МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики інформаційних технологій та інженерії

"До захисту допущена"
Зав. кафедри комп'ютерних наук та прикладної математики
_____
«___» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

Методи та алгоритми OSINT для здійснення конкурентної розвідки

Виконав: Струк Андрій Володимирович _____
(прізвище, ім’я, по батькові) (підпис)
група ІПЗ-41

Керівник: ст. викладач Рощенюк Алла Михайлівна _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2025

ЗМІСТ

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП	5
РОЗДІЛ І. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ OPEN-SOURCE INTELLIGENCE (OSINT)	7
1.1 Визначення Open-Source Intelligence (OSINT)	7
1.2 Аналіз конкурентного середовища	11
1.3 Виявлення загроз та потенційних атак	14
1.4 OSINT для бізнесу	16
Висновок	22
РОЗДІЛ ІІ. ОГЛЯД ТЕХНОЛОГІЙ ТА СЕРЕДОВИЩА OSINT	23
2.1 Проблема вибору систем моніторингу	23
2.2 Розвідка відкритих джерел OSINT	26
2.3 Джерела OSINT	29
2.4 Конкурентна розвідка	32
Висновок	35
РОЗДІЛ ІІІ. РОЗРОБКА ТЕЛЕГРАМ-БОТУ ДЛЯ OSINT РОЗВІДКИ	36
3.1 Telegram-бот як засіб автоматизації збору відкритих даних (OSINT)	36
3.2 Розробка Telegram-бота "OSINT Project bot"	40
3.3 Інтеграція Telegram-бота	47
ВИСНОВКИ	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53
ДАТОК 1	55

Реферат

Кваліфікаційна робота: 60 сторінок, 13 ілюстрацій, 12 джерел.

Метою кваліфікаційної роботи є розробка комплексної методології застосування OSINT-інструментів для проведення конкурентної розвідки, що забезпечує ефективний збір, аналіз та інтерпретацію публічно доступних даних.

Об'єкт дослідження – сучасні техніки та інструмент OSINT (Open Source Intelligence), що використовуються для аналізу конкурентного середовища.

Предметом дослідження – алгоритми обробки відкритих джерел інформації та методи їх застосування в конкурентній розвідці.

Методи дослідження – Аналіз соціальних мереж (Twitter, Facebook, LinkedIn), Веб-скрапінг (Python, BeautifulSoup, Scrapy), Робота з відкритими базами даних (WHOIS, реєстри підприємств), Геолокаційні технології (Google Maps), Аналіз медіа-контенту (TinEye, RevImg)

Результати дослідження демонструють, що застосування OSINT-методологій дозволяє отримувати цінні бізнес-інсайти, прогнозувати ринкові тенденції та виявляти конкурентні переваги. Основний внесок роботи полягає у систематизації алгоритмів збору та аналізу даних, а також розробці практичних рекомендацій щодо їх застосування в реальних бізнес-умовах..

Ключові слова: OSINT, КОНКУРЕНТНА РОЗВІДКА, ВІДКРИТІ ДЖЕРЕЛА, ВЕБ-АНАЛІТИКА, DATA MINING, СОЦІАЛЬНІ МЕРЕЖІ, ВЕБ-СКРАПІНГ, БІЗНЕС-АНАЛІТИКА, КІБЕРРОЗВІДКА, ІНФОРМАЦІЙНА БЕЗПЕКА.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

OSINT (Open Source Intelligence) – методологія збору та аналізу інформації з відкритих джерел.

SOCMINT (Social Media Intelligence) – аналіз даних з соціальних мереж для отримання корисних інсайтів.

WEBINT (Web Intelligence) – збір і обробка даних з вебресурсів, включаючи сайти, форуми та блоги.

WHOIS – протокол для отримання інформації про доменні імена та IP-адреси.

TinEye – сервіс пошуку зображень за зразком.

Maltego – інструмент для візуалізації зв'язків між різними об'єктами в OSINT-розслідуваннях.

Shodan –система пошуку виявлення підключень пристроїв до інтернету.

Data Mining – процес виявлення закономірностей у великих масивах даних.

KYC (Know Your Customer) – процедура ідентифікації клієнтів, що використовується в конкурентній розвідці.

CI (Competitive Intelligence) – систематичний збір та аналіз інформації про конкурентів.

VPN (Virtual Private Network) – технологія для забезпечення анонімності під час збору даних.

ВСТУП

У сучасних умовах глобалізації та цифровізації інформація стає ключовим ресурсом для прийняття стратегічних бізнес-рішень. Особливе значення набуває конкурентна розвідка, яка дозволяє компаніям аналізувати ринкові тенденції, виявляти слабкі сторони конкурентів та формувати власні переваги. Традиційні методи збору даних вже не відповідають потребам динамічного бізнес-середовища, тому все більшого значення набувають технології OSINT (Open Source Intelligence)- збір та аналіз отриманої інформації з відкритих джерел.

Актуальність теми обумовлена стрімким зростанням обсягів публічно доступних даних у соціальних мережах, державних реєстрах, форумах, новинних ресурсах тощо. Водночас, більшість підприємств досі не використовують повний потенціал OSINT-інструментів, обмежуючись поверхневим моніторингом конкурентів. Це призводить до втрати важливих інсайтів та зниження ефективності маркетингових і бізнес-стратегій.

Наукова новизна дослідження полягає в систематизації сучасних OSINT-методик, розробці алгоритмів автоматизованого збору даних та їх аналізу для потреб конкурентної розвідки. Особливу увагу приділено:

- інтеграції різних джерел інформації (соціальні мережі, бази даних, та інші сервіси);

- застосуванню штучних інтелектів для обробки великого масиву даних;

- візуалізації результатів для прийняття управлінських рішень.

Метою роботи є розробка комплексної методики застосування OSINT для конкурентного аналізу. Для її досягнення поставлено такі завдання:

1. Аналіз існуючих OSINT-інструментів та виявлення їх обмежень.

2. Запровадження алгоритмів збору даних із соціальних мереж, форумів.

3. Створення системи фільтрації та верифікації інформації.

4. Впровадження методів аналізу зв’язків між суб’єктами ринку.

Методи дослідження включають в себе:

- WEBIN (збір даних із вебресурсів за допомогою Python, Scrapy);

- SOCMINT (аналіз соцмереж через API Twitter, Facebook, LinkedIn);

Практична значимість полягає у створенні інструментарію, який дозволить:

- автоматизувати моніторинг діяльності конкурентів;
- оцінювати репутацію брендів у цифровому середовищі;
- прогнозувати зміни у поведінці споживачів.

Впровадження OSINT-методик у конкурентну розвідку дозволить підприємствам отримувати переваги за рахунок:

- оперативного доступу до актуальних даних;
- зниження витрат на традиційні маркетингові дослідження;
- підвищення точності стратегічного планування.

Ця робота сприятиме розвитку цифрової аналітики в Україні та створенню ефективних інструментів для бізнес-аналітики.

РОЗДІЛ І. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ OPEN-SOURCE INTELLIGENCE (OSINT)

1.1 Визначення Open-Source Intelligence (OSINT)

Open-Source Intelligence (OSINT) – це стратегічний підхід до збору, аналізу та інтерпретації інформації з відкритих джерел, який дозволяє організаціям отримувати конкурентні переваги, виявляти потенційні ризики та приймати обґрунтовані бізнес-рішення. В умовах цифрової економіки OSINT став незамінним інструментом для бізнес-аналітики, кібербезпеки та стратегічного планування.

Конкурентна розвідка на основі OSINT забезпечує комплексний підхід до аналізу ринкового середовища та дозволяє:

- Виявляти слабкі місця у бізнес-процесах конкурентів через аналіз їхньої цифрової активності, публічних звітів, вакансій та обговорень у професійних спільнотах. Це дає можливість адаптувати власні стратегії та заповнювати ринкові ніші.
- Отримувати стратегічну інформацію про ринок та галузь шляхом моніторингу змін у законодавстві, аналізу тендерної документації, дослідження патентних баз та наукових публікацій. Такі дані допомагають прогнозувати ринкові тренди та інвестувати в перспективні напрямки.
- Запобігати загрозам кібербезпеки через виявлення витоків конфіденційної інформації в darknet, моніторинг paste-сайтів та аналіз цифрових слідів співробітників. Проактивний підхід дозволяє мінімізувати ризики виникнення інцидентів.
- Контролювати репутацію компанії за допомогою аналізу соціальних мереж, форумів, ЗМІ та відгуків клієнтів. Систематичний моніторинг дозволяє оперативно реагувати на негативні публікації та формувати позитивний імідж бренду.

- Оцінювати ризики витоку даних через сканування відкритих баз даних, GitHub-репозиторіїв та мережевих ресурсів на наявність конфіденційної інформації. Це особливо актуально для захисту комерційних таємниць та інтелектуальної власності.

Сучасні OSINT-інструменти (як Maltego, SpiderFoot, Buscador) значно автоматизують процес збору даних, а застосування штучного інтелекту для аналізу великих масивів інформації дозволяє виявляти складні взаємозв'язки та приховані закономірності. Професійне застосування OSINT-методологій вимагає не лише технічних навичок, але й глибокого розуміння бізнес-процесів та галузевих особливостей.

Важливо враховувати, що ефективне використання OSINT для конкурентної розвідки має ґрунтуватися на дотриманні законодавчих норм та етичних принципів роботи з даними. Правильно налаштовані OSINT-процеси стають стратегічним активом компанії, що забезпечує інформаційну перевагу в умовах жорсткої конкуренції.

Так що ж включається в OSINT-дослідження?

Методика збору інформації з відкритих джерел (OSINT) передбачає систематичний пошук, фільтрацію та аналіз публічно доступних даних з різноманітних цифрових ресурсів. Цей процес включає кілька ключових напрямків дослідження:

а). Веб-ресурси та соціальні мережі (Сучасні соціальні платформи є найбагатшим джерелом інформації для OSINT-досліджень:

- LinkedIn дозволяє аналізувати:
  - кадрові зміни в компаніях-конкурентах
  - організаційні структури
  - професійні зв'язки співробітників
  - напрямки професійного розвитку персоналу

- Facebook надає доступ до:

- непублічних груп і спільнот
- геотегів і місць перебування
- соціальних зв'язків
- інтересів цільових аудиторій

- Twitter є цінним джерелом для:
- моніторингу трендів у реальному часі
- аналізу реакцій на події
- виявлення ключових інфлюєнсерів
- відстеження хештегів і тем

- Telegram:
- містить спеціалізовані спільноти
- дозволяє виявляти неформальні обговорення
- є джерелом експертної інформації
- часто містить дані про нові технології та продукти

Для ефективного збору даних із соцмереж використовуються спеціалізовані інструменти (Socialbearing, Tweepy, Facebook Graph API), що дозволять автоматизувати процеси збору інформації.

б) Форуми, новинні портали та бази даних

Спеціалізовані інтернет-ресурси містять критично важливу інформацію для конкурентного аналізу:

- Професійні форуми (Stack Overflow, GitHub Discussions):
- технічні обговорення продуктів
- проблеми і рішення
- експертні оцінки технологій

-Новинні портали:
- анонси продуктів і послуг
- зміни в керівництві компаній
- фінансові звіти
- аналітика ринків

- Відкриті бази даних:
  - WHOIS (інформація про домени)
  - Shodan (пошук підключених пристроїв)
  - реєстри SSL-сертифікатів
  - архіви веб-сторінок (Wayback Machine)

Для роботи з цими джерелами ефективно використовуються технології веб-скрапінгу (BeautifulSoup, Scrapy) та спеціалізовані пошукові оператори (Google Dorks).

в) Компанійні реєстри, звіти та патенти

Офіційні джерела корпоративної інформації забезпечують доступ до структурованих даних:

- Державні реєстри:
  - Єдиний держреєстр підприємств
  - Реєстри ліцензій і дозволів
  - Податкові звіти
  - Дані про участь у держзакупівлях
- Фінансова звітність:
  - річні звіти публічних компаній
  - дані про доходи і витрати
  - інвестиційні портфелі
  - структура власності
- Патентні бази (WIPO, USPTO):
  - зареєстровані технології
  - напрямки НДДКР
  - інтелектуальна власність
  - технічні рішення конкурентів

Для аналізу цих джерел використовуються спеціалізовані платформи (OpenCorporates, Orbis) та методи дата-майнінгу для виявлення зв'язків і тенденцій.

Отже, комплексне використання різноманітних відкритих джерел дозволяє формувати повноцінну картину конкурентного середовища. Ключовим аспектом є інтеграція отриманих даних в єдину аналітичну систему з подальшою візуалізацією результатів для прийняття управлінських рішень. Ефективність OSINT-дослідження безпосередньо залежить від правильного вибору джерел інформації, застосування спеціалізованих інструментів і кваліфікованої інтерпретації отриманих даних.

