

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

«На правах рукопису»
УДК 004.056

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри
_____ Дмитро ЛАНДЕ
«18» червня 2026 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи, технології та математичні
методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Автоматизація збору та нормалізація даних кіберрозвідки з використанням платформи аналізу кіберзагроз.

Виконав (-ла): здобувач (-ка) вищої освіти **IV** курсу, групи **ФБ-23**
(шифр групи)

_____ **Литвин Руслан Сергійович** _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент кафедри ІБ к.т.н., доцент Гальчинський Леонід Юрійович
(посада, науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, ім'я, прізвище) (підпис)

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

Київ – 2026 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)

«18» червня 2026 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

_____ Литвин Руслан Сергійович

(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизація збору та нормалізація даних кіберрозвідки з використанням платформи аналізу кіберзагроз.

керівник роботи доцент кафедри ІБ к.т.н., доцент Гальчинський Леонід Юрійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «29» травня 2026 р. №2001-с

2. Термін подання роботи здобувачем вищої освіти «10» червня 2026 р.

3. Вихідні дані до роботи: проблематика неповноти контексту даних кіберрозвідки в процесі оперативної діяльності SOC.

4. Зміст пояснювальної записки: вступ, теоретичні основи та проблематика кіберрозвідки, технологічні та архітектурні методи автоматизації обробки даних кіберрозвідки, програмна реалізація та тестування конектора.

5. Перелік графічного (ілюстративного) матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): презентація.

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Погодження теми дипломної роботи	15.03.2026	
2	Опрацювання наукової та технічної літератури	20.03.2026 – 20.03.2026	
3	Аналіз архітектурних методів обробки даних кіберрозвідки та графової моделі STIX	20.03.2026 – 20.04.2026	
4	Проектування конвеєра нормалізації розвідданих та аналіз цільового джерела	20.04.2026 – 25.05.2026	
5	Програмна реалізація конектора	20.04.2026 – 20.05.2026	
6	Тестування розробленого конектора та оцінка якості сформованих графів загроз	21.05.2026 – 17.05.2026	
7	Проходження переддипломної практики	20.04.2026 – 15.05.2026	
8	Оформлення дипломної роботи відповідно до методичних вказівок	28.05.2026 - 04.06.2026	
9	Передзахист дипломної роботи	10.06.2026	
10	Захист дипломної роботи	18.06.2026	

Здобувач вищої освіти

(підпис)

Руслан ЛИТВИН
(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Леонід ГАЛЬЧИНСЬКИЙ
(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота містить: 86 сторінок, 12 ілюстрацій, 2 таблиці, 1 додаток, 30 джерел літератури.

Об'єкт дослідження: процеси збору, обробки та управління даними кіберрозвідки в сучасних системах забезпечення інформаційної безпеки.

Предмет дослідження: методи та програмні засоби автоматизації збору, нормалізації і кореляції індикаторів компрометації на базі стандарту STIX 2.1 та архітектури платформи OpenCTI.

Мета роботи: підвищення ефективності обробки даних кіберрозвідки шляхом розробки конектора для автоматизованого збору, нормалізації та інтеграції неструктурованої телеметрії у стандартизований формат за допомогою платформи аналізу кіберзагроз OpenCTI.

Отримані результати: спроектовано та програмно реалізовано конектор для платформи OpenCTI. Використано багаторівневий конвеєр обробки даних для екстракції телеметрії із зовнішнього джерела Ransomfeed, який забезпечує, вирішення конфліктів атрибутів та відновлення контексту інцидентів. Застосовано алгоритм семантичного мапінгу неструктурованих індикаторів компрометації у графову модель стандарту STIX 2.1. Працездатність та ефективність створеного рішення підтверджено успішним тестуванням на масиві реальних кіберінцидентів.

Ключові слова: CTI, IoC, нормалізація даних, кіберзагрози, STIX.

ABSTRACT

The thesis contains: 86 pages, 12 figures, 2 tables, 1 appendices, and 30 references.

Object of research: processes of collecting, processing, and managing cyber threat intelligence data in modern information security systems.

Subject of research: methods and software tools for automating the collection, normalization, and correlation of indicators of compromise based on the STIX 2.1 standard and the OpenCTI platform architecture.

Aim of the work: to improve the efficiency of cyber threat intelligence data processing by developing a connector for the automated collection, normalization, and integration of unstructured telemetry into a standardized format using the OpenCTI threat intelligence platform.

Obtained results: a connector for the OpenCTI platform was designed and implemented. A multi-level data processing pipeline was used to extract telemetry from the external Ransomfeed source, which ensures the resolution of attribute conflicts and the restoration of incident context. An algorithm for the semantic mapping of unstructured indicators of compromise into the STIX 2.1 standard graph model was applied. The operability and efficiency of the created solution were confirmed by successful testing on a dataset of real cyber incidents.

Keywords: CTI, IoC, data normalization, cyber threats, STIX.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	8
ВСТУП	11
1 ТЕОРЕТИЧНІ ОСНОВИ ТА ПРОБЛЕМАТИКА КІБЕРРОЗВІДКИ	14
1.1 Роль та рівні кіберрозвідки в сучасних системах кіберзахисту	14
1.2 Критерії ідентифікації та класифікація розвідданих про загрози	16
1.3 Джерела інформації про загрози	19
1.4 Життєвий цикл обробки розвідданих	21
1.5 Проблематика збору та валідації індикаторів компрометації	24
Висновки першого розділу	26
2 ТЕХНОЛОГІЧНІ ТА АРХІТЕКТУРНІ МЕТОДИ АВТОМАТИЗАЦІЇ ОБРОБКИ ДАНИХ КІБЕРРОЗВІДКИ	27
2.1 Порівняльний аналіз систем управління кіберрозвідкою	27
2.2 Архітектура та принципи роботи платформи OpenCTI	29
2.3 Графова модель даних та стандарт STIX 2.1	31
2.4 Теоретичні основи процесів ETL в задачах CTI	32
2.5 Проблематика валідації та забезпечення цілісності розвідданих	34
2.6 Аналіз джерела розвідданих	37
2.7 Проектування конвеєра обробки даних	41
Висновки другого розділу	42
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КОНЕКТОРА	44
3.1 Програмна реалізація конектора	44
3.2 Тестування конектора	52
Висновки третього розділу	58
ВИСНОВКИ	59

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	61
ДОДАТОК А: ЛІСТИНГ КОДУ КОНЕКТОРА.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) — прикладний програмний інтерфейс;

APT (Advanced Persistent Threat) — цілеспрямована кібератака (складна стала загроза);

C2 (Command and Control) — командно-контрольний сервер інфраструктури зловмисників;

CERT (Computer Emergency Response Team) — команда комп'ютерного реагування на надзвичайні події;

CISO (Chief Information Security Officer) — керівник з інформаційної безпеки;

CTI (Cyber Threat Intelligence) — кіберрозвідка (аналітика кіберзагроз);

DFIR (Digital Forensics and Incident Response) — цифрова криміналістика та реагування на інциденти;

DGA (Domain Generation Algorithm) — алгоритм автоматичної генерації доменних імен;

EDR (Endpoint Detection and Response) — засоби виявлення загроз та реагування на рівні кінцевих точок;

ENISA (European Union Agency for Cybersecurity) — Агентство Європейського Союзу з кібербезпеки;

IDS/IPS (Intrusion Detection System / Intrusion Prevention System) — системи виявлення та запобігання вторгненням;

IEEE (Institute of Electrical and Electronics Engineers) — Інститут інженерів з електротехніки та електроніки;

IoC (Indicator of Compromise) — індикатор компрометації;

IR (Incident Response) — реагування на кіберінциденти;

ISAC (Information Sharing and Analysis Center) — галузевий центр аналізу та обміну інформацією;

L1/L2 (Line 1 / Line 2) — перша та друга лінії аналітиків центру реагування;

NIST (National Institute of Standards and Technology) — Національний інститут стандартів і технологій США;

NVD (National Vulnerability Database) — національна база даних вразливостей;

OSINT (Open Source Intelligence) — розвідка на основі відкритих джерел;

PIR (Priority Intelligence Requirement) — пріоритетні вимоги до розвідки;

RaaS (Ransomware-as-a-Service) — модель поширення шкідливого програмного забезпечення «програма-вимагач як послуга»;

SIEM (Security Information and Event Management) — система управління інформацією та подіями безпеки;

SOC (Security Operations Center) — центр оперативного реагування на інциденти кібербезпеки;

STIX (Structured Threat Information Expression) — стандартизована мова опису та обміну інформацією про кіберзагрози;

TIP (Threat Intelligence Platform) — платформа аналізу кіберзагроз (інформаційно-аналітична система);

TTPs (Tactics, Techniques, and Procedures) — тактики, техніки та процедури атакуючих;

CIRCL (Computer Incident Response Center Luxembourg) — Центр реагування на комп'ютерні інциденти Люксембургу;

ETL (Extract, Transform, Load) — процес алгоритмічного вилучення, трансформації та завантаження даних;

MISP (Malware Information Sharing Platform) — платформа спільного використання інформації про шкідливе програмне забезпечення;

REST (Representational State Transfer) — архітектурний стиль взаємодії компонентів розподіленого додатка в мережі;

SDO (STIX Domain Object) — об'єкт предметної області стандарту кіберрозвідки;

SRO (STIX Relationship Object) — об'єкт взаємозв'язку стандарту кіберрозвідки;

ВСТУП

Актуальність роботи. Щоденний обсяг індикаторів компрометації, який генерується OSINT-ресурсами та комерційними постачальниками, давно перевищив фізичні можливості ручного збору та валідації навіть укомплектованими командами SOC. Такі індикатори надходять великими обсягами та доволі швидко втрачають актуальність. Часто вони не мають необхідного контексту й технічних деталей, через що аналітику важко прийняти обґрунтоване рішення щодо алертів за їхньою участю. Таким чином, гостро постає питання автоматичного збору та збагачення даних кіберрозвідки, адже від цього напряму залежить ефективність роботи SOC.

Об’єкт дослідження: процеси збору, обробки та управління даними кіберрозвідки в сучасних системах забезпечення інформаційної безпеки.

Предмет дослідження: методи та програмні засоби автоматизації збору, нормалізації і кореляції індикаторів компрометації з використанням платформ аналізу кіберзагроз.

Метою роботи є підвищення ефективності обробки даних кіберрозвідки шляхом розробки конектора для автоматизованого збору, нормалізації та інтеграції неструктурованої телеметрії у стандартизований формат з використанням платформи аналізу кіберзагроз.

Завдання дослідження:

1. Проаналізувати теоретичні основи кіберрозвідки, життєвий цикл обробки ІоС та основну проблематику ручної валідації індикаторів компрометації.
2. Дослідити технологічні та архітектурні методи автоматизації обробки даних, включаючи порівняльний аналіз платформ аналізу кіберзагроз і застосування графової моделі стандарту STIX 2.1.
3. На основі аналізу особливостей цільового джерела даних розробити алгоритм агрегації даних кіберрозвідки та здійснити програмну реалізацію конектора.
4. Провести тестування розробленого конектора на платформі OpenCTI, оцінити ефективність та перевірити коректність формування графа загроз.

Методи дослідження. У роботі використано метод системного аналізу для дослідження процесів кіберрозвідки та її проблем, порівняльний аналіз для вибору платформи аналізу загроз, методи моделювання для створення зв'язків між об'єктами за стандартом STIX 2.1, методи програмної інженерії для безпосередньої розробки коду конектора, а також методи тестування та статистичного аналізу для перевірки роботи модуля й оцінки якості зібраних даних.

Наукова новизна одержаних результатів полягає у проектуванні багаторівневого конвеєра обробки даних кіберрозвідки з механізмом перехресної кореляції для автоматичної компенсації структурної неповноти цільового джерела.

Практичне значення одержаних результатів полягає у створенні конектора, який автоматизує процес збору, нормалізації та імпорту розвідданих із Ransomfeed до платформи OpenCTI.

Апробація результатів роботи. Основні положення, архітектурні рішення та практичні результати дипломної роботи доповідалися та обговорювалися на XXIV науково-практичній конференції студентів, аспірантів та молодих вчених Навчально-наукового фізико-технічного інституту КПІ ім. Ігоря Сікорського (м. Київ, 2026 р.).

Публікації. Основні наукові та практичні результати дипломної роботи висвітлено в 1 друкованій праці:

Литвин Р. С. Автоматизація процесів збору та систематизації розвідданих про кіберзагрози на базі платформи OpenCTI / Р. С. Литвин, Л. Ю. Гальчинський // Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених : матеріали конференції. — Київ : КПІ ім. Ігоря Сікорського, 2026. — С. 199–201.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА ПРОБЛЕМАТИКА КІБЕРРОЗВІДКИ

1.1 Роль та рівні кіберрозвідки в сучасних системах кіберзахисту

Знання IP-адрес або криптографічних хешів шкідливого програмного забезпечення не забезпечує об'єктивної оцінки намірів атакуючої сторони, доступного їй інструментарію чи первинного вектора компрометації, залишаючись набором розрізнених технічних артефактів. Формування повноцінної картини інциденту вимагає агрегації розширених метаданих та виявлення прихованих зв'язків між ідентифікованими індикаторами, що є основним завданням кіберрозвідки (CTI). Перехід від реактивного захисту до проактивного підтримання рівня захисту формує концепцію кіберрезильєнтності [21].

Відповідно до методології Palo Alto Networks, CTI розглядається як безперервний процес збору, глибинного аналізу та подальшої імплементації інформації щодо кіберзагроз, потенційних супротивників та актуальних методологій атак з метою масштабування безпекового контуру інфраструктури [1]. Фахівці інституту SANS розширюють зазначену парадигму, визначаючи кіберрозвідку як кінцевий аналітичний продукт інтерпретації телеметрії, здатний задовольнити специфічні вимоги щодо протидії векторам атак із доведеними намірами та технічною можливістю завдання шкоди [2,3]. Головним завданням CTI є збагачення контексту, шляхом доповнення даних телеметрією, кореляцію з тактиками, техніками і процедурами (TTPs).

Для забезпечення релевантної інтеграції процесів обробки даних у корпоративну архітектуру кіберзахисту здійснюється сегментація розвідданих за

рівнем абстракції та категорією кінцевого споживача, що традиційно базується на трирівневій моделі.

Стратегічний рівень. Формує макроскопічну оцінку глобального ландшафту загроз, де цільовими споживачами виступають керівники з інформаційної безпеки (CISO), рада директорів та архітектори систем. Фокус дослідження зміщується на довгострокові тренди, еволюцію геополітичних чи фінансових мотивацій АРТ-угруповань, повністю виключаючи оперування мережевими пакетами або сигнатурами. Головне завдання стратегічної розвідки полягає в аналітичному обґрунтуванні превентивних інвестицій — зокрема, у протидію компрометації ланцюгів постачання на основі галузевої специфіки організації [0, 4].

Оперативний рівень. Охоплює аналіз конкретних шкідливих кампаній та інфраструктури зловмисників, генеруючи дані для команд Threat Hunting, Incident Response та керівників SOC. Цей рівень забезпечує максимально деталізовану, прив'язану до інцидентів інформацію щодо активних атак, розгорнутих C2-серверів та векторів вторгнення. Оперативна розвідка ідентифікує виконавців поточних кіберзагроз та їхній інструментарій, дозволяючи вибудовувати проактивні механізми захисту до настання стадії безпосередньої експлуатації вразливостей (Exploitation phase за моделлю Cyber Kill Chain) [0, 2].

Тактичний рівень. Визначає робочий простір інженерів SOC та аналітиків першої і другої ліній (L1/L2), де здійснюється безпосередній розбір TTPs атакуючих із подальшим мапінгом на матрицю MITRE ATT&CK. На даному етапі досліджуються механізми закріплення зловмисника в системі (Persistence), специфічні API-виклики для викрадення облікових даних та шляхи ескалації привілеїв. Згенерована тактична аналітика конвертується у формалізовані правила детектування YARA, Sigma або кастомні кореляції в SIEM-системах, що забезпечує автоматизоване блокування загроз на рівні кінцевих точок (EDR) чи мережевого екрана [0, 4].

