МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

СЕГЕДА Євген Мар'янович

Алгоритми виявлення шкідливого програмного забезпечення за допомогою платформи Wazuh та VirusTotal / Malware Detection Algorithms Using the Wazuh Platform and VirusTotal

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма –Кібербезпека

Кваліфікаційна робота

Виконав студент групи КБм -21
Є.М. Сегеда

_____

Науковий керівник
д.т.н., професор В.В. Яцків

_____

Кваліфікаційну роботу допущено
до захисту:

«____» _____ ___2025 р.

Завідувач кафедри

_____ В.В.Яцків

ТЕРНОПІЛЬ - 2025

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 - Кібербезпека та захист інформації
освітньо-професійна програма –Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
____" _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

СЕГЕДІ Евгену Мар'яновичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Алгоритми виявлення шкідливого програмного забезпечення за допомогою платформи Wazuh та VirusTotal / Malware Detection Algorithms Using the Wazuh Platform and VirusTotal
керівник роботи д.т.н., професор В.В. Яцків

затверджені наказом по університету від 20 грудня 2024 року № 938

2. Строк подання студентом закінченої кваліфікаційної роботи 5 грудня 2025року.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- проаналізувати методи та засоби виявлення ШПЗ;
- дослідити принципи роботи та функції Wazuh і VirusTotal;
- розробити модель взаємодії компонентів;
- реалізувати алгоритм виявлення ШПЗ за допомогою платформи Wazuh та VirusTotal;
- оцінити ефективність розробленого алгоритму.

5. Перелік графічного матеріалу у роботі.
– модель взаємодії компонентів системи виявлення ШПЗ;
– алгоритм виявлення ШПЗ за допомогою платформи Wazuh та VirusTotal;
– схема роботи системи на основі алгоритму виявлення ШПЗ.

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Теоретичні основи виявлення шкідливого програмного забезпечення	12.2024 р. – 03.2025 р.	
2	Аналіз інструментів моніторингу подій безпеки та аналітики загроз	03.2025 р. – 06.2025 р.	
3	Розробка комбінованого алгоритму виявлення шкідливого програмного забезпечення	06.2025 р. – 11.2025 р.	

Студент _____ (підпис) Є.М. Сегеда

Керівник роботи _____ (підпис) д.т.н., професор Яцків В.В.

АНОТАЦІЯ

Сегеда Є.М. Алгоритми виявлення шкідливого програмного забезпечення за допомогою платформи Wazuh та VirusTotal. – Рукопис.

Дослідження на здобуття освітнього ступеня «магістр» за спеціальністю 125 «Кібербезпека та захист інформації», освітньо-професійна програма «Кібербезпека». – Західноукраїнський національний університет, Тернопіль, 2025.

У роботі досліджено методи аналізу та виявлення шкідливого програмного забезпечення; запропоновано модель взаємодії компонентів для багаторівневого аналізу підозрілих об’єктів; реалізовано алгоритм виявлення шкідливого програмного забезпечення на основі інтеграції платформи Wazuh із сервісом VirusTotalз використанням API-сервісів та досліджено його ефективність у виявленні сучасних кіберзагроз.

Ключові слова: ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, WAZUH, VIRUSTOTAL, АЛГОРИТМ ВИЯВЛЕННЯ ЗАГРОЗ.

ABSTRACT

Seheda Ye.M. Algorithms for Malware Detection Using the Wazuh and VirusTotal Platforms. – Manuscript.

Doctoral studies for the education level «Master» with the title 125 «Cybersecurity and Information Protection». – West Ukrainian National University, Ternopil, 2025.

The study examines methods of malware analysis and detection; proposes a component interaction model for multi-level analysis of suspicious objects; implements a malware detection algorithm based on the integration of the Wazuh platform with the VirusTotal service using API interfaces; and evaluates its effectiveness in detecting modern cyber threats.

Keywords: MALWARE, WAZUH, VIRUSTOTAL, THREAT DETECTION ALGORITHM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AV - антивірусна платформа;

C2 - командний сервер;

FIM - контроль цілісності файлів;

SIEM - система управління інформаційною безпекою;

БД – база даних;

ІБ – інформаційна безпека;

ОС - операційна система;

ШПЗ - шкідливе програмне забезпечення.

ВСТУП

Актуальність теми. Шкідливе програмне забезпечення (ШПЗ) постійно еволюціонує, використовуючи поліморфізм і методи обходу традиційних сигнатурних механізмів захисту [1-5]. Зростання складності та варіативності кіберзагроз потребує застосування комбінованих підходів до їх виявлення [6-9].

Одним із ефективних рішень є інтеграція локальних систем моніторингу інформаційної безпеки (ІБ) із зовнішніми аналітичними центрами. Комбінування можливостей платформи Wazuh та сервісу VirusTotal забезпечить поєднання локального контролю цілісності файлів, поведінкового аналізу процесів та активного реагування на підозрілі події із глобальними базами сигнатур і репутаційними даними для валідації та класифікації підозрілих об’єктів.

Такий підхід підхід дозволяє реалізувати багаторівневу перевірку та оперативне реагування на інциденти ІБ завдяки тому, що інструменти взаємодоповнюють один одного, застосовуючи різні методи виявлення ШПЗ.

Мета і завдання дослідження. Метою роботи є розробка та дослідження алгоритму виявлення ШПЗ шляхом інтеграції локальних можливостей Wazuh з віддаленою аналітикою VirusTotal для швидкого визначення та реагування на підозрілі файли та процеси на кінцевих вузлах мережі.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати сучасні методи аналізу та сособи виявлення ШПЗ;
- дослідити архітектуру, принципи роботи та функціональні можливості платформи Wazuh і сервісу VirusTotal;
- розробити модель взаємодії компонентів системи виявлення та аналізу шкідливих об’єктів;
- реалізувати алгоритм комбінованого аналізу із використанням API-сервісів Wazuh та VirusTotal.
- дослідити ефективність розробленого алгоритму та оцінити його

результати.

Об’єкт дослідження - процес виявлення ШПЗу середовищах ІБ.

Предмет дослідження - шляхи підвищення ефективності виявлення шкідливих об’єктів.

Методи досліджень: аналіз, порівняння та моделювання, що дозволили дослідити сучасні підходи до виявлення ШПЗ, а також розробити алгоритм з використанням інструментів Wazuh і VirusTotal; експериментальний метод із практичною реалізацією у тестовому середовищі для оцінки ефективності алгоритму.

Наукова новизна отриманих результатів полягає у розробці алгоритму виявлення ШПЗ, який поєднує функції моніторингу подій ІБ в реальному часі з аналітичними можливостями багатоджерельного аналізу загроз. Запропонований підхід дозволяє підвищити точність і ефективність ідентифікації шкідливих об’єктів шляхом комбінування поведінкових і сигнатурних ознак.

Практичне значення отриманих результатів полягає у створенні алгоритму виявлення ШПЗ, який може бути інтегрований у SIEM-системи підприємств або використаний для підвищення ефективності аналітичних процесів у центрах безпеки (SOC).

Публікації та апробація кваліфікаційної роботи.

1. Сегеда Є., Давлетова А. Комбінована система моніторингу та виявлення Malware-загроз.- Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2025), Тернопіль, 2025.

2. Питель Р., Сегеда Є. Алгоритм виявлення шкідливого програмного забезпечення на кінцевих вузлах мережі.- Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2025.

1. ТЕОРЕТИЧНІ ОСНОВИ ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття та класифікація зловмисних програм

ШПЗ (malware) - це сукупність програмних кодів, сценаріїв або інструкцій, створених з метою порушення конфіденційності, цілісності чи доступності інформаційних ресурсів. У більш широкому розумінні ШПЗ охоплює будь-яке ПЗ, що здійснює несанкціоновані дії на інформаційних системах користувачів, серверів або мережевих пристроїв. Основними ознаками ШПЗ є несанкціонований запуск, приховування власної активності, модифікація системних файлів, порушення логіки роботи операційної системи (ОС) або отримання віддаленого доступу до ресурсів [10-15].

За міжнародними стандартами ISO/IEC 27032 та рекомендаціями NIST SP 800-83, ШПЗ класифікується за призначенням, способом поширення, методом активації та наслідками для системи (рисунок 1.1).

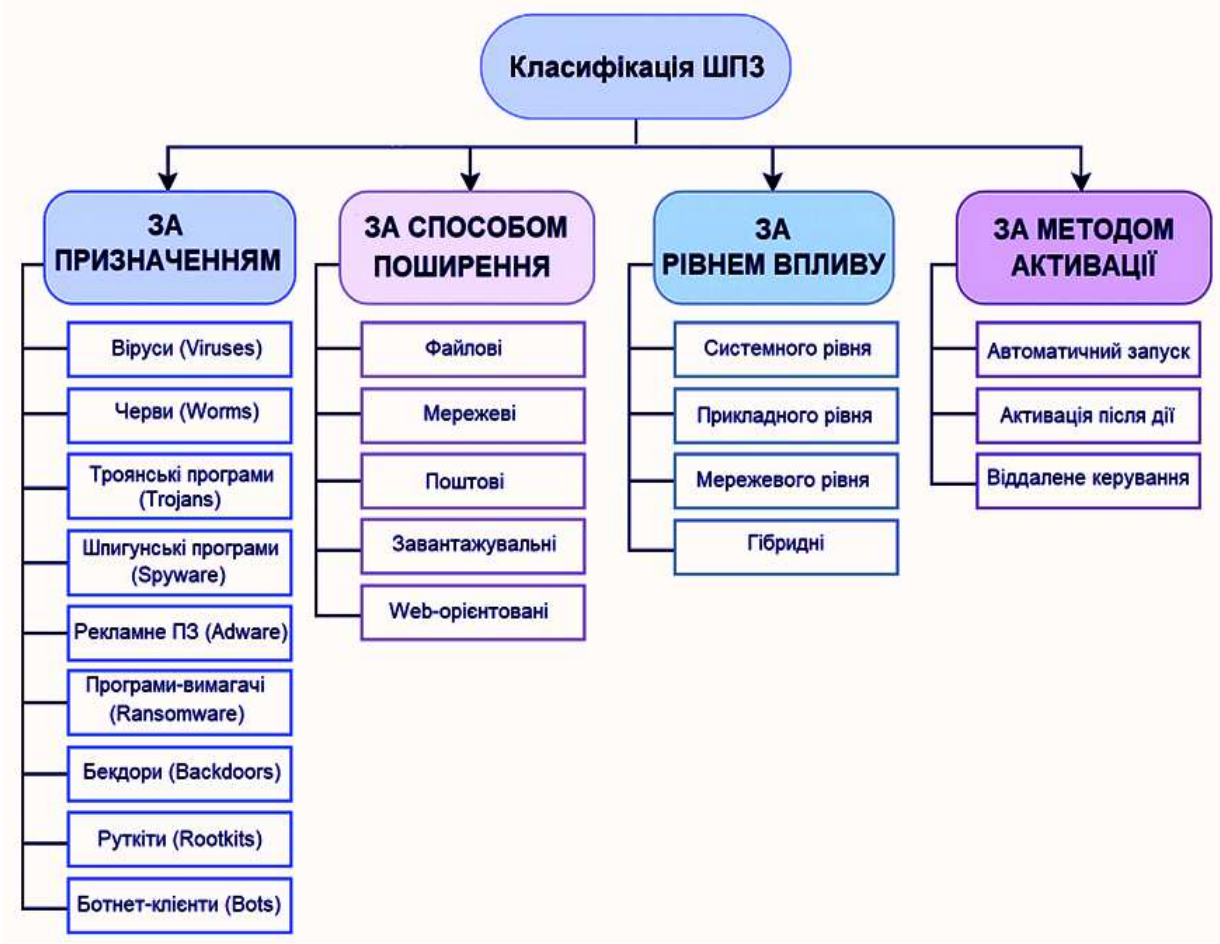


Рисунок 1.1 – Класифікація ШПЗ

Класифікація дає змогу системам виявлення загроз ефективніше застосовувати різні методи аналізу - сигнатурний, поведінковий, евристичний або гібридний [1]. Класифікація ШПЗ за призначенням включає:

- Віруси (Viruses) - самовідтворювані програми, які заражають виконувані файли або сектори диску, вставляючи у них власний код. Для свого розповсюдження потребують запуску зараженого файлу користувачем, наприклад файловий вірус Win32.Sality модифікує виконувані файли .exe і .scr, завантажуючи додаткові компоненти з мережі.
- Черви (Worms) - самопоширювані програми, що використовують мережеві протоколи для розповсюдження без участі користувача. Приклад є WannaCry (2017), що використовував вразливість SMBv1 для автоматичного зараження віддалених вузлів.
- Троянські програми (Trojans) - ШПЗ, яке маскується під легітимні програми, але після запуску виконує шкідливі дії, такі як викрадення даних або створення бекдору.
- Шпигунські програми (Spyware) - збирають інформацію про користувача, його паролі, історію браузера або натискання клавіш, наприклад, AgentTesla, що передає дані через SMTP або HTTP-з'єднання.
- Рекламне ПЗ (Adware) - нав'язує рекламу, змінює налаштування браузера, перенаправляє запити користувача.
- Програми-вимагачі (Ransomware) - шифрують дані користувача та вимагають викуп за розшифрування. Прикладом можуть слугувати LockBit або Conti - сучасні представники з розподіленою мережею операторів (Ransomware-as-a-Service).
- Бекдори (Backdoors) - забезпечують несанкціонований віддалений доступ до системи, часто є частиною складніших атак, одним із прикладів є PlugX, що використовується у цільових атаках для дистанційного керування.
- Руткіти (Rootkits) - приховують присутність іншого ШПЗ, змінюючи системні структури ядра ОС або таблиці викликів API.
- Ботнет-клієнти (Bots) - частини розподілених мереж, що

отримують команди від C2-серверів для проведення DDoS-атак або розсилки спаму.

Класифікація за способом поширення характеризує механізм проникнення ШПЗ у систему:

- через заражені файли (файлові інфектори);
- через мережу (експлуатація вразливостей TCP/IP, SMB тощо);
- через електронну пошту (вкладення або посилання);
- через зовнішні носії (USB, змінні диски);
- через веб-ресурси (drive-by-download, скрипти у браузері).

Класифікація за рівнем впливуна систему:

- системного рівня - змінюють ядро або драйвери (rootkits);
- прикладного рівня - впливають на процеси користувача (трояни, віруси);
- мережевого рівня - перехоплюють або модифікують трафік (sniffers, MITM-ін'єкції);
- гібридні - поєднують кілька рівнів впливу.

За способом впливу ШПЗ на систему виділяють наступні:

- руйнування або модифікація файлів і реєстру;
- споживання системних ресурсів (DoS, криптомайнери);
- викрадення даних;
- блокування доступу до системи чи даних.

Класифікація за способом активації ШПЗ:

- автоматичний запуск при старті ОС (через автозавантаження, служби, драйвери);
- активація після дії користувача (відкриття файлу, запуск вкладення);
- віддалене керування через командних серверів (C2-сервер).

За рівнем прихованості, що показує, наскільки ШПЗ здатне уникати виявлення можна виділити:

- стелс-програми - приховують свою активність у процесах та файловій системі;

- поліморфні віруси - змінюють структуру коду при кожному запуску;
- метаморфні віруси - переписують власний код без зміни функціоналу;
- безфайлові (fileless) загрози - працюють виключно в оперативній пам’яті, не залишаючи слідів на диску.

Сучасні загрози характеризуються високим рівнем автоматизації, поліморфізму та обфускації коду, що ускладнює виявлення традиційними сигнатурними методами. Широкого поширення набули модульні архітектури, з розділенням функцій на завантажувач, бекдор, інжектор тощо, та безфайлові атаки (fileless malware), які виконуються виключно у пам’яті процесів без створення артефактів на диску. Такі тенденції зумовилюють необхідність комбінованих систем аналізу, де поєднуються сигнатурні, поведінкові та репутаційні методи - зокрема, реалізовані на платформах типу Wazuh із інтеграцією зовнішніх сервісів, як-от VirusTotal.

1.2 Методи аналізу та виявлення шкідливих об’єктів

Аналіз ШПЗ - це процес дослідження шкідливого коду з метою визначення його функціональності, поведінки, методів поширення та потенційного впливу на систему. Залежно від рівня доступу до коду, інструментів та глибини дослідження, розрізняють три основні типи аналізу: статичний, динамічний і байтовий [16-20].

На рисунку 1.2 наведено класифікацію методів аналізу ШПЗ. Кожен тип аналізу має власну мету і рівень деталізації, статичний - швидкий і безпечний, підходить для попередньої оцінки, динамічний відображає реальну поведінку, а байтовий є найбільш глибоким і використовується для експертної розвідки або створення сигнатур.

Статичний аналіз полягає у вивченні шкідливого файлу без його виконання, зокремавідбувається перевірка контрольних сум (MD5, SHA256), дослідження заголовків файлів PE/ELF, аналіз рядків коду (strings),

виявлення підозрілих викликів API тощо. Основною метою аналізу є ідентифікація сигнатур, залежностей, метаданих та структури коду. Такий підхід швидкий і безпечний, проте не дозволяє виявити приховану або умовну поведінку програми (наприклад, дії, які виконуються лише після підключення до мережі).

