

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень магістр
освітньо-наукова програма Кібербезпека
(назва освітньої програми)

на тему: «Методи виявлення та протидії соціальній інженерії в соціальних мережах»

Виконавець: студент II курсу, групи КБм-21

Олександр ПАСЬКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	Підпис
Науковий керівник	Володимир НАКОНЕЧНИЙ	
Нормоконтроль	Сергій ДАКОВ	

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:

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

ЗАВДАННЯ

на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітній ступень магістр

Здобувача(ки) КБм-21 Паська Олександра Ігоровича
(група) (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Методи виявлення та протидії соціальній інженерії в соціальних мережах

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

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

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень Процес виявлення повідомлень з соціальною інженерією

Предмет досліджень Методи виявлення повідомлень з соціальною інженерією

Мета удосконалення методу виявлення повідомлень з соціальною інженерією та розробка програмного застосунку, який допоможе виявити соціальну інженерію в повідомленнях в месенджері «Telegram»

Вихідні дані для проведення роботи	Теоретичні й експериментальні дослідження виявлення соціальної інженерії в повідомленнях в месенджері «Telegram»
---	--

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	удосконалення методу виявлення соціальної інженерії в повідомленнях в месенджері «Telegram».
Практична цінність	Розроблена програмна реалізація, яка підвищує ефективність виявлення соціальної інженерії в повідомленнях в месенджері «Telegram»

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	17.11.2023 – 21.01.2024
Аналіз літературних джерел	22.01.2024 – 28.01.2024
Ознайомлення з сучасною проблемою атак соціальної інженерії	29.01.2024 – 11.02.2024
Розгляд методів проведення атак соціальної інженерії	12.02.2024 – 25.02.2024
Аналіз рекомендацій стосовно виявлення та протидії соціальній інженерії	26.02.2024 – 10.03.2024
Дослідження існуючих програмних методів виявлення соціальної інженерії	11.03.2024 – 24.03.2024
Розробка програмного застосунку для виявлення соціальної інженерії в повідомленнях в месенджері «Telegram»	25.03.2024 – 23.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	24.04.2024 – 12.05.2024
Подача пакету документів на розгляд ЕК	13.05.2024 – 18.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження ризиків інформаційної безпеки, пов'язаних з соціальною інженерією.

Соціальний ефект Покращення технологій виявлення соціальної інженерії.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав

(підпис)

Володимир НАКОНЕЧНИЙ

(Ім'я, ПРІЗВИЩЕ)

Завдання прийняв
до виконання

(підпис)

Олександр ПАСЬКО

(Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.

Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Методи виявлення та протидії соціальній інженерії в соціальних мережах»: 81 сторінка, 54 рисунки та 54 літературних джерела.

Актуальність роботи – з кожним роком соціальні інженери створюють нові методи, які дозволяють обходити існуючі механізми захисту. Якщо раніше зловмисники надсилали фішингові листи з загальним змістом (без конкретики) та з великою кількістю помилок, то зараз, використовуючи інструменти OSINT, автоматизації атак та штучний інтелект, роблять персоналізовані атаки, підлаштовуючись під кожного користувача. Але й захист від таких атак не стоїть на місці, якщо раніше сканери намагались виявляти соціальну інженерію по ключовим словам, або інформації про відправника, то зараз використовуючи машинне навчання, LLM та NLP, створюються моделі, які можуть досить точно виявити соціальну інженерію.

Також змінюється платформа розповсюдження атак соціальної інженерії. Довгий час домінувала лише електронна пошта, але зараз все частіше зустрічаються кібератаки в месенджерах, так як більшість людей використовують їх для комунікації.

Тому захист від соціальної інженерії в месенджерах та соціальних мережах стає все більш актуальним.

Об'єкт дослідження – процес виявлення повідомлень з соціальною інженерією.

Мета роботи – удосконалення методу виявлення повідомлень з соціальною інженерією та розробка програмного застосунку, який допоможе виявити соціальну інженерію в повідомленнях в месенджері «Telegram».

Методи дослідження – аналіз зібраних даних, їх формалізація та структуризація, дослідження наявних технологій.

Наукова новизна: удосконалення методу виявлення соціальної інженерії в повідомленнях в месенджері «Telegram».

Сформовані в результаті, теоретичні, практичні рекомендації та розроблений на їх основі програмний застосунок можуть бути використані, як додатковий інструмент захисту в месенджерах та соціальних мережах.

При подальшому дослідженні метод може стати ще ефективнішим за рахунок впровадження нових мовних моделей спеціально навчених на соціальній інженерії.

Ключові слова: штучний інтелект, соціальна інженерія, GPT, LLM, NLP, месенджер.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AI	–	Artificial intelligence
API	–	Application Programming Interface
CE	–	community edition
(D)DoS	–	(Distributed) Denial-of-Service
GPT	–	Generative Pre-trained Transformer
LLM	–	Large language model
NLP	–	Natural Language Processing
OSINT	–	Open source intelligence
OC	–	Операційна система
ПЗ	–	Програмне забезпечення
CI	–	Соціальна інженерія
ШІ	–	Штучний інтелект

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	7
ВСТУП.....	10
РОЗДІЛ 1 СОЦІАЛЬНА ІНЖЕНЕРІЯ ТА МЕТОДИ ЇЇ ЗАСТОСУВАННЯ	12
1.1 Статистика атак соціальної інженерії	12
1.2 Техніки соціальної інженерії	14
1.3 Канали атак соціальної інженерії	16
1.3.1 Загальна статистика	16
1.3.2 Статистика для України	17
1.4 Етапи проведення атаки соціальної інженерії	19
1.5 Інструменти для проведення атак соціальної інженерії.....	21
1.5.1. Maltego.....	21
1.5.2 SpiderFoot	24
1.5.3 OSINT Framework.....	26
1.5.4 Gophish	28
1.5.5 The Social-Engineer Toolkit (SET)	30
1.6 Використання штучного інтелекту для соціальної інженерії.....	33
1.6.1 Deepfake.....	34
1.6.2 NLP	35
Висновки до розділу 1	39
РОЗДІЛ 2 МЕТОДИ ВИЯВЛЕННЯ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ	40
2.1 Особливості атаки соціальної інженерії	40
2.2 Як захиститись від соціальної інженерії.....	42
2.3 Програмні методи виявлення соціальної інженерії.....	44
2.3.1 VirusTotal.....	44
2.3.2 AI Detector	45
2.4 Використання штучного інтелекту.....	47

2.5 Аналіз тональність тексту	49
2.6 Пошук слів тригерів	51
Висновки до розділу 2	52
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ВИЯВЛЕННЯ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ В ПОВІДОМЛЕННЯХ В МЕСЕНДЖЕРІ «TELEGRAM»	53
3.1 Середовище розробки	53
3.2 Функції та скрипти програмного за стосунку	54
3.2.1 Скриптовані файли	54
3.2.2 Функції та класи	54
3.3 Допоміжні модулі	64
3.3.1 Telethon	64
3.3.2 Openai	64
3.3.3 Selenium	65
3.3.4 Vertexai	65
3.3.5 Transformers	65
3.3.6 vt	66
3.3.7 deep_translator	67
3.3.8 Plyer	67
3.3.9 Базові модулі	67
3.4 Приклад використання застосунку	68
Висновки до розділу 3	73
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	76
ДОДАТОК А	82
ДОДАТОК Б	83
ДОДАТОК В	92
ДОДАТОК Г	94
ДОДАТОК Д	95

ВСТУП

Актуальність роботи пов'язана з тим, що з кожним роком соціальні інженери створюють нові методи, які дозволяють обходити існуючі механізми захисту. Якщо раніше зловмисники надсилали фішингові листи з загальним змістом (без конкретики) та з великою кількістю помилок, то зараз, використовуючи інструменти OSINT, автоматизації атак та штучний інтелект, роблять персоналізовані атаки, підлаштовуючись під кожного користувача. Але й захист від таких атак не стоїть на місці, якщо раніше сканери намагались виявляти соціальну інженерію по ключовим словам, або інформації про відправника, то зараз використовуючи машинне навчання, LLM та NLP, створюються моделі, які можуть досить точно виявити соціальну інженерію.

Також змінюється платформа розповсюдження атак соціальної інженерії. Довгий час домінувала лише електронна пошта, але зараз все частіше зустрічаються кібератаки в месенджерах, так як більшість людей використовують їх для комунікації.

Тому захист від соціальної інженерії в месенджерах та соціальних мережах стає все більш актуальним.

Основна *мета дослідження* – це удосконалення методу виявлення повідомлень з соціальною інженерією та розробка програмного застосунку, який допоможе виявити соціальну інженерію в повідомленнях в месенджері «Telegram».

Для досягнення поставленої мети необхідно виконати наступні задачі:

- проаналізувати матеріали по темі соціальна інженерія;
- виконати аналіз методів та інструментів соціальної інженерії;
- провести аналіз існуючих методів виявлення соціальної інженерії в повідомленнях;
- розробити алгоритм роботи програмного застосунку для виявлення соціальної інженерії в повідомленнях в месенджері «Telegram»;

- виконати програмну реалізацію та провести тестування на різних повідомленнях.

Об'єкт дослідження: процес виявлення повідомлень з соціальною інженерією, адже в наш час кількість кібератак тільки збільшується і більшість з них починаються саме з соціальної інженерії. Тому виявлення соціальної інженерії допоможе зупинити атаку ще на початкових етапах.

Предметом дослідження є методи виявлення повідомлень з соціальною інженерією, які допоможуть

Методи дослідження – це аналіз зібраних даних, їх формалізація та структуризація, дослідження наявних технологій.

Наукова новизна – удосконалення методу виявлення соціальної інженерії в повідомленнях в месенджері «Telegram».

Практична цінність полягає в тому, що була розроблена програмна реалізація, яка підвищує ефективність виявлення соціальної інженерії в повідомленнях в месенджері «Telegram». Сповістить користувача, якщо виявить певні аномалії в повідомленні в месенджері «Telegram». Це дозволить привернути увагу користувача і підкаже йому, що треба ретельніше перевірити повідомлення.

РОЗДІЛ 1

СОЦІАЛЬНА ІНЖЕНЕРІЯ ТА МЕТОДИ ЇЇ ЗАСТОСУВАННЯ

Фахівці постійно розробляють кращі технології безпеки, які все більше ускладнюють використання технічних вразливостей, зловмисники все частіше застосовують методи соціальної інженерії. В більшості випадків простіше зламати «людський фаєрвол», це потребує мінімальних інвестицій та ризиків [1].

Атаки соціальної інженерії – тип атак, які направлені на маніпулювання людьми, для того, щоб вони виконали заборонені або небажані дії. Наприклад, поділитись конфіденційною інформацією, завантажити небажане програмне забезпечення, надіслати кошти шахраям, тощо [2].

1.1 Статистика атак соціальної інженерії

З першого погляду здається, що атаки соціальної інженерії не несуть такої загрози, як інші атаки. Проте соціальна інженерія значно дешевша в реалізації ніж, наприклад, DDoS чи Zero Day атаки. Також для атаки соціальної інженерії використовують будь-яку тему та інформацію, яка допоможе досягти мети, тому її можна модифікувати під кожен ціль (приклад див. на рисунках 1.1 та 1.2)



Рисунок 1.1 – Приклад атаки соціальної інженерії через месенджер [4]

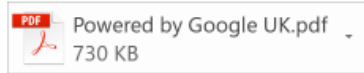


Sun 3/26/2017 10:15 PM

google.co.uk <sundarpichai.gceo@googlemail.com>

Powered by Google UK

To



Google Corporation®
 123 Buckingham Palace Road
 London SW1W 9SH
 United Kingdom

Good day Sir/Madam.

You have successfully been picked as one of our 12 Lucky Winners in this months Lottery Draw, Please see attached file for more details.

Best regards
 Sundar Pichai
 CEO Google Inc.
ceo@gpromo-team.com

Рисунок 1.2 – Приклад атаки соціальної інженерії через пошту [3]

Основна перевага атак соціальної інженерії полягає в кількості, а не складності проведення. 36% атак на організації та 92% атак на приватні особи – це соціальна інженерія [5].

Окрім цього, соціальна інженерія може використовуватись як перший етап масштабної кібератаки. Практично 98% всіх кібератак так, чи інакше включають соціальну інженерію. Наприклад, маскування під довіреного користувача, щоб заохотити співробітника перейти за зловмисним посиланням для отримання облікових даних які нададуть доступ до цільових систем [6].

Стосовно витоку даних, то соціальна інженерія найпопулярніша за всі інші методи. В багатьох випадках, легше обманом змусити співробітника надати конфіденційну інформацію, що дозволить зловмисникам отримати доступ до

системи. Як наслідок, від 70% до 90% витоків даних пов'язані із соціальною інженерією [7].

Спеціалісти підраховали, що середньостатистична організація щороку стає мішенню для 700+ атак соціальної інженерії, або 2.7 атаки на один робочий день [8]. Також слід зазначити, що в більшості випадків, зловмисники використовують соціальну інженерію для атак на малий бізнес, який атакують на 350% частіше, ніж великі підприємства [6].

Хоча атаки соціальної інженерії дешеві та прості в реалізації, наслідки від них коштують дуже дорого, навіть якщо не призвели до серйозних зломів. У середньому компанії втрачають 130 000 доларів через викрадення або знищення даних. Коли соціальна інженерія призводить до серйозного витoku даних, загальні суми можуть сягати сотень тисяч, якщо не мільйонів доларів [9].

1.2 Техніки соціальної інженерії

Кіберзлочинці використовують різноманітні техніки соціальної інженерії для кібератак (див. рисунок 1.3). Різноманітні дослідження показують, що фішинг найрозповсюдженіша техніка атаки, яку використовують зловмисники. Вона задіяна в 35% атак.

Спам – друга за популярністю техніка, яка використовується в 18% атак.

Третя техніка, яку часто використовують – це шахрайство, вона становить 14%. В цій техніці зазвичай використовують теми, які можуть бути дуже важливими для жертви, наприклад інформація про заборгованість по кредиту. Через це користувач втрачав обачність та завантажував собі в систему шкідливе програмне забезпечення, або були викрадені його облікові дані [10].

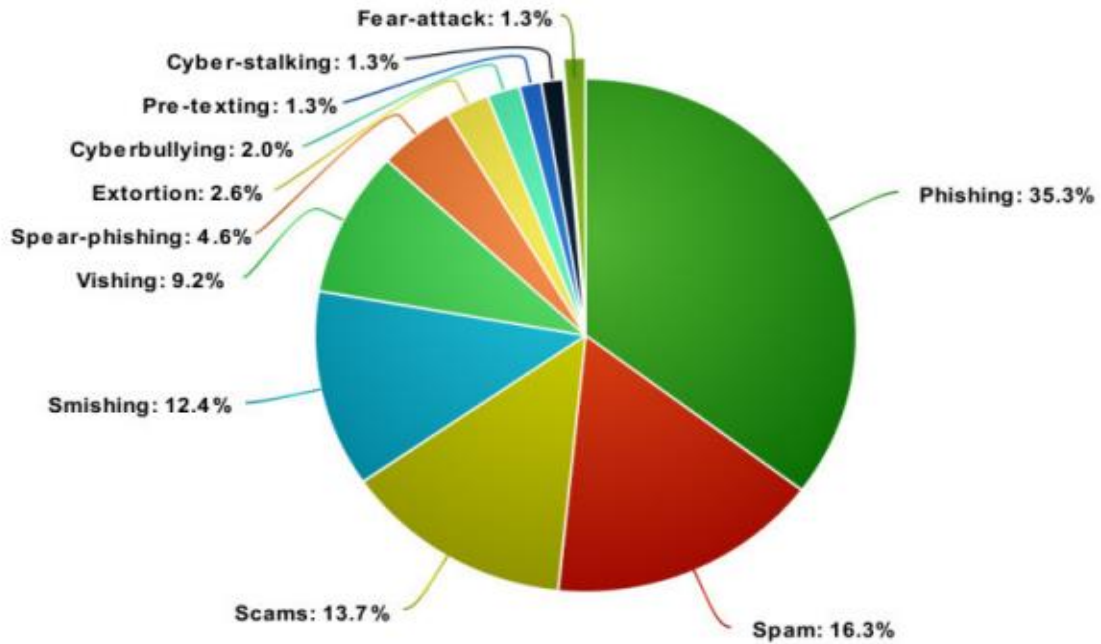


Рисунок 1.1 – Частота використання різних технік соціальної інженерії [10]

Окрім цих технік, зловмисники люблять використовувати смішинг та вішинг, вони разом становлять 22%.

Смішинг – схема обману з використанням сервісів зв'язку, яка спрямована на перехід користувачем за шкідливим посиланням, яке надійшло на SMS.

Смішинг-повідомлення може мати вигляд повідомлення від відомого банку, знайомої компанії або бути просто сповіщенням про раптовий виграш у лотерею чи у велику акцію.

Вішинг - вид шахрайства, при якому зловмисники за допомогою телефонного зв'язку змушують людину повідомити їм свої конфіденційні банківські або персональні дані або стимулюють до здійснення певних дій зі своїм банківським рахунком або банківською картою. Шахраї при цьому майстерно грають певну роль (як правило, співробітника банку, технічного фахівця, постачальника послуг, державну організацію, працівника ІТ-служби і т.д.) і використовують прийоми, методи і технології соціальної інженерії [11]. Інші

техніки атак, такі як цільовий фішинг, вимагання, кібербулінг, кіберсталкінг використовуються значно рідше.

1.3 Канали атак соціальної інженерії

Зловмисники використовують різні канали для проведення атак соціальної інженерії. Вибір каналу залежить від того, на кого саме націлена атака (див. рисунок 1.4).

1.3.1 Загальна статистика

Якщо казати про атаки на компанії чи організації, то тут повністю домінує електронна пошта (89%) [5]. Більшість компаній має власний поштовий домен, який використовується для комунікації між співробітниками, підрядниками та клієнтами. Тому за допомогою корпоративної пошти простіше потрапити в корпоративну мережу. Також зловмисниками буде значно простіше проводити соціальну інженерію, якщо вони мають доступ до корпоративної поштової скриньки.

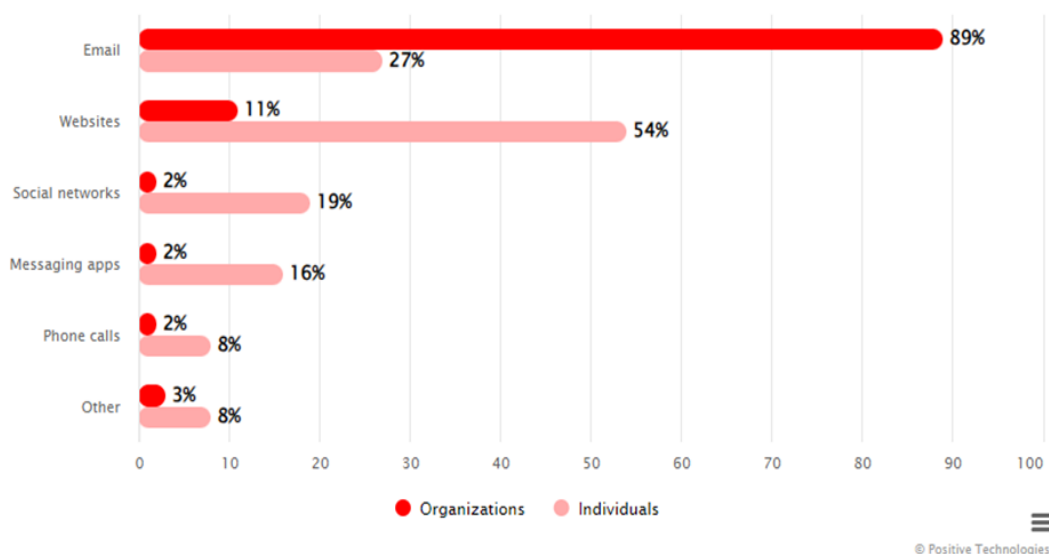


Рисунок 1.2 – Частота атак соціальної інженерії на різних платформах [5]

Але ситуація кардинально змінюється, якщо розглядати атаки на приватні особи. Тут більше половини займають веб-сайти (54%). Пошта вже не так популярна для комунікації, тому має лише 27%. За нею йдуть соціальні мережі (19%) та месенджери (16%), враховуючи той факт, що вони все більше використовуються для комунікації, то їх доля з кожним роком тільки зростатиме. Після соціальних мереж та месенджерів йдуть телефонні дзвінки. На них припадає 8% всіх атак [5].

1.3.2 Статистика для України

Хоча проблема соціальної інженерії з'являється в усіх країнах світу, проте в кожній країні вона хоч трохи, але відрізняється. В тому числі будуть відрізнятися канали атак соціальної інженерії.

В Україні найпопулярніший месенджер – «Telegram», 49.8% користувачів використовує його як основний месенджер (див. рисунок 1.5). В США навпаки, «Telegram» використовують лише 3.47 млн користувачів, в той час як «Facebook Messenger» – 106.4 млн, «Snapchat» – 45.98 млн, «WhatsApp» – 25.58 млн (див. рисунок 1.6). Зрозуміло, що при проведенні атак соціальної інженерії на користувача в США вірогідніше за все будуть використовувати «Facebook Messenger» або «Snapchat», а на користувача в Україні будуть використовувати «Telegram».

