Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«__» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: «Програмний модуль виявлення маркерів компрометації в роботі вебдодатків»

Виконавець: студентка IV курсу, групи КБ-43

_____ **Маргарита ПЕШКОВА**
(підпис) (ім'я прізвище)

	Підпис	**Ім'я, прізвище**
Керівник роботи		Олександр ТОРОШАНКО
Нормоконтроль		Юрій БАБЕНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студентці **КБ-43** **Пешковій Маргариті Романівні**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Програмний модуль виявлення маркерів компрометації в роботі вебдодатків

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та
захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Вебтехнології, методика виявлення маркерів компрометації у вебзапитах, шаблони вебатак для сигнатурного виявлення, платформи кіберрозвідки.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно дослідити існуючі рішення збору та управління індикаторами компрометації. Дослідити структуру логів вебзапитів, створити програмний модуль перевірки веблогів на наявність маркерів компрометації при атаках на вебдодатки.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка програмного модулю для виявлення

маркерів компрометації в роботі вебдодатків

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняла до виконання _____ Маргарита ПЕШКОВА
(підпис) (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 25.12.2024	*виконано*
2	Аналіз літератури	26.12.2024 –18.02.2025	*виконано*
3	Дослідження напрямку Threat Intelligence, індикаторів компрометації	19.02.2025 – 16.03.2025	*виконано*
4	Аналіз функціональних можливостей існуючих платформ кіберрозвідки	17.03.2025 – 28.03.2025	*виконано*
5	Огляд архітектури та технологій платформи OpenCTI	29.03.2025 – 10.04.2025	*виконано*
6	Вибір технологій, модулів та бібліотек для реалізації програмного модуля	11.04.2025 – 15.04. 2025	*виконано*
7	Розробка архітектури та логіки роботи програмного модуля	16.04. 2025 – 26.04. 2025	*виконано*
8	Реалізація програмного модуля	27.04.2025 – 23.05.2025	*виконано*
9	Проведення тестування модуля	24.05.2025 – 02.06.2025	*виконано*
10	Оформлення пояснювальної записки	03.06.2025 – 13.06.2025	*виконано*

Завдання видав _____ Олександр ТОРОШАНКО
(підпис) (ім'я, прізвище)

Завдання прийняла до виконання _____ Маргарита ПЕШКОВА
(підпис) (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13.06.2025 р.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, 3 додатки має 81 сторінку основного тексту, 38 рисунків та 3 таблиці. Список використаних джерел містить 33 найменування та займає 3 сторінки.

Метою роботи є розробка та апробація програмного модуля для виявлення маркерів компрометації при атаках на вебресурси.

Для досягнення зазначеної мети були визначені наступні завдання:

- дослідити напрямок threat intelligence, його важливість та поширеність;
- ознайомитись з поняттям індикатору компрометації, їх класифікацією;
- провести аналіз функціональних можливостей threat intelligence платформ, порівняти їх між собою, обрати платформу для подальшої роботи та інтеграції з програмним модулем;
- оглянути архітектуру обраної платформи та технології, що в ній використовуються;
- спроєктувати архітектуру та логіку роботи програмного модуля, визначити функції, які він має виконувати;
- реалізувати та протестувати програмний модуль на прикладі різних вебзапитів для розкриття його повного функціоналу.

Об'єктом дослідження є процес виявлення маркерів компрометації при атаках на вебресурси.

Предметом дослідження є методи управління маркерами компрометації та механізм виявлення атак на вебресурси з використанням платформи кіберрозвідки.

Методи дослідження:

- аналіз відкритих джерел;

- огляд технічної документації;
- сигнатурний аналіз;
- моделювання логіки взаємодії компонентів програмного модуля;
- тестування розробленого програмного модуля в реальному середовищі.

Практичною цінністю отриманих результатів є розробка програмного модуля для виявлення маркерів компрометації в роботі вебдодатків.

Актуальність роботи полягає в необхідності своєчасного виявлення кіберзагроз у вебдодатках шляхом аналізу логів та індикаторів компрометації, що забезпечується розробленим модулем із інтеграцією платформи кіберрозвідки.

Ключові слова: індикатор компрометації, проактивний захист, кіберрозвідка, Threat Intelligence, OpenCTI, захист вебдодатків.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕННЬ

IoC	–	Індикатор компрометації
AV	–	Антивірусне програмне забезпечення
EDR	–	Endpoint Detection and Response
CTI	–	Cyber Threat Intelligence, розвідка кіберзагроз
ПЗ	–	Програмне забезпечення
ІБ	–	Інформаційна безпека
IPS	–	Intrusion Prevention System, система запобігання вторгненням
IDS	–	Intrusion Detection System, система виявлення вторгнень
CSIRT	–	Computer Security Incident Response Team, Команда реагування на інциденти комп'ютерної безпеки
CERT	–	Computer Emergency Response Team, команда реагування на комп'ютерні надзвичайні події
TTP	–	Tactics, techniques and procedures; тактики, техніки та процедури

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій вебдодатки стали невід’ємною частиною цифрової інфраструктури. Вони забезпечують безперервну роботу сервісів, електронної комерції, хмарних платформ і управлінських систем як у приватному, так і в державному секторах. У більшості випадків такі додатки розгортаються на Linux-серверах, що забезпечують гнучкість, високу продуктивність та стабільність. Проте з огляду на свою відкриту архітектуру та поширеність, саме вебдодатки на базі операційної системи Linux усе частіше стають об'єктами цілеспрямованих атак.

Сучасні кіберзагрози мають дедалі складнішу структуру: багатоетапні атаки, прихована активність, використання вразливостей нульового дня та комбінування технічних і соціально-інженерних методів – усе це вимагає від захисних механізмів не лише виявлення факту атаки, а й здатності ідентифікувати ознаки загроз на ранніх етапах. Одним із ключових підходів у цьому напрямі є виявлення індикаторів компрометації – цифрових маркерів, які свідчать про присутність або активність зловмисника в системі. До таких індикаторів належать підозрілі IP-адреси, домени, URL, хеші файлів, а також ознаки аномальної поведінки.

Збір, обробка, класифікація та перевірка таких індикаторів у реальному часі є головним завданням напряму розвідки кіберзагроз. Цей напрям охоплює не лише вивчення актуальних атак, а й створення платформ для централізованого управління знаннями про загрози.

Таким чином, ця робота поєднує фундаментальні знання з кібербезпеки, аналізу загроз і архітектури розвідки, із практичною реалізацією інструменту для підвищення безпеки вебдодатків у реальних умовах. Розроблений модуль орієнтовано на потреби як поодиноких користувачів, так і малих організацій, які не мають у своєму розпорядженні повноцінної SOC-інфраструктури, але прагнуть проактивно захищати свої ресурси.

РОЗДІЛ 1
АНАЛІЗ ЗАГРОЗ ТА МЕТОДІВ ВИЯВЛЕННЯ ІНДИКАТОРІВ КОМПРОМЕТАЦІЇ

1.1 Сучасні кіберзагрози: складність та комплексність атак

Сучасні кіберзагрози вирізняються своєю складністю, багатоступеневістю та постійним вдосконаленням. Особливо вразливими до атак є вебсервери, що є основою цифрової інфраструктури в більшості організацій та приватних користувачів. Вебсервери, розгорнуті на Linux-системах із використанням Apache, Nginx чи подібного ПЗ, обробляють величезні обсяги HTTP-трафіку та забезпечують доступ до важливої інформації, що робить їх привабливою мішенню для зловмисників.

Зловмисники все частіше використовують комбінацію технічних засобів, таких як: експлуатація вразливостей у вебдодатках, введення шкідливого коду, атаки на відкриті порти та служби – у поєднанні із соціальною інженерією. Типовими прикладами таких атак є SQL-ін’єкції, XSS, Remote File Inclusion, а також сканування й зловживання неправильно налаштованими сервісами. Метою подібних атак зазвичай є отримання несанкціонованого доступу до системи, збирання конфіденційної інформації або повна компрометація сервера.

Комплексність атак зростає, особливо в контексті тактик, технік і процедур, що швидко адаптуються до нових умов. Багато кібератак починаються з непомітної активності: звернення до нетипових URL, використання нестандартних заголовків запитів, спроб обходу механізмів автентифікації або перевантаження журналів подій. Якщо такі дії не будуть вчасно виявлені, це може призвести до розгортання повномасштабної атаки або утримання постійного доступу з боку зловмисника [1].

Особливо важливим є впровадження засобів, що дозволяють виявляти індикатори компрометації на ранніх етапах. У контексті вебсерверів це можуть

бути підозрілі IP-адреси, спроби звернень до відомих шкідливих доменів, нетипові параметри у HTTP-запитах або файли, що містять шкідливі скрипти. Застосування автоматизованих систем виявлення таких ознак суттєво підвищує рівень безпеки серверного середовища та зменшує ймовірність успішної атаки.

Попри стереотипну думку, що мішенню є лише великі компанії або державні установи, практика свідчить, що навіть приватні вебсервери та невеликі ресурси стають об’єктами масових автоматизованих атак. Зокрема, сканери зловмисників щоденно перевіряють тисячі IP-адрес на наявність відкритих портів, застарілих CMS або некоректних конфігурацій. У багатьох випадках успішна атака стається лише тому, що користувач не використовує базових засобів захисту чи моніторингу.

Таким чином, сучасні вебсервери потребують постійного нагляду за подіями та активністю. Своєчасне виявлення підозрілих дій – це один із найважливіших чинників у забезпеченні кібербезпеки, особливо якщо мова йде про відкриті до інтернету інтерфейси. Використання IoC як засобу виявлення таких дій дозволяє запровадити проактивний підхід до безпеки й зменшити ризики як для великих інфраструктур, так і для поодиноких серверів.

1.2 Основні типи атак вебсервери

Вебсервери, що функціонують у Linux-середовищі, є невід’ємною частиною сучасної цифрової інфраструктури. Вони забезпечують доступ до вебдодатків, API, баз даних і різноманітних сервісів, що використовуються як у корпоративному, так і в індивідуальному секторі. Саме тому зловмисники активно атакують такі сервери, використовуючи широкий спектр методів: від сканування портів до складних багатоетапних атак [2]. Найпоширенішими типами атак на вебсервери Linux є наступні:

1. Атаки на вразливості вебдодатків

Це один із найбільш поширених напрямків атак. Зловмисники використовують недоліки у валідації вхідних даних або помилки в логіці

вебзастосунків, що працюють на сервері [3]. У середовищі Linux найчастіше мова йде про вебдодатки на PHP, Python (Flask/Django), Node.js та інші.

- SQL-ін’єкція – це тип атаки, коли зловмисник вводить шкідливі SQL-запити у поля введення, формуючи запит, який змінює логіку взаємодії з базою даних. Це дозволяє виводити, змінювати або видаляти дані, а іноді й отримати доступ до системи адміністрування.
- Cross-Site Scripting (XSS) – це тип атак, основною метою яких є впровадження шкідливого JavaScript-коду у сторінки сайту. Найчастіше він використовується для крадіжки cookie-файлів, перенаправлення користувачів на шкідливі ресурси або збору конфіденційної інформації.
- Remote File Inclusion (RFI) – це тип атаки, який дозволяє завантажити й виконати на сервері віддалений скрипт. Ця вразливість виникає внаслідок некоректної обробки шляхів у скриптах.
- Command Injection – це тип атаки, коли зловмисник через форму або URL підставляє системні команди, які виконуються на сервері з привілеями користувача вебсервера (наприклад, www-data або nginx).

2. Атаки на конфігурацію та відкриті сервіси

Велика кількість атак відбуваються через неправильне налаштування Linux-сервера, до таких атак відносяться:

- Directory Listing – це тип атаки, при якому зловмисник отримує можливість переглядати вміст серверних директорій у браузері, якщо вебсервер неправильно налаштований і не обмежує індексацію каталогів. У результаті він може ознайомитися зі структурою файлової системи, отримати доступ до конфігураційних файлів, журналів лог-записів, резервних копій або інших потенційно чутливих даних.
- Відкриті панелі керування (phpMyAdmin, Webmin, CPanel) – це прямий шлях до повної компрометації системи при їх доступності ззовні та слабких паролях.

- Відкриті порти (особливо критичними є ті, що надають можливість віддаленого підключення: 21, 22, 23 тощо) при відсутності брандмауерів та захисту можуть стати об’єктом атак грубої сили.
- HTTP Method Exploits – це тип атаки, коли деякі сервери не обмежують методи типу PUT, DELETE, OPTIONS, що може дозволити зловмиснику записувати, модифікувати або видаляти файли прямо на сервері.

3. Атаки грубої сили (bruteforce) та зловживання автентифікацією

Зловмисники часто автоматизують процес підбору облікових даних, використовуючи словники або бази витоків паролів.

- Bruteforce – це масове перебирання логінів та паролів до панелей адміністрування, CMS, SFTP, SSH чи інших сервісів.
- Credential Stuffing включає в себе використання вкрадених пар із попередніх витоків, що часто виявляється ефективним через повторне використання паролів користувачами.
- Атаки на токени доступу, особливо у випадках API, якщо авторизація відбувається без належного контролю часу життя токенів чи обмеження доступу за IP.

4. Впровадження бекдорів (backdoors) і вебшелів (webshells)

Після успішної атаки зловмисники прагнуть зберегти стійкий доступ до системи (тактика Persistence MITRE ATT&CK Map).

- Вебшел (webshell) – це PHP, Python або Perl-файли, які дозволяють виконувати команди через HTTP-запити. Вони маскуються під легітимні файли й дозволяють перегляд вмісту серверу, редагування конфігурацій і запуск додаткових атак.
- Реверсивні шелли (reverse shells) – атаки, які ініціюють з’єднання з контрольованим сервером зловмисника, дозволяючи йому повністю керувати скомпрометованою системою.

5. DDoS-атаки на вебсервер

Навіть якщо сервер на Linux добре захищений, він все одно не застрахований від атак на відмову в обслуговуванні.

- HTTP Flood – це атака, при якій виконується надмірна кількість запитів до головної сторінки, API чи ресурсоємного скрипта.
- Slowloris – це атака, яка тримає з’єднання з сервером відкритими якнайдовше, вичерпуючи перелік доступних чи можливих TCP-з'єднань.
- Amplification Attacks – це тип атаки, коли використанні неправильно налаштованих серверів DNS, NTP, Memcached та інших, зловмисники можуть направити великий обсяг відповідей на IP-адресу жертви.

6. Атаки на логіку обробки HTTP-запитів

Сервери часто не здатні правильно оброблювати на нетипові запити, тобто ті, що не передбачені логікою роботи конкретного вебсервера. Прикладами атак, що націлені на логіку обробки HTTP-запитів, можуть бути наступні:

- Path Traversal – це тип вразливості, який дозволяє отримати доступ до файлів за межами вебдиректорії, тобто це надає зловмиснику отримати доступ до файлів, до яких він не має отримати доступу; зазвичай мова йдеться про критично важливі файли вебсерверу, наприклад, конфігураційні, з обліковими даними тощо. При реалізації атаки до URL дописується запит на перехід з основної директорії до цільового файлу в такому форматі: ../../etc/passwd, ../../.env тощо.
- Header Injection – це тип вразливості, коли спеціальні заголовки HTTP (наприклад, User-Agent, Referer) використовуються для впровадження шкідливого коду або передачі команд.
- Log Poisoning – це вразливість, коли шкідливий код записується в логи (через поля user-agent, cookie тощо), а потім виконується, якщо лог-файли обробляються неналежним чином.

7. Вразливості в серверному ПЗ

Навіть якщо вебдодатки захищені, тобто мають встановлені останні патчі безпеки та належні налаштування, то сам вебсервер (Apache, Nginx) чи компоненти ОС (OpenSSL, systemd, PHP) можуть містити критичні вразливості [4]. Прикладами таких вразливостей для вебсерверів є:

- CVE-2024-38475 – Вразливість у mod_rewrite Apache HTTP Server.

Суть вразливості полягає в тому, що через неправильне екранування вихідних даних у модулі mod_rewrite зловмисник отримує можливість зіставити URL-адреси з файловими системами, які не призначені для прямого доступу, що може призвести до виконання коду або розкриття вихідного коду. Вразливими версіями є Apache HTTP Server версії 2.4.59 та раніше.

- CVE-2023-44487 – HTTP/2 Rapid Reset Attack. Це вразливість у реалізації протоколу HTTP/2, що дозволяє зловмиснику створювати додаткове навантаження через швидке створення та скасування потоків, що, у свою чергу, може призвести до відмови в обслуговуванні. Вразливими є вебсервери з підтримкою HTTP/2 без відповідних патчів.

Розуміння типів атак на вебсервери Linux є критично важливим для розробки ефективних рішень із захисту, враховуючи подальше написання програмного модулю для виявлення вебатак за шаблонами. Оскільки більшість атак залишають цифрові сліди, вони можуть бути виявлені через індикатори компрометації. Саме на цьому принципі і базується підхід до створення автоматизованого модуля виявлення загроз, що стане предметом практичної реалізації в цій роботі.

1.3 Розвиток та поширеність напрямку Threat Intelligence

У сьогоденні, коли кіберзагрози стають дедалі складнішими та частішими, напрямок Threat Intelligence (розвідка кіберзагроз) відіграє критично важливу роль у забезпеченні інформаційної безпеки організацій. Він охоплює процеси збору, аналізу та інтерпретації даних про потенційні або активні загрози з метою їх попередження та ефективного реагування [5]. Ключовими аспектами Threat Intelligence є наступні:

1. Збір та аналіз даних, що включає в себе використання різноманітних джерел, таких як файли логів, мережевий трафік та зовнішні інформаційні канали, за допомогою яких фахівці з кіберрозвідки ідентифікують

індикатори компрометації та моделі поведінки зловмисників, що у свою чергу, дозволяє виявляти та нейтралізувати загрози на ранніх етапах.

2. Проактивний захист є методом, коли замість реагування на вже здійснені атаки, засобами Threat Intelligence можна передбачати можливі вектори атак та вживати заходів для їх запобігання. Такий підхід значно знижує ризик успішних атак та мінімізує потенційні збитки.

3. Інтеграція з існуючими системами безпеки надає можливість інтеграції розвідувальних даних в системи управління інформаційною безпекою (SIEM), рішення для виявлення та реагування на загрози на кінцевих точках (EDR) та інші інструменти, що підвищує їх ефективність.

Зі зростанням кількості та складності кіберзагроз, попит на фахівців з Threat Intelligence стрімко зростає. Організації усвідомлюють необхідність мати в своєму штаті експертів, здатних аналізувати та інтерпретувати інформацію про загрози для захисту критично важливих активів [6].

Для підтвердження кваліфікації та компетентності саме в цьому напрямку існує низка сертифікацій [7]:

- Certified Threat Intelligence Analyst (CTIA) – це програма від EC-Council, яка навчає методам збору та аналізу розвідувальних даних, а також їх застосуванню для виявлення та нейтралізації загроз.
- GIAC Cyber Threat Intelligence (GCTI) – це сертифікація від GIAC, що підтверджує знання та навички в області стратегічної, оперативної та тактичної кіберрозвідки.
- CREST Registered Threat Intelligence Analyst (CRTIA) – це окрема сертифікація від CREST, орієнтована на фахівців, які надають послуги з розвідки загроз.
- CREST Practitioner Threat Intelligence Analyst (CPTIA) – це сертифікація початкового рівня для тих, хто прагне розпочати вдосконалювати знання в сфері розвідки загроз.

Виходячи з тих умов, які наразі диктують зловмисники та сучасні кіберзагрози, Threat Intelligence є невід’ємною складовою сучасної стратегії

кібербезпеки. Цей напрям забезпечує можливість не лише реагувати на інциденти, але й проактивно запобігати їм, що критично важливо в умовах зростаючих кіберзагроз. Інвестування в розвиток цього напряму та підвищення кваліфікації фахівців сприяє зміцненню загальної безпеки та стійкості організації до потенційних атак.

1.4 Індикатори компрометації та їх класифікація

Індикатори компрометації – це артефакти, які вказують на можливу несанкціоновану активність в інформаційній системі. Вони є важливим елементом сучасної кібербезпеки та застосовуються в процесах виявлення, аналізу і реагування на інциденти безпеки. Виявлення та аналіз IoC дозволяє ефективніше виявляти шкідливу активність, навіть якщо атака є складною, цілеспрямованою чи тривалою за виконанням [8].

IoC можуть бути отримані з різних джерел: систем виявлення вторгнень, AV програм, систем моніторингу трафіку, журналів подій, аналізу зразків шкідливого ПЗ, а також з платформ обміну кіберзагрозами (наприклад, MISP, VirusTotal, ThreatFox тощо).

IoC мають різну природу, тому умовно їх поділяють на кілька основних типів за джерелом походження. Основними категоріями є мережеві, файлові та поведінкові індикатори [9].

