КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.29.865
УДК 004.056.5

Опірський Іван Романович
д.т.н., професор, завідувач кафедри захисту інформації
Національний Університет «Львівська Політехніка», Львів, Україна
ORCID ID: 0000-0002-8461-8996
ivan.r.opirskyi@lpnu.ua

Дзьобан Тарас Іванович
студент кафедри захисту інформації
Національний Університет «Львівська Політехніка», Львів, Україна
ORCID ID: 0009-0003-4446-6240
taras.dzoban.kb.2023@lpnu.ua

Василишин Святослав Ігорович
PhD, доцент кафедри захисту інформації
Національний Університет «Львівська Політехніка», Львів, Україна
ORCID ID: 0000-0003-1944-2979
sviatoslav.i.vasylyshyn@lpnu.ua

ОБХІД EDR У ПОЄДНАННІ З SIEM: АНАЛІЗ МЕТОДІВ ПРИХОВУВАННЯ АТАК У ЛОГАХ — ДОСЛІДЖЕННЯ ТАКТИК, ЩО ВИКОРИСТОВУЮТЬ ЗЛОВМИСНИКИ ДЛЯ УНИКНЕННЯ ДЕТЕКЦІЇ

Анотація. У статті розглядається актуальна проблема — методи обходу систем виявлення та реагування на кінцевих точках (EDR) у поєднанні з платформами управління інформацією та подіями безпеки (SIEM), які є основними компонентами сучасної кіберзахисної інфраструктури. Попри постійне вдосконалення цих технологій, зловмисники також розвивають тактики, що дозволяють уникати виявлення та зберігати присутність у скомпрометованих системах. У статті здійснено класифікацію методів уникнення виявлення, зокрема таких як маніпуляція логами, підробка подій, вимкнення сервісів журналювання та використання малочастотних атак, які не потрапляють у пороги спрацювання систем попередження. Особлива увага приділена методам, які базуються на концепції «Living-off-the-Land» — використання вбудованих інструментів операційної системи (PowerShell, WMIC, CertUtil) для виконання шкідливого коду з мінімальними індикаторами активності. Також проаналізовано техніки обфускації, включаючи інжекцію зайвого коду, шифрування, перекомпіляцію та застосування кастомних завантажувачів, які дозволяють уникати виявлення як сигнатурними, так і евристичними аналізаторами. У роботі детально описано й методи атаки на рівні ядра операційної системи, зокрема DKOM (пряма маніпуляція об’єктами ядра), unhooking бібліотек DLL та атаки на рівні UEFI/BIOS, які дозволяють обійти системи моніторингу завдяки роботі поза межами контрольованого середовища ОС. Окремо розглянуто методи уникнення контролю з боку SIEM-систем: очищення логів, підробка часових міток, перевантаження сенсорів та генерація надмірної кількості подій, що ускладнює роботу аналітиків та знижує ефективність реагування. У роботі наведено приклади із застосуванням популярних рішень, таких як Elastic, Splunk, CrowdStrike та SentinelOne. У підсумку автори роблять висновок про необхідність впровадження поведінкового аналізу, довготривалої кореляції, міжплатформенної телеметрії та моделей машинного навчання як ключових засобів протидії складним технікам уникнення та забезпечення видимості кіберзагроз у сучасному гібридному IT-середовищі.

Ключові слова: Endpoint Detection and Response (EDR); Security Information and Event Management (SIEM); ухилення від виявлення; маніпуляції з журналами; повільні атаки; перевантаження подіями; Living-off-the-Land (LotL); обфускація коду; unhooking; ядрові атаки; DKOM; BIOS/UEFI; поведінкова аналітика; міжплатформна телеметрія; кореляція; кіберзагрози; PowerShell; WMIC; CertUtil.

ВСТУП

У сучасному цифровому світі, де підприємства, державні установи та критична інфраструктура дедалі більше покладаються на інформаційні технології, кібербезпека перетворилася з технічного аспекту на стратегічний пріоритет. Збільшення складності інформаційних систем та розширення цифрової взаємодії зумовлюють нові виклики у сфері захисту даних. Кожен компонент цифрового середовища — від кінцевого користувача до хмарних платформ — стає потенційною мішенню для атак. Ситуацію ускладнює і зростання обсягу даних, які циркулюють через відкриті мережі, що робить їх привабливими для зловмисників, які прагнуть отримати несанкціонований доступ або паралізувати критичні сервіси.

Останні кібератаки, зокрема з використанням програм-шифрувальників, зловмисних завантажувачів та цілеспрямованих атак на постачальницькі ланцюги (supply chain attacks), демонструють, що навіть один успішний інцидент може спричинити значні фінансові втрати, збої в роботі організацій, втрату довіри користувачів і репутаційні ризики. Крім того, дедалі частіше спостерігаються атаки, що поєднують кілька векторів впливу, таких як соціальна інженерія, експлуатація вразливостей нульового дня та використання автоматизованих інструментів для швидкого розповсюдження в мережах.

Сучасні зловмисники активно користуються широким спектром публічно доступних та комерційно доступних інструментів для автоматизації атак, які здатні обходити традиційні засоби захисту. Як відповідь на ці загрози, системи Endpoint Detection and Response (EDR) та Security Information and Event Management (SIEM) стали невід'ємними складовими архітектури захисту. Вони забезпечують постійний моніторинг кінцевих точок, агрегацію подій із різних джерел, виявлення аномальної поведінки та швидке реагування на інциденти. Установлення EDR і SIEM на робочих станціях, серверах і в хмарному середовищі дає змогу створити останню лінію оборони, яка може виявити загрози, що обійшли периметрову або антивірусну оборону.

Попри значний прогрес у розвитку цих технологій, останні наукові дослідження (Carvalho H. (2024); Defeating EDR-Evading Malware with Memory Forensics (2024); HookChain: A New Perspective for Bypassing EDR Solutions (2024); MITRE ATT&CK Evaluations) вказують на суттєві обмеження виявлення при зіткненні з тактиками ухилення (evasion techniques). Зокрема, аналіз свідчить про зростаючу популярність обфускації, унікалізації шкідливого коду, використання легітимних утиліт (LotL), фальсифікації логів, модифікацій API на рівні ядра, а також атак через BIOS/UEFI, що залишаються невидимими для традиційних сенсорів. У такому середовищі здатність до глибокого аналізу поведінки, оперативного реагування та міжсистемної кореляції подій стає критично важливою умовою забезпечення кіберстійкості.

Аналіз останніх досліджень і публікацій. Праця [14] «HookChain: A New Perspective for Bypassing EDR Solutions» пропонує нову методику обходу EDR шляхом маніпуляцій із ланцюгами хуків (hook chains). Цей підхід дозволяє змінювати логіку перехоплення системних викликів без виявлення з боку EDR. Аналогічно, у [15] Case демонструє ефективність використання волатильного аналізу пам'яті (memory forensics) для виявлення шкідливих компонентів, що залишаються невидимими для традиційного моніторингу.

Джерела [4] та [21] детально описують обфускаційні методики, включно з junk-кодом, розбиттям функцій (function splitting) та шифруванням рядків. Ці прийоми дозволяють суттєво знизити ймовірність виявлення зловмисного ПЗ, особливо у випадку використання сигнатурного аналізу.У дослідженнях [12], [13] підкреслюється використання вбудованих Windows-утиліт, таких як certutil або wmic, для здійснення шкідливих дій без завантаження

додаткового ПЗ. Такий підхід дозволяє залишатися непоміченим у системах, що базуються на поведінковому аналізі, адже утиліти є частиною стандартного середовища.

У статтях [1] та [10] розглядаються техніки «відключення» (unhooking) бібліотек Windows API, які найчастіше використовуються EDR-агентами для моніторингу. Ці техніки, зокрема DLL unhooking, дозволяють зловмисному коду виконуватися без перехоплення з боку EDR. Роботи [2], [3] та [9] описують приклади використання UEFI та BIOS як векторів атаки. Зокрема, відомий кейс з Lojax став першим підтвердженим руткітом у прошивці UEFI у реальному середовищі. Вони демонструють можливість збереження стійкого доступу навіть після перевстановлення ОС, що виходить за межі традиційних можливостей EDR. У [16] – [18] та [20] досліджується ефективність та еволюція систем EDR і XDR. Авторам вдається показати обмеження сучасних рішень, пов'язані з обсягом телеметрії, складністю кореляції даних і значним навантаженням на аналітиків SOC. Особливо актуальною є проблема вигорання працівників SOC, розглянута в [11], що опосередковано впливає на якість реагування на ухилення. Матеріали [5] – [8] ілюструють, як інструменти автоматизації й обробки подій (наприклад, Splunk, Elastic Cloud, Coccinelle) можуть як сприяти виявленню загроз, так і слугувати основою для обходу захисту — зокрема, якщо логіка аналізу не адаптується до нових векторів атак.

Проблематика дослідження: існуючі рішення EDR і SIEM не завжди здатні ефективно ідентифікувати високорозвинені, приховані або повільно активні атаки, які застосовують комбіновані методи ухилення. Це створює критичні вразливості в захисних системах, особливо в умовах швидкої еволюції загроз і обмежених можливостей людського аналізу великого обсягу подій без контексту.

Мета роботи: дослідити методи і тактики, які використовують зловмисники для обходу систем Endpoint Detection and Response та Security Information and Event Management, з акцентом на аналіз найбільш дієвих тактик ухилення від EDR станом на момент публікації статі.

