включає огляд літератури щодо попередніх досліджень, використаних методологій збору та обробки даних, розробку моделі машинного навчання, оцінку та порівняння результатів для визначення ефективності розробленої системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Презентація доповіді, виконана у Microsoft Point.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 31 серпня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Написання вступу та збір матеріалу	12.05.2023	виконано
2	Написання першого розділу	14.05.2023	виконано
3	Написання другого розділу	20.05.2023	виконано
4	Написання третього розділу	26.05.2023	виконано
5	Написання четвертого розділу	04.06.2023	виконано
6	Підготовка та оформлення презентації для доповіді	08.06.2023	виконано

Студент  (підпис) Фик М.Р. (прізвище та ініціали)

Керівник роботи  (підпис) Венгерський П.С. (прізвище та ініціали)

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО БОТНЕТ ТА ДОМЕННІ ІМЕНА.....	6
1.1 Що таке ботнет?	6
1.2 Доменні імена та їх структура.....	7
РОЗДІЛ 2. ПРОБЛЕМА ВИЯВЛЕННЯ DGA.....	9
2.1 DGA та принцип його роботи.....	9
2.2 Проблема виявлення DGA.....	10
РОЗДІЛ 3. МЕТОДИ ВИЯВЛЕННЯ DGA.....	12
3.1 Застарілі методи виявлення DGA.....	12
3.2 Використання машинного навчання у боротьбі з DGA.....	14
РОЗДІЛ 4. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ РІЗНОМОВНОГО DGA ЗА ДОПОМОГОЮ ML.....	17
4.1 Пошук даних та вибір інструментів.....	17
4.2 Підготовка даних.....	20
4.3 Тренування моделі.....	26
4.4 Тестування моделі на нових даних.....	30
ВИСНОВОК.....	35
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	37

ВСТУП

Зі стрімким розвитком технологій та зростанням цифровізації в усіх сферах нашого життя, мережа Інтернет стала невід'ємною частиною нашого повсякденного існування. Ми щодня заглиблюємося у величезний океан кіберпростору, завантажуючи триліони терабайт персональних даних, від особистих фотографій та текстових повідомлень до банківських даних та медичних записів. Ми цінуємо комфорт та легкість, які нам пропонує Інтернет, тасподіваємося на безпеку наших даних у цій всесвітній павутині.

Проте, зі збільшенням обсягів даних зростає й інтерес до них з боку зловмисників. Кіберзлодії постійно шукають слабкі місця в системах захисту, використовуючи їх для отримання приватних даних користувачів без їх відома або згоди. Вони можуть заважати власникам даних отримувати до них доступ, щоє прямим порушенням закону про захист персональних даних.

Останніми роками ми стали свідками безлічі спалахів шкідливого програмного забезпечення, які спричинили значні збитки в урядових установах, енергетичному сектору, промисловості та інших ключових областях інформаційної інфраструктури. Викрадена інформація, що включає конфіденційні дані, фінансові записи, звіти, плани та особисту інформацію, може бути використана зловмисниками для маніпуляцій, шантажу або навіть нанесення катастрофічної шкоди організаціям або індивідуальним користувачам.

Захист від шкідливого програмного забезпечення стає життєво важливим для забезпечення інтернет-безпеки. Незважаючи на значні прогресивні кроки, зроблені в галузі технологій безпеки, існують певні види атак, які все ще складно виявити та нейтралізувати фахівцям з кібербезпеки. Однією з таких складних атак є поширення вірусів за допомогою ботнетів. Ці мережі зловмисних ботів можуть незпомітно інфікувати комп'ютери та інші пристрої, призводячи до широкомасштабних порушень безпеки.

1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО БОТНЕТ ТА ДОМЕННІ ІМЕНА

1.1 Що таке ботнет?

Ботнет – це мережа серверів під'єднаних до інтернету, керування якими відбувається дистанційно через Command and Control (далі С&С) канали серверу. Розмір групи заражених девайсів може сильно різнитись: це може бути як і декілька поодиноких жертв, так і величезна павутина на тисячі пристроїв.

Комп'ютер, що став жертвою ботнету називається ботом. Ботнет використовується для викрадення особистих даних користувачів, таких як номери і паролі кредитних карт, паролі та інша незаконна діяльність, включаючи поширення спаму і запуск шкідливого програмного забезпечення. Бот встановлюється на комп'ютері користувача без його відому і являє собою зачасти приховану програму. Найефективнішим способом поширення ботнетів є використання Інтернету або мережі комп'ютерів. Вони використовують DNS- систему доменних імен.

«Техніка потоку доменів» використовується для збереження шкідливого ботнету в роботі шляхом постійної зміни доменного імені С&С сервера. Сервери С2 слугують командними центрами, які шкідливі програмні забезпечення, пов'язані з цілеспрямованими атаками, використовують для зберігання викрадених даних або отримання команд. Керівний центр С&С забезпечує вузол зв'язку між всіма ботами для передачі даних та апдейтів між ними. Щоб залишатися в роботі, ботнет та його мозок С2 постійно розвиваються, оновлюються та шукають нові шляхи для приховування своєї діяльності.

Ланцюжок комунікації виглядає так:

1. Майтсер ботів надсилає бот-мережі завдання.
2. Ботнет виконує команду майстера.
3. Боти передають результати роботи назад.

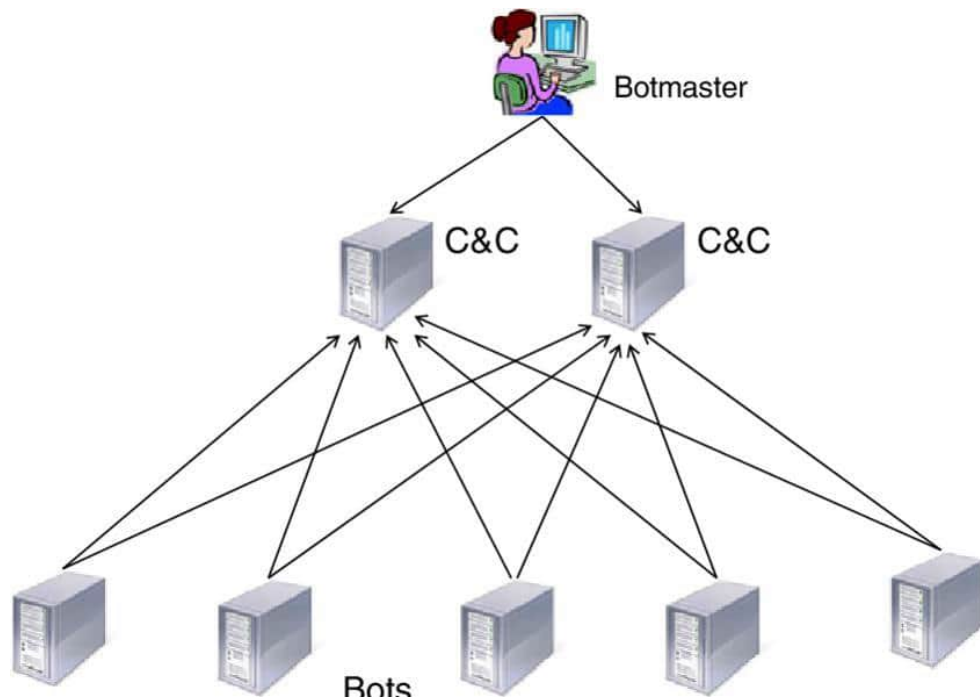


Рисунок 1.1 – Діаграма комунікації ботнету.

Виявлення ботнету можливе лише тоді, коли встановлено місцезнаходження сервера C&C і присутній надійний зв'язок між серверами та комп'ютерами.

Існують три типи архітектури C&C - централізована, децентралізована та гібридна. У централізованій архітектурі, майстер ботів керує всіма активними ботами з централізованого сервера C&C. Цей тип архітектури набагато легше контролювати, але дизайн зазвичай є складним, хоч і затримка повідомлень та живучість - короткими. У децентралізованій архітектурі мережі ботів містять більше одного сервера C&C, і кожен бот діє як C&C сервер та клієнт. Цей тип архітектури є складнішим для виявлення, проте затримка повідомлень та живучість вища, хоч і шанси на провал менші, порівняно з централізованими архітектурами. Гібридна архітектура поєднує елементи як централізованої, так і децентралізованої архітектури. Завдання виявлення та моніторингу в гібридній архітектурі складніші, ніж централізовані та децентралізовані архітектури. Однак гібридна архітектура відрізняється від інших архітектур своєю спрощеною конструкцією

1.2 Доменні імена та їх структура

Доменні імена складаються з двох частин: домену верхнього рівня (TLD) та домену другого рівня (SLD), які розділені крапкою. Наприклад, у google.com, com – це TLD, а google – SLD. Це забезпечує зручність та легкість використання Інтернету, оскільки користувачам не потрібно запам'ятовувати числові IP-адреси,

а замість цього вони можуть використовувати звичайні та легкі для запам'ятовування доменні імена. DNS має розподілену базу даних, яка зберігає інформацію про доменні імена та відповідні їм IP-адреси, тому що ця інформація може бути розповсюджена по різних серверах по всьому світу. Доменні імена зазвичай є стабільними та пов'язаними з товарним знаком компанії або організації. Для створення доменних імен не обов'язково використовувати англійську мову. Доменні імена можуть складатись з будь-яких символів, включаючи літери різних алфавітів (наприклад, кирилиці, китайської мови, арабської мови), цифр і спеціальних символів. Однак, більшість доменних імен, особливо тих, які використовуються в Інтернеті, зазвичай складаються з англійських літер і цифр. Це пов'язано з історією розвитку Інтернету та технічними обмеженнями на довжину доменних імен, які були встановлені в початковому періоді розвитку Інтернету.

