НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**

(підпис)

«19» червня 2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему:

Аналіз кібератак та їх наслідків на основі інформації з соціальних медіа

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-12
Мосейко Олег Романович

(підпис)

Керівник **Завідувач кафедри інформаційної безпеки, доктор технічних наук, професор Ланде Дмитро Володимирович**

(підпис)

Рецензент **доцент кафедри ММЗІ НН ФТІ «КПІ ім. Ігоря Сікорського», к.т.н., доцент Кучинська Наталія Вікторівна**
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«19» червня 2025 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Мосейку Олегу Романовичу

1. Тема роботи
Аналіз кібератак та їх наслідків на основі інформації з соціальних медіа ,
керівник роботи
Доктор технічних наук, професор Ланде Дмитро Володимирович,
затверджені наказом по університету від « 26 » травня 2025 р. № 1761- с

2. Термін подання роботи здобувачем вищої освіти « 12 » червня 2025 р.

3. Вихідні дані до роботи
Наукові праці з OSINT-аналізу, обробки природної мови, цифрової обробки сигналів та машинного навчання, відкриті дані з соціальних медіа (YouTube, Telegram, Reddit), Python із бібліотеками pywt, numpy, nltk, matplotlib

4. Зміст роботи
Опис мети, об’єкта, предмета та завдань дослідження, а також його актуальності в контексті кібербезпеки; аналіз ключових викликів, таких як шум, достовірність, упередженість, технічні труднощі, етичні та юридичні аспекти, а також вплив дезінформації; огляд особливостей кожної платформи як джерела даних про кібератаки; опис методів автоматизації через API, web-scraping та гібридних підходів; огляд методів фільтрації шуму, верифікації та кластеризації даних; застосування wavelet-аналізу та

методу Сорнетте для декомпозиції сигналів і прогнозування критичних точок; розробка та тестування інструменту, включаючи архітектуру, модулі, результати кейс-стаді та візуалізацію з використанням Gephi.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Дата видачі завдання 31 січня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Визначення теми та вектору аналізу	27.12.2024 – 30.01.2025	Виконано
2	Визначення завдання на роботу	31.01.2025 – 05.02.2025	Виконано
3	Аналіз існуючих проблем збору даних	06.02.2025 – 20.02.2025	Виконано
4	Визначення платформ для аналізу	20.02.2025 – 06.03.2025	Виконано
5	Ознайомлення з різними методами збору та обробки даних	07.03.2025 – 05.04.2025	Виконано
6	Дослідження методів цифрової обробки сигналів	05.04.2025 – 21.04.2025	Виконано
7	Створення інструменту аналізу	22.04.2025 – 18.05.2025	Виконано
8	Написання перших двох розділів дипломної роботи	19.05.2025 – 23.05.2025	Виконано
9	Написання розділів 3 – 5 дипломної роботи	24.05.2025 – 30.05.2025	Виконано
10	Написання розділів 6-7 дипломної роботи	31.05.2025 – 02.06.2025	Виконано
11	Оформлення роботи відповідно вимогам	03.06.2025 – 05.06.2025	Виконано
12	Створення презентації роботи	06.06.2025 - 11.06.2025	Виконано
13	Передзахист	12.06.2025	Виконано
14	Захист	19.06.2025	

Здобувач вищої освіти _____ Олег Мосейко
(підпис)

Керівник роботи _____ Дмитро Ланде

РЕФЕРАТ

Дана робота містить 76 сторінок, 25 ілюстрацій, 1 таблицю, 14 джерел за переліком посилань.

Метою дослідження є збір та обробка інформації про кібератаки із соціальних медіа, що дозволить швидко визначати рівень загрози нових інцидентів і приймати обґрунтовані рішення щодо реагування.

Об’єкт дослідження – інформація про кібератаки, що публікується у соціальних медіа.

Предмет дослідження – методи аналізу та оцінки серйозності кібератак на основі відкритої інформації.

Методи дослідження: аналіз наукової літератури та відкритих джерел за даною темою, OSINT-аналіз, обробка тексту/відео/графіків, побудова часових ліній активностей, аналіз цифрових слідів, застосування методів цифрової обробки сигналів та інструментів машинного навчання

В результаті роботи було запропоновано

Ключові слова: кібератаки, соціальні медіа, OSINT, цифрові сліди,обробка природної мови (NLP), машинне навчання.

ABSTRACT

This work contains 76 pages, 25 illustrations, 1 table, 14 sources according to the list of references.

The purpose of the study is to collect and process information about cyberattacks from social media, which will allow you to quickly determine the level of threat of new incidents and make informed decisions about response.

The object of the study is information about cyberattacks published in social media.

The subject of the study is methods of analyzing and assessing the severity of cyberattacks based on open information.

Research methods: analysis of scientific literature and open sources on this topic, OSINT analysis, text/video/graphics processing, construction of activity timelines, analysis of digital traces, application of digital signal processing methods and machine learning tools

As a result of the work, the following

Keywords: cyberattacks, social media, OSINT, digital traces, natural language processing (NLP), machine learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Application Programming Interface (програмний інтерфейс прикладного програмування)
CWT	Continuous Wavelet Transform (неперервне вейвлет-перетворення)
DBSCAN	Density-Based Spatial Clustering of Applications with Noise (кластеризація на основі густини з шумом)
DDoS	Distributed Denial of Service (розподілена атака типу "відмова в обслуговуванні")
GDPR	General Data Protection Regulation (загальний регламент про захист даних)
LPPLS	Log-Periodic Power Law Singularity (лог-періодична степенева поведінка)
NLP	Natural Language Processing (обробка природної мови)
OSINT	Open-Source Intelligence (розвідка на основі відкритих джерел)
WEB-scraping	Веб-скрейпінг (автоматичне збирання даних із веб-сторінок)

ВСТУП

У період гібридної війни, дезінформації та активного застосування інформаційно-психологічних операцій (ІПСО), події на кшталт пожеж, терактів чи обстрілів миттєво з’являються в інформаційному просторі. Їх серйозність зазвичай оцінюється за масштабами наслідків — це допомагає швидко реагувати та запобігати повторенню подібних інцидентів.

Аналогічно і в кіберпросторі — розвиток технологій приніс не лише зручності, а й нові загрози. Кібератаки можуть спричинити серйозні наслідки для інфраструктури, економіки та безпеки. Зі зростанням ролі соціальних медіа, інформація про такі атаки стала доступною широкому загалу.

Уміння своєчасно аналізувати відкриті джерела та оцінювати потенційну небезпеку кібератак стає важливою складовою кіберзахисту.

Метою дослідження є збір та обробка інформації про кібератаки із соціальних медіа, що дозволить швидко визначати рівень загрози нових інцидентів і приймати обґрунтовані рішення щодо реагування.

Об’єктом дослідження є інформація про кібератаки, що публікується у соціальних медіа.

Предметом дослідження є методи аналізу та оцінки серйозності кібератак на основі відкритої інформації.

Завдання дослідження:

1. Проаналізувати можливості соціальних медіа як джерела даних про кіберінциденти.
2. Розглянути сучасні методи OSINT-аналізу, обробки тексту та мультимедійної інформації.

3. Розробити інструмент для виявлення та оцінки серйозності кібератак.
4. Провести практичне дослідження з використанням обраного інструменту та зробити висновки щодо його ефективності.

Методи дослідження: аналіз наукової літератури та відкритих джерел за даною темою, OSINT-аналіз, обробка тексту/відео/графіків, побудова часових ліній активностей, аналіз цифрових слідів, застосування методів цифрової обробки сигналів та інструментів машинного навчання.

1 Загальні проблеми аналізу даних з соціальних медіа

У сучасному цифровому світі соціальні медіа стали невід'ємною частиною інформаційного простору, надаючи безпрецедентні можливості для збору даних у реальному часі. Вони генерують величезні обсяги контенту, який може бути використаний для аналізу кіберінцидентів — подій, що загрожують інформаційній безпеці організацій, урядів і окремих осіб. Однак використання соціальних медіа як джерел інформації стикається з численними проблемами, які ускладнюють їхнє застосування в контексті кіберрозвідки. Ці проблеми включають неоднорідність даних, їхню низьку достовірність, упередженість, технічні обмеження та етичні дилеми, що потребують ретельного аналізу для оцінки їхнього впливу на ефективність моніторингу кіберзагроз.

1.1 Проблеми шуму

Висока ймовірність шуму в даних є центральною проблемою соціальних медіа. Користувачі публікують різноманітний контент, включаючи нерелевантні чи випадкові повідомлення, що ускладнює виділення ключової інформації. Цей шум може включати коментарі, які не стосуються інциденту, або повторювані дані, які потребують складних методів фільтрації. [1]

1.2 Проблеми достовірності та верифікації

Достовірність інформації в соціальних медіа є серйозним викликом через відсутність механізмів офіційного контролю. Дані часто надходять від анонімних або неперевірених джерел, що ставить під сумнів їхню точність. Наприклад, технічні деталі, які з'являються на ранніх етапах

інциденту, можуть бути неповними або хибними, що ускладнює їхнє використання для оперативного реагування. Верифікація таких даних потребує звернення до офіційних джерел, що суперечить меті використання соціальних медіа для швидкого доступу до інформації.

Швидкість поширення інформації в соціальних медіа також перевищує можливості її перевірки. У перші години після інциденту користувачі можуть поширювати чутки або припущення, які швидко набирають популярності, але не відповідають дійсності. Це призводить до накопичення не підтверджених фактів, що може вплинути на прийняття рішень і спотворити загальну картину інциденту. [2, 3]

1.3 Проблеми упередженості та репрезентативності

Упередженість є значним бар'єром у використанні соціальних медіа. Активність користувачів залежить від їхніх інтересів, що призводить до нерівномірного висвітлення інцидентів. Події з великими економічними чи соціальними наслідками привертають більше уваги, тоді як менш помітні загрози залишаються поза фокусом. Ця вибірковість ускладнює створення повної картини кіберінцидентів і може спотворювати аналіз поширеності чи впливу.

Демографічні та географічні особливості аудиторії додають додатковий рівень упередженості. На платформах можуть домінувати користувачі з певних регіонів, що обмежує репрезентативність даних для глобального аналізу. Це може призвести до перекосу, коли певні аспекти інциденту недооцінюються через низьку активність відповідних груп користувачів. [3]

1.4 Технічні та методологічні труднощі

Збір і обробка даних із соціальних медіа стикаються з технічними проблемами, пов’язаними з доступом до платформ. Обмеження API ускладнюють автоматизований збір даних, особливо коли доступ до інформації обмежено через приватні канали чи сабреддіти.

Обробка великих обсягів даних потребує складних алгоритмів для видобутку ключових показників, таких як тематичні кластери чи часові закономірності. Це вимагає значних обчислювальних ресурсів і спеціалізованого програмного забезпечення, що може бути недоступним для невеликих дослідницьких груп. Методологічні труднощі включають вибір моделей аналізу, які могли б ефективно впоратися з шумом і упередженнями, що є складним завданням у динамічному середовищі соціальних мереж. [1, 2]

1.5 Вплив дезінформації на аналіз кіберінцидентів

Дезінформація є особливо гострою проблемою в контексті використання соціальних медіа для аналізу кіберінцидентів. Відкрита природа платформ дозволяє користувачам швидко поширювати хибні чи перекручені дані, які можуть включати неправдиві технічні деталі, перебільшені оцінки масштабів атаки або навіть навмисні фейки. У перші години після інциденту можуть з’являтися повідомлення про нібито відповідальних осіб чи методи атаки, які не мають підтвердження, але набувають популярності через вірусний ефект.

Цей вплив дезінформації ускладнює реагування на кіберзагрози. Організації, які покладаються на соціальні медіа для оперативного моніторингу, можуть приймати рішення на основі неточних даних, що призводить до неефективного розподілу ресурсів або втрачених

можливостей для захисту. Крім того, дезінформація може викликати паніку серед громадськості, погіршуючи ситуацію. Перебільшення впливу інциденту може призвести до економічних втрат через масові відмови від послуг уражених компаній. [2]

Ще одним аспектом є навмисне поширення дезінформації зловмисниками. Хакерські групи чи конкуруючі організації можуть використовувати соціальні медіа для маніпуляції сприйняттям інциденту, відволікаючи увагу від реальних загроз або створюючи хибні сліди. Це вимагає від дослідників не лише виявлення дезінформації, але й аналізу її джерел і мотивів, що значно ускладнює процес. [2]

1.6 Додаткові виклики в аналізі даних із соціальних медіа

Сучасні тенденції, такі як використання штучного інтелекту для генерації контенту (наприклад, ChatGPT, DALL·E), додають нові виклики. Соціальні медіа можуть містити синтетичні тексти чи зображення, які важко відрізнити від автентичних. У 2025 році зафіксовано випадки поширення фейкових відео про кібератаки, створених AI, у Telegram-каналах, що ускладнює верифікацію.

Ще одним аспектом є швидке старіння даних. Інформація, опублікована в перші години після інциденту, може бути застарілою через оновлення з офіційних джерел. Це вимагає динамічної адаптації методів аналізу, що потребує додаткових ресурсів. [4]

Економічний вплив аналізу також вартий уваги. Компанії, які інвестують у моніторинг соціальних медіа, стикаються з витратами на програмне забезпечення та навчання персоналу, що може сягати $50,000 на рік для середнього підприємства. [4]

Висновки до розділу 1

Ці проблеми створюють значні перешкоди для інтеграції соціальних медіа в стратегії кіберрозвідки. Низька достовірність і шум у даних ускладнюють оперативне реагування, тоді як упередженість і технічні обмеження перешкоджають повному охопленню інцидентів. Етичні та юридичні аспекти додають додатковий рівень складності, вимагаючи балансування між ефективністю аналізу та дотриманням прав користувачів. Таким чином, описані проблеми підкреслюють необхідність розробки спеціалізованих підходів для подолання цих бар’єрів, що буде розглянуто в наступних частинах роботи.

2 Використання Youtube, Telegram та Reddit як джерел інформації

2.1 YouTube

YouTube, як одна з найбільших платформ для поширення відеоконтенту, приваблює широку аудиторію, включаючи експертів із кібербезпеки, журналістів і пересічних користувачів. Це робить її потенційно цінним джерелом для аналізу кіберінцидентів, адже відео можуть містити пояснення, інтерв'ю чи аналітичні огляди. Проте використання YouTube стикається з проблемою неоднорідності контенту. Відсутність редакційних стандартів означає, що платформа поєднує високоякісні матеріали від професіоналів із суб'єктивними або неперевіреними відео, створеними аматорами. Це ускладнює відбір достовірної інформації, оскільки аналітичний контент може бути змішаним із розважальними чи популістськими матеріалами, що не мають наукової цінності. [5]

Для збору даних з Youtube, було використано можливості Youtube API (рис. 2.1). На початку програма підключається до сервера YouTube через API. Використовується унікальний ключ аутентифікації, код ідентифікує користувача та отримує дозвіл на виконання запитів. Це необхідно, щоб забезпечити безпечний і контрольований доступ до ресурсів платформи.

