

О.В. Коваленко, А.С. Коваленко, Р.О. Ткачук, В.В. Пукалов

ПРАКТИЧНІ АСПЕКТИ РЕАГУВАННЯ НА ІТ ІНЦИДЕНТИ ТА РОЗСЛІДУВАННЯ ЦИФРОВИХ ЗАГРОЗ

Навчальний посібник



Digital Forensics
Incident Response
Malware Analysis
Reverse engineering

Кропивницький - 2026

**О.В. КОВАЛЕНКО, А.С. КОВАЛЕНКО,
Р.О. ТКАЧУК, В.В. ПУКАЛОВ**

**ПРАКТИЧНІ АСПЕКТИ РЕАГУВАННЯ
НА ІТ ІНЦИДЕНТИ ТА РОЗСЛІДУВАННЯ
ЦИФРОВИХ ЗАГРОЗ**

навчальний посібник

**Кропивницький
2026**

УДК 004.056

Рекомендовано до видання рішенням вченої ради
Центральноукраїнського національного технічного університету
(протокол № 10 від 25.05.2026 р)

Рецензенти:

Смірнов Олексій Анатолієвич, доктор технічних наук, професор, завідувач кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету.

Дресєв Олександр Миколайович, кандидат технічних наук, доцент, доцент кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету.

Самостійне електронне текстове видання

Авторський колектив: Коваленко О.В. док. техн. наук, професор, Коваленко А.С. канд. техн. наук, доцент, Ткачук Р.О. асистент, Software Engineer, Україна м. Харків «EPAM Systems», Пукалов В.В. канд. техн. наук, доцент.

Практичні аспекти реагування на ІТ інциденти та розслідування цифрових загроз: навч. посіб. / О.В. Коваленко, А.С. Коваленко, Р.О. Ткачук, В.В. Пукалов, Кропивницький: ЦНТУ, 320 с.

Навчальний посібник присвячено вивченню ключових аспектів цифрової криміналістики, реагування на інциденти, аналізу шкідливого програмного забезпечення та зворотного проєктування. Розглянуто інструменти й методи, що використовуються різними ІТ-фахівцями для виявлення, аналізу та нейтралізації ІТ-інцидентів. Посібник містить практичні завдання та приклади використання таких інструментів, як Autopsy, The Volatility Framework, KAPE, Sysinternals Utilities, Wireshark, ProcDOT, Ghidra, dnSpy, PeStudio, CFF Explorer, Process Hacker, x64dbg та багато інших. Рекомендовано для студентів закладів вищої освіти галузі знань «Інформаційні технології».

ЗМІСТ

ВСТУП.....	7
 РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ	13
1.1. АРХІТЕКТУРА ОПЕРАЦІЙНИХ СИСТЕМ СІМЕЙСТВА WINDOWS.	13
1.2. ВЗАЄМОДІЯ З ФАЙЛОВОЮ ПІДСИСТЕМОЮ	24
1.3. ПІДСИСТЕМА РЕЄСТРУ ТА АВТОЗАВАНТАЖЕННЯ ПРОГРАМ	30
1.4. ПІДСИСТЕМА ПРОЦЕСІВ ТА ПРІОРИТЕТІВ	37
1.5. ПІДСИСТЕМА СЛУЖБ	43
1.6. ФОРМАТ ВИКОНАВЧОГО ФАЙЛУ ПРОГРАМИ ОС WINDOWS (PE FORMAT).....	48
1.7. ПЛАТФОРМА .NET FRAMEWORK	54
1.8. ХЕШ-ФУНКЦІЇ MD5, SHA-1, SHA-256, SSDEEP, IMPHASH	58
1.9. ШАБЛОНИ ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ YARA	62
 РОЗДІЛ 2. ІНСТРУМЕНТАРІЙ ДЛЯ ВИРІШЕННЯ ЗАДАЧ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ.....	65
2.1. ПЗ FAR MANAGER (FILE AND ARCHIVE MANAGER).....	65

2.2. ПЗ AUTOPSY	72
2.3. ПЗ THE VOLATILITY FRAMEWORK	82
2.4. ПЗ ERIC ZIMMERMAN KAPE (Kroll Artifact Parser and Extractor)	92
2.5. ПЗ SYSINTERALS UTILITES.....	96
2.6. ПЗ WIRESHARK	115
2.7. ПЗ ProcDOT.....	117
2.8. ПЗ GHIDRA.....	125
2.9. ПЗ DNSPY	129
2.10. ПЗ PESTUDIO.....	133
2.11. ПЗ CFF EXPLORER	132
2.12. ПЗ PROCESS HACKER	136
2.13. ПЗ x64dbg	140
2.14. ОНЛАЙН-СЕРВІС VIRUSTOTAL	144
2.15. ОНЛАЙН-СЕРВІС CUCKOO SANDBOX.....	148
2.16. ПЗ Malware Information Sharing Platform (MISP).....	152
2.17. ПЗ ERIC ZIMMERMAN TOOLS.....	155
2.18. ОНЛАЙН-СЕРВІС HEXED.IT	159
2.19. ПЗ REDLINE	162
2.20. ПЗ VELOCIRAPTOR.....	165
2.21. ПЗ FTK IMAGER.....	170
2.22. ПЗ YARA GENERATOR (YARGEN).....	174

2.23. ПЗ FLARE-FLOSS (FIREEYE LABS OBFUSCATED STRING SOLVER)	179
2.24. ПЗ API MONITOR	183
2.25. ПЗ WINMERGE	187
2.26. ПЗ UPX (THE ULTIMATE PACKER FOR EXECUTABLES).....	191
2.27. ПЗ REGSHOT-ADVANCED.....	195
2.28. ПЗ SYSTEM INFORMER	199

РОЗДІЛ 3. ПРАКТИЧНІ ЗАВДАННЯ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ.....

203

3.1 Практичне завдання №1 - Моніторинг файлової підсистеми інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	203
3.2 Практичне завдання №2 - Моніторинг підсистеми реєстру інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	209
3.3 Практичне завдання №3 – Моніторинг протоколів прикладного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	215
3.4 Практичне завдання №4 – Моніторинг протоколів мережного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	221
3.5 Практичне завдання №5 – Моніторинг протоколів транспортного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	228

3.6 Практичне завдання №6 – Моніторинг параметрів запуску інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	235
3.7 Практичне завдання №7 – Моніторинг паралельних обчислень процесів інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.....	242
3.8 Практичне завдання №8 – Аналіз символічних даних двійкового коду ПЗ	250
3.9 Практичне завдання №9 – Формат бінарних виконуваних файлів, об'єктного коду та динамічних бібліотек (DLL).....	259
3.10 Практичне завдання №10 – Інструменти аналізу поведінки підсистем ОС	277
3.11 Практичне завдання №11 – Аналіз дамів пам'яті операційних систем.....	274
3.12 Практичне завдання №12 – Умови використання дизасемблерів.....	281
3.13 Практичне завдання №13 – Умови використання декомпіляторів	288
3.14 Практичне завдання №14 – Ідентифікація та класифікація зразків зловмисного ПЗ	294
3.15 Практичне завдання №15 – Аналіз структури USB накопичувача та пошук прихованих даних.....	302
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	313

ВСТУП

Цифрова криміналістика, реагування на інциденти, аналіз шкідливого програмного забезпечення, зворотна розробка є одними з найважливіших складових сучасного ІТ ринку [1-2,12-15]. З розвитком інформаційних технологій і збільшенням кількості ІТ інцидентів, знання та вміння в цих галузях стали необхідними для всіх фахівців, що займаються інформаційними системами [3-11].

Навчальний посібник присвячено вивченню ключових аспектів цифрової криміналістики, реагування на інциденти, аналізу шкідливого програмного забезпечення та зворотної розробки. У посібнику подано теоретичні відомості й практичні підходи, необхідні для виявлення, аналізу та нейтралізації сучасних ІТ-інцидентів, зокрема кіберзагроз, програмних і апаратних помилок.

Останнім часом, незважаючи на посаду ІТ спеціаліста час від часу виникають нетривіальні запитання під час виявлення аномалій у системі. Є потреба у відповіді на три основні питання під час розробки програмного забезпечення.

*«Що відбулося на проєкті?
Як це сталося?
Хто в цьому винен?»*

*© Менеджерський склад
проєкту типової ІТ компанії*

Для відповіді на ці запитання необхідно знати ряд міждисциплінарних напрямків, а саме:

– **Реагування на інциденти** (Incident Response) [1,12-15] відповідає на запитання "Що відбулося в інформаційній системі або на проєкті?". Цей напрям охоплює виявлення, оцінювання та реагування на кіберінциденти з метою мінімізації шкоди й відновлення нормальної роботи системи.;

– **Зворотне проєктування** (Reverse engineering) [13-15,57] та **аналіз шкідливих програм** (Malware Analysis) [3-9,13-15] відповідає на запитання "Як це сталося?". Reverse Engineering – відповідає за аналіз програмного забезпечення для розуміння їхньої структури, функціональності та поведінки, зазвичай без доступу до вихідного коду якщо це шкідливе програмне забезпечення Malware Analysis проводить дослідження шкідливого програмного забезпечення для ідентифікації його функцій, механізмів зараження та впливу на систему, а також для розробки методів захисту.;

– **Цифрова криміналістика** [2,43-45] (Digital Forensics) відповідає на запитання "Хто в цьому винен?". Відповідає за збір, збереження, аналіз та представлення цифрових доказів у рамках розслідувань, пов'язаних з кіберзлочинністю або іншими інцидентами ІТ фірми.

Правові та доказові аспекти цифрової криміналістики

Цифрові докази є особливою формою фактичних даних, що існують в електронному середовищі та можуть бути використані для встановлення обставин ІТ інциденту, кіберзлочину або іншої події, пов'язаної з використанням інформаційних систем. До таких доказів належать файли, журнали подій, метадані, мережеві з'єднання, дампи пам'яті, електронні повідомлення, записи активності користувачів, дані веб браузерів та інші цифрові артефакти. Їх правова природа визначається тим, що вони поєднують технічний зміст і процесуальне значення, тому потребують одночасно коректного технічного вилучення, збереження цілісності та належного документування дій фахівця [16-20].

Належність і допустимість цифрових доказів залежать від того, чи мають вони зв'язок з обставинами події та чи отримані вони у спосіб, який не порушує встановлених правових вимог [17-20, 22-32]. Для практики реагування на інциденти це означає, що технічний фахівець повинен не лише виявити артефакт, а й забезпечити можливість перевірки його походження, часу створення, способу отримання та незмінності після вилучення. Особливого значення набувають хешування, фіксація часових міток, опис середовища дослідження, ведення

журналу дій і збереження первинного носія або його криміналістичної копії [19,25,30].

Необхідно знати процедуру роботи спеціалістів, під час вилучення техніки, гаджетів та інших пристроїв електронних носіїв інформації або їх огляду потрібно враховувати, **що будь-яка некоректна дія** може змінити стан даних, часові мітки, службові записи файлової системи або інші важливі артефакти. Тому у практичній роботі доцільно застосовувати процедури мінімального втручання, створення побітових копій, перевірки контрольних хеш-сум, опису обладнання, умов доступу та послідовності виконаних дій тобто вміти створювати та працювати зі копіями робочої системи (ПК, ноутбук, тощо). Такий підхід дає змогу зберегти доказове значення цифрових даних і забезпечити можливість їх подальшого аналізу, експертної перевірки або використання у звітних матеріалах.

Комп'ютерно технічна експертиза є важливим етапом дослідження цифрових об'єктів [21,35], оскільки вона дає змогу встановити властивості носіїв, структуру даних, ознаки запуску програм, факти зміни файлів, сліди мережевої активності та інші обставини, що мають значення для розслідування. У межах такої експертизи можуть досліджуватися персональні комп'ютери, сервери, мобільні пристрої, накопичувачі, журнали подій, образи дисків, дампи пам'яті, виконувані файли та програмні артефакти. Для навчального посібника це важливо, оскільки більшість розглянутих інструментів спрямована саме на підготовку, збирання або аналіз таких об'єктів [35,39-42].

Крім того любі дані з глобальної мережі Інтернет також можуть мати доказове значення, і необхідно вміти належно її зафіксувати. До таких даних належать веб сторінки, доменні імена, IP адреси, дані DNS, електронне листування, публікації в соціальних мережах, журнали веб серверів, історія браузера, кеш, файли cookie та інші сліди мережевої активності. Під час роботи з такими джерелами потрібно враховувати динамічність веб середовища, можливість швидкого видалення або зміни даних, а також необхідність фіксації

часу, адреси ресурсу, технічних параметрів доступу та способу отримання інформації [24,30].

ОСНОВНА МЕТА НАВЧАЛЬНОГО ПОСІБНИКА – надати студентам закладів вищої освіти ґрунтовні знання та практичні навички, які допоможуть їм ефективно працювати у сфері моніторингу, аналізу та реагування на ІТ-інциденти.

Розглянемо приклади можливих напрямів ІТ інцидентів:

– Виявлення підроблених або зламаних компонентів системи що розробляється. Використання вільно розповсюджуваних сторонніх бібліотек, частин вихідного коду з сторонніх ресурсів, незрозумілого коду розробника що звільнився з можливими back door функціями чи time bomb механізмами.

– Розслідування підозрілих дій користувачів, якщо поведінка користувачів у застосунку вказує на можливі зловживання, наприклад спроби обійти валідацію форм або маніпулювання клієнтськими скриптами, розробник повинен знати, як зібрати й проаналізувати дані для визначення суті інциденту.

– Аналіз логів для виявлення підозрілої активності. Розробник може аналізувати журнали доступу до API або баз даних для виявлення підозрілих запитів, спроб SQL-ін'єкцій або інших атак.

– Реагування на інциденти, пов'язані з витоком даних коли розробник повинен знати, як зібрати та зберегти докази витоку даних із бази даних для подальшого аналізу та розслідування інциденту.

– Захист та аналіз безпеки коду у моментах коли Backend-розробник повинен розуміти, як аналізувати власний код на наявність вразливостей і як захистити серверну частину від атак, що можуть призвести до компрометації системи.

– Інтеграція та захист сторонніх компонентів. Low code розробник повинен вміти перевіряти надійність та безпеку сторонніх компонентів, які інтегруються в додаток, і знати методи перевірки наявності шкідливого коду.

– Реагування на інциденти, пов’язані з конфігурацією коли розробник повинен уміти реагувати на інциденти, спричинені неправильною конфігурацією, яка може призвести до витоку даних або зловживань.

– Розслідування підозрілої активності в додатках. Якщо виявлено підозрілу активність у додатку, розробник повинен знати, як зібрати та проаналізувати відповідні дані для виявлення джерела загрози.

– Розслідування складних атак та загроз [1,2,12]. Розробник має бути готовий до розслідування складних та багатоступеневих атак, використовуючи методи DFIR для збору доказів, аналізу інцидентів та створення звітів для менеджерського складу ІТ фірми та можливо для правоохоронних органів.

Як показано на рисунку 1, посібник складається з трьох основних розділів.

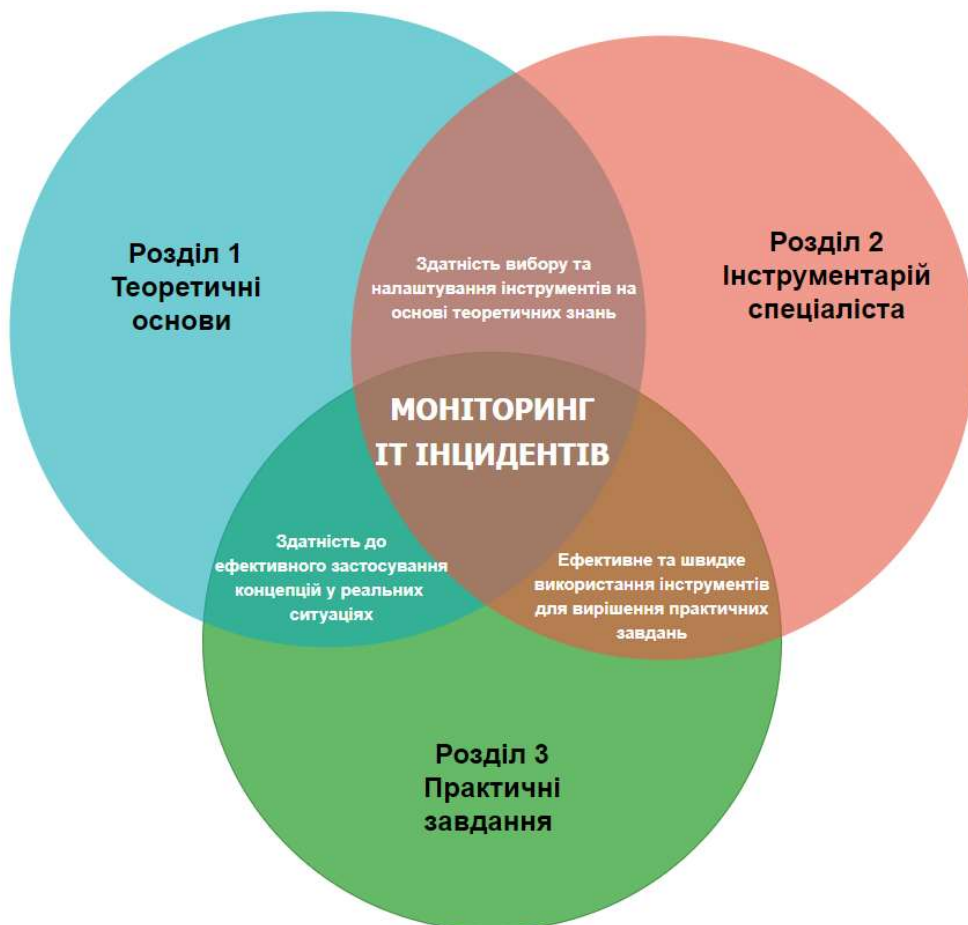


Рисунок 1 – Складові частини навчального посібника

Розділ 1. Теоретичні основи реагування на інциденти, аналізу шкідливих програм та цифрової криміналістики. Цей розділ охоплює фундаментальні поняття та концепції, пов'язані з архітектурою операційних систем, файловими системами, реєстром Windows, процесами та пріоритетами, службами Windows, форматом виконуваних файлів PE, платформою .NET Framework та хеш-функціями. Також розглядаються шаблони виявлення шкідливого програмного забезпечення YARA [57,79].

Розділ 2. Інструментарій для вирішення завдань реагування на інциденти, аналізу шкідливих програм та цифрової криміналістики. У цьому розділі детально описано різноманітні програмні засоби та утиліти, що використовуються для виявлення, аналізу та розслідування ІТ-інцидентів. Зокрема, розглядаються такі інструменти, як Autopsy, The Volatility Framework, KAPE, Wireshark, ProcDOT, Ghidra, dnSpy, PeStudio, CFF Explorer, Process Hacker, x64dbg, VirusTotal, Cuckoo Sandbox, MISP, та багато інших [70-80].

Розділ 3. Практичні завдання з реагування на інциденти, аналізу шкідливих програм та цифрової криміналістики. Розглядаються питання практичної реалізації завдань, які допоможуть закріпити теоретичні знання та набуті практичних навичок використання різних інструментів для аналізу й розслідування ІТ-інцидентів. Завдання включають моніторинг файлової підсистеми, реєстру, протоколів мережного рівня, аналіз дамів пам'яті, використання дизасемблерів та декомпіляторів, ідентифікацію та класифікацію зразків зловмисного ПЗ та багато іншого [1-2,13-15].

Навчальний посібник розроблений для студентів закладів вищої освіти галузі знань «Інформаційні технології» і допоможе їм оволодіти знаннями та практичними навичками, необхідними для захисту інформаційних систем від різноманітних ІТ-інцидентів. Посібник поєднує теоретичні й практичні аспекти, що дає змогу студентам сформуванати комплексне розуміння предмета та підготуватися до реальних викликів сучасного ринку ІТ-послуг.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ

1.1. АРХІТЕКТУРА ОПЕРАЦІЙНИХ СИСТЕМ СІМЕЙСТВА WINDOWS

Архітектура Windows – це комплексна структура, яка складається з декількох рівнів та компонентів, що взаємодіють для забезпечення стабільного, безпечного та ефективного функціонування операційної системи [46,47,73]. Вона є основою для забезпечення взаємодії між апаратним забезпеченням комп'ютера та програмами, що працюють на ньому.

Перед тим як розглядати саму структуру необхідно зрозуміти деякі вимоги, що висувуються до ОС Windows, а саме:

1. Можливість розширення. Код повинен бути написаний так, щоб системи можна було легко нарощувати і модифікувати у міру зміни потреб ринку.

2. Можливість перенесення коду. Система має функціонувати на різних апаратних архітектурах і бути придатною до відносно простого перенесення на нові платформи у разі появи такої потреби на ринку.

3. Відмовостійкість і надійність. Система має бути надійно захищена як від внутрішніх помилок, так і від зовнішніх шкідливих впливів. Програмне забезпечення не повинно створювати ризику порушення роботи операційної системи чи інших програм.

4. **СУМІСНІСТЬ.** Попри те, що Windows має розвивати наявні технології, її користувацький інтерфейс та API повинні зберігати сумісність із попередніми версіями Windows і MS-DOS. Крім того, система повинна забезпечувати можливість взаємодії з іншими платформами, такими як UNIX, OS/2 та NetWare.

5. Продуктивність. З урахуванням обмежень, що накладаються поставленими цілями, система повинна бути максимально швидкою і чуйною незалежно від апаратної платформи.

Із розглянутих вище вимог найважливіший для нас – це сумісність. Зворотна сумісність зі старими ОС це одночасно перевага та головний недолік.

Перевага – більше ПЗ, привабливість для бізнесу.

Недолік - величезна кількість проблем, пов'язаних з безпекою, стабільністю та захистом даних. Саме це слід пам'ятати при дослідженні об'єкту.

Якщо розглядати зі спрощеннями то архітектуру Windows можна побачити на рисунку 1.1 [46, 47].



Рисунок 1.1 – Спрощена схема архітектури Windows

Загалом архітектура Windows є багаторівневою та гнучкою, що дає змогу операційній системі адаптуватися до різних апаратних і програмних конфігурацій, забезпечуючи високу продуктивність, надійність і безпеку.

Архітектура Windows забезпечує управління ресурсами системи, такими як пам'ять, процесори, пристрої вводу-виводу та мережеві компоненти, а також надає розробникам програмного забезпечення інтерфейси для створення додатків, що можуть працювати в цьому середовищі.

Ядро Windows це серце операційної системи. Саме воно займається управлінням процесами, потоками, пам'яттю, обробляє апаратні переривання та планує, яка задача коли виконуватиметься. Воно тісно взаємодіє з Hardware Abstraction Layer (HAL), який приховує особливості конкретного «заліза» і дозволяє Windows нормально працювати і на Intel, і на AMD, і на різних платформах.

Крім ядра є ще підсистеми користувацького режиму. Вони включають Windows API той самий набір функцій, яким користуються всі розробники. Завдяки цим підсистемам Windows може запускати не тільки звичайні Win32-програми, а й старі додатки для POSIX чи навіть OS/2 (хоча останніх вже майже ніхто не бачив).

Окремо хочу згадати управління пам'яттю. Virtual Memory Manager дає кожному процесу свій власний віртуальний адресний простір, завдяки чому один «кривий» додаток не може завалити весь комп'ютер. А кеш-менеджер постійно тримає в оперативці найчастіше використовувані дані, щоб система працювала швидше.

Процеси та потоки є основними одиницями виконання програм у Windows. Процеси містять виконуваний код та ресурси, необхідні для виконання задач, тоді як потоки дозволяють виконувати кілька операцій одночасно в рамках одного процесу, забезпечуючи паралельне виконання.

Система вводу-виводу (I/O) в Windows координується менеджером вводу-виводу (I/O Manager), який забезпечує обробку запитів вводу-виводу між

програмами та апаратним забезпеченням. Він підтримує роботу драйверів пристроїв та файлових систем, що забезпечує ефективне управління даними.

Безпека в Windows це одна з найсильніших сторін системи. За неї відповідають кілька ключових компонентів, насамперед Security Reference Monitor (SRM) і Local Security Authority Subsystem Service (LSASS). SRM фактично вирішує, чи має той чи інший процес або користувач право доступу до певного ресурсу, а LSASS займається автентифікацією (логін-пароль, токени тощо) і перевіряє привілеї. Саме завдяки їм система захищає дані і не дає одному «киво» написаному додатку або зловмиснику легко зламати весь комп'ютер.

Щодо мережі, то Windows використовує NDIS (Network Driver Interface Specification) і Winsock (Windows Sockets). NDIS це такий стандартизований шар, який дозволяє драйверам мережних карток спілкуватися з операційною системою однаково, незалежно від виробника. А Winsock дає програмістам зручний API для роботи з мережею. Завдяки цьому Windows добре «розуміє» різні мережеві протоколи, стабільно тримає з'єднання і ефективно передає дані.

Windows використовує два режими роботи процесора

Режим користувача - ПЗ виконується в захищеному адресному просторі процесу. Код ПЗ користувача запускається в режимі користувача, а код операційної системи (наприклад, системні служби і драйвери пристроїв) запускається в режимі ядра.

Режим ядра - режим роботи процесора, в якому ПЗ надається доступ до всієї системної пам'яті і до всіх інструкціям центрального процесора. Надаючи програмному забезпеченню ОС більш високий рівень привілеїв, ніж прикладному ПЗ, процесор гарантує, що ПЗ з неправильною поведінкою не зможуть в цілому порушити стабільність роботи системи.

Основні типи процесів режиму користувача:

1. Фіксовані процеси підтримки системи (system support processes). Наприклад, процес обробки входу в систему і диспетчер сеансів, які не є сервісами Windows (тобто не запускаються диспетчером управління сервісами);

2. Процеси сервісів (service processes). Носії Windows сервісів на зразок Task Scheduler і Spooler. Багато серверні додатки Windows, наприклад Microsoft SQL Server і Microsoft Exchange Server, теж включають компоненти, що виконуються як сервіси;

3. ПЗ користувача (user applications). Бувають шести типів: для 32 розрядної Windows, 64 розрядної Windows, 16 розрядної Windows, 3.1, 16 розрядної MS-DOS, 32 розрядної POSIX та OS/2 32;

4. Підсистеми оточення (environment subsystems). Реалізована як частина середовища ОС, наданої користувачам і програмістам (Windows, POSIX і OS/2).

Windows включає наступні компоненти режиму ядра

1. **Виконавча система** (executive) Windows, що містить базові сервіси операційної системи, які забезпечують управління пам'яттю, процесами і потоками, захист, введення-виведення і взаємодія між процесами.

2. **Ядро** (kernel) Windows включає низькорівневі функції операційної системи, які забезпечують, зокрема, планування потоків, обробку переривань і виключень, а також синхронізацію під час роботи на кількох процесорах. Крім того, воно містить набір процедур і базових об'єктів, що використовуються виконавчою системою для створення структур вищого рівня.

3. **Драйвери пристроїв** (device drivers), до складу яких входять драйвери апаратних пристроїв, що транслюють користувальницькі виклики функцій вводу-виводу в запити, специфічні для конкретного пристрою, а також мережеві драйвери і драйвери файлових систем.

4. **Рівень абстрагування від обладнання** (hardware abstraction layer, HAL), забезпечує ізоляцію ядра, драйверів та виконавчої системи Windows від особливостей конкретної апаратної платформи. Завдяки цьому система не залежить від відмінностей у пристроях, наприклад, між різними типами материнських плат.

5. **Підсистема підтримки вікон і графіки** (windowing and graphics system), відповідає за реалізацію графічного інтерфейсу користувача (GUI). Вона охоплює функціональність, що належить до модулів USER і GDI, які формують

основні візуальні можливості Windows. Ці функції забезпечують підтримку вікон, елементів управління користувальницького інтерфейсу.

Основні системні файли Windows:

- **Ntoskrnl.exe** виконавча система і ядро.
- **Ntkrnlpa.exe** (тільки 32 розрядні системи) Виконавча система і ядро з підтримкою механізму Physical Address Extension (PAE), що дозволяє адресувати 64 Гб фізичної пам'яті.
- **Hal.dll** рівень абстрагування від обладнання.
- **Win32sk.sys** частина підсистеми Windows, яка у режимі ядра.
- **Ntdll.dll** внутрішні функції підтримки та інтерфейси (stubs) диспетчера системних сервісів з функціями виконавчої системи.
- **Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll** основні DLL підсистеми Windows.

Операційні системи сімейства Windows належать до операційних систем з витісняючою багатозадачністю

Розподіл процесорного часу між потоками реалізується за принципом «каруселі». **Кожному потокові послідовно надається квант часу (приблизно 20 мс), причому тривалість та порядок залежать від його пріоритету.** Після завершення кванту ОС припиняє виконання потоку й передає керування наступному.

Також потік може добровільно відмовитися від свого кванту часу. У такому разі система одразу перехоплює керування, навіть якщо квант ще не завершився, і передає його іншому потоку. Під час перемикання контексту ОС зберігає значення всіх процесорних регістрів у спеціальній структурі пам'яті – контексті потоку, що дозволяє відновити виконання з того самого місця.

Квант часу - час який виділяється кожному з потоків по черзі (за умови, що всі потоки мають однаковий пріоритет).

Windows є симетричною мультипроцесорною (symmetric multiprocessing, SMP) операційною системою (рис.1.2 [46]). У ній немає головного процесора - як операційна система, так і потоки користувача можуть бути сплановані для

роботи на будь-якому процесорі. Крім цього, всі процесори спільно використовують один і той же адресний простір.

Ця модель відрізняється від асиметричної мультипроцесорної обробки (asymmetric multiprocessing, ASMP), при якій кожній операційній системі зазвичай виділяється один процесор для виконання коду ядра операційної системи, а на інших процесорах виконується тільки користувацький код.

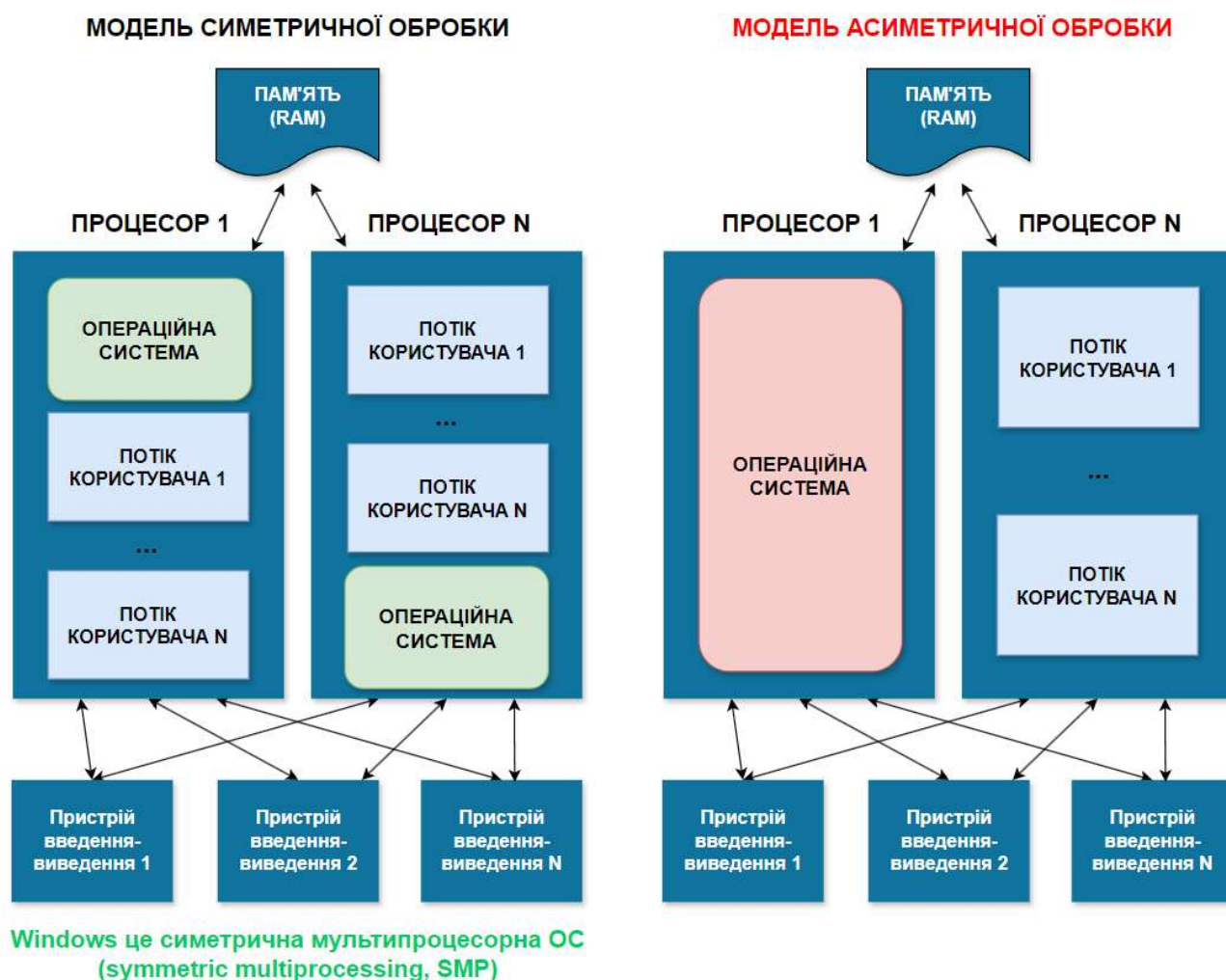


Рисунок 1.2 – Моделі взаємодії з підсистемою процесора

Підсистема Windows складається з наступних основних елементів

1. Процесу підсистеми оточення (Csrss.exe), що надає:
 - Підтримку консольних (текстових) вікон;
 - Підтримку створення та видалення процесів і потоків;

- Часткову підтримку процесів 16 розрядної віртуальної DOS машини (VDM);

- багато інших функцій, наприклад GetTempFile, DefineDosDevice, ExitWindowsEx.

2. Драйвера режиму ядра (Win32k.sys), що включає:

- Диспетчер вікон, який керує виведенням вікон на екран, приймає введення з клавіатури, миші та інших пристроїв;

- Graphics Device Interface (GDI) бібліотека, яка надає програмістам функції для роботи з пристроями графічного виведення. У складі GDI є засоби для малювання геометричних фігур, виведення тексту, роботи з растровими зображеннями та керування атрибутами графіки.

3. DLL модулів підсистем (Kernel32.dll, Advapi32.dll, User32.dll і Gdi32.dll), що транслюють виклики документованих функцій Windows API у виклики відповідних (і в більшості своїй не документованих) сервісів режиму ядра з Ntoskrnl.exe і Win32k.sys .

4. Драйверів графічних пристроїв, що становлять специфічні для конкретного обладнання драйвери графічного дисплея, принтера і мініпорт-драйверів відеоплат.

Підсистеми POSIX

POSIX, «portable operating system interface based on UNIX» (портативний інтерфейс операційної системи на основі UNIX). Це сукупність міжнародних стандартів на інтерфейси операційних систем типу UNIX. Стандарти POSIX стимулювали виробників підтримувати сумісність реалізованих ними UNIX подібних інтерфейсів, тим самим дозволяючи програмістам легко переносити свої додатки між системами.

Підсистема OS/2 (застаріла підсистема застосовується для сумісності)

Підсистема оточення OS/2, як і підсистема POSIX, володіє досить обмеженою функціональністю і підтримує лише 16 розрядні додатки OS/2 версії 1.2 з символьним або графічним введенням-виведенням (в/в). Крім того, Windows забороняє прикладним програмам прямий доступ до обладнання і тому

не підтримує програми OS/2, що використовують розширене в/в відео. Програми, що видають машинні команди CLI/STI, можуть працювати в Windows, але на час виконання команди STI всі інші програми OS/2 в системі і потоки процесів OS/2, що видають команди CLI, призупиняються.

Підсистеми оточення та DLL

При виклику програмою однією з функцій DLL підсистеми можливо наступні наслідки [47].

1. Функція повністю реалізована в режимі користувача всередині DLL підсистеми. Ніяких повідомлень процесу підсистеми оточення не посилається, і виклик сервісів виконавчої системи Windows не відбувається. Після виконання функції в режимі користувача результат повертається програмі. Прикладами таких функцій можуть служити GetCurrentProcess (завжди повертає -1, значення, визначене для посилання на поточний процес у всіх функціях, пов'язаних з процесами) і GetCurrentProcessId (ідентифікатор процесу не змінюється протягом його терміну життя, тому його можна отримати з кешу, що дозволяє уникнути перемикання в режим ядра).

2. Функція вимагає одного або більше викликів виконавчої системи Windows. Наприклад, Windows функції ReadFile і WriteFile звертаються до внутрішніх недокументованих сервісів в/в відповідно до NtReadFile і NtWriteFile.

3. Функція вимагає виконання будь яких операцій в процесі підсистеми оточення (такі процеси, що працюють в режимі користувача, відповідають за обслуговування клієнтських ПЗ, які виконуються під їх контролем). В цьому випадку підсистемі оточення видається клієнт-серверний запит через повідомлення з вимогою виконати операцію.

Ядро складається з набору функцій в Ntoskrnl.exe, що надають фундаментальні механізми (у тому числі планування потоків та синхронізації), які використовуються компонентами виконавчої системи та низькорівневими апаратно залежними засобами підтримки (диспетчерізації переривань і винятків), різними в кожній процесорній архітектурі.

Код ядра написаний в основному на С, асемблер використовували лише на вирішення специфічних завдань, важко реалізованих на С [46].

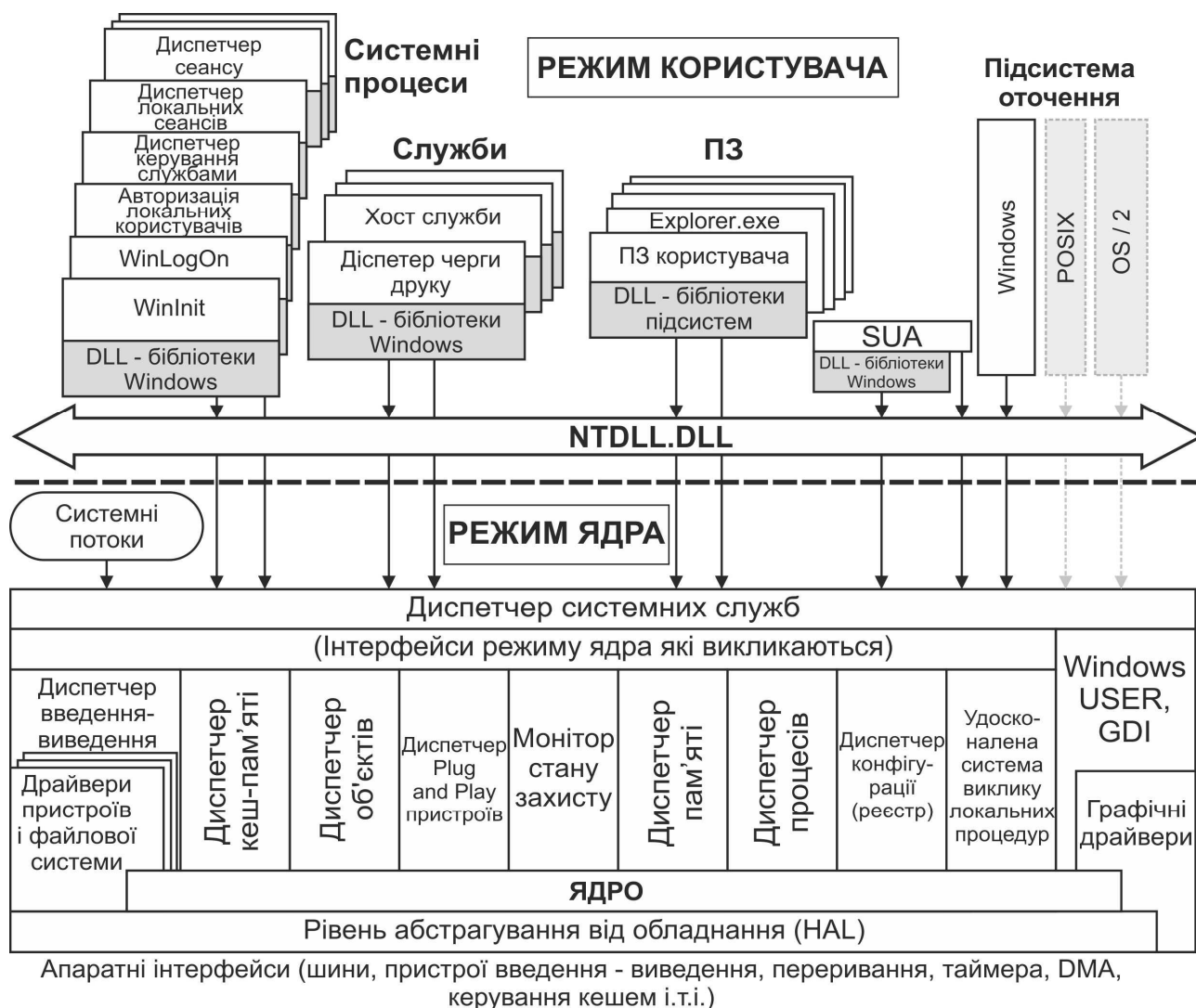


Рисунок 1.3 – Архітектурне представлення ОС Windows [46, 47]

Об'єкти ядра ОС Windows

Ядро складається з низькорівневих, чітко визначених та добре передбачуваних примітивів та механізмів операційної системи, що дозволяють компонентам виконавчої системи вищого рівня виконувати свої функції.

Перша група ядра, званих керуючими (control objects), визначає семантику управління різними функціями ОС. До цієї групи входять об'єкти

APC, DPC (deferred procedure call) та кілька об'єктів, які використовуються диспетчером введення виводу (наприклад, об'єкт переривання).

Друга група об'єктів за назвою об'єкти диспетчера (dispatcher objects) реалізує засоби синхронізації, що дозволяють змінювати планування потоків. До групи таких об'єктів входять потік ядра (kernel thread), м'ютекс (mutex), подія (event), семафор (semaphore), таймер (timer), очікуваний таймер (waitable timer) та деякі інші.

За допомогою функцій ядра виконавча система створює об'єкти ядра, маніпулює ними і конструює складніші об'єкти, що надаються в режимі користувача.

1.2. ВЗАЄМОДІЯ З ФАЙЛОВОЮ ПІДСИСТЕМОЮ

Файлова система [46-48,52-54] це організаційна структура яка використовується операційною системою для зберігання та доступу до даних на різних носіях інформації, таких як жорсткі диски, флеш-накопичувачі та інші. Вона забезпечує спосіб організації даних у вигляді файлів та директорій (папок), надає механізми для їх іменування, зберігання, читання, запису та видалення.

Файлова система також відповідає за ефективне управління місцем на диску. Вона розподіляє дані по блоках, веде індексацію файлів і постійно відстежує, скільки вільного простору залишилось. Завдяки цьому система може швидко знаходити потрібні файли і не допускати хаосу на носії.

Крім того, файлова система забезпечує захист даних. Сюди входить контроль доступу (хто і до яких файлів може заходити), шифрування інформації та механізми відновлення даних після збоїв або несподіваного вимкнення комп'ютера.

Різні файлові системи мають свої особливості. Наприклад, багато сучасних використовують журналювання (journaling) це коли всі зміни спочатку записуються в спеціальний журнал, що сильно підвищує надійність і швидкість відновлення після проблем. Також вони відрізняються підтримкою дуже великих файлів і дисків, швидкістю роботи з SSD чи жорсткими дисками тощо.

Основні поняття файлових систем

1. Файл основна одиниця зберігання даних у файловій системі. Фактично, майже все, що лежить на вашому диску (документи, фото, відео, музика, програми), зберігається у вигляді файлів.

2. Директорія (папка) – структура, яка містить посилання на файли та інші директорії, утворюючи ієрархічну структуру зберігання даних.

3. Файловий дескриптор – унікальний ідентифікатор, який використовується операційною системою для доступу до файлу.

4. Шлях (Path) – послідовність імен директорій, яка веде до конкретного файлу або директорії (наприклад C:\TEMP\file.txt).

Типи файлових систем та їхні особливості (рис. 1.4)

1. FAT (File Allocation Table). FAT12, FAT16, FAT32.

FAT одна з найстаріших файлових систем, розроблена для ранніх операційних систем Microsoft і використовується на багатьох типах носіїв. FAT12 використовувалась на дискетах, FAT16 – на старіших версіях DOS та Windows, а FAT32 підтримує більші розміри дисків і файлів.

Взаємодія FAT з операційними системами.

MS-DOS. Використовує FAT12 та FAT16 для зберігання даних на дискетах та жорстких дисках.

Windows 95/98/ME. Використовують FAT16 та FAT32 для забезпечення зберігання даних.

Windows NT/2000/XP/Vista/7/8/10/11. Підтримують FAT16 та FAT32 для сумісності з іншими пристроями та старішими версіями Windows.

Linux. Підтримує FAT32 для сумісності з іншими операційними системами та зовнішніми носіями.

macOS. Підтримує FAT32 для забезпечення сумісності з Windows та іншими пристроями, такими як USB-накопичувачі та SD-карти.

2. NTFS (New Technology File System). NTFS розроблена корпорацією Microsoft і пропонує розширені функції, такі як шифрування, контроль доступу, журналювання (journaling) та підтримка великих розмірів файлів [48,52-54]. NTFS забезпечує надійність та безпеку даних.

Взаємодія NTFS з операційними системами.

Windows NT/2000/XP/Vista/7/8/10/11 - Використовують NTFS як основну файлову систему, забезпечуючи розширені функції та надійність.

Linux – Підтримує NTFS через драйвери, такі як ntfs-3g, що дозволяє читати та записувати дані на NTFS розділах.

macOS – Підтримує NTFS для читання, але для запису потрібно використовувати додаткове програмне забезпечення, таке як Paragon NTFS або Tuxera NTFS.

FAT (FAT12, FAT16, FAT32)	Використовується у: MS-DOS, Windows 95/98/ME, Windows NT/2000/XP/Vista/7/8/10, Linux, macOS
NTFS	Використовується у: Windows NT/2000/XP/Vista/7/8/10, Linux (через ntfs-3g), macOS (тільки для читання)
ext (ext2, ext3, ext4)	Використовується у: Linux, Android
HFS+	Використовується у: macOS (до High Sierra), iOS (до версії 10.3)
APFS	Використовується у: macOS High Sierra і новіші версії, iOS 10.3 і новіші версії
exFAT (FAT64 - flash memory)	Використовується у: Windows XP і новіші версії, macOS, Linux (через exfat-utils, exfat-fuse)

Рисунок 1.4 – Типи файлових систем та операційні системи, що їх використовують

3. ext (Extended File System). ext2, ext3, ext4.

Файлова система ext була розроблена спеціально для операційних систем на основі Linux. ext2 була першою версією, ext3 додала підтримку журналювання, а ext4 є найновішою версією з покращеннями продуктивності, підтримкою великих файлів та іншими розширеними функціями.

Взаємодія ext з операційними системами.

Linux. Використовує ext2, ext3 та ext4 як основні файлові системи. ext4 є стандартною файловою системою для багатьох сучасних дистрибутивів Linux, таких як Ubuntu, Fedora та Debian.

Android. Використовує ext4 для зберігання даних на внутрішніх та зовнішніх носіях.

4. HFS+ (Hierarchical File System Plus). HFS+ була розроблена компанією Apple як сучасна заміна старої файлової системи HFS. Вона отримала підтримку дуже великих файлів, механізм журналювання (journaling), а також інші розширені функції. Завдяки цьому дані на Mac'ax і інших пристроях Apple зберігаються набагато надійніше, а робота з ними стала швидшою й ефективнішою.

Взаємодія HFS+ з операційними системами.

macOS. Використовувала HFS+ як основну файлову систему до виходу macOS High Sierra, після чого була замінена на APFS.

iOS – Використовувала HFS+ до переходу на APFS з версії iOS 10.3.

5. APFS (Apple File System).

Файлова система APFS розроблена Apple для твердотільних накопичувачів (SSD), оптимізована під швидку роботу з флеш-пам'яттю, забезпечує високу продуктивність, надійність, захисту даних. Забезпечує повноцінне шифрування, ефективне управління вільним простором, миттєві знімки (snapshots).

Взаємодія APFS з операційними системами.

macOS High Sierra і новіші версії – Використовують APFS як основну файлову систему, замінивши HFS+.

iOS 10.3 і новіші версії – Використовують APFS для зберігання даних на всіх підтримуваних пристроях Apple.

6. exFAT (Extended File Allocation Table)

exFAT була розроблена корпорацією Microsoft як сучасна альтернатива FAT32, яка усуває обмеження на розмір файлів (4 ГБ) та підтримує великі об'єми даних. Вона широко використовується на флеш-накопичувачах і SD-картах.

Взаємодія exFAT з операційними системами.

Windows XP і новіші версії. Повністю підтримують exFAT.

macOS. Підтримує exFAT, забезпечуючи сумісність з Windows і використання на флеш-накопичувачах.

Linux – Підтримує exFAT через встановлення відповідних драйверів, таких як exfat-utils і exfat-fuse.

Переваги та недоліки різних файлових систем

FAT32

Переваги: широка сумісність з різними операційними системами, простота використання.

Недоліки – обмежений розмір файлу (4 ГБ), відсутність розширених функцій, таких як контроль доступу та шифрування.

NTFS

Переваги: підтримка великих розмірів файлів, шифрування, контроль доступу, журналювання.

Недоліки: Обмежена сумісність з не-Windows операційними системами без додаткового програмного забезпечення.

ext4

Переваги: Підтримка великих розмірів файлів, журналювання, висока продуктивність на Linux.

Недоліки: Обмежена сумісність з не-Linux операційними системами.

HFS+

Переваги: Добра інтеграція з macOS, підтримка великих розмірів файлів, журналювання.

Недоліки: Обмежена сумісність з не-macOS операційними системами.

APFS

Переваги: Висока продуктивність та надійність, підтримка шифрування, оптимізована для SSD.

Недоліки: Обмежена сумісність з не-Apple операційними системами.

exFAT

Переваги: Підтримка великих розмірів файлів, сумісність з багатьма операційними системами, включаючи Windows і macOS.

Недоліки: Відсутність розширених функцій безпеки, таких як журналювання та шифрування.

1.3. ПІДСИСТЕМА РЕЄСТРУ ТА АВТОЗАВАНТАЖЕННЯ ПРОГРАМ

РЕЄСТР WINDOWS [46,47] – це централізована база даних, яка використовується операційною системою Microsoft Windows для зберігання налаштувань та параметрів конфігурації як самої ОС, так і додатків, що на ній працюють для його перегляду необхідно запустити програму regedit.exe.

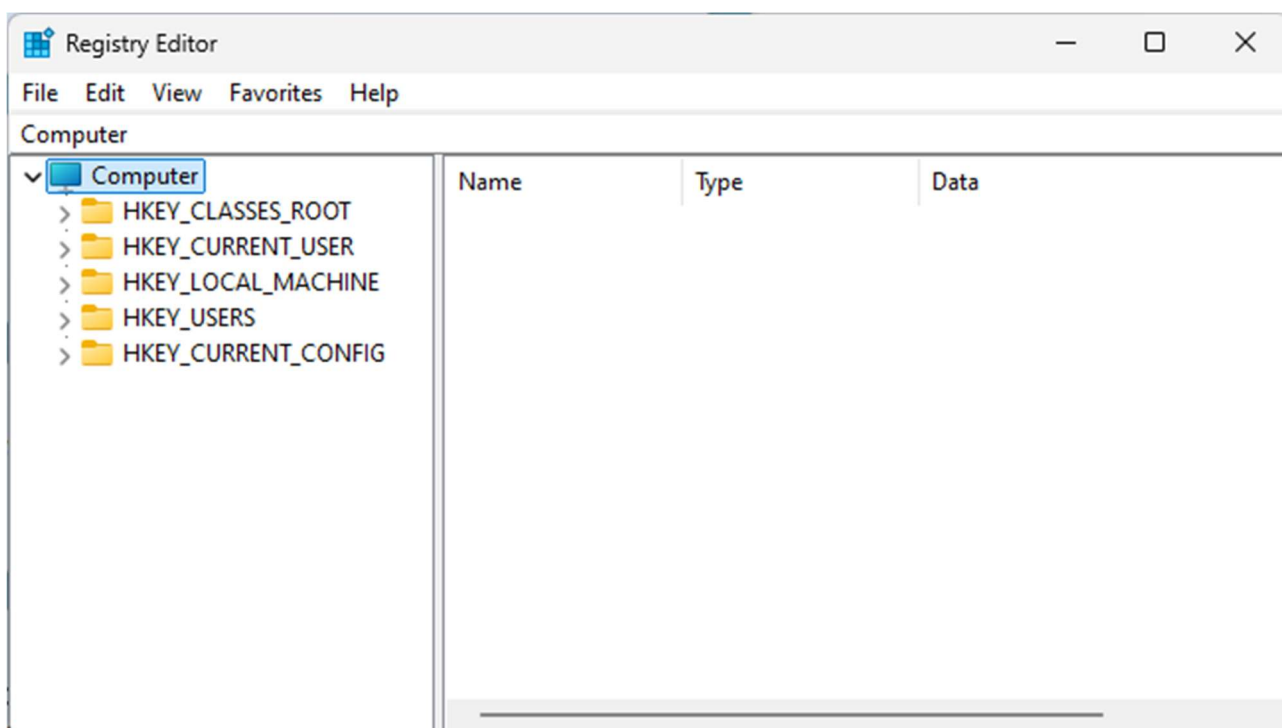


Рисунок 1.5 – Головне вікно ПЗ Regedit

Реєстр містить інформацію про апаратне забезпечення, встановлені програми, профілі користувачів, системні налаштування та інші критично важливі дані. Розуміння структури та функціонування реєстру є важливим для адміністраторів систем, розробників програмного забезпечення та фахівців з безпеки ті ін.

Реєстр Windows це один з найважливіших компонентів операційної системи. По суті, це централізоване сховище всіх конфігураційних даних:

налаштувань системи, параметрів програм, драйверів, користувацьких профілів тощо. Якщо добре розуміти, як він влаштований і як ним керувати, можна значно ефективніше налаштовувати та обслуговувати Windows, роблячи її стабільнішою і безпечнішою.

Структура реєстру

Реєстр Windows [46,47] організований у вигляді ієрархічної структури, дуже схожої на звичайну файлову систему. Він складається з кількох основних розділів, які називаються hives (вулики). Кожен такий hive містить ключі (keys) та підключі (subkeys), у яких, у свою чергу, зберігаються значення (values) з потрібними даними.

Основні розділи реєстру включають:

1. HKEY_CLASSES_ROOT (HKCR). Містить інформацію про зареєстровані типи файлів, а також асоціації файлів та COM-об'єкти.
2. HKEY_CURRENT_USER (HKCU). Зберігає налаштування та параметри конфігурації поточного користувача, що увійшов у систему.
3. HKEY_LOCAL_MACHINE (HKLM). Містить глобальні налаштування та параметри конфігурації системи, які застосовуються до всіх користувачів комп'ютера.
4. HKEY_USERS (HKU). Зберігає профілі всіх користувачів, які мають облікові записи на комп'ютері.
5. HKEY_CURRENT_CONFIG (HKCC) Містить інформацію про поточну апаратну конфігурацію системи.
6. HKEY_PERFORMANCE_DATA. Зберігає динамічні дані про продуктивність системи (тільки для читання).

Фізичне розташування основних файлів реєстру

Реєстр Windows зберігається у вигляді кількох файлів, кожен з яких відповідає певному hive реєстру.

Ці файли розташовані в різних директоріях системного диска (зазвичай це диск C:).

Нижче наведено основні файли реєстру та їх місця розташування.

1. HKEY_LOCAL_MACHINE (HKLM):

– SYSTEM. Зберігає інформацію про системну конфігурацію, драйвери та служби файл: C:\Windows\System32\config\SYSTEM.

– SOFTWARE. Містить глобальні налаштування та параметри конфігурації встановлених програм і системи файл: C:\Windows\System32\config\SOFTWARE.

– SAM. Містить базу даних з інформацією про облікові записи користувачів файл: C:\Windows\System32\config\SAM.

– SECURITY. Зберігає інформацію про політики безпеки файл: C:\Windows\System32\config\SECURITY.

– DEFAULT: Містить налаштування за замовчуванням для нових користувачів файл: C:\Windows\System32\config\DEFAULT.

2. HKEY_CURRENT_USER (HKCU). Містить налаштування та параметри конфігурації поточного користувача, що увійшов у систему файл: C:\Users\<ім'я користувача>\NTUSER.DAT.

3. HKEY_USERS (HKU):

– .DEFAULT. Містить налаштування за замовчуванням для системних облікових записів файл: C:\Windows\System32\config\DEFAULT.

– USER. Кожен користувач, який має обліковий запис у системі, має свій власний файл реєстру файл: C:\Users\<ім'я користувача>\NTUSER.DAT

4. HKEY_CLASSES_ROOT (HKCR). Містить інформацію про зареєстровані типи файлів та асоціації файлів. Цей hives фактично є частиною HKEY_LOCAL_MACHINE\Software і HKEY_CURRENT_USER\Software.

5. HKEY_CURRENT_CONFIG (HKCC). Містить інформацію про поточну апаратну конфігурацію системи. Цей hives створюється динамічно під час запуску системи.

Основні елементи реєстру

1. Ключі (Keys). Аналогічні до папок у файловій системі, ключі можуть містити підключі та значення. Кожен ключ має своє ім'я та шлях, що веде до нього від кореневого розділу.

2. Значення (Values). Кожен ключ у реєстрі може містити одне або кілька значень. Саме в цих значеннях і зберігається безпосередньо конфігураційна інформація. Кожне значення має три основні частини: ім'я, тип та дані (наприклад, рядок, число, двійкові дані тощо).

3. Типи даних:

- REG_SZ: Текстовий рядок;
- REG_DWORD: 32-бітове ціле число;
- REG_QWORD: 64-бітове ціле число;
- REG_BINARY: Двійкові дані;
- REG_MULTI_SZ: Мультистроковий текст;
- REG_EXPAND_SZ: Текстовий рядок, який може містити змінні середовища.

Управління реєстром

Для управління реєстром Windows можна використовувати різні інструменти та утиліти:

– **Registry Editor (Regedit).** Стандартний інструмент Windows для перегляду та редагування реєстру. Дозволяє створювати, змінювати та видаляти ключі та значення.

– **PowerShell.** Мова сценаріїв та командний інтерфейс Windows, що дозволяє автоматизувати багато задач, включаючи управління реєстром. Команди Get-Item, Set-Item, Remove-Item та інші використовуються для роботи з реєстром.

– **Reg.exe.** Командний рядок утиліти Windows для управління реєстром. Дозволяє виконувати операції над реєстром через командний рядок або сценарії.

Шляхи реєстру для автозавантаження програмного забезпечення

У реєстрі Windows є спеціальні ключі, які відповідають за автоматичний запуск програм під час завантаження системи або при вході користувача. Знання цих шляхів дуже допомагає контролювати автозапуск, прибирати непотрібне сміття, яке сповільнює завантаження, і оптимізувати швидкість роботи комп'ютера.

В рамках цього навчального посібника дуже важливо знати ці шляхи тому що саме їх доволі часто використовує зловмисне ПЗ [73,75-76].

Ці ключі можуть знаходитися як у розділі реєстру користувача, так і в розділі системи, що дозволяє налаштовувати автозапуск для окремих користувачів або для всіх користувачів системи.

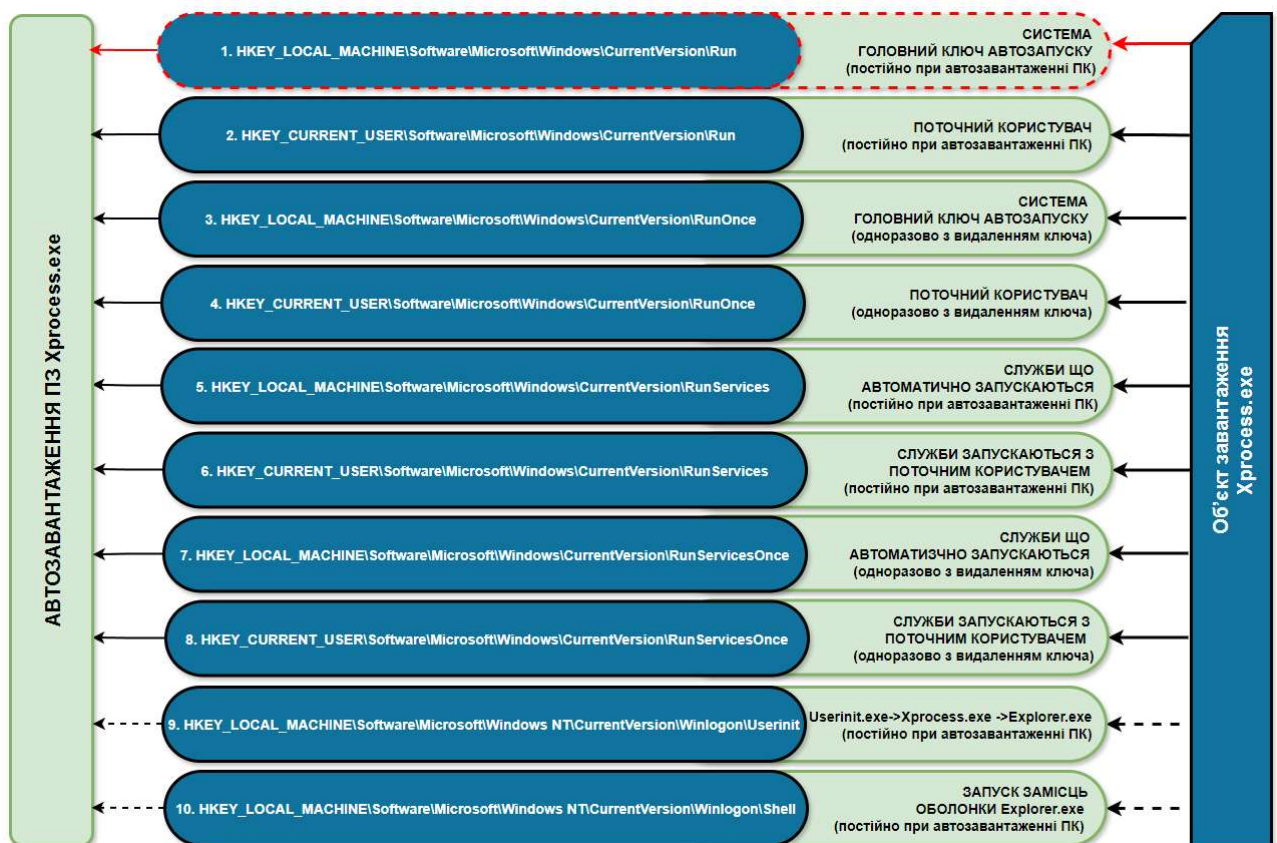


Рисунок 1.6 – Головне вікно ПЗ Regedit

Нижче наведено основні шляхи реєстру для автозавантаження програм.

1. HKEY_LOCAL_MACHINE (HKLM). Цей ключ використовується більше всіх інших - містить програми, які автоматично запускаються при завантаженні Windows для всіх користувачів системи.

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

2. HKEY_CURRENT_USER (HKCU). Цей ключ містить програми, які автоматично запускаються при вході поточного користувача в систему.

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

3. HKEY_LOCAL_MACHINE (HKLM). Цей ключ містить програми, які запускаються один раз при наступному завантаженні системи для всіх користувачів. Після запуску програма видаляється з цього ключа.

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce

4. HKEY_CURRENT_USER (HKCU). Цей ключ містить програми, які запускаються один раз при наступному вході поточного користувача в систему. Після запуску програма видаляється з цього ключа.

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

5. HKEY_LOCAL_MACHINE (HKLM). Цей ключ містить служби, які автоматично запускаються при завантаженні Windows для всіх користувачів системи.

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices

6. HKEY_CURRENT_USER (HKCU). Цей ключ містить служби, які автоматично запускаються при вході поточного користувача в систему.

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices

7. HKEY_LOCAL_MACHINE (HKLM). Цей ключ містить служби, які запускаються один раз при наступному завантаженні системи для всіх користувачів. Після запуску служба видаляється з цього ключа.

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce

8. HKEY_CURRENT_USER (HKCU). Цей ключ містить служби, які запускаються один раз при наступному вході поточного користувача в систему. Після запуску служба видаляється з цього ключа.

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce

ДОДАТКОВІ ШЛЯХИ РЕЄСТРУ ДЛЯ СПЕЦИФІЧНИХ ЗАВДАНЬ АВТОЗАПУСКУ

9. HKEY_LOCAL_MACHINE (HKLM). Цей ключ містить програми, які запускаються під час входу користувача в систему перед запуском оболонки (наприклад, Explorer.exe).

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit

10. HKEY_LOCAL_MACHINE (HKLM). Цей ключ містить оболонку, яка запускається після входу користувача в систему (за замовчуванням це Explorer.exe, але можна додати інші програми).

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

1.4. ПІДСИСТЕМА ПРОЦЕСІВ ТА ПРІОРИТЕТІВ

Процеси Windows [46,47,73] є ключовим компонентом операційної системи, яка відповідає за створення, управління та виконання процесів і потоків. Процеси є основними одиницями виконання програмного забезпечення, а потоки дозволяють виконувати кілька завдань одночасно в рамках одного процесу. Ця підсистема забезпечує ефективний розподіл ресурсів системи між процесами та керує їх взаємодією.

Пріоритети процесів у Windows відіграють дуже важливу роль у тому, як розподіляється процесорний час між різними програмами та потоками. Вони визначають, хто отримує більше ресурсів процесора, а хто може трохи почекати. Завдяки цьому система працює ефективно, не «застрягає» і швидко реагує на дії користувача.

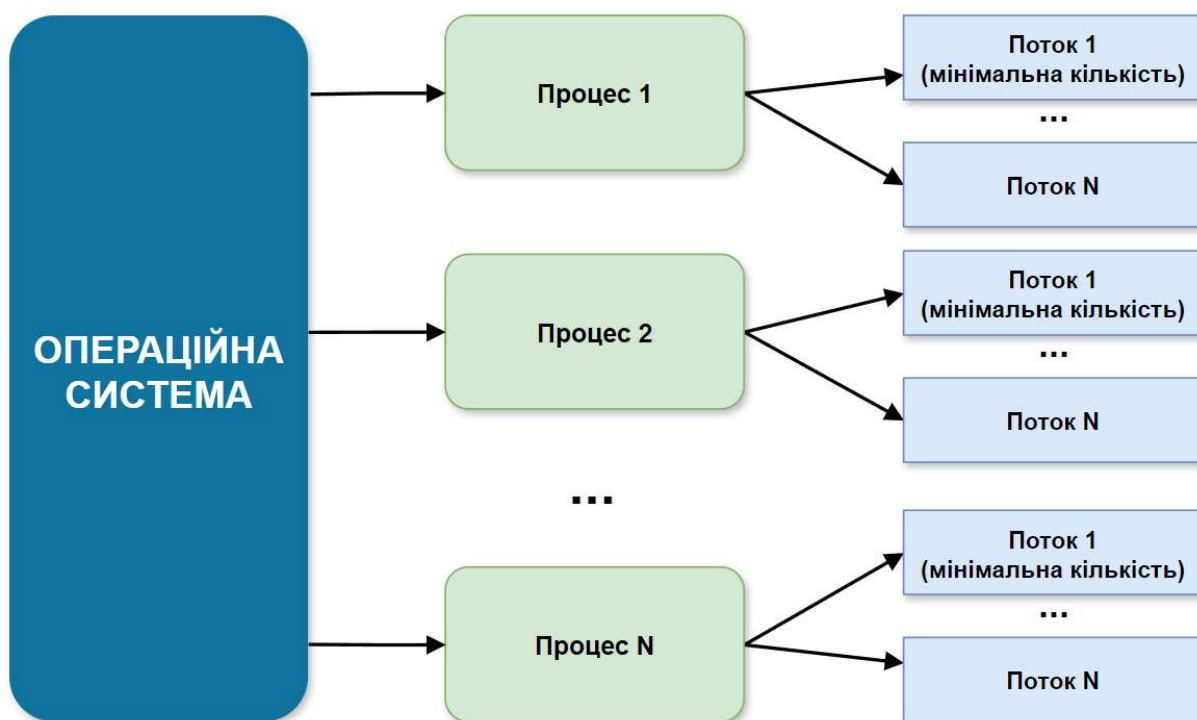


Рисунок 1.7 –Ієрархія процесів

Підсистема процесів і пріоритетів у Windows складна структура. Якщо добре її розуміти, адміністратори та розробники можуть значно краще оптимізувати роботу системи, підтримувати стабільність і швидко вирішувати проблеми з оптимізацією.

Основні поняття

1. Процес це виконувана програма, яка має власний ізольований адресний простір, набір ресурсів і дані про свій поточний стан. Кожен процес створюється ядром операційної системи і може складатися з одного або кількох потоків.

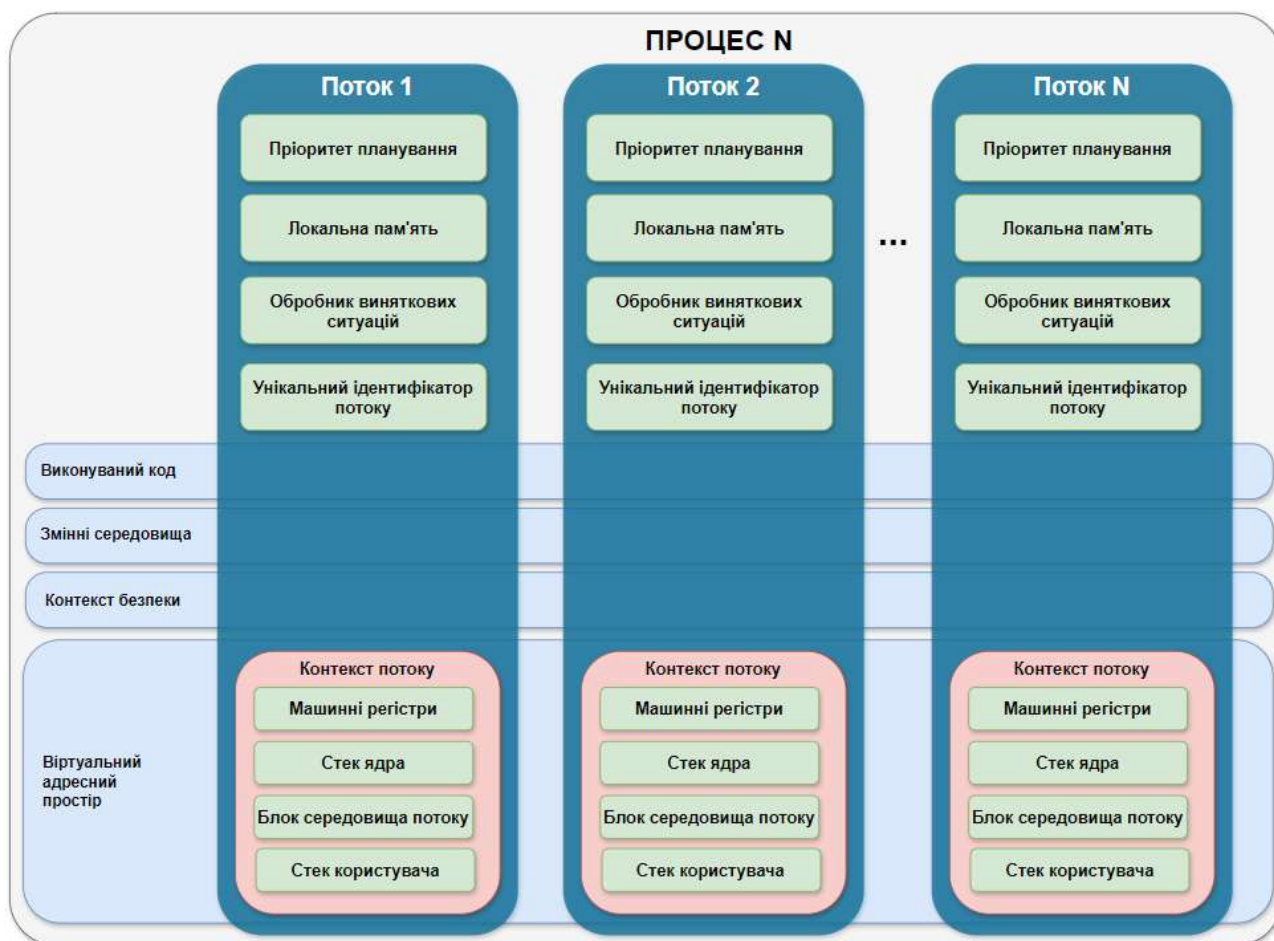


Рисунок 1.8 – Структура процесу

2. Потік це найменша одиниця виконання коду всередині процесу. У кожному процесі є хоча б один потік (основний), але їх може бути багато. Усі потоки одного процесу спільно використовують адресний простір і ресурси, що дозволяє ефективно виконувати кілька завдань одночасно.

Структура процесів

1. Процесорний контекст. Включає регістри процесора, стан викликів та інші дані, необхідні для виконання процесу.

2. Адресний простір. Кожен процес отримує свій власний віртуальний адресний простір, у якому розміщується код програми, дані та стеки. Завдяки ізоляції один процес не може випадково (або навмисно) зіпсувати пам'ять іншого це суттєво підвищує стабільність і безпеку системи.

3. Дескриптори ресурсів. Включають дескриптори відкритих файлів, сокетів, подій та інших об'єктів, якими користується процес.

4. Інформація про стан. Включає пріоритет процесу, стан виконання (готовий, працює, очікування) та інші характеристики.

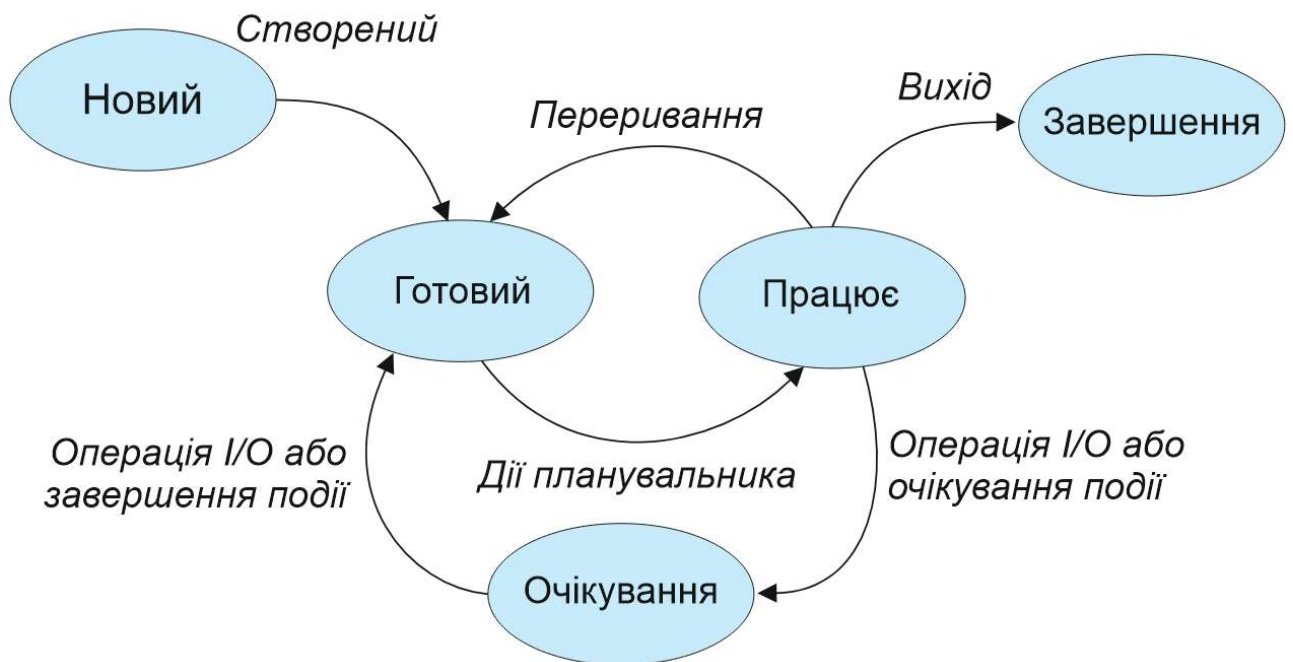


Рисунок 1.9 – Стан виконання процесу

Управління процесами

1. Нові процеси запускаються через системні функції, найчастіше через CreateProcess. При створенні система виділяє необхідні ресурси і ініціалізує контекст процесу.

2. За чергою виконання відповідає планувальник (Scheduler). Він постійно вирішує, який потік зараз має отримати процесор, враховуючи пріоритети, стан процесів та багато інших факторів.

3. Процеси та потоки часто мають взаємодіяти між собою. Для цього Windows надає **м'ютекси, семафори, події та критичні секції**, які допомагають уникати конфліктів і «гонок» даних.

4. Завершення процесів. Процеси можуть завершувати своє виконання добровільно, виконавши системний виклик ExitProcess, або бути завершені примусово операційною системою. Під час завершення процесу операційна система звільняє всі ресурси, які були йому виділені.

Рівні пріоритетів

Windows використовує багаторівневу систему пріоритетів. Усі пріоритети поділяються на дві групи в діапазоні цифрового значення пріоритетів від 1 до 31, їх розподіляють на рівня.

1. **Пріоритети від 16 до 31 реальний час (Real-Time)** використовуються для критично важливих завдань, які не можуть чекати наприклад, обробка даних від обладнання, мерева обробка пакетів даних. Високопріоритетний процес може обмежити роботу решту прикладних програм.

2. **Пріоритети від 1 до 15 звичайні (Variable)** використовуються для більшості і потоків у системі.

Пріоритети можуть бути динамічно змінені ОС залежно від активності процесу.

Класи пріоритетів

Windows визначає кілька класів пріоритетів, кожен з яких має свій діапазон значень. Основні класи пріоритетів включають:

1. Real-Time Class (**У реальному часі**). Використовується для критично важливих завдань реального часу. Діапазон: 16-31.
2. High Class (**Високий**). Використовується для завдань, які потребують високої продуктивності. Діапазон: 13-15.
3. Above Normal Class (**Вище середнього**). Використовується для завдань, які потребують вищого за середній рівня продуктивності. Діапазон: 10-12.
4. Normal Class (**Звичайний**). Використовується для звичайних користувацьких завдань. Діапазон: 8-9.
5. Below Normal Class (**Нижче середнього**). Використовується для завдань, які можуть виконуватися зниженою продуктивністю. Діапазон: 6-7.
6. Idle Class (**Низький**). Використовується для завдань, які можуть виконуватися тільки тоді, коли система не зайнята іншими завданнями. Діапазон: 1-4.

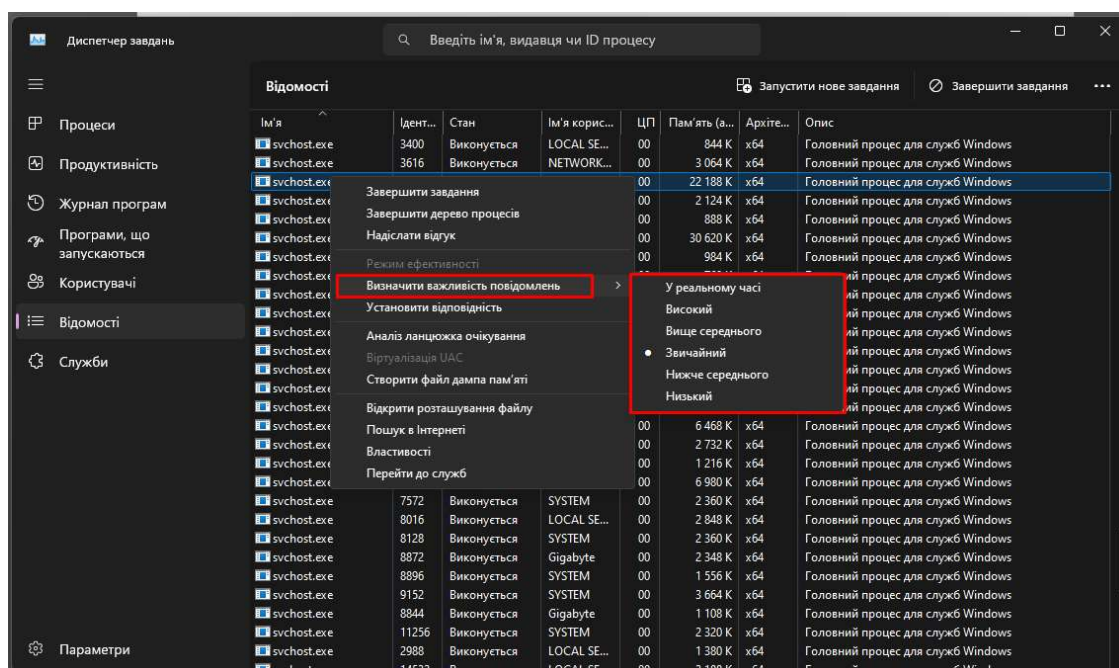


Рисунок 1.10 – Диспетчер завдань, приклад встановлення пріоритетів

Динамічне коригування пріоритетів

ОС Windows вміє динамічно змінювати пріоритети потоків залежно від їхньої поведінки. Якщо потік довго чекав на якусь подію (наприклад, на дані з диска чи мережі), система може тимчасово підвищити йому пріоритет, щоб він не «голодав».