Необхідність у доменних іменах вже була усвідомлена під час ери ARPANET. Арпанет (Advanced Research Projects Agency Network - мережа передових досліджень) - це мережа, яку вважають початком Інтернету, що була створена за дорученням Міністерства Оборони США та за допомогою декількох наукових закладів. Основне завдання полягало в об'єднанні науково-дослідницьких та військових інститутів у США, щоб збільшити швидкість та покращити зручність обміну інформацією між ними. Крім того, в умовах Холодної війни стояла задача створити інфраструктуру, яка може пережити атомний удар. Спочатку централізоване місце надавало єдиний файл "hosts.txt", що містив відповідності між людськими іменами та числовими адресами. Тим часом кількість підключених хостів постійно збільшувалася. В результаті була опублікована система доменних імен в 1983 році.

Ієрархія DNS (Domain Name System) розділена на зони, які контролюються менеджером зон. Кожен рівень в ієрархії DNS має мітку, яка містить інформацію, що стосується імені домену. Ланцюжок DNS не може складатись більше ніж 127 рівнів. Організації можуть створювати власний простір імен доменів для забезпечення приватної мережі, яка не буде доступною для глобального Інтернету. Їх часто використовують для передачі надсекретних документів та чутливої інформації. Доменні мітки грають важливу роль в ідентифікації доменних імен. Вони складаються з одного або кількох розділів або піддоменів, розділених крапками. Кількість символів в мітках обмежена від 0 до 63, що дозволяє користувачам створювати унікальні імена. Шлях від піддомену до кореня домену називається повноцінним ім'ям домену.

Оскільки загалом існує безліч можливих доменних імен, зловмисники придумали як використати це для своїх цілей.

2. ПРОБЛЕМА ВИЯВЛЕННЯ DGA

2.1 DGA та принцип його роботи

Зловмисники часто використовують алгоритм генерації домену (DGA), щоб захистити та уникнути блокування домену свого сервера C&C. Така методика називається Domain Fluxing. DGA безперервно генерує кілька адрес, використовуючи певний початковий код. Майстер ботів використовує згенеровані імена, для обходу чорного списку і евристичної методи виявлення DGA. Зловмисне програмне забезпечення припускає, що домен, створений DGA, є доменом сервера C&C, і намагається підключитися до сервера C&C. Для зв'язку з C2 сервером бот-мережі використовують дві адреси: abc.com, який не зареєстрований і не має IP-адрес, і def.com, який зареєстрований і може зв'язатися з сервером. Під час атаки Domain Flux-у бот-мережі звертаються до свого майстра власноруч створеними доменними іменами за допомогою методу звернення та випробування. Цей метод може генерувати кілька неіснуючих запитів для доменів, які не мають IP-адрес (NXDomains). Однак, в більшості випадків, бот-мережі використовують зареєстрований домен def.com для зв'язку з C&C сервером.

Таким чином, дуже важливо виявити динамічну адресу сервера C&C, оскільки DGA постійно змінює ім'я домену. Також важливо класифікувати, який DGA генерує домен, оскільки DGA мають різні цілі та типи, наприклад, програми-вимагачі та банківські трояни.

Щоб краще зрозуміти DGA, потрібно поглянути на те, як воно генерує домени та складність, пов'язану з виявленням. Зловмиснику необхідно відтворити ті ж результати, що й його шкідливе програмне забезпечення, вбудоване в DGA. Зазвичай для генерації доменів потрібно:

- Загальне насіння (seed), яке може бути будь-чим: системна дата і час, курс валюти, денна температура або навіть популярні теми Twitter і Facebook.
- Список доменів верхнього рівня: .com, .org, .cc, .net, і так далі; механізм їх додавання. Зловмисники навіть почали використовувати екзотичні TLD, такі як .tickets, .blackfriday та .feedback
- Псевдовипадковий генератор з використанням насіння.

2.2 Проблема виявлення DGA

Отож, якби усі сервери ботнету постійно мали б однакову адресу, то їх було б дуже легко знаходити та знешкоджувати. Постійна зміна адрес призводить до стійких серверів С2, які важче заблокувати правоохоронним органам, оскільки зловмисники завжди можуть зареєструвати наступний домен за допомогою алгоритму. Через те, що шкідливе програмне забезпечення намагається зв'язатись з великою кількістю згенерованих доменних імен щодня, правоохоронним органам стає важко ефективно вимкнути ботнети. Деякі автори шкідливих програм реєструють домени всього за декілька днів або годин до того, як шкідлива програма встановлює зв'язок зі своїм С&С-сервером. Використання криптографії із відкритим ключем у коді шкідливих програм робить неможливим для правоохоронних органів та інших суб'єктів імітувати команди контролерів шкідливого програмного забезпечення.

Наприклад, Conficker (відомий також як Downup, Downadup та Kido) - це комп'ютерний черв'як, який поширювався в 2008-2009 роках та спричинив значну шкоду в Інтернеті. Цей черв'як використовував різні вразливості в операційній системі Windows для швидкого поширення через мережу. Черв'як міг використовувати комп'ютери, щоб створювати ботнет, з метою вчинення злочинних дій, таких як атаки на інші комп'ютери або розсилання спаму.

Conficker також використовував складні методи захисту, щоб уникнути виявлення та видалення. Він був одним з найбільш вразливих та шкідливих комп'ютерних черв'яків свого часу. Conficker генерував 250 доменних імен на день, тоді як Conficker.c генерував 50 000 доменних імен на день. Щоб запобігти оновленню шкідливого програмного забезпечення, правоохоронним органам потрібно було б попередньо зареєструвати 50 000 нових доменних імен щодня. Також, вони могли розраховувати атаки до хвилини, оскільки динамічна природа DGA ускладнює виявлення. Інші автори шкідливих програм використовують подібну стратегію.

Згідно з даними компанії Damballa, яка спеціалізується на мережевій безпеці, у 2011 році топ-5 сімей злочинних програм на основі DGA включали Conficker, Murofet, BankPatch, Bonnana та Vobax.

Крім того, складніші версії DGA можуть використовувати словники для створення доменних імен, які можуть бути зашифровані у шкідливому програмному забезпеченні або взяті з загальнодоступних джерел. Словникова DGA створює домени, які важко виявити через їх схожість з легітимними доменами. Тобто замість незрозумілого набору символів, як `jfhdkcold.com`, такі алгоритми видають зрозумілий людині домен, як `healthcareclinic124.com`. Таким доменам люди схильні довіряти набагато

більше. Як було згадано раніше, для створення доменних імен не обов'язково використовувати англійську мову.

Доменні імена можуть складатись з будь-яких символів, включаючи літери різних алфавітів. Факт того, що DGA може використовувати будь-яку мову для створення доменних імен, створює проблеми для їх виявлення. Багато з традиційних методів виявлення шкідливого програмного забезпечення ґрунтуються на аналізі підозрілих доменних імен англійською мовою. Однак, якщо DGA використовує іншу мову, то ці методи можуть стати непридатними.

Також, DGA може використовувати словникові або набір символів, що містять літери, що дуже схожі на латинські літери. Це також ускладнює виявлення DGA, оскільки ці доменні імена можуть бути дуже схожими на легітимні домени англійською мовою.

Таким чином, змінність мови і символів, які використовуються в DGA, ускладнюють виявлення шкідливого програмного забезпечення за допомогою традиційних методів. Для успішного виявлення DGA необхідно використовувати більш складні методи аналізу, які можуть розпізнавати іншомовні та символічні доменні імена.

