

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

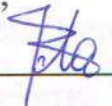
ПОЯСНЮВАЛЬНА ЗАПISKA

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект підсистеми моніторингу кінцевих вуз-
лів організації та аналізу цифрових артефактів на базі
платформи Velociraptor»

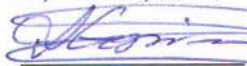
Виконав: студент 4-го курсу групи КБ-22-1
спеціальності 125 «Кібербезпека»

 Максим КУЛШ

Керівник: асистент кафедри комп'ютерної інженерії та
кібербезпеки,

 Артем ЛАЙЧУК

Рецензент: завідувач кафедри комп'ютерної інженерії та
кібербезпеки, к.т.н., доцент,

 Андрій ЄФІМЕНКО

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

02 лютого 2026 р.

ЗАВДАННЯ

на випускн у кваліфікаційну роботу бакалавра

Студент: Куліш Максим Віталійович

Тема роботи: «Проект підсистеми моніторингу кінцевих вузлів організації та аналізу цифрових артефактів на базі платформи Velociraptor»

затверджена Наказом університету 02 лютого 2026 р. №58/с

Термін здачі студентом закінченої роботи 08 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

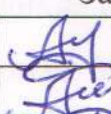
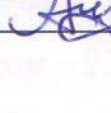
Предметом дослідження є методи та технології проактивної цифрової криміналістики (Digital Forensics and Incident Response – DFIR), архітектура та функціональні можливості платформи Velociraptor, методологія збору, обробки та аналізу цифрових артефактів на базі мови VQL, а також стратегії полювання за загрозами (Threat Hunting) з використанням матриці MITRE ATT&CK.

Об'єктом дослідження є інформаційна інфраструктура організації (корпоративний/державний сегмент), зокрема мережевий сегмент кінцевих вузлів під керуванням операційних систем сімейства Windows.

Обладнання та програмне забезпечення:

1. Velociraptor v0.75.6.
2. Oracle VirtualBox.
3. ОС Microsoft Windows 10 та Microsoft Windows 7.
4. ОС Kali Linux та Linux Ubuntu.

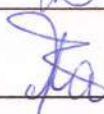
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Шелуха О.О.	 02.02.2026	02.02.2026 
2	Шелуха О.О.	 02.02.2026	02.02.2026 
3	Шелуха О.О.	 02.02.2026	02.02.2026 

Студент  Максим КУЛШ
Керівник  Артем ЛАЙЧУК

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз отриманого завдання та уточнення поставлених завдань	02.02.2026 р.	Виконано
2	Аналіз літературних джерел	01.05.2026 р.	Виконано
3	Аналіз об'єкта дослідження	14.05.2026 р.	Виконано
4	Постановка задачі дослідження	19.05.2026 р.	Виконано
5	Розробка структурної моделі	23.05.2026 р.	Виконано
6	Вибір технології для реалізації	25.05.2026 р.	Виконано
7	Вибір методів захисту	26.05.2026 р.	Виконано
8	Побудова тестової мережі	30.05.2026 р.	Виконано
9	Налаштування тестової мережі	31.05.2026 р.	Виконано
10	Перевірка працездатності	01.06.2026 р.	Виконано
11	Проведення дослідження безпеки	02.06.2026 р.	Виконано
12	Оформлення пояснювальної записки	08.06.2026 р.	Виконано
13	Захист		

Студент  Максим КУЛШ
Керівник  Артем ЛАЙЧУК

АНОТАЦІЯ

Випускна кваліфікаційна робота бакалавра складається з пояснювальної записки, що містить вступ, три розділи, висновки, список використаної літератури та додатки. Текстова частина викладена на 67 сторінках друкованого тексту. Пояснювальна записка має 3 додатки на 8 сторінках. Список використаних джерел містить 27 найменувань і займає 4 сторінки. В роботі наведено 61 рисунок та 5 таблиць. Загальний обсяг роботи – 93 сторінок.

Метою кваліфікаційної роботи є проектування та практичне розгортання підсистеми моніторингу кінцевих вузлів організації, а також автоматизація процесу збору та аналізу цифрових артефактів для задач реагування на інциденти кібербезпеки на базі платформи Velociraptor.

В рамках роботи було розглянуто теоретичні основи цифрової криміналістики та реагування на інциденти (DFIR), класифікацію ключових цифрових артефактів операційної системи Windows та методики оцінки стану безпеки хостів. Окрему увагу приділено клієнт-серверній архітектурі платформи Velociraptor та мові запитів VQL (Velociraptor Query Language). Виконано практичне проектування підсистеми, розгортання центрального сервера та агентів моніторингу, а також розробку та тестування спеціалізованих VQL-артефактів для виявлення ознак закріплення шкідливого ПЗ, аналізу системних логів та низькорівневого дослідження файлової системи.

КЛЮЧОВІ СЛОВА: ЦИФРОВА КРИМІНАЛІСТИКА, ФОРЕНЗИКА КІНЦЕВИХ ВУЗЛІВ, VELOCIRAPTOR, VQL, ЦИФРОВІ АРТЕФАКТИ, РЕАГУВАННЯ НА ІНЦИДЕНТИ.

					ККІК.КР.Б – 125 – 07 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми моніторингу кінцевих вузлів організації та аналізу цифрових артефактів на базі платформи Velociraptor	Лім.	Арк.	Аркуші	
Розроб.		Куліш М.В.		16.06.26					
Керівник		Лайчук А.А.		16.06.26			4	90	
Реценз.		Єфіменко А.А.		16.06.26					
Н. контр.		Лайчук А.А.		16.06.26					
Затверд.		Єфіменко А.А.		16.06.26					
						ФІКТ, Гр. КБ-22-1			

ANNOTATION

The final qualification work of the bachelor consists of an explanatory note, including an introduction, three chapters, conclusions, a list of references, and appendices. The text part is set out on 67 pages of printed text. The explanatory note includes 3 appendices on 8 pages. The list of references contains 27 sources and occupies 4 pages. The work contains 61 figures and 5 tables. The total volume of the work is 93 pages.

The purpose of this qualification work is to design and practically deploy an endpoint monitoring subsystem for an organization, as well as to automate the collection and analysis of digital artifacts for incident response tasks based on the Velociraptor platform.

Within the framework of the work, the theoretical foundations of digital forensics and incident response (DFIR), the classification of key Windows OS digital artifacts, and methods for assessing the security posture of endpoints were examined. Special attention was given to the client-server architecture of the Velociraptor platform and the VQL (Velociraptor Query Language). A practical design of the subsystem was performed, including the deployment of the central server and monitoring agents, as well as the development and testing of specialized VQL artifacts to detect malware persistence mechanisms, analyze system logs, and conduct low-level file system examination.

KEYWORDS: DIGITAL FORENSICS, ENDPOINT MONITORING, VELOCIRAPTOR, VQL, DIGITAL ARTIFACTS, INCIDENT RESPONSE.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ МОНІТОРИНГУ КІНЦЕВИХ ВУЗЛІВ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ	11
1.1 Аналіз концепції Threat Hunting та забезпечення видимості кінцевих вузлів (Endpoint Visibility).....	11
1.2 Аналіз архітектури платформи Velociraptor.....	16
1.3 Аналіз Velociraptor Query Language.....	20
1.4 Порівняльний аналіз існуючих SIEM/DFIR/EDR/XDR рішень та ключова різ- ниця між ними.....	26
1.5 Класифікація та безпековий потенціал критичних цифрових артефактів опе- раційної системи Windows (Prefetch, Amcache, Shimcache, USN Journal).....	30
РОЗДІЛ 2 ПРОЄКТУВАННЯ ПІДСИСТЕМИ МОНІТОРИНГУ.....	37
2.1 Аналіз інфраструктури організації та вибір топології розгортання підсистеми (Local / Cloud / Hybrid).....	38
2.2 Розробка схеми мережевої взаємодії та забезпечення безпеки каналів зв'язку на основі взаємної автентифікації (mTLS та PKI).....	43
2.3 Моделювання загроз для кінцевих вузлів на основі матриці MITRE ATT&CK та формування стратегій полювання на загрози.....	47
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ ПІДСИСТЕМИ НА БАЗІ VELOCIRAPTOR.....	50
3.1 Підготовка до розгортання тестового лабораторного середовища	50
3.2 Практичне розгортання сервера Velociraptor та інсталяція агентів моніторингу на кінцевих пристроях.....	52
3.3 Збір та аналіз доказів компрометації у вигляді спеціалізованих VQL-артефак- тів.....	65

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВИСНОВКИ.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	81
ДОДАТКИ.....	85

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CSV – Comma-Separated Values

DFIR – Digital Forensics and Incident Response

DNS – Domain Name System

EDR – Endpoint Detection and Response

gRPC – Google Remote Procedure Call

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

IoC – Indicator of Compromise

IP – Internet Protocol

JSON – JavaScript Object Notation

MD5 – Message Digest Algorithm 5

MFT – Master File Table

mTLS – Mutual Transport Layer Security

PKI – Public Key Infrastructure

SHA – Secure Hash Algorithm

SIEM – Security Information and Event Management

SOC – Security Operations Center

TCP – Transmission Control Protocol

TLS – Transport Layer Security

UDP – User Datagram Protocol

VQL – Velociraptor Query Language

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасне кібербезпекове середовище характеризується стрімким зростанням складності та цілеспрямованості кібератак (APT, програми-вимагачі, безфайлове шкідливе ПЗ), які легко обходять традиційні периметрові засоби захисту. У цих умовах компрометація кінцевих вузлів (робочих станцій та серверів) організацій стає основним вектором проникнення зловмисників. Традиційні антивірусні рішення часто виявляються неефективними проти новітніх тактик закріплення та прихованого збору даних, що зумовлює гостру потребу впровадження концепцій Threat Hunting (активного полювання на загрози) та DFIR (цифрової криміналістики та реагування на інциденти).

Забезпечення видимості інфраструктури (Endpoint Visibility) та оперативний збір низькорівневих цифрових артефактів операційної системи є критично важливими для своєчасного локалізування інцидентів. Традиційний централізований збір логів (SIEM) потребує значних ресурсів і часто не дає змоги провести глибокий криміналістичний аналіз оперативно. Використання сучасних open-source інструментів інженерії безпеки, зокрема платформи Velociraptor, відкриває нові можливості завдяки застосуванню мови запитів VQL (Velociraptor Query Language). Це дозволяє здійснювати хірургічно точний, масштабований та швидкий збір артефактів безпосередньо з тисяч кінцевих пристроїв одночасно. З огляду на це, проєктування та практичне розгортання ефективної підсистеми моніторингу на базі Velociraptor для вітчизняних підприємств та організацій є надзвичайно актуальним науково-практичним завданням.

Метою кваліфікаційної роботи є проєктування, практичне розгортання та дослідження ефективності підсистеми моніторингу кінцевих вузлів організації, а також автоматизація процесів збору та аналізу цифрових артефактів на базі платформи Velociraptor для підвищення загального рівня захищеності та оперативності реагування на інциденти.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Дослідити теоретичні основи моніторингу кінцевих вузлів, концепцію Threat Hunting та роль цифрових артефактів у сучасній системі кіберзахисту.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Проаналізувати архітектуру, компоненти та принципи функціонування платформи Velociraptor, а також синтаксис і можливості мови запитів VQL.

3. Дослідити безпековий потенціал та структуру основних цифрових артефактів ОС Windows (Prefetch, Amcache, Shimcache, USN Journal) як джерел доказів компрометації.

4. Здійснити моделювання загроз для інфраструктури організації на основі фреймворку MITRE ATT&CK та спроектувати топологію підсистеми моніторингу із забезпеченням безпеки каналів зв'язку на базі mTLS та PKI.

5. Практично розгорнути сервер підсистеми, автоматизувати процес інсталяції агентів на кінцевих пристроях та розробити (skonфігурувати) спеціалізовані VQL-артефакти для виявлення аномальної активності.

6. Провести тестування розробленої підсистеми шляхом симуляції типових сценаріїв кібератак, оцінити повноту і швидкість збору цифрових доказів та сформулювати рекомендації щодо її експлуатації.

Об'єктом дослідження є процеси моніторингу стану безпеки кінцевих вузлів та збору (аналізу) цифрових артефактів в інформаційних системах організацій в інтересах протидії актуальним кіберзагрозам.

Предметом дослідження є архітектурні рішення, методи, алгоритми та інструментальні мовні засоби автоматизованого збору цифрових артефактів операційної системи та виявлення індикаторів компрометації на базі Velociraptor.

Результатом виконання кваліфікаційної роботи є спроектована та практично реалізована підсистема моніторингу кінцевих вузлів організації на основі клієнт-серверної архітектури Velociraptor. У ході роботи розгорнуто тестову IT-інфраструктуру, налаштовано захищену взаємодію між сервером та агентами, створено комплекс спеціалізованих VQL-скриптів для виявлення ознак закріплення зловмисників (Persistence), створення несанкціонованих процесів та модифікації критичних файлів. Ефективність підсистеми підтверджено експериментально під час симуляцій атакуючих дій, що дозволило автоматизувати збір цифрових артефактів та скоротити час первинного реагування на інциденти (Incident Response).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ МОНІТОРИНГУ КІНЦЕВИХ ВУЗЛІВ ТА ЦИФРОВОЇ КРИМІНАЛІСТИКИ

1.1 Аналіз концепції Threat Hunting та забезпечення видимості кінцевих вузлів (Endpoint Visibility).

Постійне ускладнення ландшафту кіберзагроз та поява цільових таргетованих атак (APT) зумовили перехід парадигми забезпечення інформаційної безпеки від реактивної до проактивної. Проактивний пошук загроз (Threat Hunting), також відомий як кіберполювання – це цілеспрямований, ітеративний та керований аналітиками процес інспектування комп'ютерних систем та мережевих сегментів з метою виявлення прихованих шкідливих дій, які змогли успішно обійти традиційні автоматизовані превентивні засоби захисту [1]. У науковій та спеціалізованій літературі Threat Hunting визначається як стратегічний підхід, який не базується на очікуванні автоматичних сповіщень (алертів) від засобів захисту, а полягає у безперервному висуненні та перевірці гіпотез щодо можливої компрометації інфраструктури [2], [3]. Згідно з дослідженнями провідних інституцій у сфері кібербезпеки, цей процес орієнтований на виявлення зловмисників, які вже закріпилися в мережі та намагаються мімікрувати під легітимну діяльність користувачів або системних адміністраторів [4], [5].

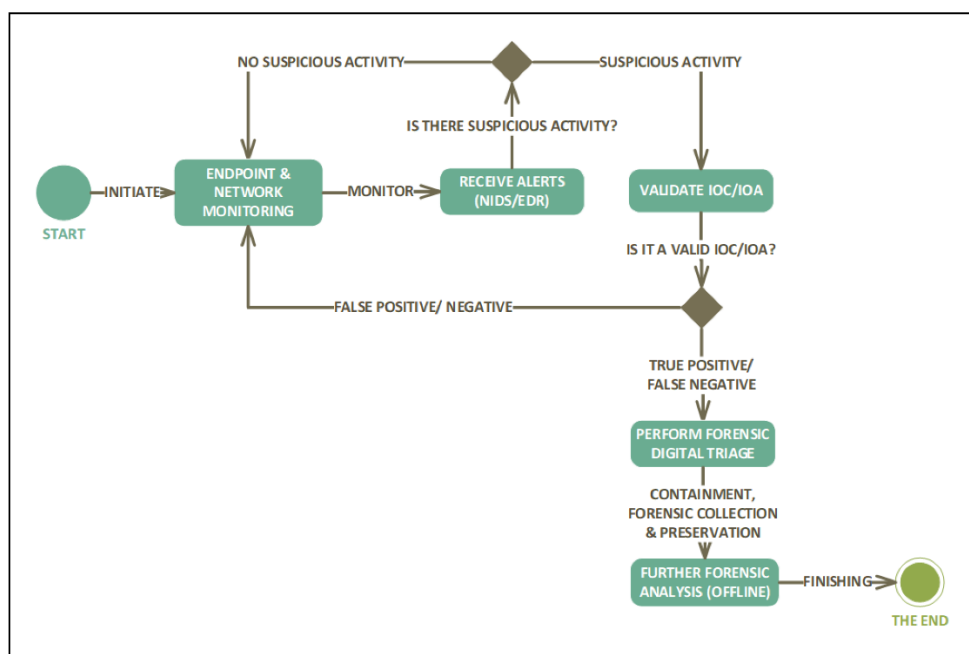


Рисунок 1.1 - Модель розгортання для проактивної цифрової криміналістики

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Використання низькорівневого аналізу артефактів операційних систем дозволяє фахівцям реконструювати повний контекст розвитку інциденту та забезпечити інструментарій для точкового стримування загроз на ранніх етапах розвитку атаки [2]. Сучасні інструменти автоматизованого збору телеметрії фіксують та аналізують стан кінцевих точок інфраструктури (хостів), забезпечуючи командам центрів моніторингу безпеки (SOC) та фахівцям з розслідування інцидентів (DFIR) критично необхідну видимість для виявлення аномальної активності, яка тривалий час може залишатися непоміченою. Процес кіберполювання передбачає комплексне відстеження стану файлових систем, об'єктів системного реєстру, активності процесів у пам'яті та мережевих з'єднань, що забезпечує як видимість поточних операцій у режимі реального часу, так і можливість ретроспективного криміналістичного аналізу історії хоста [1]. Важливим аспектом сучасного полювання за загрозами є концепція інженерії виявлення (Detection Engineering), яка трансформує результати успішних сесій кіберполювання у постійні оптимізовані правила детектування для корпоративних систем захисту [6]. Спеціалізовані програмні рішення для автоматизованого полювання повинні володіти високою масштабованістю, низьким рівнем утилізації системних ресурсів кінцевих точок та гнучкістю, що дозволяє виконувати експрес-опитування тисяч пристроїв одночасно за допомогою декларативних низькорівневих мов запитів (зокрема, Velociraptor Query Language – VQL) [7]. Такий підхід дає змогу ефективно порівнювати отримані артефакти з наявними тактиками, методами та процедурами (TTP) за міжнародною класифікацією MITRE ATT&CK, а також здійснювати швидкий пошук індикаторів компрометації (IoC) [8].

Ефективне практичне впровадження комплексу автоматизованого Threat Hunting вимагає детального проектування архітектури збору даних, розгортання оптимізованих агентів на кінцевих точках та регулярного оновлення пошукових запитів для мінімізації кількості хибних спрацьовувань. Необхідність глибокої видимості кінцевих вузлів обумовлена появою складних прихованих каналів передачі даних та методів маскування, які експлуатують легітимні протоколи та довірені корпоративні застосунки, залишаючись непомітними для сигнатурних

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

засобів захисту [9]. Розширені інженерні стратегії кіберполювання передбачають тісну інтеграцію локальних платформ збору цифрових доказів із зовнішніми модулями автоматизації, розробленими за допомогою мови програмування Python, та інтерфейсами програмування додатків (API) сучасних месенджерів [10]. Це дозволяє трансформувати складні сирі результати низькорівневого аналізу в інформативні структуровані алерти, що суттєво підвищує загальний рівень захищеності корпоративного контуру та скорочує час перебування загрози в мережі.

Операційний процес Threat Hunting включає чотири ключові етапи функціонування системи проактивного пошуку загрози. Етапи – збір цифрових артефактів, застосування проактивних запитів, аналіз контексту та тріаж, а також автоматизоване сповіщення та реагування [1].

Платформи кіберполювання орієнтовані на виявлення складних цільових кібератак, а також безфайлового шкідливого програмного забезпечення (Living off the Land), яке використовує легітимний системний інструментарій операційних систем для приховування своєї діяльності. Для реалізації цих завдань здійснюється таргетний збір телеметричних даних з кінцевих точок, який охоплює:

1. Структуру, метадані та метафайли файлових систем (Master File Table – MFT, USN Journal).
2. Стан, модифікації та ієрархічну структуру ключів системного реєстру Windows [8].
3. Журнали реєстрації подій операційної системи (Windows Event Logs, Sysmon) [2].
4. Сліди, артефакти та кеші запуску виконуваних файлів (Prefetch, Amcache, Shimcache) [11].

Організація точкового збору та парсингу зазначених джерел дозволяє виявити зловмисні відхилення від базового стану системи. Особлива цінність такого підходу полягає у здатності детектувати загрози на найнижчих рівнях взаємодії з ОС, де спроби видалення чи маскування слідів атаки самі по собі стають аномальними маркерами.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Технічний алгоритм функціонування сучасних комплексів кіберполювання складається з кількох взаємопов'язаних етапів, кожний з яких виконує специфічну роль у загальному контурі детектування загроз:

1. Збір та локальний парсинг цифрових артефактів (Artifact Collection): Клієнтські компоненти (агенти), розгорнуті на кінцевих точках, за командою керуючого сервера звертаються напряму до низькорівневих бінарних структур ОС [7]. Замість неефективної та ресурсомісткої передачі гігабайтів сирих файлів логів через мережу, агент здійснює високошвидкісний розбір (парсинг) структур безпосередньо в оперативній пам'яті хоста, що мінімізує навантаження на дискову підсистему та канали зв'язку [1].

2. Застосування проактивних фільтруючих запитів (Querying & Filtering): Отримані структуровані дані піддаються первинній аналітичній фільтрації безпосередньо на кінцевій точці за допомогою декларативних мов запитів. Цей етап дозволяє миттєво відсіювати гігабайти легітимної активності. Наприклад, аналіз запусків командних інтерфейсів (powershell.exe, cmd.exe) з нетиповими аргументами, енкодowanними командами Base64 або їх ініціалізація з тимчасових папок користувачів (Temp, AppData) відразу виокремлюється із загального масиву [8].

3. Аналіз контексту, кореляція та тріаж (Analysis & Triage): Відфільтровані компактні масиви даних збираються на централізованому сервері, де аналітик отримує доступ до консолідованої інформації. Система забезпечує можливість побудови наскрізних часових шкал (Timeline) подій на багатьох хостах одночасно, що дозволяє фахівцю чітко диференціювати дії зловмисника від легітимних операцій з адміністрування мережі та встановлювати першопричину (Root Cause) інциденту [11].

4. Автоматизоване сповіщення та оркестрація реагування (Alerting & Orchestration): Сучасні комплекси Threat Hunting інтегрують гнучкі модулі зовнішнього оповіщення [10]. Замість статичного очікування перевірки веб-панелей, критичні аномалії, виявлені в результаті виконання запитів, обробляються за допомогою фонових скриптів на базі Python і через захищений протокол HTTPS

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

перенаправляються на API зовнішніх сервісів. Це забезпечує мобільність аналітика та суттєво скорочує метрику середнього часу реагування (MTTR).

Впровадження автоматизованого комплексу збору артефактів є критично необхідним для сучасних організацій, оскільки зловмисники за наявності достатнього часу та ресурсів здатні обійти будь-які периметрові бар'єри. Можна виділити кілька ключових причин інтеграції Threat Hunting у загальну концепцію кібербезпеки підприємства:

1. Неефективність традиційних засобів проти безфайлових атак (Fileless Attacks). Класичні антивірусні рішення орієнтовані на пошук сигнатур файлів на диску, тоді як сучасні атаки все частіше використовують техніки Living off the Land, експлуатуючи довірені системні компоненти (LOLBins), які виявляються лише через аналіз артефактів виконання типу Shimcache та логів Sysmon [8].

2. Значний час прихованого перебування зловмисника в мережі (Dwell Time). Без проактивного пошуку хакери здатні перебувати в інфраструктурі місяцями, створюючи приховані облікові записи та канали витоку даних, оскільки їхня активність повністю мімікрує під дію легітимних користувачів [1].

3. Брак прозорості та деталізації при розслідуванні інцидентів. У разі настання деструктивної фази атаки (наприклад, активації програм-вимагачів) відсутність деталізованих цифрових доказів з низькорівневих джерел (MFT, USN Journal) залишає команду безпеки у «сліпій зоні», унеможливляючи встановлення вектору проникнення [7].

4. Потреба у швидкому таргетному доступі до сирих даних на вимогу. Під час розвитку інциденту критично важливо мати інструмент, здатний за один запит миттєво опитати тисячі хостів щодо наявності конкретного маркера (наприклад, специфічного хендла чи ключа реєстру), надаючи аналітику точну вибірку без перевантаження мережі [1].

5. Проблема перевантаження аналітиків сповіщеннями (Alert Fatigue). Традиційні SIEM-системи акумулюють терабайти подій, що призводить до пропуску важливих інцидентів аналітиками. Пошук на основі чітко сформульованих гіпотез

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

у Threat Hunting локалізує область дослідження та знижує навантаження на персонал [2].

6. Колосальна вартість та тривалість ручного відновлення інфраструктури. Без інструментів точкової локалізації меж компрометації адміністратори змушені проводити повну переінсталяцію ОС на всіх пристроях мережі, що зупиняє бізнес-процеси. Криміналістичні артефакти дозволяють локалізувати та зачистити лише уражені вузли [11].

Історично концепція Threat Hunting сформувалася як відповідь на обмеження реактивних систем моніторингу на початку 2010-х років, коли парадигма «захисту периметра» остаточно продемонструвала свою неспроможність проти складних цільових загроз [1]. Ранні підходи до збору цифрових доказів вимагали створення повних криміналістичних образів накопичувачів, що займало години і було непридатним для динамічних корпоративних мереж [11]. Технологічний прорив відбувся з появою та розвитком легковажних агентських платформ (зокрема, рішень класу Velociraptor), які запропонували концепцію використання експрес-запитів до кінцевих точок у режимі реального часу. Сучасний етап розвитку кіберполювання характеризується активною інтеграцією засобів збору телеметрії із гнучкими мовами програмування (Python) та хмарними API комунікаційних платформ, що дозволяє повністю автоматизувати рутинні операції тріажу та забезпечити безперервний, всеосяжний захист інформаційних ресурсів організації.

1.2 Аналіз архітектури платформи Velociraptor.

Платформа Velociraptor є високоефективним спеціалізованим інструментом класу Endpoint Visibility, Digital Forensics and Incident Response (DFIR), призначеним для проактивного пошуку загроз та низькорівневого аналізу стану кінцевих точок інфраструктури. На відміну від традиційних систем класу SIEM або важко-вагових XDR-рішень, які фокусуються на безперервному дзеркалюванні та агрегації сирих логів у централізовані бази даних, архітектурна концепція Velociraptor

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

базується на принципі відкладеного збору даних (on-demand forensics) [12]. Як зазначає розробник платформи Майк Коен, система залишає цифрові артефакти безпосередньо на кінцевих хостах і здійснює їхній цільовий пошук, парсинг та фільтрацію локально у режимі реального часу лише за запитом аналітика, що мінімізує мережеве навантаження та виключає потребу у високовартісних підсистемах індексації великих масивів даних [13].

Згідно з офіційною технічною документацією проєкту, архітектура Velociraptor спроектована за модульним клієнт-серверним принципом і складається з єдиного кросплатформного бінарного файлу, компільованого мовою Go, який залежно від конфігураційних прапорців може виконувати ролі різних системних компонентів [14]:

1. Клієнтський агент (Velociraptor Client): Легковажна служба (демон), яка встановлюється на цільових кінцевих точках інфраструктури (підтримуються ОС Windows, Linux, macOS). Агент функціонує з найвищими системними привілеями (SYSTEM або root), що необхідно для отримання прямого доступу до заблокованих файлів, об'єктів реєстру та оперативної пам'яті хоста. Клієнтський компонент не містить жорстко закодованої логіки детектування; замість цього він містить вбудований інтерпретатор мови VQL, який приймає декларативні скрипти від сервера, виконує їх локально у безпечній пісочниці й повертає виключно структурований результат фільтрації [13].

2. Фронтенд-сервер (Velociraptor Frontend): Центральний високопродуктивний вузол системи, який відповідає за автентифікацію підключених агентів, підтримання постійного зв'язку з ними, маршрутизацію та оркестрацію потоків масового полювання (Hunts) або індивідуальних запитів [9]. Фронтенд приймає вхідні з'єднання від тисяч клієнтів одночасно, розподіляє завдання та збирає кінцеві результати. Відсутність потреби у постійному потоковому парсингу важких реляційних баз даних на боці сервера дозволяє ефективно масштабувати систему [13].

3. Графічний інтерфейс користувача (Velociraptor GUI / Web Console): Веб-орієнтована панель адміністрування та моніторингу. Вона надає аналітику інструменти для керування парком хостів, написання кастомних артефактів, візуалізації

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

результатів збору, моніторингу стану серверної інфраструктури, а також конфігурування інтеграційних шлюзів. Доступ до GUI захищається протоколом HTTPS та механізмами рольового розмежування доступу (RBAC).

4. Рівень збереження зібраної інформації та конфігураційних файлів. Velociraptor не потребує складних реляційних або NoSQL СУБД (таких як Elasticsearch чи PostgreSQL). За замовчуванням платформа використовує просту та надійну файлову структуру (File Register Datastore) безпосередньо на диску сервера, що гарантує високу швидкість запису та лінійне масштабування [13]. За потреби зберігання терабайтів криміналістичних образів файлова система може бути винесена у хмарні сховища.

Стратегії та режими розгортання

Залежно від масштабів контрольованої IT-інфраструктури, географічного розподілу хостів та вимог до відмовостійкості, архітектура розгортання центральних компонентів підтримує кілька основних топологій [7]:

1. Автономне універсальне розгортання (Standalone / All-in-One): Усі функціональні ролі (Frontend, GUI, Datastore) виконуються одним бінарним процесом на єдиному виділеному сервері (наприклад, під керуванням ОС Ubuntu). Даний режим є ідеальним для розгортання дослідницьких кіберлабораторій, проведення оперативного криміналістичного аналізу обмеженого пулу хостів та тестування нових індикаторів компрометації.

2. Кластеризоване багатовузлове розгортання (Multi-Frontend Cluster): Для великих корпоративних середовищ компоненти розділяються. Один вузол виділяється виключно під роботу GUI та координацію, а прийом клієнтського трафіку розподіляється між декількома серверами-фронтендами, які об'єднуються за балансувальником навантаження (Load Balancer). Сховище даних у такому випадку підключається як загальний мережевий масив або хмарне сховище, що забезпечує високу доступність (High Availability) та стійкість до відмов [9].

Мережеві комунікації та протоколи захисту каналів зв'язку

Забезпечення безпеки та цілісності телеметрії, що передається між клієнтами та сервером через незахищені канали зв'язку, є критично важливим аспектом

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

архітектури. Взаємодія компонентів Velociraptor регламентується суворими криптографічними протоколами:

1. Зв'язок Клієнт – Фронтенд: Агент ініціює підключення до сервера через захищений протокол mTLS (Mutual TLS). Для захисту від атак типу «людина посередині» (MitM) обидві сторони примусово валідують сертифікати одна одної за допомогою вбудованого самопідписаного центру сертифікації (Internal CA) [13], [9]. Під час первинного налаштування сервера в конфігураційний файл клієнта зашивається публічний сертифікат сервера, а також адреса підключення (server_urls). Агенти здійснюють періодичні короткі опитування сервера (Polling) через криптографічний тунель, очікуючи на появу нових VQL-задач. Порт взаємодії за замовчуванням визначається конфігурацією фронтенду (найчастіше використовуються порти 8000/TCP або 8001/TCP), причому сервер повинен бути налаштований на прослуховування всіх мережевих інтерфейсів (bind_address: 0.0.0.0).

