КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (34), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.34.1256
УДК 004.056.5

Завалій Станіслав
здобувач освітнього ступеня «бакалавр» спеціальності 125 «Кібербезпека та захист інформації»
Сумський державний університет, м. Суми, Україна
ORCID:0009-0002-9651-776X
s.zavalij@student.sumdu.edu.ua

Коваль Віталій
кандидат наук, Старший викладач
Сумський державний університет, м. Суми, Україна
ORCID:0000-0002-1593-5605
v.koval@dcs.sumdu.edu.ua

МЕТОДИ ТА ЗАСОБИ ФОРЕНЗИЧНОГО АНАЛІЗУ ОПЕРАТИВНОЇ ПАМ'ЯТІ ДЛЯ ВИЯВЛЕННЯ ПРИХОВАНИХ ПРОЦЕСІВ У КОМП'ЮТЕРНИХ СИСТЕМАХ

Анотація. У статті розглянуто проблему виявлення прихованих процесів та ін'єктованого шкідливого коду в оперативній пам'яті комп'ютерних систем під керуванням Microsoft Windows 11. Актуальність дослідження обумовлена стрімким зростанням частки безфайлових атак, які повністю розгортаються в RAM і не залишають слідів, доступних класичним засобам антивірусного та EDR-моніторингу. Метою статті є формалізація методики виявлення прихованих процесів через аналіз структур пам'яті – VAD-дерева та сторінкової організації – з використанням фреймворку Volatility 3 та подальшою зовнішньою валідацією результатів. У роботі систематизовано теоретичні засади організації віртуальної пам'яті Windows, проведено порівняльний аналіз шести інструментів збору дампів та чотирьох фреймворків аналізу, простежено еволюцію методів приховування шкідливого коду. Запропонована методика побудована як шестиетапний процес: підготовка, збір волатильних даних, тріаж, поглиблений аналіз у трьох паралельних гілках (процеси, пам'ять, мережа), крос-верифікація через зовнішні платформи кіберрозвідки та звітування з документуванням ланцюга збереження доказів. Експериментальна верифікація методики проведена на сценарії контрольованої атаки з використанням Metasploit Meterpreter та подальшої міграції шкідливого коду в адресний простір легітимного процесу explorer.exe Windows 11 за технікою T1055.001 згідно класифікації MITRE ATT&CK. Виявлено стійкий комплекс індикаторів компрометації: два приватні RWX-регіони пам'яті, характерний x86-64 пролог shellcode-у з типовим патерном звертання до PEB через сегмент GS, асиметрію VAD-дерева з відсутністю зв'язку аномальних регіонів з файлами на диску, а також жорстко закодовану адресу командно-керуючого сервера. Незалежна валідація через платформу VirusTotal підтвердила належність артефакту до сімейства shellcode/marte з рівнем виявлення 17 з 62 антивірусних двигунів. Порівняльний аналіз з еталонним дампом однозначно довів, що виявлені аномалії є наслідком модельованої атаки. Виявлені артефакти зіставлено з шістьма техніками матриці MITRE ATT&CK. Методика придатна для безпосереднього застосування у навчальному процесі підготовки фахівців з кібербезпеки та у практиці інцидент-респонсу.

Ключові слова: форензика оперативної пам'яті; приховані процеси; Reflective DLL Injection; Meterpreter; Volatility 3; Magnet RESPONSE; безфайлові атаки; Windows 11; VAD-дерево; MITRE ATT&CK; цифрова форензика.

ВСТУП

Сучасний ландшафт кіберзагроз протягом останнього десятиліття зазнав фундаментальної трансформації. Якщо раніше переважна більшість шкідливих програм залишала чіткі сліди на жорсткому диску у вигляді виконуваних файлів, що дозволяло ефективно протидіяти їм засобами сигнатурного антивірусного захисту, то сьогодні все більшого поширення набувають безфайлові атаки. Вони повністю розгортаються в оперативній пам'яті, використовуючи легітимні системні інструменти та техніки ін'єкції

коду в довірені процеси, і фактично не залишають слідів, доступних класичним методам цифрової форензики [1].

За даними галузевих дослідницьких звітів компанії Check Point Research, у другому кварталі 2024 року глобальна кількість кібератак зросла на 30 % і досягла в середньому 1999 атак на одну організацію щотижня – це найвищий показник за два попередні роки [2]. Суттєву частку у цьому потоці становлять саме безфайлові кампанії, що використовують техніки Process Injection, Reflective DLL Loading та Living off the Land. У звіті MITRE ATT&CK техніка T1055 (Process Injection) розширилася до п'ятнадцяти підтехнік (T1055.001-T1055.015), що ілюструє різноманітність сучасних методів ухилення зловмисників від виявлення [3]. Подібна динаміка робить форензичний аналіз оперативної пам'яті не допоміжним, а критично важливим компонентом сучасного інцидент-респонсу.

Незважаючи на наявність потужних інструментальних засобів – передусім фреймворку Volatility 3, який у 2025 році остаточно витіснив застарілу другу версію та став де-факто стандартом аналізу пам'яті [4] – для практикуючого аналітика залишається відкритою методологічна проблема послідовного, відтворюваного застосування цих засобів у межах єдиного робочого процесу. Існує суттєвий розрив між теоретичним описом окремих плагінів і алгоритмів та практичним досвідом їх інтегрованого використання у конкретному сценарії розслідування. Цей розрив особливо відчутний у контексті дослідження систем Windows 11, для яких україномовні методичні матеріали з memory forensics ще тільки формуються.

Постановка проблеми. Класичні засоби виявлення шкідливого програмного забезпечення – сигнатурні антивіруси, IDS-системи та поведінкові EDR-рішення – мають принципові обмеження при протидії атакам, що повністю розгортаються в оперативній пам'яті. Антивіруси не можуть проаналізувати код, який не існує у вигляді файлу на диску. EDR-системи, орієнтовані на моніторинг подій, не завжди контролюють поведінку довірених системних процесів, таких як explorer.exe чи svchost.exe. Мережеві IDS не помічають одиничного TCP-з'єднання у легітимному діапазоні портів. Натомість будь-яка ін'єкція коду або міграція шкідливого процесу в легітимний неминуче залишає характерні артефакти в адресному просторі пам'яті, виявлення яких потребує цілеспрямованого структурного аналізу. Проблема полягає у формалізації покрокового алгоритму такого аналізу, який забезпечував би відтворюваність результатів, об'єктивну верифікацію виявлених індикаторів та сумісність висновків з міжнародною практикою інцидент-респонсу.

Аналіз останніх досліджень і публікацій. Питання форензичного дослідження оперативної пам'яті розглядаються у численних роботах закордонних дослідників. Фундаментальні засади аналізу VAD-дерева як ключової структури адресного простору процесу Windows закладено у роботі Б. Долан-Гевітта [5], а класичні методики ін'єкції коду через Reflective DLL Loading формалізовано С. Фьюером у 2008 році [6]. Дослідницька група Microsoft Defender ATP опублікувала ґрунтовний аналіз технік виявлення Reflective DLL Loading через моніторинг подій пам'яті, відзначивши, що події виділення пам'яті є основним сигналом для виявлення цих атак [7]. У роботі П. Пілаї та співавторів [1] представлено структурований підхід до виявлення безфайлового шкідливого ПЗ через Volatility Framework, однак автори не приділяють достатньої уваги етапу зовнішньої валідації виявлених артефактів.

У ґрунтовному систематичному огляді 2025 року, опублікованому в ACM Computing Surveys, автори застосовують методологію OSCAR (Obtain, Strategize, Collect, Analyze, Report) до аналізу пам'яті, однак фокусуються переважно на загальних принципах без прив'язки до конкретних сценаріїв атак Windows 11 [8]. Дослідження 2025 року, опубліковане у журналі Cluster Computing, демонструє ефективність застосування глибокого навчання (CNN, RNN) для класифікації шкідливого ПЗ за артефактами пам'яті з точністю до 97,8 %, однак вимагає попередньої наявності великих розмічених наборів даних, що обмежує його використання у щоденній практиці [9]. У роботі Volatility Foundation 2024-2025 років деталізовано аналіз новітніх Linux-зразків з резидентною у пам'яті руткіт-функціональністю sedexp, що засвідчує постійну актуальність розвитку анти-форензичних методів [10]. Свіжі публікації галузевих дослідників TrustedSec [11] та Cynet [12] деталізують сучасні модифікації техніки Reflective DLL Injection і вказують на необхідність їх систематичного виявлення засобами memory forensics. Аналітичний звіт Pen Test Partners 2025 року узагальнює ключові переваги аналізу пам'яті у виявленні шкідливих DLL-ін'єкцій та аномальної поведінки пам'яті [13]. Незважаючи на значну кількість публікацій, у відкритому доступі бракує практично орієнтованих україномовних методик, що інтегрують усі етапи дослідження – від збору дампу до зовнішньої валідації – на прикладі сучасних версій Windows 11.