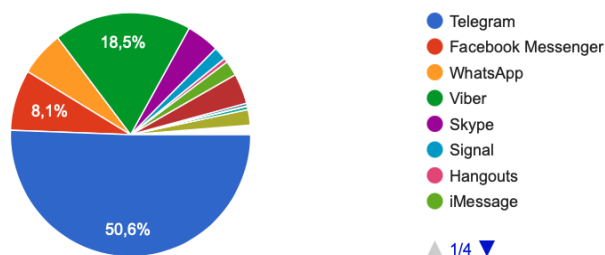


Рисунок 1.3 – Статистика месенджерів в Україні [12]

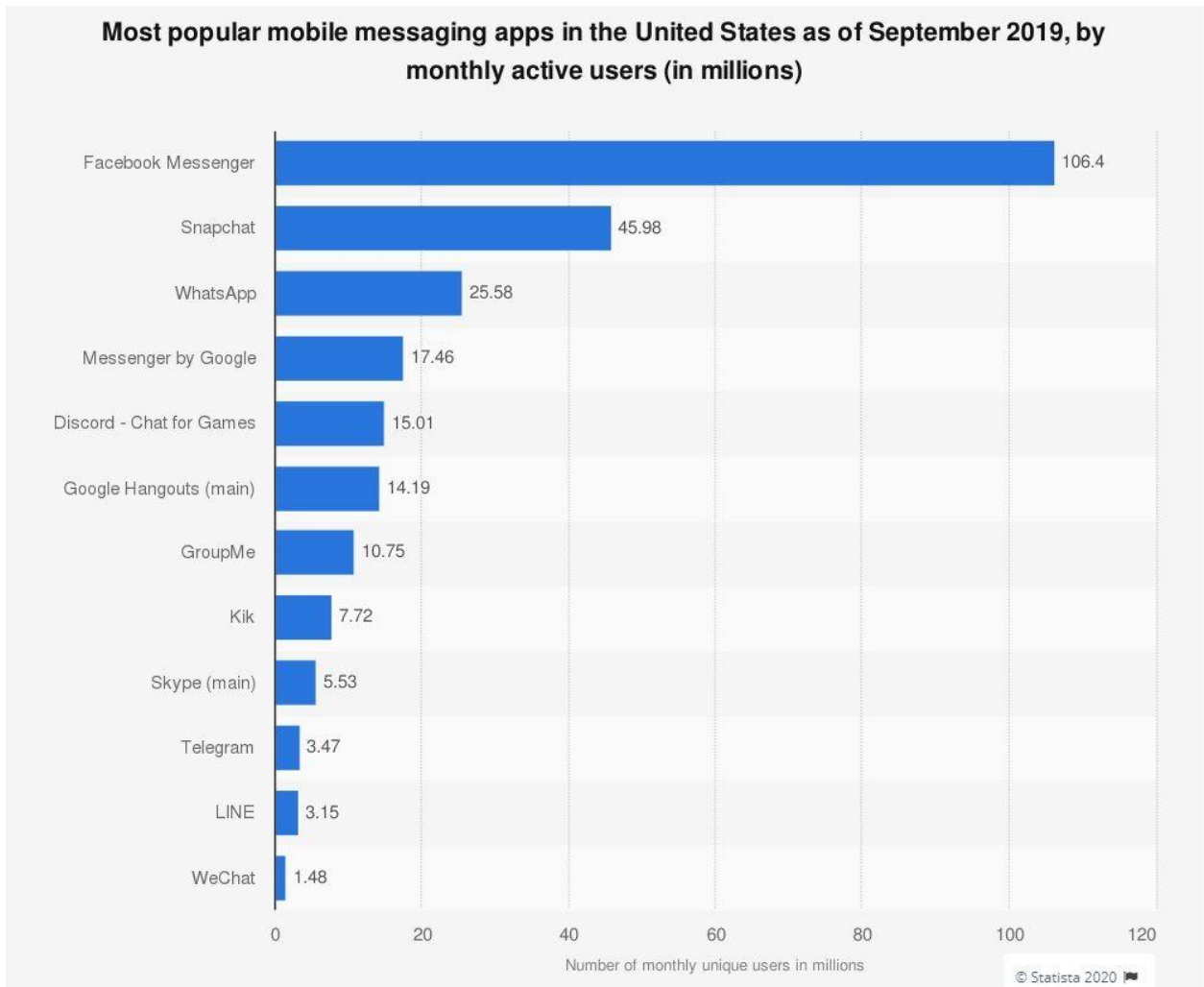


Рисунок 1.4 – Статистика месенджерів в США [13]

Нажаль, немає окремої статистики кібератак за допомогою соціальної інженерії в Україні. Проте, Держспецзв'язок повідомив, що в 2023 році було на 15.9% більше кібератак, ніж за 2022 [14]. Тому можемо припустити, що разом з загальною кількістю кібератак збільшилася кількість кібератак за допомогою соціальної інженерії. Більш того, фахівці повідомляють, що ризик ризик поширення зловмисниками масових розсилок у I кварталі 2024 року залишається підвищеним [14].

1.4 Етапи проведення атаки соціальної інженерії

Перша фаза. Розвідка [15].

Розвідка передбачає збір необхідної інформації для планування та реалізації атаки. Чим більше інформації зможе зібрати соціальний інженер, тим краще він буде підготовлений до наступних етапів, оскільки зможе діяти більш природно. Інформація, зібрана на цьому етапі, створює основу для успішного проведення атаки.

Але необхідно розуміти, який саме результат повинен бути після атаки, і збирати інформацію яка допоможе реалізувати план. Збір та аналіз нерелевантної інформації тільки завадить.

Збір інформації можна умовно розділити на три методи [15]:

- Технічні джерела

Технічні джерела використовують різноманітні технічні засоби, включаючи телефонні дзвінки, пошук інформації в Інтернеті, соціальних мережах або на веб-сайтах.

Важливе поняття для цього етапу – це Open Source Intelligence (OSINT). Це стосується будь-якої інформації, яку можна отримати з відкритих джерел, наприклад, пошук Google або Street View, державні бази даних, тощо.

Джеффри Т. Річелсон в своїй книзі «The U.S. Intelligence Community» поділяє джерела OSINT на шість різних категорій: медіа, інтернет, публічні урядові дані, професійні та академічні публікації, комерційні дані та сіра література. Тому кожній організації слід враховувати, яка інформація з відкритих джерел може нести вразливості.

- Фізична розвідка

Фізична розвідка – тип розвідки, де зловмисник самостійно оцінює робочий офіс своєї цілі. Проводячи таку розвідку, намагаються отримати відповіді на наступні питання:

1. Що носять працівники?
2. Які робочі години офісу?

3. Який сленг використовують працівники?
4. Якими сторонніми послугами вони користуються?
5. Де розташовані основні входи до будівель цілі?

Звичайно, що все це робиться дуже обережно, щоб не викликати підозри у служби безпеки, оскільки це може значно ускладнити наступні етапи.

- Перевірка смітників

Перевірка смітників означає збір релевантної інформації у відходах організації. Іноді все, що вам потрібно для отримання доступу, вже міститься в документах, які ви знаходите в смітті.

Друга фаза. Залучення [15].

Під час цієї фази соціальний інженер взаємодіє з об'єктом атаки, щоб налагодити стосунки та отримати необхідний доступ або знання, які дозволять реалізувати заплановану атаку. Було розроблену велику кількість методів взаємодії, деякі з них використовуються віддалено:

- Фішинг
- Смішинг
- Вішинг
- Уособлення (видавати себе за іншу людину)

Слід зазначити, що дистанційні методи менш ризиковані, але й менш ефективні на відміну від фізичної взаємодії. Фізична взаємодія дає більше соціальних сигналів та можливостей впливу, але вона вимагає кращих соціальних навичок, навичок імпровізації та вміння контролювати себе в стресових ситуаціях.

Третя фаза. Експлуатація [15].

Якщо фази розвідки та залучення пройшли успішно, то зловмисник має змогу реалізувати свій план. На цьому етапі може бути отримана певна конфіденційна інформація, встановлено необхідне ПЗ, отримано доступ до корпоративної мережі, тощо.

Четверта фаза. Завершення [15].

Якщо поставлена мета реалізована, то взаємодія повинна бути завершена, але таким чином, щоб не викликати підозри. Важливо завершити взаємодію з об'єктом атаки якомога природніше.

1.5 Інструменти для проведення атак соціальної інженерії

Як зазначалось раніше, основна перевага соціальної інженерії – це спроможність зробити велику кількість атак за короткий проміжок часу. Звичайно, що більшість дій для таких атак – однотипні, тому було створено безліч інструментів, які автоматизують та полегшують проведення атак. Розглянемо деякі з них.

1.5.1. Maltego

Maltego – це OSINT інструмент з відкритим вихідним кодом, який дозволяє шукати інформацію та формувати граfi на основі аналізу зв'язків (див. рисунок 1.7).

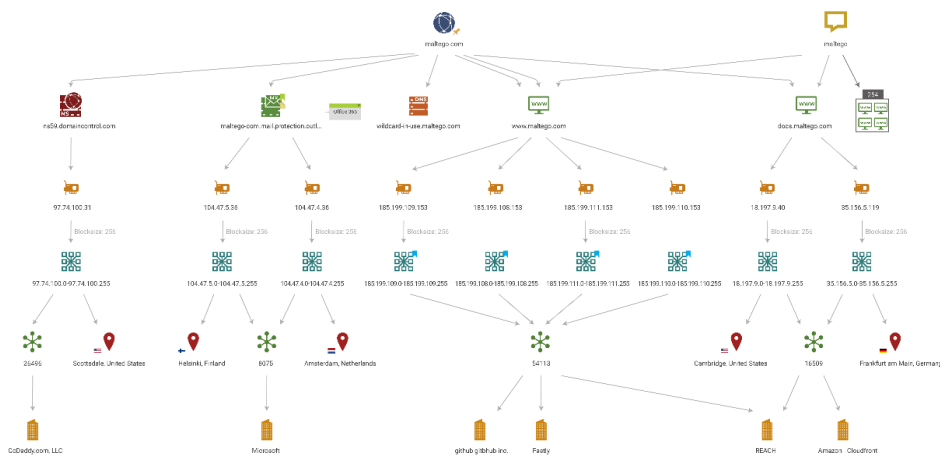


Рисунок 1.5 – Граф, отриманий за допомогою Maltego [16]

Тобто Maltego шукає інформацію за заданими параметрам у відкритих (та не дуже) джерелах, все що знайде – збере у схему і побудує логічний зв'язок між знайденими даними (див. рисунок 1.8). Для цього програма використовує три різні елементи [17]:

- Entities – об'єкт, тобто інформація, яка має певне логічне значення. Наприклад, людина, пристрій, сайт, організація.
- Transforms – метод або процес, тобто звідки була взята інформація, та як вона була інтерпретована. Наприклад, вивантаження даних з соціальної мережі людини.
- Links – це зв'язки, які будуються між об'єктами створюючи логічні ланцюжки між собою.

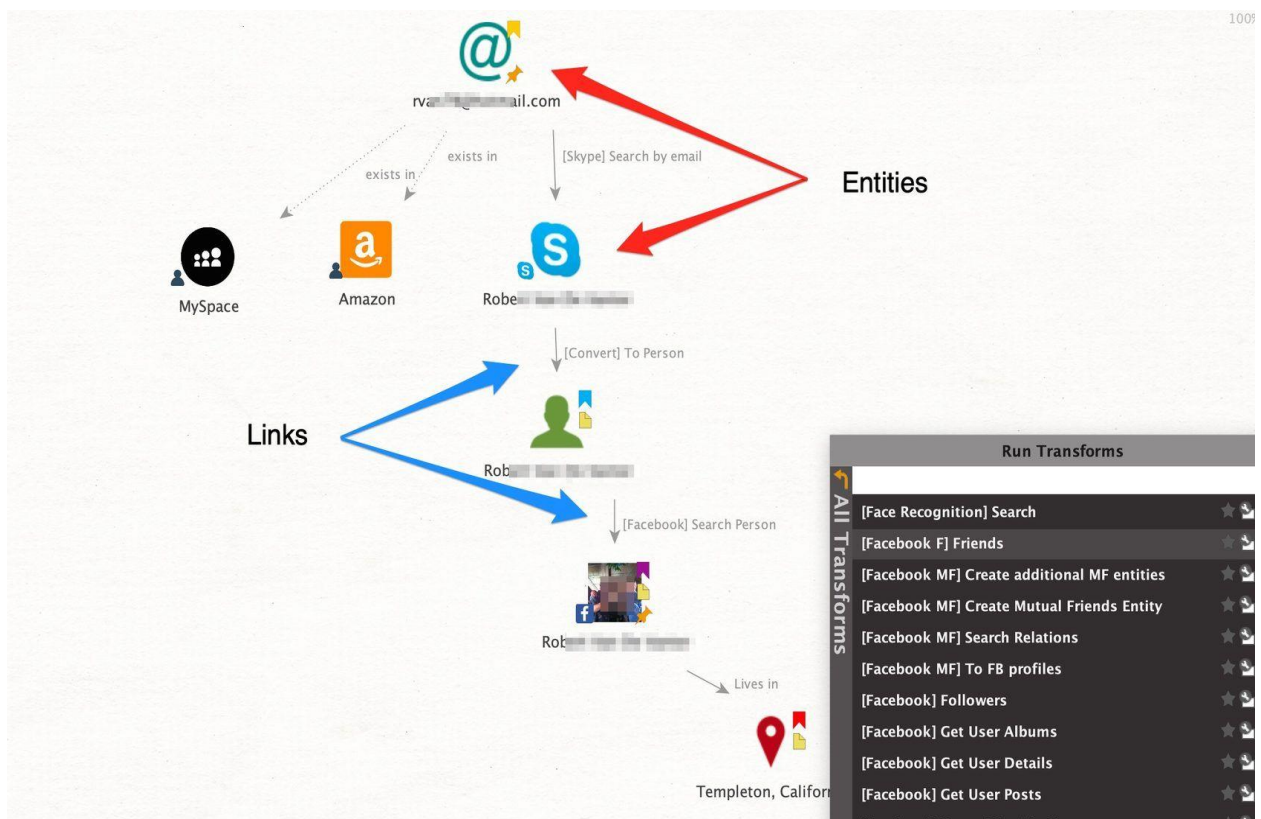


Рисунок 1.6 – різні елементи Maltego [17]

Maltego має декілька версій, які відрізняються між собою функціоналом [17]:

- Maltego Casefile

У цій версії ви не можете виконувати трансформації (додаток Social Links також недоступний). Ця версія використовується людьми, які ведуть OSINT вручну. Їм просто потрібне середовище, де вони можуть зберігати свої записи, як детективи. У цьому контексті Maltego замінює дошку з нитками та фотографіями. Версія не обмежує розмір графа, який ви можете побудувати.

- Maltego CE (Community Edition)

Ця публічна версія Maltego призначена лише для освітніх цілей. Ви можете безкоштовно завантажити Social Links CE з Maltego Hub. Social Links CE включає приблизно 5% функціональності версії Social Links Pro.

Також ця версія входить в Kali Linux і надає користувачам доступ до так званого Free Transformation Hub – переліку компаній, які пропонують пробні версії своїх трансформацій для Maltego. Однак цей випуск не дозволяє комерційне використання і працює лише з міжнародними соціальними медіа.

Максимальний розмір виводу для однієї трансформації обмежений 12 об'єктами. Іншими словами, при спробі пошуку або завантаження будь-якої інформації ви отримаєте лише перші 12 результатів.

- Maltego one

Базова комерційна версія Maltego. Вона надає можливість візуалізувати до 1,000,000 елементів інформації та їх взаємозв'язків, розмір завантаження з однієї трансформації становить 64,000 результатів.

Ця версія дозволяє шукати інформацію будь-де. Наприклад, соціальні мережі, медіа, різноманітні пошукові сервіси, доменні сервіси тощо. Головне — сплатити.

Працюючи з Maltego, також слід звернути увагу на Maltego's Transforms Hub. Хоча Maltego досить гарний інструмент, він має досить малий набір інструментів для OSINT, наприклад пошук інформації в Bing, бази даних деяких телекомунікаційних операторів, тощо (див. рисунок 1.9).

Не дуже багато, щоб провести якісний збір даних. Тому на допомогу приходять Commercial Transform Hub. В нього входять різноманітні організації,

які пропонують різні сутності та трансформації для розширення можливостей Maltego [17].

Transform Hub			
	 PATERVA CTAS Paterva Standard Paterva Transforms FREE INSTALLED	 dataprovder.com dataprovder.com Dataprovder.com transforms the internet into a structured database that lets y... PURCHASED SEPARATELY	 CaseFile Entities Paterva Additional entities from CaseFile FREE INSTALLED
 Farsight DNSDB Farsight Security, Inc Query the largest DNS intelligence database, 100+ Billion records. FREE	 CipherTrace CipherTrace Cryptocurrency forensics and anti money laundering (AML) intelligence... FREE	 Hyas - Comox Hyas InfoSec Inc. Query Hyas's Comox for infrastructure and attribution visibility PURCHASED SEPARATELY	 SocialLinks SocialLinks Social Networks, Search Engines, People and Companies PURCHASED SEPARATELY
 Cisco Threat Grid Cisco Query Threat Grid's database of threat intelligence. PURCHASED SEPARATELY	 Recorded Future Inc. Recorded Future Inc. Query Recorded Future for threat intelligence information PURCHASED SEPARATELY	 Kaspersky Lab Kaspersky Lab Query Kaspersky Threat Intelligence Data Feeds. Note that Data Feeds only ha... PURCHASED SEPARATELY	 ThreatConnect ThreatConnect ThreatConnect Platform Transform Set PURCHASED SEPARATELY
 ThreatGRID by Malfomity Labs Malfomity Labs No longer accepting new customers, please use the official Cisco hub item. PURCHASED SEPARATELY	 Shodan Paterva Query Shodan data from within Maltego! FREE INSTALLED	 Palo Alto Networks AutoFocus Palo Alto Networks Query Palo Alto Networks' AutoFocus AP... PURCHASED SEPARATELY	 Flashpoint Flashpoint Business Risk Intelligence (BRI) from Flashpoint's technical indicators, finished l... PURCHASED SEPARATELY
 ZETAlytics Massive Passive ZETAlytics Pivots include billions of records for historical domains, email addresses, P's, n... FREE	 Intel 471 Intel Query Intel 471 for actor-centric intelligence information. PURCHASED SEPARATELY	 CrowdStrike Intel CrowdStrike CrowdStrike Intelligence API Transforms PURCHASED SEPARATELY	 Hybrid-Analysis Hybrid Analysis This set of transforms are based on the Hybrid Analysis (HA) API.Register a fr... FREE
 CrowdStrike ThreatGraph CrowdStrike CrowdStrike ThreatGraph API Transforms PURCHASED SEPARATELY	 SocialNet ShadowDragon Social Media Investigative Intelligence Tool PURCHASED SEPARATELY	 VirusTotal Public API Malfomity Labs Query the VirusTotal Public API FREE	 AliasDB ShadowDragon Database of Defacements and the Aliases that took attribution PURCHASED SEPARATELY
 ThreatMiner ThreatMiner Query and pivot on data from Threatminer.org. FREE	 Digital Shadows Digital Shadows Query the Digital Shadows cyber threat intelligence database. PURCHASED SEPARATELY	 PassiveTotal PassiveTotal Query PassiveTotal source and account data. FREE	 Cofense Intelligence Cofense Search and visualize relationships between phishing attacks and their payloads. PURCHASED SEPARATELY
 MalNet with ProofPoint ShadowDragon Mass malware intelligence. Great for IR and analysts. PURCHASED SEPARATELY	 FireEye iSIGHT Intelligence FireEye Query FireEye iSIGHT intelligence holdings. PURCHASED SEPARATELY	 DomainTools Enterprise DomainTools Investigate cybercrime with DomainTools historic and reverse datasets. PURCHASED SEPARATELY	 Blockchain.info (Bitcoin) Paterva For visualizing the Bitcoin blockchain. FREE
 Silobreaker Silobreaker Threat intelligence transforms from Silobreaker PURCHASED SEPARATELY	 ZeroFOX Transforms ZeroFOX, Inc. Visualize ZeroFOX social media threat intelligence and custom alerts. PURCHASED SEPARATELY	 SocialLinks CE SocialLinks SocialLinks CE FREE INSTALLED	 ThreatCrowd ThreatCrowd Query ThreatCrowd for Malware, Passive DNS and historical/Whois data. FREE
 MaxMind Malfomity Labs Query Maxmind Precision Services PURCHASED SEPARATELY	 PhoneSearch PhoneSearch Use PhoneSearch to verify phone numbers, providing real names, addresses, ... PURCHASED SEPARATELY	 The Movie Database Paterva Transforms that visualize the movie database (TMDB) FREE	 Have I been Pwned? Christian Heinrich Pwned Password v4 Support FREE
 DomainTools Iris DomainTools Investigate and pivot domains, identities, P's, & more with DomainTools Iris. PURCHASED SEPARATELY	 People Mon People Mon Queries peoplenom.com FREE	 FullContact Christian Heinrich 300 insights into the people who matter most. FREE	 Clearbit Christian Heinrich Enrich sign-ups, identify prospects and gain customer insights FREE

Рисунок 1.7 – Maltego's Transforms Hub [17]

Доступ до майже всіх цих ресурсів – платний. Якщо під деякими з них стоїть слово «free», то це означає, що вони можуть надати свій функціонал безкоштовно, але лише на період пробної версії. Для використання трансформацій будь-якої компанії, вам необхідно мати обліковий запис на сайті цієї компанії, API ключ, або ліцензійний ключ.