1. Мережеві IoC описують аномальну або підозрілу мережеву активність, яка може свідчити про спробу проникнення, з’єднання з C2-серверами (command&control), розповсюдження шкідливого коду тощо. Такі індикатори є одними з найпоширеніших і легко піддаються автоматизованому аналізу. До них відносяться:

- IP-адреси зловмисників, які були помічені в атаках, фішингу або пов’язані з шкідливим ПЗ, також часто вони використовуються для створення фільтрів та чорних списків в організаціях.

- Домени. Зловмисники часто використовують доменні імена для маскування своїх серверів, зокрема фішингових сайтів, C2-серверів або для доставки шкідливого ПЗ.
- URL-адреси. Конкретні вебадреси, які використовуються для завантаження шкідливого коду або перенаправлення користувача на фішингові сайти, а також для експлуатації різних вразливостей вебсервера, наприклад, Path Traversal, RFI/LFI та інші.

Використання мережевих IoC дозволяє налаштовувати засоби захисту (IDS/IPS, брандмауери, проксі, AV, EDR) для блокування підозрілих з'єднань та запитів.

2. Файлові IoC вказують на шкідливі або модифіковані файли, що виявлені в системі. Вони застосовуються в антивірусних рішеннях, системах контролю цілісності файлів (FIM), EDR-рішеннях тощо. Основні файлові IoC включають:

- Хеші файлів – це криптографічні підписи (наприклад, MD5, SHA-1, SHA-256), які дозволяють ідентифікувати конкретний файл. Якщо хеш збігається з відомим зразком шкідливого ПЗ, то файл вважається небезпечним.
- Сигнатури шкідливого ПЗ – це бітові шаблони, які антивірусні системи використовують для розпізнавання шкідливого коду. Це може бути набір байтів, що зустрічається лише у відомих вірусах або троянах. Найчастіше вони використовуються для визначення в антивірусних механізмах або YARA-правилах [10].
- Метадані файлів, це включає в себе, наприклад, незвичні шляхи розміщення файлів, час створення, автора або цифрові підписи, тобто все, що може бути ознакою компрометації.

3. Поведінкові IoC вказують на аномальну або підозрілу поведінку в системі або мережі. Вони є більш динамічними, ніж інші, за своєю природою та застосовуються в контексті поведінкового аналізу, машинного навчання та систем виявлення загроз нового покоління. До них належать:

- Аномальна активність користувачів або процесів, що включає в себе, наприклад, входи в систему у незвичний, неробочий час або з нетипових локацій, зміна паролів без причини, запуск скриптів користувачами, які не мають на це відповідних причин, масова кількість новостворених облікових записів, тобто все, що не є нормальною або є підозрілою поведінкою в системі.
- Нетипові запити або обмін даними, тобто, наприклад, велика кількість вихідного трафіку до зовнішніх IP-адрес, підключення до незвичних доменів або використання нестандартних портів.
- Підозрілі зміни у конфігурації, наприклад, раптове вимкнення журналювання, зміна правил брандмауера або відключення антивірусу.

Поведінкові IoC є особливо цінними в умовах, коли атака є новою або використовує техніки обходу традиційних засобів виявлення, оскільки вони орієнтуються не на підписи, а на виявлення відхилень від нормального функціонування системи (baseline). Такі IoC є ефективними проти невідомих або обфускованих загроз, де немає сигнатур.

Усі зазначені типи індикаторів часто використовуються спільно, тобто, наприклад, мережева активність може бути пов’язана з конкретним шкідливим файлом, а файл – із підозрілою поведінкою процесу. Ефективна система моніторингу подій та реагування на інциденти повинна враховувати всі класи IoC для повного охоплення потенційних загроз. Комбіноване використання мережевих, файлових і поведінкових IoC дозволяє створити цілісний підхід до виявлення загроз.

Інформацію про джерела виявлення та застосування індикаторів компрометації наведено в таблиці 1.1.

Таблиця 1.1

Класифікація IOC

Тип IoC	**Приклад IoC**	**Джерело виявлення**	**Призначення/ застосування**

Мережевий	IP-адреса:	Firewall , IDS та IPS, SIEM та TI Feeds	Виявлення з'єднань до шкідливих або

Продовження таблиці 1.1

	117[.]209[.]88[.]219; Домен: s13[.]cnzz[.]com; URL: hxxps[://]check[.]boguj[.]icu/gkcxv[.]google		нетипових ресурсів, фільтрація трафіку
Файловий	Хеші файлів: fec61…ed1f0c4bb78, YARA-сигнатури, модифіковані системні файли	AV-рішення, системи EDR, а також для аналізу зразків шкідливого ПЗ	Ідентифікація шкідливих або модифікованих файлів, модуль FIM
Поведінковий	Аномальна активність користувачів, незвичні мережеві з’єднання, процеси з нетиповими параметрами	UEBA (User Entity Behavior Analytics), SIEM, події з розслідувань SOC	Виявлення нових, обфускованих чи інших комплексних атак

1.5 Методи передачі інформації про індикатори компрометації

Передача інформації про індикатори компрометації є ключовою складовою сучасних систем кібербезпеки. Для цього використовуються стандартизовані формати та протоколи, що забезпечують однорідність, точність та автоматизацію обміну даними між різними організаціями та системами [11]. Серед найбільш поширених рішень слід виокремити наступні:

1. STIX (Structured Threat Information eXpression) – це структурована мова для опису інформації про кіберзагрози, що дозволяє організаціям стандартизувати та автоматизувати обмін даними про загрози. Вона охоплює різні аспекти загроз, включаючи індикатори, TTP атакуючих, інциденти та інше. STIX сприяє уніфікації представлення інформації, що полегшує її аналіз та обмін між різними системами [12].

2. TAXII (Trusted Automated eXchange of Indicator Information) – це протокол, що визначає, як інформація про загрози, що представлена у форматі STIX, може бути автоматично передана між організаціями. Він забезпечує стандартизовані сервіси та обмінні механізми для спільного використання даних про загрози, підтримуючи різні моделі обміну, такі як “hub and spoke”, “peer-to-peer” та “клієнт-сервер”.

3. OpenIOC (Open Indicators of Compromise) – це відкритий формат, розроблений компанією Mandiant, для опису технічних характеристик, що ідентифікують відомі загрози або методології атакуючих [13]. Він використовує XML-схему для представлення індикаторів компрометації та дозволяє організаціям створювати, обмінюватися та використовувати ці індикатори для виявлення загроз у своїх системах.

4. CybOX (Cyber Observable eXpression) – це мова для опису подій або властивостей, що пов’язані зі зловмисними діями в кіберпросторі. Вона дозволяє стандартизувати представлення даних про спостережувані об'єкти, такі як файли, ключі реєстру, мережеві з'єднання тощо. CybOX часто використовується у поєднанні з іншими стандартами, такими як STIX, для детального опису технічних аспектів загроз.

5. IODEF (Incident Object Description Exchange Format) – це формат, визначений IETF у RFC 5070, призначений для обміну інформацією про комп'ютерні інциденти між командами реагування на інциденти. Він надає структуру для опису деталей інцидентів, включаючи час виявлення, методи атаки, оцінку впливу та іншу релевантну інформацію.

6. MAEC (Malware Attribute Enumeration and Characterization) – це стандарт для опису характеристик та поведінки шкідливого програмного забезпечення. Він дозволяє детально документувати атрибути та дії зловмисного ПЗ, що сприяє обміну інформацією між дослідниками та автоматизації процесів аналізу шкідливих програм.

На сьогодні це найпоширеніші формати та протоколи передачі інформації про IoC, що сприяють ефективному обміну інформацією про загрози між організаціями, тим самим забезпечуючи підвищення рівня кібербезпеки та швидкого реагування на інциденти. Вони є найбільш поширеними та загальновизнаними методами представлення й передачі інформації про індикатори компрометації.

Вони широко використовуються в платформах Threat Intelligence, антивірусних рішеннях та SIEM-системах, інструментах зворотного аналізу шкідливого ПЗ, CSIRT/CERT-командах, інтеграціях між організаціями та державними структурами.

Причиною їх популярності та загального визнання є наступні критерії:

- STIX/TAXII – це де-факто стандарт, що підтримується MITRE та OASIS, рекомендований у США та ЄС.
- IODEF використовується для стандартизованого обміну інцидентами через CERT/CSIRT.
- MAEC/CybOX – це спеціалізовані формати, які часто застосовуються в інструментах з аналізу зразків шкідливого ПЗ.

1.6 Джерела отримання індикаторів компрометації

Одним із ключових елементів у виявленні, реагуванні та попередженні кіберзагроз є своєчасне отримання та актуалізація індикаторів компрометації – технічних артефактів, які свідчать про можливу або фактичну присутність шкідливої активності в інформаційно-комунікаційній системі.

Для збору, обробки, структуризації та обміну цими даними активно використовуються платформи оцінки загроз – Threat Intelligence платформи. Вони забезпечують централізований доступ до джерел IoC, підтримують стандарти взаємодії та можуть бути інтегровані з рішеннями IDS/IPS, SIEM-системами, AV продуктами, рішеннями класу EDR/XDR та іншими рішеннями безпеки.

Threat Intelligence платформи – це спеціалізовані середовища для агрегації, обробки та розповсюдження даних про кіберзагрози, включаючи індикатори компрометації. Вони можуть бути як комерційними (наприклад, Anomali, Recorded Future, ThreatConnect), так і з відкритим кодом (OpenCTI, MISP, ThreatFox). Такі платформи підтримують імпорт та експорт IoC у стандартизованих форматах (STIX, TAXII), автоматичне оновлення, API для взаємодії з іншими системами та зручні інтерфейси для аналітичного опрацювання інформації. Важливою особливістю є можливість побудови зв’язків між об'єктами, наприклад, файл – хеш – домен – кампанія – техніка атаки, що дає змогу формувати повноцінну картину загрози.

1. OpenCTI (Open Cyber Threat Intelligence) – це платформа оцінки кіберзагроз з відкритим вихідним кодом, призначена для управління даними про кіберзагрози. Вона поєднує концепції з Threat Intelligence, Digital Forensics та Threat Hunting у єдиній системі. Платформа базується на об'єктно-орієнтованому підході, що включає в себе, що кожен індикатор, кампанія, зловмисник, вектор атаки або техніка MITRE ATT&CK подається як окремий об’єкт, що дозволяє будувати взаємозв’язки між ними.

OpenCTI підтримує стандарт STIX 2.1, що забезпечує сумісність із більшістю сучасних інструментів. Система дозволяє імпортувати дані з різних джерел кіберрозвідки таких, як MISP, Shodan, VirusTotal, та експортувати IoC у формати JSON та STIX. Вбудовані графічні інтерфейси дозволяють візуалізувати ланцюжки атак, розгортання шкідливих кампаній, та досліджувати залежності між індикаторами. За допомогою GraphQL API платформа може бути інтегрована в робочі процеси аналітиків SOC, CERT- та IR-команд.

Надання інформації про IoC на платформі OpenCTI представлено на рисунку 1.1.

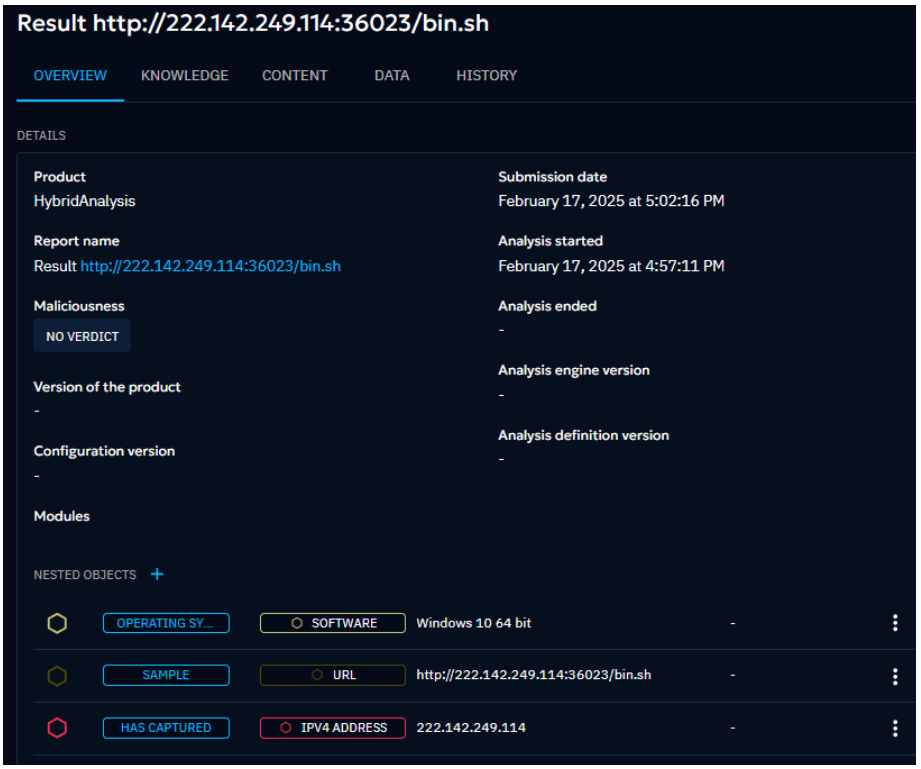


Рисунок 1.1 – Подача інформації про індикатор компрометації на платформі OpenCTI

2. MISP (Malware Information Sharing Platform & Threat Sharing) – одна з найбільш розповсюджених opensource платформ для обміну IoC, яка підтримується активною спільнотою експертів з кібербезпеки. Основна мета платформи полягає в полегшенні обміну структурованою інформацією про кіберінциденти між організаціями, командними реагуваннями (CSIRT/CERT), державними органами та приватними компаніями. MISP дозволяє створювати події, кожна з яких може містити набір атрибутів: IP-адреси, домени, хеші файлів, URL, email, сертифікати та іншу інформацію. Усі атрибути

супроводжуються тегами, рівнем достовірності, часом виявлення, категорією та контекстом загрози.

Платформа також підтримує обмін даними через API та TAXII-сервіси, що дає змогу автоматизувати збір IoC до SIEM-систем або рішень IDS та IPS. Важливою функціональністю є підтримка таксономій (наприклад, MISP Galaxy), що дозволяє класифікувати загрози за типами шкідливого ПЗ, зловмисників, груп тощо [14]. Надання інформації про IoC на платформі MISP представлено на рисунку 1.2.

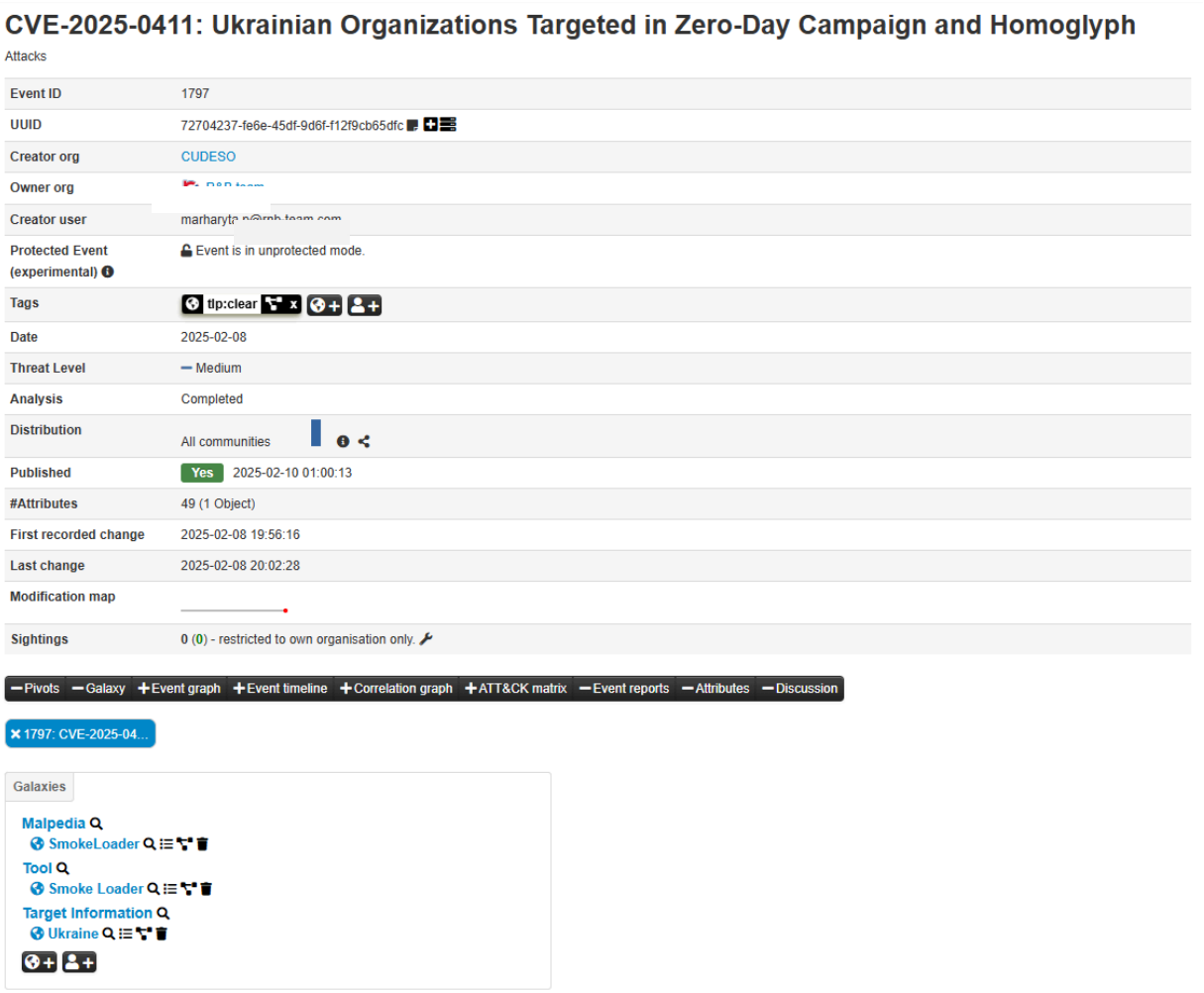


Рисунок 1.2 – Подача інформації про індикатор компрометації на платформі MISP

3. ThreatFox – це спеціалізований сервіс від організації Abuse.ch, орієнтований на швидке розповсюдження актуальних індикаторів компрометації, пов’язаних із шкідливою активністю (зокрема malware, C2-серверами, phishing-ресурсами тощо). Платформа надає відкритий доступ до

своєї бази IoC, яка постійно оновлюється фахівцями з кіберзахисту та волонтерами.

Індикатори доступні у вигляді хешів файлів, IP-адрес, URL, доменів, а також містять супровідні атрибути, такі як: тип загрози, опис, дату виявлення, рівень достовірності. ThreatFox підтримує формат JSON, CSV, та має REST API, що дозволяє використовувати дані в автоматизованих системах моніторингу та реагування. Через свою оперативність і відкритість ThreatFox активно використовується як джерело TI для SIEM-систем, IDS та IPS рішень, firewall та аналітичних систем. Надання інформації про IoC на платформі ThreatFox представлено на рисунку 1.3.

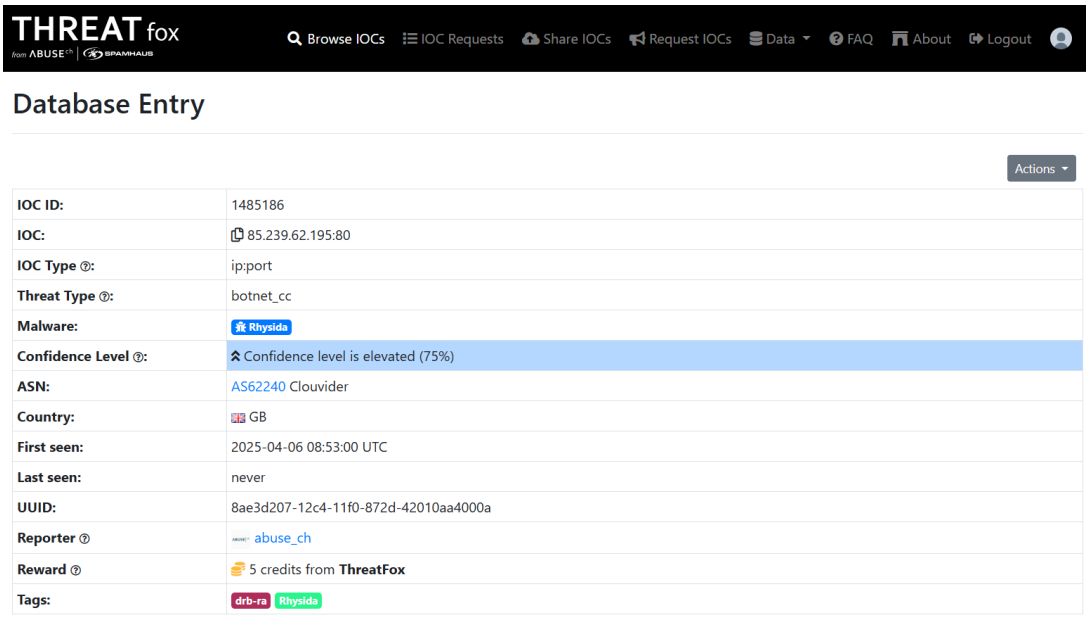


Рисунок 1.3 – Подача інформації про індикатор компрометації на платформі ThreatFox

Проаналізувавши інформацію як про відкриті, так і про комерційні рішення платформ кіберрозвідки, було наведено порівняльні характеристики TIP в таблиці (див. табл. А).