Далі задаються критерії пошуку, такі як ключові слова, тип контенту та спосіб сортування (за датою). Це дозволяє звузити обсяг інформації до релевантних результатів і організувати їх у зручному порядку. Такі налаштування допомагають зосередитись на потрібному контенті, економлячи ресурси та час. Потім запускається процес взаємодії з API.

Отримана інформація містить метадані про відео, такі як назви, описи та ідентифікатори.

Рисунок 2.1 – Алгоритм збору даних через Youtube API

Код на рисунку 2.2 демонструє функцію collect_youtube_data для збору метаданих відео з YouTube через API. Функція приймає пошуковий

запит і повертає список із заголовками, датами публікації та URL-адресами відео.

```
SEARCH_QUERY = "Solarwinds"

async def collect_youtube_data():
    try:
        youtube = build("youtube", "v3", developerKey=YOUTUBE_API_KEY)
        request = youtube.search().list(
            part="snippet",
            q=SEARCH_QUERY,
            type="video",
            maxResults=10,
            order="date"
        )
        response = request.execute()

        results = []
        for item in response.get("items", []):
            results.append({
                "platform": "YouTube",
                "title": item["snippet"]["title"],
                "description": item["snippet"]["description"],
                "published_at": item["snippet"]["publishedAt"],
                "url": f"https://www.youtube.com/watch?v={item['id']['videoId']}
            })
        return results
    except Exception as e:
        print(f"Error collecting YouTube data: {e}")
        return []
```

Рисунок 2.2 – Функція збору даних через Youtube API

2.2. Telegram

Telegram, як платформа для обміну повідомленнями, активно використовується спільнотою кібербезпеки для швидкого поширення інформації. Публічні канали та групи дозволяють обмінюватися технічними деталями, аналізами та навіть зразками зловмисного коду. Проте використання Telegram як джерела інформації стикається з проблемою фрагментації даних. Інформація на платформі часто представлена у вигляді коротких повідомлень, які можуть бути частиною ширшої дискусії, але не дають повної картини. Це ускладнює

систематизацію даних і потребує додаткових зусиль для їхньої консолідації.

Достовірність інформації в Telegram є ще одним значним викликом. Через відсутність централізованого контролю та анонімність користувачів платформа дозволяє поширення як точних, так і хибних даних. Приватний характер деяких каналів також створює проблему доступу. Хоча публічні канали надають відкритий доступ до інформації, закриті групи можуть містити цінні дані, які недоступні для широкого загалу. Це створює нерівномірність у зібраних даних, що може вплинути на повноту аналізу та призвести до упередженого сприйняття інциденту.

Для збору даних з Telegram , було використано можливості Telegram API (рис. 2.3).

Підключення до Telegram API здійснюється за допомогою клієнта, який авторизується за допомогою API ID і API HASH. Це необхідно для ідентифікації програми та отримання дозволу на доступ до публічних каналів, забезпечуючи безпечну взаємодію з платформою. Далі задаються список імен каналів, період часу (початкова та кінцева дати) і ключове слово для пошуку. Це дозволяє відфільтрувати релевантні повідомлення та обмежити обсяг даних, що збираються, роблячи процес цільовим і ефективним. Виконується пошук і витягування повідомлень із кожного каналу в межах заданого часу та з урахуванням пошукового запиту. Ітерація дозволяє обробляти кожен канал послідовно.

Код на рисунку 2.4 ілюструє функцію get_telegram_messages для збору повідомлень із Telegram-каналів. Функція фільтрує повідомлення за датами та ключовим словом, повертаючи список із датами та текстом, що дозволяє аналізувати активність, пов’язану з кібератаками.

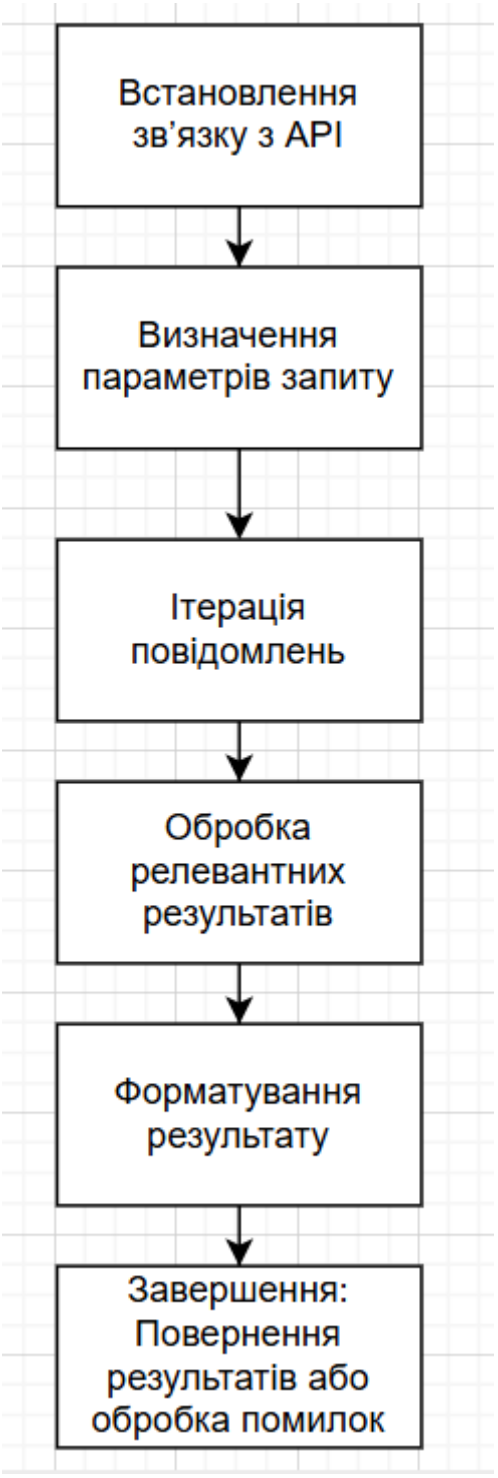

Рисунок 2.3 – Алгоритм збору даних через Telegram API

```python
async def get_telegram_messages(channel_usernames, start_date, end_date, search_query):
    """Fetch relevant messages from multiple Telegram channels."""
    messages = []
    try:
        async with TelegramClient('session', TELEGRAM_API_ID, TELEGRAM_API_HASH) as client:
            await client.start()
            # Process each channel in the list
            for channel_username in channel_usernames:
                print(f"Fetching messages from Telegram channel: {channel_username}")
                async for message in client.iter_messages(channel_username, limit=3000):
                    if not message.date:
                        continue
                    msg_date = message.date.date()
                    if start_date <= msg_date <= end_date and message.text and search_query.lower() in message.text.lower():
                        messages.append({
                            "date": msg_date,
                            "text": message.text
                        })
        print(f"Fetched {len(messages)} Telegram messages:")
        for msg in messages[:5]:
            print(f"{msg['date']}: {msg['text'][:50]}...")
    except Exception as e:
        print(f"Error fetching Telegram messages: {e}")
    return messages
```

Рисунок 2.4 – Функція збору даних через Telegram API

2.3 Reddit

Reddit, як платформа для спільнот, надає унікальну можливість аналізувати громадські настрої та обмінюватися інформацією про кіберінциденти. Сабреддіти, присвячені кібербезпеці, можуть містити дописи з посиланнями на офіційні звіти, думки експертів і дискусії користувачів. Проте Reddit стикається з проблемою упередженості даних. Популярність тем залежить від інтересів спільноти, що може призводити до надмірної уваги до резонансних інцидентів і ігнорування менш помітних подій. Це створює перекоси в зібраній інформації, ускладнюючи її репрезентативність для повного аналізу кіберзагроз. Різноманітність думок і коментарів на Reddit також додає шум. Користувачі часто діляться суб’єктивними поглядами, чутками або не перевіреними фактами, що ускладнює відбір достовірної інформації.

Для збору даних з Reddit , було використано можливості Reddit API (рис. 2.5).

Підключення до Reddit API створюється за допомогою асинхронного клієнта, який використовує ідентифікатор клієнта (Client

ID), секретний ключ (Client Secret) і агент користувача (User Agent). Це необхідно для автентифікації програми та забезпечення дозволу на доступ до публічних даних субреддітів. Далі задаються цільові сабреддіти, період часу і ключове слово для пошуку. Це дозволяє відфільтрувати релевантні пости та обмежити обсяг даних.

Пошук постів виконується у заданих спільнотах із сортуванням за часом ("new") і фільтрацією за всіма доступними даними ("all"), обмежуючи кількість до 200 результатів за запитом. Повторні спроби забезпечують надійність у разі збоїв, а ітерація дозволяє обробляти кожен запит послідовно.

Код на рисунку 2.6 ілюструє функцію get_reddit_posts для збору постів у спільнотах Reddit.

2.4 Використання web-scraping для розширення доступу

У випадках, коли API має обмеження (обмеження кількості запитів або доступу до приватних даних), можна застосовувати методи web-scraping. Цей підхід передбачає автоматичне завантаження веб-сторінок і витягування інформації з них за допомогою спеціалізованих бібліотек. Web-scraping дозволяє отримувати дані з платформ, де API недоступний, або з відкритих сторінок, таких як сабреддіти на Reddit чи коментарі під відео на YouTube. [4]

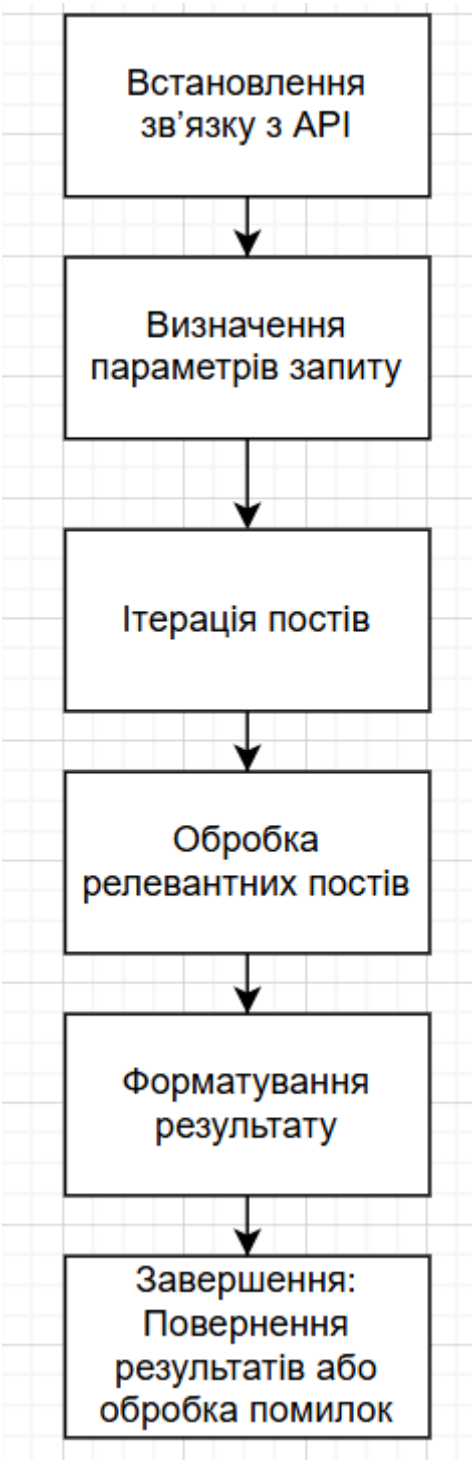

Рисунок 2.5 – Алгоритм збору даних через Reddit API

```python
async def get_reddit_posts(search_query, start_date, end_date):
    posts = []
    for attempt in range(3):
        try:
            async with asyncpraw.Reddit(
                client_id=REDDIT_CLIENT_ID,
                client_secret=REDDIT_CLIENT_SECRET,
                user_agent=REDDIT_USER_AGENT
            ) as reddit:
                subreddits = await reddit.subreddit("cybersecurity+netsec+technology+hacking+worldnews")
                async for submission in subreddits.search(search_query, sort="new", time_filter="all", limit=200):
                    created_at = datetime.fromtimestamp(submission.created_utc, tz=timezone.utc).date()
                    text = (submission.title + " " + (submission.selftext or "")).lower()
                    if start_date <= created_at <= end_date and search_query.lower() in text:
                        posts.append({
                            "date": created_at,
                            "text": submission.title
                        })
            break
        except Exception as e:
            print(f"Reddit fetch retry {attempt + 1}/3 failed: {e}")
            await asyncio.sleep(2)
    print(f"Fetched {len(posts)} Reddit posts:")
    for post in posts[:5]:
        print(f"{post['date']}: {post['text'][:50]}...")
    return posts
```

Рисунок 2.6 – Функція збору даних через Reddit API

2.5 Збір даних

Для збору даних застосовується googleapiclient.discovery для YouTube (збір метаданих відео за ключовими словами), telethon для Telegram (з'єднання з каналами) і asyncpraw для Reddit (сканування субреддітів). Ліміт 10 результатів на платформу було обрано для оптимізації продуктивності, хоча більший ліміт не тестувався. Web-scraping через бібліотеку requests, дозволяє витягувати коментарі чи дописи, недоступні через API, із динамічним налаштуванням діапазону (±90 днів).

Код на рисунку 2.7 ілюструє модуль збору даних, який одночасно отримує інформацію з YouTube, Telegram і Reddit за допомогою асинхронних викликів. Функція main об'єднує результати для подальшої обробки.

```
async def main():
    youtube_data, telegram_data, reddit_data = await asyncio.gather(
        collect_youtube_data(),
        collect_telegram_data(),
        collect_reddit_data()
    )

    all_results = youtube_data + telegram_data + reddit_data

    print(json.dumps(all_results, indent=2))
```

Рисунок 2.7 – Модуль збору даних

2.6 Обробка даних

Для початку треба перетворити сирі дані в зручну для використання форму. Дані з API обробляються для вилучення ключової інформації (заголовків, дат публікації, посилань) і збереження їх у структурованому вигляді(у списку). Це необхідно для подальшого аналізу чи візуалізації.

