



DOI 10.28925/2663-4023.2026.35.1426

УДК 004.056:004.49

Полотай Орест Іванович

кандидат технічних наук, доцент,
доцент кафедри управління інформаційною безпекою
Львівський державний університет безпеки життєдіяльності, Львів, Україна
ORCID: 0000-0003-4593-8601
orest.polotaj@gmail.com

Ткаченко Артур Мар'янович

викладач кафедри управління інформаційною безпекою
Львівський державний університет безпеки життєдіяльності, Львів, Україна
ORCID: 0009-0009-6830-4741
tkachenko.am14@gmail.com

Кухарська Наталія Павлівна

кандидат фізико-математичних наук, доцент,
доцент кафедри безпеки інформаційних технологій
Національний університет Львівська Політехніка, Львів, Україна
ORCID: 0000-0002-0896-8361
kukharska.n@gmail.com

Каспрівський Дмитро Володимирович

здобувач вищої освіти 4 курсу групи КБ-42,
кафедра управління інформаційною безпекою
Львівський державний університет безпеки життєдіяльності, Львів, Україна
ORCID: 0009-0006-6954-2424
dmitrokasprivskij@gmail.com

АНАЛІЗ ПРОГРАМНОГО ІНСТРУМЕНТУ THREATLENS ДЛЯ КРИМІНАЛІСТИЧНОГО-ДОСЛІДЖЕННЯ ШКІДЛИВИХ ФАЙЛІВ ТА ВИЯВЛЕННЯ ІНДИКАТОРІВ КОМПРОМЕТАЦІЇ

Анотація. У статті розглянуто питання автоматизації аналізу цифрових артефактів у задачах кібербезпеки та цифрової криміналістики. Актуальність роботи зумовлена необхідністю швидкого опрацювання значної кількості файлів, мережевих даних і журналів подій під час розслідування інцидентів інформаційної безпеки. Для вирішення цієї задачі представлено розроблений програмний засіб ThreatLens, призначений для автоматизованого статичного аналізу потенційно небезпечних цифрових артефактів без їх безпосереднього виконання. Розглянуто архітектуру та основні функціональні можливості ThreatLens, зокрема вилучення та валідацію індикаторів компрометації, аналіз PE-файлів, PCAP та журналів, визначення хешів, мережевих адрес, доменів, URL та інших характерних ознак. Реалізовано механізми визначення потенційно небезпечної поведінки з використанням правил MITRE ATT&CK, перевірки індикаторів за допомогою зовнішніх джерел Threat Intelligence та сигнатурного аналізу з використанням YARA. Додатково передбачено формування часової шкали подій, оцінювання рівня загрози, збереження результатів аналізу та побудову кореляційного графіку у веб інтерфейсі, що дає змогу встановлювати взаємозв'язки між виявленими артефактами та ознаками компрометації. Особливу увагу приділено безпечній обробці недовірених даних. Потенційно небезпечні операції ізольовано в окремому процесі з обмеженням часу виконання, а статичний характер аналізу виключає необхідність запуску досліджуваного програмного забезпечення. Визначено також основні обмеження програмного засобу, пов'язані з підтримкою окремих мережевих протоколів, обсягом YARA-сканування та можливостями перевірки цифрових підписів. Запропоновані напрями подальшого розвитку стосуються розширення підтримуваних форматів і протоколів, оптимізації аналізу великих обсягів даних та вдосконалення механізмів перевірки отриманих результатів.



Ключові слова: цифрова криміналістика, інциденти кібербезпеки, кіберзагрози, безпека програмного забезпечення, шкідливе програмне забезпечення, сигнатурний аналіз, програмування, HTTP-трафік, інформаційно-комунікаційні системи, Python, YARA, MITRE ATT&CK

ВСТУП

Сучасні інформаційно-комунікаційні технології розвиваються досить швидко, а разом із ними постійно з'являються нові види кіберзагроз. Через це виникає потреба у вдосконаленні способів їх виявлення та дослідження. Одним із важливих напрямів кібербезпеки є комп'ютерна або цифрова криміналістика (*digital forensics*) [13], яка передбачає збір, аналіз та дослідження цифрових даних для виявлення ознак несанкціонованої діяльності. Важливу роль у цьому процесі відіграє аналіз файлів, адже вони можуть містити відомості про шкідливе програмне забезпечення, особливості його роботи, способи потрапляння та закріплення в системі, а також інші ознаки її компрометації.

Для пошуку потенційно небезпечної активності часто використовують індикатори компрометації (Indicators of Compromise, IOC). До них можуть належати хеші файлів, назви та шляхи до файлів і папок, IP-адреси, доменні імена, URL-адреси, окремі фрагменти коду та інші ознаки, які можуть вказувати на взаємодію системи з підозрілим або шкідливим об'єктом. Автоматичний пошук таких індикаторів дає змогу швидше виконувати первинний аналіз цифрових даних і водночас зробити результати дослідження більш послідовними та відтворюваними.

У зв'язку з цим актуальним є використання програмних засобів, які можуть автоматично перевіряти файли та знаходити в них ознаки можливої компрометації. Одним із таких інструментів є *ThreatLens* – програмне рішення, розроблене мовою *Python* для криміналістичного аналізу файлів і пошуку в них індикаторів компрометації. Аналіз програмної реалізації *ThreatLens* дає можливість розглянути, як саме організована обробка файлів, яким чином формуються та виявляються IOC, як побудовані основні компоненти програми та які особливості має її робота.

Постановка проблеми.

Ефективне виявлення ознак компрометації інформаційних систем потребує швидкого опрацювання великої кількості різних цифрових даних. До них можуть належати виконувані файли, документи, журнали подій, мережеві дампи та інші цифрові артефакти. У таких даних можуть зустрічатися різні індикатори компрометації: IP-адреси, доменні імена, URL-адреси, хеш-значення файлів, шляхи до файлів і каталогів, ключі реєстру, адреси електронної пошти, CVE-ідентифікатори та інші ознаки, які можуть бути пов'язані з підозрілою або шкідливою активністю. Якщо виконувати пошук і перевірку таких ознак вручну, це може займати багато часу, особливо коли потрібно одночасно дослідити велику кількість файлів.

Ще однією складністю є правильна інтерпретація знайдених індикаторів. Наявність певного IOC сама по собі не завжди означає, що система була скомпрометована, оскільки один і той самий індикатор може зустрічатися і в цілком легітимному програмному забезпеченні або мережевій активності. Тому під час криміналістичного аналізу доцільно використовувати декілька способів дослідження одночасно. Зокрема, це може бути аналіз структури файлів, пошук прихованих або обфускованих рядків, перевірка PE-метаданих, дослідження мережевого трафіку, використання YARA-правил, а також пошук поведінкових ознак. Отримані результати можна додатково пов'язувати з відповідними техніками MITRE ATT&CK. У *ThreatLens* такий підхід реалізований у вигляді послідовного процесу статичного аналізу. Отримані під час нього результати використовуються не лише для пошуку окремих ознак, а й для визначення загального рівня ризику, встановлення послідовності подій та побудови взаємозв'язків між знайденими артефактами.

Разом із функціональними можливостями важливо враховувати і технічні особливості самого програмного інструменту. Результати аналізу залежать від того, наскільки правильно налаштовані правила пошуку та фільтрації хибних спрацювань. Аналогічно, можливості дослідження мережевого трафіку залежать від того, які протоколи та формати підтримує програма. Під час аналізу *ThreatLens* було виявлено декілька таких обмежень. Зокрема, модуль роботи з PCAP-файлами розрахований переважно на Ethernet II та IPv4, аналіз сирих даних за допомогою YARA має певні обмеження щодо обсягу інформації, яка перевіряється, а перевірка цифрового підпису PE-файлів фактично зводиться до визначення наявності Security Directory без повноцінної криптографічної перевірки сертифіката.

Отже, у роботі розглядається проблема визначення можливостей, принципів роботи та технічних обмежень програмного інструменту, який використовується для автоматизації криміналістичного аналізу файлів і пошуку індикаторів компрометації. Для цього досліджується архітектура *ThreatLens*, способи пошуку та перевірки IOC, механізми статичного аналізу файлів і мережевого трафіку, підходи до поведінкового аналізу та використання MITRE ATT&CK. Також розглядаються механізми оцінювання рівня загрози, об'єднання результатів різних видів аналізу та формування підсумкового звіту.

**Аналіз останніх досліджень і публікацій.**

Проблематика цифрової криміналістики та виявлення ознак компрометації інформаційних систем охоплює широкий спектр методів дослідження цифрових артефактів, мережевого трафіку, файлових об'єктів і системних даних. Одним із базових підходів залишається поєднання криміналістичного аналізу файлів, операційних систем, мережевого трафіку та прикладних даних у межах процесу реагування на інциденти. Відповідні підходи систематизовані, зокрема, у рекомендаціях NIST SP 800-86, де цифрова криміналістика розглядається як складова процесу дослідження інцидентів інформаційної безпеки [4].

