Методи збору даних для інформаційно-аналітичного забезпечення органів військового управління

Вступ: Сучасна інформаційна розвідка та роль даних в ІАЗ ОВУ

В умовах сучасних гібридних конфліктів інформація перетворилася на ключовий ресурс та водночас на поле бою. Інформаційно-аналітичне забезпечення органів військового управління (ІАЗ ОВУ) зазнало фундаментальної трансформації, змістивши акценти з традиційних методів розвідки на комплексне використання даних з відкритих джерел. Ця еволюція є не просто технологічним оновленням, а зміною самої парадигми розвідувальної діяльності.

Еволюція збору даних у військовій справі

Історично розвідувальна діяльність спиралася на обмежені, важкодоступні джерела. Агентурна розвідка (HUMINT) надавала унікальні дані, але була пов'язана з високими ризиками та обмеженим охопленням. Радіоелектронна розвідка (SIGINT) дозволяла перехоплювати комунікації, але вимагала значних технічних ресурсів та була вразливою до шифрування. Сьогодні до цих класичних дисциплін додався потужний інструмент – розвідка на основі відкритих джерел (Open Source Intelligence, OSINT). Інтернет, соціальні мережі, публічні бази даних, супутникові знімки та медіаресурси генерують безпрецедентні обсяги інформації, доступ до якої є відносно простим.

Цей перехід до OSINT кардинально змінив ландшафт розвідки. Якщо раніше головним викликом був *доступ* до інформації, то сьогодні ним стали *фільтрація* та *верифікація*. Дані

стали надлишковими, і аналітик стикається з ризиком "потонути" у потоці нерелевантної, застарілої або, що найнебезпечніше, навмисно сфальсифікованої інформації (дезінформації), яку активно використовує противник. У цих умовах цінність аналітика зміщується від ролі "збирача даних" до ролі "верифікатора" та "синтезатора сенсів". Технічні інструменти, що розглядаються в цьому посібнику, є лише засобами; ключовою компетенцією залишається критичне мислення, вміння виявляти зв'язки та формувати цілісну картину з розрізнених, перевірених фрагментів інформації.

Класифікація джерел та методів

Для ефективної роботи аналітик повинен чітко класифікувати методи збору даних та типи самої інформації.

- **Активний vs. Пасивний збір:**
  - **Пасивний збір** передбачає отримання інформації без прямої взаємодії з цільовим ресурсом, що залишає мінімальний цифровий слід. Прикладами є моніторинг публічних трансляцій, використання офіційних API (Application Programming Interfaces), аналіз кешу пошукових систем. Цей метод є пріоритетним з точки зору операційної безпеки.
  - **Активний збір** включає прямі запити до серверів, сканування сайтів (веб-скрапінг), взаємодію з веб-додатками. Такі дії можуть бути зафіксовані системами безпеки цільового ресурсу, що створює ризик деанонімізації.
- **Структуровані vs. Неструктуровані дані:**
  - **Структуровані дані** організовані у фіксованому форматі, що робить їх готовими до автоматизованого аналізу. Це можуть бути таблиці, бази даних, відповіді від API у форматах JSON або XML.
  - **Неструктуровані дані** не мають заздалегідь визначеної моделі. До них належать тексти новин, статті, дописи в соціальних мережах, коментарі, зображення, відео та аудіозаписи. Цей тип даних складає переважну більшість інформації в Інтернеті і потребує попередньої обробки (парсингу, розпізнавання сутностей, аналізу тональності) перед використанням.

Операційна безпека (OPSEC) як фундаментальний принцип

У контексті ІАЗ ОВУ операційна безпека (OPSEC) не є опцією, а абсолютною вимогою. Будь-яка діяльність зі збору даних в Інтернеті залишає цифровий слід (IP-адреса, User-Agent браузера, параметри системи), який може бути використаний противником

для ідентифікації джерела розвідувального інтересу або для проведення контрзаходів. Кожна операція зі збору даних повинна починатися з ретельної оцінки ризиків та розробки плану заходів з їх мінімізації, включаючи використання VPN, проксі-серверів, мережі Tor, віртуальних машин та неатрибутованих облікових записів. Нехтування OPSEC може призвести не тільки до провалу операції, але й до компрометації персоналу та методів роботи.

Частина 1. Автоматизований збір даних з мережі Internet: "Полювання" на інформацію

Ця частина присвячена методам активного видобутку інформації з веб-ресурсів, які часто не призначені для автоматизованого збору. Цей процес можна порівняти з "полюванням": аналітик вистежує, ідентифікує та вилучає потрібні дані з динамічного та іноді ворожого середовища. Це вимагає не лише технічних навичок, але й глибокого розуміння принципів функціонування Інтернету та здатності адаптуватися до постійних змін.