3. МЕТОДИ ВИЯВЛЕННЯ DGA

3.1 Застарілі методи виявлення DGA

Спершу слід визначити чіткі вимоги до системи виявлення DGA:

1. Високий відсоток істинних спрацьовувань, щоб виявити всю зловмиснуактивність.
2. Низький рівень помилкових спрацьовувань, щоб уникнути перевантаження операторів мережі нешкідливими даними.
3. Висока швидкість виявлення, щоб оператори могли швидко помістити заражені комп'ютери в карантин
4. Правильна кластеризація заражених хостів, щоб отримати уявлення про підмножини мережі, заражених одним і тим же шкідливим програмним забезпеченням. Це дозволить мережевим операторам легко відстежити, як шкідливе програмне забезпечення потрапило в систему.

Найпростішим, але неефективним способом боротьби з доменами DGA є використання чорного списку, де кожен домен блокується окремо. Проте цей підхід не є поганим та застарілим вибором, оскільки системою неперервно створюється велика кількість нових доменів. Зворотна інженерія також використовується для виявлення DGA шляхом ідентифікації їх у шкідливих доменах і створення блокувальних правил для цих доменів. Однак, така методика може виявити лише обмежену кількість DGA і потребує значних зусиль та часу від експертів. Таким чином, метод зворотної інженерії є повільним і практично непридатним для широкомасштабного виявлення і блокування DGA.

Хороший приклад – це ботнет Srizbi, також відомий як Sberplay та Exchanger, був одним з найпотужніших та шкідливих ботнетів свого часу. Виникнення цього ботнету спостерігалось в період з 2007 по 2008 рік. Він був відповідальний за широкомасштабну діяльність, включаючи розсилку спаму, поширення шкідливих програм та викрадення конфіденційної інформації. Одним з особливих аспектів Srizbi було якраз використання алгоритмів доменної генерації (DGA) для забезпечення комунікації між інфікованими комп'ютерами та C&C-сервером. Такий підхід дозволяв Srizbi уникнути перешкод і продовжувати свою шкідливу діяльність.

Боротьба з Srizbi була викликом для експертів в кібербезпеці. Існувала потреба у вдосконаленні наявних методів виявлення та блокування DGA доменів. Експерти займалися зворотною інженерією шкідливих доменів,

вивчали їх алгоритми генерації та створювали блокувальні правила для доменів, пов'язаних з Srizbi. Однак, зворотна інженерія була повільним та ресурсомістким процесом, оскільки зловмисники постійно вносили зміни у свої алгоритми. Внесення змін згодом займало лічені хвилини, у той час як експерт витрачав дні на дослідження патернів у доменах.

Помітні успіхи в боротьбі з Srizbi були досягнуті завдяки спільним зусиллям багатьох організацій і фахівців. Якраз застосування алгоритмів машинного навчання та аналізу поведінки мережі дозволило виявити характеристики, що вказували на наявність Srizbi ботнету. У результаті цього дослідження були розроблені алгоритми виявлення Srizbi, які базувалися на машинному навчанні та статистичних методах. Виявлені аномалії трафіку допомогли виявити комп'ютери, що були уражені ботнетом. Боротьба з Srizbi також включала співпрацю з реєстраторами доменних імен та інтернет-провайдерами для блокування шкідливих доменів та комп'ютерів, що з'єднувалися з C&C-сервером Srizbi. Варто відзначити, що Srizbi ботнет послужив каталізатором для покращення заходів безпеки Інтернету та вдосконалення методів виявлення та протидії шкідливим ботнетам. Вчинені в цьому напрямку кроки сприяли забезпеченню кращої безпеки для користувачів та інтернет-інфраструктури загалом.

Такі великі ботнети, як Srizbi показали фахівцям, що існує потреба в більш прогресивних підходах до боротьби з DGA, як от використання алгоритмів машинного навчання та штучного інтелекту, що можуть допомогти виявляти та блокувати DGA динамічно, аналізуючи патерни генерації доменів та виявляючи відхилення. Крім того, використання аналізу поведінки мережі, виявлення аномалій та використання характеристик шкідливих доменів можуть також допомогти в боротьбі з DGA. Продовження досліджень у цій галузі та розробка більш ефективних методів захисту є важливими завданнями для боротьби з цією загрозою.

Основна ідея виявлення динамічних доменів полягає в тому, що послідовності символів, що використовуються в легітимних доменних іменах, відрізняються від послідовностей символів доменних імен, отриманих за допомогою DGA, тому що легітимний домен часто має смислове навантаження. Отже, якщо доменне ім'я складається з випадкового набору букв та цифр, то це ім'я швидше за все було згенероване DGA.

Щоб подолати недоліки ранніх методів чорного списку та зворотної інженерії, багато досліджень намагалися використовувати статистичні характеристики доменних імен: розподіл символів і чисел у межах доменних імен використовували для розрізнення кластерів доменів DGA та не DGA. Однак ці методи забирають багато часу і зловмисник може їх легко уникнути.

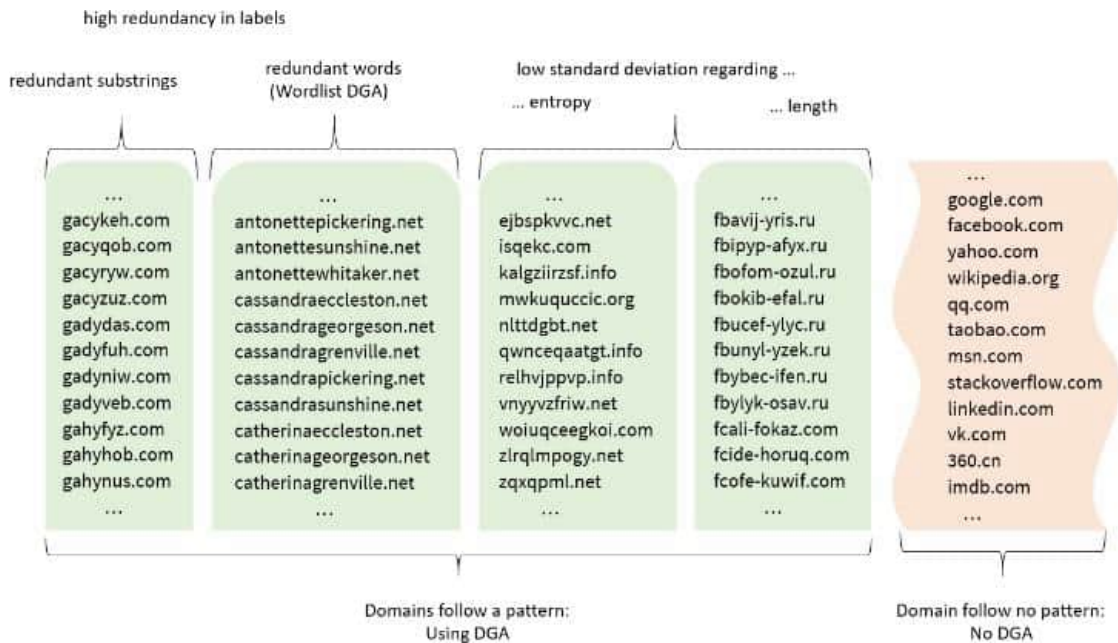


Рисунок 3.1 – Показ різниці між типами доменів.

Виявлення DGA за допомогою більшості сьогоденних тактик має недолік у тому, що вони засновані на аналізі пакетів трафіку, що означає, що домени DGA, швидше за все, встановлять зв'язок до того, як їх виявлять. В ідеалі ми повинні виявляти домени DGA в режимі реального часу, прогножуючи на основі кожного домену, щоб запобігти комунікації C&C.

3.2 Використання машинного навчання у боротьбі з DGA

Одним з найновітніших і водночас найбільш перспективних методів боротьби з розростанням бот-мереж у цифровому просторі є використання механізмів машинного навчання. Ці адаптивні, розумні технології відрізняються здатністю розпізнавати можливі атаки ще на зародковому етапі, іноді навіть перед тим, як вони фактично виникнуть.

Машинне навчання, що використовується для виявлення бот-мереж, базується на принципах нечіткої логіки. Це дає цим системам велику перевагу перед традиційними методами детекції, які можуть бути менш ефективними у невпевненому і непередбачуваному світі кіберзлочинів.

Розробники, що працюють над створенням систем на основі машинного навчання для виявлення бот-мереж, обирають відповідні особливості або так звані класифікатори. Ці класифікатори використовуються як основа для створення майбутніх моделей, що відповідають за прийняття рішень і виявлення можливих атак.

При розробці систем виявлення DGA (Domain Generation Algorithms), є два ключові класифікатори, які вважаються незмінними: це доменне ім'я та його тип (чи було воно сгенероване за допомогою DGA, чи належить воно до людського

домену). Ця інформація дозволяє моделі перевіряти власну ефективність, отримуючи зворотний зв'язок. І хоча є інші важливі дані, які можуть бути включені в модель, їх використання не є обов'язковим, проте їх наявність може значно покращити якість роботи моделі.