Таким чином, проаналізувавши функціональні можливості та характеристики різноманітних сучасних платформ для кіберрозвідки загроз, для подальшого дослідження та розробки програмного модуля для виявлення індикаторів компрометації в атаках на вебресурси було обрано OpenCTI завдяки

його архітектурі, гнучкості та розширюваності, що робить його ідеальною основою для інтеграції з власним програмним модулем [15]. На відміну від MISP, що орієнтований переважно на спільне використання та структурування індикаторів компрометації, OpenCTI забезпечує об’єктно-орієнтовану модель даних на основі STIX 2.1 та підтримує глибоку аналітику зв’язків між об’єктами (наприклад, техніками, групами, кампаніями), що критично важливо для побудови контексту навколо загроз.

Крім того, OpenCTI пропонує розвинене GraphQL API, що дозволяє здійснювати складні вибірки з бази даних у зручному форматі, що ідеально підходить для автоматизованої взаємодії програмного модуля з платформою. У порівнянні з ThreatFox, який здебільшого надає списки шкідливих об’єктів у вигляді потоків даних, OpenCTI дозволяє гнучко інтегрувати сторонні джерела через систему з’єднувачів, підтримує багатопоточну обробку, модульність, і має сучасний web-інтерфейс для подальшого аналізу результатів.

Саме такі переваги, як масштабованість, структурованість та можливість повноцінної двосторонньої інтеграції – визначили вибір OpenCTI як платформи кіберрозвідки для реалізації системи виявлення індикаторів компрометації в логах вебдодатків.

Висновки до розділу 1

У першому розділі було розглянуто основи напрямку Threat Intelligence, який є важливим компонентом сучасної кібербезпеки, оскільки його актуальність зростає прямопропорційно з розробкою нових технік та тактик атак зловмисниками. Досліджено, що цей напрямок забезпечує можливість своєчасного виявлення, аналізу та попередження кібератак завдяки використанню структурованих даних про загрози, що є особливо важливим в умовах зростаючої складності загроз та високої динаміки змін у середовищі інформаційної безпеки.

Також було розглянуто поняття індикаторів компрометації, їх класифікацію та джерела виявлення, їх роль в напрямку Threat Intelligence та проактивному захисті систем від зловмисних дій.

Окрему увагу було приділено методам обміну індикаторами компрометації, серед яких найбільш поширеними є формати STIX, TAXII, OpenIOC, CybOX та інші. Таким чином було досліджено, що використання уніфікованих стандартів сприяє швидкому, масштабованому та точному обміну розвідувальною інформацією між організаціями, урядовими структурами та постачальниками послуг безпеки.

Розгляд розвитку галузі показав, що Threat Intelligence є одним із актуальних і динамічно зростаючих напрямів, з розвиненою інфраструктурою навчання, професійної сертифікації та великою кількістю дослідницьких і практичних ініціатив. Саме тому знання у цій сфері вважаються критично важливими для фахівця з кібербезпеки, незалежно від специфіки його діяльності чи робота у SOC-команді, CERT-групі, чи впровадження стратегічної безпеки на рівні підприємства.

Огляд існуючих платформ для роботи з розвідкою кіберзагроз (зокрема MISP, OpenCTI та інших) дозволив порівняти їх функціональні можливості, формати підтримуваних даних, способи інтеграції та гнучкість в адаптації до потреб організацій. Порівняльний аналіз показав, що різні системи мають свої сильні сторони залежно від контексту використання – наприклад, відкриті рішення можуть бути більш гнучкими, а комерційні – більш комплексними у функціоналі. В результаті порівняння функціональних характеристик та можливостей було обрано платформу кіберрозвідки для подальшого використання та інтеграції з програмним модулем для виявлення індикаторів компрометації в атаках на вебресурси.

Таким чином, цей розділ закладає аналітичну та технічну основу для подальшого дослідження в межах кваліфікаційної роботи. Він формує розуміння важливості роботи з розвідкою загроз як з точки зору інфраструктури, так і з точки зору практичного використання технологій.

Надалі це дозволить більш глибоко дослідити методи реалізації захисту, оцінки ризиків, виявлення вторгнень та впровадження проактивних заходів у кіберзахисті.

РОЗДІЛ 2
ПЛАТФОРМА OPENCTI ЯК ІНСТРУМЕНТ РОЗВІДКИ КІБЕРЗАГРОЗ

2.1 Загальна характеристика платформи OpenCTI

OpenCTI (Open Cyber Threat Intelligence) – це платформа з відкритим вихідним кодом, яка забезпечує централізовану систему управління, обробки, кореляції та візуалізації даних кіберрозвідки. Її основна ідея полягає у створенні уніфікованого середовища, де структуровані дані про кіберзагрози можуть зберігатися, аналізуватися та інтегруватися з іншими інформаційними системами безпеки організації. OpenCTI дозволяє фахівцям з кібербезпеки не просто реєструвати індикатори компрометації, а працювати з повним контекстом загроз, включно з тактиками, техніками, суб'єктами загроз, кампаніями, інцидентами, цільовими організаціями тощо.

Основні цілі платформи є наступними:

- Акумулювання інформації про відомі та нові загрози у структурованому вигляді.
- Побудова взаємозв'язків між різними об'єктами загроз, від IP-адрес і доменів до груп зловмисників, інструментів і TTP.
- Інтеграція з різними рішеннями безпеки, такими як SIEM, SOAR, EDR, Threat Intelligence Feeds.
- Автоматизація аналітичної роботи над кіберзагрозами за рахунок інтеграцій, API та стандартів обміну.
- Обмін інформацією в межах організації або між різними організаціями чи партнерами.

В цілому, платформа OpenCTI заснована на ідеї використання онтології загроз, тобто всі об'єкти, які створюються у системі (індикатори, TTPs, групи зловмисників, інциденти тощо), пов'язані між собою, утворюючи цілісну

картину загроз. Завдяки цьому платформа не лише збирає конкретні факти, а й в цілісності дозволяє бачити повну картину розвитку атак і логіку дій атакуючих.

Сама платформа була створена та підтримується французькою компанією Filigran, що спеціалізується на рішеннях у сфері Threat Intelligence, а також за участі низки партнерів, серед яких ANSSI (Агентство з кібербезпеки Франції), CERT-EU, Thales, SEKOIA.IO, Maltego та інші [16].

OpenCTI – це повністю opensource проєкт, доступний на GitHub, що, відповідно, дозволяє:

- Безкоштовно використовувати його у будь-яких організаціях (від приватного бізнесу до державних установ).
- Вільно модифікувати функціонал та інтегрувати його з іншими системами.
- Спільно розвивати платформу, завдяки чому користувачі можуть брати участь у внесенні змін, звітуванні про помилки, створенні плагінів та з’єднувачів.

Ліцензія OpenCTI GNU Affero General Public License v3 (AGPLv3) гарантує, що навіть при модифікації та використанні в корпоративному середовищі код залишається відкритим [17].

Також існує активна спільнота користувачів: від окремих фахівців із кіберзахисту до великих міжнародних корпорацій. Спільнота підтримує обмін знаннями, надає рекомендації, ділиться готовими інтеграціями, а також обговорює нові загрози та підходи до розвідки.

Як вже було зазначено, сучасна кіберрозвідка все частіше стикається з проблемами надлишку даних та відсутності контексту. Це виникає через те, що традиційні платформи TI часто просто генерують мільйони IoC без опису контексту: «хто», «навіщо» і «як». OpenCTI вирішує цю проблему, надаючи контекстуалізовану Threat Intelligence інформацію, де зв’язки між даними мають ключове значення.

Завдяки підтримці стандартів STIX 2.1, TAXII 2.0, GraphQL API, OpenCTI забезпечує такі вимоги:

- сумісність з іншими системами CTI, такими як MISP, VirusTotal, Shodan, TheHive, Maltego, Elastic та іншими.
- Є масштабованим рішенням для SOC, CERT, приватних компаній, які мають потребу в управлінні інформацією про загрози.
- Використовується в автоматизованих сценаріях аналізу та реагування, особливо у поєднанні з SOAR-системами (наприклад, TheHive або Cortex XSOAR).

Таким чином, OpenCTI перетворюється з простої бази знань у центральний інтелектуальний хаб, що забезпечує ефективний аналіз, кореляцію, розподіл та інтерпретацію даних про загрози. У час, коли інформація має вирішальне значення, OpenCTI дозволяє організаціям перетворити «сирі» дані на цінні аналітичні знання.

2.2 Архітектура OpenCTI

Розуміння архітектури OpenCTI є ключовим для побудови ефективної взаємодії програмного модуля з платформою розвідки загроз. Кожен компонент цієї системи виконує критичну роль у процесі прийому, обробки, аналізу та зберігання інформації про потенційні компрометації. Завдяки глибокому аналізу внутрішніх механізмів можливо забезпечити коректну інтеграцію та передати дані у форматі, придатному для подальшої обробки та аналітики.

Використані технології забезпечують гнучкість, масштабованість та швидкість, необхідні для реагування на кіберзагрози в реальному часі. Саме ці аспекти лягли в основу логіки проєктування програмного модуля, що аналізує веблоги та автоматично передає виявлені потенційно шкідливі індикатори до платформи. Технічна обґрунтованість кожного елемента OpenCTI дозволяє не лише спростити реалізацію модулю, а й підвищити його надійність, точність і адаптивність у реальних умовах.

2.2.1 Основні компоненти платформи OpenCTI

Платформа OpenCTI складається з кількох ключових компонентів, що забезпечують її ефективну роботу, таких як:

- Backend. Серверна частина OpenCTI відповідає за обробку логіки та управління даними. Вона реалізована на Node.js, що забезпечує асинхронну обробку запитів та високу продуктивність. Backend включає GraphQL API, яке дозволяє клієнтським додаткам запитувати та отримувати дані з платформи. Це забезпечує гнучкість та ефективність у взаємодії з даними.
- Frontend. Клієнтська частина є інтерфейсом користувача, розробленим на React. Він забезпечує зручний доступ до даних, що зберігаються в платформі, та надає інструменти для їх візуалізації та аналізу. Користувачі можуть взаємодіяти з даними через вебінтерфейс, виконувати пошук, переглядати звіти та інші аналітичні матеріали. React забезпечує швидку та динамічну роботу інтерфейсу.
- Worker’и (обробники) – це процеси на Python, які асинхронно обробляють повідомлення STIX 2.1, отримані з черг повідомлень. Вони відповідають за створення відповідних елементів у платформі OpenCTI через GraphQL API. Кількість worker’ів можна масштабувати залежно від обсягу даних та вимог до продуктивності системи. Це дозволяє ефективно обробляти великі обсяги даних та забезпечувати швидку реакцію на нові загрози [18].
- З’єднувачі (connectors) є окремими службами, які працюють з платформою OpenCTI та можуть бути реалізовані практично на будь-якій мові програмування, яка підтримує STIX2. Вони використовуються для розширення функціональності OpenCTI, дозволяючи інтегрувати зовнішні джерела даних, обробляти їх та передавати в платформу. Наприклад, з'єднувачі можуть імпортувати дані з різних платформ розвідки загроз (AlienVault, Virustotal, AbuseCh тощо), виконувати збагачення даних або експортувати їх до інших систем. Це забезпечує гнучкість та масштабованість платформи, дозволяючи адаптувати її під різні потреби [19].

2.2.2 Основні технології платформи OpenCTI

Платформа OpenCTI використовує низку передових технологій для забезпечення ефективного збору, обробки та аналізу даних про кіберзагрози. Основні з них наступні:

1. GraphQL – це мова запитів для API, яка дозволяє клієнтам запитувати лише необхідні дані, що зменшує обсяг переданої інформації та підвищує ефективність взаємодії між клієнтом і сервером. В OpenCTI GraphQL використовується для забезпечення гнучкого та ефективного доступу до даних, дозволяючи формулювати складні запити та отримувати точні відповіді. Це сприяє оптимізації роботи з великими обсягами інформації про загрози.

2. Elasticsearch – це розподілена пошукова та аналітична система, яка дозволяє швидко індексувати та шукати великі обсяги даних у реальному часі. В OpenCTI Elasticsearch використовується для зберігання та швидкого пошуку індикаторів загроз, що забезпечує оперативний доступ до критично важливої інформації. Завдяки цьому користувачі можуть ефективно аналізувати та виявляти потенційні загрози в режимі реального часу [20].

3. RabbitMQ – це брокер повідомлень, який підтримує різні протоколи обміну повідомленнями. В реалізації платформи RabbitMQ використовується для асинхронної обробки завдань, таких як обробка даних із з’єднувачів або виконання фонових задач. Це забезпечує масштабованість та надійність системи, дозволяючи ефективно розподіляти навантаження між компонентами та обробляти великі обсяги даних без затримок.

4. Redis – це високопродуктивне сховище структурованих даних у пам'яті, яке підтримує різні типи даних, такі як рядки, списки, множини тощо. В контексті платформи OpenCTI Redis використовується для кешування та управління чергами завдань, що дозволяє зменшити навантаження на основні бази даних та покращити продуктивність системи. Зокрема, Redis використовується для зберігання сесій користувачів, що забезпечує швидкий доступ та високу швидкодію при обробці запитів [21].

5. MinIO – це високопродуктивне об'єктне сховище, сумісне з API Amazon S3, яке дозволяє зберігати великі обсяги неструктурованих даних, таких як файли, зображення, резервні копії тощо. MinIO використовується для зберігання великих обсягів даних, забезпечуючи масштабованість та ефективність зберігання. Це дозволяє системі обробляти та зберігати значні обсяги інформації про загрози, забезпечуючи швидкий доступ та надійність зберігання.

Інтеграція всіх цих технологій у архітектуру OpenCTI дозволяє створити потужну, гнучку та масштабовану платформу, яка відповідає сучасним вимогам до обробки та аналізу великих обсягів даних у реальному часі.

Підсумовуючи підрозділи 2.2.1 та 2.2.2, було розроблено схему, що демонструє взаємодію основних компонентів та технологій, що лежать в основі реалізації платформи кіберрозвідки OpenCTI (рис. 2.1).

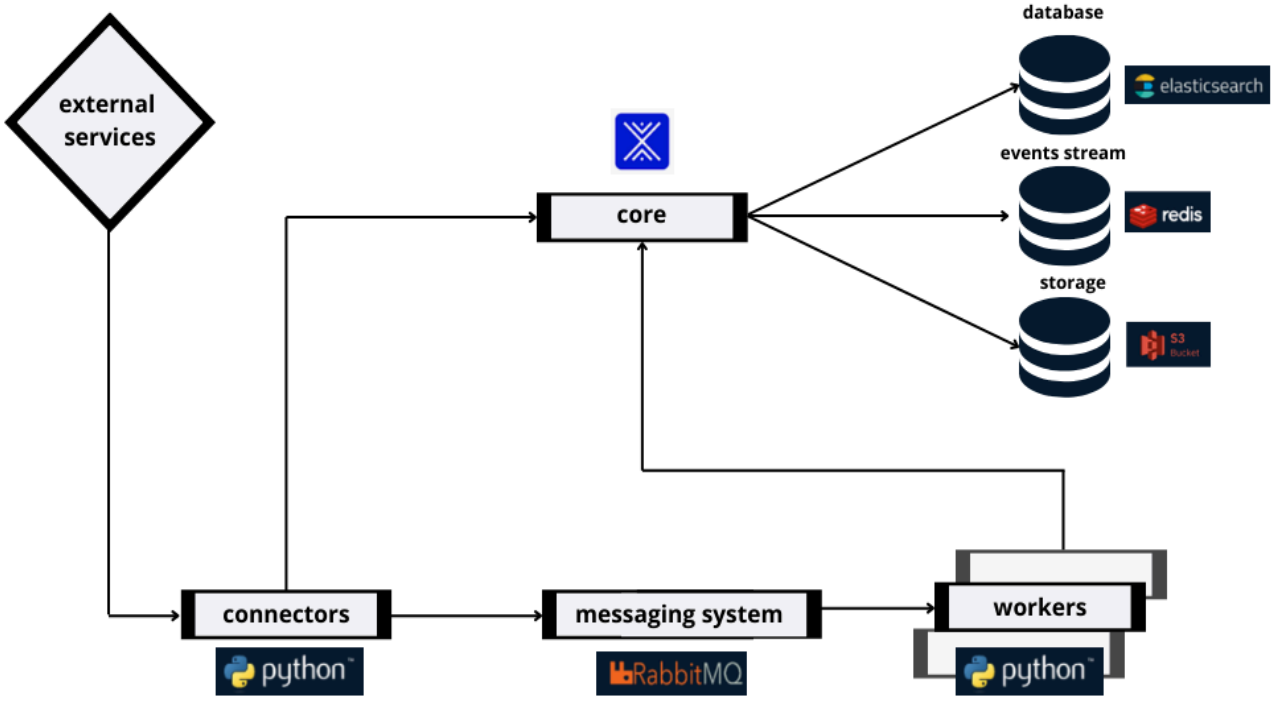


Рисунок 2.1 – Основні компоненти та технології, що забезпечують архітектуру платформи OpenCTI

2.2.3 Взаємодія між компонентами та розподіл завдань між модулями

Ефективність роботи платформи OpenCTI, яка побудована за модульною архітектурою, забезпечується злагодженою взаємодією її компонентів, кожен з яких виконує чітко визначені функції в межах загальної архітектури. Комунікація між модулями реалізується переважно через GraphQL API, RabbitMQ, Redis, а також безпосередню взаємодію зі сховищами даних (Elasticsearch, MinIO) [22].

Описати взаємодію компонентів та технологій в архітектурі платформи, можна розглянувши наступні логічні пари:

1. Frontend – Backend (серверна частина).

Користувачі взаємодіють із платформою через інтерфейс, реалізований на React. Усі запити на отримання, створення чи зміну інформації направляються до серверної частини через GraphQL API, яка, у свою чергу, відповідає за обробку запитів, валідацію даних та комунікацію з базами даних.

2. Backend (серверна частина) – Elasticsearch / Redis / MinIO.

Backend безпосередньо звертається до Elasticsearch для збереження та пошуку об’єктів. Redis використовується для кешування даних і управління сесіями користувачів. Для зберігання великих обсягів файлів, таких як вкладення до інцидентів чи звітів, використовується MinIO.

3. З’єднувачі – RabbitMQ – Backend (серверна частина).

З’єднувач отримує дані з зовнішніх джерел (наприклад, Virustotal, MISP та інші джерела IoC), обробляє їх, і надсилає у вигляді повідомлень до RabbitMQ. Backend асинхронно отримує ці повідомлення з черги, обробляє STIX-структуровані об'єкти та додає їх до платформи через API.

4. Worker’и – RabbitMQ – Backend.

Worker-и «підписані» на певні черги RabbitMQ. Вони автоматично обробляють повідомлення, наприклад, створення нових об'єктів, оновлення або видалення. Використовуючи GraphQL API, вони взаємодіють з Backend для внесення змін у базу даних OpenCTI.

5. З’єднувачі – GraphQL API

Деякі з’єднувачі можуть не використовувати RabbitMQ напряму, а надсилати оброблені дані одразу до Backend через GraphQL API. Такий підхід характерний для імпортерів з повільною або періодичною обробкою даних.

Всю інформацію про розподіл завдань між модулями зведено в таблиці 2.1, що надає розуміння принципу роботи платформи та підґрунтя для подальшого написання програмного модуля, особливо, в частині побудови запиту до платформи OpenCTI.