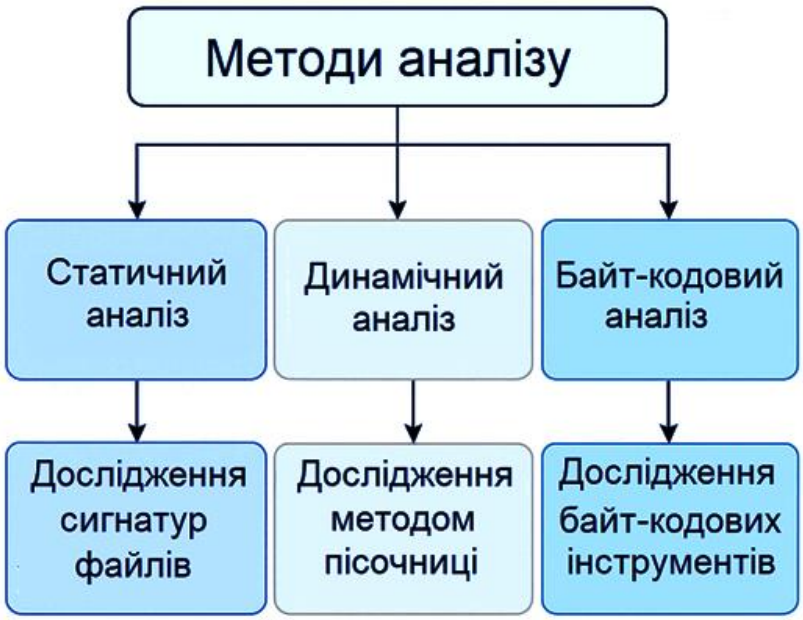


Рисунок 1.2 - Класифікація методів аналізу ШПЗ

Динамічний аналіз здійснюється під час виконання ШПЗ у контрольованому середовищі (пісочниці, sandbox), з подальшим моніторингом створення файлів, процесів, мережевих запитів та фіксації спроб підключення до С2-серверів. Його метою є спостереження за реальною поведінкою програми, взаємодією з файловою системою, мережею, процесами та реєстром. Перевага даного методу аналізу полягає у можливості виявлення поведінки, яку неможливо побачити при статичному аналізі. Недоліком є необхідність у безпечному середовищі та ризик витоку ШПЗ за межі пісочниці.

Байтовий аналіз, іноді називають глибинним або реверс-інжинірингом. Він виконується на рівні машинного коду та застосовується у вмпадках, коли потрібно точно зрозуміти алгоритми роботи програми, схеми шифрування або механізми обходу захисту. Основними прийомами, що використовуються при аналізі є дизасемблювання та декомпіляція, аналіз таблиць

імпорту/експорту, відновлення логіки шкідливих модулів та виявлення обфускації або пакування (UPX, Themida тощо). Цей метод потребує глибоких знань архітектури процесора, асемблера та принципів роботи ОС.

У сучасних системах виявлення загроз різні методи аналізу комбінуються, утворюючи гібридну модель, що забезпечує підвищену точність, гнучкість та ефективність детекції як відомих, так і нових типів ШПЗ.

Виявлення ШПЗ є ключовим завданням у сфері кібербезпеки, від якого залежить ефективність систем захисту інформації, антивірусних продуктів і платформ моніторингу подій ІБ (SIEM, EDR, XDR). Методи виявлення ШПЗ є прикладними підходами, які використовують різні типи аналізу для дослідження об’єктів (файлів, процесів, трафіку тощо). Існує декілька основних підходів до виявлення, що відрізняються за принципами дії, точністю, швидкістю обробки та здатністю реагувати на нові невідомі загрози [15-18]. До найпоширеніших належать сигнатурні, евристичні, поведінкові та гібридні методи (рисунок 1.3).

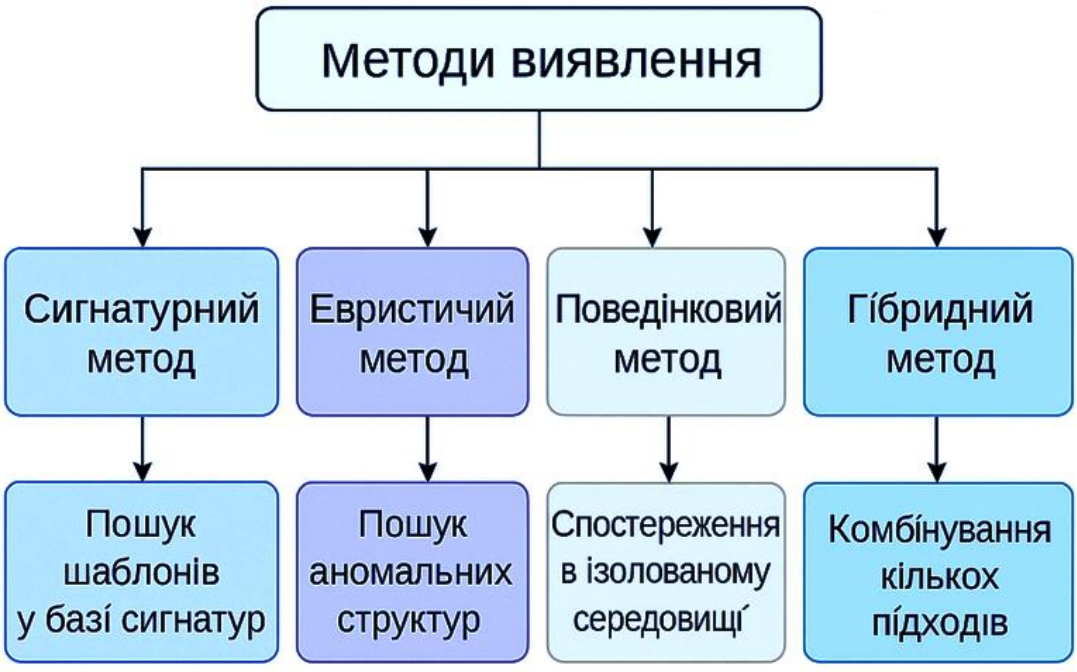


Рисунок 1.3 – Класифікація методів виявлення ШПЗ

Методи виявлення ШПЗ базуються на різних типах його аналізу (таблиця 1.1). Зокрема, сигнатурні методи використовують статичний

аналіз для пошуку відомих шаблонів шкідливого коду; поведінкові методи - динамічний аналіз, що дозволяє дослідити активність програм у контрольованому середовищі; евристичні підходи спираються на низькорівневий аналіз, який виявляє нетипові ознаки у структурі коду. Гібридні системи об'єднують кілька типів аналізу, забезпечуючи більш високу точність і стійкість до нових, ще не ідентифікованих загроз.

Таблиця 1.1 – Взаємне відображення методів виявлення та аналізу ШПЗ

Тип аналізу	Метод виявлення	Інструменти
Статичний аналіз	Сигнатурні методи	pefile, Ghidra, IDA Pro, BinText, VirusTotal.
Динамічний аналіз	Поведінкові методи	Cuckoo Sandbox, Any.Run, ProcMon, Wireshark, Wazuh
Байтовий / низькорівневий аналіз	Евристичні та гібридні методи	Ghidra, Radare2, x64dbg, OllyDbg

1.2.1 Сигнатурний метод

Сигнатурний (або детермінований) підхід є класичним і базується на пошуку у файлах чи мережевому трафіку специфічних ознак - сигнатур, що унікально ідентифікують відомі зразки ШПЗ. Сигнатура може бути представлена як послідовність байтів, фрагмент коду, хеш, або комбінація ознак, властивих певній загрозі [10, 11].

Алгоритм сигнатурного аналізу передбачає:

- створення бази відомих сигнатур (на основі еталонних зразків);
- сканування об'єкта (файлу, процесу, пакету);
- пошук збігів з еталонами;
- фіксацію інциденту у разі виявлення відповідності.

Для оптимізації застосовуються структури даних, наприклад дерев пошуку Aho-Corasick або Rabin-Karp hashing, що дозволяють швидке

порівняння великої кількості сигнатур. Бази сигнатур постійно оновлюються антивірусними лабораторіями або через відкриті сервіси, такі як VirusTotal, Hybrid Analysis чи Malpedia.

Переваги даного методу полягають у високій точності і швидкості при мінімальній кількості хибних спрацьовувань. Проте даний метод не забезпечує виявлення нових, невідомих або модифікованих зразків (zero-day).

1.2.2 Евристичний метод

Евристичний підхід базується на аналізі потенційно шкідливих ознак або поведінкових моделей коду, навіть якщо конкретна сигнатура невідома. Його метою є виявлення нових або модифікованих зразків, які маскуються під нешкідливі програми [7, 10]. Як правило, використовуються два типи евристичного аналізу:

- статичний евристичний аналіз - досліджує структуру та код файлу без його виконання (наприклад, таблиці імпорту, розмір секцій PE-файлів, підозрілі API-виклики VirtualAlloc, WriteProcessMemory, CreateRemoteThread);
- динамічний евристичний аналіз (sandbox) - виконує код у віртуалізованому середовищі та фіксує його дії: створення процесів, зміну реєстру, мережеву активність, спроби шифрування даних тощо.

Алгоритм виявлення полягає у наступному: система розраховує евристичний бал ризику на основі кількості виявлених підозрілих дій. Якщо бал перевищує порогове значення, об’єкт класифікується як потенційно шкідливий.

До переваги методу можна віднести здатність виявляти невідомі загрози, поліморфне ПЗ та zero-day атаки. Проте він характеризується високою обчислювальною складністю та можливістю хибних спрацьовувань.

1.2.3 Поведінковий метод

Поведінковий метод (behavior-based detection) ґрунтується на моніторингу дій програм під час виконання та порівнянні цих дій із

профілями “нормальної” поведінки [10, 20]. На відміну від евристичного методу, цей підхід працює у реальному часі та дозволяє ідентифікувати складні багатоступеневі атаки, навіть якщо код не містить відомих сигнатур.

Основними джерелами даних для поведінкового методу виявлення є системні журнали (Windows Event Log, syslog), API-виклики та системні події та мережеві пакети, процеси, звернення до файлової системи.

Прикладом поведінкових ознак можуть бути наступні:

- несанкціоноване створення процесу з правами адміністратора;
- шифрування великої кількості файлів у короткий час;
- спроба підключення до відомих C2-доменів;
- ін’єкція коду у легітимні процеси.

Сучасні системи управління ІБ (SIEM), зокрема Wazuh, реалізують поведінковий аналіз через механізми правил кореляції та моделі відповідності шаблонів подій (OSSEC Rules Engine). Це дозволяє виявляти не лише відомі типи атак, а й відхилення від нормальної поведінки користувачів або систем.

Переваги даного методу включають детекцію складних атак, що не мають сигнатур. Але він вимагає значних обчислювальних ресурсів і точного налаштування правил.

1.2.4 Гібридні підходи до виявлення загроз

Гібридні системи поєднують розглянуті методи для досягнення максимальної ефективності виявлення [6, 9, 16]. Вони використовують комбінацію правил, моделей машинного навчання, баз знань і зовнішніх сервісів репутації файлів, таких як VirusTotal, AbuseIPDB, MISP тощо.

Принцип гібридного аналізу:

- застосування попереднього фільтру - сигнатурний аналіз;
- при відсутності збігу - використовується евристичний або поведінковий аналіз;
- запит до зовнішніх репутаційних баз, перевіряються хеші, IP, URL;

- інтегральна оцінка ризику (combined score), що визначає остаточне рішення.

Такий підхід зменшує навантаження на систему та підвищує точність, адже дозволяє виявляти невідомі загрози при мінімальній кількості хибних спрацьовувань. Саме гібридні методи стали основою сучасних платформ безпеки типу EDR/XDR і інтегрованих рішень, до яких належить Wazuh, що у подальших розділах роботи використовується для реалізації комбінованого алгоритму детекції.

Перевагою даного методує універсальність, масштабованість, баланс точності й чутливості. Недоліком можна вважати складність налаштування, потребу у постійному оновленні зовнішніх баз і моделей.

1.3 Порівняльна характеристика сучасних антивірусних платформ та систем виявлення й реагування на загрози

У сучасних умовах кіберзагроз традиційні антивірусні засоби вже не забезпечують достатнього рівня захисту, оскільки більшість атак здійснюються у багатоступеневій або безфайловій формі. Ефективний захист потребує комплексного підходу, що поєднує методи виявлення, аналізу, кореляції подій і автоматизованого реагування. Як правило, розрізняють дві основні категорії таких систем:

- антивірусні платформи (AV) [21, 22];
- системи виявлення й реагування на загрози (IDS/IPS, EDR, SIEM, XDR) [23-26].

AV є базовим рівнем захисту кінцевих пристроїв. Їхнє основне завдання - виявлення та блокування шкідливих об’єктів до їхнього виконання у системі. Класичні антивірусні рішення (наприклад, Kaspersky, ESET, Avast, Norton, Bitdefender) використовують сигнатурний та евристичний аналіз, а також компоненти репутаційного контролю файлів.

У сучасних поколіннях антивірусів, так званих Next Generation Antivirus – NGAV, інтегровано додаткові механізми, зокрема поведінкове

виявлення, машинне навчання, cloud reputation, а також механізми запобігання експлойтам.

Переваги AV полягають у високій точності виявлення відомих зразків ШПЗ, простоті розгортання на кінцевих вузлах та централізованому оновленні сигнатур та політик. Проте обмежена ефективність проти нових або безфайлових атак, часте оновлення баз даних (БД), а також не можливість відстежувати міжпроцесні взаємодії та кореляцію подій між хостами обмежує їх використання.

На відміну від антивірусів, системи виявлення і реагування на загрози, такі як Snort, Suricata, Zeek (Bro), орієнтовані не лише на ізоляцію шкідливих файлів, а й на моніторинг поведінки системи, журналів безпеки та мережевого трафіку.

IDS/IPS (Intrusion Detection / Prevention Systems) - це мережеві або хостові рішення, які аналізують потоки даних і події з метою виявлення ознак вторгнень:

- IDS (система виявлення вторгнень) - лише повідомляє про аномалії;
- IPS (система запобігання вторгненням) - додатково блокує трафік або процес.

EDR (Endpoint Detection and Response) рішення (як CrowdStrike Falcon, SentinelOne, Microsoft Defender for Endpoint) збирають дані про всі процеси, події реєстру, мережеву активність, дозволяючи проводити післяінцидентний аналіз і віддалене реагування. EDR працює на рівні окремих вузлів, забезпечуючи глибоку видимість у поведінку системи.

Системи SIEM (Security Information and Event Management), до яких належить Wazuh, є централізованими платформами збору, нормалізації та кореляції подій ІБ з усіх компонентів інфраструктури (серверів, робочих станцій, мережевих пристроїв, хмарних сервісів). SIEM поєднує функції журналювання, аналітики, оповіщення та інтеграції з механізмами автоматичної відповіді (SOAR).Приклади: Wazuh, Splunk, IBM QRadar, ArcSight, Elastic Security, Graylog.

XDR (Extended Detection and Response) - це сучасний розвиток концепції EDR/SIEM, що поєднує різні джерела даних (endpoint, network, cloud) у єдину модель безпеки. Приклади: Palo Alto Cortex XDR, Sophos Intercept X, Trend Micro Vision One.

В таблиці 1.2 наведено порівняльні характеристики розглянутих систем виявлення ШПЗ.

Таблиця 1.2 - Порівняльна характеристика основних систем

Клас системи	Основні приклади	Джерела даних	Методи аналізу	Рівень реагування	Особливості
AV / NGAV	Kaspersky, ESET, Bitdefender	Файли, процеси	Сигнатурний, евристичний, ML	Локальний	Захист кінцевих вузлів
IDS/IPS	Snort, Suricata, Zeek	Мережевий трафік	Сигнатурний, статистичний	Мережевий	Аналіз пакетів у реальному часі
EDR	SentinelOne, CrowdStrike, Defender	Процеси, події ОС	Поведінковий, аналітичний	Локальний / централізований	Глибокий аналіз подій
SIEM	Wazuh, Splunk, QRadar	Логи, трафік, агенти	Кореляційний, поведінковий, репутаційний	Централізований	Моніторинг усієї інфраструктури
XDR	Cortex XDR, Sophos XDR	Endpoint, Network, Cloud	ML, поведінковий, аналітичний	Централізований	Єдина система керування подіями

Таким чином, еволюція засобів виявлення загроз демонструє перехід від автономних AV до комплексних інтегрованих систем аналізу подій.

Платформа Wazuh посідає проміжне місце між класичними SIEM і сучасними XDR-рішеннями. Вона є відкритою платформою з підтримкою інтеграцій, забезпечує необхідну гнучкість і масштабованість для реалізації комбінованих алгоритмів виявлення. Завдяки цьому Wazuh може функціонувати як основа для побудови гібридного алгоритму виявлення ШПЗ.

2. АНАЛІЗ ІНСТРУМЕНТІВ МОНІТОРИНГУ ПОДІЙ БЕЗПЕКИ ТА АНАЛІТИКИ ЗАГРОЗ

2.1 Архітектура та принципи роботи платформи Wazuh

Платформа Wazuh є відкритою SIEM, що поєднує збір, аналіз, кореляцію та реагування на події безпеки [27]. Її архітектура базується на агентно-централізованій моделі, яка включає компоненти Wazuh Agent, Wazuh Manager та Wazuh Dashboard.

