

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки**

«На правах рукопису»
УДК 004.056.53:004.451.9

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри
_____ Дмитро ЛАНДЕ
«__» _____ 20__ р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи, технології та
математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»**

на тему:

"Методи виявлення та протидії підвищенню привілеїв в операційній системі Windows
для запобігання прихованим загрозам"

Виконав (-ла): здобувач (-ка) вищої освіти **IV** курсу, групи _____ **ФБ-23**
(шифр групи)

_____ **Ващасв Тимофій Вікторович** _____
(прізвище, ім'я, по батькові) (підпис)

Керівник

доцент кафедри ІБ, к.т.н., доцент, Гальчинський Леонід Юрійович _____
(посада, науковий ступінь, вчене звання, ім'я, прізвище) (підпис)

Рецензент

_____ (посада, науковий ступінь, вчене звання, науковий ступінь, ім'я, прізвище) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2026 року
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
 Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 125 «Кібербезпека»

Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Дмитро ЛАНДЕ
 (підпис)

«_____» _____ 20__ р.
 (дата захисту 2026 р.)

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти
 Ващаєву Тимофію Вікторовичу
 (прізвище, ім'я, по батькові)

1. Тема роботи _"Методи виявлення та протидії підвищенню привілеїв в операційній системі Windows для запобігання прихованим загрозам"_____

керівник роботи _____ Гальчинський Леонід Юрійович, к.т.н., доцент _____,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «29» травня 2026 р. №2001-с

2. Термін подання роботи здобувачем вищої освіти «10» червня 2026 р.

3. Вихідні дані до роботи: архітектура безпеки операційної системи Windows та механізми управління привілеями; офіційна документація Microsoft щодо Windows API, маркерів доступу та дескрипторів безпеки; база знань MITRE ATT&CK; телеметрія підсистеми Sysmon та журнали подій безпеки Windows; принципи побудови та функціонування SIEM-систем

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): провести аналіз моделей безпеки та механізмів контролю доступу в операційній системі Windows; проаналізувати програмні механізми управління привілеями на рівні Windows API з метою виявлення архітектурних недоліків та потенційних векторів ескалації привілеїв; розробити методи виявлення підвищення привілеїв на основі аналізу журналів подій безпеки Windows із застосуванням правил кореляції в SIEM-системі.

5. Перелік графічного (ілюстративного) матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): презентація

6. Дата видачі завдання "1" жовтня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання, визначення об'єкта та предмета дослідження	01.10.2025	Виконано
2	Огляд науково-технічної літератури за темою дипломної роботи	13.12.2025	Виконано
3	Дослідження моделі безпеки операційної системи Windows	20.02.2026	Виконано
4	Аналіз методів підвищення привілеїв та програмних механізмів Windows API	05.04.2026	Виконано
6	Розгортання тестового середовища та конфігурація засобів моніторингу	19.04.2026	Виконано
5	Проходження переддипломної практики	20.04.2026 - 17.05.2026	Виконано
7	Практичне моделювання та дослідження техніки підміни динамічних бібліотек	22.05.2026	Виконано
8	Практичне моделювання та дослідження техніки крадіжки маркера безпек	27.05.2026	Виконано
9	Розробка методів виявлення ескалації привілеїв та формування кореляційних правил для SIEM-систем	01.06.2026	Виконано
10	Передзахист дипломної роботи	10.06.2026	Виконано
11	Доопрацювання дипломної роботи		
12	Захист дипломної роботи		

Здобувач вищої освіти

(підпис)

Тимофій ВАЩАЄВ

(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Леонід Гальчинський

(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Робота обсягом 88 сторінок містить 31 рисунок, 7 таблиць, 4 додатки та 31 джерело літератури.

Тема: методи виявлення та протидії підвищенню привілеїв в операційній системі Windows для запобігання прихованим загрозам.

Мета роботи: підвищення ефективності захисту інформаційних систем від прихованих загроз шляхом систематизації методів ескалації привілеїв в операційній системі Windows, розробки методів їх виявлення за допомогою аналізу подій безпеки в SIEM-системах, а також формування рекомендацій щодо протидії несанкціонованому підвищенню привілеїв.

Об'єкт дослідження: процеси авторизації, управління привілеями та контролю доступу в архітектурі безпеки операційної системи Windows.

Предмет дослідження: методи та механізми ескалації привілеїв на рівні операційної системи Windows, а також підходи до їх виявлення та протидії за допомогою SIEM-систем на основі аналізу подій безпеки, що генеруються службою підсистеми локальних авторитетів безпеки, монітором довідок безпеки та механізмами маніпулювання токенами доступу.

Методи дослідження: аналіз літератури з безпеки операційних систем Windows, вивчення офіційної документації Microsoft та бази знань MITRE ATT&CK, систематична класифікація технік ескалації привілеїв та експериментальні методи практичного відтворення атак в ізольованому віртуальному середовищі.

Ключові слова: Windows API, підвищення привілеїв, список контролю доступу, маркер доступу, SIEM.

ABSTRACT

The thesis contains 88 pages, 31 figures, 7 tables, 4 appendices and 31 sources of literature.

Topic: methods for detecting and counteracting privilege escalation in Windows operating system to prevent hidden threats.

Purpose of the work: increasing the effectiveness of protecting information systems from hidden threats by systematizing methods for privilege escalation in the Windows operating system, developing methods for their detection using analysis of security events in SIEM systems, and formulating recommendations for counteracting unauthorized privilege escalation.

Object of research: authorization processes, privilege management and access control in the security architecture of the Windows operating system.

Subject of research: methods and mechanisms for privilege escalation at the Windows OS level, as well as approaches to their detection and counteraction using SIEM systems based on the analysis of security events generated by the local security authority subsystem service, the security information monitor and access token manipulation mechanisms.

Research methods: analysis of the literature on the security of Windows operating systems, study of official Microsoft documentation and the MITRE ATT&CK knowledge base, systematic classification of privilege escalation techniques and experimental methods for practical reproduction of attacks in an isolated virtual environment.

Keywords: Windows API, privilege escalation, access control list, access token, SIEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
1 ТЕОРЕТИЧНІ ОСНОВИ АРХІТЕКТУРИ БЕЗПЕКИ ТА МЕХАНІЗМІВ АВТОРИЗАЦІЇ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS.....	11
1.1 Модель безпеки та контролю доступу в операційній системі Windows.....	11
1.2 Роль Local Security Authority Subsystem Service та Security Accounts Manager у забезпеченні безпеки системи.....	14
1.3 Функціонування Security Reference Monitor у забезпеченні контролю доступу.....	17
1.4 Ідентифікатори безпеки та їх роль у визначенні суб'єктів доступу..	19
1.5 Архітектурні особливості та функціонування маркерів доступу.....	22
1.6 Модель контролю доступу в операційній системі Windows.....	25
Висновки до розділу 1.....	28
2 МЕТОДИ ПІДВИЩЕННЯ ПРИВІЛЕЇВ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS ТА СПОСОБИ ЇХ ВИЯВЛЕННЯ.....	29
2.1 Класифікація та концепція методів підвищення привілеїв.....	29
2.2 Техніки підвищення привілеїв на рівні операційної системи Windows.....	34
2.3. Аналіз програмних механізмів управління привілеями на рівні Windows API.....	37
2.4 Архітектура та принципи функціонування SIEM-систем у сучасній інфраструктурі.....	43
Висновки до розділу 2.....	48
3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТЕХНІК ЕСКАЛАЦІЇ ПРИВІЛЕЇВ ТА МЕТОДІВ ЇХ ВИЯВЛЕННЯ.....	49
3.1 Постановка завдання та опис робочого середовища.....	49
3.2 Атака підміни динамічно пов'язуваних бібліотек.....	50
3.3 Атака крадіжки маркера безпеки.....	61
Висновки до розділу 3.....	71
ВИСНОВКИ.....	72
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SRM - Security reference monitor, монітор контролю безпеки

LSASS - Local Security Authority subsystem, підсистема служби локальної безпеки

SAM - Security Accounts Manager, диспетчер облікових записів безпеки

SID - Security Identifier, ідентифікатор безпеки

LSA - Local Security Authority, локальний центр безпеки

RID - Relative Identifier, відносний ідентифікатор

DAC - Discretionary Access Control, дискреційна модель контролю доступу

UAC - User Account Control, служба контролю облікових записів користувачів

ACL - Access Control List, список контролю доступу

ACE - Access Control Entries, записи контролю доступом

DACL - Discretionary Access Control List, дискреційний список керування доступом

SACL - System Access Control List, системний список керування доступом

SD - Security Descriptor, дескриптор безпеки

API - Application Programming Interface, інтерфейс програмування застосунків

SIEM - Security Information and Event Management, управління інформацією та подіями безпеки

DLL - Dynamic-link library, динамічно пов'язувана бібліотека

ВСТУП

Механізми, що регулюють доступ до ресурсів та управління привілеями користувачів, є невід'ємною складовою безпеки сучасних інформаційних систем. Оскільки Windows є домінуючою платформою в корпоративній та урядовій інфраструктурі, вона реалізує складну та багаторівневу архітектуру авторизації та контролю доступу. Розуміння внутрішньої логіки цих механізмів дозволяє розробити та впровадити ефективне рішення захисту від загроз, що використовують слабкі місця в управлінні привілеями.

Актуальність роботи визначається зростанням кількості кібератак, де ескалація привілеїв є ключовим етапом успішного вторгнення в системи. Водночас виявлення технік підвищення привілеїв залишається нетривіальним завданням, оскільки їх виконання часто відбувається в межах легітимних системних механізмів, що ускладнює виокремлення шкідливої активності від нормальної поведінки операційної системи. Відсутність систематизованих підходів до кореляції подій безпеки, що генеруються компонентами авторизації Windows у системах SIEM, знижує ефективність виявлення загроз у режимі реального часу, що підкреслює практичну необхідність даного дослідження.

Мета роботи: підвищення ефективності захисту інформаційних систем від прихованих загроз шляхом систематизації методів ескалації привілеїв в операційній системі Windows, розробки методів їх виявлення за допомогою аналізу подій безпеки в SIEM-системах, а також формування рекомендацій щодо протидії несанкціонованому підвищенню привілеїв.

Завдання дослідження:

1. Проведення аналізу моделей безпеки та контролю доступу в операційній системі Windows.
2. Проведення аналізу програмних механізмів управління привілеями на рівні Windows API.

3. Розробка методів виявлення підвищення привілеїв на основі аналізу журналів подій безпеки Windows.

Об'єкт дослідження: процеси авторизації, управління привілеями та контролю доступу в архітектурі безпеки операційної системи Windows.

Предмет дослідження: методи та механізми ескалації привілеїв на рівні операційної системи Windows, а також підходи до їх виявлення та протидії за допомогою SIEM-систем на основі аналізу подій безпеки, що генеруються службою підсистеми локальних авторитетів безпеки, монітором довідок безпеки та механізмами маніпулювання токенами доступу.

Методи дослідження: аналіз літератури з безпеки операційних систем Windows, вивчення офіційної документації Microsoft та бази знань MITRE ATT&CK, систематична класифікація технік ескалації привілеїв та експериментальні методи практичного відтворення атак в ізольованому віртуальному середовищі.

Практична цінність одержаних результатів полягає в удосконаленні налаштувань кореляційних правил в системах SIEM, спрямованих на виявлення методів ескалації привілеїв в операційних системах Windows.

Апробація результатів роботи: Результати роботи були представлені на X Міжнародна науково-практична конференція «INTERNATIONAL EXPERIENCE IN SCIENTIFIC RESEARCH» (14-16.05.2026 року Чикаго, США).