Таблиця 2.1

Розподіл завдань між модулями

Компонент	**Основні завдання**
Frontend	● Надання інтерфейсу користувача ● Візуалізація об’єктів ● Виконання запитів через GraphQL
Backend	● Обробка логіки додатку ● Інтеграція з БД (Elasticsearch, Redis, MinIO) ● Обробка API-запитів ● Підтримка черг RabbitMQ
Worker’и	● Асинхронна обробка повідомлень STIX ● Масове завантаження об'єктів ● Створення або оновлення об’єктів через API
З’єднувачі	● Отримання даних з зовнішніх джерел ● Збагачення даних ● Імпорт та експорт даних через API або RabbitMQ
Elasticsearch	● Швидкий пошук та індексація даних ● Підтримка аналітичних запитів

Продовження таблиці 2.1

Redis	● Тимчасове кешування ● Керування сесіями користувачів ● Швидкий доступ до даних
MinIO	Зберігання великих обсягів файлів (наприклад, PDF, зображень, логів)
RabbitMQ	● Посередник для асинхронної взаємодії ● Черги повідомлень для worker’ів та з’єднувачів

2.3 Принцип роботи платформи

Ефективність використання платформи кіберрозвідки значною мірою визначається її технічною архітектурою, а також тим, як вона оперує інформацією на концептуальному рівні. Платформа OpenCTI базується на чітко сформованих принципах роботи з даними про загрози, що охоплюють увесь цикл: від отримання індикаторів і TTP зловмисників до їх зберігання, аналізу, структурування та подальшого використання в контексті виявлення та протидії атакам.

Ключову роль у цьому процесі відіграють стандартизовані моделі даних, зокрема STIX 2.1, що забезпечує уніфікацію підходів до представлення IoC, TTP та інших об’єктів кіберзагроз. Розуміння принципів організації знань у межах OpenCTI є необхідною умовою для ефективної побудови сценаріїв виявлення загроз, побудови ланцюгів атак, встановлення зв’язків між подіями та ідентифікації суб’єктів, що стоять за атаками. У цьому контексті платформа виступає не просто як сховище, а як аналітичне ядро, навколо якого можуть бути побудовані гнучкі захисні механізми та розроблені власні рішення з автоматизованої взаємодії з індикаторами компрометації.

2.3.1 Принцип отримання, обробки та зберігання інформації на платформі

Для отримання даних платформа OpenCTI використовує з’єднувачі (connectors) – окремі служби, які інтегруються з зовнішніми джерелами даних, такими як платформи розвідки загроз (наприклад, MISP, VirusTotal, AbuseIPDB та іншими) або внутрішні системи організації, тобто SIEM-системи, EDR та AV-рішення, IDS- і IPS-системи та інші. З’єднувачі можуть бути реалізовані на різних мовах програмування та використовують API для отримання даних у форматі STIX 2.1. Це дозволяє автоматизувати процес збору інформації про загрози та забезпечити її актуальність.

Після отримання даних, вони передаються до черги повідомлень RabbitMQ, яка забезпечує асинхронну обробку та розподіл завдань між компонентами системи. Worker-и – це процеси на Python, які споживають повідомлення з черги, обробляють їх та передають до GraphQL API для подальшої обробки та збереження. Така концепція дозволяє масштабувати систему та обробляти великі обсяги даних у реальному часі, що є однією з головних переваг платформи.

Оброблені дані зберігаються в Elasticsearch, що забезпечує швидкий пошук та аналітику великих обсягів інформації. Для кешування та управління сесіями користувачів використовується Redis, що підвищує продуктивність системи. MinIO слугує для зберігання неструктурованих даних, таких як файли та зображення, забезпечуючи їх доступність та надійність.

Користувачі взаємодіють з платформою через вебінтерфейс, розроблений на React, що надає зручні інструменти для візуалізації та аналізу даних. Інтерфейс дозволяє виконувати пошук, переглядати звіти, аналізувати зв'язки між об'єктами загроз, будувати візуалізації, дізнаватись контекст та деталі певної атаки, завдяки чому приймати обґрунтовані рішення щодо кібербезпеки.

2.3.2 Роль STIX 2.1 моделі даних

У системі OpenCTI модель даних STIX 2.1 є основою для уніфікованого представлення інформації про кіберзагрози. Це не просто набір структурованих

об'єктів, а повноцінна мова опису кіберзагроз, яка дозволяє аналітикам, автоматизованим системам і платформам ефективно обмінюватися даними в єдиному форматі [23].

STIX 2.1 виконує стандартизацію об’єктів загроз: визначає набір об'єктів, які можуть описувати всі аспекти загрози, такі як:

- Threat Actor (суб’єкти загрози), це може бути, наприклад, група APT28 – російська група зловмисників, що відома кібернетичними атаками на державні, інформаційні, військові та інші структури розвинених країн.
- Campaign (кампанія) – це націлена група шкідливої активності, яка виконується однією групою зловмисників, має одну мету (шпигунство, крадіжка даних тощо), має спільні TTP та триває протягом певного періоду. Наприклад, кампанія SolarWinds, що відбувалась у 2020 році була дуже масштабною, у якій хакери зламали ПЗ компанії SolarWinds та отримали доступ до численних організацій по всьому світу.
- Malware (зловмисне ПЗ) – це конкретний зразок ПЗ, що використовувався для потрапляння в інфраструктуру організації або для виконання зворотнього шеллу, наприклад, як «Emotet» або «TrickBot».
- Indicator – це безпосередньо артефакт доказу зловмисної діяльності, такий, як IP-адреса, хеш, URL та інші.
- Observed Data (спостережені дані) – це та інформація, за допомогою якої може бути виявлено присутність в інфраструктурі зловмисної діяльності, наприклад, лог-файли, події з SIEM, спроцювання AV, firewall, IDS та IPS.
- Attack Pattern, Vulnerability, Tool – це ті дані, використовуючи які, можна описати тактику, техніку та процедури атак.

Отже, саме такі набори даних про загрозу чи атаку забезпечують структуровану інформацію, яка надає розуміння повного контексту атаки, що забезпечує можливість її виявлення ще на ранніх етапах та вчасної протидії.

Важливою особливістю STIX 2.1 є підтримка зв’язків між об'єктами, що дозволяє створити граф зв’язків між такими наборами об’єктів, у якому вузли

графа – це, безпосередньо, самі об'єкти, а ребра – це відносини між ними (рис. 2.2) [24].

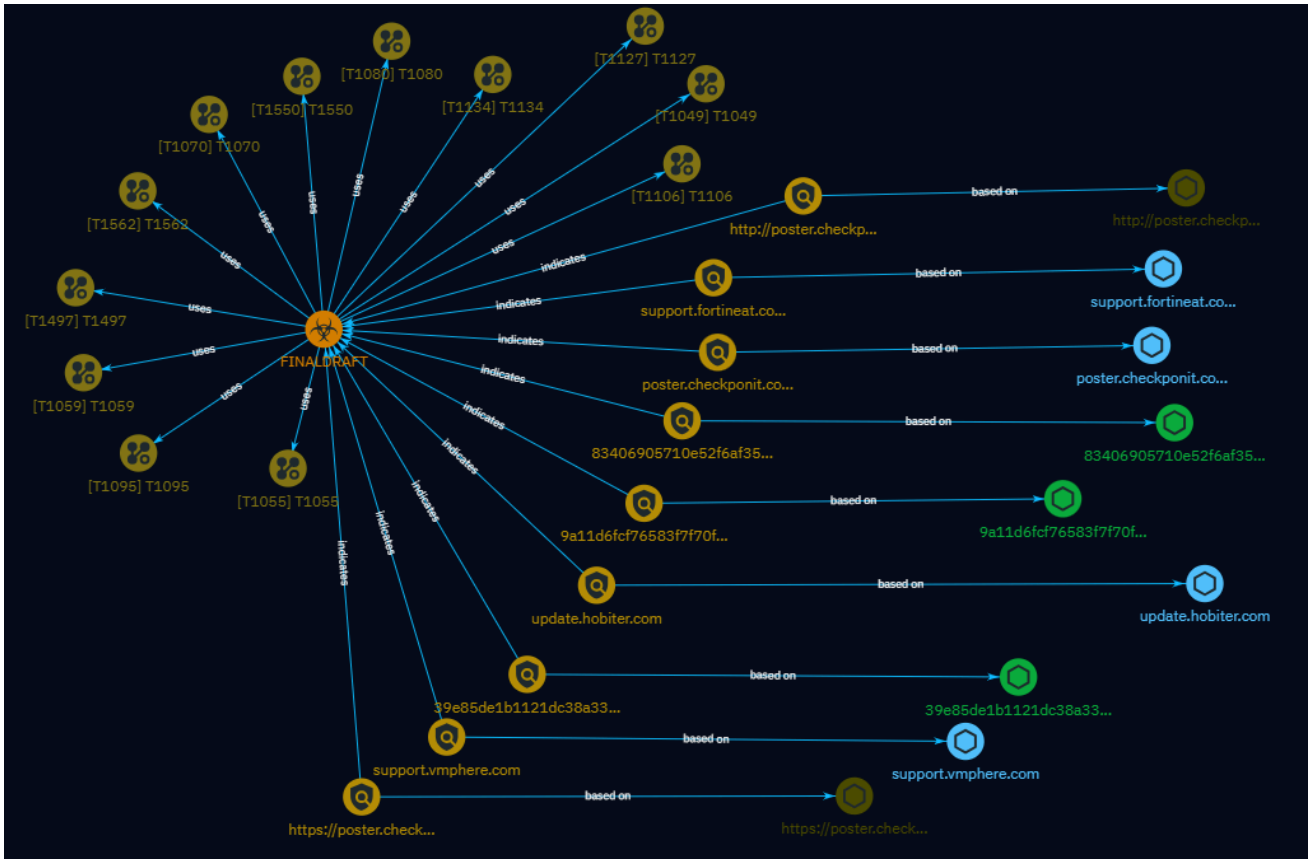


Рисунок 2.2 – Представлення взаємозв’язків між об’єктами у вигляді графу

Наприклад, з рисунку 2.2 видно, що таке зловмисне ПЗ, як FINALDRAFT, пов’язане з багатьма техніками MITRE ATT&CK такими, як T1095, T1059,T1497 та іншими, а також з представленими на рисунку IOC’ами: хеші, домени, URL.

Така модель дозволяє не тільки зберігати, а й виявляти нові залежності, шаблони поведінки або послідовності атак, використовуючи графові запити (наприклад, за допомогою GraphQL).

OpenCTI не обмежується лише базовими об’єктами STIX. Розробники платформи розширили STIX 2.1, щоб моделювати:

- дезінформацію (Narratives, Channels);
- фінансові аспекти (Cryptocurrency Wallets);

- активність у даркнеті;
- профілі користувачів, соціальні медіа, Telegram-канали тощо.

Це дозволяє адаптувати платформу до аналізу нових типів загроз, не змінюючи при цьому фундамент STIX.

Завдяки STIX 2.1 OpenCTI може безперешкодно обмінюватися даними з іншими платформами та системами:

- інтеграція з TAXII-серверами;
- експорт даних у STIX 2.1, JSON;
- сумісність з іншими CTI-інструментами, які використовують STIX (наприклад, MISP, IBM X-Force, Anomali тощо).

Такі можливості та функціонал є важливим для обміну загрозами між різними організаціями, CERT- та CSIRT-командами, державними структурами та приватними компаніями для своєчасного обміну та нормалізації інформації, а після – виявлення та реагування на інциденти ІБ.

2.3.3 Сценарії роботи з IOC, TTP, атаками та суб'єктами загроз

Платформа OpenCTI забезпечує цілісну взаємодію між різними типами об'єктів загроз, що дозволяє будувати повноцінну аналітичну картину на основі отриманих даних [25].

Коли до платформи потрапляє IP-адреса, помічена в активності зловмисників, вона автоматично перевіряється на відповідність вже відомим технікам і кампаніям. Якщо виявляються відповідності, то відбувається автоматичне збагачення: до індикатора додаються посилання на пов'язані атаки, TTPs, інструменти та іншу пов'язану з ним інформацію (див. рис. 2.3, рис. 2.4).

Таким чином, користувач не повинен ці пошуки виконувати вручну, а одразу отримує повноцінну картину.

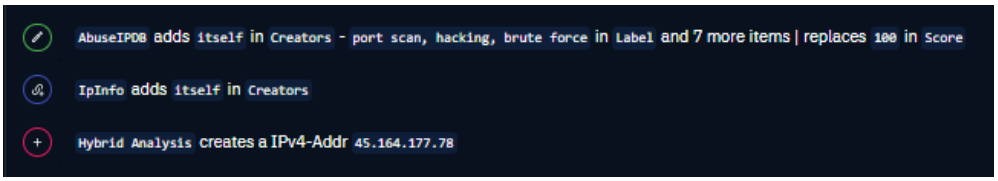


Рисунок 2.3 – Збагачення інформації про новостворений IoC

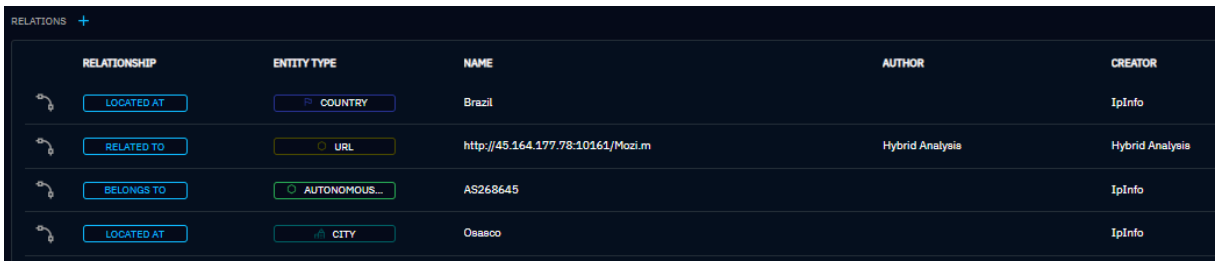


Рисунок 2.4 – Взаємопов’язані об’єкти з виявленим IoC

За кількома інцидентами ідентифікуються однакові техніки MITRE ATT&CK Map, наприклад, проникнення (persistence), уникнення виявлення (defense evasion) та інші. Після чого такі техніки об'єднуються у відповідні Attack Pattern (шаблони атак), які надалі використовуються для побудови профілю загрози (рис. 2.5). Завдяки механізмам відображення зв’язків стає зрозуміло, що це може бути проявом нової кампанії.

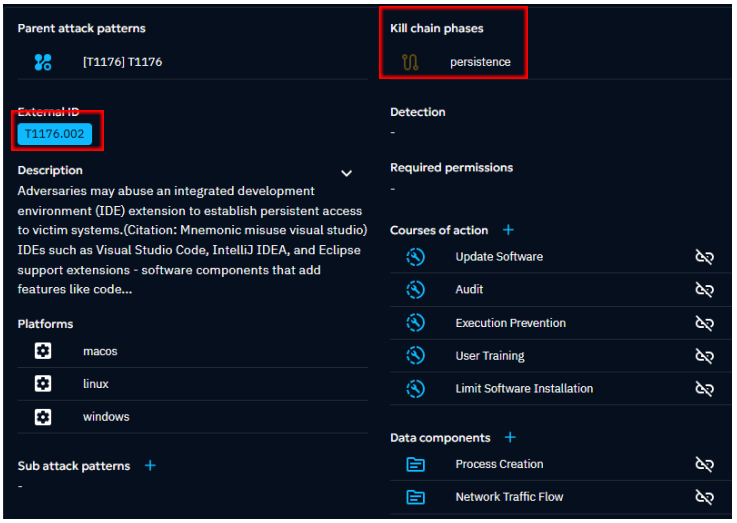


Рисунок 2.5 – Приклад шаблону атаки на основі техніки з MITRE ATT&CK Map

Також платформа автоматично виявляє пов’язані з шаблонами атак зразки зловмисного ПЗ, вразливості, набори атак та іншу залежну інформацію (рис. 2.6).

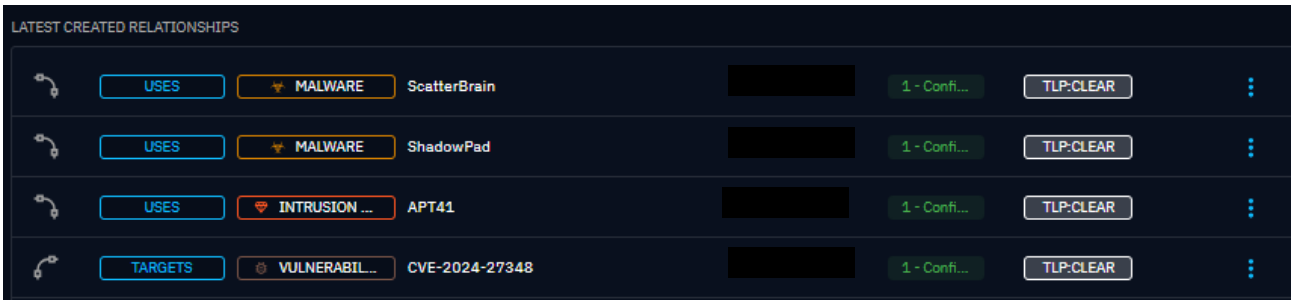


Рисунок 2.6 – Пов’язані з шаблоном атаки індикатори компрометації

Після аналізу кількох подій встановлюється, що всі вони використовують однаковий набір TTP та шкідливе ПЗ, після чого на основі цього визначається підозрювана група (Intrusion Set), до якої прив'язуються всі відповідні об’єкти (рис. 2.7).

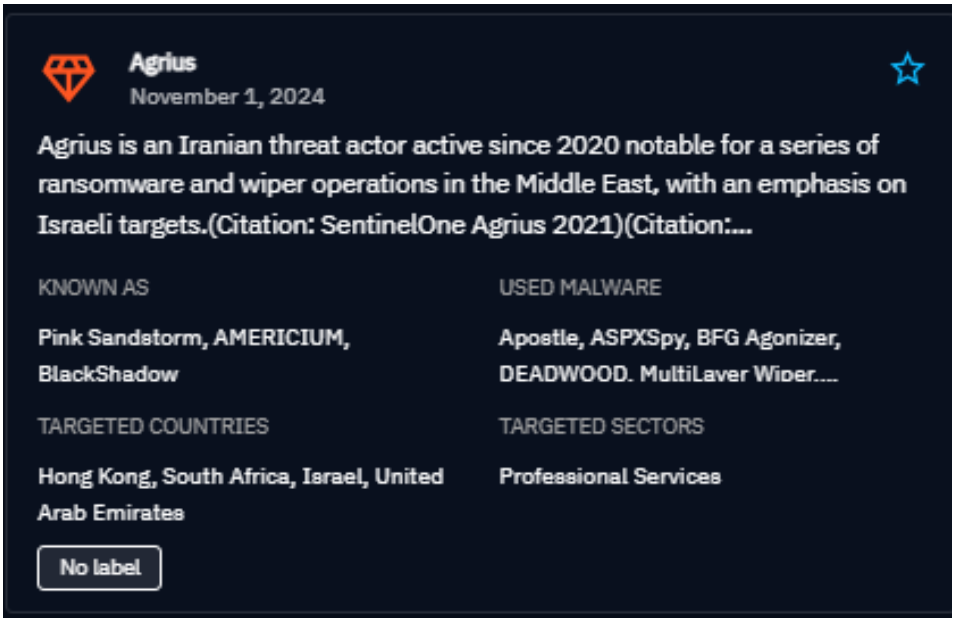


Рисунок 2.7 – Приклад створеного Intrusion Set по групі зловмисників Agrius

Після створення Intrusion Set до нього одразу ставиться у відповідність залежна йому інформація, така як IoC, звіти по атакам, що належать цій групі

зловмисників, та посилання на зовнішні джерела з інформацією про атаки (рис. 2.8).