Динамічне коригування пріоритетів включає:

1. Динамічне збільшення пріоритету відбувається автоматично ОС як приклад можна навести коли потік довго перебував у стані очікування.
2. Зниження пріоритету. Пріоритет потоків може бути знижений, якщо вони активно використовують ресурси процесора протягом тривалого часу. Це допомагає запобігти монополізації процесора одним потоком.

ІЗ моніторингу процесів та пріоритетів

1. Диспетчер задач (Task Manager). Вбудований інструмент Windows для моніторингу та управління процесами. Дозволяє користувачам переглядати список активних процесів, завершувати процеси, змінювати пріоритети та отримувати інформацію про використання ресурсів.
2. Performance Monitor. Інструмент для моніторингу продуктивності системи та процесів. Дозволяє відстежувати використання ЦП, пам'яті, дискових операцій та інших параметрів продуктивності.
3. Process Explorer. Розширений інструмент для моніторингу та діагностики процесів, розроблений Sysinternals. Надає детальну інформацію про активні процеси, бібліотеки, які вони завантажили, та їх використання ресурсів.

1.5 ПІДСИСТЕМА СЛУЖБ

Служби Windows [46,47,73] – це спеціалізовані програми, які виконують критично важливі функції у фоновому режимі без взаємодії з користувачем зазвичай вони не мають вікна взаємодії з користувачем. Вони забезпечують стабільність, безпеку та ефективність роботи операційної системи, надаючи необхідні сервіси як для самої системи, так і для прикладного програмного забезпечення. Розуміння принципів роботи служб Windows та їх налаштування є важливим для адміністраторів систем, розробників та фахівців з безпеки.

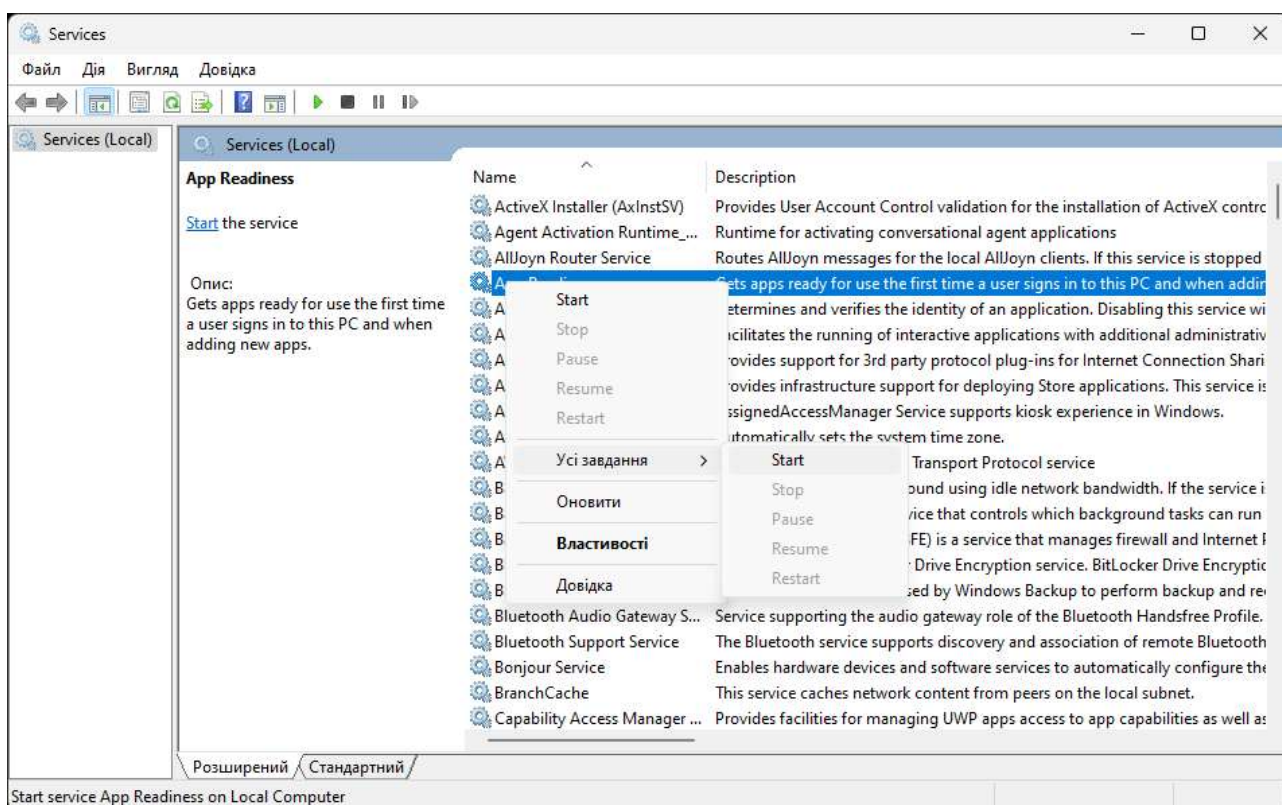


Рисунок 1.11 – Головне вікно служб

Служби Windows це фонові процеси без інтерфейсу, які забезпечують нормальну роботу системи, вони починають працювати з автентифікацією користувача в фоновому режимі.

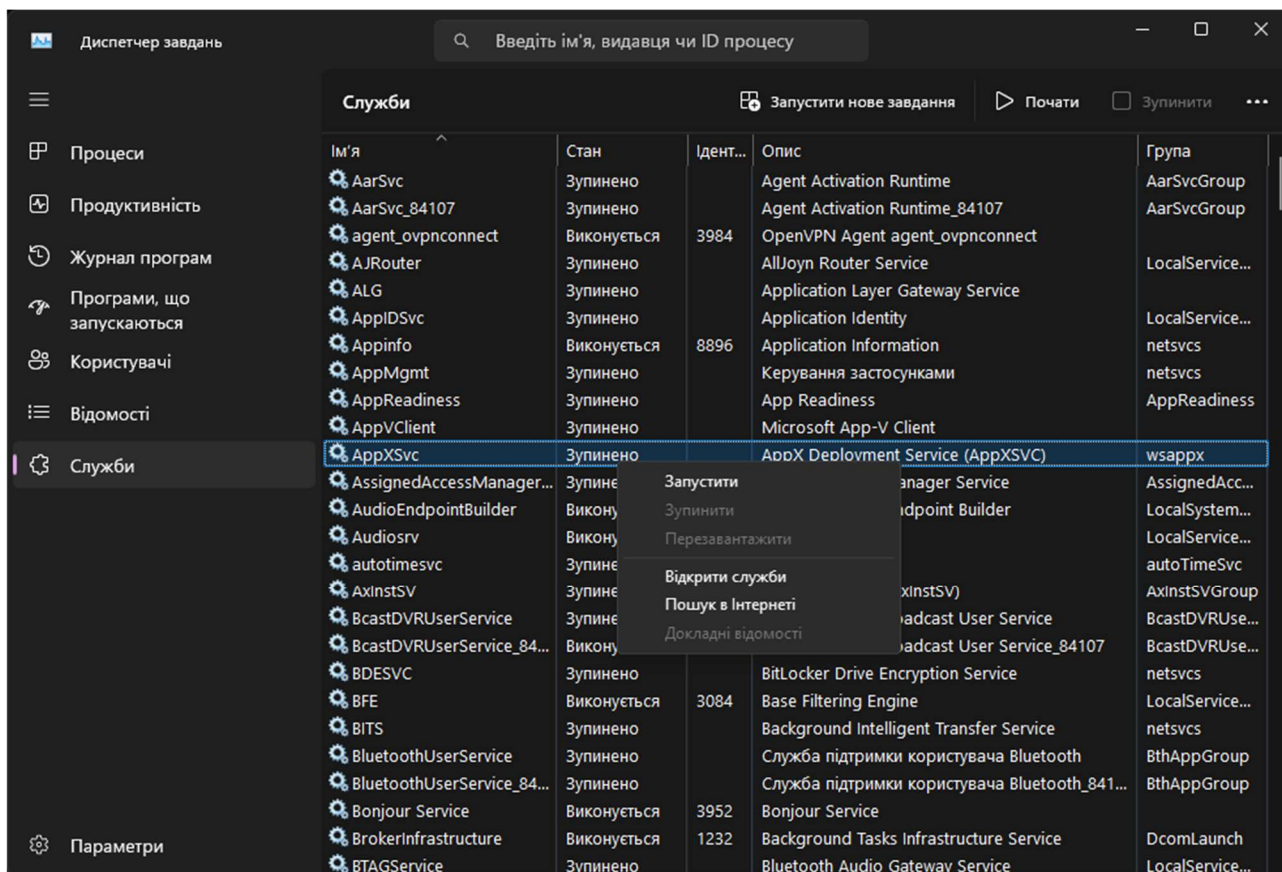


Рисунок 1.12 – Диспетчер завдань, приклад запуску служби

Основні поняття

1. Служба (Service). Програма, яка виконує завдання у фоновому режимі та зазвичай не має інтерфейсу користувача. Служби можуть бути налаштовані на автоматичний запуск при завантаженні системи, запуск вручну або у відповідь на певні події.

2. Менеджер служб (Service Control Manager, SCM). Компонент Windows, який відповідає за запуск, зупинку та управління службами. SCM контролює життєвий цикл служб та забезпечує їх взаємодію з іншими компонентами системи.

Типи служб

1. Автоматичні служби (Automatic). Служби, які запускаються автоматично при завантаженні операційної системи. Вони забезпечують

виконання важливих системних завдань та забезпечують готовність системи до роботи.

2. Ручні служби (Manual). Служби, які запускаються користувачем або іншими програмами в разі потреби. Вони не споживають системні ресурси, поки не будуть активовані.

3. Вимкнені служби (Disabled). Служби, які вимкнені та не можуть бути запущені до тих пір, поки їх стан не буде змінений адміністратором системи.

Залежності служб

Практично всі служби взаємозалежні та не можуть запускатися, поки не стартують інші служби складає ланцюг завантаження при автентифікації користувача, правильне врахування цих залежностей дозволяє уникнути проблем із порядком запуску і запобігає збоям ОС. Залежності можна налаштовувати через графічний інтерфейс Services.msc або за допомогою командного рядка sc.exe.

У ОС Windows існують критичні служби, без яких ОС не буде працювати стабільно, а саме системи:

1. Windows Update (wuauserv). Відповідає за перевірку, завантаження та встановлення оновлень для операційної системи та встановлених програм. Регулярні оновлення забезпечують безпеку та стабільність системи. Тип запуску – Автоматичний.

2. Windows Defender Antivirus Service (WinDefend) відповідає за захист від вірусів, шкідливого ПЗ та інших загроз. Виконує сканування в реальному часі та за запитом. Тип запуску – Автоматичний.

3. Print Spooler (Spooler). Управляє чергою друку, забезпечуючи можливість друку документів на підключених принтерах. Буферизує друковані документи та відправляє їх на відповідний принтер. Тип запуску – Автоматичний.

4. Remote Desktop Services (TermService). Забезпечує можливість віддаленого підключення до робочого столу комп'ютера. Використовується для адміністрування та віддаленої роботи. Тип запуску – Вручну.

5. Windows Time (W32Time). Синхронізує системний час з часом на мережевих серверах часу. Забезпечує точність системного часу, що є важливим для багатьох мережевих та безпекових протоколів. Тип запуску – Автоматичний.

6. Superfetch (SysMain) попередньо завантажує часто використовувані програми в оперативну пам'ять, завдяки чому вони запускаються помітно швидше. Тип запуску – Автоматичний.

7. Background Intelligent Transfer Service (BITS). Забезпечує передачу файлів між клієнтами та серверами у фоновому режимі, використовуючи неактивну пропускну здатність мережі. Використовується для завантаження оновлень та інших даних. Тип запуску – Вручну.

8. DHCP Client (Dhcp). Керує конфігурацією IP-адреси для комп'ютера, отримуючи її від DHCP-сервера. Забезпечує автоматичне налаштування мережевих параметрів. Тип запуску – Автоматичний.

9. DNS Client (Dnscache). Дозволяє комп'ютеру виконувати DNS-запити, кешує DNS-ім'я для прискорення наступних запитів. Забезпечує правильну роботу мережевих додатків, які використовують DNS. Тип запуску – Автоматичний.

10. Windows Event Log (EventLog). Забезпечує реєстрацію подій системи, додатків та безпеки. Дозволяє відстежувати різноманітні події, що відбуваються в системі, для моніторингу та діагностики. Тип запуску: – Автоматичний.

11. Cryptographic Services (CryptSvc). Забезпечує три керовані підсистеми: Каталог служб, що забезпечує перевірку підписів файлів Windows; послуги безпеки для управління ключами та сертифікатами, і шифрування файлів. Тип запуску – Автоматичний.

12. Windows Management Instrumentation (Winmgmt). Забезпечує інтерфейс і модель об'єктів для управління та моніторингу системи. Використовується для управління конфігурацією системи як локально, так і віддалено. Тип запуску – Автоматичний.

Інструменти для управління службами

1. Services.msc. Графічний інтерфейс для управління службами, який дозволяє переглядати список служб, змінювати їхній стан, налаштовувати параметри запуску та переглядати залежності між службами.
2. Task Manager: Менеджер завдань, який також дозволяє переглядати та керувати службами, що працюють у системі.
3. Для управління службами зручно використовувати PowerShell. Командами Get-Service, Start-Service, Stop-Service, Set-Service можна швидко отримувати інформацію, запускати, зупиняти та змінювати параметри служб.

1.6 ФОРМАТ ВИКОНАВЧОГО ФАЙЛУ ПРОГРАМИ ОС WINDOWS (PE FORMAT)

Формат файлів Portable Executable (PE) [57,58,78] є основним форматом виконуваних файлів у операційних системах сімейства Windows. Він використовується для зберігання виконуваних програм, динамічних бібліотек (DLL) та інших типів коду, який може бути завантажений і виконаний операційною системою. Формат PE базується на старішому форматі COFF (Common Object File Format) і забезпечує ефективну організацію даних для виконання коду.

Це структура даних, яка повідомляє завантажувачу ОС Windows, яка інформація потрібна для керування загорнутим виконуваним кодом. Це включає динамічні бібліотечні посилання для зв'язування, експорт API, імпорт таблиць, дані керування ресурсами та дані TLS.

Структури даних PE включають заголовок DOS, заглушку DOS, заголовок файлу PE, додатковий заголовок зображення, таблицю розділів, словники даних і розділи [58,78]. Це складна енциклопедична структура для дослідження дивись одну з нижченаведених посилань:

PE Format (офіційне джерело microsoft)

<https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

A Comprehensive Guide To PE Structure

<https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/>

What is PE ?

<https://onlyf8.com/pe-format>

PE Format Illustrated – Part 1

<https://www.codeproject.com/Articles/5356568/PE-Format-Illustrated-Part-1>

Блокова структура PE-файлу

1. Заголовок DOS (DOS Header + DOS Dtab)

Перші байти PE-файлу містять заголовок DOS, який включає невеликий DOS-завантажувальний програмний код та структуру IMAGE_DOS_HEADER. Цей заголовок також містить офсет до NT-заголовка (NT Header), що починається з магічного числа "PE\0\0".

2. NT-заголовок (NT Header)

NT-заголовок починається з сигнатури "PE\0\0" і включає два основні компоненти:

- FILE_HEADER (COFF Header). Містить загальну інформацію про файл, таку як тип машини, кількість секцій, часову мітку та розмір заголовків.
- OPTIONAL_HEADER. Містить додаткову інформацію про виконуваний файл, таку як точки входу, розміри стеку та купи, а також адреси секцій.

3. Табличні секції (Section Table)

PE-файл (виконуваний файл Windows) поділений на секції. Кожна секція містить певний тип даних: код, дані, таблицю імпорту, експорту тощо. У кожній секції є свій заголовок, де вказується розмір, розташування та характеристики.

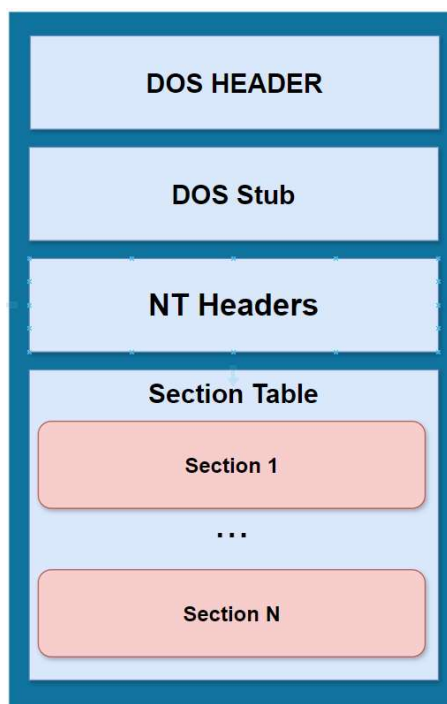


Рисунок 1.13 – Типова структура PE-файлу

PE Format

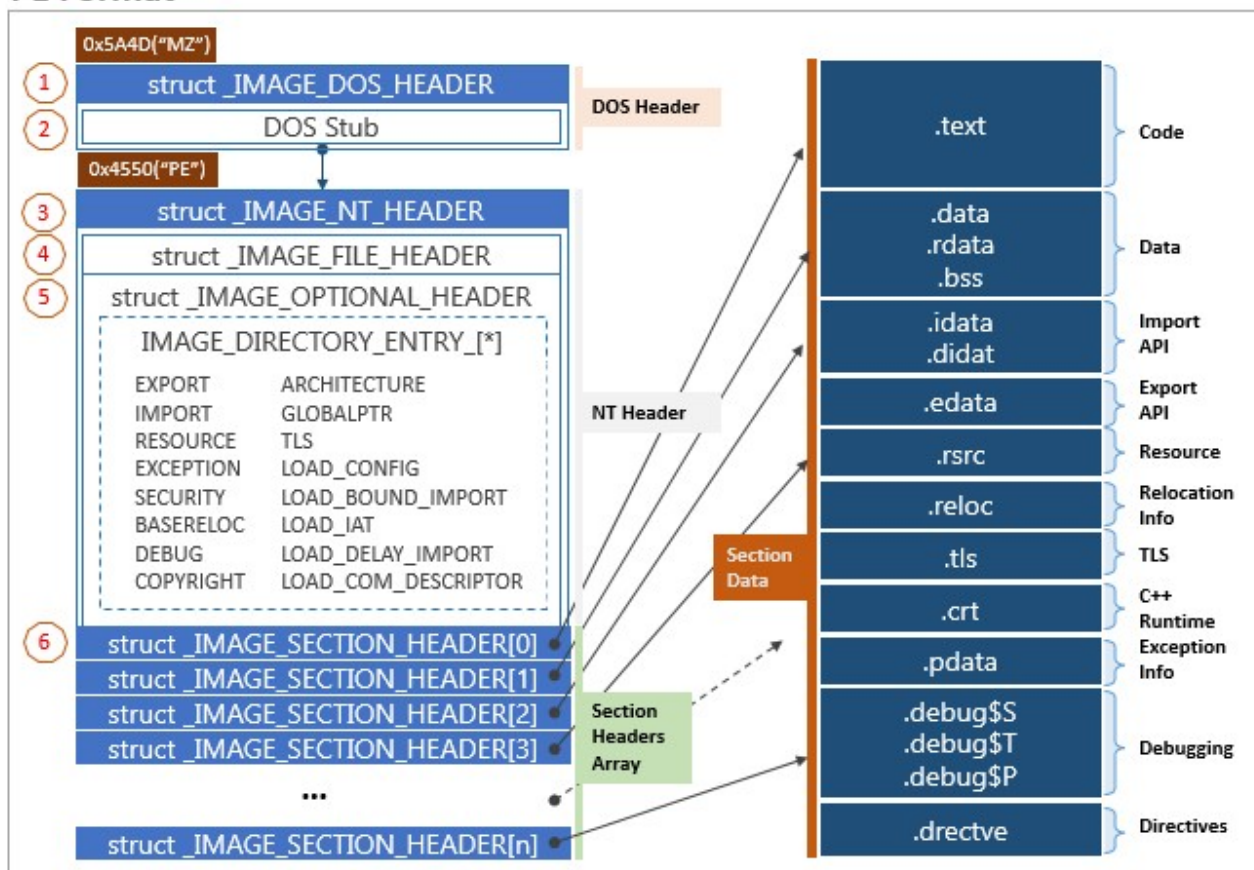


Рисунок 1.14 – Portable Executable FILE Format (<https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/>)

Стандартні секції PE-файлу які можна побачити у більшості «EXE» файлах наступні (наведені вище як Section 1 – Section N):

1. «.text» містить машинний код програми. Це основна секція, яка виконується процесором.
2. «.data» містить ініціалізовані дані, доступні програмі під час виконання.
3. «.bss» містить неініціалізовані дані, які ініціалізуються нулями під час завантаження програми.
4. «.rdata» містить дані лише для читання, такі як константи та строки.
5. «.edata» містить таблиці експорту, які використовуються для експорту функцій та змінних з динамічних бібліотек (DLL).

6. «.idata» містить таблиці імпорту, які використовуються для імпорту функцій та змінних з інших модулів.

7. «.reloc» містить інформацію про переадресацію, яка використовується, якщо базова адреса завантаження програми змінюється.

Структури даних

На основі вищенаведеного рисунку представимо вихідний код формату файлу:

1. IMAGE_DOS_HEADER (рис. 1.13). Містить інформацію про DOS-заголовок, включаючи офсет до NT-заголовка.

```
typedef struct _IMAGE_DOS_HEADER {  
WORD    e_magic;        // Магічне число DOS (4D5Ah)  
WORD    e_cblp;         // Кількість байтів у останньому сторінковому блоці  
WORD    e_cp;           // Кількість сторінок у файлі  
WORD    e_crlc;         // Кількість переліків  
WORD    e_cparhdr;      // Кількість параграфів у заголовку  
WORD    e_minalloc;     // Мінімальна кількість параграфів, що виділяються  
WORD    e_maxalloc;     // Максимальна кількість параграфів, що виділяються  
WORD    e_ss;           // Значення SS-регістру після ініціалізації  
WORD    e_sp;           // Значення SP-регістру після ініціалізації  
WORD    e_csum;         // Контрольна сума  
WORD    e_ip;           // Значення IP-регістру після ініціалізації  
WORD    e_cs;           // Значення CS-регістру після ініціалізації  
WORD    e_lfarlc;       // Офсет таблиці переліків  
WORD    e_ovno;         // Номер оверлею  
WORD    e_res[4];       // Зарезервовані слова  
WORD    e_oemid;        // OEM-ідентифікатор (для e_oeminfo)  
WORD    e_oeminfo;      // OEM-інформація; e_oemid специфічна  
WORD    e_res2[10];     // Зарезервовані слова  
LONG    e_lfanew;       // Офсет до NT-заголовка  
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

2. IMAGE_NT_HEADERS (рис. 1.13). Містить NT-заголовок, який складається з сигнатури, заголовка файлу (IMAGE_FILE_HEADER) та додаткового заголовка (IMAGE_OPTIONAL_HEADER).

```
typedef struct _IMAGE_NT_HEADERS {  
    DWORD Signature;      // Підпис "PE\0\0"  
    IMAGE_FILE_HEADER FileHeader;  
    IMAGE_OPTIONAL_HEADER OptionalHeader;  
} IMAGE_NT_HEADERS, *PIMAGE_NT_HEADERS;
```


3. IMAGE_FILE_HEADER (COFF Header) (рис. 1.13). Містить загальну інформацію про файл.

```
typedef struct _IMAGE_FILE_HEADER {  
WORD Machine;           // Тип машини (наприклад, x86 або x64)  
WORD NumberOfSections;  // Кількість секцій  
DWORD TimeDateStamp;     // Часова мітка створення файлу  
DWORD PointerToSymbolTable; // Вказівник на таблицю символів  
DWORD NumberOfSymbols;   // Кількість символів  
WORD SizeOfOptionalHeader; // Розмір додаткового заголовка  
WORD Characteristics;    // Характеристики файлу  
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

4. IMAGE_OPTIONAL_HEADER (рис. 1.13). Містить додаткову інформацію про виконуваний файл.

```
typedef struct _IMAGE_OPTIONAL_HEADER {  
WORD Magic;              // Тип файлу (PE32 або PE32+)  
BYTE MajorLinkerVersion; // Старша версія компоновника  
BYTE MinorLinkerVersion; // Молодша версія компоновника  
DWORD SizeOfCode;        // Розмір секції коду  
DWORD SizeOfInitializedData; // Розмір ініціалізованих даних  
DWORD SizeOfUninitializedData; // Розмір неініціалізованих даних  
DWORD AddressOfEntryPoint; // Точка входу в програму  
DWORD BaseOfCode;         // Базова адреса секції коду  
DWORD BaseOfData;         // Базова адреса секції даних (тільки для PE32)  
DWORD ImageBase;          // Базова адреса завантаження файлу  
DWORD SectionAlignment;   // Вирівнювання секцій в пам'яті  
DWORD FileAlignment;      // Вирівнювання секцій у файлі  
WORD MajorOperatingSystemVersion; // Старша версія ОС  
WORD MinorOperatingSystemVersion; // Молодша версія ОС  
WORD MajorImageVersion;    // Старша версія файлу  
WORD MinorImageVersion;    // Молодша версія файлу  
WORD MajorSubsystemVersion; // Старша версія підсистеми  
WORD MinorSubsystemVersion; // Молодша версія підсистеми  
DWORD Win32VersionValue;   // Зарезервовано, має бути нульовим  
DWORD SizeOfImage;         // Розмір образу в пам'яті  
DWORD SizeOfHeaders;       // Розмір заголовків  
DWORD CheckSum;            // Контрольна сума  
WORD Subsystem;            // Тип підсистеми (наприклад, GUI, консоль)  
WORD DllCharacteristics;    // Характеристики DLL  
DWORD SizeOfStackReserve;   // Розмір резерву стеку  
DWORD SizeOfStackCommit;   // Розмір стеку, виділеного відразу  
DWORD SizeOfHeapReserve;   // Розмір резерву купи  
DWORD SizeOfHeapCommit;    // Розмір купи, виділеної відразу  
DWORD LoaderFlags;         // Прапори завантажувача (зазвичай нульові)  
DWORD NumberOfRvaAndSizes; // Кількість елементів у масиві даних  
// Масив директорій даних  
IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];  
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

5. IMAGE_SECTION_HEADER (рис. 1.13). Описує кожну секцію PE-файлу.

```
typedef struct _IMAGE_SECTION_HEADER {
BYTE   Name[IMAGE_SIZEOF_SHORT_NAME]; // Назва секції
union {
DWORD  PhysicalAddress;
DWORD  VirtualSize;
        } Misc;
DWORD  VirtualAddress;
// Віртуальна адреса секції в пам'яті
DWORD  SizeOfRawData;
// Розмір секції у файлі
DWORD  PointerToRawData;
// Вказівник на початок секції у файлі
DWORD  PointerToRelocations;
// Вказівник на перелоки (не використовується у PE)
DWORD  PointerToLinenumbers;
// Вказівник на номери рядків (не використовується у PE)
WORD   NumberOfRelocations;
// Кількість переадресація (не використовується у PE)
WORD   NumberOfLinenumbers;
// Кількість номерів рядків (не використовується у PE)
DWORD  Characteristics;
// Характеристики секції
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

1.7. ПЛАТФОРМА .NET FRAMEWORK

Платформа .NET [57,58], розроблена компанією Microsoft, є потужним і універсальним середовищем для розробки різноманітних додатків – від настільних до веб-додатків, а також мобільних і хмарних сервісів. .NET забезпечує єдину платформу, яка підтримує кілька мов програмування, має багатий набір бібліотек та інструментів для розробки, а також забезпечує високий рівень продуктивності, безпеки та надійності.

Платформа .NET це потужний і сучасний інструмент для розробки. Завдяки модульності, кросплатформеності та сумісності вона дозволяє створювати швидкі, надійні та безпечні додатки. Розуміння її компонентів, мов і інструментів значно полегшує життя розробнику.

Основні компоненти .NET

1. Common Language Runtime (CLR). Основне середовище виконання для всіх додатків на платформі .NET. CLR відповідає за управління виконанням програм, автоматичне управління пам'яттю (Garbage Collection), безпеку, обробку виключень, а також забезпечення міжмовної сумісності.

2. Бібліотеки класів (Base Class Library, BCL). Набір бібліотек, що забезпечують доступ до основних функцій операційної системи та реалізують стандартні можливості, такі як введення/виведення, робота з файлами, робота з мережами, шифрування, доступ до баз даних і багато іншого.

3. .NET Core (тепер просто частина .NET) — це кросплатформена версія, яка працює на Windows, macOS і Linux. Вона має повністю відкритий код і активно розвивається спільнотою.

4. ASP.NET. Набір інструментів і бібліотек для розробки веб-додатків та веб-сервісів. ASP.NET підтримує створення динамічних веб-сторінок, RESTful сервісів, а також забезпечує високу продуктивність і безпеку.

Архітектура .NET

1. Підтримує модульний підхід до розробки додатків, що дозволяє розробникам легко додавати та видаляти функціональність за допомогою пакетів NuGet.

2. Забезпечує сумісність з попередніми версіями, що дозволяє розробникам безпечно оновлювати свої додатки та використовувати нові можливості платформи.

3. Завдяки .NET Core і Xamarin, платформа .NET дозволяє створювати додатки, що працюють на різних операційних системах, що значно розширює можливості розробників.

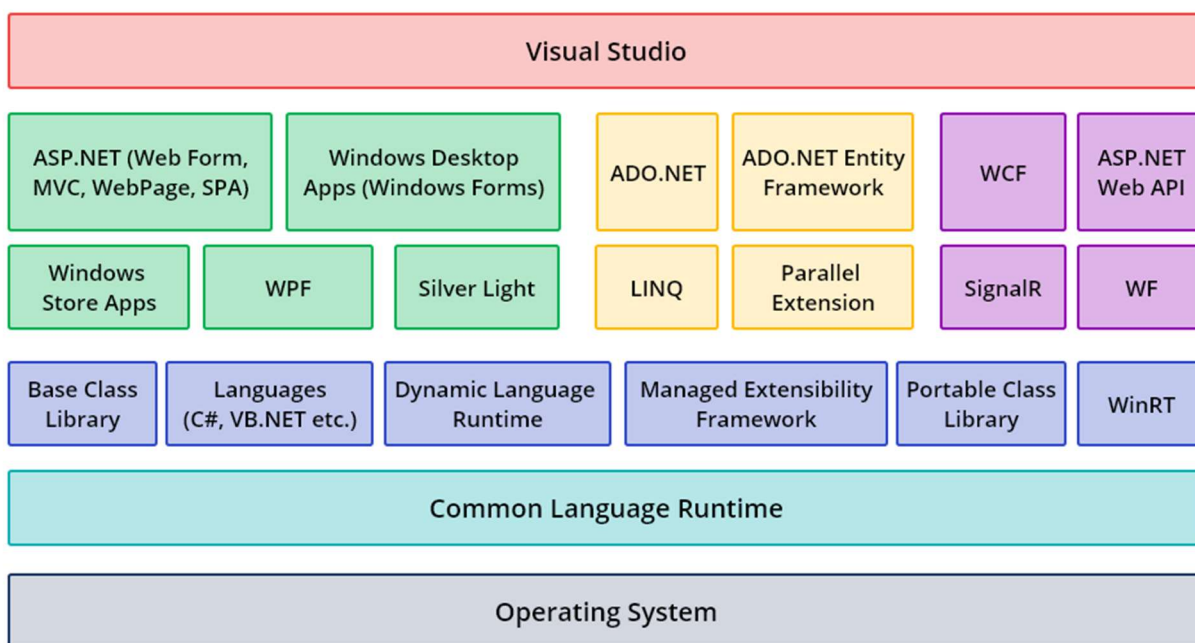


Рисунок 1.15 – Архітектура .NET FRAMEWORK (<https://celestialsys.com/blogs/an-introduction-to-net-framework/>)

Використання платформи .NET для реверс-інжинірингу

Аналіз .NET-програм набагато простіший, ніж нативних програм. Однак розробники можуть захищатися через обфускацію та шифрування.

Основні особливості .NET для реверс-інженера:

1. Керований код (Managed Code) компілюється в Intermediate Language (IL), а потім JIT-компілятор перетворює його в машинний код уже під час виконання.
2. Метадані містять детальний опис типів, методів, властивостей. Це дозволяє швидко зрозуміти структуру програми.
3. IL-код досить високорівневий і читабельний, тому його легше аналізувати, ніж звичайний асемблер.
4. Дебаг символи та PDB файли. Файли програмної бази даних (Program Database, PDB) містять відладочну інформацію, яка може бути використана для аналізу коду. Ці файли містять інформацію про вихідні файли, методи, змінні та інші дані, корисні для реверс-інжинірингу.

HOW .NET LANGUAGES WORK

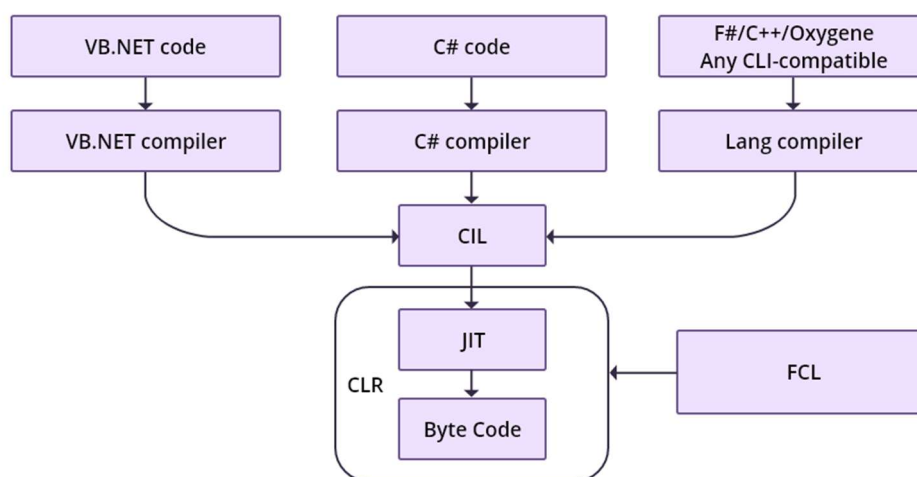


Рисунок 1.16 – Етапи компіляції вихідного коду (<https://celestialsys.com/blogs/an-introduction-to-net-framework/>)

Процес реверс-інжинірингу

Перший крок реверс-інжинірингу .NET-додатка це збір інформації: аналіз маніфестів, метаданих, структури збірок тощо.

У подальшому проведення декомпіляції використовуючи інструменти декомпіляції, реверс-інженер перетворює IL-код у високорівневий код на C# або VB.NET. Це дозволяє досліджувати логіку роботи додатка та взаємодію між його компонентами.

Аналіз вихідного коду дозволяє виявити вразливості, логічні помилки або інших аспекти, які можуть бути корисні для розуміння роботи ПЗ.

При необхідності, реверс-інженер може вносити зміни у код, повторно компілювати збірку та тестувати модифікований додаток. Це може бути використано для виправлення помилок, додавання нових функцій або виявлення вразливостей.

Існуючі вразливості вихідного коду

Для захисту коду від реверс-інжинірингу розробники можуть використовувати **обфускацію** це процес перетворення коду у менш читабельну форму. Інструменти обфускації можуть змінювати імена змінних, методів, класів та інші елементи, що ускладнює аналіз коду.

Інші методи захисту можуть включати **кодування та шифрування** важливих частин коду або даних. Це може запобігти несанкціонованому доступу до критично важливих частин додатка.

Використовуючі спеціалізоване ПЗ та технік для аналізу реверс-інженери можуть використовувати діючі методи аналізу захисту для виявлення та обходу механізмів захисту, таких як обфускація або шифрування.

1.8. ХЕШ-ФУНКЦІЇ MD5, SHA-1, SHA-256, SSDEEP, IMPHASH

Хеш-функції дозволяють створювати унікальні цифрові відбитки (хеші) даних [19,38,48], за допомогою цих відбитків можна однозначно ідентифікувати інформацію, перевіряти її цілісність і забезпечувати безпеку [19,38,52].

Окрім звичайного (чіткого) хешування існує ще нечітке хешування (fuzzy hashing). Воно допомагає знаходити схожі файли або дані, навіть якщо вони не повністю ідентичні.

Це особливо корисно в галузі цифрової криміналістики та аналізу шкідливих програм, де необхідно виявляти варіанти відомих зразків шкідливого ПЗ.

Хеш-функція MD5. MD5 (Message Digest Algorithm 5) – це криптографічна хеш-функція, розроблена Рональдом Рівестом у 1991 році. Вона приймає на вхід повідомлення довільної довжини і генерує 128-бітний (16-байтовий) хеш.

MD5 розбиває вхідні дані на блоки по 512 біт кожен. Кожен блок проходить через кілька етапів обробки, включаючи додавання заповнювача, обчислення контрольної суми та серію побітових операцій. Результатом є 128-бітний хеш-код.

Недоліки MD5 – більше не вважається криптографічно безпечним, оскільки є відомі методи для знаходження колізій (двох різних повідомлень з однаковим хешем). Тому MD5 не рекомендується для використання в критично важливих системах безпеки.

MD5 використовується для перевірки цілісності файлів при завантаженні або передачі даних, але не для криптографічних цілей.

Хеш-функція SHA-1 (Secure Hash Algorithm 1) – це криптографічна хеш-функція, розроблена Національним інститутом стандартів і технологій (NIST). Вона приймає на вхід повідомлення довільної довжини і генерує 160-бітний (20-байтовий) хеш.

SHA-1 розбиває вхідні дані на блоки по 512 біт кожен. Кожен блок проходить через кілька етапів обробки, включаючи додавання заповнювача, обчислення контрольної суми та серію побітових операцій. Результатом є 160-бітний хеш-код.

SHA-1 вже не вважається безпечним. У 2017 році дослідники продемонстрували реальну колізію для цього алгоритму. Через це фахівці рекомендують відмовитися від SHA-1 на користь більш стійких варіантів, наприклад SHA-256.

Хеш-функція SHA-256 (Secure Hash Algorithm 256-bit) – це член сімейства алгоритмів SHA-2, розроблених NIST. Вона приймає на вхід повідомлення довільної довжини і генерує 256-бітний (32-байтовий) хеш.

SHA-256 розбиває вхідні дані на блоки по 512 біт кожен. Кожен блок проходить через кілька етапів обробки, включаючи додавання заповнювача, обчислення контрольної суми та серію побітових операцій. Результатом є 256-бітний хеш-код.

SHA-256 вважається криптографічно безпечним і стійким до колізій. Вона забезпечує високу ступінь безпеки для сучасних криптографічних застосувань.

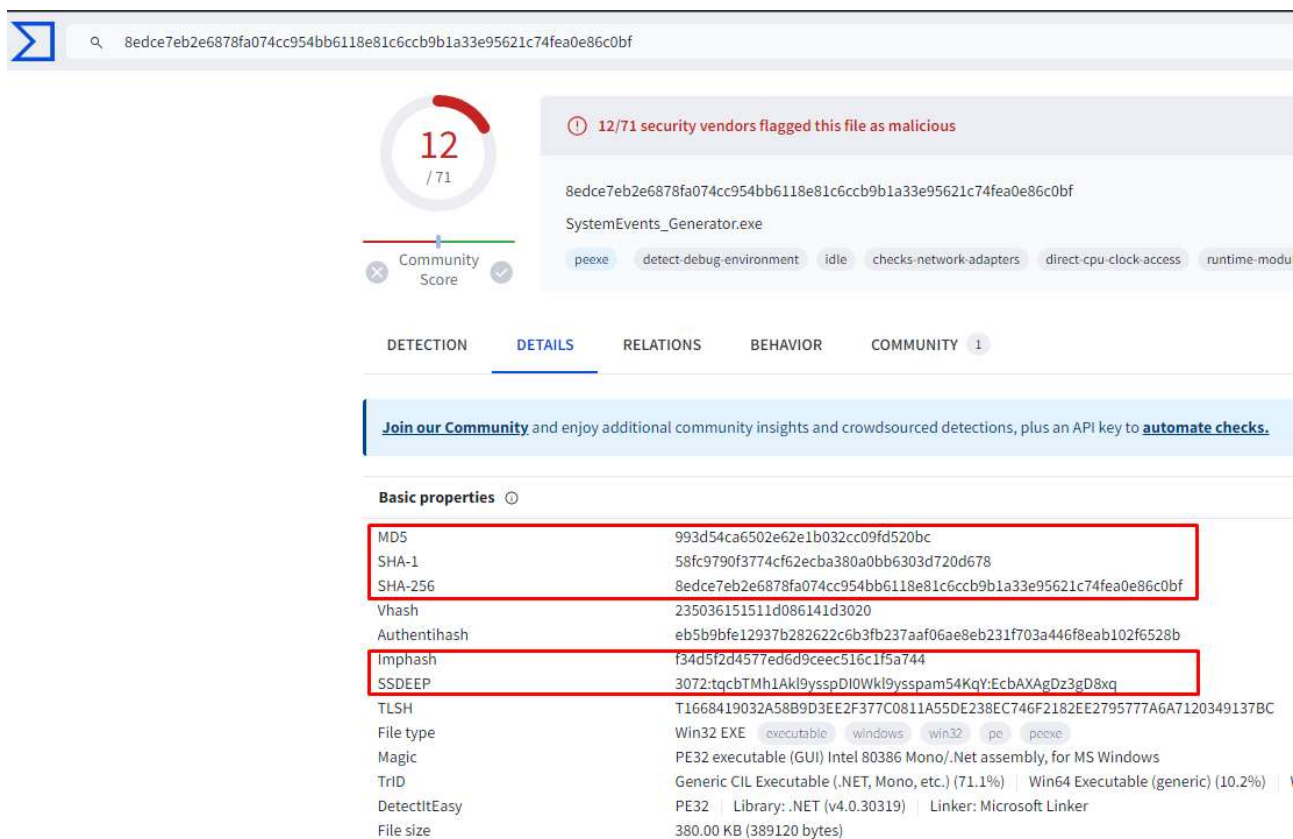
Приклад використання – широко використовується у криптографічних застосуваннях, таких як цифрові підписи, сертифікати SSL/TLS, блокчейн та інші.

SSDEEP (Context Triggered Piecewise Hashing) алгоритм нечіткого хешування здатний знаходити схожість між файлами навіть після невеликих змін.

Алгоритм розбиває файл на частини фіксованого розміру, обчислює хеші для кожного блоку і потім порівнює їх за допомогою метрики відстані Левенштейна.

У цифровій криміналістиці SSDEEP часто використовують для пошуку схожих версій шкідливого ПЗ або документів, які трохи відрізняються один від одного.

Як перевага можна зарахувати можливість виявлення схожих файлів навіть при наявності незначних змін. Недолік – обмежена точність порівняння для дуже малих або дуже великих файлів; можливість хибних спрацьовувань при порівнянні дуже різних файлів.



8edce7eb2e6878fa074cc954bb6118e81c6ccb9b1a33e95621c74fea0e86c0bf

12 / 71

12/71 security vendors flagged this file as malicious

8edce7eb2e6878fa074cc954bb6118e81c6ccb9b1a33e95621c74fea0e86c0bf

SystemEvents_Generator.exe

peexe detect-debug-environment idle checks-network-adapters direct-cpu-clock-access runtime-modu

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 1

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	993d54ca6502e62e1b032cc09fd520bc
SHA-1	58fc9790f3774cf62ecba380a0bb6303d720d678
SHA-256	8edce7eb2e6878fa074cc954bb6118e81c6ccb9b1a33e95621c74fea0e86c0bf
Vhash	235036151511d086141d3020
Authenticash	eb5b9bfe12937b282622c6b3fb237aaf06ae8eb231f703a446f8eab102f6528b
Imphash	f34d5f2d4577ed6d9ceec516c1f5a744
SSDEEP	3072:tcqbTMh1AKl9ysspDl0Wkl9ysspam54KqY:EcbAXAgDz3gD8xq
TLSH	T1668419032A58B9D3EE2F377C0811A55DE238EC746F2182EE2795777A6A7120349137BC
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
TrID	Generic CIL Executable (.NET, Mono, etc.) (71.1%) Win64 Executable (generic) (10.2%)
DetectItEasy	PE32 Library: .NET (v4.0.30319) Linker: Microsoft Linker
File size	380.00 KB (389120 bytes)

Рисунок 1.17 – Приклад створення хешів за допомогою Virustotal

Хеш-функція IMPHASH (Import Hash) – це алгоритм хешування, який використовується для створення сигнатур на основі таблиці імпортів виконуваних файлів PE (Portable Executable). Він особливо корисний для аналізу шкідливих програм, оскільки дозволяє виявляти схожість між файлами на основі їхніх імпортів.

IMPHASH обчислює хеш на основі списку імпортованих функцій та бібліотек у виконуваному файлі. Для цього він спочатку сортує список імпортів, а потім застосовує криптографічну хеш-функцію (зазвичай MD5) до отриманого

списку. Працює для ідентифікації варіантів шкідливих програм, які мають схожі імпортовані функції та дозволяє знаходити нові модифікації.

Перевага це можливість виявлення схожих виконуваних файлів на основі їхнього функціоналу; корисність у класифікації шкідливих програм.

Недолік – може бути обійдено шляхом мінімальних змін у списку імпортів; не застосовується до файлів без таблиці імпортів.

MD5, SHA-1 та SHA-256 залишаються важливими в криптографії та контролі цілісності даних, однак MD5 і SHA-1 вже застаріли через вразливість до колізій. На сьогодні SHA-256 є одним з найнадійніших алгоритмів і рекомендується для використання в системах безпеки.

Нечітке хешування SSDEEP та IMPHASH, є важливим інструментом для цифрової криміналістики та аналізу шкідливих програм

1. SSDEEP дозволяє виявляти схожі файли навіть при наявності незначних змін, що робить його корисним для аналізу файлів загального призначення.

2. IMPHASH, з іншого боку, орієнтований на аналіз виконуваних файлів та шкідливого ПЗ, дозволяючи виявляти схожість на основі таблиці імпортів.

1.9. ШАБЛОНИ ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ YARA

YARA [79] це скрити які активно застосовують для виявлення та класифікації шкідливого програмного забезпечення, сформовані скриптові правила описують характерні ознаки malware у файлах або в пам'яті.

Основні поняття YARA

1. Правило YARA. Правило складається з назви, секції метаданих (опціонально), секції умов і секції тіла, яка містить шаблони, за якими буде проводитися пошук.

2. Секції правил:

– Meta. Містить метадані про правило, такі як автор, опис, версія тощо.

Ця секція є опціональною.

– Strings. Визначає строки або шаблони, які будуть шукатися у файлі або пам'яті. Ця секція є обов'язковою.

– Condition. Описує умови, за яких правило буде спрацьовувати. Ця секція є обов'язковою.

Структура правила YARA

1. Загальний синтаксис правила YARA.

```
rule RuleName {  
  meta:  
    author = "Author Name"  
    description = "Description of the rule"  
    version = "1.0"  
  strings:  
    $string1 = "malicious string"  
    $string2 = { 6A 40 68 00 30 00 00 6A 14 8D 91 }  
    $string3 = /malware[0-9]+/  
  condition:  
    $string1 or $string2 or $string3  
}
```

2. Секція Meta. Містить метадані, які описують правило. Метадані не впливають на роботу правила, але допомагають документувати його.

```
meta:
  author = "Author Name"
  description = "Description of the rule"
  version = "1.0"
```

3. Секція Strings у правилах YARA дозволяє задавати текстові рядки, шістнадцяткові послідовності або регулярні вирази, які потрібно шукати у файлах.

```
strings:
  $string1 = "malicious string"
  $string2 = { 6A 40 68 00 30 00 00 6A 14 8D 91 }
  $string3 = /malware[0-9]+/
```

4. Секція Condition. Описує умови спрацьовування правила. Умова може включати логічні оператори, оператори порівняння та інші конструкції.

```
condition:
  $string1 or $string2 or $string3
```

Приклади формування правил YARA

1. Правило для виявлення конкретної строки

```
rule ExampleRule1 {
  meta:
    author = "John Doe"
    description = "Detects a specific malicious string"
    version = "1.0"
  strings:
    $malicious_string = "evil_malware"
  condition:
    $malicious_string
}
```

2. Правило для виявлення шістнадцяткового шаблону

```
rule ExampleRule2 {
  meta:
    author = "Jane Smith"
    description = "Detects a specific hex pattern"
    version = "1.0"
  strings:
    $hex_pattern = { E8 ?? ?? ?? ?? 68 74 65 78 74 }
```

```
condition:  
    $hex_pattern  
}
```

3. Правило з використанням регулярного виразу

```
rule ExampleRule3 {  
    meta:  
        author = "Alice Brown"  
        description = "Detects a pattern using regular expressions"  
        version = "1.0"  
    strings:  
        $regex_pattern = /malware[0-9]+/  
    condition:  
        $regex_pattern  
}
```

РОЗДІЛ 2. ІНСТРУМЕНТАРІЙ ДЛЯ ВИРІШЕННЯ ЗАДАЧ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ

2.1. ПЗ FAR MANAGER (FILE AND ARCHIVE MANAGER)

ЛІЦЕНЗІЯ

Open Source (GPLv2). поширюється під ліцензією GNU General Public License версії 2. Це означає, що програма є відкритим програмним забезпеченням і може вільно використовуватися, модифікуватися та розповсюджуватися будь-ким.

РОЗРОБНИК

FAR Manager був розроблений Євгеном Рошалем.

ОФІЦІЙНИЙ РЕСУРС

<https://farmanager.com/index.php?l=ru>

ОПИС

FAR Manager (File and Archive Manager) – це потужний текстовий менеджер файлів для Windows, створений для ефективного управління файлами та архівами. Програма надає зручний інтерфейс з широким набором функцій для професійного та повсякденного використання.

Основні функції

Управління файлами - FAR Manager дозволяє копіювати, переміщати, перейменовувати, видаляти файли та каталоги. Програма підтримує роботу з великою кількістю файлів і каталогів, забезпечуючи зручне сортування та фільтрацію.

Робота з архівами - підтримка багатьох форматів архівів, таких як ZIP, RAR, TAR, GZ. FAR Manager дозволяє створювати, розпаковувати та переглядати вміст архівів безпосередньо з програми.

Налаштування інтерфейсу

Конфігурація панелей дає змогу налаштовувати зовнішній вигляд робочого середовища та обирати однопанельний або двопанельний режим. Панелі можуть відображати різні відомості про файли, зокрема розмір, дату зміни, атрибути та інші службові параметри.

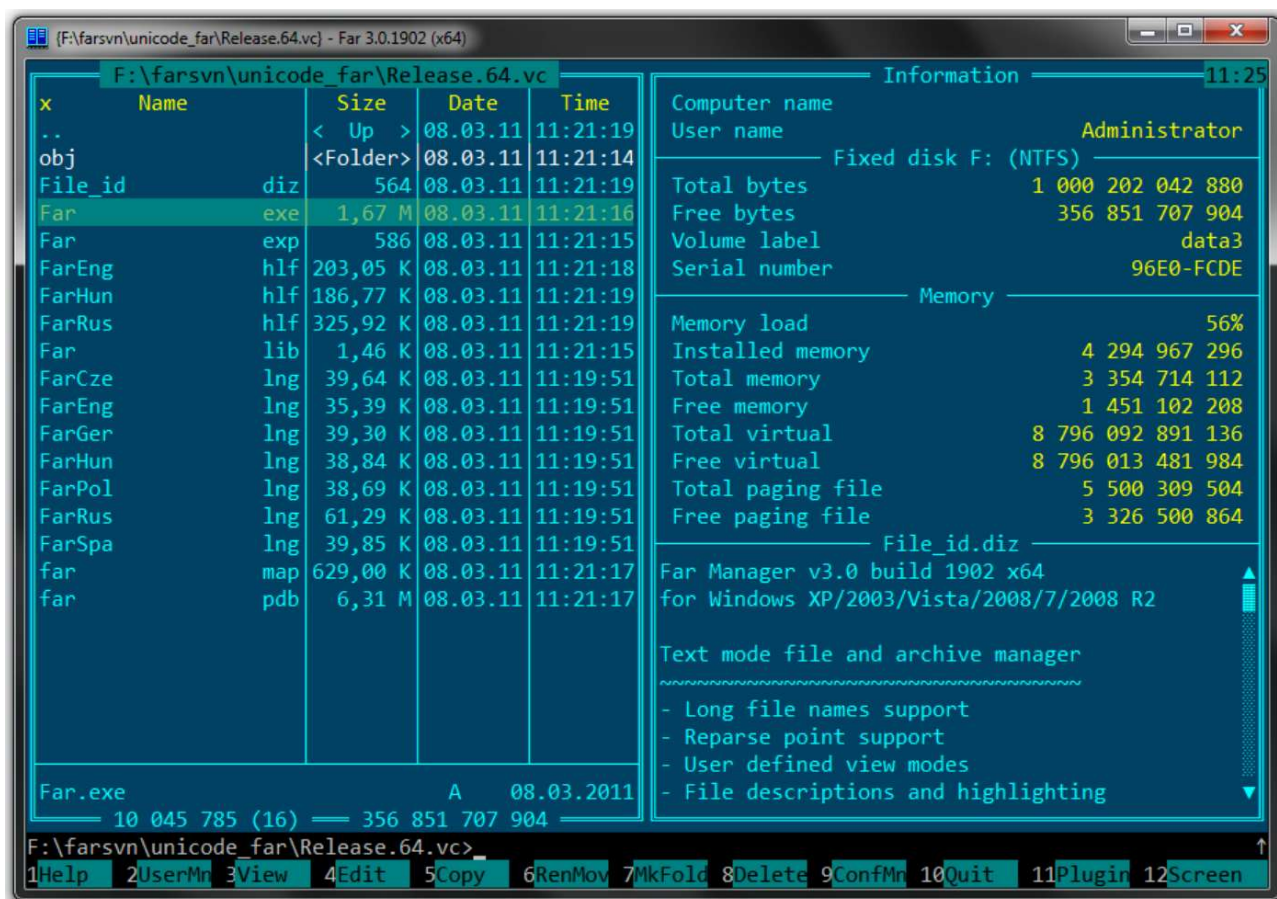


Рисунок 1.1 – Головне вікно ПЗ FAR Manager

Теми та плагіни - FAR Manager підтримує зміну тем оформлення та установку плагінів, що розширюють функціональність програми. Наприклад,

плагіни можуть додавати підтримку нових форматів файлів, інтеграцію з системами контролю версій, та інші корисні функції.

Редагування файлів

Текстовий редактор - вбудований текстовий редактор FAR Manager дозволяє редагувати текстові файли безпосередньо в програмі. Редактор підтримує підсвічування синтаксису для різних мов програмування, що робить його корисним для розробників.

Редагування у HEX форматі дає змогу працювати з файлами на рівні окремих байтів. Це корисно для детального аналізу, перевірки структури файлів і внесення точкових змін до даних різних типів.

Пошук файлів

FAR Manager має розвинені засоби пошуку файлів і вмісту файлів. Пошук може виконуватися за розширенням, розміром, датою зміни, текстовим вмістом та іншими критеріями. Окремою перевагою є підтримка пошуку всередині архівів, що дає змогу швидко знаходити потрібну інформацію навіть у стиснених даних.

Підтримка різних операційних систем

Windows - FAR Manager розроблений для роботи в операційній системі Windows і повністю підтримує всі її версії, починаючи з Windows 2000 до найновіших версій Windows 10 і 11.

Linux (через Wine) - FAR Manager може працювати в операційній системі Linux за допомогою середовища Wine. Це дозволяє користувачам Linux використовувати всі функції FAR Manager без необхідності перемикатися на Windows.

Підтримка макросів дає змогу автоматизувати повторювані дії та підвищити ефективність роботи користувача. Це робить FAR Manager зручним інструментом для професійного використання, особливо під час виконання типових операцій з файлами.

Плагіни

Розширення функціональності - FAR Manager підтримує установку плагінів, які значно розширюють можливості програми. Серед популярних плагінів є FTP-клієнт, плагіни для роботи з системами контролю версій (Git, SVN), інтеграція з програмами для обробки зображень та багато інших.

Відкритий API для плагінів користувачів дозволяє розширювати можливості. Користувачі можуть створювати власні розширення, адаптувати програму під свої спеціалізовані задачі та інтегрувати її з іншими інструментами.

Найнеобхідніші плагіни

1. FAR FTP. Плагін для інтеграції FTP-клієнта, який дозволяє зручно підключатися до FTP-серверів, завантажувати та вивантажувати файли безпосередньо з FAR Manager.

2. PE Analyzer дозволяє аналізувати виконувані файли формату PE (Portable Executable) у Windows. Плагін надає детальну інформацію про структуру PE-файлів, включаючи заголовки, таблиці імпорту/експорту, секції та інші метадані. Це корисно для розробників програмного забезпечення та спеціалістів з безпеки для аналізу та діагностики виконуваних файлів.



n	Table	Size	n	Name	Name
..			ShellNew	bootstat.dat	
Export		0	SoftwareDistribution	applauncher.mif	
Import		14	Speech	explorer.exe	
Import (delay load)		0	symbols	fveupdate.exe	
Resource		6	system	HelpPane.exe	
			System32	hh.exe	

notepad.exe	0	Ln	2505/10817	Col 1	Ch 1	48
00000000000003647:	488D1542940000		lea	rdx, qword ptr ds:[		
0000000000000364E:	488D0D2B940000		lea	rcx, qword ptr ds:[		
00000000000003655:	E816FDFFFF		call	0000000010000328Bh		
0000000000000365A:	C70500CA000002000000		mov	dword ptr ds:[000000		
00000000000003664:	85F6		test	esi, esi		
00000000000003666:	7509		jne	0000000010000357Bh		
00000000000003668:	33C0		xor	eax, eax		
0000000000000366A:	4887058FC90000		xchg	qword ptr ds:[000000		
00000000000003671:	48833D97DA000000		cmp	qword ptr ds:[000000		
00000000000003679:	0F8595410000		jne	0000000010000770Bh		
0000000000000367F:	488B05728F0000		mov	rax, qword ptr ds:[		
00000000000003686:	488B38		mov	rdi, qword ptr ds:[		
x64 00000000000003689:	48897C2428		mov	qword ptr ss:[rsp+2		
0000000000000368E:	8B742420		mov	esi, dword ptr ss:[		
> 00000000000003692:	803F20		cmp	byte ptr ds:[rdi],		
ilLeft 00000000000003695:	7624		jbe	00000000100003596h		

Рисунок 2.2 –FAR Manager, плагін PE Analyzer

3. HexEditor плагін для редагування файлів у шістнадцятковому (hex) форматі. Він дозволяє переглядати та змінювати вміст файлів на рівні байтів, що є корисним для детального аналізу та модифікації бінарних файлів, зокрема для розробників, інженерів та спеціалістів з безпеки.

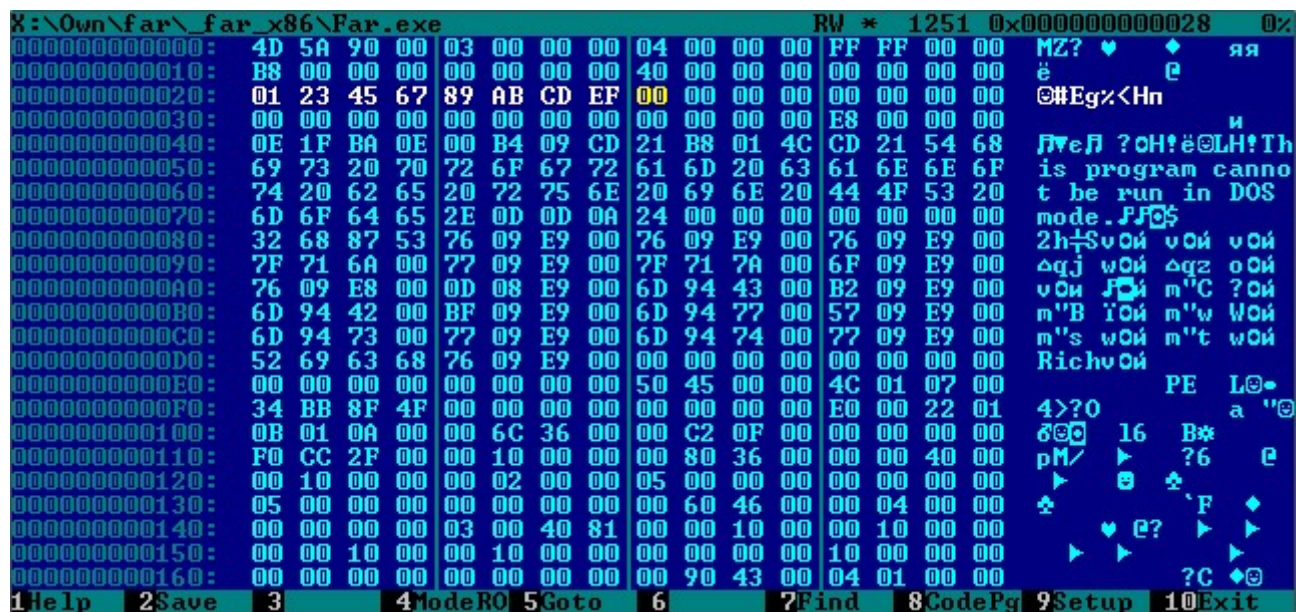


Рисунок 2.3 –FAR Manager, плагін HexEditor

4. GitBranch інтегрує систему контролю версій Git у FAR Manager. Плагін дозволяє виконувати основні операції з репозиторіями Git, такі як створення та перемикавання гілок, перегляд статусу репозиторію, коміти та інші дії безпосередньо з інтерфейсу FAR Manager.

5. FileTypes надає можливість налаштовувати асоціації файлів у FAR Manager. Плагін дозволяє визначити, які програми будуть використовуватися для відкриття файлів певних типів, а також створювати користувацькі команди для обробки файлів різних форматів.

6. VirusTotal інтегрує сервіс VirusTotal у FAR Manager. Плагін дозволяє перевіряти файли на наявність шкідливого програмного забезпечення за допомогою багатьох антивірусних систем, надаючи швидкий доступ до результатів перевірки прямо з інтерфейсу FAR Manager.

7. DarkCrypt FAR є плагіном для шифрування та дешифрування файлів. Він підтримує криптографічні операції безпосередньо у FAR Manager і забезпечує зручний спосіб захисту даних під час роботи з файлами.

8 File Unlock дозволяє розблоковувати файли, які використовуються іншими процесами. Плагін корисний у випадках, коли файл не може бути видалений, переміщений або змінений через те, що він заблокований системою або іншими програмами.

9. WiFi плагін додає функції управління бездротовими мережами у FAR Manager. Він дозволяє переглядати доступні мережі, підключатися до них, перевіряти стан підключення та налаштовувати параметри бездротової мережі прямо з інтерфейсу FAR Manager.

ПРИКЛАДИ ВИКОРИСТАННЯ

1. FAR Manager часто використовується системними адміністраторами для управління файлами та каталогами на серверних системах. Програма дозволяє швидко копіювати, переміщувати, перейменовувати та видаляти файли, що значно спрощує рутинні завдання. Як приклад адміністратор може за допомогою FAR Manager створювати резервні копії конфігураційних файлів сервера, перевіряти журнали подій та видаляти тимчасові файли для оптимізації роботи системи.

2. FAR Manager використовується програмістами для керування проектами та вихідним кодом. Програма підтримує зручну навігацію файловою структурою, редагування текстових файлів і роботу з інструментами розробника. Наприклад, розробник може застосовувати FAR Manager для впорядкування файлів проекту, редагування коду та взаємодії із системами контролю версій, зокрема Git.

3. Користувачі, що працюють з великими обсягами даних легко маніпулюють даними. ПЗ дозволяє створювати, розпаковувати та переглядати вміст архівів різних форматів (ZIP, RAR, TAR тощо). Це зручно для користувачів, які часто працюють з архівованими даними. Як приклад

користувач може створити архів для передачі великої кількості файлів, розпакувати архів з отриманими даними, а також переглянути вміст архіву перед його розпакуванням.

ВИСНОВКИ

FAR Manager є потужним інструментом для управління файлами та архівами, який надає користувачам широкі можливості для ефективної роботи з даними.

Основні переваги FAR Manager:

1. FAR Manager дозволяє швидко виконувати операції з файлами та каталогами, що значно підвищує продуктивність користувачів.
2. Програма підтримує велику кількість плагінів, що дозволяє розширювати її функціональність відповідно до потреб користувача.
3. FAR Manager підтримує роботу з архівами різних форматів, що робить його універсальним інструментом для роботи з даними.

До недоліків FAR Manager можна віднести потребу в попередньому навчанні нових користувачів і відсутність традиційного графічного інтерфейсу, що може бути незручним для частини користувачів. Водночас програма залишається популярною серед DFIR спеціалістів, системних адміністраторів, розробників та інших ІТ фахівців, які цінують швидкість, гнучкість і ефективність роботи з файлами та даними.

2.2. ПЗ AUTOPSY

ЛІЦЕНЗІЯ

Apache License 2.0. Відкрита ліцензія, яка дозволяє вільно використовувати, змінювати і розповсюджувати програму.

РОЗРОБНИК

Розроблена компанією Basis Technology.

ОФІЦІЙНИЙ РЕСУРС

<https://www.autopsy.com/>

ОПИС

Autopsy це комплексний інструмент цифрової криміналістики, який використовують для аналізу та відновлення даних з різних носіїв [45,52,70].

Система орієнтована на правоохоронців, корпоративних спеціалістів і експертів з кібербезпеки. Вона має зручний інтерфейс і підтримує розширення через плагіни з відкритим кодом (<https://github.com/markmckinnon/Autopsy-Plugins/blob/master/README.md>).

Аналіз файлових систем

Autopsy працює з великою кількістю файлових систем, зокрема NTFS, FAT, exFAT, HFS+, Ext2/3/4 та іншими.

Відновлення видалених файлів – програма може відновлювати видалені файли та каталоги, що є критично важливим для розслідування інцидентів.

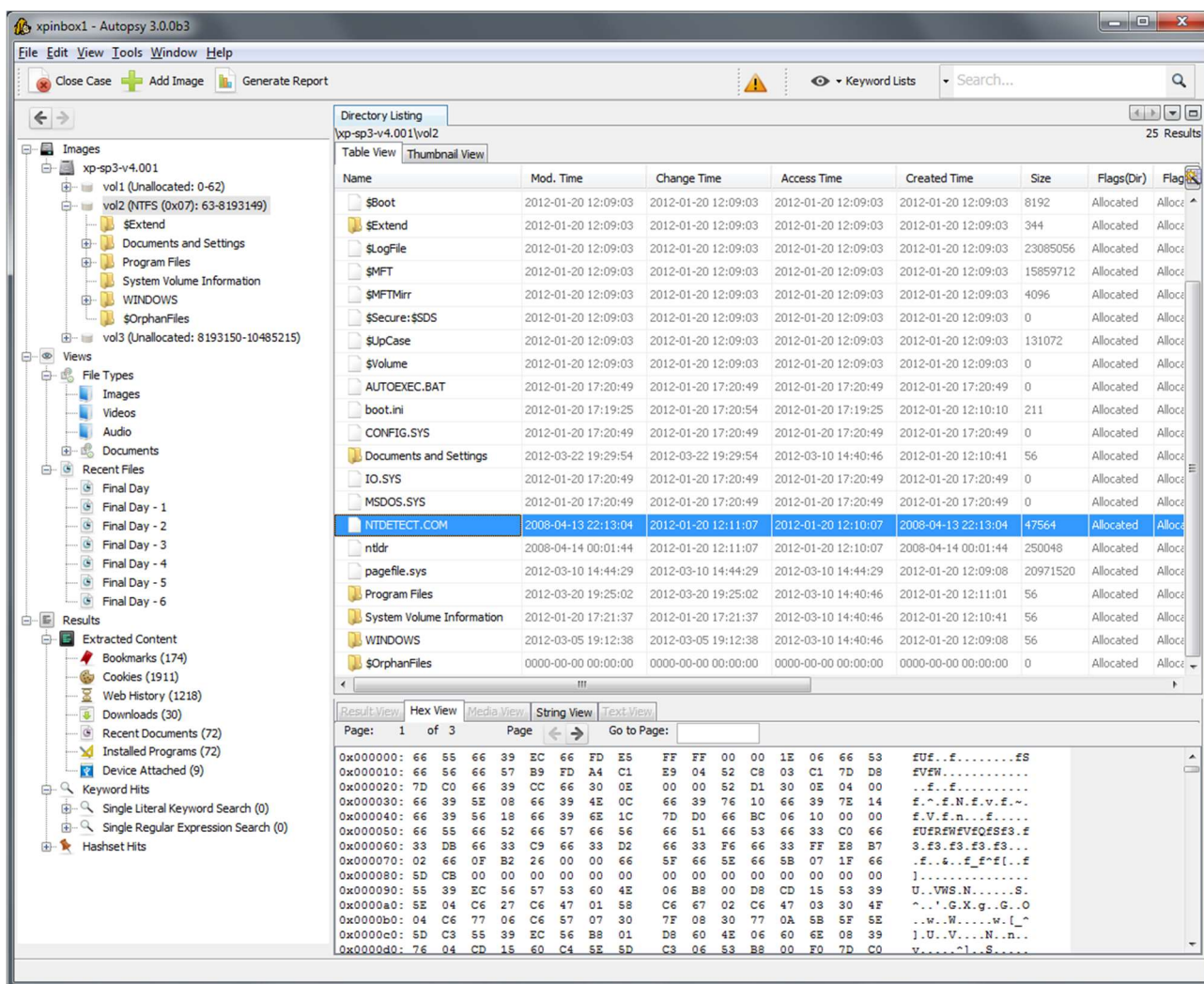


Рисунок 2.4 – Головне вікно ПЗ Autopsy

Аналіз інтернет-активності

Перегляд історії браузера – Autopsy дозволяє аналізувати історію браузера, закладки, файли cookie та завантаження для виявлення інтернет-активності підозрюваного.

Аналіз електронної пошти – програма може аналізувати файли поштових скриньок для виявлення та відновлення повідомлень електронної пошти.

Аналіз мультимедійних файлів

Під час роботи з образами дисків програма дає змогу швидко переглядати файли, вивчати їхній зміст і метадані. Це допомагає знаходити важливі цифрові сліди під час розслідування.

Аналіз відео та аудіо – програма підтримує аналіз відео та аудіо файлів, що дозволяє дослідникам відтворювати та аналізувати мультимедійні файли.

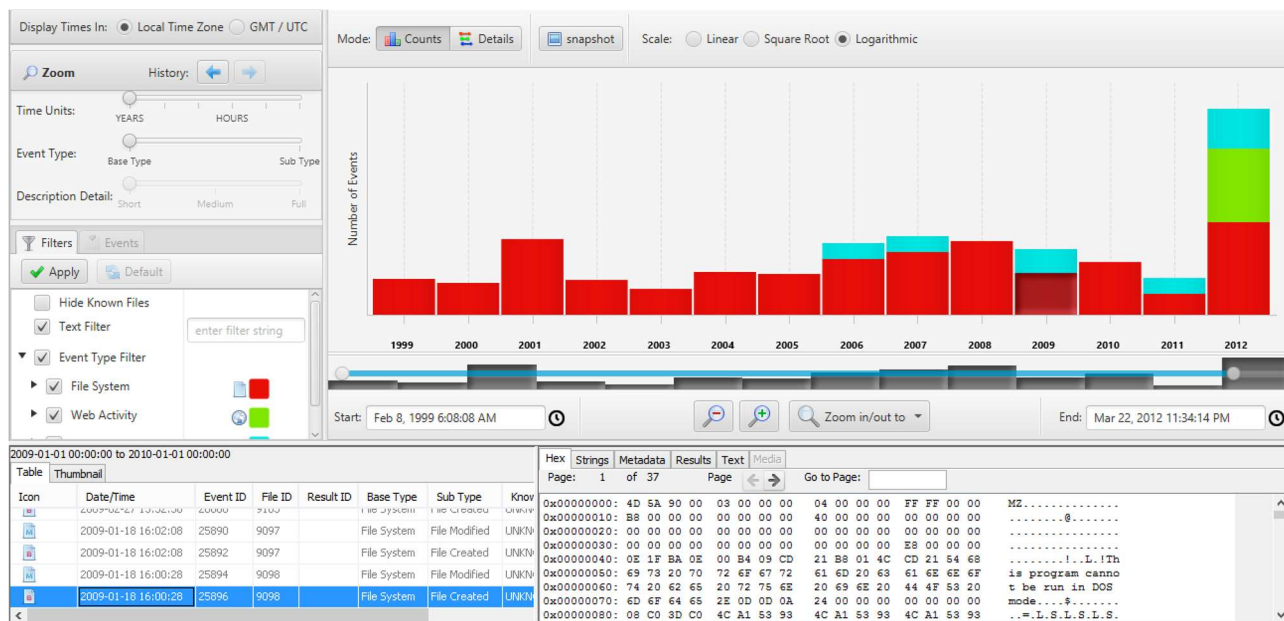


Рисунок 2.5 – ПЗ Autopsy (Timeline Analysis)

Автоматизація процесів

Автоматичне створення звітів, Autopsy може автоматично генерувати звіти на основі зібраних даних, що спрощує документування процесу розслідування.

Підтримка скриптів і засобів автоматизації дає змогу скоротити час виконання повторюваних операцій, підвищити ефективність роботи дослідника та стандартизувати окремі етапи аналізу.

Модульна архітектура

Autopsy інтегрується з The Sleuth Kit, має модульну архітектуру що дає можливість додавати нові плагіни для аналізу реєстру, пошуку malware та виконання інших завдань.

Плагіни значно розширюють функціональні можливості Autopsy, забезпечуючи інструментами для детального аналізу та розслідування цифрових інцидентів.

Стандартні плагіни

1. AD1_Extractor

Цей плагін дозволяє витягувати дані з архівів AD1, які використовуються в цифрових розслідуваннях для зберігання артефактів.

2. Amazon Echosystem Parser

Аналізує бази даних зображень Amazon Alexa, допомагаючи витягувати важливу інформацію з пристроїв Echo.

3. Atomic Wallet

Плагін для аналізу гаманців криптовалют, зокрема Atomic Wallet, забезпечуючи доступ до даних транзакцій та облікових записів.

4. CCM Recently Used Apps

Аналізує базу даних WMI для виявлення недавно використаних додатків, що може бути корисно для відстеження активності користувачів.

5. ClamAV Hashsets

Використовує антивірусні хеш-набори ClamAV для перевірки файлів на наявність шкідливого ПЗ.

6. Create Data Source Hashset

Створює хеш-набори даних для джерел, які можна імпортувати назад в Autopsy для подальшого аналізу.

7. Create Preview Data Container

Окремі плагіни Autopsy можуть створювати розширюваний віртуальний диск VHD, монтувати його, експортувати файли за визначеними розширеннями та розмонтовувати диск для подальшого додавання отриманих даних до кейсу Autopsy.

8. Cuckoo

Інтегрує Cuckoo Sandbox для перевірки статусу сервера та надсилання файлів для аналізу на наявність шкідливого ПЗ.

9. DJI Phantom Drone Parser

Аналізує дані з безпілотників DJI Phantom, витягуючи інформацію про польоти та інші важливі дані.

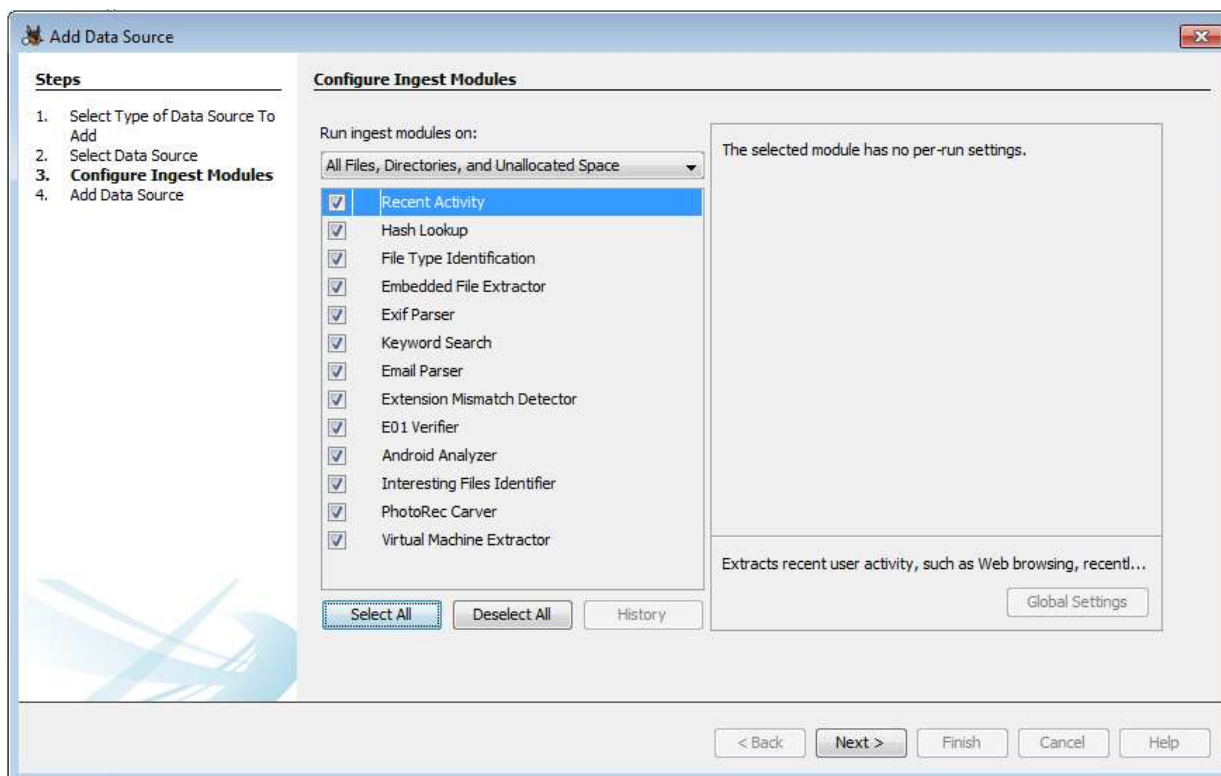


Рисунок 2.6 –ПЗ Autopsy (Ingest Modules)

10. FileHistory

Експортує файл Catalog1.edb та створює базу даних SQLite з історією файлів, яку можна імпортувати в Autopsy для аналізу.

11. Google Takeout

Аналізує дані з Google Takeout, надаючи доступ до архівів даних, завантажених з облікових записів Google.

12. LevelDB Parser

Аналізує бази даних LevelDB, які часто використовуються в сучасних додатках для зберігання даних.

13. MacFSEvents

Експортує каталог .fsevents з macOS та аналізує події файлової системи, створюючи базу даних для подальшого аналізу.

14. MacOSX Recent

Експортує та аналізує недавні документи та файли в macOS, допомагаючи відстежувати останню активність користувача.