Розділ 1.1. Веб-скрапінг: Видобуток неструктурованих даних з веб-сайтів

Веб-скрапінг (або парсинг) – це процес автоматизованого вилучення даних з веб-сторінок. Замість того, щоб вручну копіювати інформацію, аналітик пише або використовує програму (скрапер), яка завантажує HTML-код сторінки та витягує з нього необхідні елементи: текст, посилання, зображення, таблиці.

1.1.1. Технічні основи для аналітика

Щоб ефективно створювати скрапери, необхідно розуміти базові технології, на яких побудований веб.

- **Протокол HTTP/HTTPS:** Це основа взаємодії в Інтернеті. Коли користувач вводить адресу сайту в браузер, той надсилає HTTP-запит (request) до сервера, на якому розміщено сайт. Сервер обробляє запит і повертає HTTP-відповідь (response), яка

зазвичай містить HTML-код сторінки. Скрапер, по суті, імітує цей процес. Найбільш поширеними методами запитів є GET (для отримання даних) та POST (для відправки даних, наприклад, при заповненні форм).

- **Структура веб-сторінки (HTML, CSS, DOM):**
  - **HTML (HyperText Markup Language)** – це "скелет" сторінки. Він складається з тегів (наприклад, <p> для параграфа, <h1> для заголовка, <a> для посилання), які структурують контент. Саме в HTML-коді містяться дані, які є ціллю скрапінгу.
  - **CSS (Cascading Style Sheets)** – це "зовнішній вигляд" сторінки. CSS визначає кольори, шрифти, розташування елементів. Для скрапінгу CSS важливий тим, що елементи часто мають унікальні ідентифікатори (id) або класи (class), які можна використовувати для їх точного пошуку в HTML-коді.
  - **DOM (Document Object Model)** – це об'єктне представлення HTML-документа у пам'яті. Коли браузер завантажує сторінку, він будує DOM-дерево, де кожен HTML-тег є вузлом. Саме з цим структурованим деревом працюють програми-парсери, що дозволяє їм навігуватися по сторінці та вибирати потрібні елементи.

1.1.2. Базовий інструментарій на Python

Python є де-факто стандартом для веб-скрапінгу завдяки великій кількості потужних та простих у використанні бібліотек.

- **Бібліотека requests:** Ця бібліотека дозволяє надзвичайно просто надсилати HTTP-запити. За допомогою одного рядка коду можна отримати HTML-вміст будь-якої веб-сторінки. Це перший крок у будь-якому процесі скрапінгу.
- **Бібліотека BeautifulSoup:** Після отримання HTML-коду за допомогою requests, BeautifulSoup дозволяє його розібрати (спарсити). Вона перетворює сирий HTML-текст на об'єкт, по якому можна легко навігуватися, шукати елементи за тегами, класами, ідентифікаторами та іншими атрибутами, витягуючи потрібний текст або посилання.

Практичний приклад: Скрапінг заголовків новин з сайту умовного противника

Python

```
# Імпортуємо необхідні бібліотеки
import requests
from bs4 import BeautifulSoup
```

```
# URL цільового сайту
url = 'http://example-enemy-news.com'

# Встановлюємо заголовок User-Agent, щоб імітувати реальний браузер
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
}

# Надсилаємо GET-запит до сайту
try:
    response = requests.get(url, headers=headers)
    response.raise_for_status()  # Перевіряємо, чи запит був успішним (код 200)

    # Створюємо об'єкт BeautifulSoup для парсингу HTML
    soup = BeautifulSoup(response.text, 'html.parser')

    # Знаходимо всі заголовки новин. Припустимо, вони знаходяться в тегах <h3> з класом 'news-title'
    # Цю структуру потрібно визначити, проаналізувавши HTML-код цільової сторінки
    headlines = soup.find_all('h3', class_='news-title')

    # Виводимо текст кожного заголовка
    print("Останні заголовки новин:")
    for headline in headlines:
        print(f"- {headline.get_text(strip=True)}")

except requests.exceptions.RequestException as e:
    print(f"Помилка під час запиту: {e}")
```

1.1.3. Просунуті техніки та інструменти

Багато сучасних веб-сайтів є динамічними, тобто їх контент завантажується за допомогою JavaScript вже після завантаження основної HTML-сторінки. Прості скрапери на базі requests не можуть отримати такі дані.

- **Робота з динамічними сайтами (Selenium):** Selenium – це інструмент для автоматизації веб-браузерів. Він дозволяє програмі керувати реальним браузером (наприклад, Chrome або Firefox), відкривати сторінки, клікати на кнопки,

заповнювати форми та чекати, поки JavaScript завантажить весь контент. Після цього можна отримати повний HTML-код сторінки і передати його BeautifulSoup для парсингу. Це значно потужніший, але й повільніший метод.
- **Масштабний збір даних (Scrapy):** Scrapy – це не просто бібліотека, а повноцінний фреймворк для створення складних скраперів, так званих "павуків" (spiders). Він призначений для великих завдань: обходу цілих сайтів, автоматичного переходу за посиланнями, паралельного виконання запитів та ефективного збереження даних у структурованому вигляді (наприклад, у JSON, CSV або базі даних). Scrapy є оптимальним вибором для завдань постійного моніторингу великої кількості ресурсів.