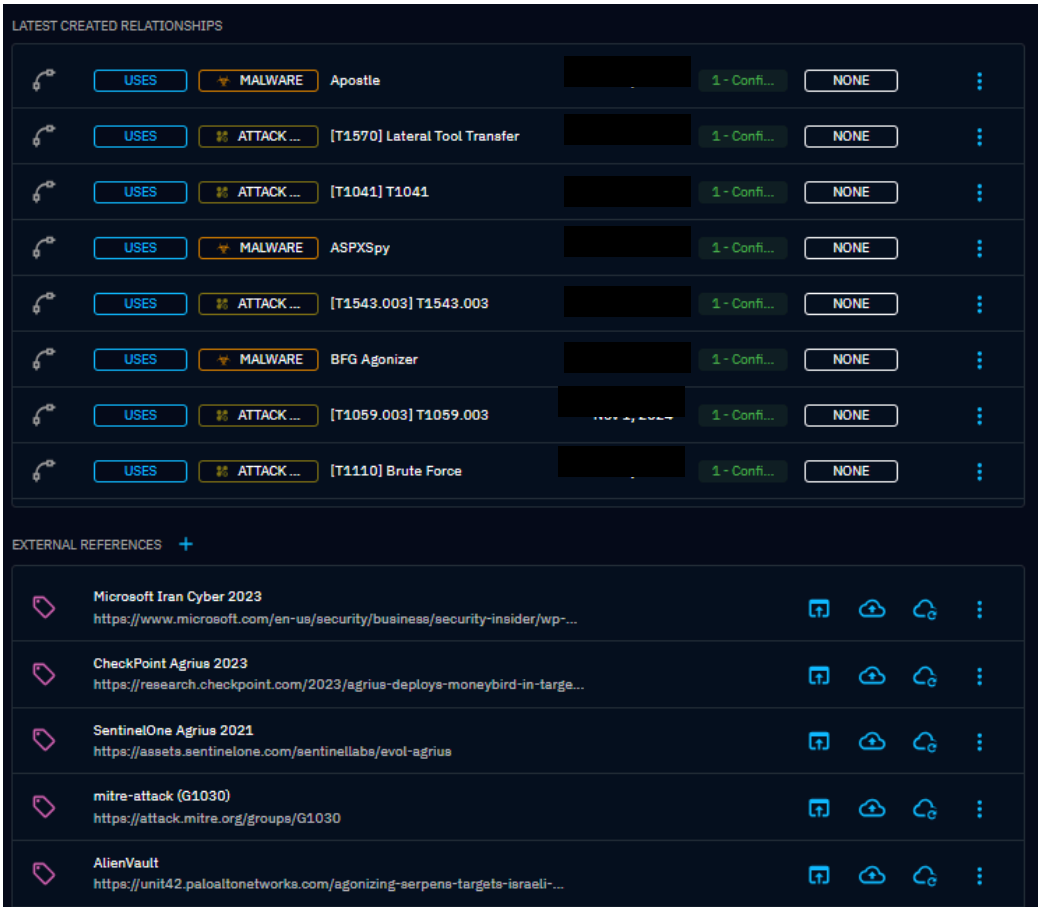


Рисунок 2.8 – Інформація, що пов’язується з Intrusion Set

Таким чином, створений профіль зловмисника дає змогу відстежувати його активність, а також прогнозувати потенційні наступні кроки.

2.4 Переваги та недоліки платформи OpenCTI

У ході дослідження платформи кіберрозвідки OpenCTI, аналізу її функціональних, технічних та архітектурних характеристик і можливостей, можна сформувати її переваги та недоліки для використання та впровадження як системи захисту інформації в інформаційно-комунікаційну систему.

До переваг платформи можна віднести її наступні можливості:

- Модульна архітектура та гнучкість інтеграцій. Платформа OpenCTI побудована з урахуванням можливості інтеграції з іншими системами через GraphQL API, REST API та плагіни-розширення (наприклад, з MISP, TheHive, MITRE ATT&CK, ElasticSearch, Shodan тощо), що, у свою чергу, дозволяє адаптувати платформу під конкретні потреби, побудувати власну екосистему Threat Intelligence.
- Підтримка моделі STIX 2.1. Використання стандарту STIX 2.1 забезпечує сумісність з іншими платформами та дозволяє структурувати інформацію відповідно до міжнародних норм. Це підвищує якість даних та полегшує їх збереження, обробку, аналіз, та передачу.
- Візуалізація зв’язків і атак. Інтерфейс платформи OpenCTI дозволяє будувати графічні візуалізації зв’язків у вигляді дашбордів та графів між об'єктами: атаками, загрозами, суб’єктами, індикаторами компрометації, кампаніями, використаними TTP та іншим наборами даних. Це значно полегшує розуміння контексту загрози та її поширення в IT-інфраструктурі. До того ж, саме ця можливість суттєво вирізняє OpenCTI на фоні інших платформ кіберрозвідки.
- Автоматизація аналізу загроз. Завдяки підключенню з’єднувачів OpenCTI може автоматично отримувати дані з відкритих джерел або платних провайдерів (наприклад, AlienVault OTX, AbuseIPDB, MalwareBazaar та інших), що спрощує моніторинг нових загроз, а також забезпечує оновлення даних про загрози в режимі реального часу.
- Підтримка об’єднання та обміну інформацією між командами. Платформа дозволяє багатьом користувачам одночасно працювати над інформацією про загрози, коментувати, робити перегляди, відслідковувати зміни. Це особливо важливо для SOC-команд або груп реагування на інциденти, оскільки це забезпечує миттєвий обмін статусами виконання задач, а також спільну роботу над проєктом в режимі реального часу.
- Відкритий вихідний код і активна спільнота. Наявність відкритого вихідного коду забезпечує повну прозорість логіки роботи, можливість

кастомізації, тобто виконання доопрацювань, підключення власних програмних модулів, та активну підтримку з боку спільноти користувачів. Крім того, це дозволяє безкоштовно використовувати платформу без ліцензійних витрат, що є важливим і для окремих користувачів, і для невеликих організацій, що прагнуть налаштувати власну систему розвідки кіберзагроз без великих затрат.

Проте, як і всі рішення, дана платформа має певні недоліки, як в архітектурі, так і в принципах роботи самої платформи, такі як:

- Високі вимоги до інфраструктури. Для повноцінного розгортання платформи OpenCTI в організації необхідна складна інфраструктура на базі Docker, включаючи такі компоненти, як Elasticsearch, RabbitMQ, Redis, MinIO та інші компоненти. Це може викликати труднощі на етапі розгортання для користувачів з обмеженими ресурсами.
- Відсутність вбудованого механізму верифікації даних. Платформа не забезпечує автоматичну перевірку достовірності загроз із зовнішніх джерел, що вимагає додаткової валідації отриманої інформації вручну або через зовнішні сервіси.
- Вимоги до навчання персоналу та користувачів. Робота з платформою вимагає попереднього ознайомлення з моделлю STIX, механізмами фільтрації та зв’язків, конфігурацією з’єднувачів. Для початківців у сфері розвідки кіберзагроз це може бути складним і зайняти більше часу.
- Обмежений функціонал без плагінів. У стандартній інсталяції платформа надає лише базовий набір можливостей. Для повного функціоналу необхідно налаштовувати додаткові модулі (наприклад, ingestion з’єднувачі, enrichment tools), що потребує окремої підтримки.

Отже, проаналізувавши всі переваги та недоліки, можна стверджувати, що саме це рішення дуже вирізняється своїми можливостями та функціоналом, в порівнянні з іншими, оскільки воно може бути доповнено власними розробками, працювати в автоматизованому режимі, підтримувати велику кількість інтеграцій, надає можливості для візуалізації та побудови зв’язків між інформацією. І, незважаючи на недоліки системи, вона вирізняється серед

аналогів спектром можливостей та функціоналу, що надає їй перевагу у виборі. З огляду на відкритий код, підтримку сучасних стандартів передачі даних, а також наявність широкої спільноти користувачів і документації, ця платформа є ідеальним середовищем для побудови модулів автоматичного виявлення маркерів компрометації.

Висновки до розділу 2

У цьому розділі було проведено комплексний аналіз платформи OpenCTI як інструменту для збору, обробки та аналізу інформації про кіберзагрози. Розглянуто загальні характеристики платформи, її цілі, основні функціональні можливості та роль у сучасному світі розвідки кіберзагроз. Окрему увагу приділено архітектурі OpenCTI: детально описані основні компоненти системи – серверна та клієнтська частини, обробники даних та з’єднувачі, а також ключові технології, що використовуються для забезпечення масштабованості, продуктивності, стабільності та ефективності роботи. Окрім цього було проведено аналіз процесу взаємодії цих елементів між собою для забезпечення безперервної обробки інформації, досліджено можливість адаптації платформи до змін у структурі даних та зовнішніх інтеграціях.

Було розкрито принципи отримання, обробки та зберігання інформації на платформі, акцентовано увагу на важливості використання стандарту STIX 2.1 для структурованого опису об'єктів загроз та забезпечення сумісності з іншими рішеннями у сфері кібербезпеки. Також особливу увагу приділено сценаріям роботи з індикаторами компрометації, тактиками і техніками атак, кампаніями та суб'єктами загроз, що дозволяє будувати цілісну аналітичну картину інцидентів. Саме можливість побудови зв’язків між об’єктами у вигляді графів є беззаперечною перевагою платформи OpenCTI серед інших. Такий підхід забезпечує не лише гнучкість у роботі аналітика, але й створює основу для побудови ефективних систем реагування на інциденти. Графова модель

дозволяє виявляти приховані залежності між подіями, що часто залишаються поза увагою при традиційному аналізі.

Крім того, було окреслено основні переваги використання OpenCTI, серед яких гнучкість інтеграцій, розвинена система візуалізації, автоматизація аналізу загроз та підтримка відкритого коду. Водночас проаналізовано і певні обмеження платформи, пов’язані з високими вимогами до інфраструктури, потребою у глибоких технічних знаннях для налаштування, розгортання та підтримки, а також необхідністю додаткової валідації отриманої інформації.

Результати проведеного аналізу формують необхідне теоретичне підґрунтя для реалізації практичної частини цієї роботи – розробки власного програмного модуля для виявлення індикаторів компрометації в атаках на вебресурси, в якому буде реалізовано інтеграцію з обраною платформою. Отримані знання про архітектуру, принципи роботи та сценарії взаємодії з платформою забезпечать можливість побудови ефективного рішення для автоматизованого моніторингу загроз у реальному часі.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ ІНДИКАТОРІВ КОМПРОМЕТАЦІЇ В РОБОТІ ВЕБДОДАТКІВ

3.1 Принцип роботи програмного модуля

Розроблений програмний модуль призначений для моніторингу запитів до вебсервера з метою виявлення потенційно шкідливої активності в роботі вебдодатків на основі аналізу логів вебсервера. Його принцип роботи полягає у здійсненні безперервного контролю над вхідними вебзапитами, що записуються у журнал подій вебсервера, та виконанні ряду перевірок на основі формалізованих шаблонів атак і відомих індикаторів компрометації.

Мета роботи програмного модуля – безперервний моніторинг цільового лог-файлу у режимі реального часу. Він функціонує як фоновий системний сервіс (systemd), автоматично запускається при старті системи та не потребує додаткового втручання для запуску.

У разі появи нового запису, який дописується в кінець лог-файлу, він аналізується за допомогою регулярного виразу, адаптованого до стандартного формату Apache-логів. З отриманого рядка вилучається структурована інформація: IP-адреса джерела запиту, дата та час, HTTP-метод, запитуваний URL ресурсу, код статусу відповіді сервера та інші параметри.

На наступному етапі URL запиту порівнюється з набором заздалегідь визначених шаблонів, які описують характерні ознаки найпоширеніших типів атак на вебресурси. Шаблони визначено для таких типів атак: SQL-ін’єкції, XSS (Cross-site Scripting), командна ін’єкція, локальне або віддалене включення файлів, directory traversal, а також зловмисні перенаправлення. У разі відповідності запиту певному шаблону, модуль автоматично визначає тип атаки.

Окрім сигнатурного підходу, у модулі реалізовано rule-based механізм виявлення аномальної поведінки. Згідно з заданими правилами, підозрілим

вважається запит, якщо він: перевищує нормальну довжину (понад 200 символів), або містить небезпечні символи чи escape-послідовності (наприклад, <, |, %00), або використовує нестандартні HTTP-методи (PUT, OPTIONS тощо).

Паралельно з класифікацією запиту виконується вилучення додаткових артефактів, які є потенційними індикаторами компрометації. До них належать: IP-адреси, URL, доменні імена, а також хеш-значення, що можуть бути присутніми у параметрах запитів. Виявлені IoC передаються на перевірку до платформи кіберрозвідки OpenCTI.

Інтеграція з OpenCTI реалізована через GraphQL API з використанням TLS-сертифіката клієнта для автентифікації. У разі успішного з'єднання формується запит на пошук об'єктів типу STIX (наприклад, ipv4-addr, url, domain-name, file) у базі відомих індикаторів. Якщо хоча б один із переданих IoC виявляється в базі, це вважається підтвердженням потенційної компрометації.

Інформація про виявлену активність фіксується в логах та надсилається користувачеві у вигляді структурованого повідомлення в месенджер Telegram. Якщо запит містив IoC, що збігається з даними платформи OpenCTI, модуль також додає до повідомлення релевантні теги, пов'язані з цим індикатором (наприклад, cobalt strike, ransomware, C2).

Для зменшення навантаження на OpenCTI реалізовано кешування перевірених IoC з тайм-аутом у 30 хвилин, що дозволяє уникати повторних запитів для нещодавно перевірених значень.

З метою підтримки аналітики модуль накопичує статистику виявлених атак за їх типами. Раз на добу, якщо протягом періоду було зафіксовано хоча б одну атаку, модуль формує узагальнене текстове сповіщення та надсилає його користувачеві у чат в Telegram. Якщо атак не було, повідомлення не формується.

У завершальній фазі обробки формується фінальне повідомлення для Telegram, що містить дані про тип атаки, її джерело, час, параметри запиту, а

також, за наявності, список виявлених індикаторів компрометації з підтвердженням їхньої присутності в базі загроз OpenCTI.

Таким чином, розроблений програмний модуль виконує роль автоматизованого засобу первинного аналізу HTTP-трафіку з функціональністю виявлення вебатак, ідентифікації IoC та інтеграції з платформами кіберрозвідки, що підвищує ефективність реагування на інциденти інформаційної безпеки.

Схему роботи програмного модуля представлено на рисунку 3.1.

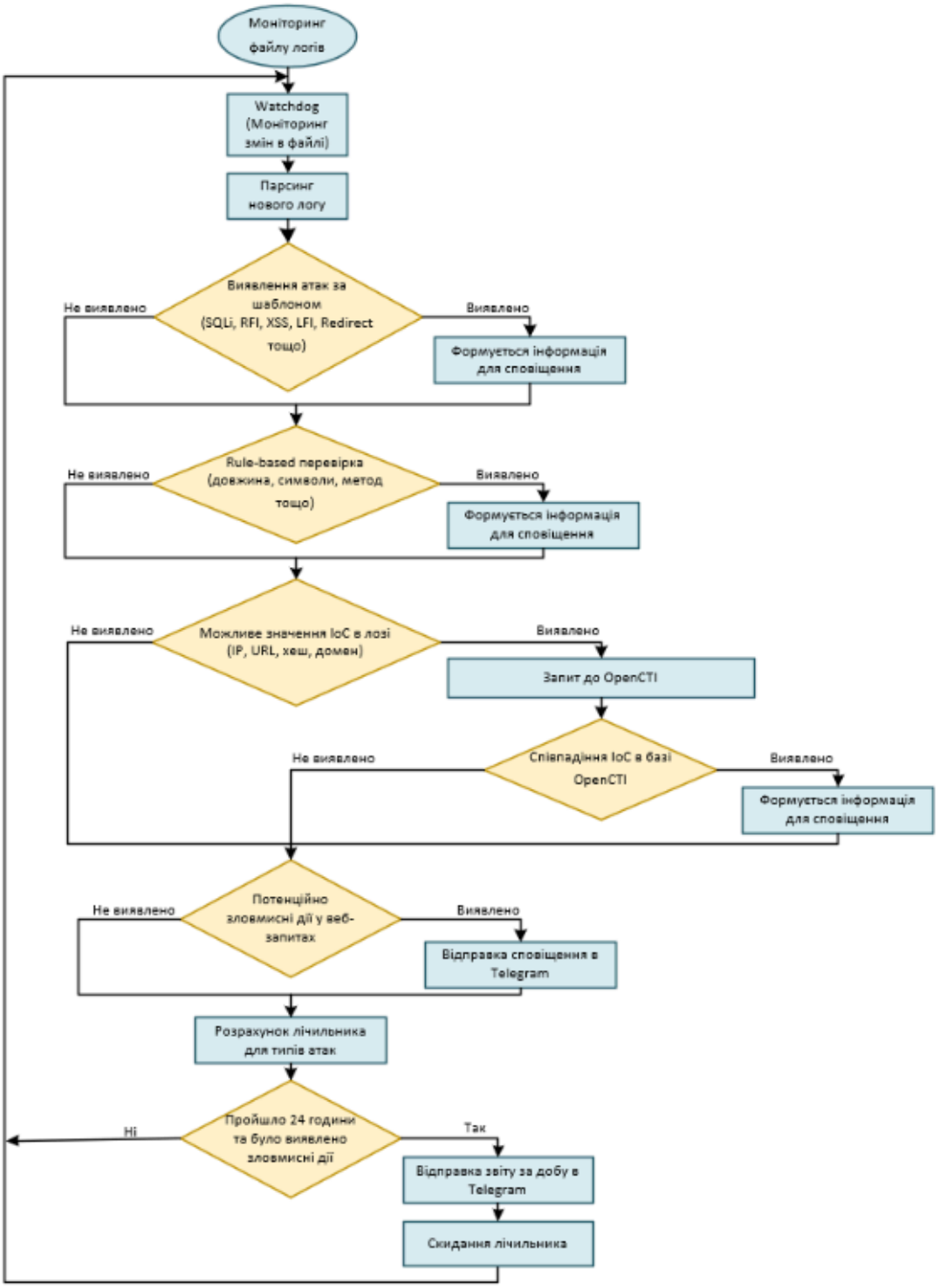


Рисунок 3.1 – Схема, що демонструє принцип роботи програмного модуля

3.2 Обґрунтування вибору мови програмування для реалізації програмного модуля

Для розробки програмного модуля було обрано мову програмування Python. Це зумовлено сукупністю практичних переваг, таких як: зручність

роботи з даними, широкі можливостями інтеграції з API, а також підтримка інструментів, необхідних для вирішення прикладних завдань у сфері кібербезпеки.

Python є однією з найпоширеніших мов програмування у галузі ІБ, що зумовлено гнучкістю, синтаксисом, великою кількістю спеціалізованих бібліотек і активною професійною спільнотою. Python дозволяє ефективно реалізовувати інструменти для автоматичного збору, аналізу й візуалізації загроз, що робить його зручним для створення скриптів та сервісів.

Окремо варто відзначити, що використання Python у цьому проєкті є не лише зручним з погляду розробки, а й технологічно сумісним з платформою OpenCTI. Основна логіка OpenCTI, зокрема її GraphQL API, бекенд-воркери та з'єднувачі, реалізовані саме на Python. Це дозволяє досягти високої інтеграційної сумісності між модулем моніторингу та платформою кіберрозвідки без потреби у проміжних рішеннях або адаптаційних інтерфейсах.

Крім того, Python забезпечує підтримку асинхронної обробки подій, що є критично важливим для реалізації систем реального часу. У межах даного модуля це дозволяє паралельно здійснювати моніторинг логів, перевірку індикаторів у OpenCTI та надсилання сповіщень у Telegram.

Також з метою обгрунтування доцільності вибору мови програмування було проведено порівняльний аналіз різних мов програмування на основі найважливіших критеріїв для написання програмного модуля та повної реалізації його функціоналу (див. табл. 3.1).

Таблиця 3.1

Порівняння функціональних можливостей мов програмування

Критерій	**Python**	**C++**	**Java**	**Go**	**JavaScript**

Простота синтаксису	Висока	Низька	Середня	Середня	Середня
Підтримка API та мережевих протоколів	Напряму, зручна	Потреба в додаткових бібліотеках	Потреба в сторонніх фреймворках	Обмежена	Висока
Робота з логами, текстовими файлами	Зручна	Ускладнена, не оптимізована	Середня	Середня	Середня
Асинхронна обробка подій	Доступно (asyncio)	Вимагає складних реалізацій	Потреба в сторонніх фреймворках	Доступно	Доступно
Поширеність для автоматизації процесів	Дуже висока	Низька	Середня	Низька	Середня
Швидкість виконання	Середня	Висока	Висока	Висока	Середня
Сумісність з OpenCTI	Повна	Відсутня	Низька	Відсутня	Відсутня

Таким чином, проаналізувавши функціональні можливості різних мов програмування на основі найважливіших критеріїв, що необхідні для розробки програмного модуля та забезпечення виконання всіх вимог до нього, Python забезпечує найкращий баланс між зручністю розробки, функціональністю, інтеграційною сумісністю та гнучкістю. Важливо також, що він дозволяє створювати фонові служби, що здатні працювати у режимі постійного моніторингу з мінімальними системними вимогами.

Таким чином, вибір Python як основної мови програмування для реалізації даного модулю є обґрунтованим як з технічної, так і з архітектурної точки зору, забезпечуючи ефективну взаємодію із зовнішніми системами, підтримку роботи з логами в реальному часі та високу швидкість розгортання рішення.

3.3 Реалізація програмного модуля

Реалізація програмного модуля базується на ідеї побудови легкого, незалежного інструменту, що працюватиме в фоновому режимі і безперервно спостерігатиме за подіями в логах вебсервера, здійснюватиме аналіз вмісту запитів і виявлятиме ознаки атак чи індикатори компрометації.

Особливістю реалізації є поєднання кількох методів аналізу: сигнатурного (на основі шаблонів відомих атак), rule-based (за формальними ознаками аномальної поведінки), а також семантичного – шляхом перевірки потенційно зловмисних ознак у платформі OpenCTI. Усі виявлені події обробляються у асинхронному режимі, з використанням черги повідомлень та відправкою релевантної інформації користувачеві через Telegram, що дозволяє реагувати на загрози з мінімальною затримкою.