Для вирішення поставленої мети необхідно вирішити наступні завдання:

1. Провести порівняльний аналіз між антивірусами, EDR та XDR
2. Здійснити дослідження методів обфускації
3. Здійснити дослідження методів засліплення EDR
4. Провести загальний огляд на методи уникнення детекції SIEM
5. Оцінити, які саме техніки ухилення найменш виявляються сучасними засобами безпеки.
6. Визначити доцільність комбінованого застосування EDR та SIEM у захисті інфраструктури.

ОСНОВНА ЧАСТИНА

EDR — це система безпеки, яка фокусується на кінцевих пристроях, таких як комп'ютери, сервери чи мобільні пристрої. Її основна функція — виявлення, розслідування та реагування на загрози на цих пристроях. EDR збирає дані про активність, аналізує їх для виявлення підозрілих дій, наприклад, аномальної поведінки, і дозволяє ізолювати пристрій чи видаляти шкідливий код. Агенти на кінцевих пристроях відстежують процеси, мережеві з'єднання та зміни у файлах. Дані аналізуються за допомогою машинного навчання та сигнатур. У разі виявлення загрози аналітики отримують сповіщення і можуть ізолювати пристрій або провести розслідування.XDR — це розширена система виявлення та реагування, яка інтегрує дані з кінцевих пристроїв,

мереж, хмарних середовищ та інших компонентів IT-інфраструктури. Вона поєднує можливості EDR, SIEM та інших інструментів, забезпечуючи комплексний підхід до виявлення та реагування на загрози. XDR збирає інформацію з усієї IT-інфраструктури, аналізує її за допомогою штучного інтелекту та кореляції, автоматично ізолює загрози, блокує атаки та надає рекомендації аналітикам.

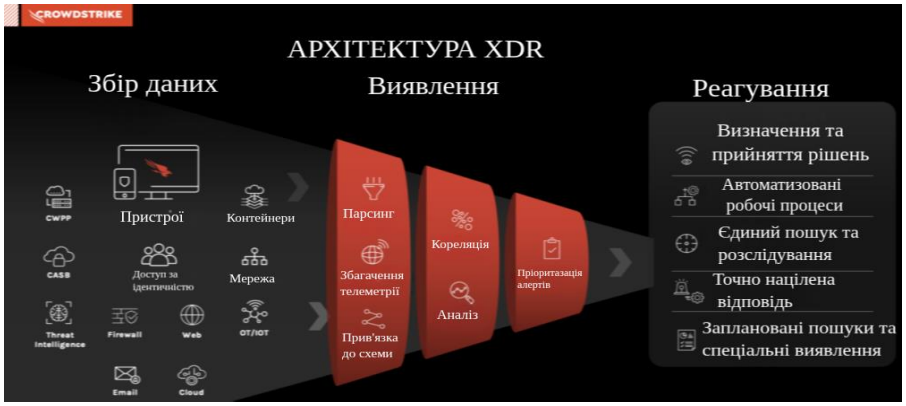

Рис. 1. Принцип роботи CrowdStrike Falcon XDR

SIEM — це система управління інформацією та подіями безпеки, яка об'єднує дані з різних джерел, таких як мережі, сервери чи додатки. Вона призначена для моніторингу, аналізу та реагування на інциденти безпеки шляхом кореляції подій і створення сповіщень на основі встановлених правил. SIEM агрегує логи та події з різних систем, аналізує їх для виявлення патернів, що вказують на загрози, і генерує сповіщення та звіти для аналітиків.

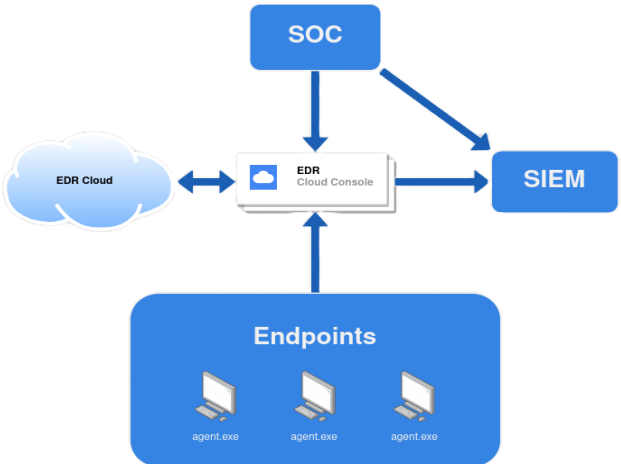

Рис. 2. Схема зв'язку між EDR, SIEM та SOC

Дослідження та порівняльний аналіз антивірусів, EDR та XDR

Перш за все, перед тим як говорити про особливості та ефективності використання EDR+SIEM, потрібно розуміти чим EDR відрізняється від того ж звичного для всіх антивіруса(AV) та Extended Detection and Response(XDR). Основна мета AV - виявлення та блокування відомих загроз, тоді як мета EDR — виявлення, розслідування і реагування на загрози на кінцевих пристроях, щодо XDR — мова йде про розширене виявлення та реагування на загрози в усій IT-інфраструктурі організації. Простішими словами XDR поєднує в собі все, на що здатні AV і EDR та дозволяє аналітику побачити повну картину. Детальніше див. табл. 1.

Таблиця 1

Порівняння AV, EDR та XDR

Характеристика	Антивірус (AV)	EDR (Endpoint Detection and Response)	XDR (Extended Detection and Response)
Джерела даних	Кінцеві пристрої	Кінцеві пристрої	Кінцеві пристрої, мережа, хмара, email, сервери
Метод виявлення	Сигнатури, евристика	Сигнатури, евристика, поведінковий аналіз, телеметрія, машинне навчання	Кореляція подій з кількох джерел, аналітика
Можливість реагування	Обмежене (переважно карантин файлів)	Розширене (ізоляція пристрою, завершення процесів тощо)	Централізоване реагування на загрози у всьому середовищі
Контекст та аналітика	Мінімальні	Глибокий аналіз подій, ланцюг атак	Повна картина атаки, зв'язки між подіями в різних системах
Підтримка аналітика SOC	Обмежена	Так	Так (з глибшим охопленням і автоматизацією)
Придатний для великих середовищ	Обмежено	Так	Так (особливо для розподілених середовищ)
Приклад рішень	Avast, ESET, Kaspersky	CrowdStrike, SentinelOne, Microsoft Defender for Endpoint	Palo Alto Cortex XDR, Trend Micro Vision One, Microsoft Defender XDR

Обфускація

Обфускація (від англ. obfuscation — затемнення, заплутування) — це техніка, яка використовується для ускладнення розуміння, аналізу або reverse engineering даних чи комунікацій, зберігаючи при цьому їхню функціональність.

Головний спосіб уникнення засобів захисту кінцевих пристроїв полягає в тому, щоб ваш код і бінарні файли не були одразу ідентифіковані як зловмисне програмне забезпечення. Оскільки сучасні інструменти EDR, навіть із традиційним статичним аналізом, також застосовують евристичну перевірку, важливо, щоб фрагменти коду у файлах і бінарних даних не розпізнавалися, інакше вони можуть бути класифіковані як відомий шкідливий код.

Це можуть здійснювати наступними методами:

1. Використання шифрування (див. рис. 3) base64, AES, XOR, hex або ж власні кастомні скрипти і розшифрування безпосередньо в пам'яті. Поширеними прикладами є шифрування коду, який працюватиме доти, доки його не розшифрують під час виконання(див. рис. 4), або скремблювання команд, що виконуються, у довільному порядку, а під час виконання — перекомпонування їх у правильні набори команд. Як і у випадку з перекомпіляцією, ця техніка найбільш корисна для обходу статичного аналізу. Це ускладнює статичний аналіз,оскільки EDR не бачить оригінального коду на диску.

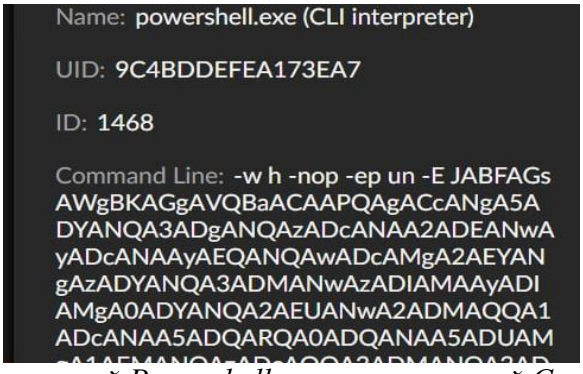

Рис. 3. Обфускований Powershell скрипт виявлений Crowdstrike EDR

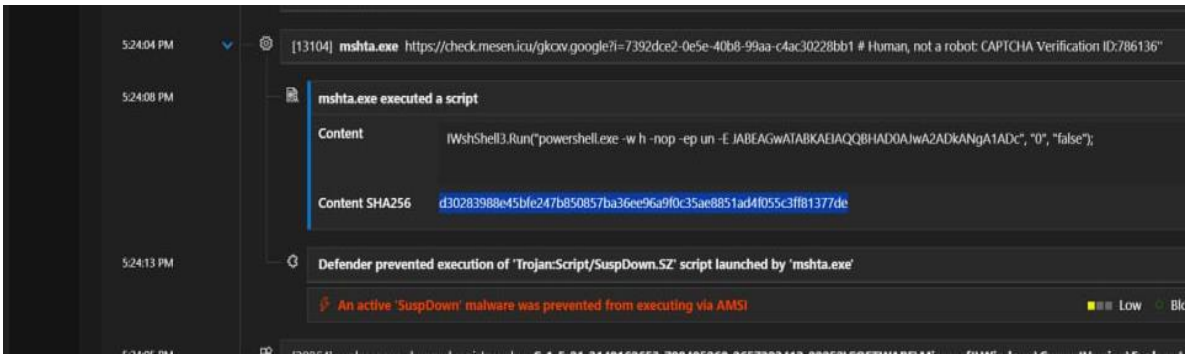

Рис. 4. Обфускований командлайн виявлений Microsoft Defender

2. Додавання junk code. Посеред програми можна вставляти багато «сміттєвого коду», які будуть виглядати як певні інструкції тощо, вони не будуть впливати на безпосередню логіки самої програми, але аналіз дуже ускладнять(див. рис. 5). Це можуть бути математичні операції або якісь випадкові цикли. Поширеними прикладами є вставка коментарів (які насправді нічого не виконують).Прикладом може також слугувати вставка тисяч рядків NOP (No Operation) у код.