1.2 Аналіз конкурентного середовища

Комплексний аналіз конкурентів передбачає вивчення ключових аспектів їх діяльності:

а) Структура бізнес-моделі:

- Джерела доходів і прибутковість
- Цінові стратегії та політика знижок
- Канали дистрибуції продукції
- Партнерські мережі і альянси

б) Маркетингові активності:

- Рекламні кампанії (digital та традиційні)
- Активність у соціальних мережах
- Програми лояльності
- Участь у виставках і івентах

в) Продуктова стратегія:

- Асортиментна політика
- Життєвий цикл продуктів
- Інноваційні розробки
- Портфель брендів

Для збору даних використовуються:

- Аналіз фінансової звітності
- Дослідження прес-релізів і інтерв'ю керівництва
- Моніторинг рекламних матеріалів
- Аналіз вакансій і вимог до персоналу

Окрім того системний підхід до аналізу ринку включає в себе:

а)Макроекономічні фактори:

- Динаміка ринкових сегментів
- Зміни в законодавчому полі
- Технологічні тренди галузі

б)Фінансовий аналіз:

- Коефіцієнти ліквідності та платоспроможності
- Рентабельність активів і капіталу
- Інвестиційна активність
- Структура витрат

в)SWOT-аналіз конкурентів:

- Сильні і слабкі сторони
- Можливості і загрози
- Конкурентні переваги

Провівши оцінку ризиків та вразливостей, необхідно провести аналіз потенційних загроз , а саме:

а)Операційні ризики:

- Залежність від постачальників
- Кадрові проблеми
- Логістичні вразливості

б)Репутаційні ризики:

- Скандали і судові процеси
- Негатив у ЗМІ

- Відгуки споживачів

в)Технологічні ризики:

- Застарілість виробництва
- Відставання в НДДКР
- Залежність від ключових технологій

Напевне самим важливим в цьому питанні є кібербезпеки та цифровий слід кожної компанії, і в свою чергу це включає в себе комплексний аналіз напрямків:

а) Моніторинг цифрового сліду:

- Аналіз соціальних профілів співробітників
- Виявлення конфіденційної інформації
- Моніторинг paste-сайтів і форумів

б) Технічний аналіз інфраструктури:

- Сканування відкритих портів
- Аналіз конфігурацій серверів
- Перевірка SSL-сертифікатів
- Виявлення вразливостей CMS

в) Заходи з попередження витоків:

- Моніторинг darknet-майданчиків
- Аналіз git-репозиторіїв
- Перевірка конфігурацій хмарних сховищ

Важливо сказати що для проведення аналізу також використовуються спеціалізовані інструменти, такі як:

- Maltego робить візуалізацію зв'язків
- Shodan робить пошук підключених пристроїв
- Have I Been Pwned для перевірки витоків даних
- Nmap для сканування мереж
- Burp Suite для тестування веб-додатків

Отже, правильний системний підхід для аналізу конкурентного середовища та кібербезпеки дозволяє не лише виявляти поточну загрозу, а й прогнозувати потенційність ризиків. Інтеграція отриманих даних в стратегічне планування забезпечує формування конкурентних переваг і зменшення потенційних загроз для бізнесу. Особливу увагу потрібно приділяти частому моніторингу змін у конкурентному середовищі та оперативному реагуванню на виявлені ризики.

1.3 Виявлення загроз та потенційних атак

Проводячи аналіз персоналу та інформаційних загроз, ми підходжимо до питання дослідження співробітників та ключових осіб, які володіють певною інформацією. Тут необхідно описати системний підхід до аналізу персоналу який включає в себе :

а)Профільний аналіз:

- Верифікація професійної історії через LinkedIn
- Аналіз академічного бекграунду
- Виявлення сумнівних зв'язків і конфліктів інтересів
- Моніторинг публічної активності в соцмережах

б)Техніки розкриття інформації:

- Пошук у спеціалізованих базах (RocketReach, ZoomInfo)
- Аналіз метаданих публічних документів
- Виявлення альтернативних ідентифікаторів (email, нікнейми)
- Геолокаційний аналіз цифрового сліду

в)Оцінка ризиків:

- Виявлення співробітників з доступом до критичної інформації
- Аналіз моделі поведінки в соцмережах
- Оцінка вразливості до соціальної інженерії

Що дозволяє слідкувати та виявляти шахрайські схем атакож боротися із кіберзлочинністю. Для цього необхідно проводити комплексний підхід до ідентифікації зловмисної активності, а саме:

а) Методи виявлення:

- Аналіз транзакційних шаблонів
- Виявлення підозрілих мережевих з'єднань
- Моніторинг darknet-майданчиків
- Аналіз логів доступу до систем

б) Інструменти дослідження:

- Використання платформ типу Recorded Future
- Застосування AI для аналізу поведінки
- Блокчейн-аналіз підозрілих транзакцій
- Використання Threat Intelligence платформ

в) Типові індикатори:

- Аномальна активність в не робочий час
- Спробу доступу з незвичних локацій
- Використання компрометованих облікових записів
- Наявність даних у витоках (Have I Been Pwned)

Важливим етапом є боротьба з дезінформацією, і тут є багато ефективних методів протидії, а саме:

а) Техніка виявлення:

- Аналіз графу соціальних зв'язків
- Оцінка шаблонів поведінки ботів
- Виявлення координатованих кампаній
- Аналіз часових паттернів активності

б) Інструментарії:

- Використання Botometer
- Застосування CrowdTangle
- Аналіз метаданих медіафайлів
- Використання геолокаційних маркерів

в) Профілактичні заходи:

- Розробка системи для раннього попередження

- Розробка стратегій комунікації
- Підготовка кризових сценаріїв
- Тренінги з медіаграмотності

Для керівництва дуже важливим моментом є фіксації всіх подій, тому таке питання як звітність та рекомендації стоять понад усе. Інколи структура фінального звіту складатиметься із результати дослідження, які включають в себе: матрицю виявлених загроз, карту ризиків з оцінкою ймовірності, візуалізацію ключових зв'язків та хронологію інцидентів.

Наступним необхідним пунктом є стратегічна рекомендація до дій. Яка включатиме в себе: пріоритетні напрямки захисту, план реагування на інциденти, оптимізацію політик безпеки та програму підвищення обізнаності завдяки технічним рішення: оновлення систем моніторингу, впровадження DLP-систем, рекомендаціям з конфігурації мережі, покращенням процедур автентифікації.

Як висновок, ми можемо сказати, що комплексний підхід до аналізу персоналу та інформаційних загроз дозволяє формувати проактивну стратегію безпеки. Важливо слід приділяти більше уваги постійному моніторингу цифрового середовища, регулярному оновленню процедур безпеки та підвищенню обізнаності співробітників. Ефективна реалізація рекомендацій дозволить значно знизити операційні ризики та захистити критичні активи компанії.

1.4 OSINT для бізнесу.

1. Оцінка конкурентного середовища – отримання даних про діяльність конкурентів, аналіз їхніх стратегій та потенційних ризиків. Аналіз конкурентів – це процес вивчення діяльності інших гравців ринку, що пропонують схожі продукти чи послуги. Завдяки цьому можна оцінити їх сильні та слабкі сторони, дослідити асортимент, цінову політику та маркетингові стратегії. І на підставі отриманої інформації створити унікальну торгову пропозицію та залучити більше клієнтів, відрізняючись від інших брендів чи компаній.

Якщо підприємство стартує без ретельного вивчення конкурентів, його успіх буде короткочасним. У довгостроковій перспективі така стратегія призводить до провалу: аналіз життєвого циклу клієнта виявить низьку ефективність, оскільки покупці швидше перейдуть до більш привабливих альтернатив. Наприклад, пропонує сервіс краще чи ставить нижчу ціну.

Вивчення ринку перед запуском нового продукту — ключовий етап. Конкурентний аналіз дає змогу виявити невикористані можливості для залучення аудиторії, особливо в соцмережах, і це варто робити вже на початковому етапі. У подальшому це дозволить експериментувати як з самим продуктом, так і з методами його просування.

Не менш важливо проаналізувати, як конкуренти представлені онлайн. У цифрову епоху їхня слабкість — ваш шанс: якщо покупки в інтернеті для клієнтів стали звичкою, ваша digital-стратегія має не лише враховувати дії суперників, але й запропонувати щось унікальне, щоб бренд запам’ятався.

2. Захист від витоків інформації – моніторинг згадок компанії та співробітників у відкритих джерелах. Витік конфіденційної інформації – загроза, яка може реалізуватися в будь-який момент. Щоб мінімізувати ризики, компанії варто заздалегідь розробити чіткий алгоритм дій: від запобіжних заходів до плану реагування на інцидент.

Чому потрібна політика інформаційної безпеки?

Навіть якщо витік станеться, наявність внутрішніх протоколів дозволить:

оперативно вжити заходів;

зменшити фінансові та репутаційні втрати;

уникнути порушень законодавства.

Ключові елементи базової політики

Відповідальність керівництва – хто контролює процес?

Дотримання законів – як забезпечити правову відповідність?

Механізми захисту – технічні та організаційні рішення.

Управління даними – класифікація та обмежений доступ.

Навчання персоналу – як запобігти людському фактору?

План реагування – дії при витоку.

Регулярні оновлення – адаптація до нових загроз.

Ця політика може бути не лише внутрішнім документом, а й публічним інструментом – це підвищує довіру клієнтів і партнерів.

Персонал – найслабша ланка

Більшість витоків виникають через помилки співробітників. Тому в правилах варто передбачити:

Підписання NDA (угоди про нерозголошення) при наймі та звільненні.

Чіткі обмеження – доступ до даних лише за необхідності.

Заборона несанкціонованого використання інформації.

Регулярні інструктажі з кібербезпеки.

Проактивність – найкращий захист. Політика безпеки має бути не формальністю, а робочим інструментом, який регулярно перевіряється та оновлюється. Інакше будь-який інцидент може перерости в кризу.

Ефективний захист від витоку даних та кіберзагроз: стратегії для бізнесу

1. Навчання персоналу – основа безпеки

Підписання NDA (угод про нерозголошення) – лише перший крок. Для повноцінного захисту даних необхідні:

Регулярні тренінги з інформаційної безпеки;

Чіткі внутрішні регламенти щодо обробки конфіденційних даних;

Практичні кейси – імітація атак та алгоритми дій при витоках.

Без системної освітньої роботи співробітники часто недооцінюють ризики, що робить компанію вразливою.

2. Кібербезпека: чому економити – невигідно

Сьогодні 47% підприємств стикаються з кібератаками щороку. Основні помилки бізнесів:

Повна відсутність інвестицій у захист – "нас це не стосується";

Неправильний розподіл бюджету – купівля рішень "на всі випадки" без аналізу реальних потреб;

Формальний підхід – дороге ПЗ не використовується на повну.

Наслідки: фінансові втрати, зупинка виробництва, падіння довіри клієнтів.

3. Як інвестувати розумно?

Оцінка ризиків – аналіз слабких місць IT-інфраструктури.

Пріоритети – захист найважливіших активів (бази даних, фінансові системи).

Гнучкість – регулярне оновлення захисту з урахуванням нових загроз.

Просунуті технології – VPN, двофакторна аутентифікація, системи виявлення вторгнень.

Приклад: Після атаки на одну з українських корпорацій її збитки склали $2,3 млн – сума, яку можна було запобігти.

4. Ключові переваги інвестицій у кібербезпеку

Захист даних – уникнення штрафів за GDPR та інші нормативи.

Стабільність бізнесу – мінімізація простоїв через атаки.

Довіра клієнтів – 81% споживачів уникають компаній, які втрачали їхні дані.

Конкурентна перевага – демонстрація відповідального підходу до безпеки.

Кібербезпека – не витрата, а інвестиція в майбутнє компанії. На відміну від фінансових криз, кіберзагрози можна прогнозувати та запобігати. Головне – діяти на випередження.

4. Контроль репутації – аналіз відгуків, коментарів і публікацій про компанію. У цифрову епоху думка клієнтів формує бренд навіть сильніше, ніж маркетингові кампанії. Один негативний відгук може відштовхнути десятки потенційних покупців, тоді як щирі позитивні коментарі стають потужним драйвером продажів. Ось чому системний моніторинг і аналіз зворотного зв'язку – не просто корисна практика, а стратегічна необхідність для бізнесу.

Як відстеження відгуків зміцнює ваш бренд?

Швидке вирішення проблем

– Виявлення скарг у реальному часі дозволяє усунути недоліки до того, як вони переростуть у публічний скандал.

– Приклад: клієнт залишив скаргу на затримку доставки – компенсація або персональний виклик менеджера знизить ризик втрати лояльності.

Довіра через прозорість

– Відкрита комунікація в соцмережах або на сайтах-вітринах (наприклад, Google Reviews чи Rozetka) показує: ви дійсно дбаєте про клієнтів.

– Дослідження показують: 89% покупців перевіряють відгуки перед рішенням про покупку.

Безкоштовні маркетингові інсайти

– Відгуки – це готовий аудіт потреб ринку. Наприклад:

Часті скарги на складність оформлення замовлення? Час спростити інтерфейс сайту.

Постійна згадка про «довге очікування»? Оптимізуйте логістику.

Покращення сервісу

– Аналіз коментарів допомагає визначити «вузькі місця» у роботі персоналу чи технічних процесах.

– Кейс: мережа ресторанів після хвилі негативу про холодні страви запровадила термопакування – і кількість скарг впала на 40%.

Створення продуктів, які хоче клієнт

– Відгуки часто містять ідеї для нових функцій чи послуг.

– Приклад: бренд взуття додав лінійку з підвищеною амортизацією після численних запитів у відгуках.

Ефективніший маркетинг

– Тональність відгуків показує, які переваги бренду клієнти цінують найбільше (наприклад, «швидка доставка» або «зручний обмін»). Це допомагає точніше налаштувати рекламні меседжі.

Лояльні клієнти = безкоштовні амбасадори

– Активні прихильники, які залишають позитивні коментарі, – ваш найкращий рекламний канал. Заохочення таких клієнтів (наприклад, персональними знижками) збільшує їхню ентузіастичність.

Кризове управління

– Публічна реакція на негатив (наприклад: «Шановний Іван, вибачте за незручності! Ми вже вирішуємо вашу проблему») знижує шкоду для іміджу.

– Важливо: 34% користувачів видаляють негативний відгук, якщо компанія допомогла його вирішити.

Конкурентна перевага

– Компанії, які аналізують відгуки, на 25% швидше адаптуються до змін ринку (дані HubSpot).

– Приклад: бізнес, який відстежував коментарі про «відсутність екопакувань», запустив «зелену» лінію – і переманив клієнтів у конкурентів.

Відгуки = Ваш GPS для успіху

Ігнорування зворотного зв’язку – це рулетка з репутацією і прибутком. Натомість проактивний моніторинг перетворює кожен коментар (навіть негативний) на можливість:

Зростання лояльності – клієнти бачать, що їхня думка важлива.