Важливим напрямом сучасних досліджень є використання індикаторів компрометації (Indicators of Compromise, IOC) для виявлення та відстеження шкідливої активності. У RFC 9424 систематизовано основні типи IOC, особливості їх виявлення та використання на різних етапах протидії кіберзагрозам. При цьому наголошується, що практична цінність IOC залежить не лише від факту їх виявлення, а й від можливості надійного використання таких індикаторів у засобах моніторингу, аналізу та блокування загроз [7]. Це формує потребу в автоматизованих засобах, здатних не тільки вилучати окремі індикатори з цифрових артефактів, а й забезпечувати їх подальше структурування та інтерпретацію.

Окремий напрям становлять дослідження автоматизованого вилучення IOC із текстових та інших джерел досліджуваної інформації. Зокрема, у роботі Н. Tang та співавторів запропоновано підхід до вилучення індикаторів кібербезпеки із різноманітних джерел за допомогою використання контекстних знань. Автори звертають увагу на проблему хибних спрацьовувань та недостатньої якості отриманих результатів, що виникає під час автоматичного визначення IOC. Це свідчить про перехід від простого пошуку формальних ознак до їх контекстного аналізу [10].

Іншим важливим напрямом є підвищення достовірності та інформативності отриманих IOC. У дослідженні S. S. Chen та співавторів запропоновано використовувати графове представлення STIX для пов'язування різноманітних джерел кіберрозвідки та оцінювання таких характеристик IOC, як актуальність, повнота, точність, своєчасність і узгодженість. Такий підхід показує, що ефективна робота з IOC передбачає не лише їх вилучення, але й оцінювання контексту та взаємозв'язків між окремими індикаторами [2].

Для аналізу шкідливого програмного забезпечення широко застосовуються методи статичного аналізу, зокрема сигнатурний пошук. Одним із практичних інструментів такого аналізу є YARA, який дає змогу описувати шкідливі або характерні для певних класів програмних об'єктів за допомогою правил, що містять текстові та бінарні шаблони. Використання правил YARA забезпечує можливість автоматизованого виявлення відомих ознак шкідливого програмного забезпечення без необхідності його виконання [11]. Водночас сучасні системи аналізу поступово переходять до поєднання сигнатурних механізмів із іншими методами статичного та динамічного аналізу. Зокрема, у дослідженні SLIFER розглянуто послідовний конвеєр, що поєднує YARA-правила, статичне виявлення шкідливого програмного забезпечення та динамічний аналіз, що дає змогу зменшити навантаження на ресурсоємні етапи дослідження [8].

Значна увага в сучасних роботах приділяється представленню результатів аналізу не лише у вигляді окремих IOC, а й через поведінкові характеристики шкідливого програмного забезпечення. Для цього широко використовується база знань MITRE ATT&CK, яка забезпечує уніфіковане представлення тактик і технік, що характеризують дії зловмисника. Дослідження В. Al-Sada, А. Sadighian та G. Oligeri демонструє можливість використання MITRE ATT&CK для систематизованого аналізу та характеристики кіберзагроз на основі їх поведінкових ознак [1]. У свою чергу, у роботі Malware2ATT&CK запропоновано автоматизоване відображення ознак шкідливого програмного забезпечення на техніки MITRE ATT&CK на основі статичної інформації, API-викликів та методів машинного навчання [9].

Таким чином, аналіз сучасних досліджень показує поступовий перехід від ізольованого пошуку окремих ознак компрометації до комплексного аналізу цифрових артефактів, у якому IOC поєднуються з інформацією про поведінку програм, мережеві взаємодії, контекст загрози та взаємозв'язки між отриманими даними. Водночас значна частина наукових робіт зосереджена на окремих складових цієї задачі – автоматизованому вилученні IOC, класифікації шкідливого програмного забезпечення, зіставленні поведінки з MITRE ATT&CK або оцінюванні якості аналізованих даних [5]. Тому актуальним залишається дослідження програмних інструментів, які поєднують зазначені підходи в межах єдиного процесу статичного криміналістичного аналізу.

У цьому контексті становить практичний інтерес програмний інструмент *ThreatLens*, який поєднує вилучення IOC із аналізом файлів і мережевого трафіку, статичний аналіз, визначення поведінкових ознак та їх зіставлення з MITRE ATT&CK, оцінювання ризику, кореляцію результатів і формування структурованих звітів. Це дає змогу розглядати його не лише як засіб пошуку окремих індикаторів, а як комплексний інструмент підтримки криміналістичного дослідження цифрових артефактів.

Мета статті.



Метою дослідження є вивчення можливостей та особливостей роботи програмного інструменту *ThreatLens* під час проведення криміналістичного аналізу цифрових даних, пошуку індикаторів компрометації та визначення можливих ознак кіберзагроз. У межах дослідження увага зосереджується на тому, як інструмент виконує статичний аналіз файлів і мережевого трафіку, знаходить та обробляє ІОС, визначає поведінкові ознаки й пов'язує їх із техніками MITRE ATT&CK. Також розглядаються способи оцінювання рівня ризику, об'єднання результатів різних видів аналізу та формування підсумкової інформації за результатами криміналістичного дослідження.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Цифрова криміналістика є важливою частиною забезпечення інформаційної безпеки. Вона охоплює пошук, отримання, збереження та аналіз цифрових даних, які можуть бути пов'язані з інцидентами інформаційної безпеки. За допомогою криміналістичного дослідження можна встановити, за яких обставин стався інцидент, яким способом зловмисник отримав доступ до системи, які ресурси могли бути скомпрометовані та які цифрові дані можуть підтверджувати факт порушення безпеки. Оскільки значна кількість сучасних інцидентів інформаційної безпеки пов'язана зі шкідливим програмним забезпеченням, його пошук і дослідження є одним із важливих завдань цифрової криміналістики.

Шкідливі програми можуть використовуватися для отримання доступу до інформаційних ресурсів без дозволу, викрадення чи зміни даних, порушення роботи систем, прихованого керування пристроями та виконання інших небажаних дій. Сучасне шкідливе програмне забезпечення при цьому може мати досить різноманітну функціональність, а також використовувати різні способи приховування своєї присутності [12]. Через це під час аналізу необхідно застосовувати такі методи, які дають змогу швидко отримати початкову інформацію про досліджуваний файл і визначити, чи може він бути пов'язаний із відомими шкідливими програмами.

Одним із найпоширеніших способів такого первинного аналізу є сигнатурний підхід [14]. Його основна ідея полягає в тому, щоб порівняти досліджуваний файл із набором заздалегідь визначених ознак, які характерні для відомих зразків шкідливого програмного забезпечення. У ролі таких ознак можуть використовуватися хеші файлів, окремі фрагменти байтового коду, характерні послідовності команд, текстові рядки, назви файлів, особливості структури виконуваних PE-файлів та інші характеристики. Якщо знайдено відповідність із відомою сигнатурою, отриманий результат може стати підставою для подальшої перевірки та детальнішого аналізу об'єкта.

Однією з переваг сигнатурного аналізу є його відносна простота та швидкість. Він дає змогу перевіряти велику кількість файлів без необхідності запускати кожен із них. Це відрізняє його, наприклад, від поведінкового аналізу, під час якого програму потрібно виконувати в спеціальному або ізольованому середовищі та спостерігати за її діями. Для цифрової криміналістики така можливість є досить корисною, особливо на початковому етапі дослідження, коли серед великої кількості цифрових артефактів потрібно швидко знайти об'єкти, які можуть становити загрозу [3].

Разом із тим сигнатурний аналіз не може гарантувати виявлення всіх шкідливих програм. Його результат залежить від того, чи є відповідна ознака в базі сигнатур і чи можна коректно встановити її відповідність досліджуваному об'єкту. Якщо файл був змінений, перепакований або обфускований, його характеристики можуть відрізнятися від відомого зразка. У такому випадку звичайне порівняння сигнатур може не дати позитивного результату. Тому цей метод доцільно використовувати як частину комплексного аналізу, а не як єдиний спосіб визначення шкідливості файлу [6].

У цифровій криміналістиці результати сигнатурної перевірки можуть бути відправною точкою для подальшого дослідження. Наприклад, після виявлення відповідності певній сигнатурі можна додатково перевірити структуру файлу, його метадані, мережеві з'єднання, активність процесів та інші пов'язані артефакти. Таким чином, сигнатурний аналіз фактично допомагає відсіяти частину об'єктів і зосередити увагу спеціаліста на тих файлах, які мають потенційно небезпечні ознаки.

Особливо корисною автоматизація такого процесу стає під час аналізу інформаційно-комунікаційних систем та мереж. У сучасних мережах одночасно функціонує велика кількість пристроїв, передається значний обсяг даних і постійно створюються нові цифрові артефакти. Перевіряти всі ці дані вручну досить складно та довго. Використання автоматизованих засобів дає змогу швидше вилучати необхідні ознаки, порівнювати їх із відомими характеристиками та передавати отримані результати фахівцю для подальшого дослідження.