3. Перекомпіляція файлу шкідливого програмного забезпечення, написаного на C++ на C#,як файла компонента у двійковому коді. Мета полягає в тому, щоб створити новий файл, який при проходженні через операцію математичного хешування видає зовсім інший хеш, ніж оригінальний файл. Також можна використовувати інструменти такі як Coccinelle (інструмент для трансформації вихідного коду на мові C, який використовує семантичні патчі для заміни конструкцій коду еквівалентними). Цей метод дозволяє обійти статичний аналіз, оскільки після запуску файлу на виконання він все одно виконує зловмисні дії, які виявить динамічний аналіз.

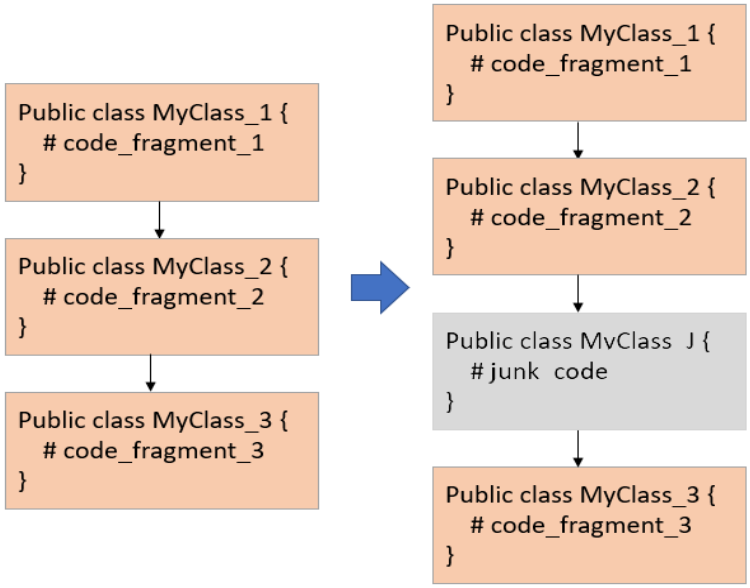

Рис. 5. Схема Додавання junk code (ResearchGate)

```
./scripts/coccicheck --mode=report --cocci=scripts/coccinelle/array_size.cocci
```

Рис. 6. Приклад використання Coccinelle

ISSN 2663 - 4023

Запустивши цю команду (див рис. 6) *Coccinelle* виконає наступну частину скрипту SmPL (див. рис. 7):

```
<smpl>

@r depends on (org || report)@
type T;
T[] E;
position p;
@@
(
(sizeof(E)@p /sizeof(*E))
|
(sizeof(E)@p /sizeof(E[...]))
|
(sizeof(E)@p /sizeof(T))
)

@script:python depends on report@
p << r.p;
@@

msg="WARNING: Use ARRAY_SIZE"
coccilib.report.print_report(p[0], msg)

</smpl>
```

Рис. 7. Частина SmPL скрипта використана Coccinelle

Цей фрагмент SmPL генерує записи на стандартному виводі, як показано нижче на рис. 8:

```
ext/hal/nxp/mcux/drivers/lpc/fsl_wwdt.c:66:49-50: WARNING: Use ARRAY_SIZE
ext/hal/nxp/mcux/drivers/lpc/fsl_ctimer.c:74:53-54: WARNING: Use ARRAY_SIZE
ext/hal/nxp/mcux/drivers/imx/fsl_dcp.c:944:45-46: WARNING: Use ARRAY_SIZE
```

Рис. 8. Записи standard output

Засліплення EDR

Якщо EDR можна «засліпити» одним або декількома способами, то шкідливий файл матиме змогу виконуватися без перешкод. Слід зазначити, що сучасні рішення EDR докладають значних зусиль, щоб уникнути цього типу обходу і більшість з них використовують кілька методів візуалізації виконання, це зроблено спеціально для того, щоб зменшити ймовірність успішного обходу. Обхід може бути здійснений за допомогою широкого спектру методів, але, як і у випадку з обфускацією, вони, як правило, відносяться до однієї з декількох категорій:

1. Unhooked processes: під час роботи звичайного користувача, де зазвичай виконуються програми та відбувається взаємодія з користувачем — будь-який процес, який не виключено з перевірки EDR, буде зафіксований або ж зачеплено(hooked). Під фіксацією мається на увазі, що EDR відслідковує виконання, моніторячи процеси, які їх ініціюють, тим самим візуалізуючи активність у системі. Ця техніка передбачає видалення або нейтралізацію хуків (hooks), встановлених EDR у системних бібліотеках (наприклад, ntdll.dll), щоб запобігти моніторингу шкідливої активності. Наприклад, у Windows, коли програма запускається, вона виконується як екземпляр процесу ntdll. Перехоплюючи кожен екземпляр ntdll, який створюється, EDR може відстежувати всі запуски, щоб визначити, чи вказують вони на зловмисні наміри. Реальний процес значно складніший, але саме так EDR може відстежувати всі виконання на рівні користувача на високому рівні. Якщо суб’єкт загрози може викликати екземпляр ntdll, який не перехоплюється EDR, або якщо він може

ISSN 2663 - 4023

видалити перехоплення після того, як екземпляр створено, то дії, виконані цим шкідливим програмним забезпеченням, стануть невидимими для самого EDR. Існує багато різних методів, які використовуються для створення відчеплених процесів або для відчеплення процесів. Перший метод — завантаження чистої копії DLL із диска. Сам процес виглядає приблизно ось так: Завантажується чиста копія бібліотеки (наприклад, ntdll.dll) із файлової системи (наприклад, C:\Windows\System32\ntdll.dll) — Знаходиться секція .text (виконуваний код) у захуканій бібліотеці в пам'яті процесу-Секція .text із чистої копії копіюється поверх захуканої версії за допомогою функцій, таких як WriteProcessMemory-Відновлюються оригінальні права доступу до пам'яті за допомогою VirtualProtect.

main.cpp

```cpp
#include <Windows.h>
#include <winternl.h>
#include <psapi.h>

int main() {
    HANDLE process = GetCurrentProcess();
    MODULEINFO mi = {};
    HMODULE ntdllModule = GetModuleHandleA("ntdll.dll");
    GetModuleInformation(process, ntdllModule, &mi, sizeof(mi));
    LPVOID ntdllBase = (LPVOID)mi.lpBaseOfDll;

    HANDLE ntdllFile = CreateFileA("C:\\Windows\\System32\\ntdll.dll", GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL);
    HANDLE ntdllMapping = CreateFileMapping(ntdllFile, NULL, PAGE_READONLY | SEC_IMAGE, 0, 0, NULL);
    LPVOID ntdllMappingAddress = MapViewOfFile(ntdllMapping, FILE_MAP_READ, 0, 0, 0);

    PIMAGE_DOS_HEADER hookedDosHeader = (PIMAGE_DOS_HEADER)ntdllBase;
    PIMAGE_NT_HEADERS hookedNtHeader = (PIMAGE_NT_HEADERS)((DWORD_PTR)ntdllBase + hookedDosHeader->e_lfanew);

    for (WORD i = 0; i < hookedNtHeader->FileHeader.NumberOfSections; i++) {
        PIMAGE_SECTION_HEADER hookedSectionHeader = (PIMAGE_SECTION_HEADER)((DWORD_PTR)IMAGE_FIRST_SECTION(hookedNtHeader) + ((DWORD_PTR)IMAGE_SIZEOF_SECTION_HEADER * i));
        if (!strcmp((char*)hookedSectionHeader->Name, ".text")) {
            DWORD oldProtection = 0;
            VirtualProtect((LPVOID)((DWORD_PTR)ntdllBase + hookedSectionHeader->VirtualAddress), hookedSectionHeader->Misc.VirtualSize, PAGE_EXECUTE_READWRITE, &oldProtection);
            memcpy((LPVOID)((DWORD_PTR)ntdllBase + hookedSectionHeader->VirtualAddress),
                   (LPVOID)((DWORD_PTR)ntdllMappingAddress + hookedSectionHeader->VirtualAddress),
                   hookedSectionHeader->Misc.VirtualSize);
            VirtualProtect((LPVOID)((DWORD_PTR)ntdllBase + hookedSectionHeader->VirtualAddress), hookedSectionHeader->Misc.VirtualSize, oldProtection, &oldProtection);
        }
    }

    CloseHandle(process);
    CloseHandle(ntdllFile);
    CloseHandle(ntdllMapping);
    FreeLibrary(ntdllModule);
    return 0;
```

Рис. 9. Приклад коду програми на C++ для створення unhooked process

Однак тут, як і всюди є ризик — читання ntdll.dll із диска може бути виявлено EDR через моніторинг файлових операцій. Також потребує відповідності версії DLL із системою.