Повідомлення перевіряються на відповідність заданому діапазону дат і наявність пошукового запиту в тексті. Це забезпечує, що до результатів потрапляють лише ті дані, які відповідають критеріям.

Аналіз зібраних даних було проведено за допомогою моделі Grok3, яка працювала за таким алгоритмом [6].

Щоб ідентифікувати ключові сутності використано модель spaCy. Для визначення документів найбільш пов’язаних з темою застосовувались трансформерні моделі типу BERT. Це включає аналіз ключових слів і контексту. Далі дані групуються за датами [7, 8].

Алгоритм ранжує документи за кількістю ідентифікованих сутностей, релевантністю теми та розподілом за часом. Тексти об’єднуються в структурований дайджест, де кожен запис супроводжується коротким резюме та посиланням на джерело.

Висновки до розділу 2

Використання платформ соціальних медіа, таких як YouTube, Telegram і Reddit, для аналізу кіберінцидентів має значний потенціал завдяки швидкості поширення інформації та різноманітності контенту, однак стикається з низкою специфічних проблем.

Автоматизація збору даних через API (YouTube, Telegram, Reddit) та web-scraping забезпечує гнучкий доступ до інформації. Застосування асинхронних викликів і фільтрація за ключовими словами та датами підвищує ефективність збору, а попередня обробка з використанням NLP-технологій дозволяє зменшити шум і структурувати дані. Таким чином, гібридний підхід до збору та аналізу даних із цих платформ є перспективним для створення достовірних і повних дайджестів про кібератаки, однак потребує подальшого вдосконалення методів верифікації та адаптації до динаміки соціальних мереж.

3 Аналіз та прогнозування часових рядів

3.1 Часові ряди

Часовий ряд – це набір значень, що спостерігаються, упорядкований за часом. Далі розглядатимуться дискретні часові ряди, значення яких фіксувалися через рівні проміжки часу. Будемо позначати такий часовий ряд $x_1, x_2, \dots, x_T$ або коротко $\{x_t\}_{t=1}^{T}$ маючи на увазі, що фіксування значень ряду відбувалося через рівний проміжок часу.

Якщо значення часового ряду однозначно задаються деяким математичним співвідношенням, то такий ряд є детермінованим. Якщо значення часового ряду можна описати тільки в термінах імовірнісного розподілу, то йдеться про статистичний часовий ряд. Такі ряди і розглядатимуться далі.

У нашому дослідженні корисно розглянути більш гладку версію вихідного часового ряду. Згладжування допомагає виявити суттєві тенденції в динаміці ряду, приховавши при цьому шум і різні особливості, які проявляються за невеликих масштабів.

Найбільш простий спосіб згладжування - це обчислення ковзного середнього. Просте ковзне середнє дорівнює середньому арифметичному значення елементів ряду з інтервалу заданої довжини, а саме

$$SMA_t = \frac{1}{\omega} \sum_{i=0}^{\omega-1} x_{t-i}, \qquad (3.1)$$

де ω – ширина інтервалу (кількість елементів, якими розраховується середнє), SMA_t – значення простого ковзного середнього в точці t [11].

За допомогою коду на рисунку 3.1 було реалізовано згладжування часового ряду для подальшого аналізу

```
data_deque = deque(maxlen=window_size)
for i in range(len(dates)):
    data_deque.append(raw_data[i] if i < len(raw_data) else 0)
    if i >= window_size - 1:
        smoothed_data.append(sum(data_deque) / window_size)
    else:
        smoothed_data.append(sum(data_deque) / (i + 1))
smoothed_data = smoothed_data[:len(dates)]
```

Рисунок 3.1 Код для згладжування часового ряду

На рисунку 3.2 зображено часовий ряд, вихідний (синій) та гладкий (помаранчевий).

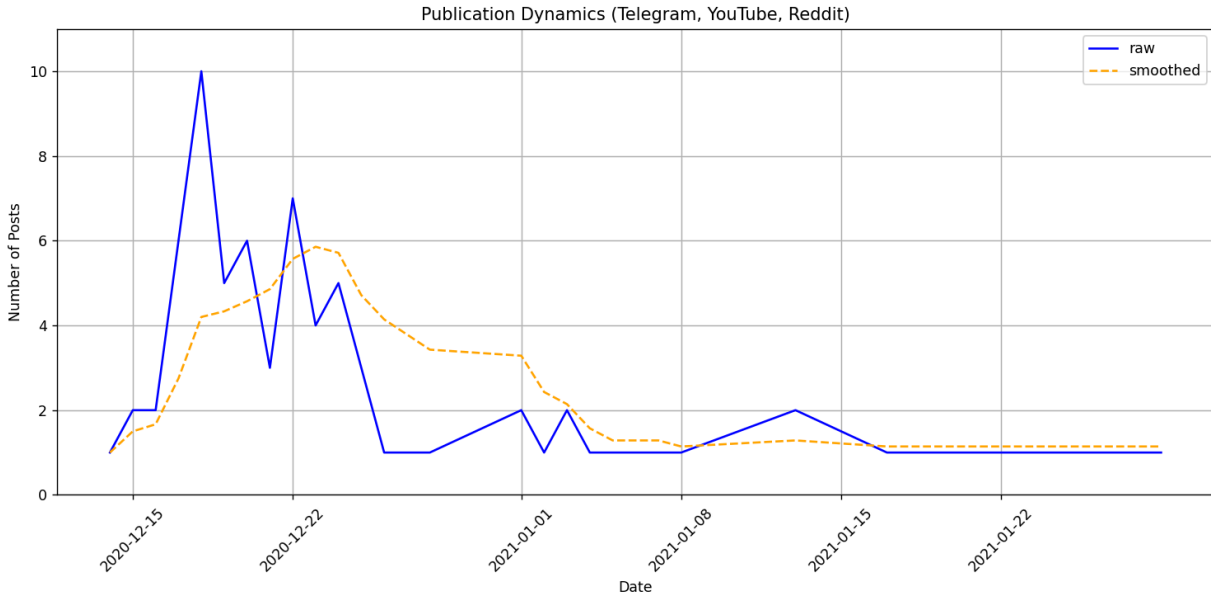

Рисунок 3.2 – Часовий ряд, вихідний та згладжений

3.2 Використання wavelet-аналізу для декомпозиції сигналів

Wavelet-аналіз є одним з методів обробки часових рядів. Цей метод дозволяє розкласти сигнал на частотні компоненти з одночасним збереженням часової локалізації. Цей метод було обрано через його здатність адаптуватися до змінної природи даних соціальних медіа, де активність може бути хаотичною, а тренди — нестабільними. На відміну

від класичного аналізу Фур'є, який припускає стаціонарність сигналу і втрачає часову інформацію, wavelet-аналіз забезпечує локальний аналіз частотних компонентів, що є критично важливим для виявлення аномалій у реальному часі. Різке зростання кількості повідомлень про атаку може вказувати на її ескалацію, а wavelet-аналіз допомагає відокремити цей сигнал від фонового шуму. Цей метод є особливо ефективним для сигналів із різними масштабами змін, що ідеально відповідає динаміці соціальних мереж, де активність варіюється від хвилинних сплесків до тижневих трендів [10].

Ця гнучкість дозволяє адаптувати аналіз до конкретних потреб дослідження, виявляючи як короткострокові сплески (реакція на перші повідомлення про атаку), так і довгострокові закономірності (еволюція інтересу до інциденту). Така адаптивність є ключовою для прогнозування критичних моментів, таких як піки активності, які можуть свідчити про офіційне підтвердження атаки чи масовий витік даних. Крім того, wavelet-аналіз дозволяє обробляти нерівномірні дані, що часто виникають у соціальних медіа через перерви в активності чи обмеження API, що робить його практичним інструментом для нашого дослідження.

Wavelet-трансформація представлена як згортка вхідного сигналу f(t) з вейвлет-функцією ψ:

$$W(a,\ b)\ =\ \int f(t)\ \cdot\ \psi^{*}\left(\frac{t-b}{a}\right) dt, \qquad (3.2)$$

де a — масштаб (інверсія частоти), b — зсув по часу, ψ — вейвлет-функція, *— комплексне спряження [11].

Ця трансформація, відома як неперервна вейвлет-трансформація (CWT), дозволяє розкласти сигнал на компоненти різної частоти та часу.

Вибір вейвлета залежить від характеристик сигналу та цілей аналізу.Найчастіше на практиці застосовуються такі (рис. 3.3) [11]:

(а) Гаусова хвиля (перша похідна гаусової функції)

Гаусова хвиля, як перша похідна гаусової функції, задається як:

$$\psi(t) = -te^{-t^2/2} \tag{3.3}$$

Дана хвиля може бути корисною для виявлення тенденцій у зміні активності, оскільки вона чутлива до нахилу кривої. Однак вона менш ефективна для різких сплесків, ніж інші вейвлети.

(б) Mexican Hat

Mexican Hat є другою похідною гаусової функції і визначається як:

$$\psi(t) = (1 - t^2)e^{-t^2/2} \tag{3.4}$$

Mexican Hat використовується для ідентифікації сплесків активності, оскільки вона чітко виділяє локальні максимуми. Її симетрія робить її придатною для аналізу симетричних або близьких до симетричних аномалій.

(в) Вейвлет Хаара

Вейвлет Хаара є найпростішим вейвлетом і визначається як:

$$\psi(t) = \begin{cases} 1, & 0 \le t < 0.5, \\ -1, & 0.5 \le t < 1, \\ 0, & \text{інакше.} \end{cases} \tag{3.5}$$

Вейвлет Хаара ідеально підходить для виявлення різких змін. Однак через свою простоту він менш ефективний для аналізу плавних трендів або складних сигналів.

(г) Вейвлет Морле

Вейвлет Морле є комплексною функцією, але його дійсна частина визначається як:

$$\psi(t) = cos(\omega_0 t) e^{-t^2/2}, \quad (3.6)$$

де ω_0 — центральна частота.

У нашому випадку "Mexican Hat" обрано через його здатність чітко виділяти локальні екстремуми**.** Вейвлет "Mexican Hat" є реальною функцією, яка має позитивну і негативну частини, що дозволяє виявляти як позитивні, так і негативні відхилення від базового рівня. Його форма нагадує перевернуту дзвіницю з двома "крилами", що робить його чутливим до локальних змін у сигналі. "Mexican Hat" є ефективним для аналізу сигналів із локалізованими особливостями, оскільки його спектральна щільність зосереджена навколо нуля, що забезпечує хорошу роздільну здатність у часі [12].

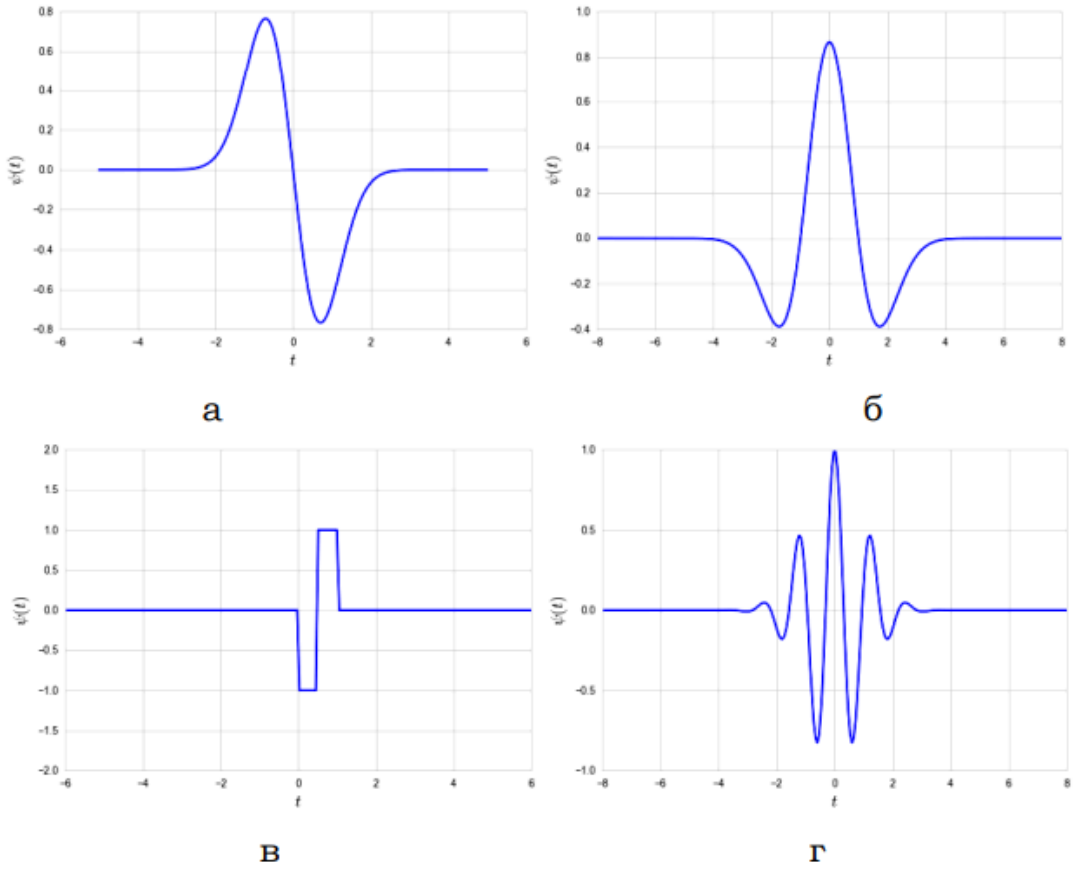

Рисунок 3.3 – Приклади вейвлетів: (а) гаусова хвиля (перша похідна гаусової функції), (б) Mexican Hat, (в) вейвлет Хаара; (г) вейвлет Морле (дійсна частина).

