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

До захисту допущено
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**

(підпис)

«_____»_____2024 р.

# Дипломна робота

**на здобуття ступеня бакалавра**

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

на тему: Формування семантичних мереж на основі збору даних про кіберзагрози

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-01
(шифр групи)

Ченський Костянтин Юрійович
(прізвище, ім'я, по батькові) (підпис)

Керівник професор, завідувач кафедри ІБ д.т.н Ланде Д. В.
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент доцент кафедри ММЗІ к.ф.-м.н. Хмельницький М. О.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

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

Київ – 2024 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)

«___»_____2024 р.

# ЗАВДАННЯ
## на дипломну роботу здобувачу вищої освіти

Ченський Костянтин Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Формування семантичних мереж на основі збору даних про кіберзагрози,
керівник роботи д.т.н., професор, завідувач кафедри ІБ Ланде Дмитро Володимирович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 31 травня 2024 р. №2251-с

2. Термін подання здобувачем вищої освіти роботи 7 червня 2024 р.

3. Вихідні дані до роботи Технічні засоби автоматизації збору інформації про кіберінциденти

4. Зміст роботи Основні поняття та відомості про кіберзагрози, збір інформації та її візуалізацію; отримання та нормалізація даних про кіберзагрози з використанням OpenCTI та Python; система збору інформації про кіберзагрози та побудови семантичних мереж на її основі.

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

6. Дата видачі завдання 30 квітня 2024 року

Календарний план

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів дипломної роботи | Примітка |
|---|---|---|---|
| 1 | Формулювання тематики дипломної роботи | 20.09.2023 – 28.09.2023 | виконано |
| 2 | Узгодження напрямку роботи з дипломним керівником | 29.09.2023 | виконано |
| 3 | Ознайомлення з джерелами інформації по обраному напрямку | 01.11.2023-20.11.2023 | виконано |
| 4 | Ознайомлення з платформами автоматизації збору інформації | 05.01.2024 – 12.01.2024 | виконано |
| 5 | Аналіз технічних засобів побудови семантичних мереж | 07.03.2024 – 18.03.2024 | виконано |
| 6 | Ознайомлення з роботою фахівця операційного центру безпеки | 15.04.2024 – 26.04.2024 | виконано |
| 7 | Отримання завдання | 29.04.2024 – 30.04.2024 | виконано |
| 8 | Реалізація програмного засобу отримання та нормалізації даних з платформи автоматизації | 01.05.2024 – 17.05.2024 | виконано |
| 9 | Реалізація програмного засобу формування семантичних мереж | 18.05.2024 – 22.05.2024 | виконано |
| 10 | Об’єднання розроблених програм у цілісну систему | 23.05.2024 – 28.05.2024 | виконано |
| 11 | Оформлення дипломної роботи | 29.05.2024 – 06.06.2024 | виконано |
| 12 | Передзахист дипломної роботи | 07.06.2024 | виконано |

Здобувач вищої освіти _____ Костянтин ЧЕНСЬКИЙ
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ Дмитро ЛАНДЕ
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

# РЕФЕРАТ

Дипломна робота має обсяг 69 сторінок, містить 33 ілюстрації, 1 додаток, 25 джерел літератури.

Метою цієї дипломної роботи є розробка власної системи для побудови семантичних мереж на основі даних, отриманих за допомогою рішення автоматизації збору інформації про кіберзагрози.

Об’єктом дослідження є процес збору, нормалізації та візуалізації інформації про кіберзагрози.

Предметом дослідження є технічні засоби автоматизації отримання звітів щодо інцидентів кібербезпеки та побудови семантичних мереж з вхідних даних.

Методами дослідження є аналіз літературних джерел, розробка та тестування програмних скриптів для автоматизації отримання даних з OpenCTI, нормалізації зібраних даних за допомогою Pandas та створення семантичних мереж для візуалізації інформації про кіберзагрози з використанням NetworkX та PyVis.

В результаті роботи було отримано програмний засіб, який може бути використаний фахівцями з кібербезпеки для виконання багатьох задач, функціоналом якого є отримання даних з OpenCTI, обробка CSV файлу заданого формату, створення вебсторінок на основі зібраної інформації та побудова інтерактивних семантичних мереж за певним запитом.

Ключові слова: кіберзагрози, OpenCTI, Python, нормалізація даних, семантичні мережі.

## ABSTRACT

The work consists of 69 pages, 33 illustrations, 1 appendix, and 25 references.

The purpose of this work is to develop a system for building semantic networks based on the data obtained with the help of an automation solution for collecting information about cyber threats.

The object of research is the process of collecting, normalizing and visualizing information about cyber threats.

The subject of the research is the technical means of automation for obtaining reports on cybersecurity incidents and developing semantic networks from input data.

The research methods are literature analysis, development, and testing of software scripts to automate data retrieval from OpenCTI, normalization of collected data using Pandas, and creation of semantic networks for visualization of information about cyber threats using NetworkX and PyVis.

As a result of the work, a software tool has been developed that can be used by cybersecurity professionals to perform many tasks. Among these tasks are retrieving data from OpenCTI, processing a CSV file of a given format, creating web pages based on the collected information, and generating interactive semantic networks based on a specific request.

Keywords: cyber threats, OpenCTI, Python, data normalization, semantic networks.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШПЗ — Шкідливе Програмне Забезпечення

API — Application Programming Interface

APT — Advanced Persistent Threat

CSV — Comma Separated Values

CTI — Cyber Threat Intelligence

CVE — Common Vulnerabilities and Exposures

HTML — HyperText Markup Language

PoC — Proof of Concept

SOC — Security Operations Center

TI — Threat Intelligence

URL — Uniform Resource Locator

# ВСТУП

У другому півріччі 2023 року кількість зареєстрованих інцидентів кібербезпеки в Україні зросла на 36%, вказується в аналітичному звіті Державної служби спеціального зв’язку та захисту інформації [1]. Така статистика має спонукати фахівців державного та комерційного секторів на постійний збір інформації щодо актуальних та минулих загроз.

Проте, пропорційно інцидентам зростає і кількість звітів, які необхідно опрацьовувати. Наприклад, вчасно внесений у корпоративний антивірус індикатор компрометації може як попередити, так і зупинити атаку на інформаційні системи. Але якщо відповідний звіт буде не помічений або пропущений, збитки можуть бути фатальні. Через це питання покращення та автоматизації процесів у цьому напрямку з кожним днем стає все більш актуальним.

Відповідно, створення візуалізації зв’язків між угрупуваннями зловмисників та їх арсеналом за допомогою семантичних мереж надасть можливість наочно визначити прогалини у процесі збору інформації про минулі інциденти та виявити неявну взаємодію, потенційно попереджуючи майбутні атаки.

**Актуальність роботи** полягає у потребі автоматизації процесів збору інформації та можливості гнучкої візуалізації отриманих даних для своєчасного впровадження методів захисту та попередження майбутніх інцидентів кібербезпеки.

**Метою роботи** є розробка власної системи для побудови семантичних мереж на основі даних, отриманих за допомогою рішення автоматизації збору інформації про кіберзагрози.

**Завданням** є аналіз наявних засобів автоматизованого збору та збереження інформації щодо інцидентів у кіберпросторі; створення

програмного рішення для отримання та обробки бажаних даних; програмна реалізація побудови семантичних мереж за певним запитом;

**Об’єктом дослідження** є процес збору, нормалізації та візуалізації інформації про кіберзагрози.

**Предметом дослідження** є технічні засоби автоматизації отримання звітів щодо інцидентів кібербезпеки та побудови семантичних мереж з вхідних даних.

**Методами дослідження** є аналіз літературних джерел, розробка та тестування програмних скриптів для автоматизації отримання даних з OpenCTI, нормалізації зібраних даних за допомогою Pandas та створення семантичних мереж для візуалізації інформації про кіберзагрози з використанням NetworkX та PyVis.

**Наукова новизна одержаних результатів** полягає у вдосконаленні наявних рішень отримання візуалізації зв’язків між загрозами у кіберпросторі шляхом створення власної системи, яка, залежно від запиту, може надати більшу повноту.

**Практичним значенням одержаних результатів** може бути як використання у рамках подальших досліджень, шляхом виявлення неявних до цього зв’язків між кіберзагрозами, так і розгортання фахівцями у центрах забезпечення інформаційної безпеки для оцінки та наочності ефективності впровадженого процесу збору інформації.

# 1 ОСНОВНІ ПОНЯТТЯ ТА ВІДОМОСТІ ПРО КІБЕРЗАГРОЗИ, ЗБІР ІНФОРМАЦІЇ ТА ЇЇ ВІЗУАЛІЗАЦІЮ

## 1.1 Кіберзагрози

Кіберзагроза — будь-яка обставина або подія, що може негативно вплинути на діяльність (включаючи місію, функції, імідж або репутацію), активи організації або окремих осіб через інформаційну систему шляхом несанкціонованого доступу, знищення, розкриття, модифікації інформації та/або відмови в обслуговуванні. Також це потенційна можливість джерела загрози успішно використати певну вразливість інформаційної системи [2].

Поняття є доволі широким, його можна класифікувати за різними критеріями, зокрема за типом загроз, методом атаки, мотивацією суб’єктів атаки та іншими характеристиками. У межах цієї роботи кіберзагроза розглядається як арсенал зловмисників, який постійно збільшується і потребує моніторингу.

Загрози можуть набувати різного вигляду, проте найбільш поширеними типами, які створюють потребу регулярного оновлення систем захисту, можна визначити:

1. ШПЗ (шкідливе програмне забезпечення) — будь-яке нав'язливе програмне забезпечення, розроблене кіберзлочинцями для крадіжки даних, пошкодження або знищення інформаційних систем. Прикладами поширених шкідливих програм є віруси, хробаки, троянські віруси, шпигунські програми, рекламні програми та програми-вимагачі [3].
2. CVE (Common Vulnerabilities and Exposures) — це список публічно оприлюднених вразливостей. Коли хтось посилається на CVE, він має на увазі вразливість, якій присвоєно ідентифікаційний номер CVE [4].
3. Легітимні засоби, якими зловживають — ці інструменти призначалися для використання в дослідженнях безпеки та інших

законних цілях. Однак кіберзлочинці знайшли спосіб використовувати їх для проведення своїх кампаній [5].

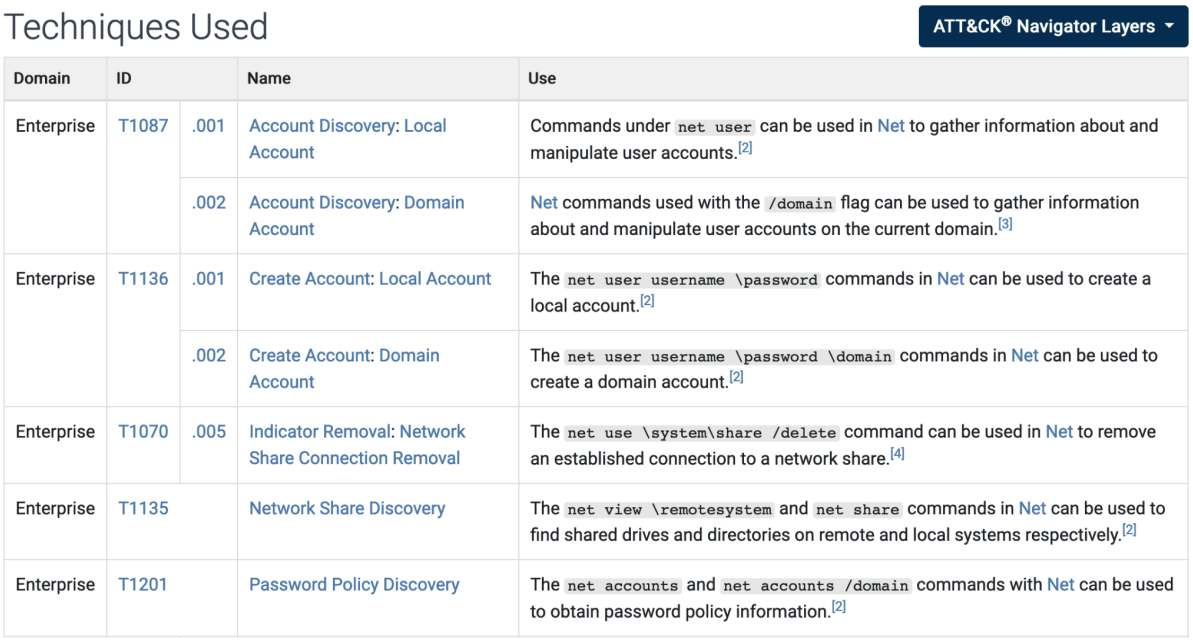
Потенційні збитки від ШПЗ та необхідність впровадження рішень корпоративного антивірусного захисту практично завжди враховуються організаціями у рамках політики інформаційної безпеки, проте, інші два зазначені типи загроз не завжди чітко висвітлюються як ті, що потребують впровадження певних процесів виявлення та реагування.

Вразливості, яким присвоєно ідентифікатор CVE з’являються щоденно, і не рідкість, коли визначений рівень критичності високий або найвищий. Доволі часто дослідники знаходять їх і у технічних засобах захисту. Зазвичай, рекомендованим усуненням ризику, обумовленого вразливістю, є оновлення до останньої версії. Проте, зустрічаються і варіанти пом’якшення ризику шляхом зміни конфігурації тощо, для організацій які не можуть в короткий термін встановити виправлення. В умовах, коли адміністратор інформаційної системи не має інструменту центрального оновлення усіх кінцевих точок, або внесення у них змін, система може залишатись вразливою впродовж тижнів, якщо не місяців.

Ситуація стає лише гіршою, якщо адміністратор не дізнається про наявність цієї вразливості, що доволі легко уявити, якщо він або фахівці відділу кібербезпеки не слідкують за новинами кіберпростору. Ризик атаки за допомогою такої загрози для організації стає найбільшим у випадках, коли для вразливості опубліковано публічно доступний PoC (proof of concept) — експлойт, який не має на меті завдати шкоди, а лише показати слабкі місця у безпеці програмного забезпечення [6]. Для досвідчених зловмисників змінити код такої програми для подальшого проведення атаки є тривіальною задачею.

Подібна ситуація й з легітимним програмним забезпеченням, яке зловмисники можуть перетворити на загрозу інформаційним системам. Прикладом є утиліта операційної системи Windows — net.exe. Вона створена для контролю користувачів, груп, служб та мережевих з'єднань [7]. Проте,

використовуючи фреймворк MITRE ATT&CK можна побачити що зловмисники використовують її для ряду технік, наприклад T1087 — виявлення облікових записів [8].



Techniques Used

ATT&CK® Navigator Layers

| Domain | ID | | Name | Use |
|---|---|---|---|---|
| Enterprise | T1087 | .001 | Account Discovery: Local Account | Commands under `net user` can be used in Net to gather information about and manipulate user accounts.[2] |
| | | .002 | Account Discovery: Domain Account | Net commands used with the `/domain` flag can be used to gather information about and manipulate user accounts on the current domain.[3] |
| Enterprise | T1136 | .001 | Create Account: Local Account | The `net user username \password` commands in Net can be used to create a local account.[2] |
| | | .002 | Create Account: Domain Account | The `net user username \password \domain` commands in Net can be used to create a domain account.[2] |
| Enterprise | T1070 | .005 | Indicator Removal: Network Share Connection Removal | The `net use \system\share /delete` command can be used in Net to remove an established connection to a network share.[4] |
| Enterprise | T1135 | | Network Share Discovery | The `net view \remotesystem` and `net share` commands in Net can be used to find shared drives and directories on remote and local systems respectively.[2] |
| Enterprise | T1201 | | Password Policy Discovery | The `net accounts` and `net accounts /domain` commands with Net can be used to obtain password policy information.[2] |

Рисунок 1.1 — Деякі з технік які використовують net.exe

Оскільки такі програми є легітимними і їх використання часто є нормальною поведінкою для середовища, впроваджені технічні засоби можуть не виявляти зловживання. Для запобігання загрозам такого типу спеціалісти з кібербезпеки та адміністратори систем мають знати список тих програм, які вже були помічені в інцидентах та слідкувати за потенційною появою нових. Озброєні такою інформацією вони зможуть протестувати вплив тих чи інших інструментів на систему та за потреби створити сповіщення або обмеження даної активності.

## 1.2 Джерела кіберзагроз

Кіберзагрози є арсеналом певного джерела їх походження, яке в загальному можна назвати угрупування кіберзлочинів, зловмисниками, але також існує класифікація, за допомогою якої можна розділити ці джерела на типи з притаманними особливостями та мотивами.