Оптимізація витрат – ви виправляєте саме те, що дратує клієнтів, а не вкладаєте гроші в «вгадування».

Стійкість до криз – навчений персонал і налаштовані процеси мінімізують ризики.

Порада: Використовуйте спеціалізовані інструменти (наприклад, Brand24, Awario або Hootsuite) для автоматизації моніторингу. Так ви не пропустите жодного важливого відгуку!

Висновок:

OSINT — це метод збору та аналізу інформації з відкритих джерел, який допомагає бізнесу краще розуміти ринок, конкурентів і потенційні ризики. Він дозволяє отримувати дані з соцмереж, форумів, державних реєстрів, фінансових звітів і технічної документації. Для обробки цієї інформації використовують спеціальні програми, штучний інтелект і візуалізацію даних, щоб знаходити корисні зв’язки і закономірності.

OSINT особливо корисний у трьох сферах. По-перше, він допомагає бізнесу аналізувати дії конкурентів, знаходити їх слабкі сторони і приймати обґрунтовані рішення. По-друге, він використовується в кібербезпеці для раннього виявлення загроз. По-третє, він дозволяє моніторити репутацію компанії в інтернеті і швидко реагувати на негатив.

Важливо застосовувати OSINT відповідно до закону і етичних норм, а також поєднувати автоматизовані інструменти з професійним аналізом людей. У результаті, OSINT — це не просто збір даних, а стратегічний інструмент, який дає бізнесу переваги в умовах жорсткої конкуренції та цифрових загроз.

РОЗДІЛ ІІ. ОГЛЯД ТЕХНОЛОГІЙ ТА СЕРЕДОВИЩА OSINT

2.1 Проблема вибору систем моніторингу

Сьогодні один необдуманий твіт або коментар може зруйнувати бізнес або кар'єру. Репутація – це як гроші: її можна накопичувати, втрачати, відновлювати і навіть інвестувати в неї. І вона безпосередньо впливає на ваш дохід.

Чому це так важливо?

50% людей не куплять товар, побачивши про нього погані відгуки.

69% відмовляться працювати в компанії з поганою репутацією.

54% керівників кажуть: чим менше негативу в інтернеті – тим вищі прибутки.

Як інтернет впливає на репутацію?

Раніше новини поширювалися повільно – через газети чи телебачення. Зараз все інакше:

Пошукові системи (Google) показують і хороше, і погане про вас мільйонам людей.

Соцмережі посилюють будь-який скандал – навіть старий твіт може стати причиною бойкоту (як у Джоан Роулінг).

Колишні співробітники або клієнти можуть опублікувати скаргу – і вона стане вірусною (як у випадку з токсичною атмосферою на шоу Еллен Дедженерес).

Що руйнує репутацію?

Ваші власні дії

Необдумані пости, лайки, коментарі.

Погана робота з клієнтами (наприклад, історія з «Книгарнею Є»).

Дії співробітників

Випадкові витоки інформації.

Грубість у соцмережах.

Зовнішні фактори

Скарги клієнтів у відгуках.

Публікації ЗМІ (як у випадку з BuzzFeed про шоу Еллен).

Що робити після кризи?

Моніторити реакції (наприклад, через Google Alerts або спеціальні сервіси).

Аналізувати, чи згадують проблему – якщо так, продовжувати працювати з репутацією.

Перевіряти, чи не постраждали інші бренди компанії (якщо є кілька).

Репутація – це не просто «образ у головах». Вона впливає на продажі, найм співробітників і довіру клієнтів. Тому слідкуйте за тим, що пишуть про вас, і реагуйте швидко – інакше наслідки можуть коштувати мільйонів.

До чого це ми ? Ах так. До систем конкурнентної розвідки. Тоді постає ще одне питання. Що таке SERM і чому нам це потрібно?

SERM (Search Engine Reputation Management) - це керування репутацією в інтернет-пошуку.

Рис. 1. Search Engine Reputation Management

Як це працює? Ми використовуємо спеціальні методи, щоб формувати позитивне враження про бренд, компанію або окрему особу в очах потенційних клієнтів. Це включає:

Створення гарного іміджу "з чистого аркуша"

Виправлення вже існуючого негативу в мережі

Простими словами:

Це налаштування пошукових систем так, щоб коли люди шукають про вас - вони бачили переважно хороші відгуки та корисну інформацію.

Чому це важливо? Тому що 9 з 10 людей перевіряють інформацію в інтернеті перед тим, як щось купити або співпрацювати з компанією. SERM допомагає керувати тим, що вони побачать.

Простими словами про SERM (управління репутацією в пошуку)

SERM – це спосіб "прибрати" погані відгуки з перших місць у Google та Яндексі. Повністю видалити негатив неможливо, але можна:

-Знизити його вплив – "закрити" хорошими новинами про компанію

-Створити позитивний образ – щоб люди бачили переважно гарні відгуки

Як це працює?

Аналізують, що зараз показує пошук (новини, відгуки, коментарі).

Створюють корисний контент – статті, офіційні сайти, соцмережі, відео.

Просувають його в топ – щоб замість скарг люди бачили правдиву позитивну інформацію.

Приклад:

Якщо в пошуку "Назва_компанії" на першому місці – скарга, SERM допомагає підняти вище офіційний сайт або добрий огляд.

Чому це потрібно?

9 з 10 клієнтів шукають інфо про бренд перед покупкою

1 поганий відгук у топі може відлякати 50% клієнтів

Конкуренти можуть навмисно псувати репутацію

Чим SERM відрізняється від ORM?

ORM – керує репутацією скрізь (форуми, соцмережі, ЗМІ)

SERM – працює тільки з пошуком (Google, Яндекс)

Найкращий результат – коли використовують обидва підходи разом.

Хто це використовує?

Бізнеси (від малого до великого)

Магазини

Відомі люди

Головне: SERM – це не разова акція, а постійна робота. Бо новий негатив може з'явитися будь-коли.

Оцінка результатів SERM

Після впровадження стратегії важливо періодично перевіряти, чи працює вона.

Як це робиться?

Аналіз пошукової видачі

Чи змінилися позиції ключових матеріалів?

Чи з'явилися нові позитивні/негативні результати?

Реакція аудиторії

Чи зросла довіра до бренду?

Чи зменшилася кількість скарг?

Рішення на основі даних

Якщо все добре → продовжуємо ту саму стратегію.

Якщо немає ефекту або стало гірше → змінюємо підхід (наприклад, більше контенту, інші ключові слова тощо).

Чому це важливо?

Пошукові системи постійно оновлюються → те, що працювало раніше, може втратити ефективність.

Конкуренти також працюють над SERM → потрібно бути на крок попереду.

Висновок: SERM – це не "зробив і забув", а постійний процес контролю та коригування..

2.2 Розвідка відкритих джерел OSINT

Open Source Intelligence (OSINT): сучасний інструмент аналітики та безпеки

OSINT (Open Source Intelligence) являє собою систематизований процес збору, аналізу та використання інформації з публічних джерел для підтримки прийняття рішень у сферах національної безпеки, бізнесу та громадської діяльності. На відміну від традиційних розвідувальних методів, OSINT ґрунтується виключно на легальних джерелах інформації, що робить його важливим інструментом у сучасному інформаційному просторі.

Основні характеристики OSINT:

- Легальність: використовує тільки публічно доступні джерела
- Універсальність: застосовується у військовій, політичній та бізнес-сферах
- Комплементарність: ефективно доповнює інші види розвідки (HUMINT, SIGINT, GEOINT)

Рис. 2. Етапи OSINT

У бізнес-середовищі OSINT тісно пов'язаний з конкурентною розвідкою (Competitive Intelligence), але принципово відрізняється від промислового шпигунства. Його основна цінність полягає в здатності перетворювати "інформаційну руду" публічних джерел у цінні аналітичні висновки ("самородки знань").

Технологічний розвиток OSINT:

Сучасні інструменти OSINT включають:

- Системи Business Intelligence для аналізу даних
- Text Mining та Knowledge Management Systems
- Social Media Analytics для моніторингу соціальних мереж
- Машинне навчання та Sentiment Analysis

Міжнародний досвід, зокрема стандарти НАТО та практика США, демонструють, що до 80-90% корисних розвідувальних даних отримують саме з відкритих джерел. Це підтверджують дослідження ЦРУ, які показують зростаючу роль OSINT у вирішенні таких завдань, як боротьба з тероризмом або контррозвідка.

В Україні розвиток OSINT-технологій проходить у декількох напрямках:

1. Академічні дослідження (роботи професора Д.Ф. Ланде)
2. Комерційні рішення (наприклад, платформа SemanticForce)
3. Військові програми (підготовка фахівців у військових інститутах)

Сучасні OSINT-системи еволюціонували до складних мережевих організмів, здатних не лише збирати інформацію, але й взаємодіяти з користувачами через чат-боти та інші інтерфейси. Це відкриває нові можливості, але й створює нові виклики, особливо у контексті гібридних загроз.

Таким чином, OSINT став невід'ємною частиною сучасних систем безпеки та прийняття рішень, поєднуючи в собі методи аналітики, технології обробки даних та стратегічне планування. Його значення продовжує зростати у всіх сферах - від національної безпеки до бізнес-аналітики.

2.3 Джерела OSINT

Категорії джерел OSINT: класифікація та особливості

1. Традиційні ЗМІ

Включають друковані та електронні медіа:

- Газети та журнали (національні, місцеві, спеціалізовані)
- Телерадіомовлення (новини, ток-шоу, документальні програми)
- Інформаційні агенції

Перевага: висока достовірність, редакційний контроль

Недолік: можлива суб'єктивність подачі інформації

2. Цифрові джерела

Найбільш динамічна категорія:

- Соцмережі (Twitter, Facebook, Telegram)
- Блоги та форуми
- Відеоплатформи (YouTube, TikTok)
- Вікі-ресурси

Особливість: оперативність оновлення, висока частка UGC (контенту користувачів)

3. Державні відкриті дані

Офіційна інформація публічного характеру:

- Законодавчі акти та нормативні документи
- Бюджетні звіти та статистика
- Транскрипти публічних виступів
- Реєстри та державні реєстрації

Ключова перевага: юридична значимість інформації

4. Наукові публікації

Академічні джерела знань:

- Рецензовані журнали
- Матеріали конференцій
- Дисертаційні дослідження
- Патентні бази

Цінність: глибина аналізу, експертна оцінка

5. Комерційна інформація

Дані бізнес-сектору:

- Фінансові звіти компаній
- Маркетингові дослідження
- Бази даних (наприклад, OpenCorporates)
- Супутникові знімки комерційного походження

6. Сіра література

Неформальні джерела:

- Технічна документація
- Внутрішні звіти організацій
- Препринти наукових робіт
- Інформаційні бюлетені

Особливість: часто містить унікальні дані, відсутні в офіційних джерелах

7. Технічні джерела

Специфічні методи збору:

- Радіомоніторинг (наприклад, ADS-B для авіації)
- Супутникові знімки (Google Earth, Sentinel Hub)
- Веб-камери онлайн-трансляцій
- Дані IoT-пристроїв

Кожна категорія джерел має унікальні характеристики, що визначають їх використання в різних сценаріях OSINT-досліджень. Оптимальна стратегія передбачає комбінування джерел з урахуванням:

- Темпи оновлення інформації
- Рівень верифікації даних
- Профільність контенту
- Юридичні обмеження доступу

Сучасні OSINT-спеціалісти повинні володіти навичками роботи з усіма типами джерел, враховуючи їх синергетичний потенціал.

Промислове шпигунство , також відоме як економічне шпигунство , корпоративне шпигунство або корпоративне шпигунство , — це форма шпигунства , що здійснюється з комерційною метою, а не виключно для забезпечення національної безпеки.

У той час як політичне шпигунство здійснюється або організовується урядами та має міжнародний масштаб, промислове або корпоративне шпигунство частіше є національним і відбувається між компаніями або корпораціями.

Коротко кажучи, метою шпигунства є збір знань про одну або кілька організацій. Економічне або промислове шпигунство відбувається у двох основних формах. Воно може включати придбання інтелектуальної власності , такої як інформація про промислове виробництво, ідеї, методи та процеси, рецепти та формули. Або ж воно може включати конфіскацію службової або операційної інформації , такої як дані про клієнтів , ціноутворення , продажі , маркетинг , дослідження та розробки , політику, потенційні пропозиції , планування чи маркетингові стратегії , або зміну складу та місць виробництва. Воно може описувати таку діяльність, як крадіжка комерційної таємниці , хабарництво , шантаж та технологічне спостереження . Окрім організації шпигунства за комерційними організаціями , уряди також можуть бути цілями – наприклад, для визначення умов тендеру на державний контракт.

Інформація може бути вирішальним фактором між успіхом і невдачею; якщо комерційну таємницю викрадено, конкурентне поле вирівнюється або навіть перехиляється на користь конкурента. Хоча значна частина збору інформації здійснюється легально за допомогою конкурентної розвідки , часом корпорації вважають, що найкращий спосіб отримати інформацію – це взяти її під контроль. Економічне або промислове шпигунство становить загрозу для будь-якого бізнесу, чиє існування залежить від інформації.

В останні роки економічне або промислове шпигунство набуло розширеного визначення. Наприклад, спроби саботажу корпорації можна вважати промисловим шпигунством; у цьому сенсі термін набуває ширших конотацій свого батьківського слова. Те, що шпигунство та саботаж (корпоративний чи інший) стали більш чітко пов'язані одне з одним, також демонструється низкою профільних досліджень, деякі урядові, деякі корпоративні. Уряд Сполучених Штатів наразі проводить перевірку на поліграфі під назвою «Тест на шпигунство та саботаж» (TES), що сприяє уявленню про взаємозв'язок між контрзаходами шпигунства та саботажу. На практиці, особливо «довіреними інсайдерами», вони зазвичай вважаються функціонально ідентичними з метою обґрунтування контрзаходів .