Ще одним методом може бути використання чистої DLL із віддаленого процесу. Процес виглядає приблизно наступним чином: Створюється новий процес у призупиненому стані (наприклад notepad.exe), який ще не був захуканий EDR — Витягується чиста копія бібліотеки (наприклад ntdll.dll) із пам'яті цього процесу — Копіюється секція .text у пам'ять цільового процесу. У цього методу є перевага — він уникає читання файлів із диска, що знижує ймовірність виявлення. Хоча сам процес важчий в реалізації, шанси бути непоміченим зростуть в рази. Використання власної копії DLL — також є ще одним методом. Хоча має два основних недоліки: завантаження файлів із мережі може бути виявлено мережевими сканерами та цей метод отребує точної відповідності версії DLL. Процес виглядає ось так: Зловмисник хостить чисту копію DLL на віддаленому сервері, яка відповідає версії ОС цільової системи — DLL завантажується в пам'ять і використовується для заміни захуканої версії.

Whisper2Shout — інша техніка unhooking, яка має одну чудову особливість — вона не потребує доступу до чистої копії DLL. Вона аналізує приватні регіони пам'яті, створені EDR для хуків, і відновлює оригінальні інструкції функцій.

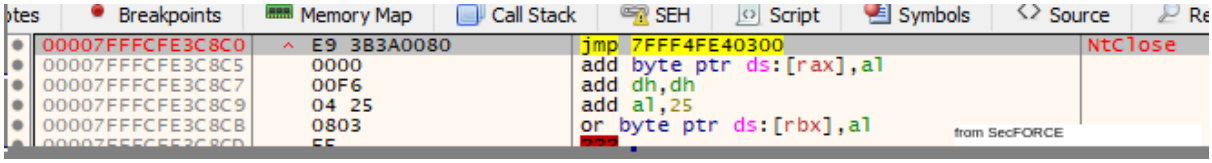

Рис. 10. Ntclose syscall «захукана» Bitdefender(SecForce)

ISSN 2663 - 4023

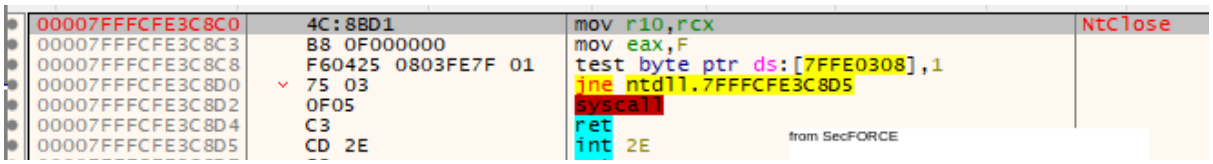

Рис. 11. Той ж syscall після unhooking'а (SecForce)

Після виконання Whisper2Shout усі цільові функції в бібліотеці (наприклад, ntdll.dll) повертаються до свого оригінального стану, і EDR втрачає можливість моніторити їх виклики. Процес стає unhooked, дозволяючи виконувати системні виклики без перехоплення.

2. Операції на рівні ядра

На додаток до операцій на рівні користувача, з якими більшість з нас знайомі з Windows, macOS та Linux, існує ще один рівень операцій, який за задумом є невидимим для користувача. Цей вищий рівень операцій називається рівнем ядра і зарезервований для використання ключових компонентів операційної системи та обмеженої кількості надзвичайно пріоритетних операцій. Операції на рівні ядра мають прямий доступ до апаратного забезпечення, пам'яті та системних ресурсів, що робить їх надзвичайно потужними, але водночас і потенційно небезпечними у разі зловмисного використання.

Оскільки операції на рівні ядра можуть завантажуватися до запуску самої платформи EDR, постачальники операційних систем докладають надзвичайних зусиль, щоб гарантувати, що на рівні ядра можуть виконуватися лише операції з високим ступенем довіри, здебільшого пов'язані з функціями самої операційної системи. Наприклад, у Windows використовується механізм цифрового підпису драйверів (Driver Signature Enforcement), а в macOS — System Integrity Protection (SIP), які обмежують можливість завантаження непідписаного або несанкціонованого коду в ядро. Проте, використовуючи вразливості або компрометуючи процес завантаження комп'ютера, зловмисники можуть запустити шкідливе програмне забезпечення, яке працює на рівні ядра, роблячи його невидимим для більшості EDR-інструментів. Такі атаки дозволяють шкідливому ПЗ приховувати свою присутність, маніпулювати системними процесами або навіть відключати захисні механізми.

Одним із найвідоміших прикладів атак на рівні ядра є зараження BIOS або UEFI, яке відбувається до завантаження операційної системи. Такі атаки дозволяють шкідливому коду працювати на найнижчому рівні системи, що робить його невидимим для EDR. Наприклад, у 2018 році було виявлено шкідливе ПЗ LoJax, яке використовувало вразливості в UEFI для створення персистентного руткіта (Дослідження ESET // Вересень 2018). LoJax модифікував прошивку материнської плати, що дозволяло йому перезавантажуватися навіть після повного переустановлення операційної системи. EDR-системи не могли виявити LoJax, оскільки він працював на рівні апаратного забезпечення, а не в межах операційної системи. Для здійснення такої атаки зловмисники зазвичай потребують фізичного доступу до пристрою або використання вразливостей віддаленого виконання коду в прошивці. Це робить подібні атаки складними, але надзвичайно ефективними проти стандартних захисних механізмів. Цікавим є те що LoJax це переписаний LoJack, який в свою чергу був створений для захисту комп'ютерів від крадіжки та втрати, розроблений корпорацією Absolute Software. Ранні версії агента були відомі під назвою Computrace. Як випливає з колишньої назви, після активації сервісу комп'ютер може звертатися до свого командного сервера — власника сповістять про місцезнаходження пристрою в разі зникнення або крадіжки.

ISSN 2663 - 4023

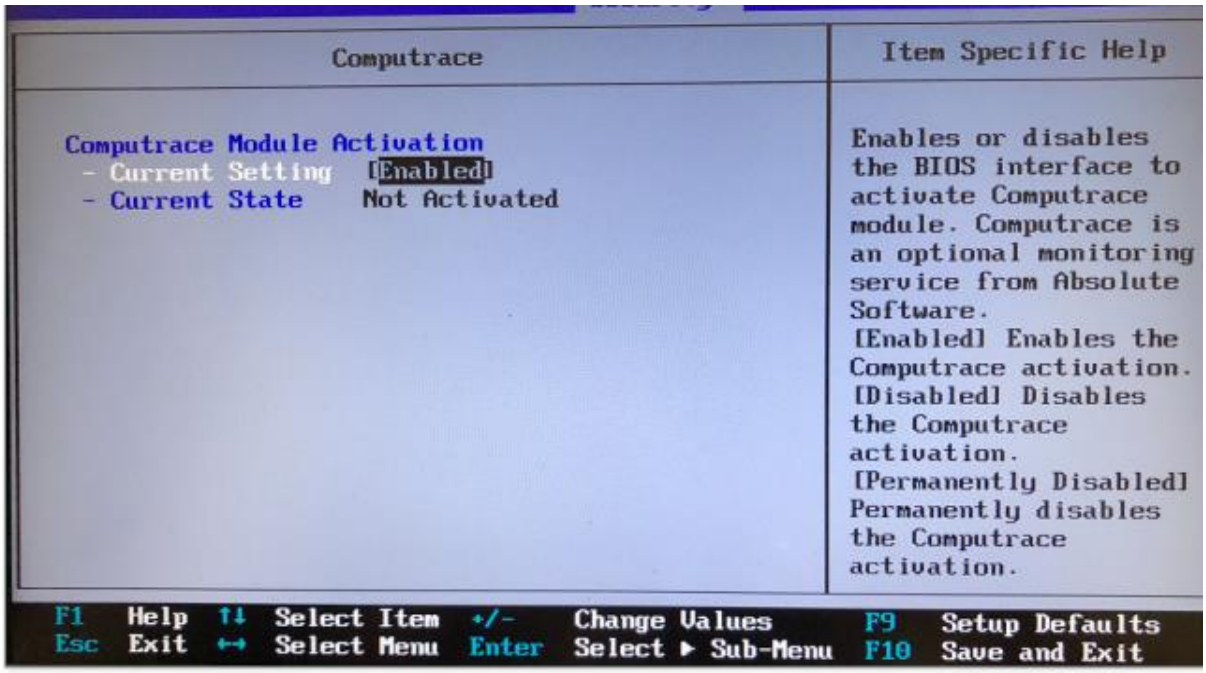

Рис. 12. Активація Computrace в BIOS(ESET)

У 2009 році було розкрито архітектуру цього продукту, процес скидання модулем UEFI/BIOS призначеного для користувача агента на диск і спосіб зв'язку агента з веб-сервером, підконтрольним Absolute Software. Схему можна зрозуміти з рисунка нижче.

Рис. 13. Механізм персистентності LoJack 2008р (ESET)

У травні 2018 року в блозі Arbor Network були описані троянізовані зразки міні-агента LoJack, rpcnetp.exe. Їхні жорстко прописані мережеві налаштування були змінені так, що шкідливі зразки встановлювали зв'язок із C2-сервером атакувальників замість легітимного сервера Absolute Software(див. рис. 14).