IBM визначає наступні типи [9]:

1. Кіберзлочинці — мотивом таких осіб здебільшого є фінансова вигода. До поширених злочинів, які скоюють кіберзлочинці, належать атаки з вимогами викупу та фішингові афери, які обманом змушують людей здійснювати грошові перекази або розголошувати інформацію про кредитні картки, облікові дані для входу в систему, інтелектуальну власність чи іншу приватну або конфіденційну інформацію.
2. Державні суб’єкти — уряди часто фінансують суб'єктів загроз з метою викрадення конфіденційних даних, збору конфіденційної інформації або підриву критично важливої інфраструктури іншого уряду. Ці зловмисні дії часто включають шпигунство або кібервійну і, як правило, добре фінансуються, що робить загрози складними й важкими для виявлення.
3. Хактивісти — ці суб'єкти загрози використовують техніки атак для просування політичних або соціальних ініціатив, таких як поширення свободи слова або викриття порушень прав людини. Хактивісти вірять, що впливають на позитивні соціальні зміни, і вважають виправданим напад на окремих осіб, організації або державні установи з метою розкриття секретів або іншої конфіденційної інформації.
4. Шукачі вражень — вони атакують комп'ютерні та інформаційні системи насамперед заради розваги. До цього типу відносять так званих “script kiddies” [10], осіб які не мають просунутих технічних навичок, але використовують вже наявні інструменти та методи для атак на вразливі

системи, в першу чергу заради розваги або особистого задоволення. Проте, вони все ж можуть завдати ненавмисної шкоди, втручаючись у кібербезпеку мережі, створюючи потенціал для майбутніх кібератак.

5. Внутрішні загрози — на відміну від більшості інших типів суб'єктів, суб'єкти внутрішніх загроз не завжди мають зловмисні наміри. Деякі з них завдають шкоди своїм компаніям через людські помилки. Але зловмисні інсайдери існують. Наприклад, незадоволений працівник, який зловживає правами доступу для крадіжки даних з метою отримання фінансової винагороди або завдає шкоди даним чи додаткам у помсту за те, що його звільнили.

Незважаючи на мотив, будь-які з перелічених джерел кіберзагроз можуть завдати збитків комерційним та державним організаціям. Заведено вважати, що державні суб’єкти є найскладнішими для протистояння через регулярне фінансування, кількість людського ресурсу та підтримку зі сторони уряду. Саме вони найчастіше асоціюються з таким поняттям як APT, хоча представниками можуть виступати й інші суб’єкти загроз.

APT (Advanced Persistent Threat) — складна, тривала кібератака, під час якої зловмисник встановлює непомітну присутність у мережі з метою викрадення конфіденційних даних протягом тривалого періоду часу. APT-атака ретельно планується і розробляється з метою проникнення в конкретну організацію, обходу присутніх заходів безпеки та непомітності [11].

Оскільки нерідко такі атаки є повторюваними, з певним показником впевненості досвідчені дослідники кібербезпеки можуть провести атрибуцію кібератаки до певного суб'єкта, що може включати поведінковий аналіз, аналіз використаного програмного забезпечення, аналіз мережевих індикаторів походження атаки тощо. Для зручності зазвичай використовують ім’я, яке вперше було присвоєно певному суб’єкту, хоча й бувають виключення коли певний центр досліджень використовує власну назву.

Прикладом державного суб'єкта, який має власне ім’я, яке використовують для атрибуції є Cozy Bear (APT29) — суб’єкт російського походження, яка, за оцінками, може діяти від імені Служби зовнішньої розвідки Російської Федерації. Було виявлено, що він використовує широкомасштабні фішингові кампанії для розповсюдження широкого спектра шкідливого програмного забезпечення в рамках атак на політичні, наукові установи та органи національної безпеки в різних галузях [11]. Даний суб’єкт регулярно націлює свої кампанії на українські підприємства, проте не обмежується цим.

Наприклад, корпорація Microsoft у січні 2024 року опублікувала статтю, де описала атаку APT29 на власні системи, зазначивши також інші асоційовані імена - Midnight Blizzard, UNC2452, Cozy Bear, NOBELIUM [12].

### 1.3 Збір інформації

Збір інформації про кіберзагрози, також відомий як CTI (Cyber Threat Intelligence), або TI (Threat Intelligence) — динамічна, адаптивна технологія, яка використовує великі дані історії загроз для проактивного блокування та усунення майбутніх зловмисних атак на мережу. Сама по собі розвідка кіберзагроз не є рішенням, але це важливий компонент архітектури безпеки. Зважаючи на еволюцію загроз, рішення для захисту є настільки ефективними, наскільки ефективною є аналітика, що їх забезпечує [13].

Слідкувати за новинами у сфері кібербезпеки, проводити їх аналіз з метою оцінки ризику для організації та вносити зміни у технічні рішення безпеки є найпростішим прикладом процесу збору інформації.

Хоч даний процес і є простим, ефективне його впровадження може значно покращити стан захищеності інформаційних систем.

Якщо фахівці департаменту кібербезпеки, або адміністратори системи візьмуть у звичку щоденний перегляд останніх новин, кількість не виявлених та не оцінених ризиків, наявних в інформаційних системах організації,

зменшиться. І навпаки, якщо не існує відповідального за аналіз новин, то відсутнє вчасне реагування на нові загрози може призвести як до фінансових, так і до репутаційних збитків.

Microsoft виділяє 4 категорії розвідки про кіберзагрози [14]:

1. Стратегічна — це аналіз високого рівня для зацікавлених осіб не технічного профілю, які мають відношення до бізнесу в цілому, таких як керівники вищої ланки. Доносити таку інформацію потрібно в широкому контексті, зважаючи на довгострокову перспективу.
2. Тактична — це інформація, необхідна фахівцям з кібербезпеки для вжиття негайних заходів з метою пом'якшення або уникнення ризиків. Вона включає технічну інформацію про найактуальніші тенденції.
3. Операційна — це знання про конкретні загрози та кампанії. Вона надає спеціалізовану інформацію для команд реагування на інциденти про особу зловмисника, його мотиви та методи.
4. Технічна — стосується ознак того, що відбувається атака, наприклад, індикатори компрометації.

Для покращення тактичної, операційної та технічної розвідки, необхідно збільшувати кількість вхідних даних, для наведеного раніше прикладу — збільшити кількість інформаційних джерел, з яких отримуються новини. Звісно, від цього збільшиться й кількість статей, які необхідно аналізувати.

Проте, навіть у великих компаніях не завжди буде відділ, чи хоча б один спеціаліст, який відповідає лише за розвідку стосовно кіберзагроз. Скоріш даний процес буде включено в обов’язки фахівців з кібербезпеки. Даний підхід не є оптимальним, оскільки виникають наступні проблеми:

1. Повторний аналіз — якщо джерела попередньо не розподілені, кожен фахівець буде окремо розглядати кожну новину, незважаючи на те чи була вона проаналізована до цього.

2. Недостатньо часу — коли збір інформації про кіберзагрози виноситься як один з обов’язків, велика ймовірність що проаналізовані будуть не всі важливі новини, звіти тощо.

3. Необхідність документації — навіть якщо джерела будуть розподілені, зважаючи на нестачу часу, або людський фактор, важлива інформація може бути пропущена. Для контролю цього питання можна вести документацію, наприклад у формі таблиці, щоб оброблені до цього дані були чітко зображені. Але й це буду додатково займати й так недостатній час.

Зважаючи на корисність збору інформації щодо новин у сфері кібербезпеки та перелічені проблеми ручної реалізації, виникає потреба автоматизації цього процесу.

### 1.4 Автоматизація збору інформації

В сучасних умовах швидкого розвитку загроз у сфері кіберпростору та постійного збільшення кількості звітів, організації стикаються з величезними обсягами даних, які потребують постійного моніторингу та аналізу. Ручний збір та обробка таких даних є надзвичайно трудомістким і неефективним процесом, що може призвести до упущень важливої інформації про загрози. Це особливо критично в умовах, коли додавання індикаторів компрометації або встановлення оновлень є ключовим фактором для успішного захисту від атаки.

Технічних рішень, які дозволяють покращити тактичну, операційну та технічну розвідку шляхом автоматизації обробки нової інформації наразі є достатньо.

Комерційні рішення пропонують власні хмарні платформи, які надають організаціям доступ до великих баз даних, з можливостями налаштування під певні сектори та інші особливості. Основним мінусом комерційних платформ для багатьох організацій може стати вартість, але він не єдиний. Деякі з них не

мають можливості внесення власних звітів стосовно інцидентів, інші автоматизують внесення індикаторів компрометації лише у технічні рішення того ж вендора.

Проте, існують і платформи з відкритим вихідним кодом, MISP та OpenCTI можна виділити як одні з найбільш поширених і доступних, зважаючи на кількість інструкцій та статей стосовно безплатно доступних платформ.

MISP — це програмне рішення з відкритим вихідним кодом для збору, зберігання, розповсюдження та обміну індикаторами кібербезпеки та загрозами, аналізу інцидентів кібербезпеки [15].

OpenCTI — це платформа з відкритим вихідним кодом, що дозволяє організаціям керувати своїми знаннями та спостереженнями про кіберзагрози. Вона була створена з метою структурування, зберігання, організації та візуалізації технічної та не технічної інформації про кіберзагрози [16].

MISP існує довше ніж OpenCTI, що впливає на кількість впроваджених спільнотою покращень. Проте, новіша платформа не менш популярна, і також постійно отримує поліпшення. На GitHub, де міститься вихідний код для обох платформ, популярність можна оцінити за кількістю зірочок, на момент написання роботи для MISP — 5 тисяч, а для OpenCTI — 4.8 тисячі.

Також варто зазначити зручність інтерфейсу користувача, та його зовнішньої привабливості, оскільки це може бути важливо для певних організацій. У MISP за замовчуванням відсутня будь-яка панель відображення наявної інформації, вона має бути налаштована власноруч користувачем. OpenCTI же, навпаки, має одразу налаштовану інформативну панель, яка надалі може бути змінена.

Інтерфейс налаштованої панелі MISP має наступний вигляд:

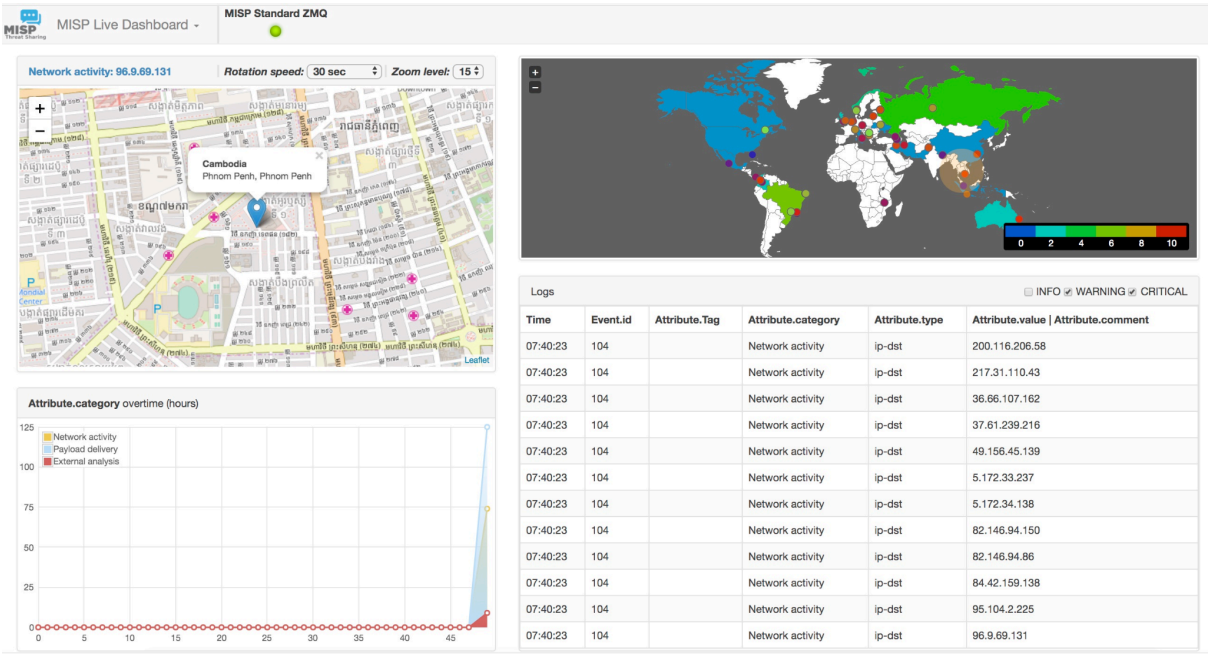



Рисунок 1.2 — Налаштована панель у MISP

Панель за замовчуванням в OpenCTI:


Рисунок 1.3 — Панель за замовчуванням в OpenCTI

Для отримання інформації стосовно інцидентів кібербезпеки обидві платформи використовують зовнішні джерела, які потребують API ключів, деякі з яких потребують платної підписки. Незважаючи на це, навіть доступні безплатно джерела можуть надати нову для організації інформацію, яка допоможе запобігти використанню кіберзагроз суб'єктами атак.

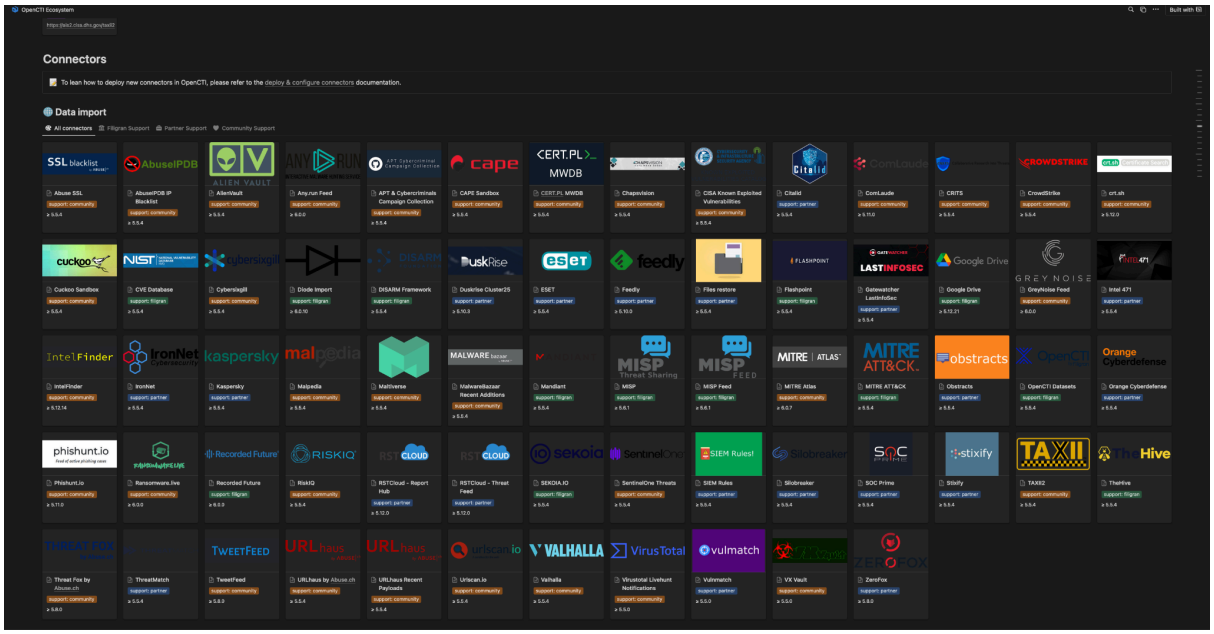



Рисунок 1.4 — Джерела інформації для OpenCTI

Оскільки OpenCTI новіша платформа, у ній передбачена можливість інтеграції з MISP, що пом'якшує потенційну складність бажаного переходу.

З опису цих платформ можна зрозуміти, що вони виконують однакові функції, але вибір однієї з них здебільшого залежить від конкретних потреб та уподобань організації. MISP, завдяки довгому існуванню, є більш розвинутою платформою. Вона забезпечує широкий набір інструментів для отримання та обробки даних про інциденти.

OpenCTI, своєю чергою, постійно вдосконалюється й інтегрує новітні технології, що робить її привабливою для організацій, які прагнуть використовувати сучасні рішення. Вона має більш зрозумілий та приємний інтерфейс, що спрощує її використання для користувачів.