1 ТЕОРЕТИЧНІ ОСНОВИ АРХІТЕКТУРИ БЕЗПЕКИ ТА МЕХАНІЗМІВ АВТОРИЗАЦІЇ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS

1.1 Модель безпеки та контролю доступу в операційній системі Windows.

Модель безпеки Windows визначає сукупність процесів у режимі користувача та режимі ядра, які забезпечують, контролюють та керують різними компонентами безпеки операційної системи, а також координують їх взаємодію [1]. Дана модель базується на принципі, де кожен компонент операційної системи Windows повинен забезпечувати безпеку об'єктів, за які він відповідає (рис 1.1).

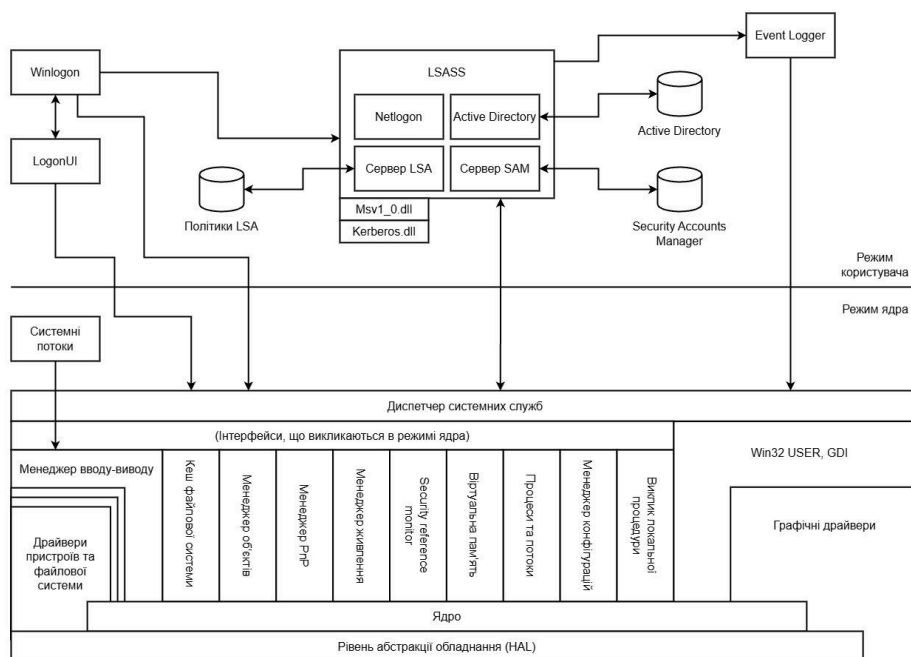


Рис. 1.1 Модель безпеки Windows

До основних компонентів безпеки, які забезпечують захист операційної системи Windows, відносять наступні [2]:

- **Security reference monitor (SRM)**, що відповідає за визначення структури даних маркера доступу для представлення контексту безпеки, виконання перевірок доступу до об'єктів та маніпулювання привілеями.

- Local Security Authority subsystem (LSASS), що відповідає за політику безпеки локальної системи та автентифікацію користувачів.
- Security Accounts Manager (SAM), що відповідає за безпеку та керування локальною базою даних облікових записів.
- Interactive logon manager (Winlogon), що відповідає за керування інтерактивними сеансами входу в систему.
- Logon user interface (LogonUI) надає зручний інтерфейс, який використовується для автентифікації в системі.
- Network logon service (Netlogon), що відповідає за перевірку користувачів та служб у будь-якій мережі, до якої користувач намагається отримати доступ.
- Authentication packages.
- Active Directory.

Процеси захисту в межах моделі безпеки характеризуються взаємодією двох основних категорій сутностей:

- Суб'єкти, що визначають активні сутності, які можуть ініціювати запити до ресурсів та виконувати певні дії в операційній системі.
- Об'єкти, що визначають пасивні сутності, які можуть бути використані суб'єктами системи.

Архітектура моделі безпеки Windows та її компоненти визначають загальний принцип, що дозволяє ізолювати процеси та виконувати перевірки автентичності суб'єктів операційної системи. Однак концепція розмежування прав доступу покладається на механізми та моделі керування доступом, які відіграють важливу роль у захисті компонентів програмного забезпечення, мережевих та системних ресурсів у корпоративному середовищі. Процес розмежування та контролю доступом становить комплексну систему, яка регулює взаємодію між суб'єктами та об'єктами на основі їхніх ролей та функціональних обов'язків.

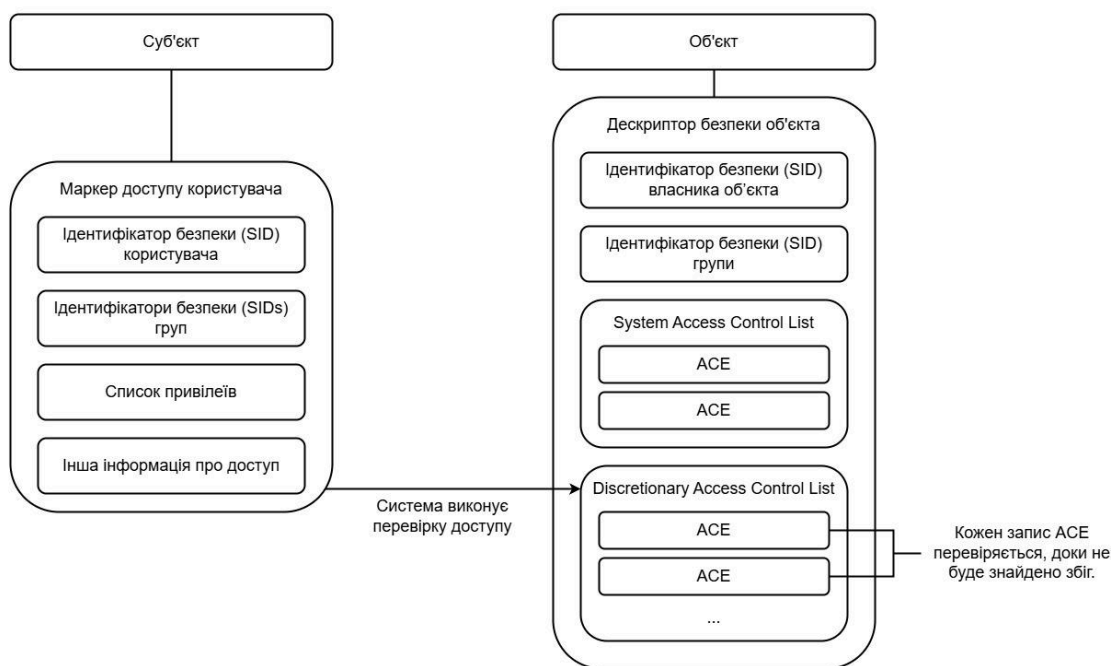


Рис. 1.2 Процес авторизації та контролю доступу

В операційній системі Windows ініціація процесу авторизації відбувається в момент, коли суб'єкт намагається отримати доступ до певного захищеного об'єкта. Контроль доступу (рис 1.2) дозволяє програмам вибірково надавати або забороняти суб'єктам доступ до певних служб, зменшуючи ризик несанкціонованого втручання в роботу системи. Підсистема безпеки порівнює інформацію в маркері доступу суб'єкта із записами контролю доступу (ACE) в дескрипторі безпеки цільового об'єкта, до якого необхідно отримати доступ [3]. На основі результатів покрокового порівняння операційна система приймає рішення щодо надання відповідного доступу або відхилення запиту суб'єкта.

Механізми безпеки та моделі контролю доступу складаються з наступних компонентів та технологій:

- Ідентифікатори безпеки (SID)
- Маркери доступу
- Дескриптори безпеки та списки контролю доступу
- Дозволи

1.2 Роль Local Security Authority Subsystem Service та Security Accounts Manager у забезпеченні безпеки системи

Забезпечення захисту в операційній системі Windows відбувається на етапі автентифікації суб'єкта та розмежування його прав за допомогою інтеграції двох основних елементів, які працюють в режимі користувача: Local Security Authority Subsystem Service (LSASS) та Security Accounts Manager (SAM). Ці компоненти утворюють єдиний комплекс захисту в архітектурі безпеки системи, де LSASS виконує функції центру керування безпекою, тоді як SAM забезпечує постійний та безпечний локальний простір для зберігання ідентифікаційної інформації суб'єкта.

Local Security Authority Subsystem Service (LSASS) – це критично важливий процес в операційних системах Windows, який виконується в режимі користувача та реалізує локальний центр безпеки (LSA), обробляючи локальні політики безпеки та автентифікацію користувачів шляхом перевірки облікових даних користувачів у Security Accounts Manager (SAM). Як основний компонент для перевірки доступу до системних об'єктів, LSASS зберігає облікові дані в пам'яті процесу для активних сеансів користувачів та створює маркери доступу, які встановлюють контекст безпеки, забезпечуючи контрольований доступ до ресурсів на основі застосованих політик [4]. Крім того, LSASS дозволяє записувати події аудиту до журналу безпеки Windows, забезпечуючи реєстрацію спроб автентифікації, порушень політик та інших дій, пов'язаних з безпекою системи. Local Security Authority Subsystem Service працює безперервно як критично важлива системна служба, що забезпечує дотримання локальних політик безпеки в операційних системах Windows, включаючи перевірку доступу користувачів та керування завданнями, пов'язаними з безпекою.

Local Security Authority (LSA) – це захищений системний процес, який автентифікує користувачів та зберігає інформацію про всі аспекти локальної безпеки на комп'ютері [5]. Підсистема безпеки LSA відстежує

політики безпеки та облікові записи, що знаходяться в системі, а також надає послуги для перевірки доступу до об'єктів та прав користувачів.

Local Security Authority застосовує всі параметри, налаштовані за допомогою локальної політики безпеки (secpol.msc), які включають вимоги до складності пароля, пороги блокування облікових записів та призначення прав користувачів. Вона постійно відстежує активність системи, перевіряє зміни паролів на відповідність вимогам складності, відстежує невдалі спроби входу, а також перевіряє права користувачів та блокує дії, що порушують налаштовані політики. Підсистема LSA також керує конфіденційною інформацією, що містить облікові дані системних облікових записів, паролі облікових записів служб та критично важливу для безпеки інформацію, яка використовується програмами та службами.

Security Accounts Manager (SAM) – це файл бази даних в операційній системі Windows, який зберігає облікові записи користувачів, зокрема їх імена та зашифровані хеші паролів. Налаштування SAM спрощують вхід користувачів до системи, а також роблять систему безпечною та надійною, захищаючи облікові дані у разі їх витоку.

