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

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.738.5:005.75

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний застосунок виявлення фішингових листів з використанням штучного інтелекту

Виконавець: Дмитро МАРИНИЧ
Керівник: к.т.н., доцент Андрій ПЕТРЕНКО
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна *ДМИТРУХА*
Нормоконтролер: к.т.н., доцент Андрій ПЕТРЕНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

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

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Маринича Дмитра Руслановича

1. Тема: *Програмний застосунок виявлення фішингових листів з використанням штучного інтелекту*

затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.

2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: Аналіз сучасних методів атак на автоматизовану систему з використанням прийомів соціальної інженерії; на основі аналізу вивести кінцевий набір даних для навчання нейронної мережі; розробка моделі нейронної мережі з навчанням без вчителя.
4. Зміст пояснювальної записки: Загальний огляд соціальної інженерії, як найвідомішого впливу на інформаційну мережу і аналіз найпоширеніших атак; проведення детального огляду бібліотек Python для їх подальшого використання; створення програмного модулю для сканування фішингових листів.

5. КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	20.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	24.11.2023	*Виконано*
4.	Збір інформації	26.11.2023	*Виконано*
5.	Аналіз найпоширеніших атак, методики зловмисників	10.11.2023	*Виконано*
6.	Дослідження бібліотек Python для роботи зі штучним інтелектом, їх подальше намагання інтеграції в проект	16.11.2023	*Виконано*
7.	Розробка навчального датасету, навчання нейронної мережі, створення віконного додатку	22.11.2023	*Виконано*
8.	Тестування створеного ПЗ	24.11.2023	*Виконано*
9.	Перевірка на антиплагіат	12.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	14.12.2023	*Виконано*
11.	Оформлення презентації	15.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Дмитро МАРИНИЧ
(підпис, дата)

Керівник кваліфікаційної роботи Андрій ПЕТРЕНКО
(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, п’яти розділів, загальних висновків, списку використаних джерел, додатків і має 87 сторінок основного тексту, 11 рисунків, 17 таблиць, 8 сторінок додатків. Список використаних джерел містить 39 найменувань і займає 4 сторінки. Загальний обсяг роботи 107 сторінок.

Метою роботи є підвищення рівня захищеності ІМ за рахунок аналізу атак за допомогою фішингу, навести перелік найвідоміших інцидентів, визначити всі необхідні аспекти соціальної інженерії.

В роботі вирішено задачу розробки чітких методики та алгоритми для виявлення підозрілих електронних листів, які можуть містити елементи соціальної інженерії, розробки та виконання серію експериментів з реальними або симульованими наборами електронних листів..

В роботі розроблено програмне забезпечення для автоматичного сканування листів Gmail з метою виявлення прикладів соціальної інженерії в тексті повідомлень.

Розроблене програмне забезпечення відносяться до галузі інформаційної безпеки і може бути використані для захищеності та свідомості користувачів у корпоративній мережі.

Можливі напрямки розвитку цієї роботи пов’язані із розширенням моделі і алгоритму програмного забезпечення відповідно до вимог міжнародних стандартів, наприклад ISO 27001, для більш повного аналізу та оцінки ризиків.

Ключові слова: аналіз, соціальна інженерія, штучний інтелект, датасет, атаки маніпулювання, фішинг, машинне навчання.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

- API - Інтерфейс програмування застосунків (Application Programming Interface);
- OAuth - Протокол авторизації (Open Authorization);
- АС – автоматизована система;
- GUI - Графічний інтерфейс користувача (Graphical User Interface);
- IDE - Інтегроване середовище розробки (Integrated Development Environment);
- ІБ – інформаційна безпека;
- ШІ – штучний інтелект;
- ІМ – інформаційна мережа.

ВСТУП

Актуальність. Сучасний світ переживає епоху, де інформація є безцінною валютою. Зростання залежності від інформаційних технологій та інтернету відкриває перед нами безмежні можливості, але разом із цим виникають нові загрози для нашої кібербезпеки. Однією з найбільш складних та хитрих загроз є соціальна інженерія, яка вимагає від нас розуміння не тільки технічних аспектів, але і глибокого знання людської психології. Ця проблема є надзвичайно актуальною в сучасному цифровому суспільстві і вимагає серйозного дослідження та заходів захисту.

Цей метод атаки ґрунтується на психологічних механізмах та маніпуляціях, які зловмисники використовують для досягнення своїх цілей. Сучасні технології дозволяють зловмисникам використовувати соціальну інженерію на різних рівнях - від атак на окремих користувачів до масштабних атак на корпоративні системи та урядові установи.

Актуальність проблеми соціальної інженерії обумовлена не лише зростанням кількості інтернет-користувачів, але і розширенням можливостей зловмисників у сфері кібератак. Зловмисники стають все більш винахідливими та вдосконалюють свої техніки. Вони використовують соціальну інженерію для впливу на людей та отримання доступу до конфіденційної інформації, фінансових ресурсів та багатьох інших цінних активів. Ця проблема має серйозний вплив на бізнес, уряд та окремих громадян.

Метою роботи є розробка програмного забезпечення для автоматичного сканування листів Gmail з метою виявлення атак з використанням соціальної інженерії в тексті повідомлень.

Виходячи з мети **завданням** для даної дипломної роботи є:

1. На основі дослідів реальних випадків розробити чіткі методики та алгоритми задля забезпечення виявлення підозрілих електронних листи, які можуть містити елементи соціальної інженерії.

2. Розробити програмне забезпечення з використанням бібліотек Python з машинним навчанням для аналізу текстової інформації в електронних листах, що дозволить розрізняти безпечні повідомлення від тих, що містять можливі елементи соціальної інженерії.

3. Провести серію експериментів з реальними або симульованими наборами електронних листів для перевірки доцільності розробленого програмного забезпечення виявлення атак соціальної інженерії.

Об'єктом дослідження є електронні листи в системі Gmail, що надходять на електронну пошту користувача.

Предметом дослідження є методи та технічні засоби, які можна використовувати для автоматичного виявлення можливих прикладів соціальної інженерії в тексті листів, спрямованих на користувача Gmail.

Методами дослідження дипломної роботи є:

1. Проведення статистичного аналізу реальних випадків атак соціальної інженерії на користувачів Gmail.

2. Дослідження та аналіз історії атак на користувачів через відкриті джерела.

3. Реалізація використання методів обробки природної мови для аналізу текстової інформації листів у Gmail з метою виявлення ключових слів.

Наукова новизна: розроблено спосіб до виявлення соціально-інженерних атак у електронних листах, поєднуючи методи обробки природної мови з машинним навчанням, що дозволить розпізнавати підозрілі патерни та фраз.

Практична цінність: розроблено програмний засіб, що дозволить користувачам проводити автоматичний аналіз поштових листів , що виявлятиме фішингові листи, які є прикладом атак з використанням соціальної інженерії, що дозволить забезпечити кібербезпеку.

РОЗДІЛ 1. ВСТУП ДО СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

1.1. Поняття та сутність.

Соціальна інженерія в сфері кібербезпеки представляє собою спосіб використання психологічних маніпуляцій та соціального впливу для отримання доступу до конфіденційної інформації, систем або мереж. Це складний арсенал технік, що базується на взаємодії з людьми з метою втягнення їх у дії, які допоможуть зловмисникам отримати доступ до цінної інформації або ресурсів.

Ця техніка атаки не ґрунтується на технічних уразливостях, але використовує вразливості в людській психології, які зазвичай важко виявити і легко використовувати. Її успішність полягає у вмінні маніпулювати довірою, бажанням допомогти, або створенні ситуацій, де люди ненавмисно розкривають конфіденційну інформацію атакуючим.

Ціль соціальної інженерії - не просто отримати доступ до інформації, але й зробити це максимально непомітно, використовуючи психологічні трюки, що дозволяють обхід захисту системи чи отримання доступу до цільових ресурсів[5].

Однією з ключових характеристик соціальної інженерії є необхідність співпраці людей, котрі використовуються як вектори атаки, щоб вони здійснювали певні дії або надавали доступ до конфіденційних даних чи систем. Це може бути використано для впливу на різні соціальні ролі - від робітників зі служби підтримки клієнтів до керівників підприємств.

Успішність соціальної інженерії в значній мірі залежить від здатності атакуючого розуміти психологічні аспекти поведінки людей, їхніх вподобань, страхів та мотивів. Отже, це вимагає не лише технічного розуміння, але й вміння ефективно спілкуватися та маніпулювати. Сутність соціальної інженерії полягає

в тому, що вона використовує людський фактор, що часто є слабким місцем у багатьох системах безпеки.

Інциденти кібербезпеки частіше викликані людськими збоями, ніж технологіями. Отже, люди є найслабшою ланкою в інформаційній безпеці. Відверто кажучи: «тільки аматори нападають на машини, професіонали атакують людей». Зловмисники використовують обман і маніпуляції, щоб змусити цільові особи сприяти їхній віктимізації[5]. Вектор атаки соціальної інженерії вважається найбільшою загрозою для інформаційних систем.

В наведеному нижче графіку ми можемо бачити наскільки високу долю з усіх атак займає саме соціальна інженерія (рис. 1.1)

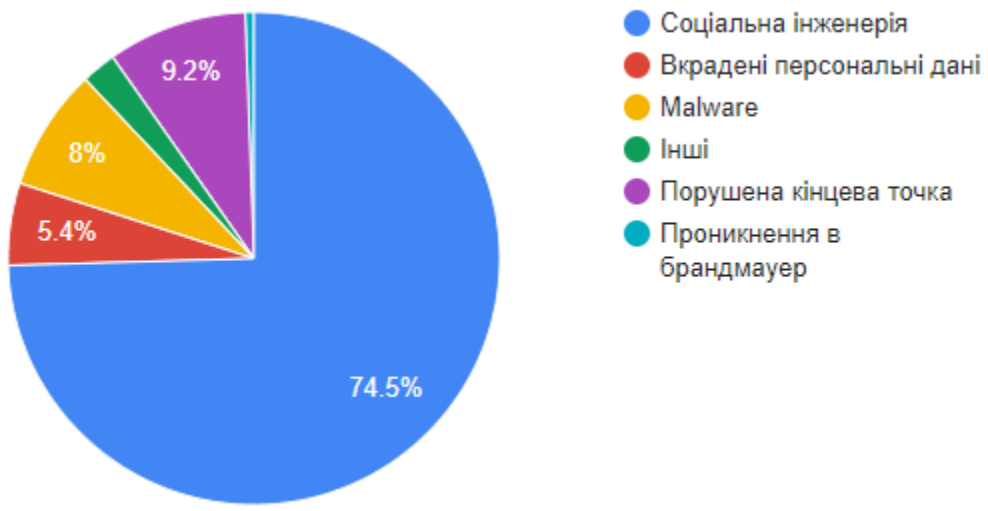

Рис.1.1. Найчастіші атаки на АС та їх співвідношення

1.2. Історія розвитку соціальної інженерії.

Соціальна інженерія не є новим явищем. Поняття "інженер соціального життя" з'явилося ще в 19 столітті і вказувало на спроби впливу на соціальні процеси та міжособистісні відносини. Проте в сучасному контексті, з поширенням інформаційних технологій та інтернету, соціальна інженерія набула нового значення[8].

Історія соціальної інженерії має корені в соціальних науках і психології, але її застосування в сучасному цифровому світі виходить далеко за рамки традиційних методів впливу на людей.

19 століття. Зародження поняття "інженер соціального життя"[7]:

- У 19 столітті в США і Європі з'явилося поняття "інженер соціального життя", що вказувало на спроби науковців і реформаторів вплинути на соціальні процеси та міжособистісні відносини.
- Це поняття визначало спроби систематичного змінення соціальних структур та змін у поведінці груп і суспільства.

20 століття. Використання соціальної інженерії в шпигунстві та розвідці:

- Під час Першої та Другої світових війн соціальна інженерія була широко використовуваною методикою для впливу на ворогів і отримання розвідувальної інформації.
- Розвідувальні служби та агенти використовували методи маніпуляції та обману для отримання конфіденційних даних та секретної інформації.

21 століття. Зростання атак соціальної інженерії в інтернеті:

- З поширенням Інтернету та цифрової технології соціальна інженерія стала однією з найважливіших загроз для кібербезпеки.
- Зловмисники використовують соціальну інженерію для отримання доступу до облікових записів, фінансових даних, атак на корпоративні мережі, розвідки та шпигунства.
- Зростання популярності соціальних мереж призвело до доступності великої кількості особистої інформації про людей, що робить соціальну інженерію ще більш ефективною.

Завдяки інтернету та соціальним мережам, зловмисники мають безпрецедентний доступ до великої кількості особистої інформації про людей, що дозволяє їм створювати реалістичні сценарії атаки. Внаслідок цього соціальна інженерія стала популярною технікою для викрадення паролів,

фінансових даних, атак на корпоративні мережі, а також для розвідки та шпигунства.

Соціальна інженерія вже досить давно використовується як вектор атаки. Нижче наведено короткий огляд шести горезвісних випадків соціальної інженерії з давньої історії:

1. Троянська війна. Можливо, одна з найдавніших розповідей про соціальну інженерію належить до Одіссея про давньогрецьку армію. Після довгого десятиліття війни з троянцями грецька армія відступила з арени і залишила троянцям колосальну дерев'яну статую коня. Троянці вітали свій подарунок у своєму місті і святкували перемогу. Вночі з коня з'явилися грецькі воїни, які завоювали місто. Міф чи ні, але цей приклад ілюструє, як людський елемент обманом змушений досягти мети, яка була неможливою лише технічними засобами.

2. Френк Абаньейл. Історія колишнього шахрая Френка Абаньейла неодноразово розповідалася в книгах, мюзиклах і фільмі «Спіймай мене, якщо зможеш». Після розлучення батьків підліток Френк втік з дому і залишився один у світі. Те, що починалося як творчий спосіб виживання, перетворилося на гру. Коли йому було всього 16 років, Френк встиг позувати пілотом авіакомпанії Pan Am. Пізніше він прикидався педіатром-ординатором, адвокатом і нібито асистентом. Френк оволодів навичками підробки чеків і, обманюючи банки у віці від 16 до 21 року, заробив 2,5 мільйона доларів. Зрештою, однак, його спіймали та посадили у в’язницю. Після свого звільнення він тісно співпрацював з ФБР майже 40 років.

3. Стенлі Марк Ріфкін. Стенлі Марк Ріфкін працював комп’ютерним консультантом у Security Pacific Bank у Лос-Анджелесі в 1978 році. Він працював над розробкою системи резервного копіювання для телефонної кімнати, яка дала йому доступ до процедур, як банк переказує гроші. Для здійснення переказу потрібен був секретний код, що змінювався щодня. Стенлі дізнався, що клерки записали код на липкій записці, щоб позбавити їх від проблем із його запам’ятовуванням. Дізнавшись цей код, Стенлі вдалося перевести 10,2 мільйона

доларів на офшорний рахунок у Швейцарії. Він здійснив найбільше пограбування банку в історії, не застосовуючи жодного насильства чи комп’ютера.

4. ABN-AMRO. У світовій діамантовій столиці Антверпені (Бельгія) викрали алмазів на 21,5 мільйона євро. Без застосування насильства зловмиснику вдалося здійснити це пограбування. Його зброя? Правопорушник, постійний клієнт, познайомився з працівниками. Використовуючи чарівність і купуючи шоколадні цукерки для персоналу, йому вдалося скопіювати ключі від сховища.

5. Кевін Мітнік. Кевін Мітнік відомий як найвідоміший у світі хакер і як людина, яка популяризувала термін «соціальна інженерія». В одному зі своїх ранніх розповідей він створив водія автобуса за допомогою соціальної інженерії, що в поєднанні з невеликим зануренням у смітник призвело до безкоштовних поїздок автобусом. Пізніше Кевін застосував свої навички соціальної інженерії через телефон. Ретельно підготувавши атаки, він зміг отримати доступ до частин багатьох телефонних систем приватних філій (PBX), що, серед іншого, дозволило йому здійснювати безкоштовні міжміські дзвінки. У 1993 році Кевіна розслідувало ФБР і було засуджено за кілька злочинів, пов'язаних з комп'ютером.

6. Порушення RSA Security LLC. У березні 2011 року група злочинців використала методи соціальної інженерії, щоб зламати RSA Security, компанію (названу на честь ініціалів співзасновників Рона Ріввеста, Аді Шаміра та Леонарда Адлемана), відому завдяки токенам двофакторної автентифікації RSA. Використовуючи соціальну інженерію електронною поштою (тобто фішинг електронної пошти), правопорушник переконав співробітника відкрити прикріплений файл електронної таблиці з шкідливим програмним забезпеченням. Зловмисне програмне забезпечення встановило бекдор, який дозволив порушнику отримати доступ до машини. Звідти зловмисник міг переглядати мережу. Орієнтовна вартість злому склала 66,3 мільйона доларів. У період після порушення валовий прибуток RSA скоротився з 67,6% до 54,1%. Іншим аспектом порушення є шкода репутації RSA Security. У охоронному

бізнесі хороша лише така репутація. Справжній вплив цієї атаки невідомий; однак це дало можливість конкуренції завершитися.

Ці шість сценаріїв показують, що соціальна інженерія існує вже деякий час і може відбуватися різними способами. У наступному розділі обговорюється, у яких формах може відбуватися соціальна інженерія.

1.3. Приклади відомих інцидентів за останні роки.

Порушення даних JPMorgan Chase у 2014 році. Витік даних JPMorgan Chase у 2014 році був кібератакою проти американського банку JPMorgan Chase, який, як вважають, скомпрометував дані, пов'язані з понад 83 мільйонами облікових записів — 76 мільйонами домогосподарств (приблизно дві з трьох домогосподарств у країні) і 7 мільйонів малих підприємств. Витік даних вважається одним із найсерйозніших вторгнень в інформаційну систему американської корпорації та одним із найбільших в історії.

Вказана інформація, яку було зламано, була контактною інформацією користувача, тобто ім'ям, адресою, номером телефону та електронною адресою, а також внутрішньою інформацією JPMorgan Chase, що стосується таких користувачів. Однак номери облікових записів, паролі, ідентифікатори користувачів, дати народження або номери соціального страхування не були включені згідно з документацією.

Коли користувач клацав посилання tinyurl, він переспрямовувався на веб-адресу, схожу на URL-адресу мобільного банкінгу, яку використовує Chase, де йому відкривався наведений нижче підроблений екран входу Chase: тоді атака покладається на те, що нічого не підозрюють користувачі, які входять до свого банку деталі. Хоча вхід є фальшивим, спільною рисою таких атак є те, що інші посилання на екрані (FAQ, Contact Us тощо) спрямовують на справжні банківські веб-сайти.

Рис 1.2. Підроблена сторінка входу в сервіс-банкінг

Щойно атаку було виявлено, її було заблоковано клієнтами наших операторів, хоча деякі абоненти інших операторів могли отримати SMS-фішинговий текст. Як завжди, якщо було отримано підозріле текстове повідомлення, не потрібно вводити свої банківські реквізити та повідомити свого оператора про те, що був отриманий спам.

Вирішальне питання полягає в тому, чи пов’язані ці два інциденти. За оцінкою експертів, сумнівно, що ці два зв’язки пов’язані. Фішингові атаки на SMS-банк, на жаль, досить поширені та постійні, багато брендів банків з часом стають цілями, тому націлювання на JPMorgan могло бути просто випадковим збігом. Крім того, ця атака була відносно невеликою, і наразі експерти не бачать жодних ознак того, що ця атака була виконана у спосіб, який відрізняється від звичайного цими зловмисниками.