## 1.5 Візуалізація зібраної інформації

Візуалізація зібраної інформації про кіберзагрози є ключовим інструментом для фахівців з кібербезпеки, що допомагає ефективніше аналізувати великі обсяги даних і прогалини у них. Графічне представлення даних дозволяє швидко виявляти неявні взаємозв'язки, які можуть бути важливими індикаторами загроз. Це особливо важливо в умовах обсягу даних, що постійно зростає та різноманіття загроз, з якими стикаються організації.

Одним з найбільш ефективних методів візуалізації зібраної інформації про кіберзагрози є використання семантичних мереж.

Семантична мережа — це форма представлення знань, яка використовує структуру графа для представлення понять (вузлів) і семантичних зв'язків між ними (ребер). Ця мережа призначена для відображення асоціативної природи людських знань, дозволяючи представляти складні взаємозв'язки між поняттями у візуальний та структурований спосіб [17].

Прикладом простої семантичної мережі у сфері кібербезпеки може бути:

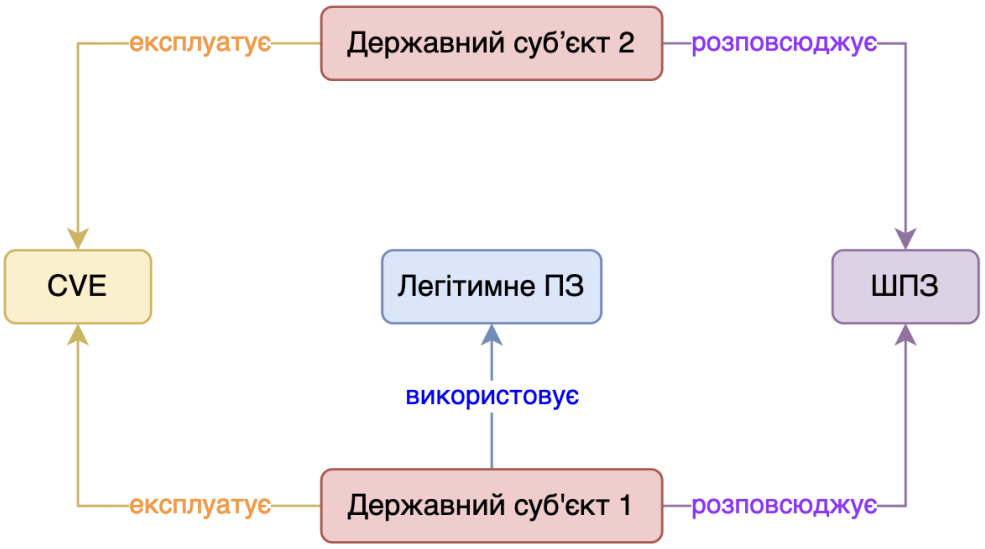



Рисунок 1.5 — Приклад семантичної мережі у сфері кібербезпеки

Зважаючи на два із трьох однакових зв’язків для Державного суб'єкта 1 та Державного суб'єкта 2, якщо перший часто атакує сектор, в якому знаходиться організація, для якої впроваджуються методи захисту, використовуючи семантичну мережу, аналітик міг би зробити висновок щодо необхідності уникнення або пом’якшення ризиків, пов’язаних і з другим.

Семантичні мережі надають наступні можливості:

1. Візуалізація неявних взаємозв'язків — дозволяють відображати не очевидні зв'язки між різними елементами системи, що сприяє кращому розумінню структури даних та взаємозв'язків між ними. Це допомагає ідентифікувати потенційні загрози, які могли бути визначені як ті, що не представляють ризику.
2. Аналіз великих обсягів даних — дозволяють ефективно аналізувати великі обсяги даних, що є критично важливим у сучасних умовах збільшення кількості кіберзагроз. Це забезпечує більш точне виявлення загроз та передбачення майбутніх атак.
3. Покращення комунікації — полегшують комунікацію між технічними та не технічними співробітниками організації, оскільки наочне представлення даних є більш зрозумілим для всіх учасників процесу. Це сприяє поліпшенню стратегічної розвідки та ефективного впровадження технічних засобів кібербезпеки.
4. Масштабованість та гнучкість — дозволяють легко масштабувати аналіз даних для великих організацій, шляхом зображення мереж, які стосуються лише важливого наразі інциденту. Це забезпечує гнучкість внесення змін в інформаційні системи під час впровадження методів захисту стосовно конкретної кіберзагрози або її суб’єкта.

У операційних центрах безпеки, також відомих як SOC (Security Operations Center) зазвичай впроваджено відеостіни, на яких зображено інформацію, яка потребує постійного спостереження. Семантична мережа, яка побудована, наприклад, на основі даних зібраних про суб’єкти атак для сектору,

який обслуговує SOC, може бути додатково зображена на одному з екранів. Звичайно, для корисності вона має регулярно оновлюватись шляхом, наприклад, планувальника задач.

**Висновки до розділу 1**

У першому розділі було детально розглянуто основні поняття та відомості про кіберзагрози, їх типи та джерела походження. Було описано такі форми кіберзагроз як шкідливе програмне забезпечення, вразливості (CVE) та легітимні засоби, які зловмисники можуть використовувати для своїх атак. Для різних суб'єктів кіберзагроз визначені їх унікальні особливості та мотиви.

Крім того, досліджено важливість збору інформації про кіберзагрози (CTI), зокрема різні категорії розвідки, які допомагають у виявленні подій кібербезпеки та реагуванні на них. Зазначено, що для ефективного захисту необхідно постійно збільшувати кількість інформаційних джерел та впроваджувати автоматизовані системи для обробки нової інформації, такі як MISP та OpenCTI.

Також висвітлено значення візуалізації зібраної інформації за допомогою семантичних мереж, які можуть бути використані для виявлення неявних зв'язків між загрозами, гнучкого аналізу та покращення комунікації у рамках стратегічної розвідки.

# 2 ОТРИМАННЯ ТА НОРМАЛІЗАЦІЯ ДАНИХ ПРО КІБЕРЗАГРОЗИ З ВИКОРИСТАННЯМ OPENCTI ТА PYTHON

## 2.1 Вибір платформи автоматизації збору

Між розглянутими у першому розділі даної роботи рішеннями автоматизації з відкритим вихідним кодом, було обрано OpenCTI. Ключовими факторами для вибору були:

1. Новизна — рішення є більш сучасним відносно MISP.
2. Зручність користувацького інтерфейсу.

Під час проходження переддипломної практики, було отримано доступ до обох рішень, проте, зважаючи на наповненість даними, яка обумовлена уподобаннями компанії, пов’язаними з вищенаведеними факторами, OpenCTI став безумовним фаворитом.

Платформа зберігає інформацію про джерела загроз у секції “Threats”, яка містить наступні класи [16]:

- Threat actors (Group) — суб’єкт загрози представляє фізичну групу зловмисників, які працюють з набором засобів вторгнення, використовують шкідливе програмне забезпечення, інфраструктуру атаки тощо.
- Threat actors (Indvidual) — суб’єкт загрози представляє одного реального зловмисника, якого можна описати фізичними та особистими атрибутами та мотивами. Він оперує набором вторгнень, використовує шкідливе програмне забезпечення та інфраструктуру тощо.
- Intrusion sets — набори вторгнень. Це послідовні дані про технічні й нетехнічні елементи, що відповідають певному суб’єкту загрози, його мотивам та технікам. Це особливо корисно для зв'язування декількох атак з певним суб’єктом, навіть без достатньої інформації про те, хто їх здійснив.

- Campaigns — кампанії. Являють собою серію атак, що відбуваються протягом певного періоду часу та/або спрямовані на певний сектор підприємств чи групу населення.

Згідно з вищевикладеним описом з офіційної документації, набори вторгнень є найповнішою інформацією про зв’язки між джерелами та загрозами, тому з метою отримання глобальної картини доцільно використовувати саме їх.

При відкритті у вебінтерфейсі, вони будуть мати подібний вигляд:

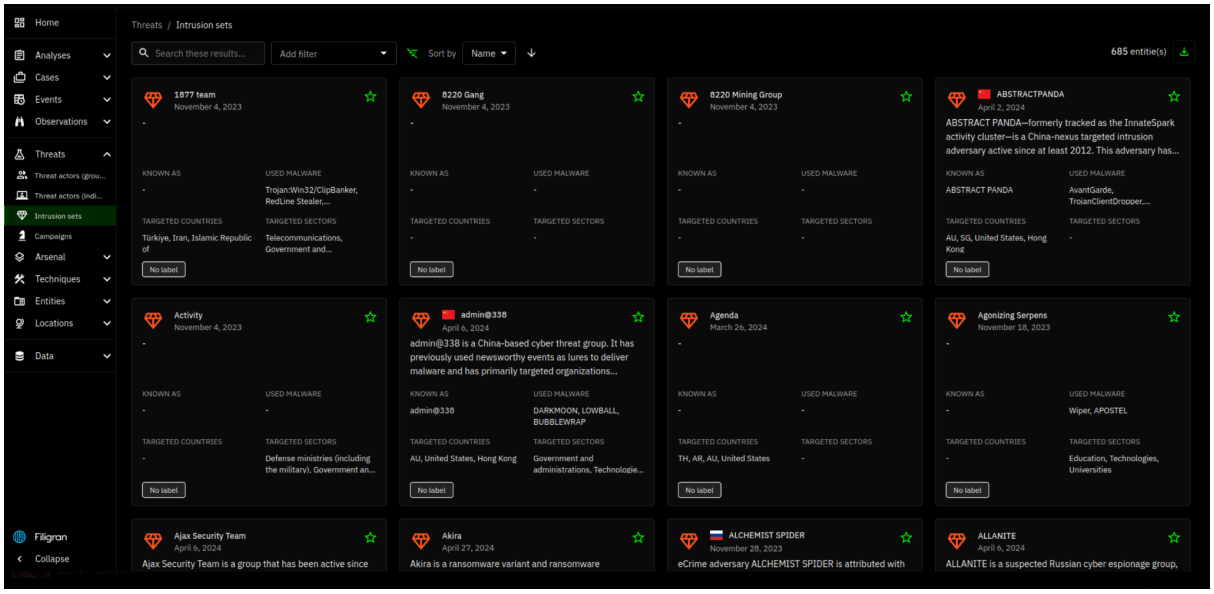



Рисунок 2.1 — Вигляд “Intrusion sets” у вебінтерфейсі OpenCTI

Іншою важливою функцією платформи є можливість отримання відношень між поняттями, таким чином уникаючи ситуацій, коли певна загроза не була додана до знань про набір вторгнень нативними засобами платформи. Такі ситуації можуть виникати через розбіжності у форматі вхідних даних, отриманих від зовнішніх джерел.

Деякі відношення будуть доступні користувачу одразу на сторінці певного набору, інші можна дістати пошуком за широкими фільтрами платформи, використовуючи секцію Data/Relationships.

Найпростішим прикладом можна вказати походження певного суб'єкта загрози, яке можна побачити на його сторінці доступній у платформі:

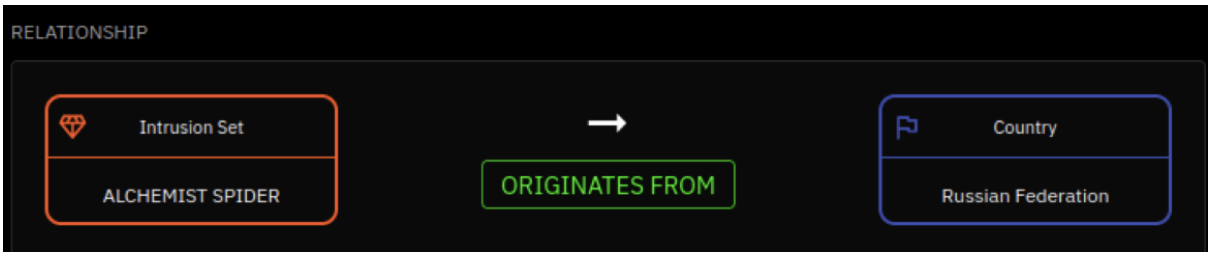



Рисунок 2.2 — Відношення типу “походить з” між набором вторгнення та країною

## 2.2 Потреба програмної реалізації

Навіть з впровадженою платформою автоматизації збору даних та додавання індикаторів у технічні рішення захисту, без достатнього аналізу інформації платформа може не покращити процес розвідки, а погіршити.

Проте, і ручна робота, яку необхідно виконати, щоб розпочати аналіз, є неефективною. При уявній ситуації, коли користувач платформи хоче дослідити необхідність впровадження додаткових методів захисту стосовно такого типу загроз як шкідливе програмне забезпечення, йому доведеться або будувати відношення між усіма наборами вторгнення і ШПЗ, вивантажуючи не згрупований CSV файл, або виконувати кількість запитів, яка дорівнює кількості цих самих наборів у системі, зберігаючи для кожного окремий файл. Звичайно, ситуація погіршиться, коли мова йде не про один тип загроз.

Через це й виникає потреба реалізувати засіб, який буде збирати й групувати всю бажану інформацію в один файл, який потім можна використати як для проведення детального дослідження, візуалізації зібраних даних, так і звітності в рамках стратегічної розвідки.

### 2.3 Вибір бажаних даних

Для цієї роботи, в рамках доказу концепції та подальшого використання для побудови семантичних мереж, мною були обрані наступні дані:

1. Назва — асоційоване з джерелом загроз ім’я.
2. Походження — країна або географічний регіон, який асоціюють з загрозою. Особливо актуально для державних суб’єктів.
3. Опис — для деяких з найбільш розвинутих та активних джерел загроз зовнішні інтеграції платформи додають власні дані.
4. Альтернативні назви — пов’язано з розбіжностями різних дослідницьких програм, згадувалось у пункті 1.2 цієї роботи.
5. Вразливості — недоліки інформаційних систем, яким присвоєно ідентифікатор CVE, що експлуатуються суб’єктами.
6. ШПЗ — програми-вимагачі, та інші, які розповсюджуються суб’єктами.
7. Легітимні засоби — ті, якими зловживає суб’єкт.
8. Звіти — посилання на зовнішні інформаційні джерела, які можуть містити індикатори компрометації, опис конкретних кампаній проведених суб’єктом тощо.

Вибір інформації, необхідної для операційної діяльності організації може відрізнятись. Загалом, перелічені вище типи загроз та інформація про їх джерела допоможе запобігти найпоширенішим векторам атак.

### 2.4 Вибір технічних засобів

Згідно з дослідженням компанії DOU, мова програмування Python посідає 3 місце серед комерційних спеціалістів України [18]. Велика кількість модулів, легкість написання коду та інтерпретованість є безумовними її перевагами. Також, при написанні програм, які потенційно можуть використовуватись на

різних операційних системах, перевагою є і широка підтримка та незалежність більшості модулів.

З використанням пакета Pandas — швидкого, потужного, гнучкого і простого у використанні інструменту аналізу та маніпулювання даними з відкритим вихідним кодом [19], мова програмування Python стає чудовим вибором для аналізу даних.

А пакет PyCTI — клієнт платформи OpenCTI для Python [20], надасть змогу отримувати інформацію з платформи за допомогою API.

Таким чином, поєднання вищезазначеної мови програмування та пакетів надасть можливість автоматизувати процес отримання та обробки даних з OpenCTI.

### 2.5 Отримання даних за допомогою OpenCTI API

Файл, який містить код програмної реалізації та інтерпретується для виконання має назву cti.py. Повний його зміст буде викладено у Додатку А.

Алгоритм програми наступний:

1. Ініціалізація підключення — користувач вводить URL OpenCTI та API-ключ, що використовуються для підключення до платформи OpenCTI.
2. Отримання списку набору вторгнень — програма використовує OpenCTI API для отримання списку всіх джерел загроз та деталей про них, наявних у платформі.
3. Отримання назви файлу — користувач вводить бажану назву для файлу, куди буде збережено отримані дані.
4. Отримання зв’язків — для кожного набору програма отримує список зв’язків, включаючи походження, вразливості, шкідливе програмне забезпечення, інструменти та звіти.

5. Збереження у файл — програма записує усі отримані дані у CSV файл, назву якого ввів користувач.

6. Завершення — за виключенням помилок під час попередніх кроків, програма повідомляє про успішний запис даних у вказаний користувачем файл.