2.4 Конкурентна розвідка

Конкурентна розвідка (КР) – це процес і перспективні практики, що використовуються для отримання знань про конкурентне середовище з метою покращення ефективності організації. Конкурентна розвідка включає систематичний збір та аналіз інформації з різних джерел та скоординовану програму конкурентної розвідки. Це дія з визначення, збору, аналізу та розповсюдження розвідувальних даних про продукти, клієнтів, конкурентів та будь-який аспект середовища, необхідний для підтримки керівників та менеджерів у прийнятті стратегічних рішень для організації.

Неперервна інтеграція (БІ) означає розуміння та вивчення того, що відбувається у світі поза бізнесом, для підвищення власної конкурентоспроможності. Це означає якомога більше та якомога швидше дізнатися про зовнішнє середовище, включаючи галузь загалом та відповідних конкурентів.

Ключові моменти:

Конкурентна розвідка — це законна ділова практика, на відміну від промислового шпигунства , яке є незаконним.

Основна увага приділяється зовнішньому бізнес-середовищу.

Існує певний процес збору інформації, її перетворення на розвідувальні дані та подальшого використання для прийняття рішень. Деякі фахівці з комунікаційної інтеграції помилково наголошують, що якщо зібрані розвідувальні дані не є корисними або не підлягають застосуванню, то вони не є розвідувальними.

Інше визначення конкурентної розвідки розглядає її як організаційну функцію, відповідальну за раннє виявлення ризиків та можливостей на ринку, перш ніж вони стануть очевидними («ранній аналіз сигналів»). Це визначення зосереджує увагу на різниці між поширенням широкодоступної фактичної інформації (наприклад, ринкової статистики , фінансових звітів , газетних вирізок), що здійснюється такими функціями, як бібліотеки та інформаційні центри, та конкурентною розвідкою, яка є поглядом на розвиток та події, спрямовані на отримання конкурентної переваги.

Термін «конкурентна розвідка» часто розглядається як синонім аналізу конкурентів , але конкурентна розвідка — це більше, ніж просто аналіз конкурентів; вона охоплює все середовище та зацікавлені сторони: клієнтів, конкурентів, дистриб'юторів, технології та макроекономічні дані. Вона також є інструментом для прийняття рішень.

Етика давно є предметом обговорення серед практиків конкурентної розвідки. Питання обертаються навколо того, що є, а що ні, з точки зору діяльності конкурентної розвідки. З цієї теми було створено кілька наукових

робіт, найбільш помітними з яких є публікації Стратегічного консорціуму фахівців з розвідки. Книга «Етика конкурентної розвідки: навігація по сірій зоні» містить майже двадцять окремих поглядів на етику в конкурентній розвідці, а також ще 10 кодексів, що використовуються різними особами або організаціями. Поєднуючи це з більш ніж двома десятками наукових статей або досліджень, знайдених у різних бібліографічних записах конкурентної розвідки, , стає зрозуміло, що не бракує досліджень, спрямованих на кращу класифікацію, розуміння та вирішення питань етики конкурентної розвідки.

Конкурентну інформацію можна отримати з публічних або передплатних джерел, шляхом спілкування з персоналом або клієнтами конкурентів, розбирання продуктів конкурентів або з польових дослідницьких інтерв'ю. Дослідження конкурентної розвідки відрізняються від промислового шпигунства , оскільки фахівці з конкурентної розвідки зазвичай дотримуються місцевих правових норм та етичних норм ведення бізнесу.

Аутсорсинг став великим бізнесом для фахівців з конкурентної розвідки. У цій галузі існує багато різних компаній, включаючи фірми з дослідження ринку та консалтингу.

Висновок:

Репутація – ключовий актив для бізнесу та кар'єри. Один негативний публічний матеріал може завдати значної шкоди. Управління репутацією включає два основні напрямки: ORM (контроль інтернет-контенту в соцмережах, форумах, ЗМІ) та SERM (оптимізація пошукової видачі в Google та Яндекс). Загрози репутації поділяються на внутрішні (необдумані дії співробітників, витоки інформації) та зовнішні (дії клієнтів, конкурентів, колишніх працівників). Ефективний захист передбачає моніторинг згадок, витіснення негативу позитивним контентом та оперативне реагування на кризи.

OSINT (розвідка відкритих джерел) – легальний інструмент збору даних з інтернету, соцмереж і держреєстрів. Він використовується для конкурентного аналізу, кібербезпеки та репутаційного менеджменту, на відміну від промислового шпигунства, яке є протизаконним.

РОЗДІЛ ІІІ. РОЗРОБКА ТЕЛЕГРАМ-БОТУ ДЛЯ OSINT РОЗВІДКИ

3.1 Telegram-бот як засіб автоматизації збору відкритих даних (OSINT)

В сучасних умовах цифровізації OSINT (Open Source Intelligence) став невід'ємною частиною стратегічного аналізу, кібербезпеки та управління репутацією. Як зазначається у висновках дослідження, OSINT дозволяє легально отримувати критично важливі дані з відкритих джерел: соціальних мереж, форумів, державних реєстрів тощо. Однак ручний збір та аналіз такої інформації є трудомістким, що обумовлює потребу в автоматизованих рішеннях

В нашій роботі ми презентуємо розробку Telegram-бота, який спростить OSINT-розвідку для бізнесу, маркетологів та ймовірних фахівців з кібербезпеки. Дозволить автоматизувати пошук публічно доступних даних за номерами телефонів, нікнеймами та email. А також спробуємо інтегрувати інструменти моніторингу репутації (SERM/ORM), що були згадані у попередньому розділі.

Цільовою аудиторіє нашого чат-боту мають стати фахівці з бізнес-аналітики (конкурентна розвідка), та PR-менеджери, які займаються конкурентним моніторингом.

От же розглянемо повний цикл реалізації Telegram-бота для OSINT-розвідки починаючи від визначення заданих цілей та архітектури до функціональних вимог самих етапів практичного впровадження. Запропоноване нами рішення базується на сучасному етапі технологічних мові програмування типу:

- Python з використанням бібліотек python-telegram-bot, requests, BeautifulSoup
- API: Telegram Bot API, із використанням telethon
- База даних: SQLite для логування запитів
- Візуалізація: matplotlib для аналізу зв'язків між знайденими даними.

```python
from telegram import Update
from telegram.ext import Updater, CommandHandler

def start(update: Update, context):
    update.message.reply_text(
        "🔍 Цей бот допомагає в OSINT-розвідці. Доступні команди:\n"
        "/search_phone — пошук за номером\n"
        "/search_username — пошук у соцмережах"
    )

updater = Updater("YOUR_BOT_TOKEN")
updater.dispatcher.add_handler(CommandHandler("start", start))
updater.start_polling()
```

Рис. 3. Search Engine Reputation Management

Також буде використано aiogram – який єявляє собою сучасний та повністю асинхронний фреймворк для Telegram Bot API, написаний на Python з використанням asyncio та aiohttp.

Рис. 4. Aiogram

Веб-скрейпінг (або парсинг даних) - це процес автоматизованого збору та структурування інформації з вебсторінок, які зазвичай призначені для перегляду людьми через браузер. Ця технологія реалізується за допомогою спеціальних програмних рішень, які можуть працювати у двох основних режимах: імітувати дії користувача в браузері або взаємодіяти з сервером напряму через HTTP-запити.

Сучасні інструменти веб-скрейпінгу включають:

- Спеціалізовані бібліотеки для різних мов програмування

- Системи автоматизованого керування браузерами

- Рішення для обробки та зберігання отриманих даних

Важливо відзначити, що веб-скрейпінг може виконуватись як повністю автоматизованими системами, так і з участю людини, коли оператор вручну копіює та систематизує дані для подальшого аналізу. Отримана інформація зазвичай зберігається у структурованому вигляді (бази даних, електронні таблиці) для подальшого використання.

Технічно скрейпінг відрізняється від стандартних методів отримання даних через API тим, що працює з інтерфейсом, призначеним для людей, а не машин. Це створює певні технічні складнощі, оскільки вебсторінки часто змінюють свою структуру, але водночас відкриває можливості для роботи з сайтами, які не надають офіційних API.

Рис. 5. Вебскрейпінг

Важливим аспектом нашої роботи є застосування синтаксичного аналізу (парсингу) - процесу, який перетворює неструктуровані або слабоструктуровані вхідні дані (переважно текстові) у чітко організовані ієрархічні структури. Синтаксичний аналізатор як програмний компонент виконує кілька ключових функцій: перевіряє коректність структури вхідних даних, виявляє синтаксичні помилки та формує деревоподібне представлення інформації (абстрактне синтаксичне дерево або інші форми структурованого подання).

Технічна реалізація парсингу передбачає різні підходи:

1. Використання окремого лексичного аналізатора (лексеру), який попередньо обробляє вхідний текст, розбиваючи його на токени

2. Застосування комбінованих рішень, де лексичний і синтаксичний аналіз інтегровані в єдиний процес

3. Використання генераторів парсерів для автоматизації створення аналізаторів

У практичній площині синтаксичний аналіз знаходить застосування у різноманітних контекстах:

- У парі з генераторами виводу (наприклад, scanf/printf у мові C)

- Як критично важливий компонент компіляторів (як на етапі аналізу вхідного коду, так і при генерації результуючого коду)

- У системах обробки природної мови

- При аналізі структурованих текстових даних (наприклад, HTML, XML)

У нашій роботі синтаксичний аналіз виступає ключовою ланкою в ланцюжку обробки даних, забезпечуючи перехід від неформалізованого представлення інформації до структурованих даних, придатних для подальшого аналізу та інтерпретації. Особливу важливість цей процес набуває при роботі з гетерогенними джерелами інформації, де потрібно виявляти та узгоджувати різні формати представлення даних.

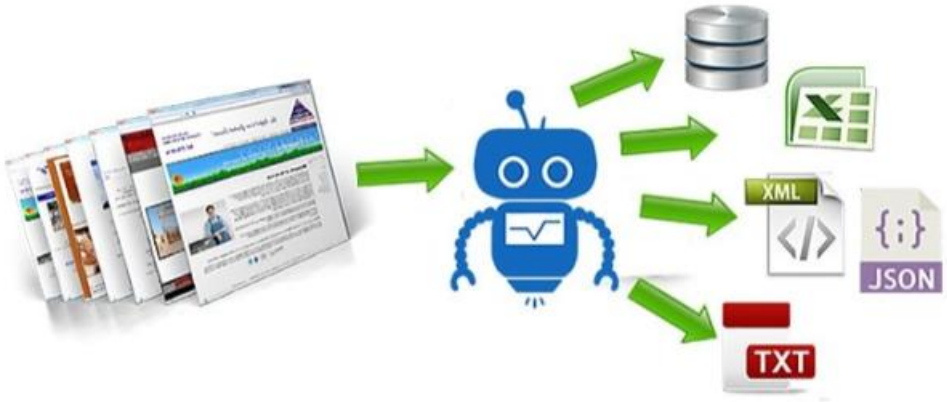

Рис. 6. Парсинг

Наш бот матиме модульну структуру, а саме користувацький інтерфейс (Telegram-клавіатури, команди), обробник запитів (парсинг введених даних: номер, email, нікнейм), та OSINT-ядро (інтеграція з API соцмереж, парсинг відкритих даних) представлену на рис.6.

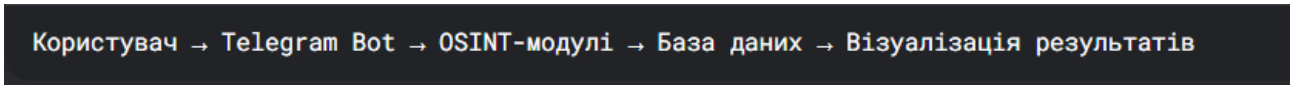

Рис. 6. Схема взаємодії

Сформулюємо вимоги до Telegram-бот OSINT-розвідки, що розроблявся з урахуванням такого ключового функціоналу, а саме- зручний користувацький інтерфейс що доступний і популярний серед молоді месенджер- Telegram. В ньому повинна бути інтерактивна клавіатура для вибору типу запиту (ЄДРПОУ / назва компанії), що буде формувати користувач. Окрім того необхідна інтеграція з інструментами вебскрапінгу та Google Dork. Важливим аспектом в пошуку даних є соціальні мережі та найголовніше для клієнта це формування стислого та структурованого OSINT-звіту.

Врахувавщи всі вище перераховані побажання опишемо архітектуру самої системи Telegram-бота, яка включатиме такі основні компоненти:

- Front-end: Telegram як клієнтська частина (через бот API)
- Back-end: FastAPI для REST-запитів, aiogram для Telegram логіки
- База даних (опціонально): SQLite або PostgreSQL для логування запитів
- Парсери та пошукові модулі: Використання requests, BeautifulSoup, Selenium, fake_useragent, DuckDuckGo, проксі-серверів

3.2 Розробка Telegram-бота "OSINT Project bot"

Опишемо структурну логіка взаємодії команд чат-бота. Після активації чат-бота ми переходимо в розділ де нам доступна наступна команда /start . Після її активації бот надає користувачеві інтерфейс із кнопками:

- Пошук за ЄДРПОУ
- Пошук за назвою компанії

Обраний запит обробляється асинхронно через FastAPI. Бот здійснює серію дій:

- Перевірка формату вводу: Валідація коду ЄДРПОУ або назви компанії.
- Пошук інформації: Здійснюється серія Dork-запитів через DuckDuckGo/Google, скрапінг державних реєстрів (Clarity, YouControl, Opendatabot).
- Парсинг результатів: Дані очищуються, агрегуються й структуровано подаються у відповідь користувачу.
- Виведення звіту: Telegram повідомлення містить основну інформацію, контактні дані, URL на реєстри, а також гіперпосилання на знайдені згадки у ЗМІ чи соцмережах.