Розподіл масиву розвідданих відповідно до трирівневої моделі забезпечує безперервну трансформацію розрізненої телеметрії в контекстуалізований аналітичний продукт, чітко орієнтований на специфічні потреби кожного ешелону захисту. Завдяки впровадженню такої ієрархічної сегментації реалізується наскрізна операціоналізація кібербезпекових процесів, що охоплює єдиний континуум дій: від макроскопічного обґрунтування превентивних інвестицій до безпосередньої трансляції ідентифікованих TTPs у формалізовані правила блокування для SIEM та EDR.

1.2 Критерії ідентифікації та класифікація розвідданих про загрози

Ідентифікація телеметричних артефактів як кіберрозвідки вимагає обов'язкового проходження процедур алгоритмічної оцінки та контекстуалізації. Відповідно до методологічних настанов Національного інституту стандартів і технологій США (NIST SP 800-150), сирі системні журнали не визнаються розвідданими, доки вони не трансформуються в оперативні знання шляхом агрегації та атрибуції [5]. Фундаментальні дослідження у сфері обробки колективної кіберрозвідки підтверджують, що аналітична цінність інформації детермінується виключно її здатністю знижувати рівень невизначеності під час прийняття архітектурних рішень [6]. Без наявності семантичних зв'язків зібрані артефакти генерують надлишковий інформаційний шум.

У науковому дискурсі масиви даних, що формують СТІ, ієрархічно класифікуються за рівнем їхньої оперативної стійкості. Ця стратифікація, концептуалізована в моделі "Піраміда болю" (Pyramid of Pain) та адаптована

сучасною академічною спільнотою, структурує розвіддані за наступними категоріями (Рисунок 1.1) [7]:

- **Атомарні індикатори компрометації.** На базовому рівні абстракції акумулюються статичні параметри: криптографічні хеш-суми шкідливих файлів, скомпрометовані IP-адреси та доменні імена. Хоча такі дані забезпечують можливість миттєвої машинної обробки та інтеграції з EDR-системами, їхня тактична цінність мінімальна. Кіберугруповання легко генерують нові індикатори за допомогою алгоритмів DGA, знецінюючи статичні бази сигнатур.
- **Мережеві та локальні артефакти.** Даний сегмент містить патерни мережевого трафіку (наприклад, специфічні User-Agent під час комунікації з C2-сервером) та залишкові сліди модифікації файлової системи (створення м'ютексів, аномальні зміни у реєстрах ОС). Фіксація цих маркерів вимагає глибшого парсингу телеметрії, проте дозволяє надійно виявляти присутність шкідливого ПЗ на етапі закріплення (persistence).
- **Інструментарій супротивника.** До цієї категорії відноситься інформація про застосовувані утиліти для розвідки, ексфільтрації даних або реалізації латерального переміщення (lateral movement). Розпізнавання специфічних експлойтів чи кастомізованих фреймворків (подібних до модифікованих профілів Cobalt Strike) дозволяє ідентифікувати рівень технічної підготовки атакуючих.
- **Тактики, техніки та процедури (TTPs).** Найвищий рівень розвідданих. Ця інформація охоплює формалізовані поведінкові патерни зловмисників. Аналіз TTPs фокусується не на змінних індикаторах чи інструментах, а досліджує безпосередню методологію проведення атаки. Використання поведінкових індикаторів забезпечує побудову стійкого проактивного контуру захисту.

- **Метадані атрибуції (Threat Actors).** Стратегічна інформація, що визначає приналежність конкретних кампаній до відомих АРТ-груп. Формується розуміння фінансової або політичної мотивації супротивника, типових цілей за галузевою ознакою та географічної прив'язки інфраструктури.

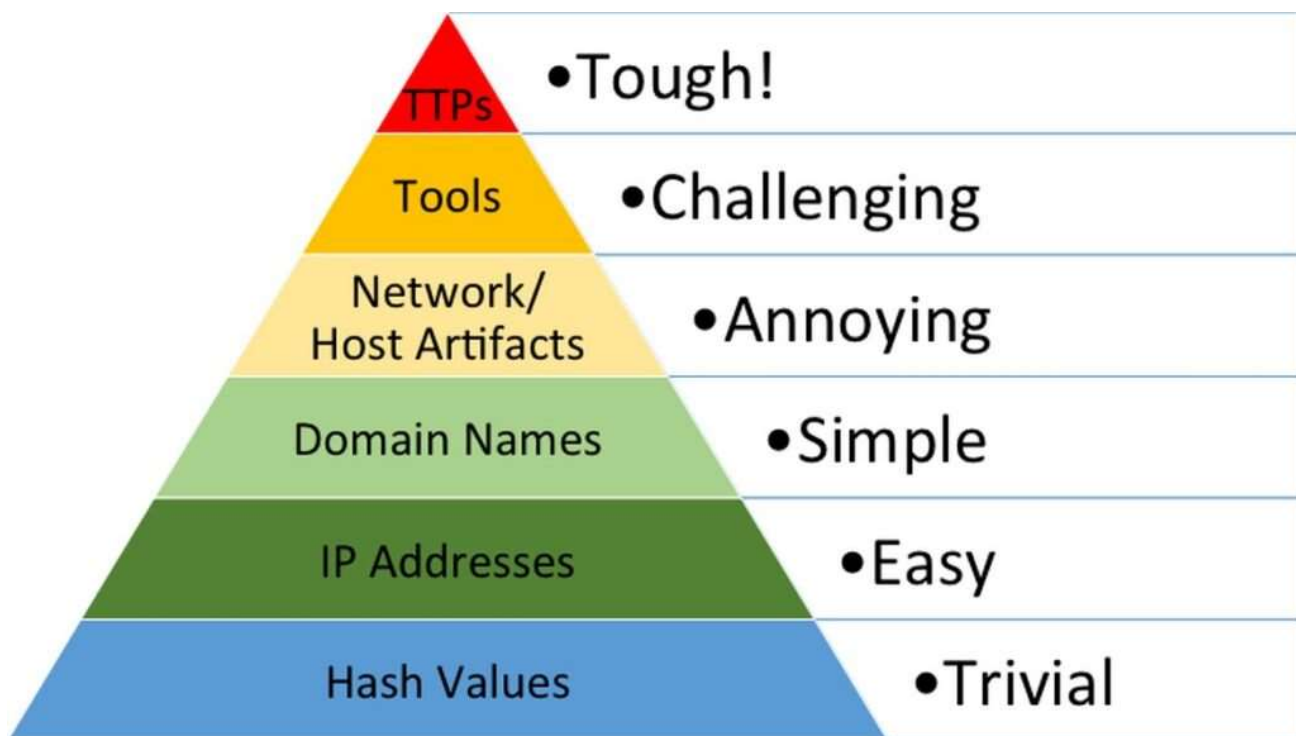


Рисунок 1.1 — Піраміда болю [7]

Наведена класифікація розвідданих дозволяє чітко визначати пріоритети під час моніторингу та раціонально розподіляти ресурси аналітиків. Зміщення фокусу з базових ІоС, які зловмисники можуть швидко змінити, на аналіз ТТРs ускладнює обхід засобів безпеки. Це дозволяє перехід від звичайного реагування на інциденти до побудови проактивної моделі кіберзахисту.

1.3 Джерела інформації про загрози

Формування об'єктивної картини ландшафту загроз безпосередньо детермінується якістю та гетерогенністю каналів збору телеметрії. У наукових дослідженнях архітектури систем колективної оборони (зокрема, у працях інституту IEEE щодо таксономії СТІ) доведено, що ізольоване використання єдиного вектора спостереження неминуче призводить до тактичної сліпоти безпекового контуру [Помилка! Джерело посилання не знайдено.]. Масиви інформації, необхідні для реалізації проактивного захисту, концептуально класифікуються за рівнем доступу, вектором походження та ступенем алгоритмічної обробки. Відповідно до методологічних рекомендацій Агентства Європейського Союзу з кібербезпеки (ENISA) та академічних праць у сфері управління інцидентами, екосистема джерел кіберрозвідки структурується за наступними напрямками [Помилка! Джерело посилання не знайдено., 10]:

- **Внутрішня телеметрія (Internal Intelligence).** Базовий рівень видимості формується за рахунок екстракції логів із SIEM, сенсорів виявлення вторгнень (IDS/IPS), EDR. Дослідження підтверджують, що власні мережеві артефакти та результати форензики (DFIR) забезпечують найвищий рівень релевантності. Вони об'єктивно відображають техніки атак, спрямовані безпосередньо на конкретну корпоративну інфраструктуру.
- **Відкриті джерела інформації (OSINT).** Глобальний масив неструктурованих даних акумулюється з публічних репозиторіїв шкідливого коду (VirusTotal, GitHub), технічних блогів вендорів, бюлетенів CERT та національних баз вразливостей (NVD). Практична експлуатація цього каналу ускладнюється критичним рівнем інформаційного шуму. Нефільтрований парсинг відкритих джерел провокує інтеграцію застарілих індикаторів, що неминуче призводить

до експоненційного зростання хибних спрацювань (false positives) під час моніторингу.

- **Комерційні фіди (Commercial Threat Feeds).** Інформаційні потоки, які розповсюджуються спеціалізованими провайдерами (Threat Intelligence Providers), характеризуються проходженням попередньої стадії нормалізації та контекстуалізації. Очищені розвіддані передаються кінцевому споживачу у машинозчитуваних форматах. Це забезпечує можливість оперативної інтеграції верифікованих TTPs безпосередньо у конфігурації засобів мережевого захисту.
- **Галузеві центри аналізу та обміну інформацією (ISAC).** Концепція колективної оборони реалізується через закриті профільні ком'юніті, де учасники певного сектору економіки (фінансового, енергетичного, державного) обмінюються специфічними індикаторами цілеспрямованих атак. У наукових працях Ф. Скопіка математично доведено, що галузевий обмін розвідданими радикально скорочує час реагування SOC-команд на профільні APT-кампанії [6].
- **Тіньовий сегмент (Dark/Deep Web Intelligence).** Агрегація інформації з цього специфічного вектору передбачає автоматизований скрейпінг хакерських форумів, тіньових маркетплейсів та ресурсів, контрольованих RaaS-утрупованнями (Ransomware-as-a-Service) [30]. Екстрагована телеметрія дозволяє фіксувати ранні маркери підготовки вторгнення, включаючи продаж скомпрометованих облікових даних (initial access brokering).

Підтримка оперативної актуальності та повноти СТІ досягається виключно шляхом безперервної агрегації даних із множинних гетерогенних джерел. Жоден ізольований канал не здатен забезпечити вичерпну видимість ландшафту загроз:

комбінування внутрішньої телеметрії з глобальними маркерами та галузевим контекстом дозволяє компенсувати недоліки окремих векторів, формуючи єдиний верифікований масив розвідданих.

1.4 Життєвий цикл обробки розвідданих

Трансформація розрізненої телеметрії у верифікований аналітичний продукт не є одноразовою або лінійною транзакцією. У наукових дослідженнях з архітектури кіберзахисту (зокрема, у фундаментальних працях В. Тунсі та Х. Раїса, опублікованих у IEEE Transactions on Information Forensics and Security) доведено, що цей механізм здатний генерувати релевантний контекст виключно за умови функціонування як безперервний ітеративний процес [11]. Концептуально ця методологія, відома як життєвий цикл кіберрозвідки (Threat Intelligence Lifecycle), була адаптована академічною спільнотою з класичних доктрин розвідувального циклу та оптимізована для алгоритмічної обробки цифрових масивів.

Відповідно до структурних моделей, затверджених інститутом SANS та дослідженнями у сфері управління кіберінцидентами, еталонний життєвий цикл розвідданих жорстко регламентується проходженням наступних фаз (Рисунок 1.2) [12, 13]:

- **Планування та визначення цілей (Direction and Planning).** Ініціалізація циклу починається з формування пріоритетних вимог до розвідки (PIRs). На цьому етапі архітектори безпеки визначають конкретні поверхні атак, що потребують моніторингу, та узгоджують метрики, за якими оцінюватиметься релевантність зібраної інформації відносно захищуваної інфраструктури.

- **Збір інформації (Collection).** Здійснюється масове акумулювання сирих даних з гетерогенних каналів: внутрішніх логів SIEM, комерційних фідів, OSINT-репозиторіїв та тіньових ресурсів. Специфіка даної фази полягає у безконтрольній генерації колосальних обсягів неструктурованого трафіку, який у первинному вигляді непридатний для машинного аналізу.
- **Обробка та нормалізація (Processing and Exploitation).** Екстраговані артефакти піддаються жорсткому парсингу та конвертації у стандартизовані машинозчитувані формати (наприклад, STIX/TAXII). Виконується алгоритмічна дедуплікація записів, фільтрація хибних спрацювань та збагачення базових індикаторів додатковими метаданими.
- **Аналіз (Analysis and Production).** Нормалізовані масиви проходять евристичну та статистичну кореляцію. Здійснюється мапінг виявлених артефактів на матрицю MITRE ATT&CK, виявляються приховані графові зв'язки між ізольованими інцидентами та формується кінцевий тактичний контекст атаки.
- **Поширення та інтеграція (Dissemination).** Згенеровані розвіддані маршрутизуються до систем споживання. Процес охоплює як автоматичний експорт сигнатур безпосередньо у конфігурації EDR/Firewall, так і доставку структурованих звітів керівникам SOC-підрозділів.
- **Зворотний зв'язок (Feedback).** Проводиться кількісна оцінка результативності застосованих TTPs. Виявлені відхилення або недоліки у детектуванні ініціюють автоматичне коригування початкових вимог PIRs на наступній ітерації циклу.

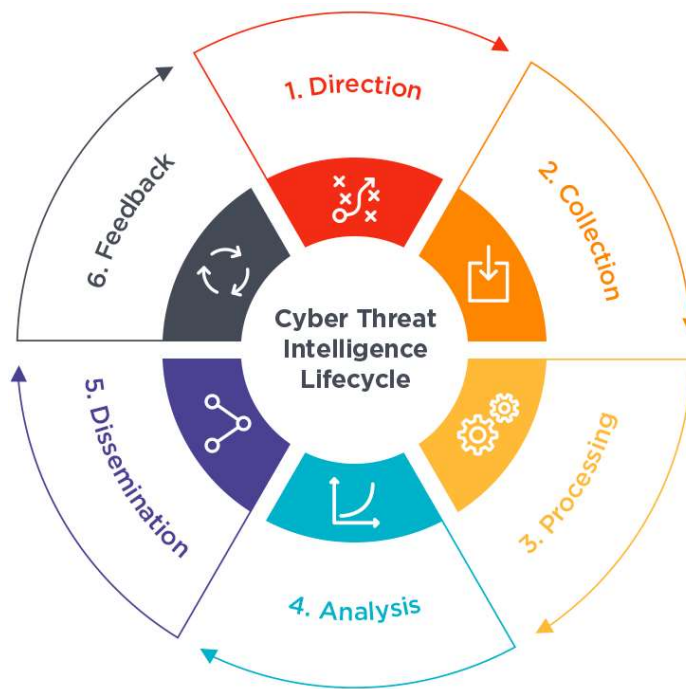


Рисунок 1.2 — Еталонна модель життєвого циклу кіберрозвідки [4]

Імплементація наведеного ітеративного фреймворку забезпечує системну трансформацію неструктурованої телеметрії у верифікований тактичний контекст. Водночас загальна операційна ефективність усього життєвого циклу критично детермінується пропускнуою здатністю на стику фаз збору та нормалізації. Саме на цих етапах акумулюється найбільший обсяг різномірних даних, що формує архітектурне «вузьке місце» та вимагає жорсткої алгоритмічної автоматизації для недопущення часових затримок перед стадією безпосереднього аналізу.