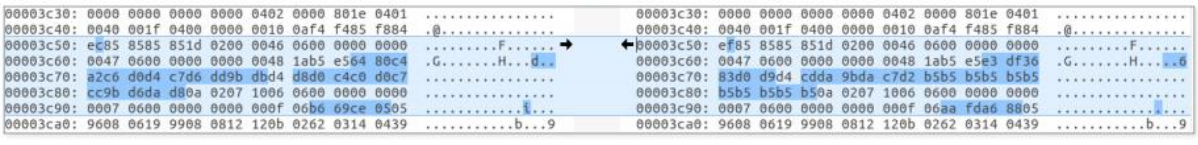

Рис. 14. Програма LoJack до зміни — зліва, після зміни LoJax 2018р — справа(ESET)

ISSN 2663 - 4023

Ще одним прикладом може слугувати техніка Direct Kernel Object Manipulation (DKOM) дозволяє зловмисникам модифікувати структури даних у ядрі, такі як список активних процесів (EPROCESS у Windows), щоб приховати шкідливі процеси. Наприклад, руткіт Turla, який використовувався в цілеспрямованих атаках, видаляв свій процес зі списку PsActiveProcessHead, що робило його невидимим для EDR-систем і стандартних утиліт, таких як Task Manager. Turla також використовував техніку перехоплення системних викликів для фільтрації запитів, які могли б виявити його присутність. Для захисту від таких атак EDR-системи можуть моніторити цілісність структур ядра або використовувати крос-перевірку даних із кількох джерел, наприклад — порівняння списку процесів із мережевою активністю. Руткіт Turla був розроблений росіянами для викрадення документів, електронних листів, облікових даних (паролів, ключів шифрування) і іншої чутливої інформації. Наприклад, у кампаніях проти дипломатичних установ у Європі та на Близькому Сході (2008–2014) руткіт використовувався для доступу до внутрішніх комунікацій і секретних звітів.

```
00000008       C (16 bits) - UTF-16LE mfa
0000000A       C (16 bits) - UTF-16LE .gov
0000000C       C (16 bits) - UTF-16LE .gouv
0000000E       C (16 bits) - UTF-16LE .ocse.
0000000A       C (16 bits) - UTF-16LE .mil
```

Рис. 15. Домени, пов'язані з держслужбами, знайдені в коді малварі

Бекдор являє собою автономну DLL, в якій є код для самовстановлення і взаємодії з поштовими клієнтами Outlook і The Bat!, навіть якщо реалізовано тільки встановлення для Outlook. Його з легкістю може скинути будь-який компонент Turla, що дозволяє виконувати додаткові процеси.

```
0000003C43 54 44 65 63 6F 64 65 2F 42 69 74 73 50 65 72 43 6F 6D 70 CTDecode/BitsPerComp
000000506F 6E 65 6E 74 20 38 2F 43 6F 6C 6F 72 53 70 61 63 65 2F 44 onent 8/ColorSpace/D
0000006465 76 69 63 65 52 47 42 2F 57 69 64 74 68 20 31 2F 48 65 69 eviceRGB/Width 1/Hei
0000007867 68 74 20 31 2F 4C 65 6E 67 74 68 20 31 39 36 39 3E 3E 0A ght 1/Length 1969>>.
0000008C73 74 72 65 61 6D 0A FF D8 FF E0 00 10 4A 46 49 46 00 01 01 stream.......JFIF...
000000A001 00 60 00 60 00 00 FF DB 00 43 00 02 01 01 02 01 01 02 02 ..`.`.....C.........
000000B402 02 02 02 02 02 03 05 03 03 03 03 03 06 04 04 03 05 07 06 ....................
000000C807 07 07 06 07 07 08 09 0B 09 08 08 0A 08 07 07 0A 0D 0A 0A ....................
000000DC0B 0C 0C 0C 0C 07 09 0E 0F 0D 0C 0E 0B 0C 0C 0C FF DB 00 43 ...................C
000000F001 02 02 02 03 03 03 06 03 03 06 0C 08 07 08 0C 0C 0C 0C 0C ....................
000001040C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C ....................
000001180C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C ....................
0000012C0C 0C 0C 0C 0C FF C0 00 11 08 00 01 00 01 03 01 22 00 02 11 ..............."...
0000014001 03 11 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 00 ....................
0000015400 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4 00 ....................
00000168B5 10 00 02 01 03 03 02 04 03 05 05 04 04 00 00 01 7D 01 02 ................}..
0000017C03 00 04 11 05 12 21 31 41 06 13 51 61 07 22 71 14 32 81 91 ......!1A..Qa."q.2..
00000190A1 08 23 42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 18 ..#B...R..$3br......
000001A419 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 ..%&'()*456789:CDEFG
000001B848 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 HIJSTUVWXYZcdefghijs
```

Рис.16. Початок PDF-документа, створеного бекдором для ексфільтрації даних Turla

Дослідження методів уникнення детекції в SIEM.

Одним із найпростіших і найефективніших способів обходу SIEM є втручання в процеси генерації, передачі або зберігання журналів. Зловмисники можуть зупиняти служби журналювання, такі як реєстрація подій Windows або syslog у Linux, щоб запобігти створенню записів. Шкідливі скрипти часто видаляють журнали безпеки чи програм одразу після виконання підозрілих дій, тому таку активність добре-адміністровані SIEM будуть детектити (див. рис. 17).

ISSN 2663 - 4023

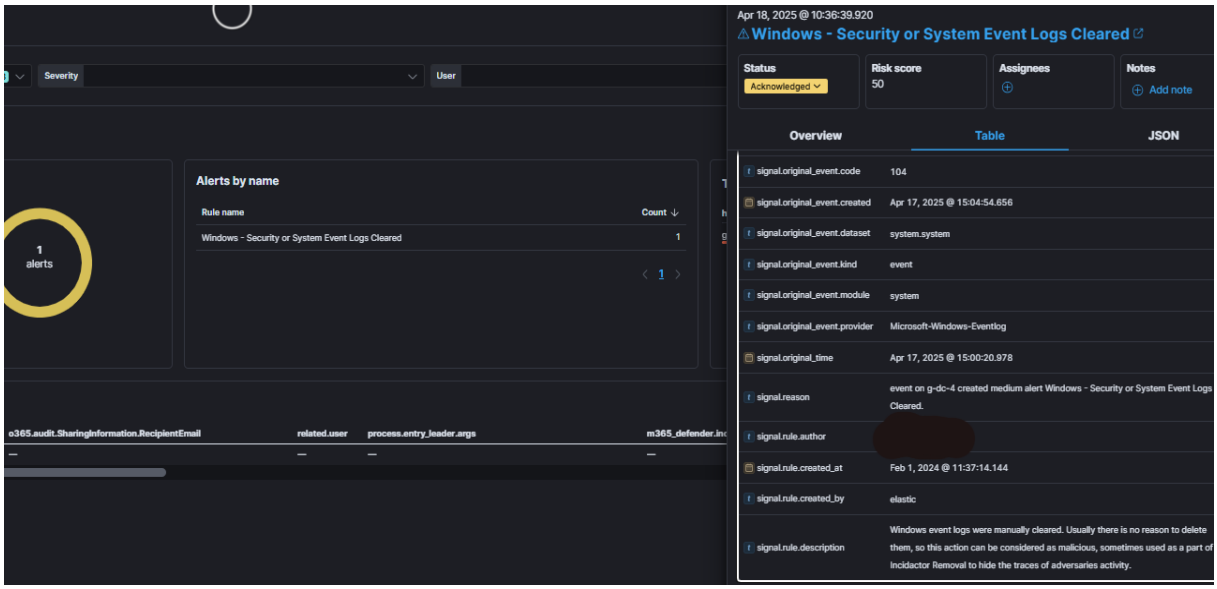

Рис. 17. Спрацювання детекції на подібну активність у Elastic Coud SIEM

Сучасне шкідливе програмне забезпечення здатне змінювати мітки часу або видаляти записи для приховування несанкціонованого доступу чи підвищення привілеїв. Якщо агенти кінцевих точок, такі як OSQuery, NXLog або Wazuh, передають журнали до SIEM, зловмисники можуть відключити або припинити їх роботу, щоб зупинити збір даних. Ці методи особливо ефективні в системах, де не забезпечується цілісність журналів, наприклад, через перевірку хешів чи доступ лише для пересилання.Хоча створення правил Healthcheck для перевірки надходження журналів за певний період є хорошою практикою, вони часто спрацьовують через певний час (5–24год) див. рис. 18 і навіть цього може бути достатньо атакеру для того,щоб досягти своєї цілі.