Метою статті є формалізація методики виявлення прихованих процесів та ін'єктованого шкідливого коду у дампах оперативної пам'яті систем Windows 11 на основі аналізу структур VAD-дерева та сторінкової організації засобами фреймворку Volatility 3, а також її експериментальна верифікація на сценарії контрольованої атаки з використанням Metasploit Meterpreter з подальшою

зовнішньою валідацією результатів через платформу VirusTotal та зіставленням виявлених артефактів з матрицею MITRE ATT&CK.

ТЕОРЕТИЧНИЙ БАЗИС ТА ВИБІР ІНСТРУМЕНТАРІЮ

Основою методики виявлення прихованих процесів є аналіз структур пам'яті, що формуються операційною системою для керування адресним простором кожного процесу. Сучасні операційні системи реалізують концепцію віртуальної пам'яті, що дозволяє кожному процесу працювати у власному ізольованому адресному просторі, який значно перевищує обсяг фізично доступної RAM. Реалізація цього механізму базується на сторінковій організації: віртуальний адресний простір розбивається на сторінки фіксованого розміру (стандартно 4 КБ для архітектури x86-64, з можливістю використання великих сторінок розміром 2 МБ або 1 ГБ), а фізична пам'ять – на відповідні фрейми. Перетворення віртуальних адрес у фізичні здійснюється апаратно за допомогою блока керування пам'яттю (Memory Management Unit, MMU) центрального процесора, який використовує ієрархічну чотирирівневу структуру таблиць сторінок: PML4 → PDPT → PD → PT [14].

Кожен процес у Windows представлений структурою _EPROCESS у режимі ядра, що містить покажчики на каталог сторінок процесу, список потоків, маркер безпеки, відкриті дескриптори об'єктів та інші критично важливі для функціонування процесу метадані. Усі активні процеси організовані у двозв'язний список ActiveProcessLinks, обхід якого дозволяє стандартним інструментам, таким як Диспетчер задач, отримати перелік запущених програм. Саме модифікація цього списку (відомий метод DKOM – Direct Kernel Object Manipulation) історично була одним із перших і найвідоміших способів приховування шкідливих процесів.

Ключовою структурою керування адресним простором процесу у Windows є дерево дескрипторів віртуальних адрес (Virtual Address Descriptor Tree, VAD). VAD-дерево являє собою самобалансоване AVL-дерево, кожен вузол якого описує безперервний діапазон віртуальних адрес у межах процесу [5]. Коли процес виконує виклик функції VirtualAlloc або завантажує бібліотеку через LoadLibrary, диспетчер пам'яті створює відповідний вузол у VAD-дереві, який містить початкову та кінцеву віртуальні адреси регіону, його тип (приватна пам'ять, відображений файл, образ виконуваного файлу), атрибути захисту та інші службові поля. Особливістю реалізації Windows є те, що сам факт виклику VirtualAlloc лише резервує та комітить діапазон у VAD, тоді як реальні записи у таблиці сторінок створюються лише при першому звертанні до відповідної сторінки за допомогою механізму lazy allocation.

Для форензичного аналізу VAD-дерево має виключне значення з кількох причин. По-перше, воно надає повну та достовірну карту адресного простору процесу безпосередньо з пам'яті ядра, незалежно від того, чи намагається шкідливе програмне забезпечення приховати окремі модулі від стандартних API. По-друге, кожен вузол VAD містить інформацію про права доступу до відповідного регіону пам'яті у вигляді набору прапорців, які відповідають константам PAGE_READONLY, PAGE_READWRITE, PAGE_EXECUTE, PAGE_EXECUTE_READ та PAGE_EXECUTE_READWRITE. Саме комбінація PAGE_EXECUTE_READWRITE (RWX), що одночасно дозволяє і запис, і виконання, є яскравим індикатором аномалії, оскільки легітимне програмне забезпечення вкрай рідко потребує таких прав, тоді як для шкідливого коду це фактично необхідна умова виконання внесеного шеллкоду [6].

Поверх VAD-дерева Microsoft реалізувала набір механізмів захисту, спрямованих на запобігання експлуатації вразливостей пам'яті. Ці механізми, хоч і мають передусім захисну природу, безпосередньо впливають і на форензичний аналіз: розуміння принципів їхньої роботи дає змогу аналітику правильно ідентифікувати аномальні зміни прав доступу до сторінок пам'яті, які зазвичай свідчать про спробу ін'єкції шкідливого коду. Ключові механізми захисту, реалізовані у сучасних версіях Windows, узагальнено у таблиці 1.

Таблиця 1

Механізми захисту адресного простору процесів Windows

Механізм	З якої версії	Принцип роботи	Способи обходу
DEP	XP SP2	Заборона виконання коду зі сторінок, позначених NX-бітом; апаратний контроль через MMU	Return-Oriented Programming (ROP), повторне використання легітимного коду
ASLR	Vista	Випадкове розміщення базових адрес виконуваного образу, стека, купи та DLL у	Heap spraying, витоки інформації (memory

Механізм	З якої версії	Принцип роботи	Способи обходу
		пам’яті	disclosure)
CFG	8.1	Бітова карта дозволених точок переходу для непрямих викликів; перевірка перед кожним call	Атаки через JIT-двигуни, обхід через невипадкові цілі
ACG	10 (1703+)	Заборона виділення нової виконуваної пам’яті у межах процесу (Arbitrary Code Guard)	Несумісність з JIT, тому не вмикається в браузерах за замовчуванням
CIG	10 (1703+)	Code Integrity Guard: вимога підпису Microsoft або WHQL для всіх модулів	Підписані вразливі драйвери (BYOVD)
HVCI	10 / 11	Hypervisor-Protected Code Integrity: захист цілісності коду ядра через VBS	Атаки на гіпервізор, апаратні вразливості процесора

Як видно з таблиці 1, кожен наступний механізм захисту вирішує специфічний клас атак, що залишався актуальним після впровадження попередніх. Однак з погляду форензичного аналізу важливо, що навіть найдосконаліші механізми захисту не виключають можливості ін’єкції коду в легітимний процес – вони лише ускладнюють шлях, яким зловмисник цього досягає. Класичні методики ін’єкції, описані в межах техніки T1055 фреймворку MITRE ATT&CK, передбачають виділення нової області пам’яті у цільовому процесі за допомогою VirtualAllocEx, запис шеллкоду через WriteProcessMemory і запуск його у вигляді нового потоку через CreateRemoteThread [3]. Усі такі дії неминуче залишають слід у VAD-дереві цільового процесу у вигляді приватного регіону пам’яті з правами PAGE_EXECUTE_READWRITE, який не відповідає жодному легітимному модулю.

Особливе місце у класифікації займає техніка Reflective DLL Injection, описана дослідником Стівеном Фьюером у 2008 році [6]. На відміну від класичної ін’єкції через LoadLibrary, рефлективне завантаження дозволяє відобразити DLL безпосередньо з пам’яті у адресний простір цільового процесу, минаючи штатний завантажувач Windows. Метод був інтегрований у фреймворк Metasploit як частина пейлоада Meterpreter і активно застосовується сучасними зловмисниками. Загальна класифікація сучасних технік приховування шкідливого коду в оперативній пам’яті узагальнена в таблиці 2.

Таблиця 2

Класифікація сучасних технік приховування шкідливого коду в RAM

Період	Техніка	Приклади/зразки	Метод виявлення у пам’яті
2001-2003	Memory-only worms	Code Red, SQL Slammer	Дамп пам’яті ураженої служби
2004-2008	Kernel rootkits/DKOM	FU Rootkit, Hacker Defender	psscan: пошук _EPROCESS за сигнатурою
2008-2012	Reflective DLL Injection	Meterpreter, Cobalt Strike	malfind: пошук RWX-регіонів без модуля
2010-2015	Process Hollowing	Stuxnet, Duqu, TrickBot	Порівняння PE-образу з дисковою копією
2014-до сьогодні	Living off the Land (LotL)	PowerShell Empire, FIN7, APT29	Поведінковий аналіз + memory forensics
2017-до сьогодні	Fileless ransomware	Sorebrect, Netwalker	EDR + memory forensics + аналіз PowerShell
2018-до сьогодні	Process Doppelgänging	SynAck ransomware	Аналіз транзакцій NTFS + аналіз пам’яті

Як демонструє таблиця 2, кожне наступне покоління технік стає все витонченішим у плані маскування, однак не може повністю позбутися слідів у пам’яті. Це фундаментальна особливість

виконання коду на сучасних процесорах: будь-який код, що виконується, мусить перебувати в адресному просторі певного процесу, а отже потрапити в поле зору правильно застосованого аналізу пам'яті [13].