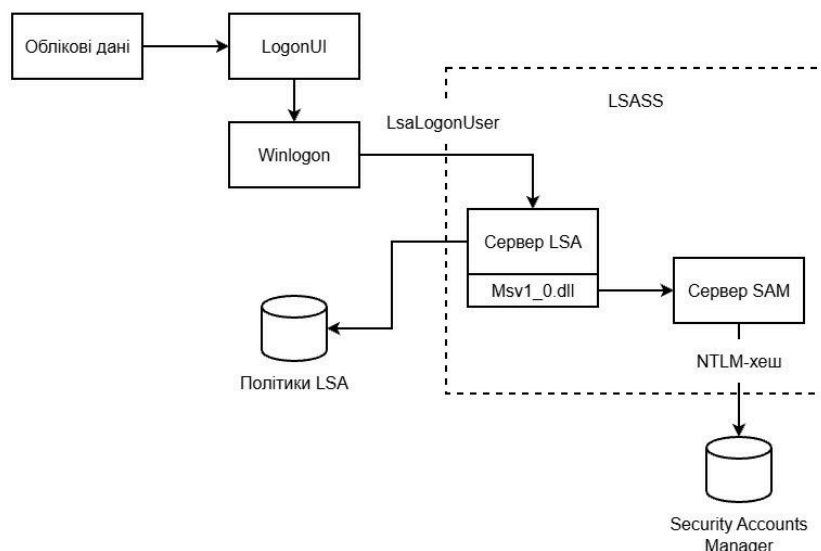


Рис. 1.3 Процес автентифікації в операційній системі Windows

Процес автентифікації у Windows (рис 1.3) містить послідовний комплекс операцій між компонентами системи в режимі користувача, які

спрямовані на безпечну перевірку ідентичності суб'єкта, не розкриваючи збережену інформацію про пароль [6]. Коли користувачі ініціюють процедуру входу в систему, системний процес LogonUI збирає введені облікові дані суб'єкта. Система пересилає отримані дані керуючій службі Winlogon, яка створює захищений канал зв'язку з Local Security Authority Subsystem Service шляхом ініціації системного виклику функції LsaLogonUser. Підсистема LSA, яка отримує запит на автентифікацію, перенаправляє масив даних до пакета автентифікації MSV1_0 (msv1_0.dll), що здійснює одностороннє криптографічне перетворення введеного пароля, обчислюючи його NTLM-хеш. Знайдений хеш надсилається службі SAM, яка звертається до зашифрованої бази даних, вилучає збережений NTLM-хеш відповідного користувача та виконує їх порівняння. Якщо обидва криптографічні відбитки збігаються, Security Accounts Manager повертає унікальний ідентифікатор безпеки (SID) користувача та ідентифікатори всіх локальних груп, до яких він входить.

В ієрархії цілей зломисників у середовищі Active Directory та на локальних робочих станціях пара LSASS та SAM посідає перше місце. Це зумовлено тим, що ці компоненти є легітимним сховищем облікових даних за дизайном самої операційної системи. Будь-яке успішне зчитування оперативної пам'яті процесу lsass.exe або копіювання файлів бази даних SAM надає атакуючому доступ до хешів або паролів у відкритому вигляді, що дозволяє миттєво скомпрометувати поточний вузол.

1.3 Функціонування Security Reference Monitor у забезпеченні контролю доступу.

Security Reference Monitor (SRM) – це основний компонент у Windows, який працює в режимі ядра та забезпечує дотримання політик контролю доступу в операційній системі. Головною метою SRM є забезпечення централізованого контролю надання суб'єктам повноважень на доступ до захищених об'єктів системи, гарантуючи, що будь-яка операція взаємодії між ними відбувається відповідно до поточної політики безпеки.

SRM виконує кілька функцій для підтримки безпеки операційної системи Windows [7]:

- Застосування контролю доступу до системних об'єктів, зокрема файлів, розділів реєстру та процесів.
- Забезпечення дотримання політик безпеки в масштабах усієї системи та реалізація принципу найменших привілеїв.
- Генерація події аудиту безпеки та відстеження дій користувача, які пов'язані з безпекою операційної системи.

Центральним елементом Security Reference Monitor є точка прийняття рішень, яка оцінює вхідні запити на доступ відповідно до встановленої політики безпеки [8]. Цей компонент, підтримуючи систему контролю доступу для забезпечення узгодженості політик безпеки системи, застосовує правила, що дозволяють оцінити відповідність запропонованої суб'єктом дії щодо об'єкта в режимі реального часу. Концепція функціонування SRM визначається трьома основними властивостями, які забезпечують дотримання політик контролю доступу:

- Повне посередництво, яке вимагає виклику Security Reference Monitor для кожної спроби доступу до захищених ресурсів.
- Ізольованість та захист від втручання, яке вимагає забезпечити захист SRM від несанкціонованої модифікації або обходу механізмів безпеки.

- Верифікованість, яка вимагає реалізацію Security Reference Monitor достатньо простою та відокремленою від інших функцій операційної системи.

Зазвичай Security Reference Monitor вбудований безпосередньо в ядро операційної системи Windows для використання апаратно-примусової ізоляції. Таке розташування мінімізує ризик компрометації засобів авторизації та забезпечує захист від несанкціонованого доступу до системних ресурсів, дозволяючи монітору блокувати несанкціоновані операції в ядрі середовища виконання. Реалізація SRM у привілейованому режимі дозволяє централізовано контролювати всі операції доступу до об'єктів ядра та підтримувати узгодженість політик безпеки на рівні всієї системи.

Важливим аспектом механізму контролю доступу Security Reference Monitor є встановлення користувацьких привілеїв, які замінюють дискреційні обмеження, визначені для захищених об'єктів. На відміну від стандартних дозволів на основі DACL, привілеї регулюють здатність суб'єкта виконувати глобальні адміністративні операції на системному рівні. Типовим прикладом є SeDebugPrivilege, активація якого в межах токена доступу процесу дозволяє отримати дескриптор будь-якого активного процесу в системі. SRM виконує динамічну перевірку наявності та статусу таких привілеїв перед ініціюванням циклу порівняння ідентифікаторів безпеки, гарантуючи пріоритет привілеїв над дискреційним контролем доступу на рівні об'єкта.

1.4 Ідентифікатори безпеки та їх роль у визначенні суб'єктів доступу

Ключовим елементом моделі безпеки Windows є ідентифікатори безпеки (SID), які працюють із компонентами технологій авторизації та контролю доступу в операційній системі Windows. Розмежування доступу до ресурсів є однією з основних концепцій безпеки Windows, оскільки операційна система повинна мати унікальний спосіб розрізняти суб'єктів, що взаємодіють із захищеними об'єктами. На відміну від імен облікових записів, які можуть змінюватися або повторно присвоюватися після їх видалення, SID залишається незмінним протягом усього життєвого циклу об'єкта безпеки, що забезпечує стабільність механізмів контролю доступу.

Ідентифікатори безпеки (SID) – це унікальне значення змінної довжини, яке генерується під час створення облікового запису та використовується для ідентифікації принципала безпеки, що представляє будь-яку сутність, яку може автентифікувати система [9]. SID забезпечує дискреційний контроль доступу шляхом визначення дозволів у налаштуваннях безпеки об'єкта, а також спрощує процеси автентифікації користувачів.

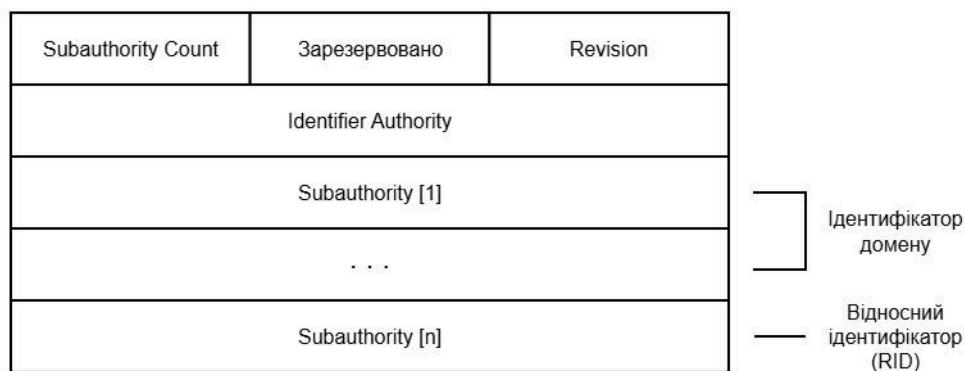


Рис. 1.4 Структура ідентифікатора безпеки (SID)

Структура SID (рис. 1.4) складається з декількох частин, що дозволяють системі визначити походження та рівень повноважень принципала безпеки [10]:

- Revision, який вказує на поточну версію структури SID.

- Sub-Authority Count, який визначає кількість компонентів підпорядкованих повноважень (Sub-Authorities), які знаходяться після значення ідентифікатора повноважень (Identifier Authority).
- Identifier Authority, який визначає найвищий рівень повноважень, що може видавати ідентифікатори безпеки (SID) для певного типу принципала безпеки.
- Sub-Authorities, що містить одне або кілька значень, які однозначно ідентифікують домен або локальний комп'ютер. Усі значення, крім останнього, складають ідентифікатор домену, який генерується з комбінації часу створення домену та псевдовипадкового початкового значення, унеможливаючи передбачення або відтворення простору SID заданого домену.
- Relative Identifier (RID) – унікальне значення у межах домену, що визначає конкретного користувача чи групу.

Таблиця 1.1 Універсальні ідентифікатори безпеки

S-1-0-0	Ідентифікує групу без учасників.
S-1-1-0	Ідентифікує групу, яка містить всіх користувачів системи.
S-1-5-7	Ідентифікує анонімний вхід до системи.
S-1-5-11	Ідентифікує автентифікованих користувачів.
S-1-5-18	Ідентифікує спеціальний обліковий запис, який використовується операційною системою.

Windows містить набір попередньо визначених ідентифікаторів безпеки (табл. 1.1), що дозволяє стандартизувати конфігурації безпеки операційної системи. Існування незмінних SID надає змогу компонентам операційної системи та програмам послідовно взаємодіяти з попередньо визначеними контекстами безпеки незалежно від конкретної конфігурації

середовища. Використання сталих ідентифікаторів дозволяє також уникнути залежності від динамічно згенерованих SID, що зменшує ризики контролю доступом та спрощує застосування політик у різних конфігураціях.

Хоча архітектура SID забезпечує надійну основу для контролю доступу в операційних системах Windows, вона не позбавлена низки вразливостей в конструкції свого механізму:

- Впровадження історії SID дозволяє зловмиснику ввести довільний ідентифікатор безпеки суб'єкта з високими привілеями безпосередньо в атрибут `sidHistory`, призначений для збереження попереднього SID принципала безпеки.
- Дублікати SID найчастіше виникають внаслідок неправильного клонування образів віртуальних машин, що обходить механізм розподілу RID Master та призводить до того, що кілька принципалів безпеки використовують однаковий SID.
- Осиротілі SID. Після видалення облікового запису користувача або групи відповідний SID може зберігатися в DACL захищених об'єктів, створюючи осиротілі записи, що ускладнює аудит безпеки. Крім того, у разі випадкового призначення новоствореному обліковому запису ідентичного SID, він може неявно успадкувати всі дозволи, що попередньо були надані видаленому профілю.

1.5 Архітектурні особливості та функціонування маркерів доступу

В архітектурі операційної системи Windows проблема представлення ідентифікаційних даних та привілеїв автентифікованого принципала безпеки у формі, яку можна ефективно використовувати під час виконання, вирішується шляхом застосування механізму маркера доступу. Маркер доступу (Access Token) – це захищений об'єкт ядра, який містить інформацію про ідентифікаційні дані та контекст безпеки автентифікованого користувача [11]. З метою уникнення повторних перевірок облікових даних під час кожної спроби доступу до ресурсу, операційна система приєднує копію токена доступу до кожного активного процесу та потоку, створеного від імені автентифікованого суб'єкта. Маркер доступу надає Security Reference Monitor механізм контролю доступу шляхом порівняння інформації з токена на відповідність дискреційному списку контролю доступу запитуваного об'єкта.