1.1.4. Виклики та методи протидії

Веб-скрапінг є по суті змагальним процесом. Аналітик знаходиться у постійній "грі в кішки-мишки" з адміністраторами сайтів, які впроваджують заходи для протидії автоматизованому збору даних. Це означає, що будь-який скрипт для скрапінгу є тимчасовим рішенням. Зміна структури HTML-коду, оновлення логіки безпеки або впровадження нових систем виявлення ботів може вивести з ладу скрапер, який бездоганно працював ще вчора.

Ця крихкість має серйозні наслідки для військової розвідки. Покладання на єдине джерело даних, що отримується шляхом скрапінгу, створює критичну вразливість. Інформаційний потік може бути перерваний у будь-який момент через просте оновлення сайту противника. Тому планування розвідувальних операцій повинно враховувати цю нестійкість. Необхідно розробляти резервні методи збору, а в ресурсне планування закладати не тільки час на початкову розробку скрапера, але й на його постійну підтримку, налагодження та адаптацію.

Основні методи захисту та способи їх обходу:

- **Ідентифікація та маскування:** Багато серверів блокують запити від програм, ідентифікуючи їх за заголовком User-Agent. Стандартний User-Agent бібліотеки requests одразу видає бота. Тому критично важливо підміняти його на User-Agent реального браузера, а в ідеалі – використовувати ротацію кількох різних User-Agent.
- **Блокування за IP-адресою:** Якщо з однієї IP-адреси надходить аномально велика кількість запитів за короткий проміжок часу, сервер може тимчасово або назавжди її заблокувати. Для обходу цього використовуються:
    - **Проксі-сервери:** Запити надсилаються через посередницькі сервери, кожен з яких має свою IP-адресу. Ротаційні проксі автоматично змінюють IP-адресу для кожного запиту або через певний інтервал часу.
    - **Мережа Tor:** Забезпечує високий рівень анонімності, маршрутизуючи трафік

через ланцюжок випадкових вузлів. Однак швидкість роботи через Tor значно нижча, і багато сайтів блокують вихідні вузли Tor.
- **CAPTCHA:** Це головна перешкода для автоматизації, призначена для того, щоб відрізнити людину від бота. Обхід CAPTCHA є складною задачею. Існують сервіси для автоматичного розпізнавання, які використовують працю людей або алгоритми машинного навчання, але їх використання пов'язане з додатковими витратами та безпековими ризиками.

1.1.5. Практичний кейс: "Моніторинг регіональних форумів та соціальних мереж на окупованих територіях для виявлення змін у суспільних настроях"

- **Постановка задачі:** Автоматично відстежувати появу нових повідомлень та коментарів на кількох визначених форумах та у відкритих групах в соціальних мережах, що стосуються життя на окупованих територіях. Мета – виявлення ключових тем обговорень, що свідчать про зміну суспільних настроїв, проблеми з логістикою, діяльність окупаційної влади тощо.
- **Вибір інструментів:** Для статичних форумів достатньо requests та BeautifulSoup. Для соціальних мереж, що динамічно завантажують контент, може знадобитися Selenium.
- **Процес:**
  1. Аналітик вручну досліджує структуру цільових сайтів, щоб визначити, в яких HTML-тегах та класах знаходяться потрібні дані (імена авторів, текст повідомлень, дата публікації).
  2. Пишеться скрипт на Python, який періодично (наприклад, раз на годину) заходить на цільові сторінки.
  3. Скрипт парсить сторінки та шукає повідомлення, що містять ключові слова (наприклад, "відключення світла", "ціни", "комендантська година", "перевірки", "мобілізація").
  4. Знайдені повідомлення (разом з посиланням на оригінал, автором та часом публікації) зберігаються у структурований CSV-файл або базу даних.
- **Результат:** Аналітичний підрозділ отримує щоденний звіт з новими релевантними повідомленнями, що дозволяє оперативно відстежувати динаміку ситуації без необхідності ручного моніторингу десятків ресурсів.

Розділ 1.2. API (Application Programming Interface): Структурований доступ до даних

Якщо веб-скрапінг – це "полювання" в диких умовах, то використання API – це отримання

інформації цивілізованим, регламентованим шляхом. API – це набір правил та інструментів, які дозволяють одній програмі взаємодіяти з іншою.