На рисунку 2.1 показано загальну архітектуру платформи Wazuh та зв’язок між компонентами [27-29].

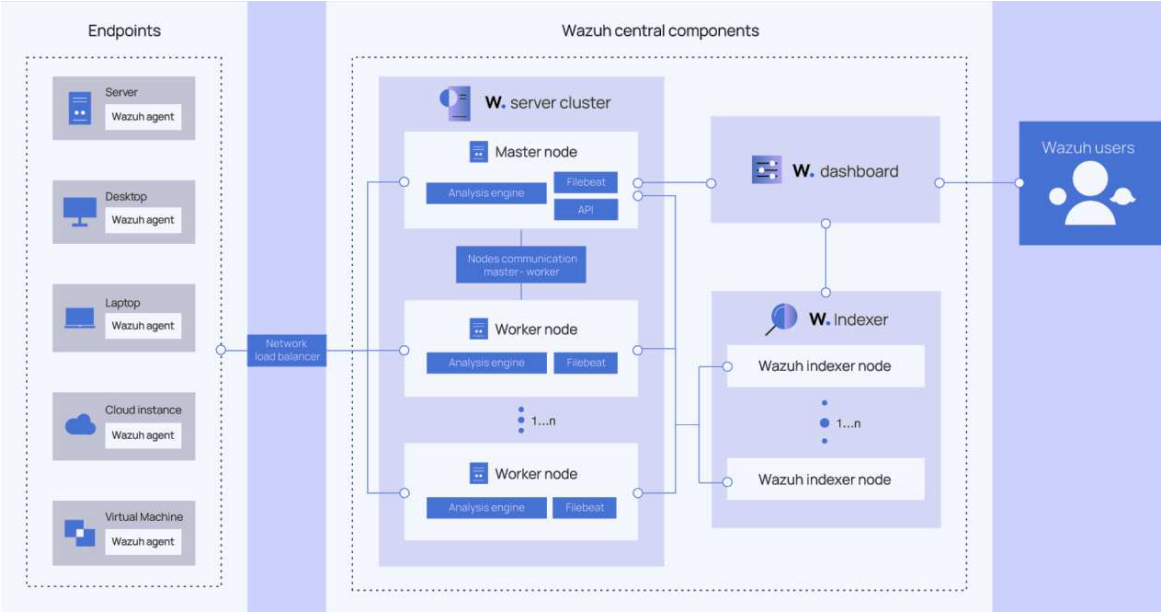


Рисунок 2.1 - Архітектура Wazuh

Основні функціональні можливості Wazuh включають:

- контроль цілісності файлів (FIM) та системних об’єктів;
- поведінковий аналіз процесів та активностей користувачів;
- YARA-сканування для виявлення відомих сигнатур ШПЗ;
- Active Response – автоматичне реагування на інциденти, зокрема блокування, карантин, видалення загроз тощо;
- централізований збір і аналіз логів з можливістю інтеграції з SIEM-системами.

Агенти, встановлені на вузлах, збирають журнали подій і передають їх

на центральний менеджер. Менеджер виконує обробку, аналіз, застосовує правила кореляції, а результати відображаються у веб-інтерфейсі Dashboard на основі Elastic Stack. Функціональні компоненти платформи Wazuh наведені на рисунку 2.2.

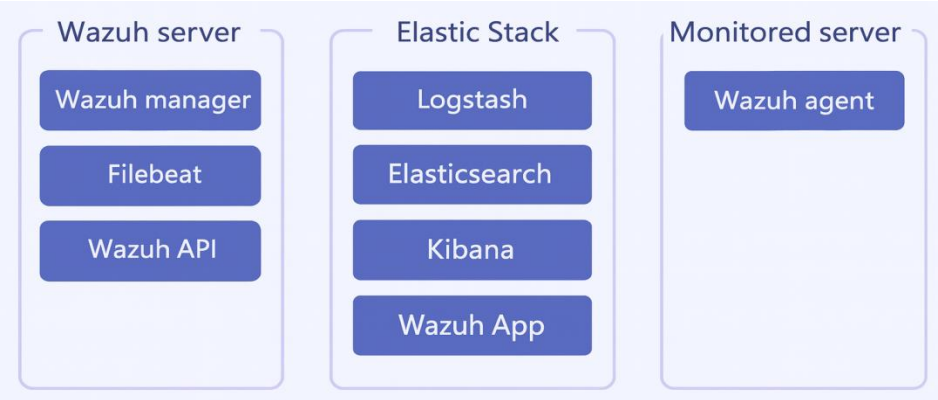


Рисунок 2.2 – Функціональні компоненти платформи Wazuh

Wazuh Agent - це компонент, який виконує збір системних логів, подій ІБ, метрик та стану конфігурації (рисунок 2.3) [27]. Основні функції агента: збір системних подій, моніторинг процесів, FIM, виконання команд.

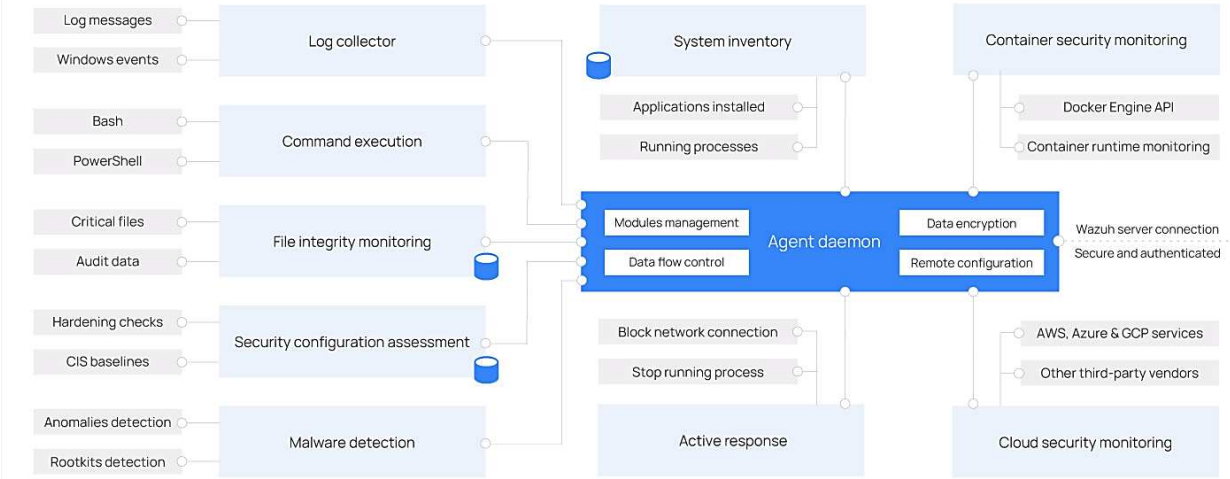


Рисунок 2.3 - Архітектура та модулі Wazuh-агента

Дані передаються на Wazuh Manager за протоколом TCP або UDP з використанням шифрування TLS 1.2/1.3. Агенти підтримують як Windows-, так і UNIX-подібні системи. На рисунку 2.4 наведено приклад, як події відбуваються в середовищі Wazuh [27-29].

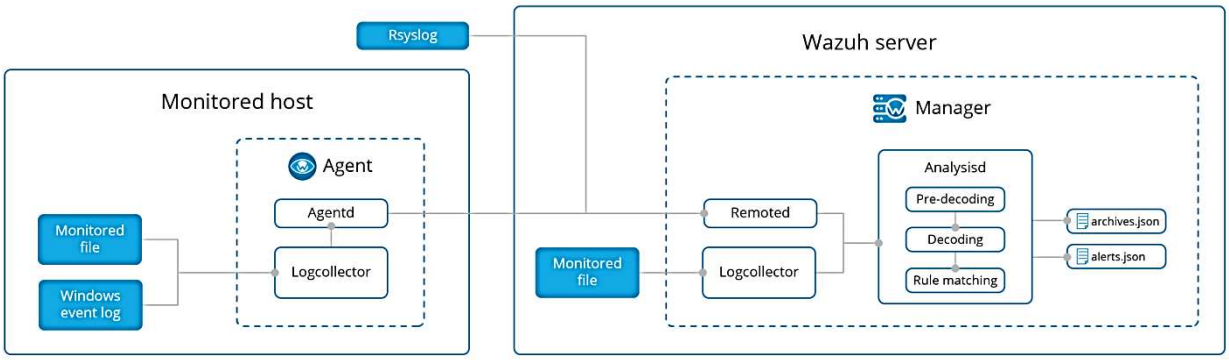


Рисунок 2.4 - Події в середовищі Wazuh

Wazuh Manager є центральним компонентом, який відповідає за обробку даних (рисунок 2.5). Він включає механізми декодування подій (Decoders), застосування правил (Rules Engine), кореляції подій (Correlation Engine) та виконання автоматичних дій (Active Response) [28-30].

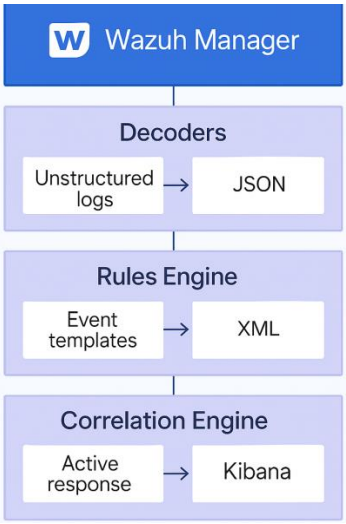


Рисунок 2.5 – Принцип роботи Wazuh Manager

Decoders аналізують неструктуровані логи та перетворюють їх у формат JSON. Rules Engine використовує XML-файли з шаблонами подій, які можуть комбінуватися у складні логічні структури. У разі спрацьовування певного правила система виконує дію реагування або передає дані у Kibana для відображення.

Модуль FIM забезпечує контроль за змінами у критичних системних файлах (рисунок 2.6). Агенти формують базу еталонних хешів (MD5, SHA1,

SHA256), а під час перевірки порівнюють поточні значення. У разі виявлення відмінностей створюється подія 'File integrity changed' [27].

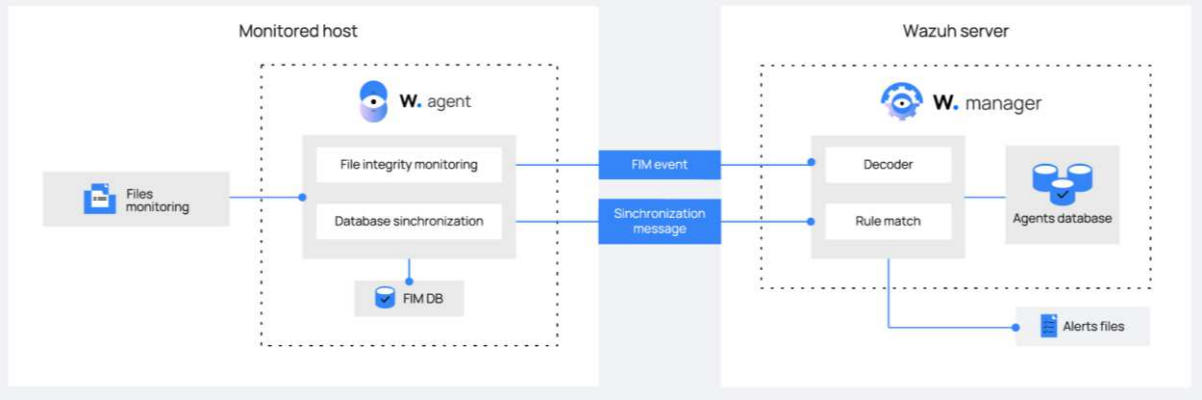


Рисунок 2.6 – Схема роботи модуля FIM

Модуль YARA-сканування реалізує механізм виявлення ШПЗ на основі сигнатурного аналізу. Технологія YARA дозволяє створювати набори правил, що описують характерні ознаки файлів або процесів, притаманні певним типам загроз. Ці правила можуть визначати унікальні шаблони коду, рядки, бінарні послідовності або інші структурні особливості, що дозволяє ідентифікувати відомі зразки шкідливих програм Під час перевірки агент Wazuh застосовує ці правила до локальних файлів і пам’яті ОС, виявляючи потенційно небезпечні об’єкти. Результати сканування передаються до менеджера Wazuh для подальшого аналізу, кореляції з іншими подіями ІБ та візуалізації результатів у Wazuh Dashboard.

Платформа Wazuh є сучасним відкритим рішенням класу SIEM, яке характеризується гнучкістю, сумісністю з різними ОС, а також поєднує функції багаторівневого виявлення загроз і автоматизованого реагування на інциденти.Водночас ефективність її застосування значною мірою залежить від коректної первинної конфігурації та належного налаштування правил безпеки, а також має певні обмеження у виявленні нових загроз без залучення зовнішніх джерел репутаційних даних. Проте, можливість інтеграції Wazuh із зовнішніми аналітичними сервісами для перевірки та кореляції даних створює передумови для побудови комбінованої системи виявлення ШПЗ.

2.2 Принцип дії та функції API сервісу VirusTotal

VirusTotal - це хмарний сервіс аналізу файлів, URL-адрес і IP-доменів, який агрегує результати понад 70 антивірусних рушіїв і систем репутаційного аналізу (рисунок 2.7) [31-34]. Сервіс використовується як для швидкої перевірки файлів, так і для інтеграції у корпоративні системи безпеки через API.

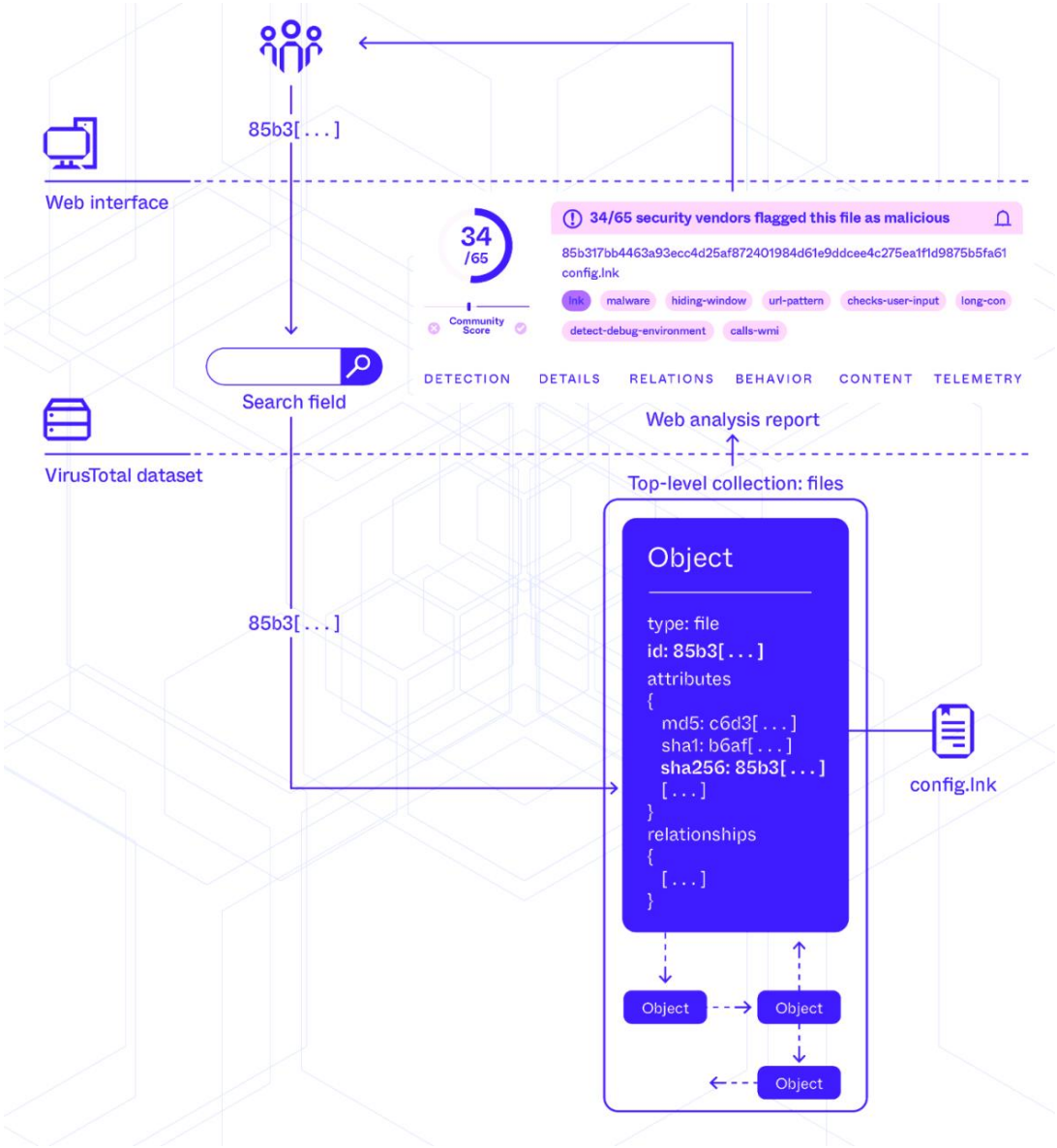


Рисунок 2.7 - Принцип роботи сервісу VirusTotal

Функції онлайн-сервіс включають:

- аналіз файлів і URL-адрес антивірусними рушіями;
- надання репутаційних даних про файли, домени та IP-адреси;

– пошук індикаторів компрометації (IoC) для підозрілих об’єктів;

– API для інтеграції з локальними системами ІБ та автоматизації перевірок.