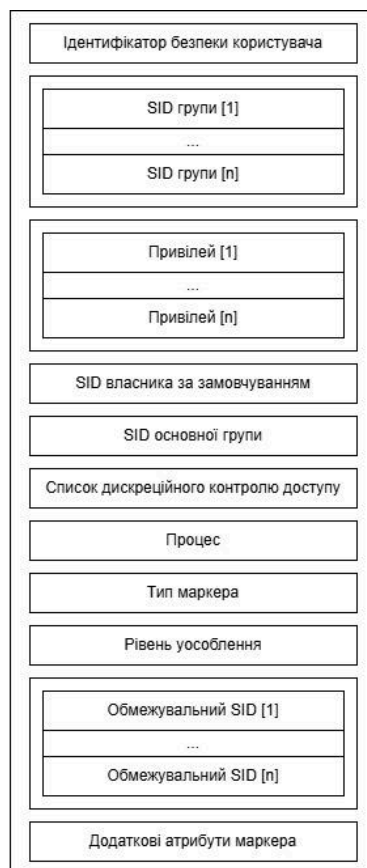


Рис. 1.5 Компоненти маркера доступу

Маркер доступу складається з компонентів (рис. 1.5), кожен з яких виконує окрему роль в оцінці контролю доступу [12]:

- Ідентифікатор безпеки облікового запису користувача.
- Список ідентифікаторів безпеки груп, до яких належить користувач.
- Список привілеїв, наданих користувачеві або його групам.
- Ідентифікатор безпеки користувача, який за замовчуванням стає власником будь-якого об'єкта, створеного користувачем.
- Ідентифікатор безпеки основної групи користувача.
- Список дискреційного контролю доступу, які операційна система за замовчуванням застосовує до об'єктів, створених користувачем.
- Ідентифікатор процесу, який спричинив створення маркера доступу.
- Тип маркера, що вказує на його належність до певної категорії токенів доступу.
- Рівень уособлення, що визначає ступінь, до якого служба може прийняти контекст безпеки клієнта, представленого даним маркером.
- Обмежувальні ідентифікатори безпеки, які дозволяють обмежити доступ потоку до рівня, нижчого за дозволений користувачеві.
- Додаткові атрибути маркера, що містять інформацію про параметри сесії та середовище виконання.

Класифікація маркерів доступу в архітектурі Windows визначається сукупністю критеріїв, що охоплюють функціональне призначення об'єкта та рівень його інтеграції компоненти операційної системи. Модель безпеки Windows розрізняє дві основні категорії маркерів доступу:

- Первинні токени (Primary Tokens)
- Токени уособлення (Impersonation Tokens)

Первинний токен призначається кожному процесу в момент ініціалізації та визначає його контекст безпеки за замовчуванням для будь-якої операції доступу до ресурсів [13, с. 174]. Дочірні процеси та потоки успадковують копію первинного токена батьківського процесу, забезпечуючи узгодженість контексту безпеки в контексті їх ієрархії.

Security Reference Monitor виконує перевірку дозволів шляхом аналізу маркера ініціюючого процесу для подальшого визначення типу доступу, який слід надати.

Токени уособлення відіграють важливу роль у функціонуванні системних служб, оскільки вони дозволяють тимчасово прийняти контекст безпеки іншого принципала з метою виконання певної операції [13, с. 181]. Необхідність у подібному механізмі виникає, зокрема, коли службі необхідно отримати доступ до об'єкта системи, що належить іншому користувачеві. Завдяки уособленню системна служба може видавати себе за користувача, що ініціював запит, та отримати доступ до об'єкта навіть тоді, коли безпосередній доступ служби буде заблоковано. Такий підхід дозволяє операційній системі нівелювати потребу надання системним службам надлишкових підвищених привілеїв та подальшого контролю доступу.

Одним з ключових механізмів захисту системи від підвищення привілеїв є технологія розділених токенів (split tokens), що лежить в основі архітектури служби контролю облікових записів користувачів. Принцип розділених токенів полягає у тому, що під час автентифікації суб'єкта з підвищеними правами система генерує фільтрований маркер (Filtered Token) з обмеженими привілеями та повний маркер доступу (Full Token), що містить адміністративні повноваження. Така дворівнева структура дозволяє примусово запускати усі процеси та додатки за замовчуванням, використовуючи Filtered Token, тоді як використання Full Token стає можливим лише після явного підтвердження запиту через службу контролю облікових записів користувачів, що забезпечує надійний захист адміністративних повноважень та перешкоджає автоматичному виконанню шкідливого коду з підвищеними правами у фоновому режимі.

1.6 Модель контролю доступу в операційній системі Windows

Модель контролю доступу визначає комплексну структуру, яка керує взаємодією між автентифікованими принципами безпеки та захищеними системними об'єктами. В операційній системі Windows цей принцип базується на концепції захищених об'єктів та встановлює набір взаємопов'язаних компонентів, які працюють на окремому рівні абстракції та дозволяють впроваджувати принцип найменших привілеїв, забезпечуючи багаторівневий механізм управління доступом на рівні об'єктів системи. Майже кожен системний ресурс контролюється за допомогою набору структур безпеки, що дозволяє приймати централізовані та узгоджені рішення щодо авторизації для різних категорій ресурсів, підтримуючи при цьому детальні політики безпеки.

В операційній системі Windows використовується переважно дискреційна модель контролю доступу (DAC). Дискреційна модель дозволяє власнику ресурсу визначати, яким користувачам дозволено отримати доступ до захищеного об'єкта, а також вказувати дії, які дозволено виконувати [14]. Такий підхід забезпечує детальний контроль над доступом, захищаючи критично важливі дані від неавторизованих осіб. Хоча операційна система Windows переважно використовує модель DAC, сучасні версії операційної системи включають додаткові механізми, включаючи контроль облікових записів користувачів (UAC), рівні цілісності та привілеї, створюючи гібридну систему контролю доступу.

Ресурсно-орієнтовані засоби контролю доступу, такі як списки контролю доступу, є найпоширенішим механізмом реалізації політик дискреційної моделі контролю доступу. Список контролю доступу (ACL) пов'язує дозволenu операцію з об'єктом та визначає всіх суб'єктів, які можуть отримати доступ до об'єкта, разом з їхніми правами на об'єкт, забезпечуючи простий спосіб надання або заборони доступу для певного користувача або груп користувачів [15, с. 11]. ACL складається з окремих записів, які називаються Access Control Entries (ACE) та визначають

конкретного принципала безпеки, вказуючи набір прав доступу, які йому надаються або в яких йому відмовляється.

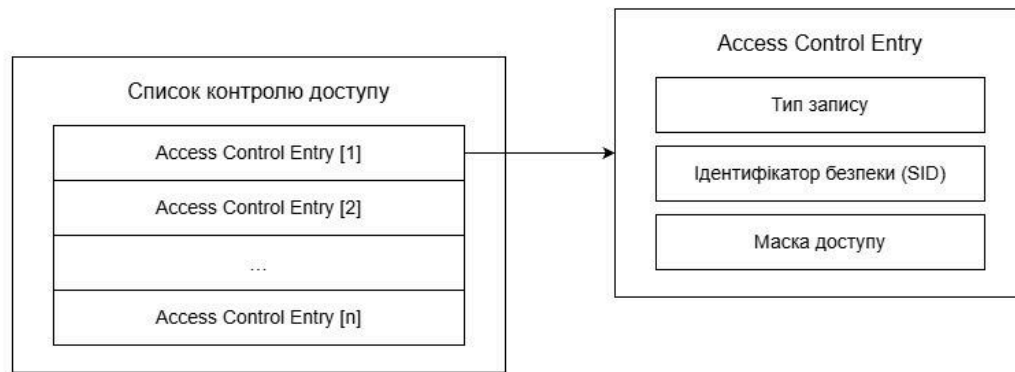


Рис. 1.6 Структура Access Control Entry (ACE)

Кожен елемент ACE (рис. 1.6) складається з трьох полів [16]:

- Тип, що визначає семантичну роль запису та розрізняє записи ACL.
- Ідентифікатор безпеки (SID), що визначає принципала або групу, до якої застосовується запис.
- Маска доступу, що визначає права доступу до об'єкта системи.

Кожен дескриптор безпеки може містити два типи ACL для захищеного об'єкта [17]:

- Дискреційний список керування доступом (DACL), що визначає користувачів та групи, яким дозволено або заборонено доступ до захищеного об'єкта.
- Системний список керування доступом (SACL), який дозволяє контролювати спроби доступу до захищеного об'єкта.

Основним елементом моделі контролю доступом є дескриптор безпеки (SD) – структура даних в операційній системі Windows, яка містить інформацію про безпеку, пов'язану з будь-яким захищеним об'єктом [18]. Основна функція SD полягає в наданні операційній системі способу перевірки власника об'єкта та механізму встановлення доступу до нього різних користувачів і груп.

Дескриптори безпеки містять набір із чотирьох основних елементів, які описують усі дозволи, що застосовуються до цього об'єкта [19]:

- Ідентифікатор безпеки (SID) власника, що визначає володільця принципала безпеки захищеного об'єкта.
- Ідентифікатор безпеки (SID) групи.
- Дискреційний список контролю доступу (DACL).
- Системний список контролю доступу (SACL).

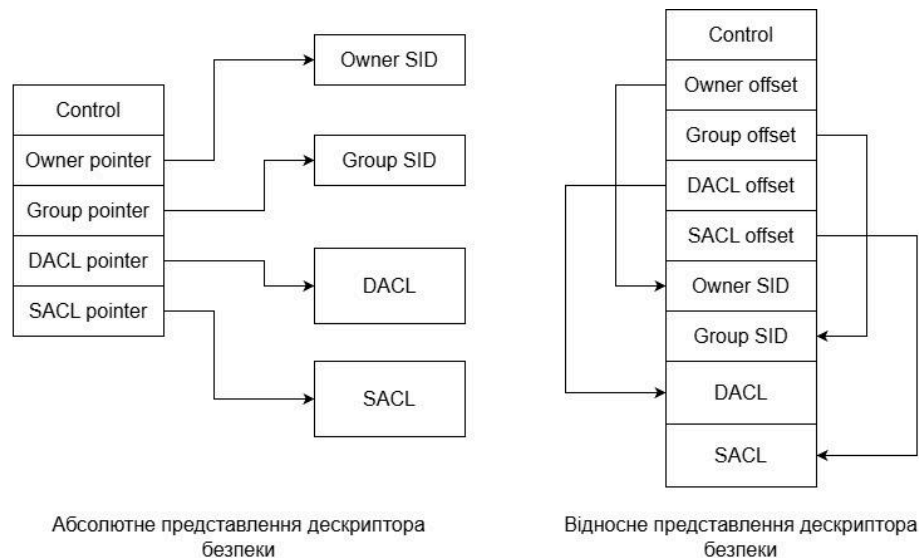


Рис. 1.7 Формати представлення дескрипторів безпеки

Ядро операційної системи Windows підтримує два формати представлення дескрипторів безпеки: абсолютний та відносний. В абсолютній формі дескриптор містить вказівники, які посилаються на абсолютну адресу пам'яті, за якою зберігаються дані, тоді як відносний дескриптор безпеки збирає компоненти в один безперервний буфер та зберігає ці розташування як позитивні зміщення відносно початку своєї структури (рис. 1.7). Менеджер об'єктів та драйвери файлової системи обробляють перетворення між цими двома формами, гарантуючи збереження логічного вмісту дескриптора безпеки незалежно від його фізичного представлення.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні основи архітектури безпеки та механізмів авторизації в операційній системі Windows. Аналіз моделі безпеки Windows визначив, що захист не забезпечується одним компонентом, а виникає в результаті взаємодії між кількома підсистемами, що працюють у режимах ядра та користувача.

Дослідження основних підсистем безпеки визначило, що LSASS та SAM утворюють основу автентифікації та управління обліковими даними у Windows. Послідовний процес автентифікації дозволяє визначити, що безпека вбудована на найпершому етапі взаємодії суб'єкта з системою, тоді як цілісність цього ланцюга є передумовою для всіх наступних рішень щодо контролю доступу. Таким чином, компрометація LSASS або бази даних SAM являє собою системну загрозу, яка обходить усі наступні механізми авторизації.

Дослідження ідентифікаторів безпеки показало, що стабільність SID протягом життєвого циклу принципала безпеки є критичною властивістю, що забезпечує послідовне застосування політик, однак також створює вразливості, зокрема через маніпуляції історією SID, неправильне клонування віртуальних машин та записи ACL, що залишилися без даних, які представляють собою реальні поверхні для атак у корпоративних середовищах. Архітектура розділених токенів, що лежить в основі контролю облікових записів користувачів, забезпечує конкретну реалізацію принципу найменших привілеїв на рівні процесу.