1.2.1. Концепція API

Найпростіша аналогія для розуміння API – це офіціант у ресторані. Клієнт (ваша програма) не йде на кухню (сервер) і не намагається самостійно зібрати собі страву з продуктів (даних). Замість цього він робить замовлення офіціанту (API) за чітко визначеним меню (документацією API). Офіціант передає замовлення на кухню, а потім приносить готову, гарно сервіровану страву (структуровані дані у форматі JSON).

- **Переваги над скрапінгом:**
  - **Надійність:** Структура даних, що повертаються, є стабільною і описана в документації. Вона не зміниться раптово через редизайн сайту.
  - **Легітимність:** Використання API відбувається за правилами, встановленими сервісом. Це "білий" метод збору даних.
  - **Швидкість та ефективність:** API повертає тільки потрібні дані без зайвого HTML-коду, CSS та JavaScript, що зменшує обсяг трафіку та час обробки.

1.2.2. Взаємодія з API

Робота з більшістю сучасних веб-API (так званих RESTful API) зводиться до надсилання HTTP-запитів на спеціальні URL-адреси.

- **Автентифікація:** На відміну від публічних сайтів, доступ до API зазвичай вимагає ідентифікації.
  - **API-ключі:** Найпростіший метод. Це унікальний рядок символів, який ви отримуєте після реєстрації в сервісі і який необхідно додавати до кожного запиту. Він працює як простий пароль для вашої програми.
  - **OAuth:** Більш складний, але й більш безпечний протокол, який дозволяє програмі отримувати доступ до даних від імені користувача, не вимагаючи від нього передачі логіна та пароля.
- **Формування запитів:** Запит до API складається з кількох частин:
  - **Endpoint:** URL-адреса, що вказує на конкретний ресурс, до якого ви звертаєтесь (наприклад, https://api.telegram.org/bot<token>/getUpdates).
  - **Параметри:** Додаються до URL для фільтрації або уточнення запиту (наприклад, ?q=keyword&count=100 для пошуку за ключовим словом та обмеження кількості результатів).

    - **Заголовки (Headers):** Містять службову інформацію, включаючи дані для автентифікації (наприклад, API-ключ).
- **Обробка відповідей (JSON):** Переважна більшість API повертає дані у форматі JSON (JavaScript Object Notation). Це текстовий формат, що представляє дані у вигляді пар "ключ-значення", подібних до словників у Python. Він легко читається як людиною, так і машиною. Python має вбудовану бібліотеку json для парсингу таких даних.

1.2.3. Огляд релевантних API для ІАЗ

- **Соціальні мережі:**
    - **Telegram:** API Telegram дозволяє створювати ботів, які можуть автоматично моніторити відкриті канали та групи, збирати повідомлення, медіафайли, реакції та коментарі. Бібліотеки Telethon та Pyrogram для Python значно спрощують цю задачу. Це потужний інструмент для відстеження ворожих пропагандистських каналів, груп колаборантів та інформаційних "вкидів".
    - **Twitter/X:** API надає доступ до публічних твітів, даних профілів, трендів. Це дозволяє аналізувати поширення дезінформації, виявляти мережі ботів (botnets), відстежувати реакцію на певні події в реальному часі.
- **Новинні агрегатори:** Сервіси на кшталт **NewsAPI** або **GDELT Project** надають API для доступу до новинних статей з тисяч джерел по всьому світу. Це незамінний інструмент для глобального моніторингу інформаційного простору, відстеження наративів, що просуваються в різних країнах, та виявлення скоординованих інформаційних кампаній.

1.2.4. Обмеження та правила гри

Використання API має свою ціну – необхідність дотримуватися правил, встановлених провайдером даних.

- **"Terms of Service" (ToS):** Кожен сервіс має документ, що регламентує умови використання його API. Ігнорування цих правил може призвести до негайного та перманентного блокування доступу.
- **Rate Limiting (Обмеження частоти запитів):** Щоб запобігти перевантаженню своїх серверів, сервіси встановлюють ліміти на кількість запитів, які можна зробити за певний проміжок часу (наприклад, не більше 100 запитів на хвилину). Скрипти повинні враховувати ці ліміти, роблячи паузи між запитами, щоб не перевищити поріг.

Використання API створює парадокс з точки зору операційної безпеки. З одного боку, це найнадійніше джерело структурованих даних. З іншого – це джерело, яке найбільше вас викриває. API-ключ є унікальним ідентифікатором. Провайдер сервісу (Telegram, Google, Twitter) точно знає, *хто* робить запит, *коли* він його робить і *про що* саме запитує. Вся ця активність логується. Для військового аналітика це становить значний ризик. Ці логі можуть бути передані розвідувальним службам країни, де розташований сервіс, або скомпрометовані хакерами. Аналізуючи патерни запитів, противник може зробити висновки про розвідувальні пріоритети (наприклад, "вони постійно запитують дані про канали, що діють у Херсонській області"). Отже, робота з API вимагає ретельного управління цифровими ідентичностями (використання неатрибутованих акаунтів для генерації ключів) та усвідомлення того, що цей канал збору є прозорим для провайдера даних. Це компроміс між якістю даних та оперативною скритністю.