Таким чином, сигнатурний аналіз можна розглядати як один із базових методів первинного дослідження цифрових артефактів. Цей метод аналізу ґрунтується на пошуку характерних ознак шкідливого програмного забезпечення та їх порівнянні з уже відомими шаблонами. Основною практичною перевагою такого підходу є можливість швидко перевірити значну кількість об'єктів і визначити ті з них,

які потребують більш детального криміналістичного аналізу. Водночас для отримання більш надійного результату сигнатурний метод доцільно поєднувати з іншими методами дослідження, зокрема статичним, поведінковим і мережевим аналізом.

В таблиці 1 наведено порівняння методів виявлення шкідливих програм.

Таблиця 1

Порівняння основних методів виявлення шкідливих програм

Метод виявлення	Принцип роботи	Переваги	Недоліки
Сигнатурний	Порівняння з базою відомих сигнатур	Висока точність для відомих загроз	Не виявляє нові загрози
Евристичний	Аналіз підозрілих структур коду	Виявлення нових модифікацій	Можливі хибні спрацювання
Поведінковий	Моніторинг дій програми	Виявлення zero-day атак	Високе навантаження
Sandbox	Запуск у ізольованому середовищі	Безпечний аналіз	Потребує ресурсів
ML/AI	Статистичний та аномальний аналіз	Автоматизація, адаптивність	Залежність від якості навчання

Рисунок 1 показує загальний алгоритм роботи системи виявлення шкідливих програм.



Рис. 1. Схема роботи системи виявлення шкідливих програм

На початку аналізу система отримує підозрілий файл або процес, після чого він послідовно перевіряється за кількома напрямками. Спочатку виконується сигнатурний аналіз, наступним кроком є евристична перевірка та моніторинг поведінки. Якщо отриманих даних недостатньо для прийняття рішення, об'єкт може бути додатково запущений в ізольованому середовищі, так званий пісочниці. За результатами всіх перевірок система визначає подальшу дію: заблокувати об'єкт, помістити його в ізоляцію або дозволити виконання. Поєднання декількох етапів перевірки дає змогу точніше виявляти потенційно небезпечні об'єкти та зменшувати ризик їх негативного впливу на систему.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У межах дослідження розроблено програмний інструмент *ThreatLens*, призначений для автоматизованого криміналістичного аналізу цифрових файлів та первинного виявлення ознак, пов'язаних із потенційною шкідливою активністю. Інструмент реалізовано у вигляді командного інтерфейсу та веб-API, що забезпечує можливість його використання як у ручному режимі під час дослідження окремих об'єктів, так і в складі автоматизованих процесів аналізу інформаційно-комунікаційних систем.



Функціональні можливості *ThreatLens* не обмежуються одним типом цифрових об'єктів. Інструмент підтримує аналіз виконуваних файлів формату PE, дамів мережевого трафіку PCAP, системних та прикладних журналів, документів та інших цифрових артефактів. Основним завданням такого аналізу є автоматизоване вилучення ІОС, виявлення ознак, характерних для потенційно шкідливої активності, та формування структурованого результату дослідження.

Одним із функціональних компонентів *ThreatLens* є виявлення та систематизація індикаторів компрометації. Залежно від типу досліджуваного об'єкта до таких індикаторів можуть належати хеш-значення, IP-адреси, доменні імена, URL-адреси, імена файлів, шляхи до об'єктів файлової системи, характерні текстові рядки та інші ознаки, які можуть бути використані під час подальшого розслідування інциденту. Виявлені ознаки структуруються та включаються до результатів аналізу, що спрощує їх подальше використання засобами моніторингу та реагування на кіберінциденти.

Окремий напрям аналізу становить виявлення поведінкових ознак, пов'язаних із відомими тактиками та техніками кіберзагроз. Для їх систематизації використовується модель MITRE ATT&CK, що дає змогу співвідносити виявлені ознаки з відповідними техніками та тактиками, застосовуваними під час реалізації кіберзагроз. Такий підхід забезпечує більш структуроване представлення результатів аналізу та дає змогу розглядати окремі виявлені ознаки не ізольовано, а в контексті потенційного сценарію атаки.

Для узагальненої оцінки результатів аналізу в *ThreatLens* передбачено розрахунок числового показника ризику (risk score). Його використання дає змогу представити сукупність виявлених ознак у компактній формі та використовувати отримане значення для первинного сортування або пріоритизації досліджуваних об'єктів. При цьому числовий показник розглядається як результат автоматизованої оцінки сукупності встановлених характеристик, а не як самостійна підстава для остаточного висновку щодо шкідливості об'єкта.

Ще однією функціональною можливістю інструмента є формування часової послідовності виявлених подій та побудова графа взаємозв'язків між цифровими артефактами. Хронологічне представлення результатів спрямоване на відтворення послідовності пов'язаних із досліджуваним об'єктом подій, тоді як графове представлення дає змогу візуалізувати зв'язки між файлами, мережевими адресами, доменами, процесами та іншими виявленими артефактами. Це створює додаткові можливості для встановлення взаємозалежностей між окремими ознаками під час подальшого криміналістичного дослідження.

Результати роботи *ThreatLens* можуть експортуватися у структурованому форматі JSON, придатному для подальшої автоматизованої обробки, а також у вигляді HTML-звіту з інтерактивним представленням результатів аналізу. Такий підхід забезпечує одночасно машинну обробку отриманих даних та їх наочне представлення для фахівця, який проводить дослідження. HTML-звіт, зокрема, забезпечує відображення виявлених індикаторів, оцінки ризику, пов'язаних поведінкових ознак, хронології та графа взаємозв'язків у єдиному інформаційному просторі.

За принципом роботи *ThreatLens* належить до класу засобів статичного вилучення індикаторів. На відміну від підходів, що передбачають виконання досліджуваного програмного коду в ізольованому середовищі, інструмент *ThreatLens* здійснює аналіз цифрових даних без запуску досліджуваного виконуваного файлу. Це дає змогу використовувати його на початковому етапі дослідження підозрілих об'єктів для отримання первинної сукупності ознак та визначення напрямів подальшого аналізу.

Таким чином, розроблений *ThreatLens* поєднує функції статичного аналізу цифрових артефактів, вилучення ІОС, зіставлення виявлених ознак із моделлю MITRE ATT&CK, оцінювання рівня ризику, побудови часової послідовності та графа взаємозв'язків. Поєднання зазначених функцій в одному інструменті створює основу для автоматизації етапу первинного первинний швидкий аналіз цифрових даних та формування структурованих результатів, які можуть використовуватися під час подальшого дослідження інцидентів інформаційної безпеки.

На рисунку 2 показано структуру репозиторію розробленого програмного продукту.

```
ioc2/
├── threatlens/.....# основний пакет Python
├── cli.py.....# точка входу командного рядка
├── web.py.....# FastAPI веб-сервіс
├── extractor.py.....# regex-екстрактор IOC з тексту/файлів
├── file_analyzer.py.....# хеші, рядки, PE-парсер, декодування
├── pcap_analyzer.py.....# парсер PCAP (Ethernet/IPv4/TCP/UDP/DNS/HTTP/TLS-SNI)
├── static_analysis.py.....# об'єднаний конвеєр статичного аналізу
├── behavior.py.....# поведінкові правила -> MITRE ATT&CK
├── intelligence.py.....# інтерфейс провайдерів Threat Intelligence
├── providers/.....# адаптери VirusTotal, AbuseIPDB, URLScan
├── risk.py.....# обчислення risk_score і threat_level
├── timeline.py.....# реконструкція хронології подій
├── correlation.py.....# граф зв'язків + інтерактивна SVG-карта
├── reporting.py.....# генерація HTML-звіту (дашборд)
├── storage.py.....# збереження історії сканувань у SQLite
├── sandbox.py.....# ізоляція аналізу в окремому процесі
├── yara_scanner.py.....# опційна інтеграція з yara-python
├── models.py.....# dataclass-моделі IOC / ScanResult
├── rules/suspicious_commands.var.....# приклад YARA-правил
├── tests/.....# 11-файлів unittest, 569-рядків тестів
└── pyproject.toml.....# опис пакету, залежності 'web' (fastapi/uvicorn)
```

Рис. 2. Структура репозиторію

Розроблений програмний засіб *ThreatLens* реалізовано мовою програмування *Python* із модульною структурою, що передбачає функціональне розділення компонентів аналізу, оброблення результатів, оцінювання ризику та формування звітності. Така організація програмного коду забезпечує можливість незалежного розвитку окремих функціональних компонентів і спрощує їх подальше тестування та розширення.