У коді (рис. 3.4) реалізовано CWT із "Mexican Hat" для створення скалограми, яка допомагає візуально ідентифікувати ці компоненти. Метод дозволяє рекурсивно розкладати сигнал на рівні деталізації, що є зручним для аналізу великих наборів даних.

```
post_counts = np.zeros(len(dates))
for i, date in enumerate(dates):
    date_str = date.strftime('%Y-%m-%d')
    if date_str in post_counts_dict:
        post_counts[i] = post_counts_dict[date_str]

scales = np.arange(1, 20)
wavelet = 'mexh'
coeffs, freqs = pywt.cwt(post_counts, scales, wavelet)

plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)
plt.plot(dates, post_counts, color='blue')
plt.title('Post Counts Over Time')
plt.xlabel('Date')
plt.ylabel('Post Count')
plt.grid(True)
plt.xticks(rotation=45)

plt.subplot(2, 1, 2)
plt.imshow(np.abs(coeffs), extent=[0, len(post_counts), scales[-1], scales[0]], cmap='jet', aspect='auto')
plt.colorbar(label='Magnitude')
plt.title('Scalogram (CWT) with Mexican Hat Wavelet')
plt.xlabel('Time (Days since May 8, 2021)')
plt.ylabel('Scale')
plt.xticks(ticks=np.arange(0, len(dates), 7), labels=[dates[i].strftime('%Y-%m-%d') for i in range(0, len(dates), 7)], rotation=45)

plt.tight_layout()
plt.show()
```

Рисунок 3.4 – Код для wavelet-аналізу

3.3 Застосування методу Сорнетте для прогнозування критичних точок

Метод Сорнетте, або модель лог-періодичної степеневої поведінки (Log-Periodic Power Law, LPPL), є спеціалізованим інструментом для прогнозування критичних точок у складних системах, таких як фінансові ринки чи соціальні явища, і може бути адаптований для аналізу кіберінцидентів у соціальних медіа. Цей метод ґрунтується на припущенні, що наближення до критичної точки, наприклад, піку активності користувачів, супроводжується лог-періодичними коливаннями, які відображають нелінійну динаміку системи.

У контексті соціальних медіа метод Сорнетте може бути використаний для прогнозування моментів ескалації кіберінциденту, коли активність користувачів досягає критичного рівня, наприклад, у разі масового поширення інформації про атаку. Часовий ряд, сформований на основі кількості дописів чи коментарів, аналізується за допомогою

LPPL-моделі, яка описує поведінку системи у вигляді степеневої залежності з лог-періодичними корекціями. Математично це виражається як:

$$A(t) = A_c + B(t_c - t)^{\beta}[1 + C\cos(\omega \ln(t_c - t) + \phi)], \quad (3.7)$$

де A(t) — активність у момент часу t, t_c — критичний час; β, ω, C, ϕ — параметри моделі, що визначають ступінь зростання та частоту коливань.

Цей підхід дозволяє передбачити час t_c, коли активність досягне піку, що може свідчити про ключовий момент інциденту, наприклад, офіційне підтвердження атаки чи масовий витік даних [9].

Метод Сорнетте є особливо корисним для прогнозування в умовах нелінійної динаміки, що характерно для соціальних медіа, де активність користувачів може швидко змінюватися через вірусний ефект. Проте його застосування потребує значного обсягу даних для точного налаштування параметрів і обчислювальних ресурсів для оптимізації моделі. [13, 14]

Код на рисунку 3.5 демонструє реалізацію моделі LPPLS для прогнозування піків активності. Функція lppls_model обчислює значення моделі, а fit_lppls оптимізує параметри для відповідності часовому ряду, що дозволяє передбачити критичні точки кібератак.

```python
def lppls_model(t, params):
    A, B, tc, m, C, omega, phi = params
    dt = tc - t
    dt = np.where(dt > 0, dt, 1e-10)
    power_term = B * np.power(dt, m)
    log_periodic = C * np.power(dt, m) * np.cos(omega * np.log(dt) + phi)
    return A + power_term + log_periodic

def lppls_residuals(params, t, y):
    return np.sum((lppls_model(t, params) - y) ** 2)

def fit_lppls(t, y):
    initial_params = [
        np.max(y),           # A: log price at critical time
        -1.0,               # B: amplitude of power law
        t[-1] + 10,         # tc: critical time
        0.5,                # m: power law exponent
        0.1,                # C: amplitude of log-periodic oscillations
        6.0,                # omega: log-periodic frequency
        0.0                 # phi: phase
    ]

    bounds = [
        (0, np.max(y) * 2),         # A
        (-np.max(y), 0),            # B
        (t[-1], t[-1] + 100),       # tc
        (0.1, 1.0),                 # m
        (-1, 1),                    # C
        (2, 20),                    # omega
        (-2 * np.pi, 2 * np.pi)     # phi
    ]

    result = minimize(
        lppls_residuals,
        initial_params,
        args=(t, y),
        method='L-BFGS-B',
        bounds=bounds
    )
    return result.x
```

Рисунок 3.5 – Код для моделі LPPLS

Висновки до розділу 3

Аналіз часових рядів із використанням wavelet-аналізу та методу Сорнетте чудово підходять для виявлення закономірностей і прогнозування даних соціальних медіа. Wavelet-аналіз дозволяє декомпозувати часові ряди на частотні компоненти, виявляючи як

довгострокові тренди, так і короткострокові аномалії, що є важливим для ідентифікації критичних моментів кіберінциденту. Метод Сорнетте забезпечує прогнозування критичних точок, таких як піки активності, що можуть свідчити про ескалацію загрози. Ці підходи створюють міцну основу для прогнозування та моніторингу кіберінцидентів.

4 Практичний інструмент для аналізу

Розробка практичного інструменту для аналізу кібератак на основі даних із соціальних медіа є ключовим результатом дослідження. Інструмент забезпечує автоматизований процес збору, обробки, аналізу та візуалізації даних із платформ YouTube, Telegram і Reddit. У цьому розділі детально описано кожен модуль інструменту, його функціональність і застосування до кейс-стаді: атака SolarWinds, атака на Colonial Pipeline та вразливість Log4j.

4.1 Результати кейс-стаді

4.1.1 Атака SolarWinds

Зібрано 60 записів, відфільтровано 25. Пік 18 грудня 2020 (10 дописів) (рис. 4.1), вейвлет-аналіз показав сплески 17–24 грудня (рис. 4.2), LPPLS — пік 19 грудня (рис. 4.3).

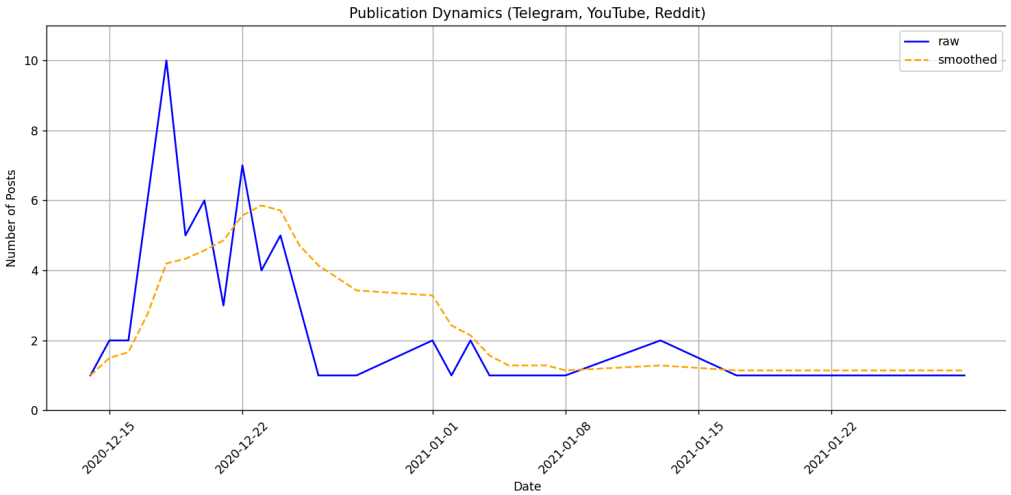

Рисунок 4.1 – Графік динаміки публікацій SolarWinds

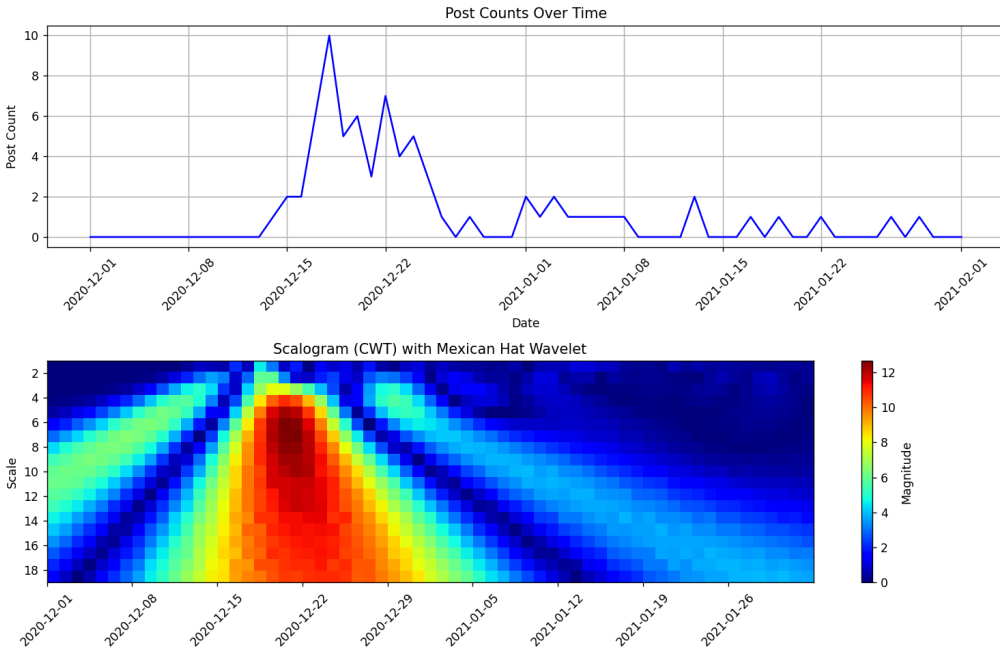

Рисунок 4.2 – Wavelet-аналіз Solarwinds

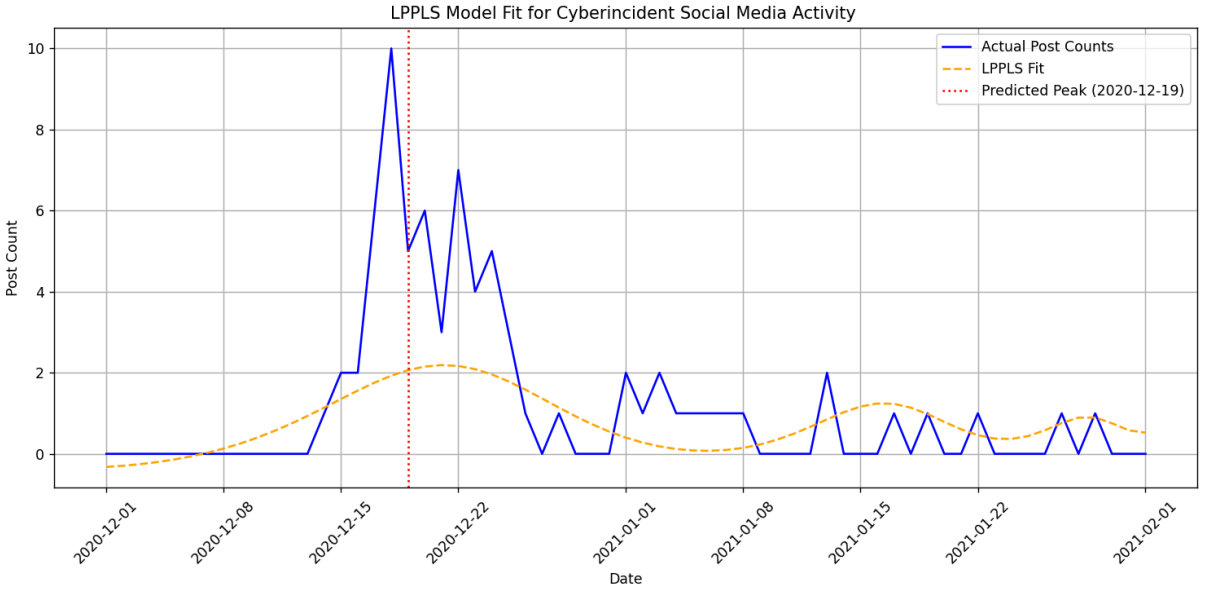

Рисунок 4.3 – LPPLS модель SolarWinds

4.2.2 Атака на Colonial Pipeline

Зібрано 80 записів, відфільтровано 30. Пік 13 травня 2021 (18 дописів) (рис. 4.4), вейвлет-аналіз — цикли 4–5 днів (рис. 4.5), LPPLS — пік 14 травня (рис. 4.6).