Хороший приклад використання додаткових класифікаторів був продемонстрований в роботі [2] професорів Дж. Средева і Поннам Пуджа. Їх модель використовувала датасет, що складався з 16 класифікаторів, включаючи такі параметри, як кількість субдоменів у адресі, середня довжина субдоменів, кількість небуквених символів, кількість цифр та інші.

Features	Ex: prata.pt	Ex: tbaxcrxirtmuusq.eu
DNL (Domain Name Length)	8	19
NoS (Number of Subdomains)	1	1
SLM (Subdomain Length Mean)	5.0	16.0
HwP (Has www Prefix)	0	0
HVTLD (Has a Valid Top Level Domain)	1	1
CSCS (Contains Single-Character Subdomain)	0	0
CTS (Contains Top Level Domain as Subdomain)	0	0
UR (Underscore Ratio)	0.0	0.0
CIPA (Contains IP Address)	0	0

Рисунок 3.2 – Класифікатори використані використані у роботі [2] професорів Дж. Средева і Поннам Пуджах .

Однак, незважаючи на потенційні переваги, використання машинного навчання в цій області має власні виклики. По-перше, ефективне тренування моделі вимагає великої кількості вхідних даних. Це означає, що для досягнення оптимальних результатів потрібно зібрати та проаналізувати мільйони прикладів доменів, створених за допомогою DGA. Крім того, ці моделі потребують постійного оновлення та підтримки, оскільки зловмисники постійно винаходять нові способи обходу систем захисту.

Ще одним викликом є очищення і форматування вхідних даних. Дані часто надходять з різних джерел, у різних форматах, і вони мають бути уніфіковані і очищені від шуму перед тим, як їх можна використовувати для тренування моделей. Це може бути досить трудомістким процесом, що ще більше ускладнює

розробку.

Використання машинного навчання для виявлення бот-мереж також може мати інші недоліки. Наприклад, залежно від якості використовуваного датасету та вибраних класифікаторів, модель може мати проблеми з виявленням нових атак або складних варіацій DGA. Крім того, залежно від розміру мережі та ресурсів, що необхідні для її підтримки та функціонування, можуть виникати проблеми з продуктивністю та швидкодією.

Ще важливо відзначити, що більшість доступних в інтернеті датасетів містять переважно англomовні домени, такі як facebook.com, gmail.com, nytimes.com та інші. Системи класифікації доменів, як правило, впливають на впізнавання доменів відповідно до даних, на яких вони були навчені. Внаслідок переважної кількості англomовних доменів у доступних датасетах, ці системи найчастіше оптимізовані для роботи з англomовними адресами. Тому, коли вони зустрічаються з доменами на інших мовах, вони можуть робити помилки у класифікації, оскільки вони раніше не бачили достатньо прикладів таких доменних імен. У цій роботі, я маю намір розглянути цю проблему та знайти способи її вирішення.

4. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ РІЗНОМОВНОГО DGA ЗА ДОПОМОГОЮ ML

4.1 Пошук даних та вибір інструментів

Розробку системи / моделі для класифікації доменів поділемо на 4 етапи:

1. Пошук даних та вибір інструментів для отримання найкращого результату найоптимальнішими шляхами.
2. Підготовка даних, їх аналіз та очищення.
3. Тренування моделі.
4. Тестування отриманої моделі на нових даних.

На щастя, у мережі є декілька доступних датасетів, що дозволить нам спробувати розробити свою систему виявлення DGA. Я знайшов датасет [5] з понад 500 тисячами доменів. “Підозрілі” домени були згенеровані 25 різними сімействами DGA. Цей набір був отриманий з репозиторію проекту Netlab Opendata (data.netlab.360.com/dga). В якості авторитетного джерела для отримання нормальних доменних імен було використано сервіс Alexa. Загалом, в цьому наборі даних було зібрано 675 000 доменних імен. Половина з них, тобто 50%, була згенерована за допомогою алгоритмів DGA, а інша половина складалась з доменів Alexa. Більша частина адрес у цьому списку є англомовними.

Оскільки мені не вдалося знайти у вільному доступі саме різномовних датасетів, я підійшов до вирішення проблеми іншим шляхом: за допомогою використання сервісів нейромережевого машинного перекладу, а саме Google Translate.

Google Translate - це безкоштовний онлайн-сервіс від Google, який автоматично перекладає тексти з однієї мови на іншу. Він використовує штучний інтелект та машинне навчання і підтримує велику кількість мов. Користувачі можуть використовувати його на веб-сайті або в мобільній програмі. Також існує зручна Python бібліотека для автоматизації перекладу.

Я обрав саме Google Translate, адже у нього є функція автоматичного розпізнавання вхідної мови, що є дуже важливим аспектом, так як ми не маємо вхідних даних щодо, до якої мови належить домен.

```
dga_prepare.ipynb  dga_domains_full.csv ×  dga_train.ipynb  dga_test.ipynb
1 type,origin, domain
2 dga,gozi,mortiscontrastatim.com
3 dga,corebot,cvylh1po636avyrsexebwbkn7.ddns.net
4 legit,alexa,plasticbags.sa.com
5 legit,alexa,mzltrack.com
6 legit,alexa,miss-slim.ru
7 dga,ranbyus,txumyqrubwutbb.cc
8 legit,alexa,myhostingpack.com
9 dga,symmi,ixekrihagimau.ddns.net
10 dga,emotet,rjyuosmhfnaedlyg.eu
11 dga,dircrypt,djqrmauttllloabj.com
12 dga,ranbyus,tqbmiuywoywolsfev.com
13 dga,matsnu,brothernerveplacebringconsult.com
14 legit,alexa,download-by-satoshi.com
15 dga,ranbyus,lrushteeoideiqbci.pw
16 dga,simda,rycoquqawyw.eu
17 legit,alexa,inthemiddlenashville.com
18 dga,fobber,cdjimfjspm.com
```

Рисунок 4.1 – Скріншот датасету.

Як мову програмування я обрав Python3, оскільки в нього є багато зручних бібліотек для машинного навчання та роботи з даними. Для початку роботи з проектами машинного навчання (ML) та глибокого навчання (DL) дуже важливо мати певні бібліотеки, API та середовища. Як середовище для розробки я обрав Jupyter Notebook - це веб-додаток, який дозволяє створювати та розповсюджувати документи, що містять живий код, рівняння, візуалізації та пояснювальний текст. Завдяки цьому він став незамінним інструментом в областях наукових досліджень, освіти, обробки даних та машинного навчання.

Jupyter Notebook підтримує десятки мов програмування (включаючи Python, R, Julia і багато інших), і допомагає виконувати код у реальному часі, створювати інтерактивні візуалізації та робити документацію, що зробило його дуже зручним інструментом для роботи з даними та машинним навчанням.

Наведу декілька ключових особливостей Jupyter Notebook:

1. Живий код: Код можна запустити в інтерактивній оболонці, і результати виконання коду відображаються безпосередньо у ноутбучі.
2. Візуалізація: Jupyter підтримує багато бібліотек для візуалізації даних, що дозволяє створювати графіки і діаграми

прямо в ноутбучі.

3. Markdown: Ноутбуки Jupyter підтримують Markdown, що означає, що ви можете включати форматований текст, зображення, гіперпосилання і навіть математичні рівняння у ваших ноутбуках.
4. Інтерактивність: Jupyter Notebook дозволяє створювати інтерактивні віджети, що робить його потужним інструментом для демонстрації динамічних концепцій або взаємодії з даними.

У розробці я використовував такі бібліотеки Python:

Pandas: надає потужні інструменти для обробки та аналізу даних, дозволяючи ефективно маніпулювати табличними даними та виконувати операції з ними, такі як фільтрація, групування, з'єднання та обчислення статистики.

Keras: є високорівневим інтерфейсом для побудови та навчання моделей глибокого навчання. Він надає зручний спосіб визначення архітектури моделі та її параметрів, а також упрощує процес навчання моделі шляхом автоматичного визначення градієнтів та оптимізації.

Sklearn (Scikit-learn): є популярною бібліотекою машинного навчання, яка надає широкий спектр алгоритмів та інструментів для класифікації, регресії, кластеризації, вибору моделі та оцінки її ефективності.

Re: дозволяє працювати з регулярними виразами для витягування даних. Буде використовуватись для очистки доменів, та розбивання їх на окермі частини.

Enchant: бібліотека, яка містить в собі словник англійської мови.

Pickle: модуль для збереження об'єктів Python для пізнішого використання без потреби в повторному виконанні коду, який створив ці об'єкти. Наприклад, можна створити модель машинного навчання, зберегти її за допомогою pickle, а потім завантажити модель з файла pickle пізніше, щоб робити прогнози без потреби повторно тренувати модель.