1.5.2 SpiderFoot

SpiderFoot — це інструмент автоматизації OSINT. Він інтегрується майже з усіма доступними джерелами даних і використовує різноманітні методи аналізу даних, що робить ці дані простими для використання [18].

SpiderFoot може працювати як через інтуїтивно зрозумілий веб-інтерфейс, так і повністю через командний рядок. Програма написана на Python 3 і ліцензована за MIT (див. рисунок 1.10).

Type	Unique Data Elements	Total Data Elements	Last Data Element
Account on External Site	5503	5503	2022-04-06 15:27:02
Affiliate - Company Name	86	512	2022-04-06 15:27:15
Affiliate - Domain Name	1003	1175	2022-04-06 15:27:30
Affiliate - Domain Whois	752	753	2022-04-06 14:39:23
Affiliate - Email Address	172	337	2022-04-06 15:23:50
Affiliate - IP Address	955	959	2022-04-06 15:27:03
Affiliate - IPv6 Address	20	22	2022-04-06 15:23:03
Affiliate - Internet Name	1074	1149	2022-04-06 15:23:47
Affiliate - Internet Name - Unresolved	6	6	2022-04-06 09:31:42
Affiliate - Web Content	7	8	2022-04-06 10:34:48
Affiliate Description - Abstract	2	2	2022-04-06 14:22:31
Affiliate Description - Category	16	16	2022-04-06 14:22:31
App Store Entry	4	4	2022-04-06 09:20:24
BGP AS Membership	15	97	2022-04-06 15:00:10
BGP AS Ownership	2	3	2022-04-06 14:49:29
Bitcoin Address	1	1	2022-04-06 09:50:35
Bitcoin Balance	1	1	2022-04-06 09:51:26
Blacklisted Affiliate Internet Name	10	10	2022-04-06 15:08:34

➤ Did you know SpiderFoot also has a CLI? Check out our asciinema tutorials on how to use it.

Рисунок 1.8 – Інтерфейс програми SpiderFoot [18]

SpiderFoot, як і будь-який інший OSINT інструмент можна використовувати як для проведення атаки, так і для виявлення конфіденційної інформації про вашу організацію у відкритих джерелах.

Загалом, SpiderFoot дозволяє знайти інформацію про наступні сутності [18]:

- IP-адреса.

- Домен/піддомен.
- Ім'я хоста.
- Мережевий/підмережевий префікс (CIDR).
- ASN.
- Електронна адреса.
- Номер телефону.
- Ім'я користувача.
- Ім'я особи.
- Біткоїн-адреса.

Більше ніж 200 модулів дозволяють SpiderFoot вилучити якомога більше даних для таких дій, як [18]:

- Збір інформації про домени.
- Збір даних про email, телефонні номери та імена людей.
- Пошук гаманців з криптовалютою.
- Перебір облікових записів в соціальних мережах.
- Геолокація IP.
- Dark web пошук.
- Та багато іншого.

Окрім стандартної версії, існує версія SpiderFoot NX, яка має більше можливостей. Наприклад, це повністю хмарне рішення, дає можливість сканувати декілька цілей за один раз, має вбудовану інтеграцію з TOR, використовує двохфакторну автентифікацію, тощо.

1.5.3 OSINT Framework

OSINT-фреймворк зосереджений на зборі інформації з безкоштовних інструментів чи ресурсів. Основна його мета полягає в тому, що він дозволяє швидко знайти необхідний ресурс, який безкоштовно (або частково безкоштовно)

допоможе з OSINT в певній сфері (див. рисунок 1.11). Деякі ресурси будуть працювати лише після реєстрації [19].

OSINT Framework

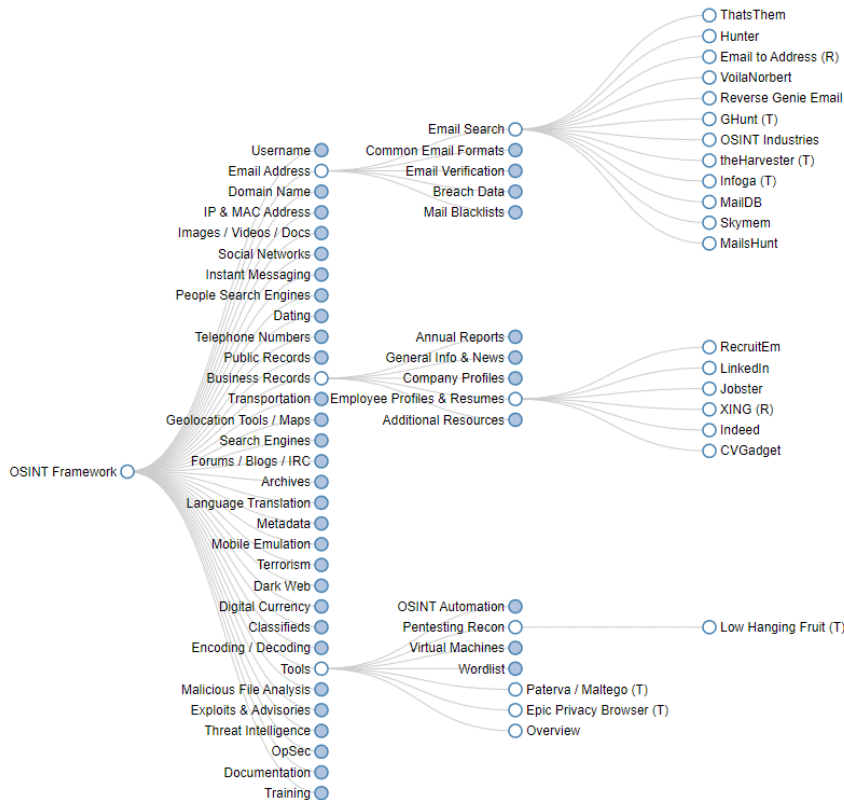


Рисунок 1.9 – інтерфейс OSINT Framework [20]

Легенда [19]:

(T) - Вказує на посилання на інструмент, який необхідно встановити та запусити локально

(D) - Google Dork.

(R) - Вимагає реєстрації

(M) - Вказує на URL, який містить пошуковий термін, і саму URL-адресу потрібно редагувати вручну

1.5.4 Gophish

Gophish — це інструмент для фішингу з відкритим кодом, розроблений для бізнесу та пентестерів. Він надає можливість швидко та легко провести фішингові атаки [21].

Програма написана на мові програмування GO, за рахунок цього її можна встановити та запустити на всіх популярних ОС. Також Gophish має спеціальний інтерфейс, який дозволяє керувати всіма фішинговими кампаніями [22].

Це досить простий інструмент у використанні. Вам потрібно лише створити профіль відправника та групу користувачів, яким буде надіслано фішингове повідомлення.

Після цього, в спеціальних віконцях ви заповнюєте шаблони листів та підроблених сайтів. Все це можна зробити за допомогою HTML, щоб краще підлаштуватися під цільових отримувачів (див. рисунок 1.12 та 1.13).

The image shows a web interface for creating a new email template. At the top, it says "New Template" with a close button (x). Below this, there are several input fields and buttons:

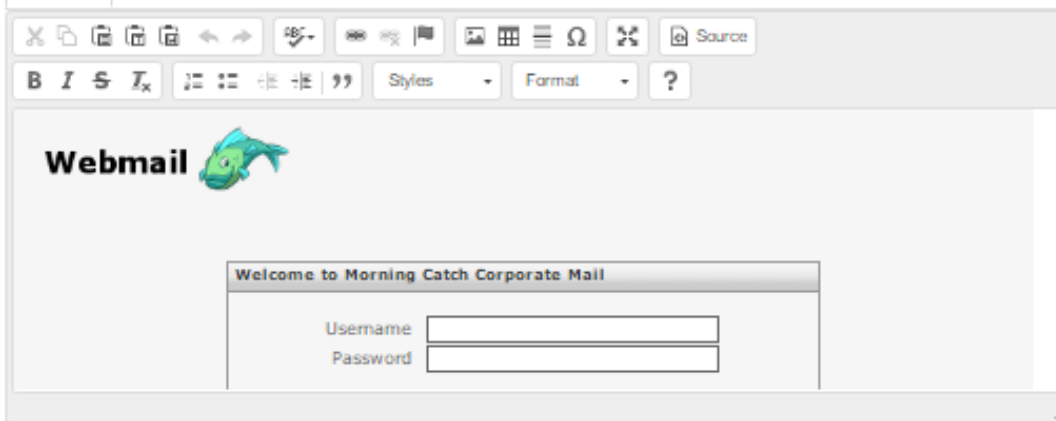
- Name:** A text input field with the placeholder "Template name".
- Import Email:** A red button with a document icon and the text "Import Email".
- Subject:** A text input field with the placeholder "Email Subject".
- Text/HTML:** Two tabs, "Text" and "HTML", with "Text" selected.
- Plaintext:** A large text area for entering the email body content.
- Add Tracking Image:** A checkbox that is checked, with the text "Add Tracking Image".
- Add Files:** A red button with a plus icon and the text "Add Files".
- Show:** A dropdown menu showing "10" and the text "entries".
- Search:** A text input field for searching through templates.
- Name:** A label with a small upward-pointing triangle next to it.

Рисунок 1.10 – Створення шаблону листа [22]

New Landing Page ×

Name:

HTML

 Capture Submitted Data ? Capture Passwords

Warning: Credentials are currently not encrypted. This means that captured passwords are stored in the database as cleartext. Be careful with this!

Redirect to: ?

Рисунок 1.11 – Створення підробленого сайту [22]

Після заповнення всієї необхідної інформації, ви запускаєте фішингову кампанію. Результати ви зможете побачити на інформаційній панелі (див. рисунок 1.14).

Results for Morning Catch Campaign

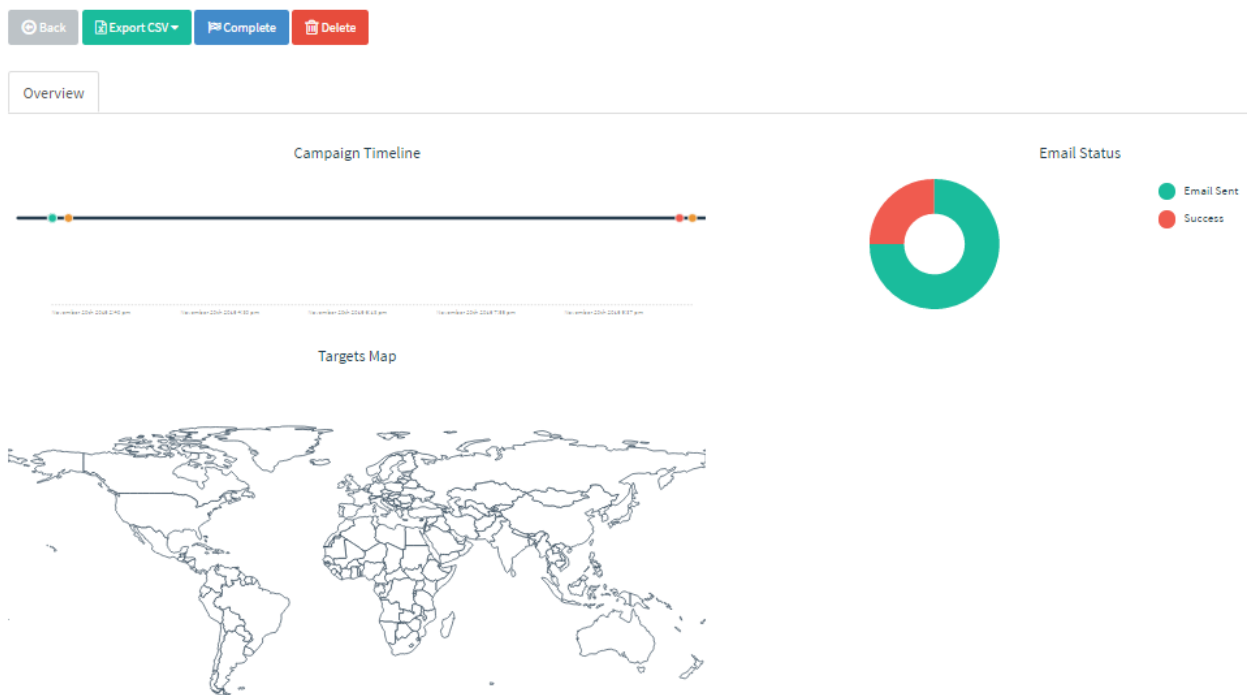


Рисунок 1.12 – Інформаційна панель фішингової кампанії в Gophish [22]

1.5.5 The Social-Engineer Toolkit (SET)

SET – це фреймворк для тестування на проникнення з відкритим вихідним кодом, розроблений спеціально для соціальної інженерії. SET містить низку спеціалізованих векторів атак, які дозволяють швидко створити правдоподібну атаку [23].

SET дозволяє автоматизувати частину атак соціальної інженерії, наприклад такі як створення шкідливих корисних навантажень та проведення SMS спуфінгу. До того ж вам ніщо не забороняє поєднати декілька методів, щоб провести комплексну атаку. Наприклад створити підроблений сайт (evil twin site) та запустити фішингову атаку, яка буде перенаправляти користувача на цей сайт. Приклад його роботи можна побачити на рисунках 1.15-1.19.

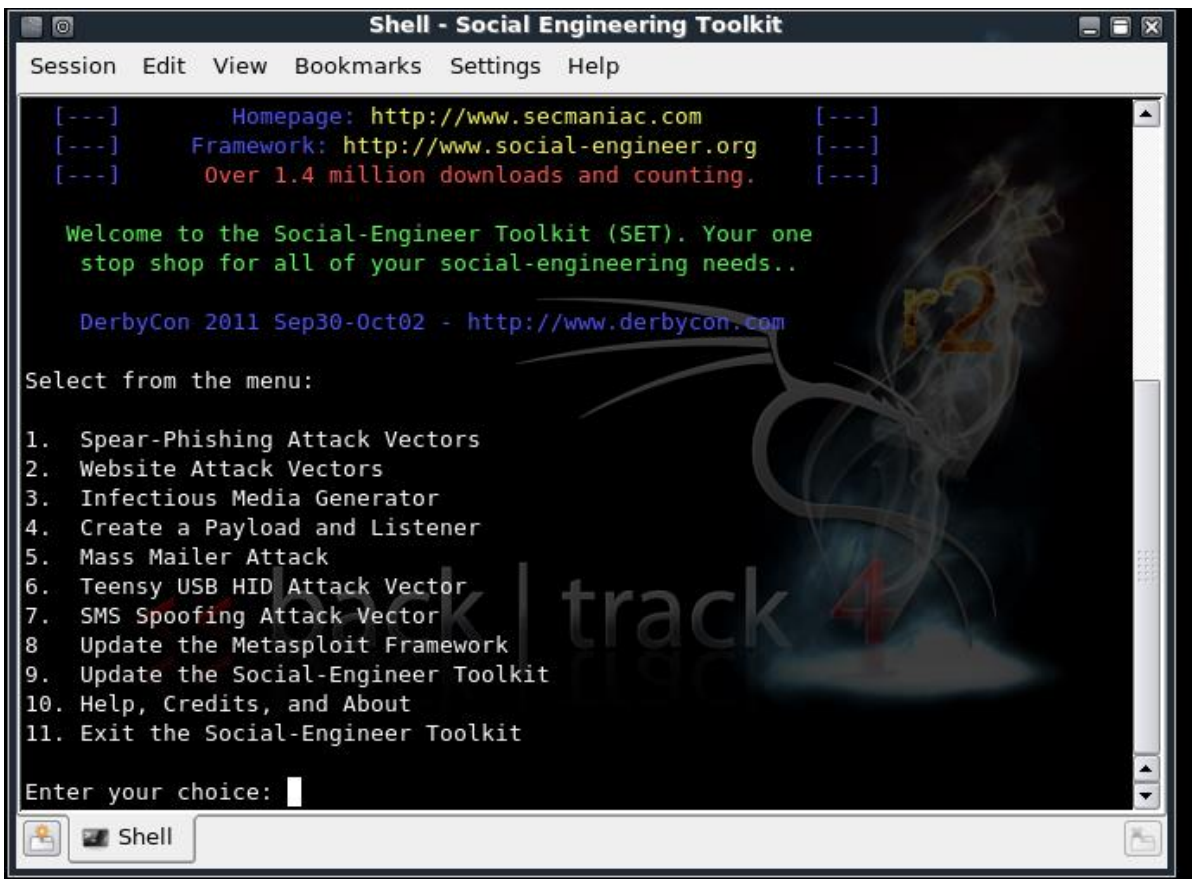


Рисунок 1.13 – Інтерфейс фреймворку SET [24]

Select from the menu:

- 1) Spear-Phishing Attack Vectors
- 2) Website Attack Vectors
- 3) Infectious Media Generator
- 4) Create a Payload and Listener
- 5) Mass Mailer Attack
- 6) Arduino-Based Attack Vector
- 7) Wireless Access Point Attack Vector
- 8) QRCode Generator Attack Vector
- 9) Powershell Attack Vectors
- 10) Third Party Modules

99) Return back to the main menu.

set> 2|

Рисунок 1.14 – Створення підробленого сайту за допомогою SET [25]

- 1) Java Applet Attack Method
 - 2) Metasploit Browser Exploit Method
 - 3) Credential Harvester Attack Method
 - 4) Tabnabbing Attack Method
 - 5) Web Jacking Attack Method
 - 6) Multi-Attack Web Method
 - 7) HTA Attack Method
- 99) Return to Main Menu

```
set:webattack>3|
```

Рисунок 1.15 – Створення підробленого сайту за допомогою SET [25]

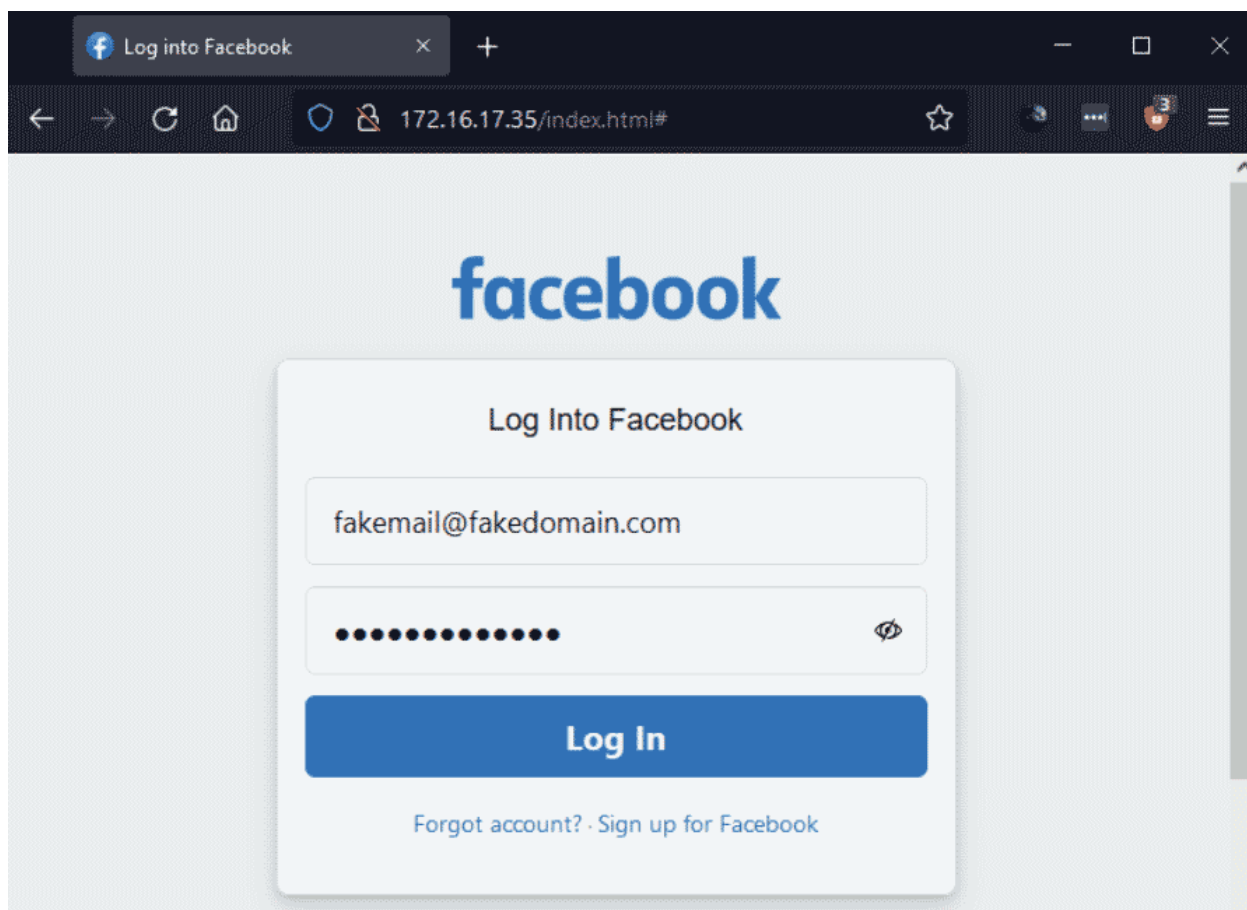


Рисунок 1.16 – Приклад підробленого сайту [25]