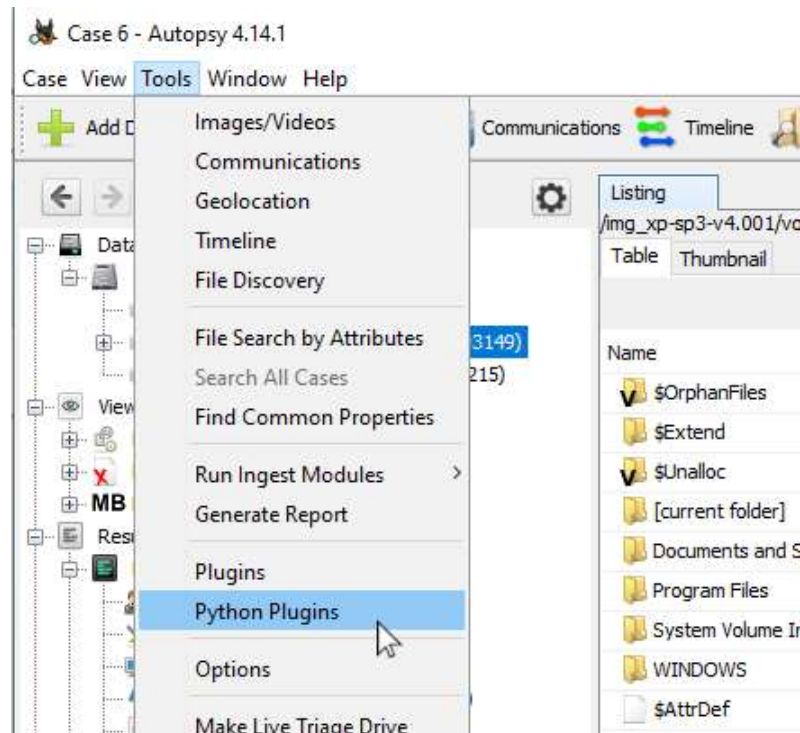


Рисунок 2.7 – ПЗ Autopsy (Python Modules)

15. MacOSX Safari

Аналізує історію браузера Safari на macOS, створюючи базу даних для подальшого аналізу активності користувача в Інтернеті.

16. Mac Mail Parser

Аналізує файли пошти macOS, витягуючи та аналізуючи повідомлення електронної пошти.

17. Mass Export By Extension

Експортує файли з певними розширеннями, що дозволяє швидко витягнути всі релевантні файли з досліджуваного диску.

18. Parse Plist

Аналізує файли Plist (Property List), які використовуються в macOS та iOS для зберігання налаштувань та конфігурацій.

19. Parse QNX Image

Аналізує зображення дисків QNX, що часто використовуються в вбудованих системах.

20. Parse SQLite DBs

Аналізує файли баз даних SQLite, імпортуючи їх в Autopsy для подальшого аналізу.

21. Parse SQLite DB Del Records

Аналізує бази даних SQLite на наявність видалених записів та створює базу даних з відновленими даними.

22. Parse Usnj

Експортує файл \$UsnJrnl:\$J з NTFS та аналізує журнал змін файлової системи.

23. Plaso

Виконує Plaso або імпортує файл Plaso для створення хронологічних ліній часу зібраних артефактів.

24. Process Activities Cache

Аналізує кеш активностей Windows 11, витягуючи дані про дії користувача.

25. Process Amcache

Експортує реєстр Amcache та аналізує його, створюючи базу даних з інформацією про встановлені програми.

26. Process Appx Programs

Аналізує базу даних SQLite для додатків Windows Store.

27. Process Appxreg Programs

Аналізує реєстр для додатків Windows Store/Metro.

28. Parse EVTX

Експортує журнали подій Windows та створює базу даних для подальшого аналізу.

29. Parse EVTX by Event ID

Аналізує журнали подій Windows за ідентифікатором події, витягуючи специфічні події для подальшого аналізу.

30. Process Extract VSS

Експортує знімки томів (Volume Shadow Copies) для аналізу.

31. Process Facebook Chats

Аналізує файли чату Facebook, витягуючи повідомлення для подальшого аналізу.

32. Process Teracopy

Аналізує журнали копіювання файлів програмою TeraCopy, витягуючи дані про перенесення файлів.

33. Process Windows Mail

Аналізує файли пошти Windows Mail, витягуючи повідомлення для подальшого аналізу.

34. Remove Artifacts

Видаляє артефакти з кейсу Autopsy, які більше не потрібні для розслідування.

35. Shimcache Parser

Аналізує кеш сумісності (Shimcache) Windows, витягуючи дані про запуснені програми.

36. Spotlight Parser

Аналізує дані індексації Spotlight з macOS, допомагаючи витягувати інформацію про файли та каталоги.

37. Thumbcache Parser

Аналізує кеш мініатюр (thumbcache) Windows, витягуючи зображення мініатюр файлів.

38. Thumbs Parser

Аналізує файли Thumbs.db з Windows, витягуючи зображення мініатюр файлів.

39. Timesketch

Інтегрується з Timesketch для створення та аналізу хронологічних ліній часу з артефактів.

40. UAL Parser

Аналізує журнали UAL (User Access Logging) Windows, витягуючи дані про доступ користувачів.

41. Volatility

Використовує Volatility для аналізу дамів пам'яті, допомагаючи витягувати інформацію про процеси, мережеву активність та інші дані.

42. Webcache

Аналізує кеш веб-браузерів Windows, витягуючи історію переглядів, файли cookie та інші дані.

43. Windows Internals

Аналізує внутрішні компоненти Windows, витягуючи дані про системні події та налаштування.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів в організаціях.

1. У корпоративному середовищі Autopsy часто застосовують для виявлення витоків даних і дослідження підозрілої активності. Наприклад, можна проаналізувати історію браузерів, пошту та інші дії користувачів.

З точки зору судової експертизи.

2. Судові експерти використовують Autopsy для відновлення видалених файлів, аналізу метаданих і побудови хронології подій.

З точки зору навчальних та дослідницьких цілей.

3. У навчальних закладах програма допомагає студентам набувати практичних навичок відновлювати видалені файли, аналізувати активність користувачів, вивчати основи комп'ютерної криміналістики.

ВИСНОВКИ

Переваги Autopsy:

1. Підтримка багатьох файлових систем і типів носіїв;
2. Можливість розширення за допомогою плагінів;
3. Інтеграція з іншими інструментами, зокрема The Sleuth Kit.

До недоліків можна віднести необхідність часу на навчання та залежність розширених функцій від якості плагінів.

2.3. ПЗ THE VOLATILITY FRAMEWORK

ЛІЦЕНЗІЯ

Використовується Open Source версії GPLv2 є відкритим ПЗ, може вільно використовуватися, модифікуватися, розповсюджуватися.

РОЗРОБНИК

Volatility була розроблена Volatility Foundation, некомерційною організацією, що займається розвитком інструментів для цифрової криміналістики.

ОФІЦІЙНИЙ РЕСУРС

<https://volatilityfoundation.org/>

ОПИС

Найпоширеніша у світі платформа криміналістики пам'яті Volatility – це потужний інструмент для аналізу дамів пам'яті, який широко використовується в цифровій криміналістиці та розслідуванні інцидентів кібербезпеки [13-15,71]. Volatility Foundation, цей інструмент забезпечує детальний аналіз оперативної пам'яті для виявлення шкідливого програмного забезпечення, прихованих процесів та інших аномалій.

Аналіз пам'яті

Інструмент підтримує дампи пам'яті з Windows, Linux, macOS та Android. Це дозволяє працювати з різними платформами.

Може витягувати текстові рядки, що зберігаються в пам'яті, а також відновлювати файли, які були відкриті під час створення дампу пам'яті.

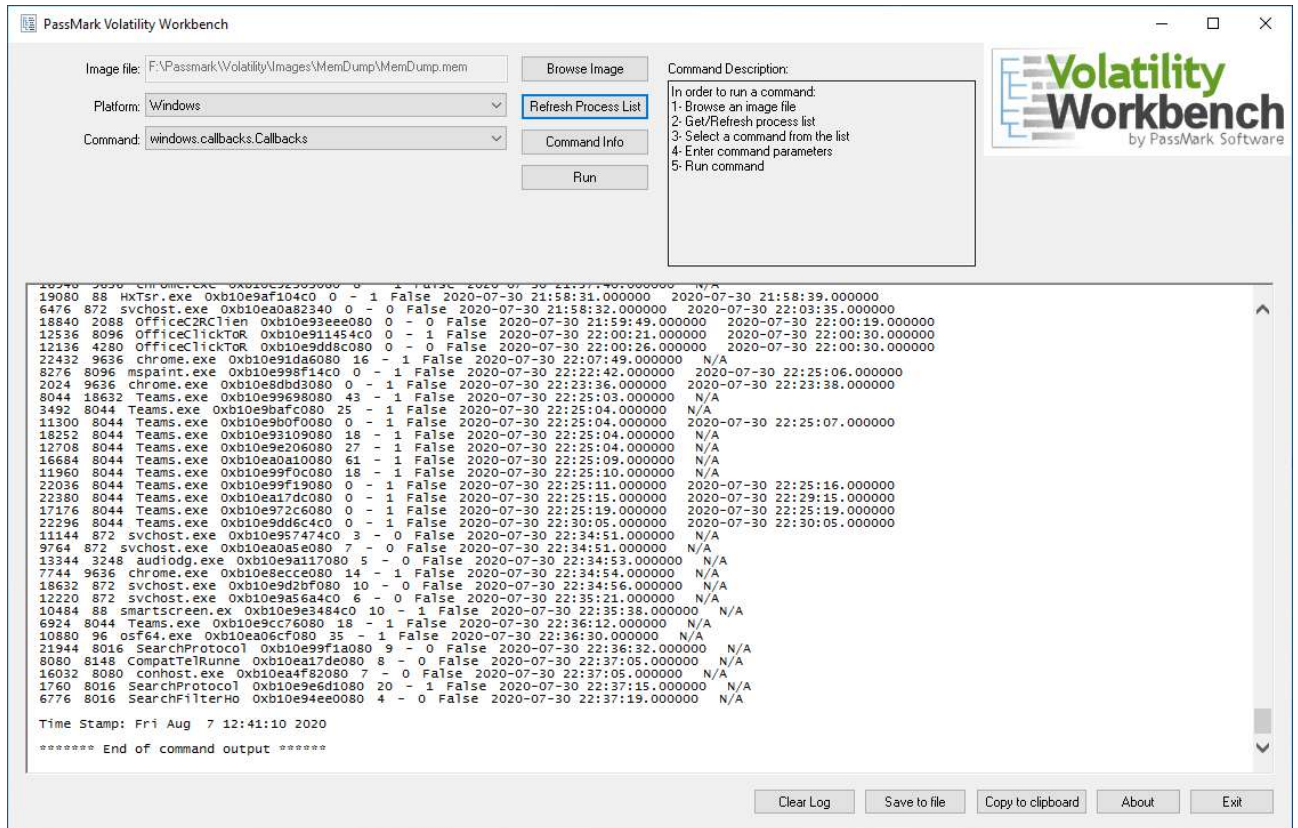


Рисунок 2.8 – Графічний інтерфейс користувача (GUI) для інструменту Volatility

Аналіз процесів

Виявлення шкідливих процесів, які можуть бути невидимими для звичайних системних інструментів. Це включає аналіз таблиці процесів та пошук аномалій.

Перевірка DLL – ПЗ аналізує завантажені DLL-файли для виявлення зловмисних або невідомих модулів, які можуть бути індикаторами компрометації системи.

Аналіз мережевої активності

Перегляд активних з'єднань включаючи активні з'єднання, відкриті порти та мережеві сесії. Це допомагає виявити підозрілу мережеву поведінку.

Аналіз мережевих пакетів – ПЗ може витягувати та аналізувати мережеві пакети, що передавалися під час створення дампу пам'яті.

Аналіз реєстру

Дозволяє аналізувати вміст реєстру Windows для виявлення змін та налаштувань, які можуть вказувати на присутність шкідливого ПЗ або інші зміни в системі.

Відновлення видалених ключів – ПЗ може відновлювати видалені ключі реєстру, що є важливим для відстеження змін в системі.

Автоматизація процесів

Volatility дає змогу використовувати скрипти і плагіни для автоматизації аналізу пам'яті.

Забезпечує можливість потокового аналізу великих обсягів даних, що допомагає оптимізувати роботу та підвищити ефективність аналізу.

Список стандартних плагінів Volatility та їх можливостей:

1. pslist

Плагін pslist відображає перелік активних процесів у дампі оперативної пам'яті, що дає змогу виявляти підозрілі або невідомі процеси, які можуть свідчити про потенційно шкідливу активність.

Командний рядок використання:

```
volatility -f memory.dmp -profile=Win7SP1x64 pslist
```

```
root@NeilFoxDellXPS: ~/volatility3
Volatility 3 Framework 2.0.0 PDB scanning finished
```

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File output
4	0	System	0xbe0a574ac040 153	-	N/A	False	2021-05-20 07:28:25.000000	N/A	Disabled	
312	4	smss.exe	0xbe0a58d18800 2	-	N/A	False	2021-05-20 07:28:25.000000	N/A	Disabled	
428	412	csrss.exe	0xbe0a59215080 11	-	0	False	2021-05-20 07:28:36.000000	N/A	Disabled	
492	312	smss.exe	0xbe0a598cd080 0	-	1	False	2021-05-20 07:28:39.000000	2021-05-20 07:28:40.000000	Disabled	
500	412	wininit.exe	0xbe0a598c2080 1	-	0	False	2021-05-20 07:28:39.000000	N/A	Disabled	
516	492	csrss.exe	0xbe0a59900280 15	-	1	False	2021-05-20 07:28:39.000000	N/A	Disabled	
584	492	winlogon.exe	0xbe0a5994e800 6	-	1	False	2021-05-20 07:28:40.000000	N/A	Disabled	
636	500	services.exe	0xbe0a599a2800 9	-	0	False	2021-05-20 07:28:41.000000	N/A	Disabled	
652	500	lsass.exe	0xbe0a599cb080 8	-	0	False	2021-05-20 07:28:41.000000	N/A	Disabled	
740	636	svchost.exe	0xbe0a59891800 25	-	0	False	2021-05-20 07:28:42.000000	N/A	Disabled	
816	636	svchost.exe	0xbe0a59a60800 17	-	0	False	2021-05-20 07:28:43.000000	N/A	Disabled	
924	584	dwm.exe	0xbe0a59ab84c0 11	-	1	False	2021-05-20 07:28:44.000000	N/A	Disabled	
956	636	svchost.exe	0xbe0a59a54800 93	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
1020	636	svchost.exe	0xbe0a59a50800 29	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
80	636	svchost.exe	0xbe0a59a4e800 38	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
496	636	svchost.exe	0xbe0a59a4c800 23	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
800	636	svchost.exe	0xbe0a59a4a800 28	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
1144	636	svchost.exe	0xbe0a59bbd800 28	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
1452	636	svchost.exe	0xbe0a58de7300 8	-	0	False	2021-05-20 07:28:44.000000	N/A	Disabled	
1600	636	svchost.exe	0xbe0a5963a5c0 11	-	0	False	2021-05-20 07:28:45.000000	N/A	Disabled	
1684	636	spoolsv.exe	0xbe0a596bf800 14	-	0	False	2021-05-20 07:28:45.000000	N/A	Disabled	
1948	636	svchost.exe	0xbe0a59cea380 15	-	0	False	2021-05-20 07:28:46.000000	N/A	Disabled	

Рисунок 2.9 – Приклад використання інструментарію Volatility, плагін psslist

2. psscan

psscan виконує глибокий скан процесів у пам'яті, включаючи ті, що були завершені, або приховані зловмисними програмами.

Командний рядок використання:

volatility -f memory.dmp --profile=Win7SP1x64 psscan

3. dlllist

dlllist відображає список завантажених DLL для кожного активного процесу, допомагаючи ідентифікувати підозрілі модулі.

Командний рядок використання:

volatility -f memory.dmp --profile=Win7SP1x64 dlllist

4. netscan

netscan аналізує мережеві підключення та відкриті порти, що допомагає у виявленні підозрілої мережевої активності.

Командний рядок використання:

volatility -f memory.dmp --profile=Win7SP1x64 netscan

Volatility 3 Framework 2.0.0

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	State	PID	Owner	Created
0xbe0a5819f380	UDPv4	0.0.0.0	5353	*	0		1144	svchost.exe	2021-07-06 17:11:06.000000
0xbe0a5819f380	UDPv6	::	5353	*	0		1144	svchost.exe	2021-07-06 17:11:06.000000
0xbe0a58520c60	UDPv6	fe80::cc9:303e:e00d:bf34		56056	*	0	3008	svchost.exe	2021-07-06 17:17:05.000000
0xbe0a58cb06a0	TCPv4	0.0.0.0	49664	0.0.0.0	0	LISTENING	500	wininit.exe	2021-05-20 07:28:43.000000
0xbe0a58cb11a0	TCPv4	0.0.0.0	49669	0.0.0.0	0	LISTENING	1684	spoolsv.exe	2021-05-20 07:28:45.000000
0xbe0a58cb11a0	TCPv6	::	49669	::	0	LISTENING	1684	spoolsv.exe	2021-05-20 07:28:45.000000
0xbe0a591ee750	TCPv4	0.0.0.0	49664	0.0.0.0	0	LISTENING	500	wininit.exe	2021-05-20 07:28:43.000000
0xbe0a591ee750	TCPv6	::	49664	::	0	LISTENING	500	wininit.exe	2021-05-20 07:28:43.000000
0xbe0a59428010	UDPv4	127.0.0.1	51722	*	0		956	svchost.exe	2021-05-20 07:28:46.000000

Рисунок 2.10 – Приклад використання інструментарію Volatility, плагін netscan

5. cmdscan

cmdscan витягує командний рядок і історію команд, що були введені користувачем, що допомагає в розслідуванні активності користувача.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 cmdscan
```

6. hivelist

hivelist знаходить і відображає всі адреси реєстрових вуликів в пам'яті, що дозволяє аналізувати ключі реєстру.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 hivelist
```

7. malfind

malfind виявляє підозрілі регіони пам'яті, що можуть містити завантажені зловмисні програми або шкідливий код.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 malfind
```

```
Volatility 3 Framework 2.0.0 PDB scanning finished

PID      Process Start VPN      End VPN Tag      Protection      CommitCharge      PrivateMemory      File output      Hexdump Disasm
904      MsMpEng.exe      0x25a04690000      0x25a0478ffff      VadS      PAGE_EXECUTE_READWRITE      256      1      Disabled
cc cc cc cc cc cc cc cc .....
cc cc cc cc cc cc cc cc .....
cc cc cc cc cc cc cc cc .....
cc cc cc cc cc cc cc cc .....
cc cc cc cc cc cc cc cc .....
cc cc cc cc cc cc cc cc .....
cc 08 00 42 00 00 00 00 ...B....
05 48 8b 45 20 48 89 c2 .H.E.H..
0x25a04690000: int3
0x25a04690001: int3
0x25a04690002: int3
0x25a04690003: int3
0x25a04690004: int3
0x25a04690005: int3
0x25a04690006: int3
0x25a04690007: int3
0x25a04690008: int3
```

Рисунок 2.11 – Приклад використання інструментарію Volatility, плагін malfind

8. memdump

memdump дозволяє витягувати окремі процеси з дампу пам'яті, що допомагає в аналізі конкретних підозрілих процесів.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 memdump -p [pid] -D /output/dir
```

9. shellbags

shellbags витягує та аналізує Shellbag ключі реєстру, що дозволяє відстежувати активність користувача щодо перегляду папок у файловій системі.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 shellbags
```

10. procdump

procdump дозволяє створювати дампи конкретних процесів, які потім можуть бути проаналізовані на наявність шкідливого коду або інших аномалій.

Командний рядок використання:

```
volatility -f memory.dmp --profile=Win7SP1x64 procdump -p [pid] -D /output/dir
```

```
root@NeilFoxDellXPS:~/volatility3# ls procdump/
file.0xdb8980509990.0xdb89804f2db0.ImageSectionObject.ntdll.dll.img
file.0xdb8980504950.0xdb8980509bd0.ImageSectionObject.ntdll.dll.img
file.0xdb898121f470.0xdb89812bad0.ImageSectionObject.rpctr4.dll.img
file.0xdb8981220bc0.0xdb89812c1db0.ImageSectionObject.SHCore.dll.img
file.0xdb89812553d0.0xdb8981462c70.ImageSectionObject.advapi32.dll.img
file.0xdb898126a5c0.0xdb897f44c40.ImageSectionObject.windows.storage.dll.img
file.0xdb8981280b60.0xdb898127b010.ImageSectionObject.wow64.dll.img
file.0xdb898128d6d0.0xdb898128c8b0.ImageSectionObject.wow64cpu.dll.img
file.0xdb898128f1a0.0xdb8981313ac0.ImageSectionObject.ws2_32.dll.img
file.0xdb89812bcef0.0xdb89812bb140.ImageSectionObject.win32u.dll.img
file.0xdb89812c9430.0xdb89812208c0.ImageSectionObject.gdi32full.dll.img
file.0xdb89812fc700.0xdb89812a0de0.ImageSectionObject.wow64win.dll.img
file.0xdb8981303080.0xdb898121f790.ImageSectionObject.user32.dll.img
file.0xdb8981304310.0xdb8981310950.ImageSectionObject.profpapi.dll.img
file.0xdb8981306080.0xdb8981304010.ImageSectionObject.crypt32.dll.img
file.0xdb8981310720.0xdb898128fad0.ImageSectionObject.shlwapi.dll.img
file.0xdb8981316480.0xdb8981312010.ImageSectionObject.powrprof.dll.img
file.0xdb8981369af0.0xdb89813c6010.ImageSectionObject.winnsi.dll.img
file.0xdb89813bbb90.0xdb89813a8c0.ImageSectionObject.cryptsp.dll.img
file.0xdb89813bea60.0xdb89813254d0.ImageSectionObject.OnDemandConnRouteHelper.dll.img
file.0xdb89813bfe0.0xdb8981396ab0.ImageSectionObject.urlmon.dll.img
file.0xdb89813cb080.0xdb8981364560.ImageSectionObject.rsanh.dll.img
file.0xdb89813fd080.0xdb89812b4140.ImageSectionObject.kernel32.dll.img
file.0xdb89813fd210.0xdb8981302de0.ImageSectionObject.gdi32.dll.img
file.0xdb898142e610.0xdb898142d010.ImageSectionObject.msvc_p_win.dll.img
file.0xdb89814306c0.0xdb898142f010.ImageSectionObject.combase.dll.img
file.0xdb8981432b70.0xdb8981432710.ImageSectionObject.oleaut32.dll.img
file.0xdb898143b6d0.0xdb8981449a40.ImageSectionObject.ucrbase.dll.img
file.0xdb898143bdf0.0xdb898143ade0.ImageSectionObject.kernel.appcore.dll.img
file.0xdb898143c760.0xdb8981439de0.ImageSectionObject.shell32.dll.img
file.0xdb898143d900.0xdb898143d470.ImageSectionObject.msaasn1.dll.img
file.0xdb898143fa00.0xdb898143ee20.ImageSectionObject.msvcr7.dll.img
file.0xdb89814539c0.0xdb89812b2010.ImageSectionObject.nsi.dll.img
file.0xdb8981453ef0.0xdb8981453ae0.ImageSectionObject.sechost.dll.img
file.0xdb89814555f0.0xdb89814519d0.ImageSectionObject.KernelBase.dll.img
file.0xdb8981455a20.0xdb8981255010.ImageSectionObject.cfmgr32.dll.img
file.0xdb8981466b60.0xdb89814520e0.ImageSectionObject.sspicli.dll.img
file.0xdb898148a140.0xdb8981425e20.ImageSectionObject.bcryptprimitives.dll.img
file.0xdb898157cef0.0xdb898157ca20.ImageSectionObject.cryptbase.dll.img
file.0xdb8981875d40.0xdb897fcbe5e0.ImageSectionObject.IPHLPAPI.DLL.img
file.0xdb8981add320.0xdb89826bc670.ImageSectionObject.wininet.dll.img
file.0xdb898212e900.0xdb89826b3740.ImageSectionObject.wtsapi32.dll.img
file.0xdb8982488570.0xdb89827b8e20.ImageSectionObject.bcrypt.dll.img
file.0xdb89826b1dd0.0xdb89827ea950.ImageSectionObject.iertutil.dll.img
file.0xdb89827bfef0.0xdb89813b31e0.ImageSectionObject.mswsock.dll.img
file.0xdb89829eab10.0xdb89818781a0.ImageSectionObject.userenv.dll.img
file.0xdb8982c24ef0.0xdb8982847db0.ImageSectionObject.dnsapi.dll.img
file.0xdb8982de6080.0xdb89813badb0.ImageSectionObject.winhttp.dll.img
file.0xdb898437e080.0xdb89805ed4f0.ImageSectionObject.smsfwdr.ex.img
file.0xdb898437e080.0xdb8980a1a20.DataSectionObject.smsfwdr.ex.dat
```

Рисунок 2.12 – Приклад використання інструментарію Volatility, плагін procdump

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів з шкідливим ПЗ.

1. Під час виявлення шкідливого програмного забезпечення Volatility використовується для аналізу об'єктів, які працюють в оперативній пам'яті або залишають у ній сліди своєї активності. Плагіни malfind та psscan допомагають ідентифікувати підозрілі процеси, приховані об'єкти, завантажені модулі та можливі ін'єкції коду. Наприклад, у разі компрометації корпоративної мережі дослідники можуть проаналізувати дамп пам'яті ураженої системи за допомогою Volatility і виявити приховані процеси, які не відображаються під час звичайного огляду операційної системи.

З точки зору аналізу активності користувачів.

2. Під час аналізу плагін cmdscan допомагає відновлювати команди, які вводилися в командному рядку. Це корисно для розуміння дій користувача перед інцидентом.

З точки зору аналізу мережевої активності.

3. Під час моніторингу netscan показує активні мережеві з'єднання, відкриті порти та підозрілу мережеву активність.

ВИСНОВКИ

Основні переваги Volatility:

1. Volatility підтримує аналіз різних форматів дамів пам'яті з операційних систем Windows, Linux, macOS та Android, що дає змогу досліджувати пам'ять широкого спектра пристроїв і програмних середовищ.

2. Volatility надає детальний аналіз процесів, модулів, мережевої активності та реєстру, що допомагає виявити приховані загрози та шкідливі програми.

3. Завдяки підтримці скриптів та плагінів, Volatility дозволяє автоматизувати процеси аналізу, що підвищує ефективність розслідування.

Як недолік Volatility вимагає певного часу на вивчення і залежить від наявності профілів для конкретних версій ОС.

2.4. ПЗ KAPE (Kroll Artifact Parser and Extractor)

ЛІЦЕНЗІЯ

КАРЕ дозволяє безкоштовно використовувати інструментарій будь-якою місцевою, державною чи міжнародною урядовою агенцією, освітньою чи дослідницькою організацією або для внутрішніх цілей компанії, окрім прав на комерційне використання [72].

Програма вимагає реєстрації на офіційному сайті, де користувачі можуть завантажити програму після погодження з умовами використання. Студенти та інші користувачі можуть використовувати програму для навчання та дослідницьких цілей.

РОЗРОБНИК

КАРЕ розроблена компанією Kroll, яка спеціалізується на кібербезпеці та цифровій криміналістиці.

ОФІЦІЙНИЙ РЕСУРС

<https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape>

ОПИС

КАРЕ (Kroll Artifact Parser and Extractor) швидко збирає найважливіші цифрові артефакти з комп'ютера або образу диска. Завдяки швидкості його часто використовують на початку розслідування, поки створюється повний образ диска.

Збір даних

КАРЕ вміє працювати з живими системами та образами, збирати журнали, дані реєстру, історію браузерів та багато інших артефактів.

Цілеспрямоване збирання даних - використовуючи попередньо визначені цілі та модулі, КАРЕ збирає лише необхідні дані, що знижує обсяг зібраних даних і прискорює подальший аналіз.

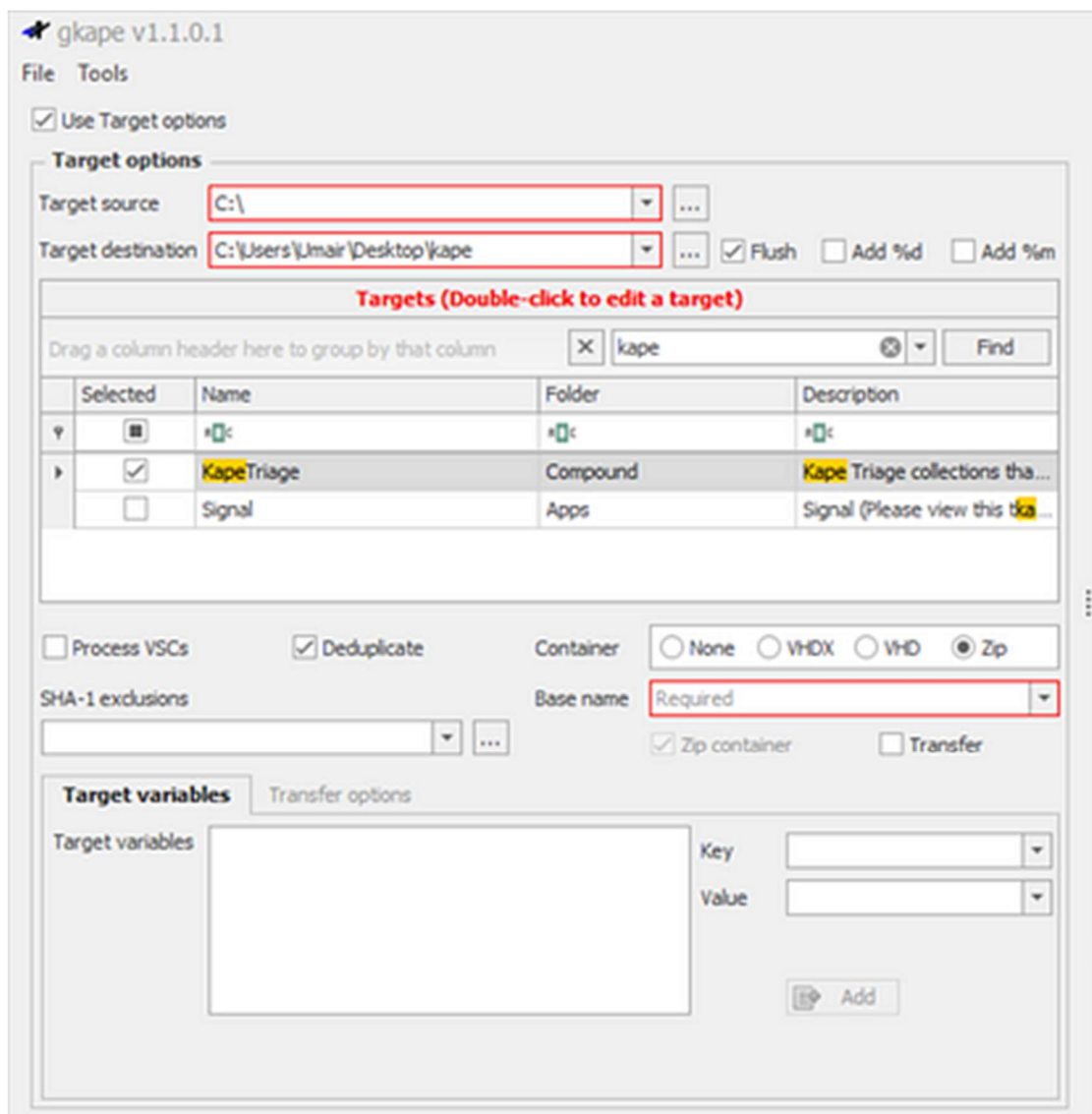


Рисунок 2.13 – Головне вікно ПЗ Kroll Artifact Parser and Extractor, ліва частина вікна – обрання додаткових умов та цілі сканування

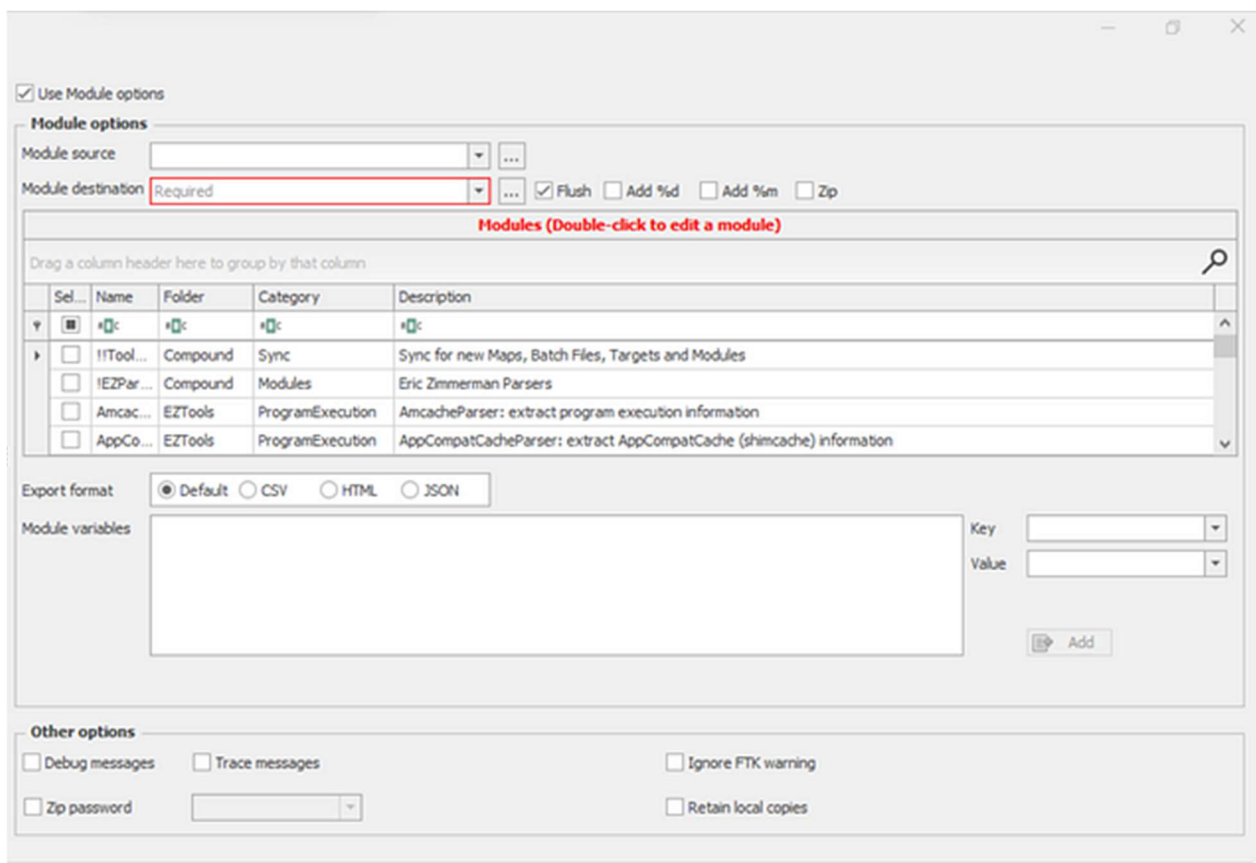


Рисунок 2.14 – Головне вікно ПЗ Kroll Artifact Parser and Extractor, права частина вікна – обрання модулів сканування

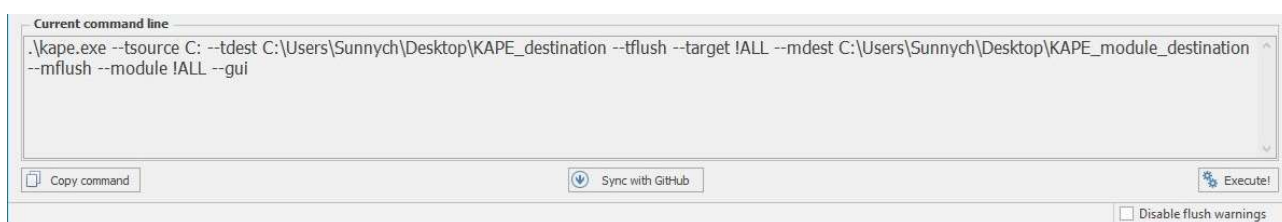


Рисунок 2.15 – Головне вікно ПЗ Kroll Artifact Parser and Extractor, нижня частина вікна - сформований результат параметрів для командного рядка

Обробка та аналіз

Після збору даних KAPE обробляє їх, розшифровує і передає в зручний формат. Інструмент добре інтегрується з Log2Timeline та RegRipper.

Звіти та візуалізація - генерує звіти та візуалізації, що допомагають фахівцям швидко зрозуміти дані та виявити ключові артефакти.

```
C:\KAPE>.\\kape.exe
KAPE version 0.9.0.1 Author: Eric Zimmerman (kape@kroll.com)

tsource      Target source drive to copy files from (C, D:, or F:\ for example)
target       Target configuration to use
tdest       Destination directory to copy files to. If --vhdx, --vhd or --zip is set, files will end up in VHD(X) container or zip file
tlist       List available targets. Use . for Targets directory or name of subdirectory under Targets.
tdetail     Dump target file details
tflush      Delete all files in 'tdest' prior to collection
tvars       Provide a list of key:value pairs to be used for variable replacement in targets. Ex: --tvars user:eric would allow for using %user% in a target
which is replaced with eric at runtime. Multiple pairs should be separated by ^
tdd         Deduplicate files from --tsource (and VSCs, if enabled) based on SHA-1. First file found wins. Default is TRUE

msource      Directory containing files to process. If using targets and this is left blank, it will be set to --tdest automatically
module       Module configuration to use
mdest       Destination directory to save output to
mlist       List available modules. Use . for Modules directory or name of subdirectory under Modules.
mdetail     Dump module processors details
mflush      Delete all files in 'mdest' prior to running modules
mvars       Provide a list of key:value pairs to be used for variable replacement in modules. Ex: --mvars foo:bar would allow for using %foo% in a module which is replaced with bar at runtime. Multiple pairs should be separated by ^
mef         Export format (csv, html, json, etc.). Overrides what is in module config

vss          Process all Volume Shadow Copies that exist on --tsource. Default is FALSE
vhdx         The base name of the VHD(X) file to create from --tdest. This should be an identifier, NOT a filename. Use this or --vhd or --zip
vhd          The base name of the VHD file to create from --tdest. This should be an identifier, NOT a filename. Use this or --vhdx or --zip
zip          The base name of the ZIP file to create from --tdest. This should be an identifier, NOT a filename. Use this or --vhdx or --vhd

scs          SFTP server host/IP for transferring *compressed VHD(X)* container
scp          SFTP server port. Default is 22
scu          SFTP server username. Required when using --scs
scpw        SFTP server password
scc          Comment to include with transfer. Useful to include where a transfer came from. Defaults to the name of the machine where KAPE is running

s3r          S3 region name. Example: us-west-1 or ap-southeast-2
s3b          S3 bucket name
s3k          S3 Access key
s3s          S3 Access secret
s3c          Comment to include with transfer. Useful to include where a transfer came from. Defaults to the name of the machine where KAPE is running

asu          Azure Storage SAS Uri
asc          Comment to include with transfer. Useful to include where a transfer came from. Defaults to the name of the machine where KAPE is running
```

Рисунок 2.16 – Можливі параметри командного рядка

Модульна архітектура

КАРЕ дозволяє дослідникам додавати нові модулі або налаштовувати існуючі для збору та обробки специфічних артефактів. Модулі написані на мові PowerShell, що полегшує їх розробку та налаштування.

КАРЕ групує артефакти в категорії, такі як EvidenceOfExecution, BrowserHistory, AccountUsage, спрощуючи процес аналізу.

Автоматизація процесів

Скрипти і командний рядок - КАРЕ підтримує автоматизацію збору та аналізу даних за допомогою скриптів та командного рядка, що скорочує час, необхідний для розслідування, особливо при великому обсязі даних.

Потокові операції - можливість налаштування поточкових операцій забезпечує ефективне управління великими обсягами даних та оптимізацію процесів аналізу.

ПРИКЛАДИ ВИКОРИСТАННЯ

КАРЕ працює у два проходи. У першому проході він копіює файли (цілі - targets), які можна зчитати, якщо ОС їх не заблокувала. Заблоковані файли передаються у вторинну чергу. У другому проході КАРЕ використовує методи прямого зчитування з диска, щоб обійти блокування ОС і копіювати ці файли. Копії зберігаються з оригінальними часовими мітками і метаданими в подібній структурі директорій. Модулі можуть бути незалежними бінарними файлами, які обробляють зібрані дані для вилучення інформації. Наприклад, модуль Prefetch Parser (PECmd) може витягувати і зберігати дані у форматі CSV.

З точки зору реагування на інциденти:

1. У разі кібератаки КАРЕ допомагає швидко знайти вектор проникнення. Під час розслідування витоків даних, зібрати потрібні докази з робочих станцій.

З точки зору цифрової криміналістики:

2. Аудит безпеки та перевірки конфігурацій систем. Наприклад, у разі підозри на витік конфіденційних даних КАРЕ дає змогу зібрати з робочої станції потрібні артефакти, зокрема історію браузера, поштові дані, системні журнали та іншу інформацію, що має значення для розслідування.

З точки зору аудиту безпеки:

3. Модульна архітектура робить КАРЕ гнучким інструментом, який можна адаптувати під різні завдання.

ВИСНОВКИ

КАРЕ є потужним та гнучким інструментом для збору, обробки та аналізу цифрових артефактів. Основні переваги КАРЕ включають:

1. КАРЕ дозволяє швидко зібрати дані з великої кількості систем, що значно скорочує час реагування на інциденти це головна перевага.
2. КАРЕ також застосовують для аудитів безпеки та перевірки конфігурацій систем.
3. КАРЕ може інтегруватися з іншими інструментами для проведення глибшого аналізу і розширення функціональних можливостей.

2.5. ПЗ SYSINTERALS UTILITES

ЛІЦЕНЗІЯ

Freeware. Sysinternals Utilities є безкоштовним набором програмного забезпечення, яке можна використовувати без обмежень для особистих і комерційних цілей. Всі утиліти доступні для завантаження безкоштовно з офіційного сайту Microsoft.

РОЗРОБНИК

Sysinternals Utilities були розроблені Марком Руссиновичем і Брюсом Когвіним. У 2006 році проєкт був придбаний компанією Microsoft.

ОФІЦІЙНИЙ РЕСУРС

<https://learn.microsoft.com/en-us/sysinternals/>

ОПИС

Sysinternals Utilities – це набір ПОТУЖНИХ ІНСТРУМЕНТІВ для управління, діагностики та усунення несправностей у операційних системах Windows [73-76]. Розроблений Марком Руссиновичем та Брюсом Когвіним, цей набір включає різноманітні утиліти, що охоплюють широке коло завдань, від моніторингу системних ресурсів до усунення проблем безпеки. Крім того Марк Руссинович видав книгу бестселер – «Windows Internals, Part 1: System architecture, processes, threads, memory management, and more, 7th Edition», «Windows Internals, Part 2, 7th Edition» (посилання на опис <https://learn.microsoft.com/en-us/sysinternals/resources/windows-internals>). Тільки в цих книгах можна прочитати практичну основу побудови ОС Windows [46, 47].

Розглянемо повний список програм набору з зазначенням найважливіших програм з офіційного сайту (<https://learn.microsoft.com/en-us/sysinternals/downloads/>) в яких немає прямих конкурентів:

1. AccessChk. Командний інструмент для перегляду дозволів на файли, реєстрові ключі, сервіси та процеси.
2. AccessEnum. Відображає доступ до директорій, файлів і реєстрових ключів.
3. AdExplorer. Переглядач і редактор Active Directory.
4. AdInsight. Інструмент для моніторингу LDAP у реальному часі.
5. AdRestore. Відновлює об'єкти Active Directory.
6. Autologon. Автоматичний вхід у систему без введення пароля.
7. **Autoruns (рекомендовано до використання).** Ця утиліта має найповніші знання про місця автозапуску серед усіх моніторів запуску. Вона показує програми, налаштовані для запуску під час завантаження системи або входу в систему, а також при запуску вбудованих додатків Windows, таких як Internet Explorer, Explorer і медіаплеєри.

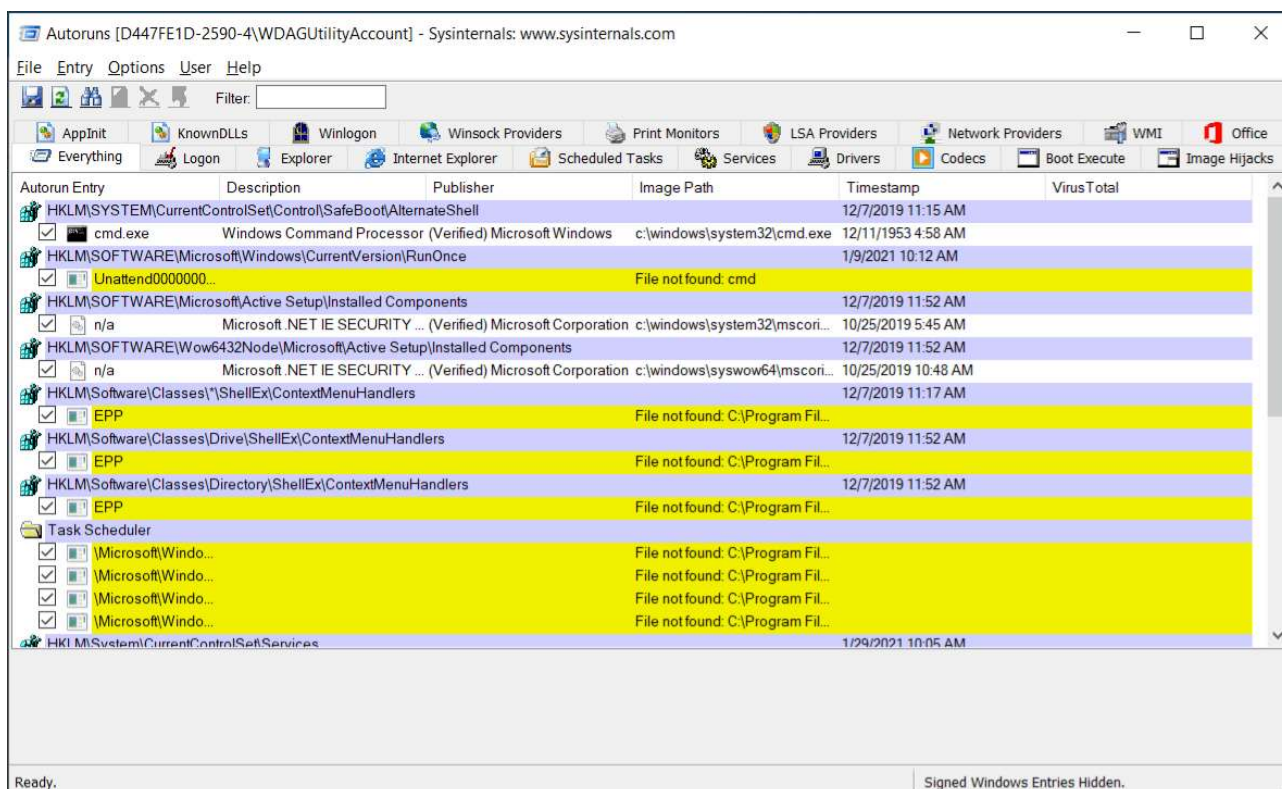


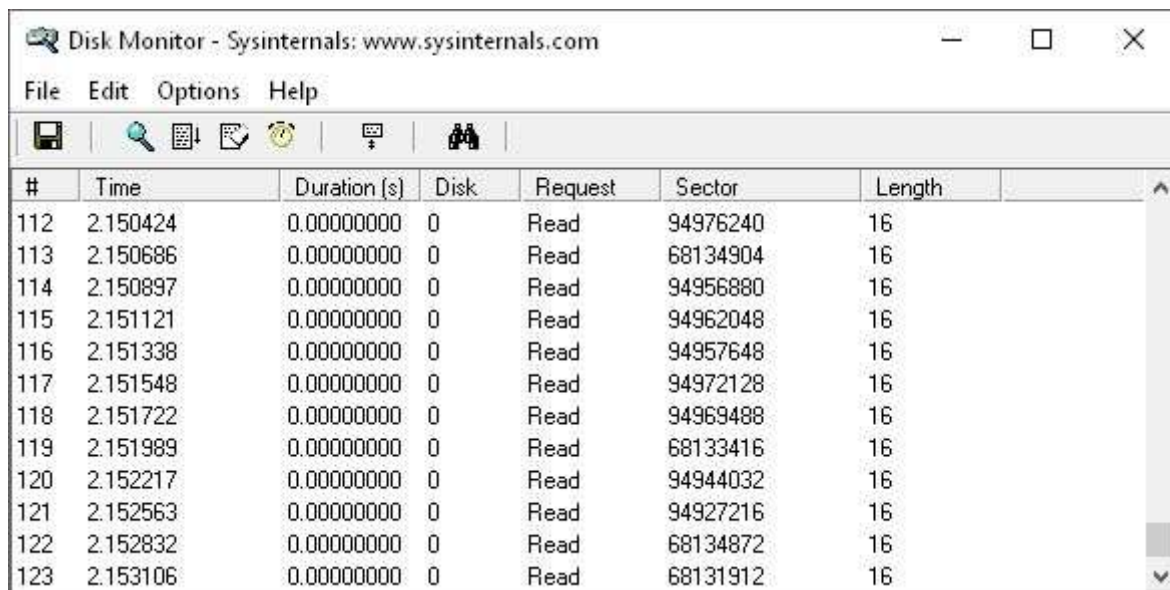
Рисунок 2.17 – ПЗ Autoruns, приклад роботи

Сюди входять програми та драйвери з папок автозавантаження, ключів реєстру Run, RunOnce та інших. Autoruns також відображає розширення оболонки Explorer, панелі інструментів, об'єкти-допоміжники браузера, сповіщення Winlogon, автозапуск сервісів тощо.

Опція "Hide Signed Microsoft Entries" допомагає зосередитися на сторонніх програмах, які додаються до автозапуску. Програма підтримує перегляд автозапуску для інших облікових записів системи. У комплекті також є командний рядок, який може виводити дані у форматі CSV, під назвою Autorunsc.

8. BgInfo. Генерує фон робочого столу з інформацією про систему.
9. BlueScreen. Імітує синій екран смерті (BSOD).
10. CacheSet. Управління розміром робочого набору кешу.
11. ClockRes. Відображає роздільну здатність системного годинника.
12. Contig. Дефрагментація окремих файлів.
13. Coreinfo. Показує відповідність між логічними процесорами та фізичним процесором.
14. Ctrl2cap. Драйвер для перетворення клавіші Caps Lock на Control.
15. DebugView. Перехоплює виклики DbgPrint драйверами.
16. Desktops. Створення віртуальних робочих столів.
17. Disk2vhd. Міграція фізичних систем у віртуальні машини.
18. DiskExt. Відображає зіставлення дисків і томів.
19. **DiskMon (рекомендовано до використання).** Захоплює активність жорсткого диска. ПЗ реєструє та відображає всю активність жорсткого диска на системі Windows. Вона може бути згорнута у системний трей, де працює як індикатор диска: зелений значок вказує на операції читання, а червоний – на операції запису.
20. DiskView. Графічна утиліта для перегляду секторів диска.
21. Disk Usage (DU). Відображає використання диска по каталогах.
22. EFSDump. Відображає інформацію про зашифровані файли.

23. FindLinks. Відображає індекс файлу та будь-які жорсткі посилання.



The screenshot shows the 'Disk Monitor' application window from Sysinternals. The window has a menu bar with 'File', 'Edit', 'Options', and 'Help'. Below the menu is a toolbar with icons for saving, finding, and other functions. The main area contains a table with the following columns: '#', 'Time', 'Duration (s)', 'Disk', 'Request', 'Sector', and 'Length'. The table displays a list of disk requests, all of which are 'Read' operations on disk '0' with a duration of 0.00000000 seconds and a length of 16 bytes. The sectors listed range from 94976240 to 68131912.

#	Time	Duration (s)	Disk	Request	Sector	Length
112	2.150424	0.00000000	0	Read	94976240	16
113	2.150686	0.00000000	0	Read	68134904	16
114	2.150897	0.00000000	0	Read	94956880	16
115	2.151121	0.00000000	0	Read	94962048	16
116	2.151338	0.00000000	0	Read	94957648	16
117	2.151548	0.00000000	0	Read	94972128	16
118	2.151722	0.00000000	0	Read	94969488	16
119	2.151989	0.00000000	0	Read	68133416	16
120	2.152217	0.00000000	0	Read	94944032	16
121	2.152563	0.00000000	0	Read	94927216	16
122	2.152832	0.00000000	0	Read	68134872	16
123	2.153106	0.00000000	0	Read	68131912	16

Рисунок 2.18 – ПЗ DiskMon, приклад роботи

24. **HANDLE** (рекомендовано до використання) показує всі відкриті дескриптори (файли, ключі реєстру, об'єкти синхронізації тощо) для будь-якого процесу в системі. Реалізований таким чином, що дескриптор Handle це структура даних, яка представляє відкритий екземпляр об'єкта ОС (файлу, ключа реєстру, об'єкта синхронізації, тощо). Ідентифікує створений об'єкт і забезпечує доступ до нього, що у результаті дає змогу читати необхідні дані, змінювати їх стан об'єкта.

25. Hex2dec. Конвертує шестнадцяткові числа в десяткові.

26. Junction. Створює символічні посилання NTFS.

27. LDMDump. Вивантажує вміст бази даних логічного менеджера дисків.

28. ListDLLs. Відображає всі завантажені DLL.


```
Copyright (C) 1997-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

Handle type summary:
ALPC Port      : 1930
Desktop        : 235
Directory      : 664
EtwConsumer    : 6
EtwRegistration : 3568
Event          : 30642
File           : 3006
IoCompletion   : 373
Job            : 1
Key            : 6214
KeyedEvent     : 163
Mutant         : 179173
PcwObject      : 95
PowerRequest   : 1
Process        : 1354
Section        : 737
Semaphore      : 6880
Session        : 14
SymbolicLink   : 9
Thread         : 6351
Timer          : 726
TmRm           : 20
TmTm           : 10
Token          : 29106
TpWorkerFactory : 8992
WindowStation  : 454
WmiGuid        : 34
Total handles: 280758

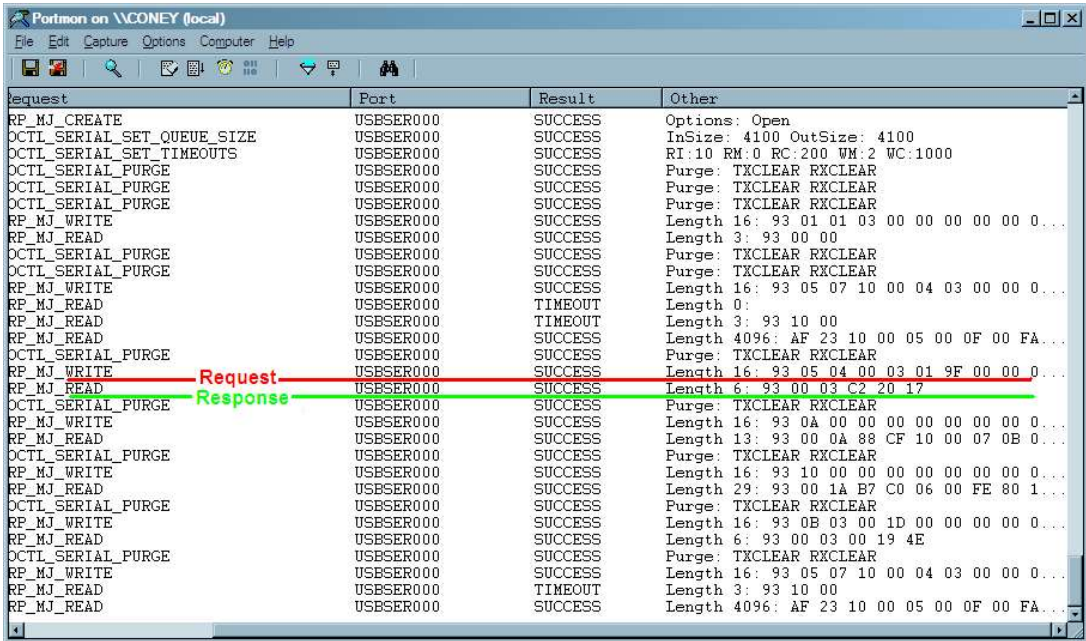
C:\Users\trlee\Desktop>handle -s

Handle v4.0
Copyright (C) 1997-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

Handle type summary:
ALPC Port      : 2344
Desktop        : 248
Directory      : 713
EtwConsumer    : 6
EtwRegistration : 3573
Event          : 22883
File           : 2805
IoCompletion   : 372
Job            : 1
Key            : 6413
KeyedEvent     : 163
Mutant         : 13679
PcwObject      : 95
PowerRequest   : 1
Process        : 1321
Section        : 741
Semaphore      : 7245
Session        : 14
SymbolicLink   : 9
Thread         : 6418
Timer          : 725
TmRm           : 20
TmTm           : 10
Token          : 29270
TpWorkerFactory : 8960
WindowStation  : 480
WmiGuid        : 34
Total handles: 108543
```

Рисунок 2.19 – ПЗ HANDLE, приклад роботи

29. LiveKd. Використовує відладчики ядра Microsoft для дослідження живої системи.
30. LoadOrder. Показує порядок завантаження пристроїв.
31. LogonSessions. Відображає активні сеанси входу.
32. MoveFile. Планує переміщення та видалення файлів при перезавантаженні.
33. NotMyFault. Створює збої, зависання та витоки пам'яті.
34. NTFSinfo. Відображає детальну інформацію про NTFS-томи.
35. PendMoves. Перераховує команди перейменування та видалення файлів.
36. PipeList. Відображає іменовані канали.
37. **PORTMON (рекомендовано до використання).** Утиліта, що моніторить і відображає всю активність на послідовних і паралельних портах системи. Вона має розширені можливості фільтрації та пошуку, що робить її потужним інструментом для дослідження роботи Windows, аналізу використання портів програмами та виявлення проблем у конфігураціях системи або додатків.



The screenshot shows the Portmon application window titled "Portmon on \\CONEX (local)". The window has a menu bar (File, Edit, Capture, Options, Computer, Help) and a toolbar. Below the toolbar is a table with four columns: Request, Port, Result, and Other. The table contains a list of USB requests and responses. A red line highlights a "Request" row, and a green line highlights a "Response" row. The "Request" row is "RP_MJ_WRITE" to "USB\\SER000" with "SUCCESS" result. The "Response" row is "OCTL_SERIAL_PURGE" from "USB\\SER000" with "SUCCESS" result. The "Other" column contains various details like "Options: Open", "InSize: 4100 OutSize: 4100", "RI:10 RM:0 RC:200 WM:2 WC:1000", "Purge: TXCLEAR RXCLEAR", and "Length" values.

Request	Port	Result	Other
RP_MJ_CREATE	USB\\SER000	SUCCESS	Options: Open
OCTL_SERIAL_SET_QUEUE_SIZE	USB\\SER000	SUCCESS	InSize: 4100 OutSize: 4100
OCTL_SERIAL_SET_TIMEOUTS	USB\\SER000	SUCCESS	RI:10 RM:0 RC:200 WM:2 WC:1000
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 01 01 03 00 00 00 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 3: 93 00 00
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 05 07 10 00 04 03 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	TIMEOUT	Length 0:
RP_MJ_READ	USB\\SER000	TIMEOUT	Length 3: 93 10 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 4096: AF 23 10 00 05 00 0F 00 FA 00 00 00 00 00 00 00 00
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 05 04 00 03 01 9F 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 6: 93 00 03 C2 20 17
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 13: 93 00 0A 88 CF 10 00 07 0B 00 00 00 00 00 00 00 00
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 29: 93 00 1A B7 C0 06 00 FE 80 10 00 00 00 00 00 00 00
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 0B 03 00 1D 00 00 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 6: 93 00 03 00 19 4E
OCTL_SERIAL_PURGE	USB\\SER000	SUCCESS	Purge: TXCLEAR RXCLEAR
RP_MJ_WRITE	USB\\SER000	SUCCESS	Length 16: 93 05 07 10 00 04 03 00 00 00 00 00 00 00 00 00
RP_MJ_READ	USB\\SER000	TIMEOUT	Length 3: 93 10 00
RP_MJ_READ	USB\\SER000	SUCCESS	Length 4096: AF 23 10 00 05 00 0F 00 FA 00 00 00 00 00 00 00 00

Рисунок 2.20 – ПЗ PORTMON, приклад роботи

38. ProcDump. Захоплює дампи процесів при пікових навантаженнях.

39. **PROCESS EXPLORER (рекомендовано до використання).**

Відображає інформацію про відкриті файли, реєстрові ключі та інші об'єкти процесів. Process Explorer надає інформацію про відкриті дескриптори та завантажені DLL-файли процесів. Відображення Process Explorer складається з двох вікон. Верхнє вікно показує список активних процесів з іменами облікових записів. Нижнє вікно відображає дескриптори або DLL-файли процесу, вибраного у верхньому вікні, залежно від режиму. Process Explorer має потужний пошук, який швидко показує, які процеси відкрили певні дескриптори або завантажили DLL-файли. Унікальні можливості Process Explorer дозволяють відстежувати проблеми з версіями DLL або витoki дескрипторів, а також отримувати уявлення про роботу Windows і додатків.

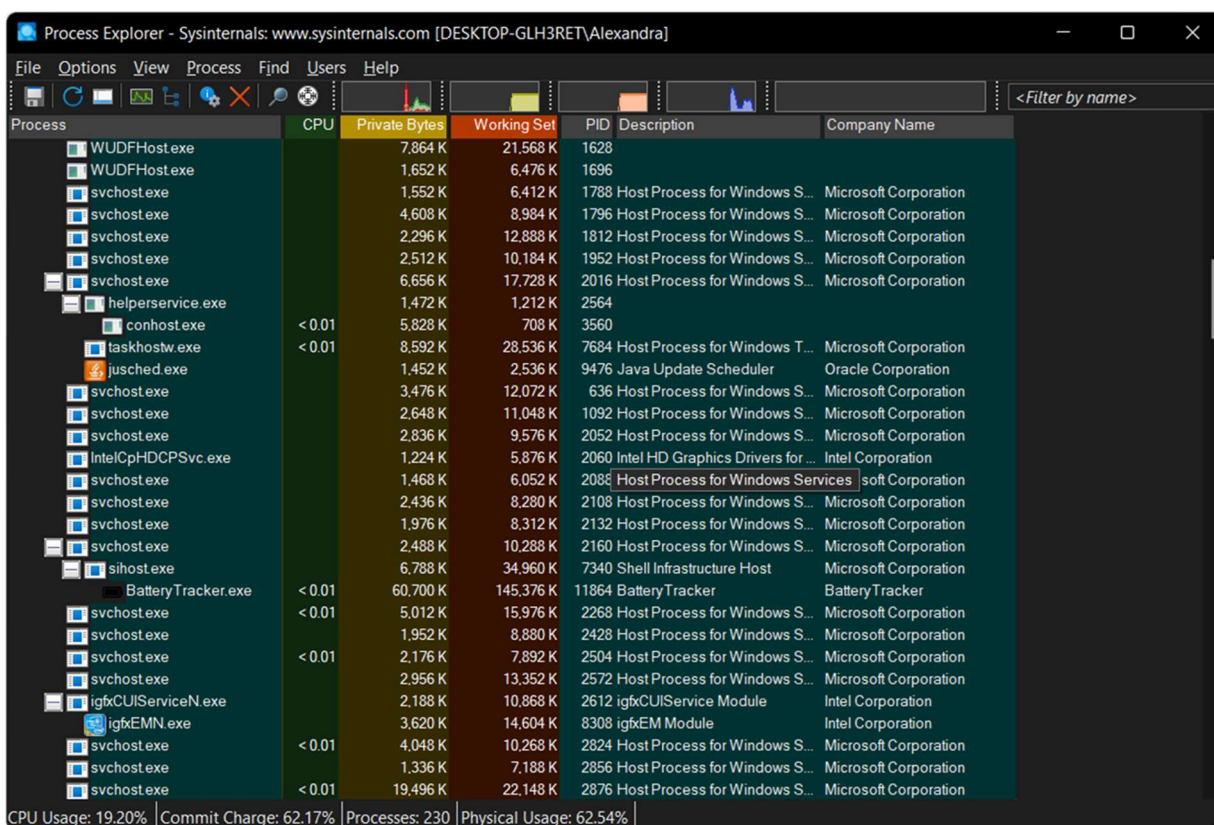


Рисунок 2.21 – ПЗ PROCESS EXPLORER, приклад роботи

40. PROCESS MONITOR (рекомендовано до використання).

Проводить моніторинг активності файлової системи, реєстру та процесів. Це вдосконалений інструмент моніторингу для Windows, який показує в реальному часі активність файлової системи, реєстру та процесів/ниток. Він поєднує функції старих утиліт Sysinternals, Filemon і Regmon, додаючи безліч поліпшень, таких як фільтр, розширені властивості подій, надійну інформацію про процеси, стеки ниток з інтегрованою підтримкою символів, одночасне логування у файл тощо. Унікально потужні функції роблять Process Monitor основним інструментом для діагностики системи та пошуку шкідливого ПЗ. Дозволяє створювати звіти які можна аналізувати в інших програмах, наприклад ProcDOT для складання дерева взаємодії файлів.

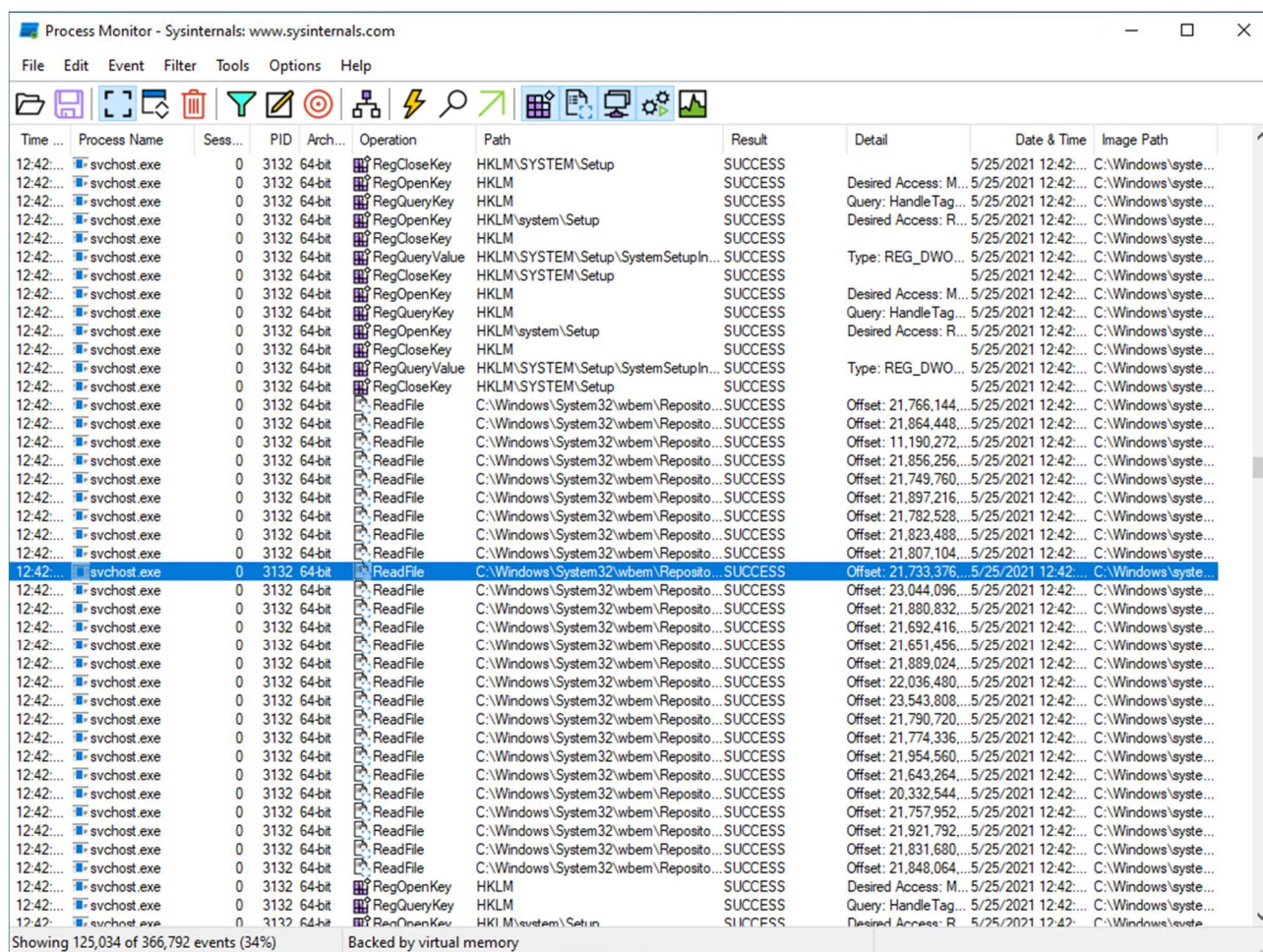


Рисунок 2.22 – ПЗ PROCESS MONITOR

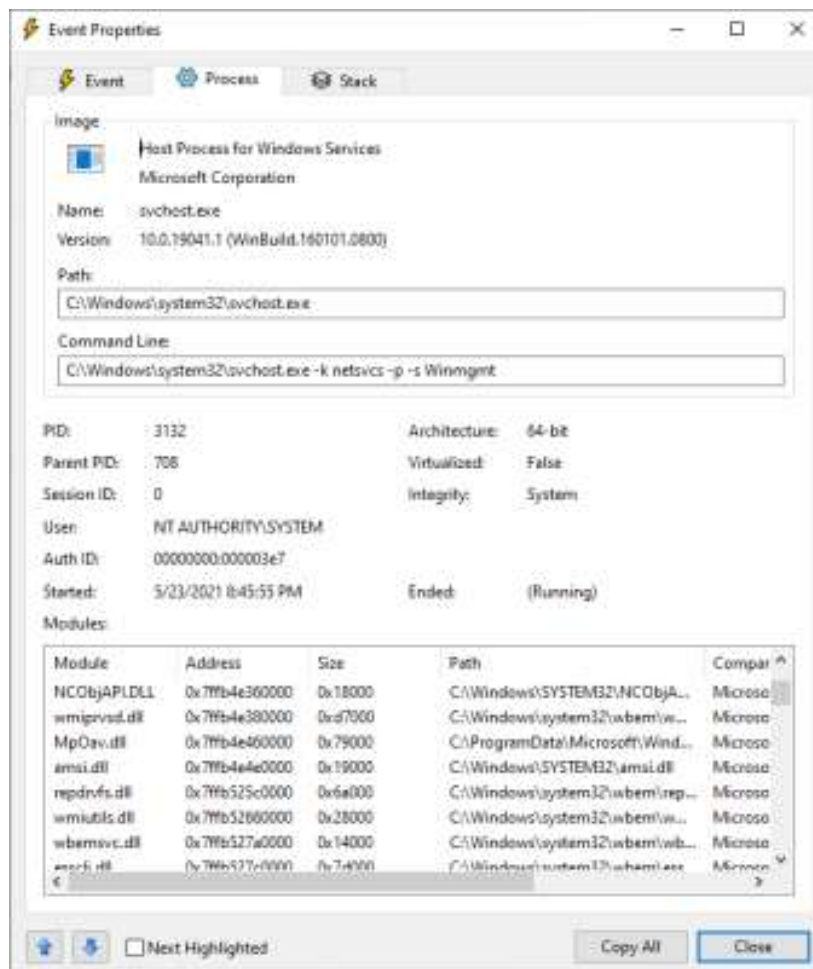


Рисунок 2.23 – ПЗ PROCESS MONITOR, відомості об'єкту дослідження

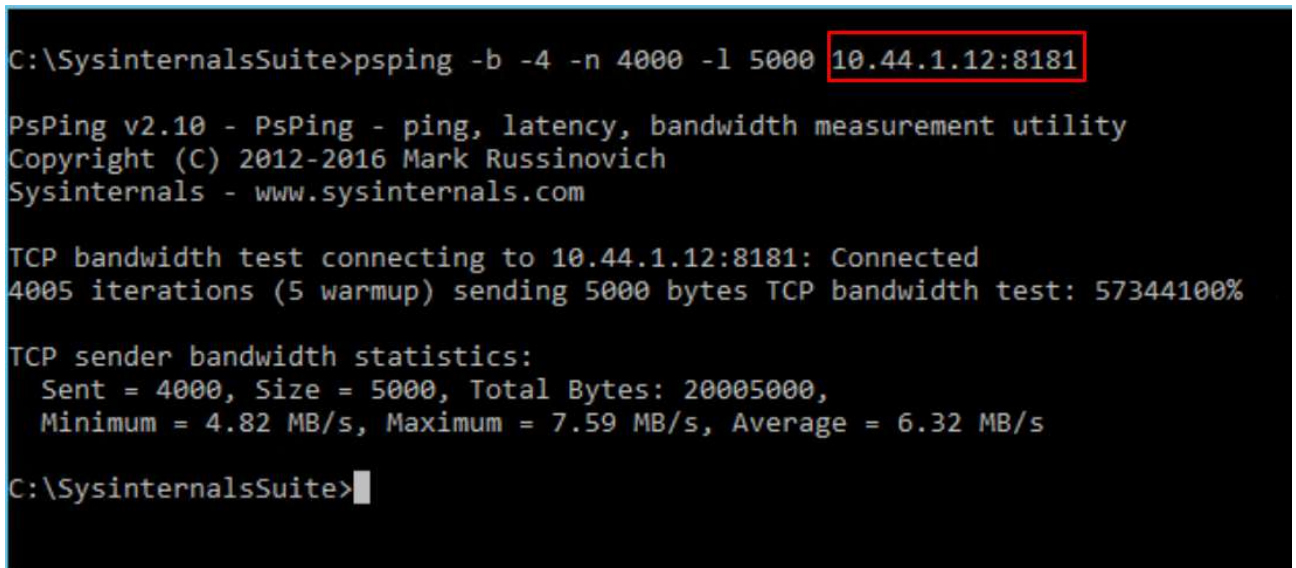
Process Monitor включає потужні можливості моніторингу та фільтрації, такі як:

- Збір більшої кількості даних для параметрів введення і виведення операцій.
- фільтри що не скидають результат, що дозволяють встановлювати фільтри без втрати даних.
- Захоплення стеків ниток для кожної операції, що дозволяє у багатьох випадках визначити першопричину операції.
- Надійне захоплення деталей процесу, включаючи шлях до зображення, командний рядок, користувача та ID сесії.
- Налаштовувані та переміщувані стовпці для будь-якої властивості події.

- Фільтри можна встановлювати для будь-якого поля даних, включаючи поля, не налаштовані як стовпці.
- Вдосконалена архітектура журналювання масштабується до десятків мільйонів захоплених подій і гігабайтів даних журналу.
- Інструмент дерева процесів показує взаємозв'язок усіх процесів, згаданих у трасуванні.
- Рідний формат журналу зберігає всі дані для завантаження в інший екземпляр Process Monitor.
- Підказка процесу для легкого перегляду інформації про зображення процесу.
- Підказка з деталями дозволяє зручно отримати доступ до форматованих даних, які не поміщаються у стовпець.
- Логування всіх операцій під час завантаження системи.

41. PsExec. Виконує процеси на віддалених системах.
42. PsFile. Показує, які файли відкриті віддалено.
43. PsGetSid. Відображає SID комп'ютера або користувача.
44. PsInfo. Отримує інформацію про систему.
45. PsKill. Завершує локальні або віддалені процеси.
46. **Ping (рекомендовано до використання).** Вимірює продуктивність мережі. Дозволяє визначити затримку мережі та пропускну здатність. Дозволяє пінгувати порти.
47. PsList. Відображає інформацію про процеси та нитки.
48. PsLoggedOn. Показує користувачів, що увійшли в систему.
49. PsLogList. Вивантажує записи журналу подій.
50. PsPasswd. Змінює паролі облікових записів.
51. PsService. Відображає та керує сервісами.
52. PsShutdown. Виконує завершення роботи або перезавантаження комп'ютера.
53. PsSuspend. Призупиняє та відновлює процеси.

54. PsTools. Набір утиліт для управління процесами на локальних та віддалених комп'ютерах.



```
C:\SysinternalsSuite>psping -b -4 -n 4000 -l 5000 10.44.1.12:8181

PsPing v2.10 - PsPing - ping, latency, bandwidth measurement utility
Copyright (C) 2012-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

TCP bandwidth test connecting to 10.44.1.12:8181: Connected
4005 iterations (5 warmup) sending 5000 bytes TCP bandwidth test: 57344100%

TCP sender bandwidth statistics:
  Sent = 4000, Size = 5000, Total Bytes: 20005000,
  Minimum = 4.82 MB/s, Maximum = 7.59 MB/s, Average = 6.32 MB/s

C:\SysinternalsSuite>
```

Рисунок 2.24 – ПЗ PsPing, приклад використання

- 55. RAMMap. Аналіз використання фізичної пам'яті.
- 56. RDCMan. Управління кількома підключеннями віддаленого робочого столу.
- 57. RegDelNull. Видаляє реєстрові ключі з нульовими символами.
- 58. Registry Usage (RU). Відображає використання реєстру.
- 59. RegJump. Переміщається до вказаного шляху реєстру у Regedit.
- 60. SDelete. Надійне видалення файлів та очищення вільного простору диска.
- 61. ShareEnum. Сканує загальні ресурси в мережі та відображає їх налаштування безпеки.
- 62. ShellRunas. Запуск програм від імені іншого користувача через контекстне меню.
- 63. Sigcheck. Перевірка цифрових підписів та версій файлів.
- 64. Streams. Виявляє альтернативні потоки NTFS.

65. **Strings.** Пошук ANSI та UNICODE рядків у бінарних образах. Дозволяє детально переглядати строкові дані різних типів кодувань.