Для такої складної роботи , нам потрібно було використати вже існуючі системи пошуку. Ми зупинили свій погляд на DuckDuckGo — пошукову систему з відкритим початковим кодом, що використовує інформацію з багатьох джерел для надання точних та різноманітних результатів.

Рис. 7. Duck Duck Go, Inc.

DuckDuckGo обрано як основну пошукову систему з наступних міркувань:

- Конфіденційність: на відміну від Google, DuckDuckGo не зберігає історії пошуку, не використовує трекери, не пов'язує запити з IP-адресами користувача, що є критичним у контексті OSINT.
- Відкритість HTML-інтерфейсу у версії https://html.duckduckgo.com/html, яка генерує просту HTML-сторінку без JavaScript, спрощує завдання парсингу.
- Швидкість і стабільність: пошукова система надає результати без складних редіректів чи динамічних блоків.

Таким чином, DuckDuckGo виступає як шлюз до великої кількості відкритих джерел, таких як офіційні державні реєстри (Clarity-project, YouControl, Opendatabot), профілі у соціальних мережах, згадки у ЗМІ, новини, тендери, судові рішення тощо.

Важливим аспектом є і створення архітектура парсингового модуля, який є ключовим елементом Telegram-бота OSINT, що відповідає за пошук інформації через веб, зокрема через пошукову систему DuckDuckGo. Для ефективного та надійного збору даних використовується набір сучасних бібліотек і методів, які дозволяють обходити типові обмеження вебскрапінгу, такі як блокування IP, обмеження за User-Agent, та складна структура вебсторінок.

Основні функції парсингового модуля реалізовано у вигляді окремого файлу duckduckgo_search.py, який включає:

- Формування коректного пошукового запиту з кодуванням символів.
- Використання кастомних HTTP-заголовків (User-Agent), щоб імітувати справжній браузер.
- Підключення до проксі для обходу обмежень по IP.
- Отримання HTML-сторінки результатів пошуку.
- Парсинг сторінки за допомогою BeautifulSoup для вилучення корисної інформації (заголовки, посилання, опис).

Файл duckduckgo_search.py

```
import time
import random
from bs4 import BeautifulSoup
from fake_useragent import UserAgent
import requests
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

PROXY_FILE = r"Telegram_bot\app\Webshare 10 proxies.txt"

def load_proxies(file_path):
    with open(file_path, "r") as f:
        return [line.strip() for line in f.readlines() if line.strip()]
def get_proxy_dict(proxy_string):
```

```
    ip, port, user, password = proxy_string.split(":")
    proxy_auth = f"http://{user}:{password}@{ip}:{port}"
    return {"http": proxy_auth, "https": proxy_auth}
def get_headers():
    return {"User-Agent": UserAgent().random}
def search_duckduckgo(query, proxy_list=None, use_selenium=False):
    if use_selenium:
        options = Options()
        options.add_argument("--start-maximized")
        driver = webdriver.Chrome(options=options)
        driver.get("https://html.duckduckgo.com/html/")
        time.sleep(2)
        search_input = driver.find_element(By.NAME, "q")
        search_input.clear()
        search_input.send_keys(query)
       search_input.submit()
        time.sleep(5)
        results = driver.find_elements(By.CSS_SELECTOR, ".result__url")
        links = [r.get_attribute("href") for r in results]
        time.sleep(3)
        driver.quit()
        return links
    else:
        if proxy_list is None or len(proxy_list) == 0:
            proxies = None
        else:
            proxy_string = random.choice(proxy_list)
            proxies = get_proxy_dict(proxy_string)

        headers = get_headers()
        url = "https://html.duckduckgo.com/html/"
        params = {"q": query}

        try:
            response = requests.post(url, data=params, headers=headers, proxies=proxies, timeout=20)
            response.raise_for_status()

            with open("duckduckgo_response.html", "w", encoding="utf-8") as f:
                f.write(response.text)

            soup = BeautifulSoup(response.text, "html.parser")
            results = soup.find_all("a", class_="result__a")
            links = [a["href"] for a in results if a["href"].startswith("http")]
            return links
        except Exception as e:
            print("❌ Помилка:", e)
            return []
```

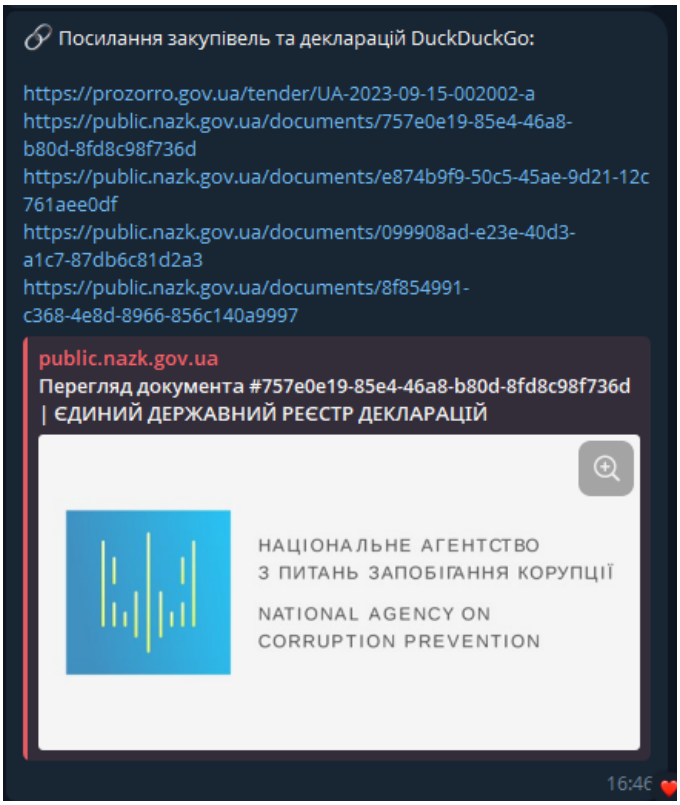

Рис. 7. Результати застосування модуля.

Отже, логіка дій нашого алгоритму є наступною:

- *Формування запиту*

  Функція приймає рядок пошукового запиту query. Для коректного вбудування його у URL використовується quote_plus, що кодує спеціальні символи у формат, сумісний з URL. Запит надсилається на адрес https://html.duckduckgo.com/html, що повертає просту HTML-сторінку без JavaScript, зручно для парсингу.

- *User-Agent*

  Для обходу блокування ботів сайтами використовується бібліотека fake_useragent, що генерує випадковий заголовок User-Agent, імітуючи різні браузери. Це знижує ймовірність блокування запитів як підозрілих.

- *Проксі*

Параметр proxy дозволяє передати адресу проксі-сервера. Це корисно при великій кількості запитів, щоб мінімізувати ризики блокування з боку DuckDuckGo або цільових ресурсів.

Файл osint_search.py виконує ключову функцію у структурі Telegram-бота OSINT: обробку запитів користувача, пошук і збір інформації з різних відкритих джерел, зокрема державних реєстрів і пошукової системи DuckDuckGo. Цей модуль організовано як набір взаємопов'язаних функцій, які забезпечують:

- Обробку запиту (назва компанії або код ЄДРПОУ).
- Парсинг локального XML-файлу з реєстром.
- Пошук і витяг інформації з зовнішнього сайту Clarity-project.
- Виконання пошуку через DuckDuckGo з підтримкою проксі.
- Формування результатів у структурованому вигляді для подальшої обробки ботом.

На наступному кроці нам необхідно провести обробку повідомлень користувача, відбувається з допомогою handlers.py. Під handler зазвичай мається на увазі обробник чогось (якихсь подій, вхідних з'єднань, повідомлень тощо). Це не є специфічним поняттям саме для Python.

```
from aiogram import F, Router
from aiogram.types import Message, ReplyKeyboardMarkup, KeyboardButton
from aiogram.filters import CommandStart
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup

from app.osint_search import run_osint_noapi, xml_search, company_name

router = Router()

class SearchState(StatesGroup):
    waiting_for_edrpou = State()
    waiting_for_name = State()

@router.message(CommandStart())
async def cmd_start(message: Message, state: FSMContext):
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="Пошук за ЄДРПОУ")],
            [KeyboardButton(text="Пошук за Назвою компанії")],
        ],
        resize_keyboard=True
    )
```

```
    await message.answer("👋 Вітаю! Оберіть тип пошуку:", reply_markup=keyboard)
    await state.clear()

@router.message(F.text == "Пошук за ЄДРПОУ")
async def choose_edrpou(message: Message, state: FSMContext):
    await message.answer("🔍 Введіть ЄДРПОУ для пошуку:")
    await state.set_state(SearchState.waiting_for_edrpou)

@router.message(F.text == "Пошук за Назвою компанії")
async def choose_name(message: Message, state: FSMContext):
    await message.answer("🔍 Введіть назву компанії для пошуку:")
    await state.set_state(SearchState.waiting_for_name)
```

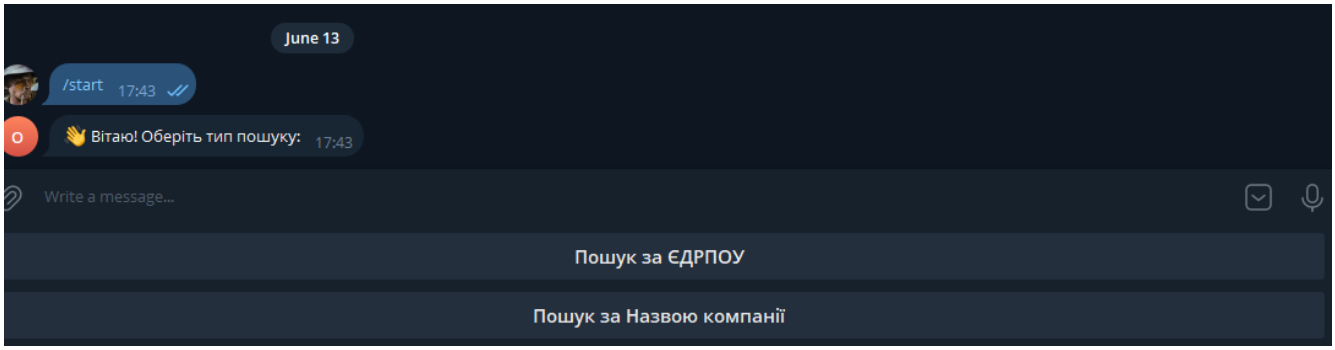

Рис. 8. Приклад головного вікна пошуку.

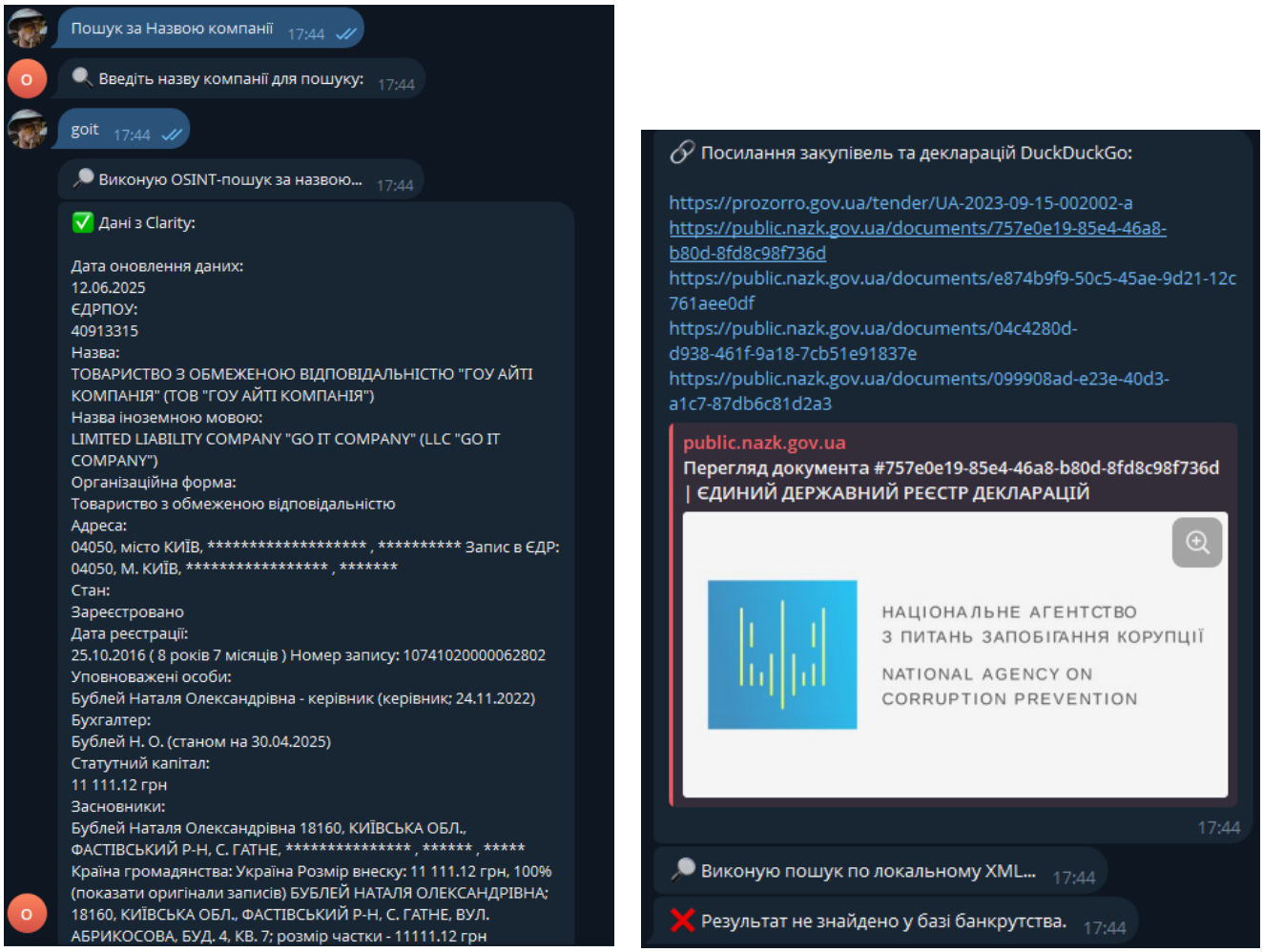

Рис. 9. Результати пошуку

3.3 Інтеграція Telegram-бота

Telegram-бот реалізований на базі асинхронного фреймворку Aiogram, що дозволяє одночасну обробку багатьох запитів без блокувань. Логіка пошуку за запитом користувача була інтегрована через функцію run_osint_noapi(query), яка асинхронно викликає duckduckgo_search() з переданим параметром.