Сучасний ринок інструментарію для форензики оперативної пам'яті є відносно зрілим і охоплює дві категорії продуктів: інструменти збору (acquisition tools), які формують саме файл дампу, та фреймворки аналізу, які приймають файл дампу на вхід і дозволяють витягувати з нього структуровану інформацію про процеси, мережеві з'єднання, завантажені модулі тощо. Серед інструментів збору пам'яті провідні позиції займають FTK Imager, Magnet RAM Capture, Belkasoft Live RAM Capturer, WinPmem, DumpIt та Magnet RESPONSE [15]. Для практичної частини дослідження обрано Magnet RESPONSE як рішення, що поєднує функціональність швидкого збору дампу з паралельним збереженням журналів подій Windows та гілок реєстру в єдиному ZIP-архіві з обчисленням хешів. Серед фреймворків аналізу безальтернативним відкритим вибором для Windows 11 є Volatility 3 [4], який, на відміну від попередньої версії, використовує систему Intermediate Symbol Files (ISF), що автоматично генеруються на основі публічно доступних PDB-символів від Microsoft. Це знімає з аналітика проблему підбору профілю пам'яті для конкретної збірки ОС. Узагальнене порівняння провідних фреймворків наведено у таблиці 3.

Таблиця 3

Порівняння фреймворків аналізу дампів оперативної пам'яті

Критерій	Volatility 3	Volatility 2	MemProcFS	Rekall
Активна підтримка	Так	Ні (з 2020)	Так	Ні (з 2020)
Повна підтримка Windows 11	Так	Часткова	Так	Ні
Профілі/символи	ISF (автозавант.)	Статичні	Symbol Server	Profiles
Інтерфейс	CLI + API (Python 3)	CLI (Python 2)	CLI + FUSE	CLI + iPython
Кількість плагінів	≈200 (зростає)	≈250	≈80	≈170
JSON експорт	Вбудований	Через плагіни	Через FUSE	Так
Інтеграція з SIEM	Висока	Середня	Висока	Низька
Ліцензія	Volatility SLA	GPLv2	AGPL	Apache 2.0

Ключовими плагінами Volatility 3, які використовуються у запропонованій методиці для виявлення прихованих процесів та ін'єкції коду, є: windows.pslist (стандартний перелік активних процесів через ActiveProcessLinks), windows.psscan (пошук прихованих _EPROCESS за сигнатурою у дампі), windows.malfind (пошук RWX-регіонів з ознаками ін'єкції коду), windows.vadinfo (детальна карта VAD-дерева заданого процесу), windows.vaddump (експорт окремих регіонів пам'яті у файл), windows.netscan (виявлення мережевих з'єднань) та windows.dlllist (перелік завантажених DLL процесу). Поєднання цих плагінів забезпечує повний цикл структурного аналізу адресного простору будь-якого процесу [16].

ЗАПРОПОНОВАНА МЕТОДИКА

Розроблена методика форензичного аналізу оперативної пам'яті побудована як послідовний шестиетапний процес, що інтегрує стандарти NIST SP 800-86, IETF RFC 3227 та матрицю MITRE ATT&CK. Основними принципами, на яких базується методика, є: (1) мінімізація впливу процесу збору дампу на стан цільової системи; (2) обчислення криптографічних хешів дампу одразу після збору для забезпечення цілісності доказів; (3) застосування кількох незалежних плагінів аналізу для підвищення надійності висновків; (4) обов'язкова крос-верифікація виявлених артефактів через зовнішні платформи кіберрозвідки; (5) ведення безперервного журналу chain of custody на всіх етапах роботи. Узагальнена структура методики представлена на рисунку 1 у вигляді блок-схеми, що відображає послідовність шести основних етапів та логіку прийняття рішень між ними.

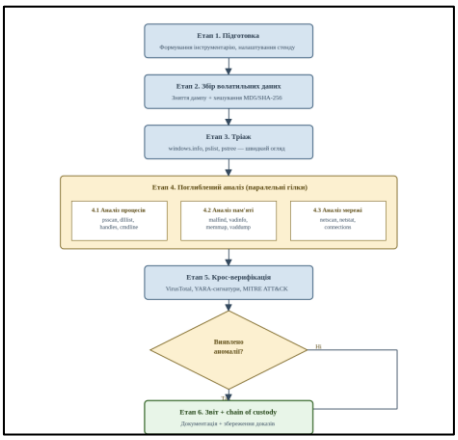

Рис. 1. Блок-схема запропонованої методики форензичного аналізу оперативної пам'яті

Етап 1 (Підготовка) передбачає визначення мети дослідження (тип інциденту, очікувані індикатори компрометації, обмеження часу), формування переліку необхідного інструментарію, підготовку робочого середовища – ізольованої аналітичної робочої станції з достатнім обсягом дискового простору (мінімум удвічі більше за обсяг очікуваного дампу), а також створення документального шаблону для журналу chain of custody, у якому фіксуватимуться усі операції з дампом: час та місце збору, ідентифікатор системи-джерела, прізвище аналітика, методологія обробки, обчислені хеші, місця зберігання копій. Невиконання цього кроку може призвести до неприйнятності зібраних доказів у судовому процесі.

Етап 2 (Збір волатильних даних) реалізує безпосереднє зняття дампу оперативної пам'яті з цільової системи. Дотримується така послідовність: підключення USB-носія з портативною версією інструменту збору; запуск інструмента з підвищеними привілеями; ініціювання збору повного дампу пам'яті у форматі raw з паралельним збором журналів подій Windows та гілок реєстру; обчислення MD5 та SHA-256 хешів отриманого дампу; занесення усіх параметрів до журналу chain of custody. Принципово важливим є вибір місця збереження дампу: запис на той самий диск, який аналізується, неприйнятний, оскільки змінює стан файлової системи цільової машини; найбільш надійним варіантом є запис на спеціально підготовлений зовнішній USB-носій.

Етап 3 (Тріаж) – швидка попередня оцінка стану системи на момент зняття дампу. Виконуються плагіни windows.info (метадані дампу: версія Windows, build, час завантаження, час знімання), windows.pslist (стандартний перелік активних процесів) та windows.pstree (ієрархічне відображення дерева процесів). Тріаж дозволяє відповісти на базові питання: чи коректно інтерпретується дамп, які процеси були активні, чи є серед них явно підозрілі (несподівано запущений PowerShell, cmd.exe з аномальними батьківськими процесами тощо). Якщо на цьому етапі вже виявлено очевидні аномалії, аналітик отримує початкову гіпотезу для подальшої перевірки.

Етап 4 (Поглиблений аналіз) є найбільш об'ємним і виконується паралельно у трьох гілках. Перша гілка – аналіз процесів. Виконуються плагіни psscan (пошук прихованих _EPROCESS за сигнатурою), dlllist (список завантажених DLL для кожного процесу), handles (відкриті дескриптори об'єктів), cmdline (командний рядок запуску процесу). Зіставлення результатів pslist та psscan дозволяє виявити процеси, видалені з ActiveProcessLinks через DKOM-маніпуляцію. Аналіз dlllist для підозрілих процесів дозволяє виявити нетипові завантажені модулі. Друга гілка – аналіз пам'яті. Виконуються плагіни malfind (пошук RWX-регіонів з ознаками ін'єкції), vadinfo (детальна карта VAD-дерева для підозрілих процесів), memmap (мапа віртуальної пам'яті), vaddump (дамп окремих регіонів пам'яті для подальшого реверс-інжинірингу). Це гілка має ключове значення для виявлення ін'єкцій типу Reflective DLL Injection та Process Hollowing. Третя гілка – аналіз мережі. Виконуються плагіни netscan (сканування мережевих структур у дампі), netstat (узагальнений список з'єднань). Виявлені з'єднання порівнюються з очікуваним мережевим профілем системи; будь-які відхилення (з'єднання з невідомими IP-адресами, нестандартні порти, з'єднання, ініційовані системними процесами) розглядаються як індикатори компрометації.

Принципово важливим є проведення усіх трьох гілок навіть якщо одна з них одразу показала позитивний результат – це необхідно для повноти картини та виключення хибнопозитивних висновків.