Графічно алгоритм можна зобразити наступним чином:

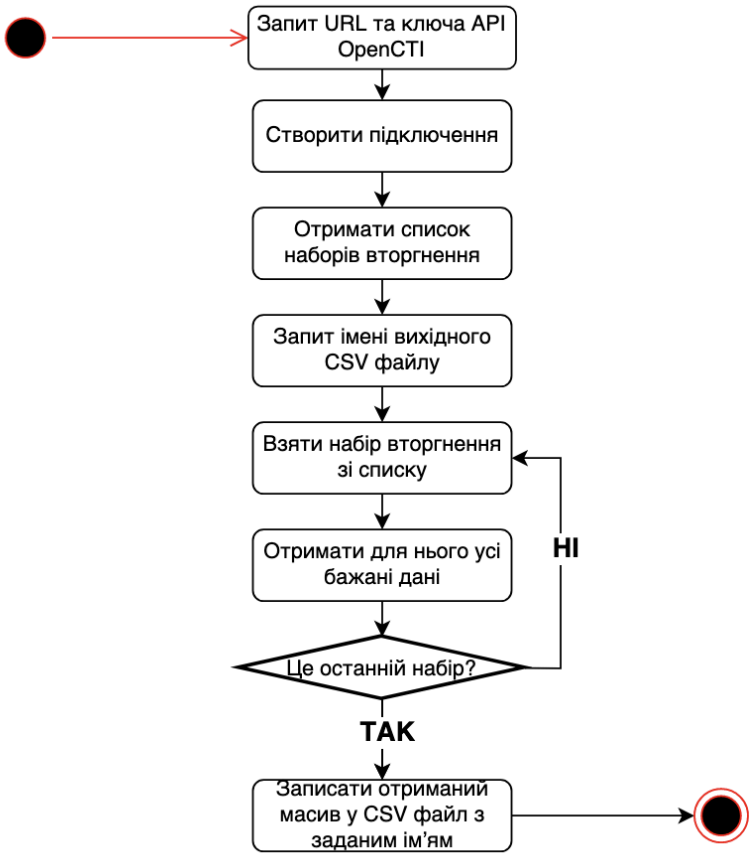



Рисунок 2.3 — Блок схема алгоритму отримання даних

Ініціалізація підключення виконується за допомогою функції openCTI, яка приймає на вхід параметри opencti_url та api_key, після чого за допомогою класу PyCTI OpenCTIApiClient створює підключення.

```
7   def openCTI(opencti_url, api_key):
8       global opencti_api_client
9       opencti_api_client = pycti.OpenCTIApiClient(opencti_url, api_key)
```

Рисунок 2.4 — Функція ініціалізації підключення

Далі для отримання списку наборів вторгнення виконується функція getThreats, яка за допомогою методу opencti_api_client.intrusion_set.list повертає масив threats з назвами, описами та альтернативними назвами.

```
108   def getThreats():
109       global threats
110       threats = opencti_api_client.intrusion_set.list()
```

Рисунок 2.5 — Функція отримання списку наборів вторгнення

Далі, програма запитує у користувача назву для бажаного CSV файлу, записуючи її у змінну output_file. Після цього викликається функція threatsCSV, яка приймає на вхід створену раніше змінну. Ця функція є основною у програмі, оскільки саме вона звертається до усіх інших з метою отримання даних про кожний з наборів вторгнення.

Першочергово у функції створюється порожній масив. Далі, циклічним перебором кожного набору записаного попередньо о у масив threats, для майбутніх колонок записуються дані, а саме:

1. Назва — береться з масиву threats.
2. Походження — викликає функцію threatOrigin та передає їй назву.
3. Опис — береться з масиву threats.
4. Альтернативні назви — береться з масиву threats.
5. Вразливості — викликає функцію threatVulns та передає їй назву.
6. ШПЗ — викликає функцію threatMalw та передає їй назву.

7. Легітимні засоби — викликає функцію threatTools та передає їй назву.

8. Звіти — викликає функцію threatReports та передає їй назву.

Словник з отриманими значеннями додається до попередньо створеного порожнього масиву. Далі, використовуючи пакет Pandas, він поміщається у клас DataFrame, структурою якого є двовимірні, змінні за розміром, потенційно неоднорідні табличні дані[19]. Завершенням функції є запис отриманого DataFrame у CSV файл з назвою заданою користувачем, використовуючи метод to_csv.

```
112  def threatsCSV(filename):
113      threat_data = []
114
115      for threat in threats:
116          threat_data.append({
117              'Name': threat["name"],
118              'Country': threatOrigin(threat["name"]),
119              'Description': threat["description"],
120              'Aliases': threat["aliases"],
121              'Vulnerabilities': threatVulns(threat["name"]),
122              'Malware': threatMalw(threat["name"]),
123              'Tools': threatTools(threat["name"]),
124              'Reports': threatReports(threat["name"])
125          })
126
127      df = pd.DataFrame(threat_data)
128      df.to_csv(filename, index=False)
```

Рисунок 2.6 — Основна функція програми cti.py

Зазначені вище функції threatVulns, threatMalw та threatTools працюють за одним принципом, змінюється лише тип поняття для якого за допомогою API шукається відношення.

Для опису вибрана функція threatMalw. Вона приймає на вхід назву набору вторгнення й створює по ньому фільтр. Далі за допомогою методу opencti_api_client.stix_core_relationship.list, який приймає ідентифікатор набору вторгнення та тип для якого шукаються відношення, отримується список усіх відношень. У цьому випадку цей тип - Malware. Далі створюється пустий масив, у який методом циклічного перебору по одному додаються назви ШПЗ, знайдені у відношеннях. Цей масив функція і повертає.

```python
def threatMalw(threatName):
    intrusion_set = opencti_api_client.intrusion_set.read(
        filters={
            "mode": "and",
            "filters": [{"key": "name", "values": [threatName]}],
            "filterGroups": [],
        }
    )

    stix_relations = opencti_api_client.stix_core_relationship.list(
        fromId=intrusion_set["id"], toTypes=["Malware"]
    )

    malw = []
    for stix_relation in stix_relations:
        malw.append(str(stix_relation["to"]["name"]))
    return malw
```

Рисунок 2.7 — Функція для отримання зв’язків з ШПЗ

Функція threatReports має невеликі відмінності від розглянутої раніше. На вхід вона також приймає назву набору вторгнення, після чого створює з нього фільтр для пошуку. Потім, за допомогою методу opencti_api_client.report.list, який приймає фільтр за ідентифікатором загрози та параметри сортування звітів (для цієї роботи було обрано такі, що спочатку записуються найновіші), отримуються усі звіти для даного набору. Далі, знову створюється пустий масив, але цього разу, для запису у нього зовнішніх посилань, використовується спочатку циклічний перебір по одному зі самих звітів, а далі аналогічний, але вже по посиланнях, які в ньому містяться. Цей масив функція і повертає.

```python
def threatReports(threatName):
    intrusion_set = opencti_api_client.intrusion_set.read(
        filters={
            "mode": "and",
            "filters": [{"key": "name", "values": [threatName]}],
            "filterGroups": [],
        }
    )
    reports = opencti_api_client.report.list(
        filters={
            "mode": "and",
            "filters": [{"key": "objects", "values": [intrusion_set["id"]]}],
            "filterGroups": [],
        },
        orderBy="published",
        orderMode="asc",
    )

    urls = []
    for report in reports:
        ext = report["externalReferences"]
        for ref in ext:
            urls.append(ref['url'])
    return urls
```

Рисунок 2.8 — Функція для отримання зовнішніх посилань

Час виконання програми напряму залежить від кількості даних зібраних у платформі OpenCTI, оскільки чим більше окремих наборів вторгнень зберігається, тим більше буде надіслано запитів за допомогою API.

Результатом виконання програми є файл формату CSV, який містить колонки Name, Country, Description, Aliases, Vulnerabilities, Malware, Tools, Reports. Вони повністю задовольняють обрані у пункті 2.3 дані.

| Name | Country | Description | Aliases | Vulnerabilities | Malware | Tools | Reports |
|---|---|---|---|---|---|---|---|
| Rancoz | [] | | | [] | ['Dissecting Rancoz', 'Vice Sc | [] | ['https://blog.cyble.c |
| Sponsoring Access | [] | | | [] | ['Ballistic Bobcat'] | [] | ['https://www.welive |
| Astaroth | [] | | | [] | ['win.ousaban', 'MEKOTIO', ' | [] | ['https://blog.talosint |
| QUILTEDTIGER | [] | | | [] | ['NJRAT', 'UknownRAT', 'Qua | [] | [] |
| CUTTING KITTEN | ['Iran'] | CUTTING KITTEN is an Iran-ba | ['CUTTINGKITTEN', 'ITS | [] | [] | [] | [] |
| UNC5051 | [] | | | [] | [] | [] | ['https://advantage.r |
| MOLOTOV JACKAL | [] | MOLOTOV JACKAL is a hacktiv | ['MOLOTOVJACKAL', 'P | [] | [] | [] | [] |
| Chinaz | [] | | | [] | [] | [] | ['https://asec.ahnlab |
| HOUND SPIDER | ['Russian Federation'] | HOUND SPIDER is a criminal g | ['HOUNDSPIDER', 'Cerb | [] | [] | [] | [] |
| Tycoon Group | [] | | | [] | [] | [] | ['https://www.trustwa |
| Migo | [] | | | [] | ['Migo'] | [] | ['https://www.cadosc |
| SABREPANDA | ['China'] | This adversary's activity was firs | ['SABRE PANDA'] | [] | ['COBALT', '9002', 'CavalryRA | [] | [] |
| BlackTech | ['China'] | [BlackTech](https://attack.mitre. | ['BlackTech', 'Palmerwor | [] | ['Waterbear', 'SOLOSHIP', 'F | ['PsExec'] | [] |
| DEEPPANDA | [] | | | [] | ['MinionRAT', 'MinionRAT', 'IF | [] | [] |
| GCMAN | [] | [GCMAN](https://attack.mitre.or | ['GCMAN'] | [] | [] | [] | [] |
| WICKEDPANDA | [] | | | [] | ['RbDoor', 'COBALTSTRIKE', | [] | [] |
| APT-C-35 | ['India', 'South Asia'] | | | [] | [] | [] | ['https://mp.weixin.q |
| UNC2565 | [] | | | [] | ['GOOTLOADER', 'FONELAU | [] | ['https://www.mandia |
| SmartApe5G | [] | | | [] | [] | [] | ['https://www.esentir |
| HURRICANEPANDA | ['China'] | | ['HURRICANEPANDA', ' | [] | ['ZXSHELL.LOWBLOW', 'MA | [] | [] |
| Manic Menagerie | [] | | | ['CVE-2017-0213', 'CVE-2018- | ['Manic Menagerie'] | [] | ['https://unit42.paloa |
| Trident Ursa | ['Russian Federation'] | | | [] | [] | [] | ['https://github.com/ |
| FERAL SPIDER | [] | FERAL SPIDER is the develope | ['FERALSPIDER', 'Death | [] | ['DeathKitty'] | [] | [] |
| ShroudedSnooper | [] | | | [] | ['TOFULOAD', 'PipeSnoop', ' | [] | ['https://blog.talosint |
| Agenda | [] | | | [] | [] | [] | ['https://documents. |

Рисунок 2.9 — Частина отриманого CSV файлу

Для простого отримання усієї бажаної інформації і її подальшого ручного аналізу, такого файлу може бути достатньо.

Проте, якщо метою є візуалізація, або підрахунок статистики, звітність тощо, усі дані повинні бути сталого формату.

### 2.6 Нормалізація отриманих даних

Для того, щоб проводити нормалізацію, яка у цьому випадку визначає приведення усіх даних у колонках до сталого формату, попереднього необхідно провести аналіз.

В його ході було виявлено наступні проблеми:

1. Походження — через використання різних інтеграцій зовнішніх джерел, формат запису країн для багатьох наборів відрізняється. Десь це пусте значення, десь код країни, десь “unknown”.
2. Дублі — в інших колонках, окрім імені, опису та звітів, зустрічались повторення назв понять.

Для розв'язання зазначених проблем було створено програму norm.py, в тілі якої міститься лише одна функція process_csv, яка приймає на вхід назви необробленого та обробленого файлів. Код, що міститься у файлі, повністю представлений у Додатку А.

Окрім описаних раніше технічних засобів, програма використовує пакет pycountry [21], який надає доступ до бази даних яка відповідає стандарту ISO 3166 “Коди країн” [22], для приведення їх до повних назв.

Алгоритм програми нормалізації:

1. Отримання необхідної інформації — програма запитує у користувача назви необробленого та бажаного нормалізованого файлів.
2. Зчитування файлу — наданий користувачем CSV файл переводиться у DataFrame.
3. Обробка стовпців — значення для Country, Aliases, Vulnerabilities, Malware, Tools, Reports конвертуються з рядків у списки.
4. Видалення дублікатів — для стовпців Country, Aliases, Vulnerabilities, Malware, Tools.
5. Обробка походжень — стовпець Country нормалізується, використовуючи пакет pycountry для заміни кодів країн на їх повні назви, після чого зводиться до сталих значень, використовуючи словник синонімів.
6. Збереження у файл — програма записує оброблений DataFrame у CSV файл з назвою заданою користувачем.

Згаданий вище словник замін:

```python
synonym_dict = {
    'Russia': 'Russian Federation',
    'United States of America': 'United States',
    'Great Britain': 'United Kingdom',
    'England': 'United Kingdom',
    'Britain': 'United Kingdom',
    'Venezuela, Bolivarian Republic of': 'Venezuela',
    'Moldova, Republic of': 'Moldova'
    # Add more synonyms here
}
```

Рисунок 2.10 — Словник синонім для назв країн

Результатом виконання програми є файл формату CSV, по структурі аналогічний отриманому під час виконання cti.py, проте він є обробленим, де усі дані зведено до єдиного формату.

Назва необробленого файлу — threats.csv, а нормалізованого — output.csv.

Проблеми, які були наявні до нормалізації даних:

| Name | Country | Malware |
|---|---|---|
| APT12 | ['China', 'China'] | ['BUGJUICE', 'BUGJUICE', 'PLAYNICE', 'PLAYNICE', 'Lena', 'PLAYNICE', 'DARKMOON', 'PLAYNICE', 'DARKMOON', 'DARKMOON', 'PLAYNICE', 'DARKMOON', 'DARKMOON', 'DARKMOON', 'DARKMOON', 'SAWDRAIN', 'DARKMOON', 'SAWDRAIN', 'StoneLoader', 'DARKMOON', 'StoneLoader', 'LoadW', 'DARKMOON', 'PLAYNICE', 'PLAYNICE', 'DARKMOON', 'SAWDRAIN', 'DARKMOON', 'PLAYNICE', 'KOADIC', 'StoneLoader', 'BUGJUICE', 'PLAYNICE', 'StoneLoader', 'HWorm', 'Quasar', 'PLAYNICE', 'PLAYNICE', 'SAWDRAIN', 'DARKMOON', 'DARKMOON', 'BUGJUICE', 'Lena', 'PLAYNICE', 'SAWDRAIN', 'DARKMOON', 'StoneLoader', 'Quasar', 'StoneLoader', 'DARKMOON', 'BUGJUICE', 'SAWDRAIN', 'SAWDRAIN', 'DARKMOON', 'SAWDRAIN', 'StoneLoader', 'PLAYNICE', 'PLAYNICE', 'PLAYNICE', 'PLAYNICE', 'PLAYNICE', 'DARKMOON', 'BUGJUICE', 'DARKMOON', 'DARKMOON', 'DARKMOON', 'Quasar', 'BUGJUICE', 'DARKMOON', 'BUGJUICE', 'SAWDRAIN', 'DARKMOON', 'SAWDRAIN', 'Bisty', 'DARKMOON', 'DARKMOON', 'BUGJUICE', 'DARKMOON', 'PLAYNICE', 'DARKMOON', 'DARKMOON', 'SAWDRAIN', 'DARKMOON', 'PLAYNICE', 'DARKMOON', 'PLAYNICE'] |
| Ember Bear | ['Russian Federation'] | |
| FRATERNAL JACKAL | ['Iran'] | |
| UAC-0184 | [] | |
| Havildar Team | ['PK'] | |
| GambleForce | [] | |
| APT 28 | [] | |
| PERCUSSION SPIDER | ['TR'] | |

Рисунок 2.11 — Частина з необробленого файлу

Для наочності результатів нормалізації, ці ж рядки в обробленому файлі:

| Name | Country | Malware |
|---|---|---|
| APT12 | ['China'] | ['BUGJUICE', 'LoadW', 'SAWDRAIN', 'HWorm', 'StoneLoader', 'Lena', 'KOADIC', 'PLAYNICE', 'DARKMOON', 'Quasar', 'Bisty'] |
| Ember Bear | ['Russian Federation'] | |
| FRATERNAL JACKAL | ['Iran'] | |
| UAC-0184 | [] | |
| Havildar Team | ['Pakistan'] | |
| GambleForce | [] | |
| APT 28 | [] | |
| PERCUSSION SPIDER | ['Türkiye'] | |

Рисунок 2.12 — Частина з нормалізованого файлу

## Висновки до розділу 2

В даній роботі другий розділ було присвячено процесу отримання та нормалізації даних про кіберзагрози з використанням API платформи OpenCTI. Було надано причини використання зазначеної платформи автоматизації.