3.3.1 Зчитування логів вебсервера та їх парсинг

Першим етапом роботи програмного модуля є зчитування нових записів із лог-файлу вебсервера Apache та трансформація кожного запису у структурований формат. Це необхідно для подальшого аналізу запитів на предмет наявності атак або індикаторів компрометації.

Для безперервного моніторингу логів використано бібліотеку watchdog, яка реалізує механізм моніторингу файлової системи [26]. У межах цього проєкту застосовується її полінговий варіант – PollingObserver. На відміну від подієвих підходів, які ґрунтуються на системних викликах (наприклад, inotify у

Linux), полінговий підхід працює шляхом періодичного опитування цільового файлу з певним інтервалом часу.

Таким чином, PollingObserver періодично перевіряє стан файлу (наприклад, його довжину або час останньої модифікації) й у разі виявлення змін ініціює обробку нових рядків (рис. 3.2). Це робить його кросплатформеним і надійним рішенням, особливо в тих середовищах, де немає повної підтримки подієвого моніторингу на рівні ОС.

Клас LogHandler, реалізований у модулі, реагує на подію on_modified() і зчитує тільки ті рядки, які були додані з моменту останнього спрацювання. Реалізований механізм дозволяє обробляти лише доданий вміст, що мінімізує навантаження та забезпечує швидке реагування на нові події.

```
def start_watcher(path, loop, queue):
    handler = LogHandler(path, loop, queue)
    observer = PollingObserver()
    observer.schedule(handler, path=os.path.dirname(path), recursive=False)
    observer.start()
    return observer
```

Рисунок 3.2 – Ініціалізація механізму моніторингу лог-файлу

Змінна self.position зберігає останню оброблену позицію у файлі, що дозволяє обмежити зчитування лише новим вмістом. Отримані рядки передаються на подальший аналіз (рис.3.3).

```
    def on_modified(self, event):
        if event.src_path != self.path:
            return
        with open(self.path) as f:
            f.seek(self.position)
            lines = f.readlines()
            self.position = f.tell()
```

Рисунок 3.3 – Фрагмент коду зчитування лише нових рядків лог-файлу

Кожен новий рядок журналу доступу вебсервера має фіксований формат, у якому зазвичай зазначено IP-адресу джерела, дату та час, HTTP-метод, запитуваний ресурс, код відповіді сервера та інші параметри. Для розбору цього

рядка використовується регулярний вираз, визначений змінною apache_log_pattern (рис. 3.4).

```
apache_log_pattern = re.compile(
    r'(?P<ip>\d+\.\d+\.\d+\.\d+) - - \[(?P<datetime>[^\]]+)\] '
    r'"(?P<method>[A-Z]+) (?P<url>[^ ]+) (?P<protocol>[^"]+)" '
    r'(?P<status>\d+) (?P<size>\d+|-)'
)
```

Рисунок 3.4 – Регулярний вираз для парсингу логів вебсервера Apache

Регулярний вираз адаптовано до Common Log Format, що використовується Apache за замовчуванням (рис. 3.5). Завдяки іменованим групам (?P<...>) кожне поле автоматично витягується у словник із ключами: ip, datetime, method, url, protocol, status, size. Якщо новий рядок лог-файлу успішно відповідає цьому шаблону, його вміст передається як словник log у наступні функції обробки.

```
127.0.0.1 - - [20/May/2025:21:15:00 +0300] "GET /download.php?file=../../../boot.ini HTTP/1.1" 200 298
```

Рисунок 3.5 – Приклад логу вебсервера Apache

Таким чином, ця частина модуля відповідає за обробку нових HTTP-запитів, гарантуючи, що кожен новий запис у журналі буде оперативно розпізнаний, структурований і переданий у подальший ланцюг аналізу.

3.3.2 Логування подій обробки

Для формування журналу подій та зберігання структурованої інформації про виявлену активність у модулі використовується бібліотека structlog, яка надає інтерфейс для логування у форматі JSON [27]. Це дозволяє формувати лог-повідомлення у вигляді об’єктів із ключами та значеннями, що значно спрощує їх подальший аналіз або обробку зовнішніми засобами (рис. 3.6).

Використання процесора JSONRenderer забезпечує вивід усіх подій у вигляді форматованого JSON, а logger стає об’єктом, через який протягом усього циклу роботи фіксуються ключові дії модуля – виявлення атак, відповідність індикаторів у OpenCTI, статистика тощо. Такий підхід до логування є важливою частиною як внутрішнього аудиту, так і забезпечення відтворюваності результатів під час тестування та верифікації роботи модуля.

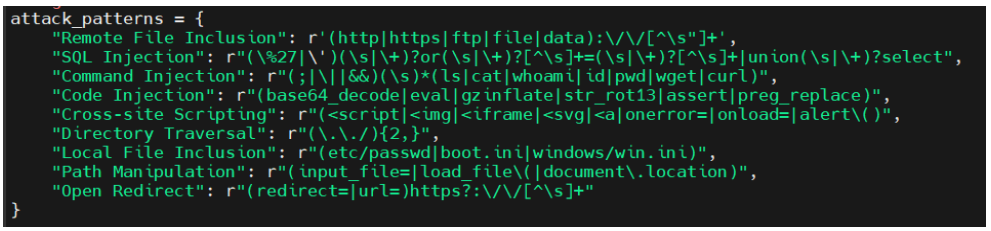
```
{
    "log": {
        "ip": "127.0.0.1",
        "datetime": "20/May/2025:21:15:00 +0300",
        "method": "GET",
        "url": "/download.php?file=../../../boot.ini",
        "protocol": "HTTP/1.1",
        "status": "200",
        "size": "298"
    },
    "attack_type": "Directory Traversal",
    "event": "Malicious activity detected"
}
```


Рисунок 3.6 – Приклад логу вебсервера Apache в форматі JSON в результаті обробки програмним модулем

3.3.3 Виявлення різних типів атак за шаблонами

Основним механізмом попереднього виявлення зловмисної активності у запитах до вебсервера є сигнатурний аналіз – це метод, що базується на пошуку ознак відомих атак шляхом зіставлення вмісту запиту з наперед визначеними регулярними виразами (шаблонами). У модулі реалізовано набір таких шаблонів для виявлення найпоширеніших класів вебатак [28].

Шаблони зберігаються у структурі attack_patterns, представленій у вигляді словника, де ключем є назва атаки, а значенням – відповідний регулярний вираз (рис. 3.7).



```
attack_patterns = {
    "Remote File Inclusion": r'(http|https|ftp|file|data):\/\/[^\s"]+',
    "SQL Injection": r"(\%27|\')(\s|\+)?or(\s|\+)?[^\s]+=(\s|\+)?[^\s]+|union(\s|\+)?select",
    "Command Injection": r"(;|\||&&)(\s)*(ls|cat|whoami|id|pwd|wget|curl)",
    "Code Injection": r"(base64_decode|eval|gzinflate|str_rot13|assert|preg_replace)",
    "Cross-site Scripting": r"(<script|<img|<iframe|<svg|<a|onerror=|onload=|alert\()",
    "Directory Traversal": r"(\.\./){2,}",
    "Local File Inclusion": r"(etc/passwd|boot.ini|windows/win.ini)",
    "Path Manipulation": r"(input_file=|load_file\(|document\.location)",
    "Open Redirect": r"(redirect=|url=)https?:\/\/[^\s]+"
}
```

Рисунок 3.7 – Регулярні вирази для визначення шаблонів атак

Кожен шаблон містить сигнатуру, характерну для певного типу атак:

- SQL Injection має такі ключові елементи конструкцій, як: OR 1=1, UNION SELECT та інші;
- XSS (Cross-site Scripting) виділяється з-поміж інших наявністю тегів <script>, <img>, JavaScript-події onerror, onload;
- RFI/LFI (віддалене або локальне включення файлів) – це використання шляхів в URL по типу file://..., ../../etc/passwd;
- Command Injection – це підрядки типу ; ls, | cat, && whoami;
- Open Redirect завжди містить параметри, наприклад, url= або redirect= з підставленими сторонніми посиланнями.

Для виконання перевірки запиту використовується функція detect_attack_type(url), яка перевіряє словник attack_patterns та застосовує регулярний вираз до URL-адреси, що міститься у лог-записі (рис. 3.8).

```
def detect_attack_type(url):
    for attack_type, pattern in attack_patterns.items():
        if re.search(pattern, url, re.IGNORECASE):
            return attack_type
    return None
```

Рисунок 3.8 – Функція визначення типу атаки на основі шаблонів

У випадку виявлення відповідності хоча б одному шаблону, функція повертає ідентифікатор типу атаки, який далі використовується для фіксації інциденту, інкременту статистики та формування повідомлення для Telegram.

Таким чином, сигнатурне виявлення у модулі реалізується у вигляді гнучкої та розширюваної системи, де кожен шаблон легко доповнюється, а механізм регулярних виразів дозволяє виявляти атаки навіть у частково обфускованому вигляді, що забезпечує високу швидкодію і точність для виявлення широкого спектру відомих атак без залучення складних моделей або зовнішніх сервісів.

3.3.4 Rule-based виявлення аномальних запитів

Окрім сигнатурного підходу, що ґрунтується на пошуку шаблонів відомих атак, у модулі реалізовано механізм перевірки на основі правил, тобто виявлення запитів, які не містять чітких ознак відомих атак, не співпадають з заданими шаблонами, але виглядають зловмисними за своєю структурою.

Такий підхід є важливим доповненням до сигнатурного аналізу, оскільки дозволяє виявляти експериментальні або модифіковані варіанти атак, ознаки сканування вебдодатку чи виклики із нетиповими параметрами або методами. У модулі ця логіка реалізована у вигляді функції is_anomalous_request (рис. 3.9).

```
def is_anomalous_request(method, url):
    if len(url) > 200:
        return "Long URL"
    if re.search(r"[<>&|%00]", url):
        return "Suspicious characters"
    if method not in {"GET", "POST", "HEAD"}:
        return "Uncommon HTTP method"
    return None
```

Рисунок 3.9 – Функція rule-based виявлення аномальних вебзапитів

Серед визначених в цій функції умов, перевіряється наступне:

1. Довжина URL: якщо вона перевищує 200 символів, такий запит вважається підозрілим. Це може свідчити про спробу SQL-ін’єкції або XSS-атаки з довгими параметрами [29].

2. Наявність небезпечних символів: регулярний вираз шукає у запиті символи <, >, &, |, а також нульовий байт (%00). Їх присутність часто пов'язана з маніпуляціями в HTML/JavaScript або shell-командами.

3. Нестандартні HTTP-методи: для вебдодатків зазвичай використовуються методи GET, POST і HEAD. Інші (наприклад, PUT, OPTIONS, TRACE) використовуються рідше та можуть свідчити про спробу тестування поведінки сервера або зловживання методами [30].

Сама функція повертає короткий опис причини, через яку запит вважається аномальним. Якщо жодна з умов не виконується, то функція повертає None, що означає відсутність підозрілої поведінки.

У контексті основного коду, у разі якщо сигнатурне виявлення не дало результату, викликається саме ця rule-based перевірка (рис. 3.10).

```
if not attack:
    reason = is_anomalous_request(log.get("method", ""), url)
    if reason:
        attack = f"Atypical behavior: {reason}"
```

Рисунок 3.10 – Інтеграція rule-based перевірки в процес аналізу запиту

Таким чином, rule-based підхід розширює діапазон інцидентів, що можна виявити, за межі шаблонних атак, дозволяючи модулю реагувати на менш очевидні, але потенційно небезпечні запити.

3.3.5 Інтеграція з платформою OpenCTI

Окрім сигнатурного та rule-based аналізу вебзапитів, програмний модуль реалізує інтеграцію з платформою OpenCTI, завдяки цьому модуль має можливість зіставляти елементи запиту з уже відомими індикаторами компрометації, що дозволяє підвищити точність виявлення атак, навіть у разі їх нетипової форми.

Після ідентифікації запиту як потенційно шкідливого, з нього виокремлюються дані, що можуть містити ознаки зловмисного IoC. Для цього використовується функція extract_iocs(log_data), яка здійснює аналіз поля url та формує словник із трьох можливих типів індикаторів (рис. 3.11).

```python
def extract_iocs(log_data):
    iocs = {"ip": log_data.get("ip"), "url": None, "domain": None, "hash": None}
    url = log_data.get("url", "")
    if url:
        if "http://" in url or "https://" in url:
            iocs["url"] = url
        domain = re.search(r"https?://([^/?#]+)", url)
        if domain:
            iocs["domain"] = domain.group(1)
        hash_ = re.search(r"\b[a-fA-F0-9]{32,64}\b", url)
        if hash_:
            iocs["hash"] = hash_.group(0)
    return iocs
```

Рисунок 3.11 – Ініціалізація структури для збереження потенційних індикаторів компрометації

У разі, якщо URL містить http:// або https://, він заноситься до поля url. Окремо з нього вилучається доменне ім’я, а також за допомогою регулярного виразу хеш-схожі значення довжиною 32-64 символи (наприклад, MD5, SHA1 або SHA256). Ці елементи можуть бути використані зловмисниками для звернення до C2-серверів, доставки шкідливих файлів або передачі даних.

Після виділення з URL IoC формується GraphQL-запит для платформи OpenCTI [31]. Структура запиту побудована відповідно до специфікації API платформи й містить змінну filters, що відповідає типу FilterGroup. Цей тип описує набір умов, за якими виконується пошук індикаторів компрометації у сховищі stixCyberObservables. (рис. 3.12).

```
query = """
query CheckObservable($filters: FilterGroup!) {
  stixCyberObservables(filters: $filters) {
    edges {
      node {
        entity_type
        observable_value
        objectLabel { value }
      }
    }
  }
}"""
variables = {
    "filters": {
        "mode": "and",
        "filters": [{ "key": "value", "values": [value] }],
        "filterGroups": []
    }
}
```

Рисунок 3.12 – GraphQL-запит для перевірки індикатора компрометації в OpenCTI

Запит надсилається на адресу API через HTTPS, з використанням клієнтського TLS-сертифіката для автентифікації (рис. 3.13).

```
r = requests.post(
    OPENCTI_API_URL,
    headers=headers,
    json={"query": query, "variables": variables},
    verify=OPENCTI_CA_PEM
    )
```

Рисунок 3.13 – Формування параметрів фільтрації для GraphQL-запиту в OpenCTI

У разі успішного отримання відповіді, перевіряється наявність елементів у полі edges. Якщо знайдено збіг, з нього вилучаються:

- тип IoC (entity_type, наприклад IPv4-Addr, url);
- безпосередньо значення IoC (observable_value);
- пов’язані теги (objectLabel), ними можуть бути такі: cobalt strike, qilin ransomware, mozi, ransomware-as-a-service.

Уся ця інформація фіксується у логах (у форматі JSON через structlog) та включається до повідомлення, що надсилається у Telegram.

3.3.6 Локальне кешування результатів перевірки IoC

Оскільки перевірка індикаторів компрометації у базі OpenCTI потребує виконання мережевого запиту через API, при великому потоці логів або повторюваних запитах це може створювати додаткове навантаження як на мережу, так і на зовнішній сервіс. Для оптимізації цього процесу в модулі реалізовано механізм локального кешування результатів перевірки IoC.

Мета кешування полягає у зниженні частоти повторних запитів до OpenCTI у випадках, коли один і той самий індикатор зустрічається у кількох запитах протягом короткого проміжку часу. Замість багаторазової перевірки модуль використовує раніше отриманий результат протягом певного періоду часу.

Кеш реалізовано у вигляді звичайного словника ioc_cache, де ключем є значення IoC, а значенням – мітка часу останньої перевірки. Тривалість життя кешу (time to live) визначено константою. Перед кожною перевіркою IoC виконується перевірка наявності у кеші та часу останньої валідації (рис. 3. 14). Якщо з моменту останньої перевірки минуло менше ніж 30 хвилин, запит до OpenCTI не виконується, і результат не оновлюється.

```
def query_opencti(iocs):
    headers = {
        "Authorization": f"Bearer {OPENCTI_API_TOKEN}",
        "Content-Type": "application/json"
    }
    matches = []
    for ioc_type, value in iocs.items():
        if not value:
            continue
        if value in ioc_cache and (time.time() - ioc_cache[value]) < IOC_CACHE_TTL:
            continue
```

Рисунок 3.14 – Умовна перевірка часу життя індикатора у кеші

Перевагами використання такої реалізації кешування є наступні:

- Зменшення кількості API-запитів, що є досить актуальним при аналізі логів, що містять повторювані запити;
- Підвищення продуктивності модуля за рахунок зменшення часу обробки кожного лог-запису;
- Зниження навантаження на платформу OpenCTI, що важливо у випадку автоматизованого використання.

Таким чином, цей підхід дозволяє зберігати баланс між оперативністю виявлення загроз і раціональним використанням зовнішніх ресурсів, завдяки чому модуль більш ефективний у довготривалому та безперервному режимі роботи.

3.3.7 Формування повідомлень та надсилання у Telegram

Оперативне інформування про виявлену шкідливу активність є критично важливою складовою роботи програмного модуля. Для цього реалізовано інтеграцію з месенджером Telegram, що забезпечує негайне надсилання структурованого сповіщення про кожен інцидент, який виявлено модулем.

Для взаємодії з Telegram Bot API у модулі використано бібліотеку python-telegram-bot, яка забезпечує зручний та надійний спосіб надсилання повідомлень у чат [32]. Надсилання реалізовано в асинхронному режимі, що, у свою чергу, дозволяє обробляти події паралельно з моніторингом логів, не блокуючи основний цикл програми (рис. 3.15). Оскільки надсилання повідомлень здійснюється мережею та може бути потенційно повільною операцією, воно винесене в окрему асинхронну чергу (asyncio.Queue()), яка обробляється окремою задачею send_notifications() [33].

```
async def send_notifications(queue):
    while True:
        msg = await queue.get()
        try:
            await bot.send_message(chat_id=CHAT_ID, text=msg, parse_mode=telegram.constants.ParseMode.MARKDOWN)
        except telegram.error.TelegramError as e:
            print(f"Telegram error: {e}")
        queue.task_done()
```

Рисунок 3.15 – Асинхронна функція надсилання повідомлень у Telegram

Відправка повідомлення виконується виключно у випадку, якщо хоча б одна з перевірок спрацювала: було виявлено атаку за шаблоном (функція detect_attack_type()), або було ідентифіковано аномальну поведінку (функція is_anomalous_request()), або було виявлено збіг з індикатором у базі OpenCTI (функція query_opencti()). Лише у разі позитивного результату однієї з перевірок формується текст повідомлення, воно ставиться у чергу, після чого надсилається у відповідний чат.

Повідомлення, що надсилається у Telegram, має структурований вигляд, де виділено: тип виявленої атаки або аномалії, IP-адресу джерела, час відправки

запиту, HTTP-метод і URL, статус відповіді сервера, перелік знайдених IoC (за наявності), теги з OpenCTI, що асоціюються з індикаторами (за наявності).

Формат повідомлення оформлено у стилі Markdown, з використанням моноширинного форматування для ключових значень, що спрощує копіювання та подальший аналіз (рис. 3.16).

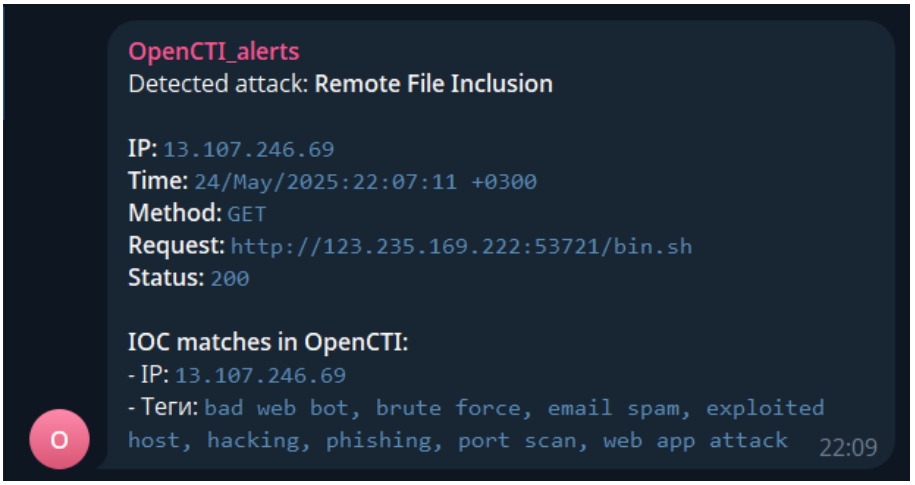


Рисунок 3.16 – Приклад сповіщення, що відправлено в Telegram, при виявленні зловмисної активності

Таким чином, реалізована система сповіщень забезпечує автоматичне, точне та своєчасне інформування про спроби атак або виявлення компрометованих артефактів, що значно підвищує швидкість реагування на можливі інциденти безпеки.