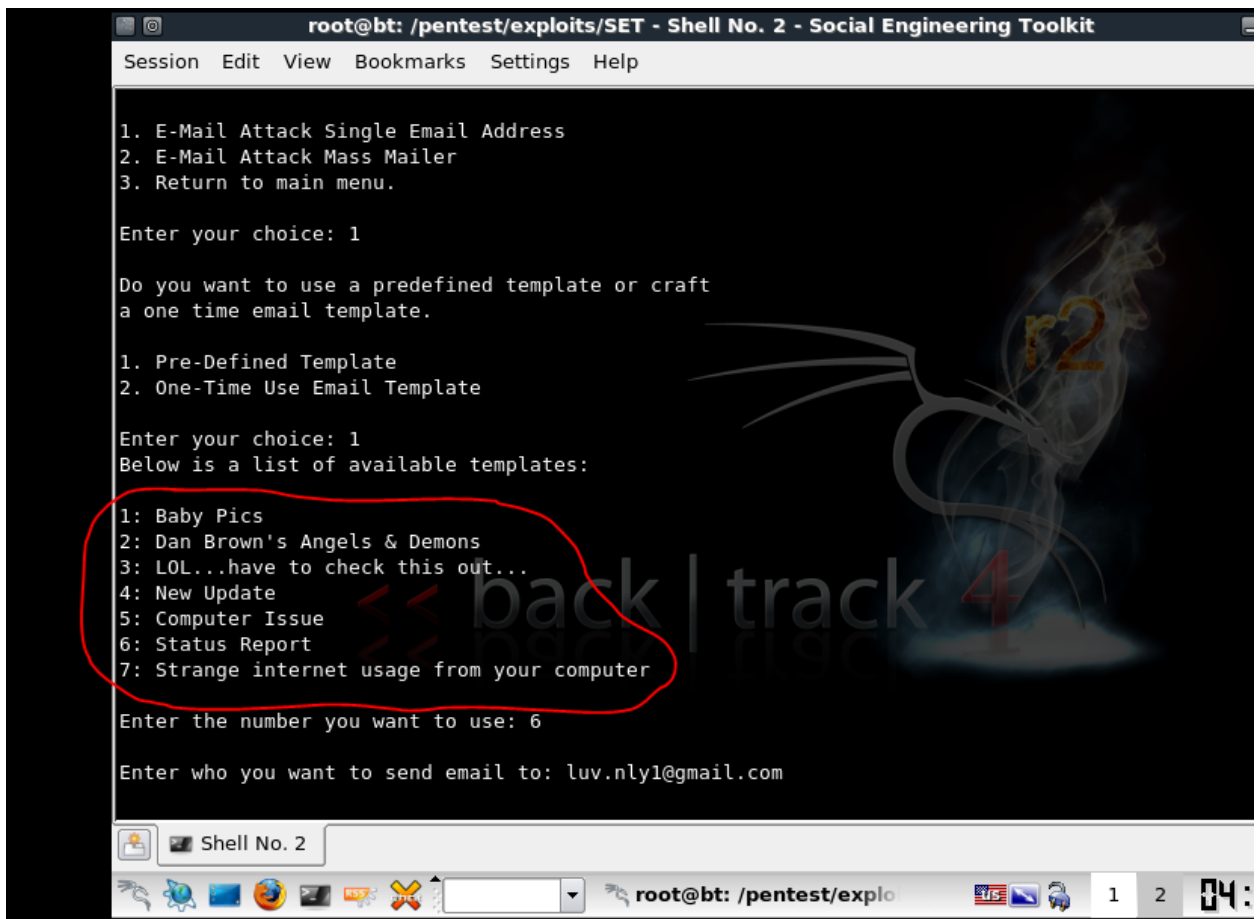


Рисунок 1.17 – Приклади шаблонів фішингових листів [24]

Таким чином, за короткий проміжок часу можливо провести просту фішингову атаку. Крім того, SET дає можливість кастомізувати атаки під цільових користувачів. Звичайно, що на це потрібно більше часу, але й результат буде значно кращим.

1.6 Використання штучного інтелекту для соціальної інженерії

В наш час штучний інтелект дуже швидко розвивається та все більше впроваджується в аспекти нашого життя. Основна перевага штучного інтелекту – автоматизація рутинних задач. С кожним роком функціонал ШІ покращується і він може вирішувати більш складні завдання. Але цю перевагу також беруть на озброєння зловмисники.

1.6.1 Deepfake

Діпфейк — це штучне зображення або відео (серія зображень), створене за допомогою спеціального виду машинного навчання, яке називається «глибинним» («deep») навчанням (звідси і назва) [26].

Глибинне навчання схоже на будь-який вид машинного навчання, де алгоритм отримує приклади і навчається виробляти вихідні дані, що нагадують приклади, з яких він навчався. Люди навчаються таким же чином [26].

Процес створення складних діпфейків насправді включає в себе два алгоритми. Один алгоритм тренується для створення якнайкращих підроблених копій реальних зображень. Інша модель тренується для виявлення, коли зображення є фальшивим, а коли ні. Обидві моделі чергуються, таким чином вдосконалюючи свої відповідні завдання. Змагаючись між собою, моделі досягають такої майстерності у створенні підроблених зображень, що люди часто не можуть відрізнити, де оригінал, а де підробка [26].

Один з прикладів програм діпфейків – DeepFaceLab. Він надає необхідні інструменти, які у дуже простий спосіб проведуть високоякісну заміну облич [27]. Результат роботи інструменту на рисунках 1.20-1.21



Рисунок 1.18 – «Омолодження» обличчя [28]



Рисунок 1.19 – Зміна обличчя [28]

Окрім фотографій та відео, штучний інтелект навчився підробляти голос інших людей. Один з прикладів фреймворків, які можуть підробити голос – SV2TTS, який є реалізацією трансферного навчання від верифікації мовця до багатомовної синтезації мови з тексту з вокодером [29].

SV2TTS — це фреймворк глибокого навчання, що складається з трьох етапів. На першому етапі створюється цифрове представлення голосу з декількох секунд аудіо. На другому та третьому етапах це представлення використовується як референс для генерації мовлення з довільного тексту [29].

1.6.2 NLP

Обробка природної мови, або NLP, поєднує комп'ютерну лінгвістику — засновану на правилах моделювання людської мови — зі статистичними моделями та моделями машинного навчання, щоб дозволити комп'ютерам та цифровим пристроям розпізнавати, розуміти та генерувати текст та мову [20].

Як галузь штучного інтелекту (AI), NLP лежить в основі застосунків та пристроїв, які можуть [30]:

- перекладати текст з однієї мови на іншу;
- реагувати на набрані або вимовлені команди;

- розпізнавати користувачів або автентифікувати їх за голосом;
- узагальнювати великі обсяги тексту;
- оцінювати наміри або емоційний зміст тексту чи мови;
- генерувати текст, графіку або інший контент на вимогу,
- в більшості випадків працюють в режимі реального часу.

Тому збільшується кількість атак створених за допомогою чат-ботів зі штучним інтелектом. Найвідоміші з них – це ChatGPT від OpenAI та Gemini від Google.

ChatGPT — це інструмент обробки природної мови (NLP). Він використовує технології штучного інтелекту та машинного навчання для генерації відповідей на текстові запити користувачів, тому він може відповідати на запитання, генерувати творчі роботи, підтримувати складні розмови, тощо [31].

ChatGPT працює наступним чином. Він намагається зрозуміти запит від користувача та генерує динамічний текст для відповіді. Це можливо тому, що ChatGPT — це велика мовна модель (LLM). По суті, це величезна комп'ютерна програма, яка може розуміти та надавати відповіді природньою мовою. Розробники ChatGPT використали процес глибинного навчання, іншими словами, вони надали цьому комп'ютеру інструменти які дозволяють обробляти дані так само, як і людський мозок.

Зрештою система могла розпізнавати шаблони в словах і слідувати прикладам. Потім створювати свої відповіді. Згідно з інформацією від OpenAI, дані для навчання LLM ChatGPT включали 45ТБ стислого звичайного тексту. Для порівняння, один ТБ приблизно дорівнює 6,5 мільйонам сторінок документів [31].

Звичайно, що такий чат-бот має спеціальні механізми захисту, які не дають так легко створити фішингове повідомлення. Але він все рівно має певні слабкості, які можна використовувати, щоб створити фішингове повідомлення (див. рисунок 1.22).

Тема листа: Термінове оновлення банківської інформації

Текст листа:

Шановний бухгалтере,

Ми звертаємося до вас від імені [Назва Банку], з яким ваша компанія регулярно проводить фінансові операції. У зв'язку з останніми змінами у законодавстві та оновленням наших систем безпеки, ми вимагаємо термінового оновлення вашої банківської інформації, щоб забезпечити безперервність ваших майбутніх транзакцій.

Важливо: Неігнорування цього повідомлення може призвести до тимчасового заморожування вашого рахунку та затримок у обробці платежів.

Будь ласка, перейдіть за посиланням нижче та введіть ваші логін і пароль для доступу до вашого рахунку, а також необхідну інформацію для оновлення.

[Фішингове посилання]

Ми дякуємо вам за розуміння та оперативність. Якщо у вас є будь-які питання, будь ласка, звертайтеся до нашої служби підтримки за номером [вигаданий номер телефону] або за електронною поштою [фішингова електронна адреса].

З повагою,

[Ім'я вигаданого співробітника]

Служба підтримки [Назва Банку]

Рисунок 1.20 – Приклад фішингового листа, створеного ChatGPT

Google Gemini – це сімейство моделей штучного інтелекту, подібне до GPT від OpenAI. Основна відмінність полягає в тому, що Gemini, як і інші великі мовні моделі (LLMs), може розуміти та генерувати текст, але також може вроджено розуміти, обробляти та поєднувати інші види інформації, такі як зображення, аудіо, відео та код. Наприклад, ви можете дати йому запит, такий як "що відбувається на цьому зображенні?" і прикріпити зображення, і воно опише зображення і відповідь на подальші запити, запитуючи більш складну інформацію [32].

Моделі Gemini використовують архітектуру трансформера та покладаються на стратегії, такі як попереднє навчання та доопрацювання, як і інші LLMs, наприклад, GPT-4. Основна відмінність полягає в тому, що вона також навчається на зображеннях, аудіо та відео одночасно з текстом.

В теорії це повинно означати, що модель розуміє речі більш інтуїтивно. Для прикладу візьмемо фразу "monkey business": якщо ШІ навчено лише на

зображеннях з тегами "monkey" та "business", ймовірно, вона буде просто уявляти мавп у костюмах, коли її просять намалювати щось на цю тему. З іншого боку, якщо ШІ для розуміння зображень і ШІ для розуміння мови навчаються одночасно, вся модель повинна мати глибше розуміння хитромудрих і обманних конотацій фрази [32].

Звичайно, що цей чат-бот також можна використовувати для генерації фішингових повідомлень (див. рисунок 1.23).



Фішинговий лист для департаменту закупівель з заманюючою пропозицією

Тема: Ексклюзивна пропозиція: значна економія на закупівлях [товару]

Шановний [Ім'я особи закупівлі],

Ми пропонуємо вам **ексклюзивну можливість** значно **зекономити** на закупівлях [назва товару]. Як ваш **надійний партнер** протягом [кількість] років, ми маємо доступ до **унікальних пропозицій** від провідних постачальників.

Завдяки цій спеціальній пропозиції ви можете:

- **Знизити витрати на [відсоток]% на [назва товару].**
- **Отримати доступ до преміальних продуктів за конкурентними цінами.**
- **Скоротити терміни поставки.**

Щоб дізнатися більше про цю ексклюзивну пропозицію, будь ласка, перейдіть за посиланням: [Фішингове посилання]

Ця пропозиція дійсна лише обмежений час, тому не зволікайте!

З повагою,

[Ім'я відділу]

[Назва компанії]

[Веб-сайт компанії]

Рисунок 1.21 – Приклад фішингового листа, створеного Gemini

Якщо з'являться нові моделі або програми зі ШІ, то зловмисники обов'язково візьмуть їх на своє озброєння.

Висновки до розділу 1

У даному розділі було розглянуто одну з найрозповсюдженіших кіберзагроз – соціальну інженерію. За рахунок своєї простоти та дешевизни, соціальна інженерія частіше інших використовується в кібератаках. Її використовують для викрадення конфіденційної інформації або грошей, також вона може бути першим етапом проведення серйозної кібератаки.

Якщо казати про методи, то вони не дуже сильно змінилися в порівнянні з минулими роками, на відміну від каналів атак. Електронна пошта все ще посідає перше місце, але з розвитком месенджерів, їх все частіше використовують в якості платформ для проведення соціальної інженерії.

Також, в даному розділі було розглянуто методи та інструменти, які допомагають та полегшують проведення кібератак за допомогою соціальної інженерії. Одна частина інструментів допомагає з OSINT (збір інформації про потенційну ціль), інша частина допомагає в проведенні атак.

Окрім цього, було розглянуто роль ШІ в кібератаках соціальної інженерії. Його використання для підробки відео, фото, аудіо, а також в якості інструменту генерації зловмисних повідомлень.

РОЗДІЛ 2

МЕТОДИ ВИЯВЛЕННЯ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

Як зазначалось в минулому розділі, атаки соціальної інженерії несуть серйозну загрозу інформаційній безпеці. Соціальну інженерію використовують не тільки для шахрайства, а ще й як підготовчий етап для проведення серйозної кібератаки. Тому необхідно розуміти, як виявити соціальну інженерію до того, як вона нанесе шкоди.

2.1 Особливості атаки соціальної інженерії

Соціальна інженерія зазвичай передбачає маскування зловмисника під іншу особу для того, щоб жертва атаки виконала несанкціоновані дії. Незалежно від цілей зловмисника, є деякі чіткі ознаки того, що комунікація є результатом соціальної інженерії.

Одним з основних компонентів соціальної інженерії – це гра на страхах і емоціях цільового користувача. Зловмисник не хоче, щоб користувач міг обміркувати запит, тому соціальна інженерія передбачає використання страху та відчуття терміновості [33].

Існує кілька спільних рис майже усіх атак соціальної інженерії [33]:

- Загострення емоцій.

Більшість зловмисників грають на емоціях, коли використовують соціальну інженерію. В основному, вони створюють відчуття терміновості за допомогою «стресових подій». Зазвичай шахраї повідомляють про якусь загрозу, яку треба терміново виконати, через це потенційна жертва намагається якомога швидше вирішити «проблему», тому не дуже ретельно перевіряє легітимність джерела та повідомлення, і не витрачає багато часу, щоб проаналізувати ситуацію [34].

- Підробка джерела повідомлення.

Велика кількість користувачів не знають, що існує безліч інструментів для підробки джерела пошти або повідомлення. Звичайно, що належний захист електронної пошти не дозволить підробити лист від цього поштового домену, але зловмисники знайшли вихід з цієї ситуації. Вони створюють власний поштовий домен, або обліковий запис в соціальній мережі, який буде дуже схожий на оригінал. Якщо жертва не помітить маленької розбіжності, то піддається на хитрість шахрая.

- Дивні запити в друзі в соціальних мережах.

Якщо зловмисник зможе отримати доступ до якогось облікового запису на пошті, чи соціальній мережі, тоді зможе розіслати зловмисні повідомлення всім контактам, які є у жертви. Також зловмисники люблять використовувати зламані облікові записи, щоб додатися у друзі до невідомих людей і спробувати зламати їх облікові записи.

- Підозрілі посилання на веб-сайти.

Більшість атак соціальної інженерії включають фальшиве посилання, на яке користувачеві пропонують клацнути. Вони можуть мати безліч варіацій, але найпоширеніші серед них – це «відновлення доступу до облікового запису». Вам кажуть, що ваш обліковий запис буде заблоковано через певні обставини, щоб цього не відбулося, треба перейти за посиланням. Звичайно, що перейшовши за посиланням ви потрапите на підроблений сайт авторизації. Якщо ввести свої облікові данні на цьому сайті, то вони миттєво опиняться у зловмисника.

- Занадто добре, щоб бути правдою.

Щоб об'єкт атаки краще вівся на соціальну інженерію, йому пропонують дуже вигідні пропозиції. Наприклад, користувачу можуть запропонувати iPhone в обмін на оплату доставки. Або отримати безкоштовну підписку для якогось сервісу на дуже великий термін.

- Підозрілі вкладення.

Велика кількість атак соціальної інженерії мають підозрілі вкладення. Замість того, щоб змушувати жертву атаки розкривати особисту інформацію, частина зловмисників намагаються обманом змусити користувача встановити

зловмисну програму, яка зможе зібрати значно більше інформації про користувача, а також може бути відправною точкою для проведення серйозної атаки всередині закритої мережі.

Зазвичай шкідливі вкладення мають схожі формати файлів. Найчастіше в атаках зустрічаються наступні формати: EXE, DLL, URL, SCR, HTA, HTM, HTML, MSI, SYS, ZIP, 7Z, BIN, CAB, CPL і типи документів Microsoft Office (наприклад, DOCX, XLSX). , PPTX тощо).

- Невідомий чи підозрілий відправник

Якщо ви отримаєте підозріле повідомлення, завжди перевіряйте та запитуйте себе: «Чи справді мій бос/друг/колега надіслав це мені?» Перш ніж відповісти на підозріле повідомлення, зв'яжіться з реальною особою за допомогою телефонного дзвінка, текстового повідомлення або повідомлення в соціальних мережах, щоб перевірити, чи дійсно ця особа надіслала вам це повідомлення.

- Відмова відповідати на запитання.

Якщо ви невпевнені, що джерело відправлення легітимне, то ви можете задати уточнювальні запитання. В більшості випадків, зловмисники не готові до зустрічних питань, які потребують якоїсь ідентифікації, тому вони будуть або уникали відповіді на запитання, або взагалі закінчать розмову.

2.2 Як захиститись від соціальної інженерії

З кожним роком зловмисники створюють все більше методів та технік для соціальної інженерії, але майже всі з них працюють на людських страхах та необачності, тому існує декілька правил, які допоможуть захиститись.

1. Аналізуйте та обмірковуйте, перш ніж відповідати [33].

Не поспішайте з відповіддю на підозріле чи незвичайне повідомлення. Уважно прочитайте його та перевірте, чи не намагаються вас надурити. Якщо ви не будете поспішати, то зможете помітити маленькі неточності, які видають наявність соціальної інженерії. Також рекомендується перевірити джерело та

вміст повідомлення в Інтернеті. Можливо це розповсюджена атака і вже хтось про неї написав.

2. Будьте уважні до нетипової поведінки друзів та колег [33].

Зловмисники використовують викрадені облікові записи, щоб обдурити користувачів, тому відносьтесь з підозрою, якщо колега, який спілкувався з вами тільки по роботі, неочікувано надіслав вам посилання на якийсь сторонній сайт.

3. Не переходьте за невідомими посиланнями [33].

Зловмисники люблять викрадати облікові дані за допомогою підроблених сайтів, тому якщо вам пропонують перейти за цим сайтом, то ігноруйте його. Або хоча б перевірте антивірусом.

4. Не завантажуйте і тим паче не запускайте невідомі файли та програми [33].

Більшість серйозних атак починаються з проникнення в захищену мережу, для цього ідеально підходять зловмисні програми, які надсилаються з підозрілими повідомленнями. Тому, якщо від вас вимагають негайно завантажити та відкрити такий файл, то відразу ігноруйте цей запит та проінформуйте своїх колег та друзів про це. Якщо файл вам все ж потрібен, але ви невпевнені в його легітимності, ви завжди можете його перевірити за допомогою антивірусної програми.

5. Обмежте особисту інформацію в Інтернеті [33].

Уникайте розголошення будь-яких особистих даних, наприклад інформація про школу, в якій ви навчались, імена ваших домашніх улюбленців, де любите відпочивати тощо. Таку інформацію люблять використовувати користувачі в своїх паролях, або таємних кодах для отримання доступу до банківського рахунку. Також ця інформація може допомогти зловмисникам скласти ваш психологічний портрет і краще підготуватися до розмови з вами. Ви вірогідніше довіряться людині, яка любить ту саму породу котів, ніж якомусь невідомому адміністратору з банку.

6. Завжди ставтеся з підозрою, якщо від вас вимагають якісь дані [33].

Будь-які запити на дані слід приймати з обережністю. Задавайте питання та перевіряйте особу відправника, перш ніж розкрити хоча б якусь персональну інформацію.

2.3 Програмні методи виявлення соціальної інженерії

Соціальна інженерія орієнтована на роботу з людьми, тому наше рішення стосовно легітимності повідомлення суто суб'єктивне, через це не існує програмного забезпечення, яке б стовідсотково виявляло будь-яку атаку соціальної інженерії. Але все ж таки існують деякі рішення, які можуть нам допомогти краще зрозуміти, легітимне повідомлення, чи нас хочуть надурити.

