Харківський національний університет імені В.Н. Каразіна

Факультет комп'ютерних наук

Безпека інформаційних систем і технологій

«Допущено до захисту»

Зав.кафедрою БІСТ

Сватовський І.І._____

« » червня 2023р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

спеціальність: 125 Кібербезпека

на тему: «Дослідження методів аналізу та можливостей існуючого інструментарію для моніторингу зловмисного програмного забезпечення»

оцінка « »

Голова ЕК

Лемешко О.В._____

Керівник проф. д.т.н. доц. Єсін В.І.
(прізвище та ініціали/підпис)

Рецензент к.т.н. Бобух В.А.
(прізвище та ініціали/підпис)

Виконавець студентка групи КБ-41

Зуган Н.С.
(прізвище та ініціали/підпис)

Харків – 2023

РЕФЕРАТ

Пояснювальна записка містить 66 сторінок, 31 рисунок, 3 таблиці, 1 додаток, 16 джерел.

Метою дипломної роботи є дослідження актуальних методів аналізу зловмисного програмного забезпечення, а також можливостей існуючого інструментарію для моніторингу цього програмного забезпечення, з наступним порівнянням ефективності застосування різних засобів для моніторингу зловмисного програмного забезпечення за результатами їх роботи.

Об'єктом дослідження є процес аналізу моніторингу зловмисного програмного забезпечення.

Предметом дослідження є методи статичного, динамічного, гібридного аналізу зловмисного програмного забезпечення.

Методи дослідження: основні розділи системного аналізу, теорія алгоритмів, експериментальне тестування.

В ході роботи були описані найбільш відомі та активні типи зловмисного програмного забезпечення, також були розкриті найпоширеніші шляхи його розповсюдження. Далі були розглянуті методи виявлення шкідливого програмного забезпечення, що проводяться на базі сигнатур або на базі евристики. Також було визначено, що основними методами аналізу є статичний, динамічний, гібридний та аналіз пам'яті, які мають достатньо інструментів, що допомагають здійснювати аналіз шкідливого програмного забезпечення.

Статичний аналіз може проводитися шляхом наступних методів: багаторазове сканування антивірусом, використання file fingerprinting, пошук по рядках, виявлення пакувальників, перевірка інформації у PE-заголовку та аналіз дизасемблерного коду.

Динамічний аналіз включає такі методи: моніторинг модифікацій файлів, моніторинг змін реєстру, моніторинг мережевої активності та системних викликів, а також відладку.

Дослідження інструментів для аналізу зловмисного програмного забезпечення включало аналіз їх можливостей та ефективності для виявлення шкідливих програм.

В результаті проведених досліджень були визначені переваги та недоліки кожного методу аналізу, була проведена оцінка ефективності інструментів для моніторингу зловмисного програмного забезпечення, шляхом порівняння їх можливостей та функцій, та надані рекомендації щодо їх використання. Також в роботі висунуті пропозиції щодо розвитку інструментарію для моніторингу зловмисного програмного забезпечення. Ще були надані рекомендації по використанню різних методів статичного та динамічного методів аналізу.

Ключові слова: ЗЛОВМИСНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОМП’ЮТЕРНИЙ ВІРУС, МЕТОДИ АНАЛІЗУ, СТАТИЧНИЙ АНАЛІЗ, ДИНАМІЧНИЙ АНАЛІЗ, ГІБРИДНИЙ АНАЛІЗ, ІНСТРУМЕНТАРІЙ МОНІТОРИНГУ.

ABSTRACT

The explanatory note contains 66 pages, 31 figures, 3 tables, 1 annex, and 16 sources.

The aim of the thesis thesis is to research current methods of analyzing malicious software, as well as the possibilities of existing tools for monitoring this software, followed by a comparison of the effectiveness of the use of various tools for monitoring malicious software based on the results of their work.

The subject matter is the process of analyzing the monitoring of malicious software.

The scope of the study is the methods of static, dynamic, hybrid analysis of malicious software.

Research methods: main sections of system analysis, theory of algorithms, experimental testing.

In the course of the work, the most known and active types of malicious software were described, and the most common ways of its distribution were also revealed. Next, methods of detecting malicious software based on signatures or heuristics were considered. It was also determined that the main analysis methods are static, dynamic, hybrid and memory analysis, which have enough tools to help perform malware analysis.

Static analysis can be carried out by the following methods: repeated scanning by antivirus, usage of file fingerprinting, search by lines, detection of packers, verification of information in the PE-header and analysis of disassembling.

Dynamic analysis includes the following methods: monitoring file modifications, monitoring registry changes, monitoring network activity and system calls, and debugging.

A study of malware analysis tools included an analysis of their capabilities and effectiveness in detecting malware.

As a result of the conducted research, the advantages and disadvantages of each analysis method were determined, the effectiveness of tools for monitoring malicious software was evaluated by comparing their capabilities and functions, and recommendations for their use were provided. Also, the paper makes suggestions for the development of tools for monitoring malicious software. Recommendations on the use of various static and dynamic analysis methods were also provided.

Keywords: MALWARE, COMPUTER VIRUS, ANALYSIS METHODS, STATIC ANALYSIS, DYNAMIC ANALYSIS, HYBRID ANALYSIS, MONITORING TOOLS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС	–	операційна система
ПЗ	–	програмне забезпечення
ПК	–	персональний комп'ютер
API	–	Application Programming Interface
CRC32	–	Cyclic redundancy check 32.
DDL	–	Data Definition Language
DLL	–	Dynamic Link Library
DDos	–	Distributed Denial of Service
ELF	–	Executable and Linkable Format
FN	–	False negative rate
FP	–	False positive rate
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
IOC	–	Indicators of compromise
IP	–	Internet Protocol
MD5	–	Message Digest 5
MS-DOS	–	MicroSoft Disk Operating System
P3P	–	Platform for Privacy Preferences
PE	–	Portable Executable

PIN	–	Personal Identification Number.
RDP	–	Remote Desktop Protocol
SHA	–	Secure Hash Algorithm
URL	–	Uniform Resource Locator
USB	–	Universal Serial Bus
XML	–	eXtensible Markup Language

ВСТУП

Зловмисне програмне забезпечення (ПЗ) може вразити програмні засоби будь-якої особи чи компанії. Останніми роками в усьому світі сталися масштабні атаки за допомогою зловмисного програмного забезпечення, через що мільйони електронних листів, паролів та іншої фінансової інформації були викрадені та скомпроментовані [1]. Оскільки залежність людства від технологій зростає, розуміння цих ризиків і способів захисту також має зростати.

Аналіз зловмисного ПЗ дозволяє виявити і зрозуміти, як саме воно працює, які дії воно виконує та які наслідки можуть бути для комп'ютерної системи та її користувачів. Це дає можливість розробити методи захисту та протидії шкідливому програмному забезпеченню, а також допомогти виявити зломисників, які створюють та поширюють це програмне забезпечення.

Метою даної роботи є дослідження актуальних методів аналізу зловмисного програмного забезпечення, а також можливостей існуючого інструментарію для моніторингу цього ПЗ, з наступним порівнянням ефективності застосування різних засобів для моніторингу зловмисного програмного забезпечення за результатами їх роботи.

Дослідження методів аналізу та інструментарію для моніторингу зловмисного програмного забезпечення може бути корисним для розробки нових підходів до кібербезпеки, допомогти вдосконалити існуючі методи та інструментарій для аналізу зловмисного програмного забезпечення.

1 ТЕРМІНОЛОГІЧНИЙ БАЗИС ПРЕДМЕТНОЇ ОБЛАСТІ , ЩО ДОСЛІДЖУЄТЬСЯ

Шкідливе програмне забезпечення — це код, який виконує шкідливі дії; він може мати форму виконуваного файлу, сценарію, коду або будь-якого іншого програмного забезпечення. Зловмисники використовують зловмисне ПЗ, щоб викрасти конфіденційну інформацію, стежити за зараженою системою або отримати контроль над системою. Зазвичай воно потрапляє систему без згоди користувача, та може бути отриманий через різні канали зв'язку, такі як електронна пошта, Інтернет або Universal Serial Bus (USB) - накопичувачі [2].

Зловмисне програмне забезпечення може заражати мережі та пристрої та певним чином завдати шкоди цим пристроям, мережам та/або їхнім користувачам.

Незалежно від методу дії, усі типи зловмисного програмного забезпечення призначені для використання пристроїв за рахунок користувача та на користь хакера – особи, яка розробила та/або розгорнула зловмисне програмне забезпечення.

1.1 Типи зловмисного програмного забезпечення

Хоча існує багато різновидів (типів) зловмисного програмного забезпечення нижче описані найбільш розповсюджені [3]:

- Програма - вимагач (Ransomware) – вимикає доступ жертви до даних до моменту сплати викупу, але немає жодної гарантії, що оплата призведе до отримання необхідного ключа розшифровки або що наданий ключ розшифровки функціонуватиме належним чином.
- Безфайлове зловмисне програмне забезпечення (Fileless Malware) спочатку вносить зміни до файлів, що є вбудованими в операційну систему (ОС). Оскільки ОС розпізнає відредаговані файли

як законні, безфайлову атаку не вловлює антивірусне програмне забезпечення.

- Шпигунське програмне забезпечення (Spyware) збирає інформацію про дії користувачів без їх відома чи згоди. Це може включати паролі, Personal Identification Number (PIN)-коди, платіжну інформацію та неструктуровані повідомлення.
- Троян (Trojans) маскується під бажаний код або програмне забезпечення. Після завантаження троян може захопити систему жертви для зловживань. Трояни можуть приховуватись в іграх, застосунках або у патчах для програмного забезпечення, або бути вбудовані в застосунки, що додаються до фішингових електронних листів.
- Черви (Worms) націлені на вразливі місця в операційних системах, щоб встановлюватися в мережі. Вони можуть отримати доступ через бекдори, вбудовані в програмне забезпечення, через вразливості програмного забезпечення або через флеш-накопичувачі. Після встановлення черви можуть використовуватися зловмисниками для запуску Distributed Denial of Service (DDoS)-атак, викрадення конфіденційних даних або проведення атак із шифруванням вимог до викупу.
- Вірус – це фрагмент коду, який вставляється в програму та виконується під час її запуску. Потрапивши в мережу, вірус може бути використаний для викрадення конфіденційних даних, запуску DDoS-атак або атак програм-вимагачів.
- Руткіт (Rootkits) – це програмне забезпечення, яке надає зловмисникам віддалене керування комп’ютером жертви з повними правами адміністратора. Руткіти можна впроваджувати в програми, ядра, гіпервізори або мікропрограми. Їх також можна використовувати для приховування інших шкідливих програм, наприклад кейлоггерів.

- Кейлогер (Keyloggers) – це тип шпигунського програмного забезпечення, яке відстежує дії користувачів.
- Вайпер (Wiper Malware) – це тип шкідливого програмного забезпечення з метою стерти дані користувача та гарантувати, що їх неможливо відновити. Вайпери використовуються для атак комп'ютерних мереж у державних або приватних компаніях у різних секторах. Зловмисники також використовують вайпери, щоб приховати сліди, залишені після вторгнення.

1.2 Методи поширення шкідливого програмного забезпечення

Зловмисне програмне забезпечення зазвичай поширюється через шкідливі веб-сайти, електронні листи та програмне забезпечення. Зловмисне програмне забезпечення також може бути приховано в інших файлах, таких як файли зображень чи документів.

Користувачі можуть ненавмисно встановити зловмисне програмне забезпечення, натиснувши посилання у фішинговому листі або завантаживши та встановивши програмне забезпечення з веб-сайту, який не має авторитету. Зловмисне програмне забезпечення також може бути встановлено на комп'ютер, коли користувач підключає заражений USB-накопичувач або коли користувач відвідує веб-сайт, заражений шкідливим програмним забезпеченням [4].

Найпоширеніші методи розповсюдження зловмисного програмного забезпечення:

- Сайти, спеціально створені для доставки шкідливих програм.
- Законні сайти, які були зламані або захоплені.
- Зловмисна реклама.
- Мережі обміну файлами/ Platform for Privacy Preferences (P3P) і ненадійні сайти.
- Фішингові листи.
- Спам у соціальних мережах .

- Протокол віддаленого робочого столу.

Сайти, спеціально створені для поширення шкідливого ПЗ. Це один із найпоширеніших способів зараження зловмисним програмним забезпеченням, його все частіше використовують через простоту налаштування доменного ім'я. Часто ці сайти не мають жодної іншої мети, окрім як заманити користувачів і зрештою надати їм зловмисне програмне забезпечення. Те, як ці сайти залучають користувачів, залежить від зловмисника [5].

Легальні сайти, які були зламані або захоплені. Інколи зловмисне програмне забезпечення потрапляє до користувачів через використання вразливостей звичайних сайтів. Якщо у фреймворку, який використовує сайт або сервер, є експлойти, це потенційно може бути шляхом до користувачів. Недоліки подібних веб-платформ можуть включати можливість запуску програм або виконання довільного коду на машинах користувача [5].

Зловмисна реклама. Способи потрапляння зловмисної реклами на активну веб-сторінку можуть бути різними. Іноді вразливість у рекламній мережі може дозволити зловмиснику скомпрометувати оголошення, через яке користувачам буде надане зловмисне програмне забезпечення. Подібні випадки траплялися особливо часто, коли багато веб-сайтів покладалися на Adobe Flash для відтворення анімованої реклами [5].

Мережі обміну файлами/P3P і ненадійні сайти. Існує певна категорія сайтів, що завжди були відомі своєю тенденцією до поширення шкідливого програмного забезпечення. У цій категорії сайтів, як правило, розміщуються незаконні торрент-трекери та веб-сайти обміну файлами. Завантажуючи файли від інших користувачів анонімно, немає гарантії, що файл буде безпечним - він може містити шкідливі програми, які можуть заражати комп’ютер. Крім того, веб-сайти обміну файлами зазвичай містять багато шкідливих оголошень, які збільшують ризик зараження [5].

Фішингові листи. Розсилка зловмисного програмного забезпечення за допомогою фішингових електронних листів є особливо ефективним методом,

оскільки він часто використовує споосби, які створюють враження достовірності, переконуючи користувачів натискати посилання та завантажувати вкладення. Коли користувач натискає посилання або завантажує вкладений файл, який міститься у фішинговому листі, на його пристрої встановлюється шкідливе програмне забезпечення. Це зловмисне програмне забезпечення може приймати різні форми, як-от програми-вимагачі, шпигунські програми або троянські програми [6].

Спам у соціальних мережах може використовувати атаки клонування фішингу. Кіберзлочинці можуть використовувати платформи соціальних мереж, щоб надсилати спамові повідомлення або публікувати посилання на шкідливі веб-сайти чи спливаючі вікна, які пропонують користувачам завантажити та встановитишкідливе програмне забезпечення замасковане під звичайну програму [6].

Протокол віддаленого робочого стола (Remote Desktop Protocol - RDP) — це популярний протокол підключення, який дозволяє користувачам віддалено підключатися до іншого комп’ютера через мережу, але його також можуть використовувати кіберзлочинці, застовуючи методи автоматичного сканування для визначення місцезнаходження комп’ютерів із відкритими з’єднаннями RDP. Коли вони знаходять такі системи, вони вгадують облікові дані для входу, часто за допомогою тактики грубої сили, і, отримавши доступ, встановлюють зловмисне програмне забезпечення для використання скомпрометованої системи. У деяких випадках хакери можуть навіть придбати облікові дані для входу в Dark Web, щоб отримати несанкціонований доступ до конфіденційної інформації [6].

2 МЕТОДИ ВИЯВЛЕННЯ ТА АНАЛІЗУ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методи виявлення зловмисного програмного забезпечення поділяються на кілька категорій за різними ознаками. Основні методи виявлення зловмисного програмного забезпечення: сигнатурний і евристичний. На рисунку 2.1 показано основні методи виявлення шкідливих програм.