Основні програмні компоненти системи зосереджено в пакеті *threatlens*. Точкою входу для командного режиму роботи є модуль *cli.py*, який забезпечує взаємодію користувача з функціями аналізу через інтерфейс командного рядка. Для реалізації програмного доступу до функцій системи передбачено модуль *web.py*, побудований на основі *FastAPI*, що забезпечує функціонування веб-API та можливість інтеграції *ThreatLens* з іншими інформаційними системами.

Первинне вилучення індикаторів компрометації реалізовано в модулі *extractor.py*. Він призначений для пошуку характерних IOC у текстових даних та вмісті досліджуваних файлів із використанням регулярних виразів. Зокрема, такий підхід дає змогу виявляти IP-адреси, доменні імена, URL-адреси та інші текстові ознаки, які можуть мати значення під час криміналістичного дослідження.

Безпосереднє дослідження файлових об'єктів виконується засобами модуля *file_analyzer.py*. Він забезпечує обчислення хеш-значень, вилучення текстових рядків, аналіз структури виконуваних файлів формату PE та виконання передбачених засобом операцій декодування. Для дослідження мережових артефактів використовується окремий модуль *pcap_analyzer.py*, призначений для аналізу дамів мережевого трафіку PCAP. Його функціональність охоплює оброблення пакетів Ethernet, IPv4, TCP, UDP, DNS, HTTP та TLS SNI, що дає змогу вилучати мережеві індикатори та пов'язані з ними характеристики.

Об'єднання окремих напрямів статичного аналізу здійснюється модулем *static_analysis.py*, який формує загальний конвеєр оброблення досліджуваного об'єкта. На цьому етапі результати окремих процедур аналізу інтегруються для подальшого визначення поведінкових ознак та оцінювання ризику.

Для інтерпретації виявлених ознак використовується модуль *behavior.py*, у якому реалізовано набір правил зіставлення встановлених характеристик із техніками та тактиками MITRE ATT&CK. Це дає змогу перейти від простого переліку виявлених ознак до їх структурованого представлення з урахуванням можливого зв'язку з відомими сценаріями шкідливої активності.

Отримання додаткової інформації з зовнішніх джерел кіберрозвідки реалізовано через модуль *intelligence.py* та каталог *providers*. Останній містить адаптери для взаємодії з такими сервісами Threat Intelligence, як VirusTotal, AbuseIPDB та URLScan. Винесення інтеграцій із зовнішніми провайдерами в окремий функціональний компонент дає змогу відокремити основну логіку аналізу від механізмів отримання зовнішніх репутаційних та контекстних даних.

Оцінювання результатів аналізу виконується модулем *risk.py*, який відповідає за розрахунок числового показника *risk_score* та визначення відповідного рівня загрози (*threat_level*). Отримане значення використовується для узагальнення сукупності виявлених ознак та подальшої пріоритизації результатів дослідження.

Для реконструкції часової послідовності пов'язаних подій передбачено модуль *timeline.py*. Він забезпечує формування хронологічного представлення виявлених артефактів та подій. Доповненням до часової моделі є модуль *correlation.py*, який відповідає за встановлення взаємозв'язків між виявленими об'єктами та формування графа зв'язків. Результати кореляційного аналізу можуть бути представлені у



вигляді інтерактивної SVG-карти, що полегшує візуальне дослідження взаємопов'язаних цифрових артефактів.

Формування підсумкових результатів здійснюється модулем `reporting.py`. Він відповідає за генерацію HTML-звіту, на якому можуть бути представлені виявлені IOC, результати поведінкового аналізу, оцінка ризику, хронологія подій та граф взаємозв'язків.

Для збереження результатів попередніх досліджень використовується модуль `storage.py`, який реалізує роботу з локальною базою даних SQLite та забезпечує збереження історії сканувань. Це створює можливість повторного перегляду результатів та накопичення даних про проведені дослідження.

Окремим компонентом є `sandbox.py`, призначений для ізоляції процедур аналізу в окремому процесі. Його використання дає змогу відокремити виконання окремих операцій аналізу від основного процесу програми. Водночас *ThreatLens* у базовому сценарії не здійснює динамічне виконання досліджуваного шкідливого коду в повноцінному ізольованому середовищі, тому цей компонент не слід ототожнювати з класичною sandbox-системою.

Додатковий механізм сигнатурного аналізу реалізовано в модулі `yara_scanner.py`, який забезпечує інтеграцію з бібліотекою `yara-python`. При цьому набір прикладних правил міститься в каталозі `rules`, зокрема у файлі `suspicious_commands.yar`. Такий компонент розширює можливості статичного аналізу шляхом використання формалізованих правил пошуку характерних ознак у досліджуваних об'єктах.

Структури даних, що використовуються для представлення індикаторів компрометації та узагальнених результатів сканування, визначено в модулі `models.py`. Використання окремих моделей IOC та *ScanResult* забезпечує уніфіковане представлення результатів між різними компонентами системи та спрощує їх подальшу передачу до модулів оцінювання, кореляції та звітності.

Окрему частину проєкту становить каталог `tests`, у якому реалізовано 11 файлів модульних тестів загальним обсягом 569 рядків. Наявність автоматизованих тестів дає змогу перевіряти коректність роботи окремих функціональних компонентів та зменшує ризик виникнення регресій під час подальшої модифікації програмного засобу. Параметри пакування проєкту, його залежності та конфігурацію визначено у файлі `pyproject.toml`, зокрема залежності, необхідні для функціонування веб-компонента на основі FastAPI та Uvicorn.

Таким чином, структура *ThreatLens* побудована за модульним принципом і охоплює послідовні етапи криміналістичного дослідження: отримання цифрового об'єкта, вилучення первинних ознак, статичний аналіз, ідентифікацію поведінкових характеристик, збагачення результатів зовнішньою інформацією, оцінювання ризику, кореляцію артефактів, збереження результатів та формування звітності. Функціональне розділення компонентів дає змогу використовувати *ThreatLens* як окремий інструмент командного рядка, веб-сервіс або компонент автоматизованого конвеєра аналізу кіберінцидентів.

Для забезпечення первинного криміналістичного аналізу в *ThreatLens* реалізовано механізм автоматизованого виявлення та класифікації індикаторів компрометації різних типів. Їх пошук здійснюється на основі регулярних виразів із подальшою перевіркою синтаксичної та контекстної коректності виявлених значень. Такий підхід дає змогу зменшити кількість хибних спрацювань, що можуть виникати під час аналізу необробленого тексту, дамів пам'яті, журналів або інших цифрових артефактів. Основні типи індикаторів, які розпізнаються системою, приклади їх представлення та особливості валідації наведено в табл. 2.

Таблиця 2

Типи індикаторів, які розпізнаються

Тип	Приклад	Особливість валідації
url	<code>https://evil.example/payload.exe</code>	Перевіряється через <code>urlsplit()</code> ; очищується від кінцевої пунктуації, обрізаних доменних зон (<code>crt/cer/com...</code>) артефактами OCR/дампу пам'яті
email	<code>user@example.com</code>	Стандартний RFC-подібний патерн
sha256 / sha1 / md5	64/40/32 hex-символи	Негативний lookahead/lookbehind, щоб не 'відкусити' частину довшого хеша
cve	<code>CVE-2026-12345</code>	Формат CVE-PIK-HOMEP(4-7 цифр)
mac	<code>00:11:22:33:44:55</code>	6 груп по 2 hex через ':' або '-'



Тип	Приклад	Особливість валідації
crypto_wallet	0x..., 1..., bc1...	Ethereum (hex40), Bitcoin legacy (Base58) та Bech32
file_path	C:\Users\Public\payload.exe	Windows- та Unix-шляхи; фільтрується не-ASCII/не-друковані
registry_key	HKLM\Software\...\Run	Префікси HKLM/HKCU/HKCR/HKU/HKCC
ip / ipv6	8.8.8.8, 2001:4860:4860::8888	Через <code>ipaddress.ip_address()</code> ; приватні й loopback адреси відкидаються
domain	evil.example	Білий список поширених TLD + чорний список 'не-доменних' розширень файлів (.exe, .dll, .json...)

За результатами аналізу наведених у табл. 2 типів індикаторів можна зазначити, що *ThreatLens* використовує диференційований підхід до їх виявлення та валідації залежно від семантики конкретного типу даних. Для IP-адрес, URL та доменів передбачена додаткова перевірка коректності значень, для хешів – контроль довжини та допустимого набору символів, а для спеціалізованих об'єктів, таких як CVE, MAC-адреси, криптовалютні адреси та ключі реєстру Windows, застосовуються специфічні структурні ознаки. Поєднання регулярного пошуку з подальшою валідацією дає змогу підвищити точність вилучення IOC та сформувати більш надійну основу для наступних етапів кореляції, оцінювання ризику й криміналістичного дослідження.