1.5 Проблематика збору та валідації індикаторів компрометації

Оперування неструктурованими масивами телеметрії в умовах реального часу формує комплекс критичних архітектурних викликів для підрозділів SOC. Як підтверджується дослідженнями ENISA щодо еволюції ландшафту загроз, безперервний потік IoCs створює інформаційне перевантаження, що фізично унеможлиблює процедуру їхньої ручної верифікації [9]. Експертний аналіз практичного застосування технічної кіберрозвідки (згідно з працями В. Тунсі та ін.) дозволяє класифікувати фундаментальні проблеми процесу інжесту та валідації за кількома ключовими векторами [12]:

- **Швидкісна деградація тактичної цінності.** Відповідно до концепції "Піраміди болю" Д. Б'янка, статичні артефакти, такі як IP-адреси або криптографічні хеш-суми, характеризуються надзвичайно коротким життєвим циклом [7]. Кіберугруповання масово використовують алгоритми генерації доменів (DGA), інструменти обфускації та динамічну зміну мережевої інфраструктури. Це призводить до миттєвого застарівання зібраних сигнатур ще до моменту їхньої остаточної імплементації в правила детектування.
- **Гетерогенність та відсутність синтаксичної уніфікації.** Агрегація телеметрії з множинних каналів (комерційні фіди, OSINT-майданчики, внутрішні сенсори) ускладнюється тотальною відсутністю єдиного формату представлення інформації. У дослідженнях інституту IEEE (М. Гонг та ін.) математично доведено, що екстракція релевантного контексту з неструктурованих текстів вимагає розробки складних кастомних парсерів [8]. Відповідно, ручний мапінг таких розрізнених сутностей генерує критичні часові затримки в загальному контурі реагування на кіберінциденти.

- **Дефіцит оперативного контексту та проблема хибних спрацювань.** Пряме завантаження "сирих" фідів безпосередньо у системи класу SIEM чи EDR закономірно провокує експоненційне зростання кількості false positives. Згідно з методологічними настановами NIST (SP 800-150), ізольований індикатор без жорсткої прив'язки до TTPs, атрибуції конкретної кампанії чи оцінки релевантності для захищуваної галузі перетворюється на звичайний інформаційний "шум" [5]. Опрацювання такого шуму стрімко виснажує когнітивні ресурси аналітиків першої та другої ліній (L1/L2).
- **Дублювання та конфлікти атрибуції.** Використання перехресних підписок на різні джерела розвідданих гарантовано призводить до отримання ідентичних індикаторів із суперечливим маркуванням. Наприклад, одна й та сама IP-адреса може мати різний рівень критичності або приписуватися до абсолютно різних АРТ-груп залежно від постачальника фіду. Вирішення подібних колізій потребує обов'язкової алгоритмічної дедуплікації та нормалізації скорингових метрик [13].

Вирішення описаних проблем вимагає комплексного та системного підходу до процесів збору й валідації даних. Для забезпечення ефективної роботи СТІ недостатньо просто агрегувати інформацію з різних джерел — необхідне обов'язкове впровадження ряду напрацьованих програмних рішень. Саме такі автоматизовані інструменти дозволяють своєчасно відкидати застарілі ІоС, усувати дублювання та приводити розрізнені формати до єдиного стандарту, гарантуючи якість та надійність розвідданих перед їхнім використанням у системах захисту.

Висновки першого розділу

У першому розділі проведено аналіз теоретичних основ СТІ. Розглянуто еталонну модель життєвого циклу обробки телеметрії, класифікацію розвідданих та ключові джерела їх збору. Визначено, що для ефективної протидії загрозам фокус моніторингу має зміщуватися з мінливих індикаторів (IoC) на аналіз поведінкових тактик зловмисників (TTPs).

Дослідження проблематики збору даних виявило критичні недоліки ручної валідації. Швидке застарівання індикаторів та відсутність єдиного формату інформації від різних постачальників створюють інформаційний шум, який перевантажує аналітиків SOC і призводить до хибних спрацювань.

Сформований теоретичний базис обґрунтовує технологічну потребу в автоматизації обробки СТІ. Розглянутий матеріал доводить необхідність розробки програмного модуля, який забезпечить автоматичну нормалізацію, фільтрацію дублікатів та стандартизацію розвідданих ще до їх потрапляння в системи захисту.

2 ТЕХНОЛОГІЧНІ ТА АРХІТЕКТУРНІ МЕТОДИ АВТОМАТИЗАЦІЇ ОБРОБКИ ДАНИХ КІБЕРРОЗВІДКИ

2.1 Порівняльний аналіз систем управління кіберрозвідкою

Архітектурне проєктування модулів агрегації вимагає вибору ТІР. На сьогодні де-факто галузевими стандартами open-source інфраструктури є платформи MISP та OpenCTI. Для обґрунтування технологічного вибору проведено порівняльний аналіз цих платформ за ключовими архітектурними критеріями:

- **Модель зберігання та онтологія даних.** Фундаментальна відмінність досліджуваних систем полягає в парадигмі структурування інформації. Відповідно до концептуальної архітектури, закладеної розробниками з організації CIRCL, платформа MISP спирається на подієво-орієнтовану реляційну модель [14]. Базовою сутністю виступає "Подія", до якої лінійно прив'язуються атомарні "Атрибути". Формування кореляційних зв'язків підтримується на рівні ядра, проте реалізується як алгоритмічна надбудова над плоскою базою даних.

Натомість ядро OpenCTI від моменту проєктування побудоване навколо концепції спрямованих графів та нативної імплементації онтології STIX 2.1 [15, 16]. Телеметрія індексується у вигляді незалежних об'єктів предметної області (SDO). Усі семантичні взаємодії між ними реєструються як самостійні об'єкти зв'язків (SRO). Для математичного моделювання складних багатовекторних інцидентів реляційна структура є архітектурно обмеженою. Графова логіка дозволяє зберегти глибокий семантичний контекст без деградації продуктивності при масштабуванні бази знань.

- **Аналітичний фокус та підтримка TTPs.** Системи вирішують різні класи інженерних завдань. Згідно з аналітичними звітами ENISA щодо платформ обміну даними, інструментарій MISP демонструє найвищу ефективність під час тактичного шарінгу розрізнених індикаторів — швидкої дистрибуції хеш-сум чи шкідливих IP-адрес на кінцеві засоби захисту [9]. Аналіз поведінкових патернів супротивника в цій моделі вторинний.

Натомість OpenCTI, жорстко сфокусована на оперативно-стратегічному рівні обробки телеметрії. Реалізовано прямий алгоритмічний мапінг зібраних артефактів на матрицю MITRE ATT&CK. Дані автоматично прив'язуються до конкретних TTPs ідентифікованих APT угруповань, формуючи цілісну модель вторгнення.

- **Механізми інтеграції та конвеєризації даних.** Автоматизація процесів інжесту висуває вимоги до транспортних інтерфейсів платформ. Інтеграція з MISP здійснюється переважно через синхронні виклики REST API за допомогою бібліотеки PyMISP. Як доводять дослідження пропускну здатності TTP-систем, обробка масивних неструктурованих фідів у синхронному режимі здатна генерувати блокувальні транзакції, виснажуючи ресурси обчислювального ядра [14, 10].

Інфраструктура OpenCTI нівелює проблему "вузьких місць" за допомогою двох технологічних рівнів. По-перше, транзакції до графової бази виконуються через інтерфейс GraphQL [25], що алгоритмічно унеможлиблює проблему надлишкової вибірки даних. По-друге, імплементовано асинхронну шину обміну повідомленнями на базі брокера RabbitMQ [16]. Зазначена архітектура першочергово адаптована під розробку ізольованих мікросервісів, які здійснюють зовнішній парсинг логів і асинхронно пушать нормалізовані STIX-об'єкти в чергу, гарантуючи абсолютну безперебійність ETL-процесів ядра платформи.

Вбудований функціонал MISP цілком задовольняє базові потреби оперативного обміну ізольованими індикаторами компрометації. Водночас, практична реалізація комплексного конектора для нормалізації гетерогенної телеметрії вимагає використання нативної графової моделі даних, підтримки поведінкового мапінгу та наявності асинхронної комунікаційної шини. Завдяки нативній графовій моделі та асинхронній шині даних платформа OpenCTI є оптимальним середовищем для розробки цільового модуля нормалізації розвідданих.

2.2 Архітектура та принципи роботи платформи OpenCTI

Проектування систем класу TIR для обробки високонавантажених потоків телеметрії зумовлює перехід до мікросервісної архітектури. Базовим обчислювальним компонентом екосистеми OpenCTI визначено серверне ядро, функціонування якого забезпечується середовищем виконання Node.js [15]. Топологію зв'язків між внутрішніми підсистемами та зовнішніми модулями інтеграції наведено на рисунку 2.1.

Маршрутизація запитів та обмін даними в межах платформи здійснюються через програмний інтерфейс GraphQL. На відміну від класичних REST-архітектур, застосування цього протоколу усуває дисбаланс вибірки даних (over-fetching/under-fetching), оскільки клієнтський рівень отримує можливість формувати специфіковані запити для екстракції цільових підграфів. Такий підхід оптимізує

мережеві транзакції під час кореляційного аналізу багатостадійних кіберінцидентів.

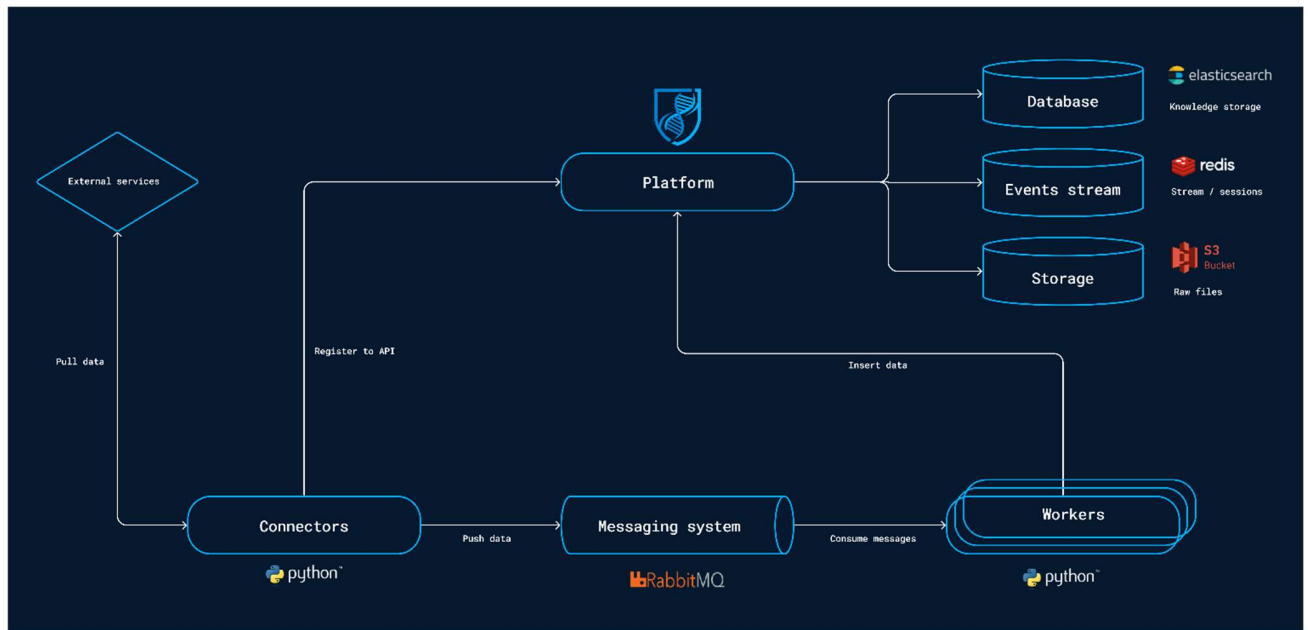


Рисунок 2.1 – Архітектура та екосистема платформи OpenCTI [16]

Архітектура рівня зберігання даних є гетерогенною, що дозволяє фізично ізолювати I/O-операції залежно від їхнього типу. Підтримка семантичної цілісності сутностей SDO та взаємозв'язків SRO покладена на спеціалізовану графову базу даних, тоді як повнотекстова індексація, логування та агрегація статистичних метрик виконуються пошуковим рушієм Elasticsearch [16, 27]. Додатково введено рівень In-Memory кешування на базі Redis, через який реалізується управління станом сесій та тимчасове зберігання транзитних об'єктів. Зазначене інженерне рішення мінімізує частоту звернень до постійної пам'яті в умовах масового інжесту індикаторів компрометації.

Конвеєризація потоків розвідданих побудована навколо брокера повідомлень RabbitMQ. Впровадження асинхронної комунікаційної шини алгоритмічно ізолює процеси екстракції зовнішньої телеметрії від її подальшої нормалізації та запису на диск [15]. Інтеграція з джерелами фідів відбувається через відокремлені сервіси (воркери), завданням яких є первинний парсинг сирих логів, трансформація

атрибутів у схеми STIX 2.1 та розміщення сформованих об'єктів у відповідних чергах. Споживання цих черг ядром платформи виконується асинхронно з урахуванням виділених апаратних квот, що запобігає внутрішнім відмовам обслуговування при аномальних сплесках вхідного трафіку.

2.3 Графова модель даних та стандарт STIX 2.1

Для стандартизації розрізнених потоків кіберрозвідки застосовується специфікація STIX 2.1, затверджена консорціумом OASIS. Замість плоских списків індикаторів компрометації, стандарт використовує принцип спрямованого графа. Обмін даними здійснюється у форматі JSON, структура об'єктів обмежена відповідними схемами. Це запобігає помилкам парсингу під час машинної обробки телеметрії. Концептуально архітектура стандарту розділяє інформацію на три базові класи сутностей [17]:

- **Об'єкти предметної області (SDO).** Клас призначено для формалізації абстрактних концепцій ландшафту кіберзагроз на стратегічному та тактичному рівнях. Номенклатура базових сутностей охоплює профілі атакуючих, масштабні кампанії, сімейства зловмисного коду та стандартизовані моделі атак. Введення цих об'єктів у граф дозволяє моделювати ресурсний потенціал та мотивацію суб'єктів загрози без прямої прив'язки до низькорівневих мережевих артефактів [15].
- **Кіберспостережувані об'єкти (SCO).** Група агрегує сиру мережеву та інфраструктурну телеметрію, що класифікується як доказова база компрометації. Множина SCO включає атомарні артефакти: IPv4/IPv6-адреси, криптографічні хеші, гілки системного реєстру та параметри TCP/UDP сесій.

У графовій топології зазначені артефакти індексуються як ізольовані вершини, репрезентуючи виключно технічні сліди вторгнення без апіорного контексту.

- **Об'єкти взаємозв'язків (SRO).** Механізм формування орієнтованих ребер виступає інструментом об'єднання розрізнених вузлів у цілісну аналітичну структуру. Типізація відношень регламентує семантику взаємодії компонентів. Зокрема, алгоритмічна приналежність мережевого індикатора до конкретного вірусного сімейства кодується ребром *indicates*, тоді як експлуатація вразливості визначеним угрупованням фіксується зв'язком *targets* [13].

Синтез вершин (SDO, SCO) та направлених ребер (SRO) генерує багатовимірний граф атрибуції інциденту. Така математична репрезентація розвідданих є архітектурною умовою для алгоритмічної ідентифікації складних патернів TTPs [22].

2.4 Теоретичні основи процесів ETL в задачах СТІ

Автоматизація збору та нормалізації розвідданих у системах класу TTP базується на застосуванні конвеєрної моделі ETL (Extract, Transform, Load). Цей підхід визначає абстрактну логіку перетворення сирової гетерогенної телеметрії на структуровані графи атрибуції. Процес розмежовується на три етапи, кожен з яких виконує регламентовану функцію машинного оброблення інформаційних масивів без прямого блокування кінцевої бази даних.

Перший етап — екстракція. Процедура полягає у вилученні первинного потоку даних із розрізнених зовнішніх джерел. Взаємодія з провайдерами кіберрозвідки реалізується через прикладні програмні інтерфейси (API) або механізми підписки на мережеві фіди. На цій стадії алгоритмічно забезпечується управління HTTP-сесіями, контроль лімітів запитів (rate limiting) та обробка пагінації. Завдання етапу зводиться до гарантованого завантаження неструктурованих масивів інформації в оперативну пам'ять обробника без внесення модифікацій у їхній початковий синтаксис.