Під час дослідження було виявлено необхідність у програмній реалізації для автоматизації процесу групування даних, що значно підвищує ефективність роботи аналітиків та зменшує необхідний для обробки інформації час.

Для програмної реалізації було обґрунтовано використання мови програмування Python з пакетами Pandas, PyCTI та pycountry.

Розроблено програми, які дозволяють отримувати дані про загрози, нормалізувати їх та зберігати у форматі CSV для подальшого використання.

Підсумовуючи, проведена робота дозволяє автоматизувати значну частину процесу збору та обробки даних про кіберзагрози, що надає можливість швидкого та зручного реагування на нові загрози.

# 3 СИСТЕМА ЗБОРУ ІНФОРМАЦІЇ ПРО КІБЕРЗАГРОЗИ ТА ПОБУДОВИ СЕМАНТИЧНИХ МЕРЕЖ НА ЇЇ ОСНОВІ

## 3.1 Постановка задачі системи

Система збору інформації про кіберзагрози та побудови семантичних мереж має на меті забезпечити автоматизацію процесу групування та нормалізації даних з платформи OpenCTI та подальшої їх візуалізації. Ця система дозволяє ефективно аналізувати великі обсяги даних, виявляти взаємозв'язки між різними елементами загроз та надавати користувачам наочне зображення результатів наявного процесу розвідки.

Цілісна система, що поєднує програми, розглянуті до цього у другому розділі, та рішення формування семантичних мереж, має наступні можливості доступні користувачу:

1. Отримання даних про кіберзагрози з OpenCTI за допомогою API.
2. Нормалізацію наданого користувачем CSV файлу заданого формату.
3. Створення вебсторінок стосовно джерел кіберзагроз та інформації про них.
4. Побудову семантичних мереж для візуалізації даних за певним запитом.

Однією з переваг об’єднання в одну систему є можливість нормалізації та отримання семантичних мереж навіть не маючи розгорнутої та наповненої платформи OpenCTI. Користувачу було б достатньо лише надати CSV файл відповідного формату. Запланована система надає можливість як отримати бажаний результат з нуля, так і вже знаходячись на певному етапі.

### 3.2 Вибір технічних засобів для формування семантичних мереж за допомогою Python

Семантична мережа — орієнтований граф, тому для її побудови необхідно використовувати пакети для роботи з графами.

Для виконання поставленої задачі було обрано:

- NetworkX — це пакет Python для створення, маніпулювання та дослідження структури, динаміки та функцій складних мереж. Який містить структури даних для різних типів графів та стандартні алгоритми для роботи з ними [23].
- PyVis — пакет для візуалізації графів, який створено для Python. Вузлам можна надавати кольори, розміри, мітки та інші метадані [24]. Одними з його переваг є збереження графів у HTML файли та можливість взаємодії користувача зі створеною візуалізацією.

Для прикладу зображено результат програмного створення мережі з Рисунка 1.5:

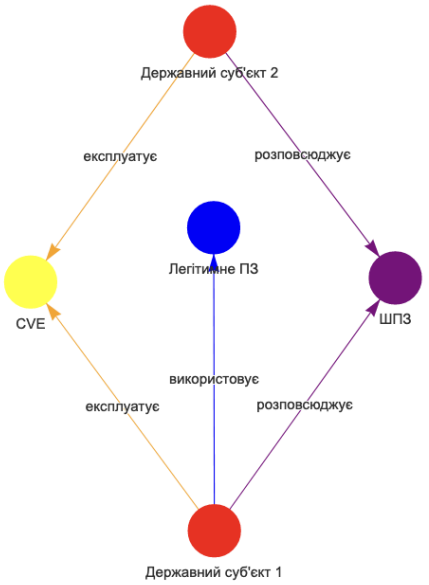



Рисунок 3.1 — Приклад програмно створеної семантичної мережі

Для отримання такого зображення виконуються наступні кроки:

1. Створення орієнтованого графа за допомогою класу пакета NetworkX DiGraph.
2. Створення словників з властивостями, які в цьому випадку є кольорами, для вузлів та ребер.
3. Створення масиву який містить необхідні ребра.
4. Додавання вузлів та ребер з вказаними властивостями за допомогою методів класу DiGraph add_node та add_edge.

```
G = nx.DiGraph()
nodes = {
    "CVE": "yellow",
    "Державний суб'єкт 1": "red",
    "Державний суб'єкт 2": "red",
    "Легітимне ПЗ": "blue",
    "ШПЗ": "purple"
}
edge_colors = {
    "експлуатує": "orange",
    "використовує": "blue",
    "розповсюджує": "purple"
}
edges = [
    ("Державний суб'єкт 1", "CVE", "експлуатує"),
    ("Державний суб'єкт 2", "CVE", "експлуатує"),
    ("Державний суб'єкт 1", "Легітимне ПЗ", "використовує"),
    ("Державний суб'єкт 1", "ШПЗ", "розповсюджує"),
    ("Державний суб'єкт 2", "ШПЗ", "розповсюджує")
]
for node, color in nodes.items():
    G.add_node(node, color=color)
for src, dst, label in edges:
    G.add_edge(src, dst, label=label, color=edge_colors[label])
```

Рисунок 3.2 — Створення графу за допомогою NetworkX

5. Створення об'єкта візуалізації орієнтованого графа за допомогою класу Network пакету PyVis з параметром directed=True.
6. Додавання вузлів та ребер з властивостями, попередньо створеними за допомогою NetworkX, використовуючи методи класу Network add_node та add_edge.
7. Зміна довжини ребер за допомогою методу repulsion з параметром spring_layout=200.

8. Збереження візуалізації у HTML файл, використовуючи метод show, передаючи йому параметр з бажаною назвою файлу. Замість show може бути використаний save_graph.

```
net = Network(notebook=True, directed=True)
for node, data in G.nodes(data=True):
    net.add_node(node, label=node, color=data['color'])
for src, dst, data in G.edges(data=True):
    net.add_edge(src, dst, label=data['label'], color=data['color'], smooth=False)
net.repulsion(spring_length=200)

net.show("network.html")
```

Рисунок 3.3 — Візуалізація графу за допомогою PyVis.

Таким чином, використовуючи обрані пакети, можна створювати візуально приємні та зрозумілі графи з гнучкими налаштуваннями, які при відкритті відображаються у браузері за замовчуванням.

### 3.3 Створення вебсторінок на основі зібраних даних

Оскільки під час вибору бажаних даних було отримано в тому числі інформацію, яка не буде зображена на семантичних мережах, наприклад, опис, розроблено програму, яка реалізує створення HTML сторінок для кожного з джерел загроз.

Таким чином, фахівець який буде працювати з цими даними може отримати доступ до зручного зображення текстових даних навіть без підключення до мережі.

Реалізація функціонала була написана у файлі web.py, який повністю викладено у Додатку А.

Алгоритм роботи програми:

1. Отримання назви CSV файлу від користувача.

2. Створення каталогу з назвою web-pages в поточному, де виконується програма.
3. Зчитування файлу вказаного користувачем у DataFrame пакету Pandas.
4. Циклічно для кожного рядку:
   a. Занесення значень для кожного стовпця в окрему змінну, порожні заміняються на “-”.
   b. Створення змінної, у якій зберігається ім’я майбутнього файлу, з заміною, такою що “APT 28” стає “apt_28”.
   c. Створення HTML-вмісту, використовуючи модуль html для переведення змінних.
   d. Збереження вмісту у файл з назвою, створеною під час кроку 4b, у каталозі, створеному під час кроку 2.

Результатом виконання програми є каталог web-pages, який містить кількість HTML файлів що дорівнює кількості рядків у заданому користувачем CSV файлі.

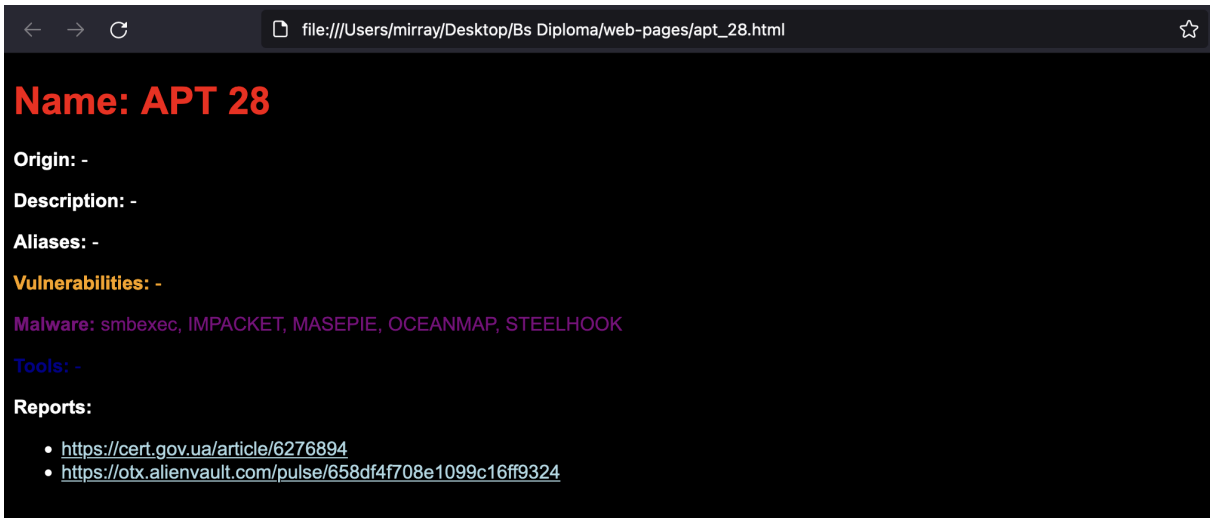



Рисунок 3.4 — Приклад відображення файлу web-pages/apt_28.html

### 3.4 Програмна реалізація семантичних мереж кіберзагроз

Для створення семантичних мереж з даних, отриманих до цього за допомогою API платформи OpenCTI, було розроблено програму graph.py, повний код якої викладено у Додатку А.

Програма складається з 6 функцій:

1. create_graph — приймає 4 параметри: data, option, filter_value, filter_column, які їй передає функція create_and_save_graph. Ініціалізує пустий орієнтований граф за допомогою класу DiGraph, та бере поточний каталог за допомогою модуля os. Основна логіка полягає у використанні фільтрованих або нефільтрованих даних для додавання вузлів та ребер до графа. Повертає граф бібліотеки NetworkX.
2. add_nodes_and_edges — приймає 4 параметри: threat_name, malwares, vulnerabilities, tools, які їй передає функція create_graph. Першочергово вона створює вузли для джерел загроз, додаючи їм як параметр гіперпосилання на відповідний файл, наявний у результаті виконання програми web.py, та червоний колір. Далі створюється вузол окремо для кожного ШПЗ, легітимного засобу, яким зловживають, та кожної вразливості, перевіряючи чи він вже не існує. При їх створенні також задано параметр кольору, фіолетовий для шкідливого програмного забезпечення, жовтий для вразливостей та синій для легітимних інструментів. Також створюються ребра — зв’язки, аналогічних кольорів.
3. filter_data — приймає 3 параметри: data, filter_value, filter_column. Викликається функцією create_graph, якщо було отримано параметри фільтрації. Фільтрує дані, перевіряючи наявність filter_value у відповідному стовпці filter_column.
4. save_graph — приймає 2 параметри: G, file_name, які отримує від функції create_and_save_graph. Встановлює розмір вузлів залежно від їх ступеня. Створює візуалізацію за допомогою класу Network з

параметрами, які відповідають за розмір та кольори, який її будує за допомогою методу from_nx, який отримує граф бібліотеки NetworkX. Далі викликається метод force_atlas_2based, який є імплементацією ForceAtlas2 — макету, спрямованого на силу: він імітує фізичну систему для просторової організації мережі. Вузли відштовхуються один від одного, як заряджені частинки, в той час, як ребра притягують свої вузли, як пружини [25]. Зберігає семантичну мережу у HTML файл, назва якого міститься в параметрі file_name.

5. create_and_save_graph — приймає 5 параметрів: input_file, output_file, option, filter_value, filter_column з основної функції програми, два останніх за замовчуванням встановлені як порожні. Зчитує дані з input_file у DataFrame, після чого передає його функції create_graph як параметр data, разом з option, filter_value та filter_column. Після цього викликає функцію save_graph, якій передає параметри G — результат виконання попереднього виклику та output_file як file_name.

6. main — забезпечує інтерфейс користувача, від якого й отримує дані для подальшої передачі до create_and_save_graph. Назва CSV файлу з даними про кіберзагрози — input_file, назва HTML файлу бажаної семантичної мережі — output_file, вибір пункту меню — option, вибір стовпця, у якому знаходиться елемент для якого буде будуватись граф — filter_column, ім’я елементу — filter_value.

При роботі з програмою graph.py, користувач отримає декілька запитань, надавши відповідь на які, отримає бажану семантичну мережу.

Перше питання — назва вхідного CSV файлу, друге — назва вихідного HTML файлу, третє — основа семантичної мережі. На третє питання користувач має обрати одну з запропонованих опцій, а саме:

1. Повна
2. Повна лише по ШПЗ
3. Повна лише по вразливостях

4. Повна лише по застосунках
5. Власна

При виборі п’ятого варіанту, буде поставлено четверте питання щодо бажаного фільтру, а саме:

1. По назві загрози
2. По назві ШПЗ
3. По назві вразливості
4. По назві застосунку
5. По країні регіону походження

Після вибору будь-якої з запропонованих можливостей буде виведено повний список доступних значень для даного фільтру.