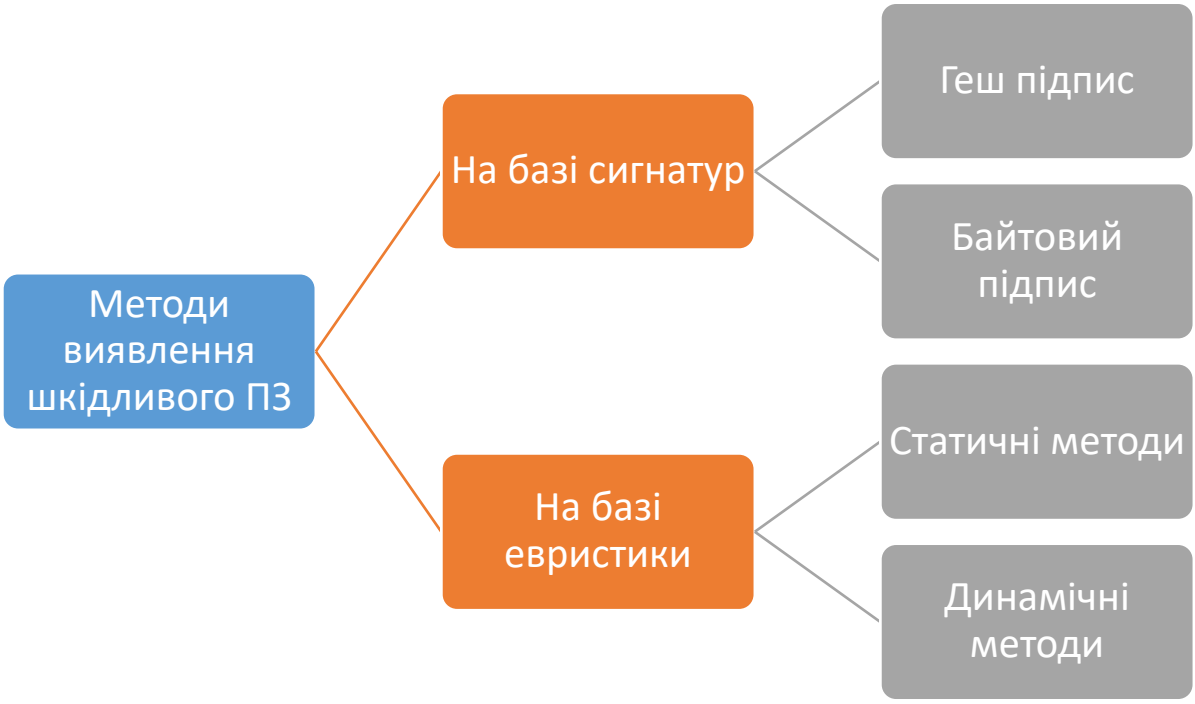

Рисунок 2.1 – Основні методи виявлення шкідливих програм

Виявлення на основі сигнатур.

Більшість доступних антивірусних програм використовують підхід на основі сигнатур. Цей підхід витягує унікальний підпис із захопленого файлу шкідливого програмного забезпечення та використовує цей підпис для виявлення подібного шкідливого програмного забезпечення. Сигнатура — це послідовність байтів або хеш файлу, який можна використовувати для ідентифікації конкретного шкідливого програмного забезпечення. Таким чином, цей метод має невелику частоту помилкових спрацьовувань. Однак для

зловмисників не важко змінити підпис шкідливого програмного забезпечення, щоб уникнути виявлення антивірусним програмним забезпеченням [7].

Сигнатура дуже ефективна та швидка у виявленні відомих зловмисних програм, але вона нездатна захопити нові випущені шкідливі програми. Підхід на основі сигнатур залежить від реалізації статичного аналізу для вилучення виняткових послідовностей байтів, відомих як позначки (marks).

Автори зловмисного програмного забезпечення створили ще одну перешкоду для підходу на основі сигнатур, використовуючи методи обфускації. Ці методи включають вставку мертвого коду (dead code insertion), перепризначення регістру, заміну інструкцій і маніпулювання кодом [7].

Виявлення на основі евристики.

Евристичне виявлення також відоме як виявлення на основі аномалії або поведінки. Під час цього виявлення дії, які виконує зловмисне програмне забезпечення під час виконання, аналізуються на етапі навчання. Після цього файл позначається як шкідливий або законний файл під час фази тестування (моніторингу) на основі шаблону, витягнутого під час навчального тесту. На відміну від сигнатурного, поведінковий підхід здатний виявляти як невідоме шкідливе програмне забезпечення, так і шкідливе програмне забезпечення, яке використовує методи обфускації [7].

Однак основними недоліками поведінкових методів є значна частота помилкових позитивних результатів (False positive rate - FP) і надмірний час моніторингу. Крім того, зменшення тисяч витягнутих функцій, оцінка подібності між ними та моніторинг активності зловмисного програмного забезпечення безпосередньо впливають на здатність виявлення атак зловмисного програмного забезпечення нульового дня. Щоб зрозуміти поведінку запущених файлів, евристичний метод зазвичай залежить від методів інтелектуального аналізу даних [7].

Аналіз шкідливого програмного забезпечення – це процес розуміння поведінки та призначення підозрілого файлу чи Uniform Resource Locator

(URL)-адреси. Результати аналізу допомагають виявити та зменшити потенційну загрозу.

Основна перевага аналізу зловмисного програмного забезпечення полягає в тому, що він допомагає спеціалістам із реагування на інциденти та аналітикам безпеки:

- прагматично сортувати інциденти за рівнем тяжкості;
- розкривати приховані індикатори компрометації (Indicators of compromise - IOC), які слід заблокувати;
- підвищувати ефективність попереджень і сповіщень IOC;
- збагачувати контекст під час пошуку загроз.

Методи, які використовуються для аналізу зловмисного програмного забезпечення, в основному поділяються на три типи: статичний, динамічний і гібридний аналіз (рисунок 2.2).

Рисунок 2.2 - Методи аналізу зловмисного програмного забезпечення

2.1 Статичний аналіз

Статичний аналіз зловмисного програмного забезпечення є важливим інструментом кібербезпеки, який дає змогу дослідникам аналізувати поведінку, структуру та цілі шкідливого програмного забезпечення. Цей тип аналізу передбачає перевірку коду та алгоритмів, які використовуються у зловмисному ПЗ, без його виконання. Користуючись цим методом, експерти можуть отримати цінну інформацію про функціональність шкідливого ПЗ і покращити своє розуміння потенційних загроз.

Методи статичного аналізу можна застосовувати до різних представлень програми. Інструменти статичного аналізу також можна використовувати для двійкового представлення програми. Під час компіляції вихідного коду програми у двійковий виконуваний файл деяка інформація втрачається. Ця втрата інформації ще більше ускладнює завдання аналізу коду. Процес перевірки заданого двійкового файлу без його виконання здебільшого виконується вручну. Наприклад, якщо вихідний код доступний, можна отримати кілька цікавих відомостей, таких як структури даних і використовувані функції. Ця інформація втрачається після компіляції вихідного коду у двійковий виконуваний файл, що перешкоджає подальшому аналізу [8].

Для статичного аналізу шкідливих програм використовуються різні методи. Деякі з них описані нижче:

- багаторазове сканування антивірусом;
- file fingerprinting;
- пошук по рядкам;
- виявлення пакувальників;
- перевірка інформації у Portable Executable (PE)-заголовку;
- аналіз дизасемблерного коду.

Спочатку дамо коротку узагальнену характеристику цих методів, після чого детально зупинимося на кожному з них.

Багаторазове сканування антивірусом: якщо досліджуваний двійковий файл є добре відомим зловмисним програмним забезпеченням, його, швидше за все, буде виявлено одним або кількома антивірусними сканерами. Використання одного або кількох антивірусних сканерів займає багато часу, але іноді стає необхідністю.

File fingerprinting: окрім дослідження очевидних зовнішніх особливостей двійкового файлу, цей процес включає операції на рівні файлу, такі як обчислення криптографічного хешу (наприклад, Message Digest 5

(MD5)) двійкового файлу, щоб відрізнити його від інших і переконатися, що він не був змінений.

Пошук за рядками: це метод, який використовується для аналізу та виявлення шкідливого програмного забезпечення шляхом пошуку певних послідовностей символів, відомих як рядки, у файлах і каталогах комп'ютерної системи. Аналізуючи ці рядки, можна визначити шаблони поведінки та характеристики, які зазвичай асоціюються зі шкідливим програмним забезпеченням.

Виявлення пакувальників: на сьогоднішній день зловмисне програмне забезпечення здебільшого поширюється в обфускованій формі, наприклад, у зашифрованому або стиснутому вигляді. Це досягається за допомогою пакера, тоді як для модифікації можна використовувати довільні алгоритми. Після пакування програма виглядає значно інакше з точки зору статичного аналізу, тому її логіку та інші метадані важко відновити. Хоча існують певні розпаковувачі, такі як PEiD2, відповідно не існує загального розпаковувача, що робить це серйозною проблемою статичного аналізу шкідливих програм.

Перевірка інформації у PE-заголовку: за допомогою метаданих певного формату файлу можна зібрати додаткову корисну інформацію. Це включає магічне число в системах UNIX для визначення типу файлу. Наприклад, із двійкового файлу Windows, який зазвичай має формат PE (портативний виконуваний файл), можна отримати багато інформації, такої як час компіляції, імпортовані та експортовані функції, а також рядки, меню та іконки.

Аналіз дизасемблерного коду: основною частиною статичного аналізу є, як правило, розбирання заданого двійкового файлу. Це виконується за допомогою інструментів, які здатні перетворити машинний код на мову асемблера. На основі реконструйованого коду складання аналітик може перевірити логіку програми і, таким чином, перевірити її наміри.

Крім того, статичний аналіз, як правило, безпечніший за динамічний, оскільки вихідний код фактично не виконується. Однак це може зайняти дуже багато часу, а отже, вимагає досвіду.

2.1.1 Багаторазове сканування антивірусом

Під скануванням антивірусом розуміється процес послідовного аналізу потенційно зловмисного ПЗ декількома антивірусами Проведення антивірусного сканування є першим кроком у вирішенні проблем з ПЗ. Перевагами такого способу є:

- Покращена точність виявлення: багаторазове сканування файлів збільшує ймовірність виявлення шкідливого ПЗ. Антивірусне програмне забезпечення може використовувати різні методи сканування або розширені алгоритми для виявлення зловмисного програмного забезпечення, яке раніше могло уникати виявлення.
- Підвищена впевненість у результатах: багаторазове сканування одного файлу різними антивірусними програмами може забезпечити вищий ступінь впевненості в результатах. Якщо кілька антивірусних програм виявляють один і той самий файл як зловмисне ПЗ, швидше за все, це справжня загроза, а не помилковий результат.
- Швидка реакція на нові загрози: багаторазове сканування гарантує, що система оновлюється з найновішими визначеннями загроз і сигнатурами вірусів. Це дозволяє швидко реагувати на нові загрози зловмисного ПЗ та знижує їхній ризик до того, як буде завдано значної шкоди.

Для сканування потенційно шкідливого програмного забезпечення можна користуватися як застосунками, що встановлюються на персональний комп’ютер, так і онлайн ресурсами. Серед антивірусного програмного забезпечення на травень 2023 року виділяють наступні: Avira, Avast, Microsoft Defender AntivirusBitdefender, AVG [9].

Для тестування можливостей антивірусів для персонального комп’ютера (ПК) виявляти зловмисне ПЗ використовувався файл з вмістом ідентичним до EICAR-Test-File.

EICAR-Test-File це комп’ютерний файл, розроблений Європейським інститутом досліджень комп’ютерних антивірусів (European Institute for Computer Antivirus Research EICAR) і Організацією дослідження комп’ютерних антивірусів (Computer Antivirus Research Organization CARO) для тестування реакції антивірусу [10]. Замість використання справжнього шкідливого програмного забезпечення, яке може завдати реальної шкоди, цей тестовий файл дозволяє людям тестувати антивірусне ПЗ без використання справжнього комп’ютерного вірусу . Його безпечно передати, оскільки він не є вірусом і не містить жодних фрагментів вірусного коду. Більшість продуктів реагують на нього так, ніби це вірус (хоча вони зазвичай повідомляють про це під очевидною назвою, наприклад «EICAR-AV-Test»). Файл є законною програмою DOS і дає результати під час запуску (він друкує повідомлення «EICAR-STANDARD-ANTIVIRUS-TEST-FILE!»).

Він також короткий і простий – насправді він повністю складається з друкованих символів ASCII, тому його можна легко створити за допомогою звичайного текстового редактора. Будь-який антивірусний продукт, який підтримує тестовий файл EICAR, повинен виявляти його в будь-якому файлі за умови, що файл починається з 68 символів і має рівно 68 байтів (Додаток А) [11].

В результаті роботи програм AVG та Avast було виявлено зловмисне ПЗ EICAR-Test-File. Обидві програми успішно справляються з виявленням зловмисного ПЗ, але одразу видаляють заражений файл, що є незручним за необхідності подальшого аналізу зловмисного файлу. Загалом, саме для виявлення зловмисного ПЗ підійде будь-який з указаних вище антивірусів, але кінцевий вибір програми залежить від потреб користувача.

Також розповсюджені онлайн сканери вірусів:

- SafetyDetectives Known Vulnerabilities Scanner;
- VirusTotal ;
- Intezer Analyze;
- ESET Online Scanner.

Онлайн сканери можуть бути зручніші, оскільки вони не потребують завантаження.

Інструмент для пошуку вразливостей SafetyDetectives — це безкоштовний онлайн-сканер, який швидко перевіряє ПК (або інші пристрої на якому він запущений) на відомі вразливості (відомі з бази даних CVE). Сканер запускається автоматично при відвідуванні веб-сайту, при чому будь-які параметри налаштування чи детальнішого сканування відсутні. Сама перевірка ПК відбувається доволі швидко (у межах 5 секунд). Після завершення сканування, у разі виявлення вразливостей, надаються пояснення по вирішенню виявлених проблем. Хоч цей інструмент визначають як ефективний [12], він не зміг визначити зловмисне ПЗ, котре було виявлене стандартними антивірусами для ПК.

VirusTotal є безкоштовним онлайн-інструментом антивірусного сканування, що також надає додаткову інформацію як, наприклад, метадані файлів і аналіз поведінки. Це робить його корисним інструментом для швидкого визначення того, чи є файл шкідливим чи ні.

Intezer Analyze — це більш просунута платформа аналізу зловмисного програмного забезпечення, яка використовує підхід під назвою «генетичний аналіз шкідливого програмного забезпечення». Цей підхід передбачає аналіз коду файлу та порівняння його з великою базою даних відомого зловмисного та безпечного коду, щоб визначити, чи є код шкідливим чи ні. Цей підхід може виявити навіть раніше невідоме шкідливе програмне забезпечення, що робить його потужним інструментом для виявлення складних загроз.

На рисунках 2.3 та 2.5 представлені результати сканування файлу calculator.exe (програма що виконує функції простого калькулятора) онлайн антивірусами VirusTotal та Intezer Analyze відповідно.

З рисунків 2.3. і 2.4 видно, що 3 антивіруси за допомогою яких VirusTotal перевіряв даний файл визначили його як зловмисний. Причина такого результату може полягати в особливостях коду програми.

Intezer Analyze позначає файл calculator.exe як підозрілий. У відсотковому співвідношенні показується подібність файлу з сім'ями зловмисного ПЗ, до яких, ймовірно, належить аналізований файл. Також надається інформація про метадані файлу та інш.