Другий етап — трансформація. Сирий потік даних проходить процедуру десеріалізації та синтаксичного парсингу для вилучення атомарних артефактів. Наступним кроком виконується семантичний мапінг: вихідні атрибути трансформуються у стандартизовані об'єкти цільової онтології із паралельним обчисленням логічних взаємозв'язків між ними. Критичною складовою трансформації є очищення телеметрії від інформаційного шуму. Процедура охоплює алгоритмічну дедуплікацію сутностей, фільтрацію легітимних значень за допомогою статичних білих списків та відхилення нерелевантних індикаторів.

Третій етап — завантаження. Фінальна фаза ініціює передачу нормалізованих об'єктів до цільової платформи аналізу кіберзагроз. Для запобігання транзакційному перевантаженню основного ядра застосовується парадигма асинхронної маршрутизації. Згенеровані пакети даних публікуються у вигляді атомарних повідомлень до спеціалізованих черг комунікаційної шини (Message Broker). Модулі екстракції передають нормалізовану телеметрію незалежно від поточного статусу центральної бази даних, що гарантує відмовостійкість архітектури в умовах експоненційних стрибків вхідного навантаження.

2.5 Проблематика валідації та забезпечення цілісності розвіданих

Агрегація масивів телеметрії з розрізнених джерел супроводжується генеруванням широкого спектра синтаксичних та семантичних аномалій. Завантаження невалідованих даних до систем аналізу кіберзагроз провокує структурну деградацію бази знань. Для гарантування цілісності графів атрибуції імплементується архітектурний патерн раннього відхилення, що передбачає алгоритмічне усунення фундаментальних інтеграційних колізій на етапі, який передує запису об'єктів у постійну пам'ять [18].

- **Синтаксична невідповідність та конфлікти типів даних.** Десеріалізовані потоки часто містять нерегламентовані типи змінних або пропускають обов'язкові ідентифікаційні атрибути. Вирішення колізії забезпечується механізмом жорсткої валідації корисного навантаження на відповідність еталонним JSON-схемам формату STIX 2.1. Транзакції зі структурними аномаліями примусово скидаються на рівні ізольованих інтеграційних модулів, що унеможливорює транзакційне перевантаження обчислювального ядра платформи.
- **Дублювання індикаторів компрометації.** Паралельний інжест телеметрії з кількох джерел експоненційно збільшує обсяг пам'яті через багаторазовий імпорт ідентичних артефактів. Усунення проблеми реалізується імплементацією алгоритмів детермінованої генерації ідентифікаторів на основі криптографічного простору імен UUIDv5. Ідентичні телеметричні маркери отримують однакові хеш-суми, що дозволяє алгоритмічно зливати їх в єдину вершину графа з кумулятивним оновленням контексту [10].
- **Топологічні колізії в мережі взаємозв'язків.** Асинхронна обробка генерує специфічні стани, коли об'єкт зв'язку посилається на неіснуючу вершину, або

порушується онтологічна логіка. Нівелювання ризиків вимагає впровадження програмної верифікації легітимності орієнтованих ребер згідно з визначеними словниками стандарту. Відхилення нелегітимних транзакцій гарантує збереження математичної спрямованості графа.

- **Хронологічні парадокси.** Порушення синхронізації часу на стороні зовнішніх провайдерів призводить до ситуацій, коли мітка модифікації вузла математично передуює мітці його створення. Нейтралізація таких аномалій досягається через запровадження жорсткого хронологічного контролю та нормалізацію відміток до формату ISO 8601, що програмно блокує інверсію часових атрибутів перед їх завантаженням у шину.
- **Забруднення бази знань хибнопозитивними артефактами.** Нефільтровані потоки сирової телеметрії неминуче інкапсулюють легітимні мережеві ідентифікатори, такі як вузли мереж доставки контенту (CDN) або публічні DNS-сервери. Запобігання цій колізії алгоритмічно реалізується шляхом контекстної фільтрації. На етапі трансформації впроваджується проміжний шар звірки артефактів з еталонними білими списками (Allowlisting), що ініціює алгоритмічне блокування транзакції або примусове зниження показника шкідливості до нульового рівня.
- **Неконтрольоване застарівання даних.** Тактичні індикатори компрометації характеризуються вкрай обмеженим періодом актуальності. Збереження ефемерних артефактів без їхньої подальшої інвалідації отруює аналітичну базу хибними спрацюваннями. Компенсація цього вектора забезпечується запровадженням алгоритмічних моделей згасання (Decay Models). Застосування математичних функцій динамічно зменшує метрику довіри пропорційно до часу, ініціюючи автоматичне відкликання сутності після перетину встановленого порогу [19].

- **Конфлікт рівнів довіри.** Інтеграція гетерогенних фідів супроводжується призначенням взаємовиключних метрик критичності для однакових інцидентів з боку незалежних постачальників. Вирішення проблеми вимагає імплементації гранульованої системи зваженої оцінки. Платформа оперує метриками репутації джерел, що дозволяє зберігати під час злиття об'єктів той атрибут, який володіє найвищим математично підтвердженим рівнем довіри.
- **Помилки кодування та ін'єкції шкідливих навантажень.** Відсутність стандартизації локалей на стороні джерел призводить до імпорту невалідних символів або XSS-патернів у текстових полях описів. Нівелювання вразливості виконується через жорстку алгоритмічну санітизацію вхідного потоку. Застосування регулярних виразів для очищення корисного навантаження та примусова перекодировка символів перед десеріалізацією виключають ризики компрометації обчислювального середовища.

Ефективність імплементованих алгоритмів препроцесингової валідації виступає фундаментальним критерієм, що прямо пропорційно детермінує загальну якість акумульованих розвідданих, мінімізуючи рівень генерації хибнопозитивних спрацювань (False Positives) та гарантуючи математичну повноту і криптографічну достовірність кінцевих графів атрибуції. Враховуючи, що пропуск невалідованих транзакцій неминуче провокує каскадну деградацію всієї аналітичної бази знань, етап глибокого семантико-синтаксичного контролю класифікується як найбільш вразливий і критично значущий вузол конвеєризації, архітектурне проєктування якого вимагає безкомпромісної інженерної деталізації в межах усього циклу агрегації даних кіберрозвідки [23].

2.6 Аналіз джерела розвідданих

Джерелом первинних даних для модуля обрано сервіс Ransomfeed.it, попередня архітектурна апробація інтеграції з яким розглядалася у роботі [24]. Платформа агрегує сирі дані про інциденти та формує масиви метаданих щодо суб'єктів загроз і скомпрометованих організацій [20]. Взаємодія з платформою та отримання даних реалізується за допомогою трьох різних методів.

Основним механізмом є використання офіційного API сервісу. Формування HTTP GET-запитів здійснюється згідно з технічною документацією розробника (рисунок 2.2). У структурі запиту параметр `offset` вказує на бажану кількість записів у вибірці. Відповідно до обмежень платформи, для оптимізації процесу збору використовується максимально допустиме значення — 1000 записів за один виклик.

Під час дослідження особливостей роботи Ransomfeed було виявлено певні розбіжності у повноті та структурі інформації. Залежно від способу доступу, платформа могла повертати різні набори даних, а також відмінні значення окремих полів для одного й того ж запису. Щоб уникнути втрати даних, було прийнято рішення отримувати інформацію всіма доступними способами для її подальшої агрегації. Це зумовило необхідність реалізації ще двох непрямих методів збору.

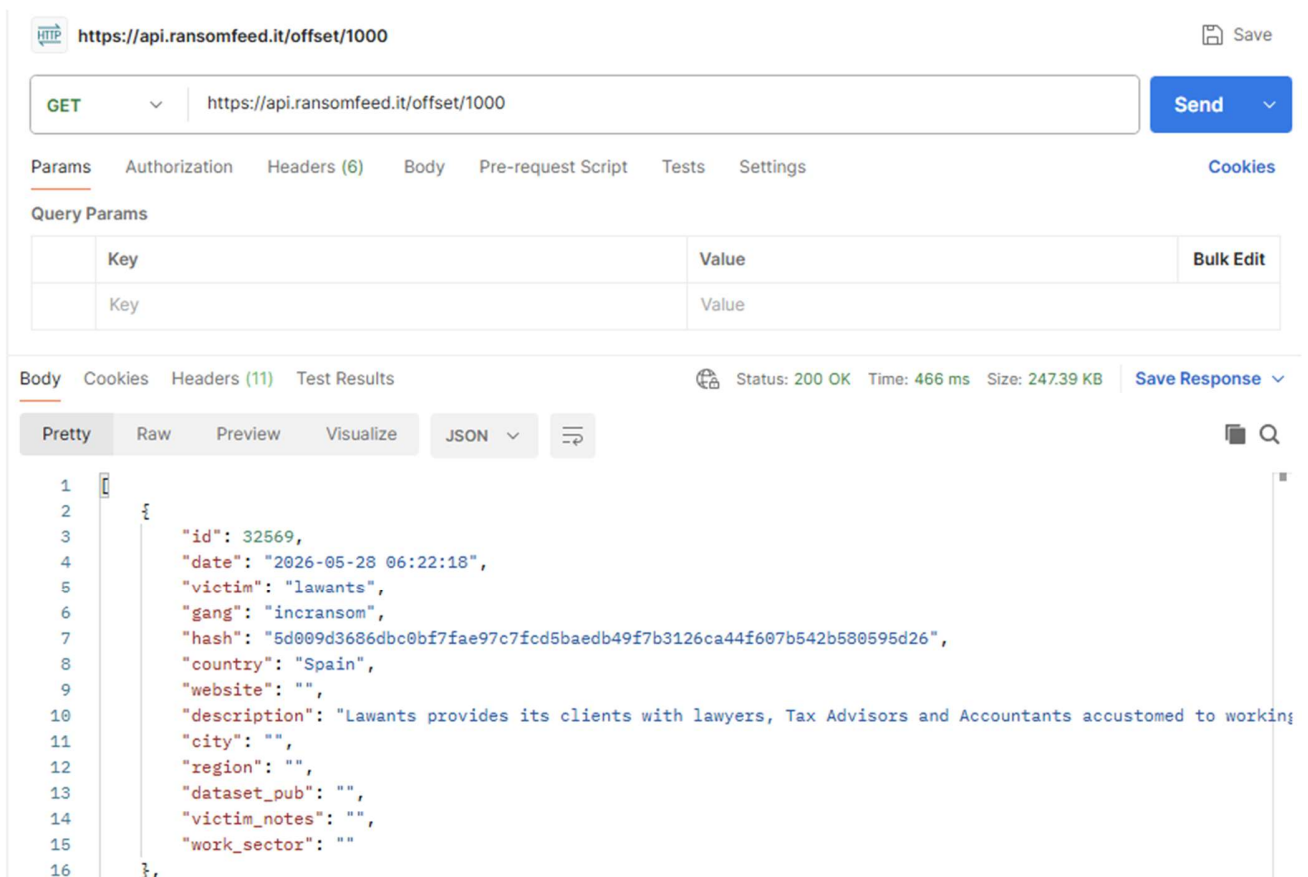
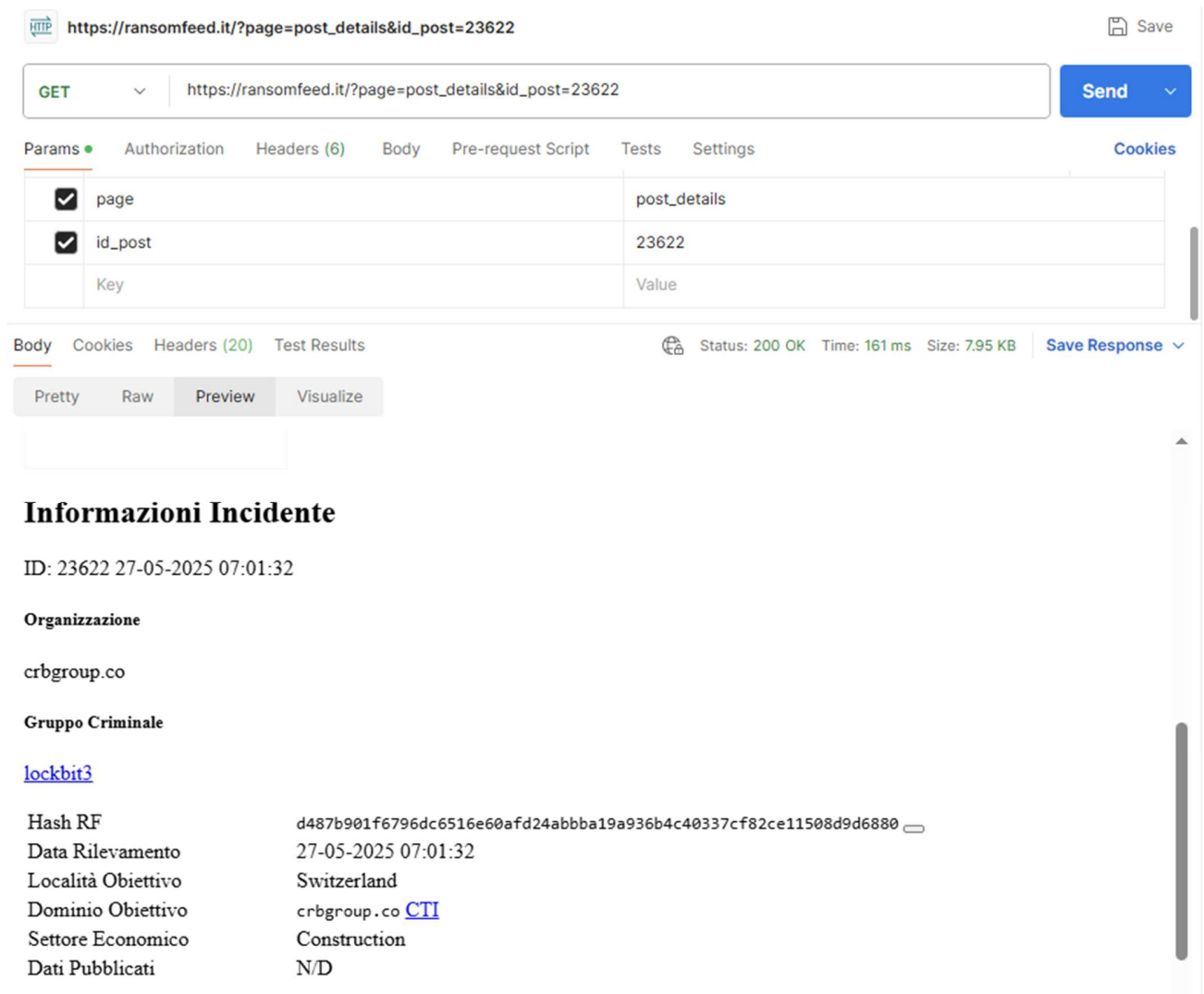


Рисунок 2.2 — Отримання даних про інциденти за допомогою API

Другий підхід полягає у скрапінгу сторінок окремих подій. На основі унікальних ідентифікаторів, попередньо отриманих через API, формуються цільові запити до відповідних сторінок інцидентів на платформі (рисунок 2.3). Оскільки сервер повертає вже згенерований HTML-документ, вилучення цільових даних здійснюється шляхом парсингу коду сторінки.



https://ransomfeed.it/?page=post_details&id_post=23622

GET https://ransomfeed.it/?page=post_details&id_post=23622 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

☒	page	post_details
☒	id_post	23622
	Key	Value

Body Cookies Headers (20) Test Results Status: 200 OK Time: 161 ms Size: 7.95 KB Save Response