1.2.5. Практичний кейс: "Створення автоматизованої системи сповіщень про активність у ворожих Telegram-каналах"

- **Постановка задачі:** Створити систему, яка в реальному часі відстежує повідомлення у списку визначених ворожих Telegram-каналів і негайно сповіщає аналітиків про появу повідомлень, що містять чутливі ключові слова.
- **Процес:**
  1. Через офіційного бота @BotFather у Telegram створюється новий бот та отримується унікальний API-ключ (токен).
  2. Створюється закритий Telegram-канал для аналітиків, куди бот буде надсилати сповіщення.
  3. Пишеться скрипт на Python з використанням бібліотеки python-telegram-bot або Telethon.
  4. Скрипт підключається до API Telegram, використовуючи отриманий ключ, і починає "слухати" нові повідомлення у цільових каналах.
  5. Кожне нове повідомлення перевіряється на наявність у ньому ключових слів зі списку (наприклад, назви конкретних підрозділів, озброєнь, прізвища командирів, географічні назви).
  6. Якщо ключове слово знайдено, скрипт миттєво формує сповіщення (включаючи текст повідомлення та посилання на оригінал) і відправляє його у закритий канал аналітиків.
- **Результат:** Аналітичний відділ отримує можливість реагувати на критично важливу інформацію не через години після її появи, а протягом кількох секунд, що значно підвищує оперативність прийняття рішень.

Таблиця 1: Порівняльний аналіз методів: Веб-скрапінг vs. API

Критерій	Веб-скрапінг	API (Application Programming Interface)
Структура даних	Неструктуровані (HTML). Потребують парсингу для вилучення даних.	Структуровані (зазвичай JSON, XML). Готові до обробки.
Стабільність	Низька. Будь-яка зміна верстки сайту може зламати скрапер.	Висока. Структура регламентована документацією і рідко змінюється.
Легітимність доступу	"Сіра зона". Часто порушує умови використання (ToS) сайту.	Офіційно дозволений та документований спосіб доступу.
Обсяг доступних даних	Потенційно все, що відображається у браузері користувача.	Тільки ті дані та в тому обсязі, які надає провайдер API.
OPSEC ризик (атрибуція)	Нижчий (при правильному маскуванні IP, User-Agent тощо).	Вищий. Ідентифікація відбувається через унікальний API-ключ.
Вимоги до навичок	Розуміння HTML/CSS, DOM, методів обходу захистів.	Розуміння HTTP, формату JSON, вміння читати технічну документацію.

Таблиця 2: OPSEC-чекліст для операцій зі збору даних в Інтернеті

Фаза операції	Пункт перевірки
Підготовка середовища	☐ Використання ізольованого середовища: віртуальна машина (VM) або контейнер (Docker).
	☐ Встановлення та постійне використання надійного VPN-сервісу або

	мережі Tor.
	☐ Створення неатрибутованих (не пов'язаних з реальною особою) акаунтів для реєстрацій.
	☐ Використання окремого, "чистого" браузерного профілю для розвідувальних завдань.
Під час виконання скрипту	☐ Ротація IP-адрес через пул надійних проксі-серверів.
	☐ Маскування User-Agent та інших заголовків для імітації реального браузера.
	☐ Встановлення випадкових затримок між запитами для імітації людської поведінки.
	☐ Дотримання лімітів запитів (rate limiting) цільового ресурсу, щоб уникнути блокування.
Робота з даними	☐ Зберігання зібраних даних на зашифрованих дисках або у зашифрованих контейнерах.
	☐ Видалення метаданих (EXIF з фото, авторство з документів) з отриманих файлів.
	☐ Використання захищених каналів зв'язку для передачі даних всередині команди.

Частина 2. Збір даних через формалізовані

документи: "Фермерство" інформації

Якщо автоматизований збір даних з Інтернету – це "полювання", то робота з формалізованими документами, такими як онлайн-форми, більше схожа на "фермерство". Замість того, щоб шукати інформацію у дикому середовищі, аналітик сам створює контрольовану систему (форму), висаджує "насіння" (питання) і збирає структурований "врожай" (відповіді) безпосередньо від джерел. Цей метод є незамінним, коли необхідну інформацію неможливо знайти у відкритих джерелах і потрібно отримати її від людей.

Розділ 2.1. Проектування ефективних форм збору даних (на прикладі Google Forms)

Якість зібраних даних напряму залежить від якості інструменту збору. Погано спроектована форма призведе до отримання неповних, неточних або упереджених даних, які не тільки не допоможуть, а й можуть ввести в оману.