Етап 5 (Крос-верифікація) передбачає незалежну перевірку виявлених артефактів засобами зовнішніх систем. Для процесів та виконуваних модулів обчислюються SHA-256 хеші та виконується пошук у базі VirusTotal [17]. Для регіонів пам'яті та витягнутих файлів застосовується сканування YARA-правилами проти баз публічних сигнатур (Florian Roth's Signature Base, ReversingLabs YARA

Rules). Підозрілі індикатори (IP-адреси C2-серверів, шляхи до файлів, мютекси) перевіряються в базах кіберрозвідки (AlienVault OTX, ThreatFox, MISP). Крос-верифікація переслідує дві цілі: (1) дозволяє атрибутувати виявлений інцидент – віднести його до конкретної кампанії, родини шкідливого ПЗ або групи зловмисників; (2) забезпечує об'єктивну валідацію: висновок аналітика, підкріплений зовнішніми незалежними джерелами, є значно вагомішим за суто внутрішнє припущення.

Етап 6 (Звіт та chain of custody) завершує цикл дослідження формуванням підсумкового звіту, що включає: вступну частину з описом цілей та обмежень дослідження; технічну частину з детальним описом виявлених індикаторів, доказовою базою (скріншоти виводів плагінів, хеш-значення, текстові артефакти); інтерпретаційну частину з реконструкцією сценарію атаки, відповідністю до MITRE ATT&CK; рекомендації щодо реагування та превентивних заходів. Окремим важливим компонентом є додаток із журналом chain of custody, що документує ланцюг роботи з доказами від моменту збору до моменту представлення звіту.

Запропонована методика має кілька принципових особливостей. По-перше, модульність: кожен етап є самодостатнім і може бути розширений або модифікований залежно від конкретного інциденту. По-друге, універсальність: хоча у межах цієї статті методика орієнтована передусім на сценарій з Meterpreter-сесією у Windows 11, сама структура є застосовною до широкого кола інцидентів – безфайлові атаки, рансомвар, APT-кампанії, інсайдерські загрози. По-третє, наголос на крос-верифікації та документуванні: жоден висновок не приймається на основі результату лише одного плагіна; жодна дія не виконується без занесення до журналу. Це забезпечує юридичну та наукову допустимість результатів.

ЕКСПЕРИМЕНТАЛЬНА ВЕРИФІКАЦІЯ

Опис експериментального стенду. Експериментальна частина дослідження проведена на ізольованому віртуальному стенді з метою забезпечення повної відтворюваності та усунення ризиків поширення шкідливого коду поза межі тестового середовища. Цільовою системою виступала віртуальна машина Windows 11 Pro (хост DESKTOP-A4G19PD, користувач User), розгорнута в Oracle VirtualBox. Як систему атакуючого використано Kali Linux зі встановленим Metasploit Framework v6.4.64-dev [18]. Цільовій машині присвоєно IP-адресу 192.168.50.211, машині зловмисника – 192.168.50.218; обидві машини функціонують у спільній віртуальній мережі без зв'язку із зовнішніми мережами. Перед початком експерименту цільову систему налаштовано на створення повного дампу пам'яті у разі критичної помилки (режим «Complete memory dump» у параметрах System failure), що є технічною передумовою для коректної роботи інструментів криміналістичного аналізу пам'яті.

Для збору оперативної пам'яті використано спеціалізований криміналістичний інструмент Magnet RESPONSE v1.7.1 [19] – безкоштовне рішення для оперативного збору летких даних з працюючої системи. На відміну від класичних утиліт (DumpIt, FTK Imager), Magnet RESPONSE додатково фіксує метадані виконуваних процесів, мережевих з'єднань, завантажених модулів та системних логів, що суттєво розширює подальші можливості аналізу. Збір еталонного («чистого») дампу виконано до проведення будь-яких атакувальних дій (рисунок 2). Результатом стала директорія, що містить файл RAMDump.dmp розміром 8,37 ГБ та ZIP-архів з додатковими криміналістично значущими даними обсягом близько 145 МБ.

Рис. 2. Збір еталонного дампу оперативної пам'яті за допомогою Magnet RESPONSE

Завершальною підготовчою операцією стало навмисне відключення вбудованого антивірусу Microsoft Defender (параметр «Real-time protection» переведено у стан «Off»), що було необхідно для подальшого етапу контрольованого зараження системи. Зауважимо, що відключення антивірусу здійснено вже після збору еталонного дампу – це гарантує, що «чистий» зразок пам'яті був отриманий зі стандартної конфігурації захисту Windows 11.

Реалізація сценарію атаки. Сценарій атаки реалізовано за моделлю Cyber Kill Chain і охоплює шість послідовних етапів: розвідку, озброєння, доставку, експлуатацію, встановлення командно-керуючого каналу та подальші дії на цілі.

На етапі розвідки атакувач з'ясовує мережеву конфігурацію цільової системи. На цільовій машині Windows 11 командою ipconfig встановлено, що системі присвоєно IPv4-адресу 192.168.50.211 у мережі 192.168.50.0/24 з маскою 255.255.255.0 та шлюзом 192.168.50.1. Версія операційної системи — Windows 10.0.26200.6584 (Windows 11 Pro). Ця інформація необхідна зловмиснику для подальшого формування корисного навантаження зі вказанням адреси зворотного з'єднання.

На етапі озброєння у середовищі Kali Linux за допомогою утиліти msfvenom зі складу Metasploit Framework згенеровано шкідливий виконуваний файл у форматі PE для систем Windows. Виконана команда має такий вигляд:

```
msfvenom -p windows/meterpreter/reverse_tcp \
        LHOST=192.168.50.218 \
        LPORT=4444 \
        -f exe > diplom_test.exe
```

Як показано на рисунку 3, у відповідь msfvenom повідомив про автоматичний вибір платформи Windows та формат raw payload без енкодера. Розмір самого payload становить 510 байтів, а підсумковий розмір згенерованого виконуваного файлу – 7168 байтів. Такий мінімальний обсяг типовий для Meterpreter-stager: основна функціональність завантажується вже після встановлення з'єднання, що відповідає принципу безфайлових атак.

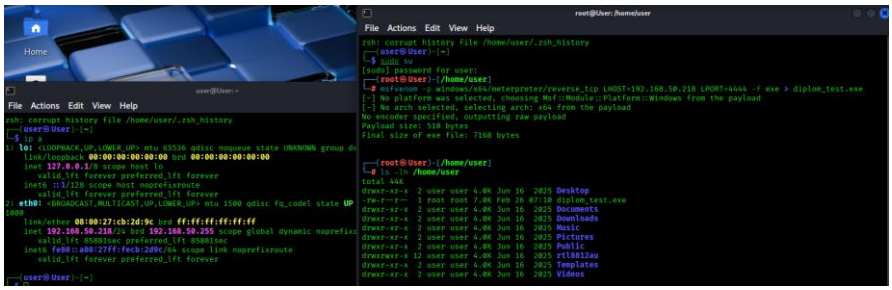

Рис. 3. Формування шкідливого виконуваного файлу засобом msfvenom

На етапі доставки сформований файл переноситься на цільову Windows-машину через спільну папку гіпервізора VirtualBox (Shared Folders). Це найпростіший спосіб у межах ізольованого стенду, що моделює ситуацію, коли користувач отримав файл через довільний канал комунікації (вкладення в листі, посилання на завантаження, USB-носій). Паралельно у Kali запускається обробник зворотних з'єднань (handler) у середовищі msfconsole з налаштуваннями PAYLOAD windows/x64/meterpreter/reverse_tcp, LHOST 192.168.50.218, LPORT 4444. Етап експлуатації реалізується запуском виконуваного файлу на цільовій системі, що призводить до встановлення зворотного TCP-з'єднання з ВМ зловмисника. У консолі Metasploit це відображається як активна Meterpreter-сесія (рисунок 4) з повідомленнями про передачу stage розміром 203 846 байтів та відкриття сесії 1.