Googletrans: це безкоштовна бібліотека Python, яка використовує неофіційне API Google Translate. Вона підтримує переклад між більш ніж 100 мовами, включаючи автоматичне визначення мови вхідного тексту. Її я буду використовувати для переладу доменів перед їх класифікацією.

З використанням цих потужних бібліотек та середовища Jupyter Notebook я зможу ефективно працювати з даними, створювати та навчати моделі машинного навчання для розв'язання різних завдань.

4.2 Підготовка даних

Першим етапом розробки, звісно, буде підготовка вхідного датасету домашнього навчання.

1. Імпортуємо бібліотеки:

```
1 import numpy as np # бібліотека для роботи з масивами даних
2 import pandas as pd # бібліотека для роботи з даними у вигляді таблиць
3 import re # бібліотека для роботи з регулярними виразами (екстракція даних)
4 import enchant # словник англійської мови
Executed at 2023.05.23 19:31:50 in 57ms
```

Рисунок 4.2 – Імпорт бібліотек.

2. Зчитуємо добрі та погані домени (DGA/не DGA) за допомогою методу `read_csv` у Pandas. Він дозволяє користуватися даними з файлу з форматом

`.csv` у Python використовуючи його зручною структуру даних під назвою `DataFrame`. `DataFrame` - це двовимірна структура даних, що представляє собою таблицю з даними. Вона складається з рядків і стовпців, де кожен стовпець має своє ім'я (`label`), а кожен рядок має свій індекс.

```
1 dga_df = pd.read_csv('data/dga_domains_full.csv') # завантаження даних
2 dga_df
Executed at 2023.05.23 19:12:37 in 548ms
```

448318 rows × 3 columns `pd.DataFrame`

	type	origin	domain
0	dga	gozi	mortiscontrastatim.com
1	dga	corebot	cvyh1po636avyrsexebwbkn7.ddns.n...
2	legit	alexa	plasticbags.sa.com
3	legit	alexa	mzltrack.com
4	legit	alexa	miss-slim.ru
...
448313	dga	pykspa	sahejo.net
448314	dga	conficker	gdpjrbb.dk
448315	legit	alexa	masbadar.com
448316	legit	alexa	gracedigital.com
448317	legit	alexa	...

Рисунок 4.3 – Перші та останні рядки датасету.

3. Створюю функцію `split_domain`, ця функція приймає домен як вхідний аргумент і розбиває його на частини за допомогою регулярних виразів. Функція знаходить і видаляє топ-домен (наприклад, `.com`, `.net`), а потім повертає рядок, який складається з частин домену, розділених пробілами, тобто розбиває на слова.

```
1 def split_domain(domain):
2     # Розділити домен на частини за будь-якими знаками (., - тощо)
3     parts = re.split(r'^a-zA-Z0-9+', domain)
4
5     # Знайти індекс першої частини, яка має більше одного символу (можливо, це TLD)
6     tld_index = None
7     for i in range(len(parts) - 1, -1, -1):
8         if len(parts[i]) > 1:
9             tld_index = i
10            break
11
12     # Якщо така частина знайдена, видалити її
13     if tld_index is not None:
14         parts = parts[:tld_index]
15
16     # Повернути рядок, який складається з частин, розділених пробілами
17     return ' '.join(parts)
```

Executed at 2023.05.23 19:12:37 in 13ms

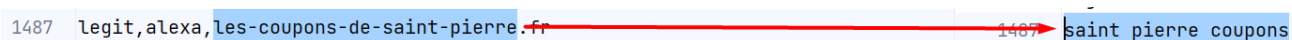
Рисунок 4.4 – Функція розділення домену на слова.

4. Створюється новий датафрейм, де домен розбивається на частини за допомогою функції `split_domain()`, створеної на попередньому кроці.

Новий датафрейм зберігається у CSV файлі за допомогою `df.to_csv()`. Це робиться для того, щоб мати можливість перекласти розбиті домени вручну за допомогою Google Translate.

Переклад на англійську мову виконується якраз для того, щоб вирішити одну з відомих проблем багатьох існуючих моделей для виявлення DGA – більшість моделей були натреновані на датасетах, що містять тільки англійські домени, як `facebook.com`, `gmail.com`, `putimes` та інших. Через це вони легко помиляються коли стикаються з іноземними доменами.

Ось приклад одного з перекладених доменів:



1487 legit, alexa, les-coupons-de-saint-pierre.fr → 1487 saint pierre coupons

Рисунок 4.5 – Функція розділення домену на слова.

```

1 df = pd.DataFrame(columns=['domain_eng']) # створити пустий датафрейм для майбутнього перекладу
   Executed at 2023.05.23 19:12:37 in 258ms

1 # розділити домени на слова
2 df['domain_eng'] = dga_df['domain'].apply(lambda x: split_domain(x))
   Executed at 2023.05.23 19:12:38 in 1s 154ms

1 df.to_csv('data/dga_domains_eng_text.csv', index=False) # зберегти датафрейм для перекладу
   Executed at 2023.05.23 19:12:39 in 518ms

```

Рисунок 4.6 – Підготовка та збереження датафрейму перед перекладом.

Переклад доменів я виконав мануально, використовуючи Google Translate File Translator, оскільки API Google Translate має обмеження на кількість запитів і тому перекласти через Python таку велику кількість доменів зайняло би багато часу.

5. Після того, як домени були перекладені вручну, завантажуються перекладений датафрейм, а переклад додається до оригінального датафрейму.

```

1 df_translated = pd.read_csv('data/dga_domains_eng_text_translated.csv') # завантажити перекладений датафрейм
   Executed at 2023.05.23 19:12:39 in 662ms

1 dga_df['domain_eng_text_translated'] = df_translated['domain_eng'] # додати переклад до оригінального датафрейму
   Executed at 2023.05.23 19:12:40 in 24ms

```

Рисунок 4.7 – Завантаження перекладених адрес.

6. З датафрейму видаляються стовпці, які більше не потрібні, за допомогою методу drop().

```

1 dga_df = dga_df.drop(columns=['origin', 'domain_eng_text']) # видалити непотрібні колонки
   Executed at 2023.05.23 19:12:40 in 176ms

```

Рисунок 4.8 – Видалення непотрібних колонок.

7. Далі я використовую бібліотеку wordninja для розбиття перекладеного домену на слова. Це потрібно для майбутньої перевірки домену на кількість англійських слів у ньому. Також витягуються топ-домени і розраховується довжина кожного домену.

```

1 # витягнути топ-домен (.com, .net тощо)
2 dga_df['top_domain'] = dga_df['domain'].apply(lambda x: x.split('.')[1])
3 # довжина домену
4 dga_df['domain_len'] = dga_df['domain'].apply(lambda x: len(x))
5 dga_df

```

Executed at 2023.05.23 19:13:27 in 297ms

Рисунок 4.9 – Отримання топ-доменів.

8. Далі я ініціалізую словник англійської мови - `enchant`, а потім кожна частинка (слово) в домені перевіряється на присутність в словнику і таквираховується к-сть англійських слів

```

1 d = enchant.Dict('en_US') # ініціалізація словника англійської мови
2 # додати колонку з кількістю англійських слів у домені
3 dga_df['eng_words'] = dga_df['domain_eng_text_translated']\
4     .apply(lambda x: sum([d.check(word) for word in x.split(' ') if word and len(word) > 3]))

```

Executed at 2023.05.23 19:17:58 in 37s 30ms

Рисунок 4.10 – Підрахунок кількості англійських слів у домені.

9. Тут ми проведемо однофакторний дисперсійний аналіз (ANOVA), щоб перевірити гіпотезу про те, чи є статистична відмінність між групами доменів за показником `length_word_ratio`. Ця змінна визначається як відношення довжини домену до кількості англійських слів у ньому.

Основна мета тесту ANOVA полягає у порівнянні середніх значень двох або більше груп. У нашому випадку ми порівнюємо домени двох типів: `dga` (шкідливі домени) та `legit` (легітимні домени). Це дає нам можливість визначити, чи мають ці дві групи статистично відмінні середні значення `length_word_ratio`.

```

1 # Проведемо тест ANOVA для перевірки гіпотези про те,
2 # що змінна length_word_ratio (довжина домену \ к-сть анг слів у ньому
3 # відрізняється для різних типів доменів (dga / legit)
4
5 df['length_word_ratio'] = df['domain_len'] / df['eng_words']
6 df['length_word_ratio'] = df['length_word_ratio'].replace(np.inf, 0)
7
8 # розділити домену на дві групи
9 dga_data = df[df['type'] == 'dga']
10 legit_data = df[df['type'] == 'legit']
11
12 # провести тест ANOVA
13 f_value, p_value = stats.f_oneway(dga_data['length_word_ratio'], legit_data['length_word_ratio'])
14
15 print("F-Value:", f_value)
16 print("P-Value:", p_value)

```

Executed at 2023.05.23 19:40:22 in 113ms

F-Value: 73792.95133750362
 P-Value: 0.0

Рисунок 4.11 – Результати тесту ANOVA.