2. Зв'язок Веб-консоль – Сервер: Доступ аналітика до GUI здійснюється через стандартний веб-браузер за протоколом HTTPS (найчастіше використовується виділений адмін-порт 8889/TCP або 8000/TCP залежно від обраного профілю генерації конфігурації). Комунікація між компонентом веб-інтерфейсу та внутрішнім ядром сервера реалізована на базі високошвидкісного протоколу gRPC, що мінімізує затримки при передачі великих масивів структурованих даних.

Рушій VQL (Velociraptor Query Language) як основа обробки телеметрії

Ядром автоматизації та гнучкості платформи є мова запитів VQL, яка за своїм синтаксисом нагадує стандартний SQL, але розширена потужними вбудованими плагінами та функціями для прямої взаємодії з низькорівневим API операційних систем [14].

VQL дозволяє аналітику абстрагуватися від складного програмування під конкретну ОС. Запит описує, які саме дані потрібно отримати, а вбудований рушій агента самостійно виконує парсинг бінарних структур операційної системи [13]. Сюди відноситься читання головної файлової таблиці MFT через плагін

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

parse_mft(), аналіз об'єктів реєстру через glob(), розбір журналів подій за допомогою parse_evtx() або моніторинг процесів у пам'яті.

Важливою перевагою є те, що фільтрація та агрегація даних відбуваються безпосередньо в оперативній пам'яті кінцевої точки [15]. Якщо запит на пошук підозрілого ключа автозавантаження опрацьовує 10 000 записів реєстру на хості, але знаходить лише 1 аномальний збіг, через мережевий канал mTLS на сервер передається лише 1 текстовий рядок результату, що робить платформу лідером за показниками мережевої оптимізації та швидкості проведення кіберполювання.

1.3 Аналіз Velociraptor Query Language.

Реалізація сучасних стратегій проактивного пошуку загроз (Threat Hunting) у межах розподіленої корпоративної інфраструктури вимагає використання інструментальних модулів, здатних здійснювати безперервний низькорівневий аналіз телеметрії хостів без створення критичного навантаження на обчислювальні ресурси кінцевих точок [15]. Децентралізована архітектура платформи Velociraptor дозволяє вирішувати ці завдання шляхом локального виконання гнучких декларативних VQL-скриптів, які класифікуються за типами артефактів безпеки та охоплюють три фундаментальні методологічні підходи: динамічний моніторинг подій, криміналістичний аналіз цілісності файлових систем та превентивне зменшення поверхні потенційних атак [14]. Розуміння контексту й ефективність застосування цих методів безпосередньо залежать від специфіки та можливостей криміналістичного двигуна мови запитів VQL.

Синтаксичний базис та архітектурна специфіка мови запитів VQL

Мова запитів Velociraptor Query Language (VQL) є фундаментальним інструментом автоматизації всієї платформи, що визначає логіку збору, аналітичного парсингу та обробки цифрових доказів безпосередньо на кінцевих точках інфраструктури. За своїм базовим синтаксисом VQL спроектована на основі класичного SQL (Structured Query Language), що дозволяє аналітику описувати умови пошуку у вигляді зрозумілих декларативних виразів типу SELECT ... FROM ... WHERE, повністю абстрагуючись від складного

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

низькорівневого програмування під конкретні інтерфейси (API) операційних систем. Проте, на відміну від стандартних реляційних СУБД, VQL не оперує статичними таблицями, збереженими на локальному диску сервера, і взагалі не використовує традиційні механізми індексації даних. Таблиці у VQL є динамічними абстракціями, які генеруються в режимі реального часу спеціалізованими програмними модулями – плагінами [13].

Основними конструктивними елементами мови VQL є:

1. Плагіни (Plugins): Виступають у ролі джерел даних і замінюють класичні таблиці у секції FROM. Плагін – це інтегрований у ядро агента код на мові Go, який приймає параметри, взаємодіє з підсистемами ОС (наприклад, читає сирі сектори диска, перелічує процеси в пам'яті, сканує гілки реєстру) і повертає потік рядків (колекцію об'єктів із фіксованими полями) [14]. Прикладами таких плагінів є `parse_mft()`, `glob()`, `pslist()`, `sockets()`.

2. Функції (Functions): Використовуються у секціях SELECT та WHERE для трансформації або оцінки окремих значень у межах кожного отриманого рядка [7-14]. Функції обробляють змінні, виконують конвертацію типів (наприклад, перетворення часових міток за допомогою `timestamp()`), обчислюють математичні вирази, аналізують бінарні структури за допомогою регулярних виразів чи здійснюють криптографічний розрахунок хеш-сум.

3. Оператори та вирази фільтрації: Дозволяють реалізовувати логіку відсікання легітимної телеметрії операційної системи за допомогою гнучких регулярних виразів (оператор `=~`), логічних зв'язок (AND, OR, NOT) та операторів покільцевого порівняння.



Рисунок 1.2 - Синтаксис та структура інструкцій мови запитів VQL

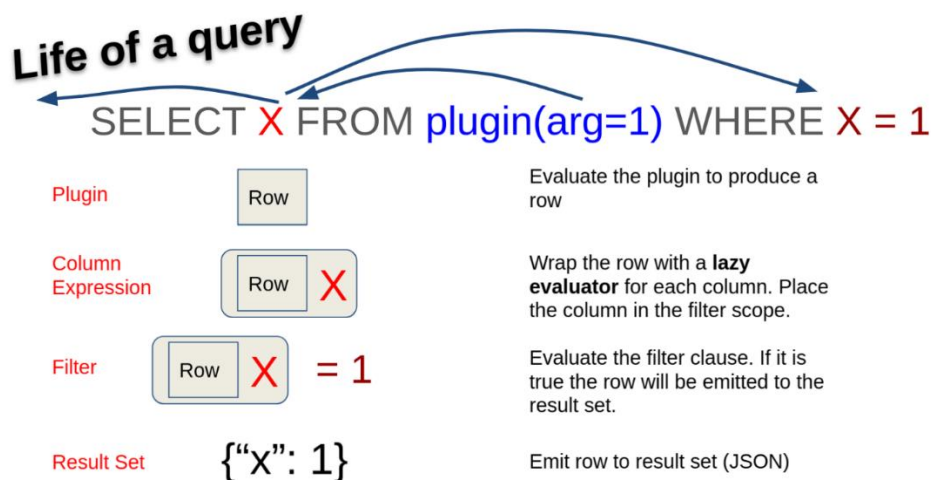


Рисунок 1.3 - Схема реалізації та виконання VQL-запитів

Унікальною архітектурною особливістю VQL є підтримка асинхронних подійних запитів (Event Queries), що відрізняє її від мов типу EQL (Event Query Language), адаптованих суворо під статичні послідовності [16]. Якщо у звичайному SQL запит виконується один раз і повертає поточний статичний зріз даних, то у VQL, якщо артефакт має тип EVENT, плагіни (наприклад, watch_evtx() або watch_monitoring()) переходять у стан безперервного фонового очікування [9], [13]. Вони створюють асинхронні підписки на системні події (через канали Windows Event Logs, ETW або підсистему ініціалізації подій ядра Linux inotify) [11]. Кожен раз, коли в операційній системі виникає відповідна дія (наприклад, створення нового процесу чи відкриття мережевого сокету), плагін миттєво генерує рядок даних, який проходить крізь фільтри VQL-запуску і, у разі виявлення аномалії, негайно відправляється на Frontend-сервер.

Ще однією важливою властивістю мови є ледаче обчислення (Lazy Evaluation) [14]. Якщо запит містить складні та ресурсомісткі операції (наприклад, розрахунок SHA256-хешу для тисяч виконуваних файлів), але секція WHERE відсікає більшість об'єктів на основі простіших критеріїв (наприклад, за назвою файлу чи його розміром), рушій VQL примусово не буде виконувати калькуляцію хешів для відфільтрованих об'єктів [13]. Це гарантує мінімальне використання оперативної пам'яті та процесорного часу на кінцевих хостах, дозволяючи

проводити глибокий Threat Hunting у висококритичних корпоративних середовищах без ризику порушення стабільності бізнес-додатків [15].

Проактивний аналіз журналів та подій у реальному часі (Event Monitoring)

Традиційний аналіз журналів реєстрації (Log Analysis) у реактивних системах захисту зводиться до безперервного копіювання сирих текстових масивів подій з кінцевих точок на центральний сервер зберігання (SIEM) для їх подальшої нормалізації та кореляції, що призводить до надмірного утилізуванню пропускної здатності мережевих каналів [17]. На противагу цьому, методологія Velociraptor базується на розгортанні клієнтських запитів подій (Artifact Type: EVENT), які виконуються агентом фоново та взаємодіють із джерелами телеметрії безпосередньо в оперативній пам'яті хоста за допомогою подійного рушія VQL [14].

На хостах під керуванням ОС Windows клієнтський бінарний файл підключається до підсистеми журналювання логів (Windows Event Logs), драйвера Sysmon, а також механізмів трасування подій для Windows (Event Tracing for Windows – ETW) [11]. Завдяки рушію VQL, аналітик отримує можливість проектувати кастомні сенсори для відстеження аномальної поведінки процесів у режимі реального часу. Наприклад, подійні артефакти типу Windows.Events.ProcessCreation дозволяють не просто фіксувати факт запуску виконуваного файлу, а миттєво аналізувати всю ієрархію батьківських процесів (Process Ancestry), розраховувати криптографічні хеш-суми бінарних об'єктів та перевіряти наявність енкодованих аргументів командного рядка (наприклад, інструкцій Base64 у сесіях PowerShell) [9].

У разі виявлення відповідностей заданій гіпотезі атаки, подія автоматично мапується на матрицю тактик і технік фреймворку MITRE ATT&CK (зокрема, етап ескалації привілеїв чи виконання шкідливого коду) [8]. Відфільтрована стиснута інформація про інцидент надходить на Frontend-сервер, де за допомогою інтеграційних скриптів обробки на базі Python трансформується у структуроване сповіщення та передається через захищене HTTPS-з'єднання на зовнішній

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

інтерфейс програмування додатків з їх власним API для оперативного тріажу аналітиком [2], [10].

Низькорівневий контроль цілісності файлових систем та криміналістичний аналіз (FIM)

Класичні інструменти моніторингу цілісності файлів (File Integrity Monitoring – FIM) функціонують шляхом періодичного рекурсивного обходу захищених директорій, порівняння поточних характеристик об'єктів з еталонними значеннями у локальних базах даних SQLite та розрахунку контрольних сум за алгоритмами MD5 чи SHA256. Недоліком такого підходу є висока інтенсивність операцій введення-виведення (I/O) на дисках та нездатність виявити маніпуляції, здійснені зловмисниками в обхід стандартних підсистем (API) операційної системи.

У платформі Velociraptor концепція FIM трансформована у підхід низькорівневої цифрової криміналістики (Digital Forensics) [7]. Замість стандартних запитів до ОС, вбудовані плагіни мови VQL (наприклад, `parse_mft()`) здійснюють прямий розбір Головної файлової таблиці (Master File Table — MFT) та Журналу оновлень USN (Update Sequence Number) файлової системи NTFS. Це дає змогу проводити аналіз структури файлових атрибутів, зокрема критично важливих полів `$STANDARD_INFORMATION` та `$FILE_NAME` [11].

Даний підхід є незамінним для виявлення техніки приховування слідів злочину – типом Timestomping (навмисна підміна часових міток створення або модифікації файлів шкідливого софту для їх маскування під легітимні системні компоненти). Оскільки більшість утиліт зловмисників змінюють часові мітки лише в атрибуті `$STANDARD_INFORMATION`, VQL-запит Velociraptor миттєво виявляє розбіжності при порівнянні з мітками в атрибуті `$FILE_NAME`, який залишається незмінним і доступним для редагування лише на рівні ядра ОС [14], [15]. Таким чином забезпечується детектування впровадження руткітів, шкідливих DLL-бібліотек чи несанкціонованих змін конфігураційних файлів без створення паразитних навантажень на накопичувачі хоста.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Мінімізація поверхні атак та оцінка відповідності конфігурацій (ASR)

Стратегія зменшення поверхні атак (Attack Surface Reduction – ASR) є комплексом превентивних заходів, спрямованих на максимальне обмеження кількості потенційних точок входу та векторів вразливостей, які зловмисник може експлуатувати для проникнення в систему [18], [19]. Можливості рушія VQL дозволяють реалізувати стратегію ASR за двома основними напрямками:

1. Безперервна інвентаризація та менеджмент вразливостей (Asset & Vulnerability Management): Агенти Velociraptor за допомогою цільових артефактів збору системного профілю (наприклад, Windows.Sys.Capabilities чи Linux.Sys.Packages) автоматично агрегують точні відомості про поточну версію ядра ОС, архітектуру, встановлені пакети безпеки (Patches), активні мережеві з'єднання та відкриті TCP/UDP порти [14]. Отримані криміналістичні зрізи зіставляються з глобальними базами даних відомих дефектів ПЗ (Common Vulnerabilities and Exposures – CVE) та сховищами кіберрозвідки Threat Intelligence (CTI), що дозволяє превентивно локалізувати слабкі місця в інфраструктурі до моменту їх реального зламу [15].

2. Оцінка конфігурації безпеки за міжнародними стандартами (SCA): Даний етап передбачає автоматизований аудит налаштувань операційних систем на відповідність еталонним бенчмаркам CIS (Center for Internet Security) або стандартам NIST. Політики оцінки конфігурації безпеки (Hardening) описуються всередині VQL-артефактів у структурованому людиночитабельному форматі YAML [14]. Модуль SCA забезпечує рекурсивну перевірку об'єктів реєстру, станів системних служб та параметрів групових політик. Зокрема, аналізуються налаштування складності паролів, активність механізмів контролю облікових записів (UAC), блокування застарілих та вразливих протоколів (таких як SMBv1 чи запуск незахищених макросів Office), а також обмеження використання неавторизованих USB-накопичувачів на основі принципу найменших привілеїв (Least Privilege) [19].

Завдяки оптимізації передачі даних, агенти надсилають на Frontend-сервер лише дельти (диференціальні відмінності) між поточним скануванням та

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

безпечним еталоном конфігурації, запобігаючи дублюванню інформації та підтримуючи актуальний статус захищеності кожної кінцевої точки в єдиній консолі керування [7].

1.4 Порівняльний аналіз існуючих SIEM/DFIR/EDR/XDR рішень та ключова різниця між ними.

Для проектування ефективного комплексу проактивного пошуку загроз (Threat Hunting) необхідно чітко диференціювати суміжний інструментарій забезпечення кібербезпеки, що розгортається в межах корпоративних інформаційних систем. Сучасний ринок захисних технологій представлений чотирьома домінуючими класами рішень: SIEM (Security Information and Event Management), EDR (Endpoint Detection and Response), XDR (Extended Detection and Response) та інструментами комплексної цифрової криміналістики й реагування на інциденти – DFIR (Digital Forensics and Incident Response) [1]. Попри те, що всі зазначені системи орієнтовані на виявлення шкідливої діяльності, вони кардинально відрізняються за своєю архітектурною філософією, глибиною доступу до даних, обчислювальними підходами та сценаріями практичного використання [15], [20].

Термінологічне розмежування та програмні реалізації захисних рішень:

1. Системи керування інформацією та подіями безпеки (SIEM): Даний клас рішень виступає в ролі централізованого агрегатора телеметрії вищого рівня. Головна архітектурна мета SIEM полягає у зборі, нормалізації, довготривалому зберіганні та інтелектуальній кореляції логів, що надходять від різнорідних джерел інфраструктури: мережевих брандмауерів, комутаторів, операційних систем, баз даних та прикладного ПЗ [17].

Класичними ринковими представниками корпоративного рівня тут є такі програмні продукти, як IBM QRadar [21] та Splunk Enterprise Security [22]. Ці платформи використовують потужні аналітичні рушії (наприклад, мову AQL в QRadar або SPL у Splunk) для історичного аналізу терабайт даних. Фокус SIEM зміщений у бік забезпечення комплаєнсу (відповідності вимогам стандартів PCI DSS, NIST, ISO) та сортування високорівневих сповіщень. Проте, SIEM володіє

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

вкрай низькою видимістю процесів безпосередньо всередині кінцевих точок і практично позбавлена можливостей швидкого низькорівневого розслідування інцидентів «на вимогу» [1].

2. Системи виявлення та реагування на кінцевих точках (EDR): EDR-рішення розгортаються безпосередньо на робочих станціях і серверах (кінцевих точках) з метою безперервного поведінкового моніторингу їхнього стану. Агенти EDR фіксують та аналізують життєвий цикл процесів, мережеві сокети хоста, модифікації реєстру та файлову активність [18].

Яскравим представником цього напрямку є гібридна open-source платформа Wazuh [23]. Вона інтегрує функції моніторингу логів та поведінкового аналізу хостів, забезпечуючи перевірку конфігурацій та базове реагування. Використовуючи правила детектування, системи цього класу здатні ідентифікувати складні та безфайлові атаки у реальному часі, а також виконувати активні дії – примусово завершувати шкідливі процеси чи ізолювати хост від мережі [2]. Основне обмеження класичного EDR полягає у звуженому колі видимості, обмеженому виключно межами хоста, без урахування контексту хмарних сервісів та мережевого трафіку усієї інфраструктури.

3. Системи розширеного виявлення та реагування (XDR): XDR є еволюційним розвитком концепції EDR, спрямованим на подолання технологічної ізолюваності захисних інструментів. Платформи класу XDR нативно інтегрують та корелюють телеметрію, що надходить одночасно з багатьох векторів: від кінцевих точок, мережевих шлюзів (NDR), систем керування ідентифікацією (IAM) та хмарних робочих навантажень.

Типовим прикладом програмного продукту, що трансформувався у повноцінне XDR-рішення, є Elastic Security (на базі Elastic Stack). Завдяки поєднанню легковажних агентів Elastic Agent, SIEM-модуля та вбудованих функцій захисту кінцевих точок (Endpoint Defend), платформа дозволяє бачити наскрізну часову шкалу складної атаки, яка розвивається паралельно у різних сегментах підприєм-

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

ства. Головна відмінність XDR від традиційних SIEM полягає у глибшій інтеграції даних на рівні агентів та наявності потужних вбудованих інструментів автоматизованого оркестрованого реагування (SOAR) [24].

4. Інструменти цифрової криміналістики та реагування на інциденти (DFIR): Даний клас систем, яскравим представником якого є платформа Velociraptor [14], займає специфічну нішу в інженерії кібербезпеки. Якщо EDR та XDR орієнтовані на безперервний пасивний збір потокової телеметрії, то DFIR-інструменти фокусуються на таргетному, високоточному та глибокому криміналістичному аналізі стану хоста «на вимогу» (on-demand triage) [7], [15].

Системи DFIR надають аналітику прямий доступ до найнижчих бінарних структур операційної системи: сирих секторів дисків, Головної файлової таблиці (MFT) та журналів оновлень USN за допомогою гнучких декларативних мов запитів типу VQL. Головна мета DFIR – надати неспростовні цифрові докази для суду або внутрішнього розслідування, встановити повний хронологічний ланцюжок дій хакера (Root Cause Analysis) та забезпечити гнучкий рушій для активного полювання (Threat Hunting) за прихованими загрозами [1].

Ключові відмінності у стратегіях використання та збору даних

Головна архітектурна різниця між розглядаємими класами систем полягає у балансі між обсягом накопичуваних даних (Data Volume) та глибиною деталізації (Investigation Depth) [7], [20].

Впровадження таких продуктів, як IBM QRadar, Splunk або Elastic XDR, вимагає розгортання великих і високовартісних серверних сховищ, оскільки ці системи змушені щомиті приймати, індексувати та зберігати терабайти подій для забезпечення можливостей історичного пошуку [21], [24]. Це створює проблему перевантаження інфраструктури даними та призводить до виникнення ефекту втоми аналітиків від великої кількості хибних сповіщень (Alert Fatigue) [17].

На противагу цьому, інструментарій DFIR (зокрема Velociraptor) реалізує парадигму децентралізованих обчислень. Замість копіювання сирих гігабайт логів, система залишає всі дані на кінцевих точках. Коли виникає потреба перевірити гіпотезу атаки, аналітик відправляє на хости точковий експрес-запит VQL.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Агент виконує парсинг бінарного артефакту локально в оперативній пам'яті комп'ютера і повертає на сервер виключно сухий, відфільтрований результат (декілька текстових рядків замість мегабайт логів) [14]. Це робить DFIR-платформи незамінними для швидкого Threat Hunting, оскільки швидкість опитування тисяч пристроїв займає лічені секунди, не утилізуючи при цьому мережевий трафік організації [9], [15].

Для наочного представлення технологічних відмінностей проведено комплексний порівняльний аналіз рішень за ключовими критеріями експлуатації (Таблиця 1.1).

Таблиця 1.1

Комплексний порівняльний аналіз рішень за ключовими критеріями експлуатації

Критерій порівняння	Системи класу SIEM (на прикладі IBM QRadar, Splunk)	Системи класу EDR (на прикладі Wazuh)	Системи класу XDR (на прикладі Elastic Security)	Інструменти DFIR (на прикладі Velociraptor)
Основний фокус системи	Централізований збір логів, кореляція та регуляторний комплаєнс	Безперервний захист та моніторинг хостів інфраструктури	Багатовекторний аналіз та нативна кореляція загроз	Глибока цифрова криміналістика та Threat Hunting
Основні джерела телеметрії	Логи з усього підприємства (FW, OS, DB, App)	Потокова телеметрія процесів, мережі та реєстру хоста	Консолідовані дані (EDR + мережа + хмари + Identity)	Низькорівневі артефакти ОС (MFT, RAM, USN, сирий диск)
Логіка обробки даних	Централізована (нормалізація та аналіз на сервері SIEM)	Гібридна (локальні правила агента + аналіз менеджера)	Централізована аналітика великих масивів даних	Децентралізована (локальний парсинг на хості через VQL)
Мережеве навантаження	Високе (постійна потокова передача сирих журналів)	Середнє (безперервне відправлення метаданих)	Високе/Середнє (залежно від моделі фільтрації)	Мінімальне (передаються лише відфільтровані результати)

Критерій порівняння	Системи класу SIEM (на прикладі IBM QRadar, Splunk)	Системи класу EDR (на прикладі Wazuh)	Системи класу XDR (на прикладі Elastic Security)	Інструменти DFIR (на прикладі Velociraptor)
Глибина криміналістичного аналізу	Низька (обмежена текстовим вмістом надісланих логів)	Середня (наявність базових ланцюжків процесів)	Середня/Висока (широкий міжсистемний контекст)	Максимальна (прямий доступ до бінарних структур ядра та файлових систем)
Режим детектування загроз	Реактивний (аналіз подій після їхнього запису в логи)	Проактивний/Реактивний (реальний час + правила)	Проактивний/Реактивний (автоматизований аналіз ланцюгів)	Проактивний (цілеспрямоване висунення та пошук гіпотез)
Можливості реагування	Створення інцидентів, запуск зовнішніх сценаріїв (скриптів)	Локальні дії: ізоляція хоста, кіл процесів, карантин	Оркестроване автоматичне реагування по всій мережі	Збір детальних криміналістичних доказів, точковий тріаж

Таким чином, розглядаємі класи систем не є взаємовиключними – вони доповнюють один одного у загальній піраміді забезпечення корпоративної безпеки [19]. Проте, для завдань автоматизованого кіберполювання, збору мережових артефактів та низькорівневої експрес-криміналістики, традиційні SIEM, EDR та XDR виявляються занадто громіздкими й обмеженими в інструментарії. Це повністю обґрунтовує необхідність проектування спеціалізованого програмного комплексу на базі DFIR-платформи Velociraptor, при цьому розуміючи її відмінності від схожих, на перший погляд, за метою дотримання інформаційної безпеки рішень. При цьому, подальша інтеграція з централізованими SIEM не виключена.

1.5 Класифікація та безпековий потенціал критичних цифрових артефактів операційної системи Windows.

У процесі проактивного пошуку загроз (Threat Hunting) та ретроспективного розслідування інцидентів кібербезпеки (DFIR) аналітики часто зустрічають сценарії, коли традиційні текстові журнали реєстрації подій операційної системи навмисно очищені зловмисником або за замовчуванням не містять деталізованих

відомостей про компрометацію [17]. Складні цільові атаки, що використовують техніки безфайлового виконання (Living off the Land) або довірений системний інструментарій (LOLBins), практично не залишають слідів у класичних антивірусних базах [8]. За таких умов основним джерелом неспростовних цифрових доказів стають низькорівневі криміналістичні артефакти операційної системи Windows (Artifacts of Execution and File Activity).

Дані об'єкти є побічними продуктами штатних підсистем ОС, призначених для оптимізації продуктивності, забезпечення сумісності додатків або підтримання транзакційної цілісності файлових систем [11]. Систематизація, класифікація та автоматизований аналіз цих артефактів за допомогою декларативних VQL-запитів платформи Velociraptor дозволяють відновити точну хронологію дій хакера, навіть якщо самі шкідливі файли були знищені або виконані виключно в адресній пам'яті хоста [20].

При цьому, поняття артефакту з точки зору цифрової криміналістики та поняття артефакту з точки зору Velociraptor все ж мають різницю. Для програмного забезпечення, це упакування одного або кількох запитів VQL та пов'язаних з ними даних у файл YAML, який читається людиною, і який зберігається у сховищі даних сервера [13]. Іншими словами, просто механізм VQL, який обробляє запит, створюючи серію рядків, і надсилає ці рядки на сервер. Артефакти у цьому випадку призначені для самодокументування за допомогою чітких описів та добре структурованого VQL, що і дозволяє виявляти ненормальну поведінку системи та її наслідки. Такий підхід сприяє обміну знаннями між користувачами з різним рівнем кваліфікації, у спільноті Velociraptor та ширшій спільноті DFIR [14].

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

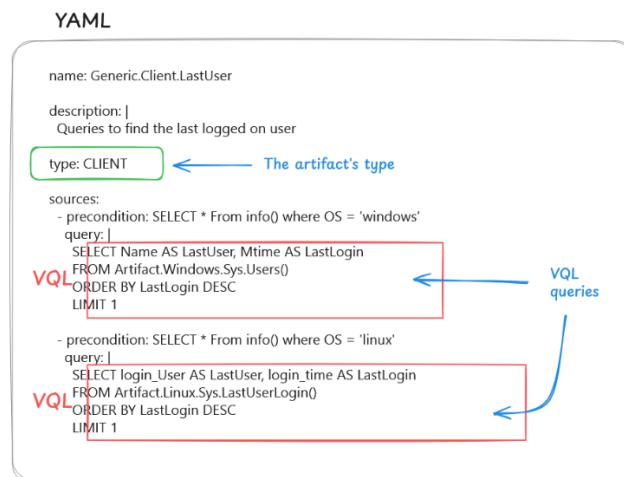


Рисунок 1.4 - Структура базового YAML-конфігураційного файлу артефакту Velociraptor

Технологічний потенціал артефакту Prefetch

Механізм попередньої вибірки (Prefetch) був інтегрований в операційну систему Windows з метою прискорення часу завантаження додатків та оптимізації взаємодії між менеджером пам'яті (Memory Manager) та дисковою підсистемою. Коли в системі ініціалізується новий процес, підсистема Prefetch відстежує перші 10 секунд його активності, фіксуючи всі файли, динамічні бібліотеки (DLL) та об'єкти метаданих, до яких звертався цей додаток. Отримані відомості компілюються у вигляді окремих бінарних файлів із розширенням .pf у захищеній системній директорії C:\Windows\Prefetch\. Назва кожного файлу формується за строгим шаблоном, який об'єднує ім'я виконуваного додатка та 8-символьний унікальний хеш-ідентифікатор шляху його запуску [11].

Для фахівця з Threat Hunting безпековий потенціал Prefetch є фундаментальним з огляду на обсяг криміналістичної інформації, яка записується всередину файлу .pf [1]:

- Факт та часові мітки виконання: У сучасних версіях Windows (Windows 10/11) у структурі Prefetch-файлу зберігається точний час останнього запуску програми, а також ретроспективна історія попередніх 7 запусків із точністю до мілісекунди [11].

- Лічильник запусків (Run Counter): Фіксує загальну кількість ініціалізацій додатка, що дозволяє диференціювати випадковий запуск малварі від стійкої активності шкідливого скрипта чи бекдору [15].

- Аналіз залежностей файлової структури: Всередині бінарної структури міститься повний перелік усіх файлових об'єктів та каталогів, які програма завантажила у пам'ять під час старту [11]. Це дає змогу аналітику чітко встановити, який саме шкідливий конфігураційний файл прочитав легітимний процес, або які тимчасові компоненти було розгорнуто дропером.

Через рушій Velociraptor за допомогою спеціалізованого VQL-плагіну `parse_prefetch()` аналітик може за кілька секунд опитати тисячі хостів інфраструктури. Це дозволяє миттєво локалізувати запуски шкідливого софту з нетипових або тимчасових директорій (наприклад, `AppData\Local\Temp\`), повністю ігноруючи спроби хакера перейменувати бінарний файл під стандартні системні компоненти (наприклад, маскування малварі під ім'ям `svchost.exe`) [9].

Безпекова специфіка реєстрового артефакту Amcache

Артефакт Amcache є спеціалізованим, прихованим кущем системного реєстру (Registry Hive), який операційна система використовує для функціонування підсистеми сумісності додатків (Application Compatibility Shim Engine). Цей об'єкт зберігається у вигляді бінарного файлу `Amcache.hve` за шляхом `C:\Windows\AppCompat\Programs\`. Головна мета Amcache полягає у фіксації метаданих про всі виконувані файли, інсталятори та динамічні бібліотеки, які хоча б один раз запускалися або були присутні в межах файлової системи хоста [11].

Криміналістична унікальність Amcache полягає в тому, що він записує статичні характеристики бінарних об'єктів, які часто не фіксуються іншими підсистемами ОС [1]:

1. Розрахунок криптографічних хеш-сум: Для кожного зафіксованого виконуваного файлу Amcache розраховує та назавжди зберігає його унікальний SHA-1 хеш-ідентифікатор. Це робить артефакт ідеальним джерелом для автоматизованого Threat Hunting на основі пошуку відомих індикаторів

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

компрометації (IoC) [20] – аналітик може зіставити бази даних зловмисних хешів із результатами збору Amcache без необхідності сканування всього дискового простору комп'ютера.

2. Час компіляції файлу (PE Compile Time): Дозволяє встановити реальний вік шкідливої утиліти, що є критично важливим при розслідуванні цільових атак (APT), де розробляється унікальний, раніше невідомий софт.

3. Повна інформація про видавця (Publisher): Фіксує дані цифрового підпису, назву компанії-розробника та версію продукту [11].