```
C:\Users\admin
strings

Strings v2.53 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

usage: C:\ProgramData\chocolatey\lib\sysinternals\tools\strings.exe [-a] [-f offset] [-b byte
s] [-n length] [-o] [-s] [-u] <file or directory>
-a      Ascii-only search (Unicode and Ascii is default)
-b      Bytes of file to scan
-f      File offset at which to start scanning.
-o      Print offset in file string was located
-n      Minimum string length (default is 3)
-s      Recurse subdirectories
-u      Unicode-only search (Unicode and Ascii is default)
-nobanner
        Do not display the startup banner and copyright message.
```

Рисунок 2.25 – ПЗ Strings, параметри командного рядка

66. **Sync.** Скидає кешовані дані на диск.

67. **Sysmon (рекомендовано до використання).**

Дозволяє проводити моніторинг створення процесів, мережеві з'єднання, зміни файлів та інші важливі події, допомагає виявляти підозрілу активність в системі.

У результаті журналювання таких подій з використанням (Windows Event Collection, SIEM) дає змогу проводити детальний аналіз та виявлення зловмисної активності в системі. Крім того дозволяє алгоритм дій та краще розуміти кроки яким чином шкідливе ПЗ діє у локальному середовищі.

Дозволяє створювати практичні гнучкі звіти для подальшого аналізу чи експорту даних в інше ПЗ. Має гнучку систему часу запису активностей в системі з можливістю ручного керування процесом.

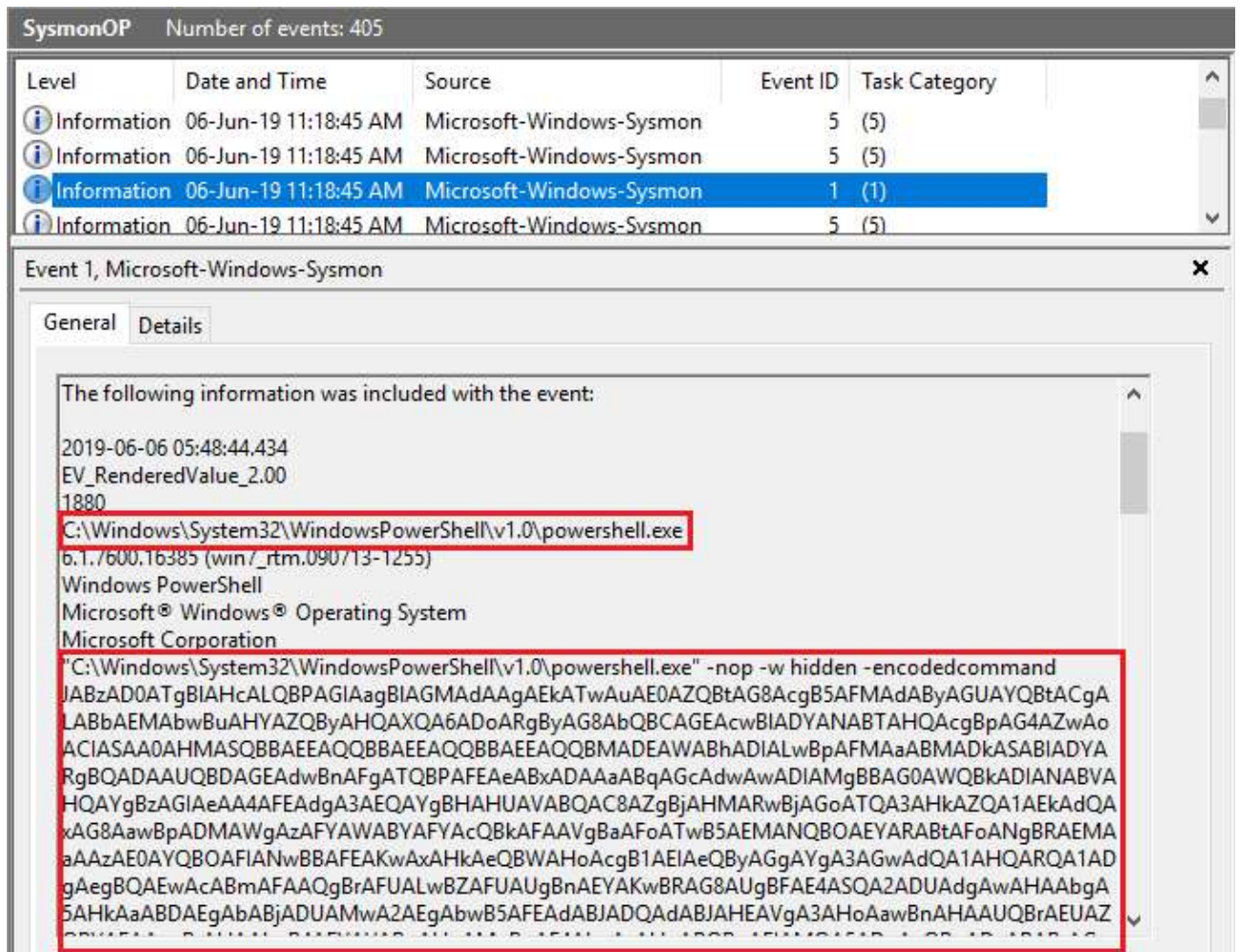


Рисунок 2.26 – ПЗ Sysmon, приклад використання

68. **TCPView (рекомендовано до використання).** Відображення активних сокетів. ПЗ відображає детальні списки всіх TCP та UDP кінцевих точок на вашій системі, включаючи локальні та віддалені адреси, а також стан TCP-з'єднань.

Вона також показує назву процесу, який використовує кінцеву точку. TCPView пропонує більш інформативну та зручну версію функцій, доступних у програмі Netstat, що постачається з Windows. У комплект входить Tcprvcon, версія командного рядка з такою ж функціональністю.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name	Sent Packets
QualysAgent.exe	1512	TCP	Established	host.docker.internal	1131	qagpublic.qg3.apps.qua...	https	03/19/21 21:05:35.149	QualysAgent	4
OUTLOOK.EXE	27004	TCP	Established	host.docker.internal	1129	40.101.92.194	https	03/19/21 21:05:35.649	OUTLOOK.EXE	3
devenv.exe	15256	TCP	Established	host.docker.internal	1126	51.107.59.180	https	03/19/21 21:05:31.734	devenv.exe	
Teams.exe	26092	TCP	Established	host.docker.internal	1123	52.114.92.151	https	03/19/21 21:05:16.124	Teams.exe	7
OUTLOOK.EXE	27004	TCP	Established	host.docker.internal	1117	52.114.128.71	https	03/19/21 21:05:04.083	OUTLOOK.EXE	6
devenv.exe	15256	TCP	Established	host.docker.internal	1116	lb-140-82-121-5-fra.gith...	https	03/19/21 21:05:03.265	devenv.exe	3
devenv.exe	15256	TCP	Close Wait	kubernetes.docker.inter...	1106	kubernetes.docker.inter...	1112	03/19/21 21:05:01.430	devenv.exe	6
Microsoft.Alm.Shared...	50384	Fin Wait 2	Listen	kubernetes.docker.inter...	1112	kubernetes.docker.inter...	1106	03/19/21 21:05:01.010	Microsoft.Alm.Shared...	6
Microsoft.Alm.Shared...	50384	TCP	Listen	kubernetes.docker.inter...	1111	0.0.0.0	0	03/19/21 21:05:01.680	Microsoft.Alm.Shared...	
devenv.exe	15256	TCP	Listen	kubernetes.docker.inter...	1106	0.0.0.0	0	03/19/21 21:04:58.789	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1105	13.66.38.99	https	03/19/21 21:04:55.470	devenv.exe	5
devenv.exe	15256	TCP	Established	host.docker.internal	1104	13.66.241.134	https	03/19/21 21:04:54.104	devenv.exe	5
devenv.exe	15256	TCP	Established	host.docker.internal	1103	13.77.157.133	https	03/19/21 21:04:53.935	devenv.exe	5
ServiceHub.IdentityHo...	17188	TCP	Established	host.docker.internal	1102	51.107.59.180	https	03/19/21 21:04:52.098	ServiceHub.IdentityHo...	8
firefox.exe	3604	TCP	Established	host.docker.internal	1101	51.107.59.180	https	03/19/21 21:04:52.682	firefox.exe	2
devenv.exe	15256	TCP	Established	host.docker.internal	1099	13.107.42.18	https	03/19/21 21:04:51.587	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1098	13.107.42.20	https	03/19/21 21:04:51.107	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1097	13.107.42.18	https	03/19/21 21:04:50.454	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1096	13.107.42.20	https	03/19/21 21:04:50.869	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1095	13.107.42.18	https	03/19/21 21:04:50.445	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1093	13.107.42.20	https	03/19/21 21:04:50.260	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1092	13.107.42.18	https	03/19/21 21:04:50.460	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1091	13.107.42.20	https	03/19/21 21:04:49.818	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1090	13.107.42.18	https	03/19/21 21:04:49.227	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1089	13.107.42.20	https	03/19/21 21:04:49.767	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1088	13.107.42.18	https	03/19/21 21:04:49.681	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1087	13.107.42.18	https	03/19/21 21:04:49.352	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1084	13.107.42.20	https	03/19/21 21:04:48.252	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1083	13.107.42.18	https	03/19/21 21:04:48.793	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1082	13.107.42.18	https	03/19/21 21:04:48.682	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1081	13.107.42.20	https	03/19/21 21:04:48.775	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1080	13.107.42.18	https	03/19/21 21:04:48.580	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1079	13.107.42.20	https	03/19/21 21:04:48.957	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1078	13.107.42.18	https	03/19/21 21:04:48.994	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1077	13.107.42.18	https	03/19/21 21:04:47.460	devenv.exe	
devenv.exe	15256	TCP	Established	host.docker.internal	1076	13.107.42.20	https	03/19/21 21:04:47.063	devenv.exe	

Endpoints: 256 Established: 175 Listening: 64 Time Wait: 9 Close Wait: 6 Update: 2 sec

Рисунок 2.27 – ПЗ TCPView, приклад використання

69. VMMap (рекомендовано до використання). Аналіз віртуальної та фізичної пам'яті процесу. Дозволяє аналізувати віртуальну та фізичну пам'ять процесів. Вона показує розподіл типів віртуальної пам'яті, що використовується процесом, а також обсяг фізичної пам'яті, призначений операційною системою для цих типів. Окрім графічного відображення використання пам'яті, VMMap також надає зведену інформацію та детальну карту пам'яті процесу. Потужні можливості фільтрації та оновлення дозволяють визначати джерела використання пам'яті процесу та вартість пам'яті для функцій програми.

VMMap підтримує гнучкі вхідні дані для аналізу процесів, а також експорт даних у різних форматах, включаючи рідний формат, що зберігає всю інформацію для подальшого завантаження. Також включені параметри командного рядка для сценаріїв скриптів.

VMMap є ідеальним інструментом для розробників, які прагнуть зрозуміти та оптимізувати використання ресурсів пам'яті своїми додатками чи проаналізувати використання пам'яті шкідливим ПЗ.

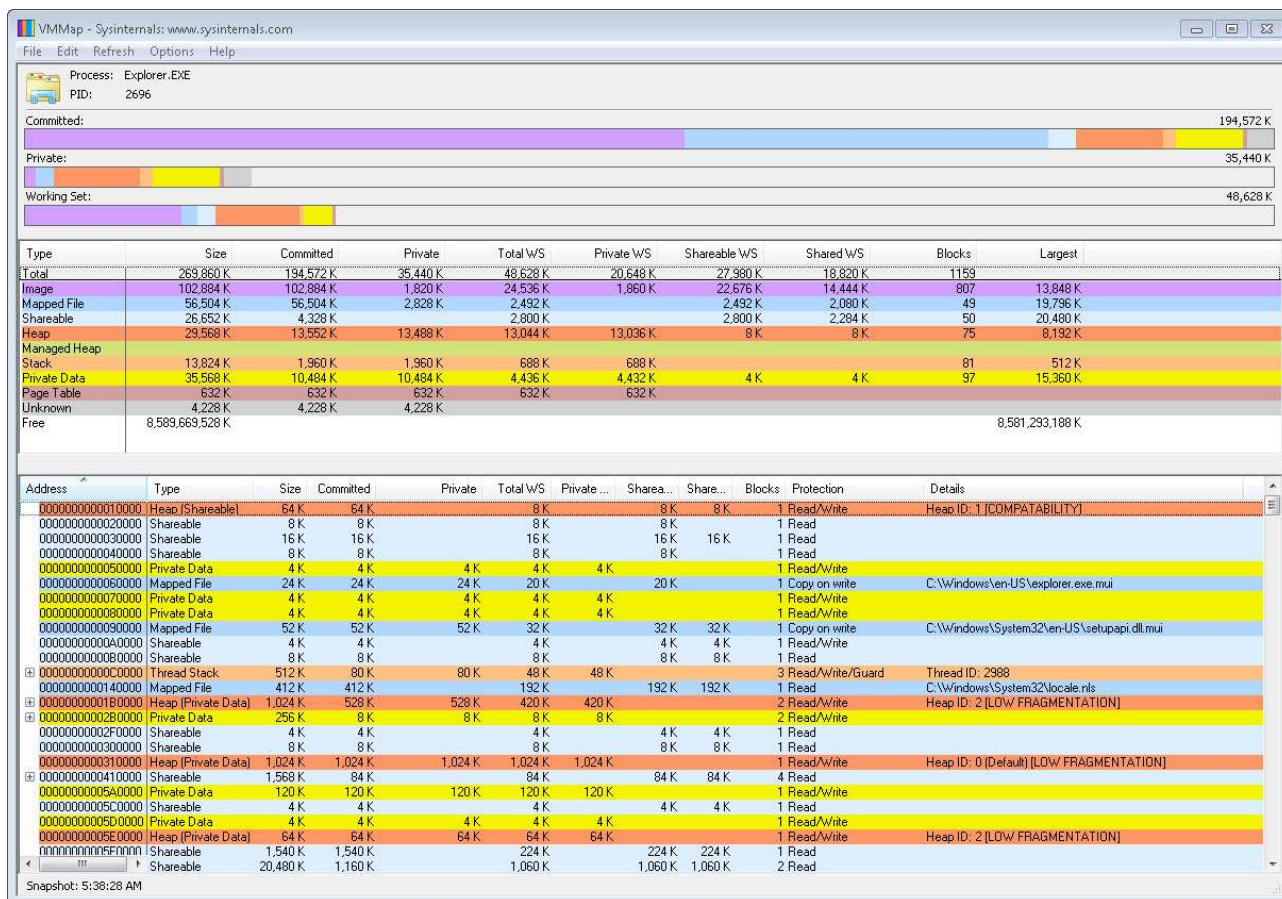


Рисунок 2.28 – VMMap, приклад використання

70. VolumeId. Встановлення ID томів FAT або NTFS дисків.
71. Whois. Показує власника інтернет-адреси.
72. WinObj. Перегляд ієрархії простору імен Object Manager.
73. ZoomIt. Утиліта для масштабування та малювання на екрані під час презентацій.

ПРИКЛАДИ ВИКОРИСТАННЯ

3 точки зору реагування на інциденти.

1. За допомогою утиліт Process Explorer і Autoruns можна швидко виявляти підозрілу активність, несанкціоновані програми та процеси, які запускаються під час старту операційної системи.

Наприклад, Process Explorer може використовуватися для виявлення зловмисного процесу, який маскується під легітимний додаток.

З точки зору оптимізації продуктивності системи.

2. Утиліти Process Monitor і RAMMap дають змогу виконувати детальний моніторинг використання системних ресурсів, аналізувати роботу процесів і оптимізувати функціонування додатків.

Наприклад, RAMMap може застосовуватися для визначення процесів, які споживають надмірний обсяг оперативної пам'яті.

З точки зору аналізу мережевої активності.

3. TCPView використовується для відстеження активних TCP і UDP з'єднань, що допомагає виявляти підозрілу мережеву активність.

Наприклад, за допомогою TCPView можна встановити наявність нетипових підключень до віддалених серверів, які можуть свідчити про роботу шкідливого програмного забезпечення.

ВИСНОВКИ

Набір Sysinternals Utilities дуже корисний для адміністраторів, розробників і фахівців з безпеки:

1. Широкі можливості моніторингу процесів, пам'яті, автозапуску, мережі та діагностики системи.

2. Розширені можливості налаштування кожної утиліти, розширені можливості фільтрації для потреб розслідування інцидентів користувача.

3. Інтеграція з іншими інструментами - Sysinternals Utilities легко інтегруються з іншими засобами діагностики та аналізу, що підвищує ефективність роботи.

Як недолік можна зазначити значну кількість застарілих системних утиліт в яких зараз немає потреби.

2.6. ПЗ WIRESHARK

ЛІЦЕНЗІЯ

GNU General Public License 2.0 (GPL). Wireshark є відкритим програмним забезпеченням і поширюється під ліцензією GPL. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму як для особистих, так і для комерційних цілей.

РОЗРОБНИК

Wireshark був розроблений Джеральдом Комбсом.

ОФІЦІЙНИЙ РЕСУРС

<https://www.wireshark.org/>

<https://github.com/wireshark/wireshark>

ОПИС

Wireshark високоефективний інструмент для аналізу мережевого трафіку, що дає змогу перехоплювати й досліджувати дані, які проходять через мережу в режимі реального часу [69,77]. Він є одним з найпопулярніших аналізаторів пакетів, що використовується як професіоналами, так і ентузіастами для діагностики проблем у мережах, забезпечення безпеки та навчальних цілей.

Захоплення трафіку

Реальний час – Wireshark дозволяє захоплювати трафік у реальному часі з різних інтерфейсів мережевих адаптерів.

Підтримка різних протоколів – ПЗ підтримує тисячі мережевих протоколів, що дозволяє детально аналізувати будь-який тип трафіку.

Аналіз пакетів

Детальний огляд ПЗ надає можливість детально розглядати кожен захоплений пакет, включаючи його заголовки та вміст.

Фільтрація та пошук у Wireshark дають змогу швидко знаходити потрібну інформацію серед великого обсягу мережевих даних, що особливо корисно під час аналізу складних інцидентів або перевірки роботи окремих протоколів.

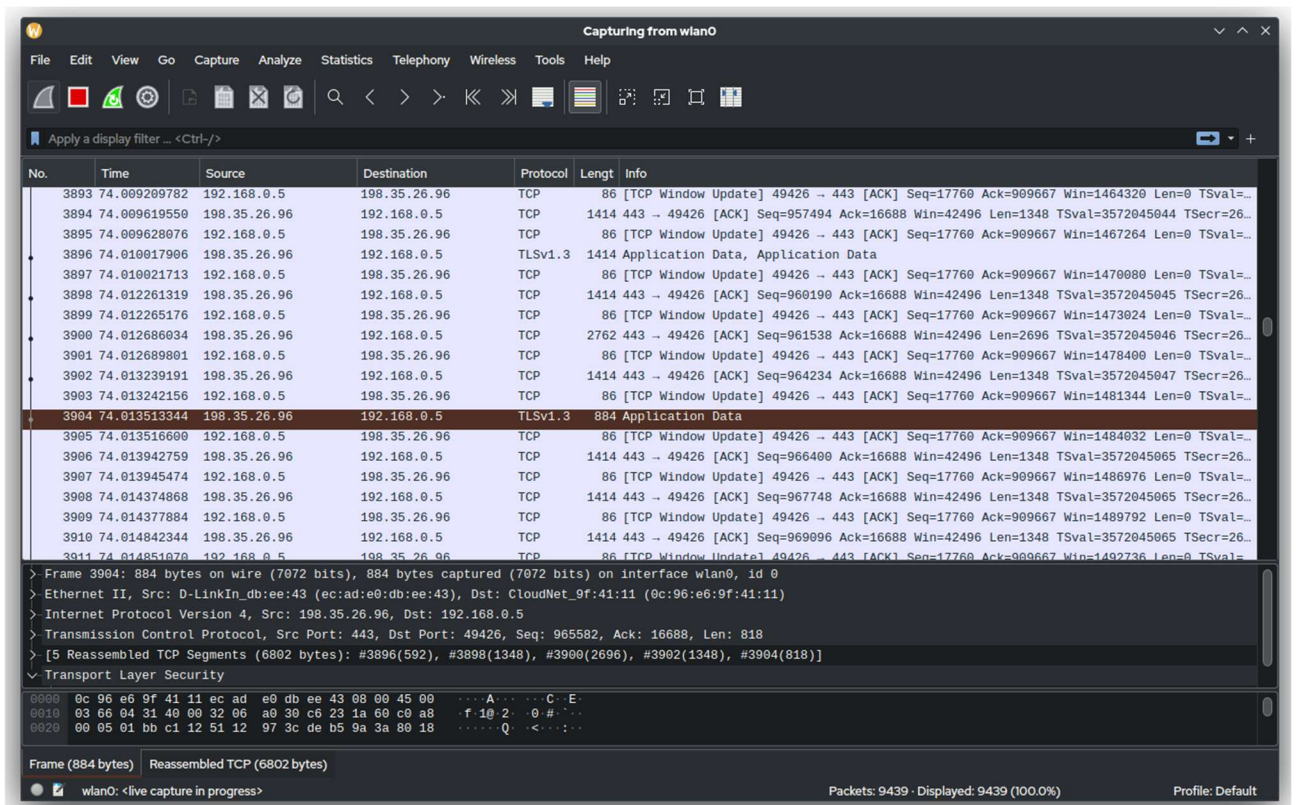


Рисунок 2.29 – Головне вікно ПЗ Wireshark

Візуалізація даних

Графічне представлення даних у Wireshark допомагає візуалізувати мережевий трафік за допомогою графіків і діаграм, що спрощує виявлення аномалій, затримок, перевантажень та інших проблем у роботі мережі.

Розширення ПЗ підтримує додаткові плагіни та розширення, що розширюють його функціональність.

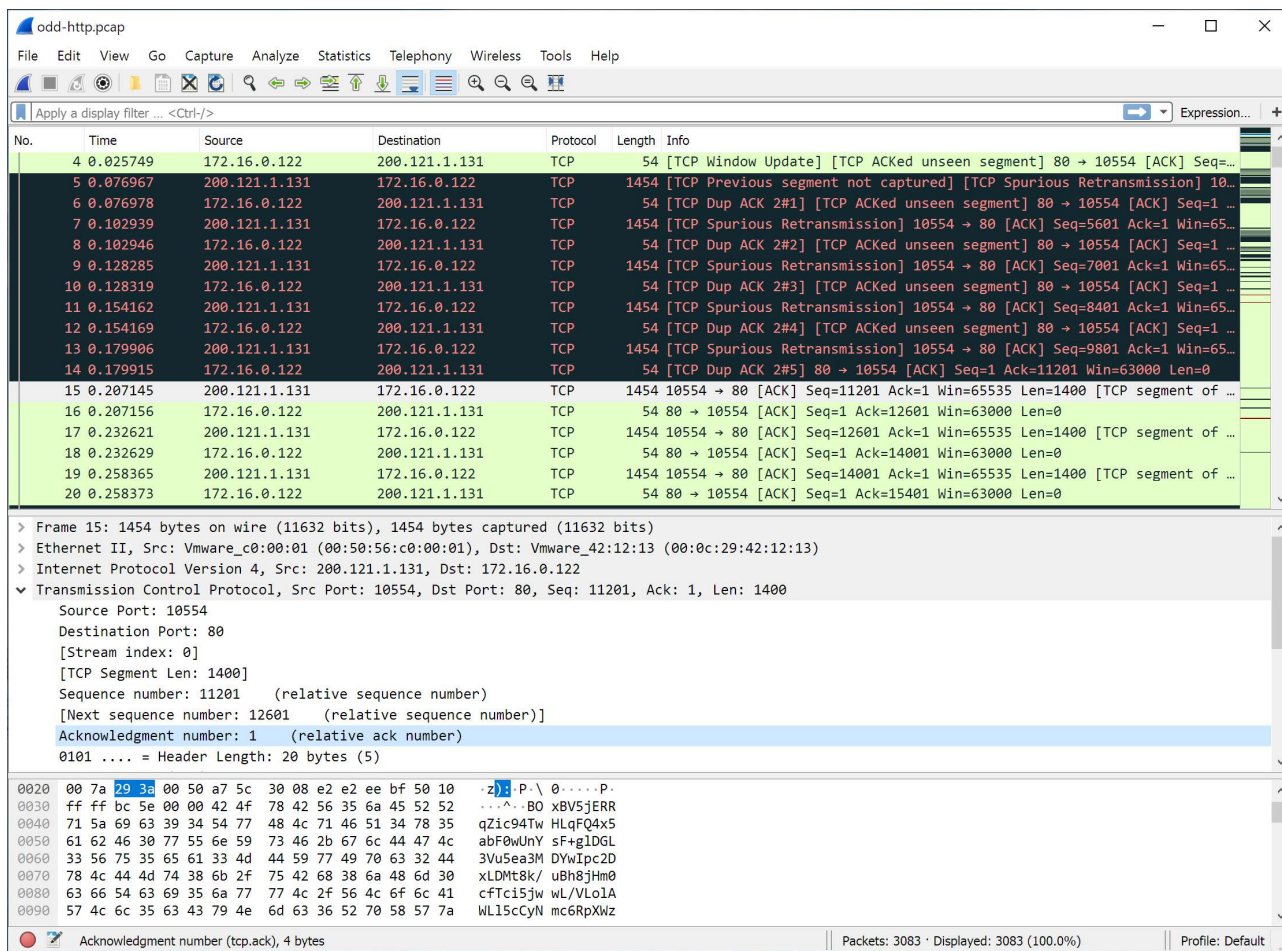


Рисунок 2.30 – ПЗ Wireshark, захват пакетів даних

Експорт даних

Різні формати - Захоплені дані можна експортувати у різні формати, включаючи XML, CSV та JSON, для подальшого аналізу або звітності.

Інтеграція – Wireshark легко інтегрується з іншими інструментами та системами для комплексного аналізу мережевої безпеки.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору діагностики мережевих проблем.

1. Wireshark використовується для діагностики мережевих несправностей, зокрема втрати пакетів, затримок передавання даних або помилок конфігурації. Наприклад, адміністратор мережі може застосувати Wireshark для аналізу трафіку між двома серверами, щоб визначити причину повільної передачі даних.

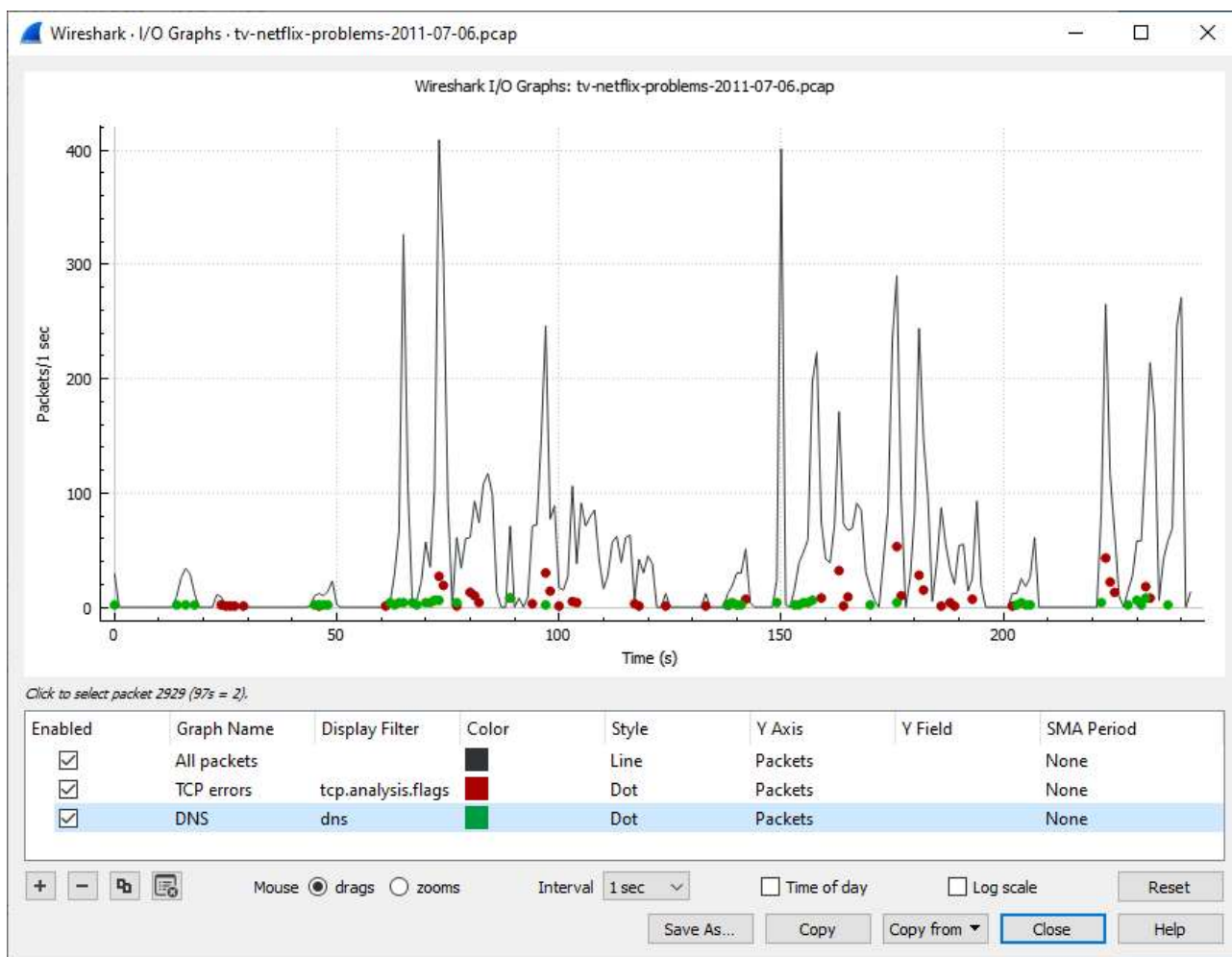


Рисунок 2.31 – ПЗ Wireshark, вікно графіків введення/виведення пакетів даних

З точки зору забезпечення безпеки.

2. Wireshark допомагає виявляти та аналізувати підозрілий трафік, який може свідчити про роботу шкідливого програмного забезпечення або проведення атаки. Наприклад, фахівець з безпеки використовує Wireshark для дослідження аномальної мережевої активності, яка може бути пов'язана з атакою типу man in the middle.

З точки зору навчальних цілей

3. Wireshark широко застосовується у навчальних закладах для вивчення мережевих протоколів, принципів роботи TCP/IP та методів аналізу трафіку. Наприклад, студенти використовують Wireshark під час лабораторних робіт для детального дослідження структури пакетів і взаємодії мережевих протоколів.

ВИСНОВКИ

Wireshark є ефективним інструментом для аналізу мережевого трафіку, який забезпечує можливості для діагностики, підвищення рівня безпеки та навчання:

1. Wireshark підтримує тисячі мережевих протоколів, що дозволяє детально аналізувати будь-який тип трафіку.
2. Програма забезпечує захоплення трафіку в реальному часі та глибокий аналіз пакетів.
3. Wireshark підтримує додаткові плагіни та розширення, а також має потужні інструменти фільтрації та пошуку.

2.7. ПЗ ProcDOT

ЛІЦЕНЗІЯ

Freeware. ProcDOT є безкоштовним програмним забезпеченням, яке можна використовувати для особистих і професійних цілей без плати за ліцензію. Програма доступна для завантаження безкоштовно і не вимагає придбання додаткових ліцензій для повного функціоналу.

РОЗРОБНИК

ProcDOT був розроблений Cuckoo Malware Analysis System.

ОФІЦІЙНИЙ РЕСУРС

<https://www.procdot.com/>

ОПИС

ProcDOT є інструментом для візуалізації та аналізу поведінки процесів у системі, який поєднує дані з різних джерел, зокрема лог файлів Procmon і мережевих дамів PCAP, та формує інтерактивне графічне представлення подій.

Це допомагає виявляти аномалії, аналізувати активність шкідливого програмного забезпечення та досліджувати взаємозв'язки між процесами, файлами, реєстром і мережевими з'єднаннями.

ProcDOT є корисним інструментом для дослідників безпеки та фахівців з аналізу шкідливого ПЗ, оскільки забезпечує наочне подання поведінки процесів і полегшує виявлення підозрілих дій у системі.

Можливості програми

Об'єднання даних ProcDOT інтегрує дані з лог-файлів Procmon та мережевих дамів PCAP для комплексного аналізу подій. Візуалізація – створює детальні графічні представлення процесів та мережевої активності, що дозволяє легко ідентифікувати підозрілі дії. Виявлення аномалій допомагає виявляти незвичайну активність, яка може свідчити про наявність шкідливого програмного забезпечення.

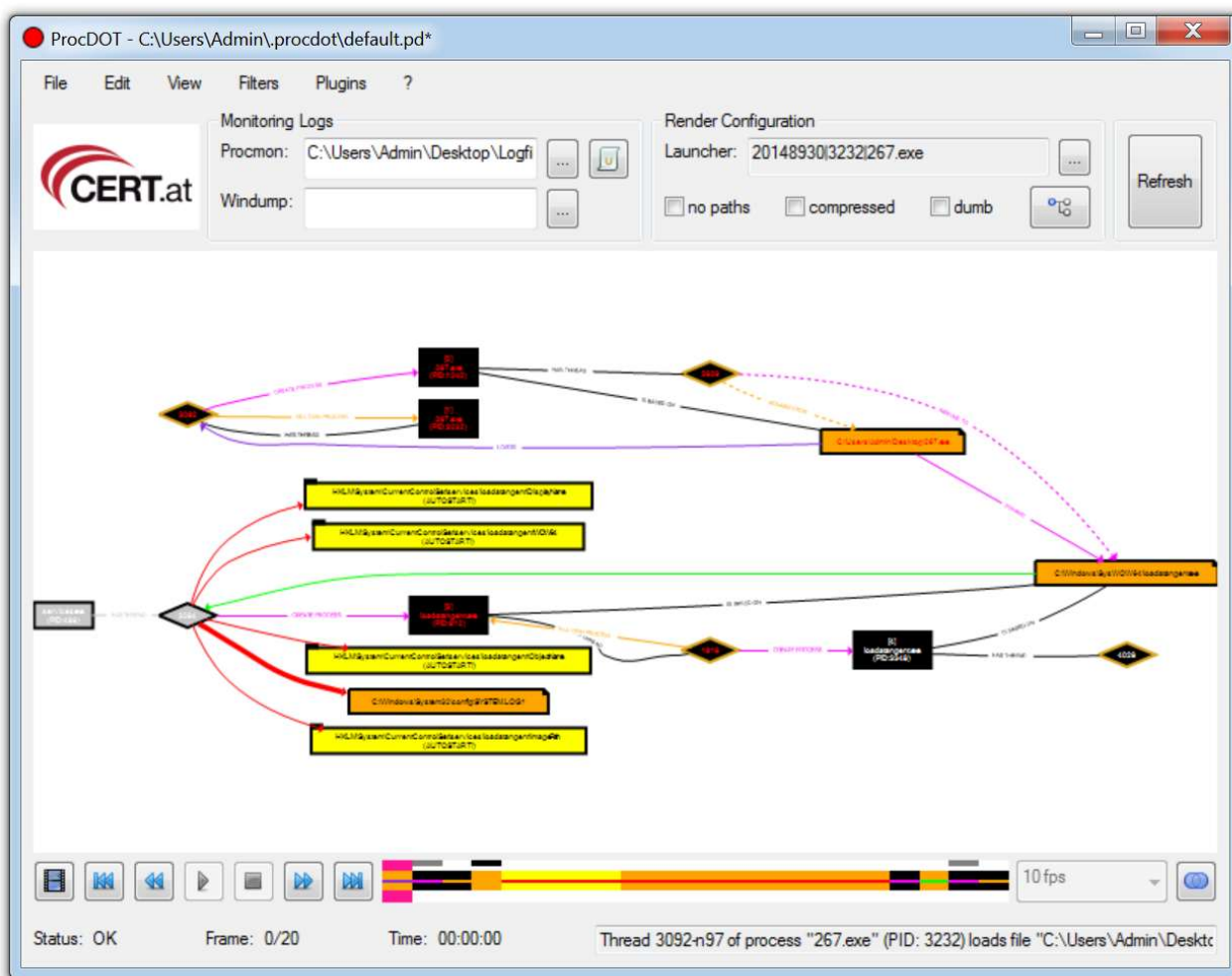


Рисунок 2.32 – Головне вікно ПЗ ProcDOT

Взаємозв'язок подій, аналізує взаємозв'язки між процесами та мережевими з'єднаннями для кращого розуміння їхньої поведінки.

Інтерактивний інтерфейс, забезпечує зручний і зрозумілий інтерфейс для аналізу та візуалізації даних.

Експорт даних, дозволяє експортувати результати аналізу у різних форматах для подальшого використання та звітності.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ

1. ProcDOT дозволяє дослідникам візуалізувати активність шкідливого програмного забезпечення, об'єднуючи дані про процеси та мережеву активність.

Як приклад спеціаліст з безпеки використовує ProcDOT для аналізу підозрілого процесу, зібраного за допомогою Procmon, і виявляє, що процес встановлює з'єднання з відомими зловмисними ІР-адресами.

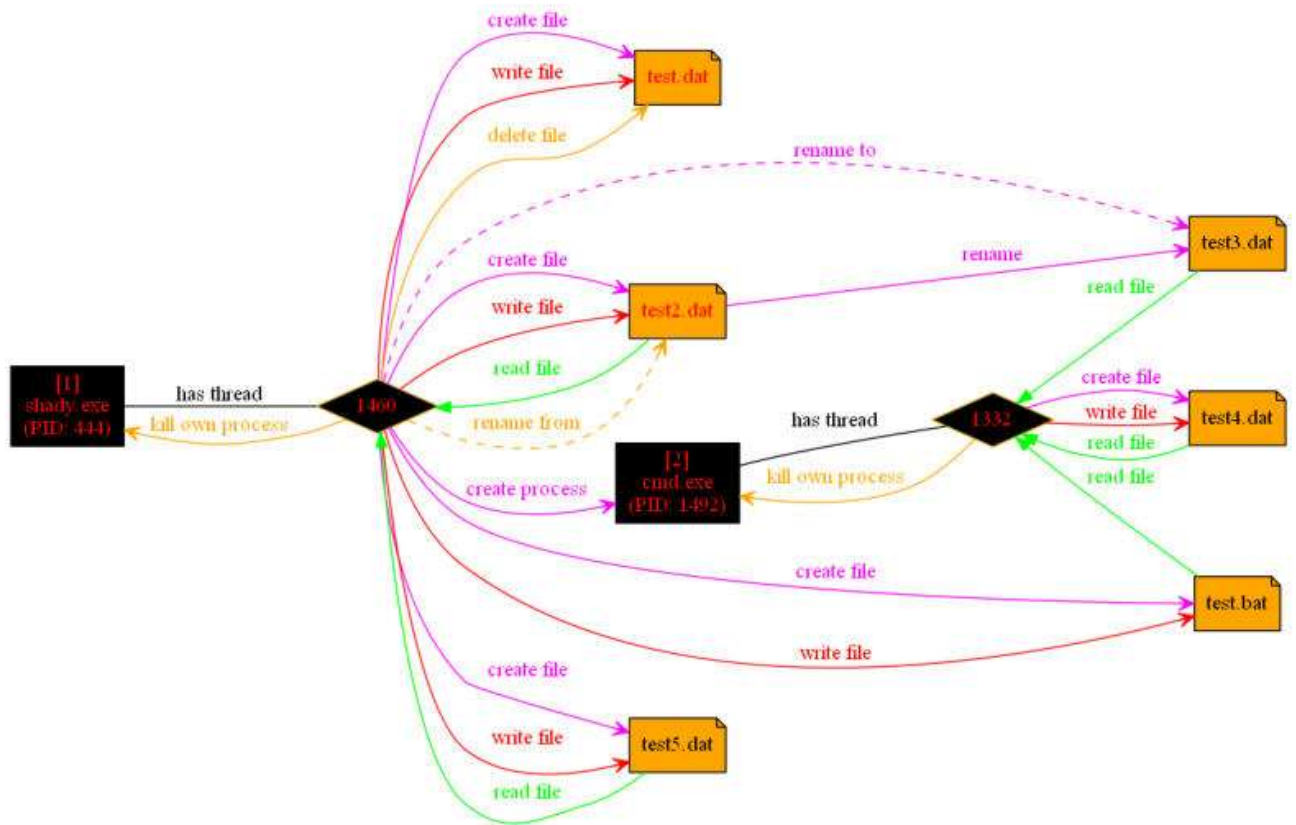


Рисунок 2.33 – Головне вікно ПЗ ProcDOT, сформована мапа взаємодії процесів об'єкта що досліджується

З точки зору розслідування інцидентів.

2. ProcDOT допомагає швидко виявляти та аналізувати аномальну активність, що дає змогу оперативно реагувати на інциденти. Наприклад, під час розслідування мережевого злому дослідник використовує ProcDOT для візуалізації процесів, файлових операцій і мережевих з'єднань, що допомагає встановити джерело підозрілої активності та визначити подальші дії для її нейтралізації.

З точки зору навчання та дослідження.

3. ProcDOT використовується у навчальних закладах для навчання студентів основам аналізу поведінки програм та мережевої безпеки. Як приклад ProcDOT використовується під час лабораторних занять, щоб показати студентам, як візуалізувати та аналізувати процеси та мережеву активність на прикладі симуляції атаки.

ВИСНОВКИ

ProcDOT є потужним інструментом для візуалізації та аналізу процесів і мережевої активності. Основні переваги включають:

1. Об'єднання даних з різних джерел для створення комплексного аналізу подій.
2. Допомогає швидко ідентифікувати незвичайну активність та шкідливе ПЗ.
3. Інтерактивний інтерфейс спрощує процес аналізу та візуалізації даних.

2.8. ПЗ GHIDRA

ЛІЦЕНЗІЯ

Apache License 2.0. Ghidra є безкоштовним програмним забезпеченням з відкритим вихідним кодом, що поширюється під ліцензією Apache 2.0. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, включаючи використання в комерційних цілях.

РОЗРОБНИК

Ghidra була розроблена Агентством національної безпеки США (NSA).

ОФІЦІЙНИЙ РЕСУРС

<https://ghidra-sre.org/>

ОПИС

Ghidra потужний інструмент для аналізу зворотної розробки програмного забезпечення, розроблений Агентством національної безпеки США (NSA). Ghidra надає дослідникам можливість аналізувати компільований код на предмет виявлення вразливостей, зловмисних модифікацій та загальних досліджень зворотної інженерії [57-58,78]. Ghidra є важливим інструментом для дослідників безпеки, інженерів зворотної розробки та розробників програмного забезпечення, оскільки надає широкі можливості для аналізу, декомпіляції та дослідження програмного коду, а також підтримує спільну роботу і розширення функціональності за допомогою скриптів.

Можливості програми

Декомпіляція – Ghidra дозволяє декомпілювати машинний код у зрозумілий для людини формат, що допомагає аналізувати програмне забезпечення на низькому рівні у форматі мов програмування ASM та C.

Підтримка різних архітектур – підтримує велику кількість архітектур, включаючи x86, x64, ARM, PowerPC та інші.

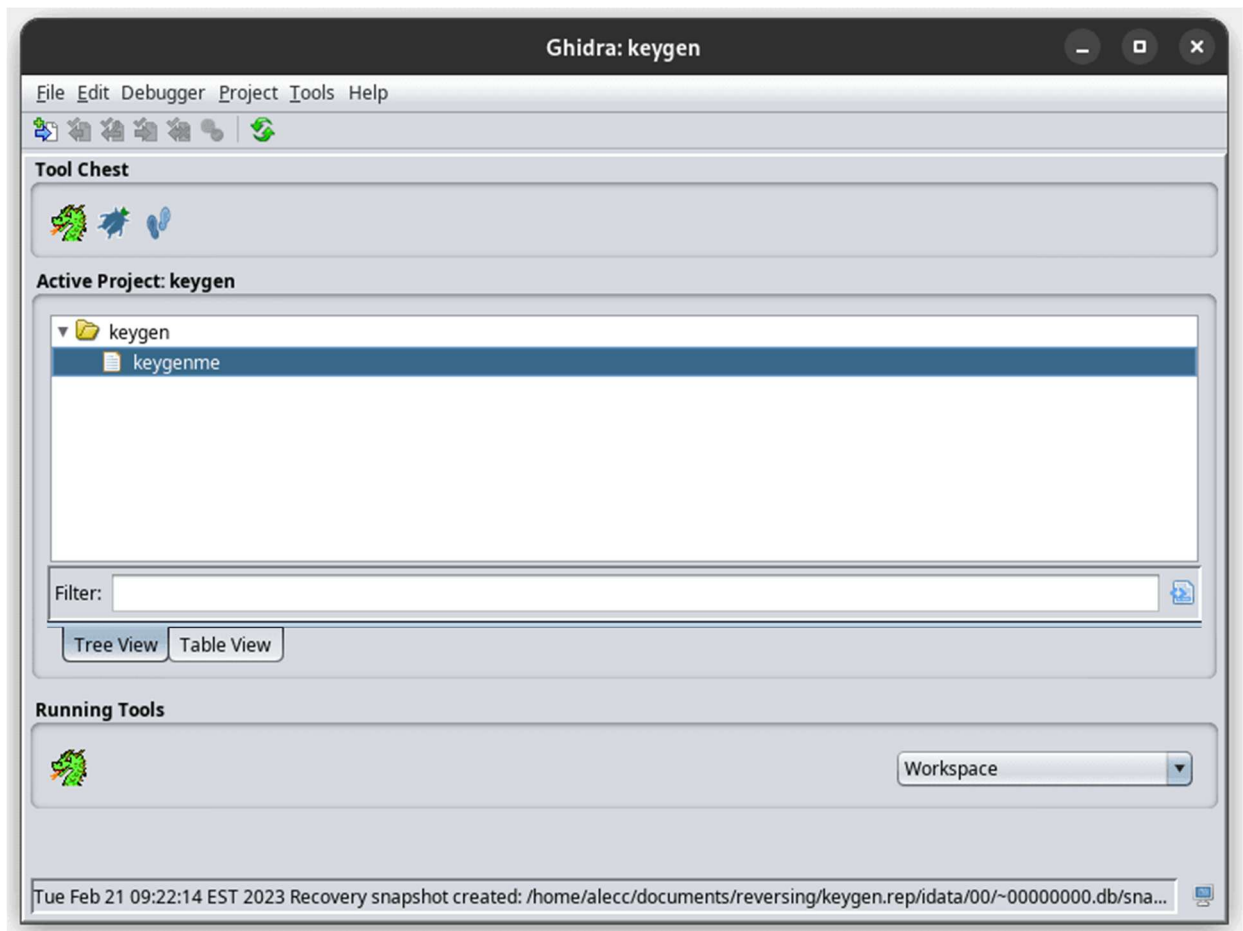


Рисунок 2.34 – Головне вікно ПЗ Ghidra, вибір об’єкту дослідження

Графічний інтерфейс Ghidra забезпечує зручне середовище для аналізу коду, перегляду функцій, дослідження потоків виконання та візуального представлення взаємодії між різними частинами програми.

Скрипти та розширення – дозволяє користувачам створювати власні скрипти та розширення для автоматизації задач та додавання нових функціональних можливостей.

Спільна робота – Ghidra підтримує можливості спільної роботи, що дозволяє декільком дослідникам працювати над одним і тим самим проєктом одночасно, синхронізуючи зміни в режимі реального часу.

Імпорт та експорт даних – підтримує імпорт та експорт даних у різних форматах, що дозволяє інтегрувати Ghidra з іншими інструментами для розширеного аналізу.

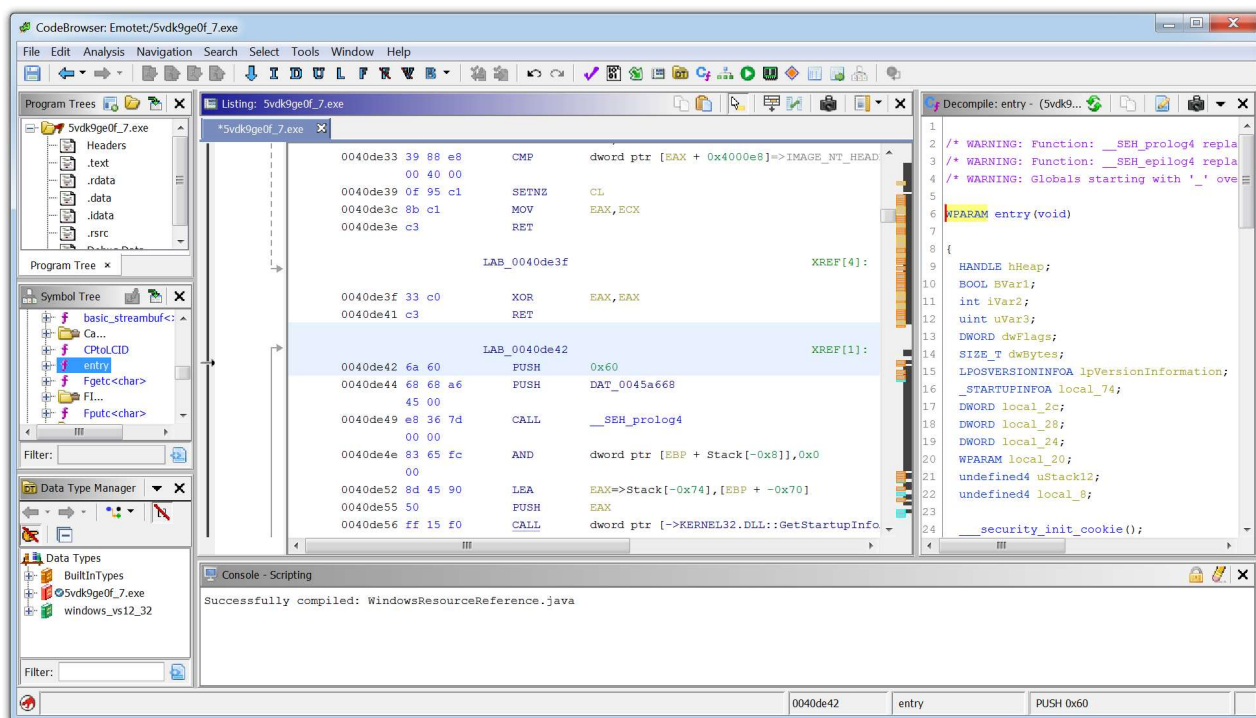


Рисунок 2.35 – ПЗ Ghidra, вікно аналізу коду ASM/C

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу зловмисного ПЗ.

1. Можливості декомпілювати код з подальшим аналізом для визначення логіки роботи, вивчення механізму проникнення в систему.

З точки зору відновлення вихідного коду.

2. Застосування методик зворотної розробки технічно та морально застарілого ПЗ, коли потрібно знати структуру, проводити відновлення логічного алгоритму роботи.

З точки зору навчальних цілей.

3. Вивчення методів зворотної розробки, аналізу ПЗ, принципів роботи виконуваних файлів. Дослідження структури ПЗ, аналіз функції, виконання декомпіляції функціонального алгоритму.

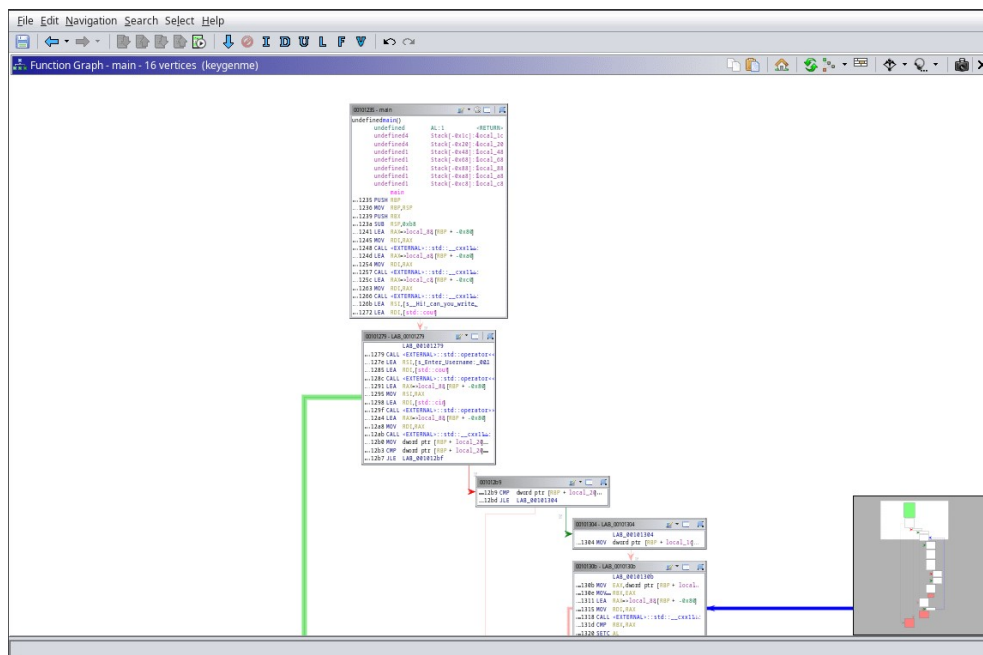


Рисунок 2.36 – ПЗ Ghidra, вікно функціонального графу залежностей

ВИСНОВКИ

Ghidra є потужним інструментом для зворотної інженерії та аналізу програмного забезпечення. Основні переваги включають:

1. Ghidra підтримує декомпіляцію для багатьох архітектур, надаючи зрозумілий для людини код на мові C.
2. Забезпечує зручне середовище для аналізу коду з графічними представленнями.
3. Підтримка спільної роботи дозволяє декільком дослідникам одночасно працювати над одним проєктом.

2.9. ПЗ DNSPY

ЛІЦЕНЗІЯ

GPLv3 (General Public License версії 3). dnSpy є відкритим програмним забезпеченням, і користувачі мають право вільно використовувати, змінювати та розповсюджувати її. Ліцензія GPLv3 також вимагає, щоб будь-які похідні роботи, створені на основі dnSpy, були випущені під тією ж ліцензією. Це забезпечує, що програмне забезпечення та його модифікації залишаються вільними та доступними для всіх користувачів. Незважаючи що оригінальний репозиторій dnSpy заархівований і доступний лише для читання все ж ПЗ є актуальною розробкою.

РОЗРОБНИК

dnSpy розробляється спільнотою розробників з відкритим кодом.

ОФІЦІЙНИЙ РЕСУРС

<https://github.com/dnSpy/dnSpy>

ОПИС

dnSpy зручний інструмент для декомпіляції, редагування та налаштування.NET-додатків.

Можливості програми

Підтримка різних мов, dnSpy підтримує декомпіляцію до C#, Visual Basic .NET та інших мов, що дозволяє переглядати вихідний код додатків .NET.

Розширений перегляд коду, інтерактивний інтерфейс дозволяє користувачам зручно навігувати по декомпільованому коду, переглядаючи структури, методи та властивості.

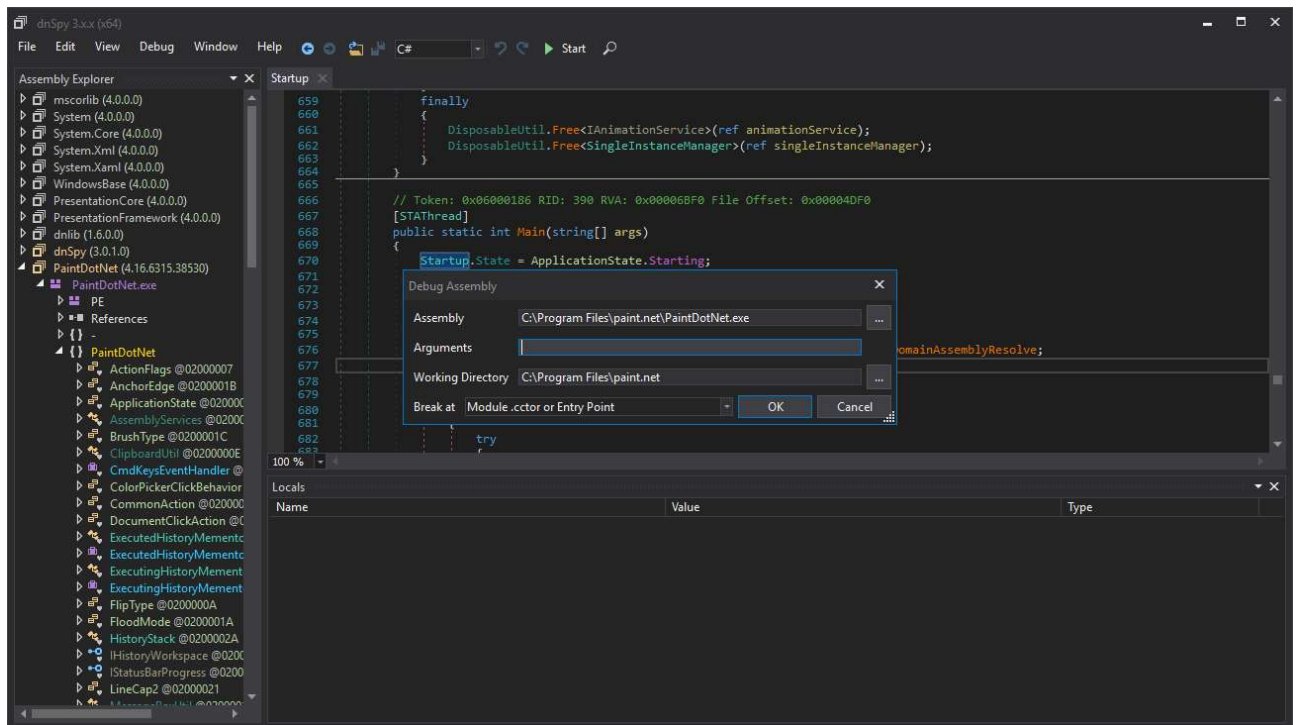


Рисунок 2.37 – Головне вікно ПЗ dnSpy

Живе редагування – можливість редагувати декомпільований код безпосередньо в програмі, що дозволяє швидко вносити зміни та тестувати їх.

Вбудований редактор – вбудований редактор з підсвічуванням синтаксису та підтримкою IntelliSense для спрощення процесу редагування.

Повноцінне налаштування – dnSpy підтримує додатки .NET, дозволяючи встановлювати точки зупину, переглядати змінні та виконувати код покроково.

Віддалене налаштування – можливість моніторингу віддалених процесів, що робить dnSpy корисним інструментом для аналізу живих додатків.

Плагіни та розширення – підтримка плагінів та розширень дозволяє користувачам додавати нові функції та інтегрувати dnSpy з іншими інструментами для розширеного аналізу.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. dnSpy дає змогу дослідникам безпеки декомпільувати та аналізувати підозрілий .NET код для виявлення його логіки роботи, потенційних

ВИСНОВКИ

dnSpy є потужним інструментом для декомпіляції, редагування та налаштування .NET додатків. Основні переваги включають:

1. Підтримка декомпіляції до різних мов, включаючи C# та Visual Basic .NET.
2. Можливість редагувати код в реальному часі та виконувати повноцінне налаштування.
3. Підтримка додаткових модулів для розширення функціональності.

2.10. ПЗ PESTUDIO

ЛІЦЕНЗІЯ

pestudio standard freeware для особистого використання. PeStudio доступний безкоштовно для особистого використання. Для комерційного використання доступні платні ліцензії pestudio pro.

РОЗРОБНИК

PeStudio розроблений компанією Winitor.

ОФІЦІЙНИЙ РЕСУРС

<https://www.winitor.com/>

ОПИС

PeStudio призначений для статичного аналізу PE-файлів. Він допомагає виявляти підозрілі ознаки без запуску файлу, що знижує ризик зараження. Має можливість співпрацювати з різними Malware БД.

Можливості програми

Аналіз без запуску PeStudio дозволяє аналізувати виконувані файли без їх фактичного запуску, забезпечуючи безпечне дослідження потенційно шкідливих файлів.

Програма використовує базу даних сигнатур для виявлення відомих загроз.

Імпорти та екпорти, відображає всі імпортовані та експортовані функції, що дозволяє виявити потенційно небезпечні бібліотеки та функції.

Інформація про ресурси – відображає детальну інформацію про ресурси файлу, такі як іконки, діалоги та інші вбудовані елементи.

Вразливості – виявляє потенційні вразливості у файлах, такі як небезпечні секції, небезпечні функції та інші ризики.

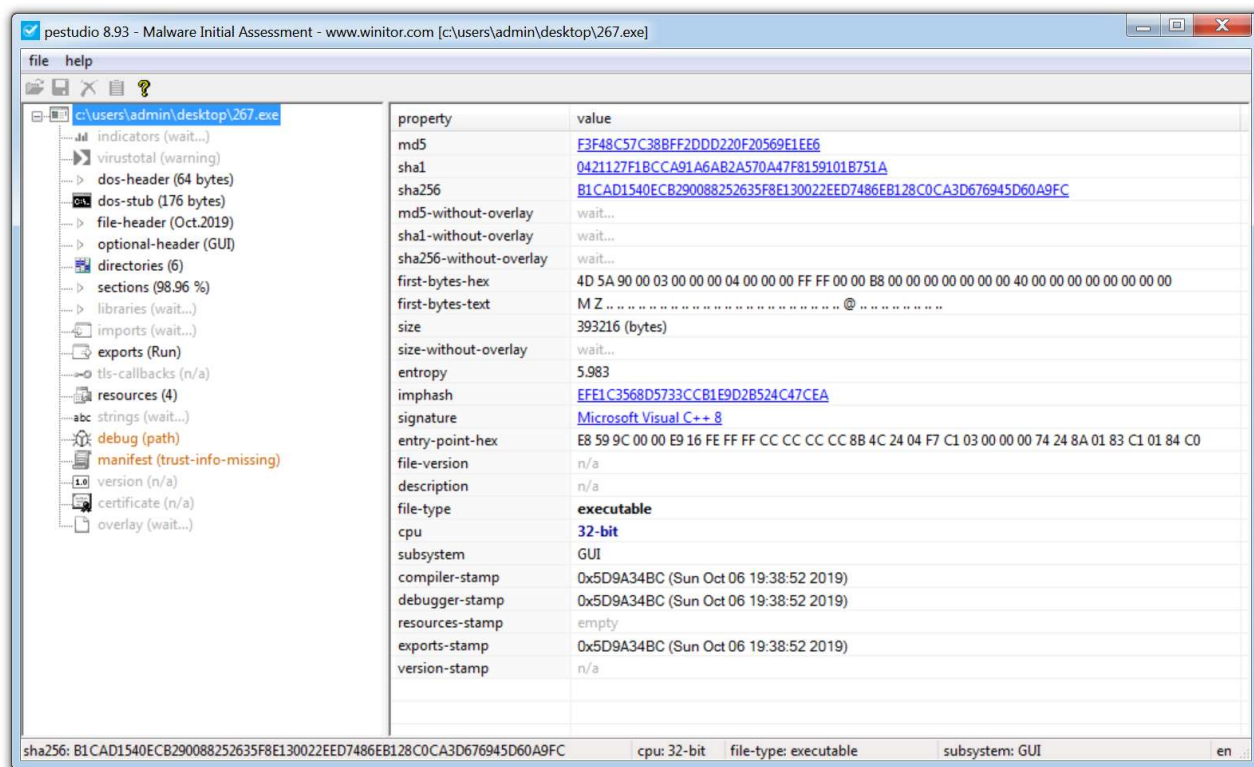


Рисунок 2.39 – Головне вікно ПЗ PeStudio

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. Аналіз структури ПЗ без запуску, Може використовувати сигнатури, індикатори підозрілої поведінки та службові характеристики файлу для виявлення відомих загроз і ознак можливої шкідливої активності. Виявляти додані структури, проводити перевірку підозрілого файлу на наявність шкідливих чи нетипічних для цього ПЗ імпортів та експортованих Dll функцій.

З точки зору перевірки програмного забезпечення.

2. Аналіз нових програм. Розробники використовують PeStudio для перевірки нових програм на наявність потенційних вразливостей перед випуском. Як приклад розробник аналізує нову версію свого програмного забезпечення за допомогою PeStudio для виявлення небезпечних функцій або ресурсів, що можуть становити загрозу.

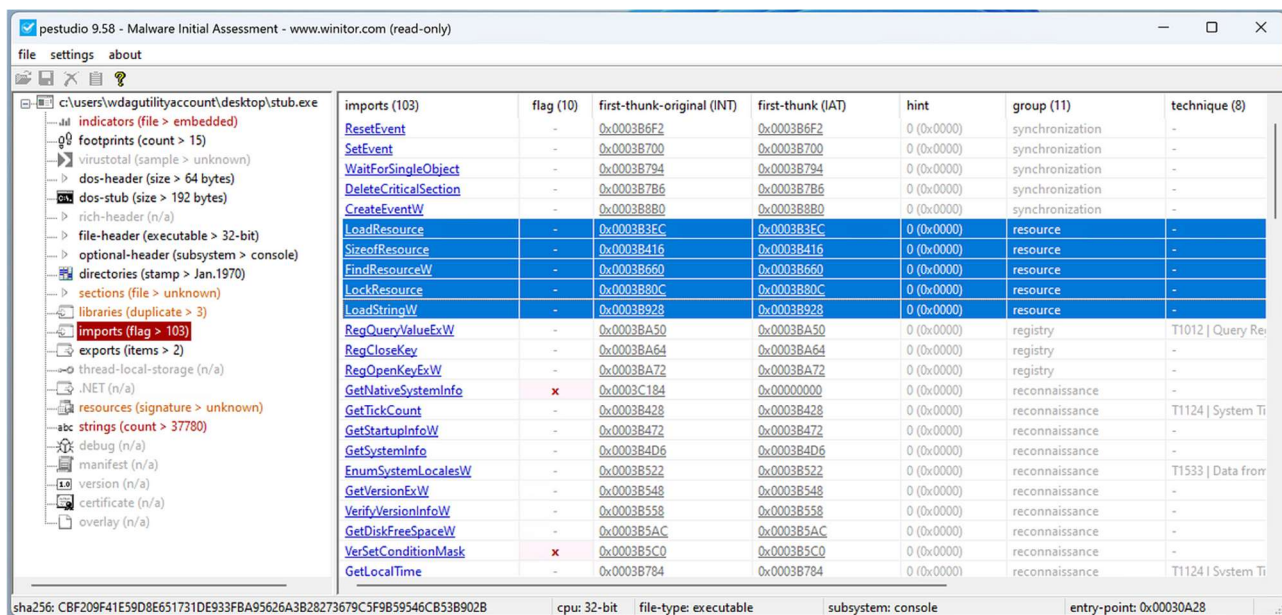


Рисунок 2.40 – ПЗ PeStudio, вікно перегляду таблиці імпорту

З точки зору навчальних цілей.

3. Освіта та тренування. PeStudio використовується в навчальних закладах для навчання студентів основам аналізу виконуваних файлів та виявлення вразливостей, демонстрація процесу аналізу та виявлення небезпечних функцій у виконуваних файлах.

ВИСНОВКИ

PeStudio є ефективним засобом для статичного аналізу виконуваних файлів, оскільки допомагає виявляти загрози, перевіряти структуру PE файлів, аналізувати імпортовані функції, цифрові підписи, рядки та інші важливі артефакти без безпосереднього виконання програми. До основних переваг цього інструменту належать:

1. Безпечне дослідження потенційно шкідливих файлів.
2. Виявлення небезпечних бібліотек та функцій.
3. Виявлення потенційних ризиків у файлах.

2.11. ПЗ CFF EXPLORER

ЛІЦЕНЗІЯ

Freeware. CFF Explorer є безкоштовним програмним забезпеченням, яке можна використовувати для особистих та комерційних цілей без обмежень.

РОЗРОБНИК

CFF Explorer був розроблений Daniel Pistelli, засновником NTCore.

ОФІЦІЙНИЙ РЕСУРС

<https://ntcore.com/explorer-suite/>

ОПИС

CFF Explorer є інструментом для аналізу та редагування виконуваних файлів формату PE, який дає змогу досліджувати їхню внутрішню структуру, змінювати заголовки, секції та інші службові атрибути. Є корисним для розробників, дослідників безпеки та аналітиків, оскільки забезпечує зручні можливості для детального вивчення виконуваних файлів, перевірки їхніх параметрів і внесення необхідних змін під час аналізу або тестування.

Можливості програми

Перегляд структури – Дозволяє детально досліджувати структуру виконуваних файлів, включаючи заголовки, секції, таблиці імпорту та експорту.

Редагування атрибутів – Можливість змінювати атрибути файлів, такі як заголовки, секції, розміри та інші параметри.

Редагування ресурсів – Інструмент для редагування вбудованих ресурсів виконуваних файлів, таких як іконки, діалоги та інші елементи.

Перегляд та зміна – Можливість переглядати та змінювати ресурси безпосередньо в PE-файлі.

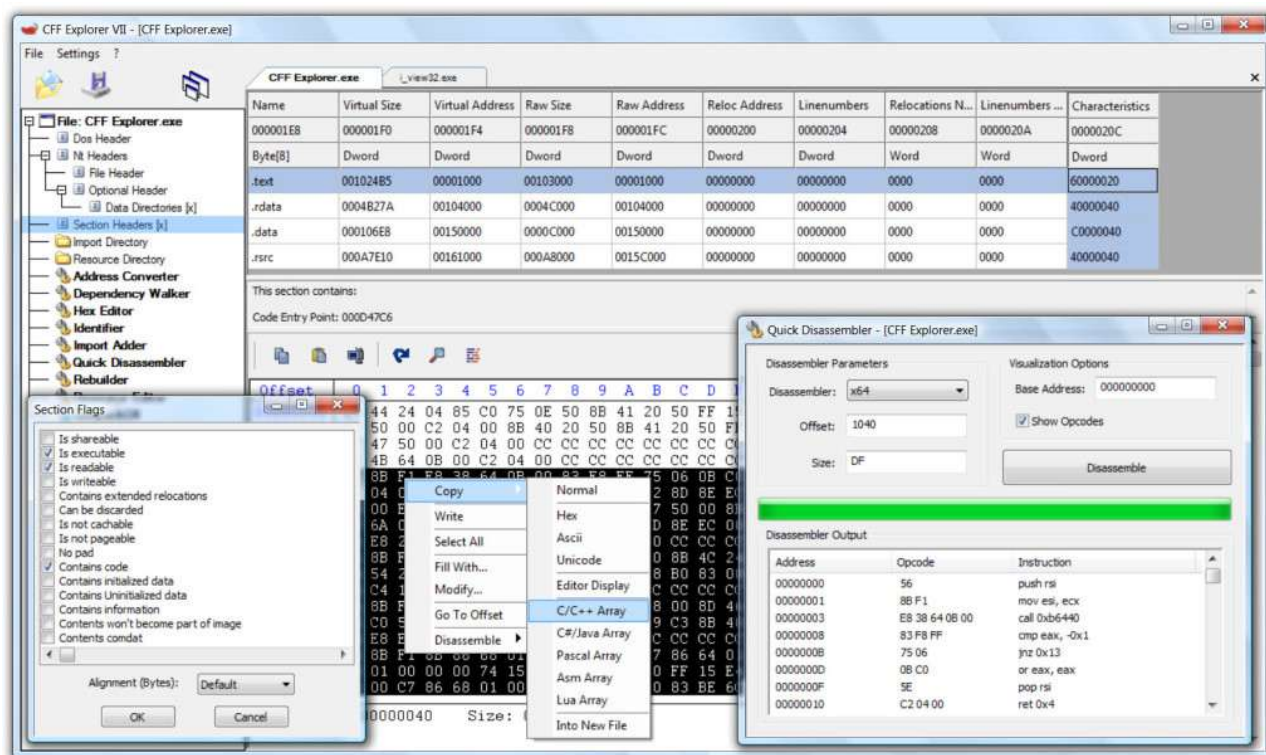


Рисунок 2.41 – Головне вікно ПЗ CFF Explorer

Автоматизація задач – Підтримка скриптів дозволяє автоматизувати різні задачі, що значно підвищує ефективність роботи.

Розширюваність – Користувачі можуть створювати власні скрипти для виконання спеціалізованих завдань.

Сумісність – CFF Explorer легко інтегрується з іншими інструментами аналізу та редагування, забезпечуючи комплексний підхід до роботи з PE-файлами.

Експорт даних – Можливість експортувати результати аналізу у різних форматах для подальшого використання.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. Виявлення та дослідження. CFF Explorer дозволяє дослідникам безпеки аналізувати структуру виконуваних файлів та виявляти потенційні загрози або аномалії. Як приклад спеціаліст з безпеки використовує CFF Explorer для

перевірки заголовків і секцій підозрілого файлу, виявляючи шкідливі модифікації.

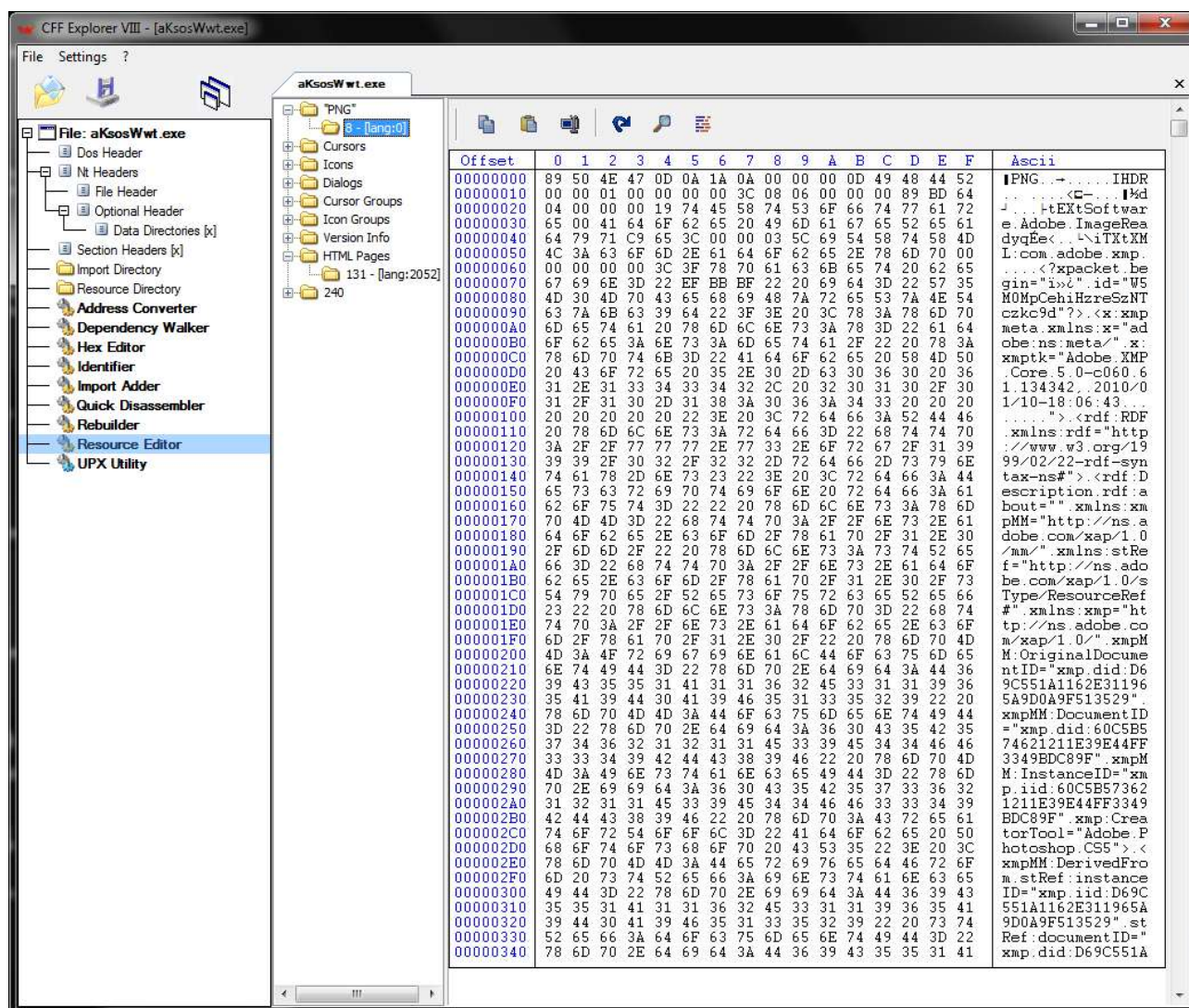


Рисунок 2.42 – ПЗ CFF Explorer, вікно перегляду ресурсів об’єкту що досліджується

З точки зору розробки програмного забезпечення.

2. Оптимізація та налагодження. Розробники використовують CFF Explorer для редагування та оптимізації виконуваних файлів, налаштування параметрів секцій та заголовків. Як приклад розробник змінює атрибути файлу, щоб оптимізувати його розмір та підвищити продуктивність.

З точки зору навчальних цілей.

3. Освіта та тренування. CFF Explorer використовується у навчальних закладах для навчання студентів основам аналізу та редагування виконуваних файлів.

ВИСНОВКИ

CFF Explorer є потужним інструментом для аналізу та редагування виконуваних файлів. Основні переваги включають:

1. Можливість детально досліджувати структуру виконуваних файлів, включаючи заголовки, секції та інші атрибути.
2. Інструменти для редагування вбудованих ресурсів файлу, таких як іконки та діалоги.
3. Автоматизація задач за допомогою скриптів, що підвищує ефективність роботи.
4. Легко інтегрується з іншими інструментами для комплексного аналізу.

2.12. ПЗ PROCESS HACKER

ЛІЦЕНЗІЯ

GPLv3 (General Public License версії 3). Process hacker є відкритим програмним забезпеченням, і користувачі мають право вільно використовувати, змінювати та розповсюджувати її. Ліцензія GPLv3 також вимагає, щоб будь-які похідні роботи, створені на основі dnSpy, були випущені під тією ж ліцензією. Це забезпечує, що програмне забезпечення та його модифікації залишаються вільними та доступними для всіх користувачів.

РОЗРОБНИК

Process Hacker розробляється спільнотою розробників з відкритим кодом.

ОФІЦІЙНИЙ РЕСУРС

<https://processhacker.sourceforge.io/>

ОПИС

Process Hacker є багатофункціональним інструментом для моніторингу та керування процесами у Windows, який дає змогу переглядати активні процеси, служби, мережеві з'єднання та завантажені модулі. Програма також надає детальну інформацію про використання оперативної пам'яті, процесорного часу та інших системних ресурсів кожним процесом, що робить її корисною для системних адміністраторів, розробників і фахівців з кібербезпеки під час діагностики, оптимізації та аналізу роботи системи.

Можливості програми

Детальна інформація – відображає детальну інформацію про кожен активний процес, включаючи використання ЦП, пам'яті та дискових ресурсів.

Ієрархія процесів – відображає процеси у вигляді дерева, показуючи батьківські та дочірні зв'язки між ними.

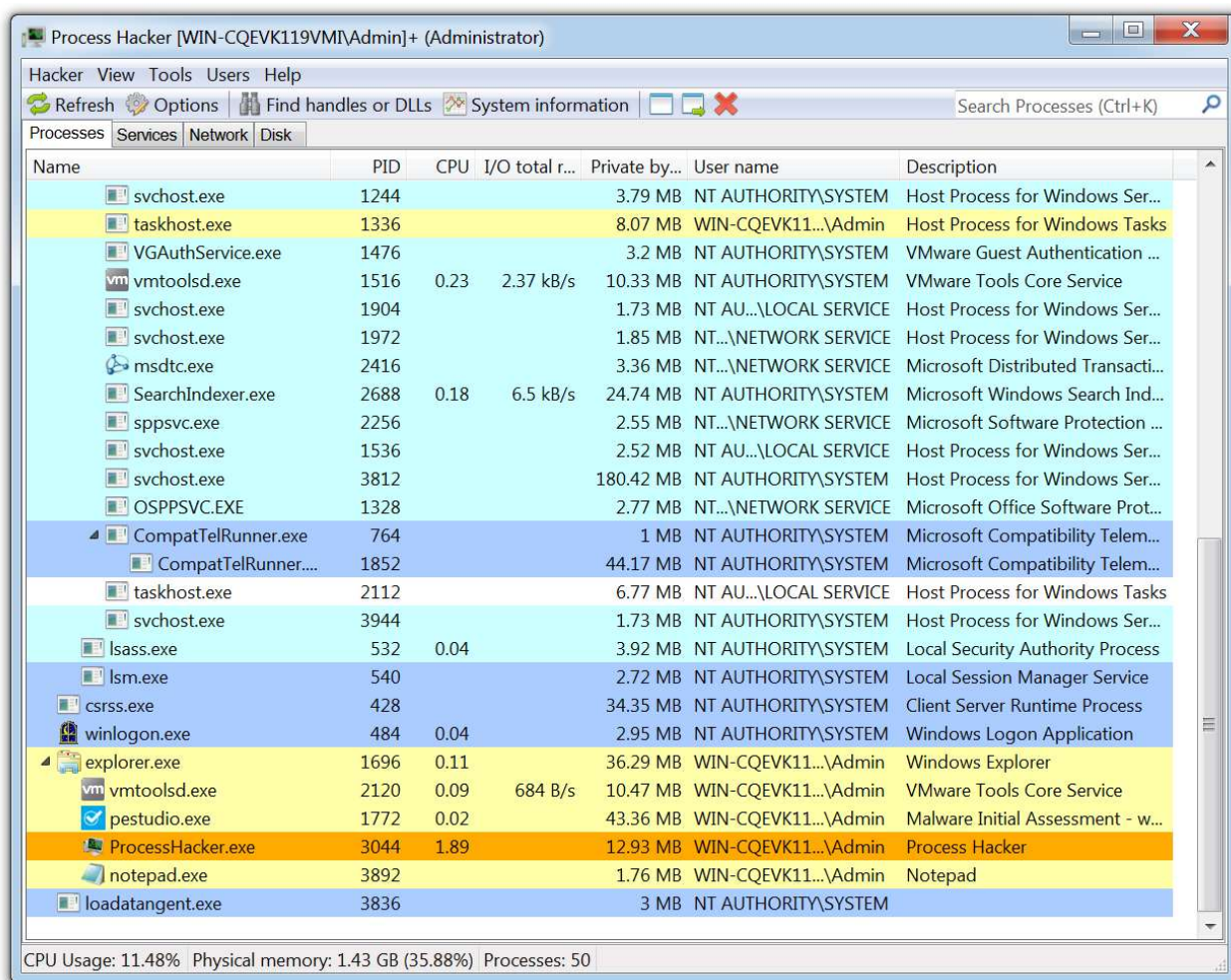


Рисунок 2.43 – Головне вікно ПЗ Process hacker

Завершення процесів – дозволяє завершувати небажані процеси або завдання, які не відповідають.

Зміна пріоритетів – можливість змінювати пріоритет процесів для оптимізації використання ресурсів системи.

Управління службами – перегляд, запуск, зупинка та налаштування служб Windows.

Інформація про драйвери – відображає завантажені драйвери та їх властивості.

Деталі пам'яті – показує використання фізичної та віртуальної пам'яті кожним процесом.

Інструменти налагодження – містить інструменти для відстеження витоків пам'яті та аналізу розподілу пам'яті.

Мережеві з'єднання – відображає активні мережеві з'єднання та відкриті порти для кожного процесу.

Відстеження активності – можливість відстежувати мережеву активність для виявлення підозрілих з'єднань.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу продуктивності системи.

1. Моніторинг ресурсів – Process Hacker дозволяє детально аналізувати використання ресурсів системи кожним процесом, що допомагає виявити процеси, які споживають надмірну кількість ЦП або пам'яті. Як приклад системний адміністратор використовує Process Hacker для ідентифікації процесу, який викликає уповільнення роботи сервера, та змінює його пріоритет.

З точки зору управління процесами та службами.

2. Керування службами – Process Hacker дозволяє зупиняти, запускати та налаштовувати служби Windows, що робить його корисним інструментом для адміністрування системи. Як приклад адміністратор зупиняє непотрібні служби, щоб звільнити ресурси системи.

З точки зору забезпечення безпеки.

3. Process Hacker дає змогу відстежувати мережеві з'єднання та відкриті порти, пов'язані з окремими процесами, що допомагає виявляти підозрілу активність у системі. Наприклад пошук та завершення роботи шкідливого процесу, який створив несанкціоновані мережеві з'єднання.

ВИСНОВКИ

Process Hacker є потужним інструментом для моніторингу та управління процесами у Windows. Основні переваги включають:

1. Відображає детальну інформацію про використання ресурсів кожним процесом.
2. Дозволяє зупиняти, запускати та налаштовувати служби Windows, а також переглядати інформацію про драйвери.
3. Відстежує мережеві з'єднання та відкриті порти для кожного процесу.

2.13. ПЗ x64dbg

ЛІЦЕНЗІЯ

GNU General Public License v3.0 (GPLv3). x64dbg є відкритим програмним забезпеченням, що поширюється під ліцензією GPLv3. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

x64dbg розробляється спільнотою розробників з відкритим кодом.

ОФІЦІЙНИЙ РЕСУРС

<https://x64dbg.com/>

ОПИС

x64dbg є налагоджувачем з відкритим вихідним кодом для Windows, який призначений для аналізу та налагодження 32 бітних і 64 бітних додатків [57,58]. Інструмент надає розробникам, дослідникам безпеки та фахівцям зі зворотної розробки функціональні можливості для поглибленого аналізу програмного забезпечення, дослідження його поведінки та перевірки логіки виконання.

Можливості програми

Інтуїтивний дизайн – x64dbg забезпечує зручний та зрозумілий інтерфейс для користувачів, що спрощує процес налагодження та аналізу.

Гнучкість налаштувань – можливість налаштовувати інтерфейс під потреби користувача.

Підтримка різних форматів – підтримує відладку PE, ELF, і різних інших форматів виконуваних файлів.

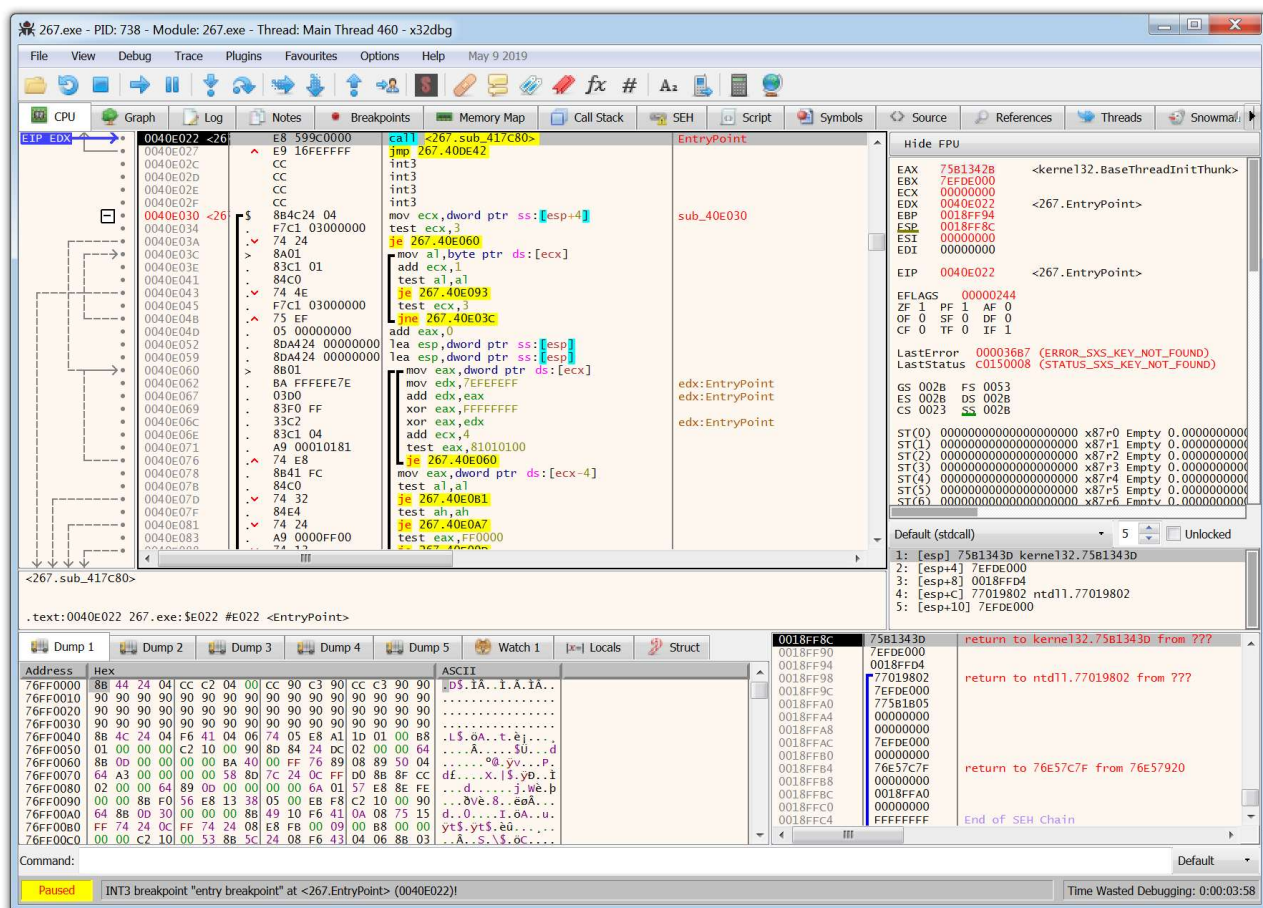


Рисунок 2.44 – Головне вікно ПЗ x64dbg

Аналіз коду – автоматично аналізує виконуваний код та надає користувачеві зрозумілі інструкції.

Підтримка скриптів – дозволяє користувачам писати власні скрипти для автоматизації задач налагодження.

Плагіни – Підтримка плагінів для розширення функціональності та інтеграції з іншими інструментами.

Активна спільнота – велика спільнота користувачів та розробників, що постійно вдосконалює програму.

Документація – доступ до детальної документації та навчальних матеріалів, які допомагають швидко освоїти інструмент.

CPU				Log	Breakpoints	Memory Map	Script	Symbols	References	Threads
	0040115A	E8 3D 02 00 00		call crackme.40139C				int MessageBoxA(...)		
	0040115F	C9		leave						
	00401160	C2 10 00		ret 10						
	00401163	8D 15 49 63 40 00		lea edx,dword ptr ds:[406349]				edx:"fdsdf", 406349		
	00401169	52		push edx				LPCTSTR lpString		
	0040116A	E8 8D 02 00 00		call crackme.4013FC				int strlenA(...)		
EAX	0040116F	8B E8		mov ebp,eax						
	00401171	B9 05 00 00 00		mov ecx,5						
	00401176	33 F6		xor esi,esi						
	00401178	33 C0		xor eax,eax						
	0040117A	8A 0C 16		mov cl,byte ptr ds:[esi+edx]				esi+edx:"fdsdf"		
	0040117D	8A D9		mov bl,cl						
	0040117F	32 98 28 63 40 00		xor bl,byte ptr ds:[eax+406328]						
	00401185	40		inc eax						
	00401186	83 F8 05		cmp eax,5						
BIP	00401189	8B 1C 32		mov byte ptr ds:[edx+esi],bl				edx+esi:"fdsdf"		
	0040118C	8B 88 27 63 40 00		mov byte ptr ds:[eax+406327],cl						
	00401192	75 02		jnz crackme.401196						
	00401194	33 C0		xor eax,eax						
	00401196	46		inc esi						
	00401197	3B F5		cmp esi,ebp						
	00401199	72 DF		jb crackme.40117A						
	0040119B	33 FF		xor edi,edi						
	0040119D	33 C9		xor ecx,ecx						
	0040119F	85 ED		test ebp,ebp						
	004011A1	76 26		jbe crackme.4011C9						
ECX	004011A3	8A 9F 2D 63 40 00		mov bl,byte ptr ds:[edi+40632D]						
	004011A9	8B F5		mov esi,ebp						
	004011AB	2B F1		sub esi,ecx						
	004011AD	4E		dec esi						
	004011AE	8A 04 32		mov al,byte ptr ds:[edx+esi]				edx+esi:"fdsdf"		
	004011B1	32 D8		xor bl,al						

Рисунок 2.45 – ПЗ x64dbg, інформація про переходи та їх значення

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. x64dbg використовується для виявлення та дослідження підозрілих або шкідливих програм, оскільки дає змогу аналізувати їхню поведінку, відстежувати виконання коду та визначати основні механізми роботи. Наприклад, фахівець з кібербезпеки може застосовувати x64dbg для дослідження шкідливого коду, щоб з'ясувати, як він обробляє дані та які дії виконує в системі.

З точки зору налаштування програм.

2. Виправлення помилок. Розробники використовують x64dbg для налагодження своїх додатків, знаходження та виправлення помилок у коді. Як приклад програміст використовує x64dbg для виявлення причин збою програми, встановлення точок зупину та аналізу змінних.

З точки зору освітніх цілей.

3. Навчання зворотної інженерії. x64dbg використовується в навчальних закладах для викладання основ зворотної інженерії та налагодження додатків.

ВИСНОВКИ

x64dbg є потужним інструментом для налагодження та зворотної інженерії програмного забезпечення. Основні переваги включають:

1. Зручний та настроюваний інтерфейс для ефективного аналізу та налагодження.
2. Підтримка різних форматів виконуваних файлів та автоматичний аналіз коду.
3. Підтримка автоматизації задач і розширення функціональності.

2.14. ПЗ VIRUSTOTAL (ОНЛАЙН-СЕРВІС)

ЛІЦЕНЗІЯ

Безкоштовна послуга з додатковими платними функціями (Freemium). VirusTotal пропонує безкоштовний аналіз файлів і URL-адрес, однак для розширених можливостей та API доступні платні підписки.

РОЗРОБНИК

VirusTotal був розроблений Hispasec Sistemas. У 2012 році сервіс був придбаний Google.

ОФІЦІЙНИЙ РЕСУРС

<https://www.virustotal.com/gui/home/upload>

ОПИС

VirusTotal сервісо для аналізу файлів і URL адрес з метою виявлення вірусів, хробаків, троянів та інших видів шкідливого програмного забезпечення.

Сервіс використовує результати багатьох антивірусних рішень та інструментів перевірки веб ресурсів, що дає змогу ефективно оцінювати потенційні загрози й формувати докладні звіти за результатами аналізу.

Можливості програми

Завантаження файлів – Користувачі можуть завантажувати файли для аналізу на наявність шкідливого ПЗ.

Перевірка URL-адрес – Аналіз URL-адрес для виявлення потенційно небезпечних сайтів.

Понад 70 антивірусних систем – Використовує численні антивірусні двигуни для точного виявлення загроз.

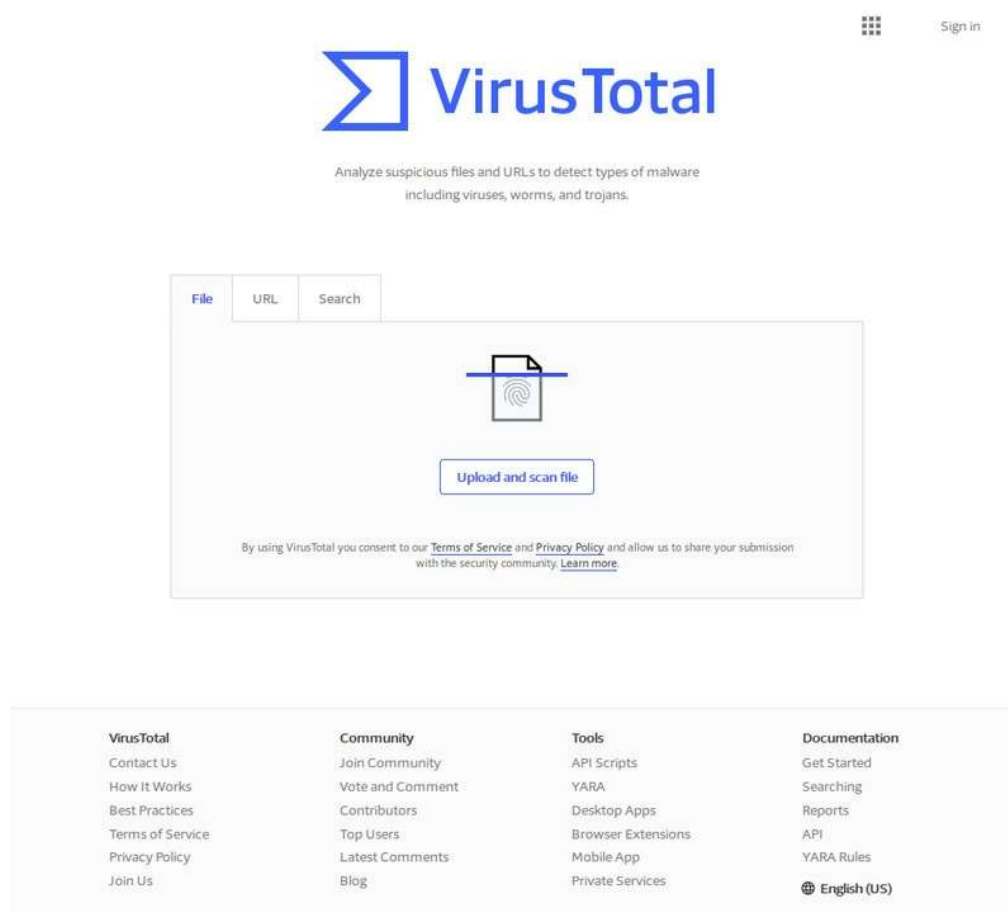


Рисунок 2.46 – Головне вікно онлайн-сервісу VirusTotal

Детальні звіти – Надає детальні звіти про виявлені загрози, включаючи інформацію про тип шкідливого ПЗ.

API доступ – Платні підписки забезпечують доступ до API для інтеграції з іншими інструментами та автоматизації процесів аналізу.

Спільний доступ до звітів – Користувачі можуть ділитися звітами та результатами аналізу з іншими.

Спільнота дослідників – Дослідники безпеки можуть взаємодіяти та ділитися результатами, сприяючи колективному виявленню загроз.

Обмін даними – Дані про виявлені загрози можуть бути обміняні з іншими користувачами для підвищення рівня безпеки.

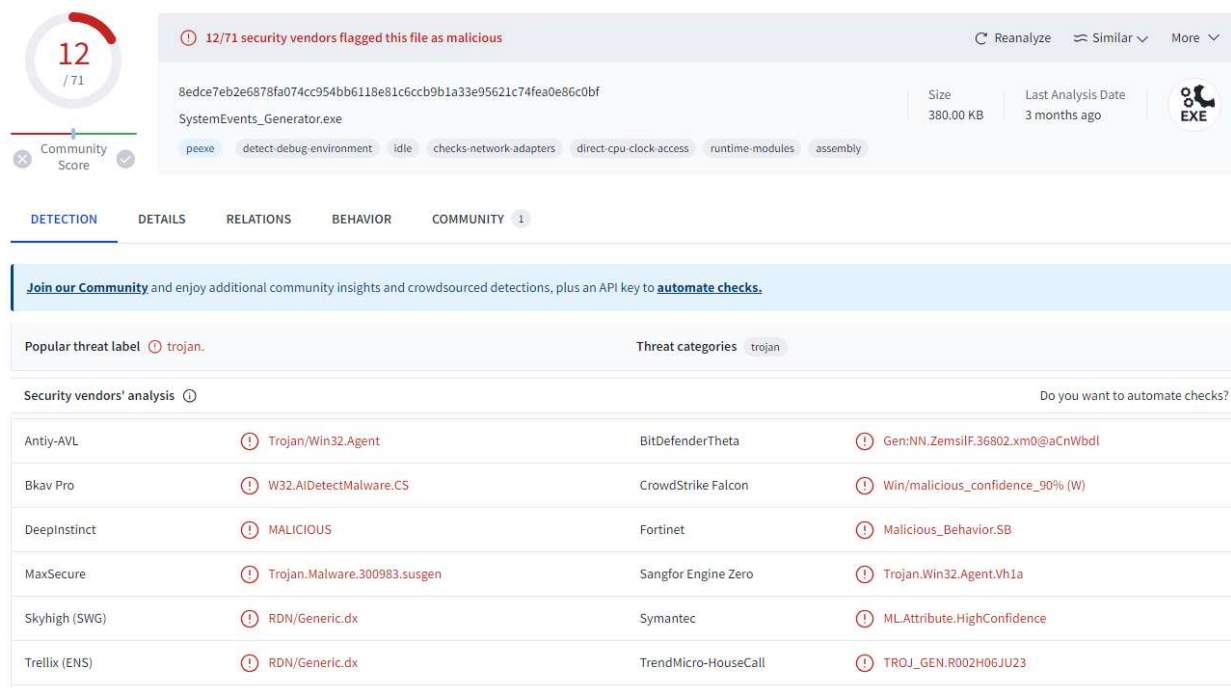


Рисунок 2.47 – Онлайн-сервіс VirusTotal, результат сканування

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. VirusTotal дає змогу користувачам завантажувати підозрілі файли для перевірки на наявність вірусів та іншого шкідливого програмного забезпечення за допомогою багатьох антивірусних систем. Наприклад, аналітик безпеки використовує VirusTotal для перевірки нових зразків шкідливого ПЗ і швидко отримує результати аналізу від різних засобів захисту.

З точки зору перевірки URL-адрес.

2. VirusTotal допомагає перевіряти URL адреси на наявність фішингових ознак, шкідливого вмісту та інших загроз, що сприяє підвищенню безпеки користувачів. Наприклад, системний адміністратор перевіряє підозрілі посилання, надані користувачами, щоб запобігти переходу на небезпечні веб сайти.

З точки зору інтеграція з іншими інструментами.

3. Автоматизація процесів. Розробники можуть використовувати API VirusTotal для інтеграції з іншими системами безпеки та автоматизації перевірок. Як приклад компанія інтегрує VirusTotal API з внутрішньою системою моніторингу для автоматичного аналізу нових файлів.

ВИСНОВКИ

VirusTotal є потужним інструментом для виявлення та аналізу загроз. Основні переваги включають:

1. Використання понад 70 антивірусних систем для точного виявлення загроз.
2. Підтримка API для інтеграції з іншими інструментами та автоматизації процесів аналізу.
3. Можливість обміну даними та звітами про загрози для підвищення рівня безпеки.

2.15. ПЗ CUCKOO SANDBOX (ОНЛАЙН-СЕРВІС)

ЛІЦЕНЗІЯ

GNU General Public License v3.0 (GPLv3). Cuckoo Sandbox є відкритим програмним забезпеченням, що поширюється під ліцензією GPLv3. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

Cuckoo Sandbox був розроблений спільнотою з відкритим вихідним кодом під керівництвом Claudio Guarnieri.

ОФІЦІЙНИЙ РЕСУРС

<https://cuckoo.cert.ee/>

ОПИС

Платформа для автоматизованого аналізу шкідливого програмного забезпечення в контрольованому ізольованому середовищі. Вона дає змогу дослідникам безпеки перевіряти підозрілі файли та URL адреси, а також отримувати докладні звіти про їхню поведінку, мережеву активність і зміни, що відбуваються в системі під час виконання.

Можливості програми

Підтримка різних форматів – Cuckoo Sandbox може аналізувати виконувані файли, документи, скрипти, зображення та багато інших типів файлів.

Автоматичний аналіз – Виконує файли у віртуальному середовищі, відстежуючи їх поведінку та взаємодію з системою.

Відстеження мережових з'єднань – Моніторинг та аналіз мережевої активності, включаючи DNS-запити, HTTP-трафік та інші протоколи.

Перехоплення трафіку – Перехоплює та аналізує мережевий трафік для виявлення підозрілих з'єднань.

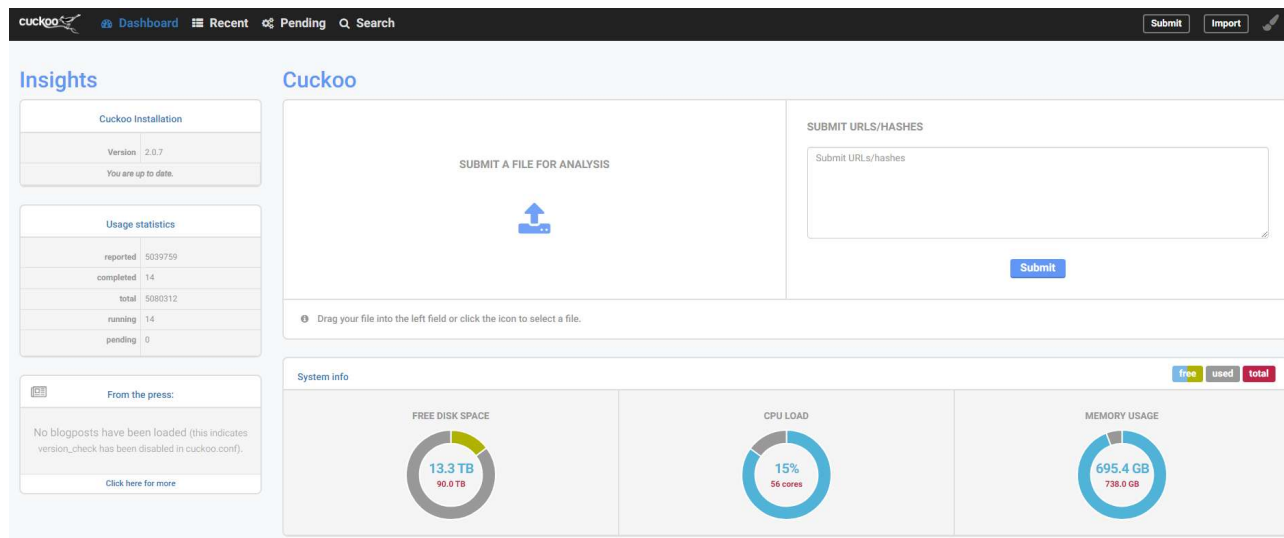


Рисунок 2.48 – Головне вікно онлайн-сервісу Cuckoo Sandbox

Детальні звіти – Генерує звіти з детальною інформацією про поведінку аналізованих файлів, включаючи системні виклики, створені процеси, зміни у файлах та реєстрі.

Графічна візуалізація – Надає візуальні представлення взаємодії шкідливого ПЗ з системою для полегшення аналізу.

Модульна архітектура – Легко інтегрується з іншими інструментами безпеки та підтримує додавання нових модулів для розширення функціональності.

API доступ – Надає API для автоматизації завдань та інтеграції з іншими системами.

Cuckoo Sandbox є корисним інструментом для дослідників безпеки та аналітиків, оскільки забезпечує автоматизований аналіз підозрілих об'єктів і допомагає виявляти ознаки потенційних загроз.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу шкідливого ПЗ.

1. Виявлення та дослідження, Cuckoo Sandbox дає змогу запускати підозрілі файли в ізольованому середовищі та отримувати звіти про їхню поведінку. Наприклад, аналітик безпеки використовує Cuckoo Sandbox для аналізу шкідливого програмного забезпечення, визначає його дії, мережеву активність, створені файли та зміни в системі.

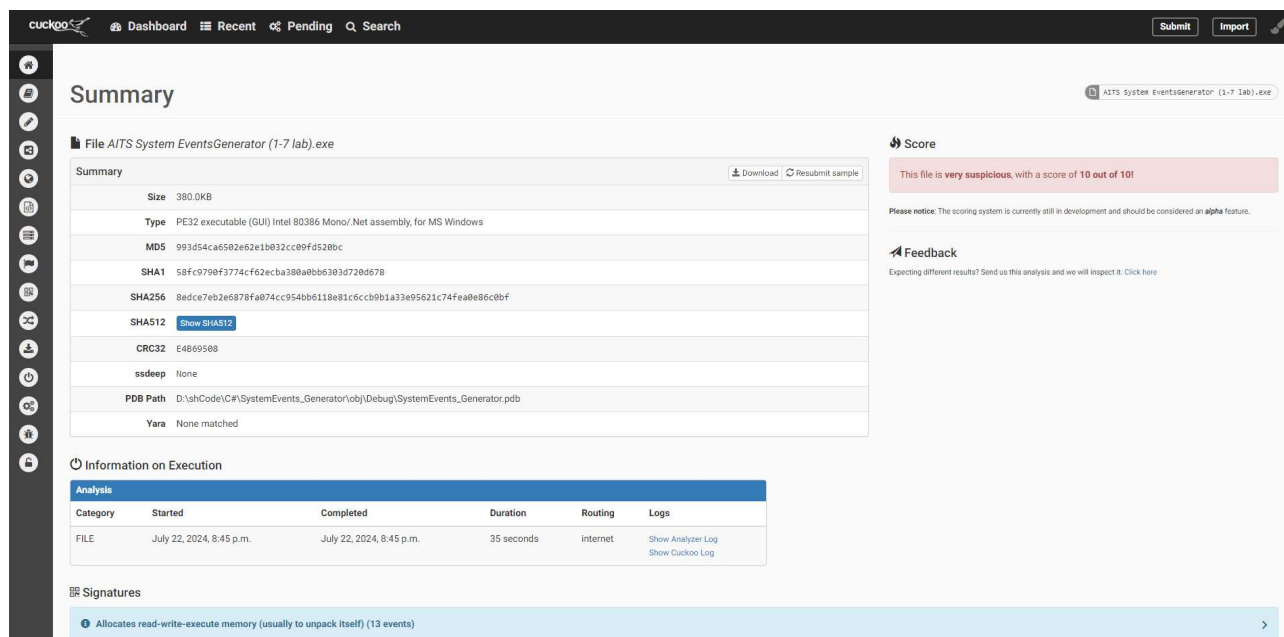


Рисунок 2.49 – Онлайн-сервіс Cuckoo Sandbox, результат аналізу

З точки зору перевірки URL-адрес.

2. Аналіз веб сайтів, Cuckoo Sandbox може використовуватися для перевірки URL адрес на наявність шкідливого вмісту, фішингових ознак або іншої небезпечної активності. Наприклад, системний адміністратор перевіряє підозрілі посилання, отримані від користувачів, щоб запобігти доступу до небезпечних веб ресурсів.

З точки зору навчальних цілей.

3. Освіта та тренування, Cuckoo Sandbox може застосовуватися у навчальному процесі для демонстрації методів аналізу шкідливого програмного забезпечення та дослідження його поведінки. Наприклад, викладач використовує платформу під час занять, щоб показати студентам етапи аналізу підозрілого файлу в контрольованому середовищі.

ВИСНОВКИ

Cuckoo Sandbox є ефективним засобом для дослідників безпеки, системних адміністраторів та освітніх установ, оскільки забезпечує аналіз підозрілих файлів і посилань в ізольованому середовищі та допомагає виявляти потенційні кіберзагрози. Основні переваги включають:

1. Виконує файли у віртуальному середовищі, відстежуючи їх поведінку.
2. Генерує звіти з інформацією про системні виклики, мережеву активність та зміни у файлах і реєстрі.
3. Легко інтегрується з іншими інструментами безпеки та підтримує розширення.

2.16. ПЗ MALWARE INFORMATION SHARING PLATFORM (MISP)

ЛІЦЕНЗІЯ

Affero General Public License (AGPL). MISP – це програмне забезпечення з відкритим кодом, що поширюється за ліцензією AGPL. Воно дозволяє вільно користуватися, модифікувати та поширювати програму, за умови, що будь-які похідні продукти також будуть розповсюджуватися під тією ж ліцензією.

РОЗРОБНИК

MISP був розроблений CIRCL (Computer Incident Response Center Luxembourg).

ОФІЦІЙНИЙ РЕСУРС

<https://www.misp-project.org/>

ОПИС

Malware Information Sharing Platform (MISP) інструмент для обміну інформацією про загрози кібербезпеки, який допомагає організаціям збирати, аналізувати та поширювати дані про шкідливе програмне забезпечення, індикатори компрометації та інші кіберзагрози [3-9,80].

Сприяє покращенню координації між фахівцями та організаціями, оскільки забезпечує централізоване середовище для накопичення інформації про загрози, її перевірки та подальшого використання під час реагування на інциденти.

Можливості програми

Централізоване зберігання – зберігає та організовує дані про загрози у централізованому сховищі.

Спільний доступ – дозволяє організаціям ділитися даними про загрози з іншими учасниками мережі.

Автоматичний аналіз – інструменти для автоматичного аналізу даних про загрози та виявлення нових загроз.



Рисунок 2.50 – Головне вікно сервісу MISP

Інтеграція з іншими інструментами – підтримка інтеграції з іншими платформами кібербезпеки та інструментами аналізу.

Графічні звіти – створення графічних звітів та візуалізацій для полегшення аналізу та розуміння даних.

Панелі моніторингу – інтерактивні панелі для моніторингу поточного стану загроз.

Плагіни та розширення – підтримка плагінів для розширення функціональності.

API доступ – надає API для автоматизації процесів та інтеграції з іншими системами.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору обміну інформацією про загрози.

1. Спільна робота між організаціями. MISP дає змогу організаціям обмінюватися даними про кіберзагрози та індикатори компрометації. Це підвищує ефективність реагування на інциденти та допомагає швидше виявляти подібні загрози в різних середовищах. Наприклад, компанія використовує MISP

для обміну інформацією про нові зразки шкідливого програмного забезпечення з іншими учасниками галузі.

З точки зору аналізу та реагування на загрози.

2. Інтеграція з іншими інструментами. MISP може використовуватися разом із SIEM системами, аналітичними платформами та іншими засобами кібербезпеки. Це дає змогу автоматизувати оброблення індикаторів компрометації, аналізувати події безпеки та формувати звіти. Наприклад, аналітик безпеки застосовує MISP разом із SIEM для виявлення аномалій у мережевому трафіку.

З точки зору навчальних цілей.

3. Освіта та тренування. MISP може використовуватися у навчальному процесі для демонстрації принципів обміну інформацією про загрози, аналізу індикаторів компрометації та організації взаємодії між учасниками кібербезпекової спільноти. Наприклад, викладач використовує MISP для ознайомлення студентів із процесом збору, структурування та аналізу даних про кіберзагрози.

ВИСНОВКИ

MISP є потужним інструментом для обміну інформацією про кіберзагрози, аналізу та співпраці між організаціями. Основні переваги включають:

1. Забезпечує централізоване сховище для даних про загрози та полегшує обмін інформацією між організаціями.
2. Підтримує автоматичний аналіз загроз та інтеграцію з іншими інструментами кібербезпеки.
3. Дозволяє розширювати функціональність за допомогою плагінів та автоматизувати завдання через API.

2.17. ПЗ ERIC ZIMMERMAN TOOLS

ЛІЦЕНЗІЯ

Freeware. Інструменти Еріка Циммермана надаються безкоштовно для особистого та комерційного використання, без обмежень.

РОЗРОБНИК

Інструменти розроблені Еріком Циммерманом, відомим фахівцем з комп'ютерної криміналістики та розробки програмного забезпечення.

ОФІЦІЙНИЙ РЕСУРС

<https://ericzimmerman.github.io/#!index.md>

ОПИС

Інструменти Еріка Циммермана (Eric Zimmerman Tools) – це набір потужних утиліт для комп'ютерної криміналістики [72], які допомагають дослідникам ефективно аналізувати цифрові докази. Вони включають утиліти для аналізу журналів подій Windows, файлів реєстру, артефактів інтернет-активності та багато іншого. Кожен інструмент спеціалізується на певному типі даних, забезпечуючи детальний аналіз та візуалізацію інформації.

Можливості комплексу програм

1. AmcacheParser – Інструмент для аналізу бази даних Amcache.hve, що містить інформацію про встановлені програми та виконувані файли. Використовується для розслідування інцидентів і відстеження дій користувачів.
2. AppCompatCacheParser – Аналізує кеш сумісності додатків (ShimCache), що містить історію виконуваних файлів. Допомагає визначити, які програми були запущені на системі.
3. bstrings є утилітою для вилучення рядків із файлів. Вона підтримує різні кодування та формати, тому використовується для аналізу бінарних файлів

і виявлення текстової інформації, яка може мати значення під час дослідження інциденту.

4. EvtxECmd є інструментом для аналізу журналів подій Windows у форматі EVTХ. Він перетворює журнали у зручний для подальшого аналізу формат і допомагає виявляти підозрілі події, помилки, ознаки несанкціонованої активності та інші важливі записи в системі.

5. JLECmd – Аналізує журнали подій у форматі JSON, створені різними програмами. Надає можливість витягувати важливу інформацію для аналізу.

6. LECmd – Утиліта для аналізу журналів подій у форматі SQLite, що дозволяє витягувати та аналізувати дані. Використовується для розслідувань і моніторингу.

7. MFTECmd – Інструмент для аналізу Master File Table (MFT) у файлових системах NTFS. Надає інформацію про створення, зміну та видалення файлів.

8. PECmd – Аналізує Prefetch файли, що містять інформацію про запуск програм. Допомагає визначити, які програми запускалися і як часто.

9. RECmd – Потужний інструмент для аналізу реєстру Windows, що дозволяє шукати та витягувати важливі дані. Корисний для форензичних розслідувань.

10. RBCmd – Аналізує об'єкти реєстру RecentDocs, які містять інформацію про недавно відкриті документи. Використовується для відстеження активності користувача.

11. SDBExplorer – Інструмент для аналізу баз даних сумісності (SDB) у Windows, що допомагає ідентифікувати виконувані файли та їх атрибути.

12. ShellBagsExplorer – Утиліта для аналізу ShellBags у Windows, що зберігають інформацію про структуру папок, які були відкриті користувачем. Допомагає відновити історію навігації по файловій системі.

13. WxTCmd – Інструмент для аналізу Windows Timeline, що надає інформацію про дії користувачів у системі. Корисний для відстеження активності користувачів.

14. ZipDump – Інструмент для аналізу та вилучення файлів з архівів ZIP. Допомагає перевіряти вміст архівів на наявність підозрілих файлів.
15. SQLECmd – Аналізує бази даних SQLite, дозволяє витягувати та аналізувати дані, що зберігаються в них. Використовується для форензичних розслідувань.
16. AppCompatProcessor – Аналізує файли ShimCache, що містять історію запуску додатків. Допомагає виявити, які програми запускалися на системі.
17. EventTranscriptParser – Інструмент для аналізу Event Transcript Logs, що містять докладні журнали дій користувачів і системи.
18. Rekall Parser – Аналізує дампи пам'яті, створені за допомогою Rekall, надаючи детальну інформацію про стан системи та запущені процеси.
19. RecentFileCache – Аналізує кеш останніх файлів, що містить інформацію про недавно відкриті файли.
20. UserAssist – Інструмент для аналізу об'єктів реєстру UserAssist, які зберігають інформацію про використання додатків.
21. Evtxtract – Витягує записи з журналів подій Windows (EVTX), що можуть бути корисними для аналізу безпеки.
22. Jumplist Parser – Аналізує списки стрибків Windows (Jumplists), що зберігають інформацію про нещодавно відкриті файли та документи.
23. TaskScheduler Parser – Інструмент для аналізу завдань, запланованих у Windows Task Scheduler, що допомагає виявляти автоматизовані дії та скрипти.
24. ThumbCache Parser – Аналізує кеш мініатюр (thumbnails) Windows, що містить зображення переглянутих файлів.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів безпеки.

1. Аналіз журналів подій. Використання EvtxECmd для детального аналізу журналів подій Windows з метою виявлення аномалій. Як приклад фахівець з безпеки використовує EvtxECmd для дослідження підозрілого входу в систему, аналізуючи журнали подій для виявлення часу і джерела атаки.

З точки зору форензичного аналізу реєстру.

2. Відновлення даних з реєстру. Використання RECcmd для витягування та аналізу даних з реєстру Windows. Як приклад аналітик використовує RECcmd для відновлення видалених ключів реєстру, що можуть вказувати на сліди шкідливого ПЗ.

З точки зору аналізу веб-активностей.

3. Перегляд історії браузера. Використання BrowsingHistoryView для відстеження веб-активності користувача. Як приклад дослідник використовує BrowsingHistoryView для відстеження дій користувача в Інтернеті, аналізуючи відвідані сайти і завантажені файли.

ВИСНОВКИ

Інструменти Еріка Циммермана є незамінними для фахівців з безпеки та комп'ютерної криміналістики, забезпечуючи ефективний аналіз та розслідування інцидентів. Основні переваги включають:

1. Детальний аналіз журналів подій Windows для виявлення аномалій.
2. Витягування та аналіз даних з реєстру Windows для відновлення ключової інформації.
3. Перегляд історії браузера для відстеження веб-активності користувача.

2.18. ПЗ HEXED.IT (ОНЛАЙН-СЕРВІС)

ЛІЦЕНЗІЯ

Freeware. HexEd.it є безкоштовним онлайн-інструментом для редагування файлів у шістнадцятковому (hex) форматі. Використання сервісу не вимагає ліцензійних платежів чи підписок, що дозволяє вільно користуватися ним для особистих та професійних потреб.

РОЗРОБНИК

HexEd.it розроблений компанією Pas de Chocolat.

ОФІЦІЙНИЙ РЕСУРС

<https://hexed.it/>

ОПИС

HexEd.it є онлайн інструментом для перегляду та редагування файлів у шістнадцятковому форматі. Він дає змогу працювати з бінарними файлами без встановлення додаткового програмного забезпечення та може використовуватися для аналізу структури файлів, пошуку службових даних і дослідження окремих байтів.

Можливості програми

Зручний інтерфейс – HexEd.it надає користувачам інтуїтивно зрозумілий інтерфейс, що дозволяє легко редагувати бінарні файли у шістнадцятковому форматі.

Підтримка великих файлів – Інструмент підтримує роботу з великими файлами, забезпечуючи стабільну та швидку роботу.

Пошук за шаблонами – HexEd.it дозволяє здійснювати пошук та заміну байтів за різними шаблонами, що робить процес редагування більш ефективним.

Гнучка система фільтрації – Можливість фільтрувати результати пошуку для точного визначення необхідних змін.

Відображення даних – Інструмент підтримує відображення даних у різних форматах, таких як шістнадцятковий, ASCII, та інші.

Імпорт та експорт файлів – Користувачі можуть легко імпортувати та експортувати файли для подальшого редагування.

Безпечне з'єднання – HexEd.it використовує HTTPS для забезпечення безпечного з'єднання та захисту даних користувачів.

Локальне зберігання даних – Всі операції виконуються локально у браузері користувача, що гарантує конфіденційність даних.

Undo/Redo – Підтримка багаторазового скасування та повторення дій для зручності роботи.

Візуалізація структури файлів – Інструмент надає можливість візуалізувати структуру файлу для кращого розуміння його вмісту.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу бінарних файлів.

1. Редагування і дослідження, HexEd.it дає змогу аналізувати та редагувати бінарні файли, які можуть належати до програм, прошивок або інших цифрових об'єктів. Наприклад, дослідник безпеки використовує HexEd.it для перегляду структури файлу, пошуку характерних сигнатур та аналізу вмісту на рівні байтів.

З точки зору відновлення даних.

2. Інструмент допомагає відновити дані з пошкоджених файлів шляхом ручного редагування та корекції байтів. Як приклад ІТ-фахівець використовує HexEd.it для відновлення важливого документа, який був пошкоджений у процесі передачі або збереження.

З точки зору розробки програмного забезпечення.

3. Відновлення пошкоджених файлів, HexEd.it може застосовуватися для ручного аналізу пошкоджених файлів і коригування окремих байтів, якщо відома структура відповідного формату. Наприклад, ІТ фахівець використовує цей

інструмент для дослідження пошкодженого документа та перевірки службових областей файлу.

ВИСНОВКИ

HexEd.it є потужним та зручним онлайн-інструментом для редагування бінарних файлів у шістнадцятковому форматі. Основні переваги включають:

1. Інтуїтивно зрозумілий інтерфейс, що полегшує редагування та аналіз файлів.
2. Потужні інструменти для пошуку та заміни байтів за різними шаблонами.
3. Всі операції виконуються локально у браузері користувача, що гарантує захист даних.

2.19. ПЗ REDLINE

ЛІЦЕНЗІЯ

Freeware. Redline надається безкоштовно для особистого та комерційного використання. Це дозволяє вільно завантажувати, використовувати та розповсюджувати програму без ліцензійних зборів.

РОЗРОБНИК

Redline розроблений компанією FireEye.

ОФІЦІЙНИЙ РЕСУРС

<https://fireeye.market/apps/211364>

ОПИС

Redline є інструментом для аналізу оперативної пам'яті та форензичного дослідження кінцевих точок. Він дає змогу дослідникам безпеки збирати, аналізувати та візуалізувати дані з комп'ютерних систем з метою виявлення ознак компрометації, шкідливого програмного забезпечення та інших потенційних загроз.

Можливості програми

Захоплення пам'яті передбачає створення повних дамів оперативної пам'яті для подальшого аналізу. Це допомагає виявляти підозрілі процеси, приховану активність та інші аномалії в роботі системи.

Аналіз активних процесів – Інструмент надає детальну інформацію про активні процеси, відкриті порти, завантажені модулі та інші системні ресурси.

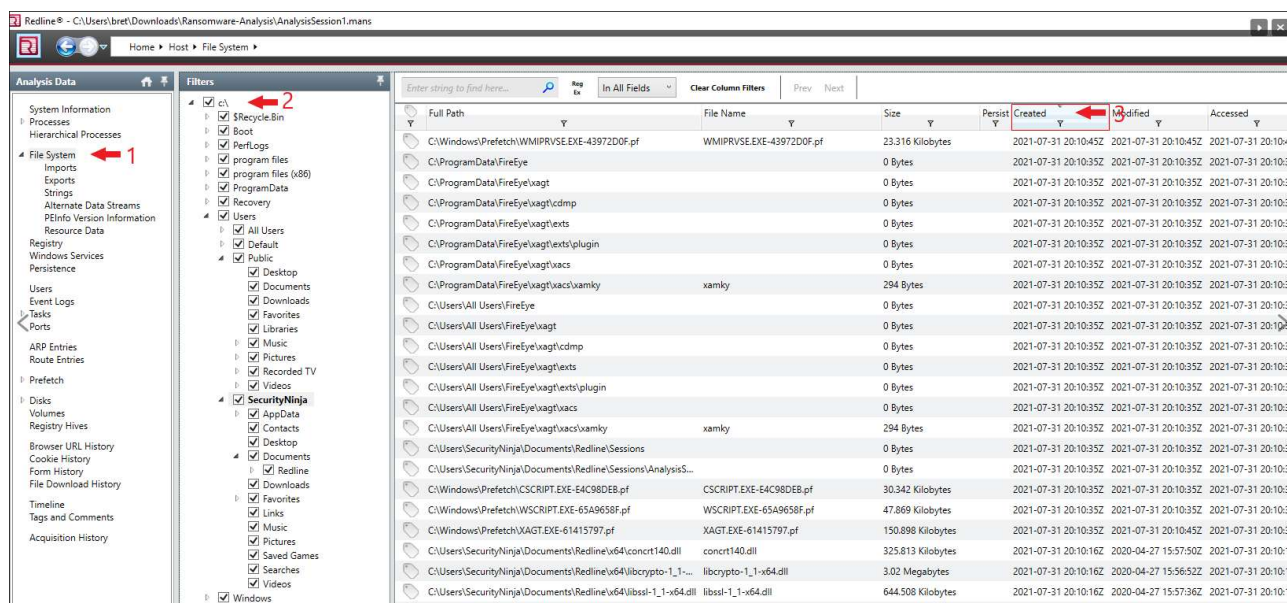


Рисунок 2.51 – Головне вікно ПЗ REDLINE

Ідентифікація шкідливого програмного забезпечення здійснюється шляхом аналізу системних артефактів, зокрема файлів реєстру, журналів подій, активних процесів, мережевих з'єднань та інших слідів діяльності в системі.

Інтерактивний аналіз – Дослідники можуть використовувати інтерактивні інструменти для візуалізації та детального аналізу зібраних даних.

Налаштовувані профілі – Redline дозволяє створювати налаштовувані профілі збору даних, що відповідають конкретним вимогам розслідування.

Автоматизація завдань – Можливість автоматизувати процеси збору та аналізу даних для підвищення ефективності розслідувань.

Детальні звіти – Інструмент генерує детальні звіти про результати аналізу, які включають інформацію про виявлені загрози та підозрілу активність.

Графічна візуалізація – Redline надає інструменти для візуалізації даних, що полегшує аналіз складних взаємозв'язків між різними артефактами.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів безпеки.

1. Redline допомагає дослідникам безпеки виявляти та аналізувати шкідливе ПЗ, яке може перебувати в оперативній пам'яті або системних файлах. Наприклад, аналітик безпеки використовує Redline для захоплення та аналізу дампа пам'яті підозрілої системи і виявляє шкідливий процес, пов'язаний із несанкціонованим доступом до комп'ютера користувача.

З точки зору аналізу пам'яті для виявлення аномалій.

2. Redline дає змогу виявляти нетипові процеси, підозрілі мережеві з'єднання та інші ознаки можливої компрометації системи. Наприклад, ІТ фахівець аналізує активні процеси й мережеві з'єднання на сервері та виявляє підозрілу активність, пов'язану з несанкціонованим передаванням даних.

З точки зору проведення форензичних розслідувань.

3. Redline забезпечує комплексний збір інформації з кінцевих точок, що дає змогу виконувати поглиблений форензичний аналіз і виявляти сліди несанкціонованих дій. Наприклад, форензичний аналітик використовує Redline для формування профілю активності користувача, аналізуючи журнали подій, записи реєстру та інші системні артефакти.

ВИСНОВКИ

Redline є потужним інструментом для збору та аналізу даних з пам'яті та кінцевих точок. Основні переваги включають:

1. Детальний аналіз пам'яті та активних процесів, що допомагає виявляти шкідливе ПЗ та інші аномалії.

2. Інструменти для аналізу різних системних артефактів, таких як файли реєстру та системні журнали.

3. Можливість інтерактивного аналізу та автоматизації завдань для підвищення ефективності розслідувань.

2.20. ПЗ VELOCIRAPTOR

ЛІЦЕНЗІЯ

GNU General Public License v3.0 (GPLv3). Velociraptor є відкритим програмним забезпеченням, яке поширюється під ліцензією GPLv3. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

Velociraptor розроблений компанією Rapid7.

ОФІЦІЙНИЙ РЕСУРС

<https://docs.velociraptor.app/>

ОПИС

Velociraptor є інструментом для розширеного пошуку, збору та аналізу цифрових доказів з кінцевих точок. Він дає змогу фахівцям з кібербезпеки оперативно отримувати необхідні дані з комп'ютерів, серверів та інших вузлів інфраструктури, виконувати моніторинг систем і виявляти ознаки потенційних загроз у реальному часі.

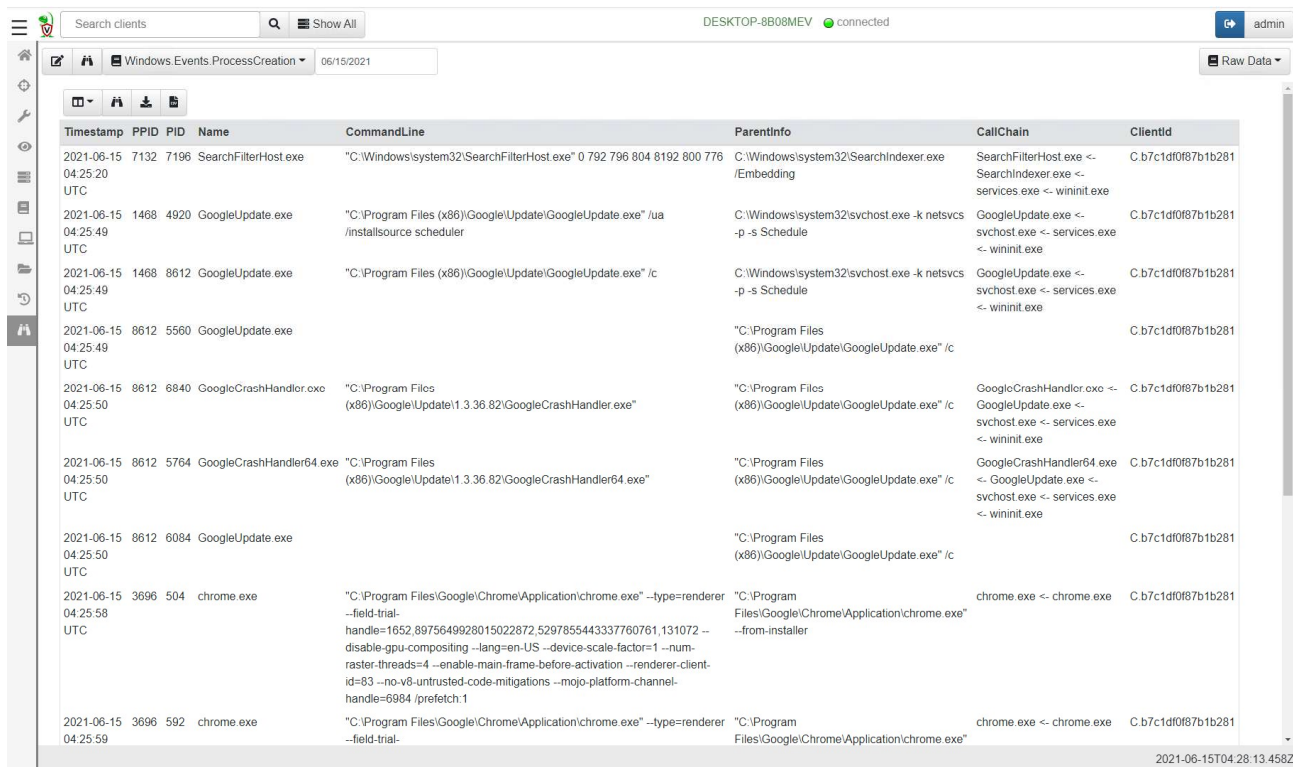
Velociraptor є корисним засобом для фахівців з кібербезпеки, оскільки забезпечує централізований збір даних, аналіз артефактів, моніторинг активності кінцевих точок та підтримку процесів реагування на інциденти.

Можливості програми

Запити VQL, або Velociraptor Query Language, використовуються для створення гнучких запитів, які дають змогу збирати, фільтрувати та аналізувати дані з кінцевих точок відповідно до потреб розслідування.

Масштабованість – Інструмент здатен ефективно працювати у великих корпоративних мережах, обробляючи тисячі кінцевих точок одночасно.

Реальний час – Velociraptor забезпечує моніторинг та аналіз систем у реальному часі, що дозволяє швидко виявляти аномалії та потенційні загрози.



Timestamp	PPID	PID	Name	CommandLine	ParentInfo	CallChain	ClientId
2021-06-15 04:25:20 UTC	7132	7196	SearchFilterHost.exe	"C:\Windows\system32\SearchFilterHost.exe" 0 792 796 804 8192 800 776	C:\Windows\system32\SearchIndexer.exe /Embedding	SearchFilterHost.exe <- SearchIndexer.exe <- services.exe <- wininit.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:49 UTC	1468	4920	GoogleUpdate.exe	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /ua	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule	GoogleUpdate.exe <- svchost.exe <- services.exe <- wininit.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:49 UTC	1468	8612	GoogleUpdate.exe	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule	GoogleUpdate.exe <- svchost.exe <- services.exe <- wininit.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:49 UTC	8612	5560	GoogleUpdate.exe	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c		C:b7c1df0f87b1b281
2021-06-15 04:25:50 UTC	8612	6840	GoogleCrashHandler.exe	"C:\Program Files (x86)\Google\Update\1.3.36.82\GoogleCrashHandler.exe"	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c	GoogleCrashHandler.exe <- GoogleUpdate.exe <- svchost.exe <- services.exe <- wininit.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:50 UTC	8612	5764	GoogleCrashHandler64.exe	"C:\Program Files (x86)\Google\Update\1.3.36.82\GoogleCrashHandler64.exe"	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c	GoogleCrashHandler64.exe <- GoogleUpdate.exe <- svchost.exe <- services.exe <- wininit.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:50 UTC	8612	6084	GoogleUpdate.exe	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c	"C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /c		C:b7c1df0f87b1b281
2021-06-15 04:25:58 UTC	3696	504	chrome.exe	"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --field-trial-handle=1652,8975649928015022872,529785544337760761,131072 --disable-gpu-compositing --lang=en-US --device-scale-factor=1 --num-raster-threads=4 --enable-main-frame-before-activation --renderer-client-id=83 --no-v8-untrusted-code-mitigations --mojo-platform-channel-handle=6984 /prefetch:1	"C:\Program Files\Google\Chrome\Application\chrome.exe" --from-installer	chrome.exe <- chrome.exe	C:b7c1df0f87b1b281
2021-06-15 04:25:59 UTC	3696	592	chrome.exe	"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --field-trial-handle=1652,8975649928015022872,529785544337760761,131072 --disable-gpu-compositing --lang=en-US --device-scale-factor=1 --num-raster-threads=4 --enable-main-frame-before-activation --renderer-client-id=83 --no-v8-untrusted-code-mitigations --mojo-platform-channel-handle=6984 /prefetch:1	"C:\Program Files\Google\Chrome\Application\chrome.exe" --from-installer	chrome.exe <- chrome.exe	C:b7c1df0f87b1b281

Рисунок 2.52 – ПЗ VELOCIRAPTOR, моніторинг

desktop-ng2qvog

DESKTOP-NG2QVOG

connected

+

▶

■

▣

↶

📁

State

Hunt ID

Description

Created

Started

Expires

Limit

Scheduled

Creator

🕒

H.d12438e6

Hunt For all Scheduled tasks

2021-01-22 04:46:04 UTC

2021-01-22 04:46:20 UTC

2021-01-29 04:46:00 UTC

2001

mic

🕒

H.a8fb5253

a8fb

2021-01-19 04:20:29 UTC

2021-01-19 07:08:54 UTC

2021-01-26 04:19:54 UTC

4002

mic

■

H.84858371

848

2021-01-19 03:30:53 UTC

2021-01-19 03:30:53 UTC

2021-01-26 03:30:53 UTC

2000

mic

🕒

H.4c1cee7b

System.VFS.ListDirectory

2021-01-19 01:48:59 UTC

2021-01-19 01:49:03 UTC

2021-01-26 01:48:46 UTC

4002

mic

Overview

Requests

Clients

Notebook

Overview

Artifact Names

Windows.System.TaskScheduler

Hunt ID

H.d12438e6

Creator

mic

Creation Time

2021-01-22 04:46:04 UTC

Expiry Time

2021-01-29 04:46:00 UTC

State

RUNNING

Ops/Sec

Unlimited

Parameters

Windows.Svstem.TaskScheduler

Results

Total scheduled

2001

Finished clients

402

Download Results

📁

Available Downloads

name

size

date

Рисунок 2.53 – ПЗ VELOCIRAPTOR, пошук з використанням криміналістичних артефактів

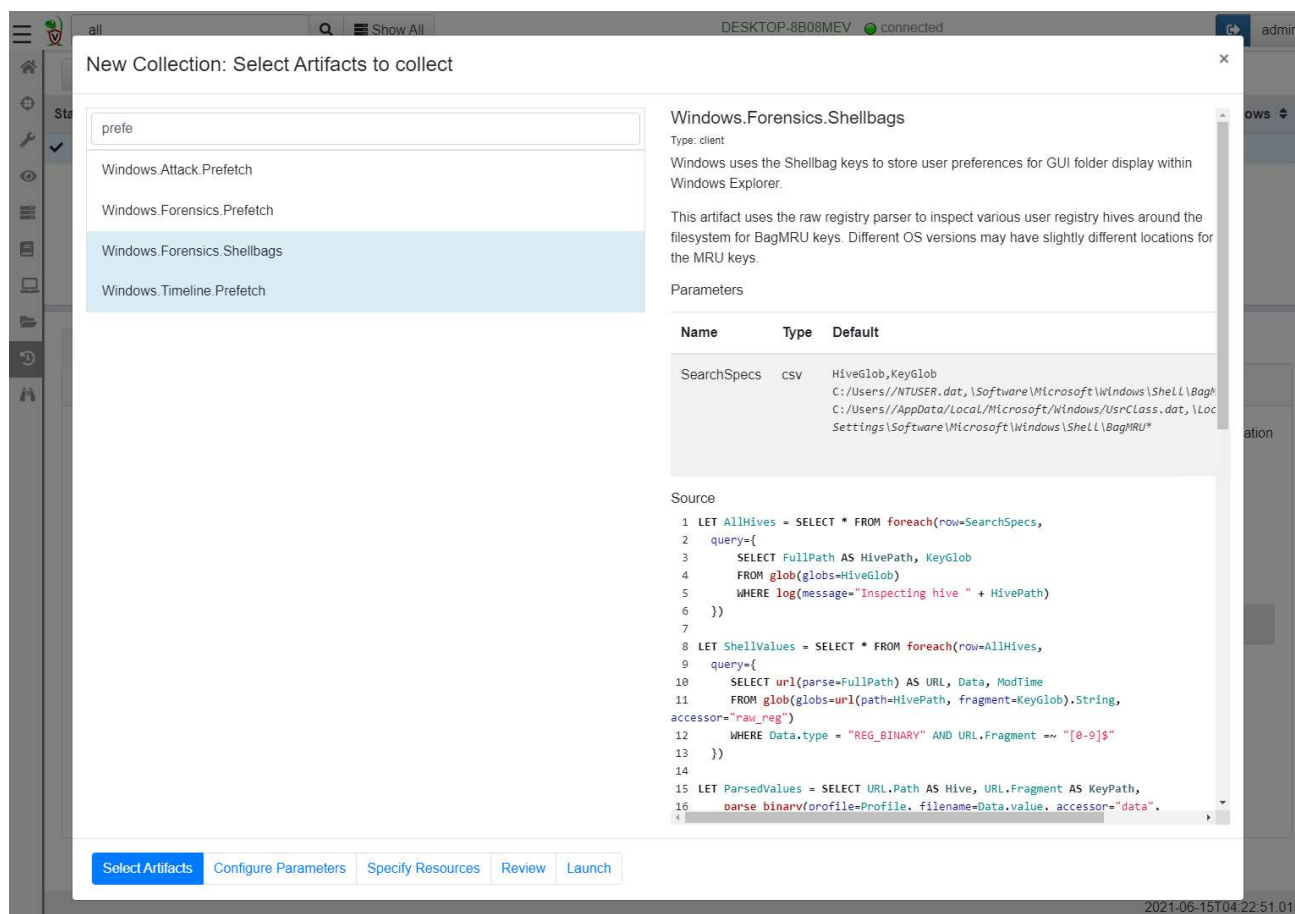


Рисунок 2.54 – ПЗ VELOCIRAPTOR, збір цифрових криміналістичних доказів

Автоматизація завдань – Можливість автоматизувати завдання моніторингу та реагування, що підвищує ефективність роботи команди безпеки.

Форензичний аналіз у Velociraptor передбачає збір і дослідження різних типів цифрових артефактів, зокрема журналів подій, файлів реєстру, даних про мережеву активність, процеси, файли та інші системні об'єкти.

Інтерактивний інтерфейс – Забезпечує зручний інтерфейс для інтерактивного аналізу та візуалізації даних, що полегшує роботу з великими обсягами інформації.

Плагіни та розширення дають змогу розширювати функціональні можливості Velociraptor, додавати нові сценарії збору даних та інтегрувати інструмент з іншими системами кібербезпеки.

API доступ – Надає API для інтеграції з іншими інструментами та автоматизації процесів.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів безпеки.

1. Velociraptor дає змогу фахівцям з кібербезпеки оперативно виявляти інциденти безпеки та реагувати на них шляхом збору даних з кінцевих точок у реальному часі. Наприклад, аналітик безпеки використовує Velociraptor для моніторингу мережевої активності та виявляє підозрілу поведінку, пов'язану з компрометацією облікового запису користувача. Зібрані дані допомагають визначити джерело загрози та своєчасно вжити заходів для її нейтралізації.

З точки зору форензичний аналіз.

2. Velociraptor надає засоби для детального аналізу цифрових артефактів, зокрема журналів подій, файлів реєстру, мережевих даних та інших системних об'єктів. Наприклад, форензичний аналітик використовує Velociraptor для дослідження даних з кількох кінцевих точок після виявлення ознак шкідливої активності. Інструмент допомагає знайти сліди шкідливого коду, встановити характер інциденту та визначити можливий вектор атаки.

З точки зору моніторингу корпоративної мережі.

3. Velociraptor дає змогу одночасно моніторити велику кількість кінцевих точок, автоматизуючи процеси збору, оброблення та аналізу даних для виявлення аномалій. Наприклад, ІТ відділ великої організації використовує Velociraptor для постійного контролю інфраструктури, автоматичного збору відомостей про активність користувачів і пристроїв. Це допомагає швидко виявляти підозрілу активність та запобігати можливим загрозам.

ВИСНОВКИ

Velociraptor є потужним інструментом для моніторингу, збору та аналізу даних з кінцевих точок. Основні переваги включають:

1. Використання мови запитів VQL для створення складних запитів та збору даних.
2. Можливість моніторингу та реагування на загрози у реальному часі.
3. Інструменти для глибокого аналізу системних артефактів, що допомагають виявляти сліди компрометації та шкідливого ПЗ.
4. Підтримка масштабованої роботи у великих мережах та можливість інтеграції з іншими інструментами безпеки.

2.21. ПЗ FTK IMAGER

ЛІЦЕНЗІЯ

Freeware. FTK Imager є безкоштовним програмним забезпеченням, яке можна використовувати для особистих та комерційних цілей. Його можна вільно завантажувати і використовувати без ліцензійних зборів, але розширені функції можуть вимагати придбання ліцензії.

РОЗРОБНИК

FTK Imager розроблений компанією AccessData.

ОФІЦІЙНИЙ РЕСУРС

<https://www.exterro.com/digital-forensics-software/ftk-imager>

ОПИС

FTK Imager є інструментом для створення образів дисків та аналізу цифрових доказів. Він дає змогу фахівцям з комп'ютерної криміналістики створювати копії носіїв інформації, вилучати файли, переглядати вміст дисків і зберігати дані для подальшого розслідування. Програма також забезпечує можливість роботи з інформацією без внесення змін до оригінальних даних.

FTK Imager є корисним засобом для фахівців з комп'ютерної криміналістики, оскільки надає можливості для створення образів дисків, аналізу цифрових доказів, вилучення потрібних файлів та перевірки цілісності даних.

Можливості програми

Повне копіювання – FTK Imager дозволяє створювати точні копії (образи) дисків, зберігаючи всі дані для подальшого аналізу. Це включає секторне копіювання, яке зберігає кожен байт оригінального диска.

Формати зображень – Підтримка різних форматів зображень, таких як E01, DD, RAW та SMART.

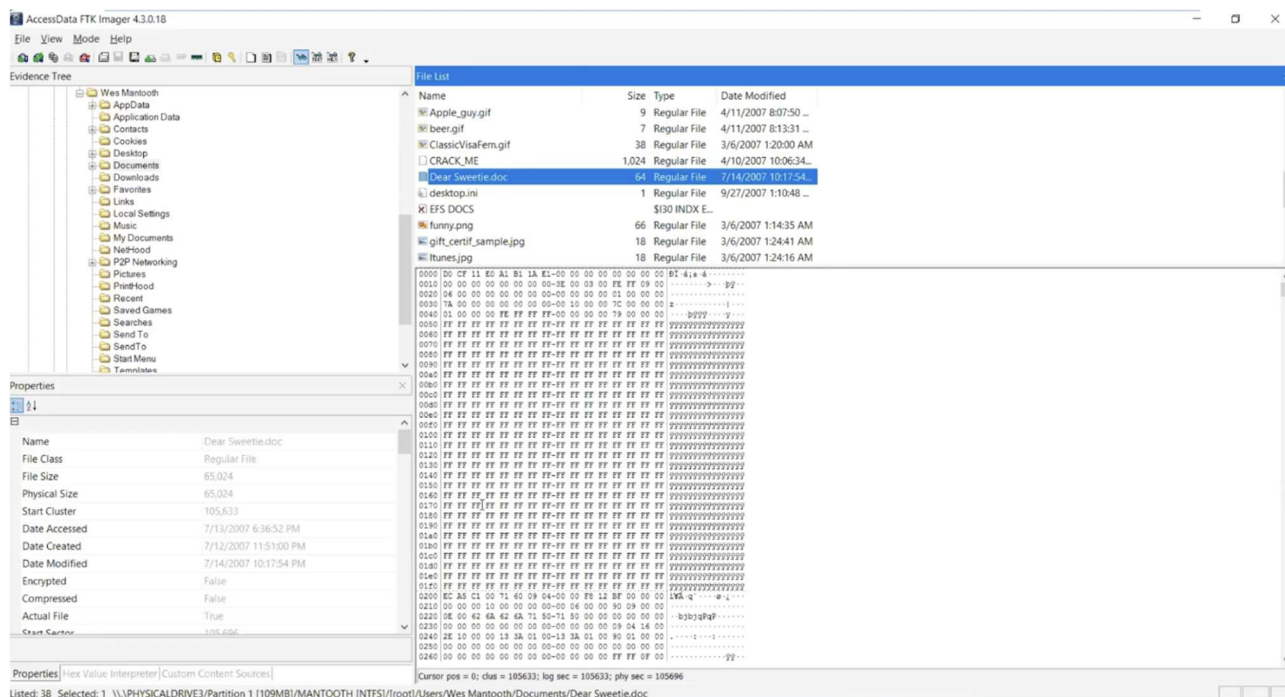


Рисунок 2.55 – Головне вікно ПЗ FTK IMAGER

Перегляд файлів – Можливість перегляду вмісту дисків та файлів без внесення змін до оригінальних даних. FTK Imager дозволяє досліджувати структуру директорій, переглядати файли та метадані.

Видобування файлів – Інструмент дозволяє витягувати файли та папки з зображень дисків для подальшого аналізу.

Хешування – FTK Imager використовує алгоритми хешування (MD5, SHA-1) для перевірки цілісності даних, що забезпечує точність копіювання та аналізу.

Звіти про хешування – Генерація звітів про хешування для документування процесу копіювання та аналізу даних.

Відновлення видалених файлів – Можливість відновлення та аналізу видалених файлів, що можуть містити важливу інформацію для розслідування.

Аналіз метаданих – FTK Imager надає детальну інформацію про метадані файлів, що допомагає виявляти сліди змін та дій користувача.

Клонування дисків – Програма дозволяє створювати точні клони дисків для використання в розслідуваннях.

Знімки в реальному часі – Можливість створення знімків системи в реальному часі для аналізу поточного стану системи.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування інцидентів безпеки.

1. Захоплення та збереження даних, FTK Imager використовується для створення образів дисків і збереження цифрових доказів без ризику зміни оригінальної інформації. Наприклад, аналітик безпеки створює образ жорсткого диска підозрілого комп'ютера за допомогою FTK Imager, щоб зберегти файли, журнали, метадані та інші дані для подальшого аналізу й розслідування. З точки зору форензичного аналізу даних.

2. Аналіз файлових систем, FTK Imager дозволяє фахівцям детально аналізувати файлові системи, відновлювати видалені файли та перевіряти метадані. Як приклад форензичний аналітик використовує FTK Imager для перегляду та видобування файлів з образу диска, включаючи видалені файли та метадані, для створення детального звіту про активність користувача.

З точки зору перевірки цілісності даних.

3. Хешування даних, FTK Imager забезпечує можливість генерувати хеш-значення для перевірки цілісності даних під час створення образів дисків. Як приклад ІТ-фахівець використовує FTK Imager для створення образу диска і генерування хеш-значень MD5 та SHA-1, щоб гарантувати, що дані не були змінені під час копіювання.

ВИСНОВКИ

FTK Imager є потужним інструментом для фахівців з комп'ютерної криміналістики та ІТ-безпеки. Основні переваги включають:

1. Можливість створювати точні копії дисків для аналізу та зберігання цифрових доказів.
2. Перегляд та видобування файлів, включаючи відновлення видалених даних.
3. Хешування даних для забезпечення їхньої цілісності та достовірності.
4. Глибокий аналіз метаданих та інших цифрових артефактів.

2.22. ПЗ YARA GENERATOR (YARGEN)

ЛІЦЕНЗІЯ

GNU General Public License v3.0 (GPLv3). Yargen є відкритим програмним забезпеченням, яке поширюється під ліцензією GPLv3. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

Yargen розроблений Дином Ферарісом (Dean Farrar).

ОФІЦІЙНИЙ РЕСУРС

<https://github.com/Neo23x0/yarGen>

ОПИС

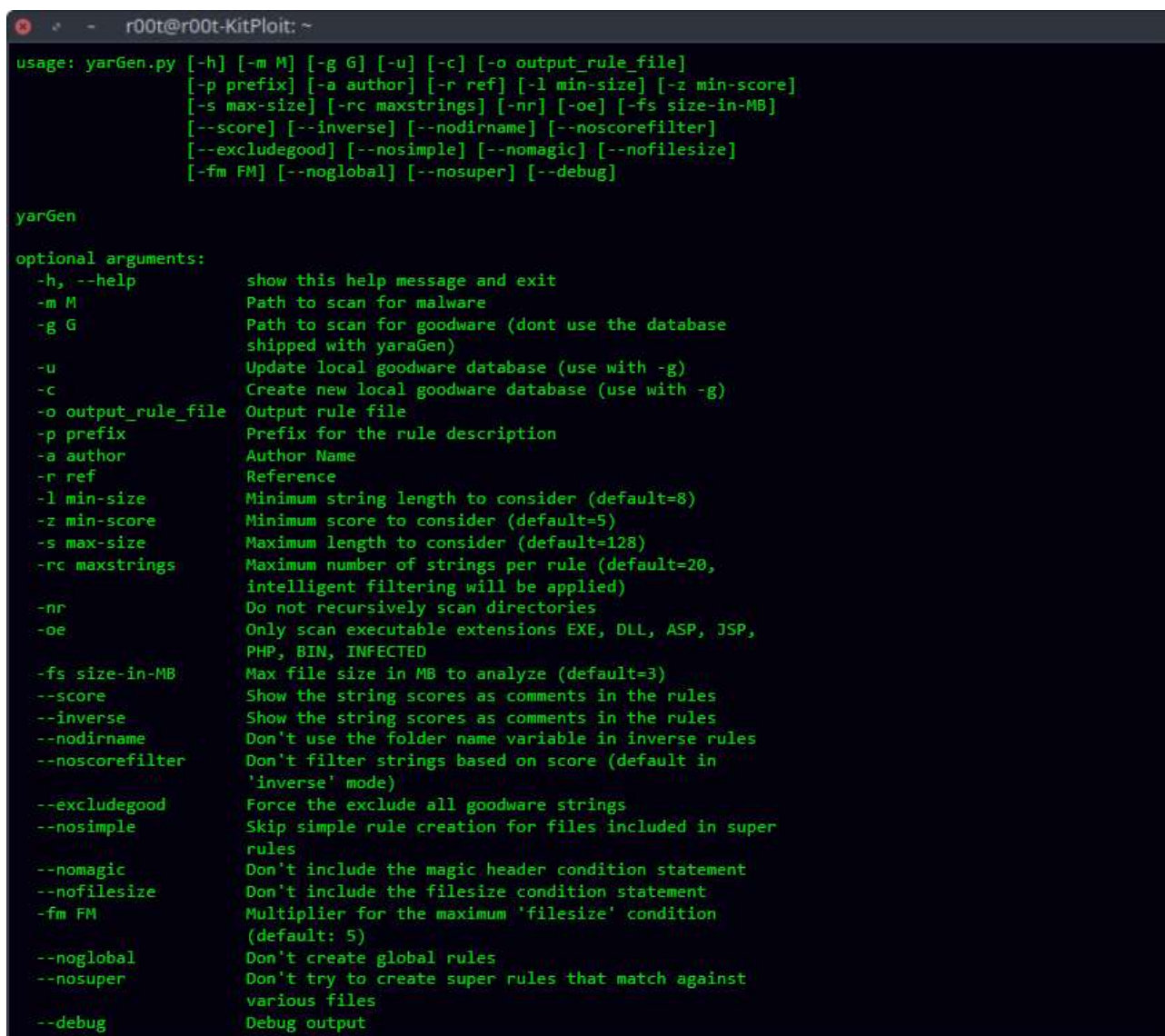
YARA Generator, або Yargen, є інструментом для автоматичного створення YARA правил, які використовуються для виявлення шкідливого програмного забезпечення та інших кіберзагроз. Він допомагає дослідникам безпеки формувати ефективні правила на основі аналізу зразків шкідливого ПЗ, що спрощує процес розроблення сигнатур для подальшого виявлення подібних загроз.

Yargen є корисним засобом для дослідників безпеки та аналітиків, оскільки забезпечує автоматизоване створення YARA правил і дає змогу швидше ідентифікувати потенційно небезпечні об'єкти, аналізувати зразки шкідливого програмного забезпечення та реагувати на виявлені загрози.

Можливості програми

Генерація на основі зразків – Yargen аналізує зразки шкідливого ПЗ та автоматично генерує YARA правила, які можуть бути використані для виявлення подібних загроз.

Конфігураційні файли – Інструмент підтримує конфігураційні файли, які дозволяють налаштовувати процес генерації правил відповідно до конкретних потреб.