Це не означає, що екперти не побачать атаки у майбутньому, які використовують цю інформацію. Як зазначалось, отримання інформації, пов’язаної з більшістю домогосподарств у США, матиме величезну цінність для фішерів. Цілком імовірно, що якщо зловмисники вирішать продати свою інформацію для використання в SMS-фішингових атаках, раптова поява мільйонів контактних даних може призвести до падіння ціни на цю інформацію

в кримінальному підпіллі через надлишок пропозиції (щось які потенційно спостерігалися під час інших порушень), що призводить до того, що цілеспрямовані фішингові атаки стають дешевшими у виконанні, і тому більше людей піддаються ризику шахрайства[25].

Атака на Equifax у 2017 році. Витік даних Equifax стався в період з травня по липень 2017 року в американському кредитному бюро Equifax. Особисті записи 147,9 мільйона американців, а також 15,2 мільйона громадян Великобританії та близько 19 000 громадян Канади були скомпрометовані в результаті злому, що зробило його одним із найбільших кіберзлочинів, пов'язаних із крадіжкою особистих даних. У рамках врегулювання з Федеральною торговою комісією Сполучених Штатів Equifax запропонувала постраждалим користувачам компенсаційні кошти та безкоштовний кредитний моніторинг.

У лютому 2020 року уряд Сполучених Штатів висунув звинувачення членам Народно-визвольної армії Китаю у зламі Equifax і викраденні конфіденційних даних у рамках масштабного пограбування, яке також включало крадіжку комерційних таємниць, хоча Комуністична партія Китаю заперечувала ці заяви.

Ключовий патч безпеки для Apache Struts було випущено 7 березня 2017 року після того, як було виявлено експлойт безпеки, і всім користувачам фреймворку було запропоновано негайно оновити його.

Як було встановлено в результаті посмертного аналізу, злам у Equifax розпочався 12 травня 2017 року, коли Equifax ще не оновив свій веб-сайт кредитних суперечок новою версією Struts. Хакери використовували експлойт для отримання доступу до внутрішніх серверів у корпоративній мережі Equifax. Інформація, яка спочатку була отримана хакерами, включала внутрішні облікові дані для співробітників Equifax, які потім дозволяли хакерам здійснювати пошук у базах даних кредитного моніторингу під виглядом авторизованого користувача. Використовуючи шифрування для подальшого маскування своїх пошуків, хакери виконали понад 9000 сканувань баз даних, витягли інформацію

в невеликі тимчасові архіви, які потім були передані з серверів Equifax, щоб уникнути виявлення, і видалили тимчасові архіви після завершення. Діяльність тривала протягом 76 днів до 29 липня 2017 року, коли Equifax виявив злам і згодом, до 30 липня 2017 року, вимкнув експлойт. Щонайменше 34 сервери в двадцяти різних країнах використовувалися в різних точках під час злому, що ускладнювало відстеження зловмисників.

Згідно з аналізом Equifax, інформація, отримана під час злому, включала імена та прізвища, номери соціального страхування, дати народження, адреси та, у деяких випадках, номери водійських прав приблизно 143 мільйонів американців. Також була скомпрометована інформація про приблизно від 400 000 до 44 мільйонів жителів Великобританії, а також 8 000 жителів Канади. Крім того, постраждали ще 11 670 канадців, як пізніше стало відомо Equifax. Також було отримано доступ до номерів кредитних карток приблизно 209 000 споживачів у США та певних документів щодо суперечок з особистою інформацією приблизно 182 000 споживачів у США.

Після першого розкриття інформації у вересні 2017 року Equifax збільшила кількість записів, до яких вони виявили доступ. Як у жовтні 2017 року, так і в березні 2018 року Equifax повідомляв, що було отримано додаткові 2,5 та 2,4 мільйона записів американських споживачів, відповідно, довівши загальну кількість до 147,9 мільйонів. У жовтні 2017 року Equifax звузила свою оцінку споживачів у Великобританії, які постраждали від злому, до 15,2 мільйона, з яких 693 665 розкрили конфіденційні особисті дані.

Експерти з безпеки очікували, що прибуткові приватні дані, отримані в результаті злому, будуть перекриті та продані на чорних ринках і в темній мережі, хоча станом на травень 2021 року не було жодних ознак продажу цих даних. Оскільки дані не з’являлися відразу в перші 17 місяців після зламу, експерти з безпеки припустили, що або хакери, які стоять за зломом, чекали значний проміжок часу, перш ніж продати інформацію, оскільки продавати її було б занадто «гаряче». близько до злому, або що за зломом стоїть національна

держава, яка планує використовувати дані в нефінансовий спосіб, наприклад, для шпигунства[26].

Атака на Twitter у 2020 році. 15 липня 2020 року, між 20:00 і 22:00, 130 відомих облікових записів Twitter були зламані сторонніми особами з метою просування шахрайства з біткойнами. Twitter та інші медіа-джерела підтвердили, що зловмисники отримали доступ до адміністративних інструментів Twitter, щоб вони могли самостійно змінювати облікові записи та безпосередньо публікувати твіти. Схоже, вони використовували соціальну інженерію, щоб отримати доступ до інструментів через співробітників Twitter. 31 липня 2020 року влада заарештувала трьох осіб і звинуватила їх у шахрайстві, відмиванні грошей, крадіжці особистих даних і несанкціонованому доступі до комп'ютера, пов'язаному з шахрайством.

Дмитро Альперович, співзасновник компанії з кібербезпеки CrowdStrike, описав інцидент як «найгірший злом великої платформи соціальних мереж». Дослідники безпеки висловили занепокоєння, що соціальна інженерія, використана для здійснення злому, може вплинути на використання соціальних мереж у важливих онлайн-дискусіях, включаючи підготовку до президентських виборів у Сполучених Штатах у 2020 році. 31 липня 2020 року Міністерство юстиції США оголосило про висунення звинувачень трьом особам у зв'язку з інцидентом.

Криміналістичний аналіз шахрайства показав, що початкові повідомлення про шахрайство вперше були опубліковані обліковими записами з короткими іменами, що складаються з одного або двох символів, наприклад «@6». Після цього близько 20:00 15 липня 2020 року з'явилися криптовалютні акаунти в Twitter, включно з обліковими записами Coinbase, CoinDesk і Binance. Потім шахрайство перейшло на більш гучні акаунти з першим таким твітом, надісланим з акаунта Ілона Маска в Twitter о 20:17. Інші ймовірно зламані облікові записи включали облікові записи таких відомих людей, як Барак Обама, Джо Байден, Білл Гейтс, Джефф Безос, MrBeast, Майкл Блумберг, Воррен Баффет, Флойд Мейвезер-молодший, Кім Кардашьян і Каньє Вест; і такі

компанії, як Apple, Uber і Cash App. Twitter вважає, що постраждали 130 облікових записів, хоча насправді лише 45 використовувалися для публікації шахрайського повідомлення. Більшість облікових записів, до яких увійшли під час афери, мали щонайменше мільйон підписників.

У твітах, пов'язаних із шахрайським зломом, стверджувалося, що відправник з благодійної мети відшкодовував би будь-якому користувачеві подвійну вартість будь-якого біткойна, який він надіслав на певні гаманці, часто в рамках заходів з надання допомоги COVID-19. Твіти послідували за поширенням шкідливих посилань низкою криптовалютних компаній; веб-сайт, на якому розміщені посилання, було видалено незабаром після публікації твітів. Хоча такі шахрайські шахрайства «подвоїти свій біткойн» були поширеними у Twitter раніше, це був перший серйозний випадок, коли вони надсилалися зі зламаних облікових записів високого рівня. Експерти з безпеки вважають, що зловмисники провели шахрайство як операцію «знищи та захопи»: знаючи, що вторгнення в облікові записи буде швидко закрито, зловмисники, ймовірно, планували, що лише невелика частка мільйонів, які стежать за цими обліковими записами, повинна потрапити на них. шахрайства за цей короткий час, щоб швидко заробити на цьому. На цих веб-сайтах було вказано кілька біткойн-гаманців; перший із спостережених отримав 12 біткойнів із понад 320 транзакцій на суму понад 118 000 доларів США та вилучив із нього близько 61 000 доларів США, а другий мав суми лише в тисячах доларів, оскільки Twitter вжив заходів, щоб припинити публікації. Незрозуміло, чи були це кошти, додані тими, кого веде шахрайство, оскільки відомо, що шахраї з біткойнами додають кошти в гаманці перед тим, як починати схеми, щоб зробити шахрайство легітимним.

Деякі зі зламаних облікових записів неодноразово публікували повідомлення про шахрайство, навіть після видалення деяких повідомлень. Твіти були позначені як надіслані за допомогою веб-додатку Twitter. Одну з фраз, задіяних у шахрайстві, було твітнуто понад 3000 разів протягом чотирьох годин, причому твіти надсилалися з IP-адрес, пов'язаних із багатьма різними країнами.

Повторне використання фраз дозволило Twitter легко видалити образливі твіти, оскільки вони вжили заходів, щоб зупинити шахрайство.

О 21:45 Twitter оприлюднив заяву, в якій говориться, що їм «відомо про інцидент безпеки, який вплинув на облікові записи в Twitter», і що вони «вживають заходів, щоб це виправити». Невдовзі після цього він вимкнув можливість для деяких облікових записів публікувати твіти або скидати їхні паролі. Twitter не підтвердив, які облікові записи були обмежені, але багато користувачів з обліковими записами, які Twitter позначив як «перевірені», підтвердили, що вони не можуть твітнути. Приблизно через три години після перших шахрайських твітів Twitter повідомив, що, на їхню думку, вдалося відновити облікові дані для всіх зачеплених облікових записів їхнім законним власникам. Принаймні одна біржа криптовалют, Coinbase, внесла біткойн-адреси в чорний список, щоб запобігти надсиланню грошей. Coinbase повідомила, що вони зупинили надсилання понад 1000 транзакцій на загальну суму понад 280 000 доларів США[27].

1.4. Цілі та мотивація зловмисників.

Фінансовий злочин - це одна з основних мотивацій для використання соціальної інженерії. Цей вид атаки спрямований на отримання фінансової вигоди через маніпуляцію і обман інших осіб. Фінансові злочини можуть приймати різні форми, і вони стають все більш вишуканими завдяки розвитку технологій та доступу до особистої інформації[20].

Нижче подано таблицю, яка ілюструє різні типи фінансових злочинів, які можуть бути вчинені з використанням соціальної інженерії:

Таблиця 1.1

Типи фінансових злочинів

Тип злочину	**Опис**
1	2
Фішинг (Phishing)	Зловмисники надсилають обманні повідомлення або електронні листи, видаваючи себе за довірених джерел, з метою викликати жертву поділитися своїми фінансовими даними, такими як паролі чи кредитні картки.
Проникнення через соціальні мережі	Зловмисники встановлюють довіру з потенційною жертвою через соціальні мережі, а потім використовують цей доступ для отримання фінансових даних.
Атаки на банківські реквізити	Зловмисники можуть виманювати банківські реквізити або інші фінансові дані через імітацію офіційних веб-сайтів банків або платіжних систем.
Обман інвестицій	Зловмисники надають неправдиву інформацію про інвестиції або фінансові можливості, що змушує жертву інвестувати гроші, які потім зникають.

1	2
Шахрайство з використанням кредитів	Зловмисники можуть відкривати кредитні лінії на ім'я жертви або використовувати їхню кредитну історію для отримання позики, яка ніколи не повертається.

Отримання конфіденційної інформації є найважливішою метою різних соціальних інженерів, оскільки ця інформація може бути використана для різних цілей, включаючи крадіжку ідентифікаційних даних, доступ до захищених ресурсів або використання інформації в інших шахрайських схемах. Для досягнення цієї мети соціальні інженери використовують різноманітні підходи та техніки, які базуються на психологічних інсайтах та соціальних взаємодіях. У цьому підпункті ми розглянемо основні методи та стратегії, які використовуються для отримання конфіденційної інформації у соціальній інженерії[21].

Іноді соціальні інженери спрямовують свої зусилля на отримання доступу до захищених ресурсів, таких як корпоративні системи, банківські облікові записи або особисті файли. Для цього вони використовують різні методи, щоб обійти захист та переконати жертву надати доступ.

Таблиця 1.2

Методи отримання доступу до захищених ресурсів

Метод	**Опис методу**
1	2
Спілкування зі службовцями	Зловмисники можуть намагатися отримати інформацію від співробітників або службовців під обманом.

1	2
Використання ідентифікаційних даних	Якщо зловмисники вже мають доступ до ідентифікаційних даних, вони можуть використовувати їх для входу до систем або облікових записів.
Соціальна інженерія в службах підтримки	Вони можуть звертатися до служб підтримки компаній або служб підтримки користувачів під якимось обманом для отримання доступу.

Зловмисники також можуть використовувати різні схеми обману для отримання конфіденційної інформації. Ці схеми можуть включати в себе фальшиві історії, підроблені документи або вигадані обставини, щоб переконати жертву надати інформацію.

Таблиця 1.3

Приклади схем обману

Схема обману	**Опис**
1	2
Схема "прохання про допомогу"	Зловмисники можуть вигадати ситуацію, в якій жертва повинна надати інформацію або гроші для допомоги.
Вигадана аварія	Повідомлення про аварію або кризову ситуацію, в якій зловмисники виступають як постраждалі, щоб отримати допомогу.

1	2
Схема "помилкова доставка"	Зловмисники можуть відправляти фальшиві повідомлення про помилкові доставки товарів або послуг, щоб вимагати інформацію для "повернення" товару.

Зловмисники можуть здійснювати спостереження за жертвами, вивчати їхні звички та збирати інформацію про них. Це дозволяє їм створювати персоналізовані атаки та використовувати отриману інформацію для маніпуляції жертвами[21].

Таблиця 1.4

Методи спостереження та аналізу

Метод	**Опис**
1	2
Спостереження в реальному часі	Зловмисники можуть фізично стежити за жертвами або використовувати технологію для спостереження в онлайні.
Аналіз інформації в соцмережах	Вони можуть аналізувати профілі та дописи жертв в соціальних мережах для збору інформації.

Розвідка та шпигунство - це ключові аспекти соціальної інженерії, які використовуються для збору інформації про цільових жертв, організації або системи. Використання цих методів дозволяє зловмисникам отримувати детальну інформацію, яка може бути використана для подальших атак або зловживань. У цьому підпункті ми розглянемо різні методи розвідки та шпигунства, їхні приклади та вплив на цільові об'єкти.

Офлайн-розвідка включає в себе фізичний доступ до цільових об'єктів або організацій для збору інформації. Це може включати в себе фізичне спостереження, інфільтрацію внутрішніх приміщень або збирання інформації зі смітників. Офлайн-розвідка дозволяє зловмисникам отримувати конфіденційну інформацію, яку неможливо отримати онлайн.

Таблиця 1.5

Приклади методів офлайн-розвідки

Метод	**Опис**
1	2
Фізичне спостереження	Спостереження за поведінкою та рухом людей або транспорту для збору інформації.
Інфільтрація	Введення агента або зловмисника в організацію або об'єкт з метою збору інформації.
Дімпстерінг	Збирання інформації з смітників та відходів, що може містити конфіденційну інформацію.

Офлайн-розвідка може бути особливо ефективною для атак на корпоративні об'єкти або для отримання фізичного доступу до приміщень або систем[20].

Ще однією технікою розвідки та шпигунства є фізична заміна та імперсонація. Зловмисники можуть видаватися за співробітників, доставкарів, технічних спеціалістів або інших осіб, що мають легальний доступ до об'єкта. Це дозволяє їм отримувати фізичний доступ до систем, приміщень або інформації[12].

Таблиця 1.6

Приклади фізичної заміни та імперсонації

Метод	**Опис**
1	2
Вигляданння як співробітник	Зловмисники можуть одягатися та діяти як співробітники підприємства або організації.
Вигляданння як службовець	Використання одягу та обладнання, яке асоціюється із службовцями певних організацій.
Імперсонація того, хто доставляє	Вигляданння як доставкар та намагання надати доставку або отримати доступ.

Фізична заміна та імперсонація можуть бути особливо небезпечними, оскільки зловмисники отримують прямий доступ до об'єктів або систем і можуть виконувати різні атаки, включаючи крадіжку інформації або встановлення шкідливого програмного забезпечення.

Зловмисники можуть здійснювати спостереження за жертвами або об'єктами для збору інформації. Це може включати в себе використання різноманітних технічних засобів, таких як камери, мікрофони, GPS-трекери тощо, для відстеження руху та дій[22].

Таблиця 2.7

Приклади спостереження та збору інформації

Метод	**Опис**
1	2
Встановлення шпигунських пристроїв	Використання мікрофонів, камер або GPS-трекерів для отримання інформації.
Спостереження з відстані	Використання телескопів, біноклів або інших оптичних засобів для спостереження здалеку.

1.5. Висновки до першого розділу.

В ході виконання першого розділу ми присвятили головну увагу основам соціальної інженерії, надали глибокий огляд цієї складної та актуальної теми. Ми розглянули основні аспекти соціальної інженерії, включаючи її визначення, історію розвитку, фінансовий злочин, цілі та мотивацію злочинців. Наш аналіз розкриває, що соціальна інженерія стала одним із найпоширеніших методів атак у сфері кібербезпеки.

Ми визначили соціальну інженерію як маніпуляцію людським фактором з метою отримання доступу до конфіденційної інформації або систем. Соціальна інженерія включає в себе використання психологічних методів та соціокультурних аспектів для досягнення своїх цілей. Цей підрозділ наголошує, що соціальна інженерія може бути так само небезпечною, як і технічні атаки.

Потім була присваячена увага цілям та мотивації зловмисників у сфері соціальної інженерії, надали глибше розуміння, чому та для чого атакують з використанням цього методу. Ми проаналізували різні мотиви та цілі зловмисників, що допомагають нам розуміти, яким чином соціальна інженерія

може бути використана для досягнення різних цілей. Зловмисники можуть використовувати фішінгові атаки, бізнес-шахрайство, обман із застосуванням кредитних карт і інші методи, щоб вкрасти гроші, особисту інформацію або доступ до фінансових ресурсів. Мотивовані фінансовими вигодами, зловмисники намагаються виникають якомога більше можливостей для грабежу та обману.

Також розглянули приклади деяких найвідоміших атак, які були здійснені за допомогою соціальної інженерії. Ці приклади демонструють, наскільки серйозними та широкомасштабними можуть бути атаки цього виду.

РОЗДІЛ 2. ТИПИ АТАК З ВИСКОРИСТАННЯМ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

Соціальна інженерія включає в себе широкий спектр атак, які базуються на маніпуляції людським фактором. Ці атаки можуть бути виконані різними способами і нерідко стають першим кроком для отримання доступу до конфіденційної інформації або вразливих систем. У цьому розділі ми розглянемо основні типи атак у соціальній інженерії, їх характеристики та приклади.

2.1. Фішинг.

Фішинг є однією з найпоширеніших та небезпечних атак соціальної інженерії. Вона полягає в тому, щоб зловмисники надсилають обманні повідомлення або електронні листи, видаваючи себе за довірених джерел, з метою викликати жертву поділитися своїми фінансовими даними, такими як паролі, номери кредитних карток або особисту інформацію[16]. Основна мета фішингу - викликати довіру та вести жертву до небезпечних дій[20].

Характеристики фішингу:

- Використання електронної пошти, веб-сайтів, соціальних мереж та інших комунікаційних засобів для спілкування з потенційною жертвою.
- Часто супроводжується ложними логотипами та офіційними виглядами, що надає обманливого характеру повідомленням.
- Зазвичай включає в себе емоційну складову або тиск на жертву (наприклад, "Ваш акаунт буде заблоковано, якщо ви не введете свій пароль").
- Мета - отримати конфіденційні дані для фінансового обману або доступу до облікового запису.