Таким чином, результати дослідження даного розділу забезпечують теоретичну основу, де буде розроблено прикладний аналіз конкретних сценаріїв авторизації та методів атаки.

2 МЕТОДИ ПІДВИЩЕННЯ ПРИВІЛЕЇВ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS ТА СПОСОБИ ЇХ ВИЯВЛЕННЯ

2.1 Класифікація та концепція методів підвищення привілеїв

У контексті безпеки сучасних операційної системи привілеї визначають дозволи облікового запису користувача або групи виконувати певної операції, які в іншому випадку були б заборонені політикою контролю доступу системи [20, с. 170]. Управління та забезпечення дотримання привілеїв складають основну функцію механізмів контролю доступу, які вбудовані в архітектури операційних систем. У моделі безпеки Windows права доступу відрізняються від привілеїв, оскільки вони контролюють доступ принципала безпеки до захищених об'єктів, тоді як привілеї керують виконання загальносистемних операцій, що виходять за межі контролю доступу на рівні об'єктів.

Теоретичну основу управління привілеями в архітектурі Windows визначає модель безпеки, виконує поділ суб'єктів операційної системи на класи привілеїв та забезпечує дотримання цих розподілів за допомогою обов'язкових механізмів контролю доступу. У даній структурі ескалація привілеїв описує несанкціоновану зміну або підвищення дозволів, що дозволяє перейти від базового облікового запису з нижчим рівнем привілеїв до іншого суб'єкта системи з вищим рівнем дозволів [21]. Головною метою цього процесу є подолання обмежень механізмів безпеки Windows та порушення цілісності моделі контролю доступу для отримання доступу до захищених ресурсів, адміністративних функцій або критичних компонентів системи. Наслідки успішного підвищення привілеїв можуть спричинити несанкціонований доступ до конфіденційних даних, повну компрометацію операційної системи або порушення роботи критичної інфраструктури, що робить таку категорію атак однією з найбільш серйозних у сучасному ландшафті загроз.

Існує два основні напрямки ескалації привілеїв, які відрізняються початковими умовами, поверхнею атаки та методами, які вони використовують:

- Вертикальне підвищення привілеїв.
- Горизонтальне підвищення привілеїв.

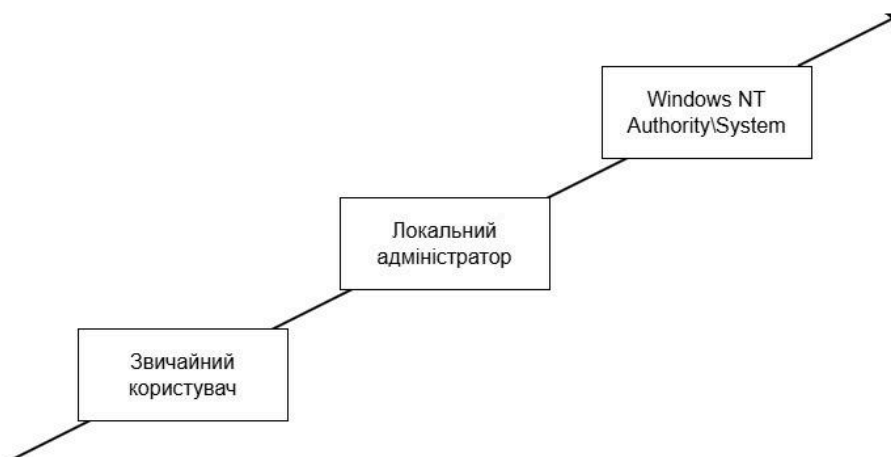


Рис. 2.1 Вертикальна ескалація привілеїв

Вертикальна ескалація привілеїв описує основну модель загрози, коли користувач, що працює в контексті з нижчими привілеями, повинен виконати певні операції для підвищення своїх прав доступу [22, с. 55]. Прикладом даного типу атак є перехід від стандартного непривілейованого облікового запису користувача до системного контексту (рис. 2.1), що надає зловмиснику повний контроль над цільовою системою. Така ескалація робить усі засоби контролю безпеки неефективними, оскільки процес системного рівня може змінювати журнали аудиту та вимикати активні засоби захисту. В операційній системі Windows вертикальне підвищення привілеїв використовується для отримання можливості змінювати конфігурацію системи, встановлювати шкідливе програмне забезпечення, маніпулювати політиками безпеки або отримувати доступ до конфіденційних даних інших користувачів.

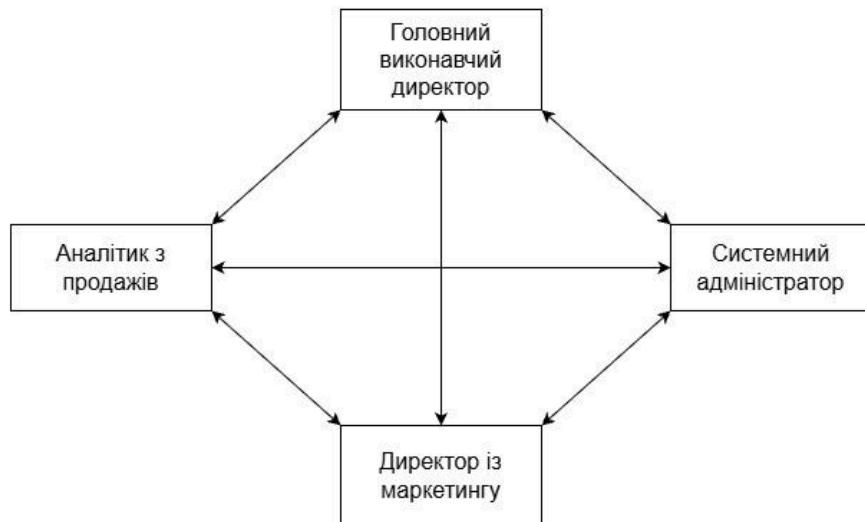


Рис. 2.2 Горизонтальна ескалація привілеїв

Горизонтальна ескалація привілеїв дозволяє отримати доступ до облікових даних іншого принципалу безпеки в мережі з вищими привілеями, ніж ті, що були використані для початкового закріплення. Гарним даного типу атак є випадок, коли зловмисник з дозволами звичайного користувача викрав облікові дані адміністратора мережі, використовуючи їх для подальшого пересування мережею або закріплення в інфраструктурі (рис. 2.2). Наприкінці фази горизонтальної ескалації привілеїв зловмисник зазвичай має декілька налагоджених точок входу віддаленого доступу до цільової системи, а також знає методи уникнення виявлення засобами захисту [22, с. 56].

У складній корпоративній інфраструктурі ланцюги складних атак поєднують обидва напрямки ескалації привілеїв, використовуючи горизонтальну ескалацію для досягнення привілейованого облікового запису та вертикальну ескалацію для отримання повного адміністративного контролю над мережею.



Рис. 2.3 Модель практичної реалізації ескалації привілеїв

Практична реалізація ескалації привілеїв побудована на структурованій моделі (рис. 2.3), що описує послідовний процес кількох взаємозалежних кроків [23]:

- **Попередня взаємодія (Pre-Engagement)** встановлює умови, визначає ціль та визначає бажаний результат, оцінюючи ресурси та методи, доступні для його досягнення. Якість підготовки до проведення атаки безпосередньо визначає ефективність та результативність усіх наступних фаз.
- **Пасивна розвідка (Passive Reconnaissance)** дозволяє виконати систематичний збір інформації про цільову інфраструктуру за допомогою методів, що не передбачають прямої взаємодії з кінцевими системами. На цьому етапі пошук інформації виконується за допомогою загальнодоступних джерел: веб-сайти організацій, профілі технічного персоналу в соціальних мережах та репозиторії коду, що містять фрагменти конфігурації. Пасивна розвідка дозволяє побудувати попередню модель інфраструктури та технологічного середовища цілі. Інформація про розгорнуті версії операційної системи, архітектуру служб каталогів, що використовуються, імена та обов'язки привілейованого персоналу безпосередньо впливає на вибір векторів ескалації на подальших етапах. Оскільки пасивна розвідка не передбачає прямого контакту з цільовими системами,

головною перевагою даного етапу є непомітність, що дозволяє уникнути будь-якого виявлення засобами захисту.

- Активна розвідка (Active Reconnaissance) передбачає пряму технічну взаємодію з цільовою інфраструктурою для визначення відкритих сервісів, конфігурацій систем та топології мережі. Цей етап охоплює такі методи, як сканування мережевих портів, ідентифікацію операційної системи та цілеспрямовані запити до відкритих адміністративних інтерфейсів. У контексті ескалації привілеїв активна розвідка створює ризик виявлення, оскільки генерує мережевий трафік для побудови карту досяжних систем та робочих служб цільової інфраструктури.
- Перерахування служб (Service Enumeration) виконує систематичний аналіз сервісів, процесів, облікових записів та конфігурацій, присутніх у цільовій системі. Сукупний результат даного аналізу дозволяє побудувати структуровану карту поверхні атаки, яка визначає пріоритетний перелік потенційно придатних для експлуатації систем.
- Експлуатація доступу (Access Exploitation) становить основу процесу ескалації привілеїв, коли зловмисник застосовує одну або декілька технік для отримання базового доступу до систем цільової інфраструктури. Цей етап характеризується використанням вразливостей, неправильних конфігурацій або слабких місць у системах, виявлених під час попередніх фазах атаки.
- Підвищення привілеїв (Privilege Escalation) охоплює подальші дії для отримання підвищеного доступу до систем цільової інфраструктури.

2.2 Техніки підвищення привілеїв на рівні операційної системи Windows.

Систематична класифікація методів ескалації привілеїв відповідно до характеру базової вразливості або механізму, що експлуатується, забезпечує аналітичну основу для розуміння ландшафту кібератак та розробки механізмів захисту критичних компонентів систем.

Для підвищення швидкості детектування та реагування на загрози було впроваджено матрицю MITRE ATT&CK [25], яка забезпечує систематизовану класифікацію тактик, технік та процедур, що використовуються зловмисниками для здійснення несанкціонованого підвищення привілеїв (табл. 2.1).