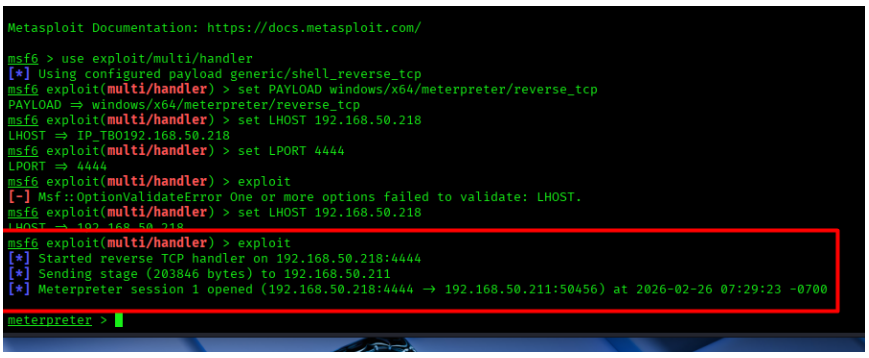

Рис. 4. Встановлення Meterpreter-сесії на скомпрометованій системі

Ключовим етапом експерименту, що становить безпосередній інтерес для форензичного аналізу, є міграція Meterpreter-процесу в адресний простір легітимного системного процесу explorer.exe (техніка Process Migration, відома також як Reflective DLL Injection, T1055.001 за класифікацією MITRE ATT&CK [20]). Для цього у Meterpreter-консолі спершу виконується команда ps, яка повертає повний перелік процесів цільової системи. У переліку було ідентифіковано цільовий процес explorer.exe із PID 4704, що виконується від імені користувача DESKTOP-A4G19PD\\User. Безпосередньо міграція виконується командою migrate 4704 (рисунок 5):

```
meterpreter > migrate 4704
[*] Migrating from 10538 to 4704...
[*] Migration completed successfully.
```

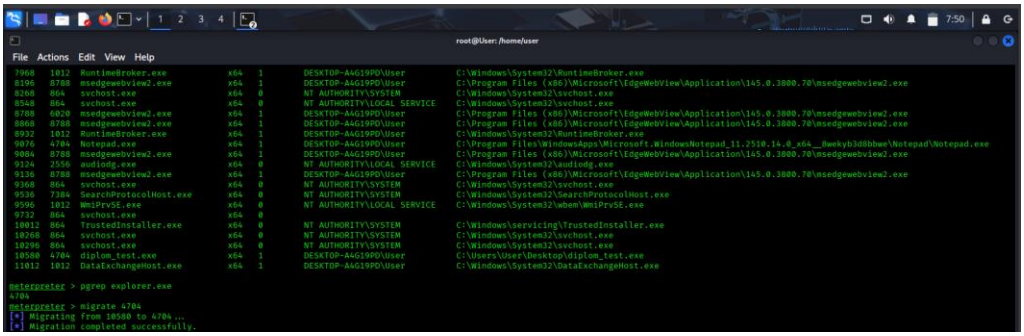

Рис. 5. Ідентифікація процесу explorer.exe та виконання міграції коду

Внаслідок виконання команди migrate Meterpreter реалізує техніку Reflective DLL Injection: у адресному просторі цільового процесу explorer.exe виділяється новий регіон пам’яті за допомогою VirtualAllocEx з прапорцями PAGE_EXECUTE_READWRITE, у який записується тіло Meterpreter-DLL, після чого створюється новий потік (CreateRemoteThread), який виконує функцію ReflectiveLoader для саморозгортання DLL без виклику стандартного завантажувача Windows [11]. З точки зору операційної системи процес explorer.exe залишається легітимним системним процесом з PID 4704. Дочірній процес diplom_test.exe (PID 10538), через який сталася початкова експлуатація, завершується одразу після успішної міграції, що додатково ускладнює виявлення інциденту звичайними засобами поведінкового моніторингу. Однак з точки зору форензичного аналізу пам’яті ця маніпуляція залишає характерні артефакти. Після завершення атаки повторно зібрано «забруднений» дамп оперативної пам’яті обсягом 8,37 ГБ за тією ж самою процедурою, що й еталонний дамп.

Аналіз дампу засобами Volatility 3. Дослідження зібраного забрудненого дампу виконано засобами фреймворку Volatility 3 версії 2.26.0 на ізольованій робочій станції аналітика. Першим кроком виконано плагін windows.pslist для отримання переліку активних процесів. У виводі коректно відображено базові системні процеси (System, smss.exe, csrss.exe, wininit.exe, services.exe, lsass.exe, svchost.exe, dwm.exe), а також VBoxService.exe, що підтверджує середовище VirtualBox. Серед перелічених процесів особливий інтерес становить процес explorer.exe з PID 4704, його батьківським процесом виступає userinit.exe (PID 4684), кількість потоків – 97. На перший погляд процес виглядає абсолютно штатним: сам факт нормального батьківського процесу та типового користувача не дає жодних підстав для підозри. Стандартний моніторинг на цьому етапі не показав би жодних аномалій – і саме у цьому полягає ефективність техніки Reflective DLL Injection.

Ключовою для виявлення ін’єкції коду є наступна гілка аналізу – пошук RWX-регіонів плагіном windows.malfind [16]. Цей плагін перебирає VAD-дерево вказаного процесу і повідомляє про регіони з прапорцями PAGE_EXECUTE_READWRITE, які з високою ймовірністю містять ін’єктований шкідливий код. Виконання плагіна для процесу explorer.exe з PID 4704 (рисунок 5) дало ключовий результат: виявлено два приватні (тег VadS) регіони пам’яті з прапорцями PAGE_EXECUTE_READWRITE. Перший регіон починається з адреси 0x00ac0000 і має розмір 4 КБ; другий регіон, починаючи з адреси 0x03700000, займає простір до 0x03731fff, тобто має розмір приблизно 200 КБ. Розмір другого регіону практично точно збігається з розміром Meterpreter stage (203 846 байтів), що передавався по мережі.

ISSN 2663 – 4023

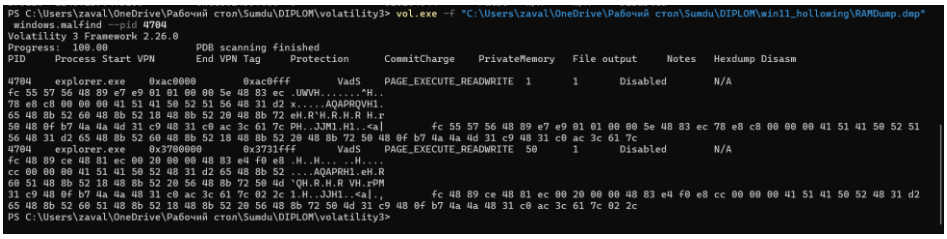

Рис. 6. Виявлення RWX-регіонів у процесі explorer.exe плагіном windows.malfind

Перший регіон починається з характерного шеллкод-прологу x86-64: байтова послідовність fc 55 57 56 48 89 e7 e9 01 01 00 00, де перший байт fc декодується як інструкція cld (очищення прапорця direction для коректної роботи рядкових інструкцій). Далі у дампі присутня послідовність 41 51 41 50 52 51 56 48 31 d2, що відповідає збереженню регістрів rcx, r8, r9, rdx, rsi на стек і обнуленню rdx – стандартна підготовка оточення перед викликом функцій. Найважливішим є наступний фрагмент 65 48 8b 52 60, який декодується як mov rdx, gs:[0x60] – звертання до сегментного регістру GS зі зміщенням 0x60 є канонічним способом отримання вказівника на структуру PEB (Process Environment Block) у 64-розрядних Windows-системах. Подальші інструкції здійснюють обхід списку завантажених модулів через PEB.Ldr.InMemoryOrderModuleList – типовий патерн пошуку експортованих функцій у Windows API без використання LoadLibrary/GetProcAddress. Цей шаблон поведінки докладно описано у роботі С. Фьюера [6] і є практично канонічною ознакою Meterpreter-payload або іншого коду на основі рефлективного завантажувача.

Другий, більший регіон починається з характерної shellcode-сигнатури: байтова послідовність fc 48 89 ce 48 81 ec 00 20 00 00, що інтерпретується як інструкції cld; mov rsi, rcx; sub rsp, 0x2000. Як і у першому регіоні, тут немає стандартного PE-заголовка з MZ-сигнатурою, який характерний для легітимних виконуваних модулів, завантажених через штатний LoadLibrary. Натомість регіон містить «сирий» виконуваний код – позиційно-незалежний shellcode, що готує власне виконавче оточення безпосередньо у пам'яті процесу-носія, без участі стандартного завантажувача Windows. Така структура – приватний регіон без зв'язку з файлом на диску, що починається безпосередньо з машинного коду без службових заголовків – є характерною ознакою payload-у, доставленого через Reflective DLL Injection або подібну техніку рефлективного завантаження. Підтвердження належності виявленого артефакту до сімейства Meterpreter буде отримано на наступному етапі через статичний аналіз вмісту регіону і зовнішню валідацію.