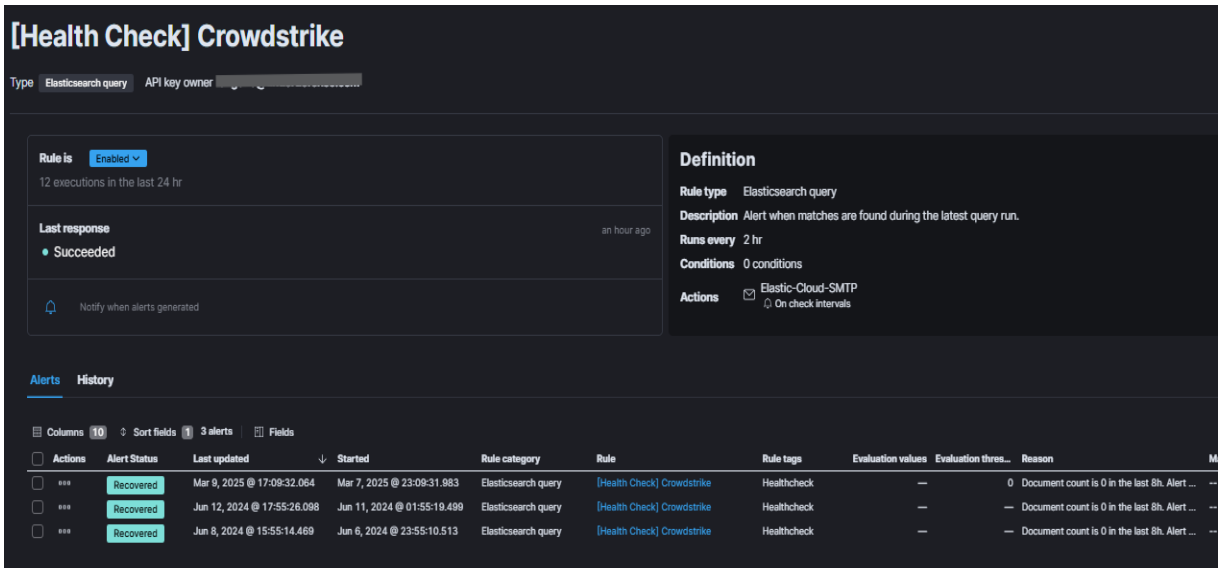

Рис.18. Healthcheck на логсорс Crowdstrike заведений в Elastic Cloud SIEM з історичними спрацюваннями

Щоб уникнути спрацьовування порогових правил виявлення, зловмисники можуть навмисно сповільнювати свої дії, залишаючись непоміченими. Наприклад, при повільному брутфорсі вони вводять один пароль кожні кілька хвилин або годин, уникаючи блокування облікового запису чи активації правил виявлення грубого перебору.

Бічний рух може відбуватися протягом кількох днів: замість швидкого переходу між хостами, зловмисники компрометують системи поступово. Викрадення даних також здійснюється поетапно — дані розбиваються на невеликі пакети, що передаються годинами або днями, щоб не викликати сповіщень про витік. SIEM-системи, які погано налаштовані або використовують короткострокові вікна кореляції або статичні пороги, часто не здатні виявити ці тонкі патерни без застосування розширеної аналітики чи довгострокового базового рівня(див. рис. 19).

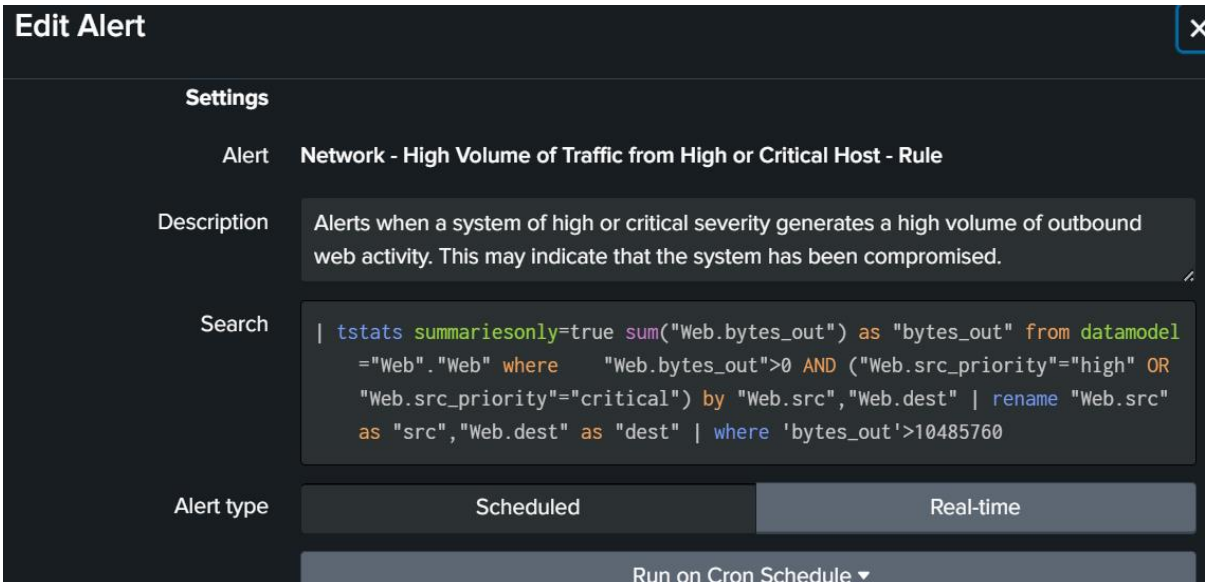

Рис. 19. Пошуковий запит алерту у Splunk,
який визначає появу великого трафіку в мережі

Методи Living-off-the-Land (LotL) дозволяють зловмисникам уникати сповіщень, використовуючи вбудовані інструменти операційної системи замість зовнішніх двійкових файлів чи програм. Наприклад, PowerShell часто застосовується для завантаження корисного навантаження, виконання скриптів у пам’яті або розвідки без запису на диск. WMIC (Windows Management Instrumentation Command-line) використовується для отримання системної інформації чи запуску віддалених процесів. CertUtil (див рис. 20) може завантажувати файли з інтернету або кодувати/декодувати рядки base64. Bitsadmin і BITS Transfer дозволяють приховано завантажувати та виконувати файли, що часто застосовується у шкідливих кампаніях. Такі методи LotL використовують довіру до нативних інструментів, генеруючи журнали, які виглядають як звичайна адміністративна діяльність, особливо в організаціях без детальних правил моніторингу. Методи LotL використовують довіру до нативних інструментів і створюють журнали, які часто виглядають як рутинна адміністративна діяльність, особливо в організаціях, що не мають детальних правил виявлення. Зазвичай зараз правила на виявлення подібної активності присутні у кожній SIEM.

ISSN 2663 - 4023

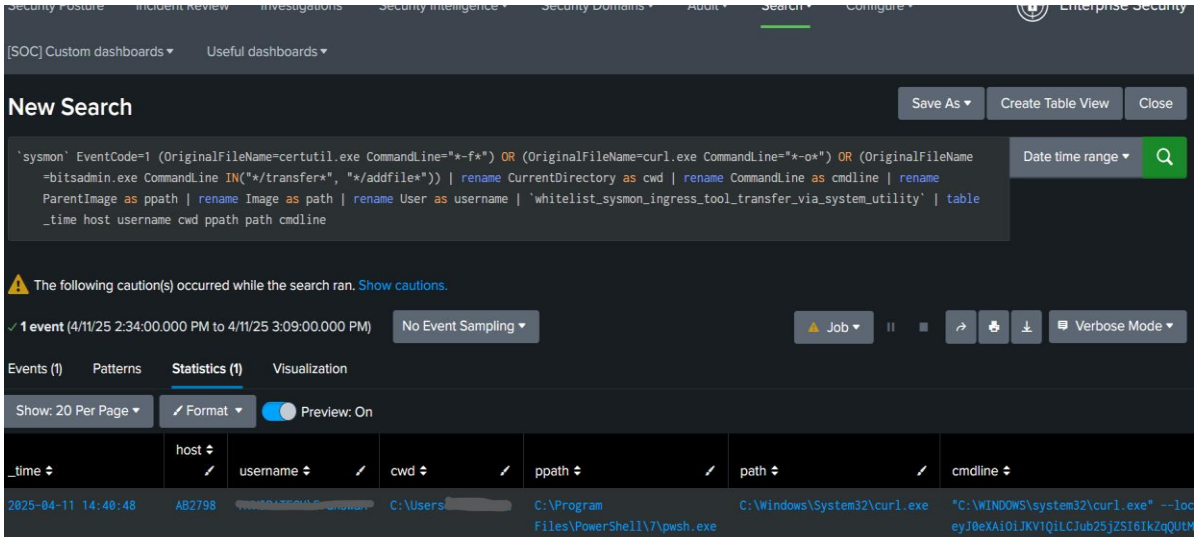

Рис. 20. Спрацювання правила Ingress_Tool_Transfer_Utility в Splunk.

Перевантаження подіями, або «затоплення журналів», — це методика, спрямована на перевантаження SIEM-систем великою кількістю доброякісних чи фальшивих подій. Генеруючи численні події, такі як невдалі входи чи нешкідливі сканування (див.рис. 21), зловмисники викликають втому від сповіщень, відволікаючи аналітиків і приховуючи справжні загрози серед шуму. Крім того, змушення SIEM оцінювати надмірну кількість правил кореляції призводить до виснаження правил, що погіршує продуктивність системи та може спричинити пропуск виявлень.