```
r00t@r00t-KitPloit: ~  
usage: yarGen.py [-h] [-m M] [-g G] [-u] [-c] [-o output_rule_file]  
                [-p prefix] [-a author] [-r ref] [-l min-size] [-z min-score]  
                [-s max-size] [-rc maxstrings] [-nr] [-oe] [-fs size-in-MB]  
                [--score] [--inverse] [--nodirname] [--noscorefilter]  
                [--excludegood] [--nosimple] [--nomagic] [--nofilesize]  
                [-fm FM] [--noglobal] [--nosuper] [--debug]  
  
yarGen  
optional arguments:  
  -h, --help            show this help message and exit  
  -m M                  Path to scan for malware  
  -g G                  Path to scan for goodware (dont use the database  
                        shipped with yaraGen)  
  -u                    Update local goodware database (use with -g)  
  -c                    Create new local goodware database (use with -g)  
  -o output_rule_file  Output rule file  
  -p prefix             Prefix for the rule description  
  -a author            Author Name  
  -r ref               Reference  
  -l min-size          Minimum string length to consider (default=8)  
  -z min-score         Minimum score to consider (default=5)  
  -s max-size          Maximum length to consider (default=128)  
  -rc maxstrings        Maximum number of strings per rule (default=20,  
                        intelligent filtering will be applied)  
  -nr                  Do not recursively scan directories  
  -oe                  Only scan executable extensions EXE, DLL, ASP, JSP,  
                        PHP, BIN, INFECTED  
  -fs size-in-MB       Max file size in MB to analyze (default=3)  
  --score              Show the string scores as comments in the rules  
  --inverse            Show the string scores as comments in the rules  
  --nodirname          Don't use the folder name variable in inverse rules  
  --noscorefilter      Don't filter strings based on score (default in  
                        'inverse' mode)  
  --excludegood        Force the exclude all goodware strings  
  --nosimple           Skip simple rule creation for files included in super  
                        rules  
  --nomagic            Don't include the magic header condition statement  
  --nofilesize         Don't include the filesize condition statement  
  -fm FM               Multiplier for the maximum 'filesize' condition  
                        (default: 5)  
  --noglobal           Don't create global rules  
  --nosuper            Don't try to create super rules that match against  
                        various files  
  --debug              Debug output
```

Рисунок 2.56 – Консольне ПЗ YARGEN, параметри командного рядка

Широкий спектр форматів – Yargen підтримує аналіз файлів різних форматів, таких як виконувані файли, документи, скрипти та інші.

Аналіз метаданих – Інструмент враховує метадані файлів для створення більш точних та ефективних правил.

Пошук рядків та сигнатур – Yargen шукає у файлах характерні рядки та сигнатури, які можуть вказувати на наявність шкідливого ПЗ.

Ідентифікація патернів – Інструмент аналізує патерни поведінки зразків шкідливого ПЗ для створення детальних правил виявлення.

```
1  /*
2  Yara Rule Set
3  Author: YarGen Rule Generator
4  Date: 2015-07-09
5  Identifier: bin
6  */
7
8  /* Rule Set ----- */
9
10 rule backdoor {
11     meta:
12         description = "Auto-generated rule - file backdoor.exe"
13         author = "YarGen Rule Generator"
14         reference = "not set"
15         date = "2015-07-09"
16         hash = "bad8c7e6836b9a5679bfac0bc7483091e8e168f2"
17     strings:
18         $s0 = "%systemroot%\System32\rundll32.exe \" fullword ascii /* PESTudio Blacklist: str
19         $s1 = "c:\\Agenti\\SimpleVector\\Release\\SimpleVector.pdb" fullword ascii /* score: '28.
20         $s2 = "GetCurrentProcessID" fullword ascii /* PESTudio Blacklist: strings */ /* score: '2
21         $s3 = "<requestedExecutionLevel level=\"highestAvailable\" uiAccess=\"false\"/>" fullword
22         $s4 = "SOFTWARE\\Microsoft\\VisualStudio\\9.0\\Setup\\VS" fullword ascii /* PESTudio Blac
23         $s5 = "BG:\\oMpp" fullword ascii /* score: '12.00' */
24         $s6 = "vvKPP80k.mKn" fullword ascii /* score: '12.00' */
25         $s7 = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\" ?><assembly xmlns=\"u
26         $s8 = "0b55581e0a49451a01584c2a1d5223224559566318244a41405b172e11161932" fullword ascii /
27         $s9 = "SimpleVector, Version 1.0" fullword wide /* score: '9.00' */
28         $s10 = "* GN3?" fullword ascii /* score: '7.00' */
29         $s11 = "SIMPLEVECTOR" fullword wide /* score: '6.50' */
30         $s12 = ".OcL/2" fullword ascii /* score: '6.00' */
31         $s13 = "VH.IYl" fullword ascii /* score: '6.00' */
32         $s14 = "uKmtnQzd78" fullword ascii /* score: '5.00' */
33         $s15 = "About SimpleVector" fullword wide /* score: '5.00' */
34     condition:
35         uint16(0) == 0x5a4d and filesize < 3785KB and all of them
36 }
37
```

Рисунок 2.57 – Консольне ПЗ YARGEN, приклад сгенерованих правил YARA

Плагіни та модулі – Підтримка плагінів та модулів дозволяє розширювати функціональність Yargen та інтегрувати його з іншими інструментами кібербезпеки.

API доступ – Надає API для автоматизації процесу створення правил та інтеграції з іншими системами.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу зразків шкідливого ПЗ.

1. Автоматичне створення правил. Yargen дає змогу дослідникам безпеки швидко створювати YARA правила на основі зразків шкідливого програмного забезпечення. Це допомагає виявляти, класифікувати та ідентифікувати нові загрози. Наприклад, аналітик безпеки використовує Yargen для аналізу зразків шкідливого ПЗ, виявлених у корпоративній мережі, після чого автоматично формує YARA правила для пошуку подібних об'єктів.