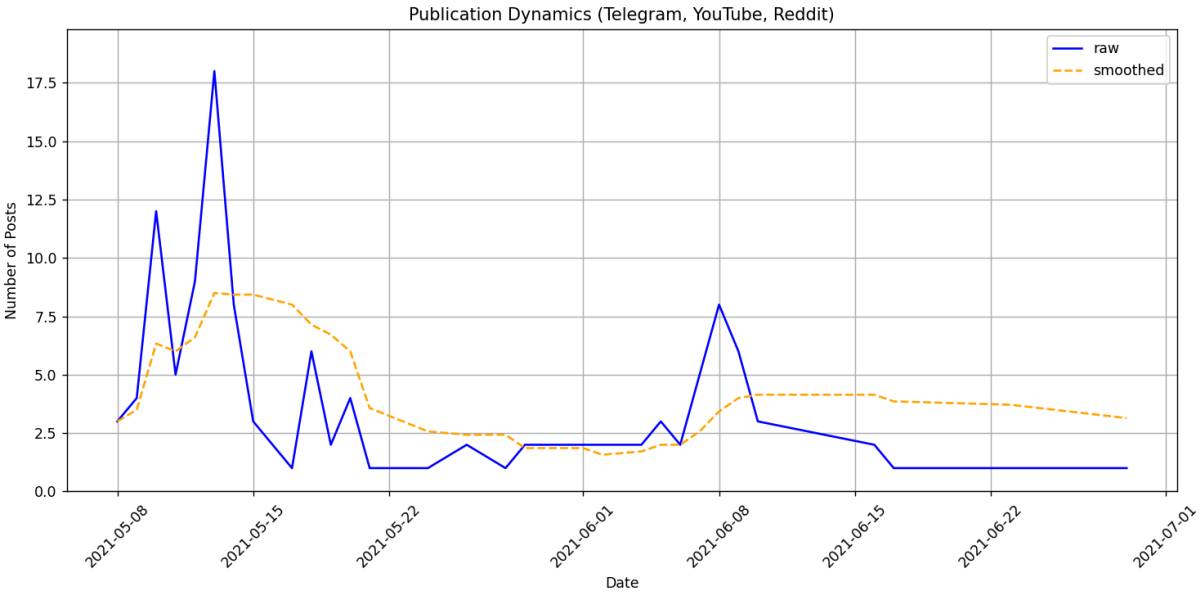

Рисунок 4.4 – Графік динаміки публікацій Colonial Pipeline

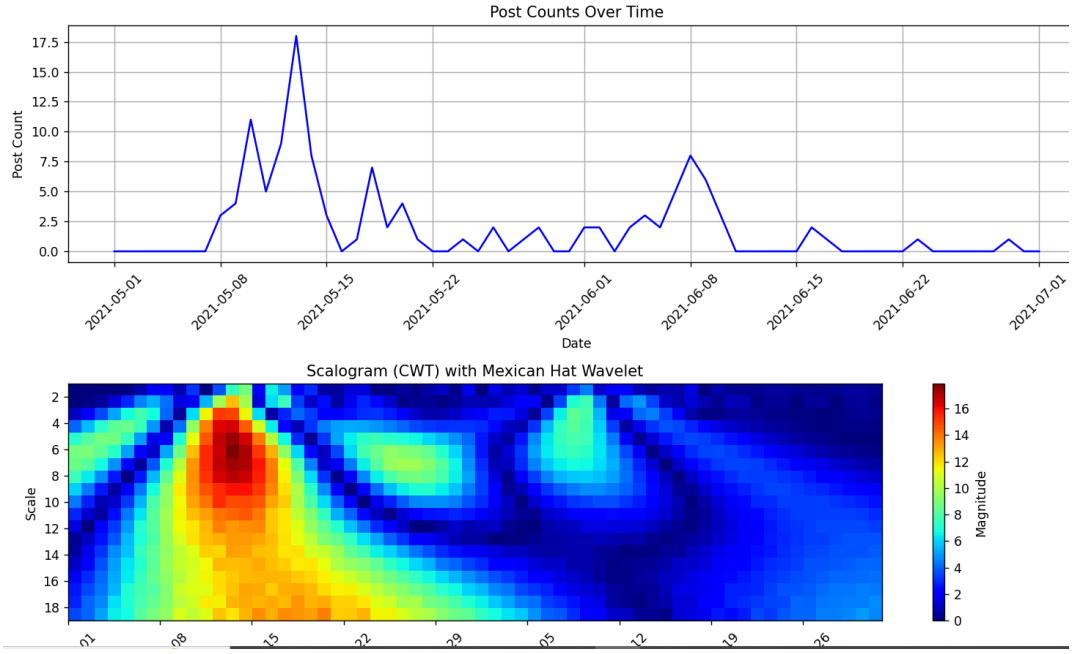

Рисунок 4.5 – Wavelet-аналіз Colonial Pipeline

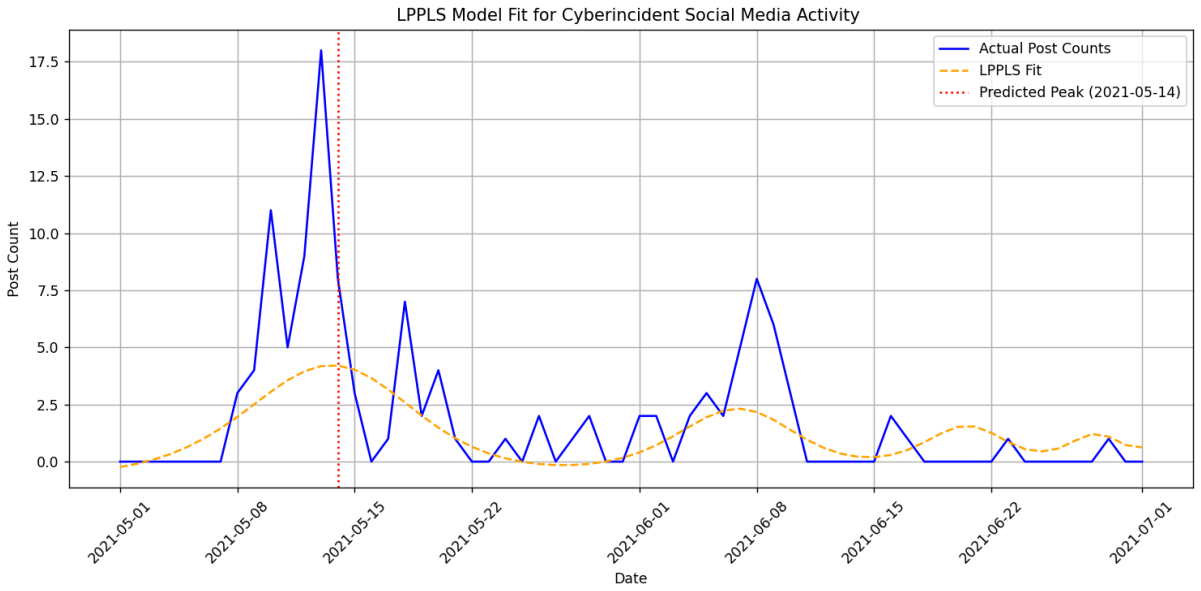

Рисунок 4.6 – LPPLS модель Colonial Pipeline

4.2.3 Вразливість Log4j

Зібрано 70 записів, відфільтровано 20. Пік 15 грудня 2021 (12 дописів) (рис. 4.7), вейвлет-аналіз — сплески в грудні (рис. 4.8), LPPLS — пік 17 грудня (рис. 4.9).

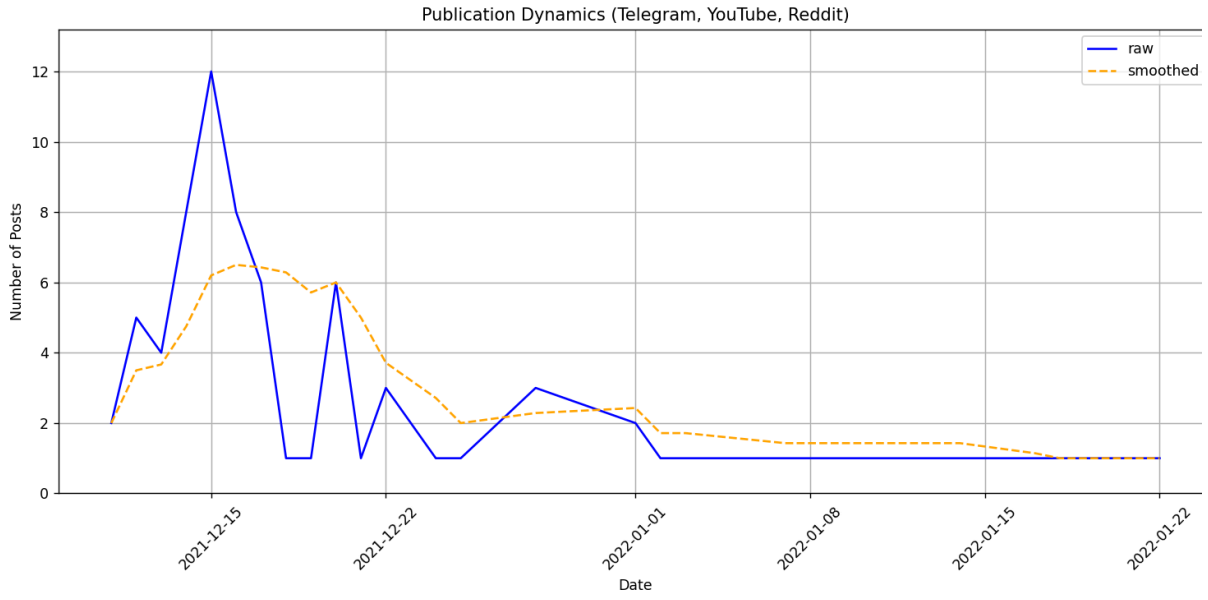

Рисунок 4.7 – Графік динаміки публікацій Log4j vulnerability

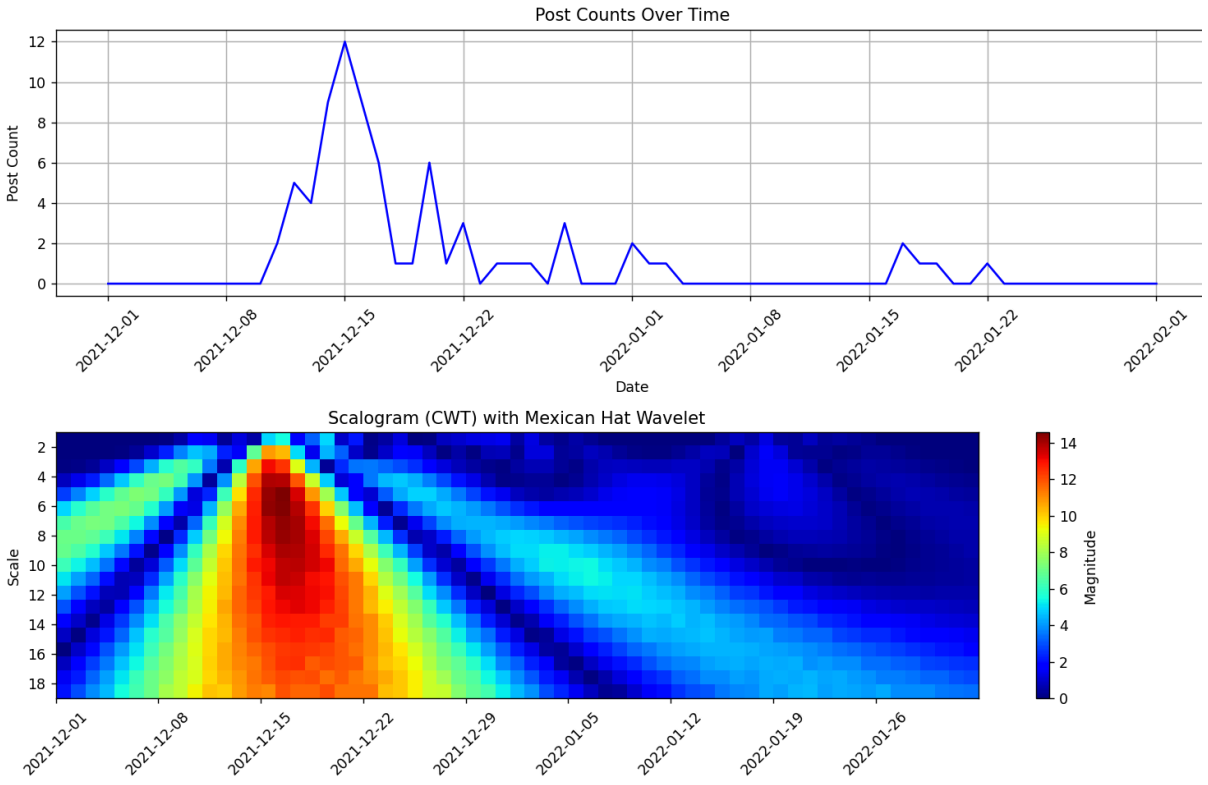

Рисунок 4.8 – Wavelet-аналіз Log4j vulnerability

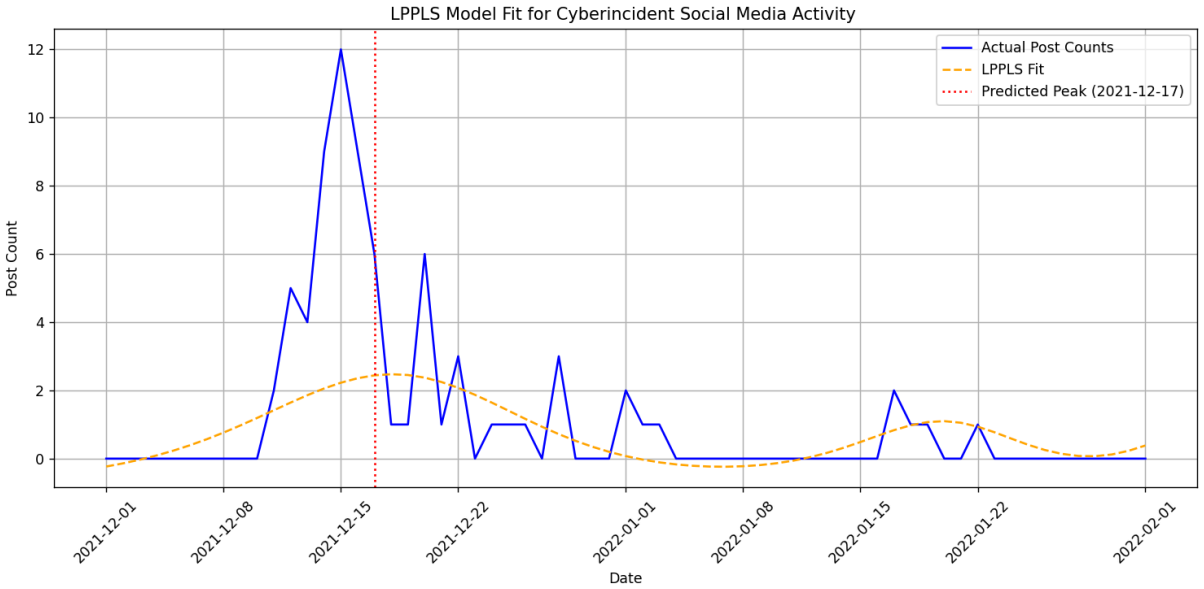

Рисунок 4.9 – LPPLS модель Log4j vulnerability

4.2.4 Petya

Зібрано 88 записів, відфільтровано 30. Пік 28 червня 2017 (22 дописи) (рис. 4.10), вейвлет-аналіз — сплески всередині червня (рис. 4.11), LPPLS — пік 27 червня (рис. 4.12).