Приклади результатів для деяких з можливих запитів користувача:

- Повна семантична мережа

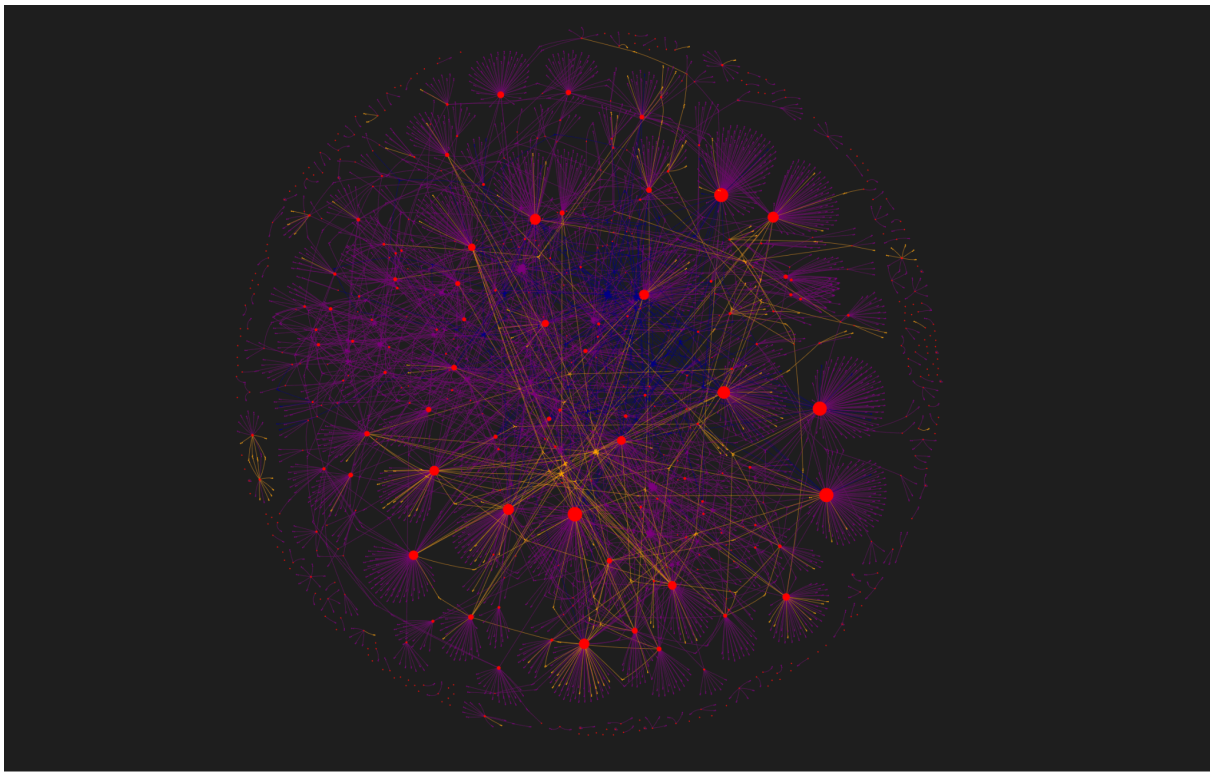


Рисунок 3.5 — Повна семантична мережа

Типово, кожен граф візуалізований за допомогою PyVis відкривається віддалено. При наближенні вузли та ребра мають наступний вигляд:

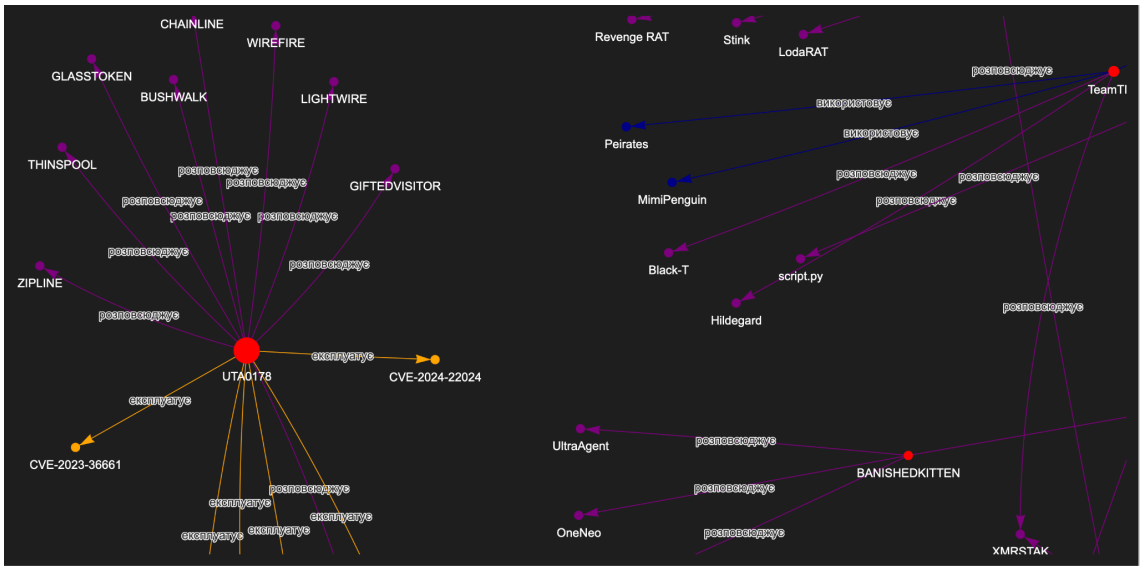



Рисунок 3.6 — Частина повної семантичної мережі

- Семантична мережа лише по вразливостях, які експлуатують суб’єкти загроз:

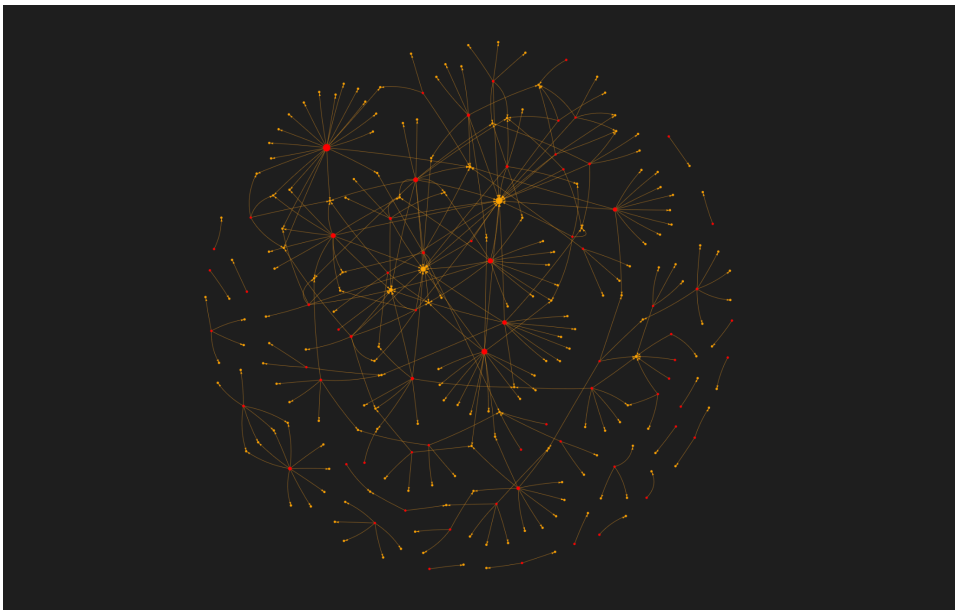


Рисунок 3.7 — Семантична мережа лише по вразливостях

Наближення до найбільшого червоного вузла, тобто джерела, для якого у зібраній з платформи OpenCTI інформації, знайдено найбільшу кількість експлуатованих вразливостей:

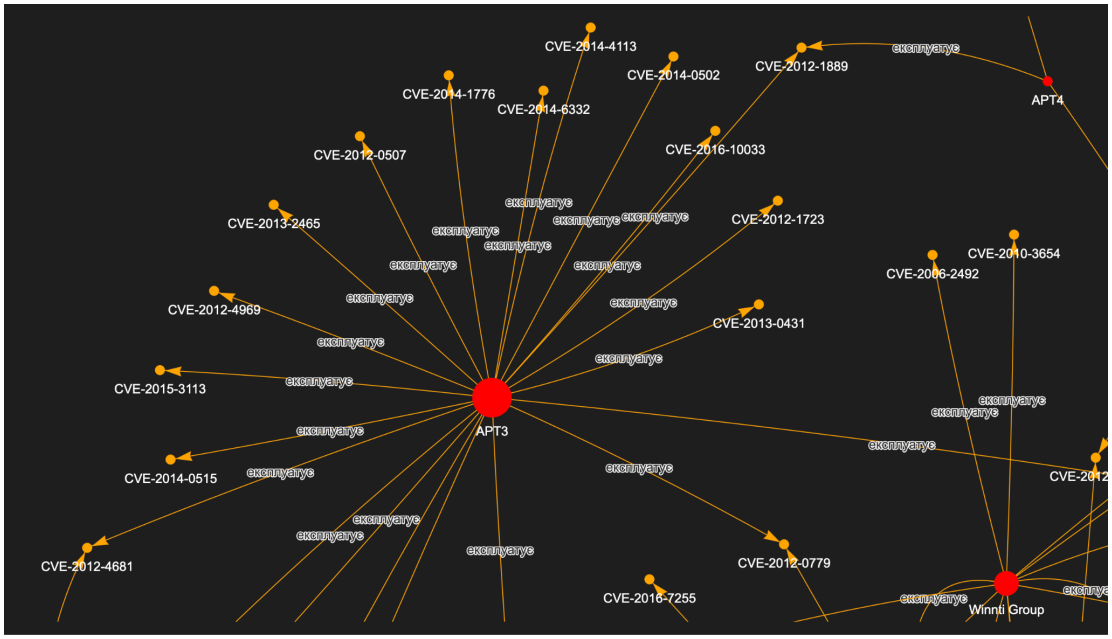



Рисунок 3.8 — Частина семантичної мережі по вразливостях

Таким чином наочно видно, що CVE-2012-1889 експлуатується суб’єктами APT3 та APT4.

- Власна семантична мережа по назві легітимного інструменту ftp

При побудові семантичних мереж за певним фільтром, програма будує граф, який містить усі зв’язки пов’язаних з запитом джерел.

Таким чином, при введеному, наприклад, ftp, користувач отримує джерела, які їм зловживають, та повний їх арсенал.

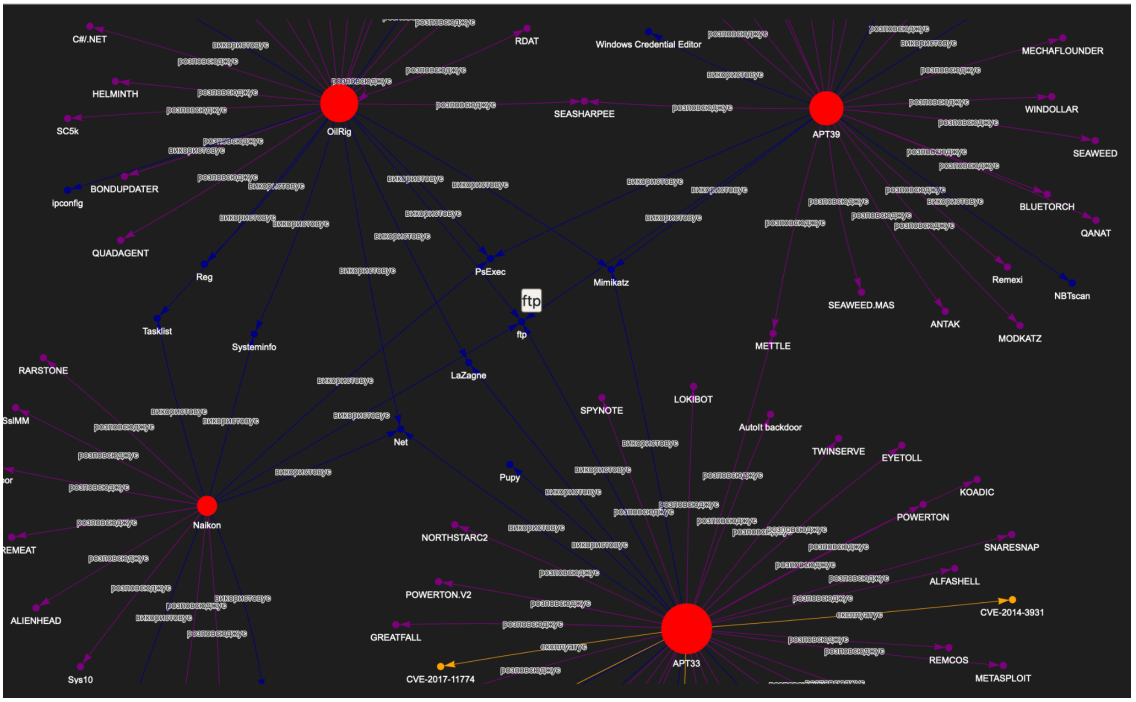



Рисунок 3.9 — Наближена семантична мережа за запитом по застосунку “ftp”

Наочно видно, що суб’єкти APT33, APT39, Nailkon та OilRig у своїх кампаніях використовували ftp, в той самий час, як й усі з них, окрім APT39, зловживали й розглянутим у Пункті 1.1 цієї роботи засобом Net.

### 3.5 Об’єднання у систему

Цілісна система являє собою об’єднання розроблених до цього програм cti.py, norm.py, web.py та graph.py. Використання мови програмування Python у цьому випадку дозволяє створити окрему програму main.py, у яку імпортуються функції вищеперелічених програм. Повний код викладено у Додатку А цієї роботи.

При виконанні інтерфейс програми відображається у командному рядку, у вигляді текстового меню, з яким взаємодіє користувач. Він надає можливість отримання даних з OpenCTI, їх нормалізації, створення вебсторінок та побудови

семантичних мереж, незалежно одне від іншого. Також присутня опція виходу з програми.

Поверхнево алгоритм програми наступний:

1. Цикл отримання бажаної дії з програмою від користувача.
2. Опція 1:
   a. Отримання URL та API-ключа платформи OpenCTI.
   b. Виклик функцій програми cti.py.
   c. Отримання бажаної назви CSV файлу.
   d. Збереження даних у вказаний користувачем файл.
3. Опція 2:
   a. Отримання назв вхідного та вихідного CSV файлів.
   b. Виклик функції програми norm.py.
   c. Збереження оброблених даних у вказаний вихідний файл.
4. Опція 3:
   a. Отримання назви вхідного CSV файлу.
   b. Виклик функції програми web.py.
   c. Збереження створених HTML файлів у каталог web-pages.
5. Опція 4:
   a. Отримання назв вхідного CSV та вихідного HTML файлів.
   b. Отримання типу семантичної мережі.
   c. Опція “5. Власна”:
      i. Отримання бажаного значення фільтра.
   d. Виклик функції програми graph.py.
   e. Збереження створеної семантичної мережі у вказаний вихідний файл.
6. Опція 5:
   a. Розрив циклу, вихід із програми.

Повна блок-схема алгоритму системи:

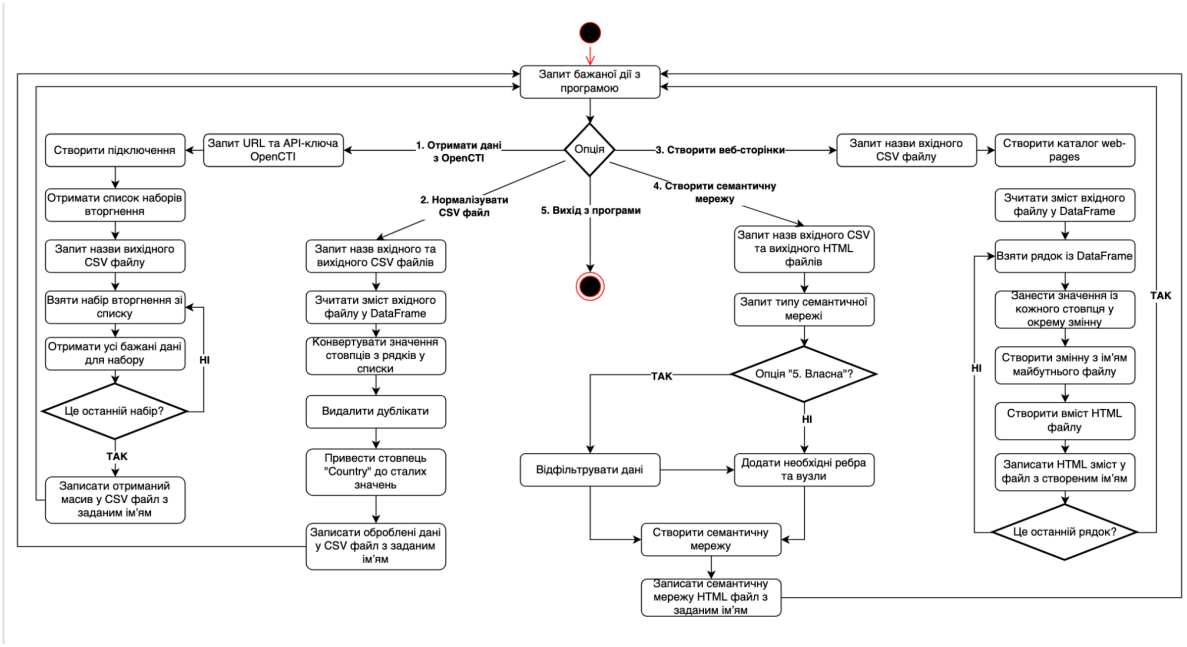



Рисунок 3.10 — Блок-схема повного алгоритму системи

### 3.6 Практичне застосування системи

Для проведення тестування працездатності системи розглядається наступна ситуація: фахівець департаменту кібербезпеки, який обслуговує українську організацію, має мету впровадити або поліпшити наявні засоби захисту проти кіберзагроз. До цього він займався підвищенням ефективності процесу розвідки, тому має впроваджене та наповнене даними рішення автоматизації OpenCTI. Враховуючи найбільш потенційних нападників — суб’єктів, які мають походження з Російської Федерації, він хоче розглянути саме їх арсенал.

Запропоноване рішення чудово впорається з наявною задачею, для цього виконуються наступні кроки:

1. Фахівець обирає опцію “Отримати дані з OpenCTI”, та вводить шлях до платформи та свій API-ключ. Назву вихідного файлу він вказує як “test-threat.csv”

Результатом першого кроку буде наступна відповідь програми:

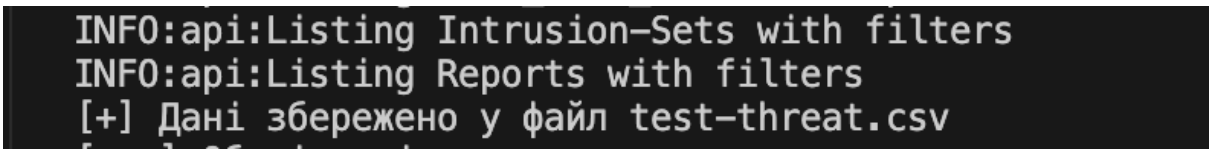



Рисунок 3.11 — Успішне отримання даних з OpenCTI

2. Далі фахівець обирає опцію “Нормалізувати CSV файл”, вказуючи щойно створений “test-threat.csv”, як вхідний, а “test-output.csv” як бажаний.