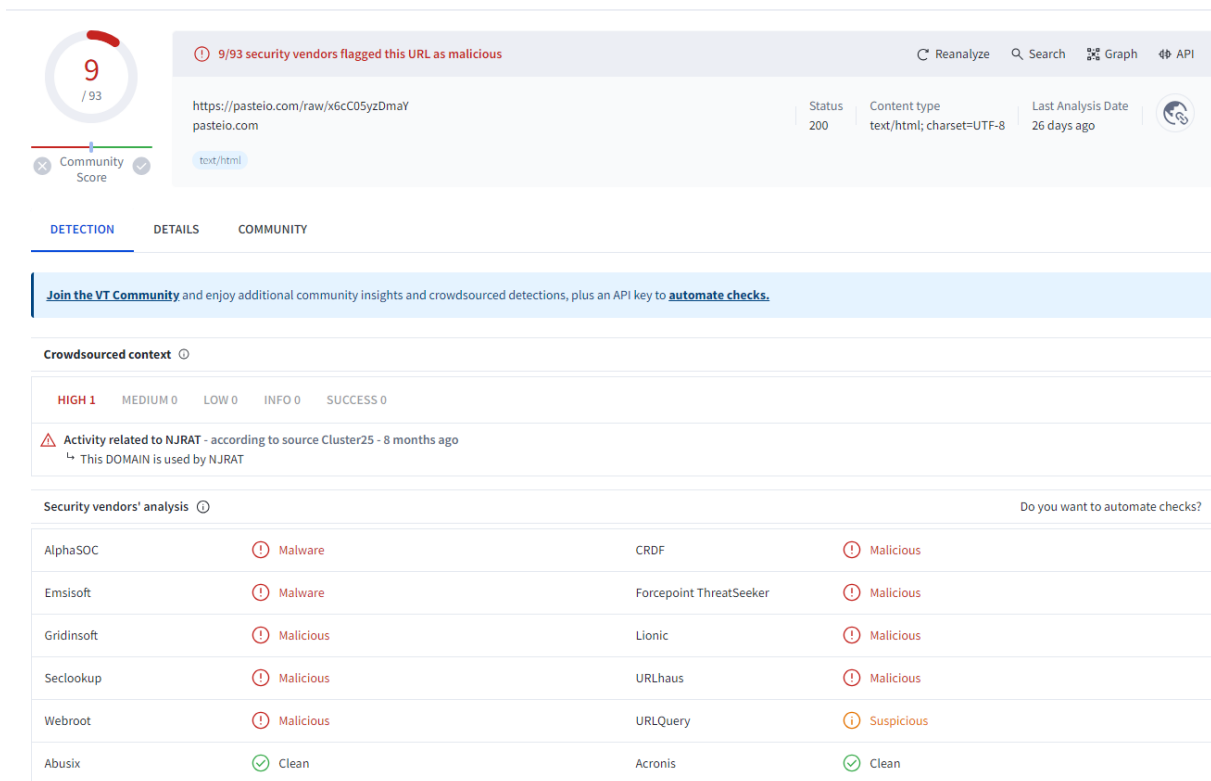
2.3.1 VirusTotal

Як зазначалось раніше, зловмисники люблять вкладати зловмисні програми та посилання у свої повідомлення, сподіваючись, що жертва їх відкриє. Тому непоганим методом виявлення соціальної інженерії буде перевірка вкладень та посилань за допомогою антивірусних програм. Якщо вони щось знайдуть, то це означатиме, що вас намагаються обдурити.

Існує велика кількість антивірусних засобів, серед них Virus Total. VirusTotal перевіряє елементи за допомогою понад 70 антивірусних сканерів і служб створення чорних списків URL/доменів, а також безліч інструментів для вилучення сигналів із досліджуваного вмісту. Будь-який користувач може вибрати файл зі свого комп'ютера за допомогою браузера та надіслати його до VirusTotal (див. рисунок 2.1).

VirusTotal пропонує низку методів надсилання файлів, включаючи основний загальнодоступний веб-інтерфейс, засоби завантаження з робочого столу, розширення браузера та програмний API. Веб-інтерфейс має найвищий пріоритет сканування серед загальнодоступних методів подання. Надсилання може бути написано будь-якою мовою програмування за допомогою публічного API на

основі HTTP. Як і у випадку з файлами, URL-адреси можна надсилати кількома різними засобами, включаючи веб-сторінку VirusTotal, розширення браузера та API [35].



The screenshot displays the VirusTotal analysis page for the URL `https://pasteio.com/raw/x6cC05yzDmaY`. The interface shows a community score of 9/93, a status of 200, and a content type of `text/html; charset=UTF-8`. The analysis was performed 26 days ago. The page is divided into sections: DETECTION, DETAILS, and COMMUNITY. A banner encourages joining the VT Community. The 'Crowdsourced context' section shows activity related to NJRAT. The 'Security vendors' analysis' section provides a table of detections from various vendors.

Security vendors' analysis			
AlphaSOC	Malware	CRDF	Malicious
Emsisoft	Malware	Forcepoint ThreatSeeker	Malicious
Gridinsoft	Malicious	Lionic	Malicious
Seclookup	Malicious	URLhaus	Malicious
Webroot	Malicious	URLQuery	Suspicious
Abusix	Clean	Acronis	Clean

Рисунок 2.1 – інтерфейс VirusTotal

Отже, якщо вам надіслали підозріле повідомлення, в якому є якийсь посилання, то краще його перевірити за допомогою VirusTotal. Якщо він щось знайде то це вже буде сигналом, що намагаються ввести в оману.

2.3.2 AI Detector

Враховуючи той факт, що зловмисники використовують ШІ, ми можемо використати деякі сервіси які покажуть вірогідність використання штучного інтелекту при написанні повідомлення.

Тому можемо скористатися спеціалізованими сайтами (наприклад: <https://www.scribbr.com/ai-detector/>). Результат його роботи можна побачити на рисунку 2.2.

AI Detector від Scribbr має показник точності – 84%. Він в більшості випадків коректно виявляє текст написаний людиною та текст написаний штучним інтелектом. Крім того він зміг виявити 60% текстів, де було поєднання тексту написаного людиною та штучним інтелектом [36]. Цей інструмент орієнтований на ChatGPT, Gemini та Copilot. GPT2, GPT3 і GPT3.5 виявляються з високою точністю, тоді як виявлення GPT4 підтримується на експериментальній основі [37].

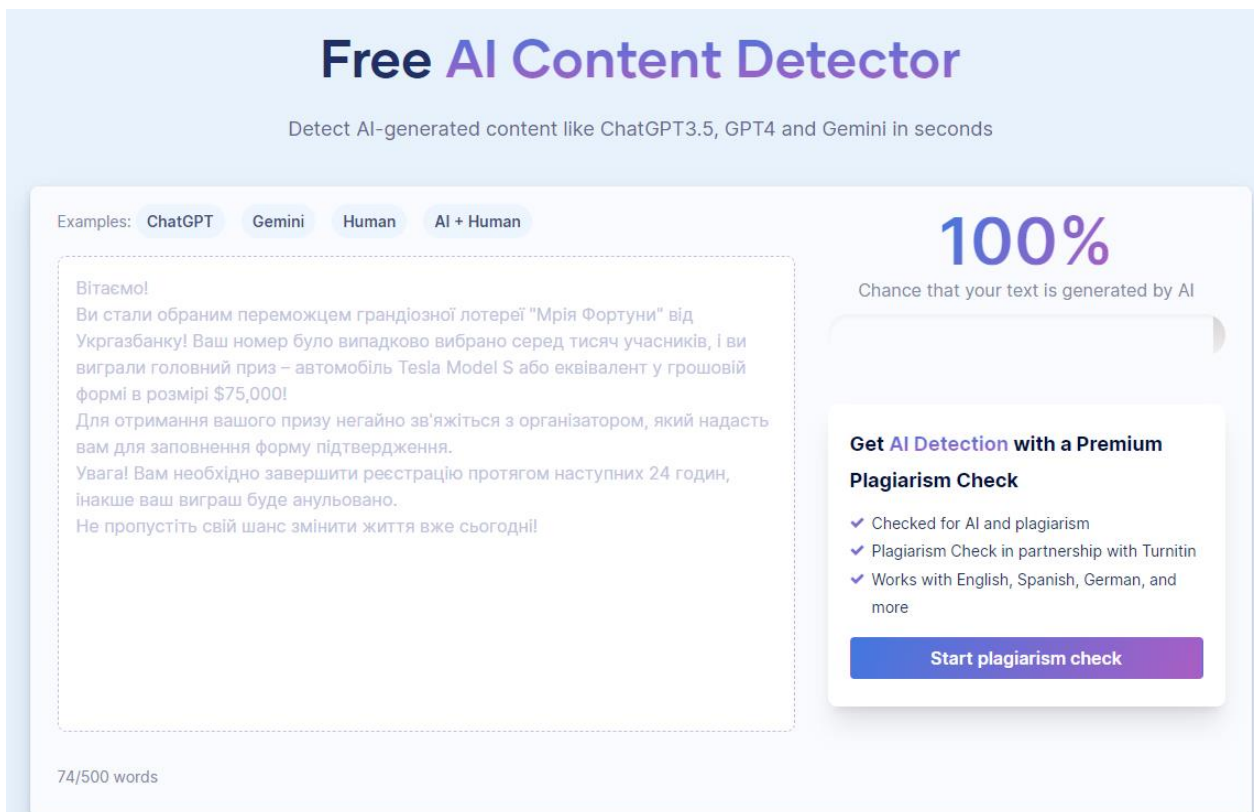


Рисунок 2.2 – Перевірка вірогідності використання ШІ в повідомленні за допомогою сервісу Scribbr

Звичайно, що цей інструмент не скаже 100%, що текст – легітимний, чи це соціальна інженерія, але якщо він щось виявив, то слід більш ретельно перевірити повідомлення та не робити поспішних дій.

2.4 Використання штучного інтелекту

У минулому розділі зазначалось, що зловмисники почали часто використовувати штучні інтелекти для проведення атак соціальної інженерії. Але їх також можна використати і в зворотну сторону, для аналізу підозрілих повідомлень (див. рисунок 2.3-2.4).

З березня 2024 року в Україні є можливість використання ChatGPT та відносно нещодавно став доступний штучний інтелект від Google – Gemini (раніше мав назву Bard).

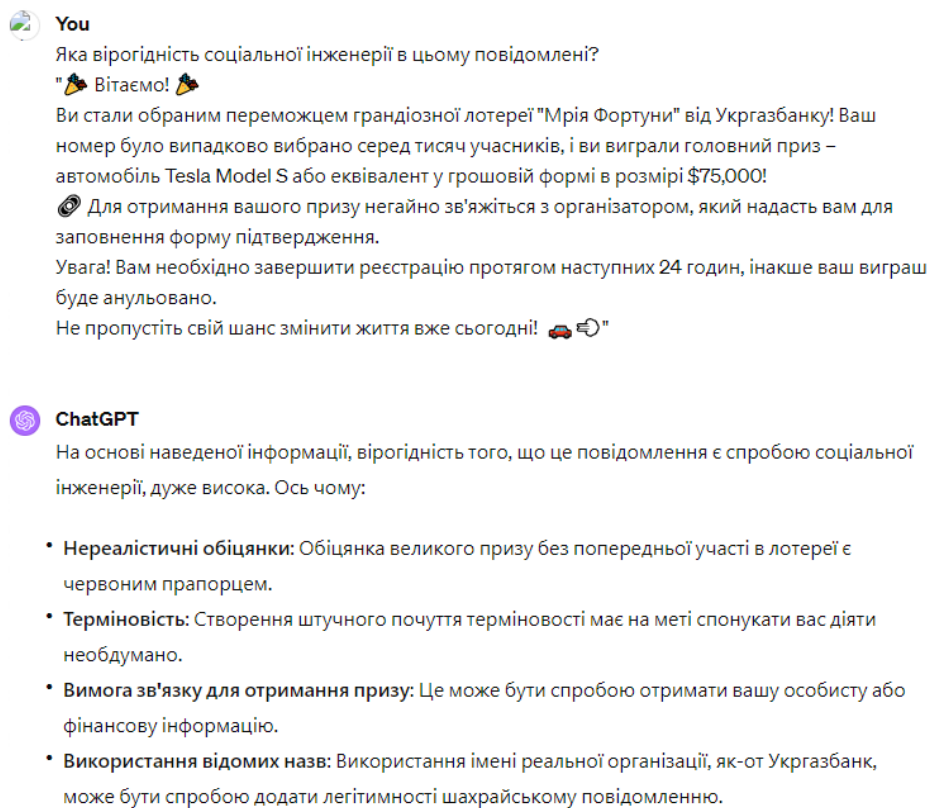


Рисунок 2.3 – Перевірка вірогідності використання соціальної інженерії в повідомленні за допомогою ChatGPT



Яка вірогідність соціальної інженерії в цьому повідомленні? ^

" 🎉 Вітаємо! 🎉

Ви стали обраним переможцем грандіозної лотереї "Мрія Фортуни" від Укргазбанку! Ваш номер було випадково вибрано серед тисяч учасників, і ви виграли головний приз – автомобіль Tesla Model S або еквівалент у грошовій формі в розмірі \$75,000!

👁️ Для отримання вашого призу негайно зв'яжіться з організатором, який надасть вам для заповнення форму підтвердження.

Увага! Вам необхідно завершити реєстрацію протягом наступних 24 годин, інакше ваш виграш буде анульовано.

Не пропустіть свій шанс змінити життя вже сьогодні! 🚗👉"

Показати черновики ✓ 🔊



Вірогідність соціальної інженерії в цьому повідомленні висока.

Ось декілька ознак, які на це вказують:

1. Незнайомий відправник:

Повідомлення не містить інформації про те, хто його надіслав, окрім згадки про "Укргазбанк".

2. Нечітка мова:

Використовуються емоційні слова та фрази, такі як " Вітаємо! ", "грандіозна лотерея", "Мрія Фортуни", "ви щасливчик", "не пропустіть свій шанс".

3. Терміновість:

В повідомленні створюється штучний ажіотаж, адже виграш потрібно "отримати негайно" і "завершити реєстрацію протягом наступних 24 годин".

Рисунок 2.4 – Перевірка вірогідності використання соціальної інженерії в повідомленні за допомогою Gemini

Як можемо побачити, чат-боти не тільки вказали вірогідність соціальної інженерії, а ще й вказали на тригери, які вказують на шкідливість повідомлення.

2.5 Аналіз тональність тексту

Існує таке поняття, як аналіз тональність тексту. Аналіз тональності тексту – це процес виявлення позитивного чи негативного емоційного забарвлення у тексті [38].

Зазвичай, аналіз тональності фокусується на полярності тексту (позитивний, негативний або нейтральний), іноді виходить за ці рамки, щоб виявити конкретні почуття та емоції (злий, щасливий, сумний тощо), терміновість (терміново, не терміново) та наміри (зацікавлений, не зацікавлений).

Для більшої точності існують категорії полярності тексту [38]:

- Дуже позитивний
- Позитивний
- Нейтральний
- Негативний
- Дуже негативний

Аналіз тональності тексту базується на NLP та алгоритмах машинного навчання, щоб автоматично визначити тон, який стоїть в онлайн-розмові.

Існують різні алгоритми для аналізу, але загалом їх можна розділити на три категорії [28]:

Засновані на правилах – алгоритми автоматично проводять аналіз емоційного забарвлення на основі наборів правил створених людиною.

Ці правила можуть включати різні техніки обробки природної мови, розроблені в області комп'ютерної лінгвістики, такі як:

- Стемінг, токенизація, розмітка частин мови та парсинг.
- Лексикони (тобто списки слів та виразів).

Базовий приклад роботи такого алгоритму:

1. Визначає два списки поляризованих слів (наприклад, негативні слова, такі як "поганий", "найгірший", "потворний" тощо, та позитивні слова, такі як "добрий", "найкращий", "красивий" тощо).

2. Підраховує кількість позитивних та негативних слів, які з'являються у даному тексті.

3. Якщо кількість випадків позитивних слів перевищує кількість негативних слів, система повертає позитивне емоційне забарвлення, і навпаки. Якщо числа рівні, система поверне нейтральне емоційне забарвлення.

Нажаль такі алгоритми не завжди дають бажаний результат, оскільки вони не враховують комбінацію слів у послідовності.

Автоматичні – ці системи використовують техніки машинного навчання. Задача аналізу тональності тексту зазвичай моделюється як проблема класифікації, де класифікатор отримує текст і повертає категорію, наприклад позитивну, негативну або нейтральну (див. рисунок 2.5).

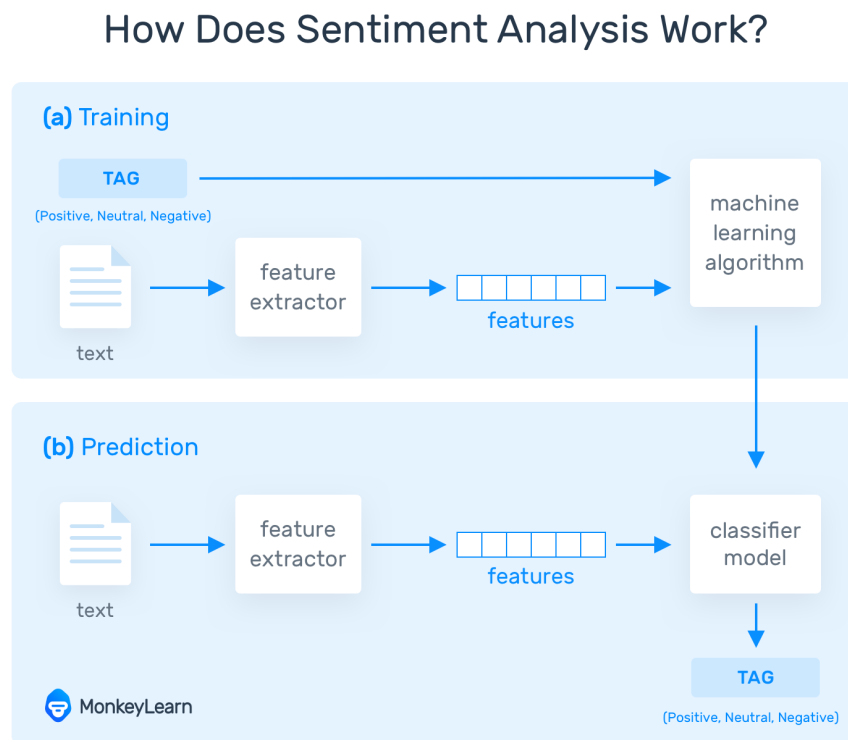


Рисунок 2.5 – Реалізація класифікатора на основі машинного навчання [38]

Гібридні системи: поєднують перший та другий підходи.

Головна перевага гібридних систем – більш точні результати.

Оскільки зловмисники при використанні соціальної інженерії намагаються грати на емоціях, то аналіз тональності їх повідомлень може допомогти виявити їх наміри.

2.6 Пошук слів тригерів

Як би зловмисники не намагались приховати свої наміри та максимально підлаштуватися під свою ціль, все одно будуть певні слова тригери які вкажуть, що повідомлення підозріле і до нього слід ставитися обачніше.

На жаль, у відкритих джерелах не виявилось списку слів на українській мові, які можуть використовувати хакери для соціальної інженерії, проте можемо використати аналоги з англійської мови.

Слова, які майже завжди використовуються в соціальній інженерії [39]: *доступ, швидко, надати, поспішати, необхідно, терміново, отримати.*

Якщо соціальна інженерія використовується для того, щоб отримати доступ до системи, то часто зустрічаються наступні слова [39]: *адміністратор, конфіденційний, дані, бази даних, пристрій, інформація, обліковий запис, логін, пароль, фаєрвол, IP, система.*

Тому якщо надходить повідомлення від невідомого контакту і там зустрічаються ці слова, то це привід більш детально перевірити його.

Висновки до розділу 2

У даному розділі було розглянуто методи, які допомагають виявити соціальну інженерію. Так як соціальна інженерія спрямована на комунікацію з людиною, то її виявлення – це суб'єктивна оцінка. Якщо узагальнити рекомендації, які надають для виявлення соціальної інженерії, то користувачу треба ставитись з підозрою до любого повідомлення від невідомого відправника, який щось від нього вимагає.

Також, було розглянуто програмні методи, які 100% не скажуть, соціальна інженерія це, чи ні, але вони можуть виявити певні аномалії та сповістити користувача про них. Тоді користувач буде детальніше перевіряти повідомлення на наявність соціальної інженерії.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ВИЯВЛЕННЯ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ В ПОВІДОМЛЕННЯХ В МЕСЕНДЖЕРІ «TELEGRAM»

Як зазначалось раніше, виявлення соціальної інженерії – це суб'єктивна задача, але є можливість застосувати певні інструменти, які допоможуть виявити вірогідність того, що повідомлення не зовсім легітимне.

Для перевірки повідомлень було розроблено програмний застосунок, який в автономному режимі отримує та перевіряє нові повідомлення в особистих чатах в месенджері «Telegram».

3.1 Середовище розробки

В якості середовища розробки було обрано мову програмування Python версії 3.9.

Python — це інтерпретована, об'єктноорієнтована, високорівнева мова програмування з динамічною семантикою. Її високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв'язуванням роблять її дуже привабливою для швидкої розробки програм, а також для використання в якості сценарної мови для з'єднання чинних компонентів. Простий синтаксис Python, легкий у навчанні та досить читабельний, а отже, знижує вартість обслуговування програм. Python підтримує модулі та пакети, що сприяє модульності програм і повторному використанню коду. Інтерпретатор Python та обширна стандартна бібліотека доступні у вигляді вихідного коду або бінарної форми безплатно для всіх основних платформ і можуть бути вільно розповсюджені [40].