Оскільки Amcache записує дані навіть у тому випадку, якщо програма була запущена з зовнішнього USB-накопичувача чи мережевої папки й одразу після виконання видалена зловмисником, аналіз куща Amcache.hve через інтерпретатор VQL дає змогу виявити сліди утиліт, які хакер намагався повністю приховати.

Механізми виявлення аномалій за допомогою Shimcache

Артефакт Shimcache, офіційно відомий у структурі Windows як Application Compatibility Cache (Кеш сумісності додатків), функціонує у тісному зв'язку з ядром операційної системи та менеджером процесів. Його основне завдання – мінімізувати навантаження на дискову підсистему шляхом кешування характеристик файлів, які потенційно можуть потребувати застосування механізмів сумісності при запуску. Дані Shimcache зберігаються в оперативній пам'яті хоста і записуються безпосередньо у системну гілку реєстру HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache виключно в момент штатного завершення роботи (перезавантаження чи вимкнення) комп'ютера [11], [15].

Основні аналітичні характеристики Shimcache [1]:

1. Обмеження за обсягом: На сучасних версіях Windows 10/11 кеш здатний одночасно зберігати відомості про максимум 1024 останні виконувані об'єкти; при перевищенні цього ліміту записи циклічно перезаписуються за алгоритмом FIFO.

2. Зміст запису: На відміну від Amcache, Shimcache не містить хеш-сум файлів. Він фіксує виключно повний абсолютний шлях до файлу, його точний

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

розмір у байтах, а також часову мітку останньої модифікації об'єкта (Last Modified Time) відповідно до стандартів файлової системи [15].

Головний безпековий потенціал Shimcache для Threat Hunting полягає у специфіці формування його записів: файл потрапляє до цього кешу не лише у разі його безпосереднього виконання, а й тоді, коли до нього здійснювалися певні типи звернень з боку операційної системи [11]. Порівнюючи часові мітки модифікації файлів із Shimcache та Prefetch, аналітики SOC отримують можливість ефективно виявляти аномалії та встановлювати факт горизонтального переміщення (Lateral Movement) зломисників усередині корпоративної мережі [1]. Оскільки цей артефакт тривалий час перебуває в RAM, живий збір даних з реєстру за допомогою швидких VQL-скриптів Velociraptor є критично важливим елементом захисту від атак [18].

Роль Журналу змін NTFS USN Journal у побудові часових шкал інцидентів Журнал оновлень послідовних номерів утилізації (Update Sequence Number Journal, або USN Journal) є невід'ємним низькорівневим компонентом файлової системи NTFS. Він призначений для забезпечення відмовостійкості та швидкої індексації змін об'єктів. USN Journal функціонує як прихований метафайл із назвою \$Extend\\$UsnJrnl (а саме його альтернативний потік даних \$J), куди драйвер файлової системи NTFS примусово записує строгий хронологічний лог будь-яких модифікацій, що відбуваються з файлами та каталогами на конкретному дисковому томі [11], [15].

Кожен запис у USN Journal містить фіксовану бінарну структуру:

1. Повну назву файлу або папки;
2. Унікальний 64-бітний індекс об'єкта у Головній файловій таблиці (MFT Reference Number);
3. Точний час проведення операції;
4. Конкретний прапорець причини модифікації (Reason Flags), наприклад: DATA_CREATION (створення), DATA_OVERWRITE (перезапис), FILE_DELETE (видалення), RENAME_NEW_NAME (перейменування) [11], [19].

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Безпековий потенціал USN Journal для проактивного кіберполювання є фундаментальним, оскільки цей журнал неможливо непомітно модифікувати або очистити частково – його можна лише повністю вимкнути, що саме по собі є критичним тригером для систем моніторингу [1]. Аналіз записів USN Journal дозволяє детектувати таргетні шкідливі кампанії на ранніх стадіях:

1. Виявлення масових хвиль шифрування (Ransomware): Скрипти автоматизації на базі Python, підключені до API сервера Velociraptor [25], здатні в режимі реального часу відстежувати аномальний сплеск причин DATA_RENAME та DATA_OVERWRITE для сотень користувацьких файлів за секунду, що свідчить про початок роботи програми-вимагача [10].

2. Детектування збору та підготовки даних до витоку (Data Staging): Фіксація створення архівованих або прихованих масивів даних у нетипових директоріях перед їх відправкою на сервери зловмисників [15].

3. Ідентифікація техніки Timestomping: Прямий порівняльний аналіз часових міток у USN Journal та Головній файлової таблиці (MFT) дозволяє гарантовано виявити підміну часу створення файлів малварі. Записи всередині USN-журналу формуються послідовно на основі внутрішнього лічильника ядра ОС і не піддаються ручному коригуванню з боку користувацьких утиліт [11], [15].

Використовуючи декларативні можливості VQL-плагіну `parse_usn()`, розроблений програмний комплекс отримує можливість проводити миттєву фільтрацію мільйонів записів транзакцій безпосередньо на кінцевих хостах, виокремлюючи виключно ті аномалії, які відповідають специфічним тактикам зловмисників за MITRE ATT&CK. Спільна інтеграція інструментів симуляції атак (таких як Atomic Red Team) дозволяє валідувати коректність налаштованих фільтрів та забезпечувати високу точність збору даних [26], [27].

Висновки до першого розділу: У першому розділі кваліфікаційної роботи проведено комплексне теоретичне дослідження засад моніторингу кінцевих вузлів, концепції проактивного полювання за загрозами (Threat Hunting) та методів цифрової форензики інфраструктури. Доведено, що стрімке ускладнення ландшафту кіберзагроз, поява цільових таргетнованих атак (APT) та безфайлового шкідливого

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

програмного забезпечення за концепцією Living off the Land (LOLBins) роблять класичні реактивні та сигнатурні засоби захисту неефективними. Тоді як забезпечення безперервної видимості кінцевих вузлів (Endpoint Visibility) у поєднанні з методологією Detection Engineering є критично важливою умовою для зниження часу прихованого перебування зловмисника в мережі (Dwell Time).

Встановлено, що модульний клієнт-серверний принцип платформи Velociraptor, компільованої мовою Go, базується на інноваційній концепції відкладеного збору даних (on-demand forensics). Перенесення логіки парсингу та первинної фільтрації телеметрії безпосередньо на кінцеві точки (агенти) дозволяє лінійно масштабувати систему без потреби у розгортанні високовартісних серверних підсистем індексації та без ризику перевантаження мережевих каналів зв'язку. Також досліджено декларативну мову запитів VQL (Velociraptor Query Language) як основу обробки хостової телеметрії. Виокремлено унікальні архітектурні властивості VQL, зокрема підтримку асинхронних подійних запитів (Event Monitoring) для фонового відстеження аномалій у реальному часі та механізму ледачих обчислень (Lazy Evaluation), що мінімізує утилізацію процесорного часу та RAM на кінцевих пристроях і гарантує стабільність бізнес-додатків.

Виконано порівняльний аналіз систем класу SIEM, EDR, XDR та DFIR. Встановлено, що для завдань експрес-криміналістики та проактивного полювання традиційні лог-агрегатори є громіздкими, оскільки орієнтовані на текстові логи подій, тоді як DFIR-платформа Velociraptor забезпечує максимальну глибину дослідження інцидентів завдяки децентралізованій обробці та прямому доступу до бінарних структур ядра.

Оцінено безпековий потенціал низькорівневих криміналістичних структур операційної системи Windows (Prefetch, Amcache, Shimcache, USN Journal). Доведено, що їхній цілеспрямований аналіз засобами VQL-плагінів дозволяє успішно реконструювати часові шкали інцидентів, розраховувати криптографічні хеш-суми шкідливих файлів та детектувати техніки обходу захисту (зокрема, Timestomping), навіть за умов навмисного очищення зловмисником системних журналів подій EVTخ.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

РОЗДІЛ 2 ПРОЄКТУВАННЯ ПІДСИСТЕМИ МОНІТОРИНГУ

2.1 Аналіз інфраструктури організації та вибір топології розгортання підсистеми (Local / Cloud / Hybrid).

Проектування підсистеми проактивного моніторингу та полювання за загрозами (Threat Hunting) вимагає попереднього моделювання архітектурних особливостей та параметрів ІТ-інфраструктури підприємства. Для забезпечення практичної цінності дослідження та верифікації розроблюваного інструментарію було сформовано базову модель інфраструктури умовної організації корпоративного сектору. Відповідно до міжнародних стандартів інвентаризації, контролю та захисту активів підприємства [18], зокрема нормативів CIS Critical Security Controls (CIS Controls 01/02) та методологій NIST, контур об'єкта захисту містить три базові структурні сегменти [17]:

1. Корпоративний сегмент (Локальний контур): Об'єднує внутрішню локальну обчислювальну мережу (ЛОМ), службу каталогу на базі Active Directory Domain Services (AD DS) під керуванням Windows Server, внутрішні файлові сховища, бази даних та сервери додатків [11].

2. Робочі станції користувачів: Кінцеві пристрої співробітників (ноутбуки та персональні комп'ютери під керуванням ОС Windows та Linux), які підключені до локальної мережі підприємства через дротові або бездротові канали зв'язку.

3. Периметровий захист: Наявність брандмауерів та маршрутизаторів, що відокремлюють внутрішню довірену мережу від публічної мережі Інтернет.

Організація безперервного збору криміналістичної телеметрії у такому середовищі створює серйозні виклики для класичних реактивних засобів моніторингу. Згідно з рекомендаціями NIST щодо керування журналами безпеки, постійна потокова передача сирих подій від хостів до центрального накопичувача спричиняє критичне перевантаження каналів зв'язку та вимагає надлишкових обчислювальних потужностей для індексації [17]. Це обумовлює необхідність децентралізованого підходу до збору артефактів, за якого базова фільтрація та аналіз даних переносяться безпосередньо на кінцеві пристрої (on-demand triage) [1], [15].

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Порівняльний аналіз топологій розгортання підсистеми моніторингу

Для визначення оптимального місця розташування керуючих компонентів підсистеми моніторингу (сервера агрегації та автоматизації) було проведено порівняльний аналіз трьох основних топологічних моделей: локальної, хмарної та гібридної [20].

1. Локальна топологія (Local / On-Premises): Передбачає розгортання сервера моніторингу виключно всередині захищеного периметра локальної мережі організації. Переваги: Максимальний рівень контролю за криміналістичними даними, відповідність жорстким нормативним вимогам щодо захисту інформації [19], відсутність залежності від зовнішніх провайдерів та сторонніх каналів зв'язку. Недоліки: Обмеження видимості пристроїв, які фізично перебувають за межами офісу без підключення до VPN, а також необхідність самостійного обслуговування апаратної бази.

2. Хмарна топологія (Cloud-Native): Передбачає розміщення сервера моніторингу на базі зовнішніх хмарних платформ (наприклад, Microsoft Azure, AWS). Переваги: Висока еластичність обчислювальних ресурсів, спрощене масштабування, постійна доступність сервера для агентів з будь-якої точки світу через Інтернет [15]. Недоліки: Виникнення постійних операційних витрат, ризику транскордонної передачі технічних логів та потенційна можливість перехоплення даних при некоректній конфігурації хмарних доступів [19].

3. Гібридна топологія (Hybrid Deployment): Поєднує локальні робочі навантаження із зовнішніми хмарними шлюзами підключення. Центральний сервер або його фронтенд-компоненти виносяться в демілітаризовану зону (DMZ) чи хмару для прийому трафіку як від внутрішніх, так і від віддалених користувачів, що робить її привабливою з точки зору гнучкості покриття розподілених інфраструктур [9].

Враховуючи специфіку та цілі розроблюваного проєкту, для підсистеми моніторингу було обрано та обґрунтовано Локальну топологію (Local / On-Premises). Даний вибір зумовлений такими інженерними та безпековими факторами:

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

1. Конфіденційність та суверенітет криміналістичних даних: Платформа Velociraptor під час виконання VQL-запитів здатна витягувати низькорівневі бінарні артефакти (дампи оперативної пам'яті, копії кушів реєстру, записи Master File Table) [11], [13]. Ці дані містять конфіденційну інформацію про внутрішню структуру ОС, облікові записи та системні процеси. Локальне розгортання сервера гарантує, що жоден криміналістичний артефакт не залишить меж захищеного корпоративного периметра, що повністю відповідає вимогам стандарту NIST SP 800-53 щодо контролю та захисту даних [19].

2. Оптимізація витрат та автономність системи: Оскільки архітектура Velociraptor базується на принципі відкладеного збору даних (on-demand forensics) і не потребує важковагових баз даних (Elasticsearch тощо) для постійної індексації, локальний сервер використовує звичайну плоску файлову структуру [7], [13]. Це дозволяє розгорнути ефективне ядро моніторингу на мінімальних локальних апаратних потужностях (або в ізольованому лабораторному середовищі віртуалізації) без необхідності оплати регулярних хмарних підписок [9], [18].

3. Консолідація контуру керування та автоматизації: Вибір локальної топології дозволяє розмістити сервер Velociraptor та розроблюваний оркестраційний двигун на мові Python в одному довіреному мережевому сегменті. Взаємодія між Python-контролером та сервером реалізується локально через gRPC API (порт 8001/TCP). Це виключає потребу відкривати критично важливі порти керування системою безпеки у публічний Інтернет, що суттєво мінімізує власну поверхню атак комплексу [10].

Таким чином, обрана локальна конфігурація забезпечує максимальний рівень захищеності криміналістичних доказів, повну автономність функціонування, нульові витрати на сторонню інфраструктуру та високу швидкість обробки телеметрії, що повністю виправдовує її вибір для реалізації даного проєкту. При цьому ми приймаємо ті особливості, які в окремих аспектах побудови та функціонування не переважають варіанти з хмарою.

Після визначення інфраструктурних вимог здійснюється безпосереднє проєктування мережевої архітектури. Оскільки сама мережа у межах практичної

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

перевірки реалізована засобами ізольованої NAT-мережі у середовищі віртуалізації Oracle VirtualBox, наведена далі схема є демонстрацією типової побудови корпоративного контуру на основі класичної трирівневої структури Cisco (ядро, розподіл, доступ), зосереджуючи увагу на ключових кінцевих точках інфраструктури: сервері моніторингу, агентах та вузлі атакуючої сторони.

Спроектowana мережа представлена на рисунку 2.1.

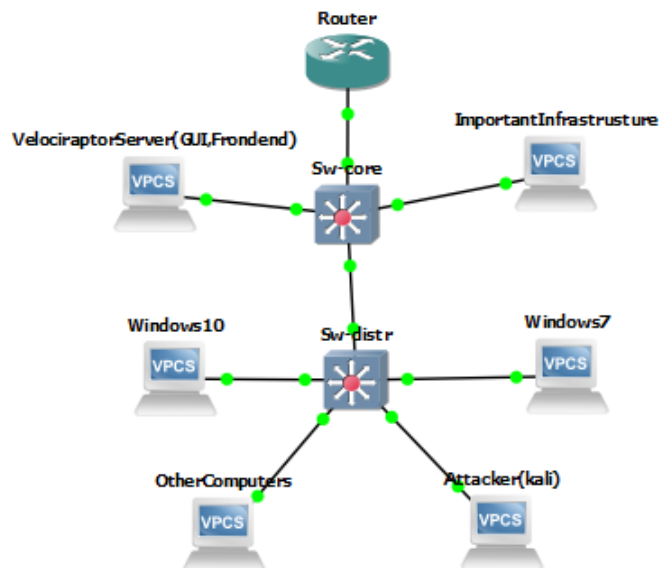


Рисунок 2.1 - Загальна топологія досліджуваної мережі в середовищі моделювання GNS3

Рівень ядра (Core Layer) складається з маршрутизатора та комутатора, з'єднаних між собою за допомогою інтерфейсів Ethernet. До рівня ядра включено центральні елементи управління інфраструктурою та сервер Velociraptor на базі ОС Linux Ubuntu Server, яка обрана через високу стабільність, оптимізоване використання системних ресурсів та захищеність на рівні ядра ОС.

Компоненти рівня ядра представлені на рисунку 2.2.

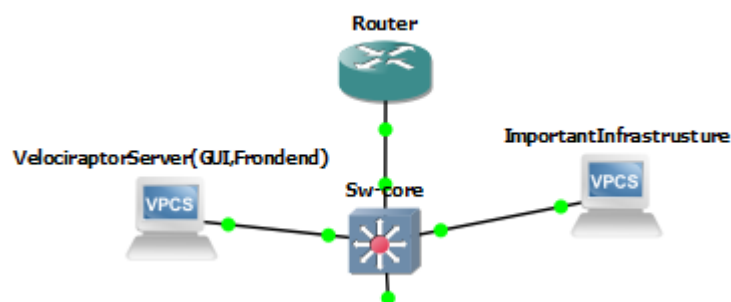


Рисунок 2.2 - Схема та конфігурація пристроїв рівня ядра мережі (Core Layer)

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Рівень розподілу (Distribution Layer) представлений комутаційним обладнанням, що під'єднане до ядра мережі за допомогою магістральних ліній зв'язку. Цей рівень відповідає за агрегацію потоків даних, маршрутизацію між підмережами (VLAN) та забезпечення високошвидкісного зв'язку між робочими станціями користувачів та ресурсами рівня ядра. Конфігурація пристроїв розподілу наведена на рисунку 2.3.

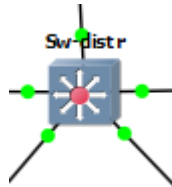


Рисунок 2.3 - Схема та конфігурація пристроїв рівня розподілу мережі (Distribution Layer)

Рівень доступу (Access Layer) представлений кінцевим устаткуванням користувачів, що забезпечує безпосереднє підключення пристроїв до мережевої інфраструктури за допомогою ліній Ethernet на витій парі. У нашому випадку на цьому рівні передбачено комп'ютери, які братимуть участь безпосередньо в експериментальному тестуванні підсистеми моніторингу: атакуючий вузол Kali Linux (емуляція несанкційованих дій), робоча станція під керуванням Windows 10 (корпоративний стандарт клієнтських ОС) та робоча станція під керуванням Windows 7 (імітація застарілого та вразливого сегмента інфраструктури). Структура рівня доступу наведена на рисунку 2.4.

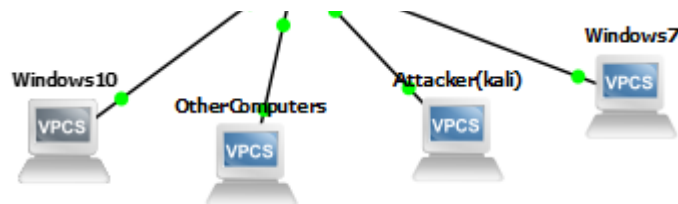


Рисунок 2.4 - Схема та конфігурація пристроїв рівня доступу мережі (Access Layer)

2.2 Розробка схеми мережевої взаємодії та забезпечення безпеки каналів зв'язку на основі взаємної автентифікації (mTLS та PKI).

Для забезпечення коректного функціонування та інформаційного обміну між усіма компонентами спроектованої архітектури було розроблено єдину схему адресації. Параметри розподілу адресного простору IPv4 наведено в таблиці 2.1.

Таблиця 2.1

Параметри адресації мережі

Пристрій	Мережа	Інтерфейс/ Мережний адаптер/Шлюз/ DNS	IPv4 адреса та пре-фікс
Маршрутизатор Router	VLAN1	Інтерфейс	207.41.7.1 /24
Сервер Velociraptor Server (Ubuntu)	VLAN1	Мережний адаптер	207.41.7.2 /24
Робоча станція Windows 10	VLAN1	Мережний адаптер	207.41.7.10 /24
Робоча станція Windows 7	VLAN1	Мережний адаптер	207.41.7.11 /24
Робоча станція Attacker (kali)	VLAN1	Мережний адаптер	207.41.7.20 /24

Оскільки підсистема проактивного пошуку загроз (Threat Hunting) за своєю архітектурною логікою оперує конфіденційною низькорівневою телеметрією кінцевих пристроїв та системними артефактами (включаючи копії реєстру, метадані файлових систем та процеси в пам'яті), безпека каналів транспортування даних є критично важливим інженерним викликом [15]. Класичне одностороннє шифрування TLS (Transport Layer Security), що повсюдно використовується у веб-інфраструктурах, захищає лише клієнта від підміни сервера, але залишає серверний контур вразливим до атак із використанням неавторизованих пристроїв або підроблених запитів. Для забезпечення цілісності, автентичності та конфіденційності криміналістичних доказів у межах локального контуру автоматизованого комплексу було розроблено криптографічну архітектуру на основі розгортання Інфраструктури відкритих ключів (Public Key Infrastructure – PKI) та реалізації протоколу взаємної автентифікації (Mutual TLS – mTLS) [9], [13].

Для виключення технологічної залежності від зовнішніх публічних комерційних центрів сертифікації (Certificate Authorities), а також підвищення рівня ізолюваності On-Premises архітектури, підсистема моніторингу реалізує концепцію внутрішнього замкненого кореня довіри (Root of Trust) [19]. Згідно з технічними регламентами платформи Velociraptor, під час первинної генерації серверної конфігурації за допомогою кросплатформного бінарного коду автоматично ініціалізується автономний, внутрішній Центр сертифікації (Internal Embedded CA) [7].

Цей вбудований СА виконує такі функції:

1. Випуск унікального самопідписаного кореневого сертифіката сервера (Root CA Certificate), який виступає найвищою точкою криптографічної довіри в підсистемі.
2. Автоматичне підписання локальних сертифікатів для вхідних інтерфейсів сервера (Frontend сертифікати) [13].
3. Випуск, авторизація та суворі валідація унікальних криптографічних сертифікатів для кожного підключеного клієнтського агента та інтеграційного API-інтерфейсу [9].

Такий підхід повністю захищає внутрішній контур кіберполювання від компрометації публічних SSL/TLS шлюзів і гарантує суворий контроль над життєвим циклом усіх ключів у повній відповідності до вимог безпеки стандарту NIST SP 800-53 щодо захисту каналів передачі даних [19].

Фундаментом безпеки мережевої взаємодії між клієнтськими агентами на кінцевих хостах та центральним Frontend-сервером є протокол взаємної автентифікації mTLS [9]. Замість пасивного очікування з'єднань, процес ініціалізації сесії та обміну телеметрією підпорядковується суворому криптографічному алгоритму двостороннього рукошлякування (Mutual Handshake) [13]:

1. Ініціалізація та запит сервера: Клієнтський агент (Velociraptor Client) надсилає запит на встановлення сесії (Client Hello) до порту взаємодії сервера (за замовчуванням 8000/TCP). Сервер відповідає надсиланням свого публічного сертифіката, а також примусовим криптографічним запитом сертифіката від клієнта (Certificate Request).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

2. Валідація сервера клієнтом: Агент перевіряє сертифікат сервера на відповідність зашитому всередині конфігураційного файлу `client.config.yaml` публічному ключу внутрішнього СА. Якщо валідація успішна, клієнт переходить до наступного кроку; у разі найменшої невідповідності з'єднання миттєво розривається, що нівелює ризики перехоплення трафіку через атаки класу Man-in-the-Middle (MitM) [15].

3. Автентифікація клієнта сервером: Клієнт надсилає свій унікальний сертифікат, підписаний внутрішнім СА, а також об'єкт `CertificateVerify`, який містить цифровий підпис попередніх повідомлень рукописання, створений за допомогою закритого ключа клієнта [7]. Сервер валідує підпис та перевіряє, чи дійсно цей сертифікат було випущено його власним вбудованим СА.

4. Формування сесійних ключів: Після успішної взаємної автентифікації сторони за допомогою алгоритму Діффі-Хеллмана (на базі еліптичних кривих ECDHE у TLS 1.3) генерують симетричні сесійні ключі для шифрування всього подальшого трафіку VQL-телеметрії [9].

Впровадження mTLS на рівні ядра системи повністю ліквідує вектор атак із використанням підроблених агентів (Rogue Agents). Навіть якщо зловмисник отримає доступ до бінарного файлу клієнта та спробує підключити свій контрольований пристрій до сервера моніторингу, сервер відхилить запит через відсутність валідного криптографічного сертифіката, підписаного внутрішнім Root СА [19].

Критично важливою особливістю безпеки каналів зв'язку у розроблюваній підсистемі є спосіб зберігання та валідації криптографічних доказів всередині конфігураційних файлів агентів. Під час автоматизованої інсталяції клієнтського модуля на кінцеві пристрої, у системну директорію копіюється файл конфігурації `client.config.yaml` [9]. Публічний сертифікат сервера та кореневий сертифікат внутрішнього СА імпортуються (зашиваються) безпосередньо у тіло цього YAML-документа у вигляді рядків коду Base64.

Під час ініціалізації мережевого стеку агент Velociraptor використовує виключно ці вбудовані сертифікати для перевірки автентичності сервера [14]. Він

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

повністю ігнорує стандартне системне сховище сертифікатів операційної системи (Windows Certificate Store або директорію /etc/ssl/certs у Linux) [11]. Це створює додатковий інженерний ешелон захисту: навіть якщо хакер отримає права локального адміністратора на кінцевому хості та скомпрометує сховище сертифікатів ОС (наприклад, примусово встановить довірений шкідливий корінь для перехоплення HTTPS-трафіку), він не зможе вклинитися у канал передачі криміналістичних артефактів, оскільки агент валідує сервер точково, спираючись лише на власну закриту PKI-конфігурацію [13], [15].

Така багатоконтурна схема забезпечення безпеки мережевої взаємодії (Таблиця 2.2) гарантує, що всі потоки даних у підсистемі – від сирієї хостової телеметрії до адміністративних gRPC-команд автоматизації – захищені від модифікації, аналізу та компрометації зловмисниками [2], [19].

Таблиця 2.2

Криптографічні контури безпеки каналів зв'язку моніторингу

Напрямок взаємодії модулів	Базовий протокол	Порт за замовчуванням	Метод автентифікації	Цільовий рівень захисту інформації
Клієнтський агент – Frontend Сервер	mTLS (Mutual TLS 1.3)	8000 / TCP	Взаємна перевірка сертифікатів через вбудований Root CA	Захист від атак Rogue Agent, перехоплення трафіку MitM, конфіденційність артефактів
Python-двигун – Серверний API	gRPC (поверх HTTP/2)	8001 / TCP	Авторизація за закритими ключами профілю api.config.yaml	Захист інтерфейсу керування, запобігання несанкційованому запуску Hunts зловмисниками

2.3 Моделювання загроз для кінцевих вузлів на основі матриці MITRE ATT&CK та формування стратегій полювання на загрози.

Ефективність підсистеми проактивного захисту безпосередньо залежить від точності математично-логічних моделей загроз та коректності висунутих гіпотез полювання [15]. Сучасні кіберзлочинці все частіше відмовляються від використання унікального шкідливого коду, який легко детектується сигнатурними методами, на користь легітимних адміністративних інструментів операційної системи

(концепція Living off the Land) [1]. Для систематизації знань про потенційні вектори компрометації кінцевих вузлів та формування ефективних стратегій Threat Hunting у межах локального контуру організації за базову модель було обрано глобальну базу знань тактик і технік зловмисників MITRE ATT&CK [8].

Важливим кроком на цьому етапі є застосування принципів інженерії детектування (Detection Engineering), яка розглядає моделювання загроз не як статичний опис, а як безперервний процес проектування та валідації аналітичних правил [6]. Процес моделювання загроз передбачає декомпозицію дій зловмисника на окремі тактичні етапи з їх подальшою прив'язкою до конкретних цифрових артефактів та подійних джерел операційної системи [1], [15]. У межах розроблюваного комплексу особливу увагу приділено детектуванню технік, які мають найвищий безпековий потенціал компрометації кінцевих точок за класифікацією MITRE [8]:

Execution (Виконання) – Техніка T1059 (Command and Scripting Interpreter): Передбачає запуск шкідливих інструкцій через вбудовані інтерпретатори команд операційної системи, такі як PowerShell або cmd.exe. Зловмисники маскують прапори виконання за допомогою Base64-кодування або обфускації рядків. Моніторинговим джерелом для цієї техніки виступає подійний аудит створення процесів (Process Creation), що фіксує ієрархію батьківських і дочірніх об'єктів [11].

Credential Access (Доступ до облікових даних) – Техніка T1003 (OS Credential Dumping): Спрямована на отримання зашифрованих паролів або LSASS-дампів оперативної пам'яті для подальшого просування мережею [8]. Полювання на цю загрозу вимагає низькорівневого аналізу дескрипторів доступу до пам'яті процесів та відстеження несанкціонованих звернень до кушів реєстру SAM і SYSTEM [11].

Defense Evasion (Обхід захисту) – Техніка T1070.006 (Timestomping): Навмисна модифікація часових міток файлів (створення, модифікації, доступу) для їх приховування від аналітиків під час криміналістичного аналізу [8]. Ця техніка моделюється через порівняння низькорівневих атрибутів MFT-таблиць та аналіз транзакцій NTFS USN Journal [11], [15].

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Impact (Вплив) – Техніка T1486 (Data Encrypted for Impact / Ransomware): Процес несанкціонованого масового шифрування файлової системи програмами-вимагачами [8]. Індикаторами цієї загрози є стрімкий сплеск модифікацій файлових розширень, перезапис заголовків та видалення тіньових копій тома, що фіксується безпосередньо через моніторинг операцій введення-виведення на рівні драйвера NTFS [10], [11].

На основі сформованої моделі загроз розробляється аналітична логіка функціонування підсистеми, яка трансформує теоретичні техніки MITRE ATT&CK у прикладні стратегії Threat Hunting, що реалізуються декларативними засобами мови VQL [8]. Для автоматизації та регулярної перевірки стійкості спроектованих правил доцільно використовувати концепцію емуляції атак за допомогою фреймворків типу Atomic Red Team, що дозволяє симулювати конкретні техніки MITRE та валідувати відгук сенсорів у контрольованому середовищі [26], [27]. Стратегія полювання будується за циклічним принципом і складається з трьох ключових етапів [1]:

1. Формулювання гіпотези: Аналітик висуває припущення про наявність прихованої загрози в мережі, наприклад: «У системі присутні шкідливі процеси, запущені з тимчасових директорій, які маскуються під легітимну утиліту svchost.exe» [15].

2. Проектування декларативного сенсора: Гіпотеза переводиться на мову конкретних криміналістичних перевірок. Замість повного сканування жорстких дисків, VQL-запит точково звертається до артефактів Prefetch та Amcache для перевірки шляхів запуску та валідації SHA-1 хешів виконуваних файлів [11].

3. Диференціальний аналіз та тріаж: Рушій VQL на хості відсікає відомі легітимні системні процеси на основі правил білих списків (Whitelisting), реалізуючи парадигму ледачих обчислень (Lazy Evaluation) [9]. На сервер повертається виключно чистий залишок аномалій, готовий для фіксації та автоматизованого сповіщення через Python-оркестратор [10].

Висновки до другого розділу: У другому розділі дипломної роботи на основі аналізу сучасних векторів кіберзагроз було здійснено проектування, мережеве

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

структурування та алгоритмічне моделювання підсистеми проактивного полювання за загрозами.