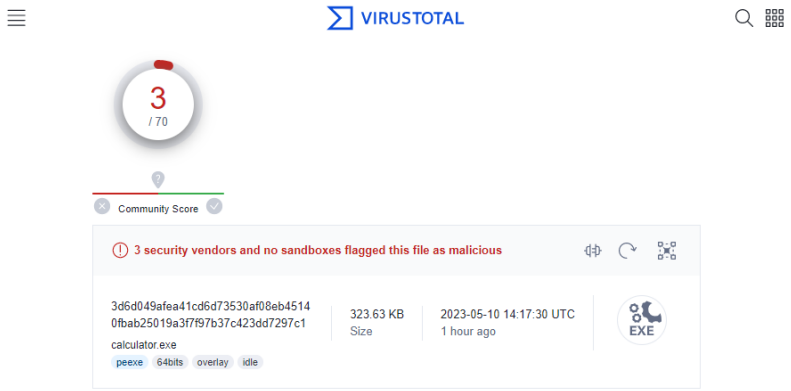

Рисунок 2.3 – Результати сканування VirusTotal файлу calculator.exe

Google	Detected	Ikarus	Trojan.Bazar
MaxSecure	Trojan.Malware.300983.susgen	Acronis (Static ML)	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
BitDefenderTheta	Undetected	Bkav Pro	Undetected
ClamAV	Undetected	CMC	Undetected
CrowdStrike Falcon	Undetected	Cybereason	Undetected
Cylance	Undetected	Cynet	Undetected
Cyren	Undetected	DeepInstinct	Undetected
DrWeb	Undetected	Elastic	Undetected
Emsisoft	Undetected	eScan	Undetected
ESET-NOD32	Undetected	F-Secure	Undetected
Fortinet	Undetected	GData	Undetected
Gridinsoft (no cloud)	Undetected	Jiangmin	Undetected

Рисунок 2.4 – Антивірусні сканери, що використовує VirusTotal

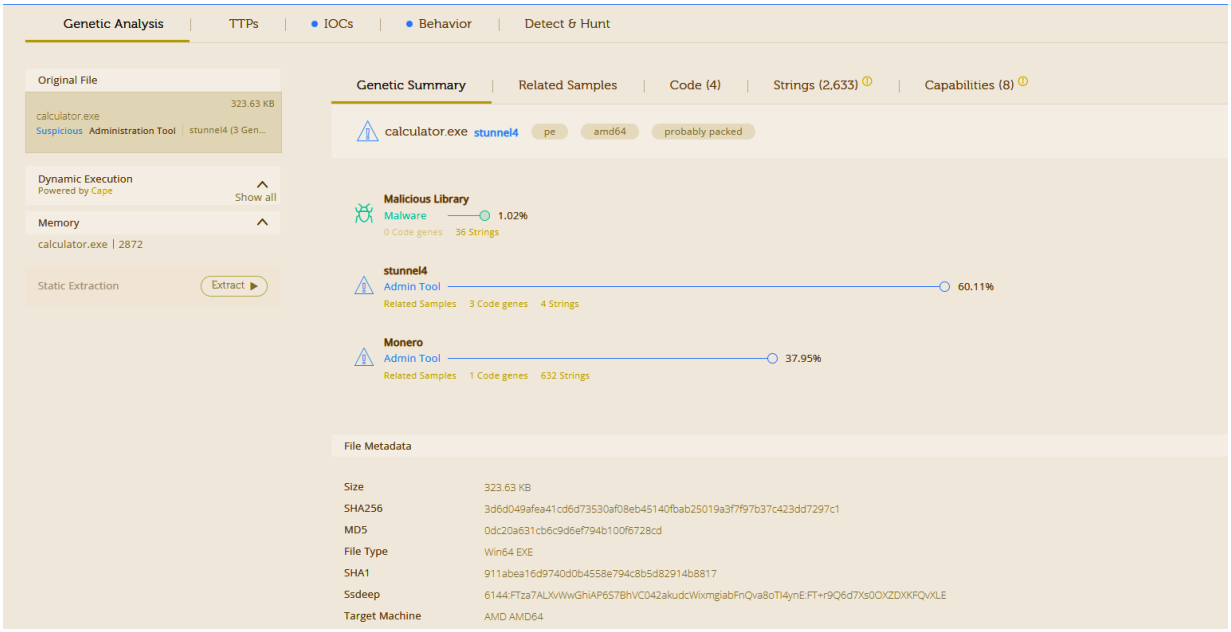

Рисунок 2.5 – Результати сканування Intezer Analyze файлу calculator.exe

На рисунках 2.6 та 2.7 представлені результати сканування файлу з вмістом ідентичним до EICAR-Test-File онлайн антивірусами VirusTotal та Intezer Analyze відповідно. Обидва антивіруси успішно визначили даний файл як зловмисний.

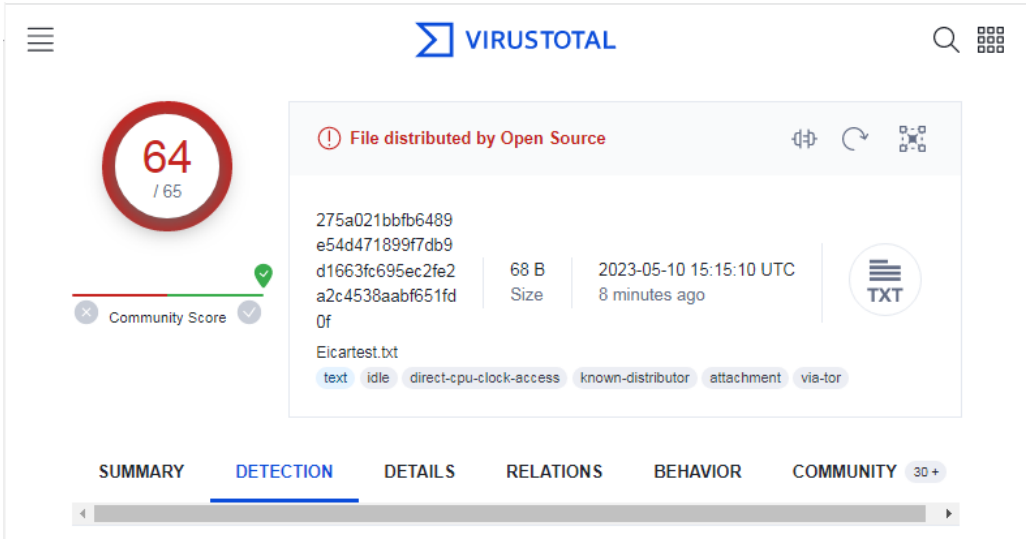

Рисунок 2.6 – Результати сканування VirusTotal файлу EICAR

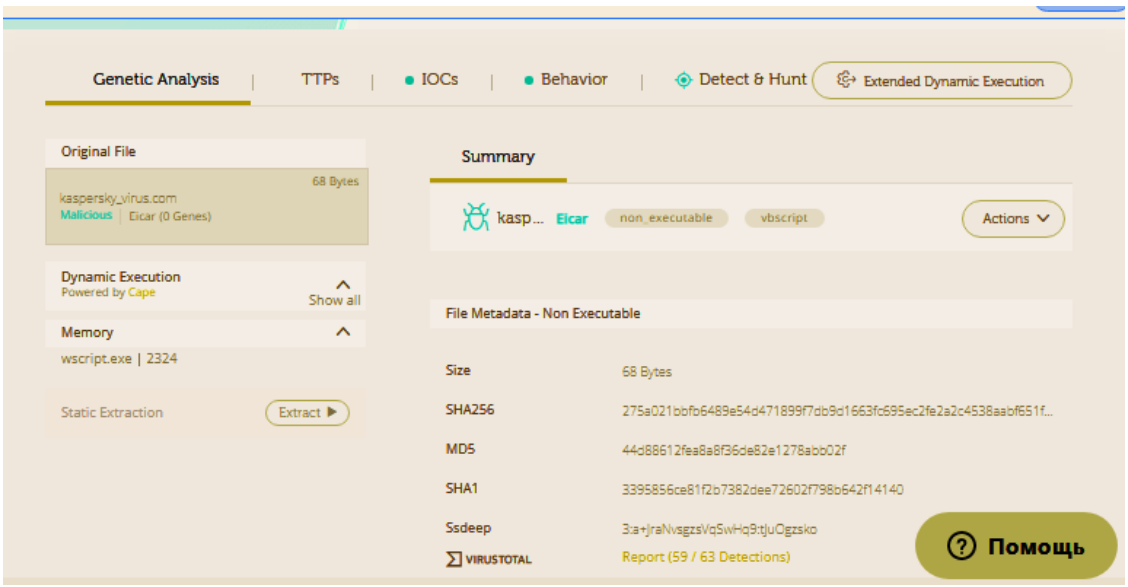

Рисунок 2.7 – Результати сканування Intezer Analyze файлу EICAR

Підсумовуючи описане, однією з головних переваг Intezer Analyze над VirusTotal є його здатність забезпечити детальний аналіз коду та його походження. Однак Intezer Analyze є платним інструментом і орієнтований більше на спеціалістів та дослідників, а не на звичайних користувачів. VirusTotal, з іншого боку, є безкоштовним інструментом, який доступний кожному та більше підходить для швидкого та легкого аналізу зловмисного програмного забезпечення. Тож, хоча обидва інструменти пропонують аналіз і виявлення зловмисного програмного забезпечення, вони відрізняються підходом, функціями та цільовою аудиторією. Вибір відповідного інструменту залежить від конкретних потреб і вимог користувача.

Хоча багаторазове сканування файлів антивірусом може бути корисним методом для статичного аналізу шкідливого програмного забезпечення, воно також має кілька недоліків.

- Споживання системних ресурсів: багаторазове сканування файлів може бути ресурсомістким процесом і може сповільнити продуктивність комп’ютерної системи, що аналізується. Це може бути особливо проблематично у великих системах або при роботі з великим обсягом файлів.

- Помилкове спрацьовування (FP): повторне сканування іноді може призвести до хибних спрацьовувань, коли законні файли позначаються як потенційні загрози. Це може призвести до непотрібних сповіщень і втрати часу на дослідження помилкових спрацьовувань.
- Помилкове не спрацьовування (False negative rate – FN): повторне сканування також може призвести до помилкових не-спрацьовувань, коли файли, які насправді є шкідливими, не виявляються.

2.1.2 File fingerprinting

Перевірка відбитків файлів (File fingerprinting) — це метод статичного аналізу, який використовується для виявлення та ідентифікації шкідливого програмного забезпечення. Ця техніка передбачає створення «геш-значення» або «цифрового відбитка» для кожного файлу в комп'ютерній системі, який служить унікальним ідентифікатором цього файлу.

Порівнюючи геш-значення файлу з базою даних відомих відбитків шкідливих програм, з'являється можливість швидко ідентифікувати файли, які раніше були визначені як шкідливі. Це може допомогти виявити та запобігти поширенню зловмисного програмного забезпечення.

Однією з найбільш розповсюджених функцій гешування, які застосовують аналітики з питань інформаційної безпеки, є MD5, однак серія SHA (Secure Hash Algorithm) також є популярною [13].

Загалом, знайдений геш може бути використаний у двох якостях: як ідентифікатор шкідливої програми та як мітка для пошуку в базах даних вірусів. Геш програми можна знайти за допомогою відповідних інструментів, таких як геш-шифратори.

Для створення гешу файлу можна скористатися наступними застосунками: Hash Generator, HashMyFiles, OpenHashTab, QuickHash .

На рисунках 2.8 і 2.9 представлені результати гешування файлу calculator.exe за допомогою програм Hash Generator та HashMyFiles відповідно.

Hash Generator дає можливість генерувати геші та контрольні суми з файлів і тексту. Він підтримує широкий спектр типів гешів і здатний генерувати їх всього за кілька секунд.

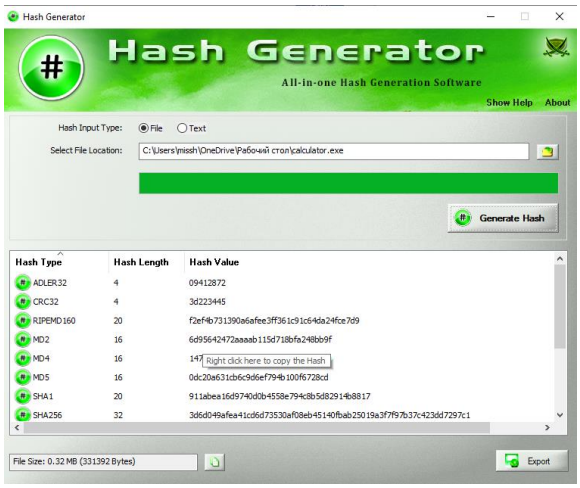

Рисунок 2.8 – Результати гешування файлу за допомогою Hash Generator

HashMyFiles – невелика безкоштовна утиліта, яка дозволяє обчислити геш одного або кількох файлів за допомогою алгоритмів MD5, SHA1 та Cyclic redundancy check 32 (CRC32). Програма також може бути запущена з контекстного меню стандартного Провідника та відображати MD5/SHA1 геш-суми вибраних файлів і папок.

Список геш-сум можна копіювати в буфер обміну або експортувати їх до текстового, HyperText Markup Language (HTML) або eXtensible Markup Language (XML) файлу.

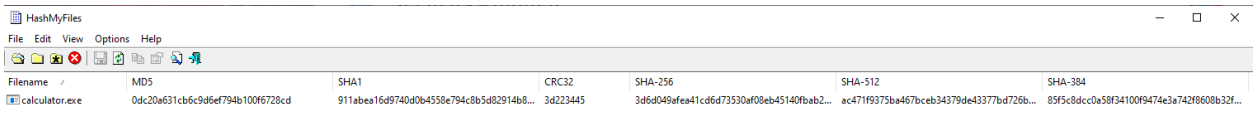

Рисунок 2.9 – Результати гешування файлу за допомогою HashMyFiles

Також гешування файлів можна здійснювати за допомогою онлайн ресурсів, як Md5file, Sisik або Hash-file. Результати їх роботи при гешуванні файлу calculator.exe представлені на рисунках 2.10, 2.11 та 2.1.2 відповідно.

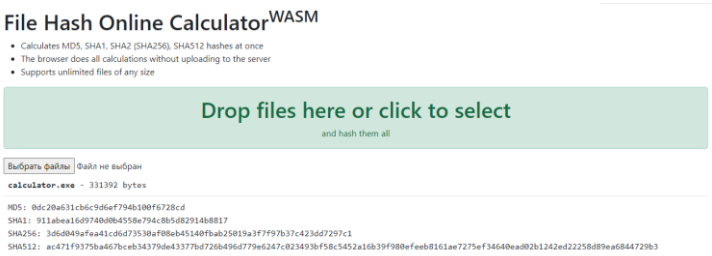

Рисунок 2.10 – Результати гешування файлу за допомогою Md5file

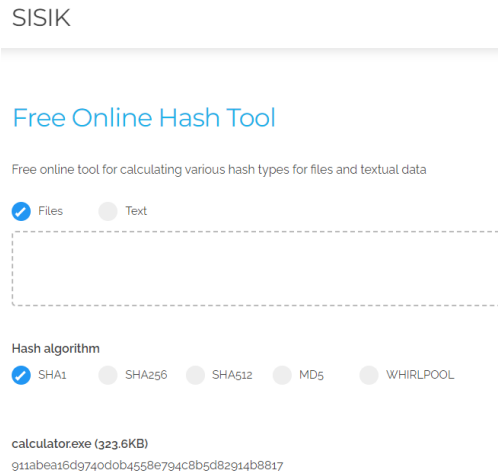

Рисунок 2.11 – Результати гешування файлу за допомогою Sisik

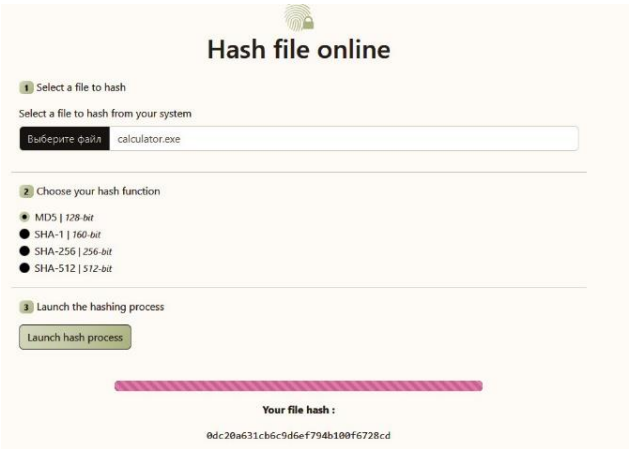

Рисунок 2.12 – Результати гешування файлу за допомогою Hash-file

Для створення гешу файлу зручніше користуватися онлайн сервісами, оскільки вони не потребують завантаження на встановлення на ПК користувача, що в певній мірі економить ресурси. Хоча також необхідно звернути увагу на довіру до подібних сервісів: вони мають мати хороший рейтинг та довіру користувачів, бути надійними та довіреними, безпечними. Щодо онлайн сервісів, деякі з них надають вибір, який алгоритм гешування

використовувати, а деякі виводять одразу список з результатами гешування усіма доступними на сторінці алгоритмами. В незалежності від того користуватися першим типом інструментів, чи другим, результати гешування надаються миттєво.

Отримані геші можна перевірити за допомогою бази вже зазначеного вище VirusTotal, Malware Hash Registry або MalwareBazaar Database. На рисунку 2.13 представлений пошук по гешу в базі VirusTotal, та результати пошуку на рисунку 2.14.