Модуль *behavior.py*, який є елементом репозиторію, містить невеликий, але змістовний набір regex-правил (таблиця 3), кожне з яких прив'язане до конкретної техніки MITRE ATT&CK. Це дає змогу перекласти 'сирі' знахідки (рядки в файлі, HTTP-події з PCAP) на мову, зрозумілу SOC-аналітику.

Таблиця 3

Набір regex-правил

ID правила	Опис	Рівень	MITRE ID / назва
powershell	PowerShell з прапорцем <code>-enc/-encodedcommand</code>	HIGH	T1059.001 – PowerShell
command-shell	<code>cmd.exe /c</code> або <code>command.com /c</code>	MEDIUM	T1059.003 – Windows Command Shell
download	<code>curl/wget/bitsadmin/certutil</code> із завантаженням по <code>http(s)</code>	MEDIUM	T1105 – Ingress Tool Transfer
credential-dumping	<code>mimikatz</code> , <code>sekurlsa::</code> , <code>lsadump::</code>	HIGH	T1003 – OS Credential Dumping
rundll32	виклик <code>rundll32.exe</code> (прокси-виконання підписаним бінарним)	HIGH	T1218.011 – Rundll32
registry	<code>reg add/delete/save/import</code>	MEDIUM	T1112 – Modify Registry
payload-delivery-url	URL з ≥ 3 сегментами шляху і <code>query-рядком</code> , або підтверджене HTTP-завантаження бінарного вмісту з PCAP	HIGH	T1105 – Ingress Tool Transfer

Показово, що *analyze_iocs()* поєднує дві евристики: (1) прямий доказ з мережевого рівня – якщо в PCAP зафіксовано `http_download` із `payload_candidate=True`, це відразу дає HIGH-детект з деталями (`host/uri/статус/Content-Type/розмір`); (2) евристику за формою самого URL – якщо в статично витягнутому URL є щонайменше 3 сегменти шляху та `query-рядок` (типовий патерн `'download.php?id=...&file=...'`), це також позначається як підозріла URL-доставка *payload*.



Підсумковий числовий показник (0–100) складається з кількох незалежно обмежених доданків, що не дає одному фактору 'задавити' решту:

- IOC-складова (макс. 35 балів): сума вагових коефіцієнтів за типом IOC (наприклад CVE важить 12, URL – 5, домен/IP – 4, email – 1), помножена на confidence/100.
- Поведінкова складова (макс. 55 балів): HIGH-детект = 15, MEDIUM = 10, LOW = 5 балів кожен.
- YARA-складова (макс. 30 балів): 15 балів за кожен окремий збіг правила.
- Threat Intelligence складова (без явного капю, але фінальний score обрізається до 100): +20 і бонус за впевненість, якщо знахідка позначена malicious; +12 і менший бонус, якщо suspicious.

Підсумковий бал вказує на один з 5 рівнів загрози (таблиця 4).

Таблиця 4

Рівні загрози

Поріг балів	Рівень
81–100	CRITICAL
61–80	HIGH
41–60	MEDIUM
21–40	MODERATE
0–20	LOW

Модуль *intelligence.py* реалізує уніфікований механізм взаємодії з джерелами *Threat Intelligence* на основі протоколу *IntelProvider*, визначеного засобами структурної типізації *Protocol*. Протокол містить єдиний метод *lookup(value, ioc_type)*, який повертає результат типу *ThreatIntelFinding* або *None*. Така абстракція дає змогу використовувати однаковий механізм оброблення як локальних, так і зовнішніх джерел кіберрозвідки. Для офлайн-режиму передбачено *JsonIntelProvider*, що отримує вердикти з локального JSON-файлу. Онлайн-збагачення реалізовано через адаптери до VirusTotal, AbuseIPDB та URLScan.io. VirusTotal використовується для перевірки IP-адрес, доменів, URL та хешів MD5/SHA1/SHA256 з урахуванням статистики останнього аналізу; AbuseIPDB здійснює репутаційну оцінку IP-адрес на основі *abuseConfidenceScore*; URLScan.io використовується для пошуку раніше просканиваних URL і доменів та отримання відповідних вердиктів. Ключі доступу до зовнішніх сервісів не зберігаються у програмному коді, а передаються через змінні середовища, що зменшує ризик їх ненавмисного розкриття.

Для виконання HTTP-запитів використовується стандартний модуль *urllib*, без залучення додаткових HTTP-клієнтів. Важливою особливістю реалізації є ізоляція помилок зовнішніх сервісів: винятки, пов'язані з помилками HTTP-запитів або мережевим з'єднанням, перехоплюються та перетворюються на внутрішній *RuntimeError*, не перериваючи загальний процес сканування. Крім того, результати запитів кешуються за комбінацією «провайдер – тип IOC – значення», що запобігає повторному виконанню однакових мережевих запитів. Окремим механізмом розширення статичного аналізу є модуль *yara_scanner.py*, який забезпечує опційну інтеграцію з YARA. Передбачено два режими сканування: безпосередній аналіз файлу з обмеженням часу виконання та криміналістичний режим, у якому досліджуються як сирі байти файлу, так і декодовані текстові рядки. Останній підхід дає змогу виявляти сигнатури навіть у випадках, коли характерні рядки попередньо закодовані. Для великих файлів аналіз сирих байтів обмежується першими 4 MiB. Автоматичний вибір вбудованих YARA-правил здійснюється залежно від типу досліджуваного об'єкта: правила з *suspicious_commands.yar* застосовуються до PE-файлів, а також до файлів журналів та текстових файлів, де пошук характерних команд має практичне значення.

Модуль *correlation.py* відповідає за кореляцію результатів криміналістичного аналізу та їх представлення у вигляді графа взаємозв'язків. Граф формується на основі всіх основних артефактів, отриманих під час сканування, зокрема досліджуваного файлу, виявлених індикаторів компрометації, відповідних технік MITRE ATT&CK, результатів зовнішнього *Threat Intelligence*, спрацювань YARA-правил та доменів, вилучених із URL-адрес. Кожному елементу графа призначається унікальний ідентифікатор, сформований відповідно до його типу та значення, наприклад *ioc:domain:evil.example*, *mitre:T1105* або *yara:Suspicious_PowerShell*. Додатково для вузлів зберігаються атрибути, що характеризують підстави їх виявлення, рівень впевненості та критичність. Ребра графа відображають



встановлені зв'язки між окремими артефактами, що дає змогу перейти від ізольованого переліку IOC до структурованого представлення взаємопов'язаних ознак досліджуваного об'єкта.

Для візуального представлення сформованого графа реалізовано функцію *render_threat_map()*, яка генерує автономний HTML-компонент із використанням SVG, CSS та вбудованого *JavaScript*-коду. Розміщення вузлів здійснюється за допомогою власної реалізації силового алгоритму, що моделює взаємне відштовхування вузлів, пружинні зв'язки між ними та тяжіння до центральної області; розрахунок виконується протягом 260 ітерацій без використання зовнішніх бібліотек візуалізації, таких як *D3.js*. Інтерактивність карти передбачає масштабування в діапазоні 0,45–2,8×, панорування робочої області та перетягування окремих вузлів із використанням Pointer Events. При виборі вузла користувач отримує доступ до його метаданих, зокрема типу, доказової ознаки, рівня впевненості та кількості встановлених зв'язків; при наведенні курсора додатково відображається текстова підказка щодо призначення відповідного типу артефакту. Для розмежування категорій використовується кольорова легенда, яка виділяє файли, IOC, результати *Threat Intelligence*, техніки MITRE ATT&CK та YARA-правила. З метою збереження читабельності візуалізації кількість представлених на карті вузлів обмежено 48 найбільш значущими елементами з урахуванням їх типу та пріоритету. Водночас повний набір даних, включно з хешами файлів і локальними шляхами, зберігається у структурованому JSON-результаті, але не відображається на графі, щоб уникнути його перевантаження другорядними деталями.

На рис. 2 показано HTML-звіт, згенерований самим інструментом ThreatLens після сканування справжнього PCAP-файлу з мережевим трафіком, що постачається разом з проектом (2021-09-10-traffic-analysis-exercise.pcap, звіт real-report6.html). Для експерименту було обрано шкідливий файл, який був реалізований у 2021 році. Аналогічно можна брати для дослідження будь які файли. Це показовий приклад роботи модуля *pcap_analyzer.py*: захоплення на 9221 пакет і понад 1.28 години тривалості дало 189 витягнутих IOC. В результаті можна побачити, що існує реальна поведінкова MITRE-детекція, оскільки в досліджуваному трафіку виявлено завантаження справжнього виконуваного PE-файлу через HTTP зв'язок.