Здійснено аналіз структурних сегментів IT-інфраструктури умовної організації та проведено порівняльне дослідження моделей розміщення керуючих компонентів. На основі безпекових та інженерних факторів обґрунтовано вибір локальної топології (On-Premises), яка гарантує повний суверенітет і конфіденційність вилучених криміналістичних даних (дампів пам'яті, копій реєстру, метаданих), мінімізує власну поверхню атак комплексу та забезпечує високу автономність без хмарних операційних витрат.

Розроблено схему захисту каналів мережевої взаємодії, а саме спроектовано схему адресації IPv4 на основі класичної тривірневої структури Cisco (ядро, розподіл, доступ), та криптографічну архітектуру захисту інформаційних потоків. Впровадження інфраструктури відкритих ключів (PKI) та протоколу взаємної автентифікації (mTLS) на базі вбудованого замкненого кореня довіри (Internal CA) повністю ліквідує вектори атак класу Man-in-the-Middle (MitM) та унеможливорює несанкціоноване підключення підроблених пристроїв порушника.

Побудовано прикладну модель критичних загроз для кінцевих вузлів на основі міжнародної бази знань MITRE ATT&CK. Сформовано циклічні стратегії проактивного полювання (Threat Hunting), які дозволили перевести теоретичні хакерські техніки виконання шкідливого коду (T1059), дампування облікових даних (T1003), обходу захисту (T1070.006) та деструктивного впливу (T1486) у площину прикладних низькорівневих VQL-перевірок стану системних об'єктів.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ ПІДСИСТЕМИ НА БАЗІ VELOCIRAPTOR

3.1 Підготовка до розгортання тестового лабораторного середовища

Практична реалізація та експериментальне тестування ефективності розробленої підсистеми автоматизованого Threat Hunting вимагає розгортання ізолюваного лабораторного середовища віртуалізації (симуляційного стенду). Головна вимога до такого середовища – забезпечення повної безпеки під час симуляції кібератак та виконання шкідливого програмного забезпечення, що унеможливорює вихід деструктивного коду за межі тестового контуру, та одночасне надання повного доступу до системних ресурсів для збору цифрових доказів.

Для виключення будь-якої взаємодії з публічними мережами та запобігання витоку шкідливого трафіку, всі чотири віртуальні машини об'єднані в єдиний ізолюваний мережевий сегмент типу Internal Network (Внутрішня мережа). Мережеві порти, маршрутизація та розподіл IP-адрес налаштовані вручну (статична конфігурація). Локальний сервер Velociraptor не має прямого виходу в Інтернет.

Детальні апаратно-програмні специфікації та параметри розподілу ресурсів для кожного вузла лабораторного полігону наведені у таблиці 3.1.

Таблиця 3.1

Програмно-апаратні параметри віртуального лабораторного середовища

Назва ВМ	Операційна система (ОС)	Системна роль у підсистемі	Виділені процесори (vCPU), Оперативна пам'ять (RAM), Обсяг диска	Статична IP-адреса в мережі	Встановлене профільне ПЗ
Ubuntu	Ubuntu Server 22.04 LTS (64-bit)	Керуючий сервер моніторингу та ядро автоматизації	4 ядра, 4 ГБ RAM, 35 ГБ SSD	207.41.7.2	Velociraptor Server, Python 3.10, gRPC Core, grpcio, protobuf
Win10	Windows 10 Enterprise (64-bit)	Сучасна кінцева точка (об'єкт захисту)	1 ядро, 3 ГБ RAM, 70 ГБ SSD	207.41.7.10	Velociraptor Client (Agent), Microsoft Sysmon v14

Продовження таблиці 3.1

Назва ВМ	Операційна система (ОС)	Системна роль у підсистемі	Виділені процесори (vCPU), Оперативна пам'ять (RAM), Обсяг диска	Статична IP-адреса в мережі	Встановлене профільне ПЗ
Win7	Windows 7 Professional (32-bit)	Застаріла кінцева точка (об'єкт захисту)	1 ядро, 1 ГБ RAM, 100 ГБ HDD	207.41.7.11	Velociraptor Client (Agent), .NET Framework 4.5
Kali-Attacker	Kali Linux (64-bit)	Вузол симуляції кібератак (зловмисник)	2 ядра, 2 ГБ RAM, 80 ГБ SSD	207.41.7.20	Nmap, Metasploit Framework, CrackMapExec, Impacket Tools

Розподіл обчислювальних ресурсів, представлений в таблиці 3.1, повністю враховує легковажну архітектуру обраного інструментарію. Оскільки агенти Velociraptor не виконують постійного потокового копіювання даних, виділення 1 Гб RAM для Windows 7 та 3 Гб для Windows 10 є більш ніж достатнім для стабільного функціонування операційних систем під час навантажувальних тестів. Локальний парсинг VQL-запитів відбувається безпосередньо в межах оперативної пам'яті хостів, передаючи на IP-адресу сервера лише компактні структури.

Розгорнутий на сервері автоматизований контролер на мові Python ініціалізує постійну gRPC-сесію прослуховування сервера за допомогою локального порту 8001/TCP. Таке компонування лабораторного стенду створює надійну, масштабовану та безпечну інженерну базу для виконання наступних практичних етапів дослідження: від автоматизованого розгортання клієнтських конфігурацій до активної емуляції шкідливого впливу із Kali Linux та моментальної фіксації уражених низькорівневих артефактів.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

3.2 Практичне розгортання сервера Velociraptor та інсталяція агентів моніторингу на кінцевих пристроях

Розбір налаштування функціонування системи на сервері та агентах:

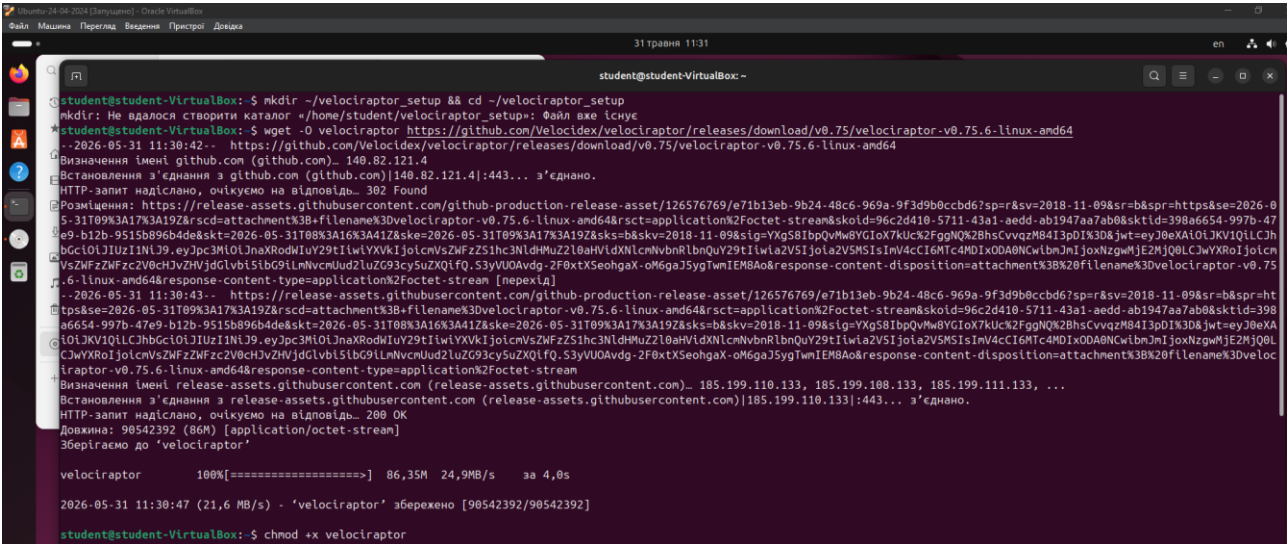


Рисунок 3.1 - Ініціалізація директорії та завантаження інсталяційного пакету Velociraptor на сервер

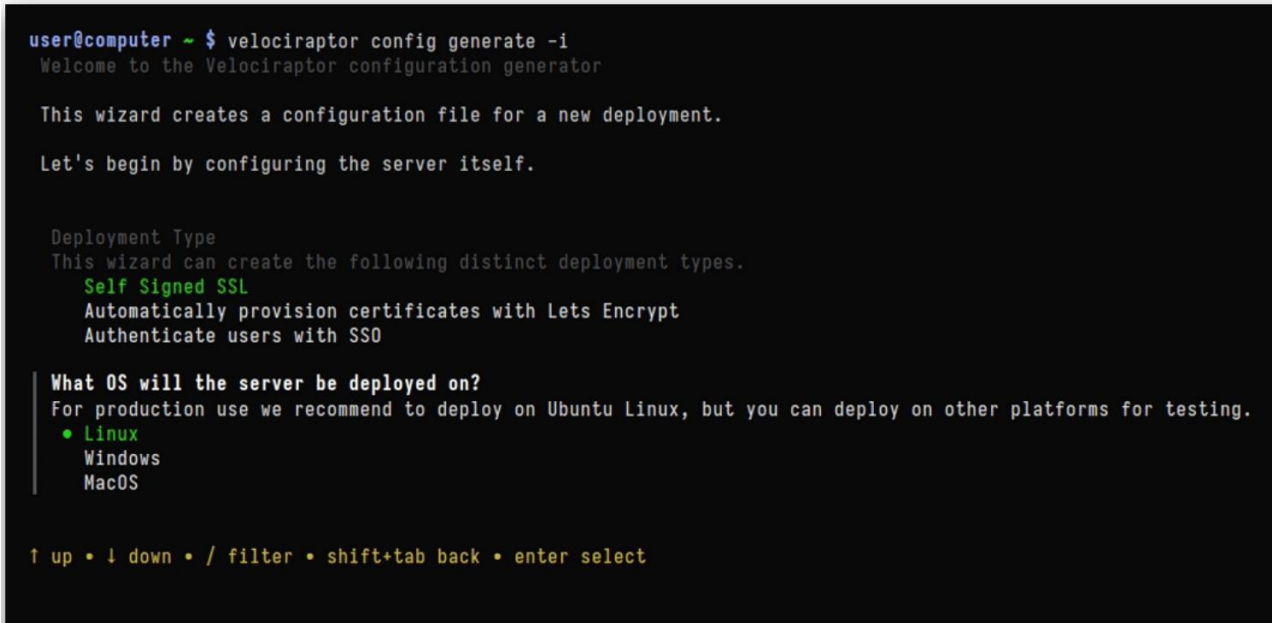


Рисунок 3.2 - Процес генерації базового конфігураційного файлу сервера

```
The datastore directory is where Velociraptor will store all files. Make sure there is sufficient disk space available!  
> /opt/velociraptor  
  
Path to the logs directory.  
Velociraptor will write logs to this directory. By default it resides within the datastore directory but you can place it  
anywhere.  
> logs  
  
Internal PKI Certificate Expiration  
By default internal certificates are issued for 1 year.  
  
If you expect this deployment to exist part one year you might  
consider extending the default validation.  
1 Year  
2 Years  
10 Years  
  
Do you want to restrict VQL functionality on the server?  
  
This is useful for a shared server where users are not fully trusted.  
It removes potentially dangerous plugins like execve(), filesystem access etc.  
  
NOTE: This is an experimental feature only useful in limited situations. If you do not know you need it select N here!  
  
Yes No  
  
Use registry for client writeback?
```

Рисунок 3.3 - Процес генерації конфігураційного файлу сервера (продовження)

```
This section configures the server's network parameters.  
  
What is the public DNS name of the Master Frontend  
Clients will connect to the Frontend using this public name (e.g. https://www.example.com/ ).  
> 192.168.1.1  
  
DNS Type  
In order for the server to be reachable from the internet, you must have DNS configured.  
None - Configure DNS manually  
NOIP  
CloudFlare  
  
Would you like to try the new experimental websocket comms?  
Websocket is a bidirectional low latency communication protocol supported by  
most modern proxies and load balancers. This method is more efficient and  
portable than plain HTTP. Be sure to test this in your environment.  
  
Yes No  
  
Enter the frontend port to listen on.  
> 8000  
  
Enter the port for the GUI to listen on.  
> 8889  
  
shift+tab back • enter submit
```

Рисунок 3.4 - Процес генерації конфігураційного файлу сервера (продовження)

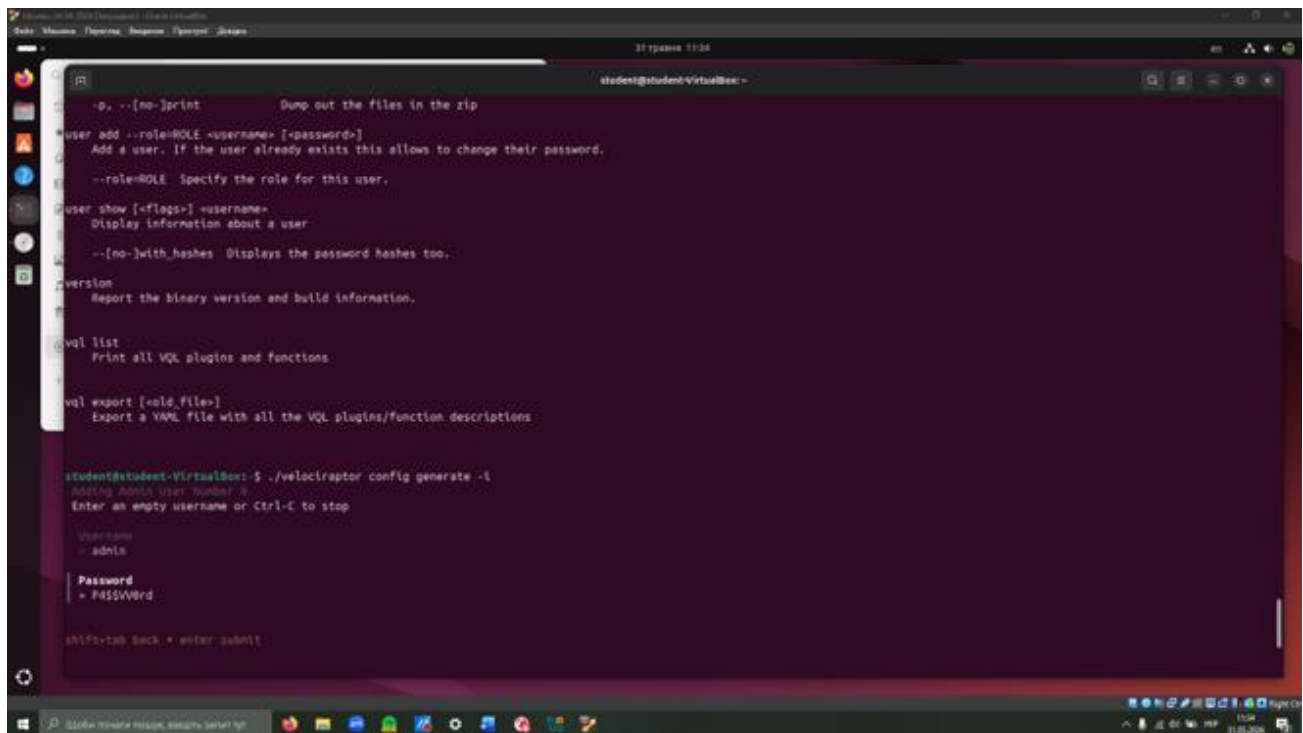


Рисунок 3.5 - Процес генерації конфігураційного файлу сервера (завершення)

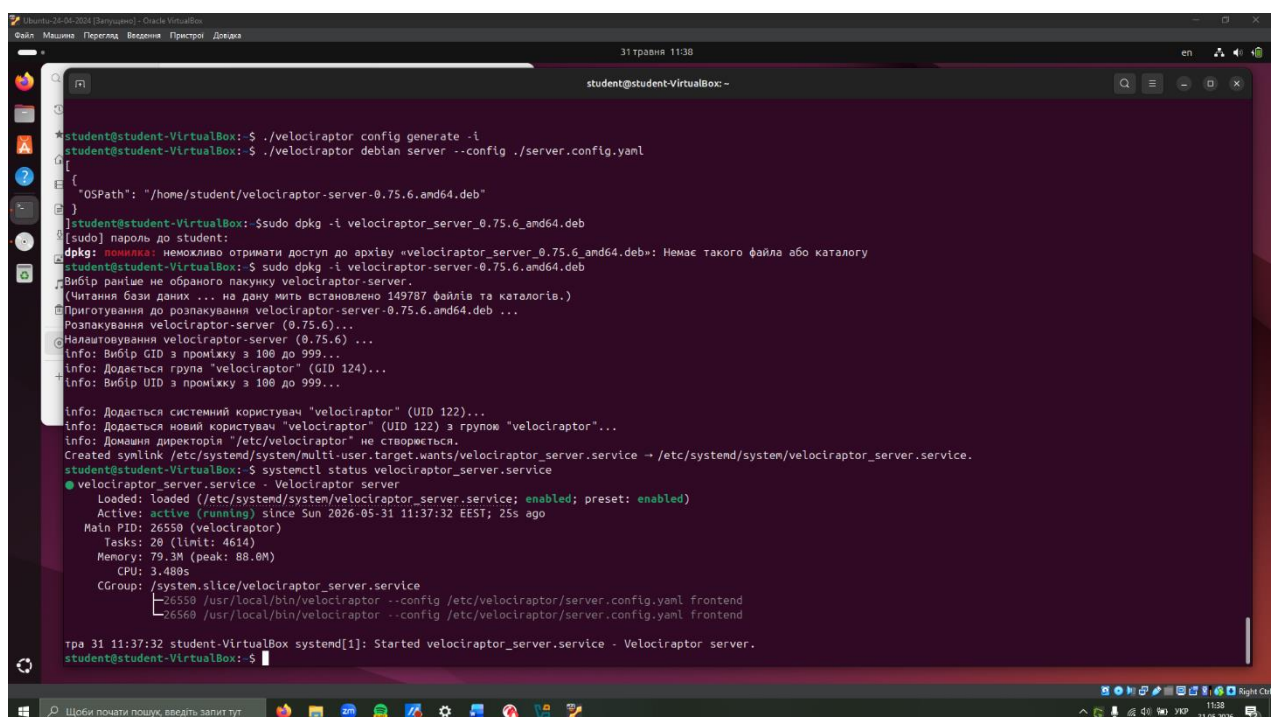


Рисунок 3.6 - Встановлення та запуск системного сервісу сервера Velociraptor

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

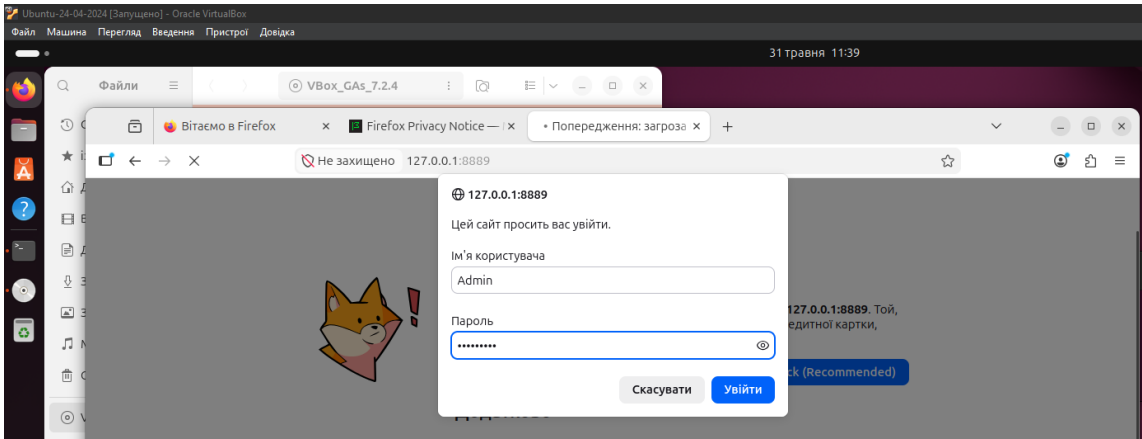


Рисунок 3.7 - Вікно автентифікації користувача у веб-інтерфейсі (GUI) сервера

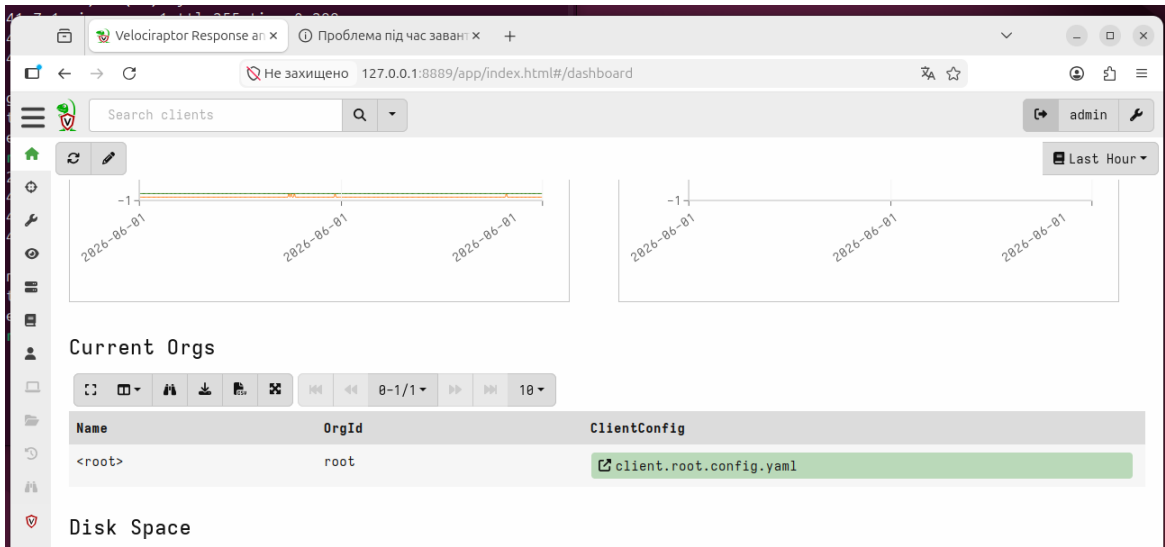


Рисунок 3.8 – Автентифікація проведена успішно

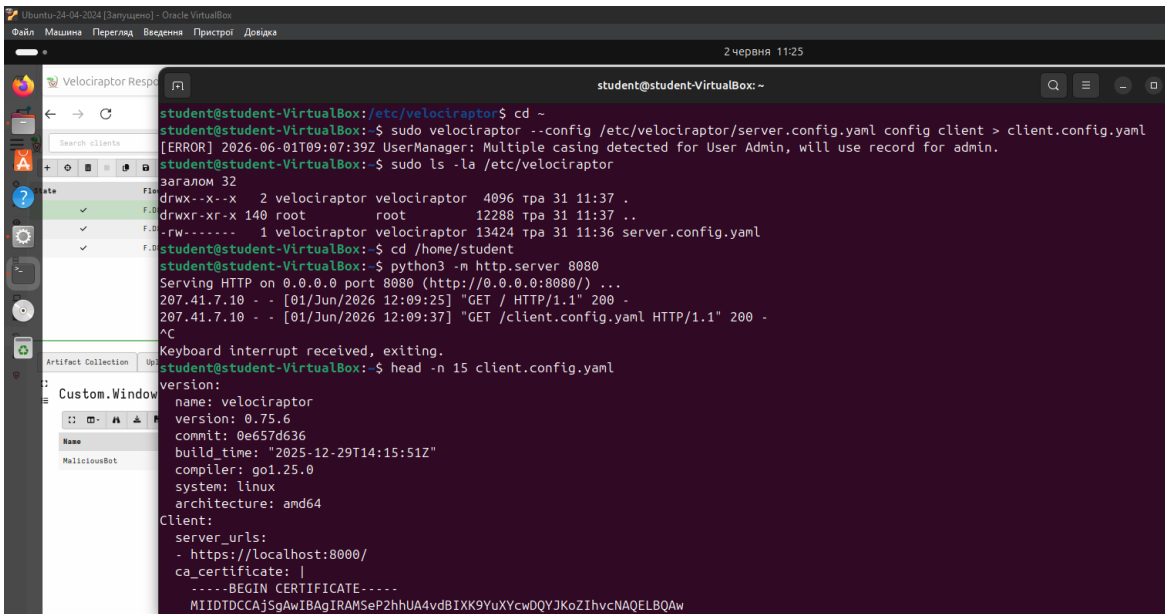


Рисунок 3.9 - Створення клієнтського конфігураційного файлу на основі серверного

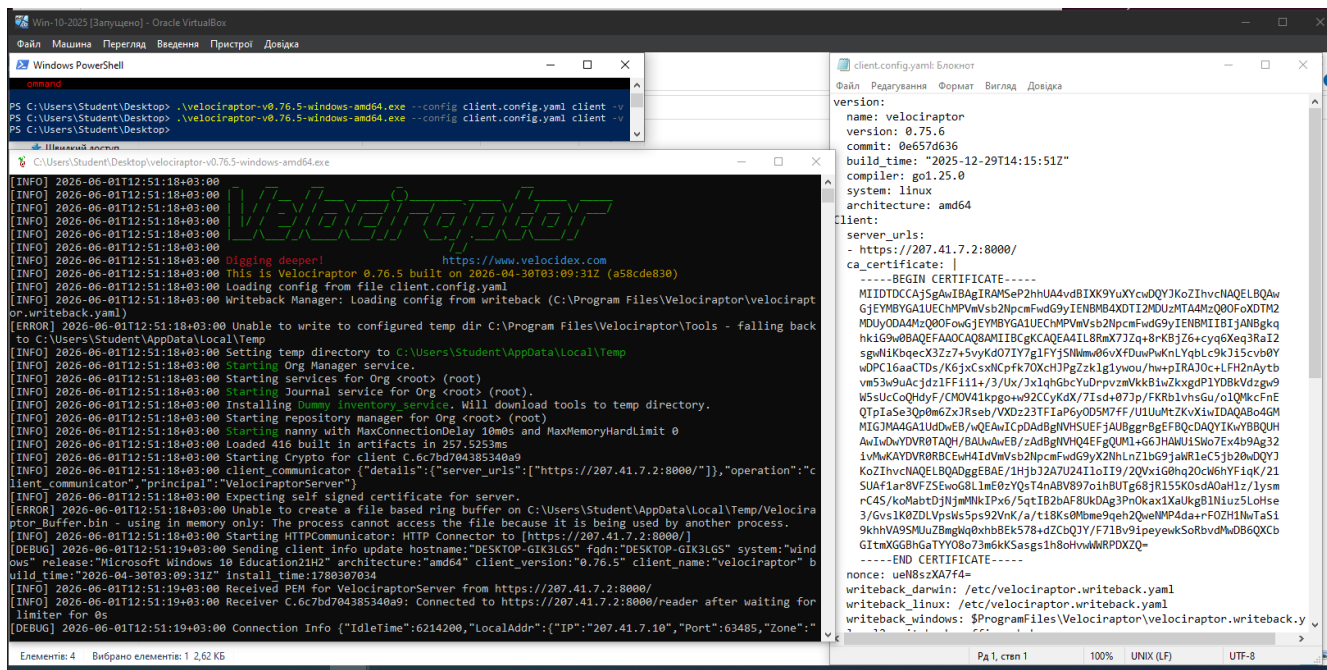


Рисунок 3.10 - Внесення мережевої адреси сервера та процес інсталяції агента на Windows 10

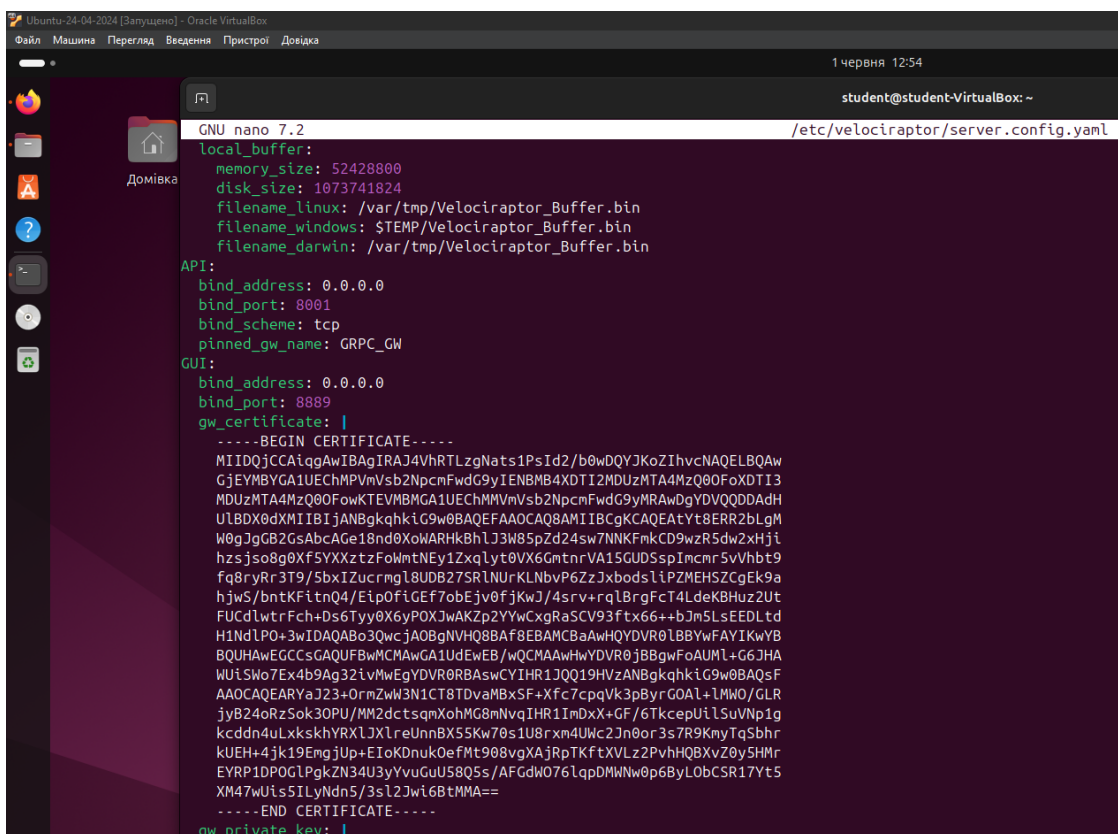


Рисунок 3.11 - Модифікація конфігураційного файлу сервера для прослуховування TCP-портів

		Куліш М.В.				Арк.
		Лайчук А.А.				56
Змн.	Арк.	№ докум.	Підпис	Дата	ККІКБ.КР.Б – 125 – 07 – ПЗ	