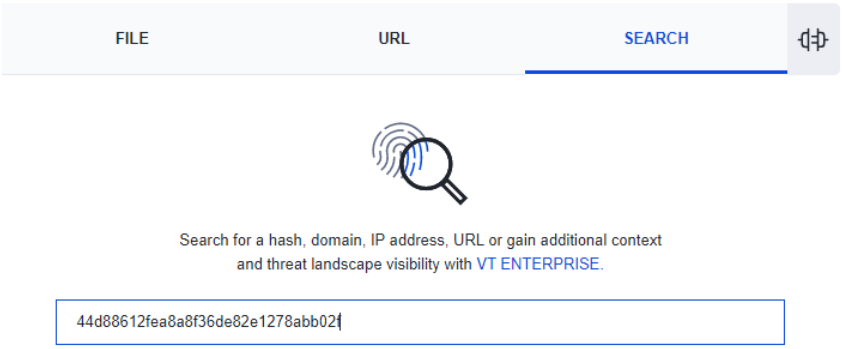

Рисунок 2.13 – Пошук по гешу у VirusTotal

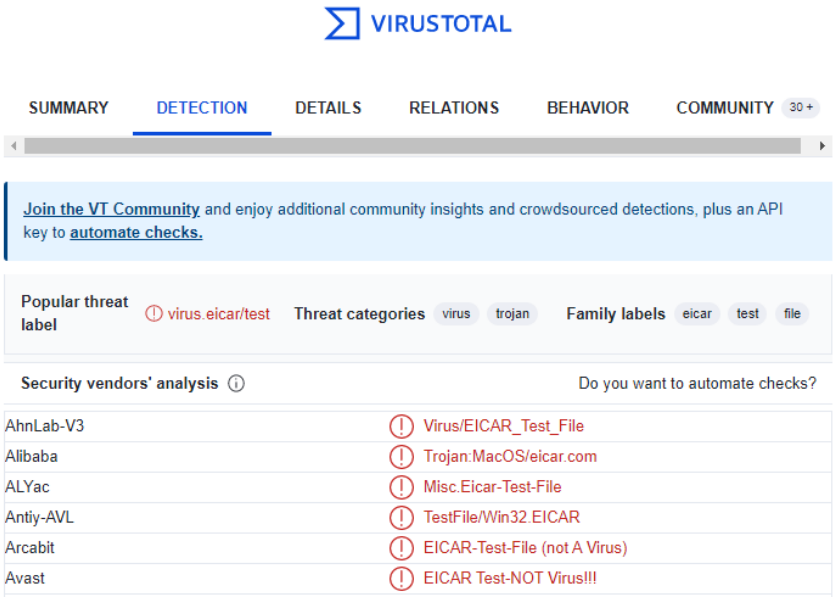

Рисунок 2.14– Результати пошуку по гешу в VirusTotal

Malware Hash Registry та MalwareBazaar Database володіють меншою базою зловмисних гешів, тому перевагу при перевірці краще надавати VirusTotal.

Переваги методу:

- Є високоточним методом ідентифікації зловмисного ПЗ. Навіть невеликі зміни у файлі призведуть у результаті до іншого геш-значення, що означає, що дуже важко підробити або змінити відбиток файлу.
- File fingerprinting є швидким методом аналізу, оскільки він передбачає лише обчислення геш-значення файлу. Це робить його корисним для швидкого аналізу великої кількості файлів.
- Для зберігання геш-значень файлів потрібен лише невеликий обсяг пам’яті.

Недоліки методу:

- Цей метод обмежується ідентифікацією відомого шкідливого програмного забезпечення, що означає, що воно може бути неефективним проти нових або невідомих загроз.
- Сприйнятливість до хибних спрацьовувань: метод може мати хибні спрацьовування, якщо два файли мають однакове геш-значення, що називається геш-колізією. Це може статися, якщо файл було навмисно змінено, щоб мати те саме геш-значення, що й відомий файл.

2.1.3 Пошук по рядках

Пошук по рядках може бути простим способом отримати інформацію щодо функціональності програми. Наприклад, якщо програма отримує доступ до URL-адреси, ви побачите URL-адресу, до якої звернулися, збережену в програмі як рядок.

Аналіз рядків може допомогти отримати значну кількість корисної інформації. Наприклад, якщо програма взаємодіє з URL-адресою, то можна знайти відповідну адресу, яка зберігається у вигляді рядку в виконавчому файлі програми. Рядки у виконавчому файлі зберігаються у форматах ASCII або Unicode, і для їх пошуку використовують спеціальні програми. Під час пошуку програма ігнорує контекст та форматування [13], що дозволяє знайти потрібні рядки в будь-якому місці та у файлах різних типів.

Доступними програмами для пошуку рядків є Midnight Commander, Multi Commander, Total Commander. На рисунках 2.15 і 2.16 представлений пошук рядків за допомогою Total Commander та Multi Commander відповідно.

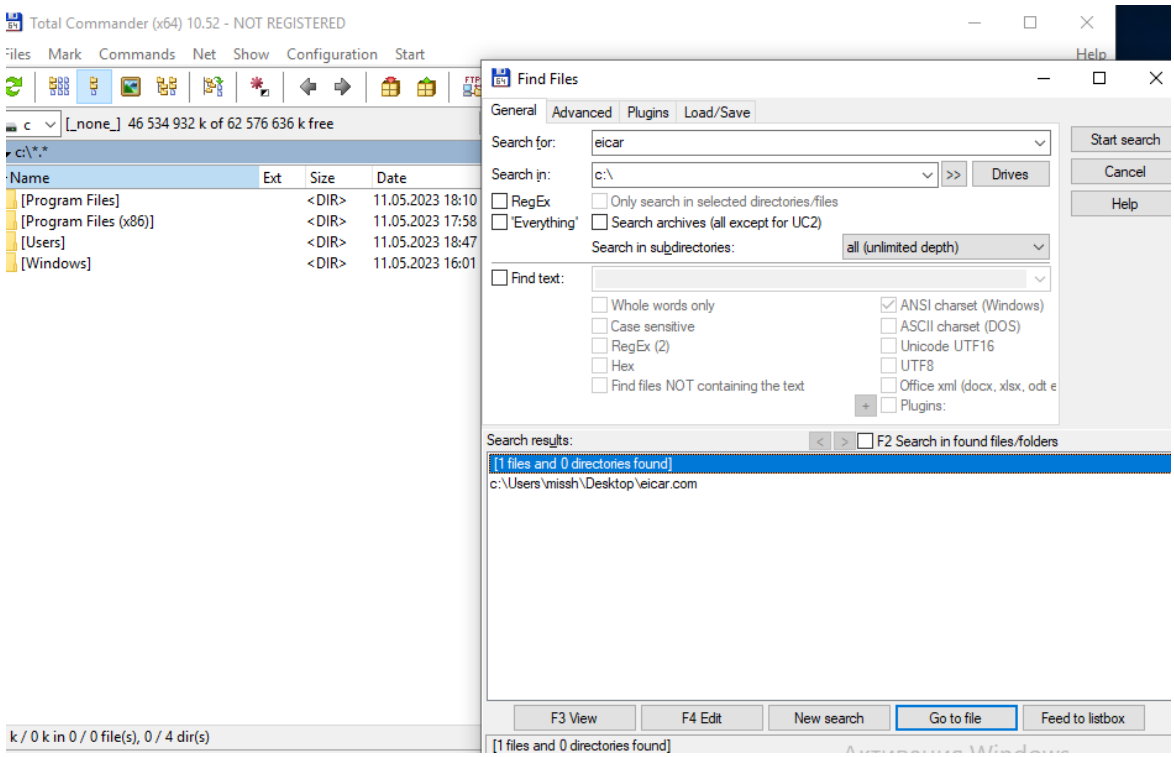

Рисунок 2.15 – Пошук рядків за допомогою Total Commander

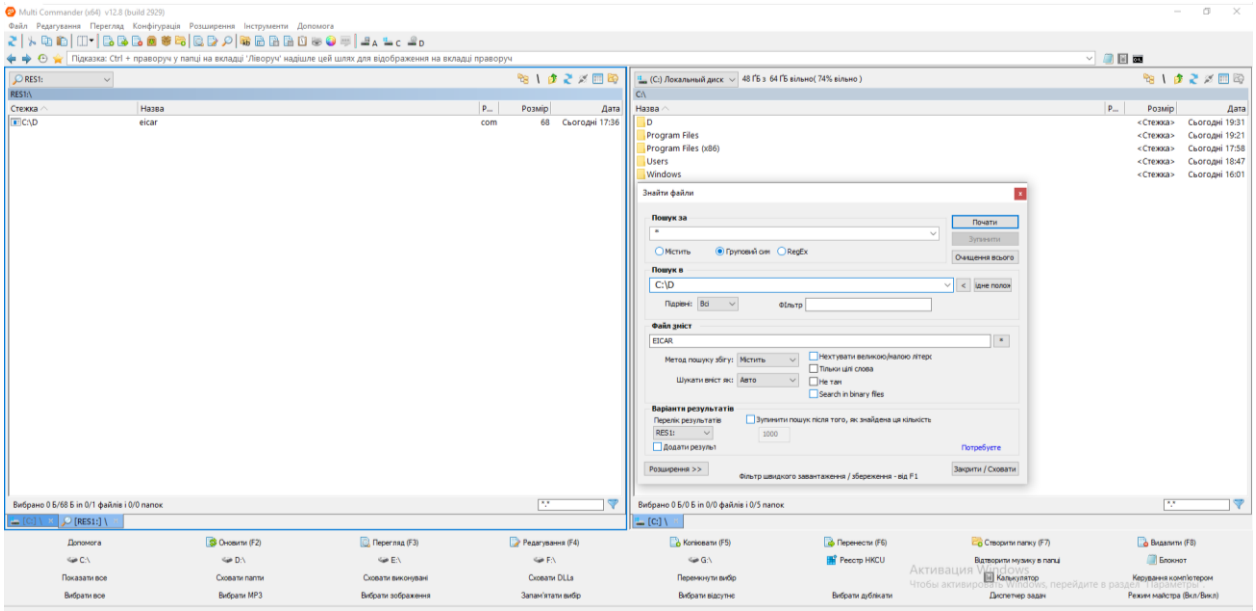

Рисунок 2.16 – Пошук рядків за допомогою Multi Commander

Якщо робити порівняння обох програм, то можна виділити наступне:

- Total Commander, і Multi Commander мають подібні можливості пошуку. Вони можуть шукати рядки в іменах файлів, вмісті файлів, а також надавати розширені параметри пошуку, такі як чутливість до регістру та регулярні вирази.
- Total Commander за низкою експерементів показав більш високу швидкість роботи, аніж Multi Commander
- Multi Commander пропонує більш розширені фільтри пошуку, ніж Total Commander. Наприклад, він може фільтрувати пошук за типом файлу, датою зміни та розміром. Це зручно при аналізі великої кількості файлів.
- Як Total Commander, так і Multi Commander відображають результати пошуку у форматі списку, де вказується ім’я файлу, розташування та інші важливі відомості. Однак Multi Commander надає більш детальну інформацію, таку як розмір файлу, дата зміни та атрибути.

Загалом і Total Commander, і Multi Commander здатні ефективно шукати рядки у файлах. Total Commander має вищу швидкість пошуку, тоді як Multi Commander пропонує більш розширені фільтри пошуку та детальніші результати пошуку.

Переваги методу:

- Ефективність як проти відомих, так і невідомих загроз: пошук рядків може бути ефективним проти відомих і невідомих загроз, оскільки він шукає певні шаблони, які часто пов’язані зі зловмисним програмним забезпеченням, наприклад рядки команд і керування або певні сигнатури зловмисного програмного забезпечення.
- Може ідентифікувати декілька типів загроз. Пошук за рядками можна використовувати для ідентифікації кількох типів загроз, таких як віруси, трояни, хробаки та шпигунські програми, шляхом пошуку певних рядків або шаблонів, пов’язаних із кожним типом загрози.

- Цей метод можна використовувати в режимі реального часу для моніторингу мережевого трафіку та визначення потенційних загроз у міру їх виникнення.

Недоліки методу:

- Пошук рядків може генерувати високу кількість хибних спрацьовувань, оскільки невинні файли чи мережевий трафік можуть містити ті самі рядки чи шаблони, що й шкідливе програмне забезпечення.
- Пошук рядків обмежується виявленням загроз, які відповідають певним рядкам або шаблонам, що означає, що він може бути неефективним проти нових або невідомих загроз, які не відповідають цим шаблонам.
- Автори зловмисного ПЗ можуть легко обійти пошук рядків шляхом обфускації або шифрування рядків або шаблонів, пов'язаних із їхнім зловмисним програмним забезпеченням. У результаті пошук рядка може бути неефективним проти складних або цілеспрямованих атак.

2.1.4 Виявлення пакувальників

Часто автори зловмисного програмного забезпечення обфускують свої коди так, що файли важко читати. За допомогою статичного аналізу дуже важко передбачити, які файли упаковані, якщо це чітко не видно. Запаковані програми є підмножиною обфускованих; їх вміст стислий, і аналізувати його неможливо. Ці два прийоми сильно обмежують застосування статичного аналізу.

Нешкідливі програми зазвичай містять безліч рядків, а ось в упакованих та обфускованих програмах їх дуже мало. Якщо при дослідженні програми виявиться лише кілька рядків, це може означати, що вона є обфускованою або запакованою і, можливо, містить шкідливий код.

Коли виконується упакована програма, програма-оболонка також запускається, щоб розпакувати її. При статичному аналізі упакованої програми можливо вичленувати лише цю оболонку.

Визначити, чи файл упакований, можна за допомогою програми PEiD (PEiD 0.95: сайт. URL: https://www.softpedia.com/get/Programming/Packers-

Crypters-Protectors/PEiD-updated.shtml). Ця утиліта дозволяє встановити тип пакувальника або компілятора, які використовувалися при складанні програми, що значно спрощує аналіз упакованих даних [13].

2.1.5 Перевірка інформації у PE-заголовку

Формат файлу PE — це структура даних, яка містить інформацію, необхідну завантажувачу операційної системи Windows для обробки програмного коду. Він використовується виконуваним файлом Windows, об’єктним кодом і бібліотеками DLL [14]. Файл PE складається із заголовка PE (Header) та розділів PE (Sections), як показано на рисунку 2.17.

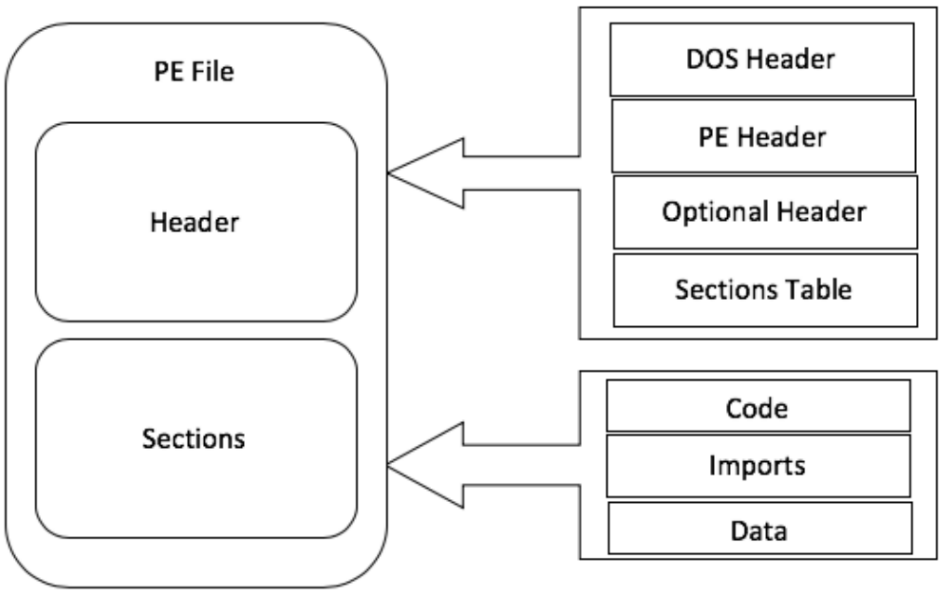

Рисунок 2.17 - Формат портативного виконуваного файлу

Формат PE-файлу організований як лінійний потік даних. Перший сегмент є заголовком і містить необхідну інформацію про код і тип його програми, функції бібліотеки чи ядра та скільки місця. Він починається з заголовка DOS (DOS header) – показує, що це за двійковий формат. Відразу за ним слідує PE заголовок (PE Header) і необов'язковий заголовок (Optional Header), що містить інформацію про виконуваний файл. Далі йде таблиця секцій (Sections Table) котра визначає, як файл буде завантажено в пам’ять. Секції (Sections) — це другий сегмент, що складається з: коду (Code), що буде виконуватися; імпорту (Imports), котрий здійснює зв’язок між виконуваним

файлом EXE та динамічними бібліотеками DLL; даних (Data) – інформація, що необхідна коду програми.