Послідовність дій при виконанні запиту:

- Користувач надсилає команду /start або вибирає одну з кнопок клавіатури: "Пошук за ЄДРПОУ" чи "Пошук за Назвою компанії".
- Бот переходить у відповідний стан FSM (Finite State Machine) і очікує текстовий запит.
- Після введення запиту запускається функція run_osint_noapi(query).
- У разі помилки (відсутність результатів, блокування, таймаут) бот повертає інформативне повідомлення з інструкцією.

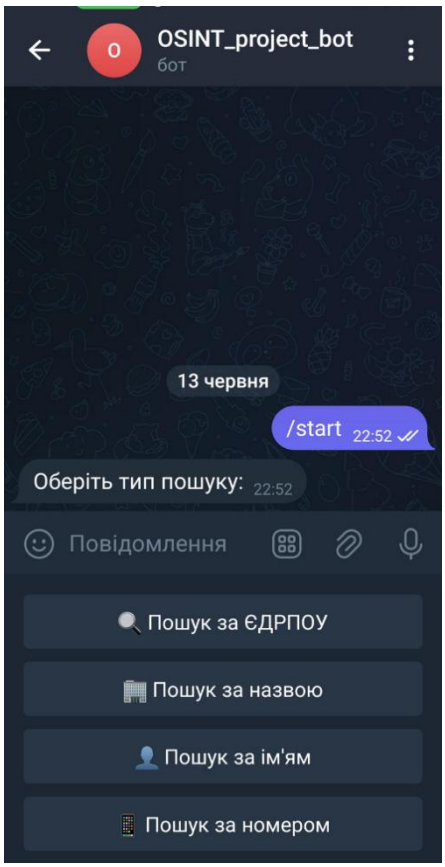

Рис. 10. Головне меню пошуку

Важливим етапом Інтеграції з OSINT-джерелами є розвідка у соціальних мережах. Вони є важливим джерелом інформації в контексті OSINT (Open Source Intelligence), оскільки вони дозволяють отримувати дані не лише про публічну активність фізичних осіб, а й про компанії, організації та їхніх представників. У рамках реалізованого Telegram-бота для OSINT-розвідки був створений окремий модуль для здійснення пошуку в соціальних мережах, орієнтований на виявлення цифрового сліду контрагента, його зв'язків, репутації та непрямих згадок.

Основна мета розвідки в соцмережах:

- Виявлення офіційних та неофіційних профілів компанії або її представників.
- Встановлення зв’язків між фізичними особами, що пов’язані з організацією.
- Отримання інформації про репутацію бізнесу в онлайн-середовищі.
- Виявлення потенційних ризиків: згадки в негативному контексті, участь у скандалах, спірних подіях тощо.

Платформи, що підлягають обробці:

Telegram-бот здійснює розвідку в таких соціальних мережах:

- Facebook — для пошуку сторінок компаній, профілів директорів, згадок у публікаціях та коментарях.
- LinkedIn — для ідентифікації бізнес-зв’язків, колишніх і нинішніх працівників, керівників та офіційної активності компанії.
- Twitter / X — для моніторингу публічної присутності компанії або ключових осіб, їх висловлювань, згадок бренду.
- Instagram — для візуальної ідентифікації бренду, публічної активності, заходів.
- YouTube — для виявлення корпоративних відео, коментарів, інтерв’ю, новинних сюжетів.
- TikTok — за наявності — для аналізу трендів, участі в акціях, репутаційних ризиків.

Головною методикою реалізації пошуку у межах нашого Telegram-бота є використання Google Dorking – техніки, яка дозволяє формувати комбіновані пошукові записи для знаходження публічної інформації з високою точністю. Цей метод широко використовується в OSINT-розвідці для виявлення даних, які не є очевидними при стандартному пошуку.

Принцип роботи Google Dorking у боті- це автоматизація створення спеціальних пошукових запитів, які включають ключові оператори Google (site:, intext:, filetype:, intitle: тощо) та логічні оператори (AND, OR, - для виключення термінів) в цільових джерелах (соцмережі, сайти компаній, форуми).

```
site:linkedin.com/in/ "Назва компанії" OR "ПІБ директора"
```

Рис. 11. Пошук профілів LinkedIn, пов’язаних із компанією

```
site:facebook.com "ТОВ Назва" OR "ПІБ керівника"
```

Рис. 12. Пошук публікацій на Facebook

```
filetype:pdf "Контактна інформація" AND "Назва компанії"
```

Рис. 13. Пошук документів (PDF, DOCX) з контактами

Як і в будь-яких системах, при пошуку даних не можливо пройти питання обмеження та обхідних шляхів для пошуку інформації.

1. Обмеження Google -Блокування при надто частих запитах (вирішується через проксі або API).

2. Капча для підозрілих запитів (потрібна інтеграція з сервісами типу 2captcha).

Варто сказати, що жодні юридичні аспекти під час пошуку інформації не були порушені. Нами отримуються лише публічно доступні дані (без порушення robots.txt). Протокол robots.txt є стандартизованим механізмом взаємодії між

веб-сайтами та автоматизованими системами сканування інтернет-простору, зокрема пошуковими роботами. Цей технічний стандарт визначає правила доступу роботів до різних частин веб-ресурсу, вказуючи, які розділи можуть бути просканованої, а які слід ігнорувати. Хоча він використовує директиви "Allow" (дозволити) та "Disallow" (заборонити), важливо розуміти, що цей протокол має суто рекомендаційний характер і не є обов'язковим до виконання.

Основне призначення robots.txt полягає в регулюванні роботи "доброякісних" роботів пошукових систем, таких як Googlebot або Bingbot, які використовують ці правила для коректної індексації сайту. Однак багато автоматизованих систем, зокрема спам-боти, збирачі електронних адрес, сканери вразливостей або інші зловмисні програми, свідомо ігнорують ці вказівки. Більше того, деякі шкідливі програми можуть використовувати інформацію з robots.txt для виявлення захищених або прихованих розділів сайту, до яких вони намагатимуться отримати доступ.

Важливо підкреслити, що robots.txt не слід розглядати як інструмент безпеки чи захисту даних. Це лише спосіб комунікації з пошуковими системами та іншими автоматизованими системами, які добровільно дотримуються цих правил. Для реального захисту конфіденційної інформації необхідні більш надійні методи, такі як автентифікація, шифрування або обмеження доступу на рівні сервера. Хоча robots.txt може використовуватись разом із Sitemaps для покращення індексації сайту, його функція обмежується лише вказівками для сумлінних веб-роботів і не забезпечує жодного технічного захисту від несанкціонованого доступу.

ВИСНОВОК

В даній кваліфікаційній роботі було описано методологію застосування OSINT-інструментів для проведення конкурентної розвідки, та реалізовано телеграм бот “OSINT Project bot” який автоматизує етапи OSINT-розвідки контрагентів для потреб бізнесу, журналістських розслідувань або аналітичної діяльності.

Реалізовано можливість пошуку за ЄДРПОУ та назвою компанії, що дозволяє отримати структуровані дані про юридичну особу, включно з її реєстраційною інформацією, місцезнаходженням, керівником, КВЕДами та станом діяльності. Інтеграція доступних відкритих джерел, таких як YouControl, Clarity Project або Opendatabot, забезпечила отримання офіційних, релевантних та верифікованих результатів.

Google Dork та DuckDuckGo-пошук були ефективно використані як ядро пошукової логіки бота. Формування специфічних Dork-запитів дозволило обійти обмеження традиційного інтерфейсу пошуку та здійснювати цільову фільтрацію інформації: за сайтами державних реєстрів, новинних видань, офіційних сторінок компаній.

HTML-парсинг за допомогою бібліотек requests, BeautifulSoup та fake_useragent забезпечив можливість автоматизованого витягування корисних фрагментів інформації зі знайдених сторінок.

Розвідка у соціальних мережах виявилась цінним джерелом додаткової інформації, яка часто не фігурує у публічних реєстрах, але може мати вирішальне значення при аналізі цифрового сліду контрагента. Пошук у Facebook, LinkedIn, Instagram, X (Twitter) дозволив ідентифікувати потенційні зв’язки, публічну активність осіб, згадки бренду або компанії, а також оцінити репутаційний фон.

Архітектура Telegram-бота виявилася зручною для взаємодії з користувачем: бот підтримує командну структуру, клавіатурні кнопки, зберігає сесії, формує звіти, що дозволяє швидко та інтерактивно здійснювати розвідку без потреби у спеціальних технічних навичках. Модульна структура на основі

aiogram та FastAPI дає змогу масштабувати систему та легко інтегрувати нові джерела або алгоритми.

Безпека та анонімність реалізовані шляхом використання проксі Webshare та заголовків User-Agent, що знижує ризик блокування під час багатократного автоматизованого доступу до веб-ресурсів.

Обмеження системи полягають у відсутності доступу до API деяких закритих сервісів, ризику фальшивих або застарілих даних у відкритих джерелах, а також потребі в постійній актуалізації пошукових шаблонів через зміну структури веб-сайтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Круковський, Ігор Анатолійович; Хомів, Богдан Арсенович; Гаврилюк, Всеволод Леонідович (2013). ІЄРАРХІЧНО-СИНЕРГЕТИЧНЕ ОБ'ЄДНАННЯ SOCIAL MEDIA ANALYTICS/SOCIAL CRM З BUSINESS INTELLIGENCE І З ГЕОГРАФІЧНОЮ ІНФОРМАЦІЙНОЮ СИСТЕМОЮ. Вісник Житомирського державного технологічного університету. Серія: Технічні науки (uk) 0 (1(64)). с. 60–69. ISSN 1728-4260. doi:10.26642/tn-2013-1(64)-60-69.
2. Запорожець, В., & Опірський, І. (2024). НЕБЕЗПЕКА ВИКОРИСТАННЯ TELEGRAM ТА ЙОГО ВПЛИВ НА УКРАЇНСЬКЕ СУСПІЛЬСТВО. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(25), 59–78. https://doi.org/10.28925/2663-4023.2024.25.5978
3. І.А. Круковський, В.Л. Гаврилюк, Б.А. Хомів Проблемні питання використання і розвитку засобів SocialMedia Analytics, їх інтеграції з Business Intelligence та з елементами ГІС – на прикладі платформи SemanticForce / "IVСічневі ГІСи": Інтелектуальна оборона" (науково-практичний форум) / Академія Сухопутних військ імені гетьмана Петра Сагайдачного: Львів, 22-24 січня 2013 р. - С. 42-45.
4. Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar (2002). «Thumbs up? Sentiment Classification using Machine Learning Techniques». Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 79–86.
5. .Дішман, П., Флейшер, К.С. та В. Кніп. «Хронологічна та категоризована бібліографія ключових стипендій з конкурентної розвідки: частина 1 (1997-2003), Journal of Competitive Intelligence and Management , 1(1), 16–78.
6. Флейшер, Крейг С., Райт, Шейла та Р. Тіндейл. «Бібліографія та оцінка ключових наукових досліджень у галузі конкурентної розвідки: частина 4 (2003–2006), Journal of Competitive Intelligence and Management , 2007, 4(1), 32–92

7. Дмитро Ланде. «Анотація "OSINT у кібербезпеці. Навчальний посібник"», Київ 2024.
8. Легомінова, С., Щавінський, Ю., Рабчун, Д., Запорожченко, М., & Будзинський, О. (2024). НЕБЕЗПЕКА ІНСТРУМЕНТІВ OSINT ТА СПОСОБИ ПОМ'ЯКШЕННЯ НАСЛІДКІВ ЇХ ВИКОРИСТАННЯ ДЛЯ ОРГАНІЗАЦІЇ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(25), 294–303. https://doi.org/10.28925/2663-4023.2024.25.294303
9. Главацька, А., Ангельська , О., & Опірський, І. (2024). ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ВИКОРИСТАННЯ OSINT ЯК НОВОЇ ЗАГРОЗИ З ДЕАНОНІМІЗАЦІЇ ОСОБИ В ІНТЕРНЕТ ПРОСТОРІ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(25), 19–50. https://doi.org/10.28925/2663-4023.2024.25.1950
10. Яровой, Т. С. (2019). OSINT, ЯК ПЕРСПЕКТИВНИЙ ІНСТРУМЕНТ КОНТРОЛЮ ЗА ЛОБІСТСЬКОЮ ДІЯЛЬНІСТЮ В КОНТЕКСТІ ДЕРЖАВНОЇ БЕЗПЕКИ. Експерт: парадигми юридичних наук і державного управління, (4(6), 201-208. https://doi.org/10.32689/2617-9660-2019-4(6)-201-208
11. Лунгол, О. (2024). ОГЛЯД МЕТОДІВ ТА СТРАТЕГІЙ КІБЕРБЕЗПЕКИ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ . Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(25), 379–389. https://doi.org/10.28925/2663-4023.2024.25.379389
12. Groosha. Введення – Пишемо Telegram-ботів з aiogram 3.x : посібник / пер. з англ. Groosha. – Режим доступу: https://mastergroosha.github.io/aiogram-3-guide/, вільний, (13.06.2025)