```
[+] Writing results to new file: check-results_retrohunt_results (2).csv

HASH: 522fd9b35323af55113455d823571f71332e53dde988c2eb41395cf6b0c15805 [COMMENT: Sofacy_Oct17_1 /subfile
VIRUS: Microsoft: Trojan:Win32/Gamafeshi.A / Kaspersky: HEUR:Trojan.Win32.Sofacy.gen / McAfee: RDN/Generic.dx / CrowdStrike: malicious_c
onfidence_100% (D) / TrendMicro: TR0J_GEN.R00EC00JH17 / ESET-NOD32: Win32/Sednit.BN / Symantec: Trojan Horse / F-Secure: Trojan.GenericK
D.12495105 / Sophos: Mal/Generic-S / GData: Trojan.GenericKD.12495105
TYPE: Win32_DLL FILENAMES: netwf.dat
FIRST_SUBMITTED: 2017-10-16 10:39:22 UTC ( 1 week ago ) LAST_SUBMITTED: 2017-10-23 00:42:09
RESULT: 40 / 66
[!] Sample is available on hybrid-analysis.com

HASH: 335565711db93cd02d948f472c51598be4d62d60f70f25a20449c07eae36c8c5 [COMMENT: Sofacy_Oct17_1
VIRUS: Kaspersky: Trojan.Win32.Agent2.mmk / McAfee: RDN/Generic.dx / CrowdStrike: malicious_confidence_100% (D) / TrendMicro: TR0J_GEN.R
00EC00JG17 / ESET-NOD32: a variant of Win32/Sednit.BN / Symantec: Trojan Horse / GData: Win32.Trojan.Agent.EH6082
TYPE: Win32_DLL FILENAMES: B0E1EB48088433797AE2A8B0A7211415
FIRST_SUBMITTED: 2017-10-11 11:26:34 UTC ( 1 week, 5 days ago ) LAST_SUBMITTED: 2017-10-17 09:30:51
RESULT: 23 / 66
[!] Imphash - appeared 1 times in this batch a2d1be6502b4b3c28959a4fb0196ea45
[!] Sample is available on hybrid-analysis.com

HASH: 3b87bfb837339445987cdf2e97169cb0c63072dc1d5bffa8ffba4f108a410988 [COMMENT: Sofacy_Oct17_2
VIRUS: Kaspersky: HEUR:Trojan.Win32.Sofacy.gen / McAfee: RDN/Generic.dx / CrowdStrike: malicious_confidence_100% (D) / TrendMicro: TR0J_
GEN.R011C00JC17 / ESET-NOD32: a variant of Win32/Sednit.BN / Symantec: Trojan.Gen.2 / F-Secure: Gen:Variant.Razy.163623 / Sophos: Mal/Ge
neric-S / GData: Gen:Variant.Razy.163623
TYPE: Win32_DLL FILENAMES: nad.dll, ED9DCB939203C17C25C6E7025B73029F
FIRST_SUBMITTED: 2017-10-11 11:25:36 UTC ( 1 week, 5 days ago ) LAST_SUBMITTED: 2017-10-18 17:23:04
RESULT: 39 / 65
[!] Sample is available on hybrid-analysis.com

HASH: 1140c624fbfe28b9ef19fef2e9aa251adfbe8c157020d5f0356d88b4d80c2c88 [COMMENT: Sofacy_Oct17_2
VIRUS: Kaspersky: HEUR:Trojan.Win32.Sofacy.gen / McAfee: RDN/Generic.dx / CrowdStrike: malicious_confidence_100% (D) / TrendMicro: TR0J_
GEN.R047C00GK17 / ESET-NOD32: a variant of Win32/Sednit.BN / Symantec: Trojan.Gen.2 / F-Secure: Gen:Variant.Razy.163623 / Sophos: Mal/Ge
neric-S / GData: Gen:Variant.Razy.163623
TYPE: Win32_DLL FILENAMES: inxtray.dll, inxt.dat
FIRST_SUBMITTED: 2017-07-13 13:54:42 UTC ( 3 months, 1 week ago ) LAST_SUBMITTED: 2017-10-05 08:18:13
RESULT: 43 / 64
[!] Imphash - appeared 1 times in this batch 13344e2a717849489bcd93692f9646f7
[!] Sample is available on hybrid-analysis.com

HASH: 522fd9b35323af55113455d823571f71332e53dde988c2eb41395cf6b0c15805 [COMMENT: Sofacy_Oct17_1
VIRUS: Microsoft: Trojan:Win32/Gamafeshi.A / Kaspersky: HEUR:Trojan.Win32.Sofacy.gen / McAfee: RDN/Generic.dx / CrowdStrike: malicious_c
onfidence_100% (D) / TrendMicro: TR0J_GEN.R00EC00JH17 / ESET-NOD32: Win32/Sednit.BN / Symantec: Trojan Horse / F-Secure: Trojan.GenericK
D.12495105 / Sophos: Mal/Generic-S / GData: Trojan.GenericKD.12495105
TYPE: Win32_DLL FILENAMES: netwf.dat
FIRST_SUBMITTED: 2017-10-16 10:39:22 UTC ( 1 week ago ) LAST_SUBMITTED: 2017-10-23 00:42:09
RESULT: 40 / 66
[!] Imphash - appeared 2 times in this batch a2d1be6502b4b3c28959a4fb0196ea45
[!] Sample is available on hybrid-analysis.com

HASH: ef027405492b0719437eb58c3d2774c87845f38c40040bbbbbcc09a4e3dd18 [COMMENT: Sofacy_Oct17_2
VIRUS: Kaspersky: UDS: DangerousObject.Multi.Generic / McAfee: RDN/Generic.hbg / CrowdStrike: malicious_confidence_100% (D) / TrendMicro:
TROJ_GEN.R002C0PJ817 / ESET-NOD32: a variant of Win32/Sednit.BQ / Symantec: Trojan.Gen.2 / F-Secure: Gen:Variant.Razy.163623 / Sophos:
Mal/Generic-S / GData: Gen:Variant.Razy.163623
TYPE: Win32_DLL FILENAMES: netwf.dll
FIRST_SUBMITTED: 2017-10-05 05:52:08 UTC ( 2 weeks, 4 days ago ) LAST_SUBMITTED: 2017-10-10 23:06:27
RESULT: 38 / 66
[!] Imphash - appeared 2 times in this batch 13344e2a717849489bcd93692f9646f7
[!] Sample is available on hybrid-analysis.com
```

Рисунок 2.58 – Консольне ПЗ YARGEN, приклад застосування автоматично сгенерованих правил YARA

З точки зору покращення існуючих YARA правил.

2. Yargen допомагає вдосконалювати наявні YARA правила шляхом аналізу нових зразків шкідливого програмного забезпечення та додавання актуальних сигнатур і патернів. Наприклад, дослідник безпеки використовує Yargen для аналізу нових зразків шкідливого ПЗ, оновлює чинні правила та підвищує точність виявлення загроз.

З точки зору форензичний аналіз.

3. Yargen застосовується для створення YARA правил, які допомагають виявляти шкідливе ПЗ під час форензичних розслідувань та аналізу даних з різних системних джерел. Наприклад, форензичний аналітик використовує Yargen для формування правил, що дають змогу знаходити ознаки шкідливої активності на комп'ютерах, які досліджуються в межах інциденту.

ВИСНОВКИ

Yargen є ефективним інструментом для автоматизованого створення YARA правил, який спрощує процес виявлення, класифікації та ідентифікації загроз. До основних переваг цього інструменту належать:

1. Швидка генерація YARA правил на основі зразків шкідливого ПЗ.
2. Аналіз файлів різних форматів для створення універсальних правил.
3. Виявлення характерних рядків та сигнатур, а також аналіз патернів поведінки.
4. Підтримка плагінів та модулів для розширення функціональності та інтеграції з іншими інструментами безпеки.

2.23. ПЗ FLARE-FLOSS (FIREEYE LABS OBFUSCATED STRING SOLVER)

ЛІЦЕНЗІЯ

Apache License 2.0. FLARE-FLOSS є відкритим програмним забезпеченням, яке поширюється під ліцензією Apache License 2.0. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, навіть у комерційних цілях, за умови дотримання умов ліцензії.

РОЗРОБНИК

FLARE-FLOSS розроблений командою FLARE (FireEye Labs Advanced Reverse Engineering) компанії FireEye.

ОФІЦІЙНИЙ РЕСУРС

Вихідний код <https://github.com/mandiant/flare-floss>

Документація алгоритму роботи <https://github.com/mandiant/flare-floss/blob/master/doc/theory.md>

ОПИС

FLARE FLOSS, або FireEye Labs Obfuscated String Solver, є інструментом для аналізу та розшифрування обфускованих рядків у бінарних файлах. Він розробляється командою FLARE компанії FireEye та допомагає дослідникам безпеки виявляти приховані рядки, які можуть свідчити про шкідливу активність або містити важливі цифрові артефакти.

FLARE FLOSS є важливим засобом для дослідників безпеки, оскільки забезпечує ефективне виявлення, аналіз і розшифрування обфускованих рядків у бінарних файлах. Його використання сприяє дослідженню шкідливого програмного забезпечення, виявленню прихованих команд та документуванню ознак потенційно небезпечної активності.

FLOSS TIGHT STRINGS (55)			
Function	Function Offset	Frame Offset	String
0x140001060	0x1400010b2	0x20	%d%02d%02d
0x140001130	0x140001198	0x40	bcrypt.dll
0x140001130	0x140001268	0x158	BCryptOpenAlgorithmProvider
0x140001130	0x140001336	0x268	BCryptImportKeyPair
0x140001130	0x14000141e	0x398	BCryptVerifySignature
0x140001130	0x14000151e	0x478	BCryptCloseAlgorithmProvider
0x1400019ec	0x140001a8f	0x20	ReadFile
0x1400019ec	0x140001b16	0xa8	kernel32.dll
0x140001bd8	0x140001c42	0x70	GetTempPathW
0x140001bd8	0x140001cc2	0x148	kernel32.dll
0x140001bd8	0x140001d8a	0x1f0	~pkg%d%S
0x140001e78	0x140001ef9	0x150	Date
0x140001e78	0x140001faa	0x308	HttpQueryInfoA
0x140001e78	0x14000202b	0x420	wininet.dll
0x140001e78	0x140002109	0x5b8	Set-Cookie
0x14000251c	0x14000266c	0x6b0	.bazar
0x14000251c	0x14000272e	0xd88	%i.%i.%i.%i
0x14000251c	0x1400028a4	0x1430	Host: %s
0x14000251c	0x140002976	0x1b88	update: %s
0x14000251c	0x140002b0a	0x2288	XTag
0x14000251c	0x140002c71	0x2938	InternetQueryDataAvailable
0x14000251c	0x140002d05	0x3050	wininet.dll
0x14000251c	0x140002e32	0x36f8	InternetReadFile
0x140002fe4	0x140003054	0x520	CoInitialize
0x140002fe4	0x1400030cc	0xa08	ole32.dll
0x140002fe4	0x14000318c	0xfd0	CoInitializeSecurity

Рисунок 2.59 – Консольне ПЗ FLARE-FLOSS, приклад роботи

Можливості програми

Автоматична розшифровка – FLARE-FLOSS автоматично ідентифікує та розшифровує обфусцьовані рядки в бінарних файлах, що дозволяє дослідникам швидко отримати приховану інформацію.

Підтримка різних методів обфускації – Інструмент здатен розшифровувати рядки, зашифровані різними методами, що робить його універсальним для аналізу різних типів шкідливого ПЗ.

Підтримка різних форматів – FLARE-FLOSS підтримує аналіз бінарних файлів у різних форматах, включаючи виконувані файли для Windows, Linux та інших операційних систем.

Інтеграція з іншими інструментами – Інструмент можна використовувати разом з іншими засобами аналізу для більш комплексного дослідження.

Командний рядок – FLARE-FLOSS працює з командного рядка, що дозволяє легко інтегрувати його у скрипти та автоматизовані процеси аналізу.

Документація та приклади – Детальна документація та приклади використання полегшують освоєння інструменту та його застосування в різних сценаріях.

Швидкий аналіз – FLARE-FLOSS забезпечує швидке виконання аналізу та розшифровки рядків, що дозволяє зменшити час, необхідний для розслідування інцидентів безпеки.

Виявлення прихованих загроз – Розшифровуючи обфусьовані рядки, FLARE-FLOSS допомагає виявляти приховані загрози та отримувати критичну інформацію для подальшого аналізу.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розслідування шкідливого програмного забезпечення.

1. Аналіз обфусьованих рядків. FLARE-FLOSS дозволяє дослідникам швидко розшифровувати приховані рядки в бінарних файлах, що допомагає виявляти критичну інформацію про шкідливе ПЗ. Як приклад аналітик безпеки використовує FLARE-FLOSS для розшифровки рядків у шкідливому файлі, виявляючи команди, що використовуються шкідливим ПЗ для зв'язку з сервером управління та контролю (C&C).

З точки зору розслідування інцидентів безпеки.

2. Інструмент допомагає виявляти приховані команди та інші важливі рядки, які можуть свідчити про потенційно шкідливу активність або дії зловмисників. Як приклад під час розслідування інциденту безпеки аналітик використовує FLARE-FLOSS для аналізу підозрілого файлу та виявляє приховані команди, що активують шкідливу функціональність.

З точки зору форензичного аналізу.

3. Аналіз підозрілих файлів. FLARE-FLOSS дозволяє швидко отримувати корисну інформацію з підозрілих файлів, полегшуючи роботу форензичних аналітиків. Як приклад форензичний аналітик використовує FLARE-FLOSS для

аналізу бінарного файлу, розшифровуючи рядки, що містять шляхи до важливих системних файлів і реєстру.

ВИСНОВКИ

FLARE-FLOSS є незамінним інструментом для розшифровки обфусцьованих рядків у бінарних файлах. Основні переваги включають:

1. Швидка і ефективна розшифровка прихованих рядків, що дозволяє дослідникам отримати важливу інформацію.
2. Аналіз файлів у різних форматах для створення універсальних правил.
3. Підтримка командного рядка і можливість інтеграції з іншими інструментами кібербезпеки.
4. Швидке виконання аналізу та розшифровки рядків, що дозволяє зменшити час розслідування інцидентів безпеки.

2.24. ПЗ API MONITOR

ЛІЦЕНЗІЯ

Freeware. API Monitor є безкоштовним програмним забезпеченням, яке можна використовувати для особистих та комерційних цілей. Його можна вільно завантажувати і використовувати без ліцензійних зборів.

РОЗРОБНИК

API Monitor розроблений компанією Rohitab Batra.

ОФІЦІЙНИЙ РЕСУРС

<http://www.rohitab.com/apimonitor>

ОПИС

API Monitor інструмент для перехоплення та моніторингу викликів API у реальному часі. Дозволяє розробникам та дослідникам безпеки відслідковувати взаємодію між ПЗ та ОС, виявляючи, які функції викликаються, з якими параметрами та який результат вони повертають. API Monitor підтримує широкий спектр Windows API, включаючи ті, що використовуються для мережевих операцій, роботи з файлами, процесами та багато іншого.

Можливості програми

Перехоплення в реальному часі – API Monitor перехоплює виклики API у реальному часі, дозволяючи користувачам бачити, які функції викликаються програмами.

Детальний аналіз – Інструмент надає детальну інформацію про кожен виклик API, включаючи вхідні та вихідні параметри, а також результати виклику.

Windows API – Підтримує перехоплення викликів широкого спектру Windows API, що охоплюють операції з файлами, процесами, мережевими з'єднаннями та інше.

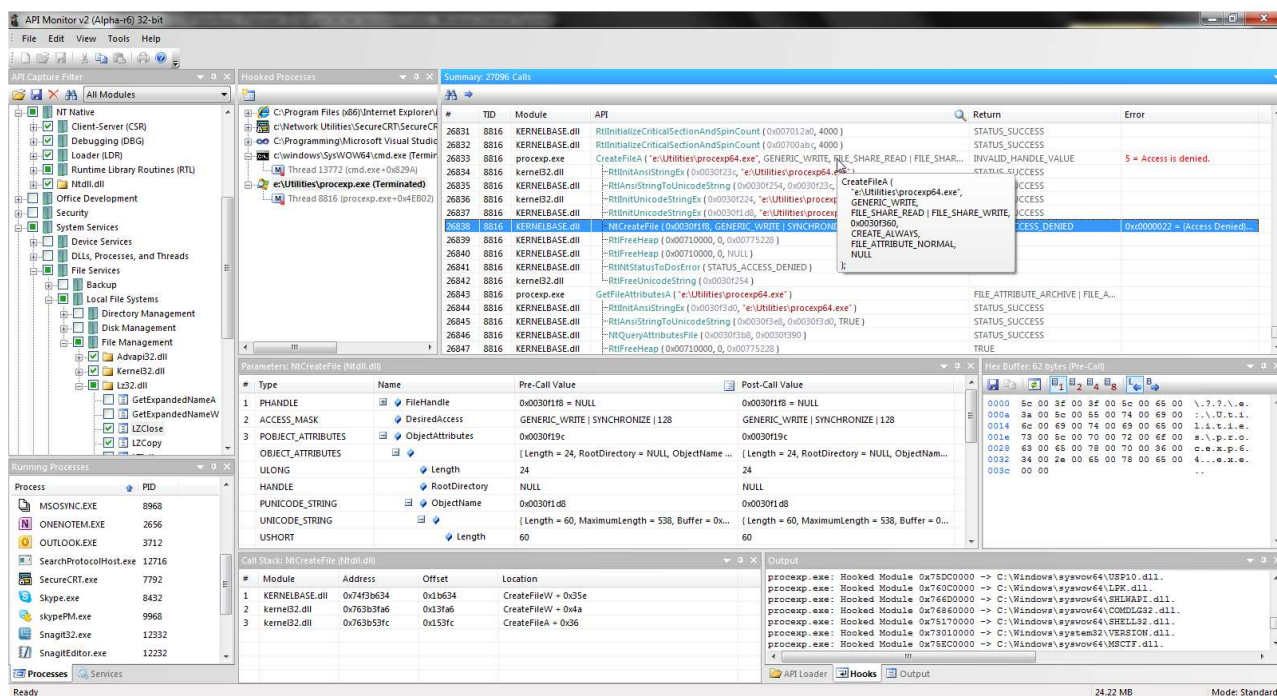


Рисунок 2.60 – Головне вікно ПЗ API MONITOR

Кастомізація – Користувачі можуть налаштовувати, які саме API вони хочуть моніторити, фільтруючи виклики за своїми потребами.

Зручний графічний інтерфейс – Інструмент має зручний графічний інтерфейс, що полегшує перегляд та аналіз викликів API.

Функція пошуку та фільтрації – Можливість шукати та фільтрувати виклики API за різними критеріями для полегшення аналізу.

Логування викликів API – API Monitor дозволяє записувати всі перехоплені виклики API у файл для подальшого аналізу.

Експорт даних – Збереження записаних даних у різних форматах, що дозволяє легко ділитися інформацією та використовувати її в інших інструментах.

Аналіз параметрів – Інструмент дозволяє детально аналізувати параметри викликів API, що допомагає виявляти потенційні проблеми та вразливості.

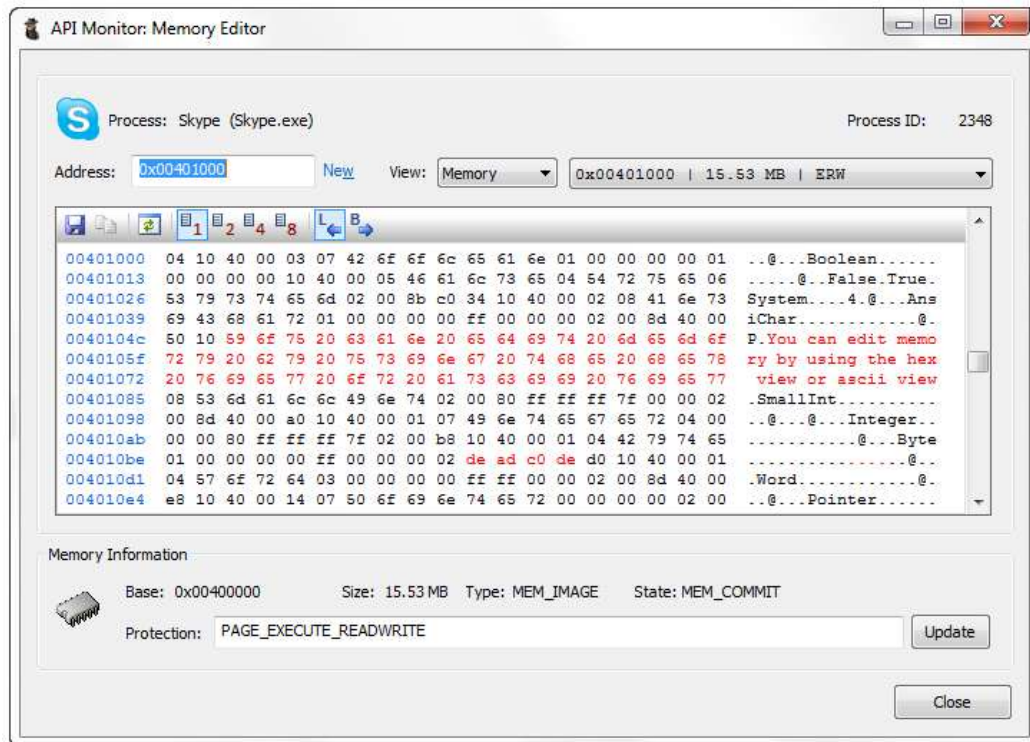


Рисунок 2.61 – ПЗ API MONITOR, редактор пам'яті процесу

Підтримка багатьох мов програмування – Підтримка програм, написаних на різних мовах програмування, що робить інструмент універсальним для різних розробницьких середовищ.

API Monitor є незамінним інструментом для розробників, тестувальників та дослідників безпеки, забезпечуючи детальний аналіз викликів API та допомагаючи виявляти проблеми та вразливості у програмному забезпеченні.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розробки та налагодження програмного забезпечення.

1. Відстеження викликів API. API Monitor дозволяє розробникам детально аналізувати, які API виклики здійснюються їхнім програмним забезпеченням, виявляючи потенційні помилки або непередбачувані виклики. Як приклад розробник використовує API Monitor для моніторингу програми, щоб виявити, які функції Windows API викликаються під час виконання специфічного сценарію, що допомагає оптимізувати код та усунути помилки.

З точки зору дослідження безпеки.

2. Дослідники безпеки використовують API Monitor для виявлення підозрілих викликів API, що можуть свідчити про діяльність шкідливого програмного забезпечення. Як приклад спеціаліст з безпеки моніторить виконання підозрілого файлу за допомогою API Monitor та виявляє виклики API, які намагаються встановити несанкціоновані мережеві з'єднання або змінити критичні системні файли.

З точки зору тестування та відлагодження ПЗ.

3. API Monitor допомагає тестувальникам аналізувати продуктивність програм, відстежуючи виклики API та виявляючи ті, що викликають затримки або надмірне використання ресурсів. Як приклад тестувальник використовує API Monitor для аналізу гри, відстежуючи виклики графічних API, щоб виявити, які функції викликають затримки в рендерингу зображень та оптимізувати продуктивність.

ВИСНОВКИ

API Monitor є потужним інструментом для перехоплення та аналізу викликів API у реальному часі. Основні переваги включають:

1. Перехоплення та детальний аналіз викликів API, включаючи параметри та результати.
2. Можливість моніторингу різних Windows API, що охоплюють різні функціональні області.
3. Інтерактивний графічний інтерфейс з функціями пошуку та фільтрації для зручності аналізу.
4. Можливість логування та експорту викликів API для подальшого аналізу.

2.25. ПЗ WINMERGE

ЛІЦЕНЗІЯ

GNU General Public License v2.0 (GPLv2). WinMerge є відкритим програмним забезпеченням, яке поширюється під ліцензією GPLv2. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

WinMerge розробляється спільнотою з відкритим вихідним кодом.

ОФІЦІЙНИЙ РЕСУРС

<https://winmerge.org/>

ОПИС

WinMerge інструмент для порівняння та об'єднання файлів і директорій. Він надає можливість аналізувати відмінності між файлами, виявляти зміни та об'єднувати їх, забезпечуючи ефективний інструмент для керування версіями та синхронізацією даних. WinMerge є важливим інструментом для розробників, тестувальників і будь-яких користувачів, які потребують аналізу та об'єднання файлів.

Можливості програми

Текстове порівняння – WinMerge дозволяє порівнювати текстові файли, виділяючи відмінності між ними. Програма підтримує різні режими порівняння, включаючи рядкове та символічне порівняння.

Порівняння директорій – Інструмент дозволяє порівнювати вміст директорій, показуючи відмінності між файлами та піддиректоріями.

Об'єднання змін – WinMerge надає можливість об'єднувати зміни між різними версіями файлів, допомагаючи користувачам зливати зміни з різних джерел у єдиний файл.

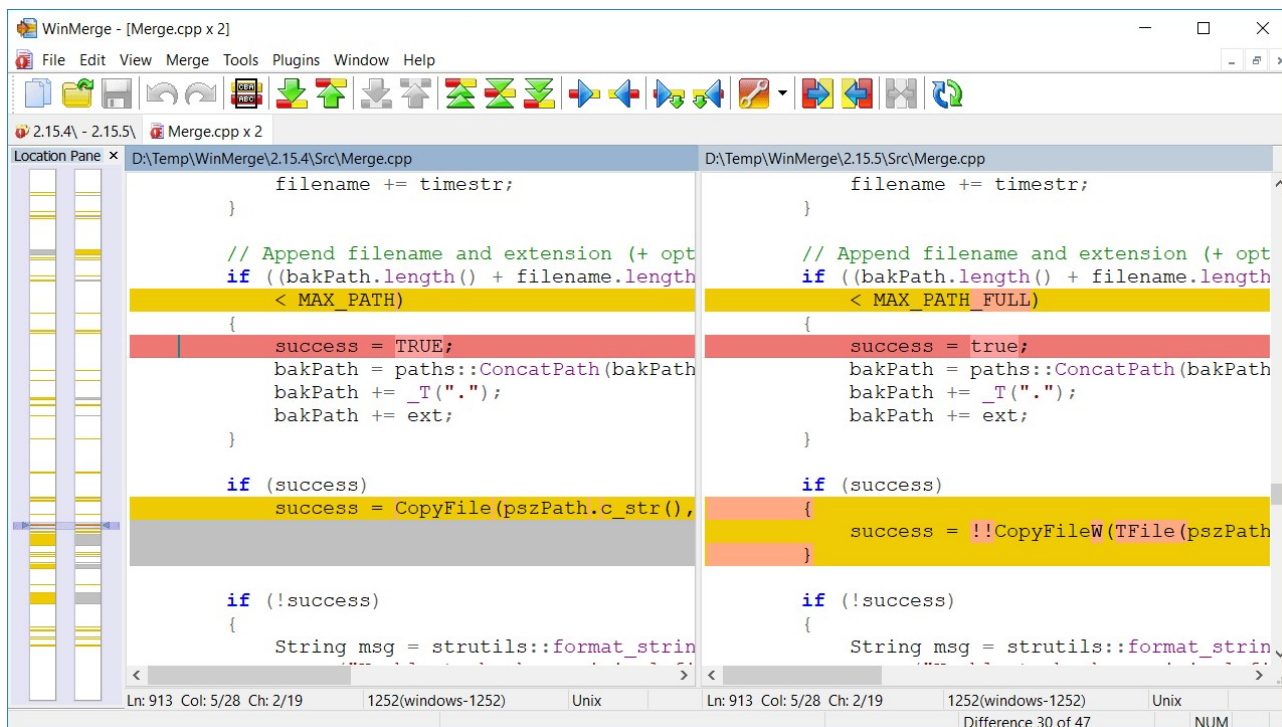


Рисунок 2.62 – Головне вікно ПЗ WINMERGE

Конфлікти об'єднання – Програма виявляє конфлікти під час об'єднання файлів та надає зручні інструменти для їх вирішення.

Різні типи файлів – WinMerge підтримує порівняння та об'єднання різних типів файлів, включаючи текстові файли, код програм, файли конфігурацій та інші.

Плагіни та розширення – Підтримка плагінів дозволяє розширювати функціональність програми для роботи з новими форматами файлів.

Графічний інтерфейс – WinMerge надає зручний графічний інтерфейс, що полегшує роботу з порівняннями та об'єднаннями файлів.

Колірне виділення відмінностей – Відмінності між файлами виділяються різними кольорами, що допомагає швидко ідентифікувати зміни.

Функція пошуку – Можливість пошуку та фільтрації відмінностей між файлами.

Експорт результатів – Збереження результатів порівняння у різних форматах для подальшого аналізу або використання.

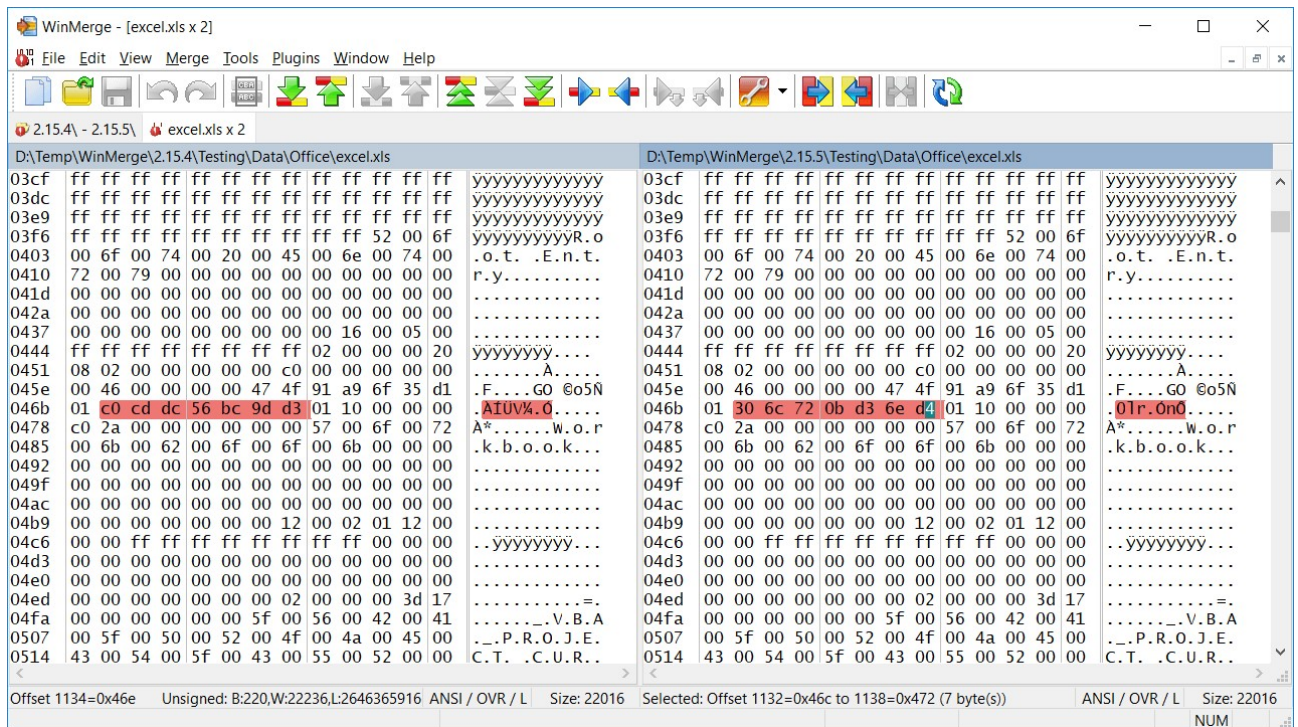


Рисунок 2.63 – ПЗ WINMERGE, приклад бінарного порівняння

WinMerge є потужним і зручним інструментом для порівняння та об'єднання файлів і директорій, що робить його незамінним для розробників, тестувальників і всіх, хто працює з великими обсягами даних.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору розробки програмного забезпечення.

1. Керування версіями. WinMerge дозволяє розробникам порівнювати різні версії вихідного коду, виявляти зміни та об'єднувати їх, забезпечуючи ефективне керування версіями. Як приклад команда розробників використовує WinMerge для порівняння змін у вихідному коді між двома гілками проєкту, швидко виявляючи нововведення та конфлікти, а також зливаючи зміни в основну гілку.

З точки зору тестування програмного забезпечення.

2. Порівняння конфігураційних файлів. Тестувальники можуть використовувати WinMerge для порівняння конфігураційних файлів різних середовищ, виявляючи невідповідності, які можуть призводити до помилок. Як

приклад тестувальник використовує WinMerge для порівняння конфігураційних файлів серверів у тестовому та виробничому середовищах, виявляючи розбіжності, які можуть спричинити проблеми під час розгортання додатка.

З точки зору синхронізації даних.

3. WinMerge дозволяє користувачам порівнювати вміст директорій, синхронізуючи файли та об'єднуючи зміни з різних джерел. Як приклад ІТ-адміністратор використовує WinMerge для синхронізації резервних копій файлів на різних серверах, забезпечуючи актуальність та цілісність даних.

З точки зору форензичного аналізу.

4. Форензичні аналітики можуть використовувати WinMerge для порівняння журналів подій та інших файлів, щоб виявити зміни або підозрілу активність. Як приклад форензичний аналітик використовує WinMerge для порівняння журналів подій до та після підозрілої активності, виявляючи конкретні зміни та дії зловмисника.

ВИСНОВКИ

WinMerge є потужним інструментом для порівняння та об'єднання файлів і директорій, що робить його корисним у багатьох сферах, включаючи розробку програмного забезпечення, тестування, синхронізацію даних та форензичний аналіз. Основні переваги включають:

1. Виділення відмінностей між файлами та директоріями, полегшуючи аналіз та об'єднання змін.
2. Зручний графічний інтерфейс з кольоровим виділенням відмінностей.
3. Підтримка різних типів файлів та можливість розширення функціональності за допомогою плагінів.
4. Пошук, фільтрація та експорт результатів порівняння для подальшого аналізу.

2.26. ПЗ UPX (THE ULTIMATE PACKER FOR EXECUTABLES)

ЛІЦЕНЗІЯ

GNU General Public License v2.0 (GPLv2). UPX є відкритим програмним забезпеченням, яке поширюється під ліцензією GPLv2. Це дозволяє вільно використовувати, змінювати та розповсюджувати програму, за умови, що похідні роботи також будуть випущені під тією ж ліцензією.

РОЗРОБНИК

UPX розроблений компаніями Markus Oberhumer, Laszlo Molnar, і John Reiser.

ОФІЦІЙНИЙ РЕСУРС

<https://upx.github.io/>

ОПИС

UPX (The Ultimate Packer for Executables) пакувальник виконуваних файлів, який використовується для стискання програм і зменшення їхнього розміру без втрати функціональності. Інструмент підтримує широкий спектр файлових форматів і операційних систем, забезпечує швидке стискання та дає змогу виконувати розпакування без втрати даних.

UPX є корисним засобом для розробників, системних адміністраторів та інших ІТ фахівців, яким потрібно зменшувати розмір виконуваних файлів зі збереженням їхньої працездатності, функціональності та цілісності.

Можливості програми

Підтримка різних форматів – UPX підтримує стискання файлів у форматах EXE, ELF, Mach-O, PE, та інших, що робить його універсальним інструментом для різних операційних систем, включаючи Windows, Linux, і macOS.

```
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2024
UPX 4.2.4      Markus Oberhumer, Laszlo Molnar & John Reiser      May 9th 2024

Usage: upx [-123456789dlthVL] [-qvfk] [-o file] file..

Commands:
  -1      compress faster          -9      compress better
  -d      decompress              -l      list compressed file
  -t      test compressed file    -V      display version number
  -h      give more help          -L      display software license

Options:
  -q      be quiet                -v      be verbose
  -oFILE  write output to 'FILE'
  -f      force compression of suspicious files
  -k      keep backup files
file..   executables to (de)compress

Type 'upx --help' for more detailed help.