Для аналізу PE-заголовку можна популярними є наступні інструменти:

- Exeinfo PE;
- PEstudio;
- CFF Explorer;
- FileAlyzer;
- PEview.

PEstudio — це спеціалізований інструмент для аналізу PE-файлів (рисунок 2.18). Він пропонує ряд функцій, призначених для детального аналізу PE-файлів і виявлення потенційних ризиків безпеці, вразливостей і підозрілої поведінки. Також, PEstudio є безкоштовним для некомерційного використання. Для комерційних користувачів доступні ліцензії.

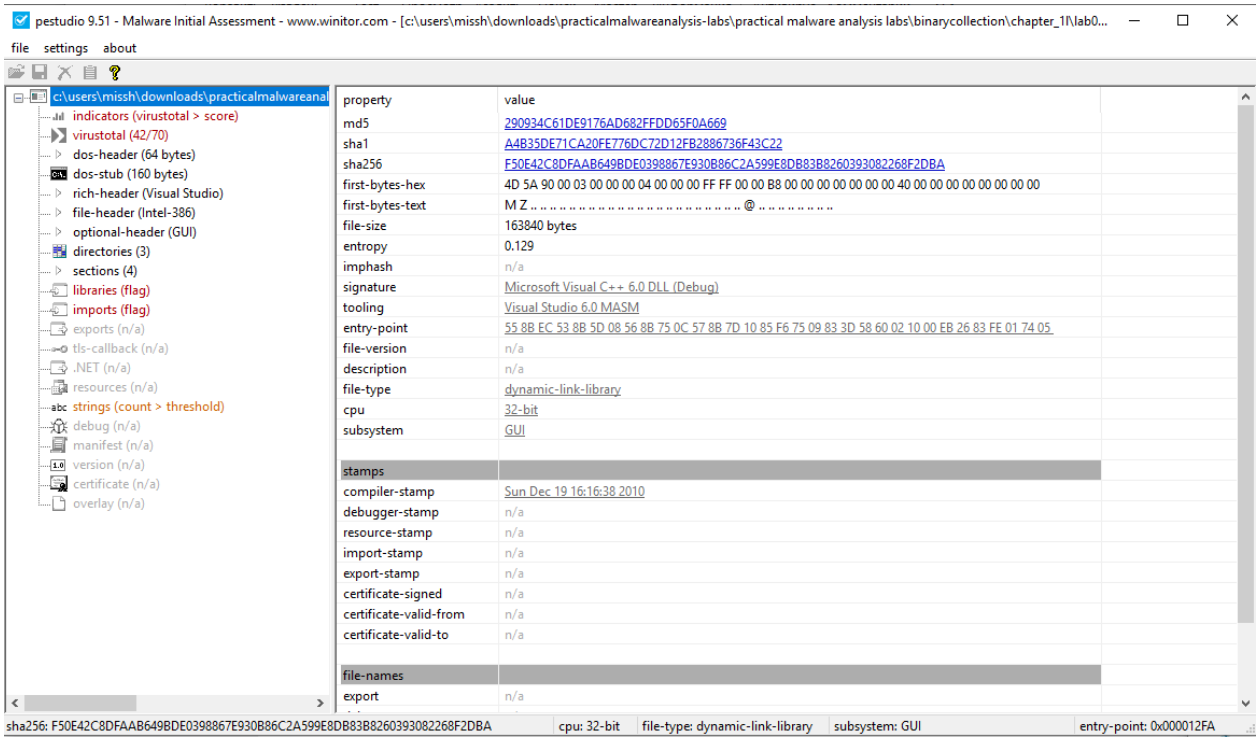

Рисунок 2.18 – Робота програми PEstudio

Exeinfo PE — це програмний інструмент, який використовується для аналізу та ідентифікації виконуваних файлів в операційній системі Windows (рисунок 2.19). Його можна використовувати для ідентифікації типу

виконуваного файлу, компілятора, використаного для створення файлу, і будь-яких пакувальників або засобів захисту, які могли бути використані для обфускації коду. Exeinfo PE також можна використовувати для виявлення та видалення шкідливих програм із виконуваних файлів.

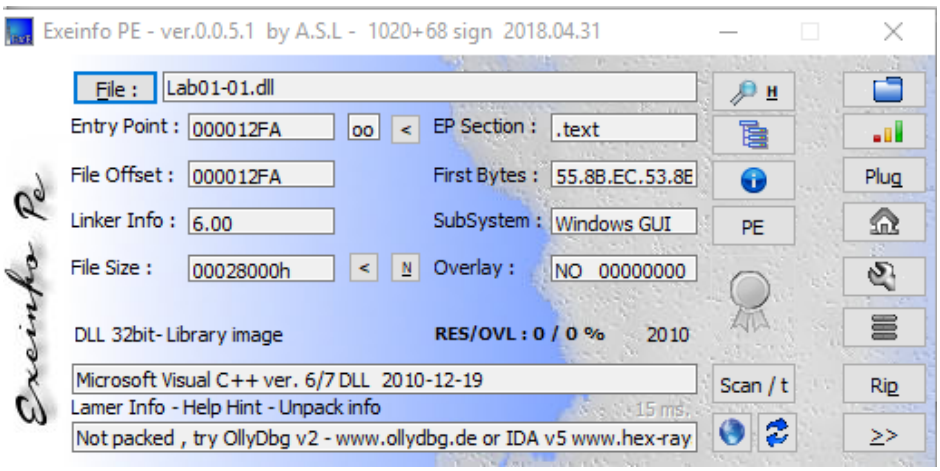

Рисунок 2.19 –Робота програми Exeinfo PE

Порівнюючи Exeinfo PE та PEstudio можна виділити наступні відмінності між цими інструментами:

- Exeinfo PE сумісний зі старішими версіями Windows, тоді як для роботи PEstudio потрібна новіша версія Windows.
- PEstudio має більше можливостей і опцій, ніж Exeinfo PE, включаючи здатність виявляти та аналізувати зловмисне програмне забезпечення, ідентифікувати пакувальники та обфускатори та аналізувати цифровий підпис файлу.
- Exeinfo PE має простіший інтерфейс користувача порівняно з PEstudio, який має більш складний інтерфейс із більшою кількістю параметрів і функцій.
- Exeinfo PE є безкоштовним інструментом, тоді як PEstudio вимагає ліцензійної плати за використання.

Загалом обидва інструменти корисні для аналізу PE-файлів, але PEstudio пропонує більш розширені функції та параметри, тоді як Exeinfo PE є простішим і зрозумілішим інструментом.

2.1.6 Аналіз дизасемблерного коду

Дизасемблювання — це аналіз двійкового коду без його виконання, який можна виконати за допомогою таких інструментів, як дизассемблери або декомпілятори, щоб отримати асемблерний код або навіть вихідний код високого рівня з двійкових файлів. Цей процес безпечний і не активує жодних шкідливих корисних навантажень. Крім того, він може розкривати такі метадані, як інформація про компілятор, назви функцій або символи налагодження, щоб допомогти ідентифікувати сімейство зловмисного програмного забезпечення або зловмисника.

Однак статичне розбирання також може бути неточним або неповним, якщо двійковий код заплутаний, зашифрований або упакований. Це може ввести в оману або заплутати, якщо певні розділи коду є нерелевантними, мертвими або умовними залежно від вхідних даних або змінних середовища. Нарешті, для аналітика може бути виснажливим і трудомістким вручну досліджувати та інтерпретувати велику кількість асемблерного коду, який може бути складним і низькорівневим. Процес розбирання програми об'ємом від п'яти до десяти мегабайт може бути тривалим і трудомістким. Хоч переважна більшість зловмисного ПЗ і має і має певні відмінні риси, які відрізняють їх від легітимних програм, надійність подібних індикаторів значно нижча, і якась кількість зловмисного ПЗ лишається непоміченою.

Значні трудовитрати роблять такий спосіб аналізу надзвичайно непривабливим і, в якійсь мірі, безперспективним [13].

Серед програм призначених для дизасемблювання виділяють наступні:

- IDA Pro;
- Radare2;
- Binary Ninja.

IDA Pro — це передовий комерційний інструмент зі зручним інтерфейсом і широкою підтримкою плагінів, що робить його придатним для професійних аналітиків, яким потрібні потужні аналітичні можливості.

Radare2 — це фреймворк із відкритим вихідним кодом, який пропонує інтерфейс командного рядка, що робить його придатним для досвідчених користувачів, які віддають перевагу налаштуванню та мають навички створення сценаріїв.

Binary Ninja — це комерційний дизассемблер із сучасним графічним інтерфейсом, орієнтований на користувачів, яким потрібен інтуїтивно зрозумілий та інтерактивний аналіз із гнучкістю налаштування плагінів.

2.2 Динамічний аналіз

У той час як статичний аналіз залежить від перевірки вмісту певних файлів і програм на наявність потенційно шкідливого вмісту, динамічний аналіз зловмисного програмного забезпечення є технікою, яка використовується для розуміння поведінки ймовірного зловмисного програмного забезпечення шляхом його запуску в контрольованому середовищі, відомому як пісочниця.

Метою динамічного аналізу є спостереження за діями зловмисного програмного забезпечення під час його роботи та виявлення будь-якої зловмисної поведінки, яке воно демонструє. Цей підхід може допомогти фахівцям із безпеки отримати уявлення про можливості та наміри зловмисного програмного забезпечення. Результати динамічного аналізу зловмисного програмного забезпечення можна використовувати для покращення захисту від подібних загроз або розроблення стратегії відновлення для заражених систем.

Існує два різні підходи [8] до динамічного аналізу зловмисного програмного забезпечення, кожен з яких забезпечує різну деталізацію та якість:

- отримання зображення повного стану системи перед запуском шкідливого програмного забезпечення та порівняння його з повним станом системи після виконання; і
- моніторинг дій шкідливого ПЗ під час виконання за допомогою спеціалізованих інструментів.

Перший варіант простіший у реалізації, але забезпечує більш грубі результати, яких іноді достатньо для отримання загального уявлення про схему дій аналізованого двійкового файлу. Цей підхід аналізує лише кумулятивні ефекти зловмисного програмного забезпечення без урахування динамічних змін як, наприклад, створення зловмисним програмним забезпеченням файлу під час його виконання та видалення цього файлу перед завершенням роботи.

Другий підхід важче реалізувати, хоча спостереження за поведінкою програми під час виконання наразі є найбільш перспективним підходом. Моніторинг дій шкідливого ПЗ під час виконання здебільшого здійснюється з використанням ізольованого програмного середовища - пісочниці. Пісочниця тут означає контрольоване середовище виконання, яке відокремлене від решти системи для ізоляції зловмисного процесу. Таке розділення зазвичай досягається за допомогою механізмів віртуалізації.

Загалом схема виконання динамічного аналізу шкідливого ПЗ виглядає наступним чином [13]:

1) Повернення віртуального середовища в початковий стан.
2) Запуск інструментів моніторингу зловмисного ПЗ.
3) Виконання зловмисного ПЗ.
4) Зупинка інструментів моніторингу.
5) Аналіз результатів.

Інструменти для моніторингу зловмисного ПЗ є різноманітними і обираються в залежності від методу динамічного аналізу.

Поширеними методами динамічного аналізу є:

- моніторинг модифікацій файлів;
- моніторинг змін реєстру ;
- моніторинг мережевої активності;
- моніторинг системних викликів;
- відладка (Debugging).

Моніторинг модифікацій файлів. Через те, що більшість зловмисного ПЗ вносить зміни у файли, що зберігаються у файловій системі, моніторинг модифікації файлів є ефективним способом аналізу поведінки такого ПЗ.

Відстеження змін реєстру. Коли зразок зловмисного програмного забезпечення виконується, він часто вносить зміни до реєстру, щоб досягти стійкості, встановити зв'язок із своїм сервером керування або виконати інші шкідливі дії. При відстеженні цих змін з'являється можливість отримати уявлення про поведінку та можливості шкідливого програмного забезпечення.

Моніторинг діяльності зловмисного ПЗ. Зловмисне програмне забезпечення часто створює дочірні процеси для виконання певних завдань, таких як зв'язок із сервером керування та завантаження додаткових компонентів та кілька потоків для виконання певних завдань, таких як зв'язок із сервером керування та викрадання даних. Відстежуючи ці потоки та процеси можна проаналізувати зловмисне ПЗ.

Моніторинг мережевої активності. Оскільки багато зловмисних програм проявляють свою реальну поведінку лише при підключенні до мережі, моніторинг мережевої активності є способом виявлення таких програм. Для імітації підключення до мережі виконання відбувається ізольованому середовищі, такому як віртуальна машина або пісочниця, де мережеву активність зловмисного ПЗ можна зафіксувати та проаналізувати.

Моніторинг системних викликів. Під час роботи в системі зловмисне програмне забезпечення взаємодіє з операційною системою через системні виклики. Системні виклики — це механізм, за допомогою якого програма запитує послугу в операційної системи. Моніторинг системних викликів може надати інформацію про функції, що використовуються виконуваним файлом.

Відладка (Debugging). Ця техніка передбачає запуск зразка зловмисного програмного забезпечення у відладчику, що дозволяє аналітикам покроково переглядати код і детально перевіряти поведінку програми.

2.2.1 Моніторинг модифікацій файлів

Оскільки більшість вірусів змінюють файли, що зберігаються у файловій системі, чудовим способом проаналізувати їх поведінку є виконання їх у тестовій системі та моніторинг змін файлів.

На травень 2023 року існує багато інструментів для моніторингу змін файлів і папок у реальному часі. Далі перераховані кілька з найпопулярніших безкоштовних варіантів: Watch 4 Folder; Disk Pulse; FileAudit; Moo0 File Monitor; SpyMe Tools.

Watch 4 Folder – це утиліта моніторингу та автоматизації папок, яка може відстежувати до 4 різних папок одночасно на 12 типів подій і реагувати на кожну подію різними видами дій. На рисунку 2.20. представлений вигляд програми та показані стандартні налаштування, серед яких можна обрати необхідні. Інструмент надає сповіщення в реальному часі (рисунок 2.21) про будь-які зміни, що робить його корисним для швидкого виявлення будь-яких несанкціонованих змін або підозрілої активності. Watch 4 Folder також пропонує ряд параметрів налаштування, таких як фільтрація за типом файлу або налаштування певних дій, які запускаються у відповідь на певні події.

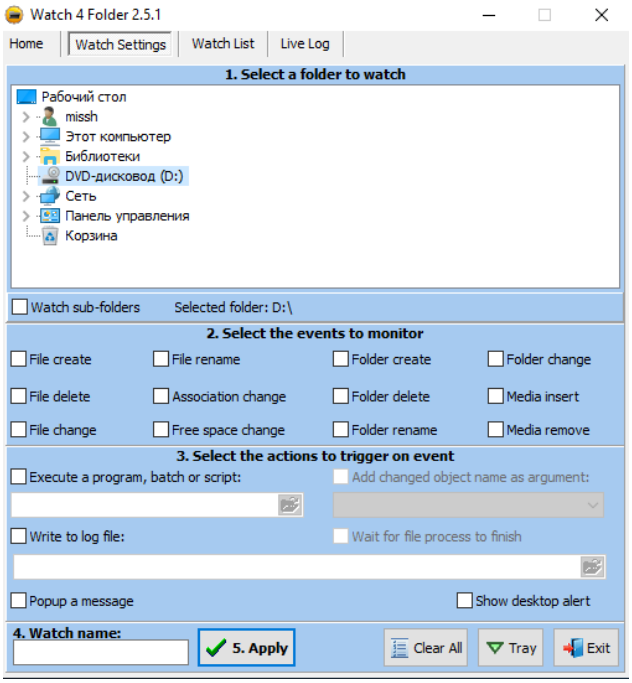

Рисунок 2.20 – Програма Watch 4 Folder

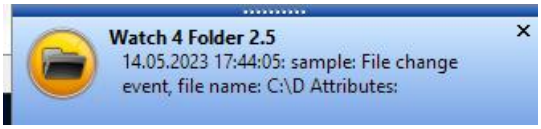

Рисунок 2.21 – Сповіщення Watch 4 Folder у реальному часі

Disk Pulse (рисунок 2.22) також дозволяє відстежувати один або кілька дисків або каталогів, виявляти зміни файлової системи, а також може надсилати повідомлення електронною поштою або виконувати команди користувача при виявленні однієї або декількох критичних змін. На додаток до моніторингу файлів і папок на наявність змін, Disk Pulse також може контролювати продуктивність системи та відстежувати будь-які зміни, внесені в реєстр або мережеві підключення.