3.3.8 Підрахунок виявлених атак та періодичне формування звітності

Для аналітичних та звітних цілей у програмному модулі реалізовано механізм накопичення статистики виявлених атак, згрупованої за їх типами. Це дозволяє отримувати не лише окремі повідомлення про інциденти, а й агреговане щоденне зведення про зловмисну діяльність, помічену в роботі вебдодатку.

У модулі використовується словник attack_stats, що створюється за допомогою структури defaultdict(int) з модуля collections. Це дозволяє автоматично інкрементувати лічильник атак відповідного типу без попередньої ініціалізації ключів. При кожному виявленні атаки, незалежно від того, чи вона виявлена за шаблоном, rule-based або як IoC, відповідний елемент словника інкрементується.

Окрема асинхронна задача send_daily_attack_summary() відповідає за відправку щоденного результату виявлень в Telegram. Ця задача виконується кожні 24 години (86400 секунд) й перевіряє, чи накопичено дані в attack_stats (рис. 3.17).

```python
async def send_daily_attack_summary(interval=86400):
    while True:
        await asyncio.sleep(interval)
        if attack_stats:
            summary = "*Malicious activity during last 24h:*\n"
            summary += "\n".join(f"- {k}: `{v}` раз(ів)" for k, v in attack_stats.items())
            try:
                await bot.send_message(chat_id=CHAT_ID, text=summary, parse_mode=telegram.constants.ParseMode.MARKDOWN)
                print("[+] Daily summary sent.")
            except telegram.error.TelegramError as e:
                print(f"[!] Telegram error: {e}")
            attack_stats.clear()
```

Рисунок 3.17 – Асинхронна функція формування щоденного зведення атак

Такий щоденний звіт формується у вигляді текстового повідомлення, що групує виявлені атаки за типами. Для кожного типу вказується кількість його спрацювань за останні 24 години (рис. 3.18).

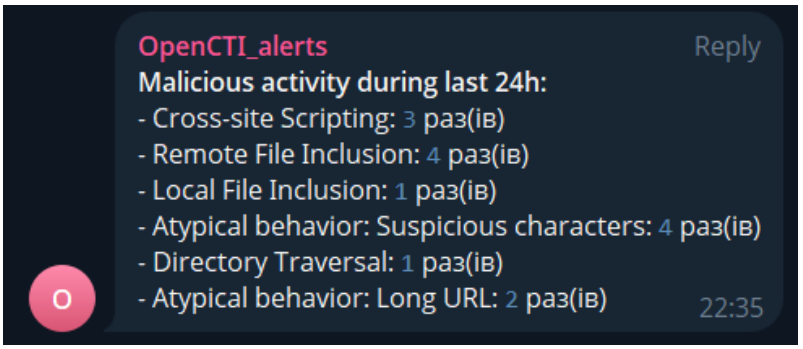


Рисунок 3.18 – Приклад повідомлення з щоденним зведенням атак у Telegram

Повідомлення надсилається лише у разі, якщо за минулу добу було зафіксовано хоча б одну спробу атаки. Якщо жодних загроз не виявлено, то зведення не формується і не надсилається, що дозволяє уникати зайвого шуму в каналі сповіщень.

Таким чином, підсумкова статистика виконує роль денного звіту про активність потенційних зловмисників. Це є зручною формою моніторингу ефективності захисту, дозволяє бачити повторювані типи атак і визначати динаміку загроз.

3.3.9. Архітектура асинхронної обробки подій

Функціонування програмного модуля базується на асинхронній архітектурі, реалізованій із використанням стандартної бібліотеки asyncio. Такий підхід дозволяє ефективно обробляти події в режимі реального часу без блокування основного потоку виконання та забезпечуючи одночасну реакцію на події в лог-файлі та відправку повідомлень у Telegram.

Ключовим елементом цієї архітектури виступає асинхронна черга asyncio.Queue, яка використовується для передачі повідомлень між різними частинами модуля. При виявленні підозрілої активності відповідне повідомлення форматується та додається до черги, незалежно від того, йдеться про атаку, виявлену за сигнатурою, rule-based перевіркою чи підтверджений IoC у базі OpenCTI. У цей момент потік обробки логів не очікує завершення надсилання, що дозволяє уникнути затримок.

В окремому асинхронному завданні (async def send_notifications) запускається нескінченний цикл, який отримує повідомлення з черги та надсилає їх користувачеві через Telegram Bot API. Всі повідомлення передаються в порядку надходження, що гарантує послідовність обробки.

Паралельно з цим працює ще одна асинхронна задача – send_daily_attack_summary, яка кожні 24 години перевіряє наявність накопиченої статистики атак та формує щоденний звіт. Таким чином,

асинхронність забезпечує незалежність та взаємну несуперечливість усіх компонентів модуля: обробки логів, взаємодії з OpenCTI, формування повідомлень та відправки результатів відбуваються паралельно, без блокування основного процесу.

Ініціалізація обох задач (send_notifications та send_daily_attack_summary) відбувається під час запуску модуля у функції main(), де обидві задачі запускаються через asyncio.create_task(...), а далі виконуються у рамках asyncio.gather(...) (рис. 3.19). Завдяки цьому, реалізовано повноцінну багатозадачність з високим рівнем продуктивності й чутливості до нових подій.

```
tasks = [
    asyncio.create_task(send_notifications(queue)),
    asyncio.create_task(send_daily_attack_summary())
]
try:
    await asyncio.gather(*tasks)
except asyncio.CancelledError:
    pass
```

Рисунок 3.19 – Створення асинхронних задач для обробки черги повідомлень та щоденного звітування

3.3.10. Запуск модуля як фонової служби

У рамках розробки програмного модуля було передбачено його безперервну роботу у фоновому режимі, без необхідності ручного запуску або втручання з боку користувача. Це забезпечується шляхом інтеграції модуля у систему ініціалізації операційної системи Linux за допомогою механізму systemd (рис. 3.20).

Systemd є стандартним менеджером системних служб для більшості сучасних дистрибутивів Linux, оскільки він дозволяє автоматично запускати програму при завантаженні системи, контролювати її життєвий цикл (перезапуск при збої, моніторинг стану), а також надає можливість зручно працювати з журналом логів (journalctl).

Фонова робота особливо важлива для програмного модуля, оскільки він повинен реагувати на події в логах у режимі реального часу й бути доступним постійно, тобто без прив'язки до активності сеансу користувача.

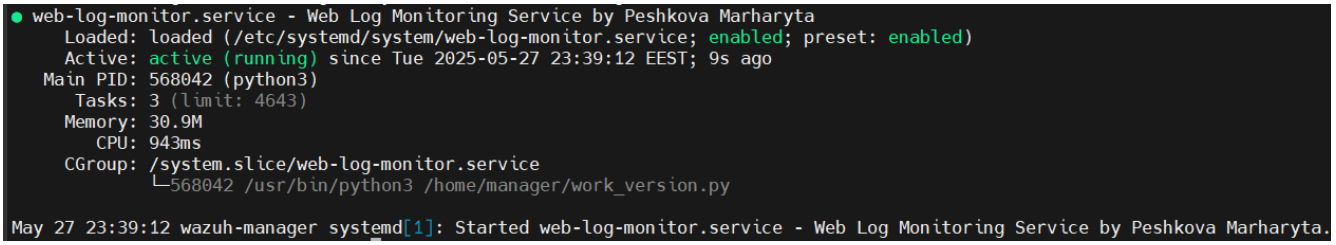


Рисунок 3.20 – Робота програмного модуля у фоновому режимі

3.3.11. Обробка винятків

У процесі роботи програмного модуля можуть виникати помилки, пов’язані як з мережею, так і з інтеграційними компонентами, такими як Telegram Bot API або GraphQL API платформи OpenCTI. З метою забезпечення надійності та стабільної роботи модуля було реалізовано обробку винятків на ключових етапах взаємодії із зовнішніми службами.

У випадку Telegram-інтеграції використовується бібліотека python-telegram-bot, яка підтримує асинхронне відправлення повідомлень. Кожна спроба надсилання обгорнута в конструкцію try-except, де перехоплюються винятки типу telegram.error.TelegramError (рис.3.21). У разі виникнення помилки (наприклад, недоступність API, неправильний chat_id, проблеми з мережею) повідомлення про виняток виводиться у консоль для подальшого аналізу, але сам модуль не припиняє виконання.

```
async def send_notifications(queue):
    while True:
        msg = await queue.get()
        try:
            await bot.send_message(chat_id=CHAT_ID, text=msg, parse_mode=telegram.constants.ParseMode.MARKDOWN)
        except telegram.error.TelegramError as e:
            print(f"Telegram error: {e}")
        queue.task_done()
```

Рисунок 3.21 – Обробка винятку при спробі надсилання повідомлення в Telegram у разі виникнення помилки на рівні API

Аналогічна методологія застосована для запитів до платформи OpenCTI через HTTP-запити бібліотеки requests. Усі звернення до API обгорнуті у try-except, де перехоплюються винятки типу requests.exceptions.RequestException (рис. 3.22). Це дозволяє модулю коректно реагувати на збої підключення, тайм-аути, проблеми з TLS-сертифікатом або некоректні відповіді сервера.

```
try:
    r = requests.post(OPENCTI_API_URL, headers=headers, json={"query": query, "variables": variables}, verify=OPENCTI_CA_PEM)
    if r.ok:
        edges = r.json().get("data", {}).get("stixCyberObservables", {}).get("edges", [])
        if edges:
            ioc_cache[value] = time.time()
            for node in [e["node"] for e in edges]:
                labels = [l["value"] for l in node.get("objectLabel", [])]
                logger.msg("Matched IoC in OpenCTI", type=node["entity_type"], value=node["observable_value"], tags=labels or None)
                matches.append(f"{ioc_type.upper()}: `{value}`")
                if labels:
                    matches.append(f"Tags: `{', '.join(labels)}`")
except requests.exceptions.RequestException as e:
    print(f"OpenCTI error: {e}")
```

Рисунок 3.22 – Обробка винятку під час запиту до OpenCTI API у разі помилки з’єднання або некоректної відповіді

Усі обробники винятків реалізовані з дотриманням принципу fail-safe, тобто у випадку збою функціональність інших частин модуля не порушується. Це означає, що навіть якщо якась інтеграція тимчасово недоступна, модуль продовжує працювати, виконуючи основну функцію – моніторинг та аналіз логів вебсервера в режимі реального часу.

Таким чином, реалізовані механізми обробки винятків дозволяють досягти високої стійкості системи до зовнішніх збоїв, забезпечуючи її безперервну роботу у продуктивному середовищі.

3.4 Тестування програмного модуля та перевірка функціональності

Для перевірки працездатності та відповідності реалізованого програмного модуля поставленим функціональним вимогам було проведено комплексне тестування з моделюванням різних сценаріїв атак на вебдодаток. Основна мета

тестування полягає в підтвердженні здатності модуля виявляти у режимі реального часу як шаблонні атаки на основі відомих сигнатур, так і аномальні запити, які виходять за межі типової поведінки. У процесі тестування також було перевірено коректність інтеграції з платформою кіберрозвідки OpenCTI, а саме: здатність модуля виокремлювати з вебзапитів потенційні індикатори компрометації (IP-адреси, URL, домени, хеші), відправляти їх на перевірку до OpenCTI, кешувати відповіді та збагачувати повідомлення про інциденти тегами. Крім того, було протестовано rule-based механізм для виявлення зловмисних параметрів запиту та функціонал відправлення сповіщень до Telegram.

Для демонстрації всіх можливостей програмного модуля було виконано наступні запити:

1. Запит, що містить зловмисні індикатори компрометації та підпадає під шаблон атаки Remote File Inclusion. RFI – цей тип атаки включає в себе спробу виконання скрипта з віддаленого сервера зловмисника, такий запит може призвести до виконання довільного коду на стороні сервера, якщо не виконується фільтрація зовнішніх URL у параметрах.

Сповіщення, що було отримане, представлено на рисунку 3.23.

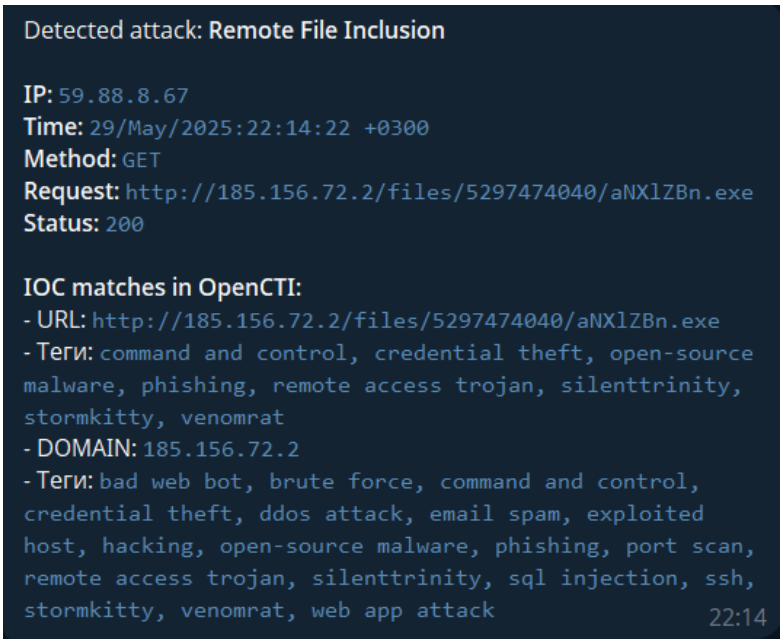


Рисунок 3.23 – Сповіщення про виявлення атаки RFI з перевіркою IoC на платформі OpenCTI

В цьому спрацювання фігурує і шаблон атаки (сигнатурне виявлення), і два індикатори компрометації: URL та домен, а також теги, що відносяться до цих IoC.

Варто зауважити, що при повторному виконанні запиту з такими ж значеннями IoC, перевірка маркерів компрометації на платформі OpenCTI не буде виконуватись повторно, оскільки це забезпечено реалізованою можливістю локального кешування. Повторивши запит, що було виконано вище, у месенджер надійде сповіщення без деталей про IoC (рис. 3.24).

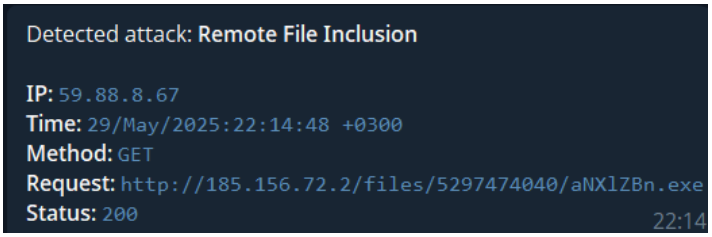


Рисунок 3.24 – Сповіщення при повторній появі IoC, що вже були перевірені

2. Запит, що не містить зловмисних індикаторів компрометації та підпадає під шаблон атаки Cross-Site Scripting, це тип атаки, коли впроваджується шкідливий JavaScript-код, що потім виконується у браузері користувача. Програмний модуль за шаблоном атаки виявив її, в надісланому запиті фігурує потенційний індикатор компрометації – це IP-адреса, але вона не міститься на платформі OpenCTI, тобто при перевірці не було виявлено співпадінь, відповідно, інформації про це немає в спрацюванні. Повідомлення, що було надіслане в месенджер, представлено на рисунку 3.25.

Рисунок 3.25 – Сповіщення про виявлення атаки XSS без входження індикаторів компрометації

3. Програмний модуль виявляє аномальну поведінку, коли відбувається спрацювання правил, що описані в програмній реалізації, це може бути занадто довга URL-адреса, нетипові символи або методи, що використовуються для звернення до вебсервера. В даному випадку було протестовано запит, де використовується нульовий байт, що є нетиповим для легітимних вебзапитів. Спрацювання, що було надіслане при виявленні аномальної поведінки, продемонстровано на рисунку 3.26.

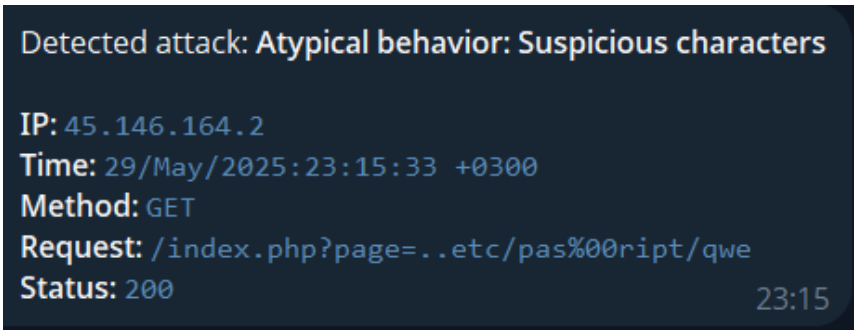


Рисунок 3.26 – Сповіщення, що було надіслане при виявленні аномальної поведінки у вебзапиті

4. Запит, що було надіслано до вебсервера, містить зловмисний індикатор компрометації, а також занадто довгу URL-адресу, що співпадає з правилом для визначення аномальної поведінки. Результат виявлення представлено на рисунку 3.27

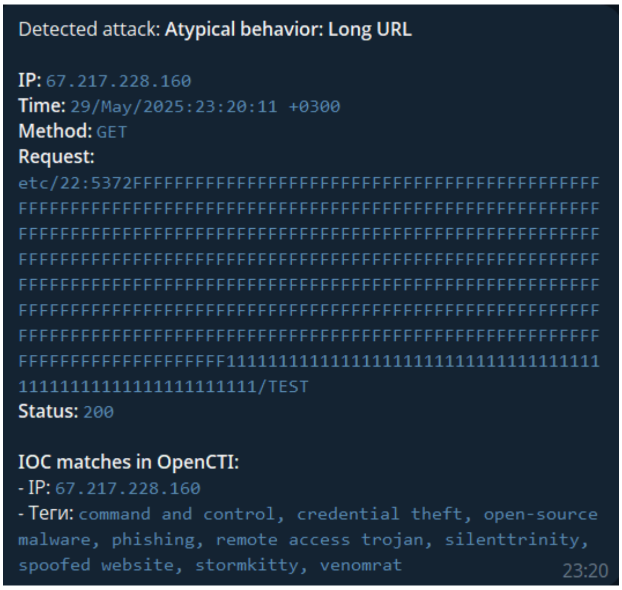


Рисунок 3.27 – Виявлення надто довгої URL-адреси в запиті і входження зловмисного індикатора компрометації

5. Вебзапити, що проходять перевірку програмним модулем та є легітимними, не співпадають з шаблонами атаки, не містять зловмисних індикаторів компрометації та аномальної поведінки в собі, ігноруються системою, не викликаючи спрацювання жодного з механізмів оповіщення.

У результаті проведеного тестування розроблений програмний модуль продемонстрував здатність ефективно виявляти різноманітні прояви потенційно шкідливої активності у запитах до вебсервера. Було успішно зафіксовано сигнатурні атаки (зокрема Remote File Inclusion, Cross-Site Scripting), а також спрацювання rule-based логіки при виявленні аномально довгих або підозрілих запитів, що містять нетипові символи. Окрему увагу приділено перевірці індикаторів компрометації через платформу OpenCTI, з можливістю виявлення тегів, пов’язаних з відомими загрозами.

Модуль виявив усі протестовані варіанти атак, коректно сформував структуровані повідомлення та надіслав їх у Telegram, що підтверджує його працездатність і відповідність визначеним функціональним вимогам. Таким чином, результати тестування підтверджують, що модуль може бути використаний як базовий інструмент первинного виявлення шкідливої активності, інтеграції з платформами кіберрозвідки та оперативного інформування відповідальних осіб про потенційні загрози.

Висновки до розділу 3

У третьому розділі було спроєктовано принцип роботи, функціональні можливості та, безпосередньо, реалізовано прикладну частину кваліфікаційної роботи, а саме створено повноцінний програмний модуль, призначений для виявлення ознак шкідливої активності в роботі вебдодатків шляхом аналізу логів вебсервера. Цей модуль є реактивним, асинхронним, розширюваним рішенням, яке поєднує класичні підходи сигнатурного виявлення атак із методами rule-based логічного аналізу та можливістю верифікації індикаторів компрометації в платформі кіберрозвідки OpenCTI.