Під час аналізу файлів VirusTotal обчислює хеш (MD5, SHA1, SHA256) та перевіряє його на наявність у базі відомих зразків. Якщо файл новий, він передається на сканування антивірусними рушіями (ClamAV, Kaspersky, ESET, Bitdefender тощо).

Результати повертаються у вигляді відсотку виявлень та класифікації загрози (рисунок 2.8) [33].

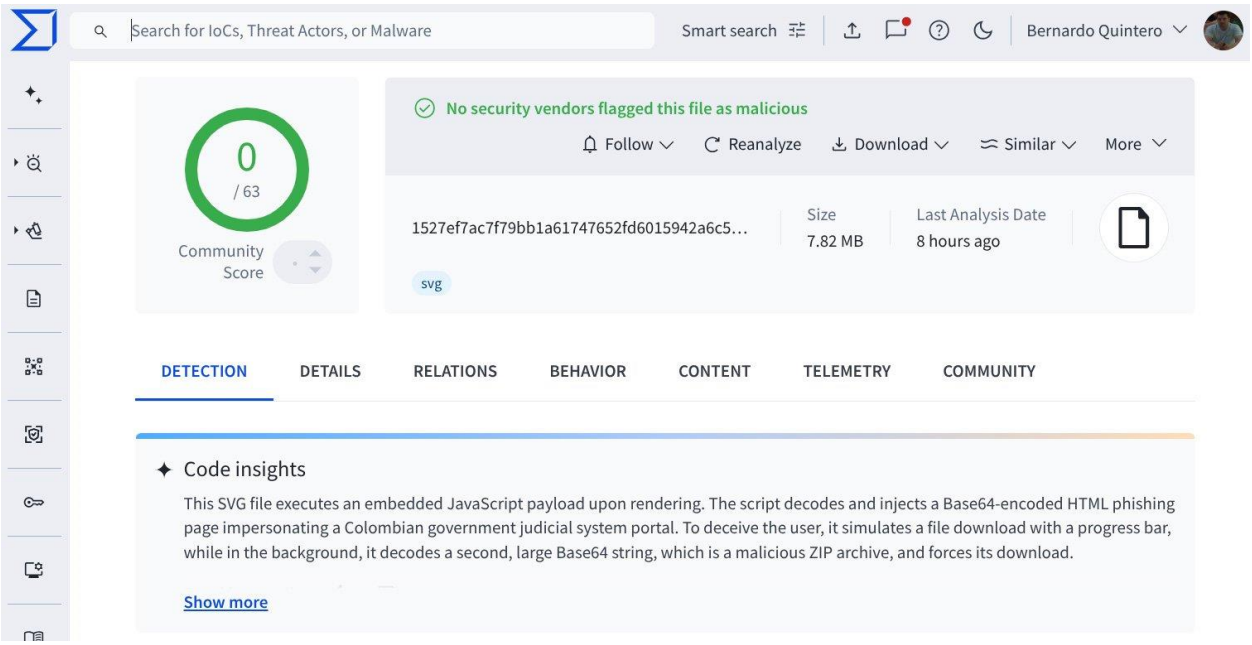


Рисунок 2.8 – Приклад процесу аналізу у сервісі VirusTotal

Система впорядковує інформацію, пов’язану з артефактами, у структуровані об’єкти, кожен із яких має унікальний ідентифікатор, тип і набір атрибутів. Ці об’єкти можуть представляти файли, URL-адреси, домени, IP-адреси тощо, а їхні зв’язки забезпечують контекстні посилання між елементами, що дозволяє відстежувати взаємозалежності між різними компонентами шкідливої інфраструктури (рисунок 2.9) [32].

Основні структурні елементи моделі даних VirusTotal:

– ідентифікатор (Identifier) - унікально визначає об’єкт і є похідним від самого артефакту, наприклад хеш SHA-256 для файлів;

– тип (Type) - вказує категорію інформації, яку представляє об’єкт,

file, URL, domain чи IP address;

– атрибути (Attributes) - містять набір даних, пов'язаних із об'єктом, що можуть бути як примітивними (розмір, дата створення), так і складними (сигнатури, метадані аналізу);

– зв'язки (Relations) - відображають логічні або причинно-наслідкові залежності між об'єктами, наприклад, зв'язок між файлом і доменом, з якого він завантажений.

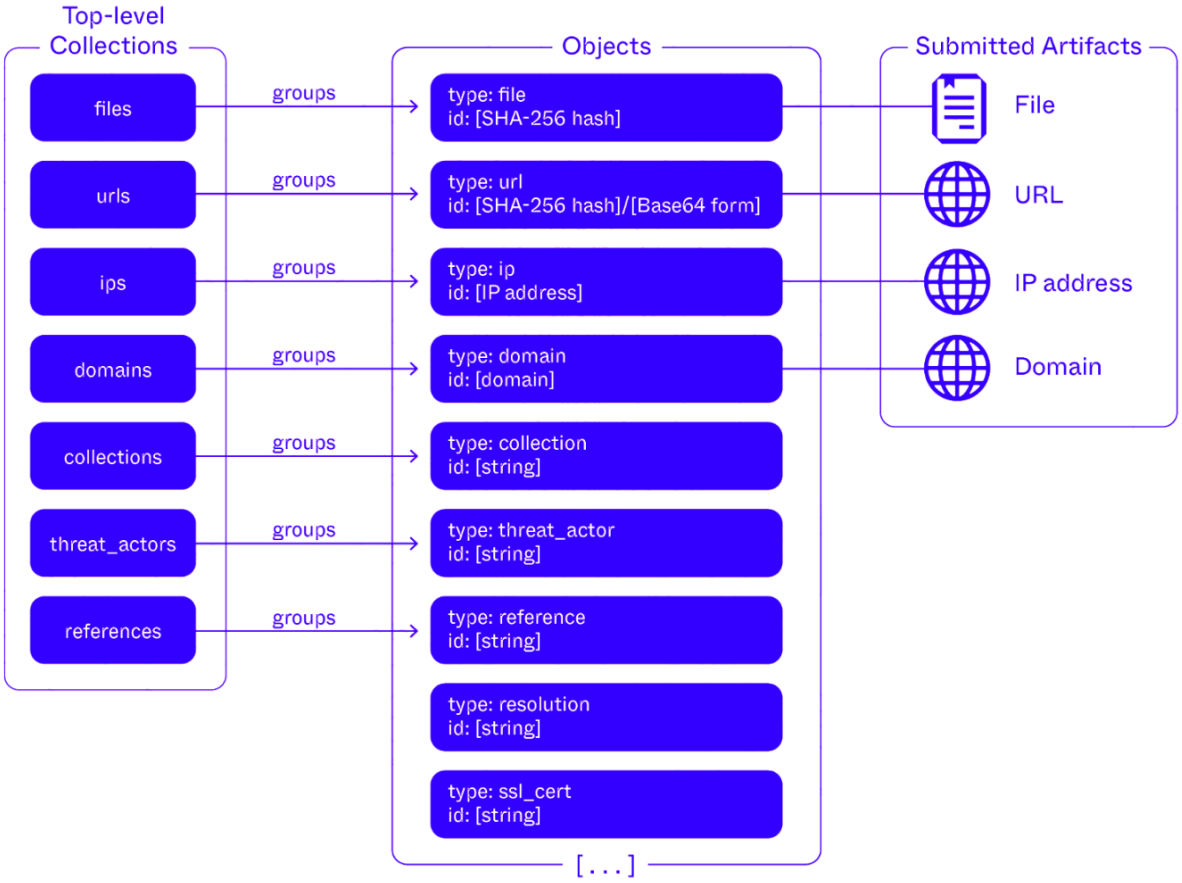


Рисунок 2.9 – Приклад структури наборів даних у VirusTotal

Для URL-адрес виконується перевірка на фішингові шаблони, шкідливі скрипти та зв'язки з відомими доменами C2-серверів. VirusTotal поєднує результати з репутаційних БД різних вендорів, що дозволяє отримати об'єктивну оцінку. Кожен рушій має власну сигнатурну базу, а результати агрегуються у єдиний показник, який відображає рівень довіри. Репутаційна інформація зберігається у хмарі та оновлюється в реальному часі.

Для взаємодії з іншими системами VirusTotal надає RESTful API, який

дозволяє здійснювати перевірку хешів, завантаження файлів, запити статистики та історії аналізу. API-запити здійснюються через HTTP-з'єднання з використанням унікального ключа API Key. Приклад відповіді містить поля positives, total і permalink, що відображають кількість спрацьовувань антивірусів, загальну кількість перевірок та посилання на детальний звіт.

Аналіз показав, що VirusTotal є ефективним сервісом з відкритим API для програмного доступу. Завдяки централізованій БД та хмарній архітектурі він забезпечує швидкий доступ до відомостей про потенційно небезпечні файли й мережеві ресурси. Серед переваг - доступність через API, масштабованість, постійне оновлення даних та підтримка різних типів об'єктів. До обмежень можна віднести відсутність глибокого поведінкового аналізу та залежність від сторонніх антивірусних механізмів.

Перспективним напрямом розвитку є використання VirusTotal у поєднанні з платформами моніторингу подій, зокрема Wazuh, що дозволяє інтегрувати репутаційні дані у процес локального аналізу загроз і підвищити точність виявлення шкідливих дій у реальному часі.

2.3 Порівняльний аналіз функцій платформи моніторингу безпеки та інструменту аналізу шкідливих об'єктів

Аналіз принципів побудови та функціонування системи моніторингу подій ІБ та сервісу аналітики загроз показав, що кожен з інструментів має власну специфіку та сферу ефективного застосування. Для визначення потенційних напрямів інтеграції Wazuh і VirusTotal, з метою підвищення точності та швидкості виявлення зловмисних програм, проведено порівняння їх основних характеристик (таблиця 2.1).

В результаті проведеного аналізу підтверджено, що платформа Wazuh орієнтована на збір подій і кореляцію інцидентів у реальному часі. Сервіс VirusTotal орієнтований на глибокий аналіз об'єктів у хмарному середовищі використовуючи репутаційні та сигнатурні бази. Такий підхід дозволяє

збагачувати дані про потенційно небезпечні артефакти та розширює можливості виявлення загроз. Це підкреслює відмінності у принципах роботи та підтверджує доцільність їх поєднання у єдиній системі.

Таблиця 2.1 – Порівняльна характеристика властивостей Wazuh і VirusTotal

Критерій	Wazuh	VirusTotal
Тип аналізу	Поведінковий, сигнатурний	Репутаційний, сигнатурний
Рівень дії	Локальний	Глобальний
Дані	Події з вузлів	Репутаційні бази, хеші
Режим роботи	Реальний час	Хмарна перевірка
Переваги	Контроль системи	Виявлення нових загроз
Обмеження	Потребує встановлення агентів, обмежений доступ до зовнішніх джерел	Не здійснює моніторинг у реальному часі, залежить від рушіїв

Для більш детального порівняння виконано аналіз функціональних характеристик обох рішень, наведених в таблиці 2.2.

Таблиця 2.2 – Порівняння функціональних характеристик платформи Wazuh і сервісу VirusTotal

Критерій	Wazuh	VirusTotal
Призначення	Моніторинг і кореляція подій	Аналіз файлів, доменів і URL
Джерела даних	Агенти, системні журнали (логи), FIM, мережеві події	Файли, URL, хеші, індикатори компрометації
Виявлення	локальні сигнатури, поведінкові правила, FIM	агреговані AV-дані та репутаційні бази
Реагування	Active Response, автоматичне блокування	Відсутнє безпосереднє реагування

Отримані результати свідчать, що Wazuh виконує активні функції виявлення й реагування на інциденти ІБ та може виступати центральним елементом SIEM-рішень. VirusTotal - доцільно використовувати як зовнішній аналітичний сервіс для перевірки підозрілих об’єктів на основі хмарних репутаційних і сигнатурних баз на етапі аналітики загроз.

У межах реалізації алгоритму виявлення ШПЗ доцільно застосовувати Wazuh як інструмент для первинного моніторингу, збору подій і фіксації аномалій, а VirusTotal для підтвердження або уточнення результатів аналізу, як додатковий етап перевірки та підтвердження шкідливості об’єктів.

Такий підхід дозволяє забезпечити ефективну взаємодію між локальним моніторингом і хмарною аналітикою, підвищити точність підвищити точність виявлення загроз, зменшити час реагування на інциденти ІБ та автоматизувати доповнення індикаторів компрометації, що, у свою чергу, забезпечує більш ефективний захист кінцевих вузлів мережі від відомих і потенційних загроз.

2.4 Особливості інтеграції Wazuh та VirusTotal

Wazuh підтримує інтеграцію з багатьма зовнішніми сервісами через модуль Active Response або API-запити. Для VirusTotal реалізується скрипт vt_lookup.py, який виконує запит до API сервісу, отримує репутаційний результат і передає його у менеджер Wazuh для подальшої кореляції [33-36].

Схема інтеграції подана на рисунку 2.4. Запропоноване рішення включає наступні компоненти:

- Wazuh Agent - встановлюється на кінцеві вузли, здійснює моніторинг та FIM, збір логів і виконання реакційних дій;
- Wazuh Manager - централізований сервер кореляції подій, аналізу правил та генерації сповіщень;
- Wazuh Integrator - модуль, що забезпечує обмін даними з зовнішніми API, зокрема з сервісом VirusTotal;
- VirusTotal API - онлайн-сервіс, який агрегує результати перевірки

файлів антивірусними рушіями, для класифікації загроз та репутаційного аналізу.

– Active Response – механізм Wazuh, що автоматично реагує на загрози: карантин, ізоляція або видалення заражених файлів/процесів;

SIEM / SOC – система централізованого збору логів і управління інцидентами ІБ де відбувається логування подій та створення інцидентів для аналітики та реагування.

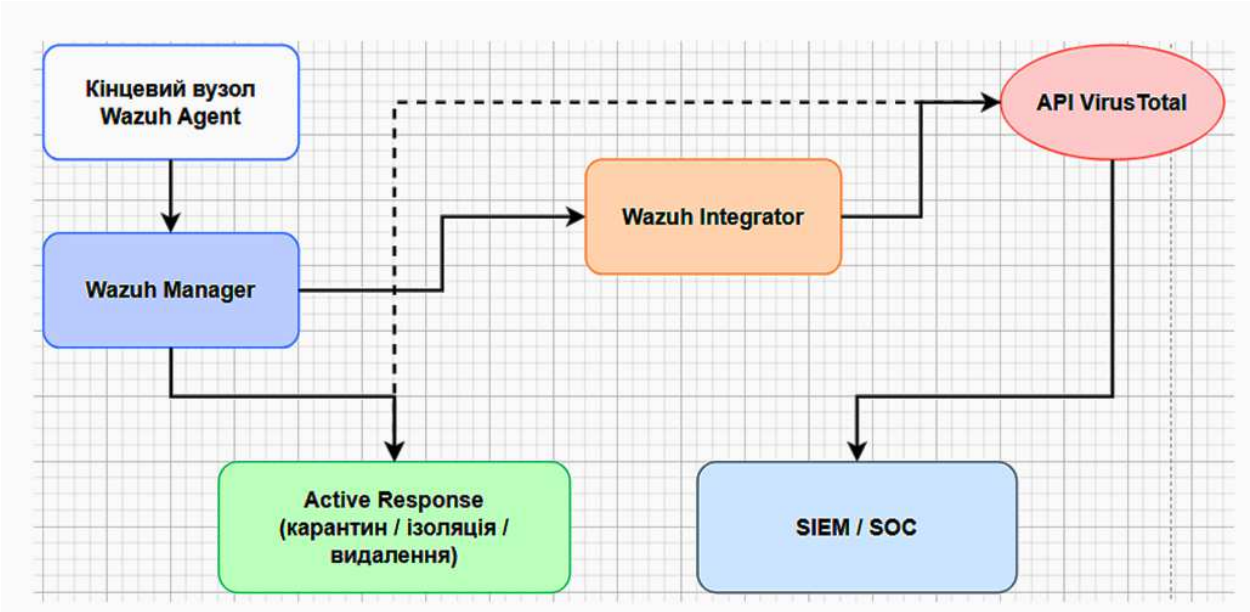


Рисунок 2.10 – Принцип інтеграції Wazuh із сервісом VirusTotal через API

Після виявлення змін у файлі модуль FIM створює подію, яка активує скрипт VirusTotal Lookup. Результат у вигляді кількості виявлень (positives/total) повертається до Wazuh Manager, де порівнюється з пороговим значенням і створюється відповідна подія безпеки.