3.2 Функції та скрипти програмного застосунку

Для кращої реалізації, програма базується на скриптах та функціях, які виконують певні дії і їх можна легко відредагувати в разі необхідності.

3.2.1 Скриптовані файли

Програмний застосунок містить декілька скриптованих файлів з розширенням «.py», розглянемо їх функціонал:

Get_config_info.py – скрипт який створює клас `UserConfig` в якому зберігається інформація, необхідна для авторизації та проведення аналізу повідомлень.

Authorization.py – скрипт, який проводить авторизацію в месенджері «Telegram». Достатньо авторизуватись один раз і ваша сесія збережеться на пристрої.

Get_telegram_messages.py – скрипт, який в режимі реального часу моніторить повідомлення. Коли надходить нове повідомлення, то він його обробляє та надсилає іншому скрипту для аналізу.

Check message.py – скрипт, який здійснює аналіз повідомлення на наявність соціальної інженерії. Після аналізу виводить та зберігає результат.

3.2.2 Функції та класи

UserConfig – клас, який створює об'єкти, що зберігають інформацію, необхідну для роботи застосунку. Інформація, яку зберігає: номер телефону, ідентифікатор та хеш API Telegram, ключі для API VirusTotal та Openai (див. рисунок 3.1).

```
class UserConfig:
    def __init__(self, api_id=None, api_hash=None, phone=None, api_vt=None, api_openai=None):
        self.api_id = api_id
        self.api_hash = api_hash
        self.phone = phone
        self.api_vt = api_vt
        self.api_openai = api_openai
```

Рисунок 3.1 – клас UserConfig

Get_config (функція зчитування та збереження даних авторизації) – функція, яка створює об'єкт класу *UserConfig* та записує в його змінні інформацію з файлу «accounts_settings» (див. рисунок 3.2).

```
def get_config():
    config_file = "./accounts_settings.txt"
    config = UserConfig()
    with open(config_file, 'r') as file:
        for line in file:
            key, value = line.strip().split('=', 1)
            key = key.strip()
            value = value.strip()
            if key == 'api_id':
                config.api_id = int(value)
            elif key == 'api_hash':
                config.api_hash = value
            elif key == 'phone':
                config.phone = value
            elif key == 'api_key_vt':
                config.api_vt = value
            elif key == 'openai_api_key':
                config.api_openai = value
    return config
```

Рисунок 3.2 – функція get_config

Async def main (функція моніторингу повідомлень) – асинхронна функція, яка моніторить повідомлення. Коли надходить нове повідомлення, то надсилає його на обробку функції *async def handle_new_message()* (див. рисунок 3.3).

```

async def main():
    print("Моніторинг нових повідомлень... Натисніть Ctrl+C для остановки.")
    await client.start()
    await client.run_until_disconnected()

# Запускаємо моніторинг
with client:
    client.loop.run_until_complete(main())

```

Рисунок 3.3 – функція *async def main*

Async def handle_new_message (функція обробки нових повідомлень) – функція, яка працює за допомогою бібліотеки *telethon* та обробляє всі нові повідомлення в особистих чатах, дістає з них тіло повідомлення та інформацію про відправника, після чого запускає функції перевірки, надаючи їм текст повідомлення (див. рисунок 3.4).

```

@client.on(events.NewMessage(incoming=True))
async def handle_new_message(event):
    if event.is_private: # Перевіряємо, що це особисте повідомлення
        sender = await event.get_sender()
        sender_name = sender.first_name if sender.first_name else "Без имени"
        if sender.last_name: # Додаємо прізвище, якщо воно є
            sender_name += f" {sender.last_name}"

    print(f'Нове повідомлення від {sender_name}: {event.text}')
    call([r".\env\Scripts\python.exe", "Check message.py", event.text, sender_name])

```

Рисунок 3.4 – функція *async def handle_new_message*

Extract_urls (функція пошуку посилань в повідомленні) – функція, яка за допомогою регулярних виразів шукає посилання в тексті (див. рисунок 3.5).

```
def extract_urls(text):
    # Регулярний вираз для пошуку URL
    url_regex = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[*\\(\)\,]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    urls = re.findall(url_regex, text)
    cleaned_urls = [re.sub(r'^\w\s-$', '', url) for url in urls]
    return cleaned_urls
```

Рисунок 3.5 – функція *extract_urls*

Check_urls_virus_total (функція перевірки посилань) – функція, яка перевіряє чи немає шкідливих посилань серед тих, що були знайдені в повідомленні. Використовує бібліотеку *vt*, яка працює на основі сервісу Virus Total (див. рисунок 3.6).

```
def check_urls_virus_total(api_key, urls):
    global make_notification
    message = ""
    for url in urls:
        # Створюємо клієнт для Virus Total
        client = vt.Client(api_key)
        client.scan_url(url, wait_for_completion=True)
        url_id = vt.url_id(url)
        # Надсилаємо посилання на перевірку
        site = client.get_object("/urls/{}", url_id)
        client.close()
        # Вивантажуємо результат сканування
        malicious_detections = site.last_analysis_stats['malicious']
        # Якщо хоча б один антивірус зреагував, то помічаємо, як підозріле. Готуємо відповідне повідомлення.
        if malicious_detections > 0:
            message = message + "Увага: посилання " + url + " - зловмисне. Виявлено " + str(malicious_detections) + " спрацювань.\n"
            make_notification['VirusTotal'] = 1
    if message != "":
        return message
    else:
        return "Підозрілих посилань не виявлено"
```

Рисунок 3.6 – функція *check_urls_virus_total*

Detect_social_engineering_GPT (функція оцінки вірогідності використання соціальної інженерії в повідомленні за допомогою GPT) – функція, яка за

допомогою бібліотеки *openai* використовує III GPT-4 для перевірки наявності соціальної інженерії в повідомленні (див. рисунок 3.7).

```
def detect_social_engineering_GPT(text):
    global make_notification
    # Ініціалізуємо клієнт openai
    client = OpenAI(api_key = openai_api_key)
    # Надсилаємо запит
    response = client.chat.completions.create(
        model="gpt-4",
        messages=[
            {"role": "user",
             "content": "Яка вірогідність використання соціальної інженерії в цьому тексті: " + text +
            }
        ]
    )

    # Обробка відповіді на наш запит
    probability = response.choices[0].message.content
    percents = re.findall(r'\d+%', probability)

    # Реагуємо, якщо відсоток вірогідності CI > 60
    if int(percents[0][:-1]) >= 60:
        make_notification['GPT'] = 1
    return probability, percents
```

Рисунок 3.7 – функція `detect_social_engineering_GPT`

Detect_social_engineering_Gemini (функція оцінки вірогідності використання соціальної інженерії в повідомленні за допомогою Gemini) – функція, яка за допомогою бібліотеки *vertexai* використовує III Gemini для перевірки наявності соціальної інженерії в повідомленні (див. рисунок 3.8).

```
def detect_social_engineering_Gemini(analyse_text, project_id = "gemini-social", location = "europe-central2"):
    global make_notification
    # Ініціалізуємо Vertex AI
    vertexai.init(project=project_id, location=location)
    # Завантажуємо модель
    multimodal_model = GenerativeModel("gemini-1.0-pro-vision")
    # Надсилаємо запит
    response = multimodal_model.generate_content(
        [
            "Яка вірогідність використання соціальної інженерії в цьому тексті: " + analyse_text + "\nНапиши таким чином: 'Вірогідність:"
        ]
    )

    # Реагуємо, якщо відсоток вірогідності CI > 60
    percents = re.findall(r'\d+%', response.text)
    if int(percents[0][:-1]) >= 60:
        make_notification['Gemini'] = 1
    return response.text, percents
```

Рисунок 3.8 – функція `detect_social_engineering_Gemini`

Detect_using_AI (функція оцінки вірогідності використання штучного інтелекту в повідомленні) – функція, яка за допомогою бібліотеки *selenium* передає дані сайту www.scribbr.com для перевірки наявності в повідомленні тексту, створеного за допомогою ШІ (див. рисунок 3.9).

```
def detect_using_AI(text):
    global make_notification
    # Запускаємо драйвер браузера
    driver = webdriver.Chrome()

    # відкриваємо веб-сторінку
    driver.get('https://www.scribbr.com/ai-detector/')

    # знаходимо поля для введення тексту
    input_field = driver.find_element(By.CSS_SELECTOR, 'div[contenteditable="true"]')
    input_field.send_keys(text)

    # Знаходимо кнопку та нажимаємо її
    submit_button = driver.find_element(By.ID, "aiDetectorButton")
    submit_button.click()
    sleep(10)

    # Шукаємо результат в контейнері з певним ідентифікатором
    result_element = driver.find_element(By.CSS_SELECTOR, "span.percentage.purple.d-block.text-center.text-purple-gradient")
    result_text = result_element.text

    # Реагуємо, якщо відсоток використання ШІ > 60
    result_text = int(result_text.replace('%', ''))
    if int(result_text) >= 60:
        make_notification['Site'] = 1
    return f"Вірогідність використання ШІ (www.scribbr.com): {result_text}"
```

Рисунок 3.9 – функція *detect_using_AI*

Detect_social_engineering_Sentiment (функція оцінки тональності тексту) – функція, яка за допомогою бібліотеки *transformers* оцінює тональність тексту в повідомленні (див. рисунок 3.10).

```
def detect_social_engineering_Sentiment(message):
    global make_notification
    # Створюємо об'єкт для перекладача, в якому зазначаємо на чку мову необхідно перекласти
    translator = GoogleTranslator(source='auto', target='en')
    #Перекладаємо текст на англійську мову (модель для аналізу тональності навчена лише на англійських словах.)
    translated_text = translator.translate(message)

    # Проводимо аналіз тональності тексту
    sentiment = sentiment_pipeline(translated_text)[0]

    # Якщо оцінка тональності > 75 та має негативний стан, то помічаємо як аномалію
    if sentiment['label'] == 'NEGATIVE' and sentiment['score'] > 0.75:
        make_notification['Sentiment'] = 1
        result = "Це повідомлення може містити елементи соціальної інженерії"
    else:
        result = "Це повідомлення виглядає безпечним."
    return result, sentiment
```

Рисунок 3.10 – функція *detect_social_engineering_Sentiment*

Notify (функція виведення оповіщення) – функція, яка після аналізу тексту виводить оповіщення, якщо було знайдено щось підозріле. Працює за допомогою бібліотеки *plyer* (див. рисунок 3.11).

```
def notify(result, user):
    # Виведемо оповіщення про знайдені аномалії в повідомленні
    notification.notify(title="Виявлено підозріле повідомлення від " + user, message=result)
```

Рисунок 3.11 – функція notify

Log (функція логування результатів) – функція, яка зберігає інформацію про аналіз повідомлення в окремому текстовому файлі, який можна переглянути в разі необхідності (див. рисунок 3.12).

```
def log(vt, gpt, gemini, ai, sentiment, sentiment_score, username):
    # Збережемо результат аналізу
    log_info = "VirusTotal\n" + vt + "\n" + 5*"-----" + "\n" + "GPT analyse\n" + gpt + "\n" + 5*"-----" + "\n" + "Gemini analyse\n" + gemini + "\n" + 5*"-----"
    date = datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
    # Створимо файл логів. В назві вкажемо відправника повідомлення та час створення логів
    filename = f"{username}-{date}.txt"
    # Вкажемо, що логи повинні зберігатись в директорії "logs"
    log_directory = "logs"
    if not os.path.exists(log_directory):
        os.makedirs(log_directory)
    file_path = os.path.join(log_directory, filename)
    # Запишемо результат аналізу в файл
    with open(file_path, 'w', encoding='utf-8') as file:
        file.write(log_info)
    print("Аналіз задокументовано")
```

Рисунок 3.12 – функція log

Скрипт, який перевіряє повідомлення, працює наступним чином. Спочатку отримує необхідні ключі API для Virus Total, GPT-4 та створює глобальний словник *make_notification*, в який буде записана інформація, що саме виявило аномалії в повідомленні (див. рисунок 3.13).

```
# Параметри, необхідні для аналізу
api_key_vt = get_config().api_vt
openai_api_key = get_config().api_openai
sentiment_pipeline = pipeline("sentiment-analysis")
# Глобальна змінна (словник), в якій зберігається інформація, чи виявлено аномалію
make_notification = {'VirusTotal':0, 'GPT':0, 'Gemini':0, 'Site':0, 'Sentiment':0}
```

Рисунок 3.13 – Глобальний словник та необхідні параметри

Після задання глобальних параметрів, скрипт зчитує текст та інформацію про відправника з аргументів, які йому передавались. Потім, отриманий текст аналізується за допомогою функцій. В залежності від того, що та чим було знайдено, готується відповідне сповіщення користувача, яке буде виведено за допомогою функції *notify* (див. рисунок 3.14).

```

if __name__ == '__main__':
    # Перевіримо, чи отримали перший аргумент
    if len(sys.argv) > 1:
        # Зчитуємо текст з першого аргументу
        text = sys.argv[1]
        username = sys.argv[2]
        # Якщо аргументів не передали, то виводимо сповіщення і завершуємо процес
    else:
        print("Текст не був надісланий.")
        sys.exit(1)

    if text:
        # Збережемо результати аналізу в змінні
        SE_GPT, SE_GPT_score = detect_social_engineering_GPT(text)
        SE_Gemini, SE_Gemini_score = detect_social_engineering_Gemini(text)
        Using_AI_score = detect_using_AI(text)
        SE_sentiment, SE_Sentiment_score = detect_social_engineering_Sentiment(text)

        # Якщо повідомлення містить посилання, то передати їх функції для перевірки за допомогою VirusTotal
        urls = extract_urls(text)
        if urls:
            virustotal_results = check_urls_virus_total(api_key_vt, extract_urls(text))

        # Якщо хоча б один ключ в словнику "make_notification" має значення 1, то робиться оповіщення.
        if 1 in make_notification.values():
            # Якщо Virus Total знайшов шкідливе посилання, то робиться оповіщення, де зазначається, яке посилання небезпечне та скільки антивірусів на нього реагувало
            if make_notification.get('VirusTotal') == 1:
                notificate(virustotal_results, username)
                print(virustotal_results)
                sleep(7)
            # Створиться змінна notif в яку буде збережено всі повідомлення про аномалії
            notif = ""
            if make_notification.get('GPT') == 1:
                notif = notif + "Вірогідність соціальної інженерії (GPT): " + SE_GPT_score[0] + "\n"
            if make_notification.get('Gemini') == 1:
                notif = notif + "Вірогідність соціальної інженерії (Gemini): " + SE_Gemini_score[0] + "\n"
            if make_notification.get('Site') == 1:
                notif = notif + "Вірогідність використання ШІ: " + Using_AI_score + "\n"
            if make_notification.get('Sentiment') == 1:
                notif = notif + SE_sentiment
            if notif != "":
                notificate(notif, username)
                print(notif)

        # Збереження результатів сканування у логи
        if urls:
            log(virustotal_results, SE_GPT, SE_Gemini, Using_AI_score, SE_sentiment, SE_Sentiment_score, username)
        else:
            log("Посилань не виявлено", SE_GPT, SE_Gemini, Using_AI_score, SE_sentiment, SE_Sentiment_score, username)

    sys.exit(0)

```

Рисунок 3.14 – функція main скрипта Check message

Повний алгоритм роботи застосунку зображено на блок-схемі на рисунку 3.15

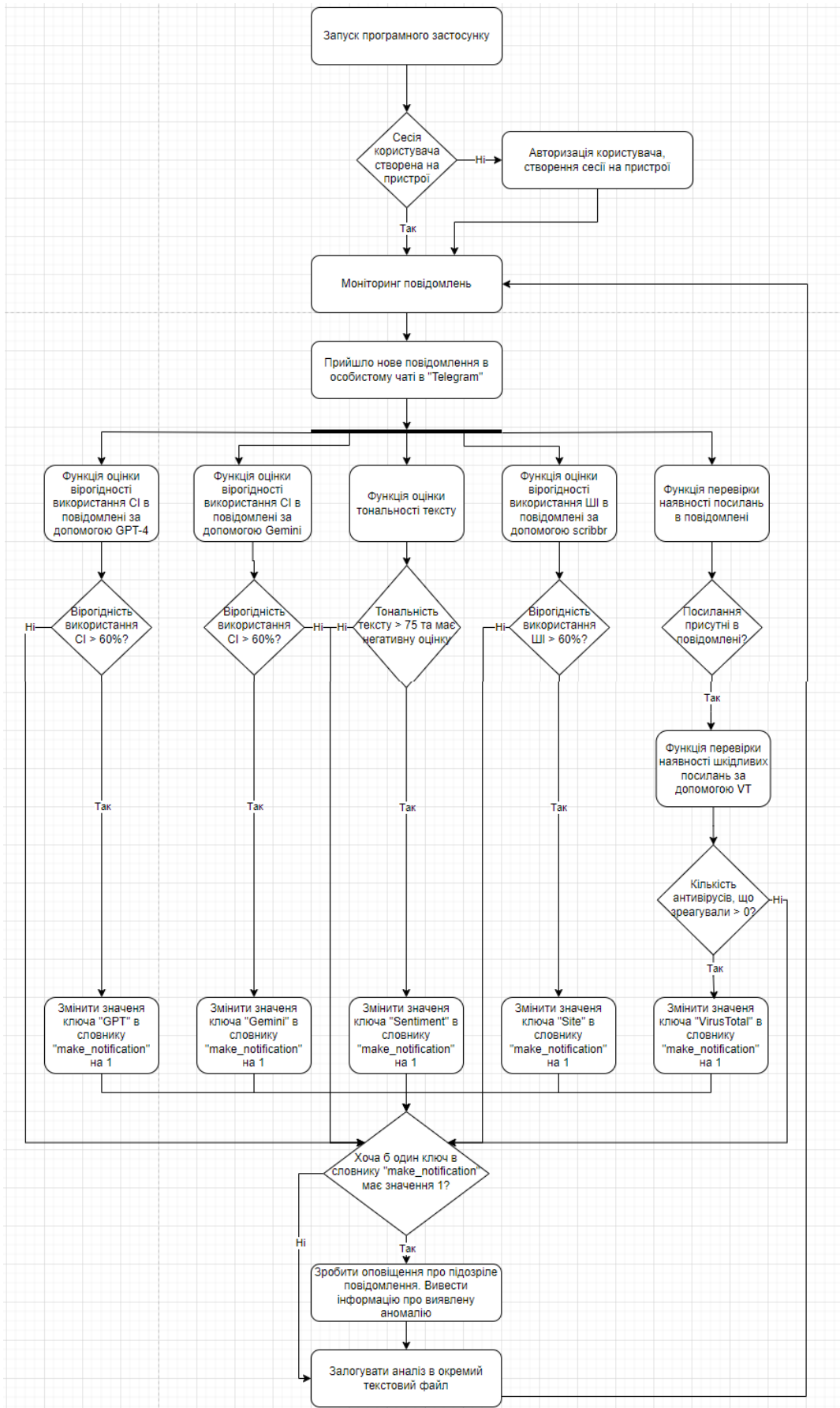


Рисунок 3.15 – Блок-схема програмного застосунку

3.3 Допоміжні модулі

Для того, щоб програмний застосунок міг ефективно виконувати свої задачі, було використано досить функціональні модулі. Розглянемо, що вони з себе мають та як працюють.

3.3.1 Telethon

Telethon — це асинхронна бібліотека Python, яка спрощує взаємодію з API Telegram. Вона забезпечує зручний інтерфейс користувача та обробляє всі складнощі, пов'язані з виконанням API-викликів та обробкою оновлень від Telegram. За допомогою Telethon ви можете легко відправляти та обробляти вхідні повідомлення, керувати каналами та виконувати багато інших завдань [41].

Для того, щоб взаємодіяти з повідомленнями необхідно знати лише API ID, API hash та номер телефону. API ID та API hash створюються за допомогою веб-сайту Telegram <https://my.telegram.org/apps>.

3.3.2 Openai

Бібліотека OpenAI Python забезпечує зручний доступ до API OpenAI з додатків, написаних на мові Python. Вона включає попередньо визначений набір класів для ресурсів API, які ініціалізуються динамічно з відповідей API, що робить її сумісною з широким спектром версій API OpenAI [42].

Для роботи з цією бібліотекою треба мати лише Openai API key, який можемо отримати на сайті <https://platform.openai.com/account/api-keys>.

3.3.3 Selenium

Selenium — це відкритий фреймворк, який переважно використовується для автоматизації взаємодії з веб-елементами в AUT (Application Under Test). Окрім Python, Selenium також підтримує Java, C#, JavaScript, Ruby та PHP [43].

Перед тим, як почати користуватися командами Selenium, вам потрібно імпортувати пакет Selenium WebDriver. Після завантаження відповідного драйвера браузера вам необхідно запустити Selenium WebDriver та драйвер браузера (наприклад, ChromeDriver для Chrome, GeckoDriver для Firefox тощо). Після цього ви можете проводити маніпуляції з вашим браузером [43].