Результати ANOVA тесту містять два значення: F-значення (F-Value) та P-значення (P-Value).

F-значення (F-Value): Велике F-значення, що в даному випадку становить приблизно 73793, свідчить про те, що середні значення співвідношення довжини домену до кількості англійських слів (length_word_ratio) для двох типів доменів (dga та legit) відмінні. Велике F-значення означає, що різниця між середніми значеннями є великою.

P-значення (P-Value): P-значення вказує на ймовірність отримати спостережувані або більш екстремальні результати, якщо нульова гіпотеза (тобто гіпотеза про відсутність різниці між групами) є істинною. У цьому випадку, P-значення дорівнює 0.0, що значно менше типового порогу (наприклад, 0.05). Це означає, що ми можемо відхилити нульову гіпотезу і прийняти альтернативну гіпотезу про те, що середні значення length_word_ratio для dga та legit доменів дійсно відрізняються.

Це пов'язано з тим, що dga домену рідко коли використовують реальні слова, зазвичай складаючи адреси з псевдорандомно вибраних букв. Або ж коли вже використовують їх, то такі dga домену виходять статистично довшими за реальні.

Отже, висновок з цього тесту полягає в тому, що тип домену впливає на співвідношення довжини до домену до кількості слів (length_word_ratio), і цю змінну можна використовувати для класифікації доменів.

Нижче наведена візуалізація різниці між двома типами доменів:

```
1 mean_ratio = df.groupby('type')['length_word_ratio'].mean()
2
3 plt.figure(figsize=(8, 6))
4 plt.bar(mean_ratio.index, mean_ratio.values)
5 plt.xlabel('Domain Type')
6 plt.ylabel('Mean Length / Word Count Ratio')
7 plt.title('Mean Ratio of Domain Length to Word Count by Type')
8 plt.show()
```

```
10 # Візуалізація результатів тесту ANOVA
```

Executed at 2023.05.23 19:46:25 in 137ms

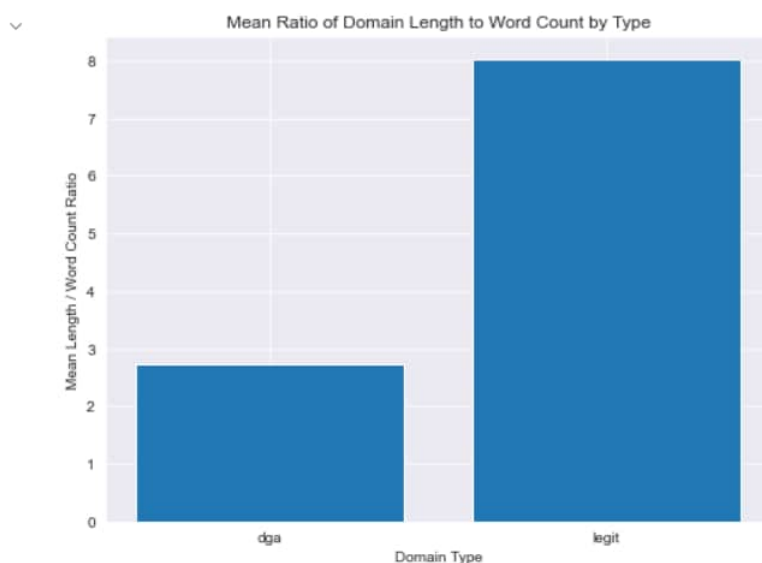


Рисунок 4.12 – Візуалізація результатів тесту ANOVA.

10. Враховуючи результати тесту, я вирішив використати такі класифікатори для створення моделі розпізнавання DGA:

'domain', 'top_domain', 'domain_len', 'eng_words', 'length_word_ratio',

```
1 # зберігаємо необхідні колонки в фінальний датафрейм
2 df[['domain', 'top_domain', 'domain_len', 'eng_words', 'length_word_ratio', 'type']]
3 .to_csv('data/dga_domains_final.csv', index=False)
```

Executed at 2023.05.23 19:50:52 in 867ms

Рисунок 4.11 – Збереження датафрейму.

Нижче наведено діаграму процесу створення усіх класифікаторів використаних для тренування моделі.

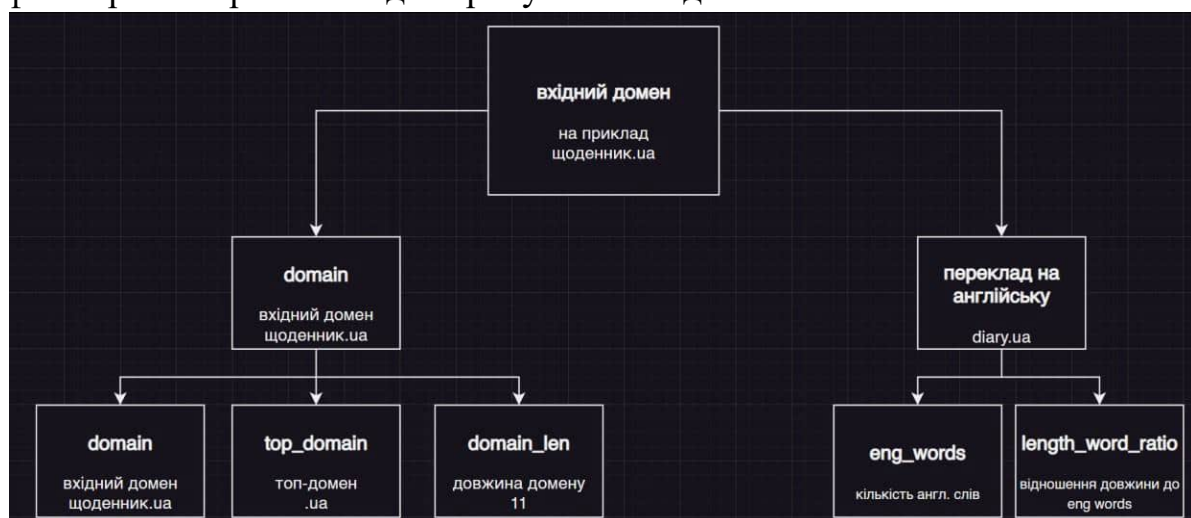


Рисунок 4.12 – Діаграма класифікаторів.

4.3 Тренування моделі

Тепер можна переходити до тренування моделі класифікації доменів:

1. Спочатку знову імпортуємо потрібні бібліотеки та зчитуємо раніше збережений датасет. Для машинного навчання ми будемо використовувати бібліотеку `sklearn`. Бібліотека `sklearn`, відома також як `Scikit-learn`, є однією з найпотужніших та найчастіше використовуваних бібліотек для машинного навчання в `Python`.

Ця бібліотека підтримує широкий спектр алгоритмів для задач класифікації, регресії, кластеризації та зниження розмірності, які покривають більшість потреб у машинному навчанні. До основних алгоритмів входять методи, засновані на ансамблях, такі як випадковий ліс (`Random Forest`), бустинг (`Gradient Boosting`), а також лінійні моделі, метод опорних векторів (`SVM`), `K`-ближніх сусідів (`KNN`) та багато інших.

Ще однією ключовою особливістю `sklearn` є набори інструментів для передопрацювання даних, включаючи кодування категорій, нормалізацію, масштабування та вибір ознак. Також вона пропонує можливості для оцінки моделі та валідації, такі як перехресна перевірка, розрахунок метрик точності, повернення та `F`-мір.

Одним з великих переваг `sklearn` є її консистентний API. Це

означає, що після того, як ви зрозуміли, як працює один алгоритм, вам буде легше освоїти інші, оскільки вони всі дотримуються однакових методів ініціалізації та використання.

За допомогою `sklearn`, ви зможете створювати потужні моделі машинного навчання, які здатні працювати з різноманітними даними і вирішувати різні проблеми.

```
1 import pandas as pd # для роботи з даними
2 # sklearn - для машинного навчання
3 from sklearn.model_selection import train_test_split
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.preprocessing import LabelEncoder
6 from sklearn.metrics import classification_report
   Executed at 2023.05.25 22:40:35 in 13ms

1 df = pd.read_csv('data/dga_domains_final_3.csv')
   Executed at 2023.05.25 22:40:35 in 407ms
```

Рисунок 4.13 – Імпорт бібліотек та датафрейму.