Таблиця 2.1 Техніки підвищення привілеїв

Abuse Elevation Control Mechanism	Більшість сучасних систем містять вбудовані механізми контролю підвищення прав, призначені для обмеження привілеїв, які користувач може виконувати в системі. Зловмисники можуть обійти дані механізми, використовуючи вбудовані можливості операційної системи у спосіб, що не передбачався розробниками для неавторизованих суб'єктів [26, с. 264].
Access Token Manipulation	Операційна система Windows використовує маркери доступу для визначення власника запущеного процесу. Зловмисник може маніпулювати маркерами доступу для зміни контексту безпеки поточних процесів, щоб створений процес виглядав так, ніби він є дочірнім процесом іншого процесу або належить іншому користувачу [26, с. 264].
Account Manipulation	Маніпуляції з обліковими записами охоплюють методи, які дозволяють зловмиснику змінювати властивості існуючих облікових записів, підвищуючи їх доступ у скомпрометованому середовищі. Поширені методи даного типу ескалації привілеїв можуть включати додавання облікових записів до привілейованих груп,

	модифікацію авторизованих ключів SSH та зміну налаштувань облікових даних для забезпечення постійного доступу до системи.
Boot or Logon Autostart Execution	Автозапуск програм під час завантаження або входу суб'єкта в систему відноситься до класу методів збереження та підвищення привілеїв, оскільки вони дозволяють налаштувати автоматичне виконання шкідливого коду під час завантаження системи або входу в обліковий запис. Найпоширеніші механізми можуть включати автоматичне виконання програм, розміщених у спеціально призначених каталогах або реєстрах Windows, що дозволяє перенаправляти їх виконання на код, контрольований зловмисником.
Boot or Logon Initialization Scripts	Скрипти ініціалізації описують окремий клас технік підвищення привілеїв, що дозволяють використовувати механізми сценаріїв для виконання шкідливого коду під час завантаження або ініціалізації входу в систему.
Create or Modify System Process	Створення або зміна системного процесу охоплює методи, які дозволяють створювати або змінювати процеси системного рівня для багаторазового виконання шкідливого навантаження. Створюючи нову службу, зловмисник може забезпечити виконання шкідливого коду з привілеями рівня SYSTEM, отримуючи повний контроль над скомпрометованою системою.
Domain or Tenant Policy Modification	Модифікація політики домену передбачає внесення змін до політик безпеки на рівні домену, зокрема об'єкти групової політики, політики автентифікації та призначення адміністративних ролей, з метою послаблення контролю доступу або створення нових можливостей отримання підвищеного доступу.
Escape to Host	Метод Escape to Host використовує слабкі місця ізолюваного контейнера або віртуалізованого середовища для отримання доступу та виконання довільного коду на базовому хості з

	системним рівнем дозволів.
Event Triggered Execution	Event Triggered Execution охоплює методи, які дозволяють виконати шкідливий код у відповідь на певні системні події, використовуючи системні механізми, що надаються операційною системою.
Exploitation for Privilege Escalation	Експлуатація вразливостей програмного забезпечення дозволяє зловмиснику використовувати недоліки в компонентах ядра, привілейованих службах або драйверах сторонніх розробників для виконання коду, контрольованого зловмисником.
Process Injection	Техніка ін'єкції в процеси дозволяє виконувати довільний код в адресному просторі окремого, зазвичай легітимного процесу [26, с. 265].
Scheduled Task/Job	Scheduled Task/Job передбачає створення або зміну запланованих завдань, якими керує служба планувальника завдань (Task Scheduler), для виконання довільного коду у вказаний час від імені привілейованого облікового запису.
Valid Accounts	Valid Accounts використовує скомпроментовані облікові дані для автентифікації в систему, що дозволяє отримати початковий доступу, підвищити привілеї або уникнути виявлення засобами захисту.

Різноманіття методів та технік підвищення привілеїв показує, що жоден архітектурний механізм безпеки чи засіб контролю доступу не є достатнім, щоб гарантувати цілісність моделі безпеки Windows. Велика площа поверхні атаки, що охоплює вразливості ядра, слабкі властивості архітектури безпеки операційної системи, неправильна конфігурація та зловживання легітимним програмним забезпеченням, вимагає використання сучасних засобів виявлення та запобігання шкідливої активності, які постійно вдосконалюються для виявлення нових типів атак.

2.3. Аналіз програмних механізмів управління привілеями на рівні Windows API

Windows Application Programming Interface (Windows API) визначає фундаментальний програмний інтерфейс Windows, що надає розробникам стандартизований набір функцій, структур даних та протоколів для взаємодії з ключовими підсистемами операційної системи.

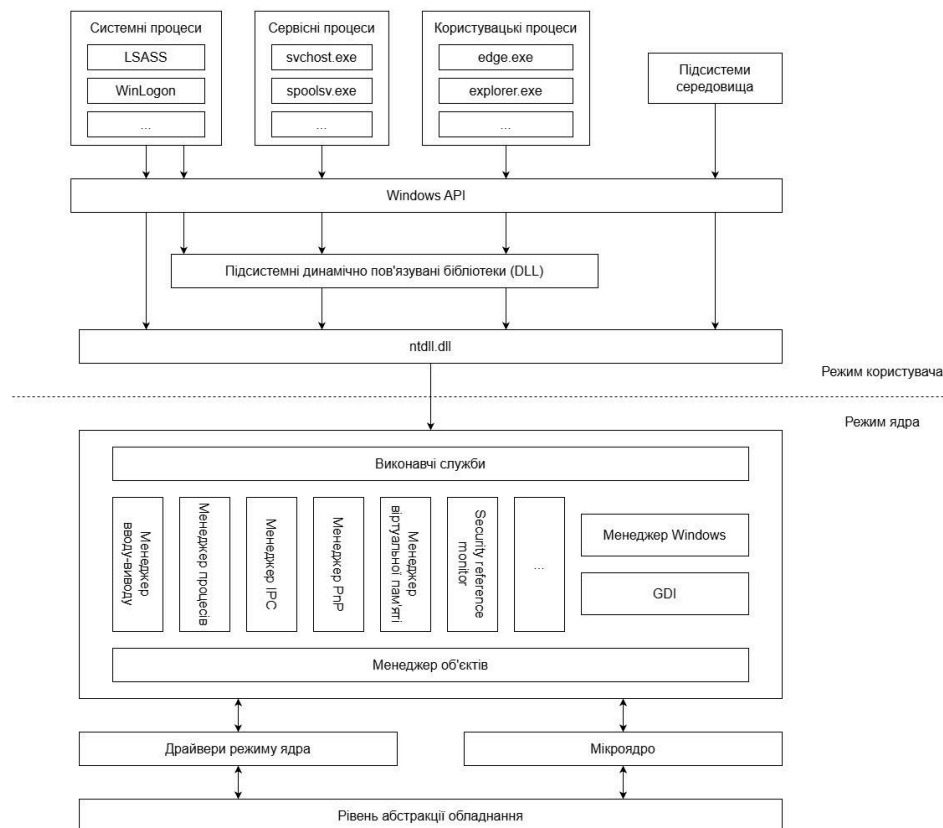


Рис 2.4 Зв'язок між суб'єктом та об'єктом через Windows API

Модель безпеки Windows реалізує розмежування доступу через багаторівневу архітектуру, яка побудована на принципі дискреційного контролю доступу, де кожен об'єкт системи асоційований з дескриптором безпеки, тоді як кожен суб'єкт діє від імені певного облікового запису за допомогою маркера доступу. Зв'язок між суб'єктом та об'єктом здійснює Security Reference Monitor (SRM), однак ініціювання та конфігурація цих механізмів відбувається за допомогою відповідних функцій Windows API (рис 2.4). Виступаючи інтерфейсом взаємодії між простором користувача та привілейованим простором ядра, API безпосередньо впливає на

стійкість операційної системи, оскільки релевантність його конфігурування визначає рівень безпеки захищених об'єктів.

Аналіз програмних інтерфейсів управління привілеями має як теоретичне, так і практичне значення, оскільки дозволяє виявити вразливі місця в архітектурі операційної системи, визначити можливі вектори атак на підвищення привілеїв та сформулювати конкретні рекомендації щодо вдосконалення механізмів моніторингу та протидії.

2.3.1. Аналіз Windows API для управління маркерами доступу

Маркер доступу (access token) є центральним об'єктом підсистеми безпеки Windows, оскільки асоціюється з кожним процесом та містить ідентифікатор безпеки користувача, перелік груп, до яких належить користувач, набір привілеїв, а також рівень цілісності. Операційна система Windows надає розгалужений набір API-функцій для програмного управління токенами доступу, що реалізовано в бібліотеці advapi32.dll.

Таблиця 2.2 Функції Windows API для управління маркерами доступу

Функція API	Призначення
OpenProcessToken	Дозволяє отримати дескриптор токена доступу для заданого процесу із зазначеним набором прав доступу.
AdjustTokenPrivileges	Дозволяє програмно вмикати та вимикати привілеї в маркері доступу.
DuplicateTokenEx	Дозволяє створити копію токена доступу з довільним рівнем уособлення.
CreateProcessWithTokenW	Дозволяє запустити довільний процес у контексті безпеки, визначеному переданим маркером доступу.
ImpersonateLoggedOnUser	Дозволяє поточному потоку

	тимчасово виконуватись у контексті безпеки іншого суб'єкта.
--	---

Незважаючи на функціональну повноту функцій API для управління маркерами доступу (табл. 2.2), їх архітектурні особливості та поведінка містять ряд недоліків для безпеки операційної системи:

- Більшість функцій управління токенами не генерують подій аудиту за замовчуванням, що унеможливорює ретроспективний аналіз дій злоумисника без попереднього налаштування розширеного аудиту.
- Функція DuplicateTokenEx у поєднанні з CreateProcessWithTokenW утворює фундамент для реалізації атак типу token theft, що дозволяє отримати дескриптор токена привілейованого процесу та запустити власний процес у його контексті [27].
- Механізм уособлення містить архітектурну вразливість, пов'язану з недостатнім процесом перевірок контексту виклику, зокрема рівень SecurityImpersonation надає суб'єкту можливість виконувати дії від імені іншого користувача в межах локальної системи, що дозволяє здійснювати атаки підвищення привілеїв.

2.3.2. Аналіз Windows API для управління дескрипторами безпеки та списками ACL

Дескриптор безпеки (Security Descriptor) визначає структуру даних, яка асоціюється з кожним об'єктом операційної системи Windows та визначає правила доступу до нього. Windows API надає широкий набір функцій для програмного управління цими компонентами, що зосереджений переважно в бібліотеці advapi32.dll.

Таблиця 2.3 Функції Windows API для управління дескрипторами безпеки та списками ACL

Функція API	Призначення
SetSecurityDescriptorDacl	Дозволяє встановити або замінити

	список контролю доступом у дескрипторі безпеки об'єкта.
SetNamedSecurityInfo	Дозволяє встановити компоненти дескриптора безпеки для іменованого об'єкта за вказаним ім'ям та типом.
AddAccessAllowedAce	Дозволяє додати до списку керування доступом запис, що явно дозволяє вказані права доступу для заданого суб'єкта безпеки.
GetSecurityDescriptorSacl	Дозволяє отримати покажчик на системний список керування доступом із заданого дескриптора безпеки.
AddAccessAllowedAce	Дозволяє додати до списку керування доступом запис, що явно забороняє вказані права доступу для заданого суб'єкта безпеки.
AddAccessDeniedAce	Дозволяє встановити дескриптор безпеки для заданого об'єкта ядра із зазначеним набором компонентів безпеки.
SetKernelObjectSecurity	Дозволяє встановити дескриптор безпеки для заданого об'єкта ядра із зазначеним набором компонентів безпеки.

Попри широкі можливості програмного управління дескрипторами безпеки та списками контролю доступу, низка API функцій створює передумови для маніпуляцій з правами доступу на рівні об'єктів системи:

- Низькорівнева структура ACL та відсутність обмежень перевірки у функціях, таких як AddAccessAllowedAce, можуть ненавмисно послабити рівень безпеки об'єкта, не викликаючи жодних попереджень на системному рівні.

- Функція SetKernelObjectSecurity дозволяє модифікувати DACL об'єктів ядра, що є основним вектором атак на підвищення привілеїв через маніпуляцію правами доступу до процесів системного рівня.
- Створення захищених об'єктів зі стандартними DACL, які успадковуються від токена процесу, можуть призводити до надмірно дозвільних конфігурацій доступу.

2.3.3. Синтез виявлених недоліків та роль систем SIEM у комплексному пом'якшенні наслідків

Дослідження функцій Windows API для керування маркерами доступу та дескрипторами безпеки виявляє послідовну закономірність структурних недоліків, які можна згрупувати в три загальні категорії:

- прогалини в аудиті
- активація привілеїв без повторної автентифікації
- неправильна конфігурація

Політика аудиту Windows за замовчуванням не охоплює більшість критично важливих для безпеки викликів API, тоді як виклики AdjustTokenPrivileges, DuplicateTokenEx, SetSecurityInfo та пов'язаних функцій не генерують записи журналу подій безпеки, що створює невидиму область, яка систематично використовується для ескалації привілеїв. Результуючий ландшафт неправильно налаштованих служб, дозвільних DACL за замовчуванням та неправильно впорядкованих ACL утворює постійну поверхню для атак, яку важко виправити за допомогою змін на рівні API без порушення зворотної сумісності.