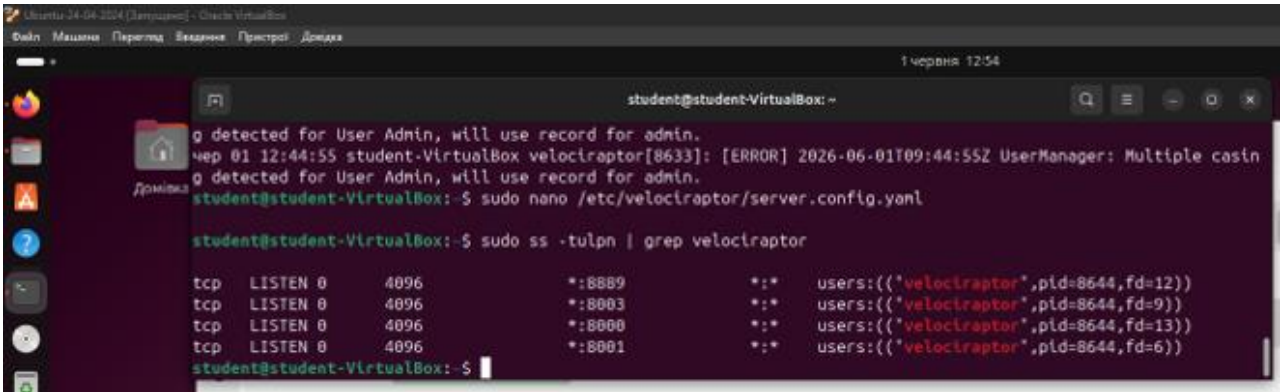


Рисунок 3.12 - Коректне функціонування прослуховування портів, що дозволить почати функціонування сервера та агентів

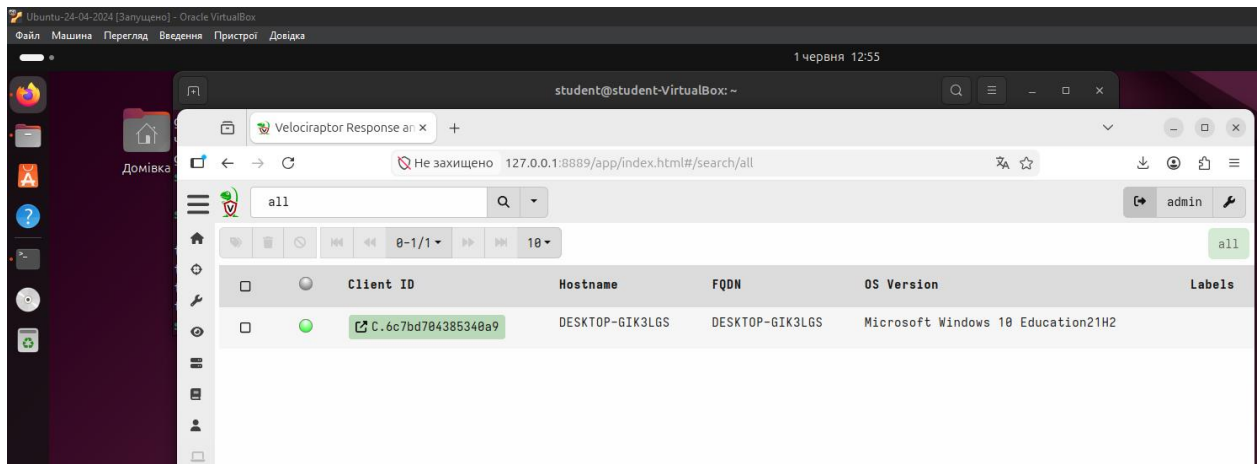


Рисунок 3.13 - Активний статус агента Windows 10 в інтерфейсі моніторингу сервера

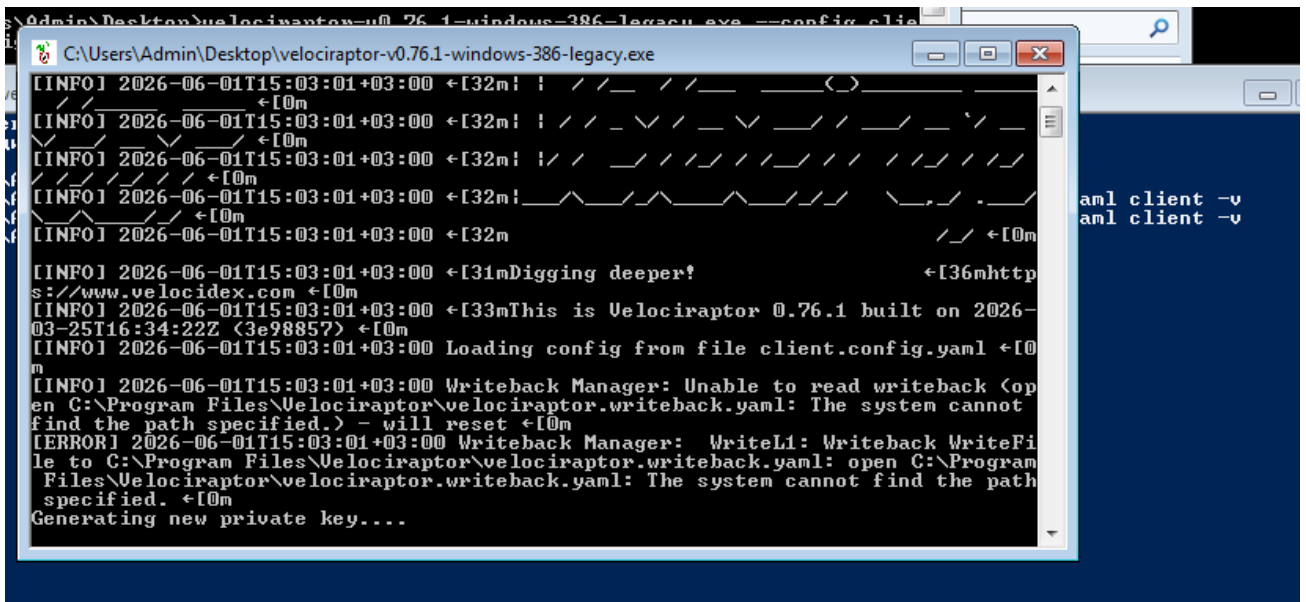


Рисунок 3.14 - Процес інсталяції програми-агента на робочу станцію Windows 7

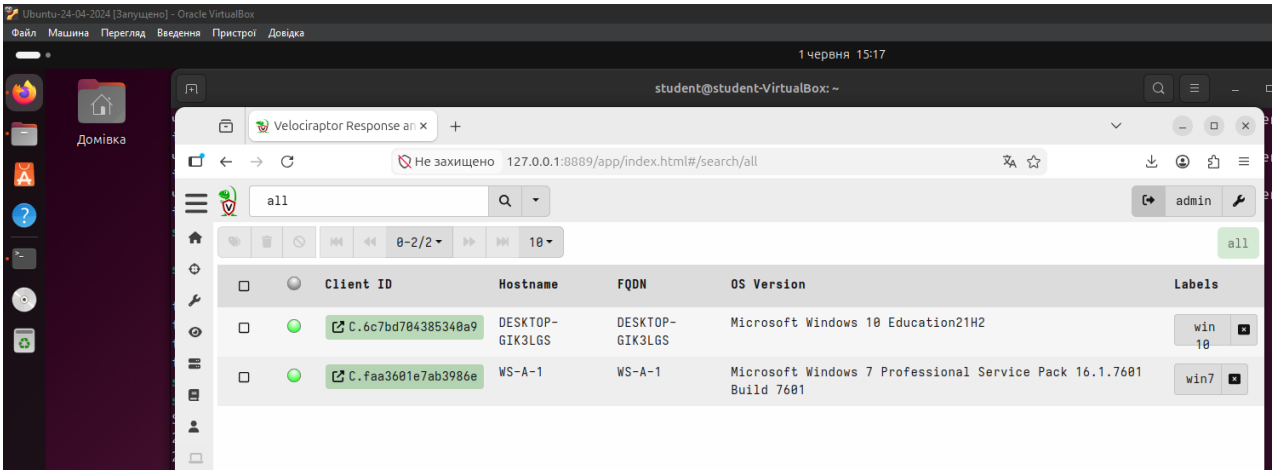


Рисунок 3.15 - Відображення активного статусу агента Windows 7 у GUI сервера

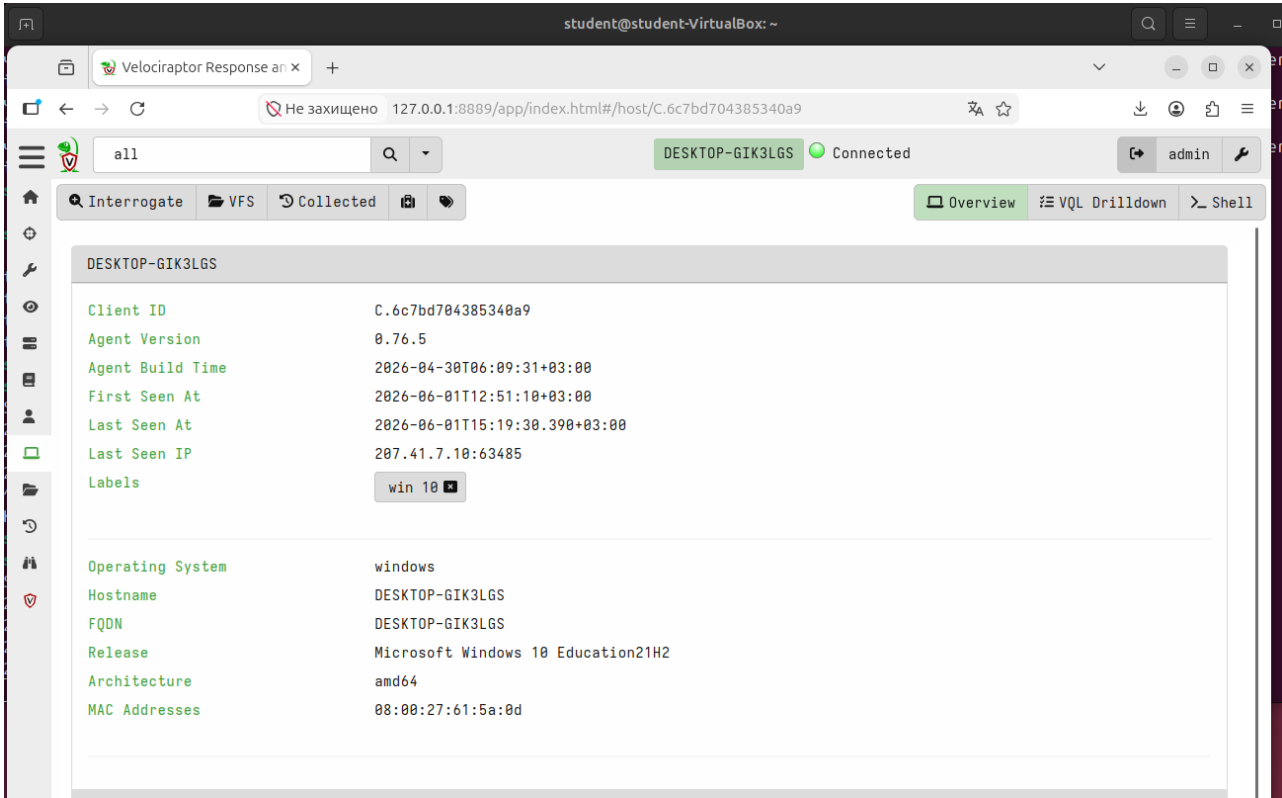


Рисунок 3.16 - Інтерфейс моніторингу системних параметрів агента Windows 10

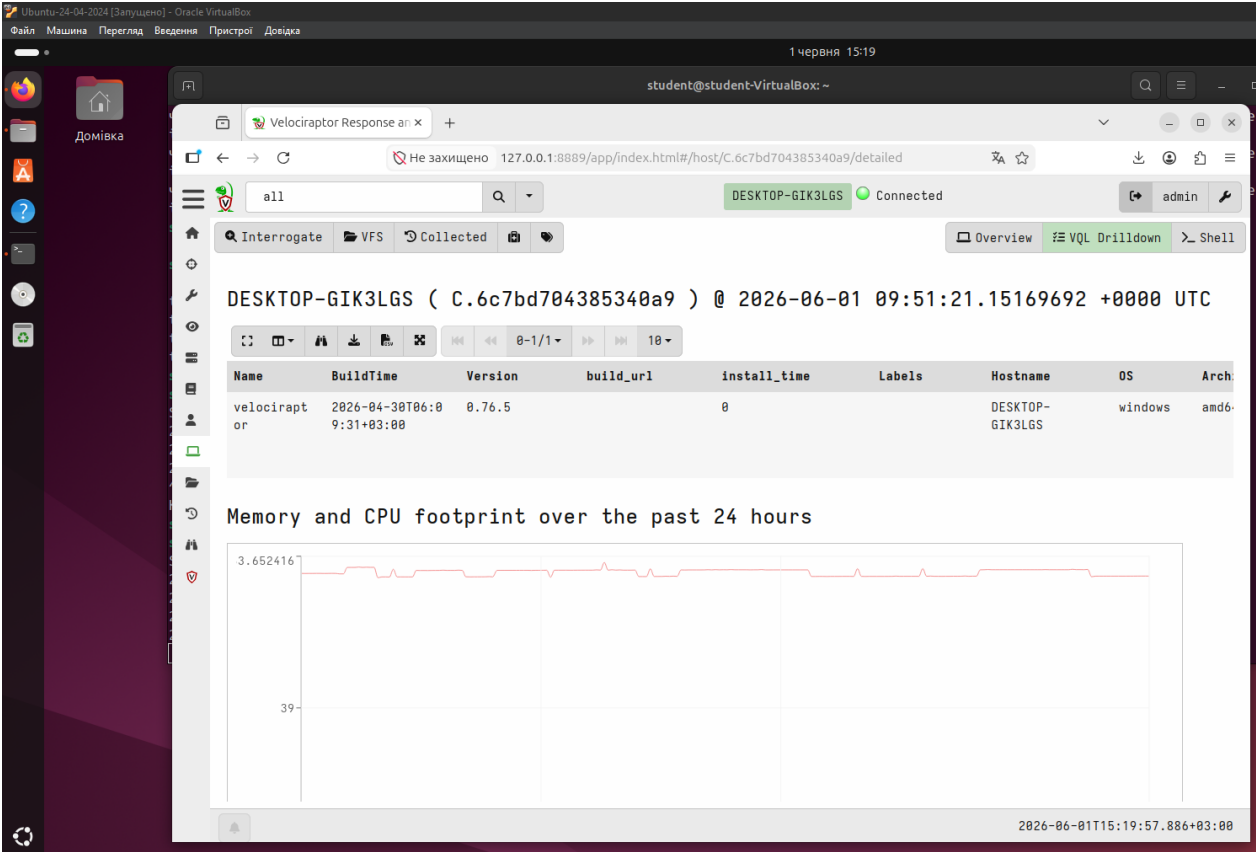


Рисунок 3.17 - Інтерфейс моніторингу системних параметрів агента Windows 10 (продовження)

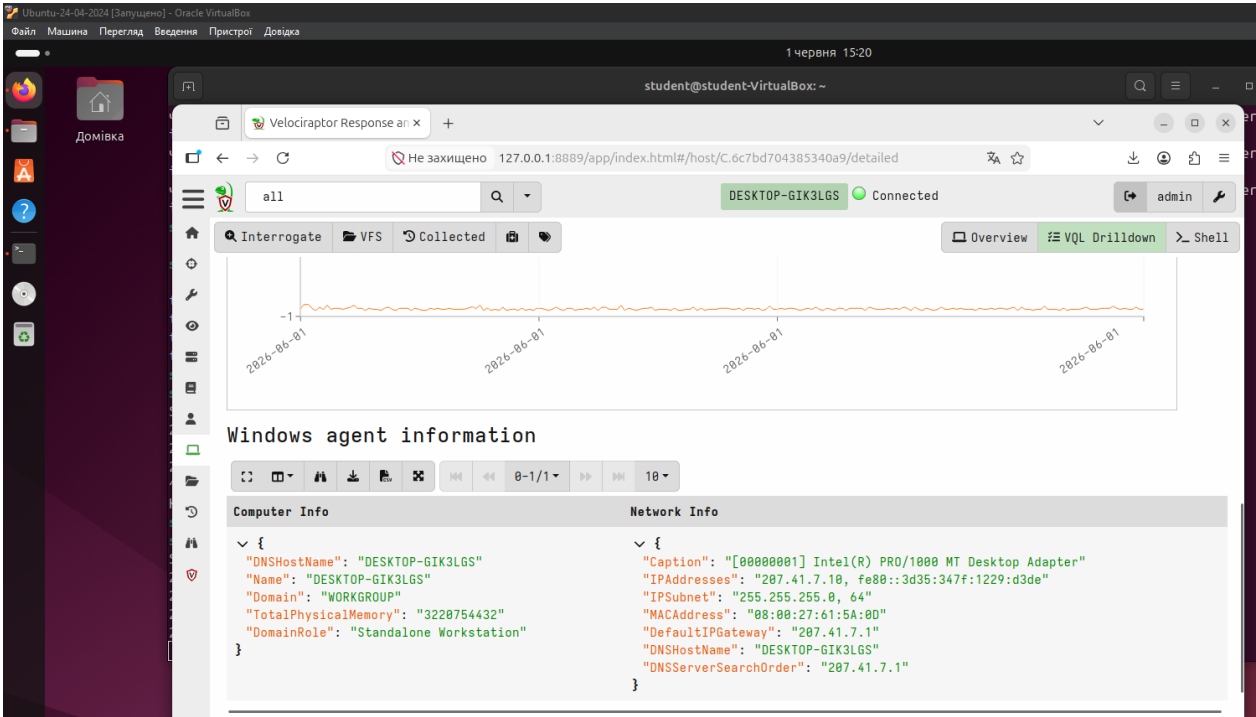


Рисунок 3.18 - Інтерфейс моніторингу системних параметрів агента Windows 10 (продовження)

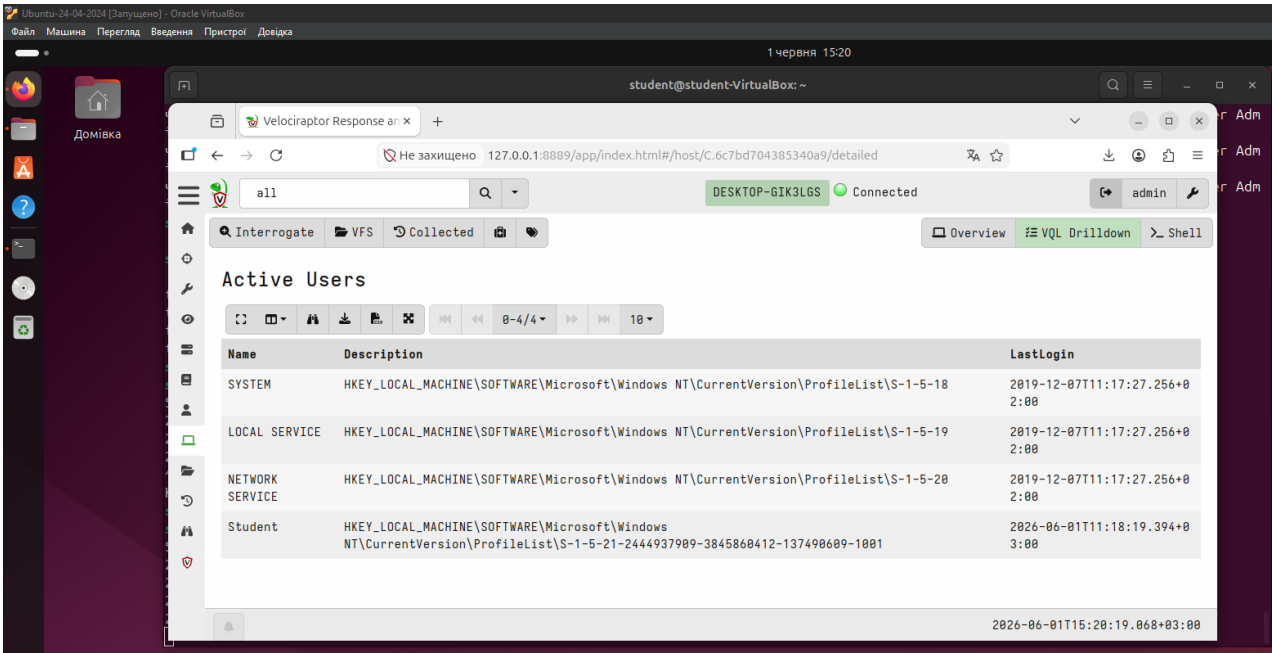


Рисунок 3.19 - Інтерфейс моніторингу системних параметрів агента Windows 10 (завершення)

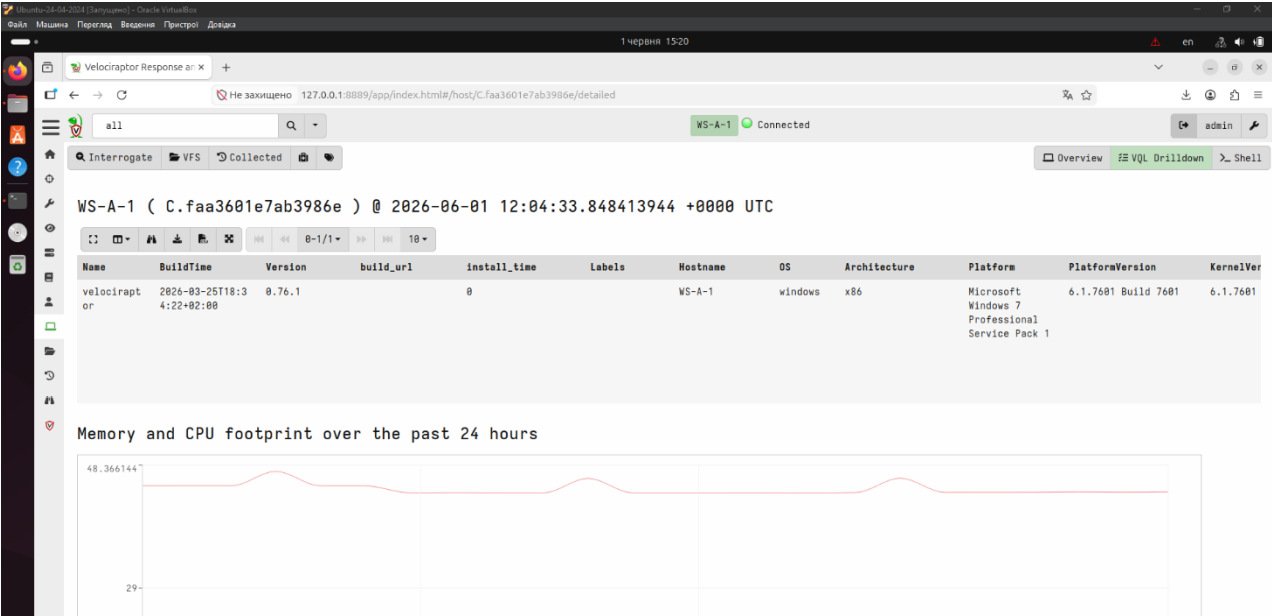


Рисунок 3.20 - Інтерфейс моніторингу системних параметрів агента Windows 7

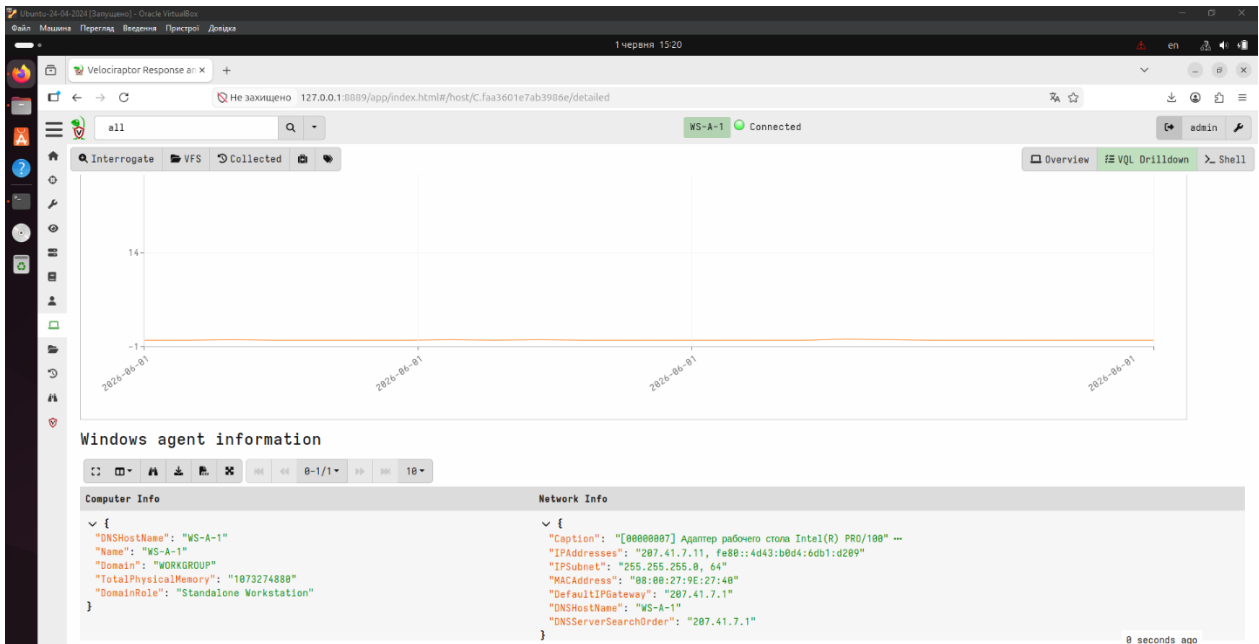


Рисунок 3.21 - Інтерфейс моніторингу системних параметрів агента Windows 7 (продовження)

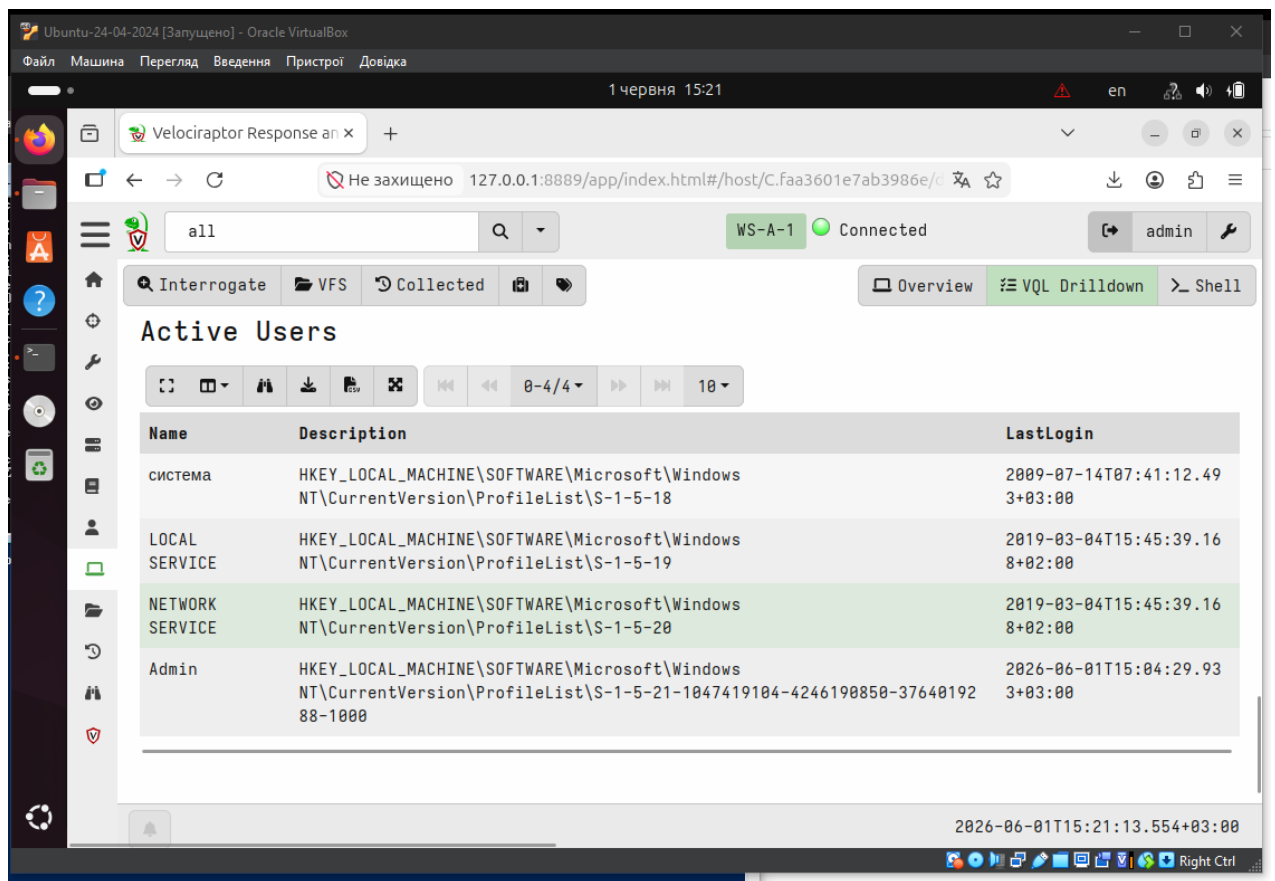


Рисунок 3.22 - Інтерфейс моніторингу системних параметрів агента Windows 7 (завершення)

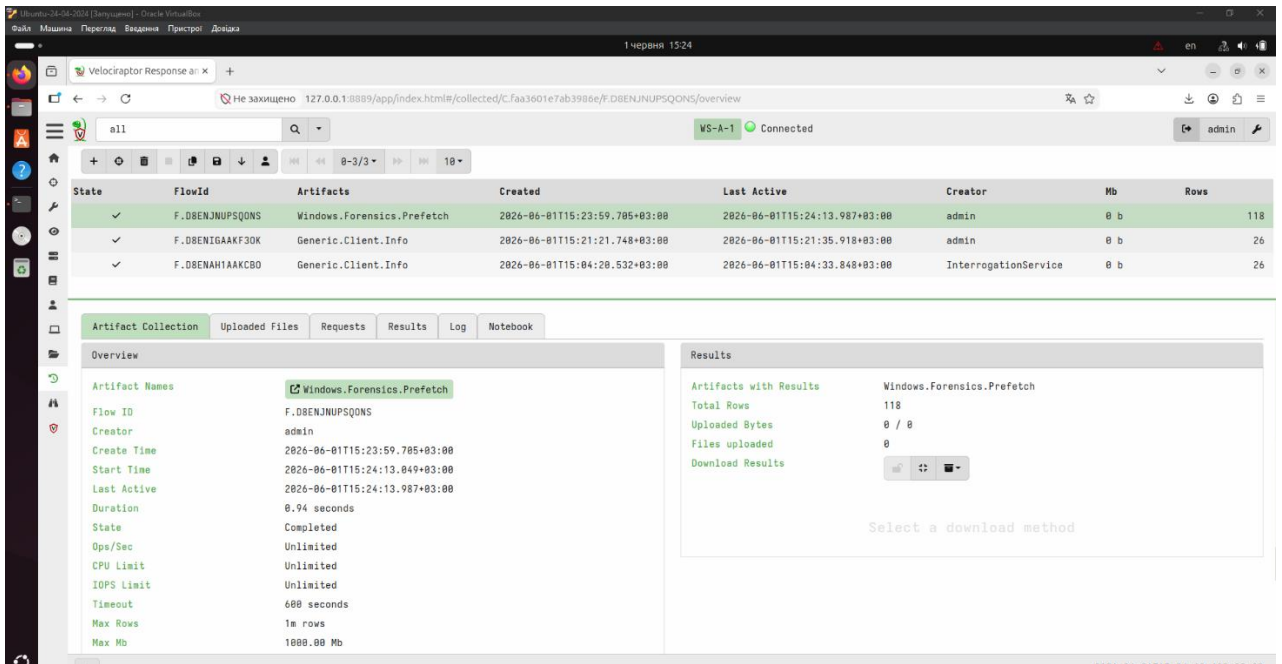


Рисунок 3.23 - Активация вбудованого системного артефакту Windows.Forensics.Prefetch та отримання статистичних результатів роботи