Pretty Raw Preview Visualize

Informazioni Incidente

ID: 23622 27-05-2025 07:01:32

Organizzazione

crbgroup.co

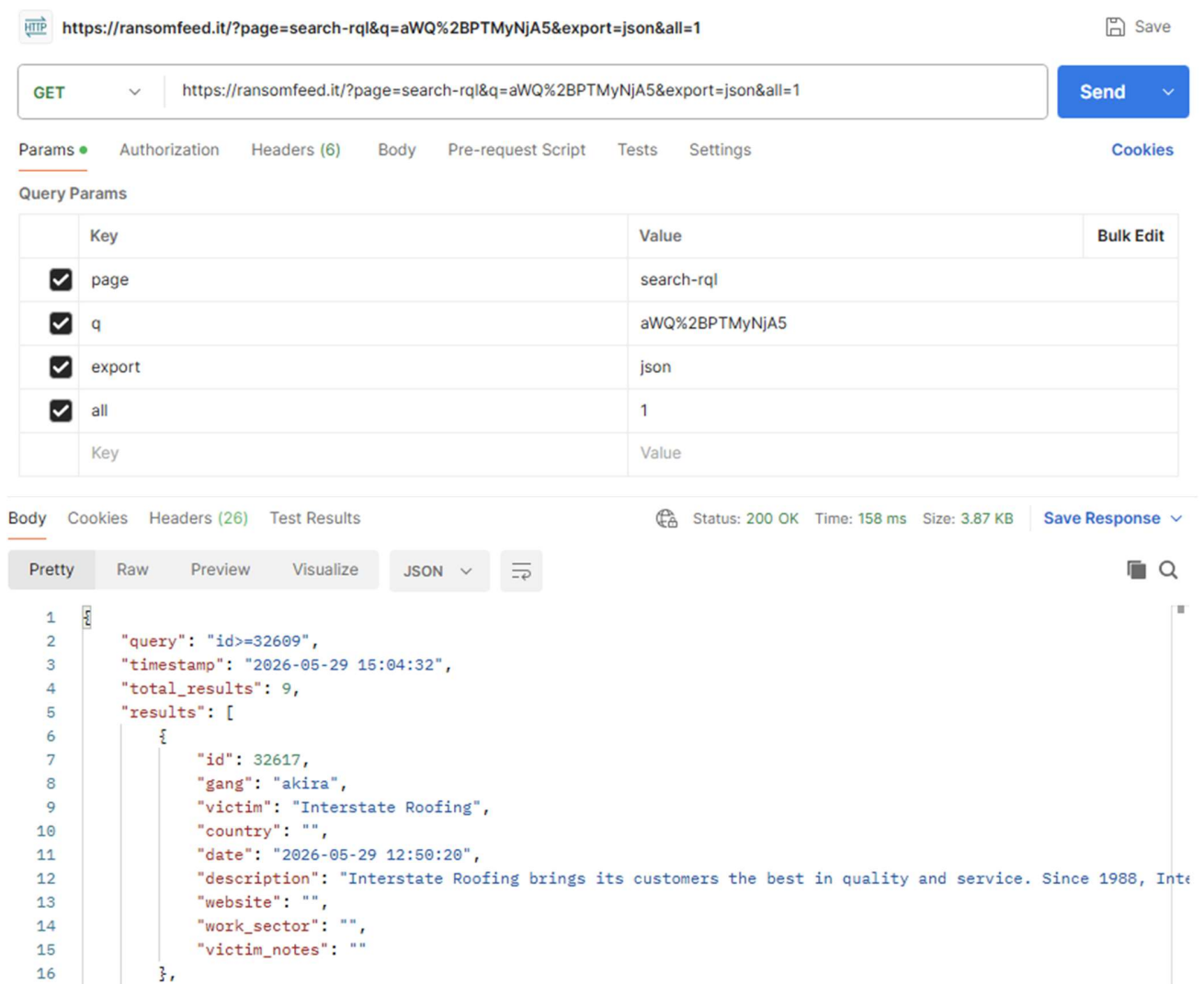
Gruppo Criminale

[lockbit3](#)

Hash RF	d487b901f6796dc6516e60afd24abbba19a936b4c40337cf82ce11508d9d6880
Data Rilevamento	27-05-2025 07:01:32
Località Obiettivo	Switzerland
Dominio Obiettivo	crbgroup.co CTI
Settore Economico	Construction
Dati Pubblicati	N/D

Рисунок 2.3 — Отримання інформації про інцидент за унікальним ID

Третій спосіб базується на використанні внутрішньої системи пошуку за допомогою мови запитів RQL. Формування запиту із зазначенням необхідного діапазону ідентифікаторів дозволяє отримати інформацію про пов'язані події (рисунок 2.4).



HTTP <https://ransomfeed.it/?page=search-rql&q=aWQ%2BTMyNjA5&export=json&all=1> Save

GET <https://ransomfeed.it/?page=search-rql&q=aWQ%2BTMyNjA5&export=json&all=1> Send

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
☒	page	search-rql	
☒	q	aWQ%2BTMyNjA5	
☒	export	json	
☒	all	1	
	Key	Value	

Body Cookies Headers (26) Test Results Status: 200 OK Time: 158 ms Size: 3.87 KB Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "query": "id>=32609",
3    "timestamp": "2026-05-29 15:04:32",
4    "total_results": 9,
5    "results": [
6      {
7        "id": 32617,
8        "gang": "akira",
9        "victim": "Interstate Roofing",
10       "country": "",
11       "date": "2026-05-29 12:50:20",
12       "description": "Interstate Roofing brings its customers the best in quality and service. Since 1988, Inte
13       "website": "",
14       "work_sector": "",
15       "victim_notes": ""
16     },

```

Рисунок 2.4 — Отримання інформації про інцидент з використанням RQL

Отримання даних одночасно кількома способами дає змогу визначити технічну специфіку розбіжностей та налаштувати коректні правила нормалізації. Такий підхід забезпечує формування достовірного та максимально повного контексту кіберзагроз перед їх подальшим аналізом.

2.7 Проектування конвеєра обробки даних

Враховуючи необхідність агрегації сирих даних з трьох різних джерел, базову архітектуру ETL (Extract, Transform, Load) було логічно розширено для забезпечення якісної попередньої обробки розвідданих. Процес трансформації даних було розділено на окремі підетапи для вирішення проблем розбіжностей у форматах. Таким чином, розроблений конвеєр інтеграції даних складається з наступних логічних кроків (рисунок 2.5):

- **Збір.** На початковому етапі здійснюється вивантаження первинної інформації про цільові об'єкти. Збір реалізується паралельно трьома методами: через офіційний API сервісу, шляхом скрапінгу HTML-сторінок та за допомогою пошукових запитів мовою RQL.
- **Нормалізація.** Оскільки кожен метод екстракції повертає дані у власній структурі, на цьому етапі розрізнені масиви приводяться до єдиного уніфікованого формату. Проводиться очищення та валідація даних. Також виконується мапінг назв країн та галузевих секторів відповідно до статичних словників для сумісності з сутностями цільової платформи.
- **Кореляція.** Наявність трьох потоків даних для ідентичних інцидентів вимагає їх точного зіставлення. На основі аналізу структури вхідних даних було розроблено правила кореляції для кожного окремого фіду. Це дозволяє усунути конфлікти атрибутів та сформувати на виході один об'єкт із максимально повним контекстом для кожної події.
- **Трансформація та завантаження.** Провалідовані об'єкти проходять фінальне перетворення у цільові формати та імпортуються до бази

даних платформи. Відправка та інтеграція сутностей реалізується за допомогою функціоналу бібліотеки русті.

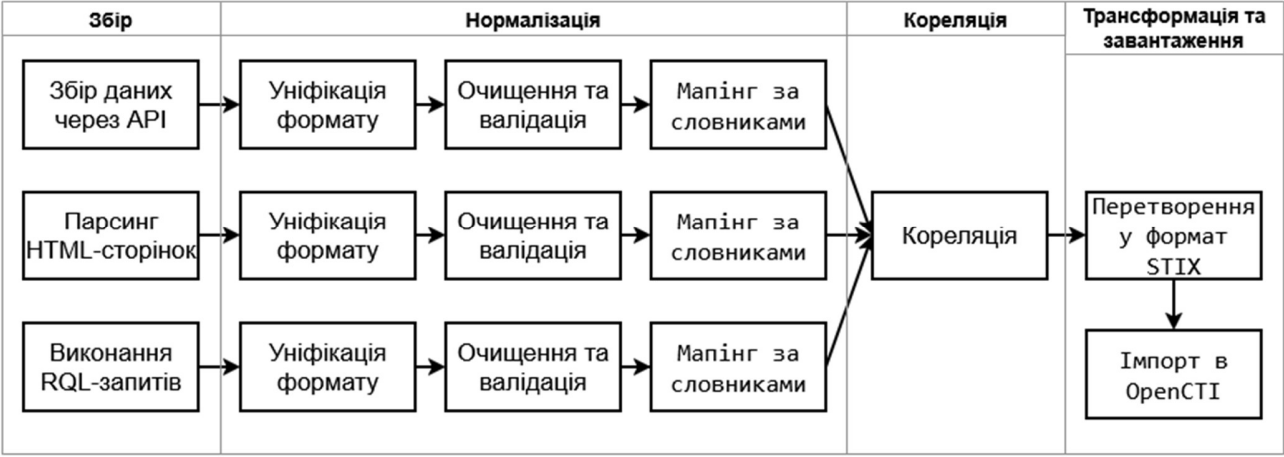


Рисунок 2.5 — Схема потоків даних конвеєра

Висновки другого розділу

У розділі проведено порівняльний аналіз платформ управління даними кіберрозвідки, за результатами якого обґрунтовано вибір OpenCTI як цільового середовища завдяки її мікросервісній архітектурі та нативній підтримці графових баз даних.

Досліджено внутрішню архітектуру платформи OpenCTI, основи моделювання загроз за стандартом STIX 2.1, концепцію побудови ETL-конвеєрів, а також ключові методи валідації та нормалізації розвідданих.

На основі сформованого теоретичного базису та практичного аналізу методів екстракції даних із цільового джерела (Ransomfeed) спроектовано багаторівневий

конвеєр. Запропонована архітектура забезпечує автоматизований збір, кореляцію та семантичну трансформацію неструктурованої інформації для подальшого імпорту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КОНЕКТОРА

3.1 Програмна реалізація конектора

Програмну реалізацію розробленого конектора виконано з використанням об'єктно-орієнтованого підходу та загальних принципів побудови конекторів. Архітектура системи базується на модульному принципі, що забезпечує логічне розділення компонентів: кожен файл відповідає за власну частину функціоналу. Повний лістинг вихідного тексту всіх програмних модулів та конфігураційних файлів наведено в Додатку А.

Для реалізації конектора було задіяно бібліотеки мови Python. Базовим інструментом для взаємодії з цільовою платформою OpenCTI виступає бібліотека `rusti` [29]. Мережева взаємодія та отримання первинних даних реалізуються за допомогою бібліотеки `requests`, а синтаксичний аналіз HTML-контенту для скрапінгу виконується за допомогою `beautifulsoup4` [28]. Робота з конфігураційними файлами та серіалізованими даними забезпечується модулями `PuYAML` і `json`. Для перевірки синтаксичної коректності мережових індикаторів застосовується бібліотека `validators`. Структурно програмний код конектора розділено на модулі.

3.1.1. Модуль конфігурації `config_loader.py`

Модуль `config_loader.py` призначений для централізованого завантаження та управління параметрами роботи конектора. Основне завдання компонента — забезпечення гнучкого налаштування інтеграційного конвеєра під час розгортання системи без необхідності внесення змін до вихідного коду.

Для обробки налаштувань реалізовано клас `RansomFeedConfig`. Логіка отримання конфігураційних значень базується на трирівневій системі пріоритетів. Програмний засіб виконує пошук та завантаження параметрів з трьох незалежних джерел у такій послідовності (від найвищого пріоритету до найнижчого):

1. **Змінні оточення.** Забезпечують динамічне налаштування модуля під час розгортання за допомогою `Docker`. Мають найвищий пріоритет і перекривають усі інші значення.
2. **Конфігураційний файл `config.yml`.** Локальний файл, який використовується для зберігання статичних параметрів середовища. Зчитування та парсинг файлу реалізовано за допомогою бібліотеки `PuYAML`.
3. **Стандартні значення.** Резервні параметри, жорстко задані на рівні програмного коду. Застосовуються лише у випадку відсутності відповідних налаштувань у змінних оточення та конфігураційному файлі.

Для забезпечення відмовостійкості етап завантаження конфігурації обгорнутий у блок обробки виключень. У разі неможливості зчитати критично важливі параметри, модуль формує звіт про помилку за допомогою бібліотеки `traceback`, фіксує його у системному журналі платформи через вбудований логер і припиняє виконання процесу.

3.1.2. Модуль збору та трансформації даних RansomFeed.py

Ключовим компонентом інтеграційного конвеєра є модуль RansomFeed.py. Він інкапсулює логіку взаємодії з платформою Ransomfeed, забезпечуючи автоматизований збір сирих даних, їх синтаксичний аналіз, приведення до єдиного формату (нормалізацію) та подальшу кореляцію. Архітектура модуля побудована на системі взаємопов'язаних класів, що розділяють відповідальність за різні етапи обробки даних.

Базовим елементом моделі даних є клас EventObj. Цей клас відповідає за валідацію та зберігання атрибутів окремого інциденту. Під час ініціалізації об'єкта відбувається первинна нормалізація сирих даних: очищення текстових полів від службових символів, форматування часових міток, перевірка синтаксису доменних імен за допомогою бібліотеки validators та мапінг країн і секторів економіки відповідно до заздалегідь визначених статичних словників.

Для уніфікації роботи з масивами подій реалізовано батьківський клас EventDict. Він не використовується як самостійний екземпляр, а виступає базовим класом для наслідування іншими класами. EventDict реалізує функціонал словника через визначення вбудованих магічних методів, де ключем виступає унікальний ідентифікатор інциденту, а значенням — екземпляр EventObj. Таке архітектурне рішення гарантує, що всі подальші класи збору даних матимуть ідентичний інтерфейс для доступу та обробки збережених об'єктів.

Безпосередня екстракція даних реалізується через три класи, які успадковують структуру EventDict та відповідають за конкретні вектори збору:

APIData — клас, що виконує базові HTTP GET-запити до офіційного API сервісу, отримує масив подій у форматі JSON та генерує відповідні пули об'єктів EventObj.

RQLData — клас для взаємодії з пошуковим рушієм платформи. Формує пошукові патерни на основі мови запитів RQL, кодує їх у формат Base64 та здійснює вибірку розширених наборів метаданих для цільового діапазону ідентифікаторів.

RansomPostDetail — модуль веб-скрапінгу. Використовує мережеві сесії requests.Session для завантаження HTML-документів сторінок інцидентів. Синтаксичний аналіз DOM-дерева виконується за допомогою бібліотеки 'BeautifulSoup', що дозволяє вилучати цільові індикатори безпосередньо з таблиць та інформаційних блоків розмітки.

Оркестрація перелічених компонентів здійснюється головним класом — MyRansomFeed. Він керує життєвим циклом об'єктів збору та відповідає за фінальну кореляцію даних. Клас ініціює вивантаження інформації через APIData, RQLData та RansomPostDetail. Отримавши три розрізнені масиви, алгоритм порівнює атрибути кожного ідентичного інциденту. Застосовуючи розроблені правила вирішення конфліктів та доповнення значень, система агрегує дані. Кінцевим результатом роботи класу є єдиний результуючий словник провалідованих інцидентів із максимально повним контекстом, готових до етапу трансформації та завантаження на цільову СТІ-платформу.

3.1.3. Модуль створення STIX-об'єктів `opencti_utils.py`

Модуль `opencti_utils.py` відповідає за перетворення підготовлених даних у стандартизований формат STIX 2.1 та їх завантаження на платформу. За цю логіку відповідає клас `OpenctiUtils`, який взаємодіє з API платформи за допомогою бібліотеки `rusti`.

Під час ініціалізації класу одразу створюються базові сутності. До них належить профіль автора (`Identity`), який вказує, хто саме додав розвіддані до системи, а також правила маркування TLP та RAP. Маркування TLP регламентує рівень конфіденційності та правила поширення інформації, тоді як RAP визначає, які активні дії можна виконувати з цими даними. Оскільки конектор запускається регулярно, система спочатку перевіряє, чи існують вже ці базові сутності на платформі. Якщо так — платформа просто повертає їхні ідентифікатори (ID), що запобігає появі дублікатів.

Основний процес формування даних реалізовано в методі `process_claim`. Він приймає словник із параметрами конкретного інциденту і будує з них зв'язані STIX-об'єкти (Рисунок 3.1). Робота методу починається з перевірки наявності ключових даних: назви компанії-жертви (`victim`) та угруповання вимагачів (`gang`).

Якщо ці дані присутні, алгоритм послідовно створює необхідні об'єкти:

1. Спочатку формується об'єкт жертви (`Organization`), до якого, за наявності, додається вебсайт компанії як зовнішнє посилання.
2. Якщо у словнику є інформація про країну та сферу діяльності компанії, створюються відповідні об'єкти локації (`Location`) та галузі (`Sector`). Вони одразу прив'язуються до компанії-жертви за допомогою STIX-зв'язків `located-at` (розташовується в) та `part-of` (є частиною).

3. Далі створюється об'єкт угруповання зломисників (Intrusion Set) та безпосередньо об'єкт самого кіберінциденту (Incident) із зазначенням дати фіксації.

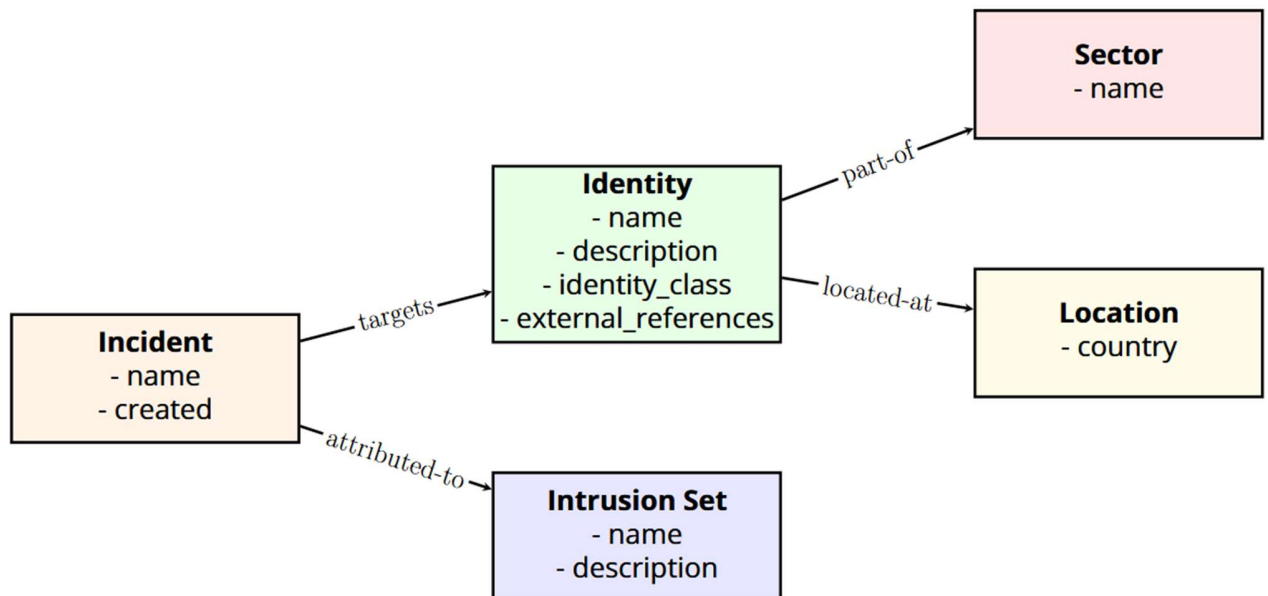


Рисунок 3.1 — Зв'язки між STIX об'єктами

На фінальному етапі всі ці розрізнені вузли поєднуються в єдиний граф. Система створює зв'язки, які вказують, що інцидент є справою рук конкретного угруповання (відношення *attributed-to*), а мішенню як самого інциденту, так і угруповання є скомпрометована організація (відношення *targets*).

Усі згенеровані об'єкти та зв'язки автоматично отримують маркування TLP і RAR, визначені на етапі ініціалізації.

3.1.4. Головний модуль ініціалізації та управління життєвим циклом `connector.py`

Модуль `connector.py` є головним оркестратором конвеєра. Управління всіма процесами реалізовано в класі `ConnectorRansomFeed`. Його ключове завдання — ініціалізувати робоче середовище та забезпечити регулярну синхронізацію розвідданих між зовнішнім джерелом і платформою OpenCTI.

Після ініціалізації конфігурацій запуск конектора здійснюється викликом методу `run`. Він використовує планувальник для циклічного виконання алгоритму, зазвичай із заданим інтервалом раз на добу.

Головною особливістю оркестратора є оптимізація мережевого навантаження завдяки роботі в інкрементному режимі. Перед початком збору метод `process_message` перевіряє поточний стан конектора на платформі, отримуючи ідентифікатор останньої успішно обробленої події (`last_id`). Це дозволяє системі не вивантажувати всю базу наново, а обробляти виключно ті інциденти, які з'явилися після попереднього циклу. Крім того, в OpenCTI ініціюється новий робочий процес, що дозволяє адміністраторам моніторити статус оновлення через графічний інтерфейс.

Далі отримане значення `last_id` передається до компонента `MyRansomFeed`, який здійснює збір та злиття нових даних трьома методами (API, RQL, скрапінг). Сформований масив нормалізованих інцидентів ітеративно передається до модуля `OpenctiUtils`, де кожен запис трансформується у набір STIX-сутностей та відправляється на платформу.

Після успішного завантаження всіх нових індикаторів конектор підбиває підсумки: фіксує кількість оброблених подій, оновлює свій глобальний стан на

платформі (зберігаючи новий `last_id` для наступного запуску) та позначає поточний робочий процес як успішно завершений.

3.1.5. Файл довідників `maps.json`

Файл `maps.json` містить словники для нормалізації назв країн та секторів економіки. Оскільки вхідні дані часто не збігаються з еталонними сутностями цільової платформи, цей файл забезпечує їх коректний мапінг та інтеграцію.

Розроблений програмний комплекс забезпечує повний цикл автоматизованого збору та попередньої обробки даних кіберрозвідки. Архітектурне рішення із застосуванням екстракції даних трьома методами та їх подальшої кореляції вирішує проблему структурних розбіжностей і неповноти первинного джерела. Завдяки вбудованим механізмам нормалізації, застосуванню статичних довідників та фінальній трансформації у формат STIX 2.1, створений конектор гарантує надходження до цільової платформи виключно валідованих розвідданих.

3.2 Тестування конектора

3.2.1. Оцінка якості зібраних даних

Для перевірки ефективності розробленого пайплайну було проведено тестування конектора на реальному масиві даних. Щоб наочно продемонструвати роботу алгоритмів нормалізації та кореляції, було згенеровано порівняльну таблицю як показано на рисунку 3.2. Вона показує події та їхні атрибути, які система автоматично виправила чи доповнила під час обробки, порівняно із даними, отриманими через API-інтерфейс джерела.

ID	Field	API (Standard)	RQL Data	HTML Data	Match	Status
32593	gang	everest	everest	everest	everest	OK
	victim	AKM	AKM	AKM	AKM	OK
	country	USA	USA	USA	USA	OK
	city					OK
	region					OK
	dataset_pub					OK
	date	2026-05-28T15:24:24Z	2026-05-28T15:24:24Z	2026-05-28T15:24:24Z	2026-05-28T15:24:24Z	OK
	website			akmenterprisesinc.com	akmenterprisesinc.com	DIFF
	work_sector	Defense industry	Defense industry	Defense industry	Defense industry	OK
	victim_notes					OK

Рисунок 3.2 — Порівняння атрибутів інциденту з різних джерел збору

Загалом під час тестування конектор проаналізував 3186 інцидентів. Отримані результати обробки даних було підраховано та зведено у загальну статистику (таблиця 3.1).

Таблиця 3.1 — Загальні результати кореляції кіберінцидентів

Показник	Кількість подій	Частка від загальної вибірки
Загальна кількість оброблених подій	3186	100 %
Подій без розбіжностей	614	19,27 %
Подій із конфліктами	2572	80,73 %

Як видно із загальних результатів, більшість записів потребувала втручання та коригування. Також було проаналізовано кількість виправлень по кожному атрибуту (таблиця 3.2).

Таблиця 3.2 — Виправлень за окремими атрибутами інцидентів

Атрибут	Кількість скоригованих подій	Відсоток від загальної вибірки (3186)
Вебсайт компанії-жертви	2392	75,08 %
Назва угруповання вимагачів	2187	68,64 %
Назва компанії-жертви	2187	68,64 %
Дата фіксації інциденту	2187	68,64 %
Країна реєстрації компанії	1982	62,21 %
Сектор економіки	1914	60,08 %
Дата публікації дампу	35	1,10 %

Статистика показує, що API-інтерфейс досліджуваного джерела є ненадійним каналом для побудови якісної СТІ-бази. Для понад 80 % інцидентів мають неточності або відсутні значення. Майже у 70 % випадків наявні неточності в назвах угруповань, жертва та дата атаки, а у 75 % випадків відсутній вебсайт.

3.2.2. Огляд конектора на платформі

Після запуску розроблений конектор успішно підключається до платформи. У розділі моніторингу відображається його активний статус та регулярний запуск процесів обробки даних (рисунок 3.3). Завдяки механізму збереження стану (state), система автоматично фіксує ідентифікатор останньої обробленої події (last_id) та час останньої синхронізації.

Процес імпорту показав, що зібрані та скориговані дані коректно переносяться у внутрішню структуру платформи. Зокрема, назви хакерських угруповань фіксуються як сутності типу Intrusion Set, а постраждалі компанії — як об'єкти типу Organization із прив'язкою до відповідного сектору економіки та географічного розташування. Історія створених об'єктів показана на рисунку 3.4.

Data / Ingestion / Ingestion monitoring / RansomFeed+

RansomFeed+ Active

Basic Information

Type	EXTERNAL_IMPORT	Last Update	Jun 3, 2026, 1:06:31 PM
Only Contextual	Not Applicable	Automatic Trigger	Not Applicable
Scope	Identity Location Intrusion-Set Incident Relationship External-Reference		

Details

State