З огляду на перелічені недоліки, комплексним підходом до їх усунення є впровадження системи Security Information and Event Management (SIEM), що агрегує дані журналів із журналів подій безпеки Windows, телеметрії Sysmon, журналювання блоків сценаріїв PowerShell та телеметрії Windows Defender для кінцевих точок. Обробляючи ці потоки

подій, SIEM може реалізувати логіку виявлення, яка вирішує кожен із виявлених недоліків:

- Активація привілеїв токенів процесами, які не належать до відомого білого списку, може викликати сповіщення високої серйозності, що вирішує проблему відсутності власного обмеження швидкості в API.
- Послідовності викликів API, характерні для крадіжки токенів, можуть бути виявлені за допомогою правил поведінкової кореляції, які охоплюють кілька ідентифікаторів подій.
- Зміни списку контролю доступом у розділах реєстру та файлах можна контролювати за допомогою правил кореляції SIEM, які позначають зміни в попередньо визначених критичних шляхах до об'єктів.
- Атаки на іменованний канал, що використовують функції Windows API, можна виявити за допомогою моделей на основі аномалій, які виявляють незвичайні шаблони рівнів привілеїв, пов'язані з процесами сервера іменованого каналу.

Важливим аспектом SIEM є застосування поведінкового аналізу шляхом навчання на нормальній активності конкретного середовища система може виявляти статистичні аномалії, що особливо ефективно проти інсайдерських загроз та атак, які використовують легітимні інструменти адміністрування.

Системи Security Information and Event Management дозволяють нівелювати архітектурні обмеження Windows API у сфері аудиту та контролю привілеїв, оскільки вони не лише збирають та зберігають відповідні події, але й забезпечують автоматизовану кореляцію, що є необхідною умовою для своєчасного виявлення атак підвищення привілеїв у реальному часі.

2.4 Архітектура та принципи функціонування SIEM-систем у сучасній інфраструктурі.

Стрімкий розвиток інформаційних технологій та поступове ускладнення загроз інформаційної безпеки зробили традиційні засоби захисту неспроможними для того, щоб забезпечити комплексний захист корпоративної інфраструктури. Величезні обсяги подій, які щоденно генеруються в різних сегментах мережі організації, унеможливають ручний аналіз логів. Системи управління інформацією та подіями безпеки стали основною платформою для вирішення завдань централізованого збору, обробки, кореляції подій, отриманих з усієї інфраструктури організації, а також оперативного реагування на інциденти безпеки.

Security Information and Event Management (SIEM) – це структурована система, яка інтегрує програмні та апаратні компоненти для полегшення збору, аналізу, кореляції та реагування на події інформаційної безпеки з різних джерел в IT-інфраструктурі організації. Системи SIEM аналізують великі обсяги подій в режимі реального часу для виявлення потенційних загроз, які можуть залишитися непоміченими, а також відхилень від нормальної моделі діяльності принципала безпеки.

SIEM-платформа об'єднує дві важливі технології, що дозволяє побудувати проактивну систему комплексного захисту всієї інфраструктури підприємства [27]:

- Security Information Management (SIM). Технологія SIM забезпечує базові можливості для збору, довгострокового зберігання та ретроспективного аналізу даних про події, отриманих з різних джерел в IT-інфраструктурі організації.
- Security Event Management (SEM). Технологія SEM фокусується моніторингу, кореляції та управління подіями безпеки в режимі реального часу для виявлення потенційних загроз безпеці та реагування на них.

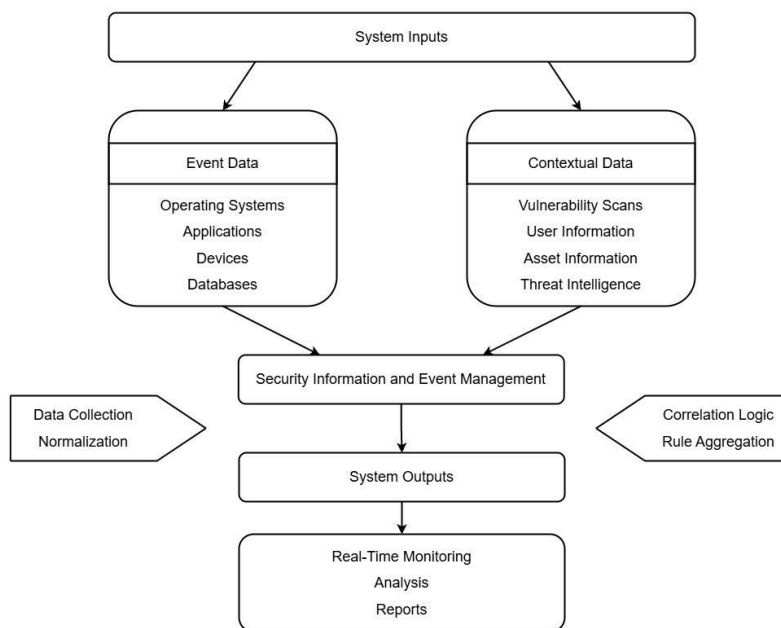


Рис. 2.5 Архітектура сучасних SIEM-систем

Архітектура сучасних SIEM-систем (рис. 2.5) складається з низки взаємопов'язаних компонентів, що виконують спеціалізовані функції на кожному етапі обробки даних [28]:

- Компоненти збору та агрегації даних (Data Collection and Ingestion) відповідають за безперервний збір журналів та телеметрії з кінцевих точок, серверів, баз даних, хмарних сервісів та міжмережевих екранів. Повнота збору даних на цьому рівні є основним визначальним фактором можливостей SIEM-систем, оскільки виявлення потенційних загроз безпеці можливо лише за умови збору подій з відповідних сегментів мережевої інфраструктури.
- Попередня обробка, парсинг та нормалізація (Parsing and Normalization). Оскільки зібрані дані надходять у різноманітних форматах, цей компонент стандартизує різноманітні логи в узгоджену схему, збагачуючи контекстною інформацією.
- Аналітика та кореляція подій (Correlation and Analytics) становить аналітичне ядро системи, що безперервно зіставляє поточкові події з кореляційними правилами для виявлення поведінки, що вказує на підозрілу активність або відомі методи атаки.

- Сховище даних (Data Storage) забезпечує довгострокове та захищене від несанкціонованої модифікації збереження нормалізованих даних.
- Панелі візуалізації та звітності (Alerting and Dashboards) надають аналітикам інтуїтивно зрозумілий інтерфейс у вигляді графіків, таблиць та діаграм для швидкого виявлення тенденцій, аномалій та патернів атак у реальному часі.

Збір та агрегація подій у SIEM-системах здійснюється за допомогою трьох основних технічних методів:

- Агентний метод передбачає встановлення спеціального програмного забезпечення на цільові пристрої. Агенти локально зчитують лог-файли, виконують первинну фільтрацію та передають структурований потік даних на центральний сервер.
- Безагентний метод покладається на налаштування систем для безпосередньої відправки даних на SIEM-сервер за допомогою стандартних протоколів (Syslog, Windows Event Forwarding, SNMP), що дозволяє зменшити навантаження на кінцеві пристрої.
- Метод на основі програмних інтерфейсів використовує інтерфейси прикладного програмування для віддалених запитів до віртуальних машин або хмарних платформ.

Функціонування сучасних SIEM-систем базується на принципі перетворення децентралізованих масивів даних у структуровану та пріоритезовану інформацію, придатну для швидкого аналізу. У процесі еволюції SIEM-платформи почали інтегрувати інструменти поведінкового аналізу та технології штучного інтелекту, що дозволило здійснювати проактивне виявлення раніше невідомих загрози на основі аномальних відхилень від легітимних поведінкових шаблонів мережевих пристроїв чи облікових записів користувачів.

Значні обсяги подій інформаційної безпеки, що генеруються в процесі функціонування засобів захисту та мережевих компонентів корпоративної інфраструктури, характеризуються високим рівнем

різномірності та неупорядкованості. Систематизація цього інформаційного потоку та його підготовка до кореляційного аналізу здійснюється в SIEM-системах за допомогою стандартизованого архітектурного конвеєра, що забезпечує стандартизацію подій до єдиного аналітичного формату.

В основі методології обробки подій безпеки лежать три ключові поняття:

- нормалізація, що дозволяє стандартизувати та приводити різномірні дані до єдиного формату.
- агрегація, що дозволяє об'єднувати схожі події для зниження навантаження на систему.
- кореляція, що здійснює зіставлення потокових подій з визначеними правилами для виявлення ознак шкідливої активності.

Нормалізація є першим критично важливим етапом обробки телеметрії, який стандартизує великі обсяги даних до єдиного формату, оскільки кожне програмне чи апаратне забезпечення використовує власні структури запису подій [29]. Під час нормалізації початкові дані проходять синтаксичний розбір, тоді як спеціальні модулі виділяють ключові семантичні поля, такі як часова мітка, ідентифікатор події, IP-адреси, порти, імена користувачів та іншу важливу інформацію. Після класифікації виокремлені атрибути формалізуються та записуються в структуровану інформаційну схему, що дозволяє оптимізувати використання сховища даних.

Окрім упорядкування та структуризації бази даних, нормалізація має багато інших переваг:

- Оптимізація простору для збереження даних.
- Зменшення часу відповіді на виконання запитів.
- Покращення цілісності даних та зменшення надмірності, що гарантує підтримку узгодженості записів.

Агрегація даних – це процес оптимізації потоку даних шляхом групування та об'єднання великої кількості подій, які відбуваються за

визначений проміжок часу, в один узагальнений масив [30]. Необроблені дані не дають багато інформації без належної агрегації та контексту з кількох джерел. Збереження великої кількості подібних подій в окремих записах може призводити до швидкого вичерпання дискового простору та перевантаження каналів зв'язку. Механізм агрегації аналізує потоки нормалізованих подій та об'єднує їх на основі збігу ключових полів протягом заданого часового відрізка, що усуває витрати на збереження дублікатів та суттєво підвищує продуктивність системи без втрати контексту безпеки.

Кореляція – це процес аналізу, спрямований на виявлення взаємозв'язків між різними нормалізованими подіями, отриманими в процесі функціонування різних компонентів корпоративної інфраструктури [31]. Основне призначення процесу кореляції в SIEM-системах полягає в побудові зв'язків між окремими подіями з метою виявлення складних кібератак.

Кореляційні правила поділяються на кілька категорій:

- Статична кореляція на основі правил використовує фіксовані логічні шаблони, що визначають комбінації типів подій, умови значень полів та обмеження часових інтервалів, для побудови пошукових запитів та ідентифікації відомих сигнатур атак.
- Поведінкова кореляція використовує моделі машинного навчання для виявлення аномальної активності, що дозволяє розпізнавати раніше невідомих методів кібератак.

Висновки до розділу 2

У другому розділі було розглянуто методи ескалації привілеїв в операційній системі Windows, що дозволить провести детальний аналіз програмних механізмів даних типів атак. Структурована модель життєвого циклу атаки, що охоплює попередню взаємодію, пасивну та активну розвідку, перелік служб, використання доступу та ескалацію привілеїв, довела, що успішне підвищення привілеїв є результатом багатоетапного процесу, ефективність якого безпосередньо залежить від якості усіх його фаз.

Огляд методів ескалації привілеїв, систематизованих за допомогою бази знань MITRE ATT&CK, продемонстрував широту поверхні атаки, доступної для зловмисника, який працює в операційній системі Windows. Значна частина методів підвищення привілеїв реалізується через легітимні інтерфейси операційної системи, що безпосередньо визначає вимоги до стратегії виявлення таких атак.