Розроблений інструмент автоматично за написаними шаблонами виявляє поширені типи атак, такі як SQL-ін’єкції, XSS, командні ін’єкції, LFI/RFI, directory traversal та інші атаки. Окрім цього, реалізовано логіку виявлення аномальних запитів, які можуть вказувати на спроби обфускації, сканування або використання нестандартних векторів атаки, це реалізовано за допомогою rule-based підходу, що виявляє аномальну поведінку. Модуль не лише класифікує запити, а й проводить глибший аналіз вмісту запитів на предмет наявності IP-адрес, доменів, URL чи хешів, що є потенційними індикаторами компрометації. Для перевірки цих маркерів зловмисності інтегровано взаємодію з платформою OpenCTI через GraphQL API.

На відміну від класичних рішень IDS/IPS, модуль не залежить від конкретного середовища або інфраструктури, оскільки він легко інтегрується до будь-якої системи на базі Linux, не вимагає агентів на хостах, що вимагає більшість рішень, і орієнтований саме на аналіз логів вебсерверів, що зазвичай є недооціненим джерелом безпекової інформації. Сервіс працює у фоновому режимі як systemd-служба, здатна самостійно запускатися разом із системою. Окрім цього, реалізовано гнучку систему повідомлень у Telegram, яка дозволяє отримувати інформацію лише за умови фактичного виявлення підозрілої активності або підтверджених IoC, що надає можливість негайного

повідомлення відповідальних осіб про потенційну спробу втручання в нормальну роботу вебдодатку.

Особливістю рішення є поєднання кількох рівнів виявлення: сигнатурного, на основі правил та перевірки на зловмисність індикаторів компрометації, що містяться у запиті за допомогою інтеграції з платформою кіберрозвідки. Така архітектура надає змогу ефективно реагувати як на відомі атаки, так і на нові, ще не класифіковані загрози. До того ж, реалізовано локальне кешування результатів перевірки індикаторів компрометації, що суттєво знижує навантаження на зовнішні API та забезпечує швидкість обробки.

На сьогодні подібні інструменти або доволі складні в налаштуванні та інтеграції (наприклад, повноцінні SIEM або платформи на зразок Suricata з розширеним інтеграційним шаром), або не мають підтримки контекстної верифікації IoC. Відсутність простих, автономних, локально контрольованих рішень для лог-аналізу з глибокою інтеграцією в екосистему кіберзахисту робить запропонований модуль унікальним у своєму роді.

Таким чином, розроблений програмний модуль демонструє високий рівень функціональності, автономності та безпеки, що робить його придатним до використання у виробничих умовах для раннього виявлення компрометації вебресурсів. Його реалізація не лише розв'язує прикладну задачу, а й створює основу для побудови більш складних систем виявлення загроз, з можливістю інтеграції у більші захисні архітектури організацій.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та проведено тестування програмного модуля для виявлення індикаторів компрометації в роботі вебдодатків шляхом аналізу логів вебсервера у реальному часі. Реалізоване рішення дозволяє виявляти загрози на основі сигнатур відомих атак, ознак аномальної поведінки та перевірки артефактів запитів за допомогою платформи кіберрозвідки OpenCTI.

У першому розділі було проаналізовано сучасні загрози, що стосуються вебінфраструктури, охарактеризовано класифікацію атак, поняття індикаторів компрометації та роль Threat Intelligence у системах інформаційної безпеки. Також розглянуто джерела та формати поширення IoC, що створило теоретичну базу для подальшої розробки та вплинуло на вибір платформи кіберрозвідки для подальшої роботи.

У другому розділі було досліджено функціональні та архітектурні особливості платформи OpenCTI як сучасного інструменту кіберрозвідки. Описано принципи її побудови, структуру даних, механізми взаємодії між модулями, а також її застосування у контексті обробки індикаторів компрометації.

У третьому розділі було безпосередньо розроблено програмний модуль. Детально описано його архітектуру, використані підходи до обробки логів, механізми виявлення атак і аномалій, інтеграцію з Telegram та OpenCTI, асинхронну логіку роботи, кешування результатів та підрахунок статистики. Функціональність модуля було підтверджено тестуванням із використанням зловмисних запитів.

Таким чином, виконавши всі поставлені завдання, було досягнуто мету роботи – розробка та апробація програмного модуля для виявлення маркерів компрометації при атаках на вебресурси

Результати розробки та тестування програмного модуля довели здатність ефективно виявляти широкий спектр атак та забезпечувати оперативне реагування без потреби у складних інфраструктурних рішеннях. Автономна робота, спрощений запуск як системного сервісу та можливість інтеграції з платформами розвідки загроз роблять цей модуль гнучким і практичним рішенням для захисту вебресурсів.

Розроблений програмний модуль є інноваційним рішенням, яке поєднує простоту реалізації з високою ефективністю виявлення загроз. В умовах зростаючої складності атак подібні інструменти стають важливими елементами захисту. Його практична цінність полягає в здатності автономно виявляти загрози на ранніх етапах без потреби в дорогих рішеннях.

У сучасних умовах інформаційної війни, де атаки можуть починатись з одного HTTP-запиту, наявність легкого, гнучкого та надійного інструменту виявлення стає необхідністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Report on the State of Cybersecurity in the Union. [Електронний ресурс]: https://www.enisa.europa.eu/sites/default/files/2024-11/2024%20Report%20on%20the%20State%20of%20Cybersecurity%20in%20the%20Union%20-%20Condensed%20version.pdf

2. CVE Details: Product details, threats and statistics [Електронний ресурс]: https://www.cvedetails.com/product/66/Apache-Http-Server.html?vendor_id=45

3. Types of Web Application Attacks and Protecting Your Organization. [Електронний ресурс]: https://www.brightsec.com/blog/8-types-of-web-application-attacks-and-protecting-your-organization/#csrf

4. Apache 2.4 Vulnerabilities. New CVE for web servers. [Електронний ресурс]: https://httpd.apache.org/security/vulnerabilities_24.html

5. The Importance of Cyber Threat Intelligence: Insights from Recent Nobelium Attacks. [Електронний ресурс]: https://www.sans.org/blog/the-importance-of-cyber-threat-intelligence-insights-from-recent-nobelium-attacks/

6. Threat Intelligence in CyberSecurity. [Електронний ресурс]: https://cyble.com/knowledge-hub/what-is-cyber-threat-intelligence/

7. Top 10 Cyber Threat Intelligence Certifications for 2023. [Електронний ресурс]: https://flare.io/learn/resources/blog/cyber-threat-intelligence-certifications/

8. Indicators of Compromise (IOC) Security Explained. [Електронний ресурс]: https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/indicators-of-compromise-ioc/

9. Indicators of compromise. [Електронний ресурс]: https://www.cisco.com/site/us/en/learn/topics/security/what-are-indicators-of-compromise-ioc.html#:~:text=in%20the%20future.-,How%20are%20IOCs%20categorized%20in%20cybersecurity%20investigations%3F,-Network%2Dbased%20IOCs

10. Yara Rules. [Електронний ресурс]: https://github.com/Yara-Rules/rules?tab=readme-ov-file

11. Cyber threat intelligence sharing – understanding the technology. [Електронний ресурс]: https://blog.apnic.net/2016/06/24/cyber-threat-intelligence-sharing-understanding-the-technology/

12. Threat Intelligence Sharing Formats. [Електронний ресурс]: https://stixproject.github.io/about/

13. Indicators of Compromise, OpenIOC and CyBOX. [Електронний ресурс]: https://saisa.eu/blogs/Guidance/?p=1565

14. OpenCTI Alternative: MISP. [Електронний ресурс]: https://www.cosive.com/opencti-alternative

15. Пешкова М.Р., Торошанко О.С. Проблеми кібербезпеки інформаційно-комунікаційних систем (PCSICS). *Платформа OpenCTI для пошуку індикаторів компрометації* : матеріали VIII міжнар. наук.-практ. конф., м. Київ, 11 квіт. 2025. С. 13-14.

16. OpenCTI: Open-Source Cyber Threat Intelligence Platform. [Електронний ресурс]: https://www.techwrix.com/opencti-open-source-cyber-threat-intelligence-platform/

17. Sharing Response Handling Information. [Електронний ресурс]: https://sappan-project.eu/wp-content/uploads/2022/11/D5.7-Sharing-Response-Handling-Information-M21.pdf

18. OpenCTI platform performances. [Електронний ресурс]: https://medium.com/filigran/opencti-platform-performances-e3431b03f822

19. OpenCTI Documentation: Connector development. [Електронний ресурс]: https://docs.opencti.io/latest/development/connectors/

20. ElasticSearch in OpenCTI Architecture. [Електронний ресурс]: https://www.remyjaspers.com/blog/opencti_rag/

21. OpenCTI Redis usage. [Електронний ресурс]: https://filigran.io/our-design-and-development-journey-to-redis-cluster-support-in-opencti

22. High Availability Deployment for the Filigran OpenCTI. [Електронний ресурс]: http://viet.pughtml.com/posts/post-5-high-availability-deployment-filigran-opencti

23. OpenCTI Data model: STIX model. [Електронний ресурс]: https://docs.opencti.io/latest/usage/data-model/

24. Difference between STIX and TAXII. [Електронний ресурс]: https://oasis-open.github.io/cti-documentation

25. Clarifying Threat Intelligence Concepts: Threat Actors vs Intrusion Sets. [Електронний ресурс]: https://filigran.io/cti-concepts-threat-actors-vs-intrusion-sets/

26. Watchdog Documentation and Examples. [Електронний ресурс]: https://python-watchdog.readthedocs.io/en/stable/quickstart.html#a-simple-example

27. JSON Logging using structlog. [Електронний ресурс]: https://betterstack.com/community/guides/logging/json-logging/

28. Wazuh Ruleset for web attacks. [Електронний ресурс]: https://github.com/wazuh/wazuh-ruleset/blob/master/rules/0245-web_rules.xml

29. 414 URI Too Long. [Електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/414

30. How to exploit HTTP Methods. [Електронний ресурс]: https://security.stackexchange.com/questions/21413/how-to-exploit-http-methods

31. GET and POST Requests in GraphQL API using Python requests. [Електронний ресурс]: https://www.geeksforgeeks.org/get-and-post-requests-in-graphql-api-using-python-requests/

32. Building Telegram Bot with Python-Telegram-Bot: A Comprehensive Guide. [Електронний ресурс]: https://medium.com/@moraneus/building-telegram-bot-with-python-telegram-bot-a-comprehensive-guide-7e33f014dc79

33. Mastering Python’s Asyncio: A Practical Guide. [Електронний ресурс]: https://medium.com/@moraneus/mastering-pythons-asyncio-a-practical-guide-0a673265cf04

Додаток А

Таблиця А.1

Порівняльні характеристики платформ Threat Intelligence

TIP	Тип рішення	Типи індикаторів	Формати даних	Інтеграції та API	Основні переваги	Основні недоліки
OpenCTI	Opensource платформа для аналізу кіберзагроз	IP-адреси, домени, URL, хеші файлів, техніки MITRE ATT&CK, кампанії, зловмисники (threat actors)	STIX 2.1, JSON	GraphQL API, інтеграції з MISP, Shodan, VirusTotal, Elastic, TheHive та іншими ресурсами IoC	Розвинена графова модель, розширене моделювання загроз, зручна візуалізація взаємозв'язків	Складна установка та конфігурація, потребує значних ресурсів
MISP	Відкрита платформа для обміну IOC	IP-адреси, домени, хеші файлів, URL, email, сертифікати	STIX, OpenIOC, CSV, JSON, MISP XML	REST API, підтримка TAXII 2.1, інтеграція з SIEM, Suricata	Активна спільнота, висока гнучкість, підтримка таксономій	Інтерфейс менш інтуїтивний, ніж в OpenCTI, потребує додаткових налаштувань

Продовження додатку А

Продовження таблиці А.1

ThreatFox	Відкритий сервіс розповсю-дження IOC	IP-адреси, домени, URL, хеші файлів, тип загрози, рівень довіри	JSON, CSV	REST API, інтеграція з SIEM та іншими засобами через скрипти	Оперативність оновлень, простота використання фокус на актуальні шкідливі активності	Відсутність повноцінної системи аналізу або візуалізації, обмежена деталізація інформації
Комерційні TIP (Threat Connect, ThreatQ)	Комерційні рішення для управління кіберзагроза-ми	Всі типи IOC, кампанії, TTP, шкідливе ПЗ, зловмисники, вектори атак	STIX, TAXII, JSON, CSV, YARA, SNORT сигнатури	Інтеграція з SIEM, EDR, IDS/IPS, SOAR, підтримка STIX/TAXII	Централізація Threat Intelligence, автоматизація процесів, підтримка SLA (для комерційних продуктів)	Висока вартість ліцензій, можливий vendor lock-in

Додаток Б

Лістинг програмного модуля

```
import re
import os
import time
import asyncio
import telegram
import structlog
import requests
from collections import defaultdict
from watchdog.observers.polling import PollingObserver
from watchdog.events import FileSystemEventHandler

# Logging
structlog.configure(processors=[structlog.processors.JSONRenderer(indent=4)])
logger = structlog.get_logger()

# Telegram config
TELEGRAM_API_TOKEN = "**********:**********************************"
CHAT_ID = "-**********"
bot = telegram.Bot(token=TELEGRAM_API_TOKEN)

# OpenCTI config
OPENCTI_API_URL = "https://my_opencti_location/graphql"
OPENCTI_API_TOKEN = "********-****-****-****-************"
OPENCTI_CA_PEM = "/location/to/authorization/certificate"

# Apache log format
apache_log_pattern = re.compile(
    r'(?P<ip>\d+\.\d+\.\d+\.\d+) - - \[(?P<datetime>[^\]]+)\] '
    r'"(?P<method>[A-Z]+) (?P<url>[^ ]+) (?P<protocol>[^"]+)" '
    r'(?P<status>\d+) (?P<size>\d+|-)'
)

# Signatures
attack_patterns = {
    "Remote File Inclusion": r'(http|https|ftp|file|data):\/\/[^\s"]+',
    "SQL Injection": r"(\%27|\')(\s|\+)?or(\s|\+)?[^\s]+=(\s|\+)?[^\s]+|union(\s|\+)?select",
    "Command Injection": r"(;|\||&&)(\s)*(ls|cat|whoami|id|pwd|wget|curl)",
    "Code Injection": r"(base64_decode|eval|gzinflate|str_rot13|assert|preg_replace)",
    "Cross-site Scripting": r"(<script|<img|<iframe|<svg|<a|onerror=|onload=|alert\()",
    "Directory Traversal": r"(\.\./){2,}",
    "Local File Inclusion": r"(etc/passwd|boot.ini|windows/win.ini)",
    "Path Manipulation": r"(input_file=|load_file\(|document\.location)",
```

Продовження додатку Б

```
    "Open Redirect": r"(redirect=|url=)https?:\/\/[^\s]+"
}

# Cache & stats
ioc_cache = {}
IOC_CACHE_TTL = 1800
attack_stats = defaultdict(int)

def is_anomalous_request(method, url):
    if len(url) > 200:
        return "Long URL"
    if re.search(r"[<>&|%00]", url):
        return "Suspicious characters"
    if method not in {"GET", "POST", "HEAD"}:
        return "Uncommon HTTP method"
    return None

def detect_attack_type(url):
    for attack_type, pattern in attack_patterns.items():
        if re.search(pattern, url, re.IGNORECASE):
            return attack_type
    return None

def extract_iocs(log_data):
    iocs = {"ip": log_data.get("ip"), "url": None, "domain": None, "hash": None}
    url = log_data.get("url", "")
    if url:
        if "http://" in url or "https://" in url:
            iocs["url"] = url
        domain = re.search(r"https?://([^/?#]+)", url)
        if domain:
            iocs["domain"] = domain.group(1)
        hash_ = re.search(r"\b[a-fA-F0-9]{32,64}\b", url)
        if hash_:
            iocs["hash"] = hash_.group(0)
    return iocs

def query_opencti(iocs):
    headers = {
        "Authorization": f"Bearer {OPENCTI_API_TOKEN}",
        "Content-Type": "application/json"
    }
    matches = []
```

Продовження додатку Б

```
    for ioc_type, value in iocs.items():
        if not value:
            continue
        if value in ioc_cache and (time.time() - ioc_cache[value]) < IOC_CACHE_TTL:
            continue
        query = """
        query CheckObservable($filters: FilterGroup!) {
          stixCyberObservables(filters: $filters) {
            edges {
              node {
                entity_type
                observable_value
                objectLabel { value }
              }
            }
          }
        }"""
        variables = {
            "filters": {
                "mode": "and",
                "filters": [{ "key": "value", "values": [value] }],
                "filterGroups": []
            }
        }
        try:
            r = requests.post(OPENCTI_API_URL, headers=headers, json={"query": query,
"variables": variables}, verify=OPENCTI_CA_PEM)
            if r.ok:
                edges = r.json().get("data", {}).get("stixCyberObservables", {}).get("edges", [])
                if edges:
                    ioc_cache[value] = time.time()
                    for node in [e["node"] for e in edges]:
                        labels = [l["value"] for l in node.get("objectLabel", [])]
                        logger.msg("Matched IoC in OpenCTI", type=node["entity_type"],
value=node["observable_value"], tags=labels or None)
                        matches.append(f"{ioc_type.upper()}: `{value}`")
                        if labels:
                            matches.append(f"Теги: `{', '.join(labels)}`")
        except requests.exceptions.RequestException as e:
            print(f"OpenCTI error: {e}")
    return matches
```

Продовження додатку Б

```
class LogHandler(FileSystemEventHandler):
    def __init__(self, path, loop, queue):
        self.path = path
        self.loop = loop
        self.queue = queue
        self.position = os.path.getsize(path)

    def on_modified(self, event):
        if event.src_path != self.path:
            return
        with open(self.path) as f:
            f.seek(self.position)
            lines = f.readlines()
            self.position = f.tell()
            for line in lines:
                match = apache_log_pattern.match(line)
                if match:
                    log = match.groupdict()
                    url = log.get("url", "")
                    attack = detect_attack_type(url)
                    if not attack:
                        reason = is_anomalous_request(log.get("method", ""), url)
                        if reason:
                            attack = f"Atypical behavior: {reason}"
                    if attack:
                        attack_stats[attack] += 1
                        ioc_matches = query_opencti(extract_iocs(log))
                        logger.msg("Malicious activity detected", log=log, attack_type=attack)
                        msg = (
                            f"Detected attack: *{attack}*\n\n"
                            f"*IP:* `{log['ip']}`\n"
                            f"*Time:* `{log['datetime']}`\n"
                            f"*Method:* `{log['method']}`\n"
                            f"*Request:* `{log['url']}`\n"
                            f"*Status:* `{log['status']}`"
                        )
                        if ioc_matches:
                            msg += "\n\n*IoC matches in OpenCTI:*\n" + "\n".join(f"- {m}" for m in
ioc_matches)
                        self.loop.call_soon_threadsafe(self.queue.put_nowait, msg)

async def send_notifications(queue):
    while True:
```

Продовження додатку Б

```
        msg = await queue.get()
        try:
            await bot.send_message(chat_id=CHAT_ID, text=msg,
parse_mode=telegram.constants.ParseMode.MARKDOWN)
        except telegram.error.TelegramError as e:
            print(f"Telegram error: {e}")
        queue.task_done()

async def send_daily_attack_summary(interval=86400):
    while True:
        await asyncio.sleep(interval)
        if attack_stats:
            summary = "*Malicious activity during last 24h:*\n"
            summary += "\n".join(f"- {k}: `{v}` раз(ів)" for k, v in attack_stats.items())
            try:
                await bot.send_message(chat_id=CHAT_ID, text=summary,
parse_mode=telegram.constants.ParseMode.MARKDOWN)
                print("[+] Daily summary sent.")
            except telegram.error.TelegramError as e:
                print(f"[!] Telegram error: {e}")
            attack_stats.clear()

def start_watcher(path, loop, queue):
    handler = LogHandler(path, loop, queue)
    observer = PollingObserver()
    observer.schedule(handler, path=os.path.dirname(path), recursive=False)
    observer.start()
    return observer

async def main():
    path = "/path/to/monitored/apache_log_file"
    queue = asyncio.Queue()
    loop = asyncio.get_running_loop()
    observer = start_watcher(path, loop, queue)
    tasks = [
        asyncio.create_task(send_notifications(queue)),
        asyncio.create_task(send_daily_attack_summary())
    ]
    try:
        await asyncio.gather(*tasks)
    except asyncio.CancelledError:
        pass
    finally:
```

Продовження додатку Б

```
        observer.stop()
        observer.join()

if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print("Stopped.")
```

Додаток В

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Тези наукових конференцій

Пешкова М. Платформа OpenCTI для пошуку індикаторів компрометації / Маргарита Пешкова, Олександр Торошанко / VIII Міжнародна науково-практична конференція “Проблеми кібербезпеки інформаційно-телекомунікаційних систем” (PCSITS)” 11 квітня 2025 року, Київ, Україна. С 13-14.