Для переходу від етапу виявлення аномалій до безпосереднього вилучення даних з метою їх аналізу утилітою strings було застосовано плагін windows.malfind із параметром –dump (рисунок 7). Це дозволило не лише остаточно локалізувати підозрілі регіони пам’яті з атрибутами PAGE_EXECUTE_READWRITE, які були ідентифіковані раніше, а й автоматично зберегти їхній повний вміст у бінарному форматі. Такий підхід забезпечує підготовку знайдених артефактів у процесі explorer.exe до поглибленого вивчення поза межами середовища Volatility, фіксуючи стан підозрілих ділянок на момент зняття дампа.

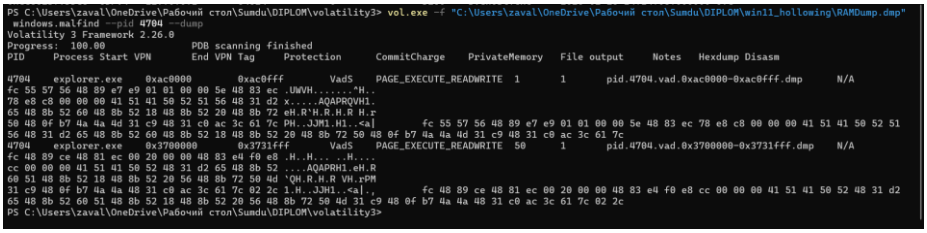

Рис. 7. Вивантаження підозрілих ділянок пам'яті процесу explorer.exe за допомогою плагіна windows.malfind

Витягнутий регіон 0x03700000-0x03731fff обсягом 200 КБ підданий статичному аналізу утилітою strings зі складу Sysinternals [21]. У результаті виявлено цілий ряд показових ознак: характерні для Meterpreter послідовності кодованих регістрів (AYAVAUATAUAVAWH), літеральне посилання на провайдер Winsock «MSAFD Tcpip [TCP/IP]», класичний DOS stub PE-заголовка «!This program cannot be run in DOS mode.», а також рядки бібліотек zlib (inflate/deflate). Найбільш значущим відкриттям, зробленим за допомогою strings, є наявність у вмісті регіону літерального рядка «tcp://192.168.50.218:4444» (рисунок 8) – жорстко закодованої адреси командно-керуючого сервера

атаки. Адреса 192.168.50.218 точно збігається з IP-адресою машини зловмисника, а порт 4444 – з параметром LPORT, заданим при генерації payload. Виявлення жорстко закодованої C2-адреси є практично абсолютним доказом зловмисної природи артефакту.

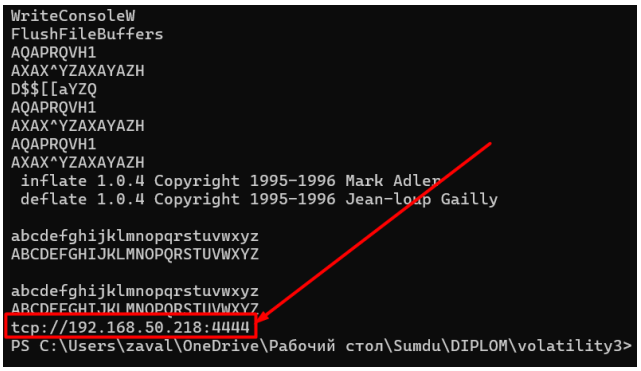

Рис. 8. Виявлення жорстко закодованої адреси C2-сервера у вмісті регіону пам'яті

Зовнішня валідація через VirusTotal. Для забезпечення незалежної верифікації висновків, отриманих засобами Volatility 3 та статичним аналізом, бінарний файл pid.4704.vad.0x3700000-0x3731fff.dmp завантажено на платформу VirusTotal [17] – публічний сервіс агрегації результатів сканування файлів більш ніж шістдесятьма антивірусними двигунами провідних вендорів галузі. Перевага такого підходу полягає в тому, що VirusTotal надає об'єктивну, многосторонню оцінку файлу і не залежить від конкретної реалізації або налаштувань будь-якого окремого антивіруса.

За результатами сканування (рисунок 9), 17 з 62 антивірусних двигунів ідентифікували завантажений файл як шкідливий, що становить рівень виявлення приблизно 27,4 %. Найбільш інформативною є мітка Popular threat label: shellcode/marte, присвоєна платформою на основі консенсусу детектів. Сімейство «marte» у вендорних класифікаціях відповідає шеллкодам Meterpreter та інших фреймворків для тестування на проникнення. Конкретні назви детектів повністю узгоджуються з цією класифікацією: BitDefender і ALYac ідентифікують файл як Generic.ShellCode.Marte.4.291CD0A3, ClamAV – як Win.Exploit.D38ba-9756522-0, Cynet присвоює максимальний рівень небезпеки (Malicious, score: 99). Сам факт ідентифікації витягнутого з пам'яті фрагмента 17 незалежними антивірусними механізмами як шкідливого є остаточним підтвердженням висновків, зроблених на попередньому етапі засобами Volatility 3. Обчислені платформою хеш-значення артефакту мають такі значення: SHA-256 – 2123eb2639d5cec8bae23c1754eb2402d1993cc9421644b62612d535327359eb. Розмір файлу – точно 200,00 КБ (204800 байтів), що ще раз підтверджує отримане раніше значення розміру другого RWX-регіону.

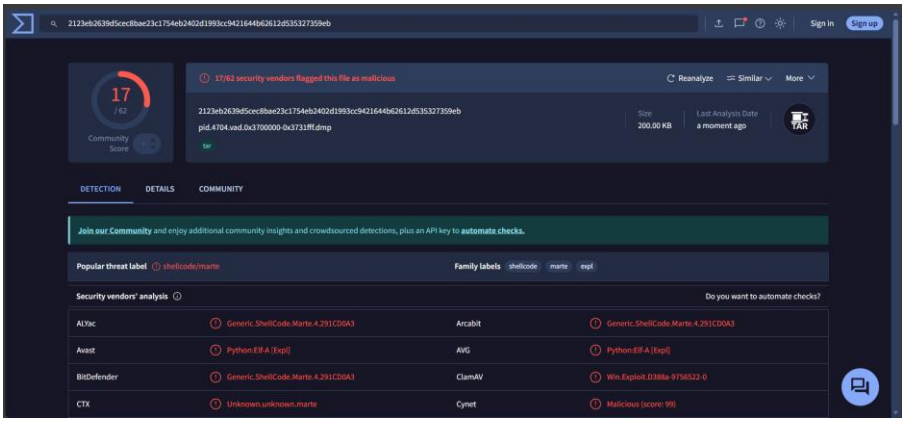

Рис. 9. Результати сканування витягнутого артефакту платформою VirusTotal

Порівняльний аналіз з еталонним дампом. Принциповим аргументом, що виключає можливі хибнопозитивні спрацювання та однозначно прив'язує виявлені артефакти до конкретного інциденту, є демонстрація відсутності цих самих артефактів у чистому (еталонному) дампі. Виконання плагіна windows.malfind для процесу explorer.exe (PID 744 у чистому дампі – інший, ніж 4704 у забрудненому, оскільки PID процесів призначаються ядром Windows у порядку їх створення і не зберігаються між перезавантаженнями системи) не виявило жодного RWX-регіону у адресному просторі процесу. Це

означає, що в штатному, незараженому стані легітимний процес explorer.exe не містить ні приватних виконуваних регіонів, ні характерних артефактів Reflective DLL Injection. Узагальнене зіставлення ключових артефактів обох дампів наведено у таблиці 4.

Таблиця 4

Зіставлення ключових артефактів у чистому та забрудненому дампах

Параметр/артефакт	Чистий дамп	Забруднений дамп
Час старту системи (UTC)	2026-02-25 21:18:48	2026-02-26 14:13:51
PID процесу explorer.exe	744	4704
Кількість потоків explorer.exe	86	97 (+11)
RWX-регіони (malfind)	0	2
VadS-вузли з RWX	0	2 (0xac0000, 0x3700000)
Тип VAD-вузла RWX-регіонів	–	VadS, поле File: Disabled
Жорстко закодована C2-адреса	Не виявлено	tcp://192.168.50.218:4444
Детект на VirusTotal	–	17/62 (shellcode/marte)

Як видно з таблиці 4, у забрудненому дампі зафіксовано стійкий комплекс відхилень від еталонного стану: збільшення кількості потоків explorer.exe на одинадцять, поява двох RWX-регіонів типу VadS без зв'язку з файлами на диску, наявність позиційно-незалежного коду з характерним shellcode-прологом, жорстко закодована адреса командно-керуючого сервера та підтвердження зловмисності 17 незалежними антивірусними двигунами.