ДОДАТОК 1

Код телеграм боту

Структура телеграм бота

- Telegram_bot
  - config.py
  - duckduckgo_search.py
  - handlers.py
  - osint_search.py
  - phone_search.py
  - social_search.py
  - xml_search.py
  - proxies.txt
  - 23-ex_xml_rba.xml
- main.py

Файл config.py

```
TOKEN_API ="********"


# Додаємо нові налаштування для сервісів перевірки номерів
PHONE_SEARCH_CONFIG = {
   "numverify_api_key": "********",
   "timeout": 10,
   "max_attempts": 3
}
```

Файл duckduckgo_search.py

```
import time
import random
from bs4 import BeautifulSoup
from fake_useragent import UserAgent
import requests
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
```

```
PROXY_FILE = r"Telegram_bot\app\proxies.txt"

def load_proxies(file_path):
    with open(file_path, "r") as f:
        return [line.strip() for line in f.readlines() if line.strip()]

def get_proxy_dict(proxy_string):
    ip, port, user, password = proxy_string.split(":")
    proxy_auth = f"http://{user}:{password}@{ip}:{port}"
    return {"http": proxy_auth, "https": proxy_auth}

def get_headers():
    return {"User-Agent": UserAgent().random}

def search_duckduckgo(query, proxy_list=None, use_selenium=False):
    if use_selenium:
        options = Options()
        options.add_argument("--start-maximized")

        driver = webdriver.Chrome(options=options)
        driver.get("https://html.duckduckgo.com/html/")

        time.sleep(2)
        search_input = driver.find_element(By.NAME, "q")
        search_input.clear()
        search_input.send_keys(query)
        search_input.submit()

        time.sleep(5)

        results = driver.find_elements(By.CSS_SELECTOR, ".result__url")
        links = [r.get_attribute("href") for r in results]
```

```
        time.sleep(3)

        driver.quit()
        return links

    else:
        if proxy_list is None or len(proxy_list) == 0:
            proxies = None
        else:
            proxy_string = random.choice(proxy_list)
            proxies = get_proxy_dict(proxy_string)

        headers = get_headers()
        url = "https://html.duckduckgo.com/html/"
        params = {"q": query}

        try:
            response = requests.post(url, data=params, headers=headers, proxies=proxies, timeout=20)
            response.raise_for_status()

            with open("duckduckgo_response.html", "w", encoding="utf-8") as f:
                f.write(response.text)

            soup = BeautifulSoup(response.text, "html.parser")
            results = soup.find_all("a", class_="result__a")
            links = [a["href"] for a in results if a["href"].startswith("http")]
            return links
        except Exception as e:
            print("❌ Помилка:", e)
            return []
```

Файл handlers.py

```
import re
from aiogram import F, Router
```

```
from aiogram.types import Message, ReplyKeyboardMarkup, KeyboardButton
from aiogram.filters import CommandStart, Command
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from app.osint_search import run_osint_noapi, xml_search, company_name
from app.social_search import search_social_networks
from app.phone_search import search_phone_number
from typing import Dict, Any

router = Router()

class SearchState(StatesGroup):
    waiting_for_edrpou = State()
    waiting_for_name = State()
    waiting_for_person_name = State()
    waiting_for_phone = State()

def format_social_media_results(social_data: Dict) -> str:
    if not social_data:
        return "❌ Не знайдено профілів у соціальних мережах"

    formatted = []
    for platform, url in social_data.items():
        if ' ' in url:
            url = url.replace(' ', '')

        icons = {
            'facebook': '📘',
            'twitter': '🐦',
            'linkedin': '💼',
            'instagram': '📸',
            'telegram': '✈️'
        }
```

```
        icon = icons.get(platform.lower(), '🔗')
        formatted.append(f"{icon} {platform.capitalize()}: {url}")

    return "🌐 Знайдені профілі у соцмережах:\n\n" + "\n".join(formatted)

@router.message(CommandStart())
async def cmd_start(message: Message, state: FSMContext):
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="🔍 Пошук за ЄДРПОУ")],
            [KeyboardButton(text="🏢 Пошук за назвою")],
            [KeyboardButton(text="👤 Пошук за ім'ям")],
            [KeyboardButton(text="📱 Пошук за номером")]  # Нова кнопка
        ],
        resize_keyboard=True
    )
    await message.answer("Оберіть тип пошуку:", reply_markup=keyboard)
    await state.clear()

@router.message(F.text == "📱 Пошук за номером")
async def start_phone_search(message: Message, state: FSMContext):
    await message.answer(
        "Введіть номер телефону у міжнародному форматі (+380...)",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[[KeyboardButton(text="↩️ Назад")]],
            resize_keyboard=True
        )
    )
    await state.set_state(SearchState.waiting_for_phone)

@router.message(SearchState.waiting_for_phone)
async def handle_phone_search(message: Message, state: FSMContext):
    phone = message.text.strip()
```

```
    if phone.lower() == "назад":
        await cmd_start(message, state)
        return

    if not re.match(r"^\+\d{7,15}$", phone):
        await message.answer("❌ Невірний формат номеру. Використовуйте міжнародний формат (+380...)")
        return

    await message.answer(f"🔎 Шукаю інформацію за номером {phone}...")

    try:
        result = search_phone_number(phone)
        await message.answer(result, disable_web_page_preview=True)
    except Exception as e:
        await message.answer(f"⚠️ Помилка пошуку: {str(e)}")

    await state.clear()
    await cmd_start(message, state)

@router.message(F.text == "🔍 Пошук за ЄДРПОУ")
async def choose_edrpou(message: Message, state: FSMContext):
    await message.answer("🔢 Введіть ЄДРПОУ для пошуку:", reply_markup=ReplyKeyboardMarkup(
        keyboard=[[KeyboardButton(text="↩️ Назад")]],
        resize_keyboard=True
    ))
    await state.set_state(SearchState.waiting_for_edrpou)

@router.message(F.text == "🏢 Пошук за Назвою компанії")
async def choose_company_name(message: Message, state: FSMContext):
    await message.answer("🏛 Введіть назву компанії для пошуку:", reply_markup=ReplyKeyboardMarkup(
```

```
        keyboard=[[KeyboardButton(text="↩️ Назад")]],
        resize_keyboard=True
    ))
    await state.set_state(SearchState.waiting_for_name)


@router.message(F.text == "👤 Пошук людини за ім'ям")
async def choose_person_name(message: Message, state: FSMContext):
        await message.answer("👤 Введіть ПІБ людини для пошуку в соцмережах:", reply_markup=ReplyKeyboardMarkup(
        keyboard=[[KeyboardButton(text="↩️ Назад")]],
        resize_keyboard=True
    ))
    await state.set_state(SearchState.waiting_for_person_name)


@router.message(F.text == "↩️ Назад")
async def back_to_main(message: Message, state: FSMContext):
    await cmd_start(message, state)


@router.message(SearchState.waiting_for_person_name)
async def handle_person_name(message: Message, state: FSMContext):
    name = message.text.strip()
    if name.lower() == "назад":
        await back_to_main(message, state)
        return

    await message.answer(f"🔎 Шукаю особу '{name}' в соцмережах...")

    try:
        with open(r"H:\Telegram_bot-20250530T004820Z-1-001\Telegram_bot\app\proxies.txt", "r") as f:
            proxy_list = [line.strip() for line in f if line.strip()]

        results = search_social_networks(query=name, name=name, proxy_list=proxy_list)

        if results:
```

```
            response = format_social_media_results(results)
            await message.answer(response)
        else:
            await message.answer("❌ Не вдалося знайти профілі за вказаним ім'ям.")

    except Exception as e:
        await message.answer(f"⚠️ Сталася помилка під час пошуку: {str(e)}")

    await state.clear()
    await cmd_start(message, state)

@router.message(SearchState.waiting_for_edrpou)
async def handle_edrpou(message: Message, state: FSMContext):
    edrpou = message.text.strip()
    if edrpou.lower() == "назад":
        await back_to_main(message, state)
        return

    await process_osint_results(message, edrpou, True)
    await state.clear()
    await cmd_start(message, state)

@router.message(SearchState.waiting_for_name)
async def handle_name(message: Message, state: FSMContext):
    name = message.text.strip()
    if name.lower() == "назад":
        await back_to_main(message, state)
        return

    await process_osint_results(message, name, False)
    await state.clear()
    await cmd_start(message, state)

async def process_osint_results(message: Message, query: str, is_edrpou: bool):
```

```
    await message.answer("🔎 Виконую комплексний OSINT-пошук...")

    results = run_osint_noapi(query)

    if clarity := results.get("clarity"):
        formatted = "\n".join([f"{k}\n{v}" for k, v in clarity.items()])
        await message.answer(f"✅ Дані з Clarity:\n\n{formatted}")
    else:
        await message.answer("❌ Нічого не знайдено в Clarity.")

    if duck := results.get("duckduckgo"):
        links_text = "\n".join(duck[:5])
        await message.answer(f"🔗 Посилання з DuckDuckGo:\n\n{links_text}")
    else:
        await message.answer("❌ DuckDuckGo не знайшов результатів.")

    await message.answer("🔎 Виконую пошук по локальному XML...")
    search_query = query if is_edrpou else company_name(query)
    xml_results = xml_search(search_query)

    if not xml_results:
        await message.answer("❌ Результат не знайдено у базі банкрутства.")
    else:
        lines = []
        for r in xml_results:
            lines.append(f'RN_NUM: {r['RN_NUM']}, REG_DATE: {r['REG_DATE']}, NAME: {r['NAME']}")
        await message.answer("✅ Результати пошуку у базі банкрутства:\n" + "\n".join(lines))
```

Файл osint_search.py

```
from bs4 import BeautifulSoup
import urllib.parse
import re, sys, json, requests
import xml.etree.ElementTree as ET
```

```
from app.duckduckgo_search import search_duckduckgo, load_proxies

sys.stdout.reconfigure(encoding='utf-8')

try:
    from fake_useragent import UserAgent
    ua = UserAgent()
    headers = {"User-Agent": ua.random}
except Exception:
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}

xml_file = r"Telegram_bot\app\23-ex_xml_rba.xml"
tree = ET.parse(xml_file)
root = tree.getroot()

def company_name(query):
    if query.isdigit():
        edrpou = query
        result = clarity_search_edrpou(edrpou)
        full_name = result.get("Назва:", "")
        match = re.search(r'\((.*?)\)', full_name)
        if match:
            short_name = match.group(1)
            return short_name
    else :
        edrpou = find_edrpou(query)
        result = clarity_search_edrpou(edrpou)
        full_name = result.get("Назва:", "")
        match = re.search(r'\((.*?)\)', full_name)
        if match:
            short_name = match.group(1)
            return short_name

def search_records(field_name, search_value):
```

```
    results = []
    for record in root.findall('RECORD'):
        field = record.find(field_name)
        if field is not None and field.text is not None and search_value in field.text:
            results.append(record)
    return results


def xml_search(query):
    short_name = company_name(query)
    if short_name:
        found_records = search_records("NAME", short_name)
    else:
        found_records = search_records("NAME", query)

    results = []
    for rec in found_records:
        rn_num = rec.find("RN_NUM").text if rec.find("RN_NUM") is not None else ""
        reg_date = rec.find("REG_DATE").text if rec.find("REG_DATE") is not None else ""
        name = rec.find("NAME").text if rec.find("NAME") is not None else ""
        results.append({
            "RN_NUM": rn_num,
            "REG_DATE": reg_date,
            "NAME": name
        })
    return results


def safe_request(url):
    try:
        res = requests.get(url, headers=headers, timeout=15)

        if res.status_code == 200:
            return res
        else:
            print(f" ⚠️ Помилка запиту {url} — код {res.status_code}")
```

```
            return None
    except requests.exceptions.RequestException as e:
        print(f" ⚠️ Запит не вдався: {url}\n→ {e}")
        return None

def clarity_search_edrpou(edrpou):
    url = f"https://clarity-project.info/edr/{edrpou}"
    res = safe_request(url)
    if not res:
        return

    soup = BeautifulSoup(res.text, "html.parser")

    table = soup.find('table', class_='table align-top mb-15 border-bottom w-100')

    date_update = soup.find("div", {"class": "updated-at"}).find("b").text
    date_update = re.sub(r"\s+", "", date_update)

    result = {}
    result["Дата оновлення даних:"] = date_update

    for row in table.find_all('tr'):
        tds = row.find_all('td')
        if len(tds) == 2:
            key = tds[0].get_text(strip=True)
            value = tds[1].get_text(separator=' ', strip=True)
            value = re.sub(r'\s+', ' ', value).strip()
            result[key] = value

    with open("company_info.txt", "w", encoding="utf-8") as out_file:
        for k, v in result.items():
            out_file.write(f"{k}\n{v}\n\n")
    return result
```

```
def find_edrpou(query):
    url = f"https://clarity-project.info/edrs?query={urllib.parse.quote(query)}"
    res = safe_request(url)
    if not res:
        return None

    soup = BeautifulSoup(res.text, "html.parser")
    edrpou_div = soup.find('div', class_='small text-secondary mb-5')
    if not edrpou_div:
        return None

    edrpou_match = re.search(r'\d{8}', edrpou_div.text)
    if not edrpou_match:
        return None

    edrpou = edrpou_match.group(0)
    return edrpou

def split_to_list(value):
    if not value:
        return []
    items = re.split(r'[;\n,•\-]+', value)
    items = [item.strip() for item in items if item.strip()]
    return items

def print_clarity_data(data):
    if not data:
        print("Дані відсутні.")
        return

    fields = [
        "Дата оновлення даних:",
        "ЄДРПОУ:",
        "Назва:",
```

```
        "Назва іноземною мовою:",
        "Організаційна форма:",
        "Адреса:",
        "Стан:",
        "Дата реєстрації:",
        "Уповноважені особи:",
        "Бухгалтер:",
        "Статутний капітал:",
        "Засновники:",
        "Кінцеві бенефіціари:",
        "Види діяльності:",
        "Контакти:",
        "Відомості про структуру власності:"
    ]

    for key in fields:
        if key in data:
            value = data[key]
            if isinstance(value, str) and len(value) > 50:
                items = split_to_list(value)
                if len(items) > 1:
                    print(f"{key}")
                    for i, item in enumerate(items, 1):
                        print(f"  {i}. {item}")
                    print()
                else:
                    print(f"{key} {value}\n")
            else:
                print(f"{key} {value}\n")

def run_osint_noapi(query):
    results = {}

    if query.isdigit():
```

```
        print("🟣 Clarity (ЄДРПОУ):")
        edrpou = query
        results["clarity"] = clarity_search_edrpou(edrpou)
    else:
        print("🟣 Clarity (назва):")
        edrpou = find_edrpou(query)
        results["clarity"] = clarity_search_edrpou(edrpou)

    print("🟠 DuckDuckGo Dorking:")
    proxy_list = load_proxies("Telegram_bot/app/proxies.txt")

    if edrpou:
        links = search_duckduckgo(f'{edrpou} (site:gov.ua OR site:org.ua)', proxy_list, True)
        results["duckduckgo"] = links
    else:
        results["duckduckgo"] = []

    return results

if __name__ == "__main__":
    query = input("Введіть назву компанії або ЄДРПОУ: ").strip()
    data = run_osint_noapi(query)

    if data:
        print("\n📄 Звіт:")
        for source, entries in data.items():
            print(f"\n--- {source.upper()} ---")
            if isinstance(entries, dict):
                print_clarity_data(entries)
            else:
                for entry in entries[:5]:
                    print(json.dumps(entry, ensure_ascii=False, indent=2))
    else:
        print("❌ Немає даних для виводу.")
```