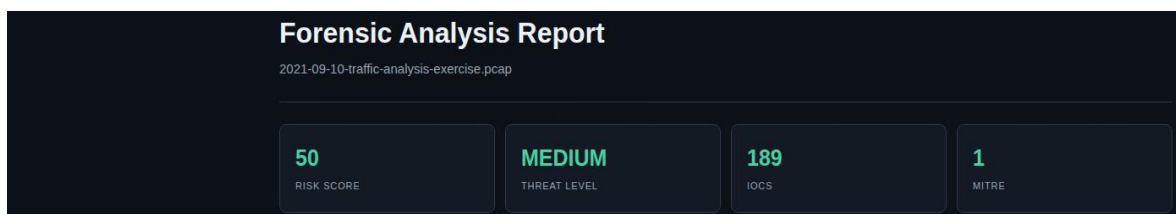


Рис. 2. Заголовок звіту і загальна статистика

На наступному рисунку показано результати MITRE ATT&CK детекції (рис. 3).

MITRE ATT&CK Detections		
SEVERITY	TECHNIQUE	DETECTION
HIGH	T1105	Executable payload delivered over HTTP

Рис. 3. Секція MITRE ATT&CK Detections: HIGH-детект T1105

На рисунку 4 показано результати IOC Inventory: службові адреси (224.0.0.x, 239.255.255.250), публічні IP та легітимні на вигляд домени Microsoft-інфраструктури (msftncsi.com, windows.com, openote.net, msn.com). Це є типовим «фоновим шумом» корпоративного трафіку поруч зі шкідливою активністю.



IOC Inventory	
TYPE	VALUE
ip	224.0.0.251
ip	224.0.0.252
ip	239.255.255.250
ip	96.17.68.66
domain	www.msftncsi.com
http_uri	/ncsi.txt
url	http://www.msftncsi.com/ncsi.txt
ip	40.83.240.146
domain	client.wns.windows.com
ip	23.3.84.67
domain	fs.microsoft.com
ip	184.85.229.68
domain	cdn.onenote.net
ip	204.79.197.203
domain	api.msn.com
ip	13.107.21.200
ip	13.107.246.254
ip	51.137.102.183
ip	13.107.49.254

Рис. 4. IOC Inventory (фрагмент з 189 записів)

Підсумковою звітною інформацією за результатами аналізу є інтерактивна карта загроз (48 вузлів, з MITRE-вузлом) (рис. 5).

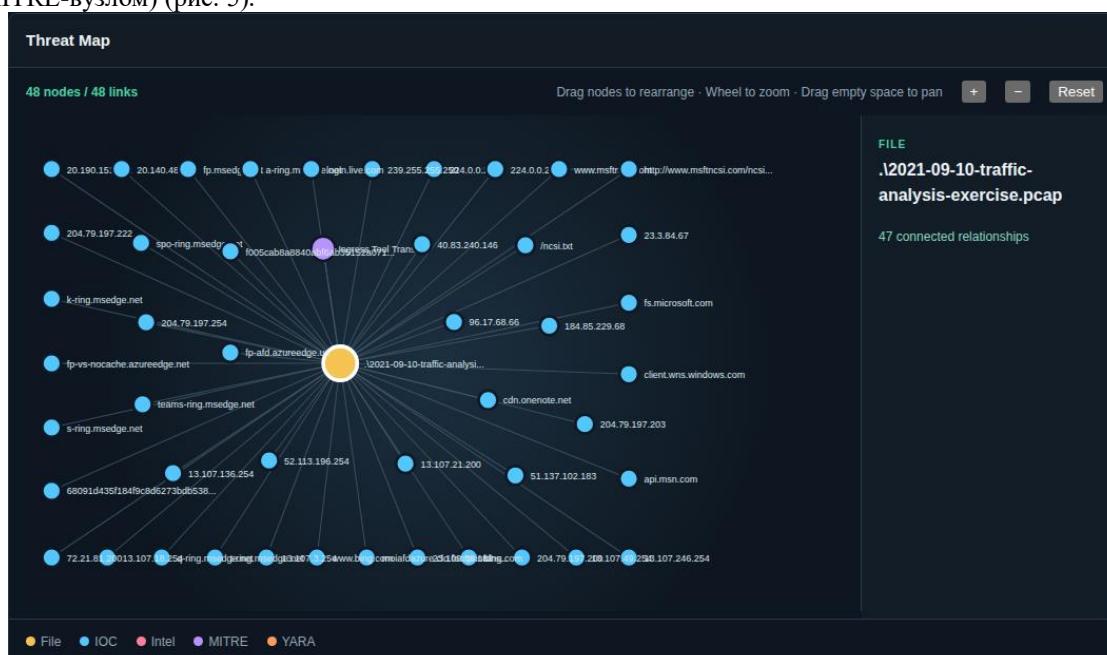


Рис. 5. Інтерактивна мапа загроз



На цьому рисунку показано граф зв'язків: файл-джерело розташований в центрі (жовтого кольору), навколо ІОС-вузли (синього кольору IP та домени msedge.net/windows.com/msn.com) і окремий фіолетовий вузол – MITRE-техніка T1105, яка пов'язана з файлом через відношення «*exhibits*».

Розроблене програмне забезпечення також дає змогу аналізувати виконувані файли, які зустрічаються в мережевому трафіку (рис. 6).

Extracted HTTP Payloads				
FILE	SIZE	FORMAT	SHA256	SOURCE
response_001.bin	14	Unknown	6137f8db2192e638e13610f75e73b9247c05f4706f0afd1fdb132d86de6b4012	www.mstncsi.c
response_002.bin	284816	PE	eed363fc4af7a9070d69340592dcab7c78db4f90710357de29e3b624aa957cf8	simpsonsaving
response_003.bin	2222	Unknown	84520bc8bb4f129be41a7ab2a1ff1064efcd060d5521c2d5c23afd8710339411	store-images.s w=150&h=150
response_004.bin	19230	Unknown	f7018b08490ebc04900ec28586710f3e93542ebe3ce390378e7f4e6665f28c43	store-images.s w=150&h=150
response_005.bin	17906	Unknown	0c58b13a7b5d78c8a9e35d5995b7352c3b38acee77d6af9886c3cf2d668bfd3c	store-images.s w=150&h=150
response_006.bin	24101	Unknown	c1c72144b320890b971bf8cd335438d6c2e50783398965e7891ccb30b4e3652	store-images.s w=150&h=150
response_007.bin	13972	Unknown	0f132c40ca33c057af0aed28534cd23b31ce36da1ad0cda3f2c2f1e03d646bca	store-images.s w=150&h=150
response_008.bin	16870	Unknown	4a11107d730c5943885081bfc3e902af14e8e83f59d110979bb2af47e325c38e	store-images.s w=150&h=150
response_009.bin	7267	Unknown	3197375f525ce4d0d339e56b343ecd510ffc72dcfa250b1352f8bd6a02d49864	download.wind
response_010.bin	2	Unknown	9b8db510ef42b8ed54a3712636fda55a4f8cfd5493e20b74ab00cd4f3979f2d	au.download.w
response_011.bin	2	Unknown	9b8db510ef42b8ed54a3712636fda55a4f8cfd5493e20b74ab00cd4f3979f2d	au.download.w

Рис. 6. Витягнутий виконуваний файл з HTTP-трафіку

На рисунку показана секція «Extracted HTTP Payloads»: response_002.bin розпізнано як формат PE розміром 284 816 байт (SHA256 eed363fc...). *ThreatLens* автоматично реасемблював його з TCP-потоків, зберіг і повторно проаналізував як окремий файл, включно з обчисленням хешів; саме ця знахідка спричинила детекцію T1105 на рисунку 3.

Також, представлений продукт має можливість показувати хронологію мережевих подій у вигляді *Timeline* формату (рис. 7).

Incident Timeline		
UTC	SOURCE	EVENT
2021-09-10T23:15:07.787295+00:00	pcap	capture_start
2021-09-10T23:15:07.787295+00:00/2021-09-11T00:31:56.054404+00:00	pcap	udp_activity
2021-09-10T23:15:09.551563+00:00	pcap	dns_response
2021-09-10T23:15:11.019480+00:00	pcap	dns_response
2021-09-10T23:15:11.022428+00:00	pcap	dns_response
2021-09-10T23:15:11.117830+00:00	pcap	dns_response
2021-09-10T23:15:14.230509+00:00	pcap	dns_response
2021-09-10T23:15:14.252737+00:00/2021-09-11T00:31:57.075695+00:00	pcap	tcp_activity
2021-09-10T23:15:14.886095+00:00	pcap	http_request
2021-09-10T23:15:14.964194+00:00	pcap	http_response
2021-09-10T23:15:14.964194+00:00	pcap	http_download
2021-09-10T23:15:15.609283+00:00	pcap	tls_client_hello
2021-09-10T23:16:38.069599+00:00	pcap	dns_response
2021-09-10T23:16:38.694974+00:00	pcap	tls_client_hello
2021-09-10T23:16:39.393702+00:00	pcap	tls_client_hello
2021-09-10T23:16:40.094805+00:00	pcap	tls_client_hello
2021-09-10T23:16:57.545292+00:00	pcap	dns_response
2021-09-10T23:16:57.663508+00:00	pcap	tls_client_hello
2021-09-10T23:16:57.663510+00:00	pcap	tls_client_hello

Рис. 7. Реконструйована хронологія мережевих подій (Timeline)



З представленого рисунку можна побачити таку хронологію подій: `capture_start` → `dns_response` → `http_request` → `http_response` → `http_download` (саме ця подія відповідає моменту завантаження PE-файлу) → `tls_client_hello`, все в строгому хронологічному порядку.