Приклад фішингу: Наприклад, зловмисники можуть відправити лист від імені банку, в якому жертву повідомляють про невідомі операції на її

банківському рахунку та просять ввести особисті дані для верифікації облікового запису. При натисканні на посилання в листі жертва потрапляє на фальшивий веб-сайт, де зловмисники збирають її дані.

2.2. Проникнення через соціальні мережі.

Проникнення через соціальні мережі є специфічним видом соціальної інженерії, який базується на встановленні довіри з потенційною жертвою через соціальні платформи. Зловмисники можуть створювати фальшиві профілі, взаємодіяти з потенційними жертвами та отримувати важливу інформацію, яка використовується для зловживання[22].

Характеристики проникнення через соціальні мережі:

- Використання соціальних мереж, таких як Facebook, LinkedIn, Twitter та інші, для встановлення контакту з потенційними жертвами.
- Збирання особистої інформації про жертву, яка може включати в себе місце роботи, родинний стан, інтереси та друзів.
- Використання цієї інформації для вигадування обманлих історій або впливу на жертву, щоб отримати фінансову вигоду або конфіденційну інформацію.

Приклад проникнення через соціальні мережі: Зловмисник може створити фальшивий профіль у соціальній мережі та встановити контакт з робітником підприємства, яке цікавить його зловмисну діяльність. Під час взаємодії з цим робітником зловмисник може вигадати обманливу історію та надати важливу інформацію, яка може бути використана для вторгнення в систему підприємства.

2.3. Атаки на банківські реквізити.

Атаки на банківські реквізити включають в себе виманювання банківських інформаційних даних, таких як номери банківських карток, PIN-коди та інші фінансові дані. Зловмисники часто вдаються до імітації офіційних веб-сайтів банків або платіжних систем для отримання цих даних від жертв.

Характеристики атак на банківські реквізити:

- Використання фальшивих веб-сайтів, які дуже схожі на офіційні ресурси банків або платіжних систем.
- Відправка спаму електронною поштою або повідомлень з проханням ввести банківську інформацію.
- Імітація листів від банків, які повідомляють про незвичайну активність на обліковому записі та просять введення даних для перевірки.

Приклад атаки на банківські реквізити: Зловмисники можуть відправити лист від імені банку, в якому повідомляють жертву про підозрілі операції на її рахунку. У листі можуть бути посилання на фальшивий веб-сайт банку, де жертва повинна ввести свої банківські дані для "перевірки облікового запису"[20].

2.4. Обман інвестицій.

Обман інвестицій включає в себе надання неправдивої інформації про інвестиційні можливості або фінансовий успіх з метою залучення жертв до інвестування грошей. Ця атака може приймати форму фіктивних інвестиційних фондів, акцій, криптовалют або інших фінансових можливостей[22].

Характеристики обману інвестицій:

- Надання інформації, яка обіцяє високий прибуток при мінімальному ризику.
- Використання психологічних методів для переконання жертви, що ця інвестиція є найкращою можливістю.
- Збирання грошей від інвесторів замість реального інвестування та виділення прибутку.

Приклад обману інвестицій: Зловмисники можуть затягувати нових інвесторів в фіктивний інвестиційний фонд, обіцяючи надзвичайний прибуток. Після того як залучені кошти стають достатньо великими, зловмисники можуть зникнути з грошима, залишивши інвесторів без коштів та можливості відновлення інвестованих активів.

2.5. Шахрайство з використанням кредитів.

Цей вид атаки полягає в тому, щоб зловмисники відкривали кредитні лінії на ім'я жертви або використовували їхню кредитну історію для отримання позики, яка ніколи не повертається. Шахраї можуть отримувати гроші та відкривати кредитні картки на ім'я іншої особи, завдаючи фінансових втрат та пошкоджень її кредитній історії[22].

Характеристики шахрайства з використанням кредитів:

- Використання особистих даних жертви для отримання кредитних продуктів.
- Передача боргів та фінансового зобов'язання на ім'я жертви.
- Порушення кредитної історії жертви, що може призвести до погіршення її кредитної репутації.

Приклад шахрайства з використанням кредитів: Зловмисники можуть отримати інформацію про жертву та відкрити новий кредитний рахунок на її ім'я.

Після видачі кредиту зловмисники можуть зникнути, не сплачуючи борги, і залишити жертву з кредитною проблемою, яку важко вирішити.

Ці типи атак у соціальній інженерії є лише частиною широкого спектру методів, які можуть бути використані зловмисниками для досягнення своїх цілей.

2.6. Етапи соціальної інженерії.

Перший етап соціальної інженерії - це збір інформації про потенційну жертву або ціль. Ця інформація допомагає зловмисникам легше визначити, які методи та тактики будуть найбільш ефективними для маніпуляції жертвою[23]. Збір інформації може включати в себе:

- Збір особистих даних: Ім'я, адреса, день народження, родинний стан, інтереси, хобі тощо.
- Вивчення соціальних мереж: Аналіз профілів в соціальних мережах для отримання додаткової інформації та встановлення зв'язків з жертвою.
- Дослідження організації: Вивчення структури та робочих процесів організації, в якій працює жертва, якщо це застосовно.

Приклад: Зловмисник, який планує виконати атаку на співробітника підприємства, може вивчити інформацію з профілю жертви в соціальних мережах, дізнатися її інтереси та зв'язки з колегами.

2.7. Встановлення довіри та побудова відносин.

Після збору інформації зловмисники переходять до етапу формування довіри. Це дуже важливий етап, оскільки успішність соціальної інженерії часто залежить від того, наскільки жертва довіряє атакуючому. Існує кілька способів, якими зловмисники можуть створити довіру[24]:

- Використання піддельних облікових записів: Зловмисники можуть створювати піддельні облікові записи або фіктивні персонажі, які виглядають довірливо.
- Використання соціальних інженерних методів: Вони можуть використовувати психологічні та маніпуляційні методи для створення довіри.
- Імітація довірених джерел: Зловмисники можуть видавати себе за представників організацій, банків, інтернет-платформ або інших довірених джерел.

Приклад: Зловмисник, який намагається виманити пароль від облікового запису жертви, може вигадати історію про проблеми з безпекою облікових записів і надати рекомендації щодо зміни пароля.

2.8. Використання маніпуляційних технік.

На цьому етапі зловмисники починають використовувати отриману інформацію та створену довіру для маніпуляції жертвою та отримання бажаної інформації або доступу до систем. Цей етап може включати в себе:

- Спеціально сплановані запити: Зловмисники можуть використовувати інформацію про жертву для формулювання запитів, які здавалися б логічними і обґрунтованими.

- Соціальні інженерні техніки: Вони можуть використовувати психологічні методи, такі як соціальний тиск, страх або вдячність, для отримання бажаної реакції від жертви.
- Вигадування ситуацій: Зловмисники можуть створювати вигадані ситуації, в яких жертва повинна надати необхідну інформацію або виконати певні дії.

Приклад: Зловмисник, який намагається отримати доступ до корпоративних даних, може надати себе за технічного підтримуючого та попросити жертву виконати певні дії на своєму комп'ютері для "виправлення проблеми", включаючи введення паролю або завантаження шкідливого ПЗ[18].

2.9. Виконання атаки та отримання доступу.

На останньому етапі зловмисники використовують отриману інформацію або отриманий доступ для досягнення своїх цілей[18]. Це може включати в себе:

- Використання отриманих облікових даних для доступу до систем або ресурсів.
- Використання зібраної інформації для вчинення фінансових злочинів або обману жертви.
- Використання доступу для отримання конфіденційної інформації або розповсюдження шкідливого ПЗ.

Приклад: Зловмисник, який отримав доступ до корпоративної мережі, може використовувати цей доступ для крадіжки конфіденційних даних або розповсюдження шкідливого ПЗ всередині організації.

Нижче ми можемо навести результуючу таблицю.

Таблиця 2.1

Етапи соціальної інженерії

Етап	**Характеристика**
1	2
Збір інформації	Збір особистих даних, вивчення соціальних мереж, дослідження організації.
Формування довіри	Використання піддельних облікових записів, соціальні інженерні техніки, імітація довірених джерел.
Маніпуляція та обман	Спеціально сплановані запити, соціальні інженерні техніки, вигадування ситуацій.
Використання отриманої інформації	Використання отриманих облікових даних, зібраної інформації для досягнення цілей.

2.10. Психологічні аспекти соціальної інженерії.

Психологія грає ключову роль у виконанні успішних соціальних інженерних атак. Зловмисники використовують різні психологічні стратегії та методи, щоб впливати на свої жертви та досягати своїх цілей. Розуміння того, як психологія впливає на успішність атаки, допомагає як жертвам, так і фахівцям з кібербезпеки легше розпізнавати та захищатися від соціальної інженерії[8].

Важливі аспекти психології, які впливають на успішність атак соціальної інженерії[14]:

1. Довіра і довірливість. Люди мають природну схильність довіряти іншим і вірити в доброчесні наміри. Зловмисники використовують цю природну нахил до довіри для своєї переваги. Вони можуть видавати себе за довірених агентів, друзів або колег та отримувати від жертви конфіденційну інформацію.

2. Емоційний вплив. Зловмисники можуть використовувати емоційний вплив, щоб створити сприятливий фон для маніпуляції. Вони можуть викликати страх, радість, співчуття або невпевненість у жертви, щоб отримати бажану реакцію. Наприклад, створюючи страх перед можливими наслідками, зловмисники можуть спонукати жертву діяти швидше і без ретельного розгляду.

3. Соціальний тиск. Соціальний тиск може бути використаний для впливу на жертву. Він включає в себе використання групової динаміки та соціальних очікувань для отримання бажаної реакції від жертви. Наприклад, зловмисники можуть створити вигадану ситуацію, в якій жертва відчуває тиск групи, і це може спонукати її до дії.

4. Психологічна довіра до авторитетів. Багато людей мають схильність слухати та вірити авторитетам або експертам. Зловмисники можуть видавати себе за представників авторитетних організацій або фахівців у певних галузях, щоб отримати довіру жертви.

5. Спроможності прийняття рішень і обмежена обробка інформації. Люди мають обмежені здібності оброблювати інформацію та приймати рішення, особливо в ситуаціях стресу або тиску. Нижче наведемо таблицю, яка ілюструє вплив психології на успішність атак соціальної інженерії.

Таблиця 2.2

Вплив психології на успішність атак

Психологічний аспект	**Вплив**
1	2
Довіра і довірливість	Збільшує шанси отримати конфіденційну інформацію від жертви.

1	2
Емоційний вплив	Сприяє створенню сприятливого фону для маніпуляції жертвою.
Соціальний тиск	Може змусити жертву діяти швидше та без ретельного розгляду.
Психологічна довіра до авторитетів	Підвищує шанси отримати довіру та сприяє маніпуляції.
Спроможності прийняття рішень та обмежена обробка інформації	Використовується для спрощення інформації та призводить до необгрунтованих дій.

2.11. Соціальна інженерія як мистецтво впливу.

Соціальна інженерія може бути розглянута як справжнє мистецтво впливу на людей і системи. Вона вимагає від зловмисників розуміння психології, маніпуляційних технік і соціальних динамік для досягнення своїх цілей. У цьому підпункті ми розглянемо соціальну інженерію як мистецтво впливу, розкриваючи її ключові аспекти та методи[24].

Одним із головних аспектів соціальної інженерії є здатність збуджувати довіру і довірливість у жертви. Зловмисники майстерно відтворюють образи довірених агентів, друзів або професіоналів, щоб здобути довіру. Вони можуть використовувати психологічні методи, такі як імітація довірених джерел і створення обмеженого часового пресу, щоб спонукати жертву до поділу конфіденційної інформації[17].

Таблиця 2.3

Психологічні методи впливу на довіру і довірливість

Метод впливу	**Опис методу**
1	2
Вигляд довіреним агентом	Виглядання як довірена особа або офіційний представник.
Імітація довірених джерел	Створення видимості довірених джерел або авторитетів.
Створення обмеженого часового пресу	Створення тиску на жертву для швидкого реагування.

Зловмисники використовують соціальні очікування та норми для маніпуляції жертвами. Вони можуть створити ситуації, які відповідають сподіванням жертви і викликають певну реакцію. Наприклад, вони можуть симулювати невідкладну ситуацію, щоб виграти час і змусити жертву діяти швидше, ніж розсуджувати[24].

Таблиця 2.4

Використання соціальних очікувань у соціальній інженерії

Соціальний метод впливу	**Опис методу**
1	2
Симуляція невідкладної ситуації	Створення вигаданої невідкладної ситуації для змусити жертву діяти швидше.
Використання соціальних норм	Виглядання як соціально прийнята поведінка або обставини.
Подія, що сприймається як закономірна	Створення ситуацій, що здаються закономірними та очікуваними.

Зловмисники можуть впливати на емоційний стан жертви, створюючи сприятливий або напружений емоційний фон. Вони можуть викликати страх,

радість, співчуття або невпевненість у жертви, в залежності від своїх цілей. Наприклад, вони можуть погрожувати можливими наслідками, щоб змусити жертву діяти на їхню користь.

Таблиця 2.5

Використання емоційного фону у соціальній інженерії

Емоційний метод впливу	**Опис методу**
1	2
Виклик страху	Створення страху перед можливими наслідками.
Виклик радості	Створення радісного емоційного фону для підвищення співпраці.
Виклик невпевненості	Збурення невпевненості для спонукання до дій.

2.12. Кібергігієна

Кібергігієна, або кібергігієнічні заходи, представляють собою сукупність стратегій та практик, спрямованих на зміцнення кібербезпеки через освіту, підвищення уваги до кібернебезпеки та формування здорових кібергігієнічних звичок серед користувачів.

Кібергігієна починається з освіти та підвищення свідомості про потенційні кібернебезпеки. Це означає не лише навчання людей про загрози, а й формування правильних підходів та звичок у використанні технологій та обробці інформації в онлайн-середовищі[6].

Головною проблемою можна вважати збір даних про користувачів. Загальний підхід можна сформулювати так: чим інтенсивніше використовується пристрій, тим більше даних про свого власника воно накопичує. Дані бувають прямими – фотографії, контакти, повідомлення, які власник сам повідомив про пристрій та Інтернет. А є непрямі дані, які обчислюються самим пристроєм з урахуванням дій власника.

До них відносяться, наприклад:

- історія встановлення та використання додатків;
- історія енергоспоживання, тобто циклів та часу зарядки, інтенсивності роботи;
- історія повідомлень та дій;
- історія магазину додатків;
- історія браузера, його кеш (переглянуті нещодавно сторінки) та DNS;
- історія переміщень містом та багато іншого.

Дані збирають виробник пристрою (наприклад, Apple, Huawei, Samsung та ін.), платформа (iOS або Android), а також легальні та нелегальні (шкідливі) програми.

Багато програм при установці вимагають дати їм дозвіл на доступ до вашої телефонної книги, файлів, фотографій і т. д. Безкоштовному додатку вони потрібні для заробітку шляхом отримання та перепродажу персональних даних[10].

Є класичний приклад безкоштовного застосування «Ліхтарик», якому для функціонування жодних дозволів не потрібно; проте для встановлення воно вимагало дозволу на доступ до дзвінків та адресної книги.

У більшості шкідливих програм на смартфоні та ноутбуці - економічні цілі, тобто їм потрібно якось отримати або вкрасти гроші користувача або нажитися на ньому іншими способами.

Основні джерела доходу таких додатків такі[19]:

1. Продаж уваги. Найдорожче продається увага користувача: це валюта, якою оплачується доступ практично до будь-яких сервісів. Монетизація уваги

може бути різною – показ реклами, створення штучних незручностей, за зняття яких необхідно заплатити, участь у спільному створенні контенту (лайки та перегляди). Кількість контактів з користувачем, щільність утримання його уваги впливають на дохідність будь-якого подібного сервісу.

2. Продаж профілю користувача. Дещо дешевше продається профіль – сукупність даних про користувача: історія перегляду сайтів, історія використання програм, список контактів, історія геолокацій пристрою, списки бездротових мереж і Bluetooth-пристроїв, записи звукового оточення, параметри середовища (версія ОС, мобільний оператор, характеристики пристрою, що використовується) і т. д. Зазвичай ці дані продаються знеособленими: сама собою особистість користувача для рекламодавців не дуже цікава, тому що його персональні дані можуть бути чутливими (і стати джерелом юридичних ризиків для продавця). Сукупність таких даних продається через ланцюжок посередників маркетинговим компаніям і дозволяє їм приймати рішення щодо потенційної цінності конкретного користувача рекламодавця. Наприклад, поєднання застарілої версії ОС і відсутність ознак активного використання сучасних технологій (мало встановлених додатків, пристрій рідко використовується) приваблює рекламодавців, які просувають сумнівні платні мобільні сервіси, які непомітно знімають гроші з рахунку недосвідченого користувача. Крім того, може бути потрібне оточення користувача. Розповсюджений сценарій: під час реєстрації в месенджері або іншому схожому сервісі користувачеві відразу доступний список його контактів з пам'яті пристрою. Причому контакти, зазвичай, отримують повідомлення у тому, що він почав користуватися сервісом. На основі списку контактів, бездротових мереж, Bluetooth-пристроїв у зоні видимості користувача можна робити висновки про його найближче оточення, приблизний рівень доходу, сімейне становище.

3. Продаж неявного доступу до пристрою. Ще дешевше продається невидимий, неявний доступ до пристрою: накрутка реклами у фоновому режимі, майнінг (видобуток) криптовалюти, розсилка спаму, «проксування трафіку» без відома користувача, тобто використання телефону як маршрутизатора

стороннього інтернет-трафіку, що дозволяє зловмисникам маскувати діяльність у Мережі під звичайну користувальницьку активність – наприклад, для накрутки показів реклами.

Все це можуть робити без вашого відома на вашому смартфоні шкідливі програми, якщо ви встановите їх - з вашої волі або мимоволі.

2.13. Висновки до другого розділу.

В ході виконання другого розділу ми розкрили різні типи атак, які можуть бути використані в соціальній інженерії. Ми описали фішінг, техніки інженерії соціальних мереж, телефонні атаки, а також атаки через викрадення ідентичності. Це дозволило нам розібратися, які методи можуть бути використані зловмисниками для отримання доступу до цінної інформації.

Ми детально розглянули етапи соціальної інженерії, які зловмисники використовують для досягнення своїх цілей. Вияснили, що спочатку ловмисники збирають якомога більше інформації про свою цільову жертву, включаючи персональні дані, соціальні зв'язки, інтереси та звички. Ця інформація використовується для персоналізації атаки та збільшення її ефективності. Після збору інформації зловмисники намагаються встановити довіру з потенційною жертвою. Вони можуть використовувати цю інформацію, щоб створити персоналізовану підману та виглядати як довірена особа або джерело інформації. Зловмисники використовують різні маніпуляційні техніки, такі як маніпулювання емоціями, створення тиску або страху, щоб отримати від жертви бажаний результат. Вони можуть використовувати ці техніки, щоб переконати жертву виконати певні дії або надати конфіденційну інформацію. На останньому етапі зловмисники виконують атаку та намагаються отримати доступ до цільових ресурсів або інформації.

Був проведений докладний аналіз того, як психологія впливає на успішність атак та як соціальна інженерія може бути розглянута як мистецтво впливу. Ми також розглянули основні психологічні механізми, які використовуються зловмисниками для досягнення своїх цілей. Один із ключових аспектів психології соціальної інженерії полягає в тому, як психологічні фактори можуть впливати на успішність атаки. Ми визначили, що велика частина успіху атаки залежить від емоційної становища жертви, її довіри до атакуючого та рівня обачності.