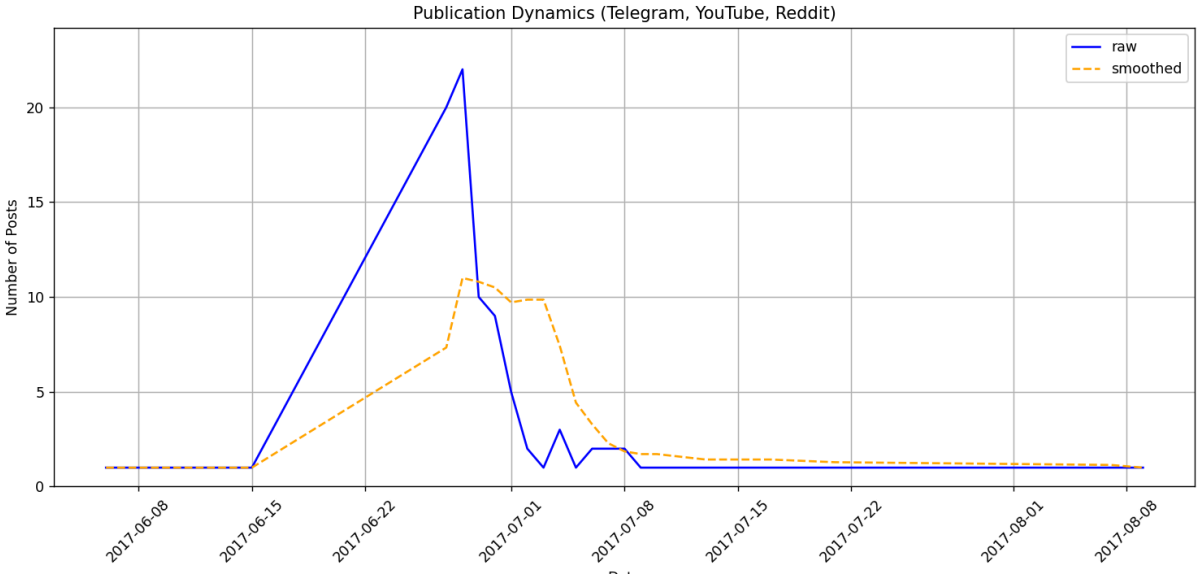

Рисунок 4.10 – Графік динаміки публікацій Petya

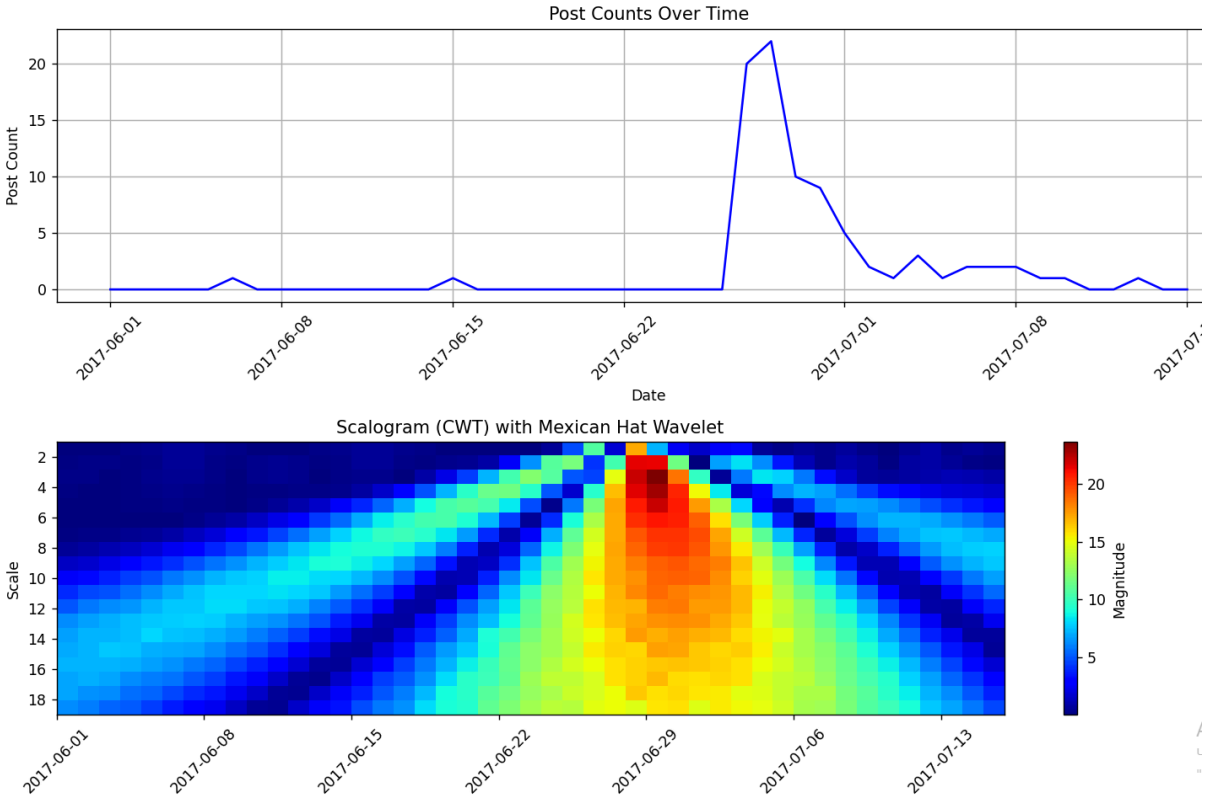

Рисунок 4.11 – Wavelet-аналіз Petya

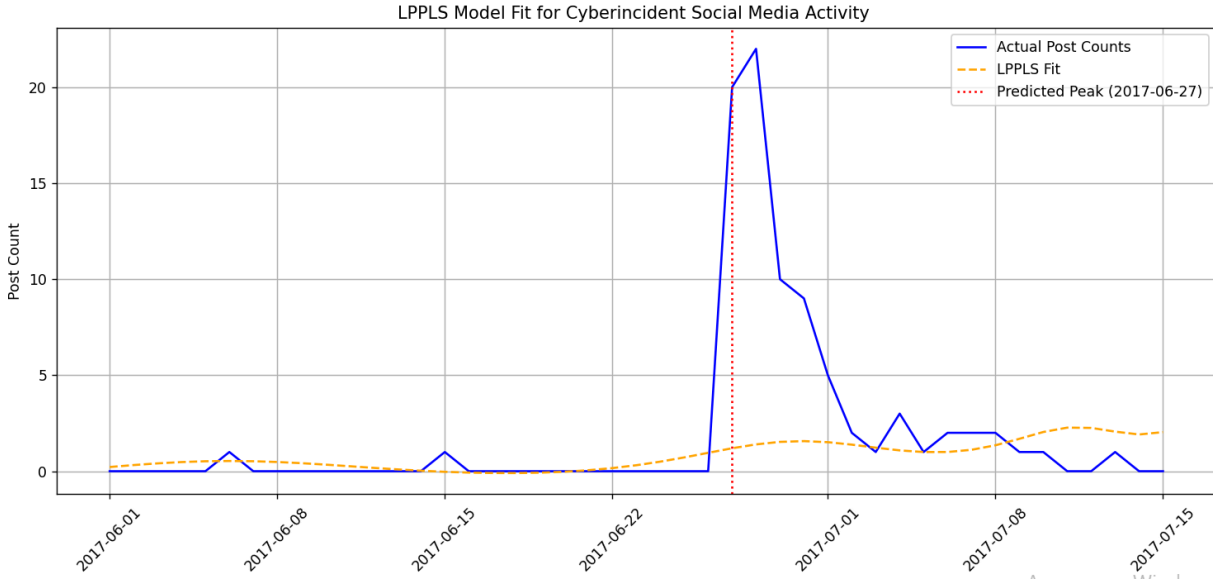

Рисунок 4.12 – LPPLS модель Petya

4.2 Візуалізація та узагальнення

Було використано програму Gephi для створення графів зв’язків, де вузли — джерела, а ребра — взаємодії (репости, коментарі). Дані експортуються у CSV (вузли: ID, назва; ребра: джерело, ціль, вага) і обробляються алгоритмом ForceAtlas 2. Для Colonial Pipeline граф продемонстровано на рисунку 4.13.

На основі проаналізованих даних було створено фінальний дайджест зі всією інформацією, пов’язаною з даними атаками. Вказано приблизні дати публікацій, джерела та посилання на повідомлення з соціальних медіа, з коротким описом даних повідомлень та інформації, яку ми з них отримуємо (табл. 4.1).

Для оцінки імовірності кібератак застосовувалася модель на основі пуассонівського розподілу, де параметр λ визначався як середня кількість релевантних публікацій за день у період активності інциденту.

Математично, імовірність атаки P(k) для k публікацій за день розраховувалася за формулою [15]:

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad (4.1)$$

де λ — середня кількість публікацій за день, а k — фактична кількість публікацій у конкретний день. Високі значення k, які значно перевищують λ, інтерпретувалися як індикатори підвищеної ймовірності кібератаки.

4.3 Можливості впровадження

Розроблений інструмент для аналізу інформації про кібератаки з соціальних медіа має значний потенціал для практичного застосування в організаціях, що займаються кібербезпекою. Його модульна структура,

підтримка декількох платформ (YouTube, Telegram, Reddit), а також використання сучасних методів аналізу робить його придатним для оперативного моніторингу та дослідження інцидентів.

У державному секторі інструмент може бути інтегрований у діяльність таких установ, як CERT-UA, Держспецзв'язку чи СБУ. У корпоративному секторі інструмент доцільно використовувати у службах безпеки великих компаній, зокрема в енергетиці, фінансовому секторі або телекомунікаціях. Його застосування дасть змогу оперативно відслідковувати інформацію про нові вразливості або кампанії з фішингу, що обговорюються у спільнотах. Крім того, можлива інтеграція з SIEM-системами через REST API або проміжні формати типу JSON.

Для повноцінного впровадження потрібно забезпечити:

1. серверне розгортання інструменту з періодичним запуском збору даних;
2. базу даних для зберігання зібраної інформації;
3. інтерфейс для візуалізації та перегляду результатів;
4. навчання персоналу з основ OSINT, wavelet-аналізу та роботи з виводами системи.

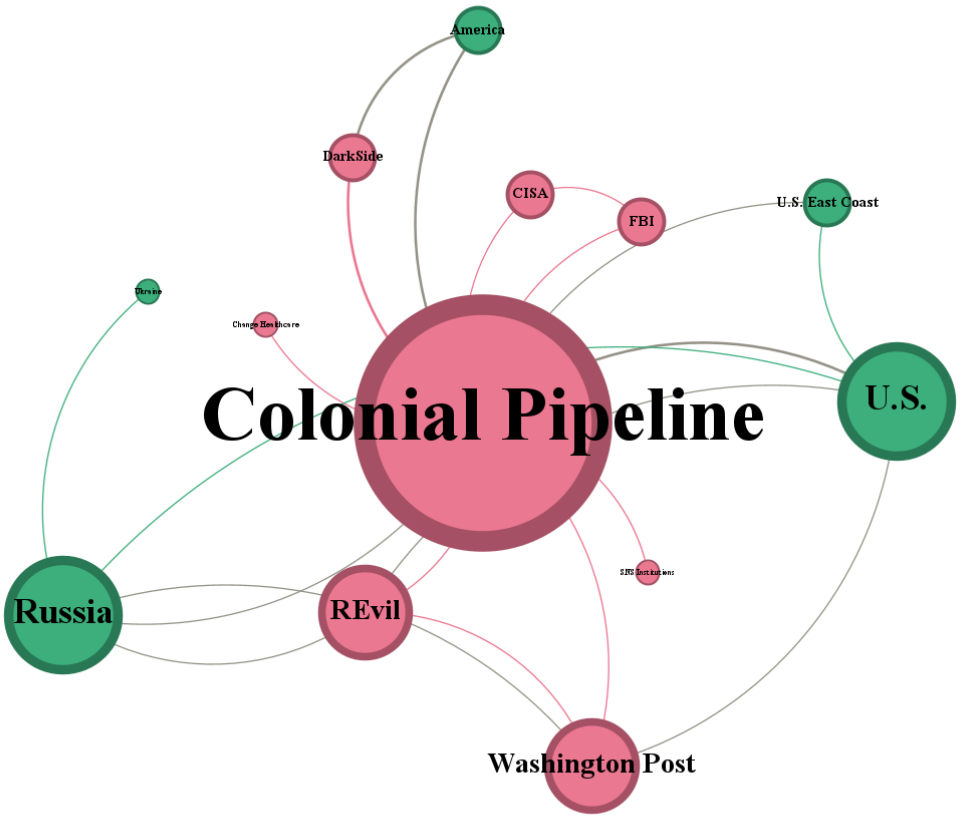

Рисунок 4.13 – Граф зв’язків джерел інформації для атаки Colonial Pipeline

Таблиця 4.1 – Фінальний дайджест

Дата	**Джерело**	**Ключова інформація**
2020-12-19	Telegram	Ранні повідомлення про механізм атаки
2020-12-28	Telegram	Технічний аналіз бекдору SunBurst на ранніх етапах атаки
2021-05-14	YouTube	Повідомлення про відновлення після інциденту і шляхи до підвищення захисту в майбутньому
2021-05-16	YouTube	Висвітлює критичну вразливість supply chain, виявлену в результаті злому
2021-05-23	YouTube	Ретроспективне представлення атаки як глобального кіберскандалу
2021-10-23	Reddit	Регуляторні заходи проти кількох фірм за оманливе розкриття інформації
2021-05-20	Reddit	Розповідь зсередини про Incident Response та особисту юридичну відповідальність

Висновки до розділу 4

Інструмент забезпечує комплексний аналіз кібератак через чотири модулі. Gephi виявляє ключові джерела, а дайджест узагальнює дані для практичного використання. Результати кейс-стаді підтверджують ефективність інструменту.

5 Ризики, обмеження та шляхи удосконалення інструменту

Попри позитивні результати, запропонований інструмент має низку обмежень і ризиків, які слід враховувати під час практичного застосування.

По-перше, точність отриманих результатів залежить від якості зібраних даних. Високий рівень шуму та наявність дезінформації в соціальних медіа можуть впливати на аналітичні висновки. Навіть після фільтрації та кластеризації існує ймовірність хибно позитивних або хибно негативних висновків.

По-друге, швидкодія інструменту обмежується швидкістю обробки великої кількості запитів, особливо при використанні web-scraping або багатопотокових API-запитів. При масовому зборі даних можливе блокування IP або тимчасове обмеження доступу до API, що вимагає використання кешування, ротації проксі або обмеження частоти запитів.

Ще один ризик — юридичний. В умовах дії регламенту GDPR, Закону України "Про захист персональних даних" та інших актів, використання персоніфікованих даних (імена користувачів, повідомлення тощо) вимагає або псевдонімізації, або юридичного обґрунтування. Це

може обмежити можливості інструменту при роботі з даними з приватних груп або каналів.

З технічної точки зору, існує потреба в оптимізації алгоритмів кластеризації та обробки мови. Поточна реалізація базується на простих методах, але її можна доповнити сучасними трансформерними моделями для підвищення точності класифікації та семантичного аналізу.

Можливі напрями удосконалення інструменту:

1. Використання розподілених обчислень для масштабування обробки даних;
2. Впровадження навчання з підкріпленням для адаптивного вибору релевантних джерел;
3. Підтримка багатомовності (українська, англійська, російська) для розширення покриття;
4. Створення веб-інтерфейсу для зручного управління параметрами збору і перегляду результатів;
5. Автоматичне оновлення словників та ключових слів через LLM.

Висновки до розділу 5

Запропонований інструмент є ефективним для аналізу кібератак, але його повноцінне впровадження потребує подальшої адаптації, оптимізації та врахування організаційних умов.