На завершення варто відзначити, що розроблений продукт дає змогу аналізувати метадані самого PCAP-файлу, який було добуто з HTTP-трафіку (рис. 8).

Hashes & File Metadata	
FORMAT	PCAP
SIZE	6180050 bytes
PACKETS	9221
CAPTURE START	2021-09-10T23:15:07.787295+00:00
CAPTURE END	2021-09-11T00:31:57.075695+00:00
DURATION (SECONDS)	4609.2884
LINK TYPE	1
MD5	11b847b2281f662331603f479c3e2440
SHA1	3f55781e2601b147217863469e989e34c7fe0109
SHA256	a9576008f7fd634740253a52b2937a205cd214cdde09f6924f0921b8e266dd12
PE metadata: None	
IMPORTED DLL	IMPORTED APIS
No import table available	

Рис. 8. Метадані PCAP-файлу

З представленого рисунку можна побачити інформацію про формат PCAP-файлу, зокрема його розмір 6180050 байт, 9221 пакет, час захоплення (2021-09-10 23:15 → 2021-09-11 00:31 UTC) і тривалість 4609.3 секунди (~1 год 17 хв), плюс MD5/SHA1/SHA256 усього файлу захоплення. Також добре видно наскрізний зв'язок усіх модулів інструмента: `pcap_analyzer.py` витягнув HTTP-потік і зібрав з нього бінарний payload → `static_analysis.py` повторно проаналізував цей payload і визначив формат PE → `behavior.py/analyze_iocs()` побачив подію `http_download` з ознакою виконуваного вмісту і згенерував детекцію T1105 → `risk.py` врахував цю HIGH-детекцію у формулі й підняв `risk_score` до 50 (MEDIUM) → `correlation.py` додав вузол MITRE-техніки в граф → `timeline.py` показав точний момент завантаження в хронології.

Результати аналізу зберігаються за допомогою модуля `storage.py`, який працює з базою даних SQLite. У цій базі даних зберігається інформація про проведені сканування, повні звіти у форматі JSON, знайдені IOC та зв'язки між різними артефактами. Це дає можливість зберігати історію перевірок і при необхідності швидко знаходити потрібні дані.

Для запуску програмного забезпечення та керування аналізом використовується командний інтерфейс `cli.py`. Через нього можна запускати перевірку файлів, налаштовувати розмір блоків під час читання та максимальний розмір файлу, вибирати джерела *Threat Intelligence* і задавати конкретні правила YARA. Також інтерфейс дає змогу створювати звіти у форматах JSON і HTML та переглядати історію попередніх сканувань. Крім командного інтерфейсу, у *ThreatLens* є можливість використовувати веб-інтерфейс на основі FastAPI. Цей веб-інтерфейс дає змогу проходити авторизацію, встановлювати обмеження розміру файлів, які завантажуються, та окремий процес для виконання тривалих операцій.

Дані в *ThreatLens* представлені за допомогою `dataclass` зі слотами. Це дає змогу зберігати інформацію в однаковому форматі та не використовувати зайву пам'ять. Основними структурами є `IOC` і `ScanResult`. `IOC` містить саму знайдену ознаку, її тип, джерело, додаткові докази та рівень впевненості у результаті. `ScanResult` зберігає загальні результати сканування, зокрема знайдені IOC, інформацію про файл, результати поведінкового аналізу, дані *Threat Intelligence* і YARA, часову послідовність подій, зв'язки між артефактами, рівень ризику та загальну оцінку загрози. Також передбачено перетворення цих даних у JSON, що дає змогу зручно зберігати та обробляти результати.

Для перевірки роботи програми було створено 11 наборів модульних тестів загальним обсягом 569 рядків коду. За допомогою тестів перевіряється пошук IOC, відсіювання помилкових результатів,



створення хешів, аналіз PE-файлів, обробка PCAP-файлів, поведінковий аналіз та його зв'язок із MITRE ATT&CK. Також перевіряється розрахунок ризику, створення часової послідовності, побудова зв'язків між об'єктами та робота бази даних SQLite.

Окремі тести стосуються роботи з *Threat Intelligence*. Для цього використовуються тестові HTTP-запити без звернення до реальних сервісів. Також перевіряється робота окремих процесів, обробка ситуацій із перевищенням часу виконання, командний інтерфейс, авторизація та обмеження розміру файлів у веб-сервісі. Таким чином, тестами перевіряються не тільки окремі частини програми, а й основні механізми їх спільної роботи. Це дає змогу оцінити стабільність і правильність роботи *ThreatLens*.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Отже, *ThreatLens* – це відносно невеликий програмний інструмент для первинного криміналістичного аналізу файлів. Основна частина його коду займає приблизно 2230 рядків. Програма написана на сучасній версії Python (3.11+) та використовує сучасні можливості мови, зокрема *dataclass* зі *slots*, *Protocol*, типізацію функцій і змінних та інші засоби *Python*.

Під час розробки особлива увага приділялася безпечній роботі з файлами, які можуть бути потенційно небезпечними. Аналіз виконується без запуску самого досліджуваного коду. Для окремих операцій використовується ізольований процес із встановленим обмеженням часу виконання. Також у програмі чітко розділено перевірку наявності даних, схожих на сертифікат, і справжню криптографічну перевірку цифрового підпису.

За своїми можливостями *ThreatLens* охоплює основні етапи первинного криміналістичного аналізу. Спочатку програма працює із сирими даними файлу, після чого шукає IOC, виконує поведінковий аналіз і пов'язує знайдені ознаки з техніками MITRE ATT&CK. Далі формується оцінка ризику, створюється графік зв'язків між знайденими об'єктами та генерується HTML-звіт із результатами аналізу. За необхідності програма також може використовувати зовнішні джерела *Threat Intelligence* і зберігати результати попередніх перевірок у базі SQLite.

Таким чином, *ThreatLens* можна розглядати як захисний інструмент, призначений для допомоги фахівцям із кібербезпеки, SOC та цифрової криміналістики під час дослідження підозрілих файлів. Його основне завдання полягає не у проведенні атак, а у виявленні та аналізі можливих ознак компрометації інформаційних систем.

У результаті проведеного дослідження розроблений програмний засіб *ThreatLens* продемонстрував придатність для автоматизованого первинного статичного аналізу шкідливого програмного забезпечення та виявлення індикаторів компрометації. До його основних переваг належать мінімальна кількість зовнішніх залежностей у базових модулях, потокова обробка великих файлів із використанням перекриття блоків, дедуплікація результатів, кешування запитів до *Threat Intelligence* та ізоляція потенційно небезпечних операцій в окремому процесі з контролем часу виконання. Важливою особливістю є те, що аналіз не передбачає виконання досліджуваного коду, що дає змогу використовувати інструмент як засіб безпечного первинного криміналістичного аналізу.

Водночас *ThreatLens* має низку обмежень, які визначають напрями його подальшого вдосконалення. Зокрема, потребують розширення можливості аналізу PCAP для підтримки IPv6 та інших канальних протоколів, збільшення або адаптації обсягу YARA-сканування для великих файлів, розширення переліку доменних зон та оптимізації декодування закодованих даних. Окремим напрямом розвитку є реалізація повноцінної криптографічної перевірки цифрових підписів PE-файлів із перевіркою сертифікаційного ланцюжка та статусу відкликання сертифікатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Al-Sada B., Sadighian A., Oligeri G. Analysis and Characterization of Cyber Threats Leveraging the MITRE ATT&CK Database. IEEE Access. 2024. Vol. 12. P. 1217–1234. DOI: 10.1109/ACCESS.2023.3344680
2. Chen S. S., Hwang R. H., Ali A., Lin Y. D., Wei Y. C., Pai T. W. Improving quality of indicators of compromise using STIX graphs. Computers & Security. 2024. Vol. 144. Article 103972. DOI: 10.1016/j.cose.2024.103972.
3. Iqbal, S., & Zulkernine, M. (2018). SpyDroid: A framework for employing multiple real-time malware detectors on Android. 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), 1–8. <https://doi.org/10.1109/MALWARE.2018.8659365>