```
{"last_run": 1780445292.561368, "last_id": 32686}
```

Last Run: Jun 3, 2026, 3:14:59 AM

Next Run: Jun 4, 2026, 3:14:59 AM

Server Capacity: 0.00 / 500 Mo

Рисунок 3.3 — Стан конектора в інтерфейсі OpenCTI

Activity		Timestamp ▲
	RansomFeed+ adds RansomFeed+ in Creators replaces TLP:WHITE in Definition	May 2, 2026, 7:42:18 AM
	RansomFeed+ creates a Marking-Definition PAP:WHITE	May 2, 2026, 7:42:18 AM
	RansomFeed+ creates a Organization RansomFeed+	May 2, 2026, 7:42:19 AM
	RansomFeed+ creates a External-Reference RansomFeed.it	May 2, 2026, 7:46:52 AM
	RansomFeed+ creates a Organization Rajagiri Hospital	May 2, 2026, 7:46:52 AM
	RansomFeed+ adds RansomFeed+ in Creators	May 2, 2026, 7:46:52 AM
	RansomFeed+ creates the relation located-at from Rajagiri Hospital (Organization) to India (Country)	May 2, 2026, 7:46:53 AM
	RansomFeed+ adds RansomFeed+ in Creators	May 2, 2026, 7:46:53 AM
	RansomFeed+ creates the relation part-of from Rajagiri Hospital (Organization) to Healthcare services (Sector)	May 2, 2026, 7:46:53 AM
	RansomFeed+ creates a Intrusion-Set thegentlemen	May 2, 2026, 7:46:53 AM
	RansomFeed+ creates a Incident Ransomware attack on Rajagiri Hospital	May 2, 2026, 7:46:54 AM
	RansomFeed+ creates the relation attributed-to from Ransomware attack on Rajagiri Hospital (Incident) to thege...	May 2, 2026, 7:46:54 AM
	RansomFeed+ creates the relation targets from Ransomware attack on Rajagiri Hospital (Incident) to Rajagiri H...	May 2, 2026, 7:46:54 AM

Рисунок 3.4 — Створення нових об'єктів на платформі

Обравши будь-який завантажений об'єкт, можна переглянути його взаємозв'язки через граф знань, де буде відображено пов'язані сутності (рисунку 3.5).

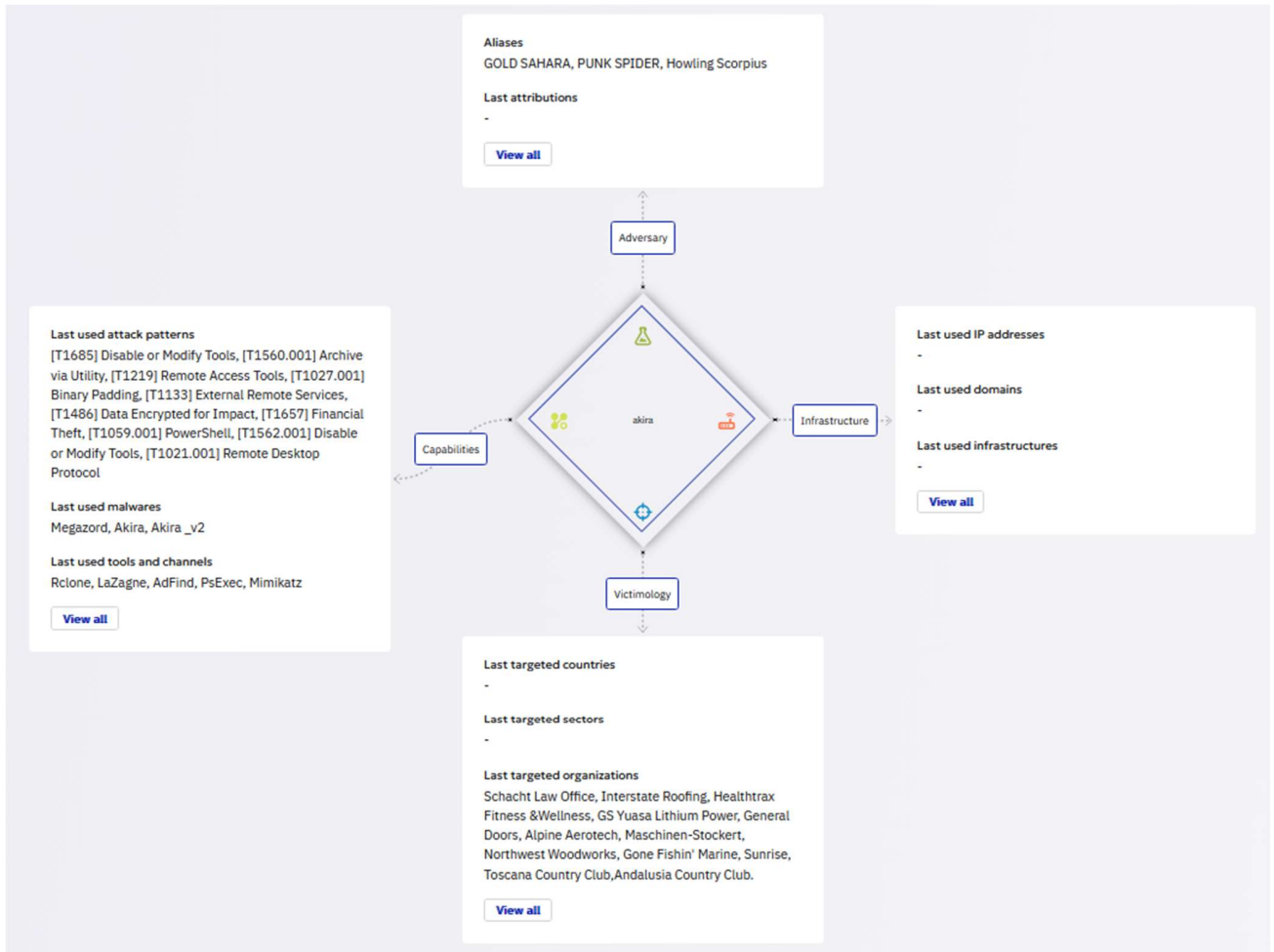


Рисунок 3.5 — Візуалізація побудованого графа знань для огрупування

Таким чином, було продемонстровано кінцевий результат роботи розробленого конектора. Він успішно завантажує зібрані та скориговані дані в платформу OpenCTI та автоматично перетворюючи їх на стандартизовані STIX-сутності.

Висновки третього розділу

У цьому розділі проаналізовано цільове джерело розвідданих, визначено оптимальні методи екстракції інформації та досліджено специфіку взаємодії з ним. На основі цих результатів спроектовано конвеєр обробки даних відповідно до поставлених завдань та виконано програмну реалізацію конектора.

Працездатність створеного модуля підтверджено під час тестування в середовищі платформи OpenCTI, яке продемонструвало успішний та коректний імпорт інформації. Крім того, здійснено оцінку якості первинних даних із базового API джерела Ransomfeed та доведено ефективність розробленого пайплайну в процесах валідації і кореляції зібраних розвідданих.

ВИСНОВКИ

У дипломній роботі вирішено актуальне науково-прикладне завдання щодо підвищення ефективності обробки даних кіберрозвідки шляхом автоматизації їх збору та нормалізації. За результатами виконання поставлених завдань зроблено такі висновки:

1. На основі аналізу теоретичних основ кіберрозвідки та життєвого циклу обробки телеметрії встановлено, що ручна валідація індикаторів компрометації є неефективною. Вона призводить до швидкого застарівання даних та перевантаження систем хибними спрацюваннями. Як результат, обґрунтовано технологічну необхідність автоматизації процесів екстракції та нормалізації розвідданих.
2. За результатами дослідження архітектурних методів обробки даних для розробки конвеєра обрано платформу OpenSTI завдяки її мікросервісній архітектурі, нативній графовій базі та наявності асинхронної комунікаційної шини. Для стандартизації розрізненої телеметрії обрано формат STIX 2.1, що дозволило зберегти семантичні зв'язки між технічними артефактами та високорівневими суб'єктами загроз.
3. За підсумками аналізу специфіки джерела виявлено структурну неповноту даних його базового API. Для вирішення цієї проблеми запропоновано багатовекторний алгоритм екстракції, що паралельно поєднує виклики API, пошукові RQL-запити та веб-скрапінг. На основі цього алгоритму розроблено та програмно реалізовано конектор мовою Python, який виконує кореляцію атрибутів та перетворення неструктурованих даних у STIX-сутності.

4. Проведено практичне тестування розробленого конектора на реальному масиві з 3186 кіберінцидентів. У результаті доведено ефективність розроблених алгоритмів кореляції, які автоматично виправили та доповнили втрачений контекст для 80,73 % подій. Експорт згенерованих об'єктів у середовище OpenCTI підтвердив коректність формування графа загроз, що свідчить про готовність конектора до впровадження в оперативні процеси SOC.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Palo Alto Networks. What Is Cyber Threat Intelligence (CTI)?. URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-cyberthreat-intelligence-cti>
2. Kime, B. P. Cyber Threat Intelligence Support to Incident Handling. SANS Institute Information Security Reading Room, 2017. URL: <https://www.giac.org/paper/gcih/27783/cyber-threat-intelligence-support-incident-handling/149720>
3. Lee, R. M. Intelligence Defined and its Impact on Cyber Threat Intelligence. 2016. URL: <https://www.robertmlee.org/intelligence-defined-and-its-impact-on-cyber-threat-intelligence/>
4. CrowdStrike. What is Cyber Threat Intelligence? [Beginner's Guide]. URL: <https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/>
5. Johnson, C., Badger, M., Waltermire, D. A., Snyder, J., & Skorupka, C. (2016). Guide to Cyber Threat Information Sharing (NIST Special Publication 800-150). National Institute of Standards and Technology.
6. Skopik, F., Settanni, G., & Fiedler, R. (2016). A problem shared is a problem halved: A survey on the dimensions of collective cyber threat intelligence. Computers & Security, 60, 154-171.
7. Bianco, D. (2014). The Pyramid of Pain. Enterprise Detection & Response.
8. Gong, M., et al. (2019). A Framework for Cyber Threat Intelligence Extraction from Heterogeneous Sources. 2019 IEEE International Conference on Intelligence and Security Informatics (ISI).

9. ENISA. (2020). Cyber Threat Intelligence: Understanding the Threat Landscape. European Union Agency for Cybersecurity.
10. Sauerwein, C., Sillaber, C., Mussmann, A., & Breu, R. (2017). Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives. *Wirtschaftsinformatik*, 837-851.
11. Tounsi, W., & Rais, H. (2018). A survey on Technical Threat Intelligence in the wild. *IEEE Communications Surveys & Tutorials*, 20(2), 1471-1502.
12. Farnham, G., & LeClair, J. (2020). Tools and Standards for Cyber Threat Intelligence Projects. SANS Institute Information Security Reading Room.
13. Mavroeidis, V., & Bromander, S. (2017). Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. 2017 European Intelligence and Security Informatics Conference (EISIC), 91-98.
14. Wagner, C., Dulaunoy, A., Wagener, G., & Iklody, A. (2016). MISP: The design and implementation of a collaborative threat intelligence sharing platform. *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 49–56.
15. Piazza, C. G., et al. (2022). Cyber Threat Intelligence representation and sharing: A survey on STIX. *Computers & Security*, 113, 102553.
16. Filigran. (2025). OpenCTI Platform Architecture and Ecosystem. Official Technical Documentation. URL: <https://docs.opencti.io/>
17. OASIS Cyber Threat Intelligence (CTI) TC. (2021). STIX Version 2.1. OASIS Standard. URL: <https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html>

18. Barnum, S. (2014). Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX). MITRE Corporation, 11, 1-22.
19. Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, 72, 212-233.
20. Ransomfeed. Corporate Victims Feed. URL: <https://ransomfeed.it/?page=corporate>
21. Lin Y. D. et al. Improving quality of indicators of compromise using STIX graphs. 2021. URL: <https://rnd.ntut.edu.tw/p/404-1042-146786.php?Lang=en>
22. Гальчинський Л., Личик В. Метрики оцінки кібервідмовостійкості (аналітичне оглядове дослідження) // Інформаційні технології та суспільство. — 2023. — 2 (8). — С. 27—33.
23. Universidad Complutense de Madrid. Comprehensive Analysis of Knowledge Management, Data Normalization, and Validation in Cyber Threat Intelligence. 2023.
24. Литвин Р. С. Автоматизація процесів збору та систематизації розвідданих про кіберзагрози на базі платформи OpenCTI / Р. С. Литвин, Л. Ю. Гальчинський // Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених : матеріали конференції. — Київ : КПІ ім. Ігоря Сікорського, 2026. — С. 199–201.
25. GraphQL Foundation. GraphQL Specification. 2025. URL: <https://spec.graphql.org/>
26. RabbitMQ. RabbitMQ Documentation. Messaging that just works. 2025. URL: <https://www.rabbitmq.com/documentation.html>

- 27.Elastic. Elasticsearch Guide. 2025. URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>
- 28.Richardson L. Beautiful Soup Documentation. 2025. URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
- 29.Filigran. OpenCTI Python Client (pycti). GitHub repository. 2025. URL: <https://github.com/OpenCTI-Platform/client-python>
- 30.Mandiant. M-Trends 2025: Ransomware Groups Evolution. Special Report. 2025. URL: <https://www.mandiant.com/resources/reports/m-trends>

ДОДАТОК А: ЛІСТИНГ КОДУ КОНЕКТОРА

Файл: src/maps.json