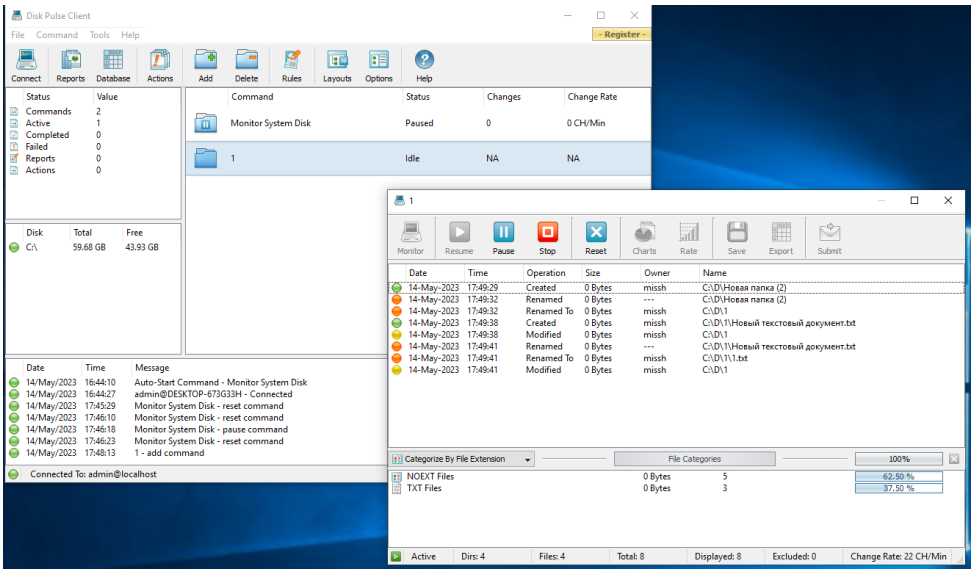

Рисунок 2.22 – Відстеження змін файлів за допомогою Disk Pulse

Загалом, Disk Pulse є більш досконалим інструментом, який пропонує широкий спектр функцій моніторингу та аналізу. Однією з головних переваг Disk Pulse над Watch 4 Folder є розширені можливості звітування та аналізу. Disk Pulse може створювати налаштовані звіти та діаграми, які надають детальну інформацію про зміни файлів, включаючи тип зміни, позначку часу та місце розташування. Це може бути корисним для виявлення закономірностей або аномалій у роботі файлів, таких як раптове збільшення кількості модифікацій файлів або сплеск мережевого трафіку.

Однак Disk Pulse є складнішим інструментом, і він може не підходити для користувачів, яким потрібна лише базова функція моніторингу файлів. Watch 4 Folder, з іншого боку, є простим і легким у використанні інструментом, який підходить для користувачів, яким потрібно стежити за певною папкою чи файлом на наявність змін і отримувати сповіщення в реальному часі.

2.2.2 Моніторинг змін реєстру

Даний метод передбачає моніторинг реєстру Windows на наявність будь-яких змін, внесених зразком зловмисного програмного забезпечення під час його виконання в контрольованому середовищі. Реєстр Windows — це централізована база даних, у якій зберігаються параметри конфігурації та параметри операційної системи Windows і встановлених програм.

Відстеження змін у реєстрі може бути особливо корисним для аналізу зразків зловмисного програмного забезпечення, яке використовує механізми збереження, наприклад створення розділів реєстру або зміну існуючих. Аналізуючи зміни реєстру, внесені зловмисним програмним забезпеченням, з’являється можливість визначити механізм збереження, який використовує зловмисне програмне забезпечення, і вжити заходів для його видалення.

Існує кілька популярних інструментів відстеження змін реєстру, які використовуються для динамічного аналізу зловмисного програмного забезпечення, зокрема:

- Regshot;
- Tiny Watcher;
- WhatChanged;
- SysTracer.

Regshot — це засіб відстеження змін реєстру з відкритим кодом, який дозволяє користувачам робити знімки реєстру до та після внесення змін. Після цього Regshot порівнює знімки та створює звіт, що містить усі змінами, які були внесені до реєстру (рисунок 2.23). Regshot відомий своєю ретельністю та надійністю, і він може виявити навіть найменші зміни в реєстрі.

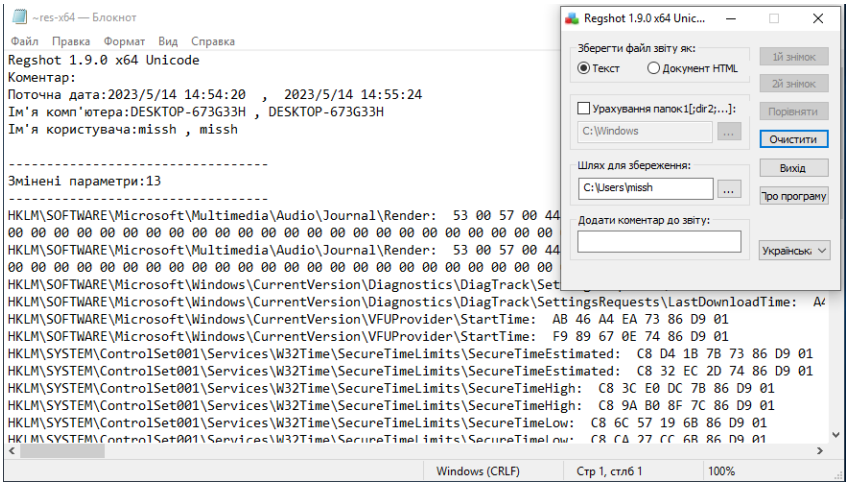

Рисунок 2.23 – Звіт та вікно програми Regshot

Tiny Watcher більш простий інструмент відстеження змін у реєстрі. При першому запуску Tiny Watcher, він робить початковий знімок реєстру. Після створення початкового знімка Tiny Watcher починає моніторинг реєстру в режимі реального часу. Він відстежує будь-які зміни, внесені в реєстр, наприклад додавання, модифікацію або видалення ключів або значень. Оскільки Tiny Watcher стежить за реєстром, він порівнює поточний стан реєстру з початковим знімком. Будь-які виявлені відмінності або зміни позначаються як зміни реєстру (рисунок 2.24).

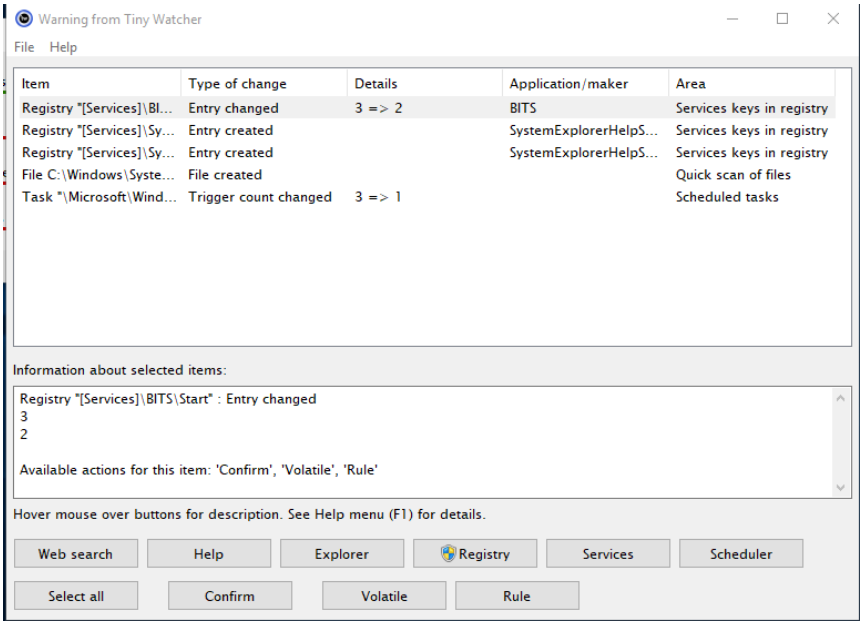

Рисунок 2.24 – Звіт програми Tiny Watcher

Підсумовуючи, Regshot потужніший і універсальніший, ніж Tiny Watcher з точки зору функцій. Regshot дозволяє користувачам робити кілька знімків реєстру, порівнювати їх і створювати звіти в різних форматах. Regshot також підтримує використання командного рядка та має більш розширені параметри фільтрації. Tiny Watcher, з іншого боку, є простішим і легшим, що робить його хорошим варіантом для користувачів, які потребують швидкого й легкого способу відстежувати зміни в реєстрі. Тож як висновок, Regshot потужніший і універсальніший, але може потребувати більше досвіду для ефективного використання, тоді як Tiny Watcher є простішим і зручнішим інструментом.

2.2.3 Моніторинг мережевої активності

Моніторинг мережі є важливою технікою, яка використовується в динамічному аналізі шкідливих програм. Він передбачає моніторинг мережевої активності зразка зловмисного програмного забезпечення під час його виконання в контрольованому середовищі, що допомагає визначити можливості та отримати уявлення про поведінку зловмисного ПЗ.

Моніторинг мережі може бути особливо корисним для аналізу зразків шкідливих програм, які використовують мережеві атаки, такі як ботнети або трояни віддаленого доступу. Аналізуючи мережевий трафік, створений зловмисним програмним забезпеченням, аналітики можуть визначити призначення трафіку, тип даних, що передаються, а також будь-які методи шифрування чи обфускації, які використовує зловмисне програмне забезпечення.

Серед найпоширеніших інструментів для моніторингу мережевої активності виділяють:

- SolarWinds Network Performance Monitor
- Wireshark
- Microsoft Network Monitor
- tcpdump
- Paessler PRTG Network Monitor

Wireshark — це безкоштовний аналізатор пакетів із відкритим кодом, який можна використовувати для захоплення та аналізу мережевого трафіку (рисунок 2.25). Wireshark має широкий спектр можливостей аналізу, включаючи здатність фільтрувати та шукати певні типи трафіку, витягувати файли та артефакти з мережевого трафіку та аналізувати моделі трафіку для виявлення аномалій і підозрілої активності.

Однією з ключових переваг Wireshark є його гнучкість і розширюваність. Він має широкий спектр плагінів і розширень, які можна використовувати для розширення його функціональності.

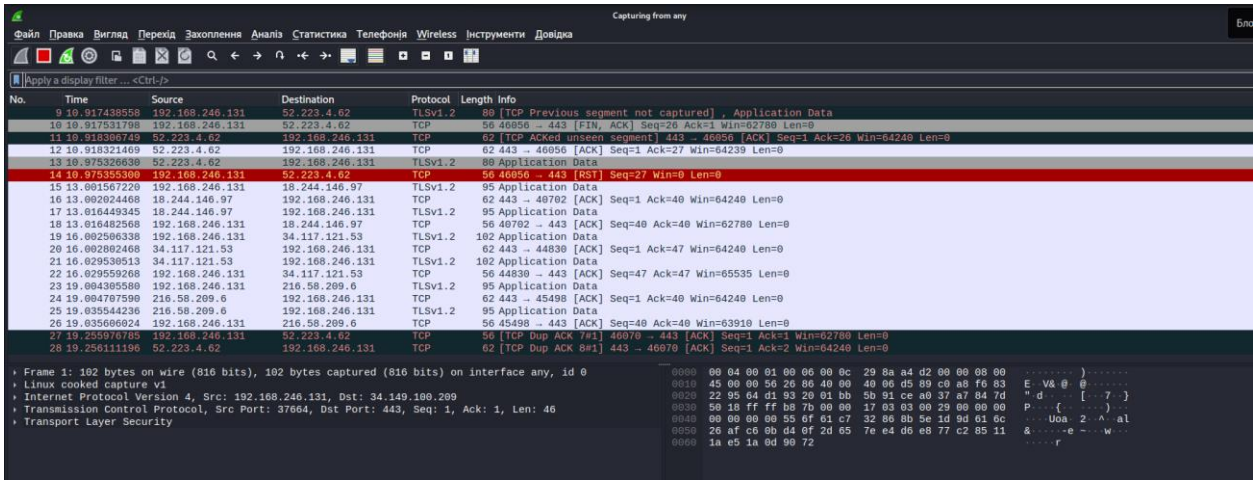

Рисунок 2.25 – Робота програми Wireshark

Microsoft Network Monitor — це інструмент на базі Windows, призначений для захоплення й аналізу мережевого трафіку (рисунок 2.26). Він має зручний інтерфейс і може використовуватися для моніторингу мережевої активності в режимі реального часу. Він також має вбудовані фільтри, які можна використовувати для захоплення певних типів трафіку, як-от трафік HTTP або трафік із певних IP-адрес.

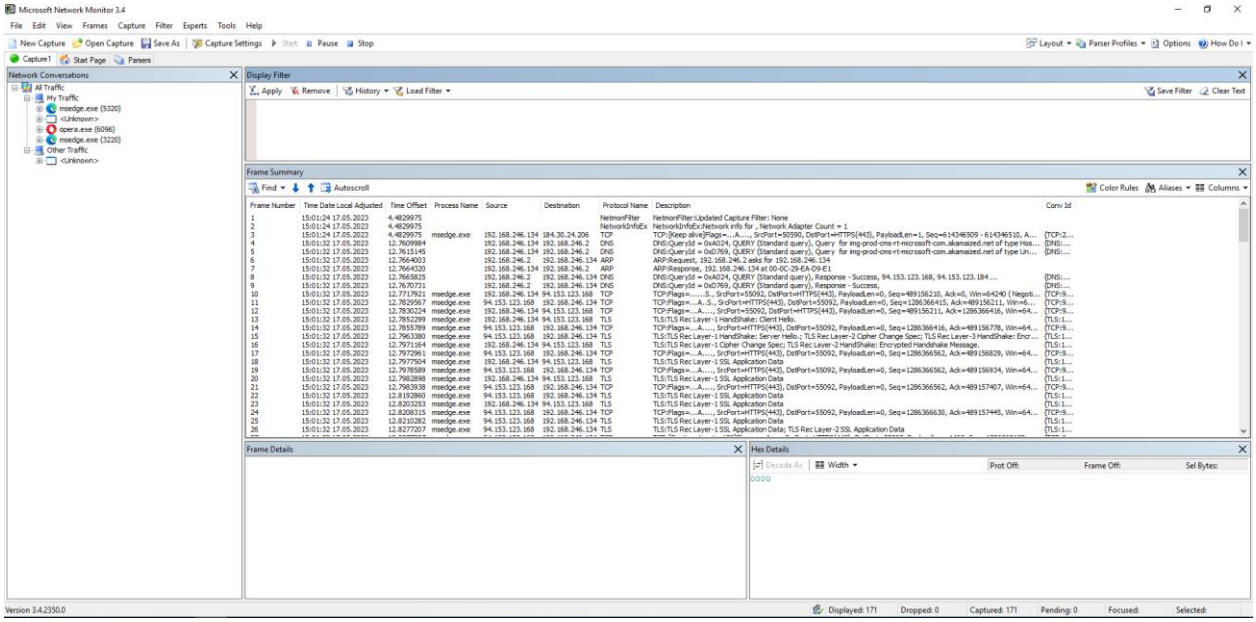

Рисунок 2.26 – Робота програми Microsoft Network Monitor

Порівнюючи обидва інструменти, можна виділити наступне:

- Microsoft Network Monitor — це інструмент на базі Windows, у той час як Wireshark є кросплатформним інструментом
- Wireshark має більш розширений інтерфейс користувача та надає більш детальні можливості аналізу порівняно з Microsoft Network Monitor.
- Microsoft Network Monitor має вбудовані фільтри, які можна використовувати для захоплення певних типів трафіку, але Wireshark має більш потужні та гнучкі параметри фільтрації.