4. Kent K., Chevalier S., Grance T., Dang H. Guide to Integrating Forensic Techniques into Incident Response. NIST Special Publication 800-86. 2006. NIST SP 800-86. URL: <https://csrc.nist.gov/pubs/sp/800/86>
5. MITRE. ATT&CK Enterprise Tactics. Офіційна база MITRE ATT&CK. URL: <https://attack.mitre.org/tactics/enterprise/>
6. Murali, R., Ravi, A., & Agarwal, H. (2020). A malware variant resistant to traditional analysis techniques. 2020 International Conference on Emerging Trends in COMPUTER TECHNOLOGIES OF PRINTING Information Technology and Engineering (ic-ETITE), 1–7. <https://doi.org/10.1109/icETITE47903.2020.264>
7. Paine K., Whitehouse O., Sellwood J., Shaw A. Indicators of Compromise (IoCs) and Their Role in Attack Defence. RFC 9424. 2023. RFC 9424. URL: <https://www.rfc-editor.org/info/rfc9424>
8. SLIFER: Investigating performance and robustness of malware detection pipelines. Computers & Security. 2024. URL: <https://www.sciencedirect.com/science/article/pii/S0167404824005704>
9. Sun H., Shu H., Kang F., Zhao Y., Huang Y. Malware2ATT&CK: A sophisticated model for mapping malware to ATT&CK techniques. Computers & Security. 2024. Vol. 140. Article 103772. DOI: 10.1016/j.cose.2024.103772
10. Tang H., Lin D., Li W., Zhang W., Zhao J. Cyber threat indicators extraction based on contextual knowledge prompt. Computer Networks. 2024. Vol. 254. Article 110839. DOI: 10.1016/j.comnet.2024.110839
11. YARA Documentation. Writing YARA rules. Офіційна документація YARA. URL: <https://yara.readthedocs.io/en/stable/writingrules.html?highlight=wild+card>
12. Гавриш, Б. М., Тимченко, О. В., Борзов, Ю. О., & Кобевко, А. Т. (2022). Класифікація шкідливого програмного забезпечення та основні методи захисту. Комп'ютерні технології друкарства, 2(48), 142–154
13. Полотай О.І. Використання комп'ютерної криміналістики для забезпечення ефективного розслідування інцидентів інформаційної та кібербезпеки. Вісник ЛДУБЖД : зб. наук. праць. Львів : ЛДУБЖД, 2023. № 28. С. 73–80.
14. Полотай О.І., Каспрівський Д.В. Сигнатурний аналіз шкідливого програмного коду в системах захисту інформаційно-комунікаційних мереж. Abstracts of III International Scientific and Practical Conference. Stockholm, Sweden. Pp. 284-289. URL: <https://eu-conf.com/en/events/science-and-society-interdisciplinary-horizons/>



Polotai Polotai

Candidate of Technical Sciences, Associate Professor,
Associate Professor of the Department of Information Security Management
Lviv State University of Life Safety, Lviv, Ukraine
ORCID: 0000-0003-4593-8601
orest.polotaj@gmail.com

Artur Tkachenko

Lecturer of the Department of Information Security Management
Lviv State University of Life Safety, Lviv, Ukraine
ORCID: 0009-0009-6830-4741
tkachenko.am14@gmail.com

Natalia Kukharska

Candidate of Physical and Mathematical Sciences, Associate Professor,
Associate Professor of the Department of Information Technology Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0000-0002-0896-8361
kukharska.n@gmail.com

Dmitro Kasprivskij

4th year higher education student, group KB-42,
Department of Information Security Management
Lviv State University of Life Safety, Lviv, Ukraine
ORCID: 0009-0006-6954-2424
dmitrokasprivskij@gmail.com

**ANALYSIS OF THREATLENS SOFTWARE TOOL FOR FORENSIC INVESTIGATION OF
MALWARE FILES AND DETECTION OF COMPROMISE INDICATORS**

Abstract. This article examines the automation of digital artifact analysis within the contexts of cybersecurity and digital forensics. The work addresses the critical need to rapidly process large volumes of files, network data, and event logs during information security incident investigations. To meet this need, the authors present ThreatLens, a software tool designed for the automated static analysis of potentially malicious digital artifacts without requiring their actual execution. The article outlines the architecture and core functionalities of ThreatLens, including the extraction and validation of indicators of compromise (IoCs); the analysis of PE files, PCAP data, and logs; and the identification of hashes, network addresses, domains, URLs, and other characteristic features. Mechanisms are implemented to detect potentially malicious behavior using MITRE ATT&CK rules, validate indicators against external Threat Intelligence sources, and perform signature-based analysis using YARA. Additionally, the tool supports event timeline generation, threat level assessment, analysis result storage, and the construction of correlation graphs within a web interface, enabling the identification of relationships between detected artifacts and signs of compromise. Particular attention is paid to the secure handling of untrusted data; potentially hazardous operations are isolated in a separate process with execution time limits, while the static nature of the analysis eliminates the need to execute the software under investigation. Finally, the article identifies key limitations of the tool regarding specific network protocol support, the scope of YARA scanning, and digital signature verification capabilities. The proposed directions for further development concern the expansion of supported formats and protocols, the optimization of large-scale data analysis, and the improvement of mechanisms for verifying the obtained results.

Keywords: digital forensics, cybersecurity incidents, cyber threats, software security, malware, signature analysis, programming, HTTP traffic, information and communication systems, Python, YARA, MITRE ATT&CK



REFERENCES

1. Al-Sada, B., Sadighian, A., & Oligeri, G. (2024). Analysis and characterization of cyber threats leveraging the MITRE ATT&CK database. *IEEE Access*, 12, 1217–1234. <https://doi.org/10.1109/ACCESS.2023.3344680>
2. Chen, S. S., Hwang, R. H., Ali, A., Lin, Y. D., Wei, Y. C., & Pai, T. W. (2024). Improving quality of indicators of compromise using STIX graphs. *Computers & Security*, 144, 103972. <https://doi.org/10.1016/j.cose.2024.103972>
3. Iqbal, S., & Zulkernine, M. (2018). SpyDroid: A framework for employing multiple real-time malware detectors on Android. In *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)* (pp. 1–8). IEEE. <https://doi.org/10.1109/MALWARE.2018.8659365>
4. Kent, K., Chevalier, S., Grance, T., & Dang, H. (2006). *Guide to integrating forensic techniques into incident response* (NIST Special Publication 800-86). National Institute of Standards and Technology. <https://csrc.nist.gov/pubs/sp/800/86>
5. MITRE. (n.d.). *Enterprise tactics*. MITRE ATT&CK. <https://attack.mitre.org/tactics/enterprise/>
6. Murali, R., Ravi, A., & Agarwal, H. (2020). A malware variant resistant to traditional analysis techniques. In *2020 International Conference on Emerging Trends in Computer Technologies and Engineering (ic-ETITE)* (pp. 1–7). IEEE. <https://doi.org/10.1109/icETITE47903.2020.264>
7. Paine, K., Whitehouse, O., Sellwood, J., & Shaw, A. (2023). *Indicators of compromise (IoCs) and their role in attack defence* (RFC 9424). Internet Engineering Task Force. <https://www.rfc-editor.org/info/rfc9424>
8. *SLIFER: Investigating performance and robustness of malware detection pipelines*. (2024). *Computers & Security*. <https://www.sciencedirect.com/science/article/pii/S0167404824005704>
9. Sun, H., Shu, H., Kang, F., Zhao, Y., & Huang, Y. (2024). Malware2ATT&CK: A sophisticated model for mapping malware to ATT&CK techniques. *Computers & Security*, 140, 103772. <https://doi.org/10.1016/j.cose.2024.103772>
10. Tang, H., Lin, D., Li, W., Zhang, W., & Zhao, J. (2024). Cyber threat indicators extraction based on contextual knowledge prompt. *Computer Networks*, 254, 110839. <https://doi.org/10.1016/j.comnet.2024.110839>
11. YARA. (n.d.). *Writing YARA rules*. <https://yara.readthedocs.io/en/stable/writingrules.html>
12. Havrysh, B. M., Tymchenko, O. V., Borzov, Y. O., & Koberko, A. T. (2022). Kласифікація шкідливого програмного забезпечення та основні методи захисту [Classification of malicious software and main protection methods]. *Комп'ютерні технології друкарства*, 2(48), 142–154.
13. Polotai, O. I. (2023). Vykorystannia kompiuternoï kryminalistyky dlia zabezpechennia efektyvnoho rozsliduvannia intsydentiv informatsiinoï ta kiberbezpeky [Use of computer forensics to ensure effective investigation of information and cybersecurity incidents]. *Visnyk Lvivskoho derzhavnoho universytetu bezpeky zhyzniedialnosti*, 28, 73–80.



14. Polotai, O. I., & Kasprivskyi, D. V. (2025). Syhnaturnyi analiz shkidlyvoho prohramnoho kodu v systemakh zakhystu informatsiino-komunikatsiinykh merezh [Signature analysis of malicious software code in information and communication network protection systems]. In *Science and society: Interdisciplinary horizons* (pp. 284–289). <https://eu-conf.com/en/events/science-and-society-interdisciplinary-horizons/>

Отримано редакцією журналу / Received: 01.06.26

Прорецензовано / Revised: 16.06.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.