2. За допомогою `LabelEncoder` перетворюємо текстові змінні в числові, оскільки більшість алгоритмів машинного навчання працюють краще або виключно з числовими даними.

```
1 le = LabelEncoder() # для перетворення текстових змінних в числові
2 # перетворюємо текстові змінні в числові
3 df['top_domain'] = le.fit_transform(df['top_domain'])
4 df['type'] = le.fit_transform(df['type'])
5 df['domain'] = le.fit_transform(df['domain'])
   Executed at 2023.05.25 22:40:36 in 1s 277ms
```

Рисунок 4.14 – Перетворення текстових змінних на числові.

3. Розділяємо датасет на два нових – X та y .

Датасет X містить ознаки (характеристики), на основі яких модель буде здійснювати прогнози.

y , з іншого боку, містить мітки, які ми будемо прогнозувати. У нашому випадку це тип домену (`dga` або `legit`). Це є цільовою змінною, яку модель машинного навчання намагається передбачити або класифікувати на основі ознак в X .

Далі кожен датасет ділимо на навчальний (training) та тестовий (testing). Ідея полягає в тому, щоб навчити модель на одній частині даних (навчальний набір), а потім перевірити, наскільки добре модель працює наданих, які вона раніше не бачила (тестовий набір).

```
1 X = df.drop(['type'], axis=1) # Features
2 y = df['type'] # Labels
   Executed at 2023.05.25 22:40:36 in 7ms

1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% тренувальних даних, 30% тестових
   Executed at 2023.05.25 22:40:36 in 59ms
```

Рисунок 4.14 – Поді даних для тренування.

4. Далі ініціалізуємо класифікатор `RandomForestClassifier` - це класифікатор, що базується на алгоритмі випадкового лісу, який є типом ансамблевого машинного навчання.

Методика випадкового лісу базується на генеруванні великої кількості дерев рішень та використання методу голосування більшості для визначення остаточного прогнозу. Під час цього процесу кожне дерево в ансамблі надає свій власний прогноз, і потім класифікатор вибирає ту категорію, яка отримала найбільшу кількість голосів з усіх дерев.

`RandomForest` є одним з найважливіших алгоритмів машинного навчання, який є виключно популярним для схожих задач класифікації. Саме тому я обрав його для цього конкретного завдання, оскільки проблема, що ми маємо вирішити, має основну мету класифікації і велику кількість ознак з нелінійними взаємозв'язками.

Один із параметрів, який ми задаємо в `RandomForestClassifier`, це `n_estimators`. Це число показує кількість дерев, що будуть сформовані в "лісі". Цей параметр дуже важливий, оскільки він вказує на кількість дерев рішень, що будуть включені до ансамблю. У даному випадку я зупинив свій вибір на створенні 100 дерев, що вважається відносно оптимальною кількістю для великої частини задач.

Тим не менш, слід зазначити, що при збільшенні числа дерев у лісу можна досягти більшої точності моделі. Однак це також призведе до збільшення обчислювальної складності і, відповідно, часу на виконання процесу. Тому завжди важливо збалансувати ці дві змінні і визначити оптимальне значення для вашого конкретного сценарію. Отож, ініціалізуємо класифікатор зі 100 деревами та починаємо навчання.

```
1 # ініціалізуємо класифікатор зі 100 деревами
2 clf = RandomForestClassifier(n_estimators=100)
```

Executed at 2023.05.25 22:40:36 in 13ms

```
1 # навчаємо модель
2 clf.fit(X_train, y_train)
```

Executed at 2023.05.25 22:41:22 in 45s 506ms

```
▼ RandomForestClassifier
RandomForestClassifier()
```

Рисунок 4.14 – Навчання моделі.

5. Після завершення тренування, тестуємо модель.

```
1 # робимо прогноз на тестових даних
2 y_pred = clf.predict(X_test)
```

Executed at 2023.05.25 22:41:26 in 3s 694ms

```
1 print(classification_report(y_test, y_pred))
```

Executed at 2023.05.25 22:41:26 in 214ms

```
▼
              precision    recall  f1-score   support

0               0.89         0.90         0.89         67482
1               0.89         0.89         0.89         67014

 accuracy                   0.89         134496
 macro avg              0.89         0.89         0.89         134496
 weighted avg           0.89         0.89         0.89         134496
```

Рисунок 4.14 – Результати моделі.

Наведена вище таблиця представляє результати оцінки моделі машинного навчання. У ній показано такі основні метрики:

- Precision (точність) - це доля правильно передбачених позитивних результатів серед усіх передбачених позитивних результатів. У цьому випадку, для класу 0 (dga) та класу 1 (legit) точність становить 0.89 - це означає, що при передбаченні обох класів модель була правильною приблизно 89% часу.

- Recall (повнота) - це доля вірно передбачених позитивних результатів серед усіх дійсно позитивних результатів. У цьому випадку, модель дала практично однаковий результат для обох типів доменів. Це означає, що датасет містив добре розподілені дані.

- F1-score - це середнє гармонічне значення точності та повноти. Воно бере до уваги обидва параметри і є корисним, коли ви хочете мати єдину метрику для оцінки моделі. Для обох класів F1-score становить 0.89, що свідчить про високу точність та повноту моделі.

- Accuracy (точність) - це загальна доля правильно передбачених випадків.

У даному випадку, точність становить 0.89, що означає, що модель правильно передбачає результати приблизно в 89% випадків.

На виході, модель RandomForest показує досить високий рівень точності та повноти, з усіма основними метриками, включаючи точність, повноту та F1-score, що варіюється від 0.89 до 0.90. Це означає, що модель добре впоралася з визначенням обох класів в наборі даних.

4.4 Тестування моделі на нових даних

Тепер щоб точно могли стверджувати, що модель є точною, потрібно потестувати її на нових даних:

1. Знову імпортуємо потрібні бібліотеки, ініціалізуємо їх, та зчитуємо збережену модель та LabelEncoder.

```

1 import pickle # для збереження моделі
2 import enchant # для перевірки чи слово англійське
3 import regex # для розбиття домену на слова
4 import wordninja # для розбиття англійських слів на слова
5 import re # для розбиття домену на частини
6 from googletrans import Translator # для перекладу домену
Executed at 2023.05.27 13:46:11 in 8ms

1 translator = Translator() # ініціалізуємо перекладач
2 d = enchant.Dict('en_US') # ініціалізуємо словник
3 # зчитуємо збережений у файл лейбл-енкодер
4 le = pickle.load(open('le.sav', 'rb'))
5 # зчитуємо збережену у файл модель
6 loaded_model = pickle.load(open('clf.sav', 'rb'))
Executed at 2023.05.27 13:46:11 in 569ms

```

Рисунок 4.15 – Імпорт бібліотек та моделі.

- Також я створив декілька функцій, а саме: `split_domain`, `prepare_domain` та `predict`, які допомагають оптимізувати процес перевірки та аналізу доменів. Ці функції створені з метою забезпечення ефективної та ретельної підготовки даних для подальшої обробки.

Функція `prepare_domain` слугує основним інструментом для підготовки даних. Її призначення - створити та обчислити набір важливих характеристик, таких як назва домену, топ-домен, довжина домену, кількість англійських слів в домені, а також співвідношення довжини домену до кількості англійських слів в ньому. Всі ці змінні відіграють ключову роль в нашому аналізі та прогнозуванні. Для забезпечення сумісності з моделлю машинного навчання, `prepare_domain` також здатна перетворювати текстові дані на числові.

Наступна функція, `split_domain`, призначена для більш специфічного аналізу. Її головна роль полягає в розбиванні домену на окремі складові. Зокрема, вона ідентифікує та видаляє топ-домен (наприклад, ".com", ".net") з повного імені домену. Залишок домену вона розділяє на окремі слова, розділені пробілами, що дозволяє нам аналізувати домен, як речення, що складається з окремих слів.

Остання функція, `predict`, є стартовою частиною робочого процесу інших функцій. Вона призначена для взаємодії з нашою моделлю машинного навчання. Функція `predict` приймає оброблені дані від функцій `prepare_domain` та `split_domain` і повертає прогнозований тип домену - "DGA" (домен, згенерований алгоритмом) або "legit" (легітимний домен). Завдяки цій функції, ми можемо визначити, чи є даний домен згенерованим алгоритмом

DGA, чи він легітимний.