Зіставлення з матрицею MITRE ATT&CK. Виявлені артефакти доцільно систематизувати за тактиками і техніками матриці MITRE ATT&CK – глобально визнаної класифікаційної системи, що дозволяє стандартизовано описувати поведінку зловмисників та забезпечує єдиний понятійний апарат для комунікації з іншими фахівцями галузі. Така прив’язка дозволяє віднести спостережений інцидент до відомих категорій загроз та полегшує подальше формування індикаторів компрометації, правил детектування і рекомендацій щодо протидії. Узагальнений мапінг наведений у таблиці 5.

Таблиця 5

Мапінг виявлених артефактів на матрицю MITRE ATT&CK

Тактика	Техніка (ID)	Спостережений артефакт	Засіб виявлення
Initial Access	T1105 (Ingress Tool Transfer)	diplom_test.exe доставлений через спільну папку	Аналіз метаданих Magnet RESPONSE
Execution	T1204.002 (User Execution: Malicious File)	Запуск шкідливого файлу користувачем	Журнали подій (Event ID 4688)
Defense Evasion	T1055.001 (Process Injection: DLL Injection)	RWX-регіони в адресному просторі explorer.exe	Volatility 3: malfind, vadinfo
Defense Evasion	T1620 (Reflective Code Loading)	Позиційно-незалежний код у приватному регіоні (VadS) без зв'язку з файлом на диску	Volatility 3 + аналіз hex-дампів
Defense Evasion	T1562.001 (Disable or Modify Tools)	Відключення Real-time protection Microsoft Defender	Аналіз стану системи
Command& Control	T1071.001 (Application Layer Protocol)	Жорстко закодована адреса tcp://192.168.50.218:4444	Volatility 3: strings, vaddump

Обговорення. Запропонована методика має визначені обмеження. По-перше, вона орієнтована переважно на виявлення ін'єкції коду в користувацькому режимі; для протидії руткітам режиму ядра з модифікацією Kernel Patch Protection або з використанням гіпервізорних технік (BYOVD — Bring Your Own Vulnerable Driver) необхідне додаткове застосування плагінів cmdscan, modules, ssdt, callbacks та аналізу Code Integrity. По-друге, відносно невисокий рівень виявлення на VirusTotal (27,4 %) свідчить, що навіть штатний Meterpreter не завжди ідентифікується класичними антивірусами, а отже зовнішня валідація має виконувати додаткову роль — наприклад, через YARA-сканування з власними правилами або інтеграцію з платформами кіберрозвідки типу MISP та AlienVault OTX. По-третє, дослідження проведене на ізольованому стенді з фіксованою конфігурацією; перенесення методики на реальні системи потребує додаткових перевірок щодо часу збору дампу та обсягу пам'яті, яка може досягати десятків гігабайтів. Перспективним напрямом удосконалення є інтеграція методики з підходами на основі машинного навчання, описаними у роботах [8, 9], що дозволить автоматизувати класифікацію виявлених артефактів за родинами шкідливого ПЗ.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У результаті проведеного дослідження сформульовано та експериментально верифіковано методику виявлення прихованих процесів і ін'єктованого шкідливого коду в дампах оперативної пам'яті систем Windows 11. Методика побудована як послідовний шестиетапний процес – підготовка, збір волатильних даних, тріаж, поглиблений аналіз у трьох паралельних гілках, крос-верифікація через зовнішні платформи кіберрозвідки та звітування – і базується на використанні фреймворку Volatility 3 у поєднанні з інструментом збору Magnet RESPONSE.

Експериментальна верифікація на сценарії контрольованої атаки з використанням Metasploit Meterpreter та подальшої міграції коду в процес explorer.exe Windows 11 продемонструвала ефективність методики: успішно ідентифіковано стійкий комплекс індикаторів компрометації, серед яких два приватні RWX-регіони пам'яті типу VadS без зв'язку з файлами на диску, характерний x86-64 пролог shellcode-у з типовим патерном звертання до PEB через сегмент GS, асиметрія VAD-дерева як ознака ін'єкції, а також жорстко закодована адреса командно-керуючого сервера, виявлена через статичний аналіз витягнутого регіону. Зовнішня валідація через VirusTotal підтвердила належність артефакту до сімейства shellcode/marte з рівнем виявлення 17 з 62 антивірусних двигунів. Порівняльний аналіз з еталонним дампом однозначно довів, що виявлені аномалії є наслідком модельованої атаки, а не фоновою активністю системи. Виявлені артефакти зіставлено з шістьма техніками матриці MITRE ATT&CK, що забезпечує сумісність висновків з міжнародною практикою інцидент-респонсу.

Наукова новизна одержаних результатів полягає у формалізації узагальненої методики виявлення прихованих процесів, що інтегрує етапи збору, тріажу, поглибленого аналізу, крос-верифікації та звітування з прив'язкою до матриці MITRE ATT&CK, на прикладі сучасної системи Windows 11. Практичне значення роботи полягає у можливості безпосереднього застосування методики у навчальному процесі підготовки фахівців з кібербезпеки бакалаврського та магістерського рівнів, у практиці інцидент-респонсу комерційних та урядових SOC, а також як методологічної основи для подальших наукових розвідок.

Перспективні напрями подальших досліджень включають: розширення методики на сценарії з ескалацією привілеїв та credential dumping (Mimikatz); адаптацію до інших операційних систем (Linux, macOS); інтеграцію з інструментами автоматизованого триажу та EDR-платформами; застосування методів машинного навчання – зокрема CNN та RNN, що демонструють точність до 97,8 % за даними [9], – для класифікації виявлених артефактів за родинами шкідливого ПЗ; розробку автоматизованого мапінгу артефактів на матрицю MITRE ATT&CK із використанням великих мовних моделей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pillai, R., Talreja, S. R., & Pamarthi, V. (2024). Memory forensics using the Volatility framework: A structured approach for detecting fileless malware. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE. https://ieeexplore.ieee.org/document/11323993
2. Check Point Research. (2024). Q2 2024 cyber attacks report. Check Point Software Technologies. https://blog.checkpoint.com/research/
3. MITRE ATT&CK. (n.d.). Process injection (Technique T1055). MITRE. https://attack.mitre.org/techniques/T1055/

4. Cyber Forensics Academy. (2025). Volatility framework: Complete memory forensics guide. https://www.cyberforensicacademy.com/blog/volatility-framework-complete-memory-forensics-guide
5. Dolan-Gavitt, B. (2007). The VAD tree: A process-eye view of physical memory. Digital Investigation, 4(1), 62–64. https://doi.org/10.1016/j.diin.2007.06.008
6. Fewer, S. (2008). Reflective DLL injection. Harmony Security White Paper. https://www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf
7. Microsoft Defender Research Team. (2025). Detecting reflective DLL loading with Windows Defender ATP. Microsoft Security Blog. Microsoft. https://www.microsoft.com/en-us/security/blog/2017/11/13/detecting-reflective-dll-loading-with-windows-defender-atp/
8. Dehfouli, Y., & Habibi Lashkari, A. (2025). Memory analysis for malware detection: A comprehensive survey using the OSCAR methodology. ACM Computing Surveys. https://doi.org/10.1145/3764580
9. Odeh, A., Taleb, A. A., Alhajahjeh, T., & Navarro, F. (2025). Advanced memory forensics for malware classification with deep learning algorithms. Cluster Computing, 28(6). https://doi.org/10.1007/s10586-025-05104-7
10. Case, A. (2025). Analysis of sedexp Linux malware: Memory-only rootkit and anti-forensics techniques. From the Source 2025. The Volatility Foundation. https://volatilityfoundation.org/from-the-source-2025/
11. TrustedSec. (2024). Loading DLLs reflections. TrustedSec Blog. https://trustedsec.com/blog/loading-dlls-reflections
12. Cynet. (2026). Process injection techniques. https://www.cynet.com/attack-techniques-hands-on/process-injection-techniques/
13. Pen Test Partners. (2024). Using Volatility for advanced memory forensics. Pen Test Partners Blog. https://www.pentestpartners.com/security-blog/using-volatility-for-advanced-memory-forensics/
14. Solomon, D. A., Russinovich, M. E., Ionescu, A., & Yosifovich, P. (2017). Windows internals. Part 1: System architecture, processes, threads, memory management, and more (7th ed.). Microsoft Press.
15. Faiz, M. N., & Prabowo, W. A. (2019). Comparison of acquisition software for digital forensics purposes. Kinetik, 4(1), 37–44. https://doi.org/10.22219/kinetik.v4i1.687
16. Volatility Foundation. (n.d.). Volatility 3 documentation: Plugin reference. https://volatility3.readthedocs.io/en/latest/
17. VirusTotal. (2025). Public API and scanning engine documentation. https://docs.virustotal.com/
18. Rapid7. (2025). Metasploit Framework: Documentation and module reference. https://docs.metasploit.com/
19. Magnet Forensics. (2024). Magnet RESPONSE: Free incident response tool. https://www.magnetforensics.com/resources/magnet-response/
20. MITRE Corporation. (2024). Process injection: Dynamic-link library injection (Technique T1055.001). MITRE ATT&CK. https://attack.mitre.org/techniques/T1055/001/
21. Microsoft. (2024). Strings v2.54. Windows Sysinternals. https://learn.microsoft.com/en-us/sysinternals/downloads/strings
22. Said Hamed, A. S., & Dewangan, O. (2025). Digital forensic: Techniques, challenges, and future direction. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2025.71562