Тож, з точки зору аналізу зловмисного ПЗ, Wireshark зазвичай вважається більш потужним інструментом. Його можна використовувати для ідентифікації конкретних типів трафіку зловмисного програмного забезпечення, наприклад трафіку керування та керування, а також для вилучення файлів і артефактів із мережевого трафіку. Microsoft Network Monitor також можна використовувати для аналізу зловмисного ПЗ, але може знадобитися більше ручного аналізу та фільтрації для виявлення зловмисної діяльності.

2.2.4 Моніторинг системних викликів

Системні виклики — це механізм, за допомогою якого процеси на рівні користувача спілкуються з ядром операційної системи. Вони забезпечують процеси для виконання таких операцій, як файловий ввід/вивід, мережевий зв’язок і керування процесами. Зловмисне програмне забезпечення часто використовує системні виклики для виконання зловмисних дій, таких як шифрування файлів, впровадження процесів або мережевий зв’язок.

Під час моніторингу системних викликів зразок зловмисного програмного забезпечення виконується в ізольованому середовищі, наприклад у віртуальній машині чи пісочниці, де його системні виклики можна перехопити та проаналізувати. Це дозволяє аналітикам спостерігати за системними викликами, зробленими зловмисним програмним забезпеченням, а також за будь-якими аргументами, переданими до цих викликів.

Серед інструментів відстеження змін реєстру, що використовуються для динамічного аналізу зловмисного програмного забезпечення поширеними є:

- Strace (утиліта Linux);
- Process Monitor;
- SystemExplorer.

Process Monitor – це монітор процесів Windows, який можна використовувати для відстеження системних викликів (рисунок 2.27). Це дозволяє аналітикам фільтрувати та шукати зафіксовані події, а також виконувати розширений аналіз, наприклад перехресне посилання з іншими системними подіями. Ця програма надає детальну інформацію про системні виклики, включаючи створення процесу, доступ до файлів, доступ до реєстру та мережеву активність. Process Monitor дозволяє користувачам фільтрувати та шукати отримані дані, полегшуючи аналіз конкретних подій системного виклику. Він також пропонує розширені функції, такі як журнал під час завантаження та можливість зберігати отримані дані для подальшого аналізу.

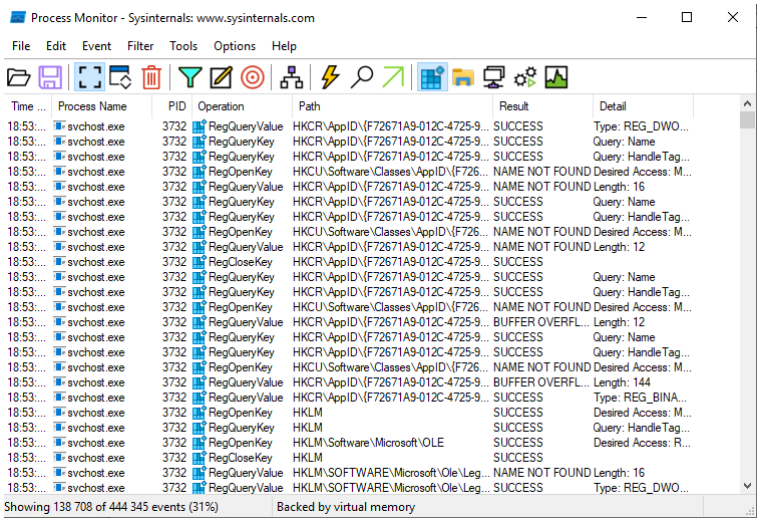

Рисунок 2.27 – Робота програми Process Monitor

SystemExplorer – це комплексний інструмент моніторингу та аналізу системи, який надає детальну інформацію про запущені процеси, служби, драйвери, модулі, мережеві підключення та системні ресурси (рисунок 2.28). Хоча він не зосереджений лише на моніторингу системних викликів, як у Process Monitor, він надає інформацію про діяльність системи, яка може допомогти проаналізувати поведінку та взаємодію процесів і модулів. SystemExplorer пропонує зручний інтерфейс із різними вкладками та видами для вивчення різних аспектів системної діяльності.

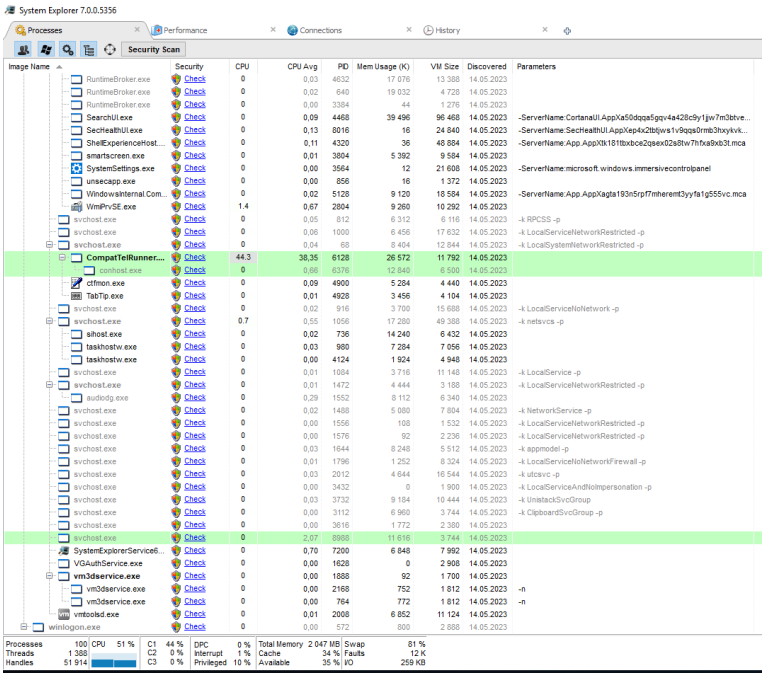

Рисунок 2.28 – Робота програми SystemExplorer

Що стосується моніторингу системних викликів, Process Monitor пропонує більш широкий і специфічний функціонал. Він фіксує широкий спектр системних подій і надає детальну інформацію про кожну подію, включаючи пов’язані процеси, часові позначки та пов’язані дані. Розширені можливості фільтрації та пошуку Process Monitor роблять його потужним інструментом для глибокого аналізу активності системних викликів. З іншого боку, SystemExplorer надає ширший огляд активності системи та пропонує розуміння різних аспектів поведінки процесів і модулів. Хоча він може не забезпечити такий самий рівень деталізації, як Process Monitor, коли йдеться про моніторинг системних викликів, він все одно може бути цінним інструментом для розуміння загальної поведінки системи та використання ресурсів.

2.2.5 Відладка (Debugging)

Відладчики— це програми, які дозволяють запускати, призупиняти, відновлювати та зупиняти програму, а також перевіряти та змінювати її пам’ять, регістри та код.

Під час відладки зразок зловмисного ПЗ виконується в ізольованому середовищі, такому як віртуальна машина або пісочниця, де його можна проаналізувати за допомогою відладчика. Відладчик може використовуватися для встановлення точки зупину в певних точках коду, наприклад перед викликом певної функції, а потім спостереження за поведінкою зловмисного програмного забезпечення, коли точка зупину досягається.

Серед відладчиків, які використовуються для динамічного аналізу зловмисного програмного забезпечення можна виділити наступні:

- OllyDbg;
- x64dbg ;
- WinDbg.

OllyDbg — це 32-розрядний відладчик аналізу на рівні асемблера для Microsoft Windows, який дозволяє користувачам аналізувати двійковий код дуже детально (рисунок 2.29). OllyDbg добре підходить для зворотного

проектування та аналізу зловмисного програмного забезпечення завдяки своїй здатності відображати та змінювати код складання, встановлювати точки зупину, відстежувати виконання програми, переглядати та редагувати пам’ять. OllyDbg також популярний завдяки своїй архітектурі плагінів, яка дозволяє користувачам розширювати його функціональність.

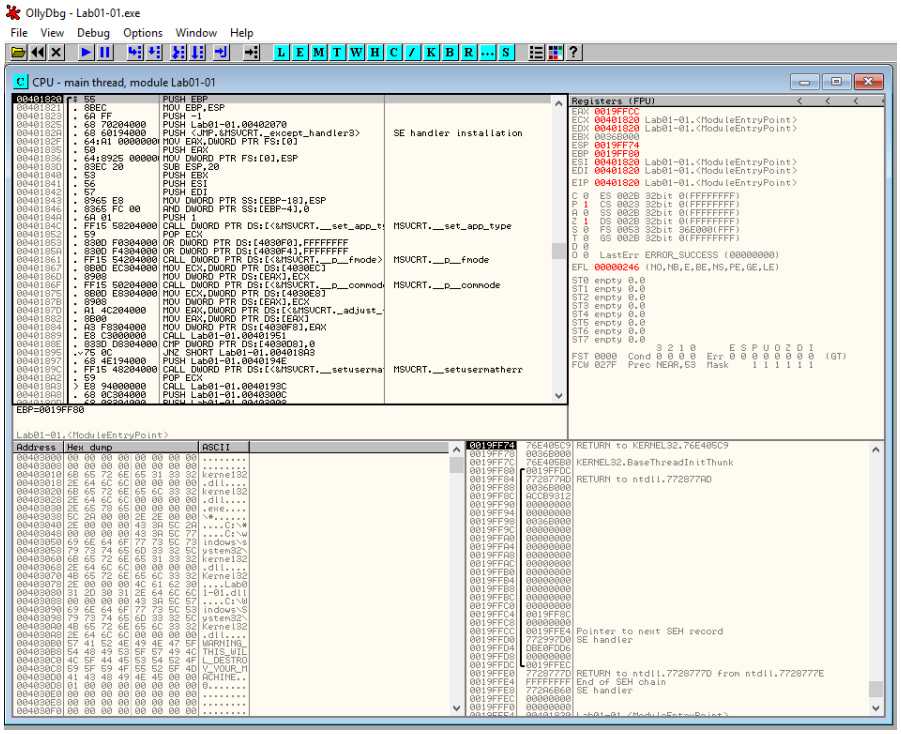

Рисунок 2.29 – Робота програми OllyDbg

x64dbg є більш сучасним інструментом налагодження, який підтримує як 32-розрядну, так і 64-розрядну архітектури (рисунок 2.30). x64dbg має відкритий код і має активну спільноту розробників, що означає, що він регулярно оновлюється новими функціями та виправляє помилки. x64dbg має багато тих самих функцій, що й OllyDbg, наприклад можливість встановлювати точки зупину, відстежувати виконання програми, переглядати та редагувати пам’ять, але він також має деякі додаткові функції, такі як графічний інтерфейс користувача, розширені можливості пошуку та вбудовану підтримка сценаріїв.

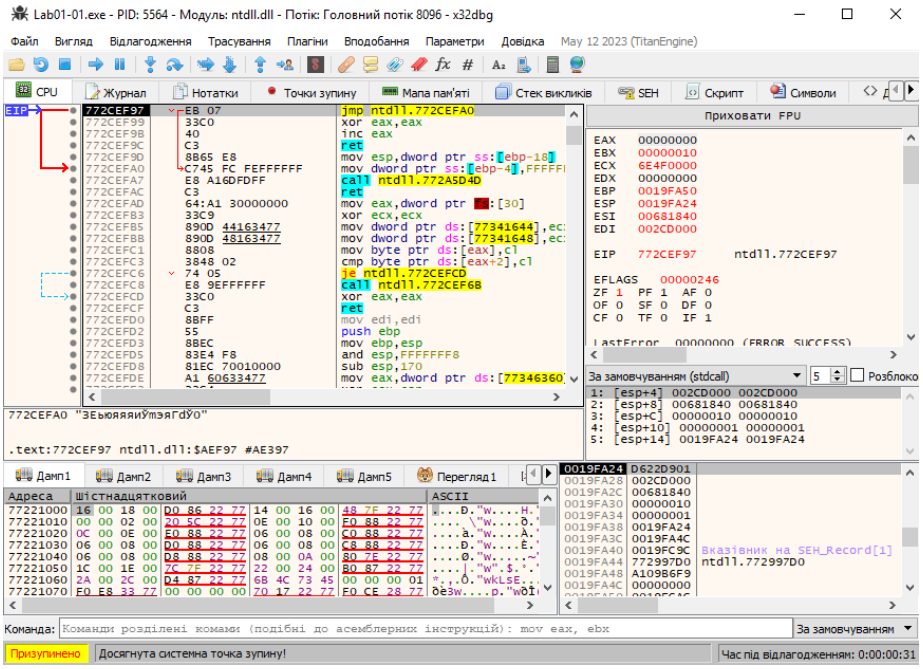

Рисунок 2.30 – Робота програми x64dbg

Тож, підсумовуючи, OllyDbg відомий своєю гнучкістю та універсальністю, що дозволяє аналітикам аналізувати код на дуже детальному рівні, однак його 32-розрядна архітектура обмежує його корисність для аналізу 64-розрядного шкідливого програмного забезпечення. З іншого боку, x64dbg підтримує як 32-бітну, так і 64-бітну архітектури, що робить його кращим для аналізу сучасного шкідливого програмного забезпечення.

2.3 Гібридний аналіз

Складне зловмисне програмне забезпечення іноді може приховуватися при виконанні в пісочниці, та простий статичний аналіз також не буде надійним при виявленні. Завдяки поєднанню статичних і динамічних методів аналізу гібридний аналіз реалізує найкраще з обох підходів – насамперед тому, що він може викривати шкідливий код, який намагається залишитися прихованим, а потім може виявляти багато інших індикаторів компрометації (IOC) за допомогою статичного та раніше невидимого коду. Гібридний аналіз допомагає виявляти невідомі загрози, навіть від найскладніших шкідливих програм.

Наприклад, одна з речей, які робить гібридний аналіз, — це застосування статичного аналізу до даних, згенерованих поведінковим аналізом, наприклад,

коли фрагмент шкідливого коду запускається та генерує деякі зміни в пам’яті. Динамічний аналіз виявить це, і аналітики отримають сповіщення про необхідність виконання базового статичного аналізу цього дампа пам’яті. У результаті буде згенеровано більше IOC і будуть виявлені експлойти нульового дня [15].

2.4 Аналіз пам’яті (Memory analysis)

Аналіз пам’яті став популярним методом, який довів свою ефективність і точність у аналізі шкідливих програм. Аналіз пам’яті приваблює аналітиків зловмисного програмного забезпечення, бо він дає всебічний аналіз зловмисного ПЗ, оскільки він здатний досліджувати програмні пастки зловмисного програмного забезпечення та код поза межами нормальної функції [7].

Аналіз пам’яті може подолати обмеження методів статичного та динамічного аналізу.

Шкідлива програма залишає певні сліди в пам'яті. За допомогою аналізу пам’яті деяку інформацію про поведінкові характеристики шкідливого програмного забезпечення можна отримати, використовуючи таку інформацію, як завершені процеси, записи Dynamic Link Library (DDL), реєстри, активні мережеві підключення та запущені служби. Робота з аналізу пам’яті складається з двох етапів: набуття пам’яті та аналіз пам’яті. Отримання пам'яті - етап отримання повного образу пам'яті машини. Аналіз пам’яті – це фаза вивчення та аналізу переміщень шкідливих програм, як правило, з використанням криміналістичного інструменту пам’яті. Таким чином стає можливим виявити приховане шкідливе програмне забезпечення за допомогою аналізу пам’яті [17].

3 ПОРІВНЯННЯ МЕТОДІВ АНАЛІЗУ ЗЛОВМИСНОГО ПЗ

3.1 Порівняння статичного, динамічного, гібридного методів аналізу та аналізу пам’яті

У порівняльній таблиці 3.1 представлена підсумовуюча інформація: описана загальна характеристика, переваги та недоліки статичного, динамічного, гібридного аналізу та аналізу пам’яті. Таблиця має на меті висвітлити ключові характеристики цих підходів до аналізу, допомагаючи дослідникам і практикам у виборі найбільш прийнятного методу для їхніх конкретних потреб. Розглядаючи сильні та слабкі сторони статичного, динамічного, гібридного аналізу та аналізу пам’яті, можна приймати обґрунтовані рішення для покращення ідентифікації та мінімізації загроз шкідливого програмного забезпечення.