Крім VirusTotal, Wazuh може використовувати інші зовнішні аналітичні джерела, такі як AlienVault OTX, MISP або AbuseIPDB. Така інтеграція забезпечує багаторівневу оцінку репутації об’єктів і дозволяє виявляти складні атаки, що використовують нові або видозмінені зразки ШПЗ.

Для реалізації інтеграції Wazuh із сервісом VirusTotal необхідно налаштувати відповідну секцію в конфігураційному файлі ossec.conf. У цій секції зазначаються назва інтеграції, API-ключ доступу до VirusTotal,

ідентифікатор правила Wazuh, при спрацюванні якого дані передаються на перевірку, а також формат повідомлень (наприклад, JSON). Завдяки цій конфігурації підозрілі файли або їхні хеші автоматично відправляються до VirusTotal для отримання репутаційної інформації.

Для зменшення навантаження на мережу та сервери VirusTotal рекомендується передавати тільки хеші файлів, а не їх повні копії. Хешу SHA256 або MD5 достатньо для визначення репутації файлу та швидкого зіставлення з БД VirusTotal.

Важливим аспектом є регулярне оновлення правил Wazuh Manager, оскільки нові варіанти ШПЗ постійно з’являються. Без актуальних правил система може пропустити нові загрози.

При обробці результатів перевірки VirusTotal необхідно встановити порогове значення підтвердження шкідливості. VirusTotal повертає кількість антивірусних движків, які виявили файл як шкідливий, і на основі цього показника приймається рішення про реагування. Порогове значення слід визначати експериментально: занадто високий поріг може призвести до пропуску реальних загроз, а занадто низький - до великої кількості хибних спрацьовувань.

У разі використання публічного API VirusTotal необхідно враховувати обмеження швидкості запитів (4 запити на хвилину). Для великих потоків подій рекомендується надсилати тільки критично важливі хеші або використовувати платну версію API з підвищеним лімітом запитів.

Завдяки дотриманню цих рекомендацій інтеграція Wazuh та VirusTotal забезпечує ефективну багаторівневу перевірку підозрілих файлів, швидке реагування на загрози та мінімізацію хибних спрацьовувань.

3. РОЗРОБКА КОМБІНОВАНОГО АЛГОРИТМУ ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Модель взаємодії компонентів

Алгоритм виявлення ШПЗ побудований на основі інтеграції Wazuh та VirusTotal, має на меті забезпечення багаторівневого контролю за подіями ІБ та автоматизованого аналізу підозрілих об’єктів. Концептуально він поєднує можливості SIEM-системи із зовнішнім аналітичним сервісом репутаційної перевірки файлів.

Архітектура реалізована за клієнт-серверною моделлю і включає такі основні компоненти:

- Wazuh Manager - центральний сервер, який приймає та обробляє журнали подій, здійснює кореляцію даних і генерує сповіщення про інциденти.
- Wazuh Agent - програмний компонент, встановлений на клієнтських хостах (робочих станціях, серверах), який збирає події, системні логи, інформацію про запущені процеси, зміни у файлах тощо.
- Filebeat / Logstash / Elasticsearch (стек ELK) - забезпечують зберігання, пошук та візуалізацію подій безпеки, а також інтеграцію з Kibana для відображення аналітичних панелей.
- VirusTotal API - зовнішній сервіс, що використовується для перевірки хешів підозрілих файлів або URL-адрес на наявність відомих ознак шкідливості.
- Модуль інтеграції (custom script або Wazuh Active Response) - проміжний рівень, який автоматизує передачу даних від Wazuh до VirusTotal, обробку відповідей API та формування подій про результати аналізу.

На рисунку 3.1 наведено загальну схему взаємодії компонентів системи для комбінованого виявлення ШПЗ, а в додатку А - її деталізовану структуру.

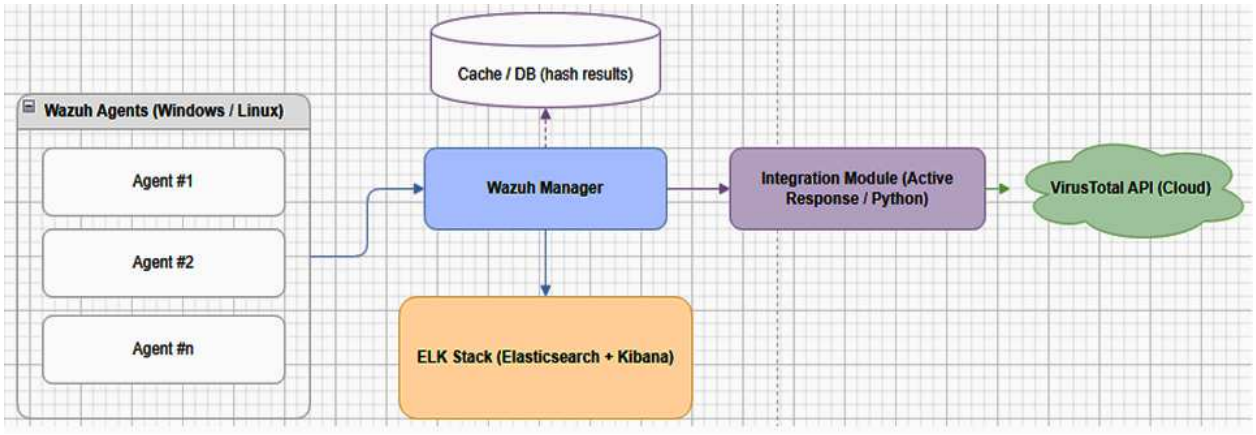


Рисунок 3.1 - Загальна модель взаємодії компонентів системи виявлення ШПЗ на основі комбінованого алгоритму

На схемі виділено основвні типи комунікацій:

- TLS / Internal Events (блакитний) - передача внутрішніх подій та логів між агентами, менеджерами й ELK;
- HTTPS to External API (зелений) - запити до сервісу VirusTotal через проксі;
- Cache / Reputation Storage (синій) - взаємодія з базою кешу;
- Audit / Compliance Logs (червоний) - формування й передача журналів аудиту.
- пунктирними лініями показано допоміжні зв’язки - резервний менеджер, кеш, аудит і легенда.

Архітектура побудована за модульним принципом і забезпечує централізований збір, аналіз, перевірку та відображення подій ІБ у корпоративному середовищі. Вона включає чотири рівні:

1. Рівень агентів (Wazuh Agents) - збір локальних подій ІБ та формування повідомлень.

2. Рівень керування (Wazuh Manager) - аналіз і кореляція подій, генерація сповіщень.

3. Аналітичний рівень (ELK Stack) - рівень зберігання та візуалізації подій.

4. Рівень зовнішнього аналізу (VirusTotal API) - репутаційна перевірка об’єктів.

При виявленні підозрілого файлу Wazuh Manager передає його хеш до інтеграційного модуля, який формує HTTPS-запит до VirusTotal API. Результати аналізу, кількість рушіїв, тип загрози, рівень довіри, повертаються в Wazuh Manager, де зберігаються у локальній базі кешу та відображаються у Kibana Dashboard. Для підвищення ефективності застосовується локальне кешування результатів запитів, що зменшує навантаження на API. Також передбачено резервний вузол Wazuh Manager для забезпечення безперервності моніторингу.

На клієнтських вузлах (робочих станціях або серверах) встановлюються Wazuh Agents, які виконують функції збору даних про події безпеки: системні журнали, зміни у файлах, запущені процеси, підозрілі виконувані об’єкти тощо.

Основним елементом архітектури є Wazuh Manager (Primary) - сервер кореляції подій, що здійснює:

- прийом журналів безпеки від агентів;
- порівняння з бібліотекою сигнатур і поведінкових правил;
- генерацію попереджень (alerts) при виявленні підозрілих активностей.

Для підвищення надійності передбачено резервний вузол Wazuh Manager (Secondary), який синхронізується з основним через канал Cluster sync. Це дозволяє реалізувати високу доступність (HA) і безперервність моніторингу навіть при відмові основного сервера.

Компоненти Elasticsearch і Kibana формують ядро аналітичного шару:

- Elasticsearch відповідає за індексацію, пошук і зберігання подій;
- Kibana забезпечує графічне відображення даних, дашборди для аналізу активностей і статистику виявлених загроз.

Події та попередження надходять до цього рівня у вигляді структурованих повідомлень JSON.

При виявленні невідомого або підозрілого файлу Wazuh Manager ініціює запит до Integration Module, який реалізований на основі Wazuh Active Response або скриптів Python. Модуль формує хеш (MD5/SHA256)

підозрілого об’єкта та через HTTPS-запит звертається до зовнішнього сервісу VirusTotal API. Для захисту зовнішнього трафіку використовується Firewall / Proxy, який фільтрує запити, контролює вихідні з’єднання та застосовує політику доступу. Через нього проходять лише перевірені HTTPS-запити до VirusTotal Cloud. Отримані результати (кількість рушіїв, що виявили шкідливість, тип загрози, рівень довіри) повертаються назад у Wazuh Manager і додаються до відповідних подій ІБ.

Наведена модель може включати також додаткові функціональні модулі:

- Cache / DB (hash results) - локальна БД для кешування результатів запитів до VirusTotal, що знижує навантаження на API та пришвидшує повторну перевірку відомих об’єктів.
- Security Audit & Logs Storage - окреме сховище журналів аудиту, у якому зберігаються всі дії системи, відповіді від API та історія кореляційних подій. Це підвищує рівень підзвітності та забезпечує відповідність політикам безпеки.

При реалізації використовується наступна топологія:

- 1 сервер Wazuh Manager + Elasticsearch + Kibana (центральний вузол SIEM);
- кілька клієнтських вузлів (Wazuh Agents) на базі ОС Windows/Linux;
- мережевий шлюз або firewall, що фільтрує зовнішній трафік;
- доступ до мережі Інтернет для комунікації з сервісом VirusTotal через HTTPS API;
- внутрішня БД для збереження результатів перевірки та хешів аналізованих файлів.

Функціональні зв’язки між компонентами системи представлено таким чином:

- Wazuh Agent фіксує підозрілу активність і надсилає дані до Wazuh Manager (Primary);
- Manager виконує первинний аналіз і, за потреби, передає хеш

об’єкта в Integration Module;

- Модуль інтеграції через Firewall / Proxy надсилає запит до VirusTotal API;
- Відповідь VirusTotal повертається, обробляється та зберігається у Cache/DB;
- Результати додаються до журналу подій і відображаються у Kibana Dashboard;
- Усі дії логуються в Security Audit Storage для подальшого аналізу.

Особливу увагу приділено безпеці самої системи моніторингу.Для комунікації між агентами та сервером використовується захищені канали TLS 1.2+ з автентифікацією за сертифікатами. Для зовнішніх запитів HTTPS, що відповідає вимогам забезпечення цілісності та конфіденційності даних. Доступ до VirusTotal API обмежений через API ключ, який зберігається у зашифрованому вигляді. Для запобігання надмірному навантаженню передбачено кешування результатів перевірок хешів на локальному сервері Wazuh. Усі запити та відповіді логуються для аудиту безпеки.

В результаті, розроблена модель взаємлдії компонентів забезпечує:

- централізований збір та аналіз подій ІБ;
- автоматизоване виявлення підозрілих об’ктів;
- перевірку файлів у глобальній базі VirusTotal;
- візуалізацію результатів та оцінку рівня загроз у реальному часі.

Запропонована модель є гнучкою, розширюваною та може масштабуватися до корпоративного рівня, що дозволяє застосовувати її як частину комплексної системи кіберзахисту.

Представлене рішення взаємодії між Wazuh і VirusTotal формує комбінований алгоритм виявлення шкідливого ПЗ, який поєднує сигнатурний і поведінковий аналіз на рівні Wazuh, репутаційний аналіз через VirusTotal, аудит і кешування результатів для підвищення ефективності. Завдяки цьому забезпечується своєчасне виявлення загроз, скорочення кількості хибних спрацьовувань і можливість інтеграції в існуючі SIEM-рішення корпоративного рівня.

3.2 Комбінований алгоритм виявлення загроз

Розроблений алгоритм комбінованого аналізу призначений для підвищення точності виявлення ШПЗ шляхом одночасного використання кількох джерел аналітичних ознак - сигнатурних, поведінкових та репутаційних. Його реалізація базується на інтеграції платформи Wazuh та автоматизованої взаємодії з сервісом VirusTotal через механізм Active Response.

Алгоритм ґрунтується на принципі зваженої агрегації показників ризику, коли кожен із трьох методів виявлення формує часткову оцінку ймовірності шкідливості об’єкта. На основі цих оцінок розраховується інтегральний бал, що визначає рівень загрози та рішення системи - створення інциденту, попередження чи інформаційне логування.

Сигнатурна складова забезпечується правилами Wazuh (ruleset), які ідентифікують відомі зразки ПЗ за хеш-значеннями або характерними ознаками. Поведінкова складова аналізує дії процесів, зміну системних файлів, створення автозапусків, нетипову мережеву активність. Репутаційна складова використовує API сервісу VirusTotal для перевірки SHA256-хешів і визначення частки антивірусних рушіїв, що позначили об’єкт як шкідливий.

Після отримання усіх трьох оцінок система виконує нормалізацію показників у діапазоні від 0 до 1 та обчислює зважений інтегральний бал:

$$Score = w_{sig}S_{sig} + w_{beh}S_{beh} + w_{rep}S_{rep},$$

де S_{sig}, S_{beh}, S_{rep} - сигнатурна, поведінкова та репутаційна оцінки відповідно,

w_{sig}, w_{beh}, w_{rep} - вагові коефіцієнти, встановлені експериментально.

На основі інтегрального балу формується рішення:

- $Score \geq 0{,}7$ - висока ймовірність шкідливості;
- $0{,}4 \leq Score > 0{,}7$ - підозріла активність;
- $0{,}4 < Score$ - подія вважається безпечною, інформація фіксується у журналі.

Реалізація алгоритму здійснена за принципом подієвого конвеєра.

Кожна подія безпеки, отримана від агента Wazuh, проходить етапи попередньої обробки - збагачення атрибутами файлу (шлях, хеш), обчислення часткових оцінок, звернення до локального кешу результатів VirusTotal, формування об’єднаного бала та, за потреби, створення попередження. У разі відсутності даних у кеші ініціюється запит до API VirusTotal, результати якого зберігаються локально на період 7 днів для зменшення навантаження на зовнішній сервіс.

На рисунку 3.2 наведено структурну блок-схему комбінованого алгоритму виявлення ШПЗ.

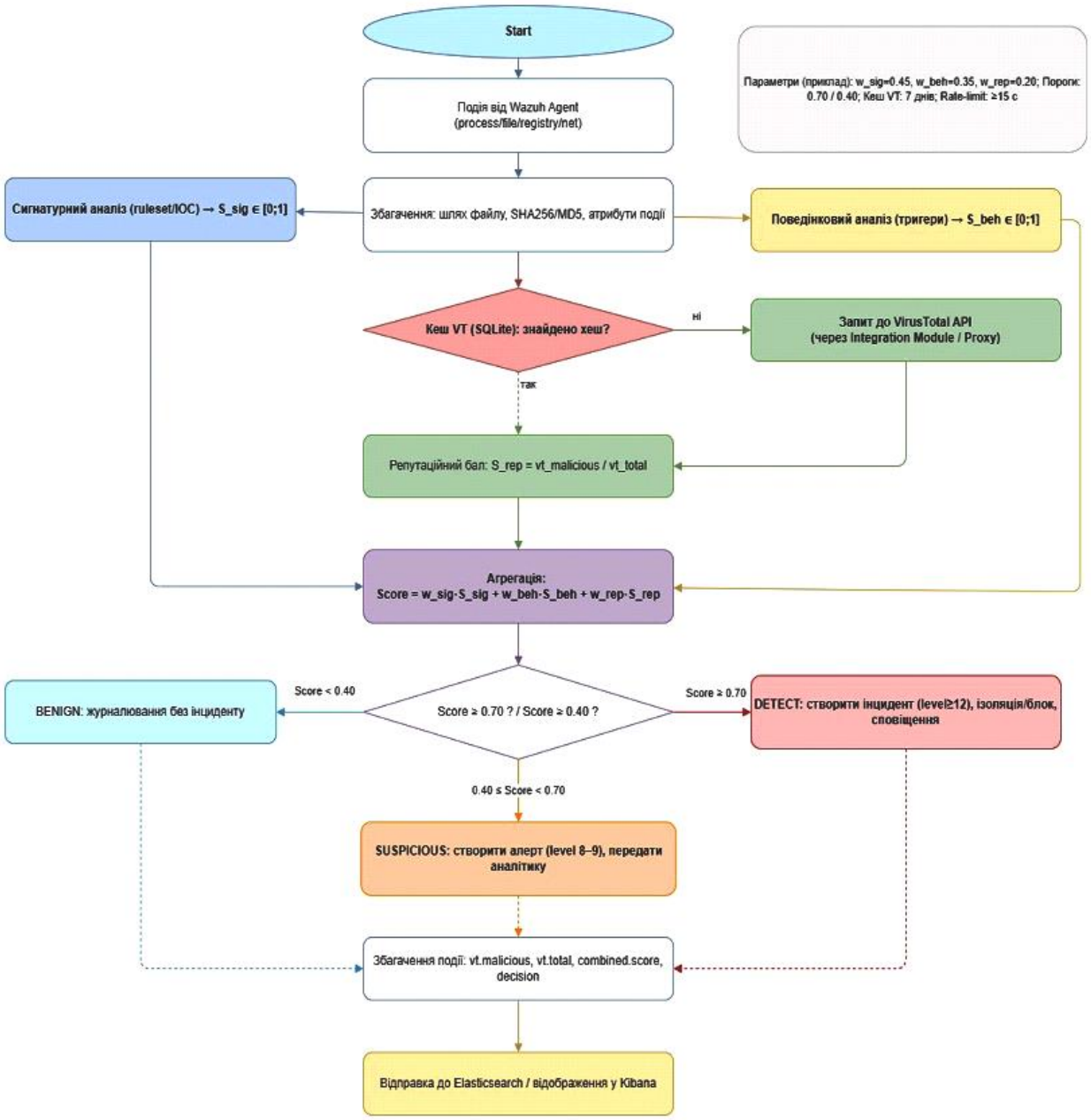


Рисунок 3.3 – Схема комбінованого алгоритму виявлення ШПЗ

У межах реалізації створено набір локальних правил Wazuh для ідентифікації поведінкових та сигнатурних індикаторів, а також скрипт vt_lookup.py, який реалізує обмін з VirusTotal, опис і вихідний код наведено у додатку Б.