У сукупності, ці три функції дозволяють нам глибше зануритися в аналіз доменів, розбиваючи їх на складові та обробляючи ключові характеристики. Вони допомагають нам працювати ефективніше, забезпечуючи точне та ретельне аналізування даних для подальшого використання в моделі машинного навчання.

```
1 def split_domain(domain):
2     # Розділити домен на частини за будь-якими знаками (., - тощо)
3     parts = re.split(r'\W', domain)
4
5     # Знайти індекс першої частини, яка має більше одного символу (можливо, це TLD)
6     tld_index = None
7     for i in range(len(parts) - 1, -1, -1):
8         if len(parts[i]) > 1:
9             tld_index = i
10            break
11
12     # Якщо така частина знайдена, видалити її
13     if tld_index is not None:
14         parts = parts[:tld_index]
15
16     # Повернути рядок, який складається з частин, розділених пробілами
17     return ' '.join(parts)
```

Executed at 2023.05.28 22:29:26 in 14ms

```
1 def prepare_domain(domain):
2     # перекладаємо домен на англійську
3     domain_eng = translator.translate(split_domain(domain)).text
4     # розбиваємо перекладений домен на слова
5     domain_eng_text_translated = ' '.join(wordninja.split(domain_eng))
6     top_domain = domain.split('.')[ -1 ] # топ-домен
7     domain_len = len(domain) # довжина домену
8     domain = ' '.join(domain.split('.')[:-1]) # домен
9     # кількість англійських слів в перекладеному домені
10    eng_words = sum([d.check(word) for word in domain_eng_text_translated.split(' ') if word and len(word) > 3])
11    eng_words = eng_words if eng_words > 0 else 1 # щоб не ділити на 0
12    length_word_ratio = domain_len / eng_words # відношення довжини домену до кількості англійських слів
13    return le.fit_transform([domain])[0], le.fit_transform([top_domain])[0], domain_len, eng_words, length_word_ratio,
    domain_eng_text_translated
```

Executed at 2023.05.28 22:29:26 in 20ms

```
1 def predict(domain):
2     result = loaded_model.predict([domain])
3     return 'legit' if result == 1 else 'dga'
```

Executed at 2023.05.28 22:29:26 in 22ms

Рисунок 4.14 – Ініціалізація необхідних функцій.

3. Я обрав п'ять доменів для тестування натренованої моделі:

Назва - Тип

щоденник.ua - legit

сайт-львівського-університету.ua - legit

英语学校.tw - legit

xn--lhrz38b.xn – dga foronpcwtr.com – dga

```
1 for domain, label in new_domains.items():
2     prepared_domain = prepare_domain(domain)
3     print(f'{domain} is {label}')
4     print(f'Translated to english: {prepared_domain[-1]}')
5     print(f'predicted label: {predict(prepared_domain[:-1])}')
6     print('-----')
```

Executed at 2023.05.29 21:07:43 in 2s 637ms

```
✓ щоденник.ua is legit
  Translated to english: diary
  predicted label: legit
  -----
  сайт-львівського-університету.ua is legit
  Translated to english: The site of Lviv University
  predicted label: legit
  -----
  英语学校.tw is legit
  Translated to english: English school
  predicted label: legit
  -----
  xn--lhrz38b.xn is dga
  Translated to english: xn lh rz 38 b
  predicted label: dga
  -----
  foronpcwtr.com is dga
  Translated to english: for on pc w tr
  predicted label: dga
  -----
```

Рисунок 4.15 – Результати моделі.

Як видно з результатів, модель правильно оприділила усі 5 адрес. Варто зазначити, що велику роль у чіткості даної моделі зіграв етап з

перекладанням доменних імен.

На приклад, адреса *щоденник.ua* дуже легко могла б здаватися підозрілою системі, що не перекладає доменні імена, або ж не була на них натренована. У логах видно, що система переклала слово щоденник як *diary* англійською, що є правильним перекладом. Це збільшило параметр *eng_words* (кількість англійських слів) та зменшило *length_word_ratio* (відношення довжини домену до кількості англійських слів), що дозволило моделі правильно передбачити тип адреси, адже *dga* домени рідко коли використовують реальні слова. Те саме сталося з наступними двома доменами: сайт-львівського-університету.ua та 英语学校.tw (перекладається з мандарину як “англійська школа”). Модель правильно оприділила їх тип за допомогою перекладу. Система також добре справилася з наступними *dga* доменами: *xn--lhrz38b.xn* та *foronpcwtr.com*. У цих іменах не було виявлено англійських слів, що знизило “рейтинг” доменів.

5. ВИСНОВОК

Використання DGA, або алгоритму генерації доменних імен, є серйозною загрозою в плані кібербезпеки, оскільки воно ускладнює виявлення та блокування C&C-сервера - головного органу управління ботнетами. C&C-сервери надають керівні команди, що використовуються для контролю за інфікованими системами. Традиційні методи забезпечення кібербезпеки, такі як створення чорного списку небезпечних доменних імен, стають менш ефективними, оскільки DGA регулярно генерує нові, непередбачувані доменні імена.

Це створює значні труднощі у процесі виявлення і блокування зловмисного трафіку. Хакери можуть миттєво змінювати доменні імена, що робить неможливим їх блокування до моменту отримання нових зразків шкідливого ПЗ та аналізу їх поведінки. Аналіз мережевого трафіку стає важливим інструментом для виявлення атак, які використовують DGA. Використовуючи спеціалізовані алгоритми та системи виявлення інтранет-атак, можна виявити підозрілий трафік, що пов'язаний з доменними іменами, згенерованими за допомогою DGA. Однак, ці системи вимагають неперервного оновлення і вдосконалення для ефективного виявлення нових варіантів DGA та адаптації до постійно змінюючихся технік зловмисників.

Враховуючи все це, організації повинні приділяти серйозну увагу заходам кібербезпеки, які включають встановлення міцних захисних систем, акцент на виявленні аномального мережевого трафіку та постійне оновлення методів боротьби з цією загрозою. Проблема розпізнавання DGA полягає в тому, що імена, що генеруються цим методом, є непередбачуваними. Через це розробники довгий час були вимушені використовувати менш ефективні методи, такі як використання чорного списку, зворотну інженерію, або ж витратити значний час на статистичний аналіз імен для кожного випадку. А саме у той час, коли розробники витрачають свій час на ці методи, зловмисники уже могли змінити зерно свого алгоритму, унеможлививши всі їхні попередні зусилля.

Відповідно до цього, ми розробили нову систему класифікації доменів за допомогою машинного навчання, фокусуючись на покращенні розпізнавання та класифікації іноземних адрес. Незважаючи на відсутність якісних багатомовних наборів даних у відкритому доступі, ми знайшли альтернативний шлях вирішення проблеми за допомогою сервісів машинного перекладу, таких як Google Translate. Використовуючи доступні дані з інтернету, Python-бібліотеки та Jupyter Notebook, ми змогли

створити ефективну систему для виявлення DGA доменів. Незважаючи на обмежений вхідний датасет, навіть така невелика модель має високу точність та здатна правильно класифікувати усі домени, що були подані для тестування.

Було виявлено, що машинне навчання є одним з небагатьох ефективних способів боротьби з постійно мінливим потоком доменів, створених з допомогою DGA. Машинне навчання дозволяє створювати гнучкі рішення та швидко адаптуватись до нових викликів. Хороша модель машинного навчання здатна не лише швидко розпізнавати домени, створені за допомогою DGA, але й є універсальною, тобто її важко обійти. За умови постійного оновлення моделі машинного навчання та додавання нових прикладів для її тренування, можна створити ефективний засіб боротьби проти кіберзлочинності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Juhong Namgung, Siwoon Son and Yang-Sae Moon. "Efficient Deep Learning Models for DGA Domain Detection" [Електронний ресурс]. – Режим доступу: <https://www.hindawi.com/journals/scn/2021/8887881/> (Дата звернення: 01.06.2023)
2. J Sreedevi, Ponnampooja. "DGA MALWARE DETECTION USING MACHINE LEARNING" [Електронний ресурс]. – Режим доступу: <http://ijream.org/papers/IJREAMV07I0375018.pdf> (Дата звернення: 01.06.2023)
3. С. М. ЛИСЕНКО, В. І. КОМАРОВ. "МЕТОД ТА ЗАСОБИ ІДЕНТИФІКАЦІЇ БОТ-МЕРЕЖ, ЩО ВИКОРИСТОВУЮТЬ ТЕХНОЛОГІЮ «ПОТІК ДОМЕНІВ»" [Електронний ресурс]. – Режим доступу: <http://journals.khnu.km.ua/vestnik/?p=1401> (Дата звернення: 2020)
4. "Getting started with the Keras Sequential model" [Електронний ресурс]. – Режим доступу: <https://faroit.com/keras-docs/1.0.1/getting-started/sequential-model-guide/> (Дата звернення: 01.06.2023)
5. Dataset. [Електронний ресурс]. – Режим доступу: https://github.com/chrmor/DGA_domains_dataset (Дата звернення: 01.06.2023)
6. Wikipedia. [Електронний ресурс]. – Режим доступу: <https://www.wikipedia.org/> (Дата звернення: 01.06.2023)