2.1.1. Архітектура форми: Від мети до структури

- **Визначення мети:** Це найважливіший крок. Перед створенням першого питання необхідно чітко відповісти: яку саме інформацію потрібно зібрати? Яку гіпотезу потрібно перевірити? Яке рішення буде прийнято на основі цих даних? Чітка мета визначає структуру форми, типи питань та цільову аудиторію.
- **Типи запитань та їх призначення:** Платформи, такі як Google Forms, пропонують різноманітні типи полів, кожен з яких слугує своїй меті:
    - **Закриті питання:**
        - *Один варіант зі списку (Multiple choice):* Ідеально для питань "так/ні" або вибору однієї категорії (напр., "Тип техніки: Танк, БМП, САУ").
        - *Кілька варіантів зі списку (Checkboxes):* Дозволяє респонденту вибрати кілька варіантів (напр., "Помічені ознаки: Маскувальна сітка, тактичні знаки, додатковий захист").
        - *Спадний список (Dropdown):* Схожий на перший варіант, але економить місце, коли варіантів багато.
        - **Перевага:** Дані, отримані з закритих питань, легко стандартизуються, підраховуються та аналізуються.
    - **Відкриті питання:**

        - *Текстове поле (Short answer):* Для коротких відповідей (ім'я, назва населеного пункту, номерний знак).
        - *Абзац (Paragraph):* Для збору детальних описів, пояснень, свідчень очевидців.
        - **Перевага:** Дозволяють отримати якісну, глибоку інформацію, яку неможливо передбачити у закритих питаннях.
    - **Шкали та сітки:**
        - *Лінійна шкала (Linear scale):* Для оцінки ставлення за шкалою (напр., від 1 до 5, "Наскільки ви довіряєте окупаційній владі?").
        - *Сітка (Multiple choice grid):* Дозволяє оцінити кілька параметрів за однаковою шкалою.
    - **Спеціалізовані поля:** *Дата* та *Час* дозволяють фіксувати точні часові рамки подій, що є критично важливим для військової звітності.

2.1.2. Забезпечення якості та валідації даних

Помилки, допущені респондентами при заповненні форми, можуть значно ускладнити подальший аналіз. Вбудовані інструменти валідації допомагають мінімізувати "сміття" в даних на етапі вводу.

- **Валідація вводу:** Google Forms дозволяє встановлювати правила для відповідей. Наприклад, можна вимагати, щоб у полі "Кількість" було введено тільки число, у полі "Координати" – текст, що відповідає певному формату, а у полі "Email" – коректна електронна адреса.
- **Обов'язкові поля:** Критично важливі питання (наприклад, дата та час події, тип техніки) слід робити обов'язковими для заповнення. Це гарантує, що ви не отримаєте анонімних звітів без ключової інформації.
- **Умовна логіка ("Перейти до розділу на основі відповіді"):** Ця потужна функція дозволяє створювати динамічні, адаптивні форми. Наприклад, якщо на питання "Чи бачили ви військову техніку?" респондент відповідає "Так", його можна перенаправити до розділу з детальними питаннями про тип, кількість та напрямок руху. Якщо відповідь "Ні", цей розділ пропускається. Це робить форму коротшою та більш релевантною для кожного респондента.

2.1.3. Психологічні та безпекові аспекти

- **Уникнення упереджених питань (Bias):** Формулювання питання може несвідомо підштовхувати респондента до певної відповіді. Слід уникати емоційно забарвленої

лексики та навідних питань. Замість "Чи згодні ви, що жорстокі дії окупантів є нелюдськими?" краще поставити нейтральне "Опишіть дії представників окупаційної влади, свідком яких ви були".
- **Анонімність та довіра:** Це ключовий аспект, особливо при зборі чутливої інформації. Якщо існує ризик для респондента, форма повинна гарантувати повну анонімність (не збирати імена, email, IP-адреси). У вступі до форми необхідно чітко пояснити, хто збирає інформацію, з якою метою, і як буде забезпечена конфіденційність. У військовому контексті, наприклад, при зборі свідчень з окупованих територій, анонімність є абсолютною умовою безпеки джерел.