РОЗДІЛ 3. ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ТАОПИС ВИКОРИСТАНИХ РЕСУРСІВ

3.1. Технічне завдання для розробки ПЗ.

Технічне завдання полягає в розробці програмного забезпечення, яке має за мету аналізувати електронні листи користувачів, отриманих через Gmail, для виявлення можливих прийомів соціальної інженерії. Програма буде використовувати штучний інтелект для сканування тексту листів з метою виявлення певних ключових слів, фраз та шаблонів, які можуть бути пов'язані з такими атаками.

Основна функціональність програми включатиме в себе підключення до поштової скриньки користувача через API Gmail, сканування листів та подальший аналіз тексту на предмет можливих загроз соціальної інженерії. Результати аналізу будуть відображені через графічний інтерфейс для зручності використання програми.

Для роботи з Gmail API планується використання відповідних бібліотек, таких як google.oauth2 та googleapiclient, щоб забезпечити доступ до поштової скриньки користувача та отримати необхідну інформацію.

Щодо аналізу тексту листів, для розпізнавання прийомів соціальної інженерії, планується використання бібліотеки scikit-learn, яка дозволить розробити модель машинного навчання для виявлення певних паттернів та вразливостей у тексті.

Одним із ключових аспектів розробки програми є забезпечення безпеки та конфіденційності даних користувача. Враховуючи чутливість інформації, отриманої з поштової скриньки, програма повинна гарантувати захист від несанкціонованого доступу та витоку даних.

Для оцінки ефективності програми та її здатності виявляти прийоми соціальної інженерії планується проведення тестування на різних наборах листів. Це дозволить оцінити точність аналізу та забезпечити відповідну функціональність програми.

У відповідності до цих завдань програма буде розроблена з використанням вказаних бібліотек та інструментів, з урахуванням необхідності забезпечення безпеки та високої ефективності аналізу тексту листів.

Цей проект покликаний підвищити рівень кібербезпеки користувачів Gmail шляхом автоматизованого виявлення можливих загроз та вразливостей через аналіз тексту отриманих листів.

3.2. Мова програмування Python.

На сьогоднішній день існує безліч мов програмування, кожна з яких має свої особливості. Але хочеться виділити Python як популярне універсальне середовище розробки програмного коду з більш ніж тридцятирічної історією.

Наприкінці 1989 року Гвідо Ван Россум створив Python - інтерпретований мову програмування, який дуже швидко став популярним і затребуваним у програмістів. На підтвердження цього можна навести компанії-гіганти, які використовують Python для реалізації глобальних проектів. Це Google, Microsoft, Facebook та багато інших[34].

Область застосування Python дуже велика. Його використовують для вирішення різних завдань: обробки наукових даних, систем управління життєзабезпеченням, ігор, веб-ресурсів, систем штучного інтелекту.

За весь час існування Python плідно використовувався та динамічно розвивався. Створювалися стандартні бібліотеки для підтримки сучасних технологій - наприклад, роботи з базами даних, протоколами Інтернету, електронною поштою, машинного навчання та багато іншого[1].

Для прискорення процесу написання програмного коду зручно використовувати спеціалізоване інструментальне середовище IDE для Python (Integrated Development Environment - інтегроване середовище розробки). Вона має повний набір засобів, необхідних для ефективного програмування на Python. Зазвичай до складу IDE входять текстовий редактор, компілятор чи інтерпретатор, налагоджувач та інше програмне забезпечення. Використання IDE дозволяє збільшити швидкість розробки програм (за умови попереднього навчання роботі з цим інструментальним середовищем).

Ще одна чудова особливість, яка надихає розробників користуватися Python, - це бібліотека Tkinter, завдяки якій можна досить швидко проектувати складні віконні інтерфейси додатків. Користувачеві досить легко перетягувати різні готові компоненти для створення власних віконних форм[38].

Мова програмування Python є досить потужним інструментальним засобом розробки систем штучного інтелекту (ШІ). Найбільшу цінність представляє навіть не так він сам, як набір бібліотек, що підключаються, на рівні яких уже реалізовані всі необхідні процедури та функції. Розробнику достатньо написати кілька десятків рядків програмного коду, щоб підключити необхідні бібліотеки, створити набір потрібних об'єктів, передати їм вихідні дані та відобразити підсумкові результати.

Рейтинг найбільш популярних мов програмування для їх використання в роботі з штучним інтелектом наведено на графіку нижче (рис. 3.1.). Як можемо бачити мова Python зайняла найбільшу частку серед усіх[9].

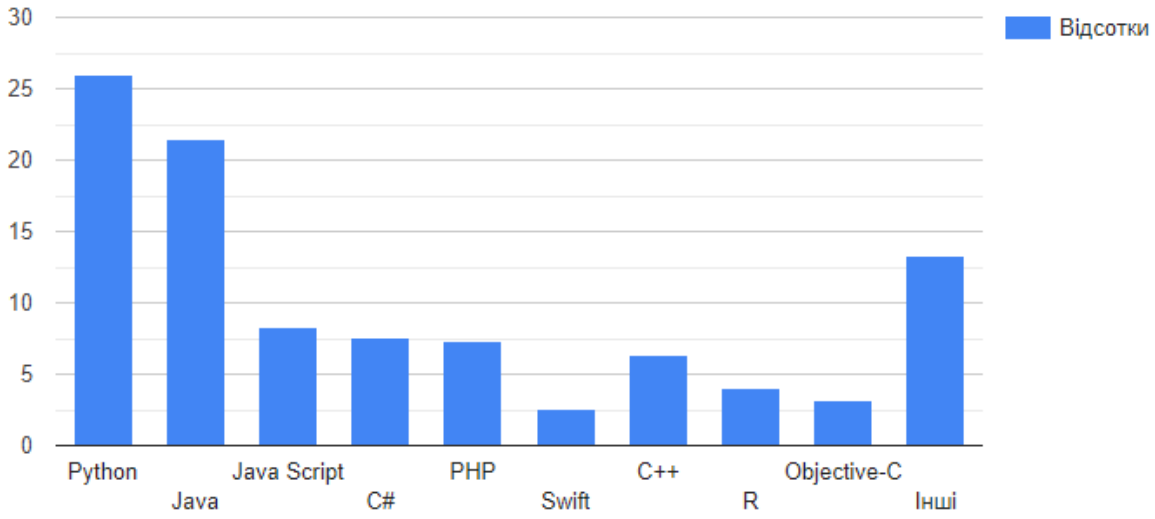

Рис 3.1. Найпопулярніші мови програмування для роботи з ШІ.

3.3. Машинне навчання без вчителя.

Машинне навчання - клас методів штучного інтелекту (ШІ), що дозволяють комп'ютерам навчатися на основі даних (зазвичай під час вирішення вузькоспеціалізованих завдань) без програмування. Термін машинне навчання було введено Артуром Семюелем, легендою в області ІІ, ще в 1959 році, проте до кінця XX століття лише небагатьом проектам вдалося досягти комерційного успіху в галузі машинного навчання, яка залишалася не більше ніж нішою для академічних досліджень[1].

Спочатку (у 1960-ті роки) багато членів спільноти ШІ були сповнені оптимізму. Наприклад, Герберт Саймон та Марвін Мінський вважали, що досягнення штучним інтелектом рівня людського розуму – справа найближчих десятиліть[2].

Осліплені оптимізмом, дослідники сфокусувалися на розвитку проектів так званого сильного ІІ. Їхньою метою було створення інтелектуальних агентів, здатних вирішувати завдання, формувати знання, навчатися і планувати свої дії, розуміти природну мову, сприймати навколишнє середовище та контролювати

моторику. Загальний ентузіазм сприяв залученню серйозного фінансування з боку таких організацій, як Міністерство оборони США, однак завдання, що стояли перед дослідниками, виявилися настільки амбітними, що спроби вирішити їх на той час були свідомо приречені на провал.

Академічні дослідження рідко доводилися рівня промислових розробок, що виявило себе у серії про " зим штучного інтелекту " . Ці "зимові" періоди (алюзія на вираз "ядерна зима": що існувало в епоху холодної війни) характеризувалися згасанням інтересу до ШІ і відповідним скороченням фінансування. До початку 1990-х років інтерес до ШІ та його фінансування були майже зведені нанівець.

Проте за останні два десятиліття галузь відродилася - спочатку в академічних колах, а потім у вигляді масового феномену, до якого залучені найкращі уми з університетського та корпоративного середовища. Вирішальну роль у відродженні ІІ зіграли три фактори: прорив у розробці алгоритмів машинного навчання, доступність великих обсягів даних та поява надшвидких комп'ютерів[2].

По-перше, замість фокусуватися на надто амбітних проектах сильного ШІ, дослідники зосередилися на вужчих завданнях, відомих як слабкий, або обмежений, ШІ. Результатом акцентування уваги на покращенні рішень для вузьких завдань став прорив у розробці алгоритмів, що проклав шлях до успішних комерційних програм. Багато з цих алгоритмів - часто з тих, які спочатку розроблялися в університетах чи приватних дослідницьких лабораторіях, - швидко перетворилися на проекти з відкритим вихідним кодом, що прискорило впровадження цих технологій у промисловому середовищі.

По-друге, більшість організацій зайнялося накопиченням даних, а вартість їхнього зберігання різко впала завдяки прогресу у створенні ефективних комп'ютерних сховищ, тоді як Інтернет забезпечив доступність великих обсягів даних у небачених раніше масштабах.

По-третє, завдяки хмарним технологіям стали доступні надпотужні обчислювальні ресурси, що дозволяє дослідникам легко та недорого

масштабувати IT інфраструктуру, не роблячи величезних інвестицій у обладнання.

Під впливом трьох перерахованих вище факторів дослідження в галузі II перемістилися з академічної сфери в промислову, що сприяло підвищенню інтересу до II та залученню більш суттєвого фінансування, яке з року в рік збільшується. Тепер штучний інтелект - як предмет теоретичних досліджень, а й повноцінна прикладна область. На рис. 3.2 показано графік з Google Trends, що свідчить про зростання інтересу до машинного навчання за останні 5 років[28].

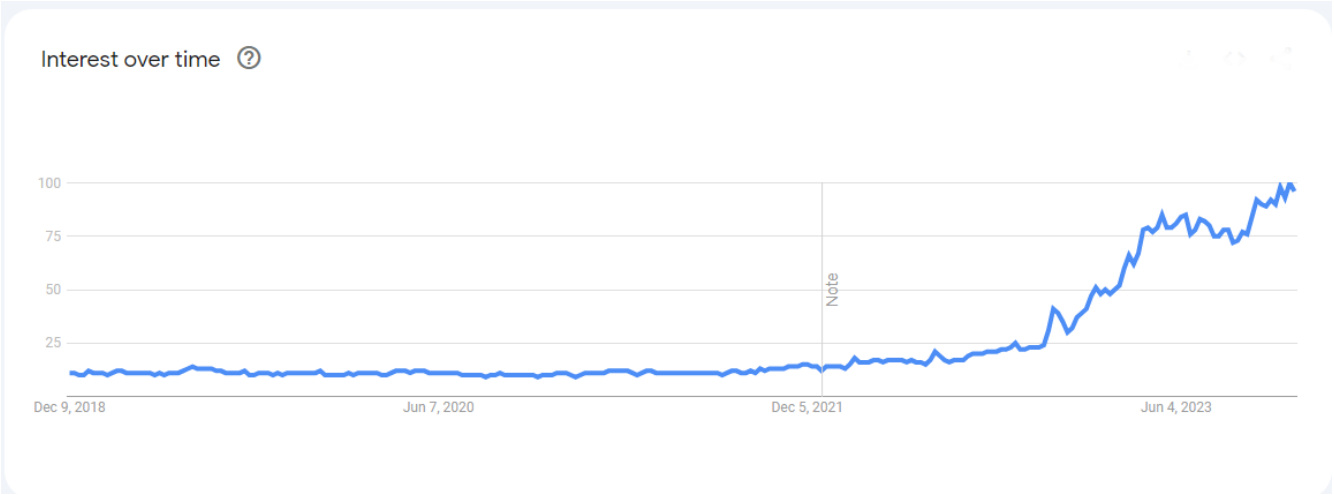

Рис 3.2. Зростання інтересу людей до машинного навчання (2018-2023).

В даний час ШІ розглядається як революційна технологія, схожа на появу комп'ютерів і смартфонів, яка протягом наступного десятиліття вплине на всі галузі промисловості[3].

Успішні комерційні застосування машинного навчання включають оптичне розпізнавання тексту, фільтрацію спаму електронної пошти, комп'ютерний зір, розпізнавання мови, машинний переклад, генерування синтетичних даних, виявлення аномалій, запобігання кіберзлочинів, виявлення шахрайських операцій з банківськими картами, прогнозування тимчасових рядів, пошукові системи, робототехніку, онлайн-рекламу, сентиментаналіз, аналіз фінансових ринків та багато іншого.

Існують дві основні методології машинного навчання: навчання з учителем (supervised learning) або контрольоване навчання і навчання без вчителя

(unsupervised learning) або неконтрольоване навчання (самонавчання). Є ще безліч методик, які є своєрідними містками між ними.

При навчанні з учителем інтелектуальний агент має доступ до міток або маркерів, які можуть бути використані для покращення продуктивності в ряді завдань. Мітки цінні тим, що вони допомагають ІІ відокремлювати спам від інших повідомлень під час навчання з учителем[9].

У разі навчання без вчителя мітки відсутні. Тому завдання ШІ не є чітко визначеним, що ускладнює точний вимір ефективності навчання. Знову звернемося до завдання фільтрації спаму, але цього разу без використання міток. Тепер інтелектуальний агент намагатиметься зрозуміти базову структуру електронних повідомлень, розбиваючи дані на групи, у кожному з яких повідомлення подібні між собою, але від повідомлень з інших груп[3].

Завдання навчання без вчителя формулюється менш чітко проти навчанням з учителем, і інтелектуальному агенту важче її вирішувати, й те водночас добре продуманий план дій дозволяє отримувати потужніші рішення. Іншими словами, оскільки завдання не задано суворо, інтелектуальний агент може виявити нові цікаві закономірності, крім тих, які ми спочатку намагалися знайти.

В наведеній нижче таблиці наведено порівняння навчання з вчителем та без вчителя.

Таблиця 3.1

Порівняння методів навчання.

Особливості	**Навчання без вчителя**	**Навчання з вчителем**
1	2	3
Тип даних	Без нагляду (наприклад, кластеризація, зниження розмірності)	З наглядом (використання маркованих даних)

1	2	3
Принцип роботи	Алгоритми знаходять патерни в даних самостійно	Використання маркованих даних та експертних знань
Приклади алгоритмів	Кластеризація K-means, аналіз головних компонент, автокодування	Нейронні мережі, рішення на основі дерева, метод опорних векторів
Переваги	Можливість роботи з непоміченими патернами, не вимагає маркованих даних	Висока точність при наявності маркованих даних, можливість використання експертних знань
Недоліки	Потребує більше даних для ефективного навчання, складніше визначити корисні патерни	Залежність від якості та обсягу маркованих даних, може бути складним для використання в складних завданнях

3.4. Бібліотека Scikit-learn.

Scikit-learn є однією з найпопулярніших бібліотек машинного навчання для Python. Вона розроблена на основі багатьох наукових бібліотек Python, таких як NumPy, SciPy та Matplotlib, та надає інтуїтивний інтерфейс для різних методів машинного навчання[35].

Відрізняється своєю універсальністю та простотою використання, проте вона може бути обмеженою для задач, пов'язаних з обробкою великих обсягів

даних через обмежену швидкість та можливості. Також, деякі складні алгоритми, такі як deep learning, не входять у численність її інтегрованих можливостей, хоча це можна компенсувати за допомогою інших спеціалізованих бібліотек[4].

Незважаючи на це, Scikit-learn відома своєю стабільністю та надійністю, що робить її популярним вибором серед дослідників, студентів та практикуючих спеціалістів у сфері машинного навчання. Її активне співтовариство та постійна підтримка забезпечують широкий спектр інструментів та можливостей для ефективної розробки та розв'язання різноманітних завдань з аналізу даних.

Scikit-learn містить широкий спектр алгоритмів машинного навчання, включаючи[4]:

1. Supervised Learning (Навчання з наглядом)

- Класифікація: Використовується для передбачення категорій або міток для даних. Наприклад, метод опорних векторів (SVM), навчання з використанням нейронних мереж, алгоритми на основі дерев рішень.
- Регресія: Використовується для передбачення числових значень. Наприклад, лінійна регресія, метод найменших квадратів, дерева рішень для регресії.

2. Unsupervised Learning (Навчання без нагляду)

- Кластеризація: Групування подібних елементів разом без маркованих міток. Наприклад, K-means, ієрархічна кластеризація.
- Зниження розмірності: Зменшення кількості змінних в наборі даних. Наприклад, аналіз головних компонент, t-distributed Stochastic Neighbor Embedding (t-SNE).