Таблиця 3.1 – Порівняння методів аналізу

	Статичний аналіз	Динамічний аналіз	Гібридний аналіз	Аналіз пам'яті
Загальна характеристика	Статичний аналіз передбачає перевірку коду або двійкового файлу програми без її виконання. Він зосереджений на аналізі структури, синтаксису та властивостей програми для виявлення потенційних уразливостей або зловмисної поведінки.	Динамічний аналіз передбачає виконання програми в контрольованому середовищі та спостереження за її поведінкою в режимі реального часу. Він фіксує інформацію про час виконання, таку як системні виклики, мережевий трафік, операції з файлами та зміни пам’яті.	Гібридний аналіз поєднує в собі статичні та динамічні методи, щоб використовувати переваги обох підходів. Він передбачає використання статичного аналізу для отримання початкового розуміння структури програми та потенційних вразливостей, а також динамічного аналізу для спостереження за її поведінкою під час виконання.	Передбачає перевірку вмісту пам’яті програми під час виконання, щоб виявити зловмисну поведінку, експлойти або артефакти, залишені зловмисним програмним забезпеченням. Він зосереджений на аналізі процесів, системних бібліотек і структур даних у пам’яті.
Переваги	Швидкий, не потребує виконання та може виявляти певні типи вразливостей або ознаки компрометації. Це також корисно для виявлення типових помилок кодування та забезпечення дотримання стандартів кодування.	Динамічний аналіз дає розуміння фактичної поведінки програми, включаючи взаємодію з операційною системою та мережею. Він може розкривати приховані функції та виявляти експлойти під час виконання чи зловмисну поведінку.	Забезпечує більш повне уявлення про поведінку програми завдяки поєднанню статичних і динамічних даних. Він може виявити приховане або поліморфне шкідливе програмне забезпечення та виявити складні методи ухилення.	Може виявити поведінку та взаємодії під час виконання, які не видно лише за допомогою статичного чи динамічного аналізу. Він може виявляти руткіти, впровадження зловмисного програмного забезпечення та інші атаки на пам’ять.

Продовження таблиці 3.1

Обмеження	При статичному аналізі можуть виникнути труднощі з виявленням складних або обфускованих зловмисних програм, оскільки він не може зафіксувати поведінку під час виконання або взаємодію з середовищем.	Динамічний аналіз може пропустити певні аспекти поведінки програми, які запускаються лише за певних умов. Це може бути ресурсомістким для широкомасштабного аналізу.	Потребує додаткових ресурсів і може зайняти більше часу порівняно зі статичним або динамічним аналізом. Це вимагає досвіду як у методах статичного, так і в методах динамічного аналізів.	Для аналізу пам’яті потрібні передові технічні навички та спеціальні інструменти. Це може бути складним і трудомістким, особливо для великих дампів пам’яті. Також може знадобитися доступ до пам’яті системи або спеціалізовані методи аналізу пам’яті.

Таким чином, кожен метод аналізу має свої сильні сторони та обмеження. Статичний аналіз виконується швидко, але може не враховувати поведінку під час виконання, тоді як динамічний аналіз фіксує поведінку в реальному часі, але може упускати певні умови або складну обфускацію. Гібридний аналіз поєднує обидва підходи для більш повного уявлення, а аналіз пам’яті зосереджується саме на поведінці пам’яті під час виконання. Вибір техніки або комбінації залежить від конкретних цілей, ресурсів і досвіду, доступних для аналізу зловмисного програмного забезпечення.

3.2 Рекомендації щодо використання методів та інструментарію для моніторингу зловмисного програмного забезпечення

У таблиці 3.2 представлені рекомендації щодо використання методів статичного аналізу та пропозиції щодо розвитку програмного інструментарію, що використовується для реалізації даних методів.

Таблиця 3.2 – Рекомендації до використання методів та розвитку інструментарію статичного аналізу

Багаторазове сканування антивірусом	Даний метод краще використовувати саме для виявлення зловмисного ПЗ, аніж саме для аналізу, хоча деякі з антивірусних сканерів (наприклад, VirusTotal) не тільки ідентифікують зловмисні програми а й надають їх більш глибоку характеристику. Метод швидкий і не дуже ресурсозатратний, тому зручний як початковий етап аналізу шкідливого ПЗ. Щодо програмного забезпечення антивіруси для ПК менше підходять саме для аналізу зловмисного ПЗ, хоч і якісно виявляють його. Для покращення інструментів можливе впровадження алгоритмів машинного навчання та розширення співпраці між постачальниками антивірусних сканерів і постачальниками аналізу загроз (threat intelligence providers).

Продовження таблиці 3.2

File fingerprinting	File fingerprinting є швидким, точним та ефективним методом аналізу. Він визначає зловмисне ПЗ, але тільки те, яке вже було ідентифіковане раніше, що робить його неефективним проти загроз нульового дня. Тож цей метод теж скоріше більш доцільно використовувати для визначення зловмисного ПЗ для того, аби бути обізнаним про вже відомі характеристики і можливості шкідливої програми. Для покращення результатів при роботі з цим методом необхідне розширення і доступність баз з гешами відомого зловмисного програмного забезпечення. Щодо рекомендацій для програмного забезпечення, то в дійсності і ПЗ для створення гешів файлів і онлайн сервіси працюють справно, більш важливо впроваджувати доступні бази з широким набором гешів для пошуку зловмисного ПЗ.
Пошук по рядкам	Пошук рядків є більш ресурзозатратним та тривалим методом аналізу зловмисного ПЗ ніж попередньо описані, хоч і при правильному налаштуванні параметрів програм для цього методу все ж буде відносно швидким. Перевагою при використання пошуку по рядкам є те, що він може виявляти раніше невідомі загрози. Метод доцільно виккористовувати для пошуку шаблонів або сигнатур у двійковому коді або ідентифікаціїї конкретних рядків (наприклад, імен функцій, ключів реєстру або URL-адрес). Для інструментів для пошуку по рядкам корисно було б реалізувати можливість аналізувати цілі набори даних, а не окремі файли для підвищення ефективності роботи. Також корисно було б реалізувати алгоритми нечіткого пошуку, що можуть допомогти ідентифікувати обфусковані або змінені рядки. Ще безумовно важливим елементом є збільшення розміру і різноманітності бази даних рядків.
Перевірка інформації у PE-заголовку	Аналіз PE- заголовків може бути цінним методом для швидкого збору інформації про двійковий файл і виявлення ознак потенційно зловмисної поведінки. Він швидкий та може бути корисним в ідентифікації обфускації або упаковки. Звичайно цей метод можна використовувати лише з виконуваним файлами Windows, тому на Linux та інших Unix-подібних системах що використовують інший формат файлу, як ELF (Executable and Linkable Format), він не зможе бути проведений.

Продовження таблиці 3.2

Аналіз дизасемблерного коду	Цей метод є дуже довготривалим та ресурсомістким, часто буває неточним або неповним. Дизасемблювання може бути використане як безпечний метод (бо не потребує запуску) для деобфускації коду та виявлення його справжнього призначення. Також даний метод можна використовувати для аналізу поведінки зловмисного програмного забезпечення та визначення конкретних векторів атак і цілей, надаючи важливу інформацію для фахівців з кібербезпеки. Невід'ємною частиною для покращення інструментів для аналізу дизасемблерного коду є покращення інтерфейсу користувача для підвищення зручності та доступності. Також можна використовувати машинне навчання та штучний інтелект для навчання інструментів статичного аналізу для визначення шаблонів, сигнатур і поведінки, пов'язаної зі зловмисним програмним забезпеченням.

Загалом, використання статичного аналізу більше підходить для ідентифікації зловмисного ПЗ, аналіз інформації отриманої з PE-заголовку надає більше інформації про характеристики файлу та його вимог, а аналіз дизасемблерного коду вже може дати уявлення про дії і призначення

У таблиці 3.3 представлені рекомендації щодо використання методів динамічного аналізу та пропозиції щодо розвитку програмного інструментарію, що використовується для реалізації даних методів.

Таблиця 3.3 – Рекомендації до використання методів та розвитку інструментарію статичного аналізу

Моніторинг модифікацій файлів	Цей метод найкраще використовувати, у разі підозри, що зловмисне ПЗ вносить зміни у файли або каталоги в системі. Відстеження змін файлів менш корисне, якщо зловмисне програмне забезпечення не покладається на зміни файлової системи. Для покращення інструментів для моніторингу модифікації файлів можливе збільшення деталізації моніторингу (що вже пропонують деякі програмні продукти), розширення можливостей аналізу та звітності. Ткож можливе використання хмарної інфраструктури - тоді інструменти моніторингу змін файлів можуть використовувати масштабованість, продуктивність і резервування, які пропонують хмарні служби.

Продовження таблиці 3.3

Моніторинг змін реєстру	Подібно до моніторингу змін файлів, відстеження змін реєстру – це техніка, яка може використовуватися для відстеження та аналізу змін, внесених у системний реєстр шкідливим програмним забезпеченням. Це особливо корисно під час спостереження за поведінкою в цільовому середовищі, оскільки дає підказки щодо того, як зловмисне програмне забезпечення виконує своє призначення. Для ефективної роботи інструментів цього методу можливе: покращення деталізації моніторингу, запровадження аналізу в режимі реального часу, запровадження використання машинного навчання та штучного інтелекту.
Моніторинг мережевої активності	Під час аналізу зловмисного ПЗ, яке проявляє мережеву активність, слід використовувати моніторинг мережевої активності. Це дозволяє аналітикам не тільки виявляти комунікації з командно-контрольними серверами та викрадання даних, а й визначати шаблони, які можуть вказувати на цілеспрямовані атаки та іншу зловмисну діяльність. Для покращення інструментів цього методу можливе: машинне навчання для виявлення аномалій; інструменти мають забезпечувати можливість декодування пакетів і відображення зрозумілої людині інформації для легкого дослідження та підтримувати можливості реконструкції потоку, що передбачає реконструкцію всього потоку зв’язку між системою та кінцевими точками мережі під час певної події.
Моніторинг системних викликів	Моніторинг системних викликів є корисним методом при неможливості виконання відладки. Відстеження системних викликів може бути особливо корисним для визначення поведінки зловмисного програмного забезпечення в мережевих інтерфейсах, системних ресурсах і розділах реєстру та може надати корисну інформацію про те, до яких ресурсів має доступ зловмисне програмне забезпечення та як воно використовує ці ресурси. Для покращення інструментів цього методу можливе: вдосконалення алгоритмів для виявлення аномальної поведінки системних викликів; моніторинг у режимі реального часу; запровадження зручного та інтуїтивно зрозумілого інтерфейсу.

Продовження таблиці 3.3

Відладка (Debugging)	Цей метод корисний, при потребі перерірки поведінки коду під час виконання, наприклад, проаналізувати потік виконання конкретного зразка зловмисного програмного забезпечення або провести відладку конкретної вразливості. Відладку також можна використовувати для ідентифікації певних типів обфускації та методів усунення помилок, які використовує зловмисне програмне забезпечення. Для покращення інструментів цього методу можливе: покращення сумісності для роботи з різними ОС; інтеграція інструментів налагодження з інструментами зворотного проектування; також інструменти налагодження мають пропонувати точне керування, для поетапного перегляду виконуваного коду, встоновлення точок зупну та перевірки вмісту пам’ті.

Підсумовуючи, відладка є універсальним методом який варто використовувати при аналізі будь-якого виду зловмисного ПЗ, а інші з представлених методів більш доцільно застосовувати в залежності від ситуації та особливостей окремої шкідливої програми.

ВИСНОВКИ

1) У ході дослідження були проведені детальні аналізи методів та можливостей існуючого інструментарію для моніторингу зловмисного програмного забезпечення.
2) Основними методами аналізу є статичний, динамічний, гібридний та аналіз пам'яті, які мають достатньо інструментів, що допомагають здійснювати аналіз шкідливого ПЗ. Однак, оскільки технології безупинно розвиваються, необхідно періодично визначати можливості програмного інструментарію, його переваги та недоліки.
3) У кожного з визначених методів аналізу зловмисного ПЗ є свої переваги та недоліки, тому його вибір залежить від конкретних цілей, ресурсів і досвіду, доступних для аналізу зловмисного ПЗ.
4) Виявлення недоліків програмного забезпечення для аналізу зловмисного ПЗ може бути корисним для розробки нових підходів та може допомогти вдосконалити існуючі методи та функції цього ПЗ. Використання більш сучасного та функціонального інструментарію сприятиме якіснішому та більш ресурсозберігаючому процесу роботи спеціалістів з аналізу зловмисного програмного забезпечення.
5) Отже, дослідження методів аналізу та можливостей існуючого інструментарію для моніторингу зловмисного програмного забезпечення є важливим напрямком в інформаційній безпеці. Результати дослідження можуть бути використані для вдосконалення існуючих методів та інструментів, а також для розробки нових підходів до аналізу зловмисного програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cybersecurity statistics: the definitive list - Windows, Mac, iOS, Android. URL: https://www.antivirusguide.com/cybersecurity/cybersecurity-statistics/ (дата звернення: 29.04.2023).
2. Monappa K. A. Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware. Packt Publishing, 2018. 510 с.
3. 12 types of malware. URL: https://www.crowdstrike.com/cybersecurity-101/malware/types-of-malware/ (дата звернення: 19.04.2023).
4. What is malware and how cybercriminals use it. URL: https://www.mcafee.com/en-us/antivirus/malware (дата звернення: 19.04.2023).
5. Checklist 28: five malware distribution methods and how to protect against them. URL: https://www.securemac.com/checklist/five-malware-distribution-methods-protect (дата звернення: 19.04.2023).
6. The top 4 ways malware is spread. URL: https://www.snaptechit.com/article/the-top-4-ways-malware-is-spread-2/ (дата звернення: 19.04.2023).
7. Sihwail R., Omar K., Zainol Ariffin K. A. A survey on malware analysis techniques: static, dynamic, hybrid and memory analysis. International journal on advanced science, engineering and information technology. 2018. №4-2, т. 8. С. 1662.
8. Bhojani N. Malware analysis. Malware Analysis. 2014. P. 1-5. DOI: https://doi.org/10.13140/2.1.4750.6889.
9. Best free antivirus software (may 2023). URL: https://www.forbes.com/advisor/business/software/best-free-antivirus-software/ (дата звернення: 05.05.2023).

10. EICAR test file. URL: https://en.wikipedia.org/wiki/EICAR_test_file (дата звернення: 25.05.2023)
11. Anti malware testfile. URL: https://www.eicar.org/download-anti-malware-testfile/ (дата звернення: 25.05.2023).
12. Cross T. 5 best online virus scanners you can trust in 2023. URL: https://www.safetydetectives.com/blog/top-online-virus-scanners/ (дата звернення: 05.05.2023).
13. Sikorski M. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. San Francisco : No Starch Press, 2012. 733 с.
14. Poudyal S., Datta Gupta K., Sen S. PEFile Analysis: A Static Approach To Ransomware Analysis. The International Journal of Forensic Computer Science. 2019. №1, т. 14. С. 34–39.
15. Malware analysis: steps & examples. URL: https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/ (дата звернення: 05.05.2023).
16. Dener M., Ok G., Orman A. Malware detection using memory analysis data in big data environment. Applied sciences. 2022. №17, т. 12. С. 8604.

ДОДАТОК А

Дамп-файл EICAR-AV-TEST

```
0C32:0100 58            POP     AX
0C32:0101 354F21        XOR     AX,214F
0C32:0104 50            PUSH    AX
0C32:0105 254041        AND     AX,4140
0C32:0108 50            PUSH    AX
0C32:0109 5B            POP     BX
0C32:010A 345C          XOR     AL,5C
0C32:010C 50            PUSH    AX
0C32:010D 5A            POP     DX
0C32:010E 58            POP     AX
0C32:010F 353428        XOR     AX,2834
0C32:0112 50            PUSH    AX
0C32:0113 5E            POP     SI
0C32:0114 2937          SUB     [BX],SI
0C32:0116 43            INC     BX
0C32:0117 43            INC     BX
0C32:0118 2937          SUB     [BX],SI
0C32:011A 7D24          JGE     0140

0C32:0110                                       45 49 43 41              EICA
0C32:0120  52 2D 53 54 41 4E 44 41-52 44 2D 41 4E 54 49 56   R-STANDARD-ANTIV
0C32:0130  49 52 55 53 2D 54 45 53-54 2D 46 49 4C 45 21 24   IRUS-TEST-FILE!$

0C32:0140 48            DEC     AX
0C32:0141 2B482A        SUB     CX,[BX+SI+2A]
```

Рисунок А.1 – Дамп-файл EICAR-AV-Test