Результатом другого кроку буде наступна відповідь програми:

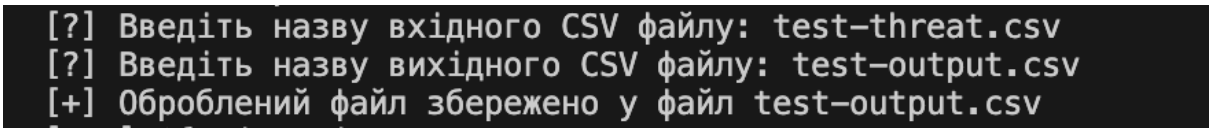



Рисунок 3.12 — Успішна нормалізація даних з CSV файлу

3. Знаючи про ймовірність відсутності підключення до мережі, фахівець обирає опцію “Створити вебсторінки”, вказуючи “test-output.csv” як вхідний файл, з метою завжди мати доступ до зручного зображення інформації про джерела загроз.

Результатом третього кроку буде наступна відповідь програми:

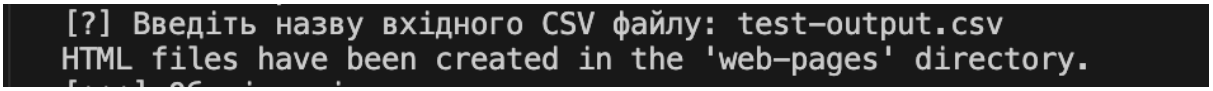



Рисунок 3.13 — Успішне створення вебсторінок

4. Для фінального групування даних та створення зображення бажаної інформації, фахівець обирає опцію “Створити семантичну мережу”, після чого виконує запит “Власна, По країні походження”, вказуючи значення “Russian Federation”.

Результатом четвертого кроку буде наступна відповідь програми:

```
[***] Оберіть фільтр:
1. По назві загрози
2. По назві ШПЗ
3. По назві вразливості
4. По назві застосунку
5. По країні регіону походження
[?] Ваш вибір: 5
[*] Доступні значення для Country:
Canada, Moldova, Spain, Ukraine, South Asia, Türkiye, Indonesia, Iran, Nigeria, Middle East, Tunisia, Syrian Arab Republic, Pakistan, Venezuela, Colombia, India
, East Asia, South America, South Korea, Palestine, Georgia, United Arab Emirates, Belarus, China, Lebanon, North Korea, North America, United States, Vietnam,
Brazil, Eastern Europe, Russian Federation
[?] Введіть значення фільтру для Country: Russian Federation
[+] Граф збережено у файлі test-graph.html
```

Рисунок 3.14 — Успішне створення семантичної мережі

Відкривши створений HTML файл, відобразиться орієнтований граф, який містить усі зв’язки джерел, для яких асоційована вказана країна походження:

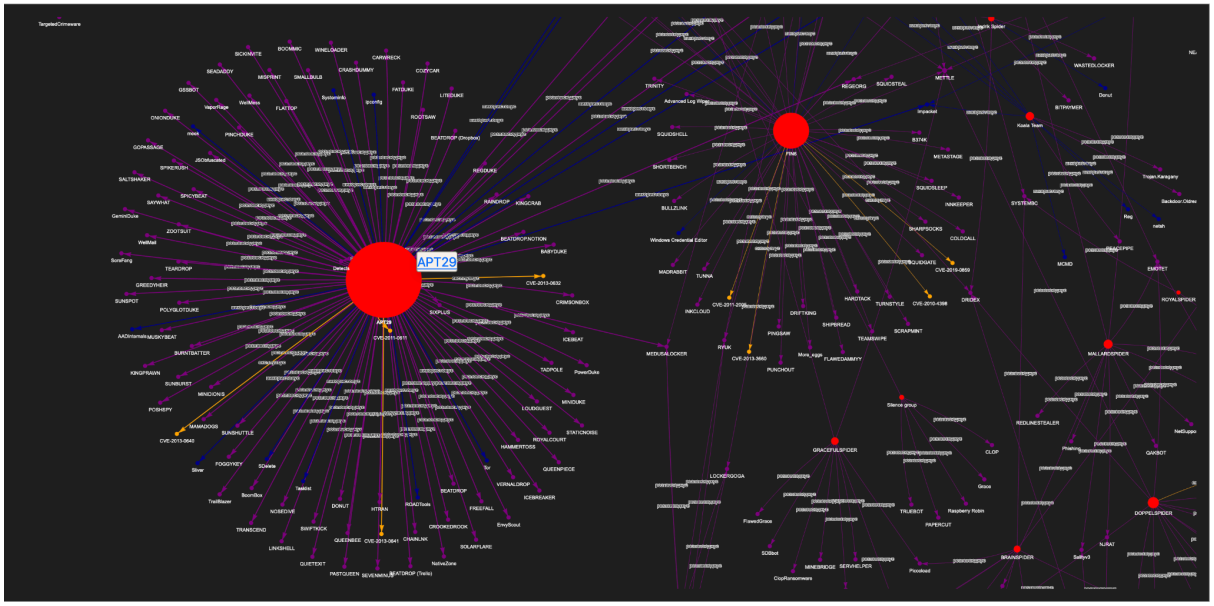



Рисунок 3.15 — Наближена семантична мережа за запитом по країні “Russian Federation”

Натискання на вузол джерела загрози відображає гіперпосилання, перейшовши по якому відкриється файл “web-pages/*.html”. В цьому випадку, натиснувши на вузол APT29 та перейшовши по гіперпосиланню, користувач потрапить на сторінку “web-pages/apt29.html”, яка має наступний вигляд:

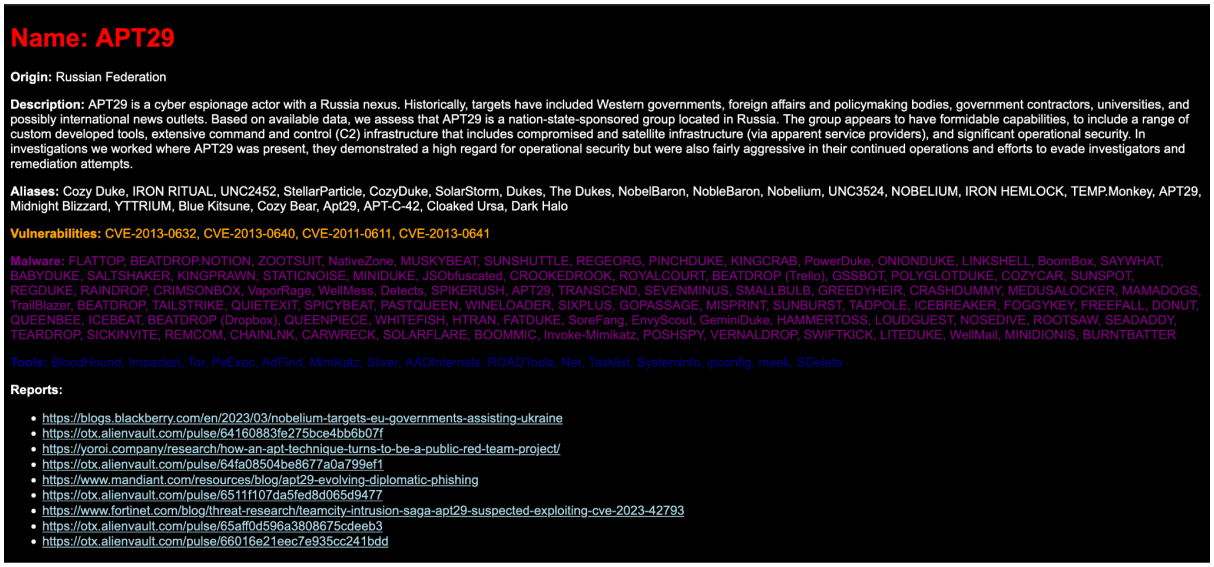



Рисунок 3.16 — Відображення файлу web-pages/apt29.html

## Висновки до розділу 3

У третьому розділі даної роботи було реалізовано власну систему збору інформації про кіберзагрози та побудови семантичних мереж на її основі. Використання мови програмування Python з пакетами NetworkX і PyVis дозволило створити ефективний інструмент для візуалізації даних з платформи OpenCTI шляхом побудови семантичних мереж.

Було розроблено програму для створення окремих вебсторінок для кожного з джерел загроз. Такий підхід дозволив впровадити гіперпосилання на семантичних мережах та отримувати зручне графічне текстове представлення зібраної інформації.

Усі попередньо реалізовані програмні рішення було об’єднано у цілісну систему з інтерфейсом у командному рядку, яка надає користувачу зрозуміле текстове меню для виконання необхідних задач.

Також було запропоновано приклад практичної потреби подібного рішення та протестовано використання системи для розв'язання поставленої проблеми.

# ВИСНОВКИ

Результатом цієї роботи є система збору інформації про кіберзагрози та побудови семантичних мереж на її основі, що включає автоматизацію процесу отримання, нормалізації та візуалізації даних з використанням платформи OpenCTI, мови програмування Python, її зовнішніх пакетів та модулів.

Надано теоретичні відомості щодо таких понять як кіберзагрози, розвідка у сфері кібербезпеки та семантичні мережі. Для загроз було розглянуто форми та джерела походження, з встановленням основних класів суб'єктів. Описано потребу у впровадженні ефективного процесу збору інформації та можливості його автоматизації за допомогою платформ з відкритим вихідним кодом.

За допомогою використання пакета PyCTI, як клієнта OpenCTI для Python, було розроблено програмну реалізацію отримання бажаних даних з платформи, з подальшою їх нормалізації за допомогою пакетів pandas та pycountry, що корисно для роботи з великими обсягами інформації.

Запропонована система дозволяє створювати вебсторінки з інформацією про кіберзагрози, тим самим забезпечує зручний доступ до детальної інформації про загрози та їхні атрибути через браузер навіть в умовах відсутності з'єднання з мережею.

Створено програмний механізм автоматизованого формування семантичних мереж за запитом користувача за допомогою пакетів NetworkX та PyVis, що створює можливість виявлення неявних зв'язків між джерелами загроз та покращує як стратегічну, так і операційну розвідку стосовно кіберзагроз. Використовуючи можливості вищезгаданих пакетів було створено гіперпосилання прямо з вузла на вебсторінку з інформацією про нього.

На прикладі практичної задачі було продемонстровано застосування розробленої системи для аналізу загроз, пов'язаних з Російською Федерацією. Це підкреслює високу корисність системи для фахівців з кібербезпеки, які з її допомогою зможуть підвищити ефективність операційного аналізу.

Таким чином, запропонована система, яка з’явилась в результаті дослідження процесу автоматизації збору інформації про кіберзагрози, може принести корисну користь багатьом організаціям при використанні її для: моніторингу потенційних загроз для інформаційних систем, аналізу ефективності впровадженого процесу розвідки, підвищення зрозумілості звітів для нетехнічних осіб.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ


1. Кібероперації рф: нові цілі, інструменти та групи. Аналітика хакерських атак проти України за 2 півріччя 2023 року [Електронний ресурс] — Режим доступу до ресурсу: https://cip.gov.ua/ua/news/kiberoperaciyi-rf-novi-cili-instrumenti-ta-grupi-analitika-khakerskikh-atak-proti-ukrayini-za-2-pivrichchya-2023-roku
2. Cyber Threat | NIST CSRC [Електронний ресурс] — Режим доступу до ресурсу: https://csrc.nist.gov/glossary/term/cyber_threat
3. What Is Malware? | Cisco [Електронний ресурс] — Режим доступу до ресурсу: https://www.cisco.com/site/us/en/learn/topics/security/what-is-malware.html
4. What is a CVE? | Red Hat[Електронний ресурс] — Режим доступу до ресурсу: https://www.redhat.com/en/topics/security/what-is-cve
5. Locked, Loaded, and in the Wrong Hands: Legitimate Tools Weaponized for Ransomware in 2021 | Trend Micro [Електронний ресурс] — Режим доступу до ресурсу: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/locked-loaded-and-in-the-wrong-hands-legitimate-tools-weaponized-for-ransomware-in-2021
6. What is proof of concept (PoC) exploit? | TechTarget [Електронний ресурс] — Режим доступу до ресурсу: https://www.techtarget.com/searchsecurity/definition/proof-of-concept-PoC-exploit
7. Net.exe Utility | Microsoft Learn [Електронний ресурс] — Режим доступу до ресурсу: https://learn.microsoft.com/en-us/previous-versions/windows/embedded/aa939914(v=winembedded.5)

8. Net, Software S0039 | MITRE ATT&CK® [Електронний ресурс] — Режим доступу до ресурсу: https://attack.mitre.org/software/S0039/
9. What is a Threat Actor? | IBM [Електронний ресурс] — Режим доступу до ресурсу: https://www.ibm.com/topics/threat-actor
10. Script Kiddies | BugCrowd [Електронний ресурс] — Режим доступу до ресурсу: https://www.bugcrowd.com/glossary/script-kiddies/
11. What is an Advanced Persistent Threat (APT)? | CrowdStrike [Електронний ресурс] — Режим доступу до ресурсу: https://www.crowdstrike.com/cybersecurity-101/advanced-persistent-threat-apt
12. Midnight Blizzard: Guidance for responders on nation-state attack | Microsoft Security Blog [Електронний ресурс] — Режим доступу до ресурсу: https://www.microsoft.com/en-us/security/blog/2024/01/25/midnight-blizzard-guidance-for-responders-on-nation-state-attack/
13. What is Cyber Threat Intelligence? | Cisco [Електронний ресурс] — Режим доступу до ресурсу: https://www.cisco.com/c/en/us/products/security/what-is-cyber-threat-intelligence.html
14. What Is Cyber Threat Intelligence? | Microsoft Security [Електронний ресурс] — Режим доступу до ресурсу: https://www.microsoft.com/en-us/security/business/security-101/what-is-cyber-threat-intelligence
15. MISP | GitHub [Електронний ресурс] — Режим доступу до ресурсу: https://github.com/MISP/MISP
16. OpenCTI Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://docs.opencti.io/latest/
17. Semantic Network | AI Glossary | OpenTrain AI [Електронний ресурс] — Режим доступу до ресурсу: https://www.opentrain.ai/glossary/semantic-network

18. Language Rating 2024 | DOU [Електронний ресурс] — Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2024/
19. Pandas Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://pandas.pydata.org/docs/
20. OpenCTI client for Python Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://opencti-client-for-python.readthedocs.io/
21. Pycountry | PyPI [Електронний ресурс] — Режим доступу до ресурсу: https://pypi.org/project/pycountry/
22. ISO 3166 — Country Codes [Електронний ресурс] — Режим доступу до ресурсу: https://www.iso.org/iso-3166-country-codes.html
23. NetworkX Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://networkx.org/
24. Interactive network visualizations — PyVis Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://pyvis.readthedocs.io
25. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software [Електронний ресурс] — Режим доступу до ресурсу: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679