Прикладом використання може слугувати наведений нижче код.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
# Завантаження даних
iris = load_iris()
```

```
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2)
# Створення та навчання моделі
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# Прогнозування
predictions = knn.predict(X_test)
# Оцінка точності
accuracy = knn.score(X_test, y_test)
print(f'Точність моделі: {accuracy}')
```

У наведеному коді для прикладу використання бібліотеки Scikit-learn, передбачається розв'язання задачі класифікації для набору даних Iris. Код завантажує дані про квіти ірису з вбудованих даних в Scikit-learn, розбиває їх на навчальний та тестувальний набори, навчає модель класифікатора k-найближчих сусідів (KNeighborsClassifier) на навчальних даних та виконує прогнози на тестувальних даних. Код самостійно завантажує дані про квіти ірису, тому жодних додаткових даних завантажувати не потрібно.

В результаті виконання коду ми отримаємо точність моделі на тестувальному наборі даних. Це число вказує, наскільки точно модель змогла зробити прогнози для нових, раніше не бачених даних, в даному випадку для квітів ірису. Точність можна використовувати для подальшого аналізу або порівняння з іншими моделями[36].

Виходячи з усіх даних можемо навести талицю переваг та недоліків цієї бібліотеки (табл. 3.2)

Таблиця 3.2

Переваги та недоліки бібліотеки Scikit-learn

Переваги	**Недоліки**
1	2
Простий у використанні	Може бути обмеженістю для складних завдань
Широкий вибір алгоритмів	Обмежена підтримка глибокого навчання
Добре документована та підтримувана	Потребує вручну налаштовувати гіперпараметри моделей
Відмінна сумісність з іншими бібліотеками	Може вимагати додаткового оптимізованого коду для обробки великих обсягів даних
Ефективна реалізація алгоритмів	Немає підтримки для вбудованих нейронних мереж
Зручний інтерфейс для вивчення та розробки моделей	-

Можна також більш детально пройтись по недолікам нашої бібліотеки:

1. Обмеженість у складних завданнях. Scikit-learn може бути обмеженою для деяких складних завдань машинного навчання, зокрема у глибокому навчанні.

2. Необхідність вручну налаштовувати гіперпараметри. Деякі моделі вимагають ретельного та вручного налаштування гіперпараметрів для досягнення кращої точності.

3. Обмежена підтримка для глибокого навчання У випадку задач, де потрібна глибока нейронна мережа або складна модель, може бути доцільно використовувати TensorFlow або PyTorch.

Scikit-learn, безсумнівно, є потужним інструментом для різноманітних завдань машинного навчання. Однак вона може мати деякі обмеження, особливо

у випадку складних завдань, які потребують глибокого навчання або обробки великих обсягів даних. Також, хоча вона має великий вибір алгоритмів, іноді для досягнення кращої точності моделі може знадобитися налаштування гіперпараметрів вручну.

Не зважаючи на це, її зручний інтерфейс, ефективна реалізація алгоритмів та здатність працювати з іншими бібліотеками Python роблять її дуже корисним інструментом для багатьох аналітичних та наукових задач.

3.5. Бібліотеки google.oauth2 та GoogleApiClient.

Google API Client є ключовою бібліотекою для взаємодії з різноманітними сервісами Google через їх API. Ця бібліотека забезпечує розробникам можливість доступатися до різноманітних сервісів Google, таких як Gmail, Google Drive, YouTube, Google Maps та багатьох інших, використовуючи стандартні інтерфейси API[31].

Основні функціональність та можливості Google API Client:

1. Доступ до різноманітних сервісів Google:

Google API Client надає можливість взаємодіяти з різноманітними сервісами Google через їх API, від електронної пошти до візуалізації даних на картах.

2. Різноманітність API:

Бібліотека має розширений набір API, які покривають широкий спектр сервісів Google, що дозволяє розробникам використовувати відповідні API для своїх потреб.

3. Документація та Підтримка:

Google API Client має добре документовану та підтримувану інфраструктуру, що дозволяє розробникам швидко засвоїти роботу з різними API.

4. Регулярні Оновлення:

Google активно оновлює та підтримує свої API, що забезпечує відповідність та зручність використання бібліотеки для розробників.

Нижче буде наведено таблицю переваг та недоліків, які стосуються бібліотеки Google API Client (табл. 3.3).

Таблиця 3.3

Переваги та недоліки бібліотеки Google API Client

Переваги	**Недоліки**
1	2
Зручний доступ до різних сервісів Google	Складнощі у використанні для початківців
Різноманіття можливостей API для розробників	Може знадобитися додаткове вивчення документації
Добре документована та підтримувана	Обмеженість функціоналу у певних API
Регулярні оновлення та підтримка від Google	Не завжди повний контроль над даними користувачів

Більш детально розглянемо кожну з переваг:

1. Зручний доступ до сервісів Google:

Google API Client забезпечує простий та зрозумілий інтерфейс для взаємодії з різноманітними сервісами Google через їх API. Але, якщо вам потрібно щось більш адаптоване до специфічних потреб вашого проекту, бібліотека Python requests може надати більше гнучкості та контролю над HTTP-запитами.

2. Різноманітність можливостей:

Google API Client має широкий спектр API, що охоплюють різні сервіси Google. Однак, якщо потрібно взаємодіяти з API інших постачальників, бібліотеки, такі як requests або httpx, можуть забезпечити універсальний інструментарій для роботи з будь-якими API.

3. Документація та підтримка:

Документація Google API Client добре структурована, а підтримка зі сторони Google забезпечує надійність та актуальність. Тим не менш, бібліотеки Python мають власні спільноти та ресурси, такі як Stack Overflow, що дозволяють швидко отримати відповіді на питання та рішення проблем.

4. Регулярні оновлення:

Google постійно оновлює та підтримує свої API, що забезпечує стабільність та актуальність Google API Client. Але, якщо потрібно мати більший контроль над версіями та підтримкою, бібліотека requests може надати більшу гнучкість у керуванні версіями API та їх оновленнями.

Тепер детальніше пройдемось по недолікам:

1. Складність для початківців:

Google API Client може бути складним для освоєння для новачків. У цьому випадку, бібліотеки Python, такі як requests, мають більш простий інтерфейс та легше освоюються початківцями.

2. Потреба в додатковому вивченні:

Деякі API можуть вимагати додаткового вивчення для розуміння їх можливостей та параметрів. У цьому випадку, бібліотеки Python дозволяють налаштувати кожен аспект HTTP-запиту вручну, що може бути корисним для більш глибокого контролю.

3. Обмеженість функціоналу:

В деяких випадках певні API можуть мати обмежені можливості, що може ускладнити розробку певних функціональних аспектів. Бібліотеки Python можуть надати більшу гнучкість у вирішенні специфічних завдань.

4. Не gовний rонтроль над даними користувачів:

При використанні деяких API через Google API Client може виникати обмежений контроль за обробкою даних користувачів. У цьому випадку, бібліотеки Python можуть дозволити більший контроль за обробкою та зберіганням даних.

Бібліотеки Python, такі як requests, httpx та інші, можуть бути корисними альтернативами для роботи з API, зокрема для більш гнучкого керування HTTP-запитами та обробки даних. Однак Google API Client є потужним інструментом для роботи з різноманітними сервісами Google через їх API і забезпечує зручний інтерфейс для взаємодії з ними[31].

Можемо також розглянути основні функції для бібліотеки Google API Client в мові програмування Python. Для ініціалізації зазвичай використовують наступний запис:

build(serviceName, version, credentials=None)

Ця функція використовується для створення об'єкту служби Google API. Вона дозволяє взаємодіяти з певними службами Google, використовуючи API Client Library. Параметри:

- serviceName: назва служби Google API (наприклад, "drive", "calendar" і т.д.).
- version: версія API.
- credentials: об'єкт, який містить автентифікаційні дані (OAuth 2.0 токени, ключі API тощо).

Прикладом використання на практиці може слугувати наступний код:

```
from googleapiclient.discovery import build
# Авторизація та ініціалізація служби Google Drive API
def initialize_drive_service():
    creds = authorize_user()
    service = build('drive', 'v3', credentials=creds)
    return service
# Приклад виклику методу API (наприклад, отримання списку файлів)
def list_files():
    drive_service = initialize_drive_service()
    results = drive_service.files().list(pageSize=10).execute()
    items = results.get('files', [])
    if not items:
```

```
            print('No files found.')
        else:
            print('Files:')
            for item in items:
                print(f'{item["name"]} ({item["id"]})')
```

У цьому прикладі використовується googleapiclient.discovery.build для створення об'єкту служби Google Drive API та ініціалізації роботи з цим API. Код:

1. Визначення функції initialize_drive_service(), яка повертає ініціалізований об'єкт служби Google Drive API.

2. Виклик методу list() для отримання списку файлів у Google Drive.

3. Виведення переліку файлів на екран.

У цьому випадку, функція list_files() використовує об'єкт служби Google Drive API, що був створений попередньо, та викликає метод API для отримання списку файлів. Після чого відбувається обробка отриманих результатів[31].

Тепер що стосується протоколу *Google OAuth2*. OAuth 2.0 - це протокол авторизації, розроблений для делегування доступу користувача до ресурсів іншому сервісу без розкриття пароля. Він дозволяє користувачам надавати обмежений доступ до своїх ресурсів (таких як фотографії, контакти, документи тощо) іншим додаткам чи сервісам через веб-протоколи[33].

До основних компонентів OAuth 2.0 можна віднести:

1. Resource Owner (Власник ресурсу). Користувач, який надає доступ до своїх ресурсів.

2. Client (Клієнт). Додаток чи сервіс, який отримує доступ до ресурсів користувача.

3. Authorization Server (Сервер авторизації). Система, що авторизує клієнта за допомогою ідентифікаційних даних користувача та надає токени доступу.

4. Resource Server (Сервер ресурсів). Сервер, що зберігає захищені ресурси користувача та приймає запити на доступ до них.

Тепер розглянемо основні принципи OAuth 2.0[33]:

1. Client Registration (Реєстрація клієнта): Клієнт реєструється на сервері авторизації для отримання ідентифікатора та секретного ключа.

2. Authorization Grant (Підтвердження авторизації): Користувач надає доступ до своїх ресурсів клієнту за допомогою авторизаційного запиту.

3. Token Generation (Генерація токенів): Після успішної авторизації сервер авторизації генерує токени доступу, які надаються клієнту для отримання ресурсів.

4. Access to Protected Resources (Доступ до захищених ресурсів): Клієнт використовує отримані токени доступу для отримання доступу до ресурсів власника.

OAuth 2.0 використовується в багатьох популярних сервісах, таких як Google, Facebook, Twitter та інші, для надання доступу до ресурсів користувачів стороннім додаткам.

Нижче наведемо таблицю про переваги та недоліки протоколу (табл. 3.4).

Таблиця 3.4

Переваги та недоліки протоколу Oauth 2.0

Переваги	**Недоліки**
1	2
Забезпечує безпеку та конфіденційність	Відсутність повної стандартизації
Дозволяє делегувати доступ до ресурсів	Ризик використання неправильної конфігурації
Легко інтегрується з багатьма сервісами	Потребує ретельної уваги при розробці
Забезпечує обмежений доступ до ресурсів	Проблеми з безпекою при необережному використанні

Можемо детальніеш пройтись по кожній з переваг:

1. Забезпечує безпеку та конфіденційність. OAuth 2.0 забезпечує безпеку та конфіденційність, оскільки не передає паролі користувачів сервісам-клієнтам, а надає обмежений доступ за допомогою токенів доступу.

2. Дозволяє делегувати доступ до ресурсів. Користувач може делегувати обмежений доступ до своїх ресурсів стороннім додаткам чи сервісам без необхідності розкривати свій пароль.

3. Легко інтегрується з багатьма сервісами. OAuth 2.0 широко підтримується багатьма популярними сервісами та додатками, що полегшує інтеграцію та використання для різноманітних платформ.

4. Забезпечує обмежений доступ до ресурсів. Протокол надає можливість обмеженого доступу до ресурсів, що дозволяє користувачам контролювати, які дані можуть бути доступні стороннім додаткам.

Тепер детальніше розглянемо кожен з недоліків протоколу:

1. Відсутність повної стандартизації. OAuth 2.0 не має повної стандартизації, що може призвести до різних реалізацій та потенційних проблем у взаємодії між сервісами.

2. Ризик використання неправильної конфігурації. Неправильна конфігурація може призвести до проблем з безпекою, таких як витік токенів або недостатньо обмежений доступ до ресурсів.

3. Потребує ретельної уваги при розробці. Для безпечної реалізації OAuth 2.0 необхідно ретельно розробляти та налаштовувати систему авторизації та інтеграції з додатками.

4. Проблеми з безпекою при необережному використанні. Необачне використання може призвести до проблем з безпекою, таких як втрата токенів доступу чи несанкціонований доступ до ресурсів користувача.

Зараз можемо розглянути дві основних операції в протоколі Oauth 2.0 в мові програмування Python:

1. Авторизація користувача. Для авторизації користувача використовується OAuth 2.0. Потрібно здійснити наступні кроки:

- Отримання доступу до Google API через OAuth 2.0.

- Отримання авторизаційного токену для доступу до конкретної служби.

2. Отримання авторизаційних даних. Для отримання авторизаційних даних (токенів) для доступу до Google API можна використовувати бібліотеку google_auth_oauthlib[37].

Прикладом використання на практиці може слугувати наступний код:

```
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
import pickle
# Параметри для доступу до API
SCOPES = ['https://www.googleapis.com/auth/drive']
# Авторизація користувача
def authorize_user():
    creds = None
    try:
        with open('token.pickle', 'rb') as token:
            creds = pickle.load(token)
    except FileNotFoundError:
        flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
        creds = flow.run_local_server(port=0)
        with open('token.pickle', 'wb') as token:
            pickle.dump(creds, token)
    return creds
```

У цьому прикладі ми використовуємо бібліотеки google_auth_oauthlib для отримання авторизаційних даних користувача, необхідних для доступу до конкретної служби Google API, наприклад, Google Drive. Кроки:

1. Імпорт необхідних класів (InstalledAppFlow, Request, pickle).

2. Визначення області (SCOPES), яку потрібно запитати для доступу.

3. Отримання авторизаційних даних (токенів) через процес OAuth 2.0, використовуючи поточні облікові дані користувача або, при їх відсутності, створення нового токену.

Код детально реалізує цей процес, він перевіряє наявність файлу з токеном (token.pickle). Якщо файл з токеном відсутній, відбувається процес авторизації через локальний сервер та отримання токену. Після цього токен зберігається у файлі token.pickle для подальшого використання.

3.6. Бібліотека Tkinter.

Tkinter - це стандартна бібліотека Python, яка використовується для створення графічного інтерфейсу користувача (GUI). Вона базується на Tk, бібліотеці для роботи зі знаками прикладної програми Tcl/Tk для різних платформ[32].

Основні функції та можливості Tkinter:

1. Створення вікна і віджетів. Tkinter дозволяє створювати вікна та різні віджети, такі як кнопки, мітки, поле вводу, списки, рамки тощо, щоб створювати інтерактивний інтерфейс.

2. Розміщення віджетів. Бібліотека надає можливість розміщувати віджети у вікні за допомогою різних менеджерів розміщення, таких як pack(), grid() та place(), що дозволяє організувати вигляд вашого GUI.

3. Обробка подій. Tkinter дозволяє визначати функції-обробники подій, які виконуються при взаємодії користувача з віджетами, такими як натискання кнопок, введення тексту, мишкові події тощо.

4. Робота з графічними об'єктами. Бібліотека також дозволяє малювати графічні об'єкти, такі як лінії, криві, прямокутники тощо, за допомогою методів роботи з канвасом (Canvas).

Нижче наведемо приклад коду використання бібліотеки та її функцій в мові програмування Python:

```
import tkinter as tk
# Створення вікна
root = tk.Tk()
root.title("Мій перший GUI")
# Створення мітки
label = tk.Label(root, text="Вітаємо у Tkinter!", font=("Arial", 18))
label.pack(pady=20)
# Функція для обробки події натискання кнопки
def on_button_click():
    label.config(text="Кнопка була натиснута!")
# Створення кнопки
button = tk.Button(root, text="Натисни мене", command=on_button_click)
button.pack(padx=10, pady=10)
# Запуск циклу обробки подій
root.mainloop()
```

У цьому прикладі створюється вікно з міткою та кнопкою. При натисканні кнопки змінюється текст мітки. Tkinter простий у використанні та дозволяє швидко створювати базовий GUI для програм Python.

Можемо також відмітити й інші корисні функції бібліотеки[32]:

1. Робота зі шрифтами та текстом. Tkinter дозволяє налаштовувати шрифти, розмір, колір тексту та робити його вирізку, вирівнювання та інші форматувальні опції.

2. Меню та панелі інструментів. Бібліотека дозволяє створювати різноманітні меню, включаючи головне меню та контекстні меню, а також панелі інструментів.

3. Робота зі зображеннями. Tkinter може відображати зображення у вікні та дозволяє виконувати різні операції з ними, такі як зміна розміру, обрізка тощо.

Нижче можемо навести таблицю переваг та недоліків (табл. 3.5)

Таблиця 3.5

Переваги та недоліки бібліотеки Tkinter

Переваги	**Недоліки**
1	2
Простота використання	Обмежена можливість кастомізації інтерфейсу
Наявність у стандартній бібліотеці	Не володіє сучасними можливостями дизайну
Підтримка на різних платформах	Обмеженість можливостей відлагодження та оптимізації

Можемо більш конкретно переглянути кожну з переваг:

1. Наявність у стандартній бібліотеці. Tkinter є частиною стандартної бібліотеки Python, тому не вимагає окремого встановлення. Вона доступна для використання після встановлення Python, що робить її доступною для всіх користувачів.

2. Простота використання. Tkinter простий у використанні для створення базових інтерфейсів користувача. Вона надає зручний спосіб створення вікон, кнопок, полів вводу та інших віджетів.

3. Підтримка на різних платформах. Tkinter є переносним та працює на різних платформах (Windows, macOS, Linux), що робить його універсальним інструментом для створення GUI.

Тепер розглянемо кожен з недоліків:

1. Обмежена можливість кастомізації інтерфейсу. Tkinter має обмежені можливості налаштування зовнішнього вигляду та дизайну інтерфейсу порівняно з деякими іншими бібліотеками.

2. Не володіє сучасними можливостями дизайну. У порівнянні з сучасними стандартами дизайну GUI, Tkinter може виглядати менш сучасно, оскільки вона не має таких розширених можливостей дизайну[39].

3. Обмеженість можливостей відлагодження та оптимізації. Tkinter може бути обмеженим у можливостях відлагодження (debugging) та оптимізації за вимогами сучасних інтерфейсів користувача, що може бути важливим у великих проектах.

3.7. Висновки до третього розділу.

У цьому розділі ми розглянули широкий спектр питань, пов'язаних із програмуванням та інструментами розробки, зосередившись на Python, бібліотеці Scikit-learn для машинного навчання, протоколі OAuth 2.0 та бібліотеці Tkinter для створення графічних інтерфейсів користувача.

Python займає одне з лідируючих місць серед мов програмування завдяки своїй простоті в освоєнні, широкому спектру застосувань та великій кількості бібліотек для різноманітних завдань. Мова володіє читабельним синтаксисом та має велику спільноту, що сприяє розвитку та підтримці.

У порівнянні між навчанням з вчителем та без вчителя, ми визначили, що обидва підходи мають свої переваги та недоліки. Навчання з вчителем надає більшу структурованість, глибшу розуміння та можливість отримати відповіді на запитання в реальному часі. У той же час, навчання без вчителя спрямоване на самостійність та дозволяє глибше освоїти матеріал за допомогою власних досліджень та практики.

Щодо розглянутих бібліотек:

- Scikit-learn вражає своєю розширеністю, дозволяючи легко працювати з алгоритмами машинного навчання та статистики.
- OAuth 2.0 є потужним протоколом для авторизації, але потребує уважного розгляду його можливостей та потенційних ризиків.
- Tkinter, хоча є стандартним інструментом для створення GUI в Python, має свої обмеження у кастомізації та сучасному дизайні.

Загалом, обираючи мову програмування або інструменти для конкретних завдань, важливо враховувати їхні можливості та обмеження, щоб забезпечити ефективність та досягнення поставлених цілей у розробці програмного забезпечення.

РОЗДІЛ 4. ПРОГРАМНИЙ ДОДАТОК ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ.

4.1. Розробка програмного забезпечення

Програма розроблялась з метою автоматизації процесу виявлення можливих загроз у повідомленнях електронної пошти. Основною метою є розпізнавання прийомів соціальної інженерії, які використовуються зловмисниками для отримання доступу до конфіденційної інформації користувачів.

До основних вимоги до програми можемо віднести наступні пункти:

1. Аналіз тексту: Програма повинна аналізувати текстову частину електронної пошти для виявлення можливих прийомів соціальної інженерії, таких як спроби фішингу, прохання надати особисту інформацію, та інші типи маніпуляцій користувачем.

2. Виявлення підозрілих шаблонів: Програма повинна використовувати навчання без учителя для виявлення підозрілих шаблонів у тексті, які можуть вказувати на наявність соціально-інженерних атак.

3. Графічний інтерфейс: Розробка програми з графічним інтерфейсом, який спрощує користувачеві взаємодію з програмою та надає зручність використання.

4. Обробка електронних листів: Програма має взаємодіяти з електронною поштою, здатна витягувати тексти листів для їх подальшого аналізу.

5. Навчання моделі: Використання бібліотек для машинного навчання (scikit-learn) для побудови моделі, яка може виявляти підозрілі ситуації у тексті.

Також при розробці були й певні обмеження:

1. Обмежені ресурси: В процесі розробки важливо врахувати обмежені ресурси, такі як обсяг доступної пам'яті та обробки даних.

2. Інтерфейс та користувацький досвід: Необхідно забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувача, щоб полегшити взаємодію з програмою.

3. Точність аналізу: Модель аналізу тексту повинна бути достатньо точною для виявлення підозрілих шаблонів у повідомленнях електронної пошти.

4. Час обробки: Програма повинна працювати достатньо швидко для обробки великого обсягу даних без значного зниження продуктивності.

Задля коректної реалізації навчання без вчителя для початку було необхідно створити навчальний датасет для нашої нейронної мережі (рис. 4.1).

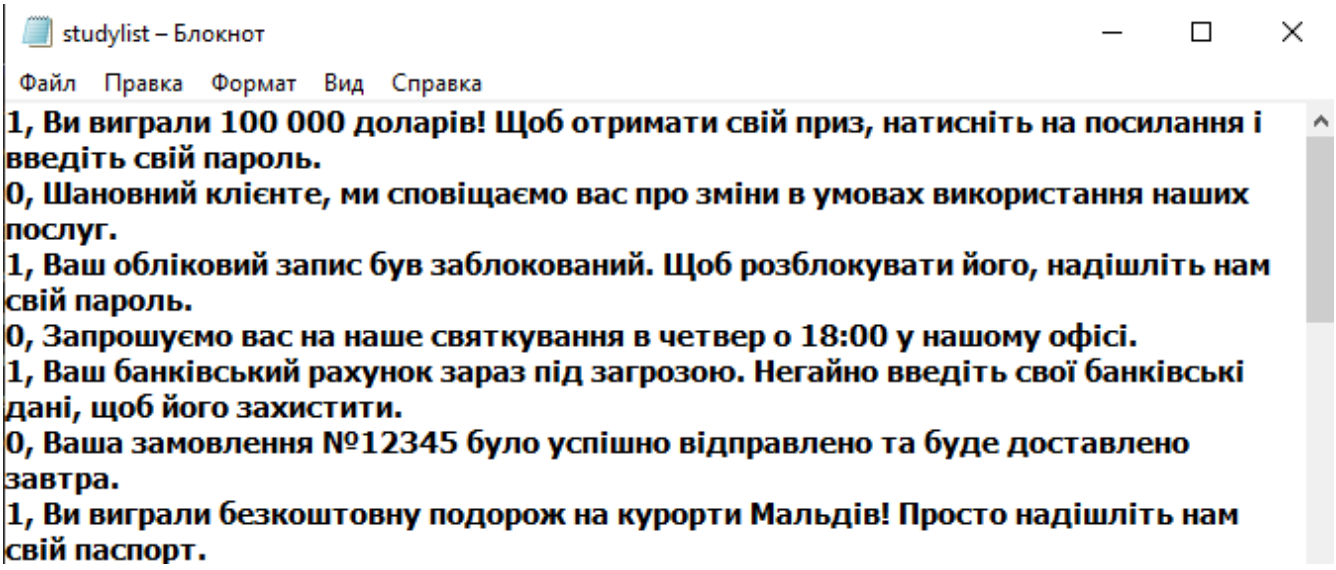

Рис. 4.1. Навчальний датасет studylist.txt

На даному скріншоті наведено лише малу частину навчальної інформації. Логіка була досить простою:

- 1 – в повідомленні наявні прийоми соціальної інженерії;
- 0 – повідомлення є безпечним.

Що ж до завантаження датасету в саме тіло програми, то за цей етап відповідає наступна функція, яка стає нам доступною після імпортування бібліотеки os:

```
with open('studylist.txt', 'r', encoding='utf-8') as file:
    lines = file.readlines()
```

Після цього необхідно було сформувати датасет так, щоб нейронна мережа розуміла яким саме чином їй необхідно навчатись. Для цього був написаний код для розділення мітки та тексту повідомлення:

```
labels = []
texts = []
for line in lines:
    parts = re.split(r',\s*', line.strip())
    labels.append(int(parts[0]))
    texts.append(parts[1])
```

Далі необхідно забезпечити процес векторизації тексту та навчання класифікатора:

```
# Векторизація тексту
vectorizer = TfidfVectorizer()
X_vectorized = vectorizer.fit_transform(texts)
# Навчання класифікатора
classifier = MultinomialNB()
classifier.fit(X_vectorized, labels)
```

Виходячи з цього вже можна було сформувати функцію для аналізу тексту:

```
def analyze_email(email_text):
   # Векторизація тексту
   text_vectorized = vectorizer.transform([email_text])
   # Використання навченого класифікатора для передбачення
   prediction = classifier.predict(text_vectorized)
   return prediction[0]
```

Можемо детально розглянути роботу даної функції:

1. Отримання тексту листа. Функція отримує текст електронного листа, який буде проаналізовано для виявлення прийомів соціальної інженерії.

2. Векторизація тексту. Використовуючи попередньо навчений векторизатор (vectorizer), функція перетворює отриманий текст листа в числовий вектор, який можна використовувати для подальшого аналізу.

3. Передбачення за допомогою класифікатора. Векторизований текст листа подається на вхід попередньо навченому класифікатору (classifier). Цей класифікатор робить прогноз щодо того, чи містить текст листа прийоми соціальної інженерії чи ні.

4. Повернення результату. Результат передбачення (1 або 0) повертається з функції. Це значення вказує на те, чи виявлені прийоми соціальної інженерії в тексті листа.

Потім було вирішено написати код для функції, яка завантажує дані про лист із сервісу Gmail, аналізує його текст на наявність прийомів соціальної інженерії, та виводить деталі листа у вікні програми. Це включає інформацію про відправника, дату, текстову частину листа, наявність вкладеного файлу та вирок щодо прийомів соціальної інженерії. Функція має назву def show_email_info(). Код для неї та опис кожного блоку буде наведено нижче.

Підготовка оточення та отримання інформації про листа. В цьому блоку зчитуються значення введених електронної пошти та шляху до файлу з обліковими даними, ініціалізується об'єкт сервісу Gmail API, а також створюється пустий рядок all_info для подальшого відображення інформації про листи.:

```
def show_email_info(msg_id):
    # Отримання введених даних: email та шляху до файлу credentials
    email = email_entry.get()
    credentials_path = credentials_entry.get()
    # Завантаження облікових даних та створення сервісу Gmail
    creds = service_account.Credentials.from_service_account_file(credentials_path, ['https://www.googleapis.com/auth/gmail.readonly'])
    service = build('gmail', 'v1', credentials=creds)
```

```
            all_info = ""  # Ініціалізація змінної для зберігання інформації про лист

          Отримання повного тексту листа та інформації про відправника та дату:
            try:
        # Отримання повного тексту листа за його ID
        msg = service.users().messages().get(userId=email, id=msg_id,
format='full').execute()
        payload = msg['payload']
        # Отримання інформації про відправника та дату листа з його заголовків
        headers = payload['headers']
        sender_name = [header['value'] for header in headers if header['name'] == 'From']
        sender_name = sender_name[0] if sender_name else 'Unknown'
        date = [header['value'] for header in headers if header['name'] == 'Date']
        date = date[0] if date else 'Unknown'

          Отримання тексту листа:
           email_text = "No text part found in this email."  # За замовчуванням -
          повідомлення про відсутність тексту
        # Функція для декодування текстових даних з base64
        def decode_text(text):
            return base64.urlsafe_b64decode(text.encode('ASCII')).decode("utf-8")
        # Функція для перевірки наявності текстової частини листа
        def check_parts(parts):
            nonlocal email_text
            for part in parts:
                if 'parts' in part:
                    check_parts(part['parts'])
                elif 'mimeType' in part and part['mimeType'] == 'text/plain':
                    email_text = decode_text(part['body']['data'])
                    break
```

```
# Пошук текстової частини листа
check_parts(payload.get('parts', []))
```

Додавання основної інформації про лист:

```
all_info = f"Email ID: {msg_id}\nSender Name: {sender_name}\nDate: {date}\nEmail Text:\n{email_text}\n"
```

Перевірка наявності вкладеного файлу та вивід його інформації:

```
# Перевірка наявності вкладення та додавання інформації про вкладений файл
        if 'parts' in payload:
            for part in payload['parts']:
                if 'filename' in part:
                    attachment_info = f"Attached file: {part['filename']}\n"
                    all_info += attachment_info
```

Перевірка наявності прийомів соціальної інженерії:

```
        # Передбачення наявності прийомів соціальної інженерії в тексті листа
        prediction = analyze_email(email_text)
        if prediction == 1:
            all_info += "Potential social engineering attempt detected in this email.\n----------------------------------------------\n"
        else:
            all_info += "No signs of social engineering in this email.\n----------------------------------------\n"
```

Потім була написана функція для коректного відображення листів та тексту в них. Функція scan_emails() виконує наступні дії:

1. Приховання віджетів вводу електронної пошти та файлу з обліковими даними. Функція викликає grid_forget() для кожного віджету, пов'язаного з

введенням електронної пошти, файлу з обліковими даними та кнопки "Scan", приховуючи їх з вікна програми. Це дозволяє зосередитись тільки на відображенні інформації про листи без відображення вхідних полів.

Блок коду:

```
email_label.grid_forget()
email_entry.grid_forget()
credentials_label.grid_forget()
credentials_entry.grid_forget()
browse_button.grid_forget()
scan_button.grid_forget()
```

2. Отримання доступу до Gmail API та списку листів користувача. Функція використовує введену адресу електронної пошти та шлях до файлу з обліковими даними для створення авторизованого з'єднання із сервісом Gmail API. Після цього вона отримує список листів користувача за допомогою Gmail API.

3. Створення вікна для відображення списку листів з прокруткою

Функція створює вікно, що має вбудовану можливість прокрутки. Це вікно буде вміщувати кнопки для кожного листа та забезпечувати можливість прокрутки у разі великої кількості листів.

Блок коду:

```
canvas = tk.Canvas(root)
canvas.grid(row=1, column=0, columnspan=3, padx=5, pady=5, sticky='news')

scrollbar = tk.Scrollbar(root, orient="vertical", command=canvas.yview)
scrollbar.grid(row=1, column=3, sticky='ns')
frame = tk.Frame(canvas)
canvas.create_window((0, 0), window=frame, anchor='nw')
canvas.configure(yscrollcommand=scrollbar.set)
canvas.bind('<Configure>', lambda e: canvas.configure(scrollregion=canvas.bbox("all")))
```

4. Створення кнопок для кожного листа та їх відображення

Для кожного листа зі списку створюється окрема кнопка, яка містить інформацію про лист (ID, відправника, дату тощо). При кліку на цю кнопку викликається функція show_email_info(), яка відображає детальну інформацію про вибраний лист.

Блок коду:

```
for message in messages:
    msg_id = message['id']
    msg = service.users().messages().get(userId=email, id=msg_id, format='full').execute()
    payload = msg['payload']
    headers = payload['headers']
    sender_name = [header['value'] for header in headers if header['name'] == 'From']
    sender_name = sender_name[0] if sender_name else 'Unknown'
    date = [header['value'] for header in headers if header['name'] == 'Date']
    date = date[0] if date else 'Unknown'
    short_id = shorten_id(msg_id)
    btn_text = f"ID: {short_id}, From: {sender_name}, Date: {date}"
    btn = tk.Button(frame, text=btn_text, command=lambda id=msg_id: show_email_info(id))
    btn.pack(fill='x', padx=5, pady=2)
```

5. Відображення інформації про листи

На завершення функція оновлює result_label, де відображається інформація про виявлені листи. Це може бути повідомлення про відсутність листів або ж сам список листів для подальшого вивчення.

Блок коду:

```
result_label.config(text=all_info)
```

Після завершення основної роботи з Gmail API та навчанням нейронної мережі було прийнято рішення перейти до роботи з вікном додатку. Використовувалась бібліотека Tkinter. Основні компоненти коду наведені нижче:

1. Створення вікна програми.

- root = tk.Tk() - створення головного вікна програми;
- root.title("Gmail Scanner") - встановлення заголовка вікна.

2. Створення віджетів (елементів) інтерфейсу.

- tk.Label: елемент для відображення тексту;
- tk.Entry: поле для введення користувацьких даних;
- tk.Button: кнопка для виклику певної дії.

У нашому випадку використовуються Label для підписів, Entry для введення адреси електронної пошти та шляху до файлу, і Button для пошуку файлу та запуску сканування.

3. Розміщення елементів у вікні.

Команда grid() використовується для розміщення елементів на вікні програми. row та column вказують на рядок та колонку, де розміщується елемент. padx та pady визначають відступи по горизонталі та вертикалі від меж елементів.

4. Призначення функцій для кнопок.

- command= вказує на функцію, яка буде викликана при натисканні кнопки.

5. Основний цикл програми.

- root.mainloop() - цей метод запускає головний цикл обробки подій Tkinter, що дозволяє вікну реагувати на дії користувача (наприклад, кліки мишею або натискання клавіш).

4.2. Тестування програмного забезпечення.

Після завершення написання коду необхідно було також протестувати роботу нашого ПЗ. Задля початку роботи з програмою необхідно отримати файл JSON для роботи з обліковими даними.

JSON (JavaScript Object Notation) - це легкий формат обміну даними, який використовується для обміну структурованою інформацією між різними системами. Він базується на синтаксисі JavaScript, але став загально прийнятим у багатьох мовах програмування.

Основні особливості JSON:

1. Синтаксис. JSON складається з пар ключ-значення, які можуть бути вкладені одне в одне. Дані представляють собою або об'єкт (набір пар ключ-значення в фігурних дужках), або масив (впорядкований список значень у квадратних дужках).

2. Легкість читання та запису. JSON легко читається як людьми, так і машинами. Він досить компактний та легкий для розуміння людиною.

3. Використання. JSON широко використовується для обміну даними між веб-серверами та клієнтами, для зберігання конфігурацій та налаштувань у файловій формі, для передачі даних у API та для багатьох інших цілей.

4. Підтримка мовами програмування. Майже всі сучасні мови програмування мають бібліотеки для роботи з JSON, що дозволяє легко обробляти дані в цьому форматі.

5. Сумісність з веб-технологіями. JSON є дуже популярним у веб-розробці, оскільки легко інтегрується з JavaScript, що дає можливість зручно взаємодіяти з AJAX запитами та обмінюватися даними між веб-сервером і клієнтом.

JSON став важливим стандартом для обміну даними у багатьох сферах інформаційних технологій та знаходить широке застосування у різних областях програмування та розробки програмного забезпечення.

Для отримання файлу JSON необхідно було пройти наступні кроки:

1. Створити проект в Google Cloud Platform (GCP):

- Перейти на сторінку Google Cloud Console (рис 4.2);
- У верхньому лівому куті вибрати або створити проект;
- У меню навігації обрати "API & Services" -> "Dashboard";
- Обрати "Enable APIs and Services", знайдіть Gmail API та увімкнути його.

2. Створити обліковий запис служби (Service Account):

- У меню навігації обрати "API & Services" -> "Credentials";
- Обрати "Create credentials" -> "Service account";
- Заповнити обов'язкові поля та обрати роль для облікового запису;
- Натиснути "Continue", а потім "Done".

3. Завантажити файл ключа для облікового запису:

- У списку служби облікового запису оберіть створений обліковий запис;
- Перейти на вкладку "Keys" та оберіть "Add key" -> "Create new key";
- Обрати JSON як формат та натисніть "Create";
- Файл credentials.json було завантажено на комп'ютер (рис 4.3).

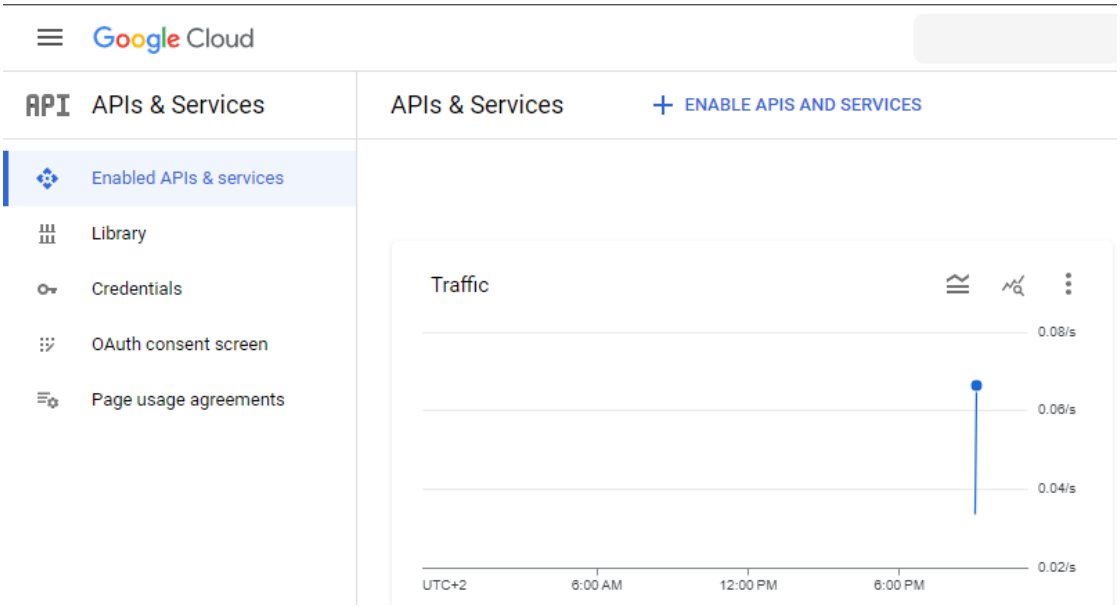

Рис. 4.2. Стартове вікно Google Cloud Console.

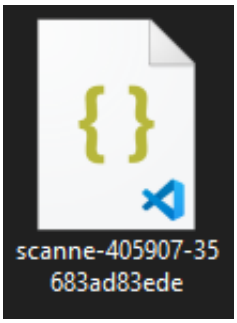

Рис. 4.3. Завантажений файл з обліковими даними.

Тепер, маючи файл з обліковими даними JSON, можна переходити до роботи в самому віконному додатку. Після запуску нас зустрічає наступний інтерфейс(Рис 4.4). В ньому користувачу буде запропоновано ввести дані про свою пошту та вказати файл зі своїми обліковими даними.

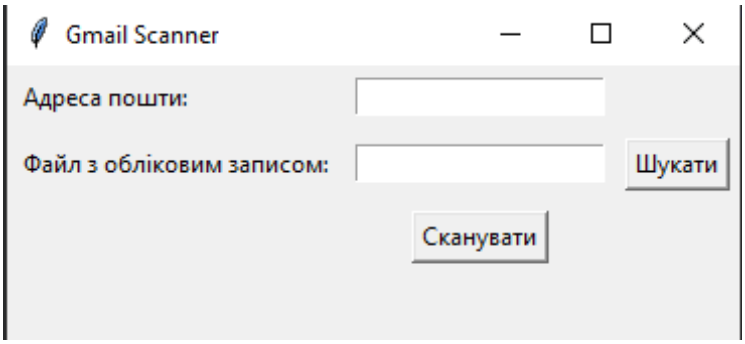

Рис. 4.4. Початковий інтерфейс програми.

Після введення всіх необхідних даним нас буде зустрічати список усіх доступних листів на пошті (рис. 4.5). Задля порівняння результатів також буде наведено скріншот з пошти Gmail (рис. 4.6). Як бачимо і там, і там в нашому списку мається 3 повідомлення. На тестову пошту були надіслані повідомлення для перевірки коректності сканування листів.

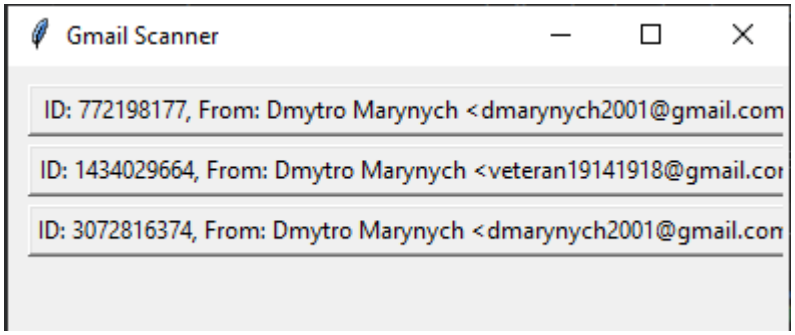

Рис. 4.5. Список листів в додатку.

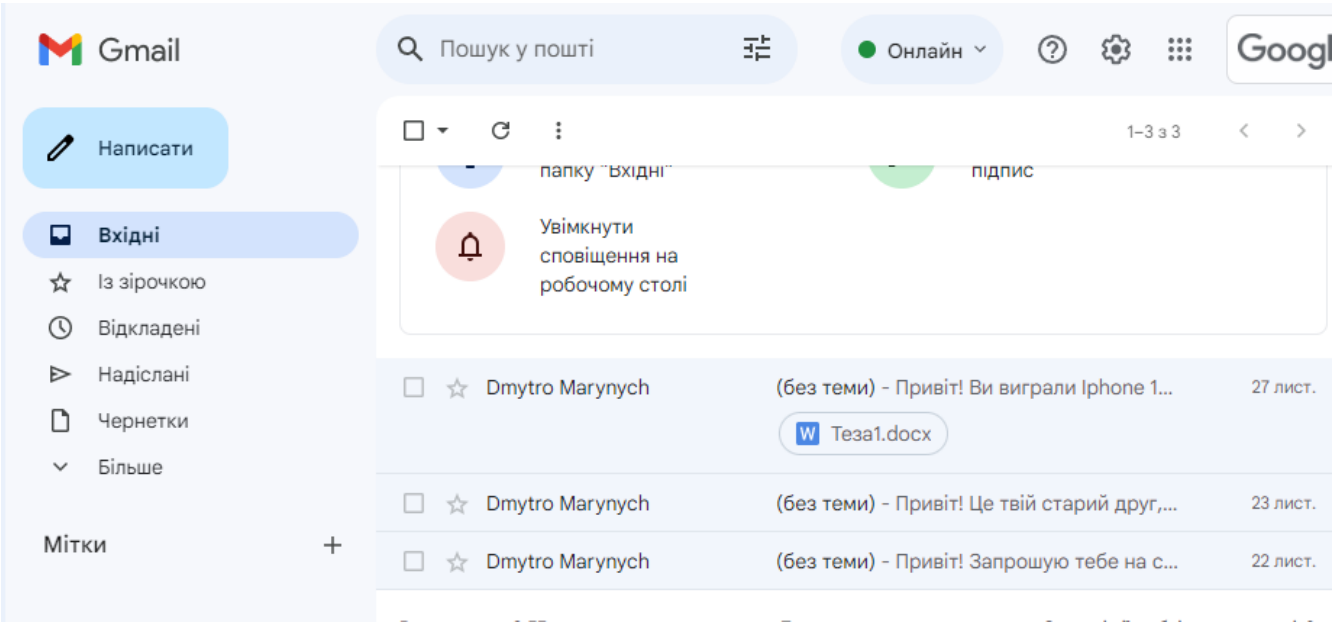

Рис. 4.6. Список повідомлень в Gmail.

Можна побачити, що при появі списку листів в додатку нам видає наступну інформацію: ідентифікаційний номер листа, ім’я відправника та адреса пошти. У більшості систем управління поштою (наприклад, Gmail) кожен лист має свій унікальний ідентифікатор (ID). Цей ідентифікатор — це унікальний номер, який присвоюється кожному листу в системі. ID листа зазвичай використовується для внутрішньої адресації та посилань на конкретний лист в межах поштової скриньки.

Зазвичай ID листа генерується системою автоматично при його створенні. Це унікальне значення, яке відрізняє кожен лист від інших. Іноді ID листа може бути використаний для доступу до конкретного листа через API, що дозволяє працювати з поштою програмно.

ID листа може мати різну довжину та формат залежно від конкретної поштової системи, але його основна функція — забезпечити унікальність та ідентифікацію кожного листа в поштовій скриньці.

При натисканні на лист в додатку нас буде зустрічати список з усією інформацією, текст листа та рішення про наявність в ньому прийомів соціальної інженерії (рис. 4.7). Для порівняння також буде показаний те й же лист, але в Gmail (рис. 4.8).

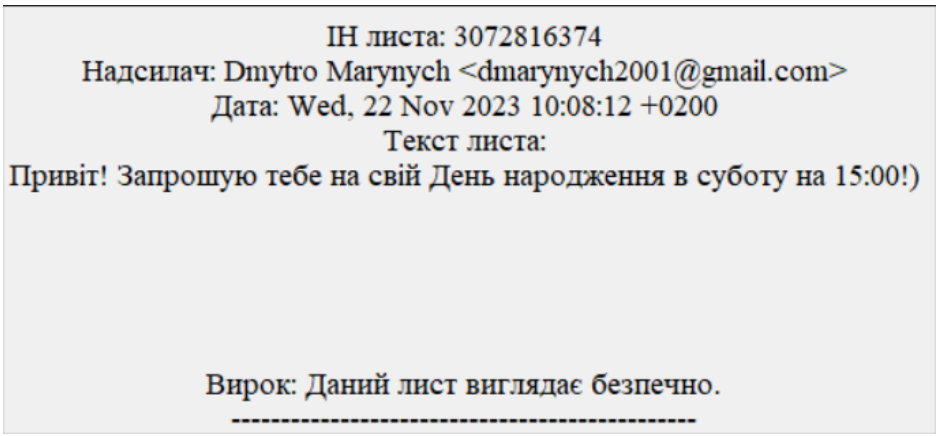

Рис. 4.7. Інформація про лист та висновок.

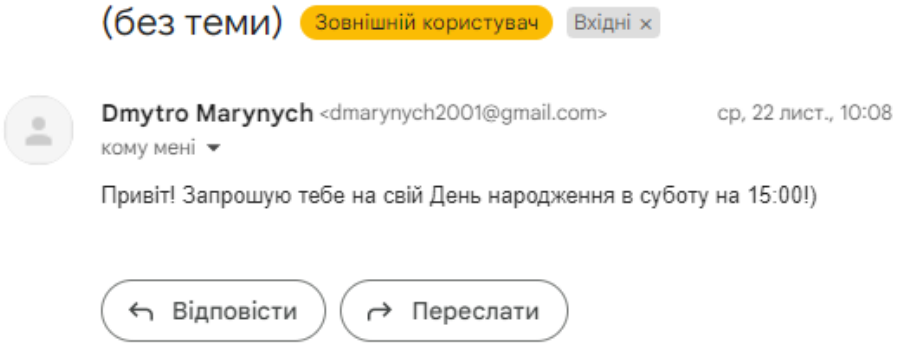

Рис. 4.8. Аналогічний лист в Gmail.

В даному прикладі був наведений лист, який дійсно не має ніяких прийомів соціальної інженерії. У висновку рішення було відповідний. Тепер спробуємо просканувати лист, де зловмисник приставляється старим другом та намагається витягнути праоль з користувача (рис. 4.9.). Як можемо бачити програма нам видає коректний висновок.

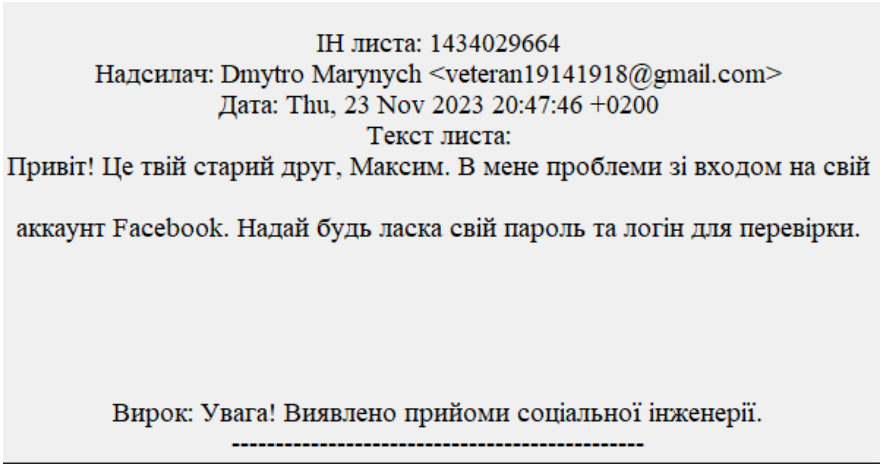

Рис. 4.9. Приклад небезпечного листа.

Тепер проведемо аналогічне тестування з небезпечним листом, але в якому є також і вкладений файл (рис. 4.10). Як можемо бачити вирок знову дається коректний. В придачу нам також відобразився список вкладених файлів. Програма буде з більшою вірогідністю давати вирок, що в листі є прийоми соціальної інженерії, коли в ньому є, наприклад, файл exe.

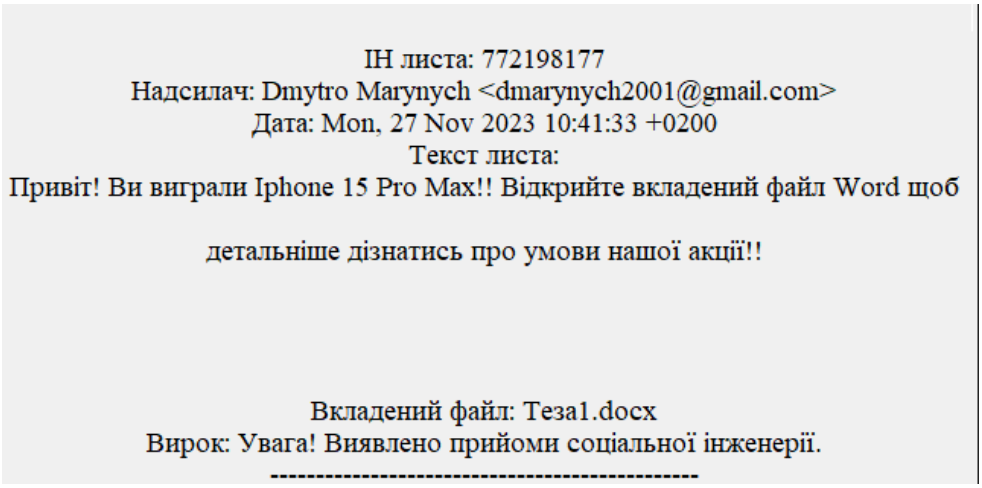

Рис. 4.10. Показ вкладеного файлу в листі.

4.3. Подальші можливості покращення ПЗ.

Розширення функціоналу сканування листів в електронній пошті може включати в себе вдосконалення методів виявлення різних видів шахрайства, спаму та соціальної інженерії. Додаткові алгоритми та функціональні можливості можуть забезпечити більш точне та повне сканування листів, що дозволить користувачам захищати свою електронну пошту від різних загроз. Розглянемо деякі з варіантів:

1. Вдосконалення аналізу заголовків листів. Аналізувати не лише вміст листів, але й їх заголовки. Заголовки містять важливу інформацію про джерело листа, його призначення та інші метадані. Деякі ознаки в заголовках можуть свідчити про те, що лист містить спам або є підозрілим.

2. Використання різних моделей машинного навчання. Додавання додаткових моделей машинного навчання для виявлення підозрілих листів. Можливо, ансамбль з різних моделей або використання більш спеціалізованих алгоритмів може підвищити точність виявлення шахрайства та спаму.

3. Оцінка надійності джерела. Аналіз джерела листа для визначення його надійності. Це може включати перевірку домену відправника, порівняння з відомими списками спам-серверів або використання методів аутентифікації, таких як SPF, DKIM та DMARC.

4. Аналіз синтаксичних особливостей листів. Виявлення підозрілих синтаксичних особливостей у листах, таких як незвичайна граматика, часте використання певних слів або фраз, які можуть вказувати на соціальну інженерію.

5. Вдосконалення обробки вкладень та посилань. Аналіз вкладень та посилань у листах для виявлення підозрілого вмісту або спамових посилань. Проведення перевірок на наявність вірусів або шкідливих файлів у вкладеннях.

6. Користувацькі налаштування правил сканування. Надання можливості користувачам створювати власні правила для виявлення підозрілих листів на основі їх власного досвіду та потреб.

Автоматизація процесу сканування та аналізу листів також може значно полегшити користування програмою та підвищити її ефективність. Розглянемо можливості автоматизації:

1. Регулярне планування сканування. Реалізація регулярного планувальника, який автоматично проводитиме сканування пошти згідно з заданим розкладом. Наприклад, сканування може відбуватися щоденно, щотижнево або за іншим заданим інтервалом.

2. Пристрої для виявлення підозрілих листів. Розробка спеціалізованих алгоритмів, які виявлятимуть підозрілі листи без необхідності втручання користувача. Це може включати автоматичне переміщення підозрілих листів у відповідний розділ чи навіть автоматичне видалення.

3. Система автоматичної реакції на підозрілі листи. Розробка системи, яка автоматично реагує на виявлення підозрілих листів. Це може бути повідомлення адміністратору системи, блокування доступу до підозрілих листів або виконання інших заданих дій.

4. Автоматична генерація звітів із результатами аналізу. Реалізація системи, яка автоматично генерує звіти з результатами аналізу листів. Ці звіти можуть бути відправлені адміністраторам або користувачам по заданому розкладу чи за запитом.

5. Опції конфігурації автоматичних завдань. Надання користувачам можливості налаштовувати автоматичні завдання. Наприклад, вказання конкретного часу або дня для проведення сканування, налаштування правил для автоматичної реакції на певні типи листів тощо.

6. Інтеграція з іншими системами. Можливість інтеграції програми з іншими системами для автоматичного обміну даними та виконання спеціалізованих дій відповідно до отриманих результатів.

4.4. Висновок до четвертого розділу.

Наш процес розробки цієї програми перейшов крізь кілька ключових етапів, включаючи концептуалізацію, реалізацію та тестування. На початковому етапі ми розглядали вимоги до програми, визначали її цільову аудиторію та функціонал. Ми вирішили зосередитися на скануванні електронних листів для виявлення можливих загроз та прийомів соціальної інженерії. Під час проектування ми розробили основну архітектуру програми, визначили необхідні бібліотеки та інструменти.

Детально розглянули кожен блок коду, наглядно описали що і за що відповідає. Наша програма складається з графічного інтерфейсу, функцій

аналізу тексту, використання методів машинного навчання для класифікації листів та інтеграції з Gmail API для отримання доступу до пошти.

Після реалізації програми ми провели тестування для перевірки її функціональності та надійності. Проведення тестів допомогло виявити деякі помилки та недоліки, які були виправлені для покращення роботи програми. Ми також розширили функціонал програми, включивши автоматичне сканування та виявлення підозрілих листів.

Також в кінці був наведений певний перелік можливих покращень, які в майбутньому зможуть зробити наше ПЗ ще більш ефективним у боротьбі з соціальною інженерією.

РОЗДІЛ 5. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1. Електромагнітні забруднення довкілля.

Електромагнітне забруднення є результатом поширення технологій та використання різноманітних пристроїв, які генерують електромагнітне випромінювання. Це випромінювання створює електричні та магнітні поля, які можуть впливати на навколишнє середовище та живі організми.

Серед основних джерел електромагнітного випромінювання можна виділити[13]:

1. Електронні прилади та побутова техніка. Мобільні телефони, мікрохвильові печі, бездротові мережі, телевізори, комп'ютери та інші електронні пристрої є основними джерелами електромагнітного випромінювання в побуті. Їх широке поширення призводить до збільшення рівня електромагнітного поля в нашому оточенні;

2. Електромережі. Електричні лінії, трансформатори та інші складові електромереж є значним джерелом магнітного поля. Високовольтні лінії передачі електроенергії також сприяють електромагнітному забрудненню;

3. Технології зв'язку. Бездротові мережі, зокрема мережі мобільного зв'язку (3G, 4G, 5G), супутниковий зв'язок та інші технології, пов'язані з передачею даних, генерують електромагнітне випромінювання;

4. Медичні прилади: Медичне обладнання, таке як рентгенівські апарати, магнітно-резонансні томографи, також може бути джерелом інтенсивного електромагнітного випромінювання.

Від розширеного використання та постійного розвитку цих джерел електромагнітного випромінювання виникає необхідність ретельного вивчення їх впливу на навколишнє середовище та здоров'я людини.

Електромагнітне випромінювання може мати вплив на природне середовище. Часто це випромінювання взаємодіє з екосистемами, впливаючи на різноманітні види, зокрема на міграцію птахів, поведінку комах, рост рослин та інші процеси. На сьогоднішній день багато досліджень зосереджені на вивченні можливих впливів електромагнітного випромінювання на здоров'я людини. Існують обговорення про можливість зв'язку між довготривалим впливом електромагнітного поля та ризиком виникнення певних захворювань, таких як рак, порушення сну, стрес, депресія та інші проблеми зі здоров'ям.

Поняття "електросмог" використовується для опису впливу електромагнітного поля на середовище та здоров'я. Це поняття походить від слова "смог", що використовується для опису забруднення повітря в містах, і вказує на те, що електромагнітне забруднення також може мати важливий вплив на здоров'я та середовище[13]. Це може включати електромагнітне випромінювання від електромереж, побутової техніки та інших джерел, які можуть спричиняти зміни в організмі та навколишньому середовищі.

Сучасна наука та технології створюють нові можливості для зменшення електромагнітного забруднення та його впливу на навколишнє середовище та здоров'я людей. Виникнення та дослідження нових матеріалів з можливістю поглинання або блокування електромагнітного випромінювання. Це можуть бути спеціальні полімерні композити, металеві або карбонові матеріали зі спеціальними властивостями.

Під розробкою екологічно чистих технологій розуміють створення нових технологій, які дозволять використовувати енергію без зайвого електромагнітного випромінювання. Наприклад, розробка електронних пристроїв, які працюють на меншій потужності, або впровадження альтернативних джерел енергії. Посилення регуляцій та встановлення стандартів щодо максимально допустимого рівня електромагнітного випромінювання для різних типів пристроїв та технологій. Це сприятиме контролю та зменшенню потенційного впливу на здоров'я.

Також продовжуються наукові дослідження щодо впливу електромагнітного випромінювання на організми та розробка заходів захисту, таких як спеціальні екрануючі матеріали, або рекомендації щодо використання техніки.

Що стосується заходів захисту від електромагнітного забруднення, то існують деякі рекомендації, спрямовані на мінімізацію впливу радіації та зменшення можливих шкідливих наслідків:

1. Під час використання мобільного телефону найбезпечнішим способом обмежити електромагнітне забруднення є використання режиму гучного зв'язку або Bluetooth. Це означає випромінювання нижчого рівня радіації;

2. Також не рекомендується користуватися мобільним телефоном у ліфтах, автомобілях, потягах чи літаках, оскільки в закритих металевих приміщеннях він споживає більше енергії та випромінює більше випромінювання;

3. Замість КЛЛ безпечніше використовувати світлодіодні лампи або лампи розжарювання;

4. РК-телевізори рекомендуються замість плазмових, оскільки вони випромінюють набагато менше випромінювання;

5. Рекомендується вимикати електроприлади в будинку під час їх використання. Будь-що, підключене до розетки, створює сильне електромагнітне поле, і коли його від'єднати, воно вже не має такого ефекту. Прикладом цього є ноутбуки, тому їх добре використовувати в режимі батареї, а потім прибрати, поки вони заряджаються;

6. Відключення маршрутизатора Wi-Fi, коли він не використовується, є ефективним заходом захисту від електромагнітного забруднення;

7. Рекомендується уникати розміщення мобільних телефонів поблизу, особливо під час сну.

Електромагнітне випромінювання може створювати потенційні ризики для населення, якщо перевищено стандартні рівні. Характеристики електромагнітного поля необхідно періодично оцінювати, щоб зрозуміти

тенденції електромагнітного випромінювання та запровадити спеціальні захисні заходи[30].

Законодавство про навколишнє середовище існує для обмеження будь-якого забруднення, і оцінка впливу на навколишнє середовище є процесом, який здійснюється відповідно до цього. Тому діяльність з можливим шкідливим впливом підлягає процесу оцінки, який називається екологічним аудитом. Крім того, компанії, які мають дозвіл на охорону навколишнього середовища, повинні співпрацювати з консультантом, який збирає найважливішу інформацію щодо діяльності, яка потенційно може спричинити різні екологічні проблеми. Таким чином, можна запропонувати ефективні рішення щодо дотримання зобов’язань, покладених законом.

Незважаючи на те, що електромагнітне випромінювання є невидимим ворогом, воно є загальною екологічною проблемою, яка швидко зростає. Тому для боротьби з цим явищем важливо якомога краще розуміти концепцію електромагнітного забруднення та вживати необхідних заходів. Хоча шкідливі наслідки до кінця не вивчені, дослідження показали, що вплив високих рівнів електромагнітних полів може бути шкідливим для здоров’я. В даний час все більше уваги приділяється можливим довгостроковим наслідкам впливу електромагнітних полів на здоров'я[30].

Трудова діяльність із залученням машин або електричних систем, що випромінюють електромагнітні поля, повинна передбачати комплекс організаційних заходів захисту для запобігання впливу на людей електромагнітних полів, що випромінюються таким обладнанням. Крім того, обладнання, яке випромінює електромагнітні поля, має бути розміщене в спеціальних робочих зонах, спеціально створених для використання цього обладнання, і на відповідній відстані від інших робочих зон.

5.2. Висновки до п’ятого розділу.

Електромагнітне забруднення – це явище, яке існує і може стати ще агресивнішим у міру розвитку технологій. Цей тип забруднення зовсім не є аспектом, яким слід нехтувати, беручи до уваги той факт, що існують прості та легкі рішення, які може застосувати кожен. Використання дротових мереж Інтернету в офісних приміщеннях замість бездротових є лише прикладом захисних заходів, які можна вжити в цьому відношенні.

Розглянуті джерела електромагнітного випромінювання, включаючи побутову техніку, електромережі та бездротові технології, виробляють електромагнітні поля, які оточують нас у повсякденному житті. На сьогоднішній день величезна кількість досліджень присвячена вивченню впливу цього поля на середовище та здоров'я людини.

Багато з цих досліджень показують потенційний вплив електромагнітного забруднення на здоров'я, такий як можливий ризик виникнення певних захворювань та вплив на екосистеми. Однак, наразі існує потреба в подальших дослідженнях та уточненні висновків, щоб з'ясувати повний обсяг цього впливу та його механізми.

Інноваційні підходи до управління електромагнітним забрудненням, такі як розробка нових матеріалів, створення екологічно чистих технологій, регулювання та встановлення стандартів, а також освітня робота, є ключовими напрямками у боротьбі з цією проблемою.

Здійснення всебічних досліджень та прийняття ефективних заходів для зменшення експозиції до електромагнітного випромінювання можуть сприяти створенню більш безпечного й екологічно чистого середовища для майбутніх поколінь.

ВИСНОВОК

В ході виконання дипломної роботи ми спрямували увагу на вплив соціальної інженерії та її наслідки в кіберпросторі. Почали ми з вивчення та розгляду психологічних аспектів соціальної інженерії, освітлюючи основні мотиви, які стимулюють зловмисників використовувати ці техніки. Розкривши різноманітні стратегії маніпуляції та психологічні механізми, ми зрозуміли, як вони можуть бути використані для отримання незаконного доступу до інформації або здійснення шкідливих дій в кіберпросторі.

Досліджуючи найвідоміші інциденти, пов'язані з соціальною інженерією, ми відзначили важливість розвитку заходів захисту в цій сфері. Аналізуючи такі інциденти, ми виявили ключові сценарії та типові прийоми, які використовуються для здійснення атак. Це розуміння стало фундаментом для розробки превентивних стратегій та заходів захисту від таких загроз.

У нашій роботі ми активно використовували різноманітні інструменти та технології для створення програмного забезпечення, спрямованого на виявлення фішингових листів. Наш підхід базувався на використанні ряду потужних бібліотек та інструментів, таких як scikit-learn для реалізації алгоритмів машинного навчання, google api client і google oauth 2.0 для взаємодії з послугами Google та tkinter для створення інтуїтивно зрозумілого графічного інтерфейсу користувача.

Бібліотека scikit-learn, яка є однією з ключових у нашій роботі, надала нам можливість використовувати різноманітні алгоритми машинного навчання для аналізу тексту електронних листів. Ми застосовували класифікаційні та кластеризаційні моделі для ідентифікації підозрілих листів на основі їхнього змісту та структури. Це дозволило нам реалізувати в додатку функціональність автоматичного аналізу тексту та виявлення підозрілих аспектів, що можуть вказувати на фішингову атаку.

Google API Client та Google OAuth 2.0 були використані для взаємодії з сервісами Google, зокрема, з доступом до листів Gmail. Ці інструменти дозволили нам отримувати доступ до електронних листів користувачів, що було необхідною передумовою для їхнього подальшого аналізу на предмет фішингових загроз. Використання API дозволило нам отримувати необхідну інформацію про листи та їх зміст для подальшого аналізу та класифікації.

Tkinter, бібліотека для створення графічного інтерфейсу користувача в мові програмування Python, була використана для створення зручного та інтуїтивно зрозумілого інтерфейсу нашого програмного забезпечення. Ми створили GUI, яке дозволяє користувачам легко взаємодіяти з додатком, завантажувати та аналізувати свої електронні листи з мінімальними зусиллями.

Процес розробки додатку включав створення навчального датасету у форматі txt, що містив найпоширеніші прийоми соціальної інженерії. Наша нейронна мережа навчалась на цих даних, аналізуючи текстову інформацію та виявляючи патерни та закономірності, що дозволило їй ефективно розпізнавати підозрілі вирази та прийоми, характерні для соціально-інженерних атак.
Проведені успішні тестування підтвердили ефективність нашого програмного забезпечення, здатного надавати коректні висновки щодо характеру листів електронної пошти. Додаток продемонстрував свою здатність до виявлення фішингових листів, що має значний вплив на підвищення рівня кібербезпеки та забезпечення захисту для користувачів.

Дана дипломна робота представляє собою не лише результат вивчення соціальної інженерії, але й практичний внесок у сферу кібербезпеки. Вона охоплює широкий спектр технічних, психологічних та технологічних аспектів, що робить її вагомим внеском у вирішенні проблеми кібербезпеки у сучасному цифровому світі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Базовые алгоритмы машинного обучения на языке Python : учебно-методическое пособие / А. Ю. Долганов, М. В. Ронкин, А. В. Созыкин ; М-во науки и высшего образования РФ. — Екатеринбург : Изд-во Урал. ун-та, 2023. — 124 с.
2. Основы искусственного интеллекта в примерах на Python. Самоучитель. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2023. — 448 с
3. Прикладное машинное обучение без учителя с использованием Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2020. - 432 с.
4. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО "Альфа-книга": 2018. - 688 с.
5. Социальная инженерия и социальные хакеры / М. В. Кузнецов, И. В. Симдянов. — СПб.: БХВ-Петербург, 2007. — 368 с.
6. Acquisti, A., Brandimarte, L., & Loewenstein, G. (2015). Privacy and human behavior in the age of information. Science, 347(6221), 509–514.
7. Advanced social engineering attacks / Katharina Krombholz // Journal of Information Security and Applications, 2015. – 120 p.
8. Benenson, Z., Gassmann, F., Landwirth, R. (2017). Unpacking spear phishing susceptibility. In Lecture Notes in Computer Science (including subseries Lecture Notes in Arti!cial Intelligence and Lecture Notes in Bioinformatics), 10323 LNCS Springer, Cham (pp. 610–627).
9. Burgoon, J. K., & Levine, T. R. (2010). Advances in deception detection. In New directions in interpersonal communication research (pp. 201–220). SAGE Publications.
10. Cain, A., Edwards, M., and Still, J. 2018. "An exploratory study of cyber hygiene behaviors and knowledge", Journal of Information Security and Applications (42:3), pp. 36-45.

11. Dmytro Marynych, Andrii Petrenko. Anti-social engineering software // Information Technology & Implementation : матеріали конференції, 20-21 листопада 2023 р. – Київ : 2023. – С. 102-103.
12. Educating and Raising Awareness on Cyber Security Social Engineering / Hussain Aldawood, Geoffrey Skinner // International Conference on Teaching, Assessment, and Learning for Engineering (TALE). – 2018.
13. Electromagnetic Pollution of the Environment and its Effects on the Materials from the Built up Media / Iosif Lingvay // Technical University of Cluj, Faculty of Electronics, Telecommunications and Information Technology. – 2018.
14. Evans, J. R., Michael, S.W., Meissner, C. A., & Brandon, S. E. (2013). Validating a new assessment method for deception detection: Introducing a psychologically based credibility assessment tool. Journal of Applied Research in Memory and Cognition, 2(1), 33–41.
15. Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., et al. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Front. Neuroinform. 5:13.
16. Holm, H., Flores, W., Ericsson, G. (2013). Cyber security for a smart grid – what about phishing? Internet Complaint Center (IC3). (2018a). 2017 internet crime report. Retrieved 23-sep-2018
17. Influence: The Psychology of Persuasion / Cialdini R. B. // HarperBusiness. – 2006.
18. Jan-Willem Bullée and Marianne Junger / Social Engineering // The Palgrave Handbook of International Cybercrime and Cyberdeviance. 2021
19. Sarah Alowais / Cyber Hygiene Practices Across Cultures: A Cross Cultural Study of the US and Saudi Arabia based Information Systems Users // Procedia Computer Science, 2023. - pages 744-750
20. Social engineering attack examples, templates and scenarios / Francois Mouton, Louise Leenen, H.S. Venter // Computers & Security, 2016. – P 189-202.
21. Social Engineering Attacks: A Survey / Fatima Salahdine, Naima Kaabouch // School of Electrical Engineering and Computer Science. – 2019.

22. Social Engineering in IT Security: Tools, Tactics, and Techniques. / Ghosh A. K., Doss R. J. // Artech House. – 2015
23. Social Engineering: Th e Art of Human Hacking / Christopher Hadnagy // Wiley Publishing. 2011
24. The Art of Deception: Controlling the Human Element of Security / Mitnick K. D., Simon W. L. // Wiley. – 2002.
25. 2014 JPMorgan Chase data breach. Wikipedia [Електронний ресурс]. – URL: https://en.wikipedia.org/wiki/2014_JPMorgan_Chase_data_breach.
26. 2017 Equifax data breach. Wikipedia [Електронний ресурс]. – URL: https://en.wikipedia.org/wiki/2017_Equifax_data_breach.
27. 2020 Twitter account hijacking. Wikipedia [Електронний ресурс]. – URL: https://en.wikipedia.org/wiki/2020_Twitter_account_hijacking.
28. Artificial Inteligence (2018-2023). Google Trends [Електронний ресурс] - URL: https://trends.google.com/trends/explore?date=today%205-y&geo=US&q=%2Fm%2F0mkz&hl=ru
29. Base16, Base32, Base64, Base85 Data Encodings. Python Docs [Електронний ресурс] - URL: https://docs.python.org/3/library/base64.html
30. Electromagnetic pollution – sources, effects and protective measures that can reduce its potential negative consequences. Stratos [Електронний ресурс] - URL: https://stratos.ro/en/poluarea-electromagnetica-surse-efecte-si-masuri-de-protectie-care-pot-diminua-potentialele-consecinte-negative-ale-acesteia/#:~:text=Some%20sources%20of%20electromagnetic%20pollution,strong%20sources%20of%20electromagnetic%20fields.
31. Google APIs / google-api-python-client. Github [Електронний ресурс] - URL: https://github.com/googleapis/google-api-python-client
32. Graphical User Interfaces with Tk. Python Docs [Електронний ресурс] - URL: https://docs.python.org/3/library/tk.html
33. OAuth 2.0 - googleapis/google-api-python-client. Github [Електронний ресурс] - URL: https://github.com/googleapis/google-api-python-client/blob/main/docs/oauth.md

34. Python. Wikipedia [Електронний ресурс] - URL: https://ru.wikipedia.org/wiki/Python
35. Scikit-learn. Wikipedia [Електронний ресурс] - URL: https://ru.wikipedia.org/wiki/Scikit-learn
36. scikit-learn: machine learning in Python [Електронний ресурс] - URL: https://scikit-learn.org/stable/
37. Авторизация OAuth 2.0 от Google. Habr.com [Електронний ресурс] - URL: https://habr.com/ru/articles/713442/
38. Регулярные выражения в Python: теория и практика. Tproger [Електронний ресурс] - URL: https://tproger.ru/translations/regular-expression-python
39. Руководство по программированию на Tkinter и Python. Metanit [Електронний ресурс] - URL: https://metanit.com/python/tkinter/

Додаток А

Загальний код на мові Python