Схема (рисунок 3.3) відображає основні етапи роботи системи: отримання події від агента Wazuh, збагачення її атрибутами, проведення сигнатурного, поведінкового та репутаційного аналізів, обчислення інтегрального показника *Score* і прийняття рішення щодо рівня загрози.

Алгоритм передбачає перевірку наявності даних у локальному кеші результатів VirusTotal, що зменшує затримку обробки подій. Якщо хеш не знайдено, ініціюється безпечний запит до хмарного API VirusTotal, після чого обчислюється репутаційна оцінка S_{rep}.

На підставі порогових значень $Score$ подія автоматично класифікується наступним чином:

- DETECT - створюється інцидент, виконується ізоляція або блокування;
- SUSPICIOUS - формується попередження для аналітика;
- BENIGN - подія журналюється без створення інциденту.

Збагачені результати передаються до Elasticsearch для подальшої візуалізації у Kibana, що забезпечує прозорість і повторюваність процесу детекції.

На рисунку 3.4 показано структуру директорій компонентів, що забезпечують інтеграцію Wazuh із сервісом VirusTotal.

```
wazuh_vt_integration
├── active-response/
│   └── bin/vt_lookup.py
├── etc/
│   ├── ossec.conf
│   ├── decoders/local_decoder.xml
│   └── rules/local_rules.xml
└── systemd/
    └── wazuh-manager.service.d/env.conf
```

Рисунок 3.4 - Файлова структура реалізованої системи

Така організація дозволяє ізолювати логіку Active Response, конфігураційні файли та системні змінні, забезпечуючи зручність розгортання та безпеку зберігання ключів.

Запропонований алгоритм дозволяє:

- поєднувати сигнатурні та поведінкові методи виявлення у межах єдиної системи кореляції подій;
- використовувати зовнішню репутаційну інформацію без ризику розкриття внутрішніх артефактів (передаються лише хеші файлів);
- гнучко налаштовувати пороги чутливості й вагові коефіцієнти залежно від типу середовища;
- забезпечити відтворюваність результатів і можливість подальшого масштабування.

Практичне впровадження комбінованого алгоритму у середовищі Wazuh показало, що поєднання трьох типів аналізу дозволяє суттєво знизити кількість хибних спрацьовувань та покращити точність класифікації інцидентів ІБ.

У процесі калібрування оптимальними виявились співвідношення

$$w_{sig} = 0.45, w_{beh} = 0.35, w_{rep} = 0.2.$$

Алгоритм придатний для подальшої оптимізації шляхом адаптивного налаштування вагових коефіцієнтів і розширення бази поведінкових тригерів.

Діаграма роботи, що відображає послідовність взаємодії компонентів у процесі роботи комбінованого алгоритму наведена в додатку В. Вона містить деталізовану послідовність виконання етапів комбінованого аналізу та взаємодію між компонентами системи. Діаграма демонструє рух події від агента Wazuh до системи візуалізації Kibana, розподілений за функціональними доріжками (Wazuh Agent, Wazuh Manager, Integration/VirusTotal, Scoring & Decision, ELK Stack). Кожна доріжка відповідає за свій етап аналізу або обробки даних, що дозволяє візуалізувати архітектурний потік та логіку прийняття рішень у системі.

3.3 Конфігурація та налаштування

Реалізація комбінованого алгоритму виявлення ШПЗ потребує узгодженої конфігурації компонентів платформи Wazuh, інтеграційного середовища та сервісу VirusTotal.

Рішення базується на моделі взаємодії «централізований менеджер - розподілені агенти». На клієнтських вузлах інстальовано Wazuh Agent, який збирає події безпеки та передає їх на Wazuh Manager. Менеджер виконує сигнатурний і поведінковий аналіз, ініціює звернення до інтеграційного модуля (Active Response), який комунікує із VirusTotal API, після чого результати записуються до сховища Elasticsearch і відображаються у Kibana.

Основні елементи реалізації системи розміщено у директоріях менеджера Wazuh (рисунок 3.5).

```
/var/ossec/
 ├── active-response/
 │    └── bin/vt_lookup.py          - скрипт інтеграції з VirusTotal
 ├── etc/
 │    ├── rules/local_rules.xml     - локальні сигнатурні й поведінкові правила
 │    ├── decoders/local_decoder.xml - користувацькі декодери подій
 │    └── ossec.conf                - основна конфігурація менеджера
 └── var/vt_cache.sqlite3           - кеш результатів VirusTotal
```

Рисунок 3.5 - Файлова структура реалізованої системи комбінованого виявлення на платформі Wazuh

Для реалізації обміну з сервісом VirusTotal у конфігураційному файлі ossec.conf визначається нова команда Active Response, яка запускає скрипт vt_lookup.py при спрацюванні певних правил:

```
<ossec_config>
  <command>
    <name>vt_lookup</name>
    <executable>vt_lookup.py</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>
```

```
  <active-response>
    <command>vt_lookup</command>
    <location>manager</location>
    <rules_id>940001,940101,940102,950100</rules_id>
  </active-response>
</ossec_config>
```

У даному прикладі Active Response спрацьовує при сигнатурних та поведінкових подіях (ідентифікатори правил 940001–950100). Менеджер Wazuh передає дані про файл у стандартному форматі JSON до скрипта vt_lookup.py, який виконує запит до VirusTotal і повертає результат для подальшого збагачення події.

Для запобігання компрометації ключа доступу до сервісу VirusTotal використано механізм системних змінних середовища. У каталозі служби Wazuh створюється файл:

```
/etc/systemd/system/wazuh-manager.service.d/env.conf
```

зі змістом:

```
[Service]
Environment="VT_API_KEY=ВАШ_КЛЮЧ_VIRUSTOTAL"
```

Після цього застосовується перезавантаження конфігурації служби:

```
sudo systemctl daemon-reload
sudo systemctl restart wazuh-manager
```

Таким чином, ключ передається процесу менеджера динамічно й не зберігається у відкритих конфігураційних файлах.

Локальні правила у файлі **`local_rules.xml`** визначають логіку реакції на сигнатурні та поведінкові індикатори. Вони також викликають комбіноване правило з підвищеним рівнем небезпеки, якщо декілька поведінкових тригерів збігаються одночасно:

```
<rule id="950100" level="12">
  <if_matched_sid>940101,940102</if_matched_sid>
  <description>Combined behavior indicates malware</description>
  <group>combined,behavior</group>
</rule>
```

Для обробки додаткових полів подій (наприклад, хешів) може бути створений простий декодер у файлі local_decoder.xml, який виділяє SHA256-значення для подальшого аналізу.

Скрипт vt_lookup.py зберігає результати останніх запитів у БД SQLite (vt_cache.sqlite3) з терміном життя записів сім днів. Це дає змогу уникнути повторних звернень до VirusTotal і зменшує навантаження на API-сервери. Крім того, реалізовано обмеження частоти запитів (rate-limit) - не частіше ніж один запит кожні 15 с, що відповідає політиці безкоштовного API-доступу.

Після розгортання конфігурації здійснюється тестування роботи алгоритму:

- Імітація спрацювання правила шляхом створення тестового файлу з відомим шкідливим хешем (наприклад, EICAR test file).
- Перевірка появи записів у журналі /var/ossec/logs/alerts/alerts.json.
- Перегляд результатів у Kibana: у події повинні відображатись поля vt.malicious, vt.total, combined.score, combined.decision.
- Підтвердження, що при високому значенні Score створюється інцидент рівня 12, а при низькому - лише інформаційне повідомлення.

Результатом конфігурації є те, що налаштована система забезпечує:

- повністю автоматизоване виявлення підозрілих дій;
- централізовану обробку та кореляцію подій;
- інтеграцію з VirusTotal без передачі оригінальних файлів, лише хешів;
- надійне зберігання результатів у БД ELK;
- візуалізацію у Kibana з відображенням оцінки ризику за комбінованим балом.

Таким чином, конфігурація підтверджує практичну реалізацію запропонованої методики комбінованого аналізу шкідливого ПЗ, описаної у попередньому підрозділі.

Перелік конфігураційних файлів і скриптів наведено у додатку Б.

3.4 Дослідження алгоритму виявлення шкідливих програм

Метою дослідження є перевірка ефективності запропонованого комбінованого алгоритму виявлення ШПЗ, реалізованого на платформі Wazuh з інтеграцією сервісу VirusTotal. Основна увага приділяється наступним критеріям:

- оцінка точності класифікації;
- зниження кількості хибних спрацьовувань;
- продуктивність роботи у реальному середовищі.

Для дослідження було створено тестове середовище, яке складається з одного менеджера Wazuh та трьох агентів, розгорнутих на віртуальних машинах Ubuntu 22.04. На агентах виконувались контрольовані дії, що імітують як легітимну активність, так і поведінку шкідливих програм. У вибірку увійшли:

- 20 прикладів зразків ШПЗ, серед яких - тестові віруси (EICAR, Mimikatz, Meterpreter, Netcat);
- 20 прикладів безпечних програм, зокрема системні утиліти, офісні застосунки та мережеві клієнти.

Кожна подія, зафіксована агентом, проходила три рівні обробки - сигнатурний, поведінковий і репутаційний. Результати зберігались у БД Elasticsearch, що дозволило проводити подальший статистичний аналіз.

Для оцінювання якості алгоритму використано наступні показники виявлення кількості:

- TP (True Positive) - подій, які правильно ідентифіковані як шкідливі;
- FP (False Positive) - хибних спрацьовувань;
- TN (True Negative) - коректно визначених безпечних подій;
- FN (False Negative) - невиявлених загроз.

На основі наведених показників обчислювались такі характеристики:

- $Precision$ – точність, відображає наскільки достовірними є спрацьовування системи:

$$Precision = \frac{TP}{TP + FP};$$

– $Recall$ – повнота, показує, яку частку реальних загроз система змогла виявити:

$$Pecall = \frac{TP}{TP + FN};$$

– $Fscore$ поєднує точність і повноту в один узагальнений показник:

$$Fscore = \frac{2 \cdot Precision \cdot Pecall}{Precision \cdot Pecal}.$$

У контексті кібербезпеки $Precision$ характеризує надійність системи при фіксації інцидентів, $Pecall$ - її здатність не пропускати загрози, а $Fscore$ - баланс між кількістю хибних спрацьовувань і пропущених загроз. В таблиці 3.1 наведено результати порівняльного тестування, яке проводилось у трьох режимах.

Таблиця 3.1 – Результати тестування

Режим аналізу	$Precision$	$Pecall$	$Fscore$	FP
Сигнатурний	0.89	0.78	0.83	5
Поведінковий	0.81	0.84	0.82	8
Комбінований	0.93	0.91	0.92	2

Як видно з таблиці 3.1, комбінований алгоритм, що інтегрує сигнатурний, поведінковий і репутаційний аналізи, забезпечив найвищі значення за всіма показниками, що свідчить про його високу ефективність. Завдяки використанню зовнішньої репутаційної інформації VirusTotal кількість хибних позитивних результатів зменшилась майже втричі у порівнянні з окремими методами аналізу.

На рисунку 3.6 наведено порівняння основних показників для різних методів аналізу.

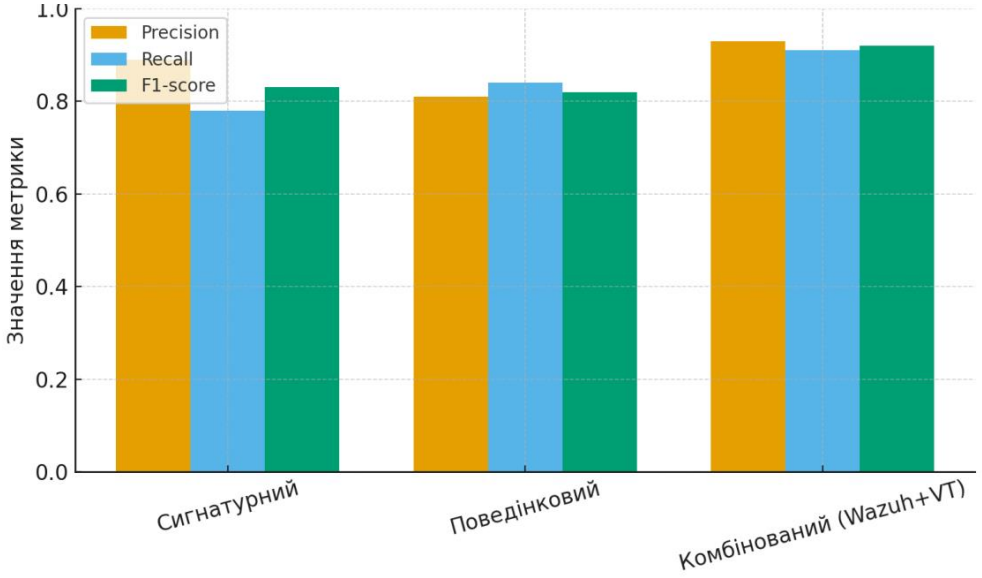


Рисунок 3.6 - Порівняння ефективності методів ШПЗ

Як видно з рисунку 3.6, комбінований алгоритм демонструє кращі результати за всіма критеріями, забезпечуючи баланс між мінімізацією хибних спрацьовувань і повнотою виявлення.

На рисунку 3.7 показано приклад результатів комбінованого аналізу у Kibana, де відображено події з різними значеннями Score, пунктирними лініями позначено пороги 0,4 та 0,7, що відповідають рівням SUSPICIOUS і DETECT та наведено прийняте системою рішення.

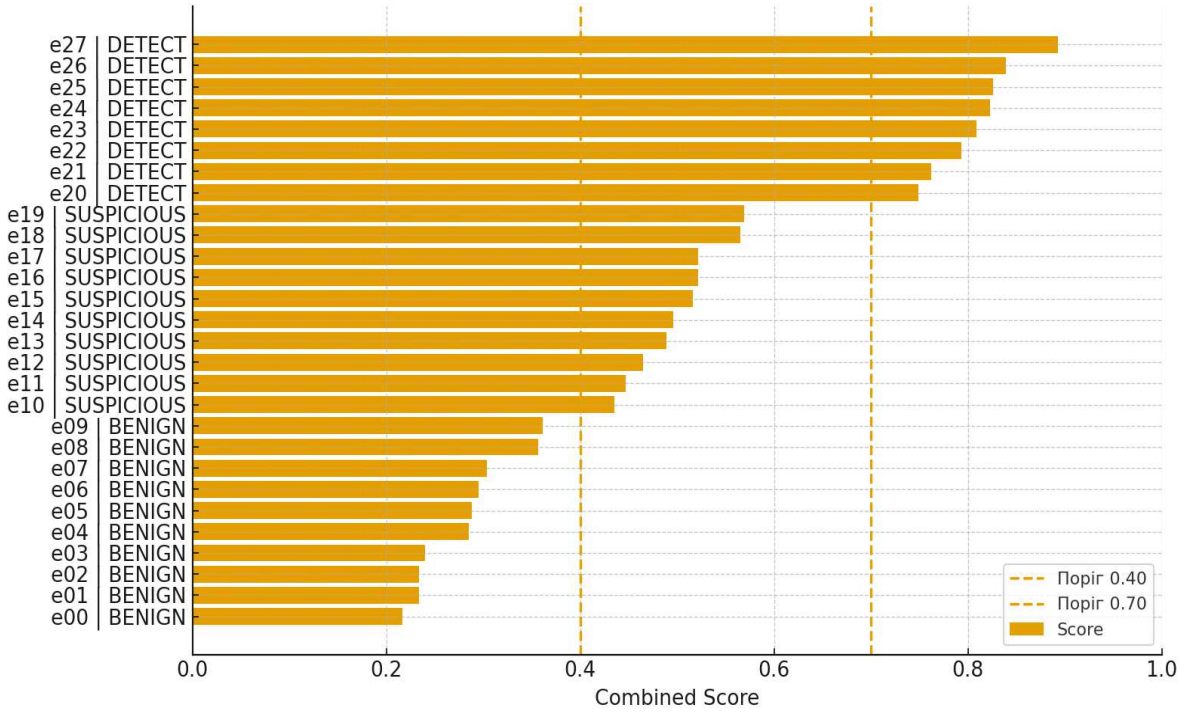


Рисунок 3.7 - Приклад візуалізації результатів комбінованого аналізу

Візуалізація результатів (рисунок 3.7) показала, що події з високими значеннями інтегрального балу Score ≥ 0,70 чітко корелюють із відомими зразками шкідливого коду, тоді як більшість легітимних дій отримали оцінку нижче 0,4, що свідчить про правильну калібровку вагових коефіцієнтів та порогів прийняття рішень.