# ДОДАТОК А

## Код отримання інформації з OpenCTI:

## cti.py

```
import pycti
import urllib3
import pandas as pd

urllib3.disable_warnings()

def openCTI(opencti_url, api_key):
	global opencti_api_client
	opencti_api_client = pycti.OpenCTIApiClient(opencti_url, api_key)

def threatOrigin(threatName):
	intrusion_set = opencti_api_client.intrusion_set.read(
	filters={
		"mode": "and",
		"filters": [{"key": "name", "values": [threatName]}],
		"filterGroups": [],
	}
	)

	stix_relations = opencti_api_client.stix_core_relationship.list(
	fromId=intrusion_set["id"], relationship_type=["originates-from"]
	)

	origins = []
	for stix_relation in stix_relations:
	origins.append(str(stix_relation["to"]["name"]))
	return origins

def threatVulns(threatName):
	intrusion_set = opencti_api_client.intrusion_set.read(
	filters={
		"mode": "and",
		"filters": [{"key": "name", "values": [threatName]}],
		"filterGroups": [],
	}
	)

	stix_relations = opencti_api_client.stix_core_relationship.list(
	fromId=intrusion_set["id"], toTypes=["Vulnerability"]
```

```
        )

        vulns = []
        for stix_relation in stix_relations:
        vulns.append(str(stix_relation["to"]["name"]))
        return vulns

def threatMalw(threatName):
        intrusion_set = opencti_api_client.intrusion_set.read(
        filters={
                "mode": "and",
                "filters": [{"key": "name", "values": [threatName]}],
                "filterGroups": [],
        }
        )

        stix_relations = opencti_api_client.stix_core_relationship.list(
        fromId=intrusion_set["id"], toTypes=["Malware"]
        )

        malw = []
        for stix_relation in stix_relations:
        malw.append(str(stix_relation["to"]["name"]))
        return malw

def threatTools(threatName):
        intrusion_set = opencti_api_client.intrusion_set.read(
        filters={
                "mode": "and",
                "filters": [{"key": "name", "values": [threatName]}],
                "filterGroups": [],
        }
        )

        stix_relations = opencti_api_client.stix_core_relationship.list(
        fromId=intrusion_set["id"], toTypes=["Tool"]
        )

        tools = []
        for stix_relation in stix_relations:
        tools.append(str(stix_relation["to"]["name"]))
        return tools

def threatReports(threatName):
        intrusion_set = opencti_api_client.intrusion_set.read(
```

```
        filters={
                "mode": "and",
                "filters": [{"key": "name", "values": [threatName]}],
                "filterGroups": [],
        }
        )
        reports = opencti_api_client.report.list(
        filters={
                "mode": "and",
                "filters": [{"key": "objects", "values": [intrusion_set["id"]]}],
                "filterGroups": [],
        },
        orderBy="published",
        orderMode="asc",
        )

        urls = []
        for report in reports:
        ext = report["externalReferences"]
        for ref in ext:
                urls.append(ref['url'])
        return urls

def getThreats():
        global threats
        threats = opencti_api_client.intrusion_set.list()

def threatsCSV(filename):
        threat_data = []

        for threat in threats:
        threat_data.append({
                'Name': threat["name"],
                'Country': threatOrigin(threat["name"]),
                'Description': threat["description"],
                'Aliases': threat["aliases"],
                'Vulnerabilities': threatVulns(threat["name"]),
                'Malware': threatMalw(threat["name"]),
                'Tools': threatTools(threat["name"]),
                'Reports': threatReports(threat["name"])
        })

        df = pd.DataFrame(threat_data)
        df.to_csv(filename, index=False)
```

```
if __name__ == "__main__":
	opencti_url = input("[?]Введіть URL OpenCTI: ")
	api_key = input("[?]Введіть API ключ OpenCTI: ")
	openCTI(opencti_url, api_key)
	getThreats()
	output_file = input("[?]Введіть назву вихідного CSV файлу: ")
	threatsCSV(output_file)
	print(f"[+] Дані збережено у файл {output_file}")
```

Код нормалізації CSV файлу:

norm.py

```
import pandas as pd
import ast
import pycountry

def process_csv(input_csv_path, output_csv_path):
	synonym_dict = {
	'Russia': 'Russian Federation',
	'United States of America': 'United States',
	'Great Britain': 'United Kingdom',
	'England': 'United Kingdom',
	'Britain': 'United Kingdom',
	'Venezuela, Bolivarian Republic of': 'Venezuela',
	'Moldova, Republic of': 'Moldova'
	# Add more synonyms here
	}

	df = pd.read_csv(input_csv_path)

	columns_to_convert = ['Country', 'Aliases', 'Vulnerabilities', 'Malware', 'Tools',
'Reports']
	for column in columns_to_convert:
	df[column] = df[column].fillna('[]').apply(ast.literal_eval)

	columns_to_deduplicate = ['Country', 'Aliases', 'Vulnerabilities', 'Malware',
'Tools']
	for column in columns_to_deduplicate:
	df[column] = df[column].apply(lambda x: list(set(x)))

	def normalize_countries(countries):
	normalized_countries = set()
	for country in countries:
		country_normalized = country.strip().lower()
		if country_normalized == 'unknown':
		continue
```

```
            full_country_name = None
            for country_info in pycountry.countries:
            aliases = getattr(country_info, 'alt_spellings', [])
            if country_normalized in ([getattr(country_info, attr, '').lower() for attr
in ['alpha_2', 'alpha_3', 'name']] + [alias.lower() for alias in aliases]):
                  full_country_name = country_info.name
                  break
            if full_country_name:
            full_country_name = synonym_dict.get(full_country_name, full_country_name)
            normalized_countries.add(full_country_name)
            else:
            full_country_name = synonym_dict.get(country.title(), country.title())
            normalized_countries.add(full_country_name)
      return list(normalized_countries)

      df['Country'] = df['Country'].apply(normalize_countries)

      df.to_csv(output_csv_path, index=False)

if __name__ == "__main__":
      input_file = input("Enter the input CSV file path: ")
      output_file = input("Enter the output CSV file path: ")
      process_csv(input_file, output_file)
```

Код створення веб-сторінок:

web.py

```
import os
import pandas as pd
import html

def create_html_pages(input_csv):
      os.makedirs('web-pages', exist_ok=True)
      df = pd.read_csv(input_csv)

      for row in df.iterrows():
      threat_name = str(row['Name'])
      country = ', '.join(eval(row['Country'])) if pd.notna(row['Country']) and
row['Country'] != "[]" else '-'
      description = str(row['Description']) if pd.notna(row['Description']) and
row['Description'] != "[]" else '-'
      aliases = ', '.join(eval(row['Aliases'])) if pd.notna(row['Aliases']) and
row['Aliases'] != "[]" else '-'
      vulnerabilities = ', '.join(eval(row['Vulnerabilities'])) if
pd.notna(row['Vulnerabilities']) and row['Vulnerabilities'] != "[]" else '-'
```

```
        malware = ', '.join(eval(row['Malware'])) if pd.notna(row['Malware']) and
row['Malware'] != "[]" else '-'
        tools = ', '.join(eval(row['Tools'])) if pd.notna(row['Tools']) and row['Tools']
!= "[]" else '-'
        reports = eval(row['Reports']) if pd.notna(row['Reports']) and row['Reports'] !=
"[]" else []

        sanitized_threat_name = threat_name.lower().replace(' ', '_').replace('/', '_')
        html_content = f"""
        <!DOCTYPE html>
        <html lang="en">
        <head>
                <meta charset="UTF-8">
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <title>{html.escape(threat_name)}</title>
                <style>
                body {{
                        background-color: black;
                        color: white;
                        font-family: Arial, sans-serif;
                }}
                .name {{
                        color: red;
                }}
                .vulnerabilities {{
                        color: orange;
                }}
                .malware {{
                        color: purple;
                }}
                .tools {{
                        color: darkblue;
                }}
                a {{
                        color: lightblue;
                }}
                </style>
        </head>
        <body>
                <h1 class="name">Name: {html.escape(threat_name)}</h1>
                <p><strong>Origin:</strong> {html.escape(country)}</p>
                <p><strong>Description:</strong> {html.escape(description)}</p>
                <p><strong>Aliases:</strong> {html.escape(aliases)}</p>
                <p class="vulnerabilities"><strong>Vulnerabilities:</strong>
{html.escape(vulnerabilities)}</p>
```

```
                <p class="malware"><strong>Malware:</strong> {html.escape(malware)}</p>
                <p class="tools"><strong>Tools:</strong> {html.escape(tools)}</p>
                <p><strong>Reports:</strong></p>
                <ul>
                {"".join(f'<li><a
href="{html.escape(report)}">{html.escape(report)}</a></li>' for report in reports)}
                </ul>
        </body>
        </html>
        """

        output_file_path = os.path.join('web-pages', f"{sanitized_threat_name}.html")
        os.makedirs(os.path.dirname(output_file_path), exist_ok=True)
        with open(output_file_path, 'w', encoding='utf-8') as output_file:
                output_file.write(html_content)

        print("HTML files have been created in the 'web-pages' directory.")

if __name__ == "__main__":
        input_csv = input("Введіть назву вхідного CSV файлу: ")
        create_html_pages(input_csv)
```

Код програми для побудови семантичних мереж:

graph.py

```
import pandas as pd
import networkx as nx
import os
from pyvis.network import Network

def create_graph(data, option, filter_value=None, filter_column=None):
        G = nx.DiGraph()
        pwd = os.getcwd()

        def add_nodes_and_edges(threat_name, malwares, vulnerabilities, tools):
        if threat_name not in G.nodes:
                G.add_node(threat_name, label=threat_name, title=f"<a
href='file:///{pwd}/web-pages/{threat_name.lower().replace(' ', '_').replace('/',
'_')}.html'>{threat_name}</a>", color='red')

        for malware in malwares:
                if malware not in G.nodes:
                G.add_node(malware, title=malware, color='purple')
                G.add_edge(threat_name, malware, label="розповсюджує", color='purple')

        for vulnerability in vulnerabilities:
```

```
                if vulnerability not in G.nodes:
                G.add_node(vulnerability, title=vulnerability, color='orange')
                G.add_edge(threat_name, vulnerability, label="експлуатує", color='orange')

        for tool in tools:
                if tool not in G.nodes:
                G.add_node(tool, title=tool, color='darkblue')
                G.add_edge(threat_name, tool, label="використовує", color='darkblue')

        def filter_data(data, filter_value, filter_column):
        if filter_column in ['Malware', 'Vulnerabilities', 'Tools', 'Country']:
                return data[data[filter_column].apply(lambda x: filter_value in eval(x) if
x else False)]
        else:
                return data[data[filter_column].apply(lambda x: filter_value in str(x))]

        if filter_column and filter_value:
        data_filtered = filter_data(data, filter_value, filter_column)
        for _, row in data_filtered.iterrows():
                threat_name = row['Name']
                malwares = eval(row['Malware']) if row['Malware'] else []
                vulnerabilities = eval(row['Vulnerabilities']) if row['Vulnerabilities']
else []
                tools = eval(row['Tools']) if row['Tools'] else []
                add_nodes_and_edges(threat_name, malwares, vulnerabilities, tools)
        else:
        for _, row in data.iterrows():
                threat_name = row['Name']
                malwares = eval(row['Malware']) if row['Malware'] else []
                vulnerabilities = eval(row['Vulnerabilities']) if row['Vulnerabilities']
else []
                tools = eval(row['Tools']) if row['Tools'] else []

                if option == 1:
                add_nodes_and_edges(threat_name, malwares, vulnerabilities, tools)
                elif option == 2 and malwares:
                add_nodes_and_edges(threat_name, malwares, [], [])
                elif option == 3 and vulnerabilities:
                add_nodes_and_edges(threat_name, [], vulnerabilities, [])
                elif option == 4 and tools:
                add_nodes_and_edges(threat_name, [], [], tools)

        return G

def save_graph(G, file_name):
```

```
        degree_dict = {node: max(5, degree) for node, degree in G.degree()}
        nx.set_node_attributes(G, degree_dict, 'size')
        net = Network(notebook=False, directed=True, height="750px", width="100%",
bgcolor="#222222", font_color="white", cdn_resources='remote')
        net.from_nx(G)

net.force_atlas_2based(gravity=-50,central_gravity=0.01,spring_length=200,spring_strength
=0.05,damping=0.4,overlap=0)
        net.save_graph(file_name)
        print(f"[+] Граф збережено у файлі {file_name}")

def create_and_save_graph(input_file, output_file, option, filter_value=None,
filter_column=None):
        data = pd.read_csv(input_file)
        G = create_graph(data, option, filter_value, filter_column)
        save_graph(G, output_file)

if __name__ == "__main__":
        input_file = input("[?] Введіть назву вхідного CSV файлу: ")
        output_file = input("[?] Введіть назву вихідного HTML файлу: ")
        print("[***] Оберіть основу семантичної мережі:")
        print("1. Повна")
        print("2. Повна лише по ШПЗ")
        print("3. Повна лише по вразливостям")
        print("4. Повна лише по застосункам")
        print("5. Власна")

        option = int(input("[?] Ваш вибір: "))

        filter_value = None
        filter_column = None
        if option == 5:
        data = pd.read_csv(input_file)
        print("[***] Оберіть фільтр:")
        print("1. По назві загрози")
        print("2. По назві ШПЗ")
        print("3. По назві вразливості")
        print("4. По назві застосунку")
        print("5. По країні регіону походження")

        filter_option = int(input("[?] Ваш вибір: "))

        filter_column_map = {
                1: "Name",
                2: "Malware",
```

```
            3: "Vulnerabilities",
            4: "Tools",
            5: "Country"
        }
        filter_column = filter_column_map.get(filter_option, "")

        if filter_column:
            unique_values = set()
            if filter_column in ['Malware', 'Vulnerabilities', 'Tools', 'Country']:
            for value_list in data[filter_column].dropna():
                    unique_values.update(eval(value_list))
            else:
            unique_values = set(data[filter_column].dropna().unique())

            valid_input = False
            while not valid_input:
            print(f"[*] Доступні значення для {filter_column}:")
            print(", ".join(unique_values))

            filter_value = input(f"[?] Введіть значення фільтру для {filter_column}: ")

            if filter_value in unique_values:
                    valid_input = True
            else:
                    print("[!] Недійсне значення, будь ласка, спробуйте ще раз.")

        create_and_save_graph(input_file, output_file, option, filter_value,
filter_column)
```

Код основної програми інтерфейсу систему:

main.py

```
from cti import openCTI, getThreats, threatsCSV
from norm import process_csv
from graph import create_and_save_graph
from web import create_html_pages
import pandas as pd

def main():
        while True:
        print("[***] Оберіть дію:")
        print("1. Отримати дані з OpenCTI")
        print("2. Нормалізувати CSV файл")
        print("3. Створити веб-сторінки")
        print("4. Створити семантичну мережу")
        print("5. Вийти")
```

```
choice = int(input("[?] Ваш вибір: "))

if choice == 1:
	opencti_url = input("[?]Введіть URL OpenCTI: ")
	api_key = input("[?]Введіть API ключ OpenCTI: ")
	openCTI(opencti_url, api_key)
	getThreats()
	output_file = input("[?]Введіть назву вихідного CSV файлу: ")
	threatsCSV(output_file)
	print(f"[+] Дані збережено у файл {output_file}")

elif choice == 2:
	input_file = input("[?] Введіть назву вхідного CSV файлу: ")
	output_file = input("[?] Введіть назву вихідного CSV файлу: ")
	process_csv(input_file, output_file)
	print(f"[+] Оброблений файл збережено у файл {output_file}")

elif choice == 3:
	input_file = input("[?] Введіть назву вхідного CSV файлу: ")
	create_html_pages(input_file)

elif choice == 4:
	input_file = input("[?] Введіть назву вхідного CSV файлу: ")
	output_file = input("[?] Введіть назву вихідного HTML файлу: ")
	print("[***] Оберіть основу семантичної мережі:")
	print("1. Повна")
	print("2. Повна лише по ШПЗ")
	print("3. Повна лише по вразливостям")
	print("4. Повна лише по застосункам")
	print("5. Власний")

	option = int(input("[?] Ваш вибір: "))

	filter_value = None
	filter_column = None
	if option == 5:
	data = pd.read_csv(input_file)
	print("[***] Оберіть фільтр:")
	print("1. По назві загрози")
	print("2. По назві ШПЗ")
	print("3. По назві вразливості")
	print("4. По назві застосунку")
	print("5. По країні регіону походження")
```

```
            filter_option = int(input("[?] Ваш вибір: "))

            filter_column_map = {
                  1: "Name",
                  2: "Malware",
                  3: "Vulnerabilities",
                  4: "Tools",
                  5: "Country"
            }
            filter_column = filter_column_map.get(filter_option, "")

            if filter_column:
                  unique_values = set()
                  if filter_column in ['Malware', 'Vulnerabilities', 'Tools',
'Country']:
                        for value_list in data[filter_column].dropna():
                        unique_values.update(eval(value_list))
                  else:
                        unique_values = set(data[filter_column].dropna().unique())

                  valid_input = False
                  while not valid_input:
                        print(f"[*] Доступні значення для {filter_column}:")
                        print(", ".join(unique_values))

                        filter_value = input(f"[?] Введіть значення фільтру для
{filter_column}: ")

                        if filter_value in unique_values:
                        valid_input = True
                        else:
                        print("[!] Недійсне значення, будь ласка, спробуйте ще раз.")

            create_and_save_graph(input_file, output_file, option, filter_value,
filter_column)

      elif choice == 5:
            print("[*] Вихід з програми")
            break

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