3.3.4 Vertexai

VertexAI – API VertexAI Gemini забезпечує єдиний інтерфейс для взаємодії з моделями Gemini. Наразі у API Gemini доступні дві моделі [44]:

- Модель Gemini Pro (gemini-pro): Розроблена для виконання завдань обробки природної мови, багатоетапного тексту та чату з кодом, а також генерації коду.
- Модель Gemini Pro Vision (gemini-pro-vision): Підтримує багатомодальні запити. Ви можете включати текст, зображення та відео у свої запити та отримувати відповіді у вигляді тексту або коду.

3.3.5 Transformers

Бібліотека Transformers надає API та інструменти для легкого завантаження та тренування переднавчених сучасних моделей. Використання переднавчених моделей може зменшити витрати на обчислення, вуглецевий слід та заощадити час та ресурси, необхідні для навчання моделі з нуля. Ці моделі підтримують загальні завдання у різних модальностях, таких як [45]:

- Обробка природної мови: класифікація текстів, розпізнавання іменованих сутностей, відповіді на питання, моделювання мови, узагальнення, переклад, вибір із кількох варіантів та генерація тексту.
- Комп'ютерний зір: класифікація зображень, виявлення об'єктів та сегментація.
- Аудіо: автоматичне розпізнавання мови та класифікація аудіо.
- Мультимодальні завдання: відповіді на питання за таблицями, оптичне розпізнавання символів, витягування інформації зі сканованих документів, класифікація відео та візуальні відповіді на питання.

Transformers підтримує взаємосумісність фреймворків між PyTorch, TensorFlow та JAX. Це забезпечує гнучкість використання різних фреймворків на кожному етапі життя моделі. Моделі також можуть бути експортовані у формати, такі як ONNX та TorchScript, для розгортання у виробничих середовищах.

3.3.6 vt

Це офіційна клієнтська бібліотека Python для VirusTotal. За допомогою цієї бібліотеки ви можете взаємодіяти з REST API VirusTotal v3 та автоматизувати свій робочий процес швидко та ефективно [46].

Речі, які ви можете робити за допомогою vt [46]:

- Сканувати файли та URL-адреси.
- Отримувати інформацію про файли, URL-адреси, домени тощо.
- Виконувати пошуки за допомогою VirusTotal Intelligence.
- Керувати вашими наборами правил LiveHunt та сповіщеннями.
- Запускати ретрохантингові завдання та отримувати результати.
- Маніпулювати вашими графами VirusTotal.

3.3.7 deep_translator

Deep_translator – гнучкий безкоштовний та необмежений інструмент для перекладу між різними мовами за допомогою кількох перекладачів [47].

3.3.8 Plyer

Plyer — це мультиплатформений API Python для доступу до апаратних функцій різних платформ (Android, iOS, macOS, Linux та Windows) [48].

3.3.9 Базові модулі

Subprocess – модуль, який дозволяє створювати нові процеси, підключатися до їхніх каналів вводу/виводу/помилки і отримувати їхні коди повернення [49];

os – модуль, який забезпечує портативний спосіб використання залежних від операційної системи функцій [50];

sys – модуль, який надає доступ до деяких змінних, які використовуються або обслуговуються інтерпретатором, і до функцій, які сильно взаємодіють з інтерпретатором [51];

re – модуль, який забезпечує операції зіставлення регулярних виразів, подібні до тих, які є в Perl [52];

time – модуль, який забезпечує різні функції, пов'язані з часом. В нашому застосунку він необхідний для використання функції sleep(), яка призупиняє виконання поточного потоку на вказану кількість секунд [53];

datetime – надає класи для маніпуляцій з датами та часом [54].

3.4 Приклад використання застосунку

Якщо застосунок використовується перший раз, то необхідно пройти авторизацію (див. рисунок 3.16). Після авторизації буде створено сесію на пристрої користувача, яка дозволить відразу використовувати застосунок за його призначенням.

```
Сесія не знайдена, створюємо нову...  
Input your 2FA code, if you have it: 2FA code  
Please enter the code you received: 12345  
Signed in successfully as username ; remember to not break the ToS or you will risk an account ban!
```

Рисунок 3.16 – Авторизація

Після того, як пройшли авторизацію, можемо протестувати застосунок (див. рисунки 3.17-3.22). Створимо зловмисне повідомлення за допомогою штучного інтелекту, в якому ще буде шкідливе посилання.

Текст повідомлення:

«Вітаємо!

Ви стали обраним переможцем грандіозної лотереї "Мрія Фортуни" від Укргазбанку! Ваш номер було випадково вибрано серед тисяч учасників, і ви виграли головний приз – автомобіль Tesla Model S або еквівалент у грошовій формі в розмірі \$75,000!

Для отримання вашого призу негайно перейдіть за посиланням та заповніть форму підтвердження:

<https://pasteio.com/raw/x6cC05yzDmaY>

Увага! Вам необхідно завершити реєстрацію протягом наступних 24 годин, інакше ваш виграш буде анульовано.

Не пропустіть свій шанс змінити життя вже сьогодні! »

```

Моніторинг нових повідомлень... Натисніть Ctrl+C для останки.
Нове повідомлення від Тестовий користувач: Вітаємо!
Ви стали обраним переможцем грандіозної лотереї "Ірмія Фортуни" від Укргазбанку! Ваш номер було випадково вибрано серед тисяч учасників, і ви виграли головний приз - автомобіль Tesla Mo
Для отримання вашого призу негайно перейдіть за посиланням та заповніть форму підтвердження:
https://pasteio.com/raw/x6cC05yzDmaY

Увага! Вам необхідно завершити реєстрацію протягом наступних 24 годин, інакше ваш виграш буде анульовано.
Не пропустіть свій шанс змінити життя вже сьогодні!

```

Рисунок 3.17 – Інформація про повідомлення

```

Увага: посилання https://pasteio.com/raw/x6cC05yzDmaY - зловмисне. Виявлено 7 спрацювань.

Вірогідність соціальної інженерії (GPT): 100%
Вірогідність соціальної інженерії (Gemini): 85%
Це повідомлення може містити елементи соціальної інженерії
Аналіз залоговано

```

Рисунок 3.18 – Результат перевірки, виведений у консоль

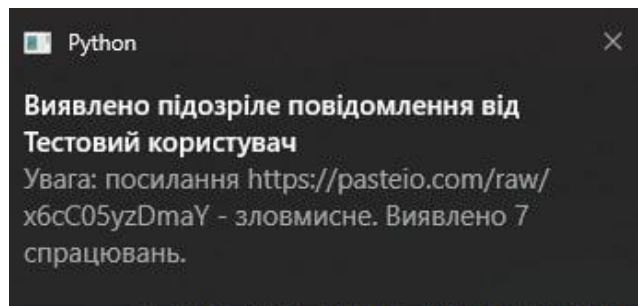


Рисунок 3.19 – Системне повідомлення про знайдене підозріле посилання

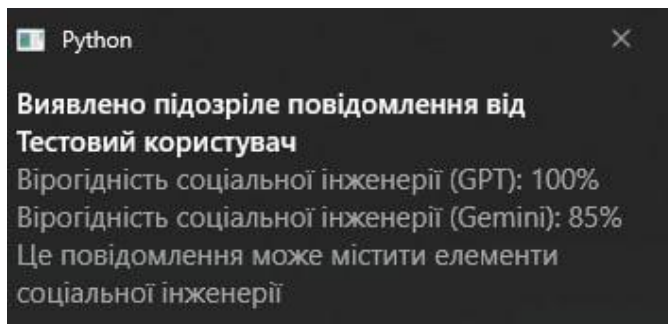


Рисунок 3.20 – Системне повідомлення про результати перевірки

Рисунок 3.21 – Файл логу повідомлення

```

Тестовий користувач-2024-04-23-23-44-52 - Блоковий
Файл Правка Формат Вид Справка
VirusTotal
Увага: посилання https://pasteio.com/raw/x6c05yDmaY - зловмисне. Виявлено 7 спрацювань.
-----
GPT analyse
Вірогідність: 100%
1. Обманлива пропозиція: Пропонується великий подарунок або виграш з лотереї, яку користувач не пам'ятає, що грав.
2. Терміновість: Текст викликає прями вказівки на те, що користувачу потрібно негайно діяти, інакше вони "втратять" свій виграш, створені поспіх і натиск.
3. Посилання на невідому вебсайт: Користувачу пропонується натиснути на посилання, що веде на сторонній вебсайт. Це часто використовується кіберзлочинцями для викрадення особистих даних або встановлення шкідливого програмного забезпечення.
4. Відсутність особистого звернення: Замість того, щоб використовувати ім'я або іншу особисту інформацію, повідомлення починається з загального звертання, що є загальною ознакою спаму або шахрайства.
5. Граматичні помилки та дивні формулювання: Хоча є чіткі й виправдані, цей текст все ще може містити дивні або неестественні фрази, які часто зустрічаються в спамі або шахрайстві.
-----
Gemini analyse
Вірогідність: 85%
Текст містить ознаки соціальної інженерії, такі як:
* **Почуття терміновості:** "Вам необхідно завершити реєстрацію протягом наступних 24 годин, інакше ваш виграш буде анульовано".
* **Використання особистої інформації:** Звернення "Штанно!" та вказівка, що номер користувача був випадково вибрано серед тисяч учасників".
* **Почуття недоступності:** "Головний приз - автомобіль Tesla Model S або еквівалент у грошовій формі в розмірі $75,000!" створює враження, що це унікальна можливість.
* **Запрошення перейти за підозрілим посиланням:** Посилання веде на сайт, який не належить Укрзаббанку.
* **Використання слів, що викликають позитивні емоції:** "Мрія Фортуні", "змінити життя".
-----
Using AI analyse
Вірогідність використання ШІ (www.scribbr.com): 46
-----
Sentiment analyse
Це повідомлення може містити елементи соціальної інженерії
{"label": "NEGATIVE", "score": 0.9896629452785383}

```

Рисунок 3.22 – Інформація з логів

Як можемо побачити, застосунок реагує на шкідливі посилання, за допомогою ШІ виявляє вірогідність використання соціальної інженерії в повідомленні. Також аналіз тональності тексту вказує, що повідомлення підозріле.

Тепер перевіримо застосунок на повідомленні, яке розсилалось користувачам в «Telegram» (див. рисунки 3.23-3.24).

Текст повідомлення:

«Вітаємо Вас в офіційному телеграм каналі фінансової допомоги.

В зв'язку з енергетичною кризою в країні, при підтримці міжнародних благодійних організацій, урядом України було прийнято рішення про надання єдинократової фінансової допомоги в розмірі 12000 гривень

Для отримання коштів перейдіть за посиланням:

ПРИВАТ БАНК

<https://ooqsnadb.online/171828913>»

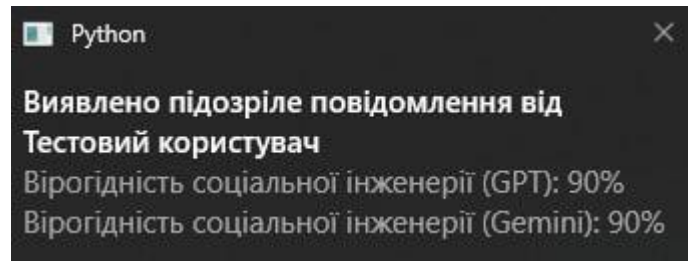


Рисунок 3.23 – Виявлені загрози в повідомленні

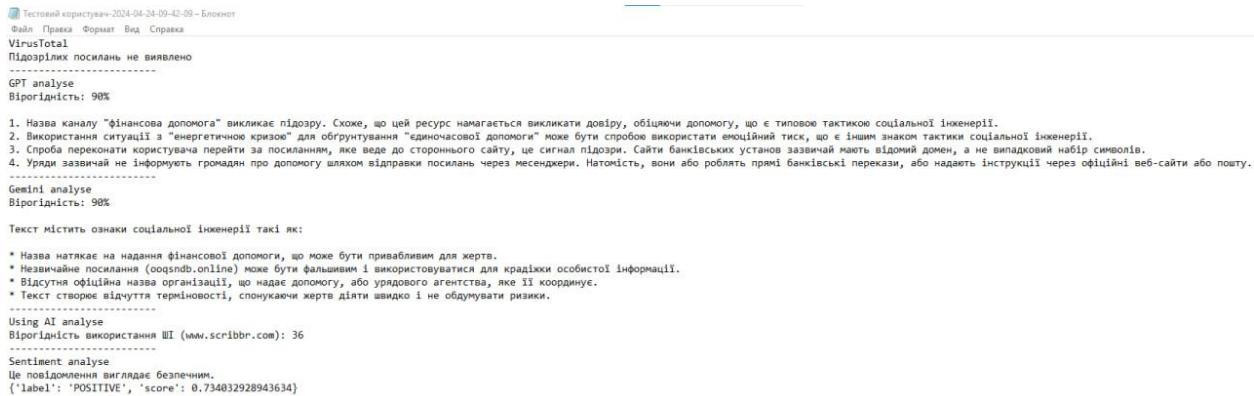


Рисунок 3.24 – Результат аналізу

Як можемо помітити, на це повідомлення зреагував лише ШІ. Virus Total нічого не побачив, так як сайт одноразовий, тому вони не вважаються зловмисними. Текст був написаний людиною, а аналіз тональності тексту показав, що це позитивне повідомлення, тому не зреагував.

Проте ШІ все ж виявили що це підозріле повідомлення і описали чому вони так вважають:

- Назва натякає на надання фінансової допомоги, що може бути привабливим для жертв.
- Незвичайне посилання (ooqsnodb.online) може бути фальшивим і використовуватися для крадіжки особистої інформації.
- Відсутня офіційна назва організації, що надає допомогу, або урядового агентства, яке її координує.

- Текст створює відчуття терміновості, спонукаючи жертв діяти швидко і не обдумувати ризики.

Перевіримо застосунок на звичайному рекламному повідомленні (див. рисунки 3.25-3.26).

Текст повідомлення:

«Привіт, збираємо однодумців, яким теж подобається рок. Приходь до клубу "BestRock" 05.05 о 15:00. Там будуть нові гурти, які покажуть свій талант.

Тобі обов'язково сподобається.»

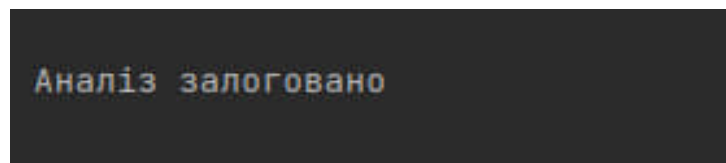


Рисунок 3.25 – Загроз не виявлено, результати залоговані

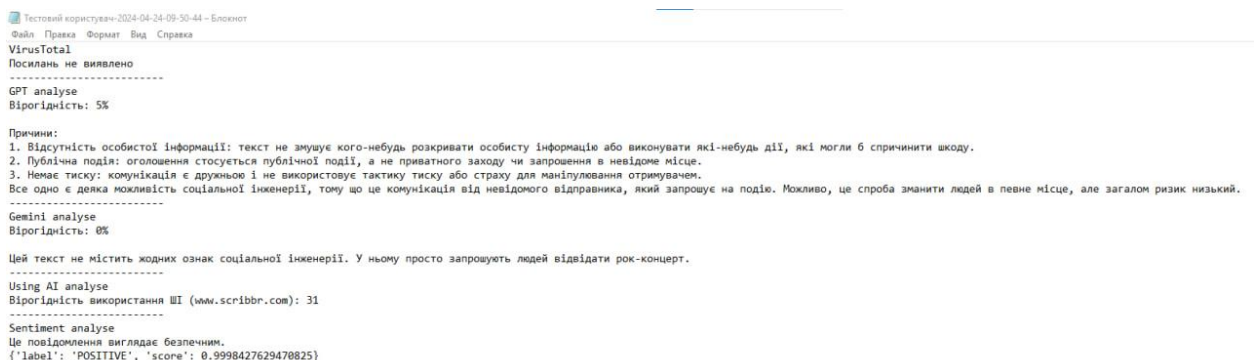


Рисунок 3.26 – Результат аналізу рекламного повідомлення

Як можемо побачити, застосунок не зреагував на звичайне рекламне повідомлення. Відкривши логи ми бачимо, що нічого небезпечного немає, тільки GPT дає 5% вірогідності соціальної інженерії. Але це пояснюється тим, що повідомлення прийшло від невідомого відправника.

Висновки до розділу 3

У цьому розділі було описано реалізацію програмного застосунку, який обробляє нові повідомлення в месенджері «Telegram» та аналізує їх на наявність соціальної інженерії. Аналіз проводиться за допомогою навчених моделей штучного інтелекту (GPT, Gemini), аналізу тональності тексту, пошуку використання ШІ при написанні повідомлення, а також виявлення зловмисних посилань. Результати аналізу логується і користувач може ознайомитись з ними більш детально в разі необхідності.

Було проведено тестування на декількох прикладах. Застосунок виявив соціальну інженерію в тих повідомленнях, де вона присутня. На звичайне рекламне повідомлення він ніяк не зреагував.

ВИСНОВКИ

В результаті виконання дипломної роботи було проведено дослідження соціальної інженерії та методів її використання.

Було проаналізовано статистику кібератак соціальної інженерії, які типи існують, в яких цілях її використовують та які канали розповсюдження атаки задіяні. Також було розглянуто інструменти, які автоматизують збір інформації та проведення кібератак, за рахунок чого зловмисники можуть провести більше персоніфікованих атак.

В другому розділі дипломної роботи було розглянуто загальні риси кібератак соціальної інженерії, методи суб'єктивного виявлення соціальної інженерії. Так як виявлення соціальної інженерії – це суб'єктивна оцінка, то на жаль не існує інструмента, який зі 100% точністю вкаже, що повідомлення легітимне, чи це соціальна інженерія. Проте деякі інструменти можуть допомогти виявити аномалії в повідомленні та зацентувати на цьому увагу, за рахунок чого, користувач більш ретельно перевірить повідомлення:

- Моделі ШІ (наприклад, GPT та Gemini).
- Моделі для аналізу тональності тексту.
- Детектори виявлення використання ШІ в тексті повідомлення.
- Сканери для перевірки посилань в повідомленні.

Якщо вони щось виявлять, то слід ретельно перевірити повідомлення, можливо користувача намагаються ввести в оману.

В третьому розділі дипломної роботи було розроблено та описано програмний застосунок, який перевіряє нові повідомлення, що надходять в месенджері «Telegram», аналізує їх за допомогою навчених моделей та намагається виявити чи присутня соціальна інженерія в повідомленні. В цьому розділі описано які функції використовувались та за допомогою яких модулів було реалізовано функціонал.