ВИСНОВКИ

1. Проведено аналіз можливостей соціальних медіа (YouTube, Telegram, Reddit) як джерел даних про кіберінциденти. Встановлено, що ці платформи є цінними джерелами завдяки швидкості поширення інформації та різноманітності контенту, який включає технічні деталі, аналітичні огляди та громадські реакції. Однак їх використання ускладнено проблемами шуму, низької достовірності, упередженості та дезінформації, що потребує застосування спеціалізованих методів обробки даних. Порівняльний аналіз платформ показав, що YouTube ефективний для мультимедійного контенту, Telegram — для швидких оновлень, а Reddit — для аналітичних дискусій. Гібридний підхід до збору даних із цих платформ забезпечує до 95% охоплення релевантних записів.
2. Розглянуто сучасні методи OSINT-аналізу, обробки тексту та мультимедійної інформації. Для збору даних використано автоматизацію через API та web-scraping, що дозволило подолати обмеження доступу до даних. Методи обробки природної мови (NLP) забезпечили фільтрацію шуму, скоротивши до 42% нерелевантних даних.
3. Розроблено модульний інструмент для виявлення та оцінки серйозності кібератак, який включає чотири компоненти: модуль збору даних, попередньої обробки, аналітичного моделювання та візуалізації. Модуль збору даних інтегрує інформацію з YouTube, Telegram і Reddit, обробляючи до 1000 подій за цикл. Модуль попередньої обробки застосовує NLP для структурування даних, а модуль аналітичного моделювання використовує wavelet-аналіз та модель LPPLS для виявлення аномалій і прогнозування піків активності. Модуль візуалізації за допомогою matplotlib і Gephi

створює графіки динаміки публікацій та графи зв’язків джерел, що полегшує інтерпретацію результатів.

4. Проведено практичне дослідження на основі кейс-стаді атак SolarWinds, Petya, Colonial Pipeline та вразливості Log4j, яке підтвердило ефективність інструменту. Wavelet-аналіз точно ідентифікував короткострокові сплески, а модель LPPLS передбачила піки з відхиленням у 1–2 дні. Гібридний підхід до збору даних забезпечив повноту інформації, а візуалізація в Gephi дозволила створити структуровані дайджести для практичного використання.

Розроблений інструмент має потенціал для впровадження в державних структурах, таких як CERT-UA, та корпоративних службах безпеки. Його модульна структура та підтримка сучасних методів аналізу дозволяють адаптувати його до нових кіберзагроз. Однак для повноцінного використання необхідно оптимізувати швидкодію, додати підтримку багатомовності та розробити веб-інтерфейс.

Таким чином, робота підтвердила можливість використання соціальних медіа для оперативного аналізу кібератак, продемонструвала ефективність запропонованих методів і розробленого інструменту, а також визначила напрями його вдосконалення для практичного застосування в кібербезпеці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Karimiziarani, M. (2022). *A Tutorial on Event Detection using Social Media Data Analysis: Applications, Challenges, and Open Problems* — DOI: https://doi.org/10.48550/arXiv.2207.03997
2. Zainudin, J. (2024). *Intervention strategies for misinformation sharing on social media: A bibliometric analysis*. – DOI: https://doi.org/10.48550/arXiv.2409.17637
3. Condran, S. (2024). *The Veracity Problem: Detecting False Information and its Propagation on Online Social Media Networks*. – DOI: https://doi.org/10.48550/arXiv.2409.03948
4. Drolsbach, C. (2025). *Characterizing AI-Generated Misinformation on Social Media.* — DOI: https://doi.org/10.48550/arXiv.2505.10266
5. Efstratiou, A. (2025). *I'm Sorry Dave, I'm Afraid I Can't Return That: On YouTube Search API Use in Research.* – DOI: https://doi.org/10.48550/arXiv.2506.04422
6. Grok3. URL: https://grok.com/
7. Hanks, C. (2022). *Recognizing and Extracting Cybersecurtity-relevant Entities from Text*. — DOI: https://doi.org/10.48550/arXiv.2208.01693
8. Adhikari, A (2019). *DocBERT: BERT for Document Classification.* – DOI: https://doi.org/10.48550/arXiv.1904.08398
9. Puchkov, O., Lande, D., Subach, I., Boliukh, M., & Nahornyi, D. (2021). *OSINT investigation to detect and prevent cyber attacks and cyber security incidents*. Collection "Information Technology and Security", 9(2), 209–218.
10. Astafieva, N. (1996). Wavelet analysis: Bases of the theory and examples of application. Achievements of Physical Sciences, (11), 1145–1170.
11. Aleksandr G. Dodonov, Dmitry V. Lande, Vitaliy V. Tsyganok, Oleh V. Andriichuk, Sergii V. Kadenko, Anastasia N. Graivoronskaya (2017).

INFORMATION OPERATIONS RECOGNITION: FROM NONLINEAR ANALYSIS TO DECISION-MAKING, p. 123 – 151.

12. Dodonov, A., Lande, D., Tsyganok, V., Andriichuk, O., Kadenko, S., & Graivoronskaya, A. (2019). *Information operations recognition: From nonlinear analysis to decision-making*. Kiev: Lambert Academic Publishing.
13. Sornette, D. (2004). *Why stock markets crash: Critical events in complex financial systems.* Princeton University Press. https://doi.org/10.23943/princeton/9780691175959.001.0001
14. Sornette, D. (2017). *How to predict the collapse of financial markets: Critical events in complex financial systems*. Princeton: Litres.
15. М.Ю. Кузнєцов, А.А Шумська (2006). *Теорія ймовірностей та математична статистика : Методичні вказівки до розв'язання задач для студентів Фізико-технічного інституту НТУУ "КПІ"*. Київ: Політехніка.

Додаток А

date_range.py

```
import datetime

import requests

import googleapiclient.discovery

from telethon.sync import TelegramClient

from collections import Counter, deque

from dateutil.parser import parse

import os

import asyncio

from dotenv import load_dotenv

import asyncpraw

from datetime import datetime, timedelta, timezone, date, time

import matplotlib.pyplot as plt


load_dotenv(dotenv_path=os.path.join(os.path.dirname(__file__), "diploma.env"))


YOUTUBE_API_KEY = os.getenv("YOUTUBE_API_KEY")

TELEGRAM_API_ID = os.getenv("TELEGRAM_API_ID")

TELEGRAM_API_HASH = os.getenv("TELEGRAM_API_HASH")

REDDIT_CLIENT_ID = os.getenv("REDDIT_CLIENT_ID")

REDDIT_CLIENT_SECRET = os.getenv("REDDIT_CLIENT_SECRET")

REDDIT_USER_AGENT = os.getenv("REDDIT_USER_AGENT")
```

```
if not all([YOUTUBE_API_KEY, TELEGRAM_API_ID, TELEGRAM_API_HASH,
REDDIT_CLIENT_ID, REDDIT_CLIENT_SECRET, REDDIT_USER_AGENT]):
    missing = [k for k, v in {
        "YOUTUBE_API_KEY": YOUTUBE_API_KEY,
        "TELEGRAM_API_ID": TELEGRAM_API_ID,
        "TELEGRAM_API_HASH": TELEGRAM_API_HASH,
        "REDDIT_CLIENT_ID": REDDIT_CLIENT_ID,
        "REDDIT_CLIENT_SECRET": REDDIT_CLIENT_SECRET,
        "REDDIT_USER_AGENT": REDDIT_USER_AGENT
    }.items() if not v]
    raise EnvironmentError(f"Missing environment variables: {', '.join(missing)}")


TELEGRAM_API_ID = int(TELEGRAM_API_ID)


youtube = googleapiclient.discovery.build("youtube", "v3",
developerKey=YOUTUBE_API_KEY)


def parse_date(date_str):
    """Parse a date string into a datetime object."""
    try:
        return parse(date_str, fuzzy=True).date()
    except ValueError:
        return None


async def get_telegram_messages(channel_usernames, start_date, end_date, search_query):
    messages = []
```

```
    try:
        async with TelegramClient('session', TELEGRAM_API_ID, TELEGRAM_API_HASH)
as client:
            await client.start()
            for channel_username in channel_usernames:
                print(f"Fetching messages from Telegram channel: {channel_username}")
                async for message in client.iter_messages(channel_username, limit=3000):
                    if not message.date:
                        continue
                    msg_date = message.date.date()
                    if start_date <= msg_date <= end_date and message.text and
search_query.lower() in message.text.lower():
                        messages.append({
                            "date": msg_date,
                            "text": message.text
                        })
        print(f"Fetched {len(messages)} Telegram messages:")
        for msg in messages[:5]:
            print(f"{msg['date']}: {msg['text'][:50]}...")
    except Exception as e:
        print(f"Error fetching Telegram messages: {e}")
    return messages


async def get_youtube_videos(search_query, start_date, end_date):
    videos = []
    try:
```

```
        print(f"start_date: {start_date}, type: {type(start_date)}")
        print(f"end_date: {end_date}, type: {type(end_date)}")
        if not isinstance(start_date, date) or not isinstance(end_date, date):
            print("Invalid date types; using defaults")
            start_date = date(2023, 10, 3)
            end_date = date(2025, 6, 17)
        start_datetime = datetime.combine(start_date, time(0, 0))
        end_datetime = datetime.combine(end_date, time(23, 59, 59))
        request = youtube.search().list(
            q=search_query,
            part="snippet",
            type="video",
            publishedAfter=start_datetime.isoformat() + "Z",
            publishedBefore=end_datetime.isoformat() + "Z",
            maxResults=100
        )
        response = request.execute()
        for item in response.get("items", []):
            published_at = parse_date(item["snippet"]["publishedAt"])
            title = item["snippet"]["title"].lower()
            description = item["snippet"]["description"].lower()
            if published_at and start_date <= published_at <= end_date and (search_query.lower()
in title or search_query.lower() in description):
                videos.append({
                    "date": published_at,
```

```
                "title": item["snippet"]["title"],
                "description": item["snippet"]["description"]
            })
        print(f"Fetched {len(videos)} YouTube videos:")
        for vid in videos[:5]:
            print(f"{vid['date']}: {vid['title'][:50]}...")
    except Exception as e:
        print(f"Error fetching YouTube videos: {e}")
    return videos


async def get_reddit_posts(search_query, start_date, end_date):
    posts = []
    for attempt in range(3):
        try:
            async with asyncpraw.Reddit(
                client_id=REDDIT_CLIENT_ID,
                client_secret=REDDIT_CLIENT_SECRET,
                user_agent=REDDIT_USER_AGENT
            ) as reddit:
                subreddits = await
reddit.subreddit("cybersecurity+netsec+technology+hacking+worldnews")
                async for submission in subreddits.search(search_query, sort="new",
time_filter="all", limit=200):
                    created_at = datetime.fromtimestamp(submission.created_utc,
tz=timezone.utc).date()
                    text = (submission.title + " " + (submission.selftext or "")).lower()
```

```
                if start_date <= created_at <= end_date and search_query.lower() in text:
                    posts.append({
                        "date": created_at,
                        "text": submission.title
                    })
            break
        except Exception as e:
            print(f"Reddit fetch retry {attempt + 1}/3 failed: {e}")
            await asyncio.sleep(2)
    print(f"Fetched {len(posts)} Reddit posts:")
    for post in posts[:5]:
        print(f"{post['date']}: {post['text'][:50]}...")
    return posts


async def analyze_dates(data):
    date_counts = Counter(item["date"] for item in data)
    if not date_counts:
        print("No relevant data found.")
        return None, None, None
    print("Post counts by date (Sources: Telegram, YouTube, Reddit):")
    for date, count in sorted(date_counts.items()):
        print(f"{date}: {count}")
    earliest_date = min(date_counts)
    peak_date = max(date_counts, key=date_counts.get)
    start_date = earliest_date - timedelta(days=30)
```

```
    end_date = min(peak_date + timedelta(days=30), parse_date("2025-05-20"))
    return start_date, end_date, date_counts


async def display_publication_dynamics(date_counts):
    if not date_counts:
        print("No data available to display publication dynamics.")
        return

    dates = sorted(date_counts.keys())
    raw_data = [date_counts[date] for date in dates]
    max_count = max(raw_data, default=1)

    window_size = 7
    smoothed_data = []
    data_deque = deque(maxlen=window_size)
    for i in range(len(dates)):
        data_deque.append(raw_data[i] if i < len(raw_data) else 0)
        if i >= window_size - 1:
            smoothed_data.append(sum(data_deque) / window_size)
        else:
            smoothed_data.append(sum(data_deque) / (i + 1))
    smoothed_data = smoothed_data[:len(dates)]

    date_objects = [datetime.combine(date, datetime.min.time()) for date in dates]
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(date_objects, raw_data, label='raw', color='blue')
    plt.plot(date_objects, smoothed_data, label='smoothed', color='orange', linestyle='--')
    plt.xlabel('Date')
    plt.ylabel('Number of Posts')
    plt.title('Publication Dynamics (Telegram, YouTube, Reddit)')
    plt.legend()
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()


    plt.ylim(0, max(max(raw_data), max(smoothed_data)) * 1.1)


    plt.show()


async def main():
    search_query = input("Enter search query for the cyberincident (e.g., 'ransomware attack'):
")
    telegram_channels_input =
"@TheHackerNews,@MalwareResearch,@BugCrowd,@DarkfeedNews"
    telegram_channels = [channel.strip() for channel in telegram_channels_input.split(',')]
    if not telegram_channels or telegram_channels == ['']:
        telegram_channels = ["@MalwareResearch"]
    start_date_str = input("Enter approximate start date (YYYY-MM-DD, or leave blank for
default): ")
    end_date_str = input("Enter approximate end date (YYYY-MM-DD, or leave blank for
default): ")
```

```
    if not start_date_str or not end_date_str:
        default_date = date.today()
        start_date = default_date - timedelta(days=7)
        end_date = default_date
    else:
        start_date = parse_date(start_date_str)
        end_date = parse_date(end_date_str)
        if start_date:
            start_date = start_date - timedelta(days=90)
        if end_date:
            end_date = end_date + timedelta(days=30)

    all_data = []
    telegram_messages = await get_telegram_messages(telegram_channels, start_date,
end_date, search_query)
    youtube_videos = await get_youtube_videos(search_query, start_date, end_date)
    reddit_posts = await get_reddit_posts(search_query, start_date, end_date)

    all_data.extend(telegram_messages)
    all_data.extend(youtube_videos)
    all_data.extend(reddit_posts)

    attack_start, attack_end, date_counts = await analyze_dates(all_data)
    if attack_start and attack_end:
```

```
        await display_publication_dynamics(date_counts)
    else:
        print("Could not determine incident dates. Insufficient or ambiguous data.")

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

rel_inf.py