UPX comes with ABSOLUTELY NO WARRANTY; for details visit https://upx.github.io
```

Рисунок 2.64 – Консольне ПЗ UPX

Високий рівень стискання – Інструмент забезпечує високий рівень стискання, зменшуючи розмір файлів до 50-70% від їх початкового розміру.

Швидке стискання і розпакування – UPX забезпечує швидке стискання та розпакування файлів, що дозволяє ефективно працювати навіть з великими обсягами даних.

Оптимізація виконання – Стиснуті файли працюють майже з тією ж швидкістю, що й оригінальні, завдяки оптимізації процесу розпакування.

Повне відновлення файлів – UPX дозволяє повністю розпаковувати файли до їх початкового стану, що забезпечує збереження оригінальних даних без втрат.

Безпека даних – Процес розпакування гарантує цілісність даних, що є важливим для програм, які потребують точного відновлення.

Командний рядок – UPX підтримує використання з командного рядка, що дозволяє інтегрувати його у скрипти та автоматизовані процеси.

Налаштування параметрів стискання – Користувачі можуть налаштовувати параметри стискання для досягнення оптимального балансу між розміром файлу та швидкістю виконання.

Доступність вихідного коду – UPX є відкритим програмним забезпеченням, що дозволяє розробникам переглядати та модифікувати вихідний код відповідно до своїх потреб.

Спільнота і підтримка – Активна спільнота розробників та користувачів забезпечує регулярне оновлення та підтримку інструменту.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору стискання виконуваних файлів для розповсюдження.

1. Зменшення розміру файлів. UPX дозволяє розробникам зменшувати розмір виконуваних файлів, що сприяє економії місця на диску та зменшенню часу завантаження. Як приклад розробник використовує UPX для стискання файлів програмного забезпечення перед його розповсюдженням, що зменшує розмір інсталяційного пакету та прискорює процес завантаження і встановлення програми.

З точки зору оптимізації портативних додатків.

2. Стискання портативних програм. UPX часто використовується для стискання портативних додатків, які запускаються з USB-накопичувачів або інших змінних носіїв. Як приклад адміністратор системи стискає портативні утиліти за допомогою UPX, щоб зменшити їх розмір і забезпечити зручніше використання та швидший запуск з USB-накопичувачів.

З точки зору зменшення розміру шкідливих програм.

3. Зловмисники можуть використовувати UPX для стискання шкідливих програм, зменшення їхнього розміру та ускладнення аналізу коду. Наприклад, дослідник безпеки під час аналізу підозрілого виконаного файлу може встановити, що він стиснутий за допомогою UPX, після чого розпаковує його для подальшого дослідження структури та поведінки шкідливого коду.

З точки зору оптимізації ресурсів вбудованих систем.

4. Економія пам'яті. Вбудовані системи з обмеженими ресурсами можуть виграти від використання UPX для зменшення розміру виконуваних файлів і економії пам'яті. Як приклад інженер використовує UPX для стиснення програмного забезпечення, що працює на мікроконтролері, забезпечуючи ефективніше використання обмежених ресурсів пам'яті пристрою.

ВИСНОВКИ

UPX є ефективним інструментом для стиснення виконуваних файлів, який дає змогу істотно зменшити їхній розмір без втрати функціональності. До основних переваг цього інструменту належать:

1. Зменшення розміру файлів до 50-70% від їх початкового розміру.
2. Швидке стиснення і розпакування файлів без втрати продуктивності.
3. Можливість повного відновлення файлів до їх початкового стану.
4. Підтримка різних форматів файлів і операційних систем, а також можливість налаштування параметрів стиснення.
5. Доступність вихідного коду та активна підтримка спільноти.

2.27. ПЗ REGSHOT-ADVANCED

ЛІЦЕНЗІЯ

GNU General Public License v2.0 (GPLv2). Regshot-Advanced – це програмне забезпечення з відкритим кодом, яке розповсюджується за ліцензією (GPLv2). Воно дозволяє вільно користуватися, модифікувати та поширювати програму, за умови, що всі похідні роботи також випускаються під тією ж ліцензією

РОЗРОБНИК

Regshot-Advanced розроблений спільнотою з відкритим вихідним кодом.

ОФІЦІЙНИЙ РЕСУРС

<https://github.com/skydive241/Regshot-Advanced>

ОПИС

Regshot Advanced є інструментом для порівняння стану реєстру та файлової системи Windows до і після внесення певних змін. Він дає змогу фахівцям з комп'ютерної безпеки та системним адміністраторам аналізувати зміни, які створюються програмами, службами або системними подіями. Програма є вдосконаленою версією оригінального Regshot і має додаткові функції та покращену продуктивність.

Regshot Advanced є корисним засобом для системних адміністраторів, дослідників безпеки та ІТ фахівців, оскільки забезпечує ефективне виявлення, порівняння й аналіз змін у реєстрі та файловій системі.

Можливості програми

Зняття знімків – Regshot-Advanced дозволяє створювати знімки поточного стану реєстру та файлової системи до і після певних дій або змін.

Порівняння знімків – Інструмент порівнює два знімки, виявляючи всі зміни, що відбулися між ними.

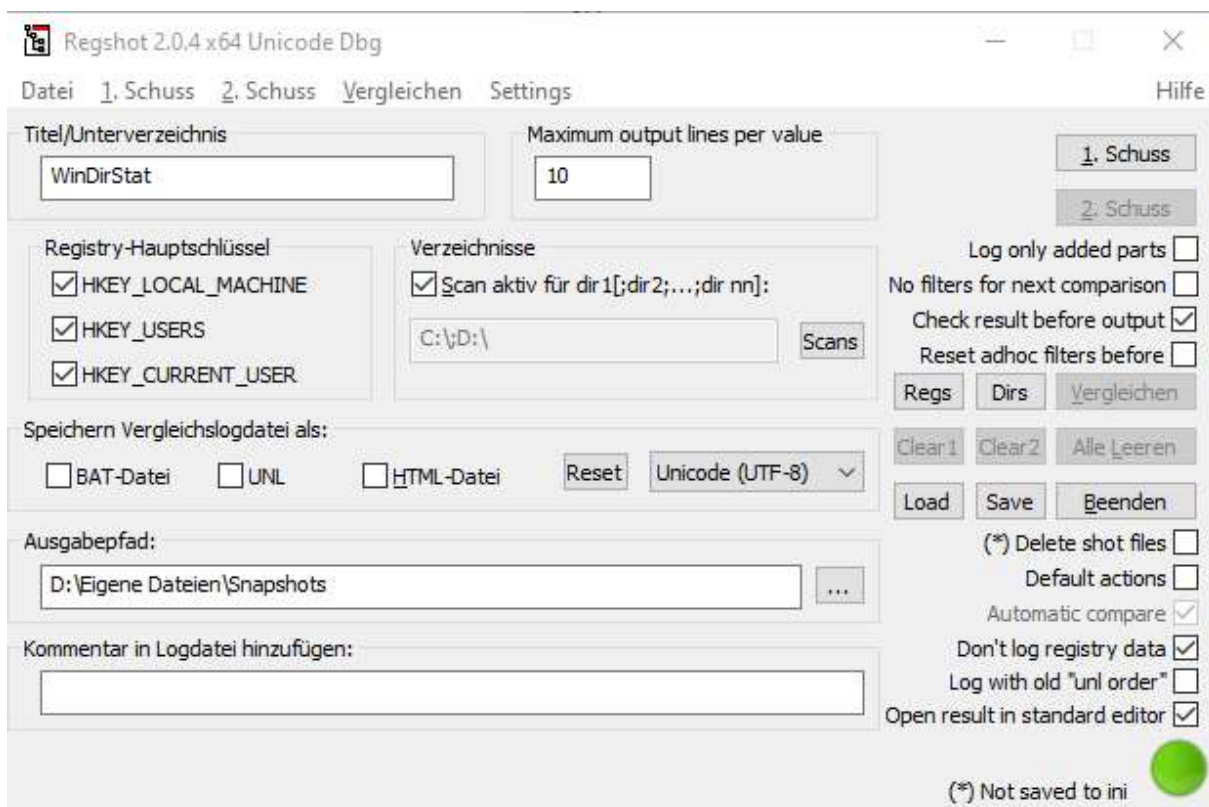


Рисунок 2.65 – Головне вікно ПЗ REGSHOT

Виявлення змін у реєстрі – Regshot-Advanced виявляє зміни в ключах реєстру, таких як створення, видалення та модифікація ключів і значень.

Аналіз файлової системи – Інструмент також виявляє зміни у файловій системі, включаючи створення, видалення та зміну файлів і папок.

Текстові звіти – Regshot-Advanced генерує докладні текстові звіти про всі зміни, які відбулися між знімками. Звіти містять інформацію про змінені ключі реєстру та файли.

Збереження звітів – Можливість збереження звітів у різних форматах для подальшого аналізу або документування.

Гнучкість у використанні – Інструмент підтримує командний рядок, що дозволяє автоматизувати процес зняття знімків та порівняння змін.

Сумісність – Regshot-Advanced підтримує різні версії Windows, включаючи старі та сучасні операційні системи.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору моніторингу змін у реєстрі після встановлення програмного забезпечення.

1. Відстеження змін у реєстрі. Regshot-Advanced дозволяє системним адміністраторам створювати знімки реєстру до і після встановлення програмного забезпечення, щоб визначити всі зміни, внесені інсталяцією. Як приклад адміністратор використовує Regshot-Advanced для порівняння стану реєстру до і після встановлення нового антивірусного програмного забезпечення, щоб переконатися, що інсталяція не внесла непередбачених змін.

З точки зору аналізу змін після підозрілої активності.

2. Виявлення шкідливих змін. Дослідники безпеки можуть використовувати Regshot Advanced для виявлення змін у реєстрі та файловій системі, які виникають унаслідок дії шкідливого програмного забезпечення або інших загроз. Наприклад, аналітик безпеки застосовує Regshot Advanced для аналізу змін, внесених до системи після підозрілої активності, щоб виявити нові ключі реєстру та файли, створені шкідливим ПЗ.

З точки зору тестування та налагодження програмного забезпечення.

3. Відстеження впливу програмного забезпечення. Розробники можуть використовувати Regshot Advanced для контролю змін у системі, які виникають під час встановлення або запуску програмного забезпечення. Це дає змогу перевірити коректність роботи додатка, виявити небажані зміни та оцінити можливі побічні ефекти. Наприклад, розробник порівнює стан системи до і після тестового запуску нового додатка, щоб визначити, які саме зміни були внесені до реєстру та файлової системи.

З точки зору форензичного аналізу.

4. Аналіз слідів діяльності. Форензичні аналітики можуть використовувати Regshot-Advanced для створення знімків реєстру та файлової системи до і після виконання підозрілих дій для виявлення слідів діяльності. Як приклад форензичний аналітик використовує Regshot-Advanced для порівняння

змін у системі після виконання підозрілого файлу, щоб виявити нові або змінені файли та ключі реєстру.

ВИСНОВКИ

Regshot-Advanced є потужним інструментом для відстеження та аналізу змін у реєстрі та файловій системі. Основні переваги включають:

1. Створення знімків до і після змін для детального аналізу.
2. Виявлення змін у ключах реєстру та файловій системі.
3. Генерація докладних текстових звітів для документування змін.
4. Підтримка командного рядка та відкритий вихідний код.
5. Підтримка старих та сучасних версій операційних систем.

2.28. ПЗ SYSTEM INFORMER

ЛІЦЕНЗІЯ

MIT License. System Informer є відкритим програмним забезпеченням, яке дозволяється вільно використовувати, копіювати, змінювати, поширювати та застосовувати у власних програмних рішеннях за умови збереження повідомлення про авторські права та тексту ліцензії.

РОЗРОБНИК

System Informer розробляється Winsider Seminars & Solutions, Inc. за участю спільноти розробників з відкритим кодом. Проєкт створений у 2008 році Wen Jia Liu як відкрита альтернатива стандартному диспетчеру завдань Windows та інструментам аналізу процесів.

ОФІЦІЙНИЙ РЕСУРС

<https://systeminformer.com/>

<https://github.com/winsiderss/systeminformer/>

ОПИС

System Informer це багатофункціональний інструмент для моніторингу системних ресурсів, аналізу процесів, налагодження програмного забезпечення та виявлення потенційно шкідливої активності у Windows. Програма є сучасним продовженням розвитку Process Hacker та надає розширені можливості для перегляду процесів, служб, драйверів, мережевих з'єднань, дескрипторів, модулів, потоків і параметрів використання пам'яті.

Можливості програми

Детальна інформація про процеси. System Informer відображає активні процеси, їх ідентифікатори, використання центрального процесора, оперативної пам'яті, диска, графічного процесора та інших ресурсів.

Ієрархія процесів. Програма показує процеси у вигляді дерева, що дає змогу визначати батьківські та дочірні зв'язки між ними.

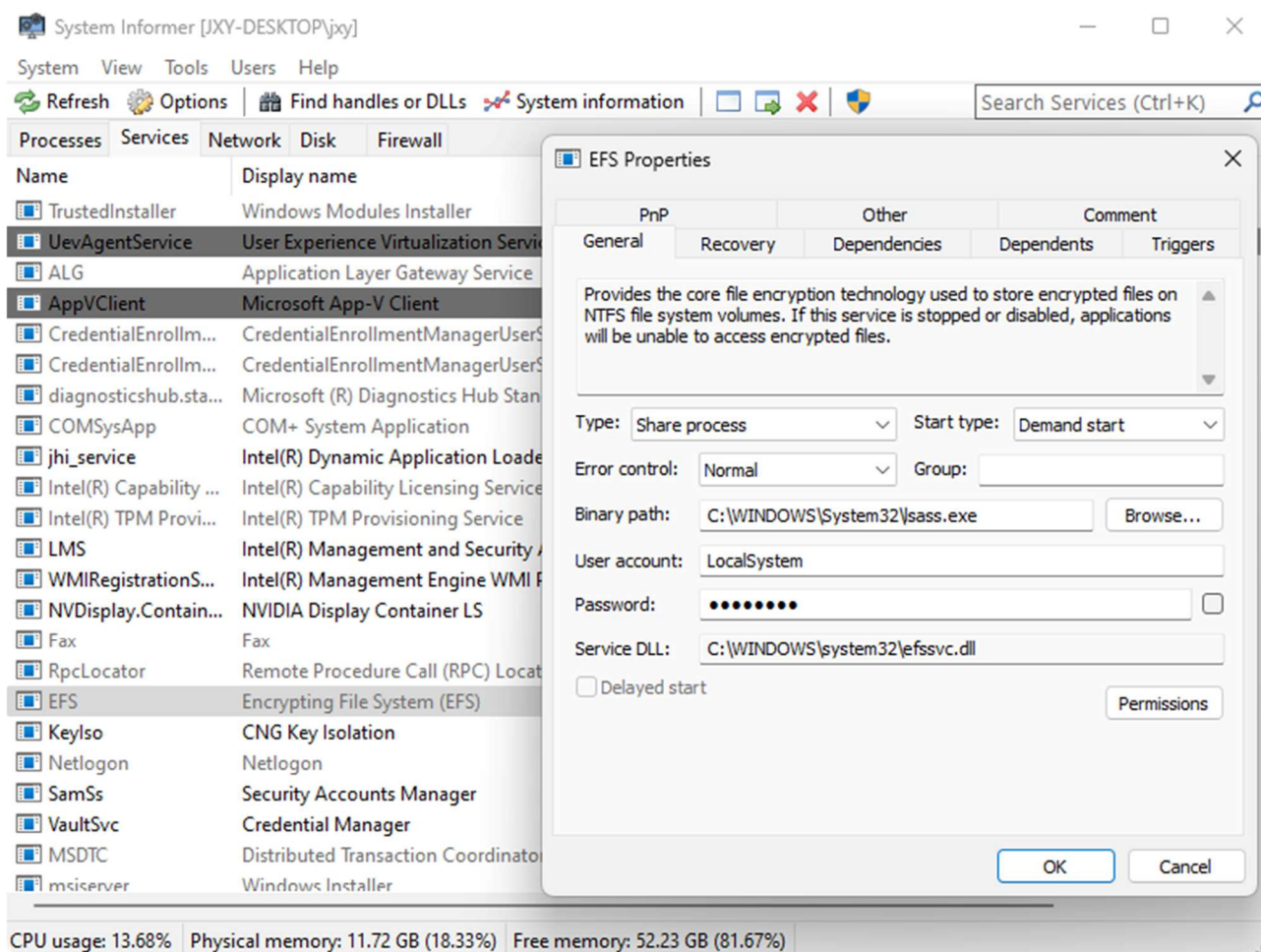


Рисунок 2.66 – Головне вікно ПЗ System Informer

Керування процесами. System Informer дозволяє завершувати, призупиняти, відновлювати та перезапускати процеси, а також змінювати їх пріоритети.

Аналіз служб Windows. Програма дає змогу переглядати служби, запускати, зупиняти та аналізувати їх стан.

Перегляд драйверів. System Informer відображає завантажені драйвери, їх розташування, властивості та стан у системі.

Аналіз мережевих з'єднань. Програма показує активні TCP та UDP з'єднання, локальні й віддалені адреси, порти та процеси, які використовують мережу.

Перегляд дескрипторів і модулів. System Informer дозволяє аналізувати відкриті файли, ключі реєстру, бібліотеки DLL та інші об'єкти, пов'язані з конкретним процесом.

Аналіз пам'яті. Програма надає інформацію про використання фізичної та віртуальної пам'яті, робочі набори, стек, heap та інші характеристики пам'яті процесів.

ПРИКЛАДИ ВИКОРИСТАННЯ

З точки зору аналізу продуктивності системи.

1. Моніторинг ресурсів. System Informer дозволяє визначити, які процеси створюють найбільше навантаження на центральний процесор, оперативну пам'ять, диск або мережу. Як приклад системний адміністратор використовує System Informer для пошуку процесу, який спричиняє уповільнення роботи робочої станції або сервера, після чого змінює його пріоритет або завершує його виконання.

З точки зору управління процесами та службами.

2. Керування службами Windows. System Informer дає змогу переглядати стан служб, запускати або зупиняти їх, а також аналізувати пов'язані з ними процеси. Як приклад адміністратор зупиняє службу, яка працює некоректно, перевіряє її залежності та повторно запускає її після усунення проблеми.

З точки зору забезпечення безпеки.

3. Виявлення підозрілих процесів. System Informer дає змогу перевіряти активні процеси, їх мережеві з'єднання, відкриті файли та завантажені модулі. Як приклад фахівець з кібербезпеки використовує System Informer для виявлення процесу, який встановлює несанкціоноване мережеве з'єднання, аналізує його властивості та завершує його виконання.

ВИСНОВКИ

System Informer є сучасним інструментом для моніторингу, адміністрування та аналізу процесів у Windows.

1. Відображає розширену інформацію про процеси, служби, драйвери, модулі, дескриптори та мережеві з'єднання.
2. Дозволяє керувати процесами та службами Windows, змінювати пріоритети, завершувати завислі процеси й аналізувати використання ресурсів.
3. Надає інструменти для налагодження програмного забезпечення, аналізу пам'яті та створення дамів процесів.
4. Допомогає виявляти підозрілу активність у системі через перевірку процесів, мережевих підключень і завантажених компонентів.

РОЗДІЛ 3. ПРАКТИЧНІ ЗАВДАННЯ РЕАГУВАННЯ НА ІНЦИДЕНТИ, АНАЛІЗУ ШКІДЛИВИХ ПРОГРАМ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ

ПРАКТИЧНЕ ЗАВДАННЯ №1

ТЕМА: Моніторинг файлової підсистеми інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем моніторингу.

ЗНАТИ: Теоретичні основи архітектурної будови ОС Windows.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

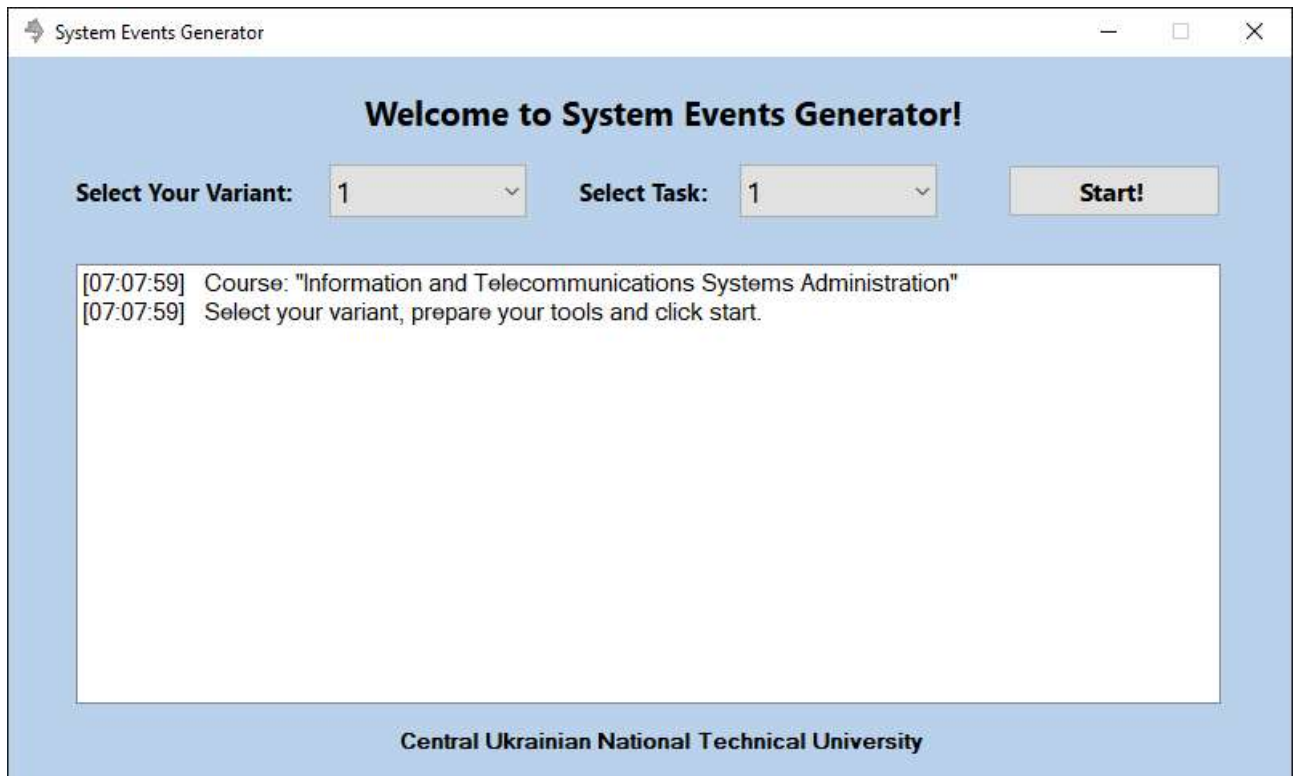


Рисунок 3.1 – Головне вікно генератора подій

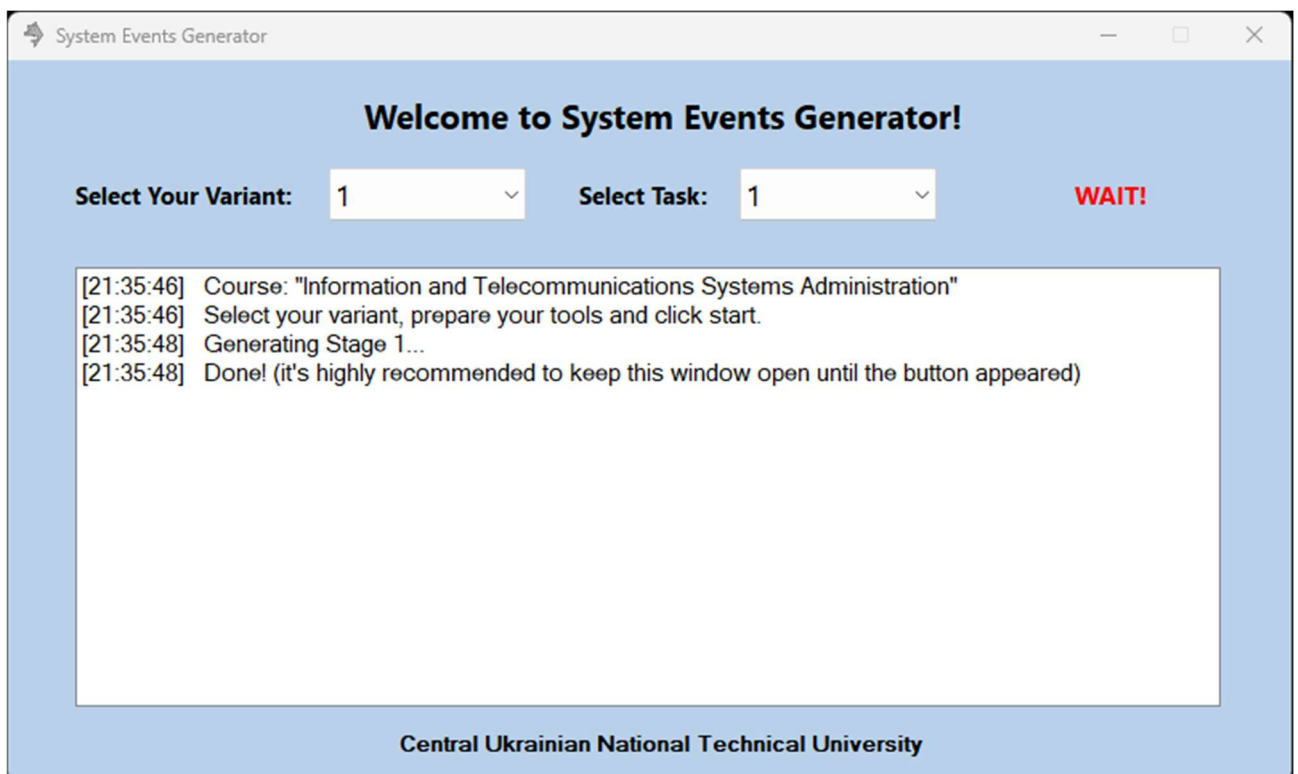


Рисунок 3.2 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 1)

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**ITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.

Посилання: <https://drive.google.com/file/d/1WQbei1Ji2l6LBMIQDohrJghL60TYpUQa9/view?usp=sharing>

3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадючих списках як це показано на рисунку 2.

4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.

5. Сформувати звіт роботи генератора “**ITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке моніторинг файлової підсистеми Windows, і чому він важливий у контексті DFIR?
2. Які основні компоненти файлової підсистеми Windows підлягають моніторингу під час розслідування інцидентів?
3. Як файловий моніторинг може допомогти у виявленні шкідливої активності або несанкціонованих змін у системі?

4. Які інструменти та методи найчастіше використовуються для моніторингу файлової підсистеми Windows у реальному часі?
5. Як моніторинг змін у файловій системі може допомогти в аналізі компрометації системи або витоку даних?
6. Як можна автоматизувати процес моніторингу файлової системи для виявлення аномалій?
7. Що таке Sysinternals Process Monitor, і як він використовується для моніторингу файлової підсистеми?
8. Як використовувати Process Monitor для відстеження змін у файловій системі та реєстрації подій, пов'язаних із створенням, видаленням або модифікацією файлів?
9. Як інструмент Windows Audit Policy може бути налаштований для відстеження доступу до критичних файлів та папок?
10. Як можна налаштувати Windows File Integrity Monitoring (FIM) для автоматичного відстеження змін у важливих системних файлах?
11. Як використовувати Windows Event Viewer для моніторингу подій, пов'язаних із файловою системою, таких як доступ до файлів або зміна їх прав доступу?
12. Які інші сторонні інструменти використовуються для моніторингу файлової підсистеми Windows, і які їхні основні можливості?
13. Як можна виявити зміни у системних файлах, що можуть вказувати на компрометацію системи або шкідливу активність?
14. Які індикатори компрометації (IoC) можуть бути виявлені під час моніторингу файлової підсистеми?
15. Як аналізувати журнали моніторингу файлової системи для виявлення підозрілої активності або несанкціонованих дій?
16. Як можна використовувати File Access Monitoring для виявлення несанкціонованого доступу до критичних файлів або папок?
17. Як аналізувати зміни у файлах конфігурації та реєстру Windows для виявлення шкідливих дій?

18. Як виявити ознаки приховування файлів або підміни легітимних файлів шкідливими?

19. Як аналізувати журнали зміни прав доступу до файлів для виявлення можливих спроб обходу системи безпеки?

20. Як моніторинг файлової підсистеми може допомогти у виявленні шкідливого ПЗ, яке маскується під легітимні системні файли?

21. Як використовувати результати моніторингу файлової підсистеми для швидкого реагування на інциденти безпеки?

22. Як моніторинг файлової системи може допомогти в ідентифікації початкової точки компрометації (Patient Zero) у разі атаки?

23. Як результати моніторингу можуть бути використані для відновлення системи після атаки або несанкціонованих змін?

24. Як моніторинг файлової підсистеми може допомогти у виявленні та усуненні слідів шкідливої активності під час процесу ремедіації?

25. Як можна використовувати моніторинг файлової системи для перевірки цілісності системи після відновлення або оновлення?

26. Як інтегрувати моніторинг файлової підсистеми Windows з SIEM-системами для автоматичного виявлення та реагування на інциденти?

27. Як можна використовувати сценарії PowerShell або інші інструменти автоматизації для розширення можливостей моніторингу файлової підсистеми?

28. Як аналізувати зміни у файловій системі у поєднанні з іншими типами даних, такими як мережевий трафік або журнали безпеки?

29. Як можна налаштувати розширені правила моніторингу для відстеження конкретних подій або типів активності у файловій підсистемі?

30. Як використовувати методи машинного навчання або штучного інтелекту для автоматичного виявлення аномалій у даних моніторингу файлової підсистеми?

31. Як розробити стратегію моніторингу файлової підсистеми для корпоративного середовища?

32. Як налаштувати оповіщення та автоматизовані дії на основі результатів моніторингу файлової підсистеми?

33. Як забезпечити баланс між продуктивністю системи та ефективністю моніторингу файлової підсистеми?

34. Як забезпечити захист даних моніторингу від несанкціонованого доступу або змін?

35. Як здійснювати аудит та регулярну перевірку налаштувань моніторингу файлової підсистеми для забезпечення його відповідності сучасним вимогам безпеки?

ПРАКТИЧНЕ ЗАВДАННЯ №2

ТЕМА: Моніторинг підсистеми реєстру інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем моніторингу.

ЗНАТИ: Теоретичні основи архітектурної будови ОС Windows.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

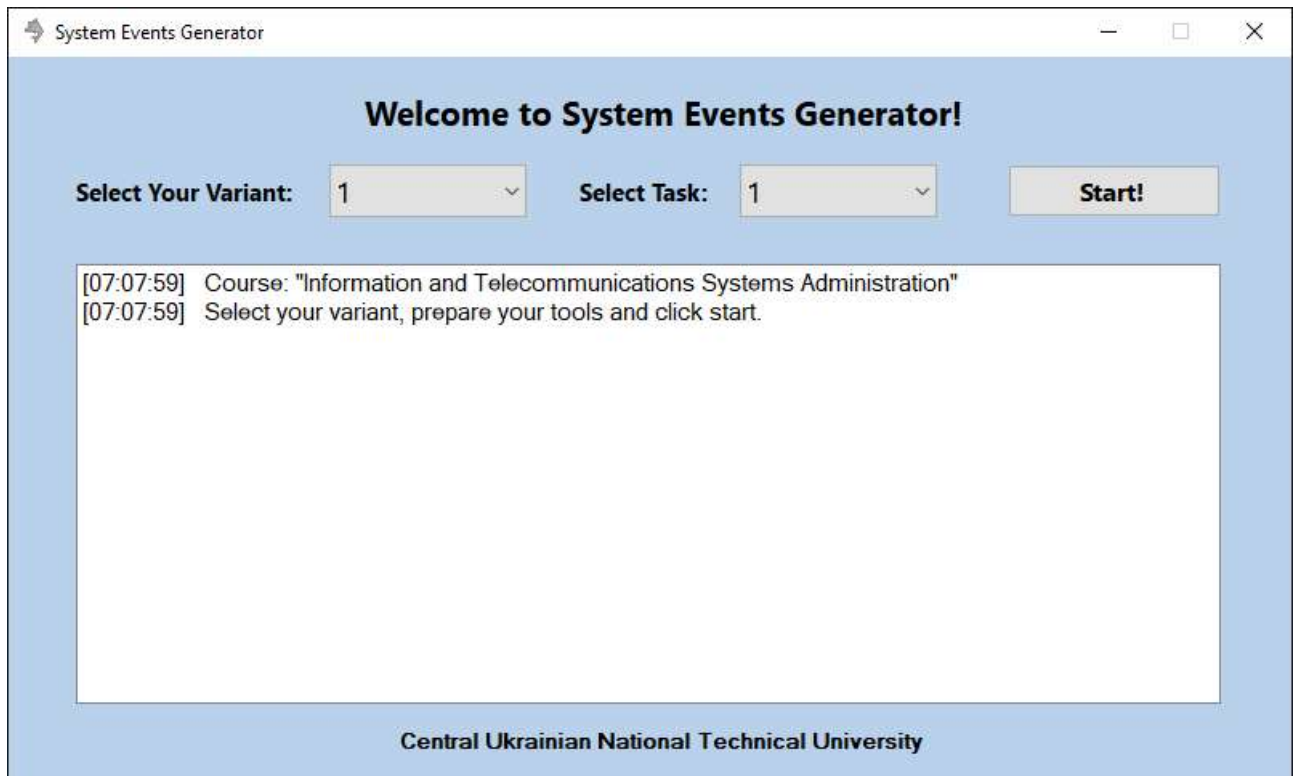


Рисунок 3.3 – Головне вікно генератора подій

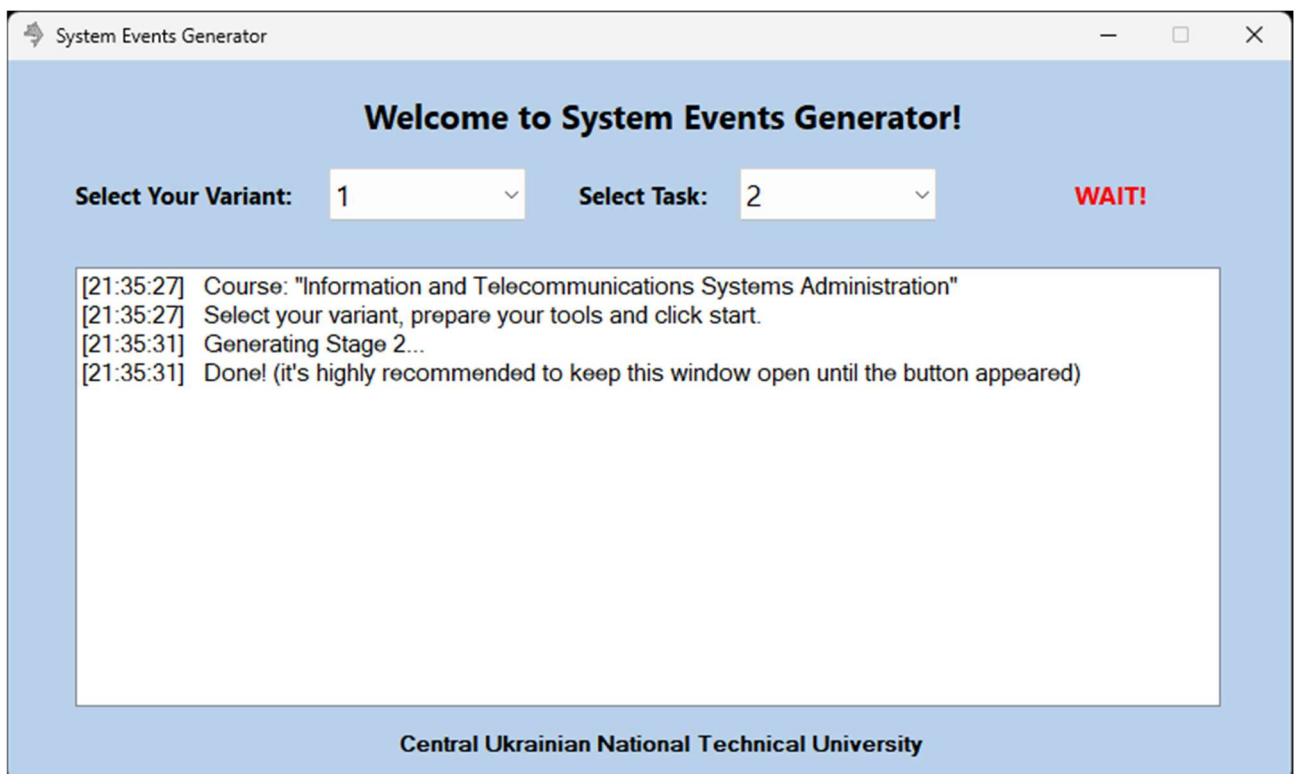


Рисунок 3.4 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 2)

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**AITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадючих списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.
5. Сформулювати звіт роботи генератора “**AITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовуються ini файли?
2. Де фізично зберігається файли системного реєстру Windows?
3. Які існують розділи реєстру та для чого вони використовуються?
4. За необхідності як відновити ключі реєстру?
5. Наведіть приклади створення скрипта редагування даних реєстру.

6. Що таке реєстр Windows? Яку функцію він виконує в операційній системі?
7. Яка структура реєстру Windows? Які основні компоненти (розділи) він містить?
8. Що таке розділи (hives) у реєстрі Windows, і які основні розділи існують?
9. Яку роль відіграє кожен з основних розділів реєстру: HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_CONFIG?
10. Як реєстр Windows зберігає інформацію про встановлене програмне забезпечення?
11. Що таке ключі та підключі в реєстрі Windows? Як вони організовані?
12. Що таке значення ключів у реєстрі Windows? Які типи значень підтримує реєстр?
13. Як змінення значень реєстру може вплинути на роботу операційної системи та програм?
14. Як працює механізм доступу до реєстру? Які існують права доступу до ключів реєстру?
15. Що таке віддзеркалення реєстру (registry reflection)? Як воно застосовується в системах 64-бітної архітектури?
16. Як операційна система Windows використовує реєстр для збереження конфігурації апаратного забезпечення?
17. Що таке точки відновлення системи, і як вони використовуються для збереження стану реєстру?
18. Як користувач може відновити попередній стан реєстру у разі пошкодження або невдалої конфігурації?
19. Що таке автозавантаження (autorun) програм через реєстр? Які шляхи реєстру використовуються для цього?

20. Як адміністратор може використовувати реєстр для налаштування політик безпеки на комп'ютері?
21. Що таке реєстр подій (Event Log) і як він взаємодіє з реєстром Windows?
22. Як працює моніторинг змін у реєстрі Windows? Які інструменти можуть бути використані для цього?
23. Як працює шифрування даних у реєстрі Windows? Які інструменти забезпечують захист даних реєстру?
24. Яка роль реєстру у процесі завантаження Windows? Як змінення в реєстрі можуть вплинути на процес завантаження?
25. Що таке редактор реєстру (Registry Editor)? Які основні функції він виконує?
26. Як працюють сценарії (scripts) для автоматизації змін у реєстрі? Які мови програмування підтримуються для цього?
27. Як реєстр зберігає конфігурації користувачів і груп? Як адміністратор може змінити ці конфігурації?
28. Що таке реєстрові файли (registry files)? Як вони використовуються для імпорту та експорту ключів і значень реєстру?
29. Як реєстр Windows зберігає інформацію про драйвери та апаратні пристрої?
30. Як працює резервне копіювання реєстру Windows? Які інструменти та методи існують для цього?
31. Як працює службове програмне забезпечення Windows, таке як службовий контролер домену (Domain Controller), з реєстром?
32. Як використовуються змінні середовища (environment variables) в реєстрі Windows?
33. Яка роль реєстру в налаштуванні мережевих параметрів операційної системи Windows?
34. Що таке "сироти" (orphaned keys) у реєстрі Windows, і як вони впливають на роботу системи?

35. Як використовувати групові політики (Group Policy) для централізованого управління реєстром на комп'ютерах у домені?
36. Що таке реєстровий сервер (Registry Server), і як він використовується у мережах?
37. Як працює синхронізація реєстру між кількома пристроями або користувачами?
38. Як захистити реєстр від шкідливих змін або несанкціонованого доступу?
39. Що таке реєстрові скрипти (REG files), і як вони використовуються для швидкого внесення змін у реєстр?
40. Як реєстр Windows зберігає інформацію про сертифікати безпеки та шифрувальні ключі?
41. Як реєстр підтримує роботу віртуалізації та контейнеризації у Windows?
42. Що таке реєстрові зміни, пов'язані з оновленням операційної системи, і як вони можуть вплинути на стабільність системи?
43. Як реєстр використовується для налаштування параметрів продуктивності Windows?
44. Що таке "реєстрові вулиці" (registry hives roaming), і як вони використовуються у мобільних робочих середовищах?
45. Як працює архітектура реєстру у різних версіях операційної системи Windows, і які зміни відбулися з часом?

ПРАКТИЧНЕ ЗАВДАННЯ №3

ТЕМА: Моніторинг протоколів прикладного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем мережного моніторингу протоколів прикладного рівня.

ЗНАТИ: Теоретичні основи базової еталонної моделі взаємодії відкритих систем – 7 рівень семирівневої моделі OSI.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

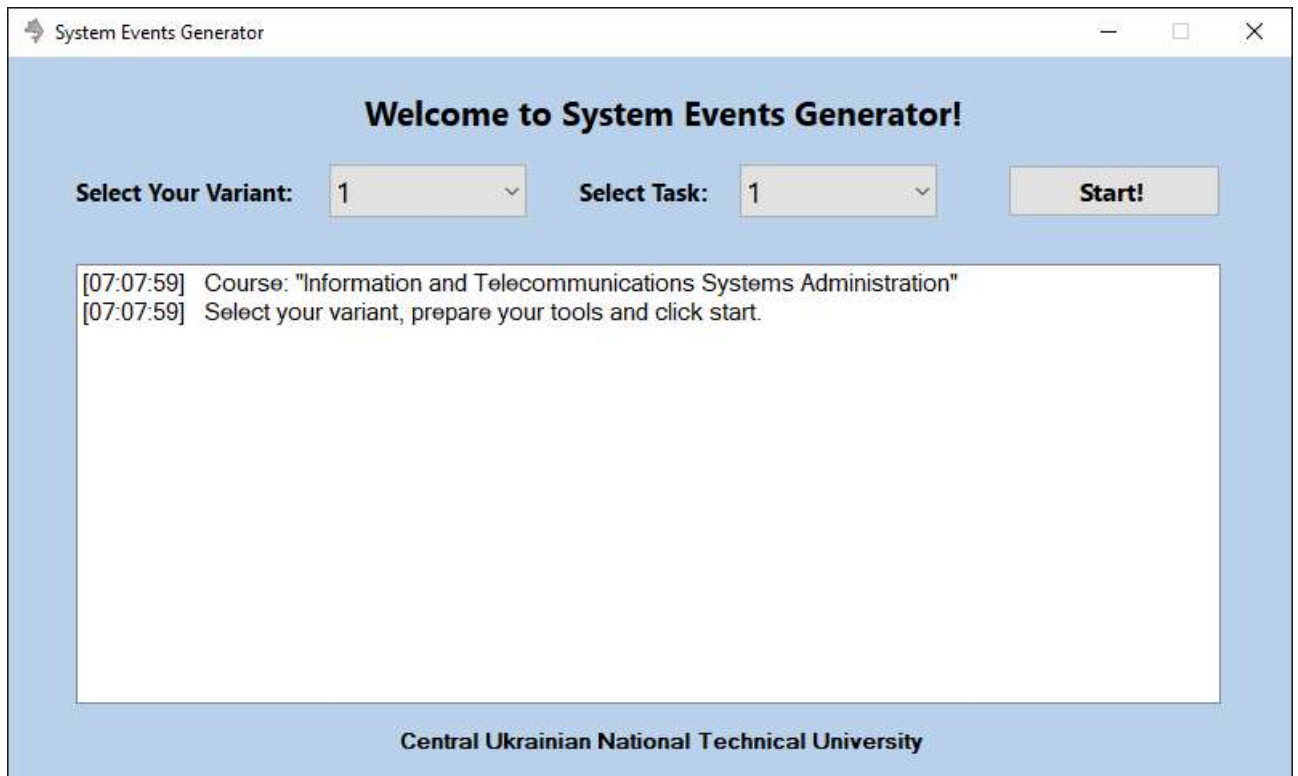


Рисунок 3.5 – Головне вікно генератора подій

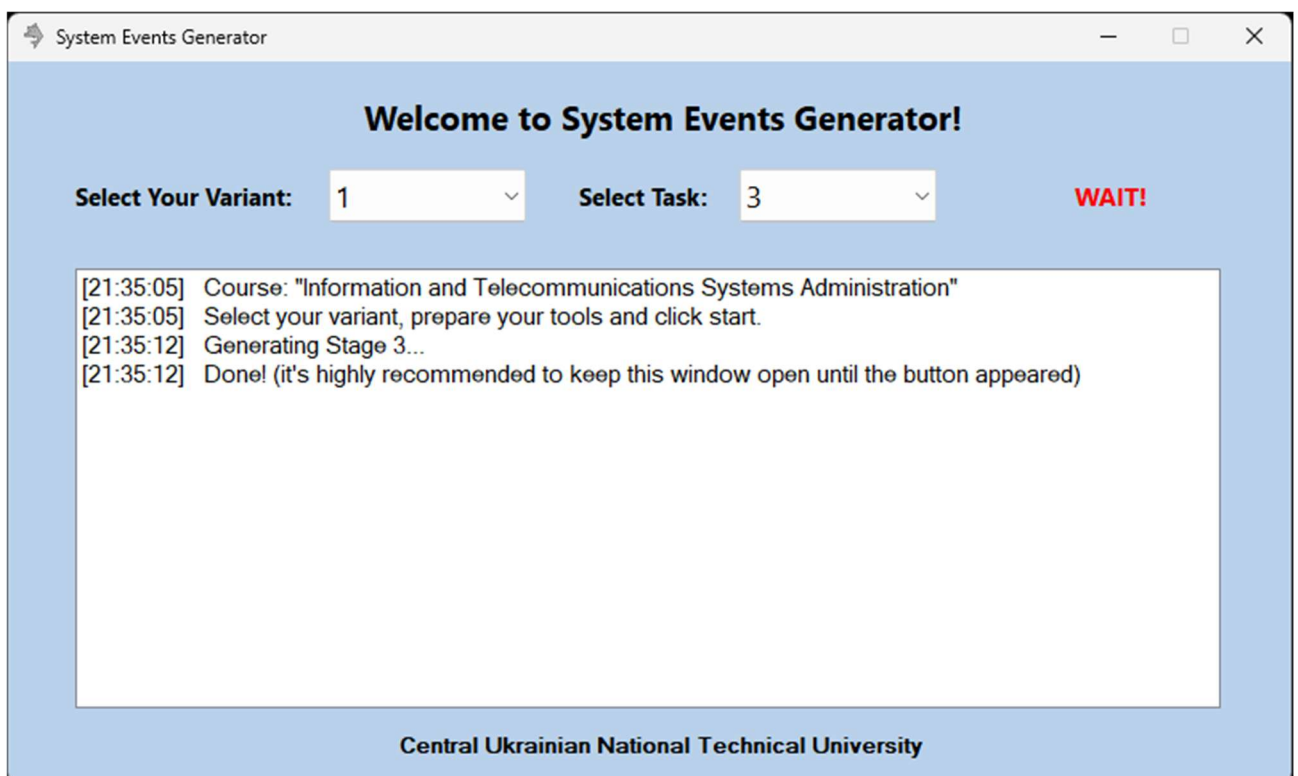


Рисунок 3.6 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 3)

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**AITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадajuчих списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.
5. Сформуваи звіт роботи генератора “**AITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовують HyperText Transfer Protocol?
2. В чому відмінність HTTP від HTTPS?
3. Наведіть структуру протоколу HTTPS.
4. Наведіть існуючі методи запитів HTTP.
5. Які групи формуються коди статусів HTTP?
6. Яка структура HTTP-запиту? Які основні частини він містить?
7. Яка структура HTTP-відповіді? Які дані вона включає?

8. Що таке методи HTTP-запитів? Які основні методи використовуються (GET, POST, PUT, DELETE, PATCH, OPTIONS, HEAD)?
9. Як працює метод GET у HTTP? У чому його особливості?
10. Як працює метод POST у HTTP? Коли він використовується?
11. Що таке HTTP-заголовки? Які існують типи заголовків у HTTP-запитах та відповідях?
12. Як працює сесія HTTP? Чому HTTP є статичним протоколом?
13. Що таке статичний та динамічний контент у контексті HTTP?
14. Як працюють куки (cookies) у HTTP? Яка їхня роль у підтримці сесій?
15. Що таке HTTPS? Які основні відмінності між HTTP та HTTPS?
16. Як працює SSL/TLS у контексті HTTPS? Що таке шифрування на рівні транспортного шару?
17. Що таке сертифікати безпеки (SSL/TLS-сертифікати)? Як вони забезпечують безпеку у HTTPS?
18. Яка роль Центру сертифікації (CA) у HTTPS? Як працює довірчий ланцюжок сертифікатів?
19. Що таке самопідписані сертифікати, і в яких випадках вони використовуються?
20. Які існують версії протоколу SSL/TLS? Чому від SSL було відмовлено на користь TLS?
21. Що таке атака "Man-in-the-Middle" (MiTM), і як HTTPS допомагає захистити від неї?
22. Як браузері перевіряють дійсність SSL/TLS-сертифікатів? Що таке відгук сертифікатів (certificate revocation)?
23. Що таке HSTS (HTTP Strict Transport Security) і яка його роль у забезпеченні безпеки HTTPS?
24. Що таке "mixed content" (змішаний контент) у HTTPS, і чому це може бути проблемою?

25. Як HTTP/2 відрізняється від HTTP/1.1? Які основні переваги HTTP/2?

26. Що таке "header compression" у HTTP/2, і як це впливає на продуктивність?

27. Що таке HTTP/3 і які його основні переваги порівняно з HTTP/2?

28. Як працюють редиректи (HTTP Redirects)? Які існують типи редиректів (301, 302, 307)?

29. Що таке HTTP-заголовок "Referer", і яка його роль у веб-безпеці?

30. Що таке коди стану (status codes) у HTTP/HTTPS? Які вони мають категорії?

31. Які існують коди стану 1xx? Що вони означають? Наприклад, 100 Continue.

32. Що означає код 200 OK? Яка його роль у комунікації між клієнтом і сервером?

33. Що таке код стану 201 Created, і в яких випадках він використовується?

34. Що означає код 204 No Content, і коли він повинен бути використаний?

35. Що таке код стану 301 Moved Permanently, і як він впливає на перенаправлення трафіку?

36. Чим відрізняється код 302 Found від коду 307 Temporary Redirect?

37. Що означає код 304 Not Modified, і як він використовується у кешуванні контенту?

38. Що таке код стану 400 Bad Request? Які можливі причини його виникнення?

39. Що означає код 401 Unauthorized? Як він використовується у контексті аутентифікації?

40. Що таке код 403 Forbidden, і які сценарії його використання?

41. Що означає код 404 Not Found? Як він використовується при роботі з веб-ресурсами?
42. Що таке код стану 405 Method Not Allowed, і як він виникає?
43. Що означає код 408 Request Timeout? Які причини його появи?
44. Що таке код 409 Conflict, і коли він використовується?
45. Що означає код 410 Gone, і коли він застосовується?
46. Що таке код 429 Too Many Requests, і які його можливі наслідки?
47. Що означає код 500 Internal Server Error? Як адміністратор може виправити цю помилку?
48. Що таке код 501 Not Implemented, і які сценарії його використання?
49. Що означає код 502 Bad Gateway, і як його виправити?
50. Що таке код 503 Service Unavailable, і які причини його виникнення?
51. Що означає код 504 Gateway Timeout? Як він пов'язаний з мережею?
52. Що таке код 505 HTTP Version Not Supported, і в яких випадках він з'являється?
53. Як HTTP/HTTPS-коди стану можуть бути використані для діагностики проблем у мережевих додатках?
54. Що таке "soft 404 error"? Як його можна виявити та виправити?

ПРАКТИЧНЕ ЗАВДАННЯ №4

ТЕМА: Моніторинг протоколів мережного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем мережного моніторингу протоколів мережного рівня.

ЗНАТИ: Теоретичні основи базової еталонної моделі взаємодії відкритих систем – 3 рівень семирівневої моделі OSI.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**AITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадających списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.
5. Сформувавши звіт роботи генератора “**AITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

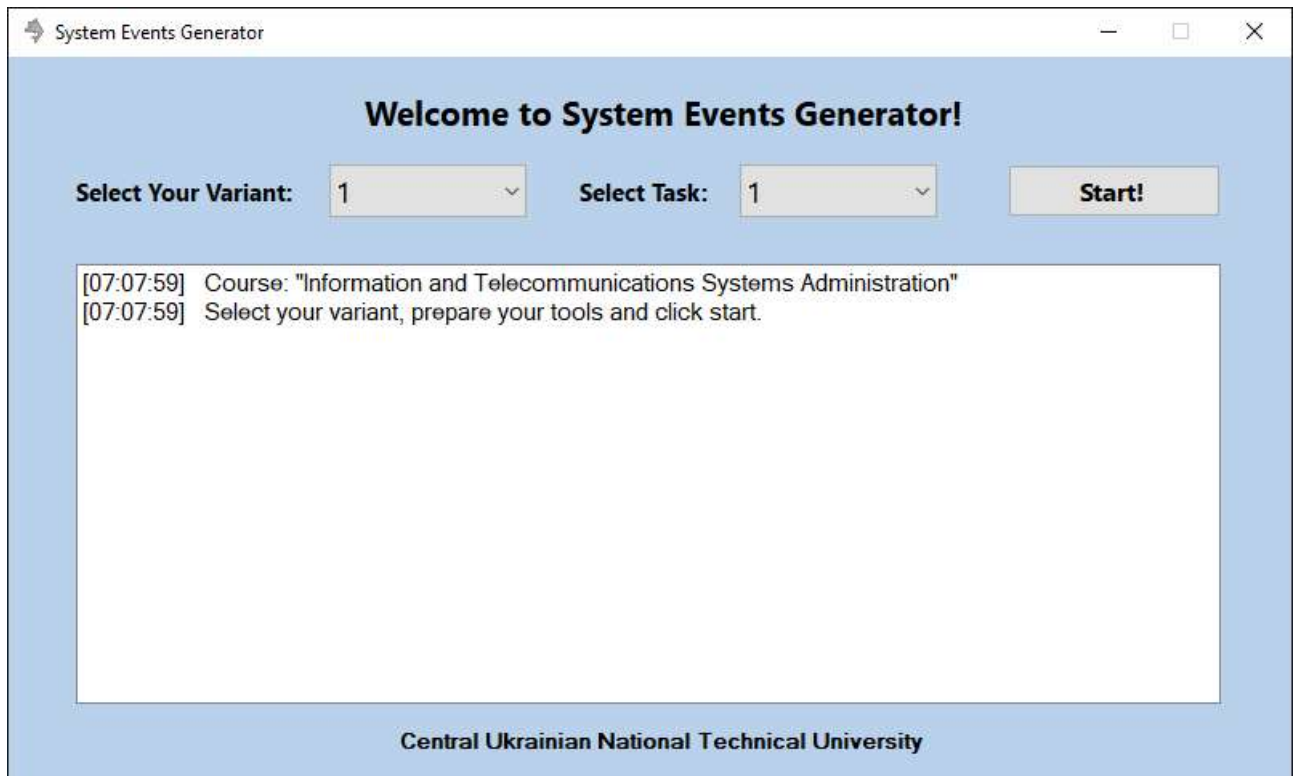


Рисунок 3.7 – Головне вікно генератора подій

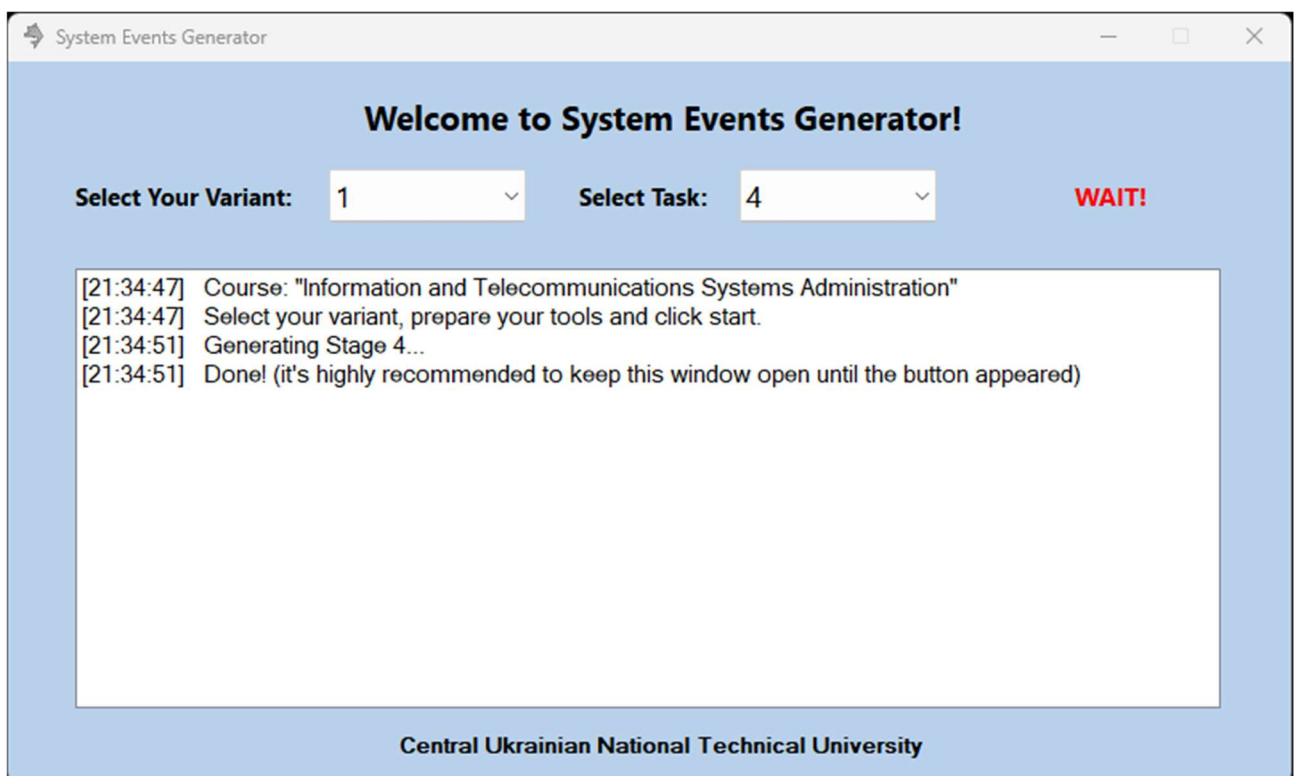


Рисунок 3.8 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 4)

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовують ЕСНО запити?
2. Яка відома мережна утиліта використовує ЕСНО запити?
3. Наведіть приклад роботи протоколу ICMP?
4. Яка використовується структура у ICMP пакеті?
5. Для чого використовується IPv4?
6. Наведіть приклад формування адреси на основі IPv4.
7. Для чого використовується IPv6?
8. Наведіть приклад формування адреси на основі IPv6.
9. Роз'ясніть термінологічне скорочення ICMP Header.
10. Роз'ясніть термінологічне скорочення ICMP Payload.
11. Що таке протокол ICMP (Internet Control Message Protocol)? Яка його основна функція в мережі?
12. Які типи повідомлень ICMP існують? Які їхні основні функції?
13. Що таке ЕСНО-запит (ICMP Echo Request)? Для чого він використовується?
14. Що таке ЕСНО-відповідь (ICMP Echo Reply)? Яка її роль у мережевих комунікаціях?
15. Як працює команда ping у контексті протоколу ICMP? Які дані вона передає та отримує?
16. Що таке TTL (Time to Live) у контексті ICMP? Як TTL використовується при виконанні ЕСНО-запитів?
17. Які можливі причини виникнення помилок при виконанні ping (наприклад, запит не отримав відповіді)?
18. Що таке ICMP-повідомлення типу "Destination Unreachable"? Які причини його виникнення?
19. Як працює ICMP-повідомлення "Time Exceeded"? У яких випадках воно надсилається?

20. Що таке ICMP Redirect (перенаправлення)? Яка його роль у маршрутизації трафіку?
21. Як ICMP використовується для діагностики проблем у мережі?
22. Які ризики пов'язані з використанням ICMP у мережах? Як можна зловживати ICMP для атак (наприклад, DDoS)?
23. Як захистити мережу від зловживань ICMP, таких як атаки типу "Smurf" або "Ping Flood"?
24. Що таке ICMPv6, і як він відрізняється від ICMP для IPv4?
25. Як ICMP-повідомлення допомагають визначати максимальний розмір пакета (MTU) у мережі?
26. Що таке "traceroute"? Як протокол ICMP використовується для визначення маршруту пакетів у мережі?
27. Яка роль ICMP у процесі розпізнавання проблем з маршрутизацією та доступністю хостів?
28. Як можна відключити або обмежити ICMP ECHO-запити для підвищення безпеки мережі?
29. Що таке ICMP-повідомлення типу "Parameter Problem"? Як вони виникають?
30. Які інші типи ICMP-повідомлень існують, крім ECHO-запитів та відповідей? Яка їхня роль?
31. Що таке IPv4 (Internet Protocol version 4)? Які його основні характеристики?
32. Як виглядає IPv4-адреса? Які частини вона містить?
33. Що таке класова адресація у IPv4? Які існують класи адрес (A, B, C, D, E)?
34. Як визначити клас IPv4-адреси за її першими бітами?
35. Що таке мережева адреса та адреса хоста у контексті IPv4?
36. Що таке маска підмережі (subnet mask)? Як вона використовується для визначення мережевої частини IPv4-адреси?

37. Що таке CIDR (Classless Inter-Domain Routing)? Як він впливає на адресацію у мережах?
38. Як працює механізм NAT (Network Address Translation) у контексті IPv4? Які його переваги та недоліки?
39. Що таке приватні IPv4-адреси? Які діапазони зарезервовані для приватних адрес?
40. Що таке публічні IPv4-адреси? Як вони відрізняються від приватних?
41. Що таке loopback-адреса в IPv4? Яке її призначення?
42. Що таке broadcast-адреса? Як вона використовується у мережах IPv4?
43. Як працює ARP (Address Resolution Protocol) у контексті IPv4? Для чого він використовується?
44. Що таке DHCP (Dynamic Host Configuration Protocol)? Як він допомагає у призначенні IPv4-адрес?
45. Що таке IP-конфлікт, і які можуть бути наслідки його виникнення?
46. Як відбувається поділ на підмережі (subnetting) у IPv4? Які переваги це дає?
47. Що таке мережа класу C з маскою /24? Скільки хостів вона може містити?
48. Як можна визначити підмережу та діапазон IP-адрес за допомогою CIDR-нотації?
49. Що таке multicast-адресація у IPv4? Яке її призначення?
50. Як працює маршрутизація в IPv4? Які існують типи маршрутів (статичні, динамічні)?
51. Що таке IPv6 (Internet Protocol version 6)? Які його основні відмінності від IPv4?
52. Як виглядає IPv6-адреса? Яка її структура?

53. Що таке подвійна двокрапка (::) у IPv6, і як вона використовується?
54. Які існують типи IPv6-адрес (унікальні локальні, глобальні, мультикаст, лінк-локальні)?
55. Що таке лінк-локальна адресація у IPv6? Як вона використовується?
56. Що таке глобальна унікальна адреса (global unicast address) у IPv6?
57. Як IPv6 забезпечує автоматичну конфігурацію адреси без використання DHCP (SLAAC)?
58. Що таке відсутність NAT у IPv6, і які переваги це дає?
59. Як працює DNS у контексті IPv6? Що таке запис AAAA?
60. Що таке ICMPv6, і як він взаємодіє з IPv6?
61. Що таке приватна адресація у IPv6? Які діапазони для цього зарезервовані?
62. Як працює механізм адресної компресії у IPv6?
63. Що таке IPv6-маршрутизація? Які існують основні протоколи маршрутизації для IPv6?
64. Що таке dual-stack технологія? Як вона допомагає у переході з IPv4 на IPv6?
65. Як працює механізм тунелювання у IPv6? Що таке 6to4 і Teredo?
66. Що таке EUI-64 формат у IPv6, і як він використовується для формування адрес?
67. Як відрізняється multicast-адресація у IPv6 від IPv4?
68. Що таке IPv6 адреса локального призначення (site-local address)? Як вона використовується?

ПРАКТИЧНЕ ЗАВДАННЯ №5

ТЕМА: Моніторинг протоколів транспортного рівня інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем мережного моніторингу протоколів транспортного рівня.

ЗНАТИ: Теоретичні основи базової еталонної моделі взаємодії відкритих систем – 4 рівень семирівневої моделі OSI.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**AITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадających списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.
5. Сформулювати звіт роботи генератора “**AITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

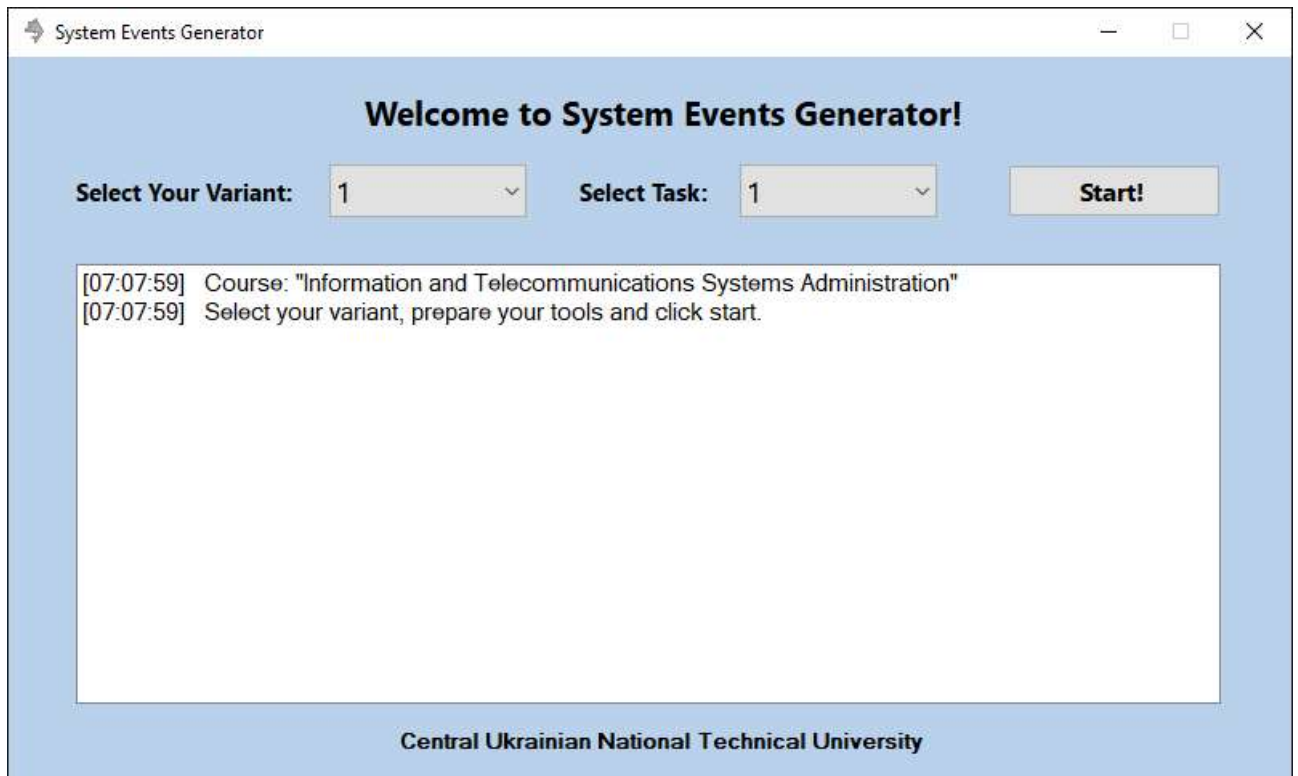


Рисунок 3.9 – Головне вікно генератора подій

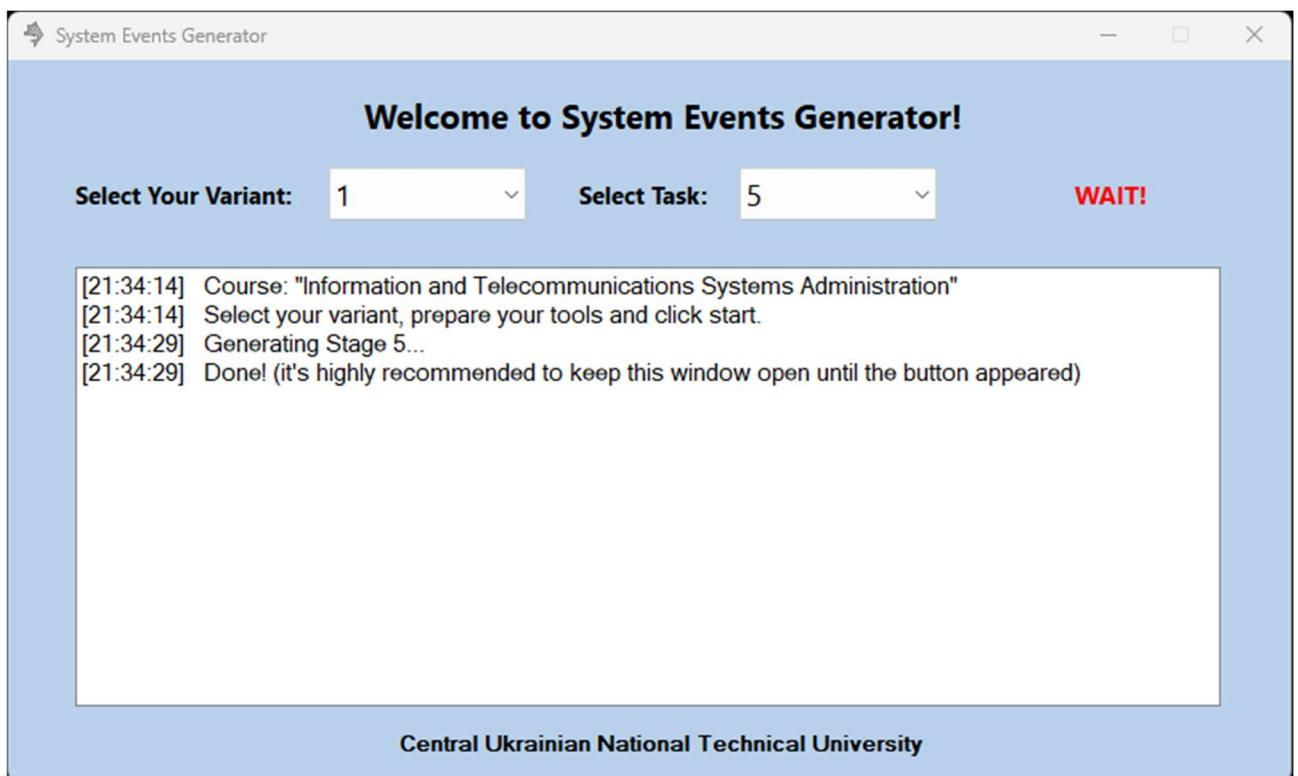


Рисунок 3.10 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 5)

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовують стек протоколів TCP/IP?
2. Наведіть формат TCP сегменту.
3. Перерахуйте поля заголовків TCP сегменту та керуючі біти.
4. Наведіть приклад використання протоколу TCP.
5. Для чого використовуються TCP-порти?
6. Наведіть приклади використання відомих TCP-портів.
7. У чому різниця TCP та UDP портів?
8. Порти TCP та UDP не залежать один від одного?
9. Яку кількість TCP портів можна відкрити одночасно?
10. Що таке стек протоколів TCP/IP? Яку роль він відіграє у розслідуваннях інцидентів?
11. Як використовувати протоколи TCP/IP для аналізу мережевого трафіку під час розслідування кіберінциденту?
12. Як протокол IP може бути використаний для відстеження джерела атаки або зловмисного трафіку?
13. Яка роль ICMP у розслідуваннях інцидентів? Як ICMP може бути використаний для діагностики мережеских проблем або для виявлення сканування мережі?
14. Як ARP (Address Resolution Protocol) може бути використаний для виявлення підміни MAC-адрес (ARP spoofing) під час розслідування?
15. Що таке фрагментація у контексті IP, і як вона може вплинути на аналіз мережевого трафіку під час інциденту?
16. Як DHCP-журнали можуть допомогти в ідентифікації пристроїв у мережі під час розслідування інциденту?
17. Як протокол DNS може бути використаний для виявлення DNS-тунелювання або інших форм зловживання DNS у мережі?
18. Яка роль протоколу FTP у розслідуваннях інцидентів? Як FTP може бути використаний для несанкціонованого завантаження файлів?

19. Як SNMP (Simple Network Management Protocol) може бути використаний для збору інформації про мережеві пристрої під час розслідування?

20. Як Telnet і SSH можуть бути використані для несанкціонованого доступу до систем, і як це можна виявити під час розслідування?

21. Що таке NTP (Network Time Protocol) і як він може впливати на синхронізацію часу під час розслідування інцидентів?

22. Як використання RARP (Reverse Address Resolution Protocol) може бути виявлене під час розслідування мережевого інциденту?

23. Яка роль протоколу HTTP/HTTPS у розслідуваннях інцидентів? Як аналіз HTTP-заголовків може допомогти виявити шкідливий трафік?

24. Як використовувати мережеві протоколи для збору цифрових доказів під час розслідування інцидентів?

25. Як маршрутизація та таблиці маршрутів можуть допомогти в ідентифікації мережевих шляхів, якими проходив зловмисний трафік?

26. Що таке NAT (Network Address Translation), і як він може впливати на ідентифікацію джерел трафіку під час розслідування?

27. Як аналіз VPN-з'єднань, побудованих на основі протоколів TCP/IP, може допомогти у розслідуванні інцидентів?

28. Що таке GRE (Generic Routing Encapsulation) і як його використання може бути виявлене під час аналізу мережевого трафіку?

29. Як IPsec може забезпечити конфіденційність трафіку, і як його використання може вплинути на розслідування?

30. Що таке TCP (Transmission Control Protocol)? Як його використання впливає на аналіз мережевого трафіку під час розслідування?

31. Як встановлення з'єднання у TCP (трирівнева установка з'єднання) може допомогти в ідентифікації активних з'єднань під час інциденту?

32. Що таке UDP (User Datagram Protocol)? Які особливості UDP слід враховувати під час аналізу інцидентів?

33. Як різниця між TCP та UDP впливає на ідентифікацію несанкціонованого трафіку у мережі?

34. Як номери портів у TCP та UDP можуть бути використані для ідентифікації додатків і сервісів під час розслідування інцидентів?

35. Як TCP-журнали можуть допомогти виявити підозрілу активність у мережі?

36. Як аналіз UDP-трафіку може допомогти в ідентифікації атак типу DDoS під час розслідування інцидентів?

37. Що таке контрольна сума у TCP та UDP, і як її перевірка може допомогти виявити зміни в трафіку під час розслідування?

38. Як TCP-сполучення можуть бути використані для створення ланцюжка подій під час аналізу інциденту?

39. Що таке SYN-сканування, і як його можна виявити під час аналізу TCP-трафіку?

40. Як затримка у передачі TCP-пакетів (jitter) може бути використана для виявлення прихованих каналів зв'язку?

41. Що таке фрагментація у контексті UDP, і як вона може вплинути на аналіз трафіку під час розслідування?

42. Як TCP та UDP можуть бути використані для тунелювання шкідливого трафіку, і як це виявити під час розслідування?

43. Як логування TCP-з'єднань може допомогти в ідентифікації джерела зловмисного трафіку?

44. Що таке порт-сканування, і як різні типи сканування (TCP SYN, TCP Connect, UDP) можуть бути виявлені?

45. Як UDP може використовуватися для прихованого передавання даних у мережі, і як це виявити під час аналізу?

46. Як TCP-підтвердження (ACK) можуть бути використані для виявлення шкідливої активності у мережі?

47. Як великі обсяги UDP-трафіку можуть вказувати на можливі атаки на сервіс (DoS/DDoS)?

48. Що таке DNS-ампліфікація, і як UDP може бути використаний для таких атак?

49. Як можна використовувати аналіз TCP та UDP для виявлення зловживань службами та сервісами у мережі?

50. Як зловмисники можуть використовувати відкриті TCP-порти для сканування та розвідки у мережі?

51. Як аналіз різних типів трафіку (TCP/UDP) може допомогти у визначенні типу шкідливого програмного забезпечення?

52. Як TCP та UDP-тунелі можуть використовуватися для обходу засобів захисту, і як це виявити під час аналізу?

53. Що таке TCP-reset атака, і як її можна виявити під час аналізу мережевого трафіку?

54. Як велика кількість відхилених UDP-пакетів може вказувати на потенційну атаку на мережу?

55. Як протоколи, побудовані на основі TCP (наприклад, HTTP/HTTPS), можуть бути використані для приховування шкідливого трафіку?

56. Як зловмисники можуть використовувати UDP для передачі команд до ботнетів, і як це виявити під час аналізу?

57. Як можливі аномалії у TCP-послідовності можуть вказувати на спроби втручання або модифікації трафіку?

58. Що таке атака "UDP Flood", і як її можна виявити під час аналізу інциденту?

59. Як аналіз обох типів трафіку (TCP та UDP) може допомогти у складанні повної картини інциденту та відновленні хронології подій?

ПРАКТИЧНЕ ЗАВДАННЯ №6

ТЕМА: Моніторинг параметрів запуску інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем моніторингу консольних параметрів.

ЗНАТИ: Теоретичні основи архітектурної будови ОС Windows.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utils/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**ITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадających списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”.
5. Сформувавши звіт роботи генератора “**ITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

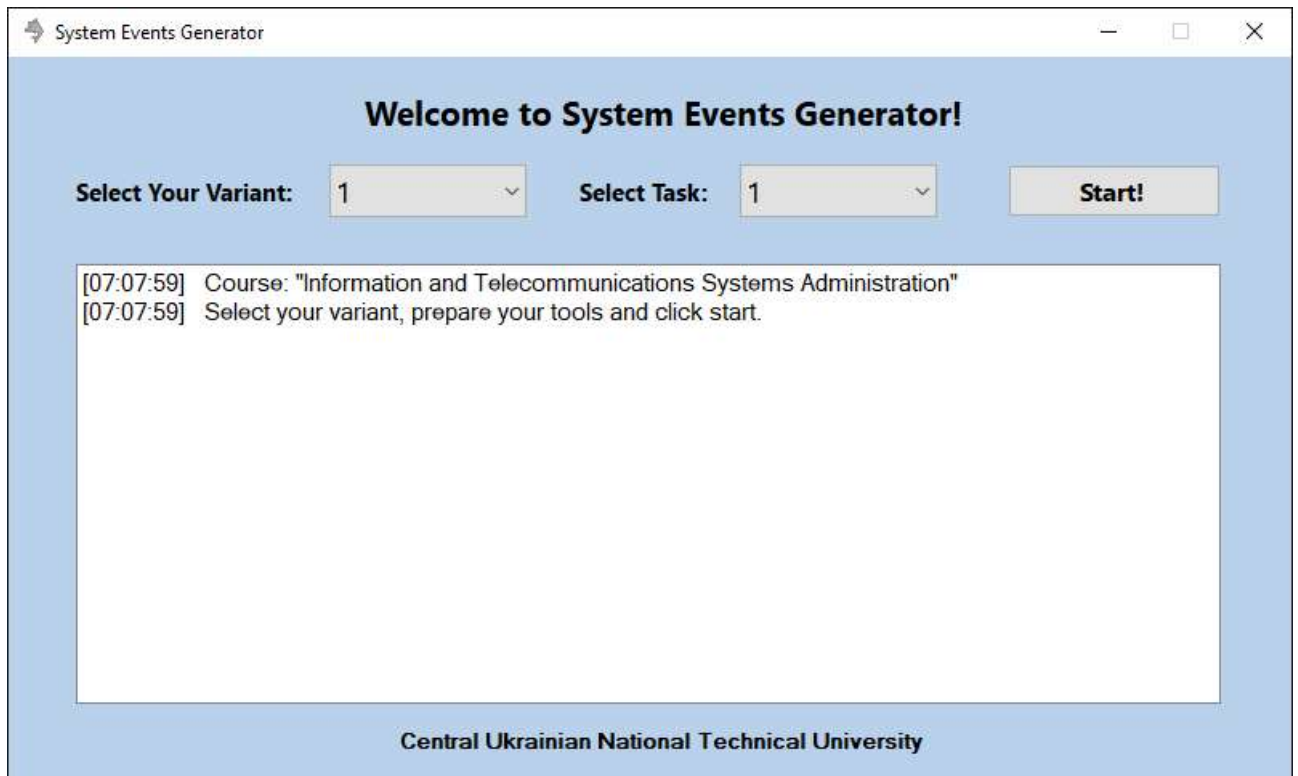


Рисунок 3.11 – Головне вікно генератора подій

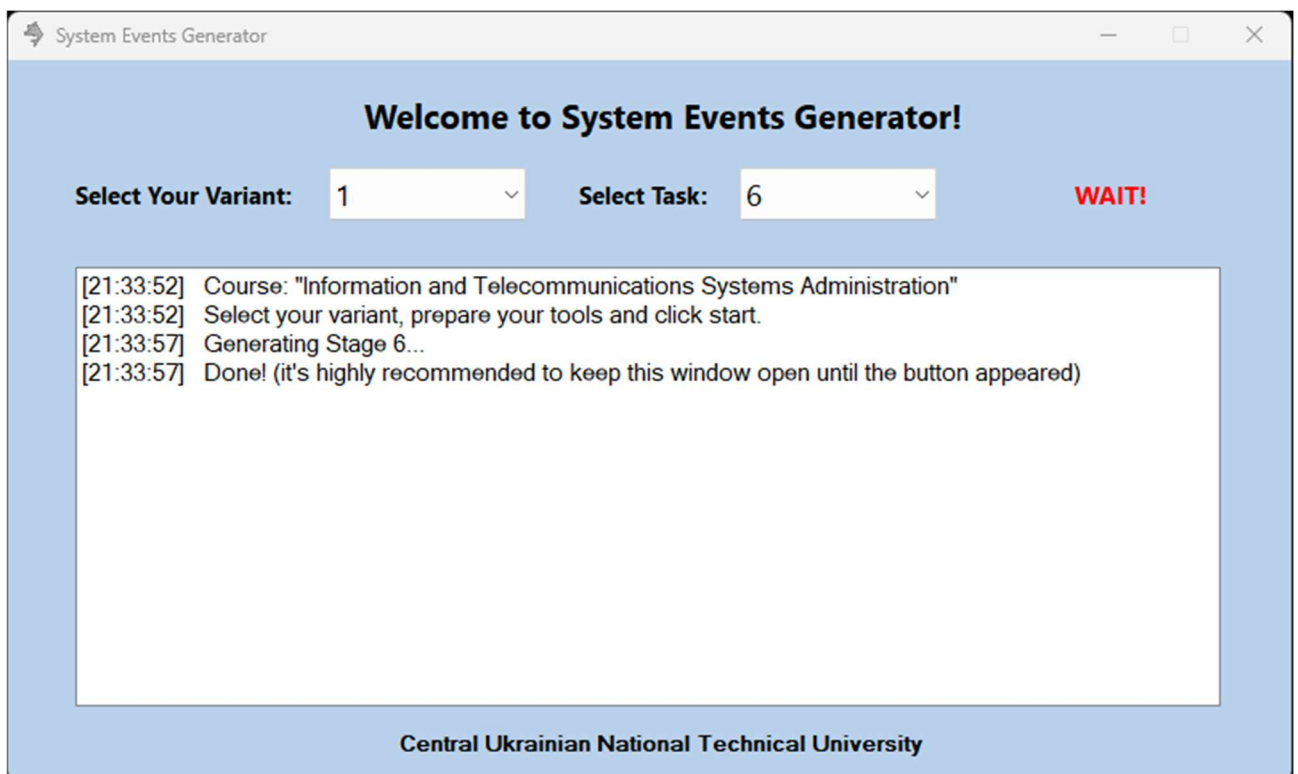


Рисунок 3.12 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 6)

КОНТРОЛЬНІ ЗАПИТАННЯ

1. В яких випадках використовується командний інтерпретатор?
2. В яких випадках використовується пакетний файл (batch file)?
3. Що таке PowerShell?
4. Для чого використовується PowerShell_ise?
5. Для чого використовують командлети?
6. Наведіть приклад формування скрипту з використанням командлети GET-CONTENT.
7. Наведіть приклад формування скрипту з використанням командлетів GET-SERVICE/ START-SERVICE/STOP-SERVICE.
8. Наведіть приклад формування скрипту з використанням командлетів GET-PROCESS/ STOP-PROCESS.
9. Що таке PowerShell, і як він відрізняється від інших інтерпретаторів командного рядка в Windows?
10. Які основні можливості PowerShell роблять його корисним для фахівців з кібербезпеки та DFIR?
11. Як PowerShell інтегрується з операційною системою Windows для виконання адміністративних завдань?
12. Що таке командлети (cmdlets) у PowerShell? Які їхні основні властивості?
13. Як фахівець з DFIR може використовувати PowerShell для збору доказів під час розслідування інциденту?
14. Яка роль PowerShell у автоматизації завдань у контексті кібербезпеки?
15. Як PowerShell може використовуватися для віддаленого виконання команд? Які ризики це несе?
16. Як відстежувати використання PowerShell у системі під час інциденту?

17. Які журнали подій (event logs) фіксують активність PowerShell, і як їх можна використовувати для розслідування?
18. Що таке PowerShell Core, і як він відрізняється від Windows PowerShell?
19. Як PowerShell може бути використаний для управління мережею під час реагування на інцидент?
20. Як відстежувати виконання скриптів PowerShell у реальному часі?
21. Що таке модулі PowerShell, і як вони можуть бути використані для розширення функціональності у контексті DFIR?
22. Як PowerShell може бути використаний для збору інформації про систему, такої як процеси, служби та підключені пристрої?
23. Які ризики пов'язані з використанням PowerShell у шкідливих цілях, і як їх виявити під час розслідування?
24. Як створити скрипт PowerShell для збору логів з різних систем?
25. Як командлет Get-Process може бути використаний для виявлення підозрілих процесів під час розслідування інциденту?
26. Як командлет Get-Service допомагає ідентифікувати аномальні або шкідливі служби на комп'ютері?
27. Як використовувати командлет Get-EventLog для аналізу подій безпеки в Windows?
28. Як командлет Get-WmiObject може бути використаний для збору інформації про систему під час інциденту?
29. Як командлет Get-ChildItem може бути використаний для пошуку файлів за специфічними атрибутами (наприклад, приховані файли)?
30. Як за допомогою PowerShell можна перевірити хеші файлів (наприклад, використовуючи командлет Get-FileHash) під час аналізу?
31. Як командлет Get-NetTCPConnection може допомогти в аналізі активних мережевих з'єднань під час інциденту?

32. Як командлет Invoke-WebRequest може бути використаний для виявлення підозрілих завантажень або доступу до шкідливих веб-сайтів?
33. Як командлет Get-Process може бути використаний для збору інформації про процеси, які можуть бути пов'язані з шкідливим ПЗ?
34. Як PowerShell можна використовувати для аналізу автозавантаження програм (наприклад, використовуючи командлет Get-ItemProperty для ключів реєстру)?
35. Як командлет Get-Volume може бути використаний для аналізу підключених дисків під час розслідування?
36. Як командлет Export-Csv може бути використаний для збереження зібраних даних у зручному для аналізу форматі?
37. Як командлет Start-Process може бути використаний для запуску сторонніх інструментів або виконання команд під час розслідування?
38. Як командлет Measure-Object може бути використаний для аналізу обсягів даних під час оцінки впливу інциденту?
39. Як створити PowerShell-скрипт для автоматизованого збору артефактів з зараженої системи?
40. Як PowerShell може бути використаний для створення списку активних мережевих з'єднань і їхньої аналізу?
41. Як PowerShell може допомогти у відновленні та аналізі файлів журналів безпеки під час розслідування?
42. Як за допомогою PowerShell можна виявити та зупинити підозрілі процеси або служби?
43. Як використовувати PowerShell для моніторингу змін у реєстрі Windows, які можуть свідчити про інцидент безпеки?
44. Як за допомогою PowerShell можна створити скрипт для автоматичного створення звіту про стан системи після інциденту?
45. Як PowerShell можна використовувати для перевірки наявності відомих вразливостей або неправильних конфігурацій у системі?

46. Як за допомогою PowerShell можна перевірити права доступу до файлів і каталогів під час розслідування?

47. Як командлети PowerShell можуть допомогти у виявленні несанкціонованого доступу до мережевих ресурсів?

48. Як PowerShell можна використовувати для створення і підтримки списку контрольованих програм та процесів у середовищі реагування на інциденти?

49. Як за допомогою PowerShell можна відстежувати та аналізувати активність користувачів у системі?

50. Як PowerShell може бути використаний для швидкого збирання інформації з декількох систем одночасно у великих мережах?

51. Як PowerShell-скрипти можуть допомогти у виявленні аномальних дій, що можуть свідчити про компрометацію облікових записів?

52. Як за допомогою PowerShell можна зібрати та проаналізувати дані про підключені USB-пристрої?

53. Як PowerShell може бути використаний для проведення пост-мортем аналізу після успішного завершення інциденту?

54. Як PowerShell-скрипти можуть бути інтегровані в автоматизовані системи реагування на інциденти?

55. Як за допомогою PowerShell можна перевірити журнали подій на наявність ознак підозрілої активності або маніпуляцій?

56. Як PowerShell може бути використаний для аналізу мережевого трафіку та виявлення аномалій?

57. Як PowerShell може допомогти у створенні резервних копій важливих системних файлів та журналів для подальшого аналізу?

58. Як PowerShell може використовуватися для віддаленого збору доказів та моніторингу під час інциденту у великих організаціях?

ПРАКТИЧНЕ ЗАВДАННЯ №7

ТЕМА: Моніторинг паралельних обчислень процесів інформаційно-телекомунікаційної системи в реальному часі з формуванням звітності для відновлення штатного функціонування.

МЕТА: Отримати практичні навички використання систем моніторингу консольних параметрів.

ЗНАТИ: Теоретичні основи архітектурної будови ОС Windows та основи паралельного програмування.

ВМІТИ: Встановлювати дистрибутиви технічних засобів та утиліт для управління, діагностики, усунення неполадок та моніторингу ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдання пропонується використовувати документацію, технічні засоби та утиліти, призначені для управління, діагностики, усунення неполадок і моніторингу системи:

- Sysinternals. <https://docs.microsoft.com/en-us/sysinternals/>
- Nirsoft <https://www.nirsoft.net/utills/index.html>
- Washington University Computer Science & Engineering
https://www.cse.wustl.edu/~jain/cse567-06/ftp/os_monitors/index.html/
- Flexense Ltd SysGauge <https://www.sysgauge.com/>

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

1. Визначити свій індивідуальний варіант завдання.
2. Запустити генератор подій “**AITS System EventsGenerator (1-7 lab).exe**” що зображено на рисунку 1.
3. Вибрати номер індивідуального варіанту (Select your variant) та номер практичного завдання (select task) у випадających списках як це показано на рисунку 2.
4. Запустити та налаштувати обраний засіб моніторингу та натиснути кнопку “Start”. Для тестування роботи засобу пропонується також використати програму “NullTestProject (7 lab)”. Це пустий скомпільований проєкт у оболонці Microsoft Visual Studio на мові програмування C#. Пустий проєкт дозволяє побачити як працює багатопотоковість по замовчанню та розглянути властивості операційної системи.

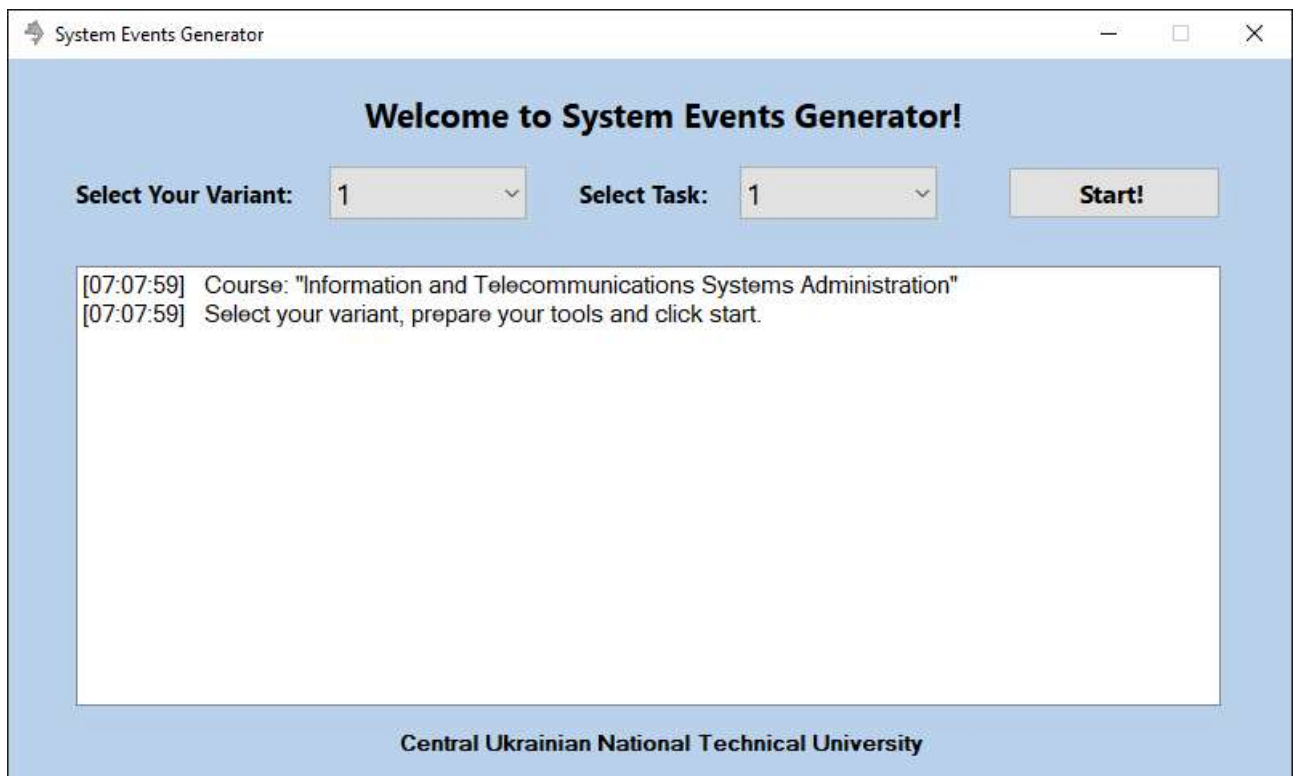


Рисунок 3.13 – Головне вікно генератора подій

5. Сформувати звіт роботи генератора “**AITS System EventsGenerator (1-7 lab).exe**” та оформити звіт практичного завдання.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт моніторингу індивідуального варіанту завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність проведеного моніторингу.
- Відповіді на контрольні питання.

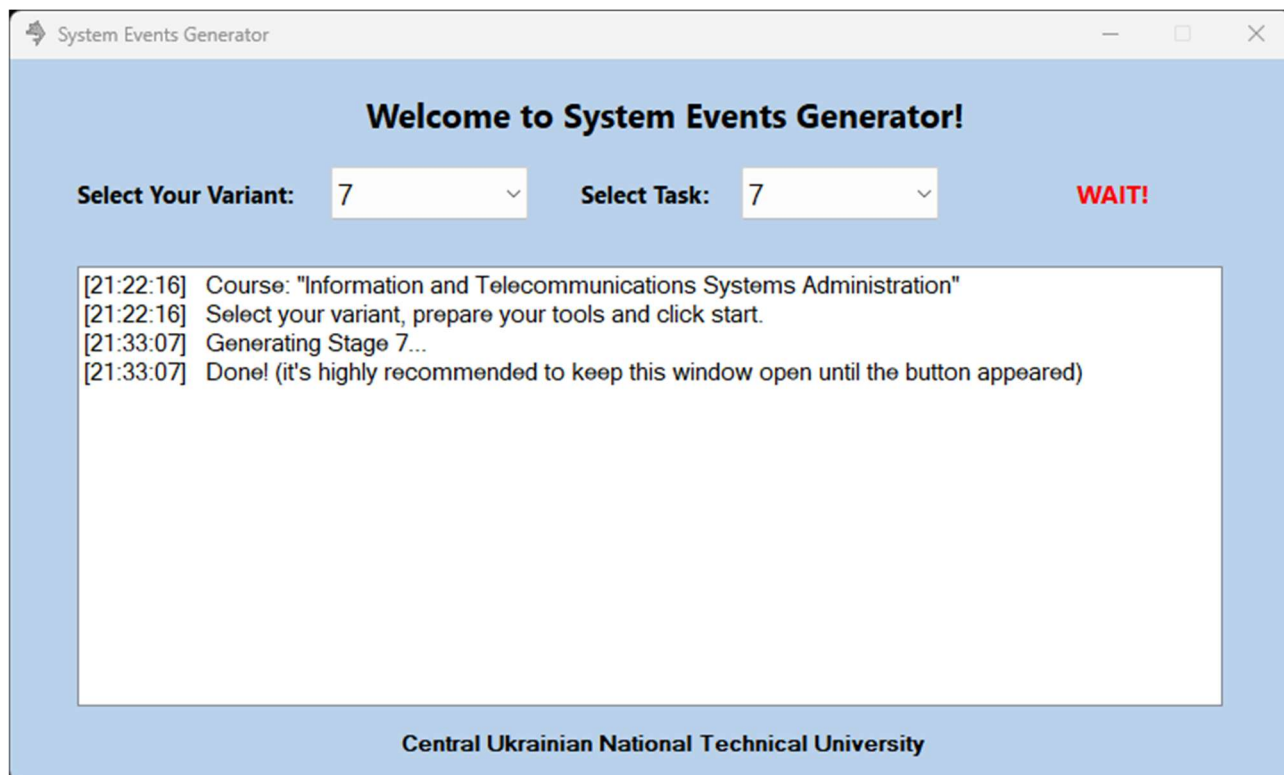


Рисунок 3.14 – Приклад встановлення та запуску генератора подій
(Варіант 7, Практичне завдання 7)

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для яких цілей використовують потоки?
2. Дайте загальну характеристику потоків.
3. Скільки потоків (thread) використовується у програмному забезпеченні по замовчанню?
4. В чому різниця між потоком та процесом?
5. Які існують переваги та недоліки використання багатопотоковості?
6. Які існують рівні пріоритету потоків?
7. Які існують стани потоків?
8. Що таке процес в операційній системі Windows? Які його основні компоненти?
9. Як Windows створює новий процес? Які системні виклики використовуються для цього?
10. Як у контексті DFIR можна відстежувати створення нових процесів у Windows?
11. Як за допомогою утиліти Task Manager або командлетів PowerShell можна визначити активні процеси на комп'ютері?
12. Що таке PID (Process Identifier), і як його використовують для ідентифікації процесів під час розслідування інцидентів?
13. Як аналіз процесів може допомогти у виявленні шкідливого програмного забезпечення на комп'ютері?
14. Як процеси взаємодіють з іншими системними ресурсами, такими як пам'ять і файлові системи?
15. Які інструменти доцільно використовувати для аналізу процесів під час розслідування інцидентів, зокрема Process Explorer, Process Hacker або інші засоби з набору Sysinternals?
16. Як можна виявити аномальні або підозрілі процеси в системі?

17. Що таке батьківські (parent) і дочірні (child) процеси? Як аналізувати їх взаємозв'язки під час розслідування?

18. Як відстежувати та аналізувати історію створення процесів у системі під час інциденту?

19. Що таке "процеси-зомбі" (zombie processes)? Як їх виявити та усунути?

20. Як шкідливі процеси можуть приховувати свою присутність у системі, і як їх виявити під час розслідування?

21. Що таке "живучі" (persistent) процеси, і як вони можуть використовуватися шкідливими програмами для утримання в системі?

22. Як аналізувати процеси, які працюють у контексті облікових записів користувачів, для виявлення потенційних загроз?

23. Що таке потік (thread) у контексті операційної системи Windows? Яка його роль у виконанні процесів?

24. Як Windows керує потоками в межах одного процесу?

25. Як потоки можуть бути використані для виконання багатозадачності в операційній системі?

26. Які інструменти можна використовувати для аналізу потоків під час розслідування інцидентів?

27. Як визначити кількість потоків, що працюють у рамках одного процесу?

28. Як аналіз потоків може допомогти у виявленні шкідливих програм, які використовують багатопоточність?

29. Що таке контекст перемикання (context switching) потоків? Як воно впливає на продуктивність і безпеку системи?

30. Як шкідливі програми можуть використовувати потоки для прихованої діяльності у системі?

31. Що таке "нитки-зомбі" (zombie threads)? Як їх виявити та усунути?

32. Як можна відстежувати створення нових потоків у процесі під час інциденту?

33. Як аналізувати поведінку потоків у реальному часі для виявлення підозрілої активності?

34. Що таке "останній потік процесу", і як його завершення впливає на стан процесу?

35. Як аналіз потоків може допомогти у виявленні шкідливих ін'єкцій коду (code injections) у легітимні процеси?

36. Як розслідувати інциденти, пов'язані з аномальною кількістю потоків у процесах?

37. Як шкідливі програми можуть використовувати низькорівневі функції управління потоками для утруднення свого виявлення?

38. Що таке пріоритет процесу в операційній системі Windows? Як він впливає на виконання процесів?

39. Які існують рівні пріоритетів процесів у Windows? Як вони визначають порядок виконання процесів?

40. Як визначити та змінити пріоритет процесу за допомогою інструментів Windows або PowerShell?

41. Як аналіз пріоритетів процесів може допомогти у виявленні шкідливої активності в системі?

42. Як шкідливі програми можуть використовувати зміну пріоритетів для впливу на роботу системи?

43. Що таке пріоритет потоку? Як він впливає на виконання завдань у межах процесу?

44. Як визначити та змінити пріоритет потоку? Які інструменти можна для цього використовувати?

45. Як аномалії у пріоритетах процесів або потоків можуть вказувати на компрометацію системи?

46. Що таке "підвищений" (high) і "реального часу" (real-time) пріоритет? Як їх використання впливає на систему?

47. Як зловмисники можуть маніпулювати пріоритетами для утруднення виявлення шкідливих процесів або потоків?

48. Як можна моніторити зміни пріоритетів процесів у реальному часі для виявлення підозрілої активності?

49. Що таке "зниження" (demotion) пріоритету, і як воно використовується для керування ресурсами системи?

50. Як змінення пріоритетів може впливати на ефективність виконання задач під час інциденту?

51. Як шкідливі програми можуть використовувати низькопріоритетні потоки для приховування своєї активності?

52. Як аналіз пріоритетів допомагає виявити процеси або потоки, які можуть бути частиною атаки на відмову у обслуговуванні (DoS)?

53. Які існують стани процесів у Windows (наприклад, Running, Waiting, Terminated)? Що вони означають?

54. Як аналіз станів процесів може допомогти у виявленні шкідливої активності або аномалій у системі?

55. Що таке "процес у стані очікування" (waiting state)? Як він може бути використаний для приховування шкідливої активності?

56. Як відстежувати зміни станів процесів під час інциденту? Які інструменти можна використовувати для цього?

57. Що таке "зупинені" (suspended) процеси? Як їх використання може бути пов'язане з шкідливими діями?

58. Як шкідливі програми можуть маніпулювати станами процесів для утруднення виявлення?

59. Що таке "процес завершено" (terminated state), і як аналіз цього стану може допомогти у розслідуванні інцидентів?

60. Як стани потоків впливають на виконання процесів і загальну продуктивність системи?

61. Як відстежувати стан потоків під час інциденту для виявлення прихованих загроз?

62. Як стани процесів можуть вказувати на присутність "процесів-зомбі" або "ниток-зомбі" у системі?

63. Що таке "процеси-примари" (ghost processes)? Як їх виявити під час розслідування?

64. Як шкідливі програми можуть використовувати змінення станів потоків для ускладнення аналізу їхньої активності?

65. Як стани процесів та потоків можуть бути використані для моніторингу загального стану системи під час інциденту?

66. Як аналіз станів допомагає визначити, які процеси або потоки є критичними для розуміння природи інциденту?

67. Як можна виявити процеси або потоки, що знаходяться у незвичних станах, що можуть вказувати на шкідливу активність?

ПРАКТИЧНЕ ЗАВДАННЯ №8

ТЕМА: АНАЛІЗ СИМВОЛЬНИХ ДАНИХ ДВІЙКОВОГО КОДУ ПЗ

МЕТА: Отримати практичні навички використання методів та інструментів статичного аналізу даних

ЗНАТИ: Основи програмування в ОС Windows.

ВМІТИ: Застосовувати інструментарій статичного аналізу даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Аналіз символічних даних – це процес вилучення читабельних символів та строк формату Ascii та Unicode із двійкового коду програмного забезпечення, що аналізується. З урахуванням можливостей заплутування даних зловмисником (obfuscated string) за допомогою потокового шифрування символічних даних (xor, base64, rc4, та ін.) та відсікання генерованих хибних текстових даних що перешкоджають аналізу.

Для вирішення поставленого завдання практичного завдання пропонується використовувати системну утиліту командного рядка **Flare-Floss** – “floss.exe”.

Можна використовувати інші підходи чи написати власний аналог (вихідний код алгоритму Flare-Floss доступний).

Flare-Floss вирішує проблему з заплутаними та зашифрованими рядками – програма автоматично витягує заплутані (obfuscated) рядки із двійкового коду ПЗ, що аналізується. ПЗ об’єднує та автоматизує різні техніки, щоб виконати декодування рядків.

Замість того, щоб надійно захищати бекдори пакувальниками (наприклад UPX), багато авторів зловмисного ПЗ уникають евристичних виявлень, маскуючи лише ключові частини виконуваного файлу.

Часто ці частини є рядками та ресурсами, які використовуються для налаштування доменів (ftp, https, ip та ін), файлів та інших артефактів зараження. Ці ключові функції не відображатимуться як звичайний текст у виводі утиліти strings.exe, яку використовують під час базового статичного аналізу.

FLARE Obfuscated String Solver (FLOSS, раніше FireEye Labs Obfuscated String Solver) використовує розширені методи статичного аналізу для автоматичної деобфускації рядків із двійкових файлів шкідливого програмного забезпечення. Це поліпшена версія утиліти командного рядка strings.exe від Sysinternals, з можливістю аналізу заплутаних ресурсів.

У зв'язку з великим обсягом інформації використовувати електронну документацію відповідно обраного інструментарію, операційної системи та мови програмування.

Пропонується використовувати:

1. FLARE Obfuscated String Solver (Офіційна сторінка):

<https://github.com/mandiant/flare-floss>

2. FLARE (Source code and exe file Linux, Mac, Windows):

<https://github.com/mandiant/flare-floss/releases>

3. FLARE (Source testfiles):

<https://github.com/mandiant/flare-floss-testfiles/tree/master/src>

Пропонується переглянути додаткові мультимедіа матеріали:

4. FLOSS Every Day: Automatically Extracting Obfuscated Strings from Malware

https://www.youtube.com/watch?v=i2gDruusO3I&pp=ugMICgJydRABGAE%3D&ab_channel=SANSDigitalForensicsandIncidentResponse

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажити файл відповідно варіанту (1.exe-40.exe).

Посилання: <https://drive.google.com/file/d/1qujA4s23LDdqS5-TvSCVjVlr2d33C9m/view?usp=sharing>

(Основне завдання)

Використовуючи обраний засіб обробки заплутаних та зашифрованих рядків проаналізувати виконуваний файл відповідно до індивідуального варіанту (1.exe-40.exe). Отримати декодований унікальний ключ у форматі «Good job key: XX-XX-XX».

(Додаткове завдання)

Реалізувати кодування та декодування даних з перевіркою Flare-Floss на основі одного з наступних вихідних кодів декодування:

- (github.com) [decode-base64](#);
- (github.com) [decode-from-global](#);
- (github.com) [decode-from-heap](#);
- (github.com) [decode-from-stack](#);
- (github.com) [decode-global-stack strings](#);
- (github.com) [decode-in-place](#);
- (github.com) [decode-local-stack strings](#);
- (github.com) [decode-rc4](#);
- (github.com) [decode-reencode-string](#);
- (github.com) [decode-single-byte-xor](#);
- (github.com) [decode-split-stackstrings](#);
- (github.com) [decode-stackstrings-move-to-global](#);
- (github.com) [decode-string-by-index](#);
- (github.com) [decode-string-with-header](#);
- (github.com) [decode-substitution-cipher](#);
- (github.com) [decode-tightstring](#);
- (github.com) [decode-to-global](#);
- (github.com) [decode-to-heap](#);

- (github.com) [decode-to-output-buf](#);
- (github.com) [decode-to-stack-req-mov](#);
- (github.com) [decode-to-stack](#);
- (github.com) [decode-wrapped-decoder](#).

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання.
- Звіт додаткового завдання з **блок схемою алгоритму роботи програми.**
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке FLARE-FLOSS (FireEye Labs Obfuscated String Solver)? Яке його основне призначення у контексті DFIR?
2. Як FLARE-FLOSS допомагає виявляти приховані та обфусковані рядки у шкідливому програмному забезпеченні?
3. Які основні функції виконує FLARE-FLOSS під час аналізу зловмисних програм?
4. Як FLARE-FLOSS може бути використаний для аналізу рядків у виконуваних файлах?
5. Як FLARE-FLOSS інтегрується з іншими інструментами аналізу шкідливих програм у контексті DFIR?
6. Які типи обфускації рядків може розпізнати та декодувати FLARE-FLOSS?

7. Як використання FLARE-FLOSS може допомогти в ідентифікації команд та параметрів, використовуваних шкідливим ПЗ?
8. Які методи використовує FLARE-FLOSS для екстракції рядків із виконуваних файлів?
9. Як FLARE-FLOSS допомагає виявляти комунікації між шкідливими програмами та їхніми серверами управління (C2)?
10. Як FLARE-FLOSS може бути використаний для аналізу динамічно завантажуваних або виконуваних модулів шкідливого ПЗ?
11. Які обмеження має FLARE-FLOSS при аналізі складних методів обфускації?
12. Як результати, отримані за допомогою FLARE-FLOSS, можуть бути використані для зворотного інжинірингу шкідливих програм?
13. Як FLARE-FLOSS допомагає в аналізі PowerShell-скриптів, які були обфусковані зловмисниками?
14. Як можна інтегрувати FLARE-FLOSS у процес автоматичного аналізу зразків шкідливого ПЗ?
15. Які типи виконуваних файлів підтримує FLARE-FLOSS для аналізу?
16. Як FLARE-FLOSS може бути використаний для аналізу прихованих API-викликів у шкідливих програмах?
17. Як FLARE-FLOSS може допомогти у виявленні конфігураційних файлів, захованих у шкідливих програмах?
18. Як результати роботи FLARE-FLOSS можна використовувати для створення YARA-правил?
19. Які додаткові заходи слід вживати, якщо FLARE-FLOSS не може декодувати всі обфусковані рядки?
20. Як FLARE-FLOSS може допомогти у визначенні мети та функціональності шкідливого ПЗ під час розслідування інциденту?
21. Що таке таблиця кодування символів, і яку роль вона відіграє у цифровій криміналістиці та реагуванні на інциденти?

22. Як кодування символів впливає на аналіз текстових файлів під час розслідування кіберінцидентів?
23. Які проблеми можуть виникнути при аналізі файлів із різними кодуваннями символів?
24. Як неправильно визначене кодування може призвести до втрати або перекручення цифрових доказів?
25. Як конвертувати текстові дані між різними кодуваннями символів без втрати інформації під час розслідування інцидентів?
26. Що таке US-ASCII, і чому його розуміння важливе під час аналізу старих або спрощених систем?
27. Як US-ASCII може бути використаний для виявлення прихованих повідомлень у текстових файлах під час розслідування інциденту?
28. Як відрізняються текстові дані в US-ASCII від інших кодувань при перегляді у hex-редакторах?
29. Як можливі конфлікти з кодуванням US-ASCII можуть вплинути на аналіз доказів, таких як логи або скрипти?
30. Як підозрілі символи або коди в US-ASCII можуть вказувати на наявність шкідливого коду або експлойтів?
31. Що таке KOI-8, і як його використання може вплинути на аналіз даних у цифровій криміналістиці?
32. Як розслідувати інциденти, пов'язані з текстами на кирилиці, які були збережені в KOI-8?
33. Які труднощі можуть виникнути при аналізі файлів із кодуванням KOI-8 на сучасних системах?
34. Як можливі проблеми з конвертацією з KOI-8 можуть призвести до втрати важливої інформації під час розслідування?
35. Як відрізнити тексти, закодовані в KOI-8, від текстів в інших кодуваннях, таких як Windows-1251 або UTF-8?
36. Що таке Windows-1251, і чому він важливий для аналізу текстів, написаних на кирилиці?

37. Як проблеми з кодуванням Windows-1251 можуть вплинути на розслідування інцидентів, що стосуються кириличних текстів?
38. Як можна визначити, що текстовий файл або документ було створено у Windows-1251?
39. Які інструменти можна використовувати для перевірки і коректного відображення текстів, закодованих у Windows-1251?
40. Як шкідливий код може використовувати специфіку Windows-1251 для уникнення виявлення під час аналізу?
41. Що таке UTF-8, і чому він став стандартом для текстів у глобальній мережі?
42. Як проблеми з кодуванням UTF-8 можуть виникати під час розслідування інцидентів, що стосуються міжнародних текстів?
43. Як UTF-8 може бути використаний для приховування шкідливих даних у текстових файлах?
44. Як правильна інтерпретація UTF-8 може допомогти уникнути втрати важливої інформації при аналізі цифрових доказів?
45. Як відрізнити текст, закодований у UTF-8, від текстів в інших кодуваннях під час аналізу?
46. Які ризики пов'язані з обробкою текстових файлів у UTF-8, коли система налаштована на інше кодування?
47. Як UTF-8 може бути використаний для виконання атак типу "Code Injection" або інших маніпуляцій з текстом?
48. Як зловмисники можуть використовувати некоректну інтерпретацію UTF-8 для приховування шкідливих дій у системі?
49. Що таке Base64, і як він використовується для кодування даних під час розслідування інцидентів?
50. Як Base64 може бути використаний для приховування шкідливого коду в текстових файлах або веб-запитах?
51. Як правильно декодувати текст, закодований у Base64, під час аналізу підозрілих файлів?

52. Як можна розпізнати шкідливі або приховані дані, закодовані у Base64, під час аналізу мережевого трафіку?
53. Які інструменти можуть бути використані для автоматичного декодування даних, закодованих у Base64, під час розслідування інцидентів?
54. Як зловмисники можуть використовувати Base64 для обфускації коду, і як це можна виявити під час аналізу?
55. Як виявити та розшифрувати приховані повідомлення або шкідливий код, який використовує Base64 у текстових або HTML-файлах?
56. Які ризики існують при використанні Base64 для кодування конфіденційних даних, і як їх виявити під час розслідування?
57. Як використання Base64 у командних рядках або скриптах може бути виявлене під час розслідування інцидентів?
58. Як Base64 може бути використаний для виконання шкідливих дій через веб-додатки, і як це можна виявити?
59. Як проблеми з сумісністю між US-ASCII, KOI-8, Windows-1251, UTF-8 та Base64 можуть вплинути на розслідування інцидентів?
60. Як зловмисники можуть використовувати слабкі сторони сумісності кодувань для проведення атак або приховування своїх дій?
61. Як можна виявити конфлікти між різними кодуваннями символів під час аналізу текстових файлів?
62. Як конвертація між різними кодуваннями може призвести до втрати даних або зміни їхньої інтерпретації під час розслідування?
63. Як налаштувати системи моніторингу для виявлення аномалій, пов'язаних із некоректним кодуванням символів?
64. Як забезпечити коректне відображення текстів із різними кодуваннями символів у журналах подій і звітах під час розслідування інцидентів?
65. Як впровадження підтримки кількох кодувань символів у системах може підвищити ефективність розслідування інцидентів?

66. Як відрізнити текст, закодований у Base64, від текстів, закодованих у інших форматах, під час аналізу доказів?
67. Як автоматизувати процес конвертації текстових даних між різними кодуваннями символів під час аналізу інцидентів?
68. Як проблеми з кодуванням символів можуть вплинути на роботу з міжнародними даними або системами, що підтримують кілька мов?
69. Таблиця кодування символів формату US-ASCII.
70. Таблиця кодування символів формату KOI-8.
71. Таблиця кодування символів формату Windows-1251.
72. Таблиця кодування символів формату KOI8-U.
73. Таблиця кодування символів формату UTF-8.
74. Таблиця кодування символів формату UTF-16LE.
75. Дайте визначення та опис терміну rc4.
76. Дайте визначення та опис терміну base64.
77. Дайте визначення та опис терміну byte-xor.
78. З якими типами строк працює FLARE?

ПРАКТИЧНЕ ЗАВДАННЯ №9

ТЕМА: ФОРМАТ БІНАРНИХ ВИКОНУВАНИХ ФАЙЛІВ, ОБ'ЄКТНОГО КОДУ ТА ДИНАМІЧНИХ БІБЛІОТЕК (DLL)

МЕТА: Отримати практичні навички використання методів та інструментів статичного аналізу даних

ЗНАТИ: Основи програмування в ОС Windows.

ВМІТИ: Застосовувати інструментарій статичного аналізу даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ

PE Format - це формат файлів, для всіх 32- і 64-розрядних Windows систем. На даний момент існує два формати PE-файлів: PE32 та PE32+. PE32 – формат для x86 систем, а PE32+ для x64.

Portable Executable (PE) – це формат виконуваних бінарних файлів, об'єктного коду та динамічних бібліотек (DLL), який застосовується в 32- та 64-розрядних версіях операційних систем Microsoft Windows.

PE представляє собою структуру даних, що містить усю необхідну інформацію для завантажувача PE, щоб відобразити файл у пам'яті. Виконуваний код включає посилання для динамічного зв'язування бібліотек, таблиці експорту та імпорту API-функцій, дані для управління ресурсами та локальну пам'ять потоків (TLS).

У Windows NT формат PE використовується для файлів EXE, DLL, SYS (драйверів пристроїв) та інших виконуваних файлів.

Платформа .NET корпорації Microsoft розширила формат PE за допомогою функцій, що підтримують загальномовне середовище виконання (Common Language Runtime - CLR). Серед додатків – заголовок CLR та секція даних CLR. Після завантаження двійкового файлу завантажувач ОС призводить до виконання CLR через посилання у таблиці імпорту PE/COFF. Потім CLR завантажує заголовок CLR та секції даних.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій.

Документація формату:

1. (microsoft.com) PE Format

<https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

2. Заголовний файл winnt.h (Win source):

<https://source.winehq.org/source/include/winnt.h>

3. Structure of a Portable Executable 32 bit

https://upload.wikimedia.org/wikipedia/commons/1/1b/Portable_Executable_32_bit_Structure_in_SVG_fixed.svg

Пропонується використовувати інструментарій:

1. PeStudio

<https://www.winator.com/download>

2. CFF Explorer

https://ntcore.com/?page_id=388

3. PEiD Tool

<https://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/PEiD-updated.shtml>

4. Resource Hacker

<http://www.angusj.com/resourcehacker/>

5. (off/on-line) HEX EDITOR

<https://hexed.it/>

Пропонується переглянути додаткові мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

1. PE file format part1 - DOS Headers, Signature, File Header

<https://www.youtube.com/watch?v=bZ->

[swh1S01A&list=PLCLxMnnAnGilosKSzfg0LCdKZSmkdbDI&ab_channel=Tech69](https://www.youtube.com/watch?v=bZ-swh1S01A&list=PLCLxMnnAnGilosKSzfg0LCdKZSmkdbDI&ab_channel=Tech69)

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажити файл відповідно варіанту (1.exe-40.exe) та бібліотеку MalwareNetCard.dll.

Посилання: <https://drive.google.com/file/d/1J2b3Mnk8ZiGUKSoOUrOTfpln5woYgu0G/view?usp=sharing>

(Основне завдання)

- 1.Виправте формат файлу.
- 2.Визначте компілятор.
- 3.Визначте оригінальне ім'я файлу.
- 4.Визначте та проаналізуйте бібліотеки що використовуються та експортуються.
- 5.Визначте та проаналізуйте строкові ресурси файлу та визначте незвичні HTTP та FTP посилання.
- 6.Визначте час та дату компіляції.
- 7.Визначте зайву секцію файлу. Відокремте та визначте її формат секції з подальшим збереженням та проведенням відповідних дій.
- 8.Знайдіть у секції унікальний ключ у форматі «LBXXXXXX».

(Додаткове завдання)

- 1.Створити власну схему та розказати повну структуру PE Format.

Звіт виконання практичного завдання повинен містити:

– Титульний лист.

- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання, з наступними відповідями: формат файлу; назва компілятора; оригінальне ім'я файлу; список бібліотеки що використовуються та експортуються; строкові ресурси файлу (НТТР, FТР); час і дата компіляції, а також унікальний ключ.
- Звіт додаткового завдання – **створена власна схема PE Format** з описами розділів та секцій.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке Portable Executable (PE) формат? Яке його основне призначення в операційній системі Windows?
2. Що таке заголовок PE-файлу (PE Header), і яку інформацію він містить?
3. Яка роль таблиці імпорту (Import Table) у PE-файлі? Як вона може бути використана у розслідуванні інцидентів?
4. Що таке таблиця експорту (Export Table) у PE-файлі, і як вона може бути проаналізована під час розслідування?
5. Як можна використовувати інформацію з таблиці завантаження конфігурації (Load Configuration Table) під час аналізу шкідливих програм?
6. Що таке секція "Entry Point" у PE-файлі, і яку роль вона відіграє у виконанні програми?
7. Як відрізнити легітимний PE-файл від шкідливого за допомогою аналізу його структури?
8. Як аномалії у структурі PE-файлу можуть вказувати на модифікації або шкідливу активність?

9. Що таке цифровий підпис у контексті РЕ-файлів? Як перевірка підпису може допомогти в аналізі підозрілих файлів?
10. Як можна використовувати РЕ-файли для приховування шкідливого коду або виконання атак на цілісність системи?
11. Як використовувати інструменти для аналізу РЕ-файлів, щоб виявити ознаки шкідливих дій?
12. Які методи використовуються для упаковки РЕ-файлів, і як це може вплинути на їх аналіз?
13. Як аналіз рідкісних або незвичних секцій РЕ-файлу може допомогти у виявленні шкідливих програм?
14. Як шкідливі програми можуть використовувати техніку "підробленого" РЕ-файлу для обходу антивірусних засобів?
15. Які стандартні секції містяться у РЕ-файлах? Наведіть приклади таких секцій.
16. Що таке секція .text, і яку роль вона відіграє у виконанні РЕ-файлу?
17. Що таке секція .data, і які дані зазвичай у ній зберігаються?
18. Що таке секція .rdata, і як її вміст може бути проаналізований під час розслідування інциденту?
19. Що таке секція .bss, і яка її роль у РЕ-файлі?
20. Як аналіз секції .rsrc може допомогти у виявленні шкідливих ресурсів, прихованих у РЕ-файлі?
21. Що таке секція .reloc, і як вона використовується у процесі завантаження РЕ-файлу?
22. Які нестандартні або спеціалізовані секції можуть бути присутні у РЕ-файлах, і як їх аналізувати?
23. Як можуть бути використані додаткові секції у РЕ-файлах для зберігання шкідливих даних або кодів?
24. Як аналіз секцій може допомогти у виявленні технік обфускації або приховування коду в шкідливих програмах?

25. Як аналіз перерозподілу секцій (section alignment) у РЕ-файлах може вказувати на модифікації або зловживання?
26. Як можна виявити зміну стандартних секцій РЕ-файлів, що можуть вказувати на компрометацію?
27. Що таке PeStudio, і як він використовується для аналізу РЕ-файлів у контексті DFIR?
28. Які основні функції надає PeStudio для аналізу підозрілих РЕ-файлів?
29. Як PeStudio може допомогти у виявленні підозрілих API-викликів у РЕ-файлі?
30. Як за допомогою PeStudio можна виявити ознаки шкідливої активності в РЕ-файлах?
31. Як PeStudio допомагає у перевірці наявності цифрового підпису у РЕ-файлах?
32. Які додаткові індикатори ризику може виявити PeStudio під час аналізу РЕ-файлів?
33. Як PeStudio може допомогти в аналізі пакетованих або упакованих РЕ-файлів?
34. Як можна використовувати PeStudio для автоматизації аналізу великої кількості РЕ-файлів?
35. Що таке CFF Explorer, і як він може бути використаний для аналізу РЕ-файлів у контексті DFIR?
36. Які основні функції надає CFF Explorer для дослідження структури РЕ-файлів?
37. Як за допомогою CFF Explorer можна аналізувати таблиці імпорту та експорту в РЕ-файлах?
38. Як CFF Explorer може допомогти у виявленні та аналізі незвичних або підозрілих секцій у РЕ-файлі?
39. Як CFF Explorer використовується для перевірки заголовків РЕ-файлів?

40. Які можливості CFF Explorer надає для редагування та модифікації PE-файлів?
41. Як CFF Explorer може допомогти у виявленні ознак компрометації PE-файлу?
42. Що таке PEiD Tool, і яке його основне призначення у контексті DFIR?
43. Як PEiD Tool допомагає у визначенні упаковок та компресорів, що використовуються у PE-файлах?
44. Як PEiD Tool може бути використаний для виявлення обфускованих або захищених PE-файлів?
45. Які індикатори PEiD Tool використовує для визначення типу упаковки або шифрування у PE-файлах?
46. Як PEiD Tool може допомогти у визначенні шкідливих програм, що використовують нестандартні упаковок?
47. Як можна використовувати результати аналізу PEiD Tool для подальшого розслідування шкідливої активності?
48. Що таке Resource Hacker, і як він використовується для аналізу ресурсів PE-файлів у контексті DFIR?
49. Як Resource Hacker допомагає виявляти приховані ресурси, такі як зображення, іконки або тексти, у PE-файлах?
50. Як за допомогою Resource Hacker можна аналізувати та редагувати секцію .rsrc у PE-файлах?
51. Як Resource Hacker може бути використаний для виявлення шкідливих ресурсів, вбудованих у PE-файл?
52. Як шкідливі програми можуть використовувати ресурси PE-файлів для приховування свого коду або даних?
53. Як Resource Hacker може допомогти у створенні підроблених або модифікованих PE-файлів для тестування засобів безпеки?
54. Що таке HEX редактор, і яке його основне призначення у контексті аналізу PE-файлів?

55. Як HEX редактор може бути використаний для детального аналізу вмісту PE-файлу на байтовому рівні?
56. Як за допомогою HEX редактора можна виявити зміни у структурі PE-файлу, які вказують на шкідливу активність?
57. Як HEX редактор може допомогти у виявленні прихованих даних або шкідливого коду в PE-файлах?
58. Як за допомогою HEX редактора можна досліджувати підозрілі секції або незвичні патерни в PE-файлах?
59. Як HEX редактори можуть бути використані для ручного виправлення або модифікації PE-файлів під час розслідування інцидентів?

ПРАКТИЧНЕ ЗАВДАННЯ №10

ТЕМА: ІНСТРУМЕНТИ АНАЛІЗУ ПОВЕДІНКИ ПІДСИСТЕМ ОС.

МЕТА: Отримати практичні навички використання методів та інструментів динамічного аналізу даних.

ЗНАТИ: Основи програмування в ОС Windows.

ВМІТИ: Застосовувати інструментарій динамічного аналізу даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ

На відміну від статичного аналізу зловмисного програмного забезпечення, динамічний аналіз шкідливого програмного забезпечення виконується шляхом аналізу коду під час його роботи. Щоб вивчити поведінку виконуваного файлу, рекомендується працювати з віртуальними середовищами по типу Oracle VirtualBox.

Щоб зрозуміти функціональність зловмисного програмного забезпечення та запобігти його розповсюдженню, реверсивні інженери використовують налаштування під час розширеного динамічного аналізу зловмисного програмного забезпечення.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій.

Пропонується використовувати інструментарій:

1. Process Monitor (Sysinternals)

<https://learn.microsoft.com/en-us/sysinternals/downloads/procmon>

2. ProcDOT

<https://www.procdot.com/downloadprocdotbinaries.htm>

3. Process Monitor plus ProcDOT install, CSV export

<https://www.aldeid.com/wiki/ProcDOT>

4. Process hacker

<https://processhacker.sourceforge.io/downloads.php>

Пропонується переглянути додаткові мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

1. Visual Analysis with ProcDOT

https://www.youtube.com/watch?v=KRctlgDTJz4&ab_channel=13Cubed

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажте файл «mainlab.exe» та допоміжний файл «1.dat». Виконати файл у відповідності до варіанту та проаналізувати результат.

Посилання: <https://drive.google.com/file/d/1vimgYORP2jM5FW-dUukU1JpXyumn0bw/view?usp=sharing>

(Основне завдання)

- 1.Інсталювати та налаштувати обраний інструментарій.
- 2.Запустити файл “mainlab.exe”, обрати у випадному списку варіант, натиснути кнопку «Go!».
- 3.Проаналізувати потокові зміни, з отриманих даних виділити унікальний ключ у форматі «XXX-XXX-XXX-XXX-XXX».

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.

- Звіт основного завдання, з схемою сформованою ProcDOT та виділеним унікальним ключем у форматі «XXX-XXX-XXX-XXX-XXX».
- Звіт додаткового завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте визначення threads processes.
2. Дайте визначення Process ID (PID).
3. Дайте визначення threads ID.
4. Дайте визначення Parent Process (PPID).
5. Дайте визначення session id.
6. Дайте визначення authentication id.
7. Дайте визначення thread duration.
8. Дайте визначення processes duration.
9. Що таке Sysinternals Process Monitor, і яке його основне призначення у контексті DFIR?
10. Як Process Monitor відслідковує та реєструє активність процесів, реєстру та файлової системи в операційній системі Windows?
11. Які типи подій можна відслідковувати за допомогою Process Monitor? Як ці дані можуть бути корисними у розслідуванні інцидентів?
12. Як можна фільтрувати та сортувати дані, зібрані Process Monitor, для виявлення підозрілих дій?
13. Як Process Monitor може допомогти в ідентифікації шкідливих процесів або змін у реєстрі під час розслідування?
14. Як використовувати Process Monitor для аналізу файлів журналів (log files) з метою виявлення аномальної активності?

15. Які дані про процеси та їхні дії зберігаються у файлі журналу Process Monitor?
16. Як Process Monitor може бути використаний для відстеження створення, модифікації та видалення файлів у системі?
17. Як відстежувати завантаження та виконання динамічних бібліотек (DLL) за допомогою Process Monitor?
18. Як Process Monitor може допомогти у виявленні програм-шпигунів або іншого шкідливого ПЗ, яке вносить зміни до системи?
19. Як аналізувати тривалість процесів та подій, зафіксованих Process Monitor, для виявлення потенційних загроз?
20. Як зберігати та експортувати журнали Process Monitor для подальшого аналізу або використання в інших інструментах DFIR?
21. Що таке ProcDOT, і яке його основне призначення у контексті DFIR?
22. Як ProcDOT інтегрується з Process Monitor для візуалізації та аналізу зв'язків між процесами?
23. Як використовувати ProcDOT для виявлення шкідливих процесів та взаємозв'язків між ними під час розслідування інциденту?
24. Як ProcDOT допомагає у візуалізації графів подій, зібраних Process Monitor, та як це може бути корисним у розслідуванні?
25. Які типи подій можуть бути візуалізовані за допомогою ProcDOT, і як це допомагає у виявленні аномальної активності?
26. Як ProcDOT може бути використаний для виявлення та аналізу ланцюжків шкідливих дій у системі?
27. Як ProcDOT допомагає у відстеженні виконання команд, створення процесів та взаємодії між ними?
28. Як налаштувати ProcDOT для оптимальної роботи з великими журналами подій?
29. Як можна використовувати ProcDOT для створення звітів та діаграм, які можуть бути використані в судових процесах?

30. Як результати, отримані за допомогою ProcDOT, можуть бути інтегровані в інші інструменти та системи DFIR?
31. Що таке Process Hacker, і яке його основне призначення у контексті DFIR?
32. Як Process Hacker відрізняється від інших інструментів моніторингу процесів, таких як Process Monitor?
33. Які основні функції надає Process Hacker для моніторингу та управління процесами в операційній системі Windows?
34. Як за допомогою Process Hacker можна виявити підозрілі процеси, які приховують свою активність?
35. Як Process Hacker допомагає у виявленні та зупинці шкідливих процесів у реальному часі?
36. Як аналізувати створення та завершення процесів за допомогою Process Hacker?
37. Як Process Hacker може бути використаний для аналізу та зміни прав доступу до процесів та їхніх потоків?
38. Як Process Hacker допомагає у відстеженні мережевих з'єднань, пов'язаних із конкретними процесами?
39. Як за допомогою Process Hacker можна здійснювати відновлення інформації про завершені процеси?
40. Як Process Hacker може бути використаний для аналізу споживання ресурсів системи процесами, що працюють?
41. Як шкідливі програми можуть уникати виявлення Process Hacker, і як це можна виявити?
42. Що таке "Processes Duration", і як це поняття використовується у контексті DFIR?
43. Як аналіз тривалості процесів ("Processes Duration") може допомогти у виявленні аномальних або підозрілих дій у системі?
44. Що таке "Thread Duration", і яка його роль у аналізі роботи процесів?

45. Як можна використовувати "Thread Duration" для виявлення шкідливих потоків, що працюють надто довго або надто коротко?
46. Як аналіз тривалості потоків ("Thread Duration") може вказувати на використання системи зловмисниками для виконання шкідливих дій?
47. Що таке "Authentication ID", і як ця інформація може бути корисною під час розслідування інцидентів?
48. Як за допомогою "Authentication ID" можна визначити, який користувач або процес виконав конкретну дію у системі?
49. Як аналіз "Authentication ID" може допомогти у виявленні несанкціонованих доступів до системи?
50. Як інформація про Processes Duration та Thread Duration може використовуватися для побудови часових ліній подій під час розслідування інциденту?
51. Як аналіз "Authentication ID" може допомогти у відстеженні дій зловмисників, які використовували компрометовані облікові записи?
52. Як можна корелювати дані про "Processes Duration", "Thread Duration" та "Authentication ID" для побудови повної картини інциденту?
53. Як аналіз тривалості роботи процесів і потоків допомагає виявляти відмови в обслуговуванні DoS або інші атаки, спрямовані на перевантаження системи?
54. Яким чином інформація про Authentication ID може використовуватися для ідентифікації внутрішніх атак, коли зловмисник застосовує облікові записи користувачів організації?
55. Як інструменти Process Monitor, Process Hacker та ProcDOT допомагають аналізувати дані про тривалість виконання процесів і потоків, а також досліджувати події, пов'язані з автентифікацією?
56. Як показники тривалості роботи процесів і потоків можуть застосовуватися для виявлення системних аномалій, зокрема шкідливих програм, які працюють у фоновому режимі?

57. Як розуміння "Processes Duration" та "Thread Duration" може допомогти у створенні ефективних політик реагування на інциденти?

58. Як можна автоматизувати збір і аналіз даних про тривалість процесів та потоків для покращення ефективності реагування на інциденти?

ПРАКТИЧНЕ ЗАВДАННЯ №11

ТЕМА: АНАЛІЗ ДАМПІВ ПАМ'ЯТІ ОПЕРАЦІЙНИХ СИСТЕМ

МЕТА: ОТРИМАТИ ПРАКТИЧНІ НАВЧКИ ВИКОРИСТАННЯ ІНСТРУМЕНТІВ АНАЛІЗУ ПАМ'ЯТІ ОПЕРАЦІЙНИХ СИСТЕМ.

ЗНАТИ: Основи програмування в ОС Windows/Linux.

ВМІТИ: Застосовувати інструментарій Memory Forensics Framework.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Volatility – це фреймворк криміналістики пам'яті з відкритим кодом для реагування на інциденти та аналізу зловмисного програмного забезпечення [13-15,71]. Він написаний на Python і підтримує Microsoft Windows, Mac OS X і Linux.

Volatility – це найпоширеніша у світі структура для вилучення цифрових артефактів із вибірок енергонезалежної пам'яті (RAM). Методи вилучення виконуються повністю незалежно від системи, що досліджується, але забезпечують видимість стану виконання системи.

Volatility підтримує різноманітні зразки форматів файлів і можливість конвертувати між цими форматами:

1. Raw/Padded Physical Memory.
2. Firewire (IEEE 1394).
3. Expert Witness (EWF).
4. 32- and 64-bit Windows Crash Dump.
5. 32- and 64-bit Windows Hibernation (from Windows 7 or earlier).

6. 32- and 64-bit Mach-O files.
7. Virtualbox Core Dumps.
8. VMware Saved State (.vmss) and Snapshot (.vmsn).
9. HPAK Format (FastDump).
10. QEMU memory dumps.
11. LiME format.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій відповідно до обраної операційної системи.

Пропонується використовувати інструментарій:

1. Volatility Workbench виключно версії 3.0

<https://github.com/volatilityfoundation/volatility3>

Пропонується переглянути додаткові текстові та мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

2. Introduction to Memory Forensics with Volatility 3

https://www.youtube.com/watch?v=Uk3DEgY5Ue8&ab_channel=DFIRScience

3. Digital Forensics and Incident Response

<https://www.crowdstrike.com/cybersecurity-101/digital-forensics-and-incident-response-dfir/>

4. Indicator of Compromise

<https://encyclopedia.kaspersky.com/glossary/indicator-of-compromise-ioc/>

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажити файл у відповідності до варіанту (Folder 1-40 “Windows7even.dmp”).

Task LR 4 Dumps_1 (1-10)

<https://drive.google.com/file/d/1oEXxTL2H9j3ecIQwwQ8L3KzvUd3wpBbR/view?usp=sharing>

Task LR 4 Dumps_2 (11-20)

<https://drive.google.com/file/d/1zG9OVZfj0z8xSCwfcfLHU1JvNfIqvOXO/view?usp=sharing>

Task LR 4 Dumps_3 (21-30)

<https://drive.google.com/file/d/1K0bK9FADyEEAoLQKJK8VABjOZn4YOhXO/view?usp=sharing>

Task LR 4 Dumps_4 (31-40)

https://drive.google.com/file/d/1r00UeG_E-DWmoXBkylLaKetPKLrwtly_/view?usp=sharing

(Основне завдання)

- 1.Інсталювати та налаштувати обраний інструментарій.
- 2.Провести аналіз дампу пам'яті у відповідності до варіанту.
- 3.Визначте наступні параметри процесу «tttttttt.exe»:
 - Параметри командного рядка під час запуску процесу;
 - Які динамічні бібліотеки використовує процес;
 - Ключі реєстру які було створено процесом;
 - Які порти TCP/IP використовує процес;
 - Які файли було відкрито чи відчинено процесом.

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання, з визначеними відповідями.
- Звіт додаткового завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте визначення терміну File System Forensics.
2. Дайте визначення терміну Memory Forensics.
3. Дайте визначення терміну Network Forensics.
4. Дайте визначення терміну Log Analysis.

5. Що таке аналіз оперативної пам'яті, і чому він важливий у контексті цифрової криміналістики та реагування на інциденти?
6. Які основні методи використовуються для аналізу оперативної пам'яті в рамках розслідування інциденту?
7. Які види артефактів можна знайти в дампі оперативної пам'яті, і як вони можуть допомогти у розслідуванні?
8. Як виявляються шкідливі процеси або приховані потоки в оперативній пам'яті?
9. Як можна використовувати аналіз оперативної пам'яті для виявлення активних мережових з'єднань або відкритих портів?
10. Які інструменти найбільш популярні для аналізу дамів оперативної пам'яті, і які їхні основні можливості?
11. Як аналіз оперативної пам'яті може допомогти у виявленні шкідливого коду, який не залишає слідів на диску?
12. Що таке волатильні дані, і чому вони важливі для аналізу оперативної пам'яті?
13. Як можна виявити ознаки приховування шкідливих програм у пам'яті (наприклад, Rootkit)?
14. Як виявити використання ін'єкції коду у процеси за допомогою аналізу оперативної пам'яті?
15. Які обмеження має аналіз оперативної пам'яті, і як їх можна врахувати під час розслідування?
16. Що таке дам оперативної пам'яті, і які методи існують для його збору?
17. Які інструменти використовуються для створення дамів оперативної пам'яті на різних операційних системах (Windows, Linux, macOS)?
18. Як вибрати правильний інструмент для створення дампу оперативної пам'яті в конкретній ситуації?

19. Які кроки слід виконати перед збором дампу оперативної пам'яті, щоб зберегти цілісність даних?
20. Як забезпечити безпеку та конфіденційність даних під час збору дампу оперативної пам'яті?
21. Які ризики пов'язані зі збором дампу оперативної пам'яті, і як їх можна мінімізувати?
22. Як збір дампу оперативної пам'яті може вплинути на роботу системи або змінити її стан?
23. Що таке Live Response, і як він пов'язаний зі збором дампу оперативної пам'яті?
24. Як перевірити цілісність дампу оперативної пам'яті після його збору?
25. Які особливості збору дамків пам'яті з віртуальних машин або хмарних середовищ?
26. Як зберігати та передавати дампи оперативної пам'яті для подальшого аналізу?
27. Що таке Volatility Workbench, і яке його основне призначення в рамках DFIR?
28. Як Volatility Workbench відрізняється від командного рядка Volatility Framework?
29. Які основні функції надає Volatility Workbench для аналізу дамків оперативної пам'яті?
30. Як використовувати Volatility Workbench для виявлення підозрілих процесів у дампі пам'яті?
31. Як Volatility Workbench допомагає у відстеженні активних мережевих з'єднань під час аналізу пам'яті?
32. Як за допомогою Volatility Workbench можна виявити ін'єкції коду у процеси?
33. Як Volatility Workbench може бути використаний для аналізу модулів ядра та драйверів?

34. Як Volatility Workbench допомагає в ідентифікації активних сесій користувачів та їх дій у системі?
35. Як Volatility Workbench інтегрується з іншими інструментами для покращення ефективності аналізу?
36. Як Volatility Workbench може бути використаний для створення звітів, які підходять для судових розглядів?
37. Які основні модулі Volatility Workbench найчастіше використовуються у розслідуваннях, і які дані вони надають?
38. Що таке Indicator of Compromise (IoC), і яку роль він відіграє у реагуванні на інциденти?
39. Які основні типи IoC існують (хеші файлів, IP-адреси, домени, підозрілі файли тощо)?
40. Як збирати, зберігати та використовувати IoC у процесі розслідування інциденту?
41. Як можна автоматизувати виявлення IoC у корпоративних мережах?
42. Як використання IoC допомагає у виявленні та зупиненні активних кіберзагроз?
43. Як відрізняються тактичні, оперативні та стратегічні IoC, і як вони застосовуються в DFIR?
44. Які інструменти та платформи використовуються для обміну IoC між різними організаціями?
45. Як IoC можуть бути використані для зворотного інжинірингу шкідливого ПЗ?
46. Як аналіз IoC допомагає в ідентифікації методів, що використовуються зловмисниками під час атак?
47. Як відстежувати ефективність використання IoC у рамках стратегій захисту?
48. Які джерела можуть бути використані для отримання надійних IoC?

49. Що таке CERT-UA, і яку роль він відіграє у національній кібербезпеці України?
50. Які основні завдання та функції виконує CERT-UA під час реагування на кіберінциденти?
51. Як CERT-UA співпрацює з іншими національними та міжнародними організаціями з кібербезпеки?
52. Як організації можуть співпрацювати з CERT-UA для реагування на інциденти та обміну інформацією?
53. Як CERT-UA допомагає у розповсюдженні ІоС та інших даних про загрози в Україні?
54. Які типи інцидентів зазвичай обробляються CERT-UA, і які методи використовуються для їх розслідування?
55. Як CERT-UA може допомогти у навчанні та підготовці спеціалістів з кібербезпеки?
56. Як CERT-UA забезпечує інформування про нові кіберзагрози та атаки?
57. Як CERT-UA працює з приватним та державним секторами для зміцнення кібербезпеки в Україні?
58. Як інформація, надана CERT-UA, може бути використана для захисту організації від майбутніх атак?
59. Що таке CSIRT, і яке його основне призначення у контексті кібербезпеки?
60. Які основні функції виконує CSIRT під час розслідування та реагування на кіберінциденти?
61. Як CSIRT структурується у великих організаціях, і які ролі зазвичай виконуються у його складі?
62. Як CSIRT співпрацює з іншими командами, такими як CERT, SOC, і Red/Blue Teams?
63. Як CSIRT організовує процес реагування на інциденти від моменту виявлення до завершення?

64. Як CSIRT використовує інструменти для моніторингу, виявлення та аналізу інцидентів у реальному часі?
65. Як CSIRT забезпечує комунікацію та координацію з іншими організаціями та партнерами під час інцидентів?
66. Які виклики та ризики можуть виникати при роботі CSIRT, і як вони можуть бути подолані?
67. Як CSIRT забезпечує збір та збереження цифрових доказів для подальшого використання у судових розглядах?
68. Як CSIRT планує та проводить навчання з реагування на інциденти для підвищення готовності організації?
69. Що таке SOAR, і яке його призначення у контексті цифрової криміналістики та реагування на інциденти?
70. Як SOAR допомагає автоматизувати процеси виявлення та реагування на кіберзагрози?
71. Які основні компоненти SOAR, і як вони інтегруються з існуючими системами безпеки?
72. Як SOAR може допомогти у скороченні часу реагування на інциденти?
73. Як SOAR використовується для автоматизації аналізу інцидентів та створення звітів?
74. Як SOAR може допомогти у координації дій між різними командами з кібербезпеки, такими як SOC, CSIRT, та CERT?
75. Які типові сценарії автоматизації можна реалізувати за допомогою SOAR?
76. Як SOAR інтегрується з інструментами моніторингу та аналізу для забезпечення комплексного підходу до кібербезпеки?
77. Як SOAR може використовуватися для виявлення та реагування на атаки, що проводяться за допомогою автоматизованих ботів або скриптів?

ПРАКТИЧНЕ ЗАВДАННЯ №12

ТЕМА: УМОВИ ВИКОРИСТАННЯ ДИЗАСЕМБЛЕРІВ.

МЕТА: Отримати практичні навички використання дизасемблерів для зворотного аналізу програмного забезпечення

ЗНАТИ: Основи програмування в ОС Windows.

ВМІТИ: Застосовувати інструментарій декомпіляції.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Дизасемблери є дуже корисним інструментом для дослідження скомпільованого виконуваного бінарного файлу та надання загальної картини розуміння того, що він робить.

Зворотна розробка, або reverse engineering, є процесом дослідження програмного забезпечення з метою з'ясування принципів його роботи, внутрішньої структури та логіки виконання. Найчастіше цей підхід застосовують для аналізу вже створеного програмного продукту, щоб зрозуміти його функціональність без прямого копіювання реалізації. Особливе значення зворотна розробка має під час дослідження шкідливого програмного забезпечення, коли потрібно визначити алгоритм його роботи, поведінку в системі та можливі наслідки виконання.

Виконувані файли містять машинний код у вигляді двійкових даних. Дизасемблери переводять машинний код на більш зручний асемблерний код чи на декілька мов одночасно (наприклад C та Asm одночасно).

Дизасемблер (disassembler) – комп'ютерна програма що трансліює мову машинних кодів.

Дизасемблер не є тим самим, що декомпілятор. Декомпілятор перетворює машинний код у представлення мовою високого рівня, тоді як дизасемблер відображає програмний код у вигляді інструкцій асемблера.

Вивід дизасемблера – дизасембльований код – зазвичай подається у відформатованому вигляді, що робить дизасемблер зручним інструментом для зворотної розробки. Його використання дозволяє розібратися з деталями функціонування коду, провести певну оптимізацію окремих критичних ділянок коду, а також може використовуватися для усунення вбудованого захисту тобто злому.

Дизасемблери можуть просто подавати код у вигляді асемблерного коду, а можуть бути інтерактивними.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій відповідно до обраної операційної системи .

Пропонується використовувати інструментарій:

Ghidra (National Security Agency)

<https://github.com/NationalSecurityAgency/ghidra/releases>

Пропонується переглянути додаткові мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

Malware Analysis With Ghidra (HackerSploit)

https://www.youtube.com/watch?v=TJhfnItRVOA&list=PLBf0hzazHTGMSlOI2HZGc08ePwut6A2Io&index=17&ab_channel=HackerSploit

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажити файл у відповідності до варіанту (1.exe-40.exe) та отримати результат.

Посилання:

<https://drive.google.com/file/d/1O4u43I5Rq2ITT9xkTMA47hYLaG8tU0Wr/view?usp=sharing>

(Основне завдання)

- 1.Інсталювати та налаштувати обраний інструментарій.
- 2.Провести дизасемблювання виконуючого бінарного файлу у відповідності до варіанту (1.exe-40.exe).
- 3.Проаналізувати дизасембльований код та отримати унікальний ключ у форматі «XXX» чи «XX».

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання, з отриманим унікальним ключем у форматі «XXX» чи «XX» та вставками дизасембльованого коду підтверджуючими проведену роботу.
- Звіт додаткового завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте визначення ASM команді безумовної передачі керування JMP.
2. Дайте визначення ASM команді умовної передачі керування JZ/JE.
3. Дайте визначення ASM команді умовної передачі керування JNZ/JNE.
4. Дайте визначення ASM команді умовної передачі керування JC/JNAE/JB.
5. Дайте визначення ASM команді умовної передачі керування JNC/JAE/JNB.
6. Які існують регістри загального призначення 8/32/64 розрядні?
7. Як зберігаються змінні у мові ASM?

8. Що таке Malware Reverse Engineering, і чому він важливий у контексті цифрової криміналістики та реагування на інциденти?
9. Які основні етапи реверс-інжинірингу шкідливого програмного забезпечення?
10. Які інструменти та техніки використовуються для реверс-інжинірингу шкідливих програм?
11. Як визначити, який підхід (статичний або динамічний аналіз) краще підходить для аналізу конкретного шкідливого ПЗ?
12. Як аналіз шкідливого ПЗ допомагає виявити його основні функції та методи атаки?
13. Що таке обфускація, і як вона використовується шкідливими програмами для утруднення реверс-інжинірингу?
14. Як зловмисники використовують пакування та шифрування для захисту шкідливого коду від аналізу?
15. Які типові патерни або поведінкові ознаки можуть вказувати на наявність шкідливого ПЗ у системі?
16. Як можна використовувати реверс-інжиніринг для виявлення команд управління та контролю (C2) у шкідливих програмах?
17. Як створювати сигнатури (наприклад, YARA-правила) на основі результатів реверс-інжинірингу для виявлення аналогічного шкідливого ПЗ?
18. Які ризики пов'язані з виконанням реверс-інжинірингу шкідливих програм, і як їх можна мінімізувати?
19. Як розпізнати та знешкодити пастки (anti-debugging, anti-VM), які використовуються шкідливими програмами для ускладнення аналізу?
20. Як використовувати результати реверс-інжинірингу для поліпшення захисних заходів та систем виявлення загроз у мережі?
21. Що таке дизасемблер, і яку роль він відіграє у реверс-інжинірингу програмного забезпечення?
22. Які популярні дизасемблери використовуються для аналізу програм під Windows?

23. Як дизасемблери допомагають у розумінні роботи виконуваного файлу без його виконання?
24. Що таке дезасемблювання, і як воно відрізняється від деобфускації?
25. Які інструменти використовуються для дезасемблювання файлів формату PE (Portable Executable)?
26. Як дизасемблери можуть бути використані для виявлення та аналізу шкідливого коду, що впроваджується у легітимні програми?
27. Як дизасемблери можуть бути використані для аналізу патчів та оновлень програмного забезпечення з точки зору безпеки?
28. Що таке умовні переходи у дезасембльованому коді, і як вони можуть бути використані для аналізу поведінки шкідливого ПЗ?
29. Як дизасемблери можуть допомогти у відстеженні виконання коду та аналізі логіки шкідливої програми?
30. Які ризики можуть виникнути при неправильному використанні дизасемблерів під час аналізу програмного забезпечення?
31. Як використовувати дизасемблери для аналізу модулів ядра та драйверів у контексті розслідування кіберінциденту?
32. Як дизасемблери інтегруються з іншими інструментами для проведення комплексного аналізу шкідливих програм?
33. Як можна ідентифікувати шкідливий код у дезасембльованому вигляді серед легітимного коду?
34. Які є обмеження у використанні дизасемблерів для аналізу сучасного шкідливого ПЗ?
35. Що таке Ghidra, і яке його призначення в контексті реверс-інжинірингу?
36. Які основні функції та можливості надає Ghidra для аналізу програмного забезпечення?
37. Як Ghidra відрізняється від інших популярних інструментів для реверс-інжинірингу, таких як IDA Pro?

38. Як використовувати Ghidra для дезасемблювання та аналізу виконуваних файлів у форматі PE?
39. Які додаткові модулі або плагіни можна використовувати з Ghidra для розширення її функціональності?
40. Як Ghidra допомагає у відстеженні та аналізі динамічних викликів API у шкідливих програмах?
41. Як Ghidra забезпечує інтерактивний аналіз коду, і які переваги це надає у контексті реверс-інжинірингу?
42. Як за допомогою Ghidra можна аналізувати складні алгоритми, використовувані шкідливим ПЗ для шифрування або обфускації даних?
43. Як Ghidra допомагає у створенні та використанні скриптів для автоматизації реверс-інжинірингу?
44. Як можна використовувати Ghidra для дослідження небезпечних або підозрілих патчів у програмному забезпеченні?
45. Як Ghidra може бути використана для аналізу вбудованого програмного забезпечення (firmware) у рамках розслідування кіберінцидентів?
46. Як Ghidra підтримує колаборацію та спільний аналіз у командах з кібербезпеки?
47. Як результати, отримані за допомогою Ghidra, можуть бути використані для створення захисних заходів або сигнатур виявлення загроз?
48. Як можна використовувати Ghidra для зворотного інжинірингу шкідливих браузерних плагінів або додатків?
49. Які обмеження та виклики можуть виникнути під час використання Ghidra для аналізу сучасного шкідливого ПЗ?

ПРАКТИЧНЕ ЗАВДАННЯ №13

ТЕМА: УМОВИ ВИКОРИСТАННЯ ДЕКОМПІЛЯТОРІВ.

МЕТА: Отримати практичні навички використання декомпіляторів для зворотного аналізу програмного забезпечення

ЗНАТИ: Основи програмування в ОС Windows.

ВМІТИ: Застосовувати інструментарій декомпіляції.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Дизасемблери є дуже корисним інструментом для дослідження скомпільованого виконуваного бінарного файлу та надання загальної картини розуміння того, що він робить, проте ASM код важко аналізувати.

Декомпілятор, який також називають детранслятором або оберненим транслятором, є програмою, що перетворює модуль у вигляді двійкового машинного або об'єктного коду, сформованого компілятором, у **функціонально подібний** вихідний код мовою високого рівня, наприклад Python, Java, C або C#.

Процес перетворення машинного коду в вихідний код високого рівня називають декомпіляцією. Декомпілятори, як і дизасемблери, застосовуються для аналізу та зворотної розробки програмного забезпечення.

Комп'ютерна компіляція є незворотною в сенсі однозначного відновлення початкового коду: значна частина інформації при цьому втрачається. Тому декомпілятори використовують різні методи й алгоритми зворотного аналізу. Отриманий вихідний код зазвичай лише функціонально схожий на оригінальний, хоча деякі техніки дозволяють відновлювати код точніше.

Декомпіляція байт-коду мов, що виконуються на віртуальній машині (наприклад, Java або C#), зазвичай значно простіша, оскільки компілятори таких мов зберігають більше інформації, ніж компілятори, що генерують машинний код для C або C++.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій відповідно до обраної операційної системи.

Пропонується використовувати інструментарій:

dnspy

<https://github.com/dnSpy/dnSpy>

Пропонується переглянути додаткові мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

Malware Analysis With Amr Thabet (HackerSploit)

https://www.youtube.com/watch?v=ZKObRxxbOCQ&list=PLBf0hzazHTGMSlOI2HZGc08ePwut6A2Io&index=16&ab_channel=HackerSploit

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, завантажити файл у відповідності до варіанту (1.exe-40.exe) та отримати результат.

Посилання:

<https://drive.google.com/file/d/10qytO7mjOzur1Xpsh9cwpC4Ns-3qutaE/view?usp=sharing>

(Основне завдання)

1.Інсталювати та налаштувати обраний інструментарій (для роботи необхідно встановити dotnet sdk).

2.Провести декомпіляцію виконуваного файлу у відповідності до варіанту (1.exe-40.exe).

3.Проаналізувати декомпільований код та отримати унікальний ключ у форматі «XXXXXXXXXXXXXXXXXXXX».

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання, з отриманим унікальним ключем у форматі «XXXXXXXXXXXXXXXXXXXX» та вставками вихідного коду підтверджуючими проведену роботу.
- Звіт додаткового завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке .NET Framework, і яке його основне призначення?
2. Які основні компоненти складають .NET Framework?
3. Як .NET Framework забезпечує виконання коду, написаного на різних мовах програмування?
4. Що таке Common Language Runtime (CLR), і яку роль він відіграє у .NET Framework?
5. Що таке Intermediate Language (IL), і як він використовується у .NET Framework?
6. Як .NET Framework забезпечує безпеку коду та виконання через механізми, такі як Code Access Security (CAS)?
7. Які типи додатків можна розробляти за допомогою .NET Framework?
8. Як .NET Framework підтримує обробку винятків і забезпечує надійність коду?
9. Що таке Global Assembly Cache (GAC), і яку роль вона відіграє у .NET Framework?

10. Як .NET Framework забезпечує управління пам'яттю через механізми, такі як garbage collection?
11. Які є версії .NET Framework, і як вони відрізняються одна від одної?
12. Як можна використовувати .NET Framework для розробки багатоплатформних додатків за допомогою .NET Core та .NET 5/6?
13. Як аналіз коду на .NET Framework може допомогти у виявленні уразливостей та шкідливих дій у програмах?
14. Як зловмисники можуть використовувати специфіку .NET Framework для створення шкідливого ПЗ, і як це виявити?
15. Що таке реверс-інжиніринг програм, написаних на .NET, і чому він важливий у контексті DFIR?
16. Які інструменти використовуються для реверс-інжинірингу .NET програм?
17. Як можна декомпілювати .NET програму до її вихідного коду або близького до нього вигляду?
18. Що таке Intermediate Language (IL), і як його аналіз може допомогти у реверс-інжинірингу .NET програм?
19. Як можна використовувати метадані .NET збірок для аналізу їхньої структури та вмісту?
20. Як аналіз коду на рівні IL може допомогти у виявленні шкідливих інструкцій або прихованих функцій у .NET додатках?
21. Які методи обфускації використовуються для захисту .NET додатків від реверс-інжинірингу, і як їх можна обійти?
22. Як зловмисники можуть використовувати реверс-інжиніринг для модифікації легітимного .NET коду з метою його використання у шкідливих цілях?
23. Які ризики пов'язані з реверс-інжинірингом .NET програм, і як їх можна мінімізувати?
24. Як аналіз шкідливих .NET програм може допомогти у створенні захисних механізмів та сигнатур для їх виявлення?

25. Як можна автоматизувати процес реверс-інжинірингу .NET програм для швидшого аналізу шкідливих зразків?
26. Як використовувати результати реверс-інжинірингу .NET програм для зворотного інжинірингу інфраструктури командування і контролю (C2)?
27. Що таке ILSpy, і яке його основне призначення у контексті реверс-інжинірингу .NET програм?
28. Які основні функції та можливості надає ILSpy для аналізу .NET додатків?
29. Як ILSpy декомпілює .NET збірки до вихідного коду, і які обмеження має цей процес?
30. Як можна використовувати ILSpy для аналізу метаданих .NET збірок?
31. Які типи проєктів і збірок підтримує ILSpy для декомпіляції?
32. Як ILSpy може допомогти у виявленні прихованих або шкідливих функцій у .NET додатках?
33. Як ILSpy обробляє обфускацію коду, і які методи використовуються для її обходу?
34. Як можна використовувати результати декомпіляції ILSpy для створення YARA-правил або інших сигнатур для виявлення шкідливого ПЗ?
35. Як ILSpy інтегрується з іншими інструментами для проведення комплексного аналізу .NET програм?
36. Як ILSpy може бути використаний для навчання та тренування фахівців з кібербезпеки у реверс-інжинірингу .NET додатків?
37. Які ризики та виклики можуть виникнути при використанні ILSpy для аналізу сучасного шкідливого ПЗ?
38. Як можна автоматизувати процес декомпіляції за допомогою ILSpy для масового аналізу зразків?
39. Що таке dnSpy, і яке його основне призначення у контексті реверс-інжинірингу .NET програм?

40. Які основні функції та можливості надає dnSpy для аналізу та модифікації .NET додатків?
41. Як dnSpy використовується для декомпіляції та аналізу коду .NET збірок?
42. Як dnSpy може бути використаний для динамічного аналізу .NET програм, включаючи налагодження та трасування?
43. Які додаткові функції надає dnSpy для редагування та перезапису коду у .NET збірках?
44. Як можна використовувати dnSpy для виявлення та знешкодження шкідливих функцій у .NET додатках?
45. Як dnSpy обробляє обфускацію коду, і які методи використовуються для її обходу?
46. Як dnSpy може бути інтегрований з іншими інструментами для проведення комплексного аналізу шкідливих .NET програм?
47. Як dnSpy може допомогти у виявленні та аналізі ін'єкцій коду в .NET додатках?
48. Як можна використовувати dnSpy для модифікації та тестування захисних механізмів у .NET додатках?
49. Які ризики та виклики можуть виникнути при використанні dnSpy для аналізу та модифікації шкідливого ПЗ?
50. Як dnSpy може бути використаний для створення навчальних матеріалів та тренування фахівців з кібербезпеки у реверс-інжинірингу .NET додатків?
51. Як dnSpy допомагає у відстеженні змін у реальному часі під час виконання .NET додатків?
52. Які обмеження має dnSpy, і як їх можна подолати у контексті аналізу сучасного шкідливого ПЗ?

ПРАКТИЧНЕ ЗАВДАННЯ №14

ТЕМА: ІДЕНТИФІКАЦІЯ ТА КЛАСИФІКАЦІЯ ЗРАЗКІВ ЗЛОВМИСНОГО ПЗ

МЕТА: Отримати практичні навички реагування на інциденти та ліквідації наслідків.

ЗНАТИ: Основи програмування в ОС Windows/Linux.

ВМІТИ: Застосовувати інструментарій ідентифікації та класифікації.

ТЕОРЕТИЧНІ ВІДОМОСТІ

YARA – це інструмент, призначений (але не обмежуючись цим) допомогти дослідникам зловмисного ПЗ ідентифікувати та класифікувати зразки зловмисного ПЗ. За допомогою YARA можна створювати описи сімейств зловмисних програм (або будь-які інші, які ви хочете описати) на основі текстових або бінарних шаблонів. Кожен опис, або правило, складається з набору рядків і логічного виразу, які визначають його логіку [57,79,80].

Давайте подивимося на приклад:

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
```

```
$b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
$c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

condition:
    $a or $b or $c
}
```