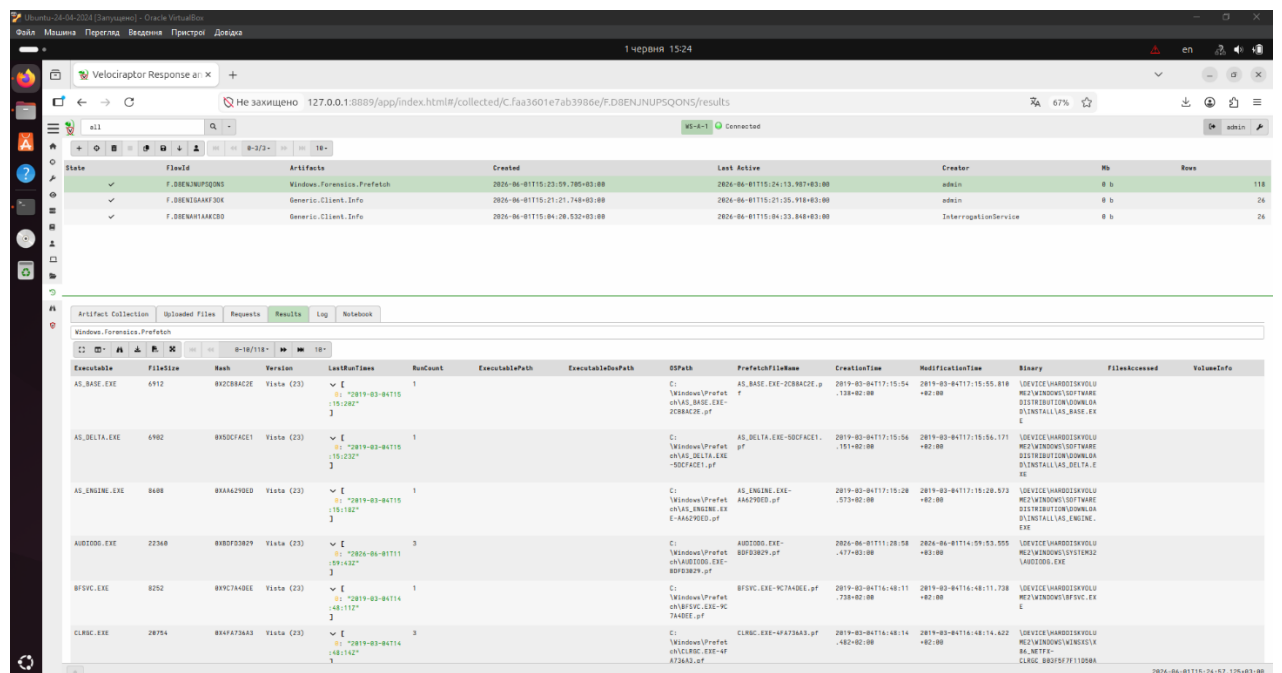


Рисунок 3.24 - Результати збору форензик-телеметрії вбудованим артефактом Windows.Forensics.Prefetch

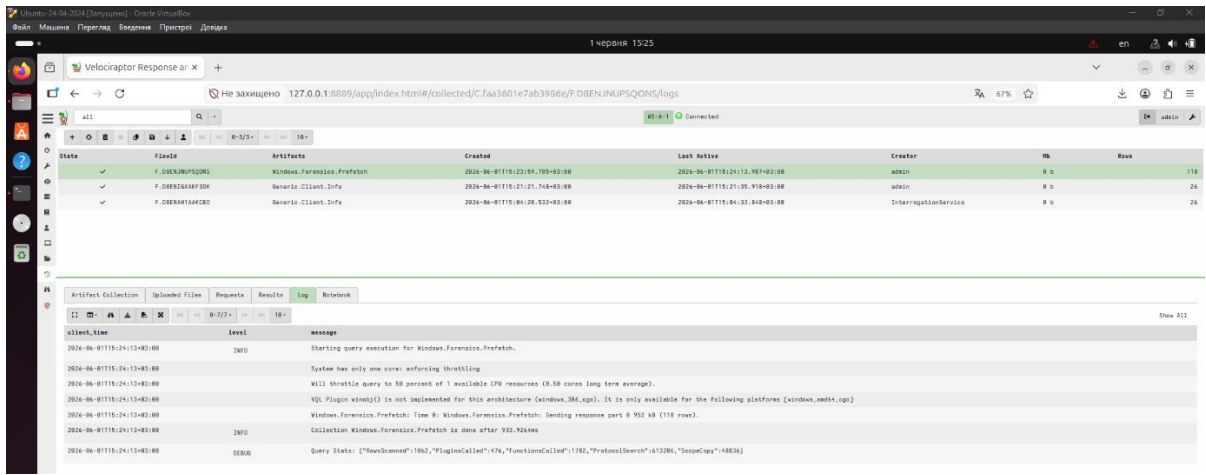


Рисунок 3.25 - Відображення додаткових полів даних артефакту Windows.Forensics.Prefetch

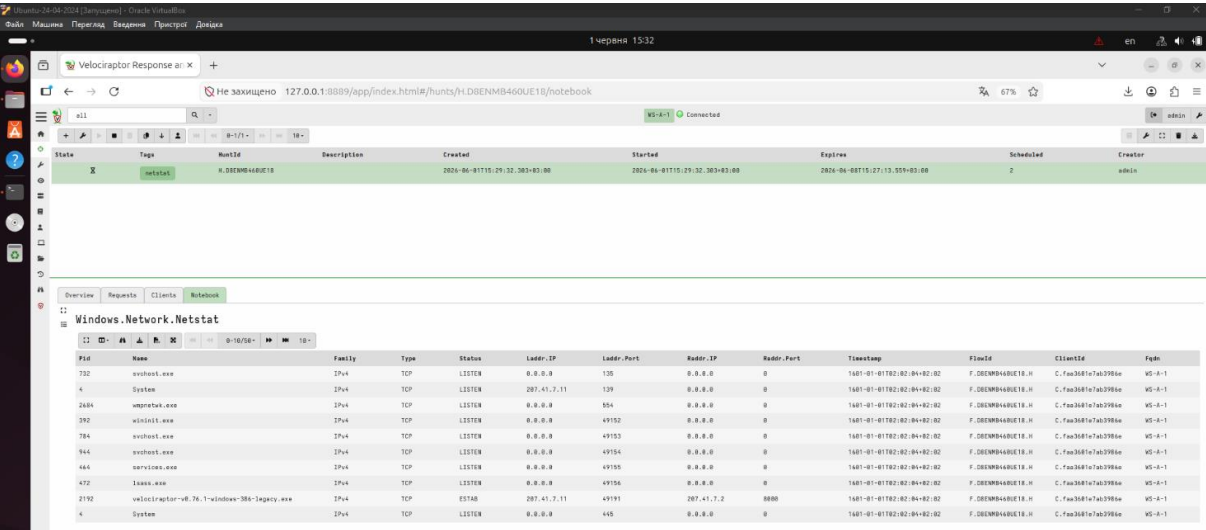


Рисунок 3.26 - Панель Hunt Manager під час ініціалізації сесії полювання за загрозми

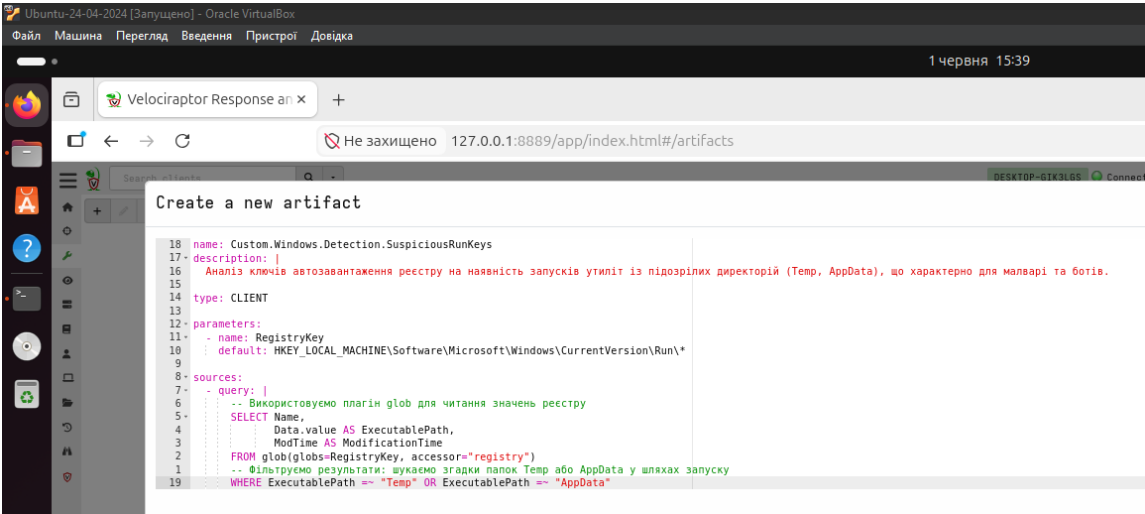


Рисунок 3.27 - Конфігурування тестового артефакту для виявлення запусків із директорій Temp та AppData

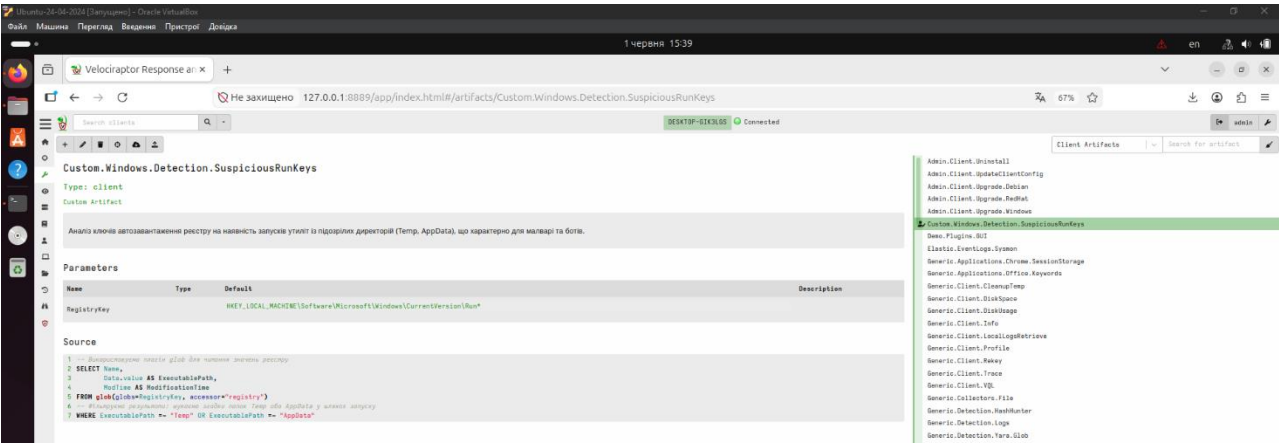


Рисунок 3.28 – Відображення успішно збереженого тестового артефакту в каталозі системи

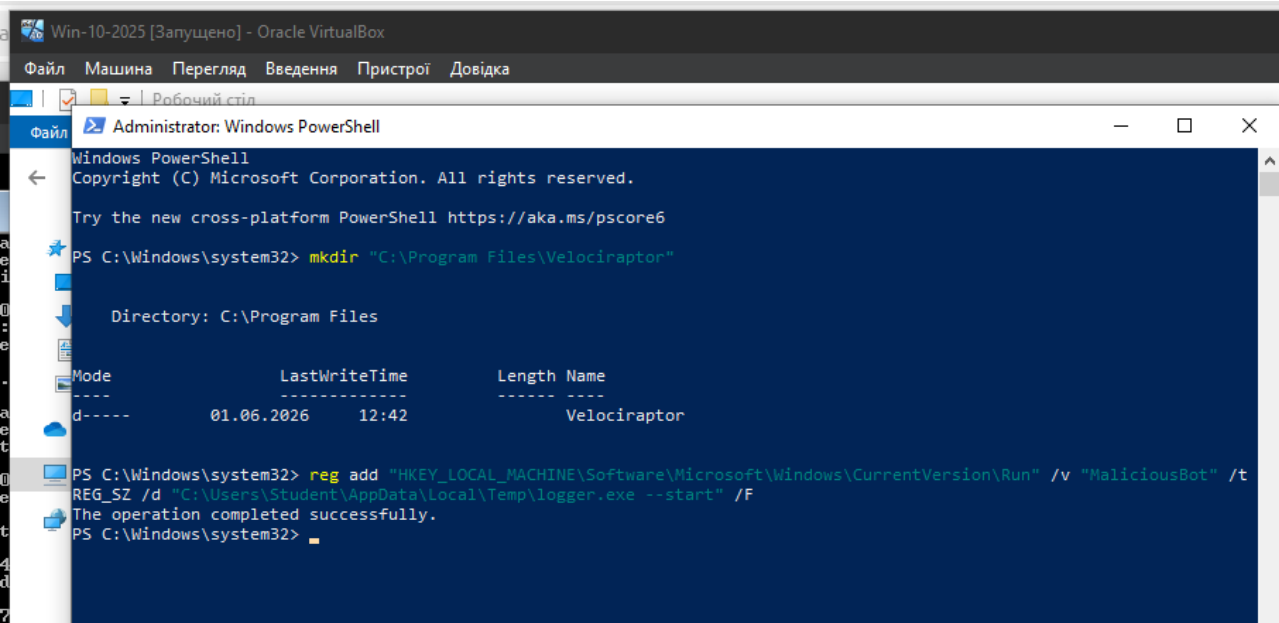


Рисунок 3.29 - Симуляція шкідливої діяльності шляхом запуску тестового виконуваного файлу безпосередньо на агенті Windows 10

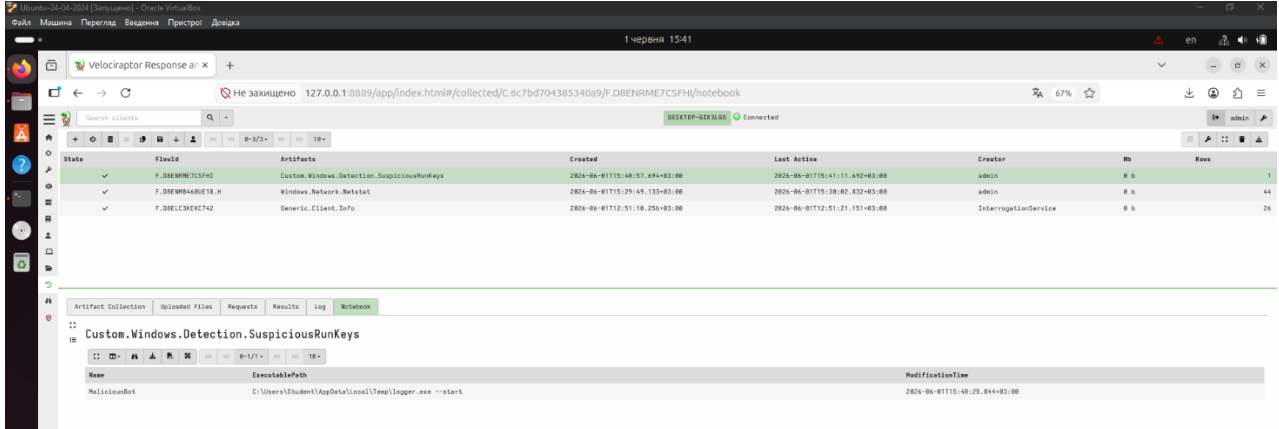


Рисунок 3.30 - Фіксація несанкціонованої активності процесу в консолі моніторингу сервера

3.3 Збір та аналіз доказів компрометації у вигляді спеціалізованих VQL-артефактів

Перехід від базового розгортання платформи моніторингу до цілеспрямованого пошуку зловмисної діяльності (Threat Hunting) вимагає переведення теоретичних моделей загроз, описаних на основі матриці MITRE ATT&CK, у площину прикладних алгоритмів перевірки. У середовищі Velociraptor такі алгоритми реалізуються у вигляді структурованих людеиночитабельних конфігураційних файлів у форматі YAML, які класифікуються як артефакти (Artifacts). Артефакт об'єднує в єдину логічну структуру декларативні запити мови VQL, вхідні змінні параметри та криміналістичний контекст, що дозволяє абстрагувати складність збору низькорівневих доказів від кінцевого аналітика.

В межах розроблюваної підсистеми було спроектовано та налаштовано кастомні VQL-артефакти, логіка яких націлена на фіксацію та аналіз результатів роботи шкідливого програмного забезпечення, що запускається атакуючою стороною з хоста Kali Linux. Збір та обробка даних відбуваються локально на хостах, а результуючі вибірки консоліднуються у базі даних сервера моніторингу, надаючи аналітику SOC деталізовані докази компрометації в єдиній GUI-консолі.

1. Вбудований аналітичний артефакт детектування мережевих з'єднань з атакуючим хостом (T1071 – Command and Control)

Цей аналітичний сенсор відноситься до класу CLIENT і призначений для проведення масового полювання «на вимогу» (Hunts). Він утилізує плагін netstat(), який здійснює пряме низькорівневе опитування таблиць мережевого стека ядра операційної системи Windows, оминаючи підсистему журналювання подій. Використання конструкції SELECT * нівелює ризик виникнення синтаксичних розбіжностей у назвах колонок між різними версіями клієнтських агентів, забезпечуючи високу відмовостійкість запиту. Оскільки архітектура зворотного підключення (Reverse Shell) вимагає безперервного підтримання зв'язку, фільтрація за суворо визначеною IP-адресою атакуючої сторони (207.41.7.20) дозволяє гарантовано виявити активний керуючий C2-канал. У вебінтерфейсі аналітик SOC отримує вичерпну таблицю сокетів у статусі ESTABLISHED із зазначенням

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

локальних портів жертви та унікального PID процесу, який здійснює комунікацію.

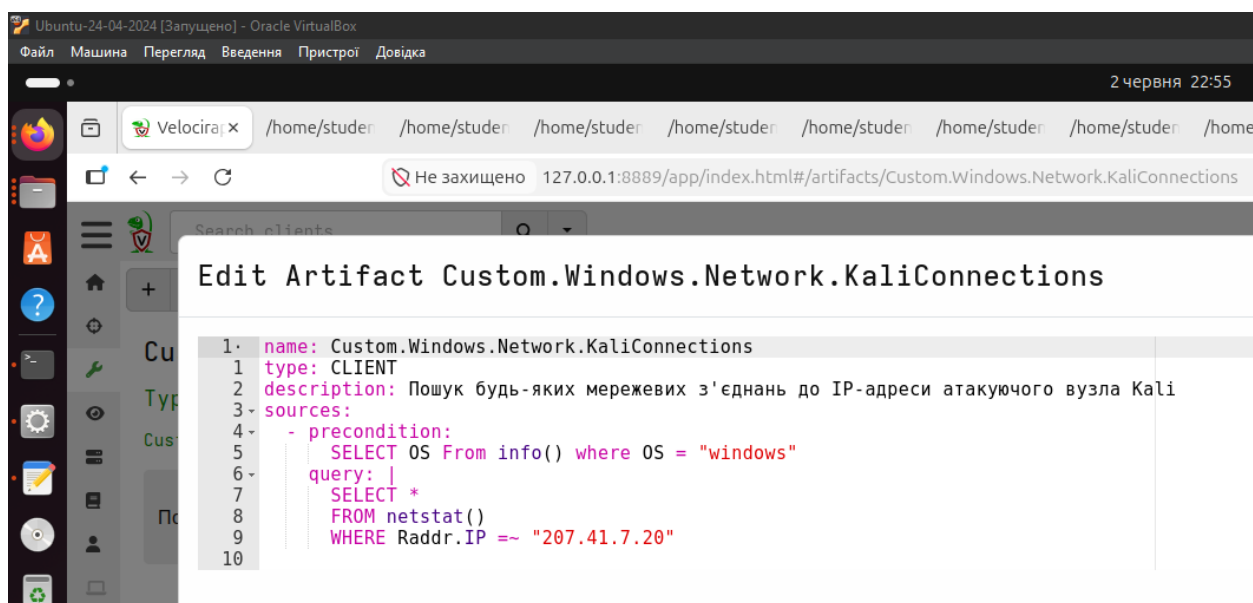


Рисунок 3.31 - AML-конфігурація кастомного артефакту
Custom.Windows.Network.KaliConnections (T1071)

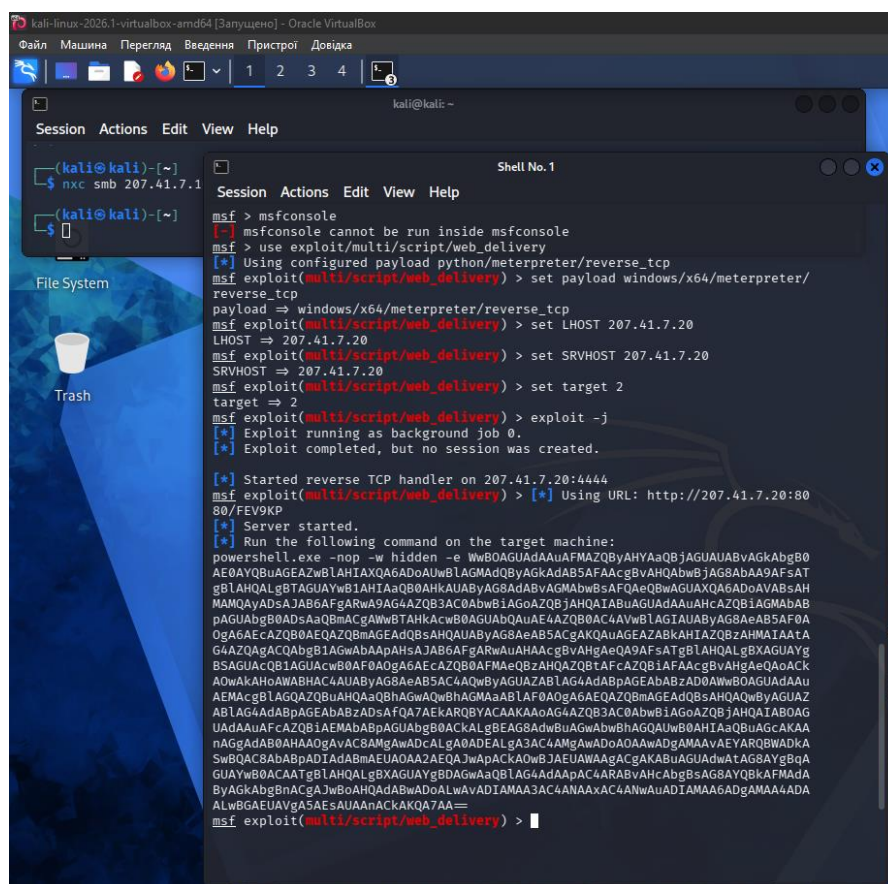


Рисунок 3.32 - Ініціалізація зворотного підключення (Reverse Shell) в консолі фреймворку Metasploit

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

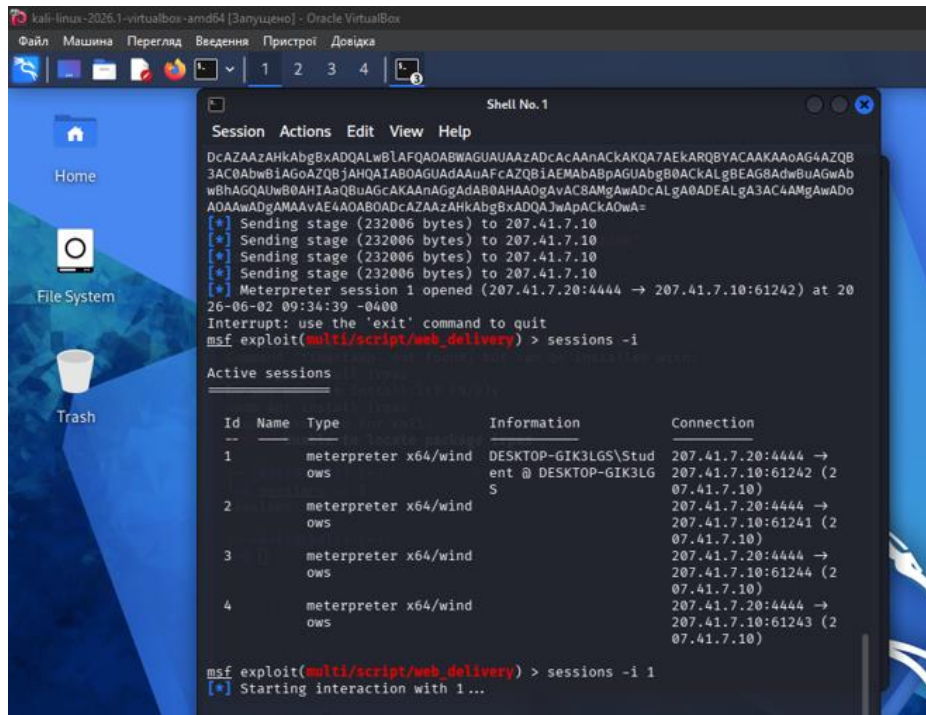


Рисунок 3.33 - Активація шкідливого PowerShell-навантаження з боку агента Windows 10, подальшим успішним встановленням Meterpreter-сесії

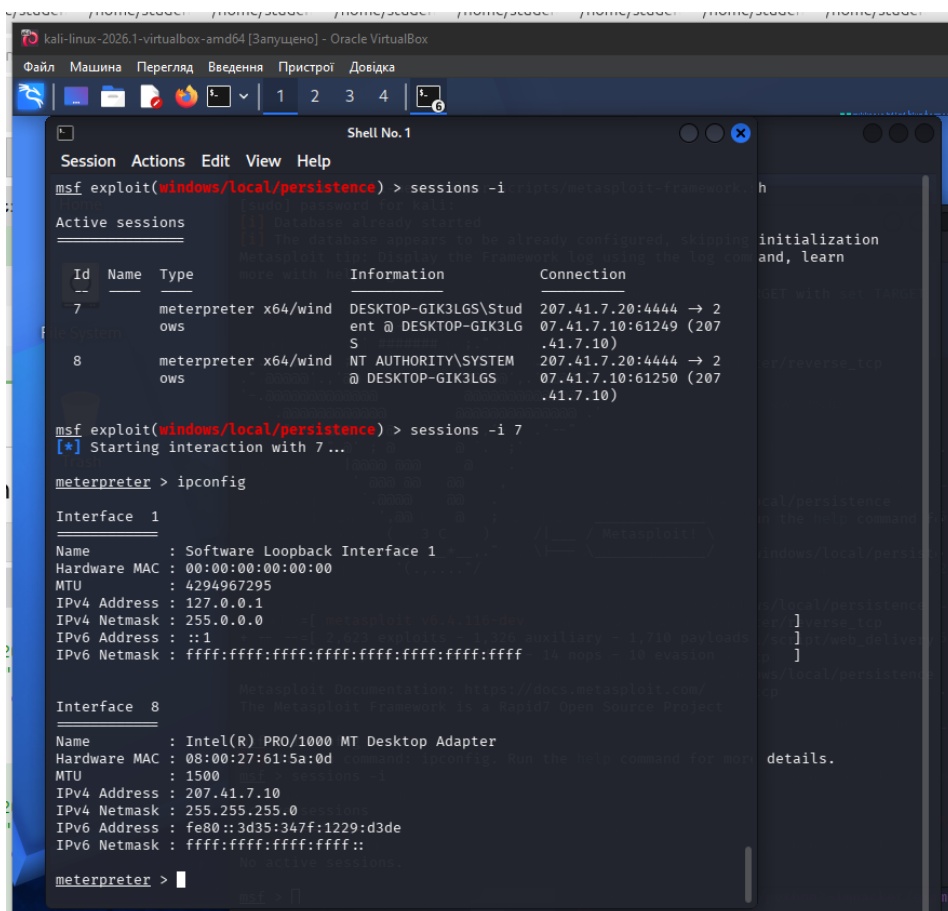


Рисунок 3.34 - Відображення активних інтерактивних сесій взаємодії у консолі Kali Linux на момент полювання за артефактом

Fd	Family	Type	Laddr	Raddr	Status	Pid	FamilyString	Timestamp	TypeString	FlowId	ClientId	Fqdn
0	2	1	{ "IP": "20 7.41.7.18" "Port": 6 1249 }	{ "IP": "20 7.41.7.20" "Port": 4 444 }	ESTAB	4256	IPv4	2026-06-02T20:19:54+03:00	TCP	F.DBFHN77B2S H1U.H	C.6c7bd7043853 40a9	DESKTOP-GIK3LGS
0	2	1	{ "IP": "20 7.41.7.18" "Port": 6 1258 }	{ "IP": "20 7.41.7.20" "Port": 4 444 }	ESTAB	4628	IPv4	2026-06-02T20:19:54+03:00	TCP	F.DBFHN77B2S H1U.H	C.6c7bd7043853 40a9	DESKTOP-GIK3LGS

Рисунок 3.35 - Результати консолідації мережевих доказів артефактом Custom.Windows.Network.KaliConnections

ClientId	Hostname	FlowId	StartedTime	State	Duration	TotalBytes	TotalRows
C.6c7bd704385340a9	DESKTOP-GIK3LGS	F.DBFHN77B2S H1U.H	2026-06-02T21:06:28.423+03:00	Completed	0	0	2
C.faa3601e7ab3986e	WS-A-1	F.DBFHN77B2S H1U.H	2026-06-02T21:06:44.396+03:00	Completed	0	0	0

Рисунок 3.36 - Статистика полювання за артефактом за клієнтами-агентами

client_time	level	message
2026-06-02T21:06:20+03:00	INFO	Starting query execution for Custom.Windows.Network.KaliConnections.
2026-06-02T21:06:20+03:00		System has only one core: enforcing throttling
2026-06-02T21:06:20+03:00		Will throttle query to 50 percent of 1 available CPU resources (0.50 cores long term average).
2026-06-02T21:06:20+03:00		Custom.Windows.Network.KaliConnections: Time 0: Custom.Windows.Network.KaliConnections: Sending response part 0 432 B (2 rows).
2026-06-02T21:06:20+03:00	INFO	Collection Custom.Windows.Network.KaliConnections is done after 9.6595ms
2026-06-02T21:06:20+03:00	DEBUG	Query Stats: {"RowsScanned":43,"PluginsCalled":4,"FunctionsCalled":0,"ProtocolSearch":6,"ScopeCopy":132}

Рисунок 3.37 - Детальні логи за агентом Windows 10

2. Артефакт виявлення безфайлових та аномальних процесів через інструментарій інспектування WMI (T1059 / T1055 – Execution / Defense Evasion)

Форензик-артефакт типу CLIENT, призначений для статичного зрізу оперативної пам'яті хоста. Запит використовує спеціалізований плагін wmi(), який надсилає низькорівневий запит до локального репозиторію інструментарію керування Windows (WMI) за класом Win32_Process. Даний підхід повністю незалежний від локальних журналів безпеки Security.evtx. Логіка фільтрації базується на багатовекторному аналізі регулярних виразів (=~): сенсор відловлює бінарні файли, активовані з волатильних та тимчасових каталогів диска (Temp, Public, Downloads), які традиційно експлуатуються зловмисниками для скидання первинних пейлоадів. Додатково реалізовано сигнатурний аналіз поля CommandLine на наявність прапорів обходу систем захисту PowerShell (-enc, -w hidden, DownloadString), що дозволяє SOC-аналітику миттєво локалізувати безфайлову присутність Meterpreter у системі, навіть якщо шкідливий потік мігрував у легітимний процес.