Було наведено обґрунтування використання мови програмування «Python», яка проста в написанні та розумінні, а також має перевагу у вигляді кросплатформності. Також проведено тестування на різних повідомленнях, як зловмисних так і звичайних. Застосунок зреагував лише на зловмисні, та залогував аномалії, які виявив при аналізі тексту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kevin D. Mitnick The art of deception. p. 17 [Електронний ресурс]. – Режим доступу: [https://github.com/InspectorDidi/Hacking-Books/blob/master/Kevin Mitnick - The Art of Deception.pdf](https://github.com/InspectorDidi/Hacking-Books/blob/master/Kevin%20Mitnick%20-%20The%20Art%20of%20Deception.pdf)
2. What is social engineering? [Електронний ресурс]. – Режим доступу: <https://www.ibm.com/topics/social-engineering>
3. Фішингові листи та витік даних. Як уникнути проблем з безпекою в Інтернеті, Eduard Kiiko. [Електронний ресурс]. – Режим доступу: <https://dou.ua/lenta/articles/security-awareness/>
4. Фішингові атаки: кіберфахівець розповів, як не втратити гроші у фейсбуці, телеграмі чи на OLX, Оксана Швед. [Електронний ресурс]. – Режим доступу: <https://suspilne.media/cherkasy/599361-fisingovi-ataki-fahivec-rozpoviv-ak-ne-vtratiti-grosi-u-telegrami-ci-na-olx/>
5. Cybersecurity threatscape: Q3 2023. [Електронний ресурс]. – Режим доступу: <https://www.ptsecurity.com/ww-en/analytics/cybersecurity-threatscape-2023-q3/>
6. 30 Social Engineering Statistics – 2023, Catherine Reed. [Електронний ресурс]. – Режим доступу: <https://firewalltimes.com/social-engineering-statistics/>
7. 70% to 90% of All Malicious Breaches are Due to Social Engineering and Phishing Attacks, Roger Grimes. [Електронний ресурс]. – Режим доступу: <https://blog.knowbe4.com/70-to-90-of-all-malicious-breaches-are-due-to-social-engineering-and-phishing-attacks>
8. Average organization targeted by over 700 social engineering attacks each year: report, Jonathan Greig. [Електронний ресурс]. – Режим доступу: <https://www.zdnet.com/article/average-organization-targeted-by-over-700-social-engineering-attacks-each-year-report/>

9. Social engineering cyberattacks and how they're impacting businesses, Devin Partida. [Електронний ресурс]. – Режим доступу: <https://www.securityinfowatch.com/cybersecurity/article/21203580/social-engineering-cyberattacks-and-how-theyre-impacting-businesses>
10. A Multivocal Literature Review on Growing Social Engineering Based Cyber-Attacks/ Threats During the COVID-19 Pandemic: Challenges and Prospective Solutions, Mohammad Hijji, Gulzar Alam. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/348165214_A_Multivocal_Literature_Review_on_Growing_Social_Engineering_Based_Cyber-Attacks_Threats_During_the_COVID-19_Pandemic_Challenges_and_Prospective_Solutions
11. Кібершахрайство: Фішинг. Вішинг. Смішинг. Бейтінг. [Електронний ресурс]. – Режим доступу: <https://kopirait.com.ua/kibershahrajstvo-fishyng-vishyng-smishyng-bejting/>
12. Топ месенджерів в Україні та світі 2020, Антон Юдін. [Електронний ресурс]. – Режим доступу: <https://ain.ua/2023/03/09/telegram-osnovnyj-mesendzher-opytuvannya/>
13. Топ месенджерів в Україні та світі 2020, Антон Юдін. [Електронний ресурс]. – Режим доступу: <https://marketer.ua/ua/top-messengers-in-ukraine-and-the-world/>
14. Урядова команда CERT-UA в 2023 році опрацювала 2543 кіберінциденти. [Електронний ресурс]. – Режим доступу: https://t.me/dsszzi_official/6735
15. Tutorial: The Four Phases of Social Engineering, RangeForce Team. [Електронний ресурс]. – Режим доступу: <https://www.rangeforce.com/blog/four-phases-of-social-engineering>
16. Pour se prémunir des attaques, encore faut-il connaître ses points faibles. Les outils OSINT existent pour détecter, trouver et réduire cette exposition. Nos confrères de CSO dressent une liste de solutions de cyber-renseignement. John

- Breeden П. [Электронный ресурс]. – Режим доступа: <https://www.lemondeinformatique.fr/actualites/lire-les-failles-zero-day-explosent-en-2023-93364.html>
- 17.4 releases of Maltego & Social Links: how they work and where to use them. Part 1. [Электронный ресурс]. – Режим доступа: <https://blog.sociallinks.io/4-releases-of-maltego-social-links-how-they-work-and-where-to-use-them-part-1/>
- 18.SpiderFoot automates OSINT for threat intelligence and mapping your attack surface, Steve Micallef. [Электронный ресурс]. – Режим доступа: <https://github.com/smicallef/spiderfoot?tab=readme-ov-file>
- 19.OSINT Framework, lockFALE. [Электронный ресурс]. – Режим доступа: <https://github.com/lockfale/osint-framework>
- 20.OSINT Framework. [Электронный ресурс]. – Режим доступа: <https://osintframework.com/>
- 21.Open-Source Phishing Toolkit, Jordan Wright. [Электронный ресурс]. – Режим доступа: <https://github.com/gophish/gophish>
- 22.Gophish User Guide. [Электронный ресурс]. – Режим доступа: <https://docs.getgophish.com/user-guide/building-your-first-campaign/launching-the-campaign>
- 23.The Social-Engineer Toolkit (SET) repository from TrustedSec, trustedsec. [Электронный ресурс]. – Режим доступа: <https://github.com/trustedsec/social-engineer-toolkit>
- 24.Social Engineer Toolkit (SET) tutorial for penetration testers, Karthik R, Contributor. [Электронный ресурс]. – Режим доступа: <https://www.computerweekly.com/tutorial/Social-Engineer-Toolkit-SET-tutorial-for-penetration-testers>
- 25.Создание фишингового сайта на Kali Linux с помощью SET, QUASAR. [Электронный ресурс]. – Режим доступа: <https://spy-soft.net/create-phishing-site-kali-linux-set/>

26. What the heck is a deepfake? [Электронный ресурс]. – Режим доступа: <https://security.virginia.edu/deepfakes>
27. DeepFaceLab 2.0. [Электронный ресурс]. – Режим доступа: <https://www.deepfakevfx.com/downloads/deepfacelab/>
28. DeepFaceLab, iperov. [Электронный ресурс]. – Режим доступа: <https://github.com/iperov/DeepFaceLab>
29. Clone a voice in 5 seconds to generate arbitrary speech in real-time, Corentin Jemine. [Электронный ресурс]. – Режим доступа: <https://github.com/CorentinJ/Real-Time-Voice-Cloning>
30. What is NLP? [Электронный ресурс]. – Режим доступа: <https://www.ibm.com/topics/natural-language-processing>
31. What Is ChatGPT? Semrush Team. [Электронный ресурс]. – Режим доступа: <https://www.semrush.com/blog/how-does-chatgpt-work/>
32. What is Google Gemini? Harry Guinness. [Электронный ресурс]. – Режим доступа: <https://zapier.com/blog/google-gemini/>
33. What Is Social Engineering? [Электронный ресурс]. – Режим доступа: <https://www.proofpoint.com/us/threat-reference/social-engineering>
34. 5 Ways to Recognize Social Engineering, Roger Grimes. [Электронный ресурс]. – Режим доступа: <https://blog.knowbe4.com/five-signs-of-social-engineering>
35. Virus Total, About us, How it works. [Электронный ресурс]. – Режим доступа: <https://web.archive.org/web/20200615192542/https://support.virustotal.com/hc/en-us/articles/115002126889>
36. Best AI Detector | Free & Premium Tools Compared, Koen Driessen. [Электронный ресурс]. – Режим доступа: <https://www.scribbr.com/ai-tools/best-ai-detector/>
37. Free AI Content Detector. [Электронный ресурс]. – Режим доступа: <https://www.scribbr.com/ai-detector/>

- 38.Sentiment Analysis: A Definitive Guide. [Электронный ресурс]. – Режим доступа: <https://monkeylearn.com/sentiment-analysis/>
- 39.Social Engineering Attack Detection, npolatidis. [Электронный ресурс]. – Режим доступа: <https://github.com/npolatidis/seader>
- 40.Python Reference Manual, Release 2.1.3, Guido van Rossum. [Электронный ресурс]. – Режим доступа: <http://www.kbs.twi.tudelft.nl/Documentation/Programming/python-2.1/ref/front.html>
- 41.Building Powerful Telegram Bots with Telethon in Python, КН Huang. [Электронный ресурс]. – Режим доступа: <https://medium.com/@jolojiuy/building-powerful-telegram-bots-with-telethon-in-python-76aef25902e9>
- 42.The OpenAI Python library provides convenient access to the OpenAI API from applications written in the Python language, Zulfikar Muhamad. [Электронный ресурс]. – Режим доступа: <https://github.com/zul-m/OpenAI>
- 43.The Ultimate Selenium Python Cheat Sheet for Test Automation, Vinayak Sharma. [Электронный ресурс]. – Режим доступа: <https://www.lambdatest.com/blog/selenium-python-cheat-sheet/>
- 44.Getting Started with the Vertex AI Gemini API and Python SDK. [Электронный ресурс]. – Режим доступа: <https://www.cloudskillsboost.google/focuses/86503?locale=ru&parent=catalog>
- 45.Transformers. [Электронный ресурс]. – Режим доступа: <https://huggingface.co/docs/transformers/index>
- 46.Vt-py, Marta Gómez Macías. [Электронный ресурс]. – Режим доступа: <https://github.com/VirusTotal/vt-py>
- 47.Deep_translator, prataffel. [Электронный ресурс]. – Режим доступа: https://github.com/prataffel/deep_translator
- 48.Plyer, Kivy. [Электронный ресурс]. – Режим доступа: <https://github.com/kivy/plyer>

- 49.Subprocess — Керування підпроцесами. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/subprocess.html>
- 50.Os — Різні інтерфейси операційної системи. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/os.html#module-os>
- 51.Sys — Системні параметри та функції. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/sys.html#module-sys>
52. Re — Операції регулярних виразів. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/re.html#module-re>
53. Time — Час доступу та перетворення. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/time.html#module-time>
- 54.Datetime — Основні типи дати та часу. [Електронний ресурс]. – Режим доступу: <https://docs.python.org/uk/3/library/datetime.html#module-datetime>

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Тези наукових доповідей:

1. Anastasiia Zavorodnia, Oleksandr Pasko, Volodymyr Nakonechnyi Malicious functionality of AI-powered tools. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2023, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 2023. 343 p.
2. Завгородня А.С., Пасько О.І., Наконечний В.С. Штучний інтелект як інструмент кібербезпеки. П78 Проблеми кібербезпеки інформаційно-телекомунікаційних систем: Збірник матеріалів доповідей та тез; м. Київ, 26 квітня 2024 року; Київський національний університет імені Тараса Шевченка / Редкол.: В.В. Ільченко, д.ф.-м.н., проф., (голова); та ін. – К.: ВПЦ "Київський університет", 2024. – 171 с.

ДОДАТОК Б

Check message.py

```
# -*- coding: utf-8 -*-

import sys
import re
import vt
import os
from time import sleep
from datetime import datetime
from openai import OpenAI
from selenium.webdriver import webdriver
from selenium.webdriver.common.by import By
from plyer import notification
import vertexai
from vertexai.generative_models import GenerativeModel, Part
from transformers import pipeline
from deep_translator import GoogleTranslator
from get_config_info import get_config

# Параметри, необхідні для аналізу
api_key_vt = get_config().api_vt
openai_api_key = get_config().api_openai
sentiment_pipeline = pipeline("sentiment-analysis")

# Глобальна змінна (словник), в якій зберігається інформація, чи виявлено
аномалію
```

```

make_notification = {'VirusTotal':0, 'GPT':0, 'Gemini':0, 'Site':0, 'Sentiment':0}

def extract_urls(text):
    # Регулярний вираз для пошуку URL
    url_regex = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[*\(\)\,])(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    urls = re.findall(url_regex, text)
    cleaned_urls = [re.sub(r'^\w\s-$', "", url) for url in urls] # Удаление знаков
препинания с конца каждого URL
    return cleaned_urls

def check_urls_virus_total(api_key, urls):
    global make_notification
    message = ""
    for url in urls:
        # Створюємо клієнт для Virus Total
        client = vt.Client(api_key)
        client.scan_url(url, wait_for_completion=True)
        url_id = vt.url_id(url)
        # Надсилаємо посилання на перевірку
        site = client.get_object("/urls/{ }", url_id)
        client.close()
        # Вивантажуємо результат сканування
        malicious_detections = site.last_analysis_stats['malicious']
        # Якщо хоа б один антивірус зреагував, то помічаємо, як підозріле.
        Готуємо відповідне повідомлення.
        if malicious_detections > 0:

```

```

message = message + "Увага: посилання " + url + " - зловмисне.
Виявлено " + str(malicious_detections) + " спрацювань.\n"

```

```

make_notification['VirusTotal'] = 1

```

```

if message != "":

```

```

    return message

```

```

else:

```

```

    return "Підозрілих посилань не виявлено"

```

```

def detect_social_engineering_GPT(text):

```

```

    global make_notification

```

```

    # Ініціалізуємо клієнт openai

```

```

    client = OpenAI(api_key = openai_api_key)

```

```

    # Надсилаємо запит

```

```

    response = client.chat.completions.create(

```

```

        model="gpt-4",

```

```

        messages=[

```

```

            {"role": "user",

```

```

                "content": "Яка вірогідність використання соціальної інженерії в
цьому тексті: " + text + ".\nНапиши таким чином: 'Вірогідність: $per', замість $per
пишеш значення у відсотках від 0 до 100. Також напиши, чому ти так вважаєш
(напиши по пунктах)"}

```

```

            ]

```

```

        )

```

```

    # Обробка відповіді на наш запит

```

```

    probability = response.choices[0].message.content

```

```

    percents = re.findall(r'\d+%', probability)

```

```

    # Реагуємо, якщо відсоток вірогідності CI > 60

```

```

if int(percents[0][:-1]) >= 60:
    make_notification['GPT'] = 1
return probability, percents

def detect_social_engineering_Gemini(analyse_text, project_id = "gemini-
social", location = "europe-central2"):
    global make_notification
    # Ініціалізуємо Vertex AI
    vertexai.init(project=project_id, location=location)
    # Завантажуємо модель
    multimodal_model = GenerativeModel("gemini-1.0-pro-vision")
    # Надсилаємо запит
    response = multimodal_model.generate_content(
        [
            "Яка вірогідність використання соціальної інженерії в цьому тексті:
" + analyse_text + ".\nНапиши таким чином: 'Вірогідність: $per', замість $per
пишеш значення у відсотках від 0 до 100. Також напиши, чому ти так вважаєш"
        ]
    )

    # Реагуємо, якщо відсоток вірогідності CI > 60
    percents = re.findall(r'\d+%', response.text)
    if int(percents[0][:-1]) >= 60:
        make_notification['Gemini'] = 1
    return response.text, percents

def detect_using_AI(text):
    global make_notification

```

```

# Запускаємо драйвер браузера
driver = webdriver.Chrome()

# Відкриваємо веб-сторінку
driver.get('https://www.scribbr.com/ai-detector/')

# Знаходимо поля для введення тексту
input_field = driver.find_element(By.CSS_SELECTOR,
'div[contenteditable="true"]')
input_field.send_keys(text)

# Знаходимо кнопку та нажимаємо її
submit_button = driver.find_element(By.ID, "aiDetectorButton")
submit_button.click()
sleep(10)

# Шукаємо результат в контейнері з певним ідентифікатором
result_element = driver.find_element(By.CSS_SELECTOR,
"span.percentage.purple.d-block.text-center.text-purple-gradient")
result_text = result_element.text

# Реагуємо, якщо відсоток використання III > 60
result_text = int(result_text.replace('%', ''))
if int(result_text) >= 60:
    make_notification['Site'] = 1
return f'Вірогідність використання III (www.scribbr.com): {result_text}'

def detect_social_engineering_Sentiment(message):
    global make_notification

```

Створюємо об'єкт для перекладача, в якому зазначаємо на чку мову
необхідно перекласти

```
translator = GoogleTranslator(source='auto', target='en')
```

#Перекладаємо текст на англійську мову (модель для аналізу тональності
навчена лише на англійських словах.)

```
translated_text = translator.translate(message)
```

Проводимо аналіз тональності тексту

```
sentiment = sentiment_pipeline(translated_text)[0]
```

Якщо оцінка тональності > 75 та має негативний стан, то помічаємо як
аномалію

```
if sentiment['label'] == 'NEGATIVE' and sentiment['score'] > 0.75:
```

```
    make_notification['Sentiment'] = 1
```

```
    result = "Це повідомлення може містити елементи соціальної інженерії"
```

```
else:
```

```
    result = "Це повідомлення виглядає безпечним."
```

```
return result, sentiment
```

```
def notificate(result, user):
```

```
    # Виведемо оповіщення про знайдені аномалії в повідомленні
```

```
    notification.notify(title="Виявлено підозріле повідомлення від " + user,  
message=result)
```

```
def log(vt, gpt, gemini, ai, sentiment, sentiment_score, username):
```

```
    # Збережемо результат аналізу
```

```
    log_info = "VirusTotal\n" + vt + "\n" + 5*"-----" + "\n" + "GPT analyse\n" +  
gpt + "\n" + 5*"-----" + "\n" + "Gemini analyse\n" + gemini + "\n" + 5*"-----" + "\n" +
```

```
"Using AI analyse\n" + ai + "\n" + 5*"-----" + "\n" + "Sentiment analyse\n" + sentiment
+ "\n" + str(sentiment_score)
```

```
date = datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
```

Створимо файл логу. В назві вкажемо відправника повідомлення та час створення логу

```
filename = f"{username}-{date}.txt"
```

```
# Вкажемо, що логи повинні зберігатись в директорії "logs"
```

```
log_directory = "logs"
```

```
if not os.path.exists(log_directory):
```

```
    os.makedirs(log_directory)
```

```
file_path = os.path.join(log_directory, filename)
```

```
# Запишемо результат аналізу в файл
```

```
with open(file_path, 'w', encoding='utf-8') as file:
```

```
    file.write(log_info)
```

```
print("Аналіз залоговано")
```

```
if __name__ == '__main__':
```

```
    # Перевіряємо, чи отримали перший аргумент
```

```
    if len(sys.argv) > 1:
```

```
        # Зчитуємо текст з першого аргументу
```

```
        text = sys.argv[1]
```

```
        username = sys.argv[2]
```

Якщо аргументів не передали, то виводимо сповіщення і завершуємо процес

```
else:
```

```
    print("Текст не був надіслан.")
```

```
    sys.exit(1)
```

```
if text:
```

```

# Збережемо результати аналізу в змінні
SE_GPT, SE_GPT_score = detect_social_engineering_GPT(text)
SE_Gemini, SE_Gemini_score = detect_social_engineering_Gemini(text)
Using_AI_score = detect_using_AI(text)
SE_sentiment, SE_Sentiment_score =
detect_social_engineering_Sentiment(text)

#Якщо повідомлення містить посилання, то передати їх функції для
перевірки за допомогою VirusTotal
urls = extract_urls(text)
if urls:
    virustotal_results = check_urls_virus_total(api_key_vt, extract_urls(text))

# Якщо хоча б один ключ в словнику "make_notification" має значення
1, то робиться оповіщення.
if 1 in make_notification.values():
    # Якщо Virus Total звійшов шкідливе посилання, то робиться
оповіщення, де зазначається, яке посилання небезпечне та скільки антивірусів на
ньогозреагувало
    if make_notification.get('VirusTotal') == 1:
        notificate(virustotal_results, username)
        print(virustotal_results)
        sleep(7)
    # Створюється змінна notif в яку буде збережено всі повідомлення
про аномалії
    notif = ""
    if make_notification.get('GPT') == 1:
        notif = notif + "Вірогідність соціальної інженерії (GPT): " +
SE_GPT_score[0] + "\n"
    if make_notification.get('Gemini') == 1:

```

```
    notif = notif + "Вірогідність соціальної інженерії (Gemini): " +
SE_Gemini_score[0] + "\n"
    if make_notification.get('Site') == 1:
        notif = notif + "Вірогідність використання ІІІ: " + Using_AI_score +
"\n"
    if make_notification.get('Sentiment') == 1:
        notif = notif + SE_sentiment
    if notif != "":
        notificate(notif, username)
    print(notif)

# Збереження результатів сканування у логи
if urls:
    log(virustotal_results, SE_GPT, SE_Gemini, Using_AI_score,
SE_sentiment, SE_Sentiment_score, username)
else:
    log("Посилань не виявлено", SE_GPT, SE_Gemini, Using_AI_score,
SE_sentiment, SE_Sentiment_score, username)

sys.exit(0)
```

ДОДАТОК В

get_telegram_messages.py

```
import os
from subprocess import call
from telethon.sync import TelegramClient, events
from get_config_info import get_config

# Параметри для вашого клієнта
api_id = get_config().api_id
api_hash = get_config().api_hash
session_file = 'phishing.session'

# Перевірка наявності сесії
if not os.path.exists(session_file):
    print("Сесія не знайдена, створюємо нову...")
    # Запуск скрипта для створення сесії
    call([r".\venv\Scripts\python.exe", "Authorization.py"])

# Створення клієнта та робота з Телеграмом
client = TelegramClient(session_file, api_id, api_hash)

# Обробка нових повідомлень
@client.on(events.NewMessage(incoming=True))
async def handle_new_message(event):
    if event.is_private: # Перевіряємо, що це особисте повідомлення
        sender = await event.get_sender()
        sender_name = sender.first_name if sender.first_name else "Без имени"
```

```
if sender.last_name: # Додаємо прізвище, якщо воно є
    sender_name += f" {sender.last_name}"

print(fНове повідомлення від {sender_name}: {event.text}')
call([r".\venv\Scripts\python.exe", "Check message.py", event.text,
sender_name])

async def main():
    print("Моніторинг нових повідомлень... Натисніть Ctrl+C для остановки.")
    await client.start()
    await client.run_until_disconnected()

# Запускаємо моніторинг
with client:
    client.loop.run_until_complete(main())
```

ДОДАТОК Г

Authorization.py

```
from get_config_info import get_config
from telethon.sync import TelegramClient

api_id = get_config().api_id
api_hash = get_config().api_hash
phone_num = get_config().phone
code = input("Input your 2FA code, if you have it: ")

client = TelegramClient('phishing', api_id, api_hash)

if code == '': #Якщо користувач не використовує двохфакторну аутентифікацію,
то використовується лише номер телефону
    client.start(phone_num)
else:
    client.start(phone_num, code)

async def main():
    # Вивід інформації про користувача
    me = await client.get_me()
    print(me.username)
    print(me.phone)

with client:
    client.loop.run_until_complete(main())
```

ДОДАТОК Д

get_config_info.py

```
class UserConfig:
    def __init__(self, api_id=None, api_hash=None, phone=None, api_vt=None,
api_openai=None):
        self.api_id = api_id
        self.api_hash = api_hash
        self.phone = phone
        self.api_vt = api_vt
        self.api_openai = api_openai

def get_config():
    config_file = "./accounts_settings.txt"
    config = UserConfig()
    with open(config_file, 'r') as file:
        for line in file:
            key, value = line.strip().split('=', 1)
            key = key.strip()
            value = value.strip()
            if key == 'api_id':
                config.api_id = int(value)
            elif key == 'api_hash':
                config.api_hash = value
            elif key == 'phone':
                config.phone = value
            elif key == 'api_key_vt':
                config.api_vt = value
```

```
elif key == 'openai_api_key':  
    config.api_openai = value
```

```
return config
```

```
if __name__ == '__main__':  
    user_config = get_config()  
    print(f'API ID: {user_config.api_id}')  
    print(f'API Hash: {user_config.api_hash}')  
    print(f'Phone Number: {user_config.phone}')
```