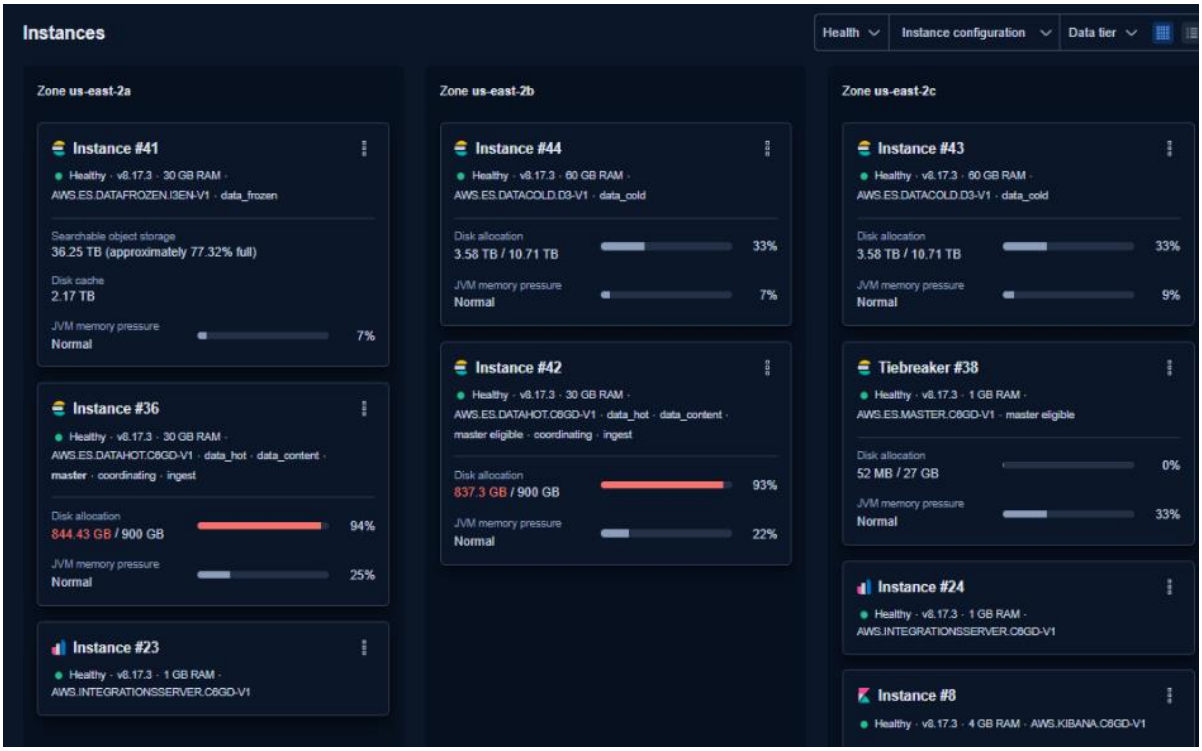

Рис. 21. Hot-node в Elastic Cloud SIEM забилася від надмірної генерації логів, в результаті деплоймент помер = атакер мав вікно для атаки

ISSN 2663 - 4023

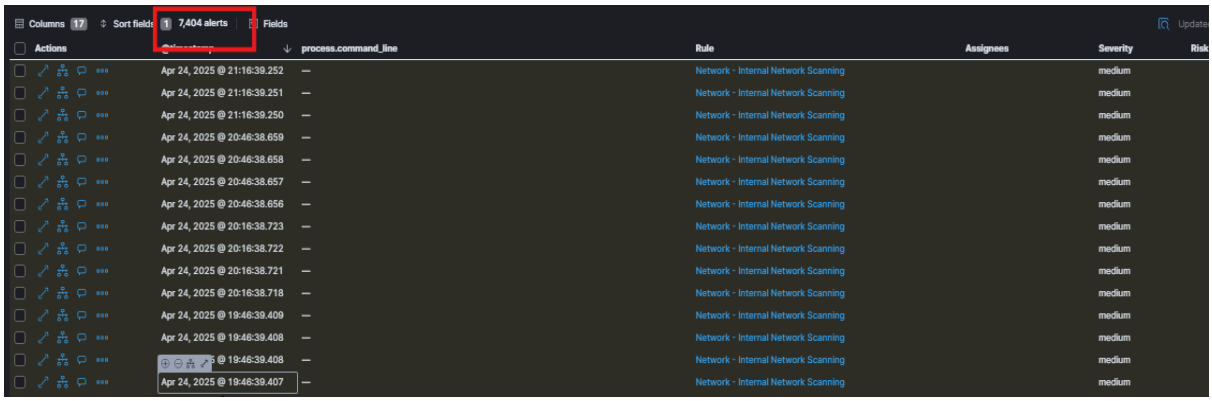

Рис. 22. Більше 7600 алертів згенеровано за два дні правилом Internal Network Scanning

Перевантаження датчиків, таких як системи IDS/IPS, також створює проблеми: при високому навантаженні вони втрачають пакети або знижують видимість, надаючи зловмисникам вікно для дій.Ця тактика особливо ефективна проти недостатньо налаштованих SIEM або перевантажених команд SOC з малим складом і може бути частиною ширшої стратегії, відомої як «вигорання SOC» див. рис. 23.

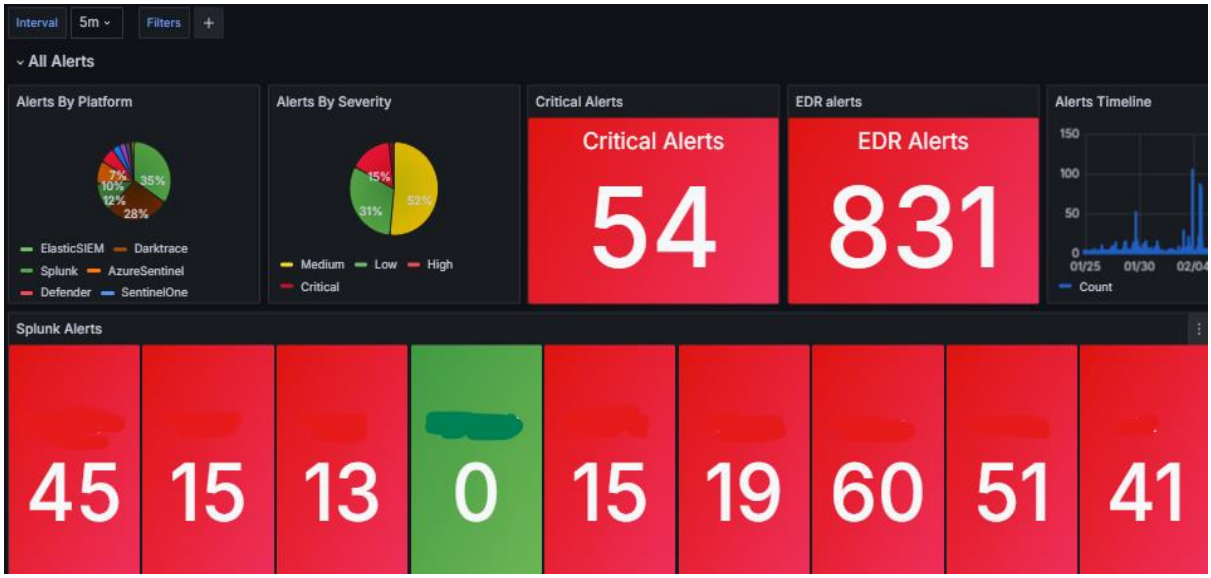

Рис. 23. Вигорання Security Operations Center очима SOC-аналітика

Таким чином, сучасні кіберзагрози вимагають від організацій не лише впровадження передових технологій, таких як EDR, XDR та SIEM, а й комплексного підходу до кібербезпеки. Це включає оптимізацію конфігурацій систем, посилення команд SOC шляхом автоматизації рутинних завдань, використання SOAR систем, а також інвестування в навчання персоналу та впровадження технологій штучного інтелекту для раннього виявлення аномалій. Лише за умови інтеграції цих заходів організації зможуть ефективно протистояти складним і багатогранним тактикам ухилення, що постійно еволюціонують у кіберпросторі.