У таблицях 3.2 та 3.3 наведено приклад фрагмента журналу подій, отриманого після роботи комбінованого алгоритму виявлення ШПЗ. Такий формат даних відповідає вигляду подій у модулі Kibana Discover, де аналітик може відфільтрувати інциденти за рівнем загрози, значенням Score або хостом.

Таблиця 3.2 - Ідентифікаційні дані зафіксованих Wazuh подій

№	Час події (UTC)	Вузол (host)	Файл / процес	SHA256
1	2	3	4	5
1	24.10.2025 10:01	agent-01	C:\Users\Admin\AppData\Temp\eicar.com	275a021bbfb6481d7e8efc0b0d4a22d0f181a8549b8b3b4c3e4f1b6c8f4b9d5f
2	24.10.2025 10:02	agent-02	/usr/bin/curl	9e107d9d372bb6826bd81d3542a419d6d1872b6d9e7d3e8c3a1a0a8d7f0e5a3b
3	24.10.2025 10:03	agent-03	C:\Windows\System32\wscript.exe	c0e84e870874dd37ed0d164c7986f03a3f9a3e3e3c87a2b1b5d59f3b9a9a3d14
4	24.10.2025 10:04	agent-01	/tmp/run.sh	a2eab9c5f5cf47df05ee0f5682b22a8c2e884a65f3b18d2e43c7d3e4f0f5c8a1
5	24.10.2025 10:05	agent-03	C:\ProgramData\svchost32.exe	b6f9a69e93c1e3b84e7c1a2f3e8d94d1b6a7d2e1c3a4f7b8e9f1c5d8a3e2b6d1

Продовження таблиці 3.2

1	2	3	4	5
6	24.10.2025 10:05	agent-02	/usr/lib/systemd/systemd-journald	f5a3c4e8b9a6d3c2e7a4b8f1c5d6a9e3b7a8f2c1e4d9b6a3f7c5e8a1d3b2f6c9
7	24.10.2025 10:06	agent-01	/tmp/installer.sh	1d4b78d39fbc67a9e1f8b3a4d5e9c2f7a0e8c3d9b1f4a7c5e6d2b9f3a8e5c4d1
8	24.10.2025 10:07	agent-03	C:\Users\Public\Music\svchost.exe	e3c9d4f7b8a1e6c3d2f9a4b7c5e8d1a9f6b3c7a2e4f5b8d9a1c6e3f2b7d5a4c8
9	24.10.2025 10:08	agent-02	/opt/test/updater.py	a8b3d5e9f4c7a2b1d6e3f5a4b9c7e2a1d3b5c8f7a4e9b1d2f3c6a5b8d7e4c9a2
10	24.10.2025 10:08	agent-01	C:\Windows\Temp\update.exe	9b8a5d6e4c3f2b1a7d9e8c5f3a2b4c6d5e1a9b8c7d4f2e3c1a6b5e7f9d8c3a4

Таблиця 3.2 містить метадані подій, що були об’єктами аналізу в системі Wazuh. Кожен запис містить часову мітку події, ідентифікатор вузла (host), шлях до виявленого файлу та унікальний SHA256-хеш, який ідентифікує файл без передачі його вмісту, а таблиця 3.3 - результати комбінованого аналізу для тих самих подій.

Таблиця 3.3 відображає результати перевірки отримані з результатів перевірки у сервісі VirusTotal - значення полів VT malicious і VT total, поле combined.score відображає зважену оцінку ймовірності шкідливості, сформовану за результатами сигнатурного, поведінкового та репутаційного аналізів, а також прийняте рішення (decision).

Таблиця 3.3 – Результати комбінованого аналізу для подій

№	VT malicious	VT total	Combined Score	Рішення (decision)
1	67	70	0.96	DETECT
2	0	62	0.18	BENIGN
3	1	65	0.27	BENIGN
4	12	67	0.62	SUSPICIOUS
5	48	68	0.88	DETECT
6	0	70	0.22	BENIGN
7	6	64	0.47	SUSPICIOUS
8	54	70	0.91	DETECT
9	5	65	0.44	SUSPICIOUS
10	0	70	0.19	BENIGN

Час обробки однієї події без запиту до VirusTotal становив у середньому 45-60 мс. При необхідності зовнішнього запиту середній час зростав до 1,3 с, що зумовлено затримками HTTP-виклику. Використання локального кешу дозволило зменшити кількість зовнішніх звернень на 72 %, що значно покращило загальну пропускну здатність системи.

Середня швидкість обробки подій на одному менеджері склала близько 320 подій за секунду, що підтверджує можливість масштабування рішення для середніх корпоративних середовищ без зниження ефективності.

Отримані результати експериментів демонструють високу ефективність запропонованого комбінованого алгоритму:

- поєднання сигнатурного, поведінкового та репутаційного аналізів дозволяє підвищити точність ідентифікації до 92 %;
- кількість хибних спрацьовувань зменшується у 2-3 рази порівняно з традиційними методами;
- використання кешування знижує навантаження на зовнішній API та забезпечує стабільність роботи системи;
- архітектура інтеграції з VirusTotal може бути масштабована без змін у логіці обробки подій.

Проведене дослідження підтверджує практичну доцільність використання запропонованого алгоритму виявлення ШПЗ за допомогою платформи Wazuh та VirusTotal для підвищення надійності систем моніторингу безпеки.

ВИСНОВКИ

Проведено аналіз сигнатурного, евристичного, поведінкового методів аналізу шкідливого коду та гібридних підходів до виявлення ШПЗ, що дозволило визначити їх переваги та обґрунтувати вибір інструментів для реалізації комбінованого аналізу загроз.

Проведено порівняльний аналіз існуючих антивірусних платформ і систем реагування на загрози, отримані результати стали теоретичною базою для розробки ефективної моделі взаємодії компонентів системи детекції шкідливих об’єктів.

Досліджено архітектуру, принципи роботи та функціональні можливості платформи Wazuh і сервісу VirusTotal з метою виявлення можливостей взаємного доповнення їх функцій для підвищення ефективності виявлення шкідливих програм. З’ясовано, що комбінація досліджених інструментів дозволяє поєднувати локальний контроль цілісності файлів і поведінковий аналіз процесів з глобальними базами сигнатур та репутаційними даними, забезпечуючи багаторівневу перевірку загроз.

Розроблено модель взаємодії компонентів системи виявлення та аналізу загроз із використанням Wazuh і VirusTotal для реалізації інтегрованого підходу до виявлення шкідливого програмного забезпечення, який використовує різні методи аналізу одночасно

Реалізовано алгоритм виявлення ШПЗ, що дозволяє поєднувати різні методи аналізу та забезпечує ефективну ідентифікацію підозрілих об’єктів й оперативне реагування на події безпеки. Спільне використання платформи Wazuh та VirusTotal забезпечує повний цикл виявлення загроз - від локального моніторингу до репутаційного аналізу.

Проведено налаштування та тестування розробленого алгоритму, який продемонстрував високу ефективність виявлення загроз у порівнянні з використанням окремих інструментів, забезпечивши оперативне реагування на інциденти безпеки та багаторівневу перевірку підозрілих об’єктів у реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abdullah M., Nawaz M.M., Saleem B., Zahra M., Ashfaq E.b., Muhammad Z. Evolution Cybercrime - Key Trends, Cybersecurity Threats, and Mitigation Strategies from Historical Data. Analytics, 2025, 4, 25. https://doi.org/10.3390/analytics4030025

2. Saeed S., Altamimi S.A., Alkayyal N.A., Alshehri E., Alabbad D.A. Digital transformation and cybersecurity challenges for businesses resilience: Issues and recommendations. Sensors, 2023, 23, 6666.

3. Federal Bureau of Investigation, Internet Crime Report 2023. 2023.- [Електронний ресурс].- Режим доступу: https://www.ic3.gov

4. Lian Z., Shi P., Chen M. A Survey on Cyber-Attacks for Cyber-Physical Systems: Modeling, Defense and Design. IEEE Internet Things J. 2024, 12, 1471–1483.

5. Aslan Ö., Aktuğ S.S., Ozkan-Okay M., Yilmaz A.A., Akin E. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics, 2023, 12, 1333

6. Vladov S., Vysotska V., Varlakhov V., Nazarkevych M., Bolvinov S., Piadyshev V. Innovative Method for Detecting Malware by Analysing API Request Sequences Based on a Hybrid Recurrent Neural Network for Applied Forensic Auditing. Appl. Syst. Innov, 2025, 8, 156. https://doi.org/10.3390/asi8050156

7. Ryu D, Lee S, Yang S, Jeong J, Lee Y, Shin D. Enhancing Cybersecurity in Energy IT Infrastructure Through a Layered Defense Approach to Major Malware Threats. Applied Sciences, 2024, 14(22):10342. https://doi.org/10.3390/app142210342

8. Zhang S., Gao M., Wang L., Xu S., Shao W., Kuang R. A Malware-Detection Method Using Deep Learning to Fully Extract API Sequence Features. Electronics, 2025, 14, 167. https://doi.org/10.3390/electronics14010167

9. Sherazi S.N.A., Qureshi A. Hybrid Analysis Model for Detecting Fileless Malware. Electronics, 2025, 14, 3134. https://doi.org/10.3390/electronics14153134

10. Гавриш Б.М., Тимченко О.В., Борзов Ю.О., Кобевко А.Т. Класифікація шкідливого програмного забезпечення та основні методи захисту.- Комп'ютерні технології друкарства, 2022, 2(48).- с. 142-154.

11. Aggarwal S., Di Troia, F. Malware classification using dynamically extracted api call embeddings. Applied Sciences, 2024, 14(13), 5731.

12. Berrios S., Leiva D., Olivares B., Allende-Cid H., Hermosilla, P. Systematic review: malware detection and classification in cybersecurity. Applied Sciences, 2025, 15(14), 7747.

13. Kamdan Pratama Y., Munzi R.S., Mustafa A.B., Kharisma I.L. Static Malware Detection and Classification Using Machine Learning: A Random Forest Approach. Engineering Proceedings. 2025; 107(1):76. https://doi.org/10.3390/engproc2025107076

14. Міжнародний стандарт ISO/IEC 27032.- [Електронний ресурс].- Режим доступу: https://www.iso.org/ru/standard/76070.html

15. NIST SP 800-83 Rev. 1 Guide to Malware Incident Prevention and Handling for Desktops and Laptops.- [Електронний ресурс].- Режим доступу: https://csrc.nist.gov/pubs/sp/800/83/r1/final

16. Qomariah N., Alwi E.I., Asis M.A. Analisis Malware Hummingbad Dan Copycat Pada Android Menggunakan Metode Hybrid. Cyber Security dan Forensik Digital, 2023, 6(2), 39-47.

17. Nithish N., Murugan R. Cypher Shield: A Simple Approach to Foil Reverse Engineering. International Journal of Innovative Research in Computer and Communication Engineering. 12. 2024. 1042-1046. 10.15680/IJIRCCE.2024.1202053.

18. Malware Analysis 101. What is malware analysis and how to respond to it?.- [Електронний ресурс].- Режим доступу: https://infosecwriteups.com/malware-analysis-101-ac6d55092c8d

19. Ashawa M., McGregor R., Owoh N.P., Osamor J., Adejoh J. Static and Dynamic Malware Analysis Using CycleGAN Data Augmentation and Deep Learning Techniques. Appl. Sci. 2025, 15, 9830. https://doi.org/10.3390/app15179830

20. Alani M.M., Alawida M. Behavioral Analysis of Android Riskware Families Using Clustering and Explainable Machine Learning. Big Data Cogn. Comput. 2024, 8, 171. https://doi.org/10.3390/bdcc8120171

21. AV-TEST. Malware Statistics & Trends Report, AV Test Malware Statistics.- [Електронний ресурс].- Режим доступу: : https://www.av-test.org/en/statistics/malware (accessed on 18 December 2023).

22. AV-ATLAS. Malware & PUA.- [Електронний ресурс].- Режим доступу: https://portal.av-atlas.org/malware (accessed on 18 December 2023).

23. Szczepaniuk E.K., Szczepaniuk H. Cybersecurity of Smart Grids: Requirements, Threats, and Countermeasures. Energies 2025, 18, 5017. https://doi.org/10.3390/en18185017

24. Bhardwaj A., Sapra L., Rahman S. Elasticsearch-Based Threat Hunting to Detect Privilege Escalation Using Registry Modification and Process Injection Attacks. Future Internet 2025, 17, 394. https://doi.org/10.3390/fi17090394

25. Chamkar S.A., Zaydi M., Maleh Y., Gherabi N. Improving Threat Detection in Wazuh Using Machine Learning Techniques. J. Cybersecur. Priv. 2025, 5, 34. https://doi.org/10.3390/jcp5020034

26. Sheeraz M., Durad M.H., Paracha M.A., Mohsin S.M., Kazmi S.N., Maple C. Revolutionizing SIEM Security: An Innovative Correlation Engine Design for Multi-Layered Attack Detection. Sensors 2024, 24, 4901. https://doi.org/10.3390/s24154901

27. Wazuh documentation.- [Електронний ресурс].- Режим доступу: https://documentation.wazuh.com

28. Getting started with Wazuh. Architecture.- [Електронний ресурс].- Режим доступу: https://documentation.wazuh.com/current/getting-started/architecture.html

29. Wazuh Architecture and Components.- [Електронний ресурс].- Режим доступу: https://medium.com/@sharjeelkh1995/wazuh-architecture-and-components-08403653217c

30. Bernardo, Louis. Targeted Attack Detection by Means of Free and Open Source Solutions, 2019.- 161p.

31. VirusTotal.- [Електронний ресурс].- Режим доступу: https://www.virustotal.com/gui/home/upload

32. Balaji N. VirusTotal for Threat Research: A Detailed Guide Released. – 2024.- [Електронний ресурс].- Режим доступу: https://cybersecuritynews.com/virustotal/

33. Sánchez Martínez J.L. Exploring the VirusTotal Dataset | An Analyst's Guide to Effective Threat Research.- 2024.- [Електронний ресурс].- Режим доступу: https://blog.virustotal.com/2024/08/VT-S1-EffectiveResearch.html

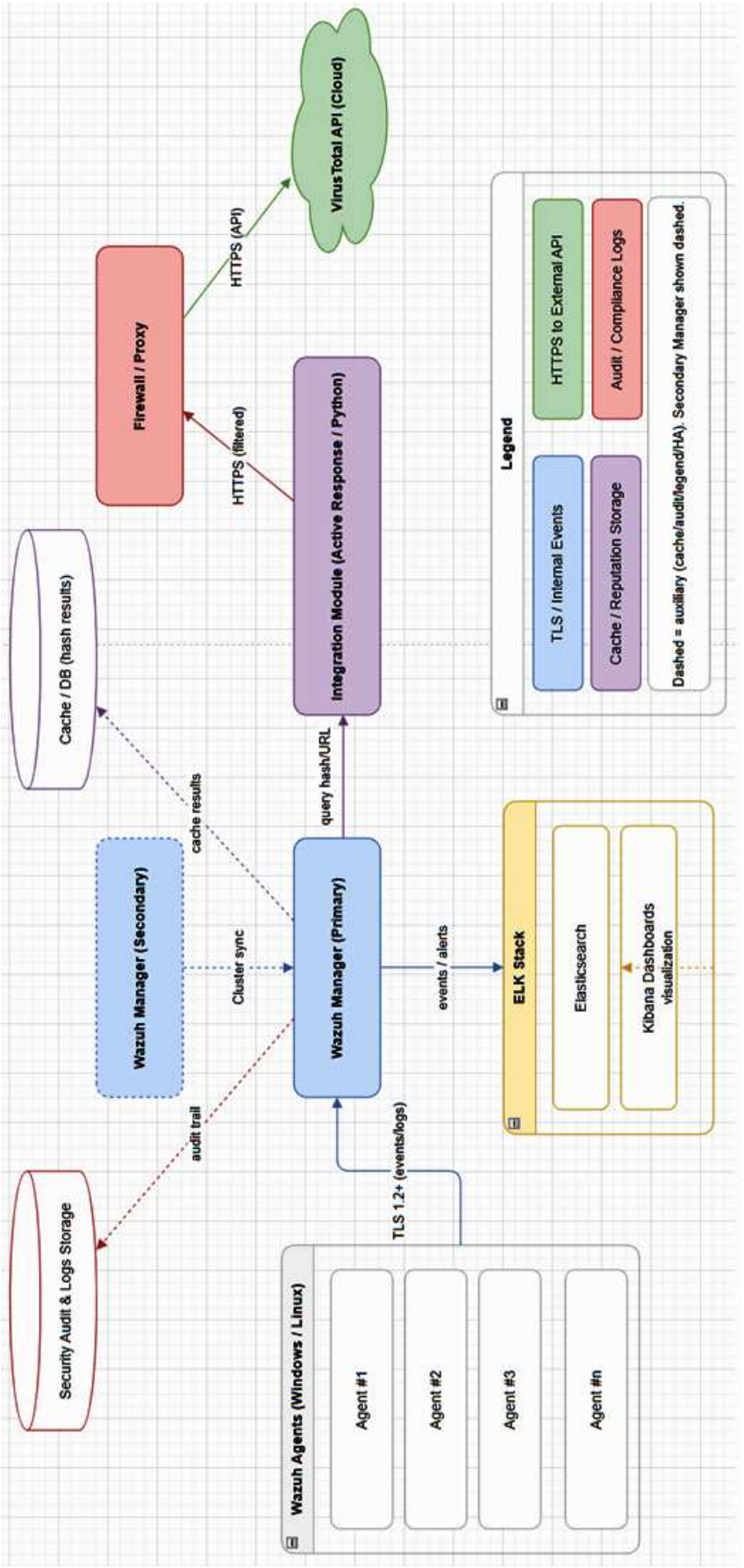
34. Ilascu I. VirusTotal cheat sheet makes it easy to search for specific results. 2022.- [Електронний ресурс].- Режим доступу: https://www.bleepingcomputer.com/news/security/virustotal-cheat-sheet-makes-it-easy-to-search-for-specific-results/