2.1.4. Практичний кейс: "Розробка форми для збору свідчень очевидців про переміщення військової техніки противника"

- **Мета:** Створити стандартизований інструмент для швидкого збору та систематизації даних від цивільного населення або розвідувальних груп.
- **Структура форми:**
  - **Розділ 1: Загальна інформація**
    - Дата спостереження (тип "Дата", обов'язкове).
    - Час спостереження (тип "Час", обов'язкове).
    - Населений пункт або координати (тип "Текстове поле", з валідацією формату).
  - **Розділ 2: Деталі про техніку**
    - Тип техніки (тип "Спадний список" з варіантами: Танк, БМП, БТР, САУ, РСЗВ, вантажівка, інше).
    - Якщо вибрано "інше", з'являється поле для текстового опису.
    - Кількість одиниць (тип "Текстове поле", валідація на число).
    - Напрямок руху (тип "Текстове поле").
    - Опишіть розпізнавальні знаки, якщо бачили (тип "Абзац").
  - **Розділ 3: Додаткові матеріали**
    - Поле для завантаження фото/відео (тип "Завантаження файлу").
    - **Важливе попередження:** "Увага! Фото та відео можуть містити метадані (геолокацію). Перед завантаженням переконайтесь, що ви перебуваєте в безпечному місці та використовуєте засоби для видалення метаданих".
- **Результат:** Дані збираються в єдиному форматі, що дозволяє їх швидко обробляти, наносити на карту та виявляти маршрути пересування колон противника.

Розділ 2.2. Технічна реалізація, розповсюдження та управління даними

Створення форми – це лише половина справи. Не менш важливим є вибір платформи, організація розповсюдження та подальша обробка зібраних даних.

2.2.1. Огляд платформ

- **Google Forms:** Це базовий, безкоштовний та надзвичайно простий у використанні інструмент. Його головні переваги – швидкість створення форм та автоматична інтеграція з Google Sheets, що дозволяє миттєво бачити відповіді у вигляді таблиці. Однак, використання Google Forms несе в собі значні ризики. Сам факт створення та розповсюдження форми є розвідувальною операцією, яка може бути виявлена. Форма має цифровий слід: обліковий запис Google, з якого її створено, унікальна URL-адреса, сервери, на яких зберігаються дані. Противник, моніторячи мережевий трафік, може помітити аномальну активність (кластери людей, що заходять за однією URL-адресою), що негайно викличе підозру. Зберігання чутливих військових даних на серверах комерційної компанії, що знаходиться поза юрисдикцією та контролем військових, є неприйнятним з точки зору безпеки. Це створює єдину точку відмови, компрометація якої може розкрити всю мережу респондентів.
- **Альтернативи:**
    - **Комерційні сервіси (SurveyMonkey, Typeform):** Пропонують більше можливостей для кастомізації та аналітики, але також є хмарними рішеннями з тими ж безпековими недоліками.
    - **Self-hosted рішення (LimeSurvey, KoBoToolbox):** Це програмне забезпечення з відкритим кодом, яке можна встановити на власних, захищених серверах. Це дає повний контроль над даними та безпекою, але вимагає технічних навичок для розгортання та підтримки. Для збору чутливої інформації цей варіант є єдиним прийнятним.

2.2.2. Розгортання та розповсюдження

- **Канали поширення:** Вибір каналу залежить від цільової аудиторії та рівня безпеки. Це можуть бути закриті групи в захищених месенджерах (Signal, Threema), QR-коди на друкованих матеріалах (якщо потрібне офлайн-поширення), захищені email-розсилки. Слід уникати поширення посилань у відкритих чатах та соціальних мережах.
- **Забезпечення доступу цільової аудиторії:** Іноді необхідно переконатися, що форму заповнюють лише певні люди. Для цього можна використовувати паролі,

унікальні одноразові посилання або попередню верифікацію респондентів через інші канали.

2.2.3. Експорт та попередня обробка даних

Сирі дані, зібрані через форму, рідко бувають ідеальними. Вони потребують очищення та підготовки до аналізу.

- **Робота з Google Sheets:** Відповіді з Google Forms автоматично потрапляють у зв'язану таблицю. Це дозволяє проводити базовий аналіз: сортування, фільтрацію, побудову простих діаграм та зведених таблиць безпосередньо в інтерфейсі Google Sheets.
- **Експорт у CSV:** Для більш складного аналізу дані слід експортувати у формат CSV (Comma-Separated Values). Це універсальний текстовий формат, який можна відкрити в будь-якому аналітичному інструменті.
- **Очищення даних за допомогою Python (Pandas):** Бібліотека Pandas є золотим стандартом для маніпуляцій з даними в Python. Вона дозволяє:
    - Завантажити CSV-файл у структуру даних DataFrame.
    - Виявити та видалити дублікати записів.
    - Знайти та виправити помилки форматування (наприклад, "Київ", "київ", "Kyiv" привести до єдиного "Київ").
    - Заповнити пропущені значення.
    - Перетворити дані у потрібні типи (наприклад, текст з датою у формат дати).