Файл phone_search.py

```
import re
import requests
import phonenumbers
import random
import logging
from typing import Dict, Optional, List
from phonenumbers import carrier, geocoder, timezone
from bs4 import BeautifulSoup
from fake_useragent import UserAgent
from app.config import PHONE_SEARCH_CONFIG

logger = logging.getLogger(__name__)

class PhoneSearcher:
    def __init__(self):
        self.ua = UserAgent()
        self.numverify_key = PHONE_SEARCH_CONFIG["numverify_api_key"]
        self.proxy_file = "proxies.txt"

    def _get_random_proxy(self) -> Dict[str, str]:
        try:
            with open(self.proxy_file, "r") as f:
                proxies = [line.strip() for line in f if line.strip()]

            if not proxies:
                logger.warning("Файл проксі знайдено, але він порожній")
                return {}
            proxy_str = random.choice(proxies)

            try:
                ip, port, user, password = proxy_str.split(":")
                return {
```

```
                "http": f"http://{user}:{password}@{ip}:{port}",
                "https": f"http://{user}:{password}@{ip}:{port}"
            }
        except ValueError as e:
            logger.error(f"Невірний формат проксі '{proxy_str}': {e}")
            return {}

    except FileNotFoundError:
        logger.warning(f"Файл {self.proxy_file} не знайдено")
        return {}
    except Exception as e:
        logger.error(f"Помилка читання проксі: {e}")
        return {}

def basic_phone_info(self, number: str) -> Dict:
    try:
        parsed = phonenumbers.parse(number)
        return {
            "Країна": geocoder.description_for_number(parsed, "uk"),
            "Оператор": carrier.name_for_number(parsed, "uk"),
            "Часовий пояс": timezone.time_zones_for_number(parsed),
            "Валідність": "Так" if phonenumbers.is_valid_number(parsed) else "Ні"
        }
    except Exception as e:
        logger.error(f"Помилка аналізу номеру: {e}")
        return {"Помилка": str(e)}

def search_social_media(self, number: str) -> Dict:
    results = {}
    platforms = {
        "Telegram": f"https://t.me/+{number}",
        "WhatsApp": f"https://wa.me/{number}",
        "Viber": f"viber://chat?number={number}"
    }
```

```
    for platform, url in platforms.items():
        try:
            proxy = self._get_random_proxy()
            response = requests.get(
                url,
                headers={"User-Agent": self.ua.random},
                proxies=proxy,
                timeout=10,
                allow_redirects=False
            )
            if response.status_code in (200, 302):
                results[platform] = url
        except Exception as e:
            logger.warning(f"Помилка пошуку {platform}: {e}")
            continue

    return results

def numverify_lookup(self, number: str) -> Dict:
    if not self.numverify_key:
        logger.warning("API ключ Numverify не налаштовано")
        return {}

    url = f"http://apilayer.net/api/validate?access_key={self.numverify_key}&number={number}"
    try:
        proxy = self._get_random_proxy()
        response = requests.get(url, proxies=proxy, timeout=10)
        return response.json()
    except Exception as e:
        logger.error(f"Помилка Numverify API: {e}")
        return {}

def search_leaks(self, number: str) -> List[str]:
```

```
        return []

    def format_results(self, data: Dict) -> str:
        formatted = []
        for section, content in data.items():
            if isinstance(content, dict):
                items = [f"{k}: {v}" for k, v in content.items() if v]
                if items:
                    formatted.append(f" 🔹 {section}:\n" + "\n".join(items))
            elif content:
                formatted.append(f" 🔹 {section}:\n" + "\n".join(content))

        return "\n\n".join(formatted) if formatted else " 🔍 Інформація не знайдена"

def search_phone_number(number: str) -> str:
    if not re.match(r"^\+\d{7,15}$", number):
        return " ❌ Невірний формат номеру. Використовуйте міжнародний формат (+380...)"

    searcher = PhoneSearcher()
    results = {
        " 📱 Базова інформація": searcher.basic_phone_info(number),
        " 🌐 Соцмережі": searcher.search_social_media(number),
        " 🔎 API Numverify": searcher.numverify_lookup(number),
        " ⚠️ Можливі витоки": searcher.search_leaks(number)
    }
    return searcher.format_results(results)
```

Файл social_search.py

```
import re
import requests
from bs4 import BeautifulSoup
from fake_useragent import UserAgent
```

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
import time
from urllib.parse import quote
import random
from typing import Dict, Optional, List
import logging

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

class SocialMediaSearcher:
    def __init__(self, proxy_list: List[str] = None):
        self.ua = UserAgent()
        self.proxy_list = proxy_list or []
        self.driver = None
        self.used_proxies = set()
        self.failed_proxies = set()

    def _get_random_proxy(self) -> Optional[Dict[str, str]]:
        available_proxies = [p for p in self.proxy_list
                             if p not in self.failed_proxies and p not in self.used_proxies]

        if not available_proxies:
            self.used_proxies.clear()
            available_proxies = [p for p in self.proxy_list if p not in self.failed_proxies]
            if not available_proxies:
                return None

        proxy_str = random.choice(available_proxies)
```

```
        self.used_proxies.add(proxy_str)

        try:
            ip, port, user, password = proxy_str.split(":")
            return {
                "http": f"http://{user}:{password}@{ip}:{port}",
                "https": f"http://{user}:{password}@{ip}:{port}"
            }
        except Exception as e:
            logger.error(f"Помилка парсингу проксі {proxy_str}: {e}")
            self.failed_proxies.add(proxy_str)
            return None

    def _get_headers(self) -> Dict[str, str]:
        return {
            "User-Agent": self.ua.random,
            "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
            "Accept-Language": "en-US,en;q=0.5",
            "Accept-Encoding": "gzip, deflate, br",
            "Connection": "keep-alive"
        }

    def _init_selenium(self, proxy: Dict[str, str] = None):
        options = Options()
        options.add_argument("--headless")
        options.add_argument("--disable-gpu")
        options.add_argument("--no-sandbox")
        options.add_argument("--disable-dev-shm-usage")

        if proxy:
            options.add_argument(f"--proxy-server={proxy['http']}")

        try:
            self.driver = webdriver.Chrome(options=options)
```

```
            self.driver.set_page_load_timeout(30)
        except Exception as e:
            logger.error(f"Помилка ініціалізації Selenium: {e}")
            raise

    def search_profiles(self, query: str, name: Optional[str] = None) -> Dict[str, str]:
        results = {}

        if name:
            results.update(self._search_by_name(name))

        if query:
            results.update(self._search_by_query(query))

        return {k: v for k, v in results.items() if v}

    def _search_by_name(self, name: str) -> Dict[str, str]:
        encoded_name = quote(name)
        results = {}

        # Facebook
        fb_result = self._check_facebook(encoded_name)
        if fb_result:
            results['facebook'] = fb_result

        # Twitter (використовує Selenium)
        tw_result = self._check_twitter(encoded_name)
        if tw_result:
            results['twitter'] = tw_result

        # LinkedIn (використовує Selenium)
        li_result = self._check_linkedin(encoded_name)
        if li_result:
            results['linkedin'] = li_result
```

```
    return results

def _search_by_query(self, query: str) -> Dict[str, str]:
    results = {}

    # Instagram
    ig_result = self._check_instagram(query)
    if ig_result:
        results['instagram'] = ig_result

    # Telegram
    tg_result = self._check_telegram(query)
    if tg_result:
        results['telegram'] = tg_result

    return results

def _check_facebook(self, name: str) -> Optional[str]:
    """Перевіряє наявність профілю на Facebook"""
    url = f"https://www.facebook.com/public/{name}"
    try:
        proxy = self._get_random_proxy()
        res = requests.get(
            url,
            headers=self._get_headers(),
            proxies=proxy,
            timeout=15
        )

        if res.status_code == 200 and "content=\"profile" in res.text:
            return url

    except requests.RequestException as e:
```

```
            logger.warning(f"Помилка пошуку Facebook: {e}")

        return None

    def _check_twitter(self, name: str) -> Optional[str]:
        """Перевіряє наявність профілю на Twitter"""
        url = f"https://twitter.com/search?q={name}&f=user"
        try:
            if not self.driver:
                proxy = self._get_random_proxy()
                self._init_selenium(proxy)

            self.driver.get(url)
            time.sleep(3)

            # Перевіряємо наявність результатів
            results = self.driver.find_elements(By.XPATH, "//div[@data-testid='UserCell']")
            if results:
                return url

        except Exception as e:
            logger.warning(f"Помилка пошуку Twitter: {e}")
            try:
                self.driver.quit()
            except:
                pass
            self.driver = None

        return None

    def _check_linkedin(self, name: str) -> Optional[str]:
        """Перевіряє наявність профілю на LinkedIn"""
        url = f"https://www.linkedin.com/search/results/people/?keywords={name}"
        try:
```

```
            if not self.driver:
                proxy = self._get_random_proxy()
                self._init_selenium(proxy)

            self.driver.get(url)
            time.sleep(3)

            # Перевіряємо наявність результатів
            results = self.driver.find_elements(
                By.XPATH, "//li[contains(@class, 'reusable-search__result-container')]")
            if results:
                return url

        except Exception as e:
            logger.warning(f"Помилка пошуку LinkedIn: {e}")
            try:
                self.driver.quit()
            except:
                pass
            self.driver = None

        return None

    def _check_instagram(self, username: str) -> Optional[str]:
        """Перевіряє наявність профілю на Instagram"""
        # Видаляємо пробіли з username
        username = username.replace(' ', '')
        url = f"https://www.instagram.com/{username}/"
        try:
            proxy = self._get_random_proxy()
            res = requests.get(
                url,
                headers=self._get_headers(),
                proxies=proxy,
```

```
            timeout=10
        )

        if res.status_code == 200 and "Page Not Found" not in res.text:
            return url

    except requests.RequestException as e:
        logger.warning(f"Помилка пошуку Instagram: {e}")

    return None

def _check_telegram(self, username: str) -> Optional[str]:
    """Перевіряє наявність профілю на Telegram"""
    url = f"https://t.me/{username}"
    try:
        proxy = self._get_random_proxy()
        res = requests.get(
            url,
            headers=self._get_headers(),
            proxies=proxy,
            timeout=10
        )

        if res.status_code == 200 and "tgme_page_description" in res.text:
            return url

    except requests.RequestException as e:
        logger.warning(f"Помилка пошуку Telegram: {e}")

    return None

def __del__(self):
    if self.driver:
        try:
```

```
            self.driver.quit()
        except:
            pass

def search_social_networks(query: str, name: Optional[str] = None, proxy_list: List[str] = None) ->
Dict[str, str]:
    searcher = SocialMediaSearcher(proxy_list)
    return searcher.search_profiles(query, name)
```

Файл xml_search.py

```
import xml.etree.ElementTree as ET
from app.osint_search import company_name

xml_file = r"Telegram_bot\app\23-ex_xml_rba.xml"
tree = ET.parse(xml_file)
root = tree.getroot()

def search_records(field_name, search_value):
    results = []
    for record in root.findall('RECORD'):
        field = record.find(field_name)
        if field is not None and field.text is not None and search_value in field.text:
            results.append(record)
    return results

def xml_search(query):
    short_name = company_name(query)
    print(f"Коротка назва компанії для '{query}': {short_name}")

    if short_name:
        found_records = search_records("NAME", short_name)
    else:
        found_records = search_records("NAME", query)
```

```
    if not found_records:
        print("Записи не знайдені.")
    else:
        for rec in found_records:
            rn_num = rec.find("RN_NUM").text if rec.find("RN_NUM") is not None else ""
            reg_date = rec.find("REG_DATE").text if rec.find("REG_DATE") is not None else ""
            name = rec.find("NAME").text if rec.find("NAME") is not None else ""
            print(f"RN_NUM: {rn_num}, REG_DATE: {reg_date}, NAME: {name}")
```

Файл main.py

```
import asyncio
from aiogram import Bot, Dispatcher
from aiogram.fsm.storage.memory import MemoryStorage
from app.handlers import router
from app.config import TOKEN_API
async def main():
    bot = Bot(token=TOKEN_API)
    dp = Dispatcher(storage=MemoryStorage())
    dp.include_router(router)
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```