35. Proof of Concept guide. Detecting and removing malware using VirusTotal integration. .- [Електронний ресурс].- Режим доступу: https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html

36. Abrams L. VirusTotal finds hidden malware phishing campaign in SVG files. 2025.- [Електронний ресурс].- Режим доступу: https://www.bleepingcomputer.com/news/security/virustotal-finds-hidden-malware-phishing-campaign-in-svg-files/

ДОДАТОК А

Модель взаємодії компонентів системи виявлення шкідливого програмного забезпечення на основі комбінованого алгоритму



ДОДАТОК Б

Вихідний код для реалізації алгоритму та конфігурації системи виявлення шкідливого програмного забезпечення

Б.1. active-response/bin/vt_lookup.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
vt_lookup.py - Active Response інтеграція Wazuh з VirusTotal v3
Функції:
  - читає вхід Wazuh (JSON з parameters або data.file.hash.sha256)
  - перевіряє локальний кеш (SQLite)
  - звертається до VT API (лише за SHA256), з урахуванням rate-limit
  - повертає JSON з результатом для подальшого збагачення події
Вимоги:
  - Python 3.8+
  - довкілля: VT_API_KEY (обов'язково), опц.: HTTPS_PROXY, HTTP_PROXY
Розміщення:
  /var/ossec/active-response/bin/vt_lookup.py
Права: 0750; власник root:wazuh
"""

import os
import sys
import json
import time
import sqlite3
import urllib.request
import urllib.error
from contextlib import closing

API_KEY = os.getenv("VT_API_KEY", "")
VT_URL  = "https://www.virustotal.com/api/v3/files/{}"

CACHE_DB = "/var/ossec/var/vt_cache.sqlite3"
TTL_SEC  = 7 * 24 * 3600              # 7 днів зберігання кешу
MIN_DELAY_SEC = 15                    # базове обмеження частоти (free key)
LAST_CALL_FILE = "/var/ossec/var/vt_last_call.ts"

def _now() -> int:
    return int(time.time())
def _rate_limit_wait():
```

```
    try:
        if not os.path.exists(LAST_CALL_FILE):
            return
        with open(LAST_CALL_FILE, "r") as f:
            last = int(f.read().strip() or "0")
        delta = _now() - last
        if delta < MIN_DELAY_SEC:
            time.sleep(MIN_DELAY_SEC - delta)
    except Exception:
        pass
def _rate_limit_touch():
    try:
        with open(LAST_CALL_FILE, "w") as f:
            f.write(str(_now()))
    except Exception:
        pass
def init_cache():
    con = sqlite3.connect(CACHE_DB)
    con.execute("""CREATE TABLE IF NOT EXISTS cache(
        sha256 TEXT PRIMARY KEY,
        ts INTEGER NOT NULL,
        malicious INTEGER NOT NULL,
        total INTEGER NOT NULL,
        permalink TEXT NOT NULL
    )""")
    con.execute("""CREATE INDEX IF NOT EXISTS idx_ts ON cache(ts)""")
    con.commit()
    return con
def cache_get(con, sha256):
    cur = con.execute("SELECT ts, malicious, total, permalink FROM cache
WHERE sha256=?", (sha256,))
    row = cur.fetchone()
    if not row:
        return None
    ts, mal, tot, link = row
    if _now() - ts > TTL_SEC:
        return None
    return {"malicious": int(mal), "total": int(tot), "url": link}
def cache_put(con, sha256, vt):
    con.execute("REPLACE INTO cache(sha256, ts, malicious, total, permalink)
VALUES(?,?,?,?,?)",
                (sha256,       _now(),       int(vt.get("malicious",       0)),
int(vt.get("total", 0)), vt.get("url", "")))
    con.commit()
```

```
def vt_query(sha256):
    if not API_KEY:
        raise RuntimeError("VT_API_KEY is not set")
    req = urllib.request.Request(VT_URL.format(sha256))
    req.add_header("x-apikey", API_KEY)
    _rate_limit_wait()
    try:
        with closing(urllib.request.urlopen(req, timeout=12)) as r:
            data = json.loads(r.read().decode("utf-8", "ignore"))
    finally:
        _rate_limit_touch()
    stats = data["data"]["attributes"]["last_analysis_stats"]
    link  = "https://www.virustotal.com/gui/file/" + sha256
    total = int(sum(int(v) for v in stats.values()))
    mal   = int(stats.get("malicious", 0))
    return {"malicious": mal, "total": total if total > 0 else 1, "url":
link}
def read_event():
    """Active Response надсилає JSON через stdin. Шукаємо hash у parameters
або в data.file.hash.sha256"""
    raw = sys.stdin.read().strip()
    if not raw:
        return {}
    try:
        return json.loads(raw)
    except Exception:
        # якщо це не JSON (деякі версії надсилають рядки) - пробуємо знайти
"sha256=<hash>"
        ev = {}
        if "sha256=" in raw:
            h = raw.split("sha256=")[-1].split()[0]
            ev["parameters"] = ["sha256", h]
        return ev
def extract_sha256(event):
    # 1) parameters: ["sha256", "<HASH>"]
    params = event.get("parameters") or []
    if len(params) >= 2 and params[0].lower() == "sha256":
        return params[1]
    # 2) data.file.hash.sha256
    try:
        return event["data"]["file"]["hash"]["sha256"]
    except Exception:
        return None
def main():
```

```
    event = read_event()
    sha256 = extract_sha256(event)
    if not sha256 or len(sha256) != 64:
        print(json.dumps({"vt": {"malicious": 0, "total": 1, "url": ""},
"error": "no_valid_sha256"}))
        return
    con = init_cache()
    cached = cache_get(con, sha256)
    if cached:
        print(json.dumps({"vt": cached, "cached": True}))
        return
    try:
        vt = vt_query(sha256)
        cache_put(con, sha256, vt)
        print(json.dumps({"vt": vt, "cached": False}))
    except (urllib.error.HTTPError, urllib.error.URLError) as e:
        # У разі помилки не блокуємо конвеєр - повертаємо нейтральну оцінку
        print(json.dumps({"vt": {"malicious": 0, "total": 1, "url": ""},
"error": str(e)}))
    except Exception as e:
        print(json.dumps({"vt": {"malicious": 0, "total": 1, "url": ""},
"error": str(e)}))
if __name__ == "__main__":
    main()
```

Б.2. etc/rules/local_rules.xml (фрагменти)

```
<!-- /var/ossec/etc/rules/local_rules.xml -->
<group name="malware,combined,vt">
  <!-- Поведінкові індикатори -->
  <rule id="940101" level="8">
    <if_group>process_creation</if_group>
    <match>\\AppData\\Local\\Temp\\|/tmp/</match>
    <description>Process launched from TEMP directory</description>
    <group>behavior,temp_exec</group>
    <mitre id="T1204.002"/>
  </rule>
  <rule id="940102" level="9">
    <if_group>registry_mod|autoruns|persistence</if_group>
    <description>Autorun persistence created</description>
    <group>behavior,persistence</group>
    <mitre id="T1060"/>
  </rule>
  <!-- Сигнатурний (IOC) - приклад точного збігу хешу -->
```

```
  <rule id="940001" level="12">
    <description>Known malicious hash (IOC match)</description>
    <field name="file.hash.sha256">^[0-9A-Fa-f]{64}$</field>
    <group>malware,ioc</group>
    <mitre id="T1204"/>
  </rule>
  <!-- Комбіноване підсилення: коли одночасно 940101 і 940102 -->
  <rule id="950100" level="12">
    <if_matched_sid>940101,940102</if_matched_sid>
    <description>Combined behavior indicates malware</description>
    <group>combined,behavior</group>
  </rule>
  <!-- Рівні для підсумкового скорингу (заповнюються скриптом/процесором) -->
  <rule id="950101" level="8">
    <description>Suspicious activity (combined score 0.40–0.69)</description>
    <group>combined</group>
  </rule>
  <rule id="950102" level="4">
    <description>Benign trace (combined score &lt; 0.40)</description>
    <group>combined</group>
  </rule>
</group>
```

Б.3. etc/decoders/local_decoder.xml

```
<!-- /var/ossec/etc/decoders/local_decoder.xml -->
<decoders>
  <!-- Декодер для збагачених записів власного формату (необов'язково) -->
  <decoder name="combined-detector">
    <prematch>combined_detector</prematch>
    <regex              offset="after_prematch">file=(\S+)\s+sha256=([0-9a-fA-
F]{64})\s+action=(\S+)</regex>
    <order>file.path,file.hash.sha256,event.action</order>
  </decoder>
</decoders>
```

Б.4. Фрагмент ossec.conf (команда та Active Response)

```
<!-- Вставити у /var/ossec/etc/ossec.conf на менеджері -->
<ossec_config>
  <!-- Реєстрація команди -->
  <command>
    <name>vt_lookup</name>
    <executable>vt_lookup.py</executable>
```

```
    <timeout_allowed>no</timeout_allowed>
  </command>
  <!-- Виклики Active Response при відповідних правилах -->
  <active-response>
    <command>vt_lookup</command>
    <location>manager</location>
    <!-- Тригеримо при поведінкових/сигнатурних і комбінованих правилах -->
    <rules_id>940001,940101,940102,950100</rules_id>
  </active-response>
</ossec_config>
```

Б.5. Збагачення полів у Logstash / Filebeat, якщо ви використовуєтьтя окремий етап для розрахунку підсумкового балу та присвоєння рішення. Приклад Processors (Ingest Pipeline Elasticsearch):

```
{
  "processors": [
    {
      "script": {
        "lang": "painless",
        "source": """
          double w_sig = 0.45, w_beh = 0.35, w_rep = 0.20;
          double S_sig = (ctx?.rule?.level != null && ctx.rule.level >= 10) ?
1.0 :
                         (ctx?.rule?.level != null && ctx.rule.level >= 7) ?
0.5 : 0.0;
          double S_beh = (ctx?.labels?.contains('behavior') ? 0.5 : 0.0) +
                         (ctx?.labels?.contains('persistence') ? 0.4 : 0.0);
          if (S_beh > 1.0) S_beh = 1.0;
          double mal = (ctx?.vt?.malicious != null) ? ctx.vt.malicious : 0;
          double tot = (ctx?.vt?.total != null && ctx.vt.total > 0) ?
ctx.vt.total : 1;
          double S_rep = mal / tot;
          double score = w_sig*S_sig + w_beh*S_beh + w_rep*S_rep;
          ctx.combined = ['score': score, 'decision': (score >= 0.70 ?
'DETECT' : (score >= 0.40 ? 'SUSPICIOUS' : 'BENIGN'))];
        """
      }
    }
  ]
}
```

Б.6. systemd довкілля для API-ключа (безпека)

/etc/systemd/system/wazuh-manager.service.d/env.conf

```
[Service]
Environment="VT_API_KEY=***ТУТ_ТВОЙ_КЛЮЧ_VT***"
Environment="HTTPS_PROXY="
```

Після додавання:

```
sudo systemctl daemon-reload
sudo systemctl restart wazuh-manager
```

Б.7. requirements.txt (якщо скрипт виконуватиметься в окремому venv)

```
# stdlib-only; немає сторонніх залежностей
```

Б.8. Тестові приклади (локальна перевірка скрипта)

```
# Імітація події Active Response (stdin JSON)
echo                          '{"parameters":                          ["sha256",
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"]}' \
 | VT_API_KEY=YOUR_KEY ./vt_lookup.py
```

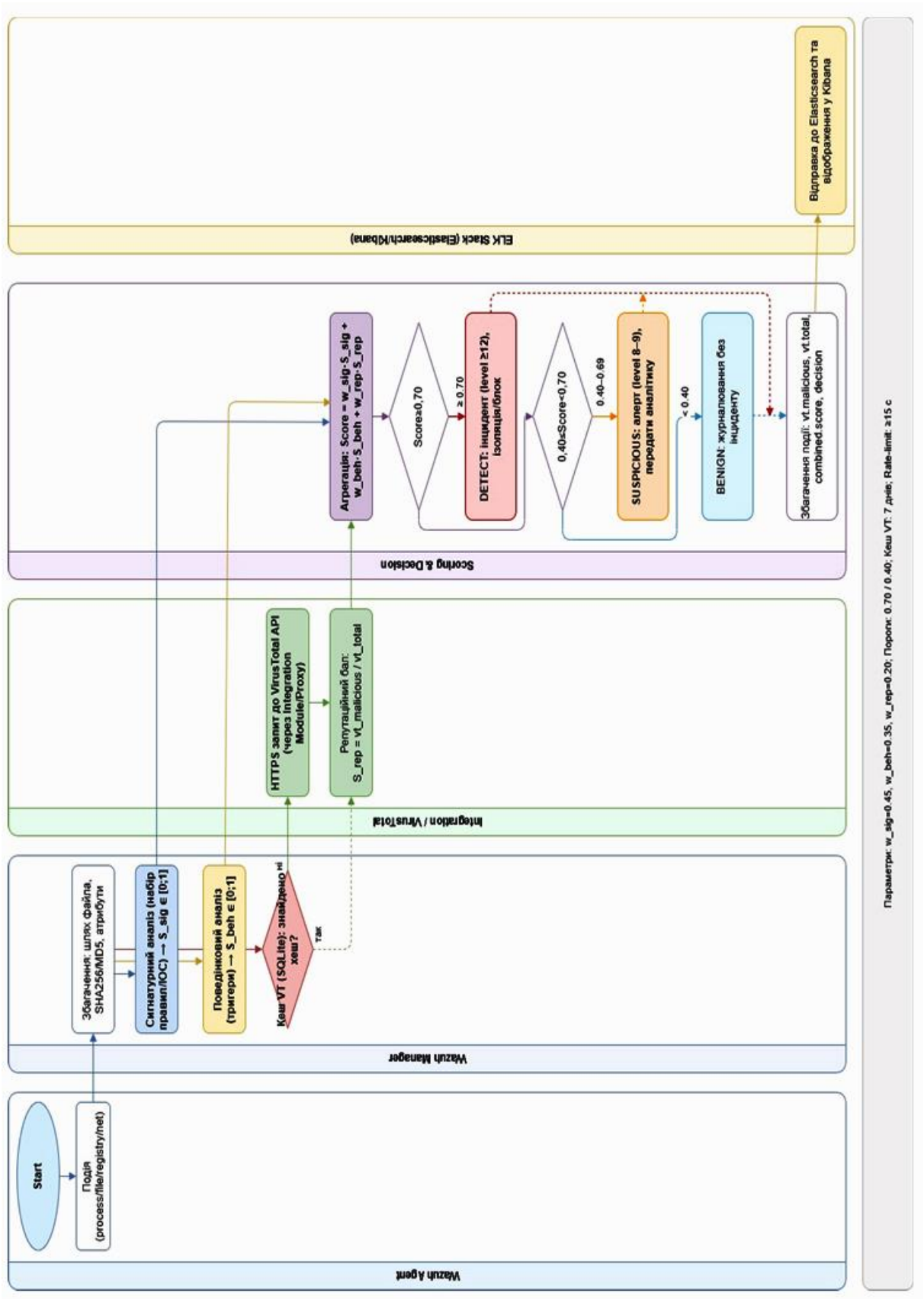
Очікувано, що повернеться JSON:

```
{"vt":{"malicious":0,"total":1,"url":""},"error":"HTTP Error ..."}
```

або валідна відповідь із VT, якщо хеш існує у базі.

ДОДАТОК В

Схема роботи системи на основі алгоритму виявлення шкідливого програмного забезпечення



ДОДАТОК Г

Копії публікацій