Наведене вище правило повідомляє YARA, що будь-який файл, який містить один із трьох рядків, має бути позначено як `silent_banker`.

Це лише простий приклад, складніші та потужніші правила можна створити за допомогою символів підстановки, рядків без урахування регістру, регулярних виразів, спеціальних операторів та багатьох інших функцій, які можна знайти в документації YARA.

У зв'язку з великим обсягом інформації використовувати електронну документацію та інструментарій відповідно до обраної операційної системи.

Пропонується використовувати інструментарій:

YARA

<https://github.com/VirusTotal/yara>

Samples yara-rules

<https://github.com/godaddy/yara-rules/blob/master/example.yara>

YARA Rule Examples

<https://support.phishingtackle.com/hc/en-gb/articles/4410170814609-YARA-Rule-Examples>

YARA's documentation

<https://yara.readthedocs.io/en/stable/>

Yara Rules Project

<https://github.com/Yara-Rules>

Writing YARA rules

<https://yara.readthedocs.io/en/stable/writingrules.html?highlight=%24mz#accessing-data-at-a-given-position>

Пропонується переглянути додаткові мультимедіа матеріали (більш детально на сторінці курсу у Moodle):

Finding Evil with YARA

https://www.youtube.com/watch?v=mQ-mqxOfopk&ab_channel=13Cubed

What are Yara Rules (and How Cybersecurity Analysts Use Them)

https://www.youtube.com/watch?v=BM23_H2GGMA&ab_channel=GeraldAuser%2CPhD-SimplyCyber

ХІД ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ

Визначити свій індивідуальний варіант завдання, на основі виконаних практичних завдань 8-10, 12-13 завантажити бінарні файли у відповідності до варіанту та провести наступні маніпуляції.

(Основне завдання)

1. ЧІТКА HASH КЛАСИФІКАЦІЯ.

Створити сигнатури hash виконуваних бінарних файлів за допомогою криптографічних алгоритмів обчислення **MD5, SHA-1, SHA-256**.

2. НЕЧІТКЕ ХЕШУВАННЯ SSDEEP

Створити сигнатури виконуваних бінарних файлів за допомогою нечіткого хешування **SSDEEP**.

3. ХЕШ ІМПОРТОВАНИХ БІБЛІОТЕК IMPHASH

Створити сигнатури виконуваних бінарних файлів за допомогою нечіткого хешування **IMPHASH**.

4. YARA КЛАСИФІКАЦІЯ.

Створити правила YARA виконуваних бінарних файлів з обов'язковим використанням наступних правил:

1. \$ascii_string, \$unicode_string, \$hex_string (YARA string modifiers).
2. xor, filesize (YARA keywords).

3. dotnet.version, dotnet.assembly.name (YARA keywords dotnet).
4. pe.exports, pe.number_of_sections (YARA keywords PE module).

Звіт виконання практичного завдання повинен містити:

- Титульний лист.
- Тема та мета практичного завдання.
- Завдання до практичного завдання.
- Звіт основного завдання зі вставками HASH даних, YARA правил.
- Звіт додаткового завдання.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної практичного завдання.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте визначення терміну MISP-UA в розрізі ефективної бази даних ІОС.
2. Функціонал та призначення VirusTotal.
3. Для яких цілей використовується параметри «-C,-c, -d, -f,-h,-i,-l,-x,-n,-N,-w» командного рядка програми yara64.exe?
4. Для яких цілей використовується параметри «-m,-D,-e,-S,-s,-L,-g,-r,-z,-k,-t,-p,-a,-v» командного рядка програми yara64.exe?
5. Які існують основні регулярні вирази програми yara64.exe?
6. Які існують додаткові модулі розширення основної функціональності YARA?
7. В яких випадках використовують зовнішні змінні YARA?
8. Що таке криптографічні хеш-функції?
9. Що таке нечіткі криптографічні хеш-функції?
10. У яких випадках використовується хеш імпортованих бібліотек?
11. Що таке YARA, і яке її основне призначення у контексті DFIR?

12. Як YARA допомагає у виявленні та класифікації шкідливого програмного забезпечення?
13. Яка структура YARA-правила, і які його основні компоненти?
14. Що таке "розділ метаданих" у YARA-правилі, і яку інформацію він містить?
15. Які типи патернів можна визначати за допомогою YARA-правил (рядки, байти, регулярні вирази)?
16. Як YARA обробляє логічні вирази та оператори в правилах для створення більш точних умов?
17. Що таке "розділ умов" у YARA-правилі, і як він визначає умови спрацювання правила?
18. Як YARA-правила можуть бути використані для ідентифікації певних сімейств шкідливого ПЗ?
19. Як визначити ефективність YARA-правила у виявленні шкідливого коду?
20. Які обмеження має YARA у виявленні складних або сильно обфускованих шкідливих програм?
21. Як тестувати та налагоджувати YARA-правила для забезпечення їхньої точності та ефективності?
22. Як YARA інтегрується з іншими інструментами та платформами для автоматизованого виявлення загроз?
23. Як створити ефективне YARA-правило для виявлення нового шкідливого ПЗ?
24. Які найкращі практики існують для написання YARA-правил з мінімальним рівнем хибних спрацювань?
25. Як використовувати регулярні вирази у YARA-правилах для підвищення точності виявлення?
26. Як можна комбінувати кілька умов у YARA-правилі для створення складніших умов спрацювання?

27. Як використовувати оператори AND, OR, NOT у YARA-правилах для визначення логічних умов?
28. Як уникнути надмірного спрацювання YARA-правил на безпечних файлах або програмах?
29. Як використовувати "Wildcard" та "Jumps" у YARA-правилах для виявлення варіативних патернів?
30. Як налаштувати порогові значення для кількості збігів, необхідних для спрацювання YARA-правила?
31. Які типи атак можуть бути виявлені за допомогою добре налаштованих YARA-правил?
32. Як використовувати YARA для виявлення певних рядків або патернів у пам'яті або на диску?
33. Як YARA може бути використана для виявлення індикаторів компрометації (IoC) у файлах або мережевих потоках?
34. Як створювати YARA-правила, що враховують специфіку операційних систем або архітектур?
35. Як YARA може бути інтегрована у процеси реагування на інциденти та розслідування кіберзагроз?
36. Як використовувати YARA для аналізу файлів під час розслідування інциденту?
37. Як YARA може допомогти у виявленні шкідливих файлів або кодів у великих наборах даних?
38. Як YARA може бути використана для пошуку та аналізу шкідливих файлів на кінцевих пристроях або серверах?
39. Як використовувати YARA для автоматичного моніторингу систем на предмет підозрілої активності?
40. Як результати роботи YARA можуть бути використані для створення звітів та надання доказів під час розслідувань?
41. Як YARA може допомогти у виявленні нових варіантів відомого шкідливого ПЗ?

42. Як використовувати YARA для аналізу дамів пам'яті або знімків системи?
43. Як інтегрувати YARA з іншими інструментами DFIR, такими як Volatility, для комплексного аналізу інциденту?
44. Як YARA може бути використана для виявлення шкідливих конфігураційних файлів або скриптів?
45. Як можна використовувати YARA для автоматизації пошуку шкідливих файлів у мережевому трафіку?
46. Як впровадити YARA у корпоративну середу для автоматичного виявлення загроз?
47. Як налаштувати YARA для автоматичного сканування файлів на сервері або кінцевих пристроях?
48. Як інтегрувати YARA з SIEM-системами для автоматизованого аналізу інцидентів?
49. Як використовувати YARA у хмарних середовищах для виявлення шкідливих файлів та активності?
50. Як автоматизувати процес оновлення YARA-правил для забезпечення актуальності захисту?
51. Як оцінити ефективність впровадження YARA у процесах кібербезпеки?
52. Як використовувати YARA для створення базових ліній поведінки та виявлення аномалій?
53. Як впровадити YARA у процеси аналізу та обробки загроз у реальному часі?
54. Як використовувати YARA у поєднанні з SOAR (Security Orchestration, Automation, and Response) для автоматизації реагування на інциденти?
55. Як можна інтегрувати YARA з платформами обміну інформацією про загрози для розповсюдження правил виявлення?

56. Як оцінити якість та ефективність існуючих YARA-правил у виявленні шкідливого ПЗ?
57. Як аналізувати продуктивність YARA-сканувань для забезпечення ефективності та мінімізації навантаження на систему?
58. Як уникнути конфліктів або дублювання правил у великому наборі YARA-правил?
59. Як можна оптимізувати YARA-правила для швидшого виконання та меншої кількості помилкових спрацювань?
60. Як забезпечити актуальність YARA-правил у постійно змінюваному ландшафті загроз?
61. Як тестувати нові YARA-правила перед їх впровадженням у виробниче середовище?
62. Як аналізувати логи та результати спрацювання YARA-правил для покращення точності виявлення?
63. Як адаптувати YARA-правила до специфіки організації або галузі?
64. Як використовувати машинне навчання або інші сучасні технології для автоматичного покращення YARA-правил?
65. Як інтегрувати зворотний зв'язок від інцидентів у процес оптимізації YARA-правил?

ПРАКТИЧНЕ ЗАВДАННЯ №15

ТЕМА: АНАЛІЗ СТРУКТУРИ USB НАКОПИЧУВАЧА ТА ПОШУК ПРИХОВАНИХ ДАНИХ

МЕТА: Отримати практичні навички створення образу USB накопичувача, перевірки його цілісності, аналізу файлової структури та пошуку прихованих даних.

ЗНАТИ: Основи роботи з USB накопичувачами, файловими системами, образами дисків, хеш сумами та базовими інструментами аналізу даних.

ВМІТИ: Створювати образ USB накопичувача, перевіряти його хеш суму, переглядати файлову структуру, знаходити приховані дані та оформлювати результати аналізу у звіті.

ТЕОРЕТИЧНІ ВІДОМОСТІ

З огляду на значний обсяг довідкової інформації, здобувачам вищої освіти рекомендується використовувати електронну документацію до обраного програмного інструментарію, операційної системи та, за потреби, мови програмування. Вибір додаткових джерел і засобів виконання роботи доцільно узгоджувати відповідно до обраної операційної системи.

З урахуванням різноманітності операційних систем, технічних середовищ, програмних засобів і можливих підходів до виконання поставлених завдань допускається використання альтернативного інструментарію, рекомендаційний інструментарій вказано на кожному кроці окремо. Обрані засоби мають забезпечувати повне виконання завдань роботи, коректність

отриманих результатів та можливість їх підтвердження у звіті за допомогою скріншотів, текстових результатів або інших доказових матеріалів.

У роботі **виконується навчальне дослідження** USB накопичувача з метою аналізу його файлової структури, перевірки цілісності даних, пошуку прихованої інформації та фіксації результатів у звіті.

Це необхідно для формування базових навичок аналізу цифрових носіїв інформації, які використовуються під час розслідування інцидентів у сфері ІТ, відновлення даних, перевірки цілісності файлів та виявлення цифрових слідів.

Знання принципів аналізу USB накопичувачів допомагає **фахівцю з інформаційних технологій** розуміти особливості зберігання даних на фізичних носіях, роботу файлової системи, призначення службових областей, що забезпечують доступ до файлів, а також способи виявлення пошкоджених, видалених або прихованих об'єктів. Це формує практичне уявлення про зв'язок між логічним відображенням файлів в операційній системі та фактичним розміщенням даних у структурі носія.

Для прикладного програміста такі знання є корисними, оскільки вони допомагають глибше розуміти роботу з файлами, потоками байтів, кодуванням даних, базами даних, архівами та службовими форматами. Програміст отримує уявлення про те, як програмне забезпечення записує інформацію на носій, за яких умов можуть виникати помилки збереження, чому важливо перевіряти цілісність даних та як правильно обробляти файли різних типів. Це сприяє підвищенню якості програмного забезпечення, яке працює з локальними файлами, журналами, резервними копіями, конфігураційними файлами та користувацькими даними.

Для комп'ютерного інженера така підготовка є важливою, оскільки вона пов'язана з апаратно програмною організацією комп'ютерних систем. Аналіз образів накопичувачів дає змогу краще зрозуміти принципи роботи носіїв інформації, завантажувальних секторів, файлових систем, службових структур, часових міток та механізмів збереження даних. Це допомагає ефективніше діагностувати проблеми з накопичувачами, оцінювати стан даних, виконувати

технічне відновлення інформації та використовувати інструменти низькорівневого аналізу.

Для фахівця з кібербезпеки ці знання мають безпосереднє практичне значення, оскільки цифрові носії часто містять сліди дій користувачів, службові журнали, приховані файли, видалені дані або фрагменти інформації, що можуть бути важливими під час аналізу інцидентів. Уміння створювати образ носія, перевіряти його хеш суму, працювати з копією даних, аналізувати файлову структуру та документувати результати належить до базових складових цифрової криміналістики.

Отримані знання також розвивають уважність до технічних деталей, уміння працювати з доказовими даними, навички **аналізу нестандартних ситуацій** та здатність обґрунтовувати власні висновки. Це важливо для будь якого ІТ фахівця, оскільки робота з даними потребує не лише технічного виконання операцій, а й розуміння причин, наслідків та достовірності отриманих результатів.

Це важливо для будь якого ІТ фахівця, оскільки робота з даними потребує не лише технічного виконання операцій, а й розуміння причин, наслідків та достовірності отриманих результатів.

У результаті виконання роботи здобувач отримує практичне уявлення про те, як поєднуються програмні інструменти, структура файлової системи, методи пошуку даних та правила оформлення технічного звіту. Такі навички можуть застосовуватися під час адміністрування комп'ютерних систем, розроблення програмного забезпечення, аналізу інцидентів, відновлення інформації та забезпечення цілісності цифрових даних.

ХІД ВИКОНАННЯ РОБОТИ

ПРЕАМБУЛА

До Вас як майбутнього ІТ фахівця спеціаліста галузі інформаційні технології надійшло умовне завдання детально проаналізувати USB накопичувач для виявлення, аналізу та нейтралізації ІТ інцидентів.

У межах розслідування вилучено USB накопичувач, який імовірно належить зловмиснику. За попередньою інформацією, на носії можуть міститися приховані файли, пошкоджені службові області файлової системи, видалені дані, фрагменти зашифрованих повідомлень та цифрові сліди, необхідні для встановлення повної картини подій.

Результатом роботи є послідовне виявлення фрагментів секретного ключа мастер пароля, складання мастер пароля за встановленою маскою та отримання фінального прапора, який підтверджує успішне завершення розслідування.

Можливі приклади фрагментів ключа мастер пароля:

Фрагмент ключа 1 «D15K»;

Фрагмент ключа 2 «Macaroni»;

Фрагмент ключа 3 «88AAA88»;

Фрагмент ключа 4 «orange»;

Фрагмент ключа 5 «code»;

Фрагмент ключа 6 «Madonna»;

Фрагмент ключа 7 «Ciccone».

Можливий приклад складеного мастер пароля:

D15K-Macaroni-88AAA88-orange-code-Madonna-Ciccone

Можливий приклад шаблон маски мастер пароля:

«XXXX-XXXXXXXX-XXXXXXXX-XXXX-XXXX-XXXXXXXX-XXXX»

4-8-7-6-4-7-6.

Завдання має навчальний характер і спрямоване на формування практичних компетентностей з основ цифрової криміналістики, аналізу носіїв інформації, захисту даних та документування результатів технічного дослідження.

КРОК 1. ПІДГОТОВКА, СТВОРЕННЯ КОПІЇ USB НАКОПИЧУВАЧА.

1.1. Скачати точну побітову копію у вигляді файлу образу по посиланню

<https://drive.google.com/file/d/1gN-Mme0v4gRbhdgAG88gPeK9PgOlJdOM/view?usp=sharing>

Усі подальші дії виконувати виключно зі створеним образом.

1.2 **Розрахувати** хеш суму створеного образу за алгоритмом MD5 або SHA1 та **порівняти** її з еталонним значенням.

SHA1: c0bfde422ff4e20ce0dcaf7513024129603686fb

MD5: caf8cc5d1ca23755f0d01d831c6f7aa5

SHA256: a3b32ba174d6838a5954e05ef5db5ab3c93792c62957ffae4c11812b7870d4d6

У ЗВІТ: Назву файлу образу, його розмір, використаний алгоритм хешування, отриману хеш суму та результат звірки з еталонним значенням.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: FTK Imager (функція Create Disk Image).
- Linux: утиліти dd (для зняття копії), md5sum, sha1sum.

КРОК 2. НИЗЬКОРІВНЕВИЙ АНАЛІЗ ДИСКА ТА ВІДНОВЛЕННЯ ДОСТУПУ

2.1. **Спробувати** змонтувати створений образ штатними засобами операційної системи або за допомогою спеціалізованого програмного забезпечення.

2. Якщо під час монтування виникає повідомлення про невідому файлову систему, помилку читання або пропозиція відформатувати диск, не виконувати форматування. У такому випадку перейти до **аналізу** завантажувального сектора Boot Sector.

3. **Знайдіть** причину пошкодження (зверніть увагу на перші байти, зокрема на поле OEM ID за зміщенням 0x03). Всі знайдені нестандартні слова чи аномалії в цій області можуть бути першою частиною фрагменту ключа мастер пароля.

2.3. Відкрити робочу копію образу у hex редакторі та проаналізувати перші байти завантажувального сектора. Особливу увагу звернути на поле OEM ID, яке розміщується за зміщенням 0x03. Нестандартні слова, символи або аномальні значення у цій області можуть містити перший фрагмент секретного ключа мастер пароля.

2.4. Усуньте пошкодження вручну (виправте байти на правильну сигнатуру), щоб відновити читабельність файлової системи. Переконайтеся, що файлова система доступна для перегляду. Після відновлення доступу перейти до наступного етапу виконання роботи.

У ЗВІТ: Опис помилки, яка виникла під час першої спроби монтування образу; Підтвердження, що після виправлення файлова система відкривається; Скріншоти hex редактора до та після внесення змін; Зміщення, початкове значення байтів, текстову інтерпретацію знайдених символів та можливий фрагмент ключа мастер пароля. Дії які підтверджують проходження кроку включаючи скріншоти або текстові дані. Якщо підтверджувальні матеріали у звіті відсутні, відповідний крок вважається невиконаним і не зараховується.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: HxD (Hex Editor), Far manager.
- Linux: hexeditor, ghex, xxd.

КРОК 3. РОЗВІДКА ТА ПОШУК "КАРТИ" РОЗСЛІДУВАННЯ

3.1. Просканувати файлову систему на наявність усіх доступних логічних файлів, каталогів, службових об'єктів і видалених файлів.

3.2 Побудувати дерево файлової структури та визначити основні каталоги, які можуть належати профілю користувача, наприклад документи, нотатки, завантаження, тимчасові файли або службові папки.

3.3. Знайти основний текстовий документ профілю користувача.

3.4. Проаналізувати вміст знайденого документа (можливо, доведеться шукати прихований текст або читати файл у режимі "сирих" рядків). **Визначити маску**, за якою формується мастер пароль. Маска показує кількість фрагментів,

їхню послідовність і приблизну довжину кожної частини. Надалі вона використовується як карта розслідування для пошуку всіх складових пароля.

У ЗВІТ: Назву використаного програмного засобу для аналізу образу; Скріншот дерева файлової системи образу; Перелік виявлених логічних і видалених файлів, які мають значення для розслідування; Назву та шлях до основного текстового документа профілю користувача; Фрагмент вмісту документа або опис знайденої підказки; Виявлену маску мастер пароля; Скріншоти або текстові докази, які підтверджують правильність знайденої маски. Дії які підтверджують проходження кроку включаючи скріншоти або текстові дані. Якщо підтверджувальні матеріали у звіті відсутні, відповідний крок вважається невиконаним.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: FTK Imager (монтування образу як Evidence Item), Autopsy.
- Linux: Autopsy, The Sleuth Kit (утиліта fls для перегляду дерева директорій).

КРОК 4. АНАЛІЗ КОНФІГУРАЦІЙ ТА ДЕКОДУВАННЯ СМІТТЄВИХ ДАНИХ

4.1 Перевірити службові області диска, тимчасові та системні файли (.conf, .ini, .log, .tmp). Не забудьте перевірити вміст файлів, які ви ідентифікували як "видалені" на попередньому етапі.

4.2 Виконати пошук, підозрілих даних на ваш погляд (текстові рядки та ін). Звертайте увагу на параметри, що містять шістнадцятковий код (у редакторах зазвичай префікс "0x"), або інші значення, які виглядають незрозуміло з набором символів зі знаками «==» у кінці тексту.

4.3 Виявити дані, які можуть бути закодованими, **декодуйте** знайдені послідовності (з Hex та Base64), щоб отримати наступні фрагменти ключа мастер пароля.

У ЗВІТ: Дії, які підтверджують проходження кроку, включають скріншоти або текстові дані. Якщо підтвердних матеріалів у звіті немає, відповідний крок вважається невиконаним.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Універсальні: CyberChef (рецепти From Hex, From Base64).
- Linux: термінальні команди base64 -d, xxd -r -p, icat (для вилучення видалених файлів).

КРОК 5. ДОСЛІДЖЕННЯ ЛОКАЛЬНИХ БАЗ ДАНИХ

5.1. **Знайти** у файловій системі образу файли локальних баз даних (формат SQLite, часто використовується для історії браузера або системних логів). **Відкрити** знайдену базу даних та **проаналізувати** вміст таблиць і визначити ті, які можуть містити корисні записи.

5.2. **Знайти** нестандартні записи, що вказують на помилки або фрагментацію даних (наприклад, розбиті частини слова).

5.3. **Зібрати** знайдені частини записів, декодуйте їх (якщо потрібно) та склейте у єдиний фрагмент ключа мастер пароля.

У ЗВІТ: Дії, які підтверджують проходження кроку, включають скріншоти або текстові дані. Якщо підтвердних матеріалів у звіті немає, відповідний крок вважається невиконаним.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows / Linux: DB Browser for SQLite, консольна утиліта sqlite3.

КРОК 6. СТЕГАНОГРАФІЯ НА РІВНІ ФАЙЛОВОЇ СТРУКТУРИ

6.1. Переглянути файлову систему образу USB накопичувача та знайти всі графічні файли на наявність прихованої інформації.

6.2. **Проаналізувати** на наявність зайвих даних, що були навмисно дописані у файл поза його стандартним шаблонним мультимедіа форматуванням, наприклад зверніть увагу на такий маркер як IEND для PNG та знайдіть прихований текст.

У ЗВІТ: Дії, які підтверджують проходження кроку, включають скріншоти або текстові дані. Якщо підтвердних матеріалів у звіті немає, відповідний крок вважається невиконаним.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: HxD, утиліта Strings (від Microsoft Sysinternals).
- Linux: strings, Hex-редактори, zsteg.

КРОК 7. ЛОГІЧНІ АНОМАЛІЇ ТА АНАЛІЗ ЧАСОВОЇ ШКАЛИ

7.1. Побудувати часову шкалу (Timeline) усіх файлів на диску, відсортувавши їх за датою модифікації або створення.

7.2. Проаналізувати отриманий список файлів і знайти об'єкти з аномальними часовими мітками. Аномальними вважаються файли, дата яких суттєво відрізняється від інших, наприклад файл датований далеким майбутнім або минулим. Відкрити файл з аномальною часовою міткою та перевірити його вміст. Якщо файл містить зрозумілий текст, **зафіксувати** знайдений фрагмент як можливу частину фрагменту ключа мастер пароля.

7.3. Проаналізуйте сусідні бінарні дампи. Якщо вміст файлу виглядає як нечитабельне сміття і поруч є підказка про "інверсію пам'яті", застосуйте до цього файлу побітові логічні операції (наприклад, Bitwise NOT або XOR з 0xFF), щоб відновити початковий текст.

У ЗВІТ: Дії, які підтверджують проходження кроку, включають скріншоти або текстові дані. Якщо підтвердних матеріалів у звіті немає, відповідний крок вважається невиконаним.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: FTK Imager (сортування колонок Date Modified), CyberChef (рецепт Bitwise NOT).
- Linux: ls -l (перегляд дат), консольна команда `tr '\000-\377' '\377-\000'` (інверсія).

КРОК 8. File Carving та отримання фінального прапора

8.1. **Зібрати** всі знайдені на попередніх кроках фрагменти ключа в єдиний мастер пароль відповідно до маски, визначеної на кроці 3. **Перевірити** правильність послідовності фрагментів мастер пароля. Звернути увагу на кількість частин, розділювачі між ними, регістр літер і наявність цифр або спеціальних символів.

8.2. **Дослідити** нерозмічену область диска (Unallocated Space) або кінцеву частину файлу образу, яка може містити дані поза межами основної файлової системи.

8.3. **Виконайте** пошук "файлів усередині файлу" за їхніми сигнатурами (Magic Numbers). Знайдіть прихований ZIP-архів (сигнатура 50 4B 03 04).

8.4. **Отримайте** цей блок даних, збережіть як окремий архів та розпакуйте його за допомогою зібраного Мастер пароля. Знайдіть всередині секретний прапор (Flag), який засвідчить успішне виконання місії.

У ЗВІТ: Зафіксувати індивідуальний **МАСТЕР ПАРОЛЬ** у звіті роботи.

РЕКОМЕНДОВАНИЙ ІНСТРУМЕНТАРІЙ:

- Windows: HxD (пошук сигнатури, копіювання блоку даних), 7-Zip.
- Linux: binwalk -e, foremost, unzip

Повний звіт виконання роботи повинен містити:

- Титульний лист.
- Тема та мета роботи.
- Отриманий сформований індивідуальний мастер пароль.
- Скріншоти та інші матеріали на думку автора які підтверджують правильність виконаної роботи.
- Відповіді на контрольні питання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке побітова копія USB накопичувача та чому під час цифрової криміналістики подальший аналіз виконують саме з образом, а не з оригінальним носієм?
2. Для чого розраховують хеш суму образу диска та яку роль вона відіграє у підтвердженні цілісності цифрового доказу?
3. Що таке Boot Sector і чому його пошкодження може призвести до неможливості монтування файлової системи?
4. Яке призначення має поле OEM ID у завантажувальному секторі та чому аномальні значення в цій області можуть мати значення для розслідування?
5. У чому полягає різниця між логічними файлами, видаленими файлами та службовими об'єктами файлової системи?
6. Чому під час аналізу образу USB накопичувача важливо фіксувати не лише знайдені дані, а й шлях до файлу, зміщення, інструмент аналізу та скріншоти виконаних дій?
7. Які ознаки можуть вказувати на те, що текстові дані у файлах закодовані у форматі Hex або Base64?
8. У чому полягає різниця між кодуванням і шифруванням даних?
9. У чому полягає суть стеганографії на рівні файлової структури та як можна виявити приховані дані, дописані після завершального маркера файлу?
10. Що таке File Carving і як пошук сигнатур Magic Numbers допомагає виявити приховані файли у нерозміченій області диска?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. NIST Special Publication 800-61 Revision 3. Incident Response Recommendations and Considerations for Cybersecurity Risk Management. Gaithersburg : National Institute of Standards and Technology, 2025. URL: <https://csrc.nist.gov/pubs/sp/800/61/r3/final>.
2. NIST Special Publication 800-86. Guide to Integrating Forensic Techniques into Incident Response. Gaithersburg : National Institute of Standards and Technology, 2006. URL: <https://csrc.nist.gov/pubs/sp/800/86/final>.
3. M-Trends 2025 Report. Google Cloud, Mandiant, 2025. URL: <https://services.google.com/fh/files/misc/m-trends-2025-en.pdf>.
4. 2025 Data Breach Investigations Report. Verizon Business, 2025. URL: <https://www.verizon.com/business/resources/reports/dbir/>.
5. Microsoft Digital Defense Report 2025. Microsoft, 2025. URL: <https://www.microsoft.com/en-us/corporate-responsibility/cybersecurity/microsoft-digital-defense-report-2025/>.
6. CrowdStrike 2025 Global Threat Report: Executive Summary. CrowdStrike, 2025. URL: <https://www.crowdstrike.com/en-us/resources/reports/global-threat-report-executive-summary-2025/>.
7. IBM X-Force Threat Intelligence Index 2025. IBM, 2025. URL: <https://www.ibm.com/think/x-force/x-force-threat-intelligence-index-2025-attackers-steal-sell-user-identities>.
8. SANS 2025 Detection and Response Survey Webcast and Forum. SANS Institute, 2025. URL: <https://www.sans.org/webcasts/sans-2025-detection-and-response-survey-webcast-forum>.
9. DFIR Threat Report 2025. CyberCX, 2025. URL: <https://cybercx.com.au/resource/dfir-threat-report-2025/>.
10. Гавловський В. Д. Аналіз стану кіберзлочинності в Україні. Інформація і право. 2019. № 1 (28). С. 108–117.

11. Гуцалюк М. В. Сучасні тенденції організованої кіберзлочинності. Інформація і право. 2019. № 1 (28). С. 118–128.
12. Thompson E. C. Cybersecurity Incident Response: How to Contain, Eradicate, and Recover from Incidents. Berkeley : Apress, 2018.
13. Luttgens J., Mandia K., Pepe M. Incident Response and Computer Forensics. 3rd ed. New York : McGraw-Hill, 2014.
14. Cybersecurity Incident Management Guide. CREST, 2025. URL: https://www.crest-approved.org/wp-content/uploads/2025/10/2025_Cybersecurity-Incident-Management-Guide_Update_v1.0.pdf.
15. Tykushin A., Ostrovskaya S., Volkov D. Incident Response for Windows : Adapt Effective Strategies for Managing Sophisticated Cyberattacks Targeting Windows Systems. Birmingham : Packt Publishing, 2024. 244 p. URL: <https://www.packtpub.com/en-KR/product/incident-response-for-windows-9781804619322>.
16. Конвенція Ради Європи про кіберзлочинність : Конвенція від 23.11.2001 р., ратифікована Законом України № 2824-IV від 07.09.2005 р. URL: https://zakon.rada.gov.ua/laws/show/994_575.
17. Кримінальний процесуальний кодекс України : Закон України від 13.04.2012 р. № 4651-VI. URL: <https://zakon.rada.gov.ua/laws/show/4651-17>.
18. Цивільний процесуальний кодекс України : Закон України від 18.03.2004 р. № 1618-IV. URL: <https://zakon.rada.gov.ua/laws/show/1618-15>.
19. Використання електронних цифрових доказів у кримінальних провадженнях : метод. рек. / М. В. Гуцалюк, В. Д. Гавловський, В. Г. Хахановський та ін. ; за заг. ред. О. В. Корнейка. Вид. 2-ге, допов. Київ : Вид-во Нац. акад. внутр. справ, 2020. 104 с.
20. Гуцалюк М. В., Антонюк П. Є. Щодо сутності електронної цифрової інформації як джерел доказів у кримінальному провадженні. Криміналістичний вісник. 2020. № 1. С. 37–49.
21. Кобець М. В. Дії слідчого під час виявлення на місці події мобільних терміналів. Криміналістичний вісник. 2023. № 1 (39). С. 52–63.

22. Коваленко А. В. Електронні докази в кримінальному провадженні: сучасний стан та перспективи використання. Вісник Луганського державного університету внутрішніх справ імені Е. О. Дідоренка. 2018. № 4. С. 237–245.
23. Козицька О. Г. Щодо поняття електронних доказів у кримінальному провадженні. Юридичний науковий електронний журнал. 2020. № 8. С. 418–421.
24. Метелев О. П. Збирання цифрової інформації як окремий спосіб отримання доказів під час кримінального провадження. Науковий вісник Ужгородського національного університету. 2020. № 60. С. 177–181.
25. Метелев О. П. Проблеми визначення допустимості і належності цифрових електронних доказів у кримінальному процесі. Вісник кримінального судочинства. 2019. № 3. С. 224–238.
26. Перцова-Тодорова Л. Електронний доказ під час обшуку. Підприємництво, господарство і право. 2020. № 6. С. 243–247.
27. Ратнова А. В. Кримінальні процесуальні та криміналістичні основи використання електронних документів у доказуванні : дис. ... д-ра філософії. Львів, 2021. 248 с. URL: https://dspace.lvduvs.edu.ua/bitstream/1234567890/3747/1/ratnova_d.pdf.
28. Самойленко О. А. Виявлення та розслідування кіберзлочинів : навч.-метод. посіб. Одеса, 2020. 112 с.
29. Сіренко О. В. Електронні докази у кримінальному провадженні. Міжнародний юридичний вісник: актуальні проблеми сучасності. 2019. № 14. С. 208–214.
30. Скрипник А. В. Використання цифрової інформації в кримінальному процесуальному доказуванні : монографія. Харків : Право, 2022. 408 с. URL: <https://pravo-izdat.com.ua/novinki/vikoristannya-cifrovoyi-informaciyi-v-kriminalnomu-procesualnomu-dokazuvanni>.
31. Столітній А. В., Каланча І. Г. Формування інституту електронних доказів у кримінальному процесі України. Проблеми законності. 2019. № 146. С. 179–191. URL: <https://www.ceeol.com/search/articledetail?id=796280>.

32. Татаренко Г. В., Болгарєва К. В., Татаренко Д. В. Електронні документи як засіб доказування: сутність та правове регулювання. Актуальні проблеми права: теорія і практика. 2019. № 1. С. 111–119.
33. Хахановський В. Г., Гуцалюк М. В. Особливості використання електронних цифрових доказів у кримінальних провадженнях. Криміналістичний вісник. 2019. № 1. С. 13–19.
34. Хахановський В. Г., Гребенькова М. С. Виявлення, збирання та дослідження електронних відображень як джерел доказів. Юридичний часопис Національної академії внутрішніх справ. 2022. Т. 12, № 4. С. 28–39.
35. Чванкін С. А. Комп'ютерно-технічна експертиза у цивільному судочинстві. Право та державне управління. 2021. № 1. С. 45–51. DOI: 10.32840/pdu.2021.1.7.
36. Школьніков В. І. Правова основа отримання інформації з мережі Інтернет у кримінальному провадженні. Вісник НТУУ «КПІ». Політологія. Соціологія. Право. 2018. № 4. С. 172–176.
37. Гаркуша А. М., Каланча І. Г. Алгоритм прийняття рішень щодо вилучення електронних носіїв інформації під час обшуку. Кримінальна юстиція в Україні: реалії та перспективи : матеріали круглого столу, м. Львів, 11 черв. 2021 р. Львів : Львівський державний університет внутрішніх справ, 2021. С. 159–165. URL: <https://dspace.lvduvs.edu.ua/handle/1234567890/3865>.
38. Каланча І. Г., Гаркуша А. М. Копія електронної інформації як доказ у кримінальному провадженні: процесуальний та технічний аспекти. Юридичний науковий електронний журнал. 2021. № 8. С. 336–340. DOI: 10.32782/2524-0374/2021-8/77.
39. Нізовцев Ю. Ю., Омелян О. С. Щодо підготовки та призначення судових експертиз у межах розслідування кримінальних правопорушень, пов'язаних із кібератаками. Криміналістичний вісник. 2021. № 2 (36). С. 59–68. DOI: 10.37025/1992-4437/2021-36-2-59.
40. Теплицький Б. Б. Завдання, об'єкти та питання комп'ютерно-технічної судової експертизи. Юридичний часопис Національної академії внутрішніх справ. 2019. № 2 (18). С. 24–32. DOI: 10.33270/04191802.24.

41. Теплицький Б. Б. Актуальні питання призначення експертизи комп'ютерної техніки і програмних продуктів під час розслідування злочинів у сфері використання електронно-обчислювальних машин, систем, комп'ютерних мереж і мереж електрозв'язку. Науковий вісник Національної академії внутрішніх справ. 2021. № 3 (120). С. 28–34. DOI: 10.33270/01211203.28.
42. Щербаковський М. Г., Коршенко В. А. Комплексні телекомунікаційно-автотехнічні експертизи. Вісник Харківського національного університету внутрішніх справ. 2019. Вип. 4 (87). С. 179–186. DOI: 10.32631/v.2019.4.18.
43. Arnes A. Digital Forensics. Hoboken : Wiley, 2018.
44. Hassan N. A. Digital Forensics Basics: A Practical Guide Using Windows OS. New York : Apress, 2019.
45. Sammons J. The Basics of Digital Forensics: The Primer for Getting Started in Digital Forensics. Amsterdam : Syngress, 2012.
46. Yosifovich P., Ionescu A., Russinovich M. E., Solomon D. A. Windows Internals. Part 1: System Architecture, Processes, Threads, Memory Management, and More. 7th ed. Redmond : Microsoft Press, 2017.
47. Allievi A., Ionescu A., Russinovich M. E., Solomon D. A. Windows Internals. Part 2. 7th ed. Redmond : Microsoft Press, 2021.
48. Гаркуша А. М. Питання визначення часових міток файлів у файлових системах FAT і NTFS. Криміналістичний вісник. 2014. № 1 (21). С. 177–181.
49. Holt T. J., Bossler A. M., Seigfried-Spellar K. C. Cybercrime and Digital Forensics: An Introduction. London : Routledge, 2018.
50. Kavrestad J. Fundamentals of Digital Forensics: Theory, Methods, and Real-Life Applications. Cham : Springer International Publishing, 2018.
51. Shavers B. Placing the Suspect Behind the Keyboard: Using Digital Forensics and Investigative Techniques to Identify Cybercrime Suspects. Waltham : Syngress, 2013.
52. Phillips A., Nelson B., Steuart C. Guide to Computer Forensics and Investigations: Processing Digital Evidence. Boston : Cengage Learning, 2019.
53. Lin X. Introductory Computer Forensics: A Hands-on Practical Approach. Cham : Springer International Publishing, 2018.

54. Parasram S. V. N. Digital Forensics with Kali Linux: Perform Data Acquisition, Digital Investigation, and Threat Analysis Using Kali Linux Tools. Birmingham : Packt Publishing, 2018.
55. Sachowski J. Implementing Digital Forensic Readiness: From Reactive to Proactive Process. Amsterdam : Elsevier, 2016.
56. Sachowski J. Digital Forensics and Investigations: People, Process, and Technologies to Defend the Enterprise. London : Taylor and Francis, 2018.
57. Sammons J. Digital Forensics: Threatscape and Best Practices. Amsterdam : Syngress, 2016.
58. Maras M.-H. Computer Forensics: Cybercriminals, Laws, and Evidence. Burlington : Jones & Bartlett Learning, 2015.
59. Li C. Emerging Digital Forensics Applications for Crime Detection, Prevention, and Security. Hershey : Information Science Reference, 2013.
60. Kavrestad J. Guide to Digital Forensics: A Concise and Practical Introduction. Cham : Springer International Publishing, 2017.
61. Britz M. Computer Forensics and Cyber Crime: An Introduction. Boston : Pearson, 2013.
62. Chawki M., Darwish A., Khan M. A., Tyagi S. Cybercrime, Digital Forensics and Jurisdiction. Cham : Springer International Publishing, 2015.
63. Chen L., Takabi H., Le-Khac N.-A. Security, Privacy and Digital Forensics in the Cloud. Hoboken : Wiley, 2019.
64. Dalrymple B. E., Smith E. J. Forensic Digital Image Processing: Optimization of Impression Evidence. Boca Raton : CRC Press, 2018.
65. Ho A. T. S., Li S. Handbook of Digital Forensics of Multimedia Data and Devices. Chichester : Wiley, 2015.
66. Horan C., Saiedian H. Cyber Crime Investigation: Landscape, Challenges, and Future Research Directions. Journal of Cybersecurity and Privacy. 2021. Vol. 1, No. 4. P. 580–596. DOI: 10.3390/jcp1040029.
67. Zarpala L., Casino F. A Blockchain-Based Forensic Model for Financial Crime Investigation: The Embezzlement Scenario. Digital Finance. 2021. Vol. 3. P. 301–332. DOI: 10.1007/s42521-021-00035-5.

68. Digital Forensic Investigation of Internet of Things IoT Devices / R. Montasari, H. Jahankhani, R. Hill, S. Parkinson. Cham : Springer, 2021.
69. Jaswal N. Mastering Network Forensics: A Practical Approach to Investigating and Defending against Network Attacks. New Delhi : BPB Publications, 2024. URL: <https://bpbonline.com/products/mastering-network-forensics>.
70. Autopsy User Documentation. The Sleuth Kit. URL: <https://www.sleuthkit.org/autopsy/docs.php>.
71. Volatility 3 Documentation. The Volatility Foundation. URL: <https://volatility3.readthedocs.io/en/latest/>.
72. KAPE Documentation. Eric Zimmerman. URL: <https://ericzimmerman.github.io/KapeDocs/>.
73. Microsoft Sysinternals Documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/sysinternals/>.
74. Process Explorer. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer>.
75. Autoruns. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/sysinternals/downloads/autoruns>.
76. Sysmon. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon>.
77. Wireshark User's Guide. Wireshark Foundation. URL: https://www.wireshark.org/docs/wsug_html_chunked/.
78. Ghidra Software Reverse Engineering Framework. National Security Agency. URL: <https://github.com/NationalSecurityAgency/ghidra>.
79. YARA Documentation. VirusTotal. URL: <https://yara.readthedocs.io/en/stable/>.
80. MISP Documentation and Support. MISP Project. URL: <https://www.misp-project.org/documentation/>.

Навчальний посібник

**ПРАКТИЧНІ АСПЕКТИ РЕАГУВАННЯ
НА ІТ ІНЦИДЕНТИ ТА РОЗСЛІДУВАННЯ
ЦИФРОВИХ ЗАГРОЗ**

Електронне видання

© Центральноукраїнський національний
технічний університет, 2026
© Коваленко О.В., Коваленко А.С.,
Ткачук Р.О., Пукалов В.В.

Навчальний посібник присвячений вивченню ключових аспектів цифрової криміналістики, реагування на інциденти та аналізу шкідливого програмного забезпечення.

Розглядаються інструменти та методи, що використовуються різноманітними ІТ фахівцями для виявлення, аналізу та нейтралізації ІТ інцидентів.

Посібник включає практичні завдання та приклади використання таких інструментів, як Autopsy, The Volatility Framework, KAPE, Sysinternals Utilities, Wireshark, ProcDOT, Ghidra, dnSpy, PeStudio, CFF Explorer, Process Hacker, x64dbg та багато інших.

Рекомендовано для студентів закладів вищої освіти галузі знань Інформаційні технології.