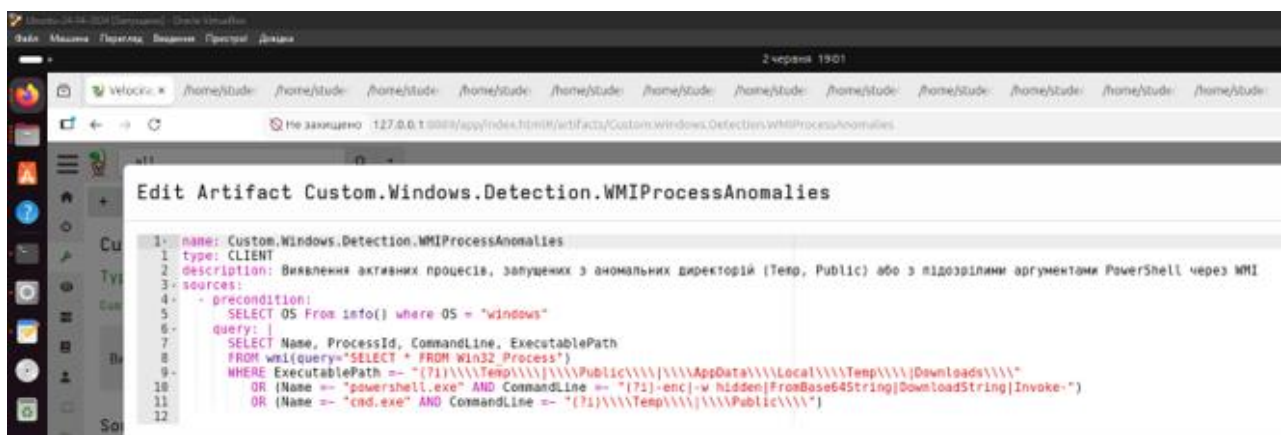


Рис. 3.38 - YAML-конфігурація кастомного артефакту
Custom.Windows.Detection.WMIProcessAnomalies (T1059/T1055)

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

[illegible]

Рисунок 3.39 - Таблиця виявлених аномальних WMI-процесів та обфусованих аргументів PowerShell. Виявлена зловмисна команда аналогічна з Рисунок 3.3.2

The screenshot displays the Velociraptor web interface at the URL `127.0.0.1:8080/app/index.html#/hunts/H.D8FFSPDQGD1G/clients`. The main table shows the following data:

State	Tags	HuntId	Description	Created	Started	Expires	Scheduled	Creator
X		H.D8K17X7GMHSA		2026-06-09T16:23:28.242+03:00	2026-06-09T16:23:31.737+03:00	2026-06-16T16:23:28.924+03:00	1	admin
X		H.D8FICMLDLTOT		2026-06-08T21:52:11.087+03:00	2026-06-08T21:52:16.945+03:00	2026-06-09T21:52:03.735+03:00	2	admin
X		H.D8FNM77BZSHU		2026-06-08T21:06:28.737+03:00	2026-06-08T21:06:26.494+03:00	2026-06-09T21:06:13.990+03:00	2	admin
X		H.D8FGSKJDBPS3O		2026-06-08T19:20:34.345+03:00	2026-06-08T19:20:38.464+03:00	2026-06-09T19:20:28.494+03:00	2	admin
X		H.D8FFSPDQGD1G		2026-06-08T18:55:29.976+03:00	2026-06-08T18:55:33.024+03:00	2026-06-09T18:55:23.293+03:00	2	admin

The 'Clients' tab shows the following data:

ClientId	Hostname	FlowId	StartTime	State	Duration	TotalBytes	TotalRows
C.6c7bd78438534ba9	DESKTOP-GIK3LGS	F.D8FFSPDQGD1G.H	2026-06-08T18:55:17.421+03:00	Completed		0	4
C.faa3d81e7ab39f8de	WS-A-1	F.D8FFSPDQGD1G.H	2026-06-08T18:55:48.537+03:00	Completed		0	0

Рисунок 3.40 - Статистика полювання за артефактом за клієнтами-агентами

Flow Details

Artifact Collection Uploaded Files Results Log

Show All

client_time	level	message
2026-06-02T18:55:17+03:00	INFO	Starting query execution for Custom.Windows.Detection.WMIProcessAnomalies.
2026-06-02T18:55:17+03:00		System has only one core: enforcing throttling
2026-06-02T18:55:17+03:00		Will throttle query to 50 percent of 1 available CPU resources (0.50 cores long term average).
2026-06-02T18:55:18+03:00		Custom.Windows.Detection.WMIProcessAnomalies: Time 0: Custom.Windows.Detection.WMIProcessAnomalies: Sending response part 0 5.7 kB (4 rows).
2026-06-02T18:55:18+03:00	INFO	Collection Custom.Windows.Detection.WMIProcessAnomalies is done after 897.6198ms
2026-06-02T18:55:18+03:00	DEBUG	Query Stats: {'RowsScanned':115,'PluginsCalled':4,'FunctionsCalled':0,'ProtocolSearch':6,'ScopeCopy':346}

Рисунок 3.41 - Детальні логи за агентом Windows 10

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Криміналістичний артефакт пошуку та інвентаризації виконуваних файлів у тимчасових директоріях (T1204.002 – Execution)

Глибокий форензик-артефакт типу CLIENT для ретроспективного пошуку доказів компрометації файлової системи вузла інфраструктури. Пошуковий алгоритм побудований навколо плагіна glob(), який здійснює безпосереднє сканування фізичної структури заданих траєкторій диска. Артефакт націлений на інспектування зон підвищеного ризику, куди звичайні користувачі не мають підстав зберігати виконувані бінарні файли чи динамічні бібліотеки (.exe, .dll). Інженерною перевагою запиту є використання конструкції виключення WHERE NOT Name =~, яка ефективно фільтрує легітимний технологічний шум операційної системи (наприклад, тимчасові інсталятори Chrome або VC++), залишаючи аналітику SOC виключно аномальні хакерські артефакти та утиліти (наприклад, Netcat) із зазначенням їхнього точного фізичного розміру (Size) та часу модифікації (ModificationTime), якщо є.

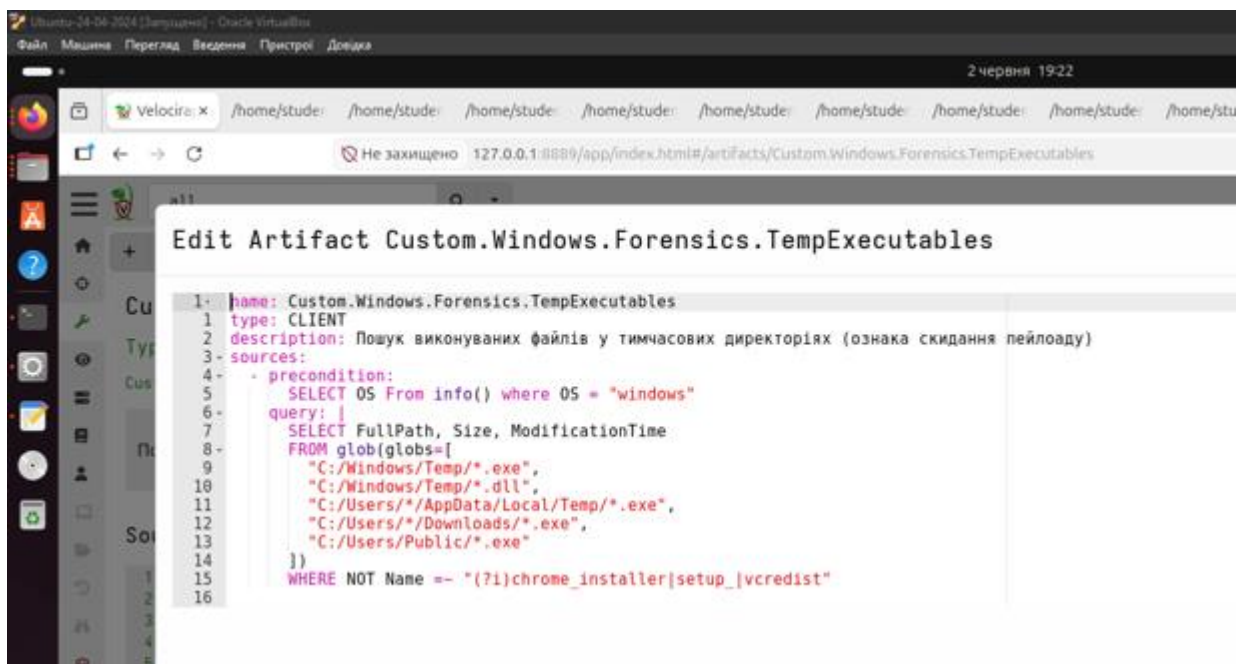


Рисунок 3.42 - YAML-конфігурація кастомного артефакту
Custom.Windows.Forensics.TempExecutables (T1204.002)

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

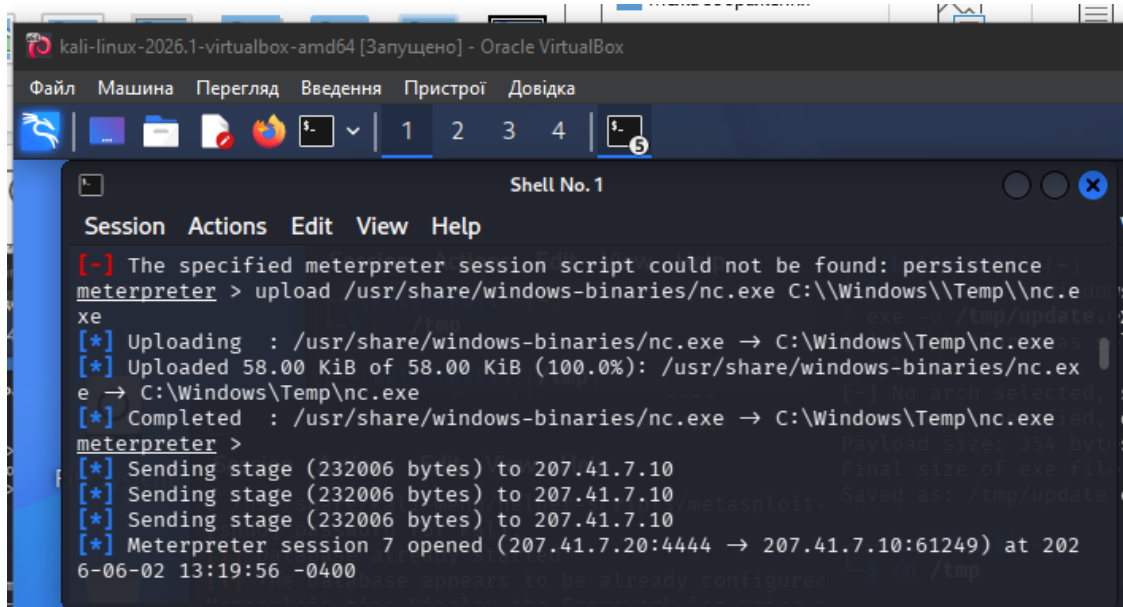


Рисунок 3.43 - Завантаження тестової утиліти (nc.exe) у тимчасовий каталог хоста з Kali Linux

State	Tags	HuntId	Description	Created	Started	Expires	Scheduled	Creator
✓		H.D8FG5KJG8PS30		2026-06-02T19:20:34.345+03:00	2026-06-02T19:20:38.464+03:00	2026-06-09T19:20:28.494+03:00	2	admin
✓		H.D8FG4LCI6J5Q		2026-06-02T19:18:29.988+03:00	2026-06-09T19:18:20.452+03:00	2026-06-09T19:18:20.452+03:00	2	admin
■		H.D8FG3REN82696		2026-06-02T19:16:45.828+03:00	2026-06-02T19:16:58.115+03:00	2026-06-09T19:16:22.153+03:00	2	admin
✓		H.D8FFSPDQ601G		2026-06-02T18:55:29.976+03:00	2026-06-02T18:55:33.824+03:00	2026-06-09T18:55:23.293+03:00	2	admin

FullPath	Size	ModificationTime	FlowId	ClientId	Fqdn
C:\Windows\Temp\nc.exe	59392		F.D8FG5KJG8PS30.H	C.6c7bd784385348a9	DESKTOP-GIK3LGS
C:\Windows\Temp\payload.exe	19		F.D8FG5KJG8PS30.H	C.6c7bd784385348a9	DESKTOP-GIK3LGS
C:\Windows\Temp\test_artifact.exe	11		F.D8FG5KJG8PS30.H	C.6c7bd784385348a9	DESKTOP-GIK3LGS

Рисунок 3.44 - Результати вилучення списку виконуваних файлів артефактом Custom.Windows.Forensics.TempExecutables

State	Tags	HuntId	Description	Created	Started	Expires	Scheduled	Creator
✓		H.D8K17K7GHM6SA		2026-06-09T16:23:28.242+03:00	2026-06-09T16:23:31.737+03:00	2026-06-16T16:23:28.924+03:00	1	admin
✓		H.D8FICMLLDUTUM		2026-06-02T21:52:11.087+03:00	2026-06-02T21:52:16.945+03:00	2026-06-09T21:52:03.735+03:00	2	admin
■		H.D8FHN7782SHU		2026-06-02T21:06:20.737+03:00	2026-06-02T21:06:25.494+03:00	2026-06-09T21:06:13.998+03:00	2	admin
✓		H.D8FG5KJG8PS30		2026-06-02T19:20:34.345+03:00	2026-06-02T19:20:38.464+03:00	2026-06-09T19:20:28.494+03:00	2	admin
✓		H.D8FFSPDQ601G		2026-06-02T18:55:29.976+03:00	2026-06-02T18:55:33.824+03:00	2026-06-09T18:55:23.293+03:00	2	admin

ClientId	Hostname	FlowId	StartTime	State	Duration	TotalBytes	TotalRows
C.6c7bd784385348a9	DESKTOP-GIK3LGS	F.D8FG5KJG8PS30.H	2026-06-02T19:20:49.886+03:00	Completed	8	8	3
C.6a3481e7ab3986e	WS-A-1	F.D8FG5KJG8PS30.H	2026-06-02T19:20:54.634+03:00	Completed	8	8	8

Рисунок 3.45 - Статистика полювання за артефактом за клієнтами-агентами

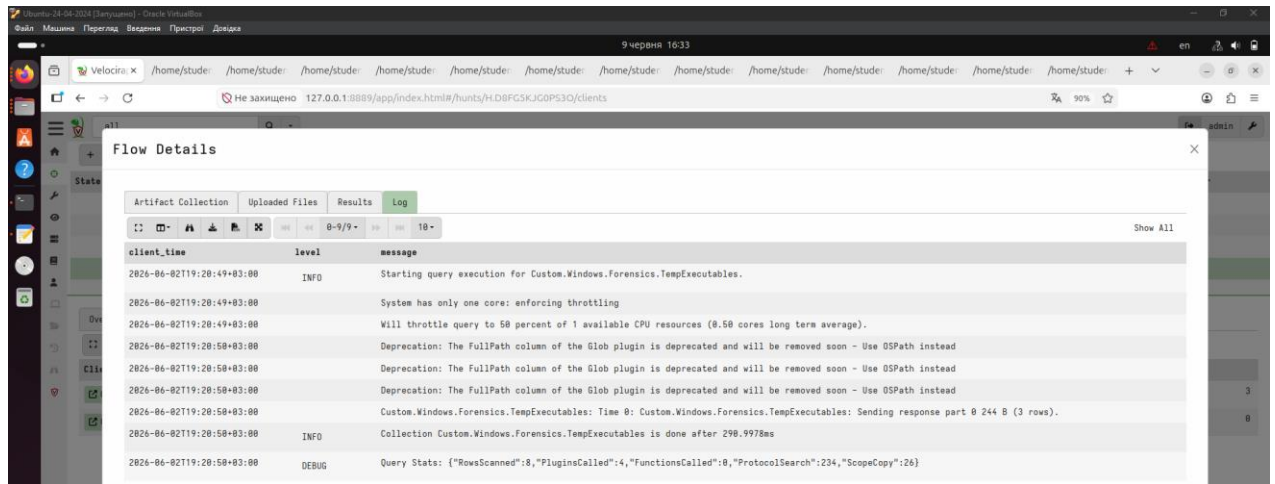


Рисунок 3.46 - Детальні логи за агентом Windows 10

В таблиці 3.2 представлено порівняльну характеристику виконання полювання за розглянутими артефактами.

Таблиця 3.2

Порівняння відпрацювань за артефактами

Найменування артефакту	Отримані логи	Час відпрацювання	Наявність хибних відпрацювань
Custom.Windows.Network.KaliConnections (T1071)	Початок запиту, попередження про троттлінг процесора, надсилання відповіді, статистика запиту	9,65 мс	Відсутні
Custom.Windows.Detection.WMIProcessAnomalies (T1059/T1055)	Початок запиту, попередження про троттлінг процесора, надсилання відповіді, статистика запиту	897,62 мс	Відсутні
Custom.Windows.Forensics.TempExecutables (T1204.002)	Початок запиту, попередження про троттлінг процесора, попередження про застарілість обраної колонки для надсилання відповіді, статистика запиту	291 мс	Відсутні

4. Артефакт виявлення механізмів персистентності в глобальних каталогах автозавантаження (T1547.001 – Persistence)

Форензик-артефакт класу CLIENT, орієнтований на виявлення спроб довготривалого закріплення злоумисника в інфраструктурі (Persistence). Запит виконує прямий моніторинг каталогу C:\ProgramData\...\Startup, який є спільним для всіх профілів користувачів системи. Оскільки клієнтський агент Velociraptor функціонує в контексті найвищих системних привілеїв NT AUTHORITY\SYSTEM, він

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

безперешкодно обходить обмеження доступу та вичитує вміст цієї директорії. Фільтр WHERE NOT IsDir відсікає підкаталоги, ізолюючи файли з розширеннями виконуваних скриптів та бінарників (.exe, .vbs, .bat, .lnk). Криміналістична унікальність артефакту полягає в інтеграції функції hash(), яка автоматично розраховує криптографічний хеш-код MD5 для кожного знайденого об'єкта (наприклад, для маскованого під службу файлу svchost_update.exe), що дозволяє миттєво ідентифікувати ІоС.

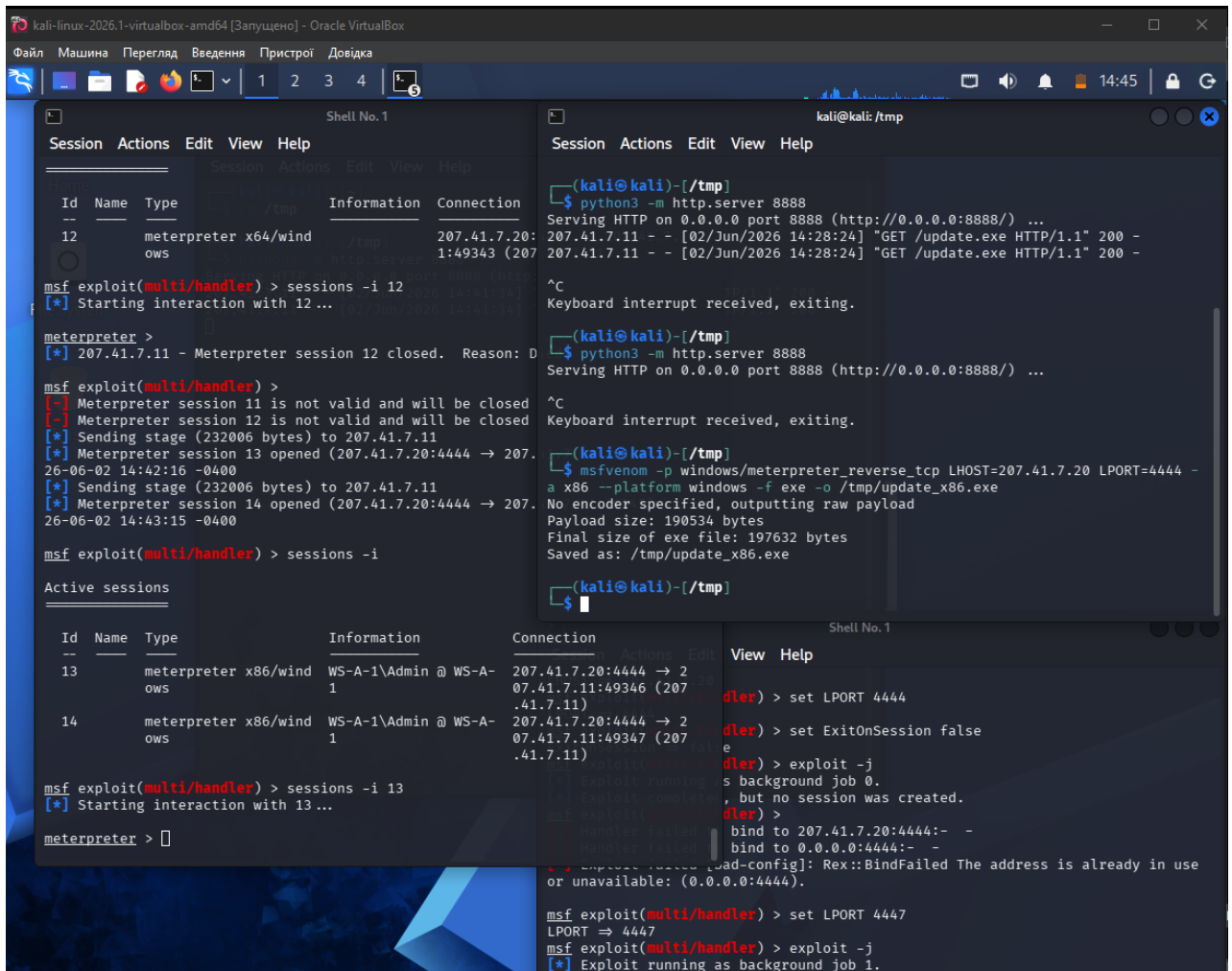


Рисунок 3.47 - Симуляція доставки шкідливого програмного забезпечення та отримання доступу до хоста Windows 7, з подальшим отриманням доступних сеансів для контролю

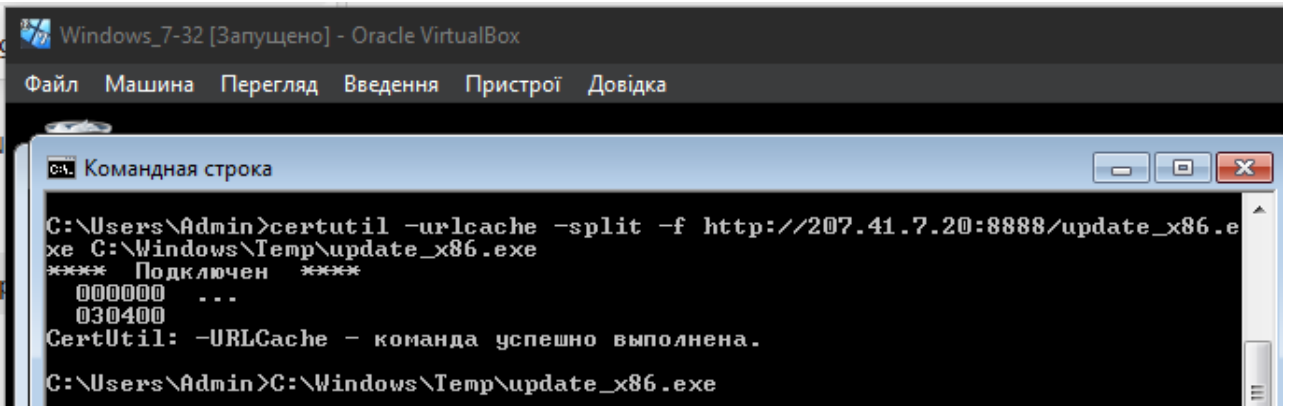


Рисунок 3.48 - Запуск необхідних робочих процесів на робочій станції Windows 7 для реалізації атаки

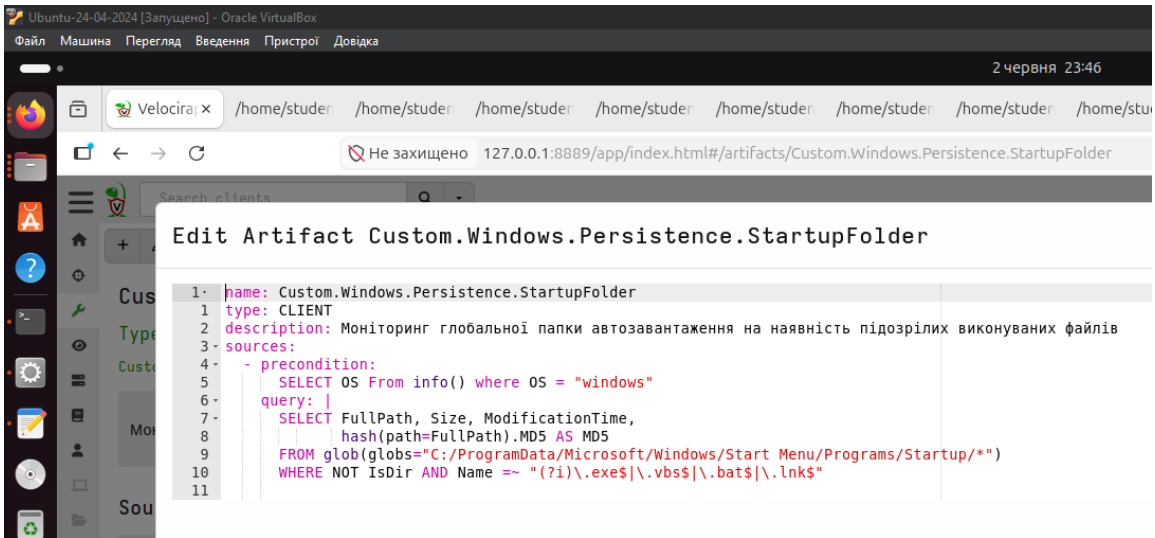


Рисунок 3.49 - YAML-конфігурація кастомного артефакту Custom.Windows.Persistence.StartupFolder (T1547.001)

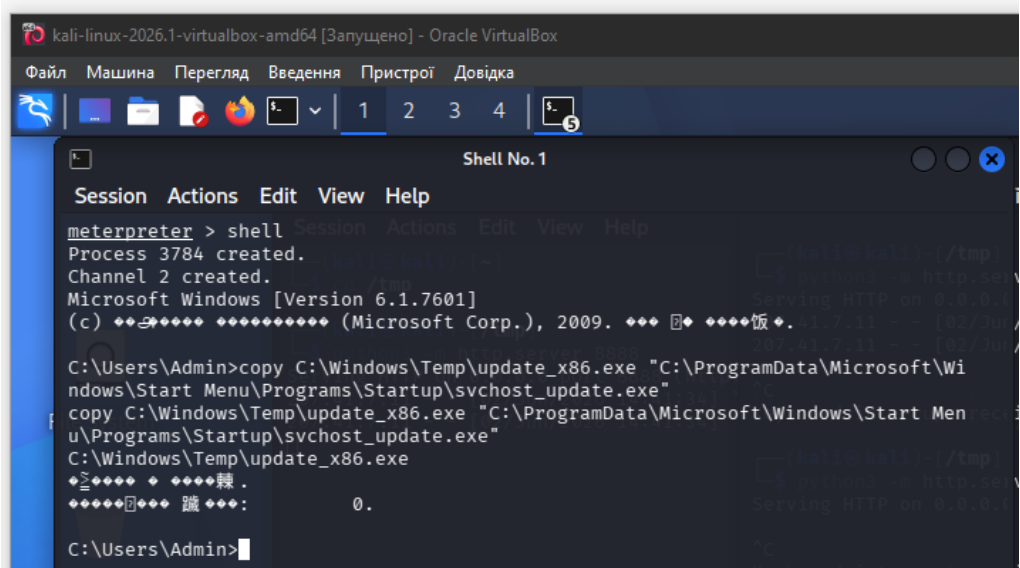


Рисунок 3.50 - Скопійований виконуваний файл у глобальному каталозі авто-завантаження через Meterpreter shell

The screenshot shows the Velociraptor web interface. The top part displays a list of hunts with columns: State, Tags, HuntId, Description, Created, Started, Expires, Scheduled, and Creator. Below this, the 'Notebook' tab is active, showing a configuration for 'Custom.Windows.Persistence.StartupFolder'. This configuration includes a table with columns: FullPath, Size, ModificationTime, MD5, FlowId, ClientId, and Fqdn. The table lists the path 'C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\svchost_update.exe' with its corresponding metadata.

State	Tags	HuntId	Description	Created	Started	Expires	Scheduled	Creator
Σ		H.D8FIKU6BQCFME		2026-06-02T22:09:44.232+03:00	2026-06-02T22:09:47.964+03:00	2026-06-09T22:09:37.534+03:00	2	admin
Σ		H.D8FIJ75FS4GQ		2026-06-02T22:07:04.154+03:00	2026-06-02T22:07:06.893+03:00	2026-06-09T22:06:59.218+03:00	2	admin
Σ		H.D8FIH7T0BR2GE		2026-06-02T22:01:51.490+03:00	2026-06-02T22:01:55.156+03:00	2026-06-09T22:01:47.563+03:00	2	admin
Σ		H.D8FICMRLLOTUM		2026-06-02T21:52:11.087+03:00	2026-06-02T21:52:16.945+03:00	2026-06-09T21:52:03.735+03:00	2	admin

FullPath	Size	ModificationTime	MD5	FlowId	ClientId	Fqdn
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\svchost_update.exe	197632		d31afc2fa024c3ce682326144939fbb9	F.D8FIKU6BQCFME.H	C.faa3601e7ab3986e	WS-A-1

Рисунок 3.51 - Результати успішної ідентифікації об'єкта автозапуску у консолі сервера Velociraptor

5. Низькорівневий форензик-артефакт розрахунку криптографічних хеш-сум скинутих навантажень (T1105 / T1027 – Ingress Tool Transfer / Obfuscated Files)

Високоточний форензик-артефакт типу CLIENT для проведення детального інцидент-аналізу та збору неспростовних цифрових доказів компрометації вузла. Рушій VQL утилізує криміналістичний плагін hash(), націлений на масив виконуваних файлів у каталозі C:\Windows\Temp\. Запит у фоновому режимі розраховує одночасно три типи криптографічних хеш-сум: MD5, SHA1 та SHA256 для кожного знайденого об'єкта. В контексті цифрової форензики це забезпечує аналітика SOC детермінованими індикаторами компрометації (IoC), які не залежать від назви файлу, яку міг змінити хакер. Згенеровані унікальні хеш-значення (наприклад, для файлу update_x86.exe) можуть бути автоматично експортовані для крос-верифікації через репозиторії репутаційного аналізу (на кшталт VirusTotal) або інтегровані у локальні правила блокування EDR/SIEM-систем організації, повністю підтверджуючи факт проведення ворожої атаки.