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (34), 2026
ISSN 2663 – 4023

Stanislav Zavalii
Bachelor's degree student in specialty 125 «Cybersecurity and Information Protection»
Sumy State University, Sumy, Ukraine
ORCID:0009-0002-9651-776X
s.zavalij@student.sumdu.edu.ua

Vitaliy Koval
Ph.D., Senior Lecturer
Sumy State University, Sumy, Ukraine
ORCID:0000-0002-1593-5605
v.koval@dcs.sumdu.edu.ua

METHODS AND TOOLS FOR FORENSIC MEMORY ANALYSIS TO DETECT HIDDEN PROCESSES IN COMPUTER SYSTEMS

Abstract. The article addresses the problem of detecting hidden processes and injected malicious code in the volatile memory of computer systems running Microsoft Windows 11. The relevance of the research is driven by the rapid growth of fileless attacks, which fully unfold in RAM and leave no traces accessible to classical antivirus and EDR monitoring tools. According to industry reports, the volume of cyberattacks increased by 30 % in Q2 2024, with a significant share constituted by fileless and memory-resident campaigns. The aim of the article is to formalise a methodology for detecting hidden processes through analysis of memory structures – Virtual Address Descriptor (VAD) tree and page organisation – using the Volatility 3 framework with subsequent external validation of the obtained results. The paper systematises the theoretical foundations of Windows virtual memory organisation, performs a comparative analysis of six memory acquisition tools and four analysis frameworks, and traces the evolution of code-hiding techniques. The proposed methodology is designed as a sequential six-stage process: preparation, volatile data acquisition, triage, deep analysis (in three parallel branches: process analysis, memory analysis, and network analysis), cross-verification through external cyberintelligence platforms, and reporting with chain-of-custody documentation. The methodology is integrated with NIST SP 800-86, IETF RFC 3227, and MITRE ATT&CK standards. Experimental verification of the methodology has been performed on a controlled attack scenario using the Metasploit Meterpreter framework with subsequent migration of the malicious code into the address space of the legitimate Windows 11 explorer.exe process via the Reflective DLL Injection technique (T1055.001 in MITRE ATT&CK classification). A robust set of compromise indicators has been identified: two private RWX memory regions, a characteristic x86-64 shellcode prologue with the typical PEB-traversal pattern through the GS segment, the asymmetry of the VAD tree with the absence of file-on-disk linkage for the anomalous regions, and a hardcoded address of the command-and-control server. Independent validation through the VirusTotal platform has confirmed the artefact's attribution to the shellcode/marte family with a detection rate of 17 out of 62 antivirus engines. Comparative analysis with the reference (clean) dump has unequivocally proven that the detected anomalies result from the simulated attack rather than background system activity. The detected artefacts are mapped onto six MITRE ATT&CK matrix techniques: T1105, T1204.002, T1055.001, T1620, T1562.001, and T1071.001, ensuring compatibility of conclusions with the international incident response practice. The scientific novelty consists in the formalisation of a generalised methodology for detecting hidden processes that integrates collection, triage, deep analysis, cross-verification, and reporting stages with explicit MITRE ATT&CK mapping, applied to modern Windows 11 systems. The practical significance of the work lies in the methodology's readiness for direct application in cybersecurity training at bachelor and master levels, in commercial and governmental SOC incident response practice, and as a methodological basis for further research in volatile memory forensics. Prospective research directions include extending the methodology to privilege escalation and credential dumping scenarios, adaptation to Linux and macOS platforms, integration with EDR platforms, and the application of deep learning methods for automated classification of detected artefacts.

Keywords: memory forensics; hidden processes; Reflective DLL Injection; Meterpreter; Volatility 3; Magnet RESPONSE; fileless attacks; Windows 11; VAD tree; MITRE ATT&CK; digital forensics.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Pillai, R., Talreja, S. R., & Pamarthi, V. (2024). Memory forensics using the Volatility framework: A structured approach for detecting fileless malware. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE. https://ieeexplore.ieee.org/document/11323993
2. Check Point Research. (2024). Q2 2024 cyber attacks report. Check Point Software Technologies. https://blog.checkpoint.com/research/
3. MITRE ATT&CK. (n.d.). Process injection (Technique T1055). MITRE. https://attack.mitre.org/techniques/T1055/
4. Cyber Forensics Academy. (2025). Volatility framework: Complete memory forensics guide. https://www.cyberforensicacademy.com/blog/volatility-framework-complete-memory-forensics-guide
5. Dolan-Gavitt, B. (2007). The VAD tree: A process-eye view of physical memory. Digital Investigation, 4(1), 62–64. https://doi.org/10.1016/j.diin.2007.06.008
6. Fewer, S. (2008). Reflective DLL injection. Harmony Security White Paper. https://www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf
7. Microsoft Defender Research Team. (2025). Detecting reflective DLL loading with Windows Defender ATP. Microsoft Security Blog. Microsoft. https://www.microsoft.com/en-us/security/blog/2017/11/13/detecting-reflective-dll-loading-with-windows-defender-atp/
8. Dehfouli, Y., & Habibi Lashkari, A. (2025). Memory analysis for malware detection: A comprehensive survey using the OSCAR methodology. ACM Computing Surveys. https://doi.org/10.1145/3764580
9. Odeh, A., Taleb, A. A., Alhajahjeh, T., & Navarro, F. (2025). Advanced memory forensics for malware classification with deep learning algorithms. Cluster Computing, 28(6). https://doi.org/10.1007/s10586-025-05104-7
10. Case, A. (2025). Analysis of sedexp Linux malware: Memory-only rootkit and anti-forensics techniques. From the Source 2025. The Volatility Foundation. https://volatilityfoundation.org/from-the-source-2025/
11. TrustedSec. (2024). Loading DLLs reflections. TrustedSec Blog. https://trustedsec.com/blog/loading-dlls-reflections
12. Cynet. (2026). Process injection techniques. https://www.cynet.com/attack-techniques-hands-on/process-injection-techniques/
13. Pen Test Partners. (2024). Using Volatility for advanced memory forensics. Pen Test Partners Blog. https://www.pentestpartners.com/security-blog/using-volatility-for-advanced-memory-forensics/
14. Solomon, D. A., Russinovich, M. E., Ionescu, A., & Yosifovich, P. (2017). Windows internals. Part 1: System architecture, processes, threads, memory management, and more (7th ed.). Microsoft Press.
15. Faiz, M. N., & Prabowo, W. A. (2019). Comparison of acquisition software for digital forensics purposes. Kinetik, 4(1), 37–44. https://doi.org/10.22219/kinetik.v4i1.687
16. Volatility Foundation. (n.d.). Volatility 3 documentation: Plugin reference. https://volatility3.readthedocs.io/en/latest/
17. VirusTotal. (2025). Public API and scanning engine documentation. https://docs.virustotal.com/
18. Rapid7. (2025). Metasploit Framework: Documentation and module reference. https://docs.metasploit.com/
19. Magnet Forensics. (2024). Magnet RESPONSE: Free incident response tool. https://www.magnetforensics.com/resources/magnet-response/
20. MITRE Corporation. (2024). Process injection: Dynamic-link library injection (Technique T1055.001). MITRE ATT&CK. https://attack.mitre.org/techniques/T1055/001/
21. Microsoft. (2024). Strings v2.54. Windows Sysinternals. https://learn.microsoft.com/en-us/sysinternals/downloads/strings
22. Said Hamed, A. S., & Dewangan, O. (2025). Digital forensic: Techniques, challenges, and future direction. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2025.71562

Отримано редакцією журналу / Received: 25.01.26
Прорецензовано / Revised: 06.02.26
Схвалено до друку / Accepted: 24.09.26