```
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
import tkinter as tk
import base64
import re
import hashlib
from tkinter import filedialog

# Функція для аналізу тексту
def analyze_email(email_text):
    # Векторизація тексту
    text_vectorized = vectorizer.transform([email_text])

    # Використання навченого класифікатора для передбачення
    prediction = classifier.predict(text_vectorized)

    return prediction[0]

def shorten_id(id):
    hashed = hashlib.sha256(str(id).encode()).digest()
    return int.from_bytes(hashed[:4], byteorder='big')  # Конвертація хешу у ціле число

def browse_file():
    filepath = filedialog.askopenfilename()  # Відкриття вікна вибору файлу
```

```
    credentials_entry.delete(0, tk.END)  # Очищення поля введення
    credentials_entry.insert(0, filepath)  # Вставлення шляху до файлу в поле введення

def show_email_info(msg_id):
    email = email_entry.get()
    credentials_path = credentials_entry.get()
    creds = service_account.Credentials.from_service_account_file(credentials_path,
                                                  scopes=['https://mail.google.com/'],
                                                  subject=email)

    service = build('gmail', 'v1', credentials=creds)
    all_info = ""

    try:
        msg = service.users().messages().get(userId=email, id=msg_id, format='full').execute()
        payload = msg['payload']

        headers = payload['headers']
        sender_name = [header['value'] for header in headers if header['name'] == 'From']
        sender_name = sender_name[0] if sender_name else 'Unknown'
        date = [header['value'] for header in headers if header['name'] == 'Date']
        date = date[0] if date else 'Unknown'

        email_text = "No text part found in this email."

        def decode_text(text):
            return base64.urlsafe_b64decode(text.encode('ASCII')).decode("utf-8")
```

```
        def check_parts(parts):
            nonlocal email_text
            for part in parts:
                if 'parts' in part:
                    check_parts(part['parts'])
                elif 'mimeType' in part and part['mimeType'] == 'text/plain':
                    email_text = decode_text(part['body']['data'])
                    break

        check_parts(payload.get('parts', []))

        all_info = f"ІН листа: {shorten_id(msg_id)}\nНадсилач: {sender_name}\nДата: {date}\nТекст листа:\n{email_text}\n"

        # Перевірка наявності вкладення
        if 'parts' in payload:
            attachment_info = ""
            file_attached = False
            for part in payload['parts']:
                if 'filename' in part:
                    file_attached = True
                    attachment_info += f"Вкладений файл: {part['filename']}\n"
            if file_attached:
                all_info += "Attachments:\n" + attachment_info

        prediction = analyze_email(email_text)

        if prediction == 1:
            all_info += "Вирок: Увага! Виявлено прийоми соціальної інженерії.\n----------------------------------------\n"
```

```
        else:
            all_info += "Вирок: Даний лист виглядає безпечно.\n----------------------------------------\n"

    except HttpError as error:
        all_info = f"An error occurred: {error}"

    result_label.config(text=all_info)

def clear_window():
    def clear_window():
        for widget in root.winfo_children():
            if widget != result_label:
                widget.grid_forget()

# Функція для сканування поштових листів
def scan_emails():
    email_label.grid_forget()
    email_entry.grid_forget()
    credentials_label.grid_forget()
    credentials_entry.grid_forget()
    browse_button.grid_forget()
    scan_button.grid_forget()
    clear_window()

    email = email_entry.get()
    credentials_path = credentials_entry.get()
    creds = service_account.Credentials.from_service_account_file(credentials_path,
                                                    scopes=['https://mail.google.com/'],
                                                    subject= email)
```

```
    service = build('gmail', 'v1', credentials=creds)
    all_info = ""

    try:
        results = service.users().messages().list(userId=email).execute()
        messages = results.get('messages', [])

        if not messages:
            all_info = "No messages found."
        else:
            canvas = tk.Canvas(root)
            canvas.grid(row=1, column=0, columnspan=3, padx=5, pady=5, sticky='news')

            scrollbar = tk.Scrollbar(root, orient="vertical", command=canvas.yview)
            scrollbar.grid(row=1, column=3, sticky='ns')

            frame = tk.Frame(canvas)
            canvas.create_window((0, 0), window=frame, anchor='nw')

            canvas.configure(yscrollcommand=scrollbar.set)
            canvas.bind('<Configure>', lambda e:
canvas.configure(scrollregion=canvas.bbox("all")))

            for message in messages:
                msg_id = message['id']
                msg = service.users().messages().get(userId=email, id=msg_id,
format='full').execute()
                payload = msg['payload']
                headers = payload['headers']
```

```
            sender_name = [header['value'] for header in headers if header['name'] ==
'From']
            sender_name = sender_name[0] if sender_name else 'Unknown'
            date = [header['value'] for header in headers if header['name'] == 'Date']
            date = date[0] if date else 'Unknown'
            short_id = shorten_id(msg_id)
            btn_text = f"ID: {short_id}, From: {sender_name}, Date: {date}"
            btn = tk.Button(frame, text=btn_text, command=lambda id=msg_id:
show_email_info(id))
            btn.pack(fill='x', padx=5, pady=2)

    except HttpError as error:
        all_info = f"An error occurred: {error}"

    result_label.config(text=all_info)

# Завантаження навчального датасету
with open('studylist.txt', 'r', encoding='utf-8') as file:
    lines = file.readlines()

# Розділення мітки та тексту повідомлення
labels = []
texts = []
for line in lines:
    parts = re.split(r',\s*', line.strip())
    labels.append(int(parts[0]))
    texts.append(parts[1])

# Векторизація тексту
vectorizer = TfidfVectorizer()
```

```
X_vectorized = vectorizer.fit_transform(texts)

# Навчання класифікатора
classifier = MultinomialNB()
classifier.fit(X_vectorized, labels)

# Графічний інтерфейс
root = tk.Tk()
root.title("Gmail Scanner")

email_label = tk.Label(root, text="Адреса пошти:")
email_label.grid(row=0, column=0, sticky='w', padx=5, pady=5)

email_entry = tk.Entry(root)
email_entry.grid(row=0, column=1, padx=5, pady=5)

credentials_label = tk.Label(root, text="Файл з обліковим записом:")
credentials_label.grid(row=1, column=0, sticky='w', padx=5, pady=5)

credentials_entry = tk.Entry(root)
credentials_entry.grid(row=1, column=1, padx=5, pady=5)

browse_button = tk.Button(root, text="Шукати", command=browse_file)
browse_button.grid(row=1, column=2, padx=5, pady=5)

scan_button = tk.Button(root, text="Сканувати", command=scan_emails)
scan_button.grid(row=2, column=1, padx=5, pady=5)

result_label = tk.Label(root, text="", font=("Times New Roman", 14))
result_label.grid(row=3, column=0, columnspan=3, padx=5, pady=5)
```

```
root.mainloop()
```

Додаток Б

Блок-схема програм