2.2.4. Практичний кейс: "Створення та адміністрування системи щоденних звітів від польових підрозділів"

- **Проблема:** Штаб отримує щоденні звіти від підрозділів у різних форматах (текстові повідомлення, фото таблиць, голосові повідомлення), що ускладнює їх систематизацію та аналіз.
- **Рішення:** Створюється стандартизована онлайн-форма (розгорнута на захищеному внутрішньому сервері).
- **Структура форми:** Форма містить чіткі поля: "Назва підрозділу" (спадний список), "Дата звіту", "Витрати боєприпасів (за номенклатурою)", "Стан техніки (справна/пошкоджена/знищена)", "Втрати (за категоріями)", "Ключові події за добу" (текстове поле).
- **Процес:** Командири підрозділів щовечора заповнюють форму зі своїх захищених пристроїв. Дані автоматично збираються в єдину базу даних.

- **Результат:** Штаб отримує структуровані дані в реальному часі. На основі цих даних автоматично будується "живий" дашборд, що візуалізує ситуацію: рівень боєздатності підрозділів, динаміку витрат БК, загальну картину втрат. Це дозволяє керівництву приймати більш обґрунтовані та оперативні рішення щодо логістики, ротації та планування операцій.

Висновок: Інтегрований підхід до інформаційно-аналітичного забезпечення

Розглянуті методи збору даних – автоматизоване "полювання" та структуроване "фермерство" – не є взаємовиключними. Навпаки, їхня справжня сила розкривається при інтегрованому використанні, що дозволяє створювати комплексну та багаторівневу аналітичну картину.

Синтез даних з різних джерел

Синергетичний ефект виникає тоді, коли дані з одного джерела використовуються для верифікації, доповнення або контекстуалізації даних з іншого. Наприклад, повідомлення про переміщення колони ворожої техніки, отримане через форму від очевидця (Частина 2), може бути початковим сигналом. Далі, використовуючи методи з Частини 1, аналітик може:

1. **Верифікувати:** Промоніторити місцеві Telegram-канали та форуми на предмет появи фото, відео або обговорень цієї колони.
2. **Доповнити:** Проаналізувати повідомлення в цих каналах, щоб уточнити точний маршрут, склад колони та час проходження.
3. **Контекстуалізувати:** Використовуючи API новинних агрегаторів, перевірити, чи не було офіційних заяв окупаційної влади про "навчання" або "передислокацію" в цьому районі, що може бути інформаційним прикриттям.

Такий підхід, що поєднує дані від людських джерел (HUMINT, хоч і опосередкований) та дані з відкритих джерел (OSINT), значно підвищує надійність та повноту розвідувальної інформації.

Цифровий розвідувальний цикл

Класичний розвідувальний цикл (планування -> збір -> обробка -> аналіз -> розповсюдження) залишається актуальним, але його наповнення адаптується до цифрової реальності. Методи, описані в цьому посібнику, є ключовими інструментами на етапах збору та обробки.

- **Планування:** Визначення інформаційних потреб та вибір адекватних методів (скрапінг, API, форма) і інструментів. Оцінка ризиків OPSEC.
- **Збір:** Запуск скраперів, взаємодія з API, розповсюдження форм.
- **Обробка:** Парсинг HTML, розбір JSON, очищення та валідація даних з форм. Приведення різнорідних даних до єдиного формату.
- **Аналіз та Розповсюдження:** Синтез оброблених даних, виявлення патернів, підготовка аналітичних звітів та їх доведення до осіб, що приймають рішення.

Майбутні тренди у зборі даних

Сфера цифрової розвідки розвивається надзвичайно стрімко. Аналітикам необхідно постійно відстежувати нові технології та адаптувати свої методи.

- **Роль ШІ та великих мовних моделей (LLM):** Інструменти на кшталт GPT-4 вже сьогодні можуть використовуватися для автоматичного узагальнення (сумаризації) великих обсягів текстових даних, зібраних за допомогою скрапінгу, класифікації повідомлень за темами, аналізу тональності (sentiment analysis) суспільних настроїв. У майбутньому їх роль в обробці неструктурованих даних буде лише зростати.
- **Аналіз зображень та відео (IMINT/VIDINT):** OSINT виходить за межі тексту. Розвиток інструментів для автоматичної геолокації за фотографіями, аналізу супутникових знімків (доступ до яких стає все більш демократичним), розпізнавання об'єктів на відео відкриває нові горизонти для збору доказової бази.
- **Етичні та юридичні виклики:** Законодавство щодо захисту персональних даних (напр., GDPR) стає все більш суворим. Хоча військові операції часто мають свої правові рамки, аналітикам необхідно розуміти юридичні та етичні межі своєї діяльності, особливо при роботі з даними цивільних осіб.

Здатність ефективно збирати, обробляти та аналізувати дані з цифрових джерел є критично важливою компетенцією для сучасних збройних сил. Опанування розглянутих методів дозволить аналітичним підрозділам забезпечувати командування якісною, своєчасною та перевіреною інформацією, що є фундаментом для переваги на полі бою XXI століття.