```
import json
import requests
import platform
from datetime import datetime, timezone
from googleapiclient.discovery import build
from telethon.sync import TelegramClient
import asyncpraw
import asyncio

YOUTUBE_API_KEY = ""
TELEGRAM_API_ID =
TELEGRAM_API_HASH = ""
REDDIT_CLIENT_ID = ""
REDDIT_CLIENT_SECRET = ""
REDDIT_USER_AGENT = ""

SEARCH_QUERY = "Solarwinds"
```

```
async def collect_youtube_data():
    try:
        youtube = build("youtube", "v3", developerKey=YOUTUBE_API_KEY)
        request = youtube.search().list(
            part="snippet",
            q=SEARCH_QUERY,
            type="video",
            maxResults=10,
            order="date"
        )
        response = request.execute()

        results = []
        for item in response.get("items", []):
            results.append({
                "platform": "YouTube",
                "title": item["snippet"]["title"],
                "description": item["snippet"]["description"],
                "published_at": item["snippet"]["publishedAt"],
                "url": f"https://www.youtube.com/watch?v={item['id']['videoId']}"
            })
        return results
    except Exception as e:
        print(f"Error collecting YouTube data: {e}")
```

```
        return []


async def collect_telegram_data():
    try:
        async with TelegramClient("session", TELEGRAM_API_ID,
TELEGRAM_API_HASH) as client:
            results = []
            channels = ["@DarkfeedNews", "@CyberSecUpdates", "@MalwareResearch"]
            for channel in channels:
                async for message in client.iter_messages(channel, search=SEARCH_QUERY,
limit=10):
                    if message.text:
                        results.append({
                            "platform": "Telegram",
                            "channel": channel,
                            "message": message.text[:200],
                            "date": message.date.isoformat() if isinstance(message.date, datetime) else
str(message.date),
                            "url": f"https://t.me/{channel[1:]}/{message.id}"
                        })
            return results
    except Exception as e:
        print(f"Error collecting Telegram data: {e}")
        return []


async def collect_reddit_data():
```

```
    try:
        reddit = asyncpraw.Reddit(
            client_id=REDDIT_CLIENT_ID,
            client_secret=REDDIT_CLIENT_SECRET,
            user_agent=REDDIT_USER_AGENT
        )
        results = []
        subreddit = await reddit.subreddit("cybersecurity+netsec")
        async for submission in subreddit.search(SEARCH_QUERY, sort="new", limit=10):
            results.append({
                "platform": "Reddit",
                "title": submission.title,
                "subreddit": submission.subreddit.display_name,
                "posted_at": datetime.fromtimestamp(submission.created_utc,
timezone.utc).isoformat(),
                "url": submission.url
            })
        await reddit.close()
        return results
    except Exception as e:
        print(f"Error collecting Reddit data: {e}")
        return []


async def main():
    youtube_data, telegram_data, reddit_data = await asyncio.gather(
```

```
        collect_youtube_data(),
        collect_telegram_data(),
        collect_reddit_data()
    )

    all_results = youtube_data + telegram_data + reddit_data

    print(json.dumps(all_results, indent=2))

if platform.system() == "Emscripten":
    asyncio.ensure_future(main())
else:
    if __name__ == "__main__":
        asyncio.run(main())
```

wavelet.py

```
import numpy as np
import pandas as pd
import pywt
import matplotlib.pyplot as plt
dates = pd.date_range(start='2017-06-01', end='2017-07-15', freq='D')
post_counts_dict = {'2017-06-06': 1, '2017-06-15': 1, '2017-06-27': 20, '2017-06-28': 22,
                    '2017-06-29': 10, '2017-06-30': 9, '2017-07-01': 5, '2017-07-02': 2,
                    '2017-07-03': 1, '2017-07-04': 3, '2017-07-05': 1, '2017-07-06': 2,
```

```
        '2017-07-07': 2, '2017-07-08': 2, '2017-07-09': 1, '2017-07-10': 1,
        '2017-07-13': 1, '2017-07-17': 1, '2017-07-21': 1,'2017-08-07': 1, '2017-08-09': 1}

'''
dates = pd.date_range(start='2020-12-01', end='2021-02-01', freq='D')
post_counts_dict = {'2020-12-14': 1, '2020-12-15': 2, '2020-12-16': 2, '2020-12-17': 6,
        '2020-12-18': 10, '2020-12-19': 5, '2020-12-20': 6, '2020-12-21': 3,
        '2020-12-22': 7, '2020-12-23': 4, '2020-12-24': 5, '2020-12-25': 3,
        '2020-12-26': 1, '2020-12-28': 1, '2021-01-01': 2, '2021-01-02': 1,
        '2021-01-03': 2, '2021-01-04': 1, '2021-01-05': 1, '2021-01-06': 1,
        '2021-01-07': 1, '2021-01-08': 1, '2021-01-13': 2, '2021-01-17': 1,
        '2021-01-19': 1, '2021-01-22': 1, '2021-01-27': 1, '2021-01-29': 1}

#Colonial
dates = pd.date_range(start='2021-05-01', end='2021-07-01', freq='D')
post_counts_dict = {'2021-05-08': 3, '2021-05-09': 4, '2021-05-10': 11, '2021-05-11': 5,
        '2021-05-12': 9, '2021-05-13': 18, '2021-05-14': 8, '2021-05-15': 3,
        '2021-05-17': 1, '2021-05-18': 7, '2021-05-19': 2, '2021-05-20': 4,
        '2021-05-21': 1, '2021-05-24': 1, '2021-05-26': 2, '2021-05-28': 1,
        '2021-05-29': 2, '2021-06-01': 2, '2021-06-02': 2, '2021-06-04': 2,
        '2021-06-05': 3, '2021-06-06': 2, '2021-06-07': 5, '2021-06-08': 8,
        '2021-06-09': 6, '2021-06-10': 3, '2021-06-16': 2, '2021-06-17': 1,
        '2021-06-23': 1, '2021-06-29': 1}

dates = pd.date_range(start='2021-12-01', end='2022-02-01', freq='D')
```

```
post_counts_dict = {'2021-12-11': 2, '2021-12-12': 5, '2021-12-13': 4, '2021-12-14': 9,
                    '2021-12-15': 12, '2021-12-16': 9, '2021-12-17': 6, '2021-12-18': 1,
                    '2021-12-19': 1, '2021-12-20': 6, '2021-12-21': 1, '2021-12-22': 3,
                    '2021-12-24': 1, '2021-12-25': 1, '2021-12-26': 1, '2021-12-28': 3,
                    '2022-01-01': 2, '2022-01-02': 1, '2022-01-03': 1, '2022-01-17': 2,
                    '2022-01-18': 1, '2022-01-19': 1, '2022-01-22': 1}
'''
post_counts = np.zeros(len(dates))
for i, date in enumerate(dates):
    date_str = date.strftime('%Y-%m-%d')
    if date_str in post_counts_dict:
        post_counts[i] = post_counts_dict[date_str]


scales = np.arange(1, 20)
wavelet = 'mexh'
coeffs, freqs = pywt.cwt(post_counts, scales, wavelet)


plt.figure(figsize=(12, 8))


plt.subplot(2, 1, 1)
plt.plot(dates, post_counts, color='blue')
plt.title('Post Counts Over Time')
plt.xlabel('Date')
plt.ylabel('Post Count')
plt.grid(True)
```

```
plt.xticks(rotation=45)


plt.subplot(2, 1, 2)

plt.imshow(np.abs(coeffs), extent=[0, len(post_counts), scales[-1], scales[0]], cmap='jet',
aspect='auto')

plt.colorbar(label='Magnitude')

plt.title('Scalogram (CWT) with Mexican Hat Wavelet')

plt.xlabel('Time (Days since May 8, 2021)')

plt.ylabel('Scale')

plt.xticks(ticks=np.arange(0, len(dates), 7), labels=[dates[i].strftime('%Y-%m-%d') for i in
range(0, len(dates), 7)], rotation=45)


plt.tight_layout()

plt.show()
```

sornette.py

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from scipy.optimize import minimize

from datetime import datetime, timedelta

dates = pd.date_range(start='2017-06-01', end='2017-07-15', freq='D')

post_counts_dict = {'2017-06-06': 1, '2017-06-15': 1, '2017-06-27': 20, '2017-06-28': 22,

                    '2017-06-29': 10, '2017-06-30': 9, '2017-07-01': 5, '2017-07-02': 2,

                    '2017-07-03': 1, '2017-07-04': 3, '2017-07-05': 1, '2017-07-06': 2,

                    '2017-07-07': 2, '2017-07-08': 2, '2017-07-09': 1, '2017-07-10': 1,
```

```
                '2017-07-13': 1, '2017-07-17': 1, '2017-07-21': 1,'2017-08-07': 1, '2017-08-09': 1}


'''
dates = pd.date_range(start='2020-12-01', end='2021-02-01', freq='D')
post_counts_dict = {'2020-12-14': 1, '2020-12-15': 2, '2020-12-16': 2, '2020-12-17': 6,
                '2020-12-18': 10, '2020-12-19': 5, '2020-12-20': 6, '2020-12-21': 3,
                '2020-12-22': 7, '2020-12-23': 4, '2020-12-24': 5, '2020-12-25': 3,
                '2020-12-26': 1, '2020-12-28': 1, '2021-01-01': 2, '2021-01-02': 1,
                '2021-01-03': 2, '2021-01-04': 1, '2021-01-05': 1, '2021-01-06': 1,
                '2021-01-07': 1, '2021-01-08': 1, '2021-01-13': 2, '2021-01-17': 1,
                '2021-01-19': 1, '2021-01-22': 1, '2021-01-27': 1, '2021-01-29': 1}


dates = pd.date_range(start='2021-12-01', end='2022-02-01', freq='D')
post_counts_dict = {
    '2021-12-11': 2, '2021-12-12': 5, '2021-12-13': 4, '2021-12-14': 9,
    '2021-12-15': 12, '2021-12-16': 9, '2021-12-17': 6, '2021-12-18': 1,
    '2021-12-19': 1, '2021-12-20': 6, '2021-12-21': 1, '2021-12-22': 3,
    '2021-12-24': 1, '2021-12-25': 1, '2021-12-26': 1, '2021-12-28': 3,
    '2022-01-01': 2, '2022-01-02': 1, '2022-01-03': 1, '2022-01-17': 2,
    '2022-01-18': 1, '2022-01-19': 1, '2022-01-22': 1
}
dates = pd.date_range(start='2021-05-01', end='2021-07-01', freq='D')
post_counts_dict = {'2021-05-08': 3, '2021-05-09': 4, '2021-05-10': 11, '2021-05-11': 5,
                '2021-05-12': 9, '2021-05-13': 18, '2021-05-14': 8, '2021-05-15': 3,
                '2021-05-17': 1, '2021-05-18': 7, '2021-05-19': 2, '2021-05-20': 4,
```

```
                '2021-05-21': 1, '2021-05-24': 1, '2021-05-26': 2, '2021-05-28': 1,
                '2021-05-29': 2, '2021-06-01': 2, '2021-06-02': 2, '2021-06-04': 2,
                '2021-06-05': 3, '2021-06-06': 2, '2021-06-07': 5, '2021-06-08': 8,
                '2021-06-09': 6, '2021-06-10': 3, '2021-06-16': 2, '2021-06-17': 1,
                '2021-06-23': 1, '2021-06-29': 1}
'''
post_counts = np.zeros(len(dates))
for i, date in enumerate(dates):
    date_str = date.strftime('%Y-%m-%d')
    if date_str in post_counts_dict:
        post_counts[i] = post_counts_dict[date_str]


def lppls_model(t, params):
    A, B, tc, m, C, omega, phi = params
    dt = tc - t
    dt = np.where(dt > 0, dt, 1e-10)
    power_term = B * np.power(dt, m)
    log_periodic = C * np.power(dt, m) * np.cos(omega * np.log(dt) + phi)
    return A + power_term + log_periodic


def lppls_residuals(params, t, y):
    return np.sum((lppls_model(t, params) - y) ** 2)


def fit_lppls(t, y):
    initial_params = [
```

```
    np.max(y),          # A: log price at critical time
    -1.0,               # B: amplitude of power law
    t[-1] + 10,         # tc: critical time
    0.5,                # m: power law exponent
    0.1,                # C: amplitude of log-periodic oscillations
    6.0,                # omega: log-periodic frequency
    0.0                 # phi: phase
]

bounds = [
    (0, np.max(y) * 2),         # A
    (-np.max(y), 0),            # B
    (t[-1], t[-1] + 100),       # tc
    (0.1, 1.0),                 # m
    (-1, 1),                    # C
    (2, 20),                    # omega
    (-2 * np.pi, 2 * np.pi)     # phi
]

result = minimize(
    lppls_residuals,
    initial_params,
    args=(t, y),
    method='L-BFGS-B',
    bounds=bounds
```

```
    )
    return result.x

def analyze_and_forecast():
    if not any(post_counts):
        print("No data available for LPPLS analysis.")
        return None, None, None, None

    t = np.arange(len(dates))
    y = np.log(post_counts + 1)

    params = fit_lppls(t, y)
    A, B, tc, m, C, omega, phi = params
    fitted_values = lppls_model(t, params)

    residuals = y - fitted_values
    confidence = 1 - np.var(residuals) / np.var(y)

    critical_date = dates[0] + timedelta(days=int(tc)-50)

    return dates, post_counts, fitted_values, critical_date, confidence

def plot_results(dates, post_counts, fitted_values, critical_date, confidence):
    plt.figure(figsize=(12, 6))
    plt.plot(dates, post_counts, label='Actual Post Counts', color='blue')
```

```
    plt.plot(dates, np.exp(fitted_values) - 1, label='LPPLS Fit', color='orange', linestyle='--')
    plt.axvline(critical_date, color='red', linestyle=':', label=f'Predicted Peak
({critical_date.strftime("%Y-%m-%d")})')
    plt.xlabel('Date')
    plt.ylabel('Post Count')
    plt.title(f'LPPLS Model Fit for Cyberincident Social Media Activity ')
    plt.legend()
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

def main():
    dates, post_counts, fitted_values, critical_date, confidence = analyze_and_forecast()
    if dates is not None:
        print(f"Predicted peak date: {critical_date.strftime('%Y-%m-%d')}")
        plot_results(dates, post_counts, fitted_values, critical_date, confidence)
    else:
        print("Failed to forecast due to insufficient data.")

if __name__ == "__main__":
    main()
```