```
{
  "sectors": {
    "Avionics": "Air transport",
    "Automotive": "Road transport",
    "Apparel": "Heavy industries",
    "Real Estate / Real Estate Management": "Construction",
    "Attorney": "Legal consulting",
    "Services": "Consulting",
    "Utility": "Manufacturing",
    "Business Services": "Consulting",
    "Marketing": "Consulting",
    "Human Resources": "Consulting",
    "Church": "Citizens",
    "Cloud service providers": "Hosting service providers",
    "Distribution": "Retail (distribution)",
    "Fintech": "Finance",
    "Airlines": "Air transport",
    "Wealth Management": "Financial organizations",
    "Personal care": "Retail (distribution)",
    "Public Health": "Health",
    "Training": "Education",
    "Architecture": "Construction",
    "Academic Institutions": "Universities",
    "Hotel": "Lodging industry",
    "Luxury": "Retail (distribution)",
    "Charity associations": "Non-Governmental Organizations (NGOs)",
    "Human Services": "Healthcare services",
```

```

"Durables": "Manufacturing",
"Shops": "Retail (distribution)",
"Public sector": "Government and administrations",
"Embassy": "Diplomacy",
"Development": "International organizations",
"Critical Industries": "Heavy industries",
"Healthcare": "Health",
"Flight": "Air transport",
"Critical Sectors": "Heavy industries"
},
"countries": {
  "UK": "United Kingdom of Great Britain and Northern Ireland",
  "Hong Kong": "Hong Kong Special Administrative Region",
  "South Korea": "Republic of Korea",
  "UAE": "United Arab Emirates",
  "Czech Republic": "Czechia",
  "Moldova": "Republic of Moldova",
  "Russia": "Russian Federation",
  "Vietnam": "Viet Nam",
  "Portorico": "Puerto Rico",
  "Marocco": "Morocco",
  "Tanzania": "United Republic of Tanzania",
  "Venezuela": "Venezuela (Bolivarian Republic of)",
  "Iran": "Iran (Islamic Republic of)",
  "Bolivia": "Bolivia (Plurinational State of)",
  "Ivory Coast": "C\u00f4te d'Ivoire",
  "Union of the Comoros": "Comoros",
  "Palestine": "State of Palestine",
  "Principaut de Monaco": "Monaco",
  "Curacao": "Cura\u00e7ao",
  "Lettonia": "Latvia",
  "Giordania": "Jordan",

```

```

    "Laos": "Lao People's Democratic Republic",
    "Ukraina": "Ukraine",
    "Repubblica di Palau": "Palau",
    "Cipro": "Cyprus",
    "usa": "United States of America",
    "San Salvador": "El Salvador",
    "Principato di Andorra": "Andorra",
    "Repubblica de Cabo Verde": "Cabo Verde",
    "Virgin Islands": "United States Virgin Islands",
    "Macedonia": "North Macedonia",
    "Papua Nuova Guinea": "Papua New Guinea",
    "CANADA": "Canada",
    "india": "India",
    "italy": "Italy",
    "Republic of Vanuatu": "Vanuatu"
}
}

```

Файл: src/main.py

```

from ransomfeed import ConnectorRansomFeed
import traceback
if __name__ == "__main__":
    try:
        connector = ConnectorRansomFeed()
        connector.run()
    except Exception:
        print(traceback.format_exc())

```

Файл: src/ransomfeed/__init__.py

```

from .connector import ConnectorRansomFeed
__all__ = ['ConnectorRansomFeed']

```

Файл: src/ransomfeed/opencti_utils.py

```
from __future__ import annotations
```

```
from datetime import datetime
```

```
class OpenctiUtils:
```

```
    def __init__(nova, config: RansomFeedConfig, helper: OpenCTIConnectorHelper):
```

```
        nova.config = config
```

```
        nova.helper = helper
```

```
        nova.tlp_marking = nova.create_marking(nova.config.tlp_marking)
```

```
        nova.pap_marking = nova.create_pap_marking(nova.config.pap_marking)
```

```
        nova.identity = nova.create_identity()
```

```
    def create_marking(nova, level: str):
```

```
        mapping = {
```

```
            "white": {"definition_type": "TLP", "definition": "TLP:WHITE"},
```

```
            "clear": {"definition_type": "TLP", "definition": "TLP:WHITE"},
```

```
            "green": {"definition_type": "TLP", "definition": "TLP:GREEN"},
```

```
            "amber": {"definition_type": "TLP", "definition": "TLP:AMBER"},
```

```
            "amber+strict": {
```

```
                "definition_type": "TLP",
```

```
                "definition": "TLP:AMBER+STRICT",
```

```
            },
```

```
            "red": {"definition_type": "TLP", "definition": "TLP:RED"},
```

```
        }
```

```
        if level not in mapping:
```

```
            level = "clear"
```

```
        args = mapping[level.lower()]
```

```
        existing = nova.helper.api.marking_definition.read(
```

```
            filters={
```

```
                "mode": "and",
```

```
                "filters": [{"key": "definition", "values": [args["definition"]]}],
```

```
                "filterGroups": [],
```

```
            }
```

```
        )
```

```

    if existing:
        return existing
    return nova.helper.api.marking_definition.create(**args)
def create_pap_marking(nova, level: str):
    mapping = {
        "white": {"definition_type": "PAP", "definition": "PAP:WHITE"},
        "green": {"definition_type": "PAP", "definition": "PAP:GREEN"},
        "amber": {"definition_type": "PAP", "definition": "PAP:AMBER"},
        "red": {"definition_type": "PAP", "definition": "PAP:RED"},
    }
    if level.lower() not in mapping:
        level = "white"
    args = mapping[level.lower()]
    return nova.helper.api.marking_definition.create(**args)
def create_identity(nova):
    author = nova.helper.api.identity.create(
        type="Organization",
        name="RansomFeed+",
        identity_class="team",
        description=(
            "Questa API restituisce dati relativi agli attacchi ransomware
monitorati da RansomFeed. (This API returns data related to ransomware attacks
monitored by RansomFeed.)"
        ),
    )
    return author
def process_claim(nova, item):
    if not item.get("victim") and not item.get("gang"):
        return
    external_refs = []
    if item.get("website"):
        external_refs.append(nova.helper.api.external_reference.create(source_name
'RansomFeed.it', url = item.get("website"))['id'])
=

```

```

victim = nova.helper.api.identity.create(
    type="Organization",
    name=item["victim"],
    description=f"Victim of ransomware group {item['gang']}.
{item['description']}{'. ' + item['victim_notes']}",
    externalReferences=external_refs,
    objectMarking=[nova.tlp_marking["id"], nova.pap_marking["id"]],
    createdBy=nova.identity["id"]
)
if victim:
    if item.get("country"):
        country = nova.helper.api.location.create(
            type="Country",
            name=item.get("country")#,
        )
        if country:
            nova.helper.api.stix_core_relationship.create(
                fromId=victim["id"],
                toId=country['id'],
                relationship_type="located-at",
                description="Victim location",
                objectMarking=[nova.tlp_marking["id"],
nova.pap_marking["id"]],
                createdBy=nova.identity["id"]
            )
    if item['work_sector']:
        sector = nova.helper.api.identity.create(
            type="Sector",
            name=item['work_sector']#,
        )
        if sector:
            nova.helper.api.stix_core_relationship.create(
                fromId=victim['id'],

```

```

        toId=sector['id'],
        relationship_type="part-of",
        description="Operates in this sector",
        objectMarking=[nova.tlp_marking["id"],
nova.pap_marking["id"]],
        createdBy=nova.identity["id"]
    )
group = nova.helper.api.intrusion_set.create(
    name=item["gang"],
    aliases=[item["gang"]],
    objectMarking=[nova.tlp_marking["id"], nova.pap_marking["id"]],
    createdBy=nova.identity["id"]
)
incident = nova.helper.api.incident.create(
    name=f"Ransomware attack on {item['victim']}",
    description=f"Claimed by {item['gang']} on {item['date']}",
    first_seen=item["date"],
    created=item["date"],
    objectMarking=[nova.tlp_marking["id"], nova.pap_marking["id"]],
    createdBy=nova.identity["id"]
)
nova.helper.api.stix_core_relationship.create(
    relationship_type="attributed-to",
    fromId=incident["id"],
    toId=group["id"],
    objectMarking=[nova.tlp_marking["id"], nova.pap_marking["id"]],
    createdBy=nova.identity["id"]
)
nova.helper.api.stix_core_relationship.create(
    relationship_type="targets",
    fromId=incident["id"],
    toId=victim["id"],

```