ВИСНОВОКИ

У результаті проведеного дослідження встановлено, що зловмисники використовують широкий спектр дієвих тактик для обходу систем Endpoint Detection and Response (EDR) та Security Information and Event Management (SIEM), зокрема обфускацію коду, маніпуляції з журналами, методи Living-off-the-Land, повільні атаки, перевантаження подіями, операції на рівні ядра та засліплення EDR. Ці техніки дозволяють ефективно приховувати шкідливу активність, ускладнюючи виявлення навіть найсучаснішими захисними рішеннями. Порівняльний аналіз показав, що XDR кращий за антивіруси та EDR завдяки інтеграції даних із різних джерел, але й ці системи не є повністю стійкими до витончених атак. Традиційні методи виявлення, що базуються на сигнатурах і статичних правилах, мають значні обмеження, тому окремого використання EDR, XDR чи SIEM недостатньо — їхня інтеграція є критично важливою. Для ефективної протидії кіберзагрозам необхідний проактивний багаторівневий підхід, який включає сильну видимість кінцевих точок, збагачені дані журналів, регулярне оновлення сигнатур, поведінкові моделі, методи обману та розслідування, керовані людьми. Розуміння методів ухилення від виявлення є ключовим для розробки стійких стратегій кібербезпеки в умовах зростання складності загроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Whisper2Shout – Unhooking Technique.* (n.d.). https://www.secforce.com/blog/whisper2shout-unhooking-technique/
2. *First UEFI Rootkit Detected in the Wild.* (n.d.). https://eset.ua/download_files/marketing/Releases/lojax_whitepaper.pdf
3. *Turla Rootkits.* (n.d.). https://uk.wikipedia.org/wiki/Turla_
4. *Junk Code Insertion.* (n.d.). https://www.researchgate.net/figure/Function-Splitting_fig3_371581054
5. *Coccinelle Basic Documentation.* (n.d.). https://docs.zephyrproject.org/latest/develop/tools/coccinelle.html
6. *Coccinelle Basic Documentation.* (n.d.). https://docs.zephyrproject.org/latest/develop/tools/coccinelle.html
7. *Splunk Documentation.* (n.d.). https://docs.splunk.com/Documentation/Splunk/9.4.1/Overview/AboutSplunkEnterprise
8. *Elastic Cloud Documentation.* (n.d.). https://www.elastic.co/docs/deploy-manage/deploy/elastic-cloud/cloud-hosted
9. *Computrace BIOS Trojan.* (n.d.). https://novikov.ua/bios-%D0%BD%D1%8B%D0%B9-%D1%82%D1%80%D0%BE%D1%8F%D0%BD-%D0%BE%D1%82-absolute-software-computrace/
10. *DLL Unhooking.* (n.d.). https://unprotect.it/technique/dll-unhooking/
11. *SOC Burnout.* (n.d.). https://medium.com/infosec-ninja/sos-for-your-soc-how-to-prevent-burnout-and-boost-retention-7c053b5b71ce
12. *SOC Burnout.* (n.d.). https://medium.com/infosec-ninja/sos-for-your-soc-how-to-prevent-burnout-and-boost-retention-7c053b5b71ce
13. *Exploring the Hidden Switches of Certutil and Certreq.* (n.d.). https://www.encryptionconsulting.com/exploring-the-hidden-switches-of-certutil-and-certreq/
14. *WMIC Guide.* (n.d.). https://learn.microsoft.com/ru-ru/windows/win32/wmisdk/wmic
15. *HookChain: A New Perspective for Bypassing EDR Solutions.* (n.d.). https://arxiv.org/abs/2404.16856
16. *Defeating EDR-Evading Malware with Memory Forensics.* (n.d.). https://www.volexity.com/wp-content/uploads/2024/08/Defcon24_EDR_Evasion_Detection_White-Paper_Andrew-Case.pdf
17. *AV Bypass Techniques through an EDR Lens.* (n.d.). https://blog.f-secure.com/av-bypass-techniques-through-an-edr-lens/
18. *Evolution of Endpoint Detection and Response (EDR) in Cyber Security: A Comprehensive Review.* (n.d.). https://www.e3s-conferences.org/articles/e3sconf/abs/2024/86/e3sconf_rawmu2024_01006/
19. *Effectiveness of Endpoint Detection and Response Solutions in Combating Modern Cyber Threats.* (n.d.). https://polarpublications.com/index.php/JACSTIC/article/view/1
20. *Bypassing Antivirus Detection: Old-School Malware, New Tricks.* (n.d.). https://arxiv.org/abs/2305.04149

21. *XDR: The Evolution of Endpoint Security Solutions – Superior Extensibility and Analytics to Satisfy the Organizational Needs of the Future.* (n.d.). https://www.researchgate.net/publication/354190628_XDR_The_Evolution_of_Endpoint_Security_Solutions_-Superior_Extensibility_and_Analytics_to_Satisfy_the_Organizational_Needs_of_the_Future
22. *A Taxonomy of Software Obfuscation Techniques for Layered Security*. (n.d.). https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00049-3

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023

Ivan Opirskyy
Doctor of Technical Sciences, Professor,
Head of the Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID ID: 0000-0002-8461-8996
ivan.r.opirskyi@lpnu.ua

Taras Dzoban
Student of the Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID ID: 0009-0003-4446-6240
taras.dzoban.kb.2023@lpnu.ua

Sviatoslav Vasylyshyn
PhD., Associate Professor, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID ID: 0000-0003-1944-2979
sviatoslav.i.vasylyshyn@lpnu.ua

BYPASSING EDR COMBINED WITH SIEM: ANALYSIS OF ATTACK CONCEALMENT TECHNIQUES IN LOGS — A STUDY OF ADVERSARIAL TACTICS FOR DETECTION EVASION

Abstract. This article addresses a highly relevant cybersecurity issue — methods for bypassing Endpoint Detection and Response (EDR) systems in combination with Security Information and Event Management (SIEM) platforms, which are key components of modern cyber defense infrastructure. Despite the continuous evolution of these technologies, attackers develop tactics to evade detection and maintain persistence in compromised systems. The paper presents a classification of evasion techniques, including log manipulation, event spoofing, disabling logging services, and low-frequency attacks that remain below alert thresholds. Special attention is given to tactics based on the "Living-off-the-Land" (LotL) concept — leveraging built-in operating system tools (e.g., PowerShell, WMIC, CertUtil) to execute malicious code with minimal indicators of compromise. Obfuscation techniques such as junk code injection, encryption, recompilation, and the use of custom loaders are analyzed for their ability to evade both signature-based and heuristic detection engines. The paper also explores kernel-level attack methods, including Direct Kernel Object Manipulation (DKOM), DLL unhooking, and firmware-level intrusions via UEFI/BIOS modifications, which allow attackers to operate outside the monitored OS environment. Furthermore, the study examines SIEM evasion methods such as log wiping, timestamp tampering, sensor overload, and alert flooding — all of which aim to degrade analyst effectiveness and reduce detection fidelity. Real-world examples are provided using popular platforms such as Elastic, Splunk, CrowdStrike, and SentinelOne. The authors conclude by emphasizing the importance of behavioral analytics, long-term correlation, cross-platform telemetry, and machine learning models as essential strategies for countering sophisticated evasion techniques and ensuring threat visibility in hybrid IT environments.

Keywords: Endpoint Detection and Response (EDR); Security Information and Event Management (SIEM); detection evasion; log manipulation; slow attacks; event flooding; Living-off-the-Land (LotL); code obfuscation; unhooking; kernel-level attacks; DKOM; BIOS/UEFI; behavioral analytics; cross-platform telemetry; correlation; cyber threats; PowerShell; WMIC; CertUtil.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. *Whisper2Shout – Unhooking Technique.* (n.d.). https://www.secforce.com/blog/whisper2shout-unhooking-technique/
2. *First UEFI Rootkit Detected in the Wild.* (n.d.). https://eset.ua/download_files/marketing/Releases/lojax_whitepaper.pdf

3. *Turla Rootkits.* (n.d.). https://uk.wikipedia.org/wiki/Turla_
4. *Junk Code Insertion.* (n.d.). https://www.researchgate.net/figure/Function-Splitting_fig3_371581054
5. *Coccinelle Basic Documentation.* (n.d.). https://docs.zephyrproject.org/latest/develop/tools/coccinelle.html
6. *Coccinelle Basic Documentation.* (n.d.). https://docs.zephyrproject.org/latest/develop/tools/coccinelle.html
7. *Splunk Documentation.* (n.d.). https://docs.splunk.com/Documentation/Splunk/9.4.1/Overview/AboutSplunkEnterprise
8. *Elastic Cloud Documentation.* (n.d.). https://www.elastic.co/docs/deploy-manage/deploy/elastic-cloud/cloud-hosted
9. *Computrace BIOS Trojan.* (n.d.). https://novikov.ua/bios-%D0%BD%D1%8B%D0%B9-%D1%82%D1%80%D0%BE%D1%8F%D0%BD-%D0%BE%D1%82-absolute-software-computrace/
10. *DLL Unhooking.* (n.d.). https://unprotect.it/technique/dll-unhooking/
11. *SOC Burnout.* (n.d.). https://medium.com/infosec-ninja/sos-for-your-soc-how-to-prevent-burnout-and-boost-retention-7c053b5b71ce
12. *SOC Burnout.* (n.d.). https://medium.com/infosec-ninja/sos-for-your-soc-how-to-prevent-burnout-and-boost-retention-7c053b5b71ce
13. *Exploring the Hidden Switches of Certutil and Certreq.* (n.d.). https://www.encryptionconsulting.com/exploring-the-hidden-switches-of-certutil-and-certreq/
14. *WMIC Guide.* (n.d.). https://learn.microsoft.com/ru-ru/windows/win32/wmisdk/wmic
15. *HookChain: A New Perspective for Bypassing EDR Solutions.* (n.d.). https://arxiv.org/abs/2404.16856
16. *Defeating EDR-Evading Malware with Memory Forensics.* (n.d.). https://www.volexity.com/wp-content/uploads/2024/08/Defcon24_EDR_Evasion_Detection_White-Paper_Andrew-Case.pdf
17. *AV Bypass Techniques through an EDR Lens.* (n.d.). https://blog.f-secure.com/av-bypass-techniques-through-an-edr-lens/
18. *Evolution of Endpoint Detection and Response (EDR) in Cyber Security: A Comprehensive Review.* (n.d.). https://www.e3s-conferences.org/articles/e3sconf/abs/2024/86/e3sconf_rawmu2024_01006/
19. *Effectiveness of Endpoint Detection and Response Solutions in Combating Modern Cyber Threats.* (n.d.). https://polarpublications.com/index.php/JACSTIC/article/view/1
20. *Bypassing Antivirus Detection: Old-School Malware, New Tricks.* (n.d.). https://arxiv.org/abs/2305.04149
21. *XDR: The Evolution of Endpoint Security Solutions – Superior Extensibility and Analytics to Satisfy the Organizational Needs of the Future.* (n.d.). https://www.researchgate.net/publication/354190628_XDR_The_Evolution_of_Endpoint_Security_Solutions_-Superior_Extensibility_and_Analytics_to_Satisfy_the_Organizational_Needs_of_the_Future
22. *A Taxonomy of Software Obfuscation Techniques for Layered Security.* (n.d.). https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00049-3