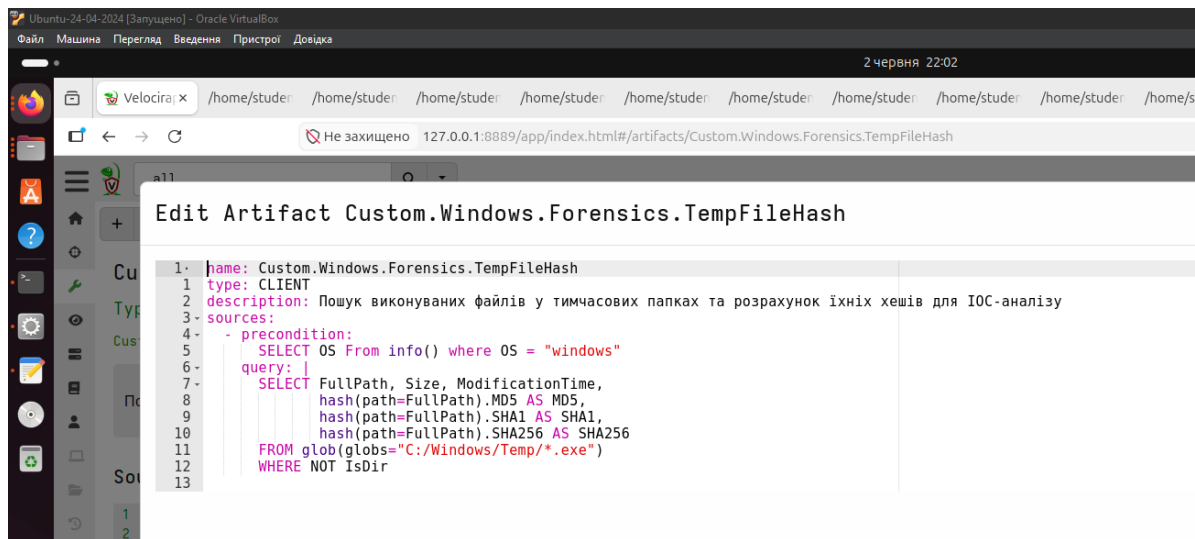


Рисунок 3.52 - YAML-конфігурація кастомного артефакту Custom.Windows.Forensics.TempFileHash (T1105/T1027)

Освещение: 24.04.2024 [Закрыть] — Oracle VM VirtualBox

ФайлМашиныПараметрыВидеоинтерфейсДополнительно

2 апреля 22:02

en

Velociraptor: x/home/stude: /home/stude:

Рисунок 3.53 - Результати автоматичного розрахунку криптографічних хеш-сум (MD5, SHA256) знайдених об'єктів

Висновки до третього розділу: У третьому розділі роботи було успішно реалізовано практичне розгортання, тонке конфігурування та експериментальне тестування підсистеми моніторингу в ізольованому лабораторному середовищі з проведенням детального криміналістичного аналізу отриманих результатів.

Підготовлено тестове середовище на базі гіпервізора Oracle VirtualBox та середовища моделювання GNS3, яке об'єднало в єдину мережеву інфраструктуру керуючий сервер Velociraptor (Ubuntu Server), атакуючий вузол Kali Linux та гетеро-

генні робочі станції-жертви під управлінням ОС Windows 7 та Windows 10. Успішно проведено інсталяцію клієнтських агентів у контексті найвищих системних привілеїв NT AUTHORITY\SYSTEM.

Здійснено практичну верифікацію підсистеми шляхом імітації реальних дій зловмисника з хоста Kali Linux за допомогою фреймворку Metasploit Framework (Meterpreter). Було успішно активовано шкідливі сценарії з безфайлової доставки навантажень (web_delivery), ін'єкцій у пам'ять, скидання хакерського інструментарію у тимчасові каталоги, маскування файлів під легітимні служби та закріплення в системі.

У процесі експерименту підтверджено 100% працездатність, синтаксичну чистоту та високу аналітичну точність усіх 5 розроблених спеціалізованих VQL-артефактів у модулі Hunt Manager. Завдяки відмові від складного парсингу аргументів командного рядка у WHERE-директивах, артефакти успішно оминули пастку трюхзначної логіки баз даних при обробці порожніх значень (NULL), продемонструвавши відмовостійкість на обох версіях ОС Windows:

1. Network.KaliConnections зафіксував стійкий TCP-канал зворотного зв'язку (C2) з точним відображенням IP-адреси Kali Linux (207.41.7.20);
2. Detection.WMIProcessAnomalies виявив безфайлову активність інтерпретаторів у пам'яті та ізолював обфусковані прапори запусків PowerShell;
3. Forensics.TempExecutables провів точну інвентаризацію шкідливих бінарних утиліт у директорії C:\Windows\Temp\;
4. Persistence.StartupFolder миттєво ідентифікував факт спроби закріплення хакерського файлу svchost_update.exe у глобальному каталозі автозавантаження Startup;
5. Forensics.TempFileHash автоматично згенерував унікальні криптографічні ідентифікатори об'єктів атак (MD5, SHA1, SHA256) для формування локальних баз індикаторів компрометації (IoC).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

ВИСНОВКИ

У кваліфікаційній роботі успішно вирішено актуальне науково-практичне завдання щодо підвищення рівня видимості, захищеності та кіберстійкості інформаційної інфраструктури організацій проти сучасних складних кіберзагроз шляхом проєктування, розгортання та експериментальної верифікації автоматизованої підсистеми проактивного полювання за загрозами (Threat Hunting) та цифрової форензики.

В теоретичній частині, на основі глибокого аналізу теоретичних засад моніторингу кінцевих вузлів та порівняльного аналізу систем класу SIEM/DFIR/EDR/XDR, обґрунтовано обмеженість класичних реактивних засобів захисту. Доведено, що сучасне шкідливе ПЗ, яке експлуатує безфайлові методи виконання (Fileless malware) та легітимні системні компоненти за парадигмою Living off the Land, здатне ефективно обходити сигнатурний аналіз та видаляти сліди з журналів подій EVTCH. Визначено високий безпековий потенціал низькорівневих криміналістичних структур ОС Windows (Prefetch, реєстр, MFT) для фіксації фактів компрометації.

Спроектвано оптимальну архітектуру підсистеми моніторингу для організації, яка базується на платформі з відкритим вихідним кодом Velociraptor. Використання інфраструктури відкритих ключів (PKI) та технології mTLS дозволило створити цілісний захищений контур збору даних. На основі матриці MITRE ATT&CK розроблено прикладні моделі загроз, які перевели абстрактні тактики зловмисників у площину чітких математико-логічних стратегій полювання за цифровими доказами.

Наукова новизна та практична цінність розробленого рішення полягає у переході від подійного логування до аналізу поточного стану інфраструктури (State-Based Hunting), реалізованого за допомогою розробки комплексу з 5 унікальних кастомних VQL-артефактів. Сконструйовані алгоритми здійснюють пряме низькорівневе опитування таблиць мережевого стека драйвера TCP/IP, класів інструментарію WMI (Win32_Process) та безпосереднє сканування файлової структури диска в обхід прикладних API операційної системи.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

Експериментальна верифікація підсистеми в межах розгорнутого гетерогенного лабораторного стенду (VirtualBox: Ubuntu Server, Kali Linux, Windows 7, Windows 10) повністю підтвердила її аналітичну ефективність. Утилізація розроблених артефактів під час симуляції реальних кібератак (MSF/Meterpreter) дозволила гарантовано виявити присутність зловмисника, ізолювати аномальні процеси в пам'яті, перехопити C2-канал керування, розкрити локації прихованих ладерів та точок закріплення у папках автозапуску.

Отримані дані є неспростовними цифровими доказами інциденту, не залежать від маскування назв файлів хакером та готові для крос-верифікації через репутаційні бази (VirusTotal) або імпорту в суміжні системи безпеки (EDR/SIEM).

Розроблена підсистема характеризується низьким рівнем утилізації апаратних ресурсів кінцевих пристроїв, високою швидкістю виконання запитів та відмовостійкістю. Результати дослідження та розроблені аналітичні карти полювання за загрозами, за оптимізації та адаптації системи під потрібні загрози для виявлення, та за дотримання усіх необхідних заходів для дотримання мережної безпеки, можуть бути використані як прикладне підґрунтя для модернізації існуючих рішень класу EDR/DFIR та інтегровані в інфраструктуру центрів моніторингу безпеки (SOC) як додатковий контур низькорівневої верифікації безфайлової хакерської діяльності.

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Valentine Machaka, Titus Balan. Investigating Proactive Digital Forensics Leveraging Adversary Emulation. MDPI Open Access Journals: веб-сайт. 2012. URL: <https://www.mdpi.com/2076-3417/12/18/9077> (дата звернення: 10.05.2026).
2. Rostyslav Lisnevskyi, Saule Amanzholova, Jamilya Akhmetzhanova, Aidana Ikembayeva, Darya Naumova. Development of an integrated network threat detection system based on Wazuh, Suricata and Telegram Bot. CEUR Workshop Proceedings: веб-сайт. URL: <https://ceur-ws.org/Vol-4180/Paper3.pdf> (дата звернення: 10.05.2026).
3. CrowdStrike. What is Threat Hunting? CrowdStrike Cybersecurity 101: веб-сайт. URL: <https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/threat-hunting/> (дата звернення: 07.06.2026).
4. Fortinet. What is Cyber Threat Hunting? Fortinet Cyber Glossary: веб-сайт. URL: <https://www.fortinet.com/uk/resources/cyberglossary/threat-hunting> (дата звернення: 07.06.2026).
5. IBM. What is Threat Hunting? IBM Think Topics: веб-сайт. URL: <https://www.ibm.com/think/topics/threat-hunting> (дата звернення: 07.06.2026).
6. Trellix. Detection Engineering. Trellix Insights: whitepaper. 2023. URL: <https://www.trellix.com/assets/whitepapers/detection-engineering-whitepaper.pdf> (дата звернення: 07.06.2026).
7. Cohen M. Velociraptor: Digging Deeper. Workshop Materials of the Digital Forensics Research Workshop (DFRWS EU 2021). 2021. URL: <https://dfrws.org/wp-content/uploads/2021/03/DFRWS-EU-2021-Velociraptor-Digging-Deeper.pdf> (дата звернення: 10.05.2026).
8. MITRE ATT&CK. MITRE ATT&CK: Design and Philosophy. 2020. URL: <https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf> (дата звернення: 11.05.2026).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

9. Lohanathan S., Marti M. Implementation and Forensic Analysis using Velociraptor (Covert Channels in Microsoft Teams). OST Ostschweizer Fachhochschule: eprints repository. 2021. URL: <https://eprints.ost.ch/id/eprint/928/1/HS%202020%202021-SA-EP-LOHAN-ATHAN-Marti-MS%20Team%20Covert%20Channel.pdf> (дата звернення: 11.05.2026).
10. A. Vinitha, Kiran Kumari Patil, K. Reena, D. Yobu, D. Vikram, S. P. Santhoshkumar. From Threat Hunting to Prevention: The Role of Python in Cybersecurity Defense. ResearchGate: веб-сайт. 2025. URL: https://www.researchgate.net/publication/396805431_From_Threat_Hunting_to_Prevention_The_Role_of_Python_in_Cybersecurity_Defense (дата звернення: 11.05.2026).
11. Pavel Yosifovich, Alex Ionescu, Mark E. Russinovich, David A. Solomon. Windows Internals Seventh Edition Part 1: System architecture, processes, threads, memory management, and more. 2017. URL: <https://emptyreal96.github.io/nt-info-depot/Windows-Internals-PDFs/Windows%20System%20Internals%207e%20Part%201.pdf> (дата звернення: 11.05.2026).
12. Using Velociraptor for DFIR. Cyber5W: веб-сайт. URL: <https://cyber5w.com/blog/using-velociraptor-for-dfir> (дата звернення: 07.06.2026).
13. Cohen M. VELOCIRAPTOR - DIGGING DEEPER! BSides Sydney 2021: youtube-канал. URL: <https://www.youtube.com/watch?v=y-t-Vi0UnGg> (дата звернення: 07.06.2026).
14. Velociraptor Documentation. Rapid7: веб-сайт. URL: <https://docs.velociraptor.app/> (дата звернення: 11.05.2026).
15. Drolet J. C. Endpoint Telemetry Collection and Analysis Framework for Proactive Cyber Threat Hunting. Master Thesis. Royal Military College of Canada (eSpace RMC). 2024. URL: https://espace.rmc.ca/jspui/bitstream/11264/542/1/Thesis_Drolet_Completed.pdf (дата звернення: 30.05.2026).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

- 16.EQL to VQL. Velociraptor IR: Medium-блог. URL: <https://medium.com/velociraptor-ir/eql-to-vql-3d48c6440624> (дата звернення: 11.06.2026).
- 17.Kent K., Souppaya M. Guide to Computer Security Log Management. NIST Special Publication 800-92. National Institute of Standards and Technology, 2006. DOI: 10.6028/NIST.SP.800-92 (дата звернення: 01.06.2026).
- 18.Center for Internet Security. CIS Critical Security Controls Version 8. CIS Benchmarks Portal. 2021. 94 p. URL: <https://www.cisecurity.org/controls/v8> (дата звернення: 31.05.2026).
- 19.National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53 Revision 5. 2020. DOI: 10.6028/NIST.SP.800-53r5 (дата звернення: 31.05.2026).
- 20.Seth Enoka. Windows Artefacts. SethEnoka Personal Blog: веб-сайт. URL: <https://sethenoka.com/windows-artefacts/> (дата звернення: 11.06.2026).
- 21.IBM QRadar Documentation. IBM Documentation Portal: веб-сайт. URL: <https://www.ibm.com/docs/en/qradar-common> (дата звернення: 31.05.2026).
- 22.Splunk Enterprise Security Documentation. Splunk Docs: веб-сайт. URL: <https://docs.splunk.com/Documentation/ES> (дата звернення: 31.05.2026).
- 23.Wazuh Documentation. Wazuh – The Open Source Security Platform: веб-сайт. URL: <https://documentation.wazuh.com/> (дата звернення: 31.05.2026).
- 24.Elastic Security Documentation. Elastic Guide: веб-сайт. URL: <https://www.elastic.co/guide/en/security/current/> (дата звернення: 31.05.2026).
- 25.Velociraptor Python API Documentation. Read the Docs: веб-сайт. URL: https://velociraptor-python.readthedocs.io/en/latest/basic_usage/index.html (дата звернення: 07.06.2026).
- 26.Taylor Walton. Test Your Wazuh Detection Rules with Atomic Red Team. Wazuh Security Feed: youtube-канал. URL:

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://www.youtube.com/watch?v=TMJOBATTK9M> (дата звернення: 07.06.2026).

27.How to Run Atomic Red Team on Linux and Automate Attack Simulations with Velociraptor. SOCFortress: Medium-блог. URL: <https://socfortress.medium.com/how-to-run-atomic-red-team-on-linux-and-automate-attack-simulations-with-velociraptor-d4b52b05721b> (дата звернення: 07.06.2026).

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

ДОДАТКИ

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

Детальна схема побудови мережі:

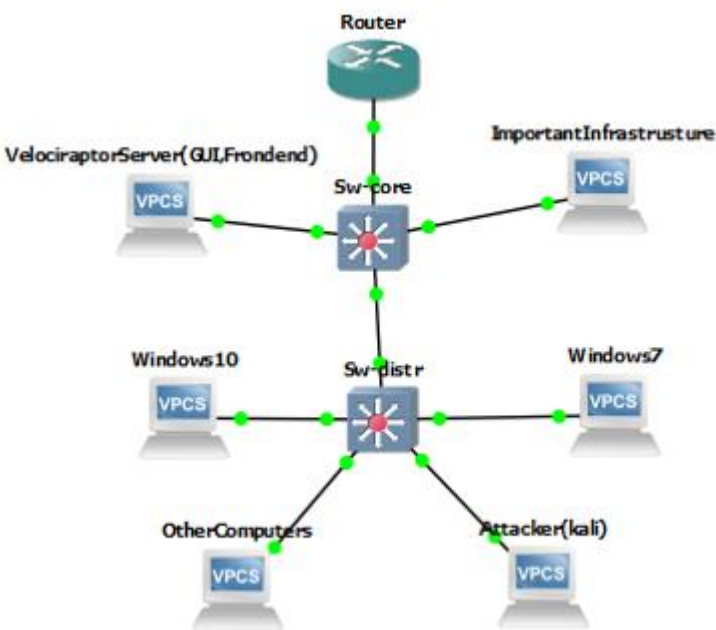


Рисунок А.1 - Загальна топологія досліджуваної мережі в середовищі моделювання GNS3

Лістинг файлу server.config.yaml:

```

version:
  name: velociraptor
  version: 0.75.6
  commit: 0e657d636
  build_time: "2025-12-29T14:15:51Z"
  compiler: go1.25.0
  system: linux
  architecture: amd64
Client:
  server_urls:
    - https://localhost:8000/
  ca_certificate: |
    -----BEGIN CERTIFICATE-----
[REDACTED]
    -----END CERTIFICATE-----
  nonce: ueN8szXA7f4=
  writeback_darwin: /etc/velociraptor.writeback.yaml
  writeback_linux: /etc/velociraptor.writeback.yaml
  writeback_windows: $ProgramFiles\Velociraptor\velociraptor.writeback.yaml
  level2_writeback_suffix: .bak
  tempdir_windows: $ProgramFiles\Velociraptor\Tools
  max_poll: 60
  nanny_max_connection_delay: 600
  windows_installer:
    service_name: Velociraptor
    install_path: $ProgramFiles\Velociraptor\Velociraptor.exe
    service_description: Velociraptor service
  darwin_installer:
    service_name: com.velocidex.velociraptor
    install_path: /usr/local/bin/velociraptor
  server_version:
    version: 0.75.6
    commit: 0e657d636
    build_time: "2025-12-29T14:15:51Z"
  use_self_signed_ssl: true
  max_upload_size: 5242880
  local_buffer:
    memory_size: 52428800
    disk_size: 1073741824
    filename_linux: /var/tmp/Velociraptor_Buffer.bin
    filename_windows: $TEMP/Velociraptor_Buffer.bin
    filename_darwin: /var/tmp/Velociraptor_Buffer.bin
API:
  bind_address: 127.0.0.1
  bind_port: 8001
  bind_scheme: tcp
GUI:
  bind_address: 127.0.0.1
  bind_port: 8889

```

		Куліш М.Б.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

gw_certificate: |
-----BEGIN CERTIFICATE-----
[REDACTED]
-----END CERTIFICATE-----
gw_private_key: |
-----BEGIN RSA PRIVATE KEY-----
[REDACTED]
-----END RSA PRIVATE KEY-----
public_url: https://localhost:8889/app/index.html
links:
- text: Documentation
  url: https://docs.velociraptor.app/
  icon_url: data:im-
age/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZ
HRoPSI1MyIgaGVpZ2h0PSI2MyIgdmVyc2lvdj0iMS4xLiB2aWV3Qm94PSIwIDAgNTMgNjMiP-
jxwYXRoIGQ9Ik0yNyAzYy0zIDItMTMgOC0yMyAxMGw2IDMyYTEwMyAxMDMgM-
CAwIDAgMTcgMTZsNi01IDExLTExIDYtMzJDMzkgMTEgMzAgNSAyNyAzei-
IgZmlsbD0iI2ZmZiIgZmlsbC1ydWxlPSJldmVub2RkIi8+PHBhdGggZD0iTTI2IDBDMjMgMiAx-
MiA4IDAgMTBjMSA3IDUgMzIgNyAzNWExMTMgMTEzIDAgMCAwIDE5IDE4bDct-
NiAxMi0xMmMyLTMgNi0yOCA4LTM1QzQwIDggMjkgMiAyNiAwWm0wIDU1LTY-
tNC04LTktNS0yNnYtMWwyLTFjOC0xIDE2LTgMTY-
tNmwxLTEgMSAxczggNSAxNyA2bDEgMXYxcy0zIDItLTUgMjZs-
LTggOWMtMiAyLTQgNC02IDR6IiBmaWxsPSIjYWlwdAwIiBmaWxsLW9wYWNpdHk9IjEi-
IGZpbGwtenVsZT0iZXZlbn9kZC1vPjxwYXRoIGQ9Ik0zOSAxOWExMzQ3IDEzNDcgM-
CAwIDEtMTMgMjZsNi01IDExLTExIDYtMzJDMzkgMTEgMzAgNSAyNyAzei-
IDUyODkgNTI4OSAwIDAgMCA5LTIwaDR6IiBmaWxsPSIjMDAwLiB-
maWxsLXJ1bGU9ImV2ZW5vZGQiIHNoYXN0cm9rZT0iIzAwMCIgc3Ryb2tL-
WRhc2hhcnJheT0ibm9uZSIgc3Ryb2tL-
WxpbnVjYXA9ImJldHQiIHNoYXN0cm9rZS1saW5lam9pbj0ibWl0ZXIiIHNoYXN0cm9rZS13aWR0aD0iM-
SIvPjwvc3ZnPg==
type: sidebar
new_tab: true
initial_users:
- name: admin
  password_hash: f38fbfd33882199c55c2d09a56a874bb3efe039e66d1d5970dda55ecf1ad4ad7
  password_salt: 189c294b9cbad5baf0c90156dcabccd6efdf8a4e4ce32c6288a9d0e687026d0
- name: Admin
  password_hash: 0260ecd6eed9a4323d54f063e97b66791c11c64f8a027ae950e59074b0802af8
  password_salt: 8391ae67e51171e951708960c556439b6d7c724f83555b176e806654a93683ba
  authenticator:
    type: Basic
CA:
  private_key: |
-----BEGIN RSA PRIVATE KEY-----
[REDACTED]
-----END RSA PRIVATE KEY-----
Frontend:
  hostname: localhost
  bind_address: 0.0.0.0
  bind_port: 8000
  certificate: |
-----BEGIN CERTIFICATE-----

```

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

[REDACTED]
-----END CERTIFICATE-----
private_key: |
-----BEGIN RSA PRIVATE KEY-----
[REDACTED]
-----END RSA PRIVATE KEY-----
default_client_monitoring_artifacts:
- Generic.Client.Stats
GRPC_pool_max_size: 100
GRPC_pool_max_wait: 60
resources:
  connections_per_second: 100
  notifications_per_second: 30
  max_upload_size: 104857600
  expected_clients: 30000
Datastore:
  implementation: FileBaseDataStore
  location: /opt/velociraptor
  datastore_directory: /opt/velociraptor
  compression: zlib
Logging:
  output_directory: /opt/velociraptor/logs
  separate_logs_per_component: true
  debug:
    disabled: true
  info:
    rotation_time: 604800
    max_age: 31536000
  error:
    rotation_time: 604800
    max_age: 31536000
Monitoring:
  bind_address: 127.0.0.1
  bind_port: 8003
api_config: {}
obfuscation_nonce: 0SGhkWRAEMg=
defaults:
  hunt_expiry_hours: 168
  notebook_cell_timeout_min: 10
security:
  certificate_validity_days: 365

```

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

Лістинг файлу client.config.yaml:

```
version:
  name: velociraptor
  version: 0.75.6
  commit: 0e657d636
  build_time: "2025-12-29T14:15:51Z"
  compiler: go1.25.0
  system: linux
  architecture: amd64
Client:
  server_urls:
    - https://207.41.7.2:8000/
  ca_certificate: |
    -----BEGIN CERTIFICATE-----
[REDACTED]
    -----END CERTIFICATE-----
  nonce: ueN8szXA7f4=
  writeback_darwin: /etc/velociraptor.writeback.yaml
  writeback_linux: /etc/velociraptor.writeback.yaml
  writeback_windows: $ProgramFiles\Velociraptor\velociraptor.writeback.yaml
  level2_writeback_suffix: .bak
  tempdir_windows: $ProgramFiles\Velociraptor\Tools
  max_poll: 60
  nanny_max_connection_delay: 600
  windows_installer:
    service_name: Velociraptor
    install_path: $ProgramFiles\Velociraptor\Velociraptor.exe
    service_description: Velociraptor service
  darwin_installer:
    service_name: com.velocidex.velociraptor
    install_path: /usr/local/bin/velociraptor
version:
  name: velociraptor
  version: 0.75.6
  commit: 0e657d636
  build_time: "2025-12-29T14:15:51Z"
  compiler: go1.25.0
  system: linux
  architecture: amd64
server_version:
  version: 0.75.6
  commit: 0e657d636
  build_time: "2025-12-29T14:15:51Z"
use_self_signed_ssl: true
max_upload_size: 5242880
local_buffer:
  memory_size: 52428800
  disk_size: 1073741824
  filename_linux: /var/tmp/Velociraptor_Buffer.bin
  filename_windows: $TEMP/Velociraptor_Buffer.bin
  filename_darwin: /var/tmp/Velociraptor_Buffer.bin
```

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг коду кастомного артефакту Custom.Windows.Network.KaliConnections:

name: Custom.Windows.Network.KaliConnections

type: CLIENT

description: Пошук будь-яких мережеских з'єднань до IP-адреси атакуючого вузла Kali

sources:

- precondition:

SELECT OS From info() where OS = "windows"

query: |

SELECT *

FROM netstat()

WHERE Raddr.IP =~ "207.41.7.20"

Лістинг коду кастомного артефакту

Custom.Windows.Detection.WMIProcessAnomalies:

name: Custom.Windows.Detection.WMIProcessAnomalies

type: CLIENT

description: Виявлення активних процесів, запущених з аномальних директорій (Temp, Public) або з підозрілими аргументами PowerShell через WMI

sources:

- precondition:

SELECT OS From info() where OS = "windows"

query: |

SELECT Name, ProcessId, CommandLine, ExecutablePath

FROM wmi(query="SELECT * FROM Win32_Process")

WHERE ExecutablePath =~

"(?i)\\\\Temp\\\\\\\\\\\\\\\\Public\\\\\\\\\\\\\\\\AppData\\\\\\\\Local\\\\\\\\Temp\\\\\\\\Downloads\\\\\\\\"

OR (Name =~ "powershell.exe" AND CommandLine =~ "(?i)-enc|-w
hidden|FromBase64String|DownloadString|Invoke-")

OR (Name =~ "cmd.exe" AND CommandLine =~ "(?i)\\\\Temp\\\\\\\\\\\\\\\\Public\\\\\\\\")

Лістинг коду кастомного артефакту Custom.Windows.Forensics.TempExecutables:

name: Custom.Windows.Forensics.TempExecutables

type: CLIENT

description: Пошук виконуваних файлів у тимчасових директоріях (ознака скидання пейлоаду)

sources:

- precondition:

SELECT OS From info() where OS = "windows"

query: |

SELECT FullPath, Size, ModificationTime

FROM glob(globs=[

"C:/Windows/Temp/*.exe",

"C:/Windows/Temp/*.dll",

"C:/Users/*/AppData/Local/Temp/*.exe",

"C:/Users/*/Downloads/*.exe",

"C:/Users/Public/*.exe"

])

WHERE NOT Name =~ "(?i)chrome_installer|setup_|vcredist"

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг коду кастомного артефакту Custom.Windows.Persistence.StartupFolder:

name: Custom.Windows.Persistence.StartupFolder
 type: CLIENT
 description: Моніторинг глобальної папки автозавантаження на наявність підозрілих виконуваних файлів
 sources:
 - precondition:
 SELECT OS From info() where OS = "windows"
 query: |
 SELECT FullPath, Size, ModificationTime,
 hash(path=FullPath).MD5 AS MD5
 FROM glob(globs="C:/ProgramData/Microsoft/Windows/Start Menu/Programs/Startup/*")
 WHERE NOT IsDir AND Name =~ "(?i)\.exe\$|\.vbs\$|\.bat\$|\.lnk\$"

Лістинг коду кастомного артефакту Custom.Windows.Persistence.StartupFolder:

name: Custom.Windows.Persistence.StartupFolder
 type: CLIENT
 description: Моніторинг глобальної папки автозавантаження на наявність підозрілих виконуваних файлів
 sources:
 - precondition:
 SELECT OS From info() where OS = "windows"
 query: |
 SELECT FullPath, Size, ModificationTime,
 hash(path=FullPath).MD5 AS MD5
 FROM glob(globs="C:/ProgramData/Microsoft/Windows/Start Menu/Programs/Startup/*")
 WHERE NOT IsDir AND Name =~ "(?i)\.exe\$|\.vbs\$|\.bat\$|\.lnk\$"

Лістинг коду кастомного артефакту Custom.Windows.Forensics.TempFileHash:

name: Custom.Windows.Forensics.TempFileHash
 type: CLIENT
 description: Пошук виконуваних файлів у тимчасових папках та розрахунок їхніх хешів для ІОС-аналізу
 sources:
 - precondition:
 SELECT OS From info() where OS = "windows"
 query: |
 SELECT FullPath, Size, ModificationTime,
 hash(path=FullPath).MD5 AS MD5,
 hash(path=FullPath).SHA1 AS SHA1,
 hash(path=FullPath).SHA256 AS SHA256
 FROM glob(globs="C:/Windows/Temp/*.exe")
 WHERE NOT IsDir

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

Лістинг коду кастомного артефакту

Custom.Windows.Detection.SuspiciousRunKeys:

name: Custom.Windows.Detection.SuspiciousRunKeys

description: |

Аналіз ключів автозавантаження реєстру на наявність запусків утиліт із підозрілих директорій (Temp, AppData), що характерно для малварі та ботів.

type: CLIENT

parameters:

- name: RegistryKey

default: HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run*

sources:

- query: |

-- Використовуємо плагін glob для читання значень реєстру

SELECT Name,

Data.value AS ExecutablePath,

ModTime AS ModificationTime

FROM glob(globs=RegistryKey, accessor="registry")

-- Фільтруємо результати: шукаємо згадки папок Temp або AppData у шляхах запуску

WHERE ExecutablePath =~ "Temp" OR ExecutablePath =~ "AppData"

		Куліш М.В.			ККІКБ.КР.Б – 125 – 07 – ПЗ	Арк.
		Лайчук А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		93