```

        f.write(str(artifact) + '\n\n')
    print('ioc_counter', 'new_last_id', len(artifacts), artifacts.last_id)
    return len(artifacts), artifacts.last_id

def process_message(nova):
    current_state = nova.helper.get_state()
    now = datetime.now(tz=timezone.utc)
    friendly_name = "RansomFeed connector"
    work_id = nova.helper.api.work.initiate_work(
        nova.helper.connect_id, friendly_name
    )
    if current_state is not None:
        if "last_run" in current_state:
            last_seen = datetime.fromtimestamp(current_state["last_run"])
            nova.helper.log_info(f"Connector last run at: {last_seen} (UTC)")
        if "last_id" in current_state:
            last_id = current_state["last_id"]
    else:
        nova.helper.log_info("Connector has never run")
        last_id = None
    st_time = now.timestamp()
    ioc_counter, new_last_id = nova._collect_intelligence(last_id)
    message = (
        "Connector successfully run ("
        + str(ioc_counter)
        + " events have been processed), storing last_run as "
        + str(now)
    )
    nova.helper.api.work.to_processed(
        work_id,
        "All operations completed, marking work as complete"
    )
    nova.helper.log_info(message)

```

```

nova.helper.set_state(
    {
        "last_run": st_time,
        "last_id": new_last_id
    }
)
def run(nova):
    nova.helper.schedule_iso(message_callback=nova.process_message,
duration_period=nova.config.duration_period,)

```

Файл: src/ransomfeed/RansomFeed.py

```

import base64
import requests
from bs4 import BeautifulSoup
from datetime import datetime
import validators
from pathlib import Path
import os
import json
class EventDict:
    def __init__(nova):
        nova.info = None
        nova.victiums = {}
    def __str__(nova):
        str_output = ''
        for v in nova.victiums.values():
            str_output += str(v) + '\n\n'
        return(str_output)
    def __getitem__(nova, item_id):
        try:
            return nova.victiums[item_id]
        except(KeyError):

```

```

        return EventObj({}, False)
def __contains__(nova, item_id):
    return item_id in nova.victiums
def __iter__(nova):
    return iter(nova.victiums.values())
def __len__(nova):
    return sum(1 for obj in nova.victiums.values() if obj.id != -1)
def values(nova):
    return nova.victiums.values()
class EventObj:
    __slots__ = [
        'id', 'hash', 'gang', 'victim', 'country', 'city',
        'region', 'dataset_pub', 'date', 'website',
        'work_sector', 'victim_notes', 'description'
    ]
    _map_sectors = None
    _map_countries = None
    def __init__(nova, obj, dayfirst:bool):
        nova._load_maps()
        for fld_name in nova.__slots__:
            fld = str(obj.get(fld_name, '')).replace('\n\n CTI', '').strip()
            if not fld:
                continue
            if fld.lower() in ['n/d', 'unknown', 'non disponibile']:
                continue
            match(fld_name):
                case 'id':
                    nova.id = int(fld) if fld != '' else -1
                case 'dataset_pub':
                    nova.dataset_pub = fld.strip(' ')[0] if fld != '' else fld
                case 'date':
                    if fld == '':

```

```

        nova.date = ''
    else:
        if dayfirst:
            tmp = datetime.strptime(fld, "%d-%m-%Y %H:%M:%S")
        else:
            tmp = datetime.strptime(fld, "%Y-%m-%d %H:%M:%S")
        nova.date = tmp.strftime("%Y-%m-%dT%H:%M:%SZ")
    case 'website':
        nova.website = nova._url_clean(fld, ('http://', 'https://',
        'https:', 'http:', 'https:', 'http;', 'https;', 'www.', 'http', 'https'))
    case 'country':
        raw_country = fld.split(',')[0]
        nova.country = nova._map_countries.get(raw_country,
raw_country)
    case 'work_sector':
        nova.work_sector = nova._map_sectors.get(fld, fld)
    case _:
        setattr(nova, fld_name, fld)

@staticmethod
def _url_clean(field, pr_list:list):
    for pr in pr_list:
        field = field.removeprefix(pr)
    domain = field.lower().strip().split('/')[0]
    if validators.domain(domain.replace('*', '')):
        return domain
    else:
        return ''

def _load_maps(nova):
    if not (EventObj._map_sectors and EventObj._map_countries):
        config_file_path = Path(__file__).parents[1].joinpath('maps.json')
        if not os.path.isfile(config_file_path):
            return
        with open(config_file_path) as f:

```

```

        conf = json.load(f)
        EventObj._map_sectors = conf.get('sectors')
        EventObj._map_countries = conf.get('countries')
def __lt__(nova, other):
    if not isinstance(other, EventObj):
        return NotImplemented
    return nova.id < other.id
def __getitem__(nova, key):
    return getattr(nova, key)
def __setitem__(nova, key, value):
    setattr(nova, key, value)
def __getattr__(nova, name):
    if name in nova.__slots__:
        return '' # <-----
---- '' | None
        raise AttributeError(f"'{type(nova).__name__}' object has no attribute
'{name}'")
def get(nova, key):
    return nova.__getitem__(key)
def __str__(nova):
    header = f"--- RQL RECORD: {nova.victim or 'UNKNOWN'} ---"
    footer = "-" * len(header)
    lines = [
        header,
        f"ID          : {nova.id}",
        f"Hash          : {nova.hash}",
        f"Gang          : {nova.gang}",
        f"Victim         : {nova.victim}",
        f"Location       : {nova.country}" + (f", {nova.region}" if nova.region
else "") + (f" ({nova.city})" if nova.city else ""),
        f"Published      : {nova.dataset_pub}",
        f>Date         : {nova.date}",
        f"Sector         : {nova.work_sector}",

```

```

        f"Website      : {nova.website}",
        f"Notes        : {nova.victim_notes}",
        f"Description    : {nova.description}",
        footer
    ]
    return "\n".join(lines)
def __repr__(nova):
    return f"EventObj(id='{nova.id}', victim='{nova.victim}')"
class RQLData(EventDict):
    def __init__(nova, rql):
        nova.rql = rql
        super().__init__()
        nova.info = nova._get_info()
        nova.report_processing()
    def report_processing(nova):
        for obj_vic in nova.info['results']:
            temp = EventObj(obj_vic, False) # 2026-01-08 18:44:45
            nova.victiums[temp.id] = temp
    def _get_info(nova):
        rql64 = base64.b64encode(nova.rql.encode('utf-8'))
        return requests.get('https://ransomfeed.it', params={'page':'search-rql',
'q': rql64, 'export':'json', 'all':'1'}).json()
class APIData(EventDict):
    def __init__(nova):
        super().__init__()
        nova.info = nova._get_info()
        nova.report_processing()
    def _get_info(nova):
        return requests.get('https://api.ransomfeed.it/offset/1000').json()
    def report_processing(nova):
        for v in nova.info:
            temp = EventObj(v, False)

```

```

        nova.victiums[temp.id] = temp
class RansomPostDetail(EventDict):
    def __init__(nova):
        super().__init__()
        nova.ss = requests.Session()
        nova.ss.headers.update({
            'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:147.0)
Gecko/20100101 Firefox/147.0',
            'Accept': 'application/json'
        })
    def _get_info(nova, id):
        try:
            res = nova.ss.get('https://ransomfeed.it/?page=post_details&id_post='+str(id)).text
            soup = BeautifulSoup(res, 'html.parser')
            first_price = soup.find(class_='table-responsive').find('table')
            if first_price is None:
                print(f'Lol: {id}')
                return None
            rows = first_price.find_all('tr')
            obj = {'id':id}
            for row in rows:
                cells = row.find_all(['td', 'th'])
                data = [cell.text.strip().replace('\n\n CTI', '').replace('N/D',
'') for cell in cells]
                match (data[0]):
                    case 'Hash RF':
                        obj['hash'] = data[1]
                    case 'Data Rilevamento':
                        obj['date'] = data[1]
                    case 'Località Obiettivo':
                        obj['country'] = data[1]
                    case 'Dominio Obiettivo':

```

```

        obj['website'] = data[1]
    case 'Settore Economico':
        obj['work_sector'] = data[1]
    case 'Dati Pubblicati':
        obj['dataset_pub'] = data[1]
    case 'Descrizione':
        obj['description'] = data[1]
    case 'Stesso Target su Altri Gruppi':
        pass
    case 'Note Interne':
        pass
    case 'Rivendicazioni Collegate':
        pass
    case _:
        raise ValueError(f"Parsing error, new field in table:
'{data[0]}:{data[1]}'")
    cards = soup.find_all("div", class_="rsf-info-card")
    for card in cards:
        header_tag = card.find("h5")
        if not header_tag:
            continue
        value_tag = header_tag.find_next("p")
        header_text = header_tag.get_text(strip=True)
        if "Organizzazione" in header_text:
            obj['victim'] = value_tag.get_text(strip=True)
        elif "Gruppo Criminale" in header_text:
            obj['gang'] = value_tag.get_text(strip=True)
    except:
        return None
    return obj
def get_range(nova, start_id, end_id):
    if start_id > end_id:

```

```

        raise ValueError(f"Range error: '{start_id}' > '{end_id}'")
    for id in range(start_id, end_id+1):
        temp = nova._get_info(id)
        if temp is None:
            continue
        event_obj = EventObj(temp, True)
        nova.victiums[event_obj.id] = event_obj
class MyRansomFeed(EventDict):
    def __init__(nova, start_id = None):
        super().__init__()
        nova.st_id = start_id
        nova.last_id = start_id
        nova.fields = [
            'id', 'hash', 'gang', 'victim', 'country', 'city',
            'region', 'dataset_pub', 'date', 'website',
            'work_sector', 'victim_notes', 'description'
        ]
        nova.api_data = None
        nova.rql_data = None
        nova.html_data = None
        nova._get_info()
    def _get_info(nova):
        nova.api_data = APIData()
        if not nova.st_id:
            nova.st_id = min(nova.api_data).id
        nova.last_id = max(nova.api_data).id
        nova.rql_data = RQLData(f'id>={nova.st_id} and id<={nova.last_id}')
        nova.html_data = RansomPostDetail()
        nova.html_data.get_range(nova.st_id, nova.last_id)
    def match_filds(nova):
        for id in range(nova.st_id, nova.last_id+1):
            api_o = nova.api_data[id]

```

```

rql_o = nova.rql_data[id]
html_o = nova.html_data[id]
if not (api_o.id or rql_o.id or html_o.id):
    continue
post = EventObj({'id':id}, False)
for fld_name in nova.fields:
    api_f = api_o[fld_name]
    rql_f = rql_o[fld_name]
    html_f = html_o[fld_name]
    if not (api_f or rql_f or html_f):
        continue
    elif api_f == rql_f == html_f:
        post[fld_name] = api_f
    else:
        match(fld_name):
            case 'country':
                if html_f:
                    if rql_f and (rql_f in html_f):
                        post[fld_name] = rql_f
                    elif api_f and (api_f in html_f):
                        post[fld_name] = api_f
                    else:
                        post[fld_name] = html_f
                elif rql_f:
                    post[fld_name] = rql_f
                elif api_f:
                    post[fld_name] = api_f
            case _:
                if html_f:
                    post[fld_name] = html_f
                elif rql_f:
                    post[fld_name] = rql_f

```

```

        elif api_f:
            post[fld_name] = api_f
            nova.victiums[id] = post
if __name__ == "__main__":
    from rich.console import Console
    from rich.table import Table
    mch = MyRansomFeed(29500)
    mch.match_filds()
    console = Console()
    table = Table(
        title="📊 Порівняння даних: API vs RQL vs HTML",
        show_lines=True,
        header_style="bold bright_white on blue"
    )
    table.add_column("ID", style="cyan", justify="center", vertical="top")
    table.add_column("Field", style="yellow")
    table.add_column("API (Standard)", overflow="fold")
    table.add_column("RQL Data", overflow="fold")
    table.add_column("HTML Data", overflow="fold")
    table.add_column("Match", overflow="fold")
    table.add_column("Status", justify="center")
    fields_to_compare = [
        'gang', 'victim', 'country', 'city',
        'region', 'dataset_pub', 'date', 'website', 'work_sector', 'victim_notes'
    ]
    csv_data = []
    for id_val in range(mch.st_id, mch.last_id):
        a_obj = mch.api_data[id_val]
        r_obj = mch.rql_data[id_val]
        h_obj = mch.html_data[id_val]
        m_obj = mch[id_val]
        first_row = True

```

```

for field in fields_to_compare:
    raw_a = getattr(a_obj, field, "-")
    raw_r = getattr(r_obj, field, "-")
    raw_h = getattr(h_obj, field, "-")
    raw_m = getattr(m_obj, field, "-")
    val_a = raw_a
    val_r = raw_r
    val_h = raw_h
    val_m = raw_m
    match_r = (val_r == val_a)
    match_h = (val_h == val_a)
    is_all_match = match_r and match_h
    style_r = "" if match_r else "[bold red]"
    style_h = "" if match_h else "[bold red]"
    style_m = "" if match_m else "[bold red]"
    status = ("[green]OK[/green]", 'OK') if is_all_match else ("[bold
red]DIFF[/bold red]", 'DIFF')
    display_id = f"[bold cyan]{id_val}[/bold cyan]" if first_row else ""
    table.add_row(
        display_id,
        f"[bold]{field}[/bold]" if not is_all_match else field,
        val_a,
        f"{style_r}{val_r}[/]" if style_r else val_r,
        f"{style_h}{val_h}[/]" if style_h else val_h,
        raw_m,
        status[0]
    )
first_row = False
csv_data.append({
    "ID": id_val,
    "Field": field,
    "API_Standard": val_a,

```

```

        "RQL_Data": val_r,
        "HTML_Data": val_h,
        "Status": status[1]
    })
    table.add_section()
    console.print(table)
    import csv
    csv_file = "discrepancies_report_689.csv"
    fieldnames = ["ID", "Field", "API_Standard", "RQL_Data", "HTML_Data", "Status"]
    with open(csv_file, mode="w", encoding="utf-8-sig", newline="") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(csv_data)
    console.print(f"\n[bold green]Дані збережено у файл:[/] {csv_file}")

```

Файл: src/ransomfeed/config_loader.py

```

from __future__ import annotations
import os
import yaml
import traceback
from pycti import get_config_variable
from pathlib import Path
RANSOMFEED_API_URL = 'https://api.ransomfeed.it'
class RansomFeedConfig:
    def __init__(nova):
        nova.load = nova._load_config()
    @staticmethod
    def _load_config():
        config_file_path = Path(__file__).parents[1].joinpath('config.yml')
        if not os.path.isfile(config_file_path):
            return {}
        with open(config_file_path) as f:

```

```

        conf = yaml.safe_load(f)
    return conf

def initialize_config(nova, helper: OpenCTIConnectorHelper) -> None:
    try:
        nova.duration_period = get_config_variable('CONNECTOR_DURATION_PERIOD', ['connector', 'duration_period'],
        nova.load, default='P1D')

        nova.verify_ssl = get_config_variable('VERIFY_SSL', ['connector', 'verify_ssl'], nova.load, default=True)

        nova.api_url = get_config_variable('RANSOMFEED_API_URL', ['RANSOMFEED', 'api_url'], nova.load, default=RANSOMFEED_API_URL or RANSOMFEED_API_URL

        nova.batch_size = get_config_variable('RANSOMFEED_BATCH_SIZE', ['RANSOMFEED', 'batch_size'], nova.load, isNumber=True, default=1000)

        nova.tlp_marking = get_config_variable('RANSOMFEED_TLP_MARKING', ['RANSOMFEED', 'tlp_marking'], nova.load, default='clear')

        nova.pap_marking = get_config_variable('RANSOMFEED_PAP_MARKING', ['RANSOMFEED', 'pap_marking'], nova.load, default='clear')
    except:
        helper.connector_logger.error(f'Init config error: {traceback.format_exc()}')
        raise

```