Аналіз функцій API Windows для керування токенами доступу та дескрипторами безпеки виявив структурні недоліки, які лежать в основі значної частини спостережуваної поверхні атаки. Поєднання цих недоліків створює постійну поверхню для атак, яку неможливо виправити за допомогою змін на рівні API без порушення зворотної сумісності, що встановлює фундаментальне обмеження для будь-якої суто превентивної стратегії безпеки.

Таким чином, системи управління інформацією та подіями безпеки представляють собою найбільш життєздатну архітектурну відповідь на обмеження рідного аудиту Windows. Архітектура сучасних систем SIEM, побудована навколо конвеєра збору даних, нормалізації, агрегації, кореляції та оповіщення, забезпечує необхідну основу для виявлення ескалації привілеїв у режимі реального часу.

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТЕХНІК ЕСКАЛАЦІЇ ПРИВІЛЕЇВ ТА МЕТОДІВ ЇХ ВИЯВЛЕННЯ

3.1 Постановка завдання та опис робочого середовища

Метою практичної частини даної роботи є демонстрація реалізації атак на підвищення привілеїв у середовищі операційної системи Windows та розробка механізмів їх виявлення за допомогою сучасних інструментів моніторингу безпеки. Для досягнення зазначеної мети були обрані дві техніки, що широко поширені в реальних кібератаках, зокрема DLL Hijacking та Token Impersonation/Theft.

Робоче середовище, в якому проводилися дослідження, складається з двох вузлів, об'єднаних у локальну мережу. Перший вузол відповідає за сервер Security Information and Event Management системи Wazuh версії 4.14, що здійснює централізований збір, агрегацію та кореляцію подій безпеки, отриманих з контрольованих кінцевих точок, а також надає відповідну аналітику.

Другий вузол є цільовою машиною під управлінням операційної системи Microsoft Windows 11 із встановленим агентом Wazuh, що функціонує як контрольована кінцева точка. На цільовій машині додатково розгорнуто утиліту Sysmon, що забезпечує розширений моніторинг та журналізацію системних подій Windows. Служба Sysmon реєструє та надає детальну інформацію про події створення процесів, мережевих з'єднань, завантаження модулів, доступ до процесів та інших, які є критично важливими для виявлення атак на підвищення привілеїв. Агент Wazuh, встановлений на цільовій машині під управлінням операційної системи Windows, зчитує відповідні журнали подій та передає їх до сервера Wazuh для подальшого аналізу та кореляції, що дозволяє виявити складні методи кібератак.

3.2 Атака підміни динамічно пов'язуваних бібліотек

3.2.1 Загальна характеристика та механізми техніки DLL Hijacking

Динамічно пов'язувана бібліотека (DLL) визначає спільний модуль, що містить виконуваний код, дані та ресурси, дозволяючи декільком процесам одночасно використовувати один функціонал. Техніка DLL Hijacking належить до категорії атак, що використовують особливості пошуку та завантаження динамічно пов'язуваних бібліотек операційною системою Windows. Під час запуску застосунку операційна система здійснює пошук необхідних бібліотек відповідно до визначеного порядку, який охоплює власний каталог програми, системні директорії Windows, поточну робочу директорію та каталоги, зазначені в системній змінній середовища PATH.

Сутність атаки полягає у розміщенні шкідливої бібліотеки з ідентичною назвою до тої, яку шукає цільовий застосунок, у директорії, що має вищий пріоритет у порядку пошуку, ніж легітимна бібліотека. Внаслідок таких дій операційна система завантажує змінену бібліотеку замість легітимної, що забезпечує виконання контрольованого коду в контексті цільового процесу. Якщо цільовий процес функціонує з підвищеними привілеями або запускається з відповідними дозволами, зломисник отримує можливість виконати код із підвищеним рівнем прав доступу.

3.2.2 Виявлення вразливих виконуваних файлів та практична реалізація атаки DLL Hijacking

Перед початком впровадження шкідливого навантаження, зломисник повинен ідентифікувати цільовий виконуваний файл, який функціонує в контексті підвищених привілеїв та намагається завантажити одну або декілька динамічно пов'язуваних бібліотек за неіснуючим шляхом. Необхідною умовою атаки DLL Hijacking є наявність системних каталогів, що доступні для запису без підвищених прав та перевіряються операційною системою раніше, ніж директорія, що містить легітимну бібліотеку.

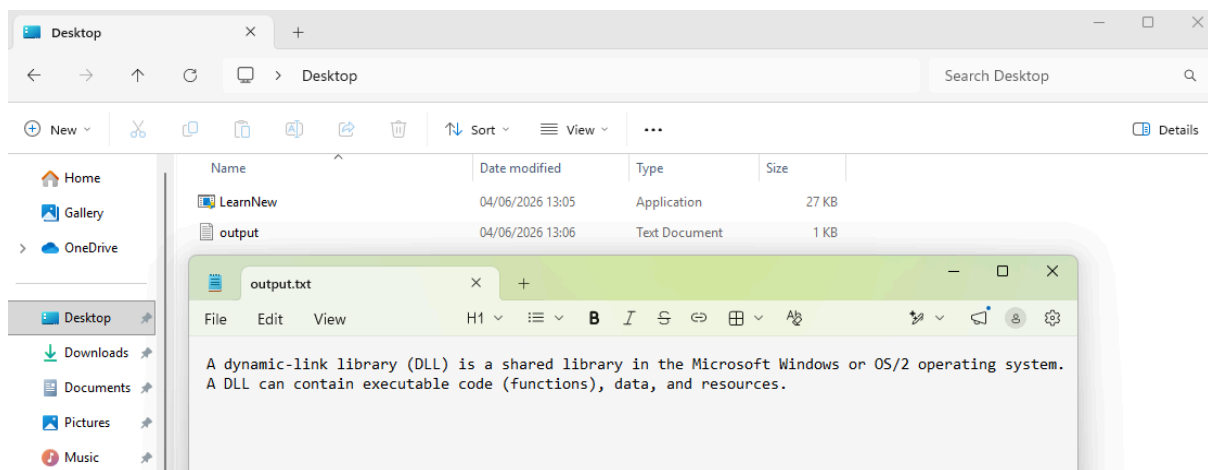


Рис. 3.1 Вразлива програма LearnNew.exe

На першому етапі реалізації атаки було створено тестову програму LearnNew.exe, яка намагається завантажити бібліотеку MyDll.dll під час виконання та здійснює запис текстового повідомлення у файл output.txt, розташованого у власному каталозі програми (рис. 3.1).

На другому етапі, використовуючи утиліту Process Monitor, необхідно знайти всі динамічно пов'язувані бібліотеки, які не було знайдено цільовим процесом за стандартними шляхами розташування.

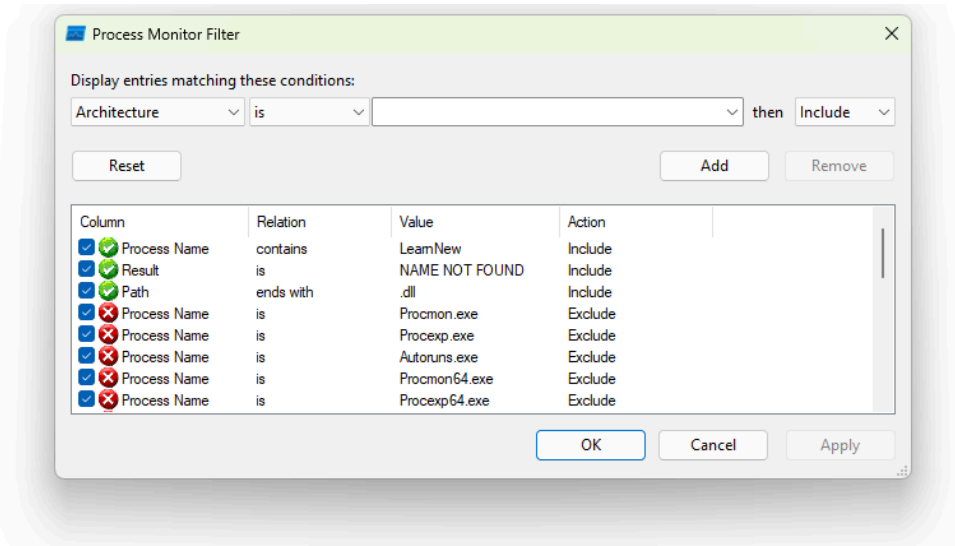


Рис. 3.2 Фільтрація подій невдалого завантаження бібліотек

Для визначення релевантних подій невдалого завантаження бібліотек було налаштовано відповідний фільтр (рис. 3.2), що знаходить усі події процесу LearnNew.exe, які одночасно відповідають трьом основним критеріям:

- Назва процесу (Process Name) містить ідентифікатор цільового виконуваного файлу.
- Результат (Result) зберігає значення "NAME NOT FOUND", що вказує на невдалий пошук файлу.
- Шлях до завантажуваного файлу закінчується на .dll, що вказує на невдалий пошук динамічно пов'язуваної бібліотеки.

Process Monitor - Sysinternals: www.sysinternals.com						
File Edit Event Filter Tools Options Help						
Time of Day	Process Name	PID	Operation	Path	Result	Detail
13:13:09.8067114	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\MSVCP140.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8067272	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\VCRUNTIME140_1.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8089590	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\VCRUNTIME140.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8100945	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\VCRUNTIME140.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8111352	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\VCRUNTIME140.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8180029	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8186407	LearnNew.exe	9464	CreateFile	C:\Windows\System32\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8190020	LearnNew.exe	9464	CreateFile	C:\Windows\System32\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8193529	LearnNew.exe	9464	CreateFile	C:\Windows\System32\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8206102	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\Desktop\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8210386	LearnNew.exe	9464	CreateFile	C:\Windows\System32\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8212913	LearnNew.exe	9464	CreateFile	C:\Windows\System32\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8215586	LearnNew.exe	9464	CreateFile	C:\Windows\System32\wbem\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8218039	LearnNew.exe	9464	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8222639	LearnNew.exe	9464	CreateFile	C:\Windows\System32\OpenSSH\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...
13:13:09.8225745	LearnNew.exe	9464	CreateFile	C:\Users\WinUser\AppData\Local\Microsoft\WindowsApps\MyDll.dll	NAME NOT FOUND	Desired Access: Read ...

Рис. 3.3 Перелік шляхів до динамічних бібліотек

У результаті фільтрації було визначено перелік шляхів (рис. 3.3), які вказують послідовність спроб знаходження бібліотеки MyDll.dll процесом LearnNew.exe, зокрема у каталозі розміщення виконуваного файлу поточного процесу, а також у системних директоріях Windows\System32 та Windows. Усі спроби повертають результат "NAME NOT FOUND", що свідчить про відсутність потрібної бібліотеки за жодним із перевірених шляхів. Для реалізації атаки зловмиснику необхідно розмістити власну бібліотеку в цій директорії, гарантуючи завантаження та виконання шкідливого коду.

```

bool IsElevated()
{
    HANDLE token;
    if (OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &token))
    {
        TOKEN_ELEVATION elevation;
        DWORD size;

        GetTokenInformation(token, TokenElevation, &elevation, sizeof(elevation), &size);

        CloseHandle(token);

        return elevation.TokenIsElevated;
    }
    return false;
}

```

Рис 3.4 Отримання інформації з маркера доступу

На третьому етапі була розроблена власна бібліотека MyDll.dll, яка реалізує перевірку рівня привілеїв поточного процесу. Код даної бібліотеки містить функція IsElevated (рис 3.4), яка використовує виклик OpenProcessToken для отримання маркера безпеки поточного процесу та GetTokenInformation для отримання інформації про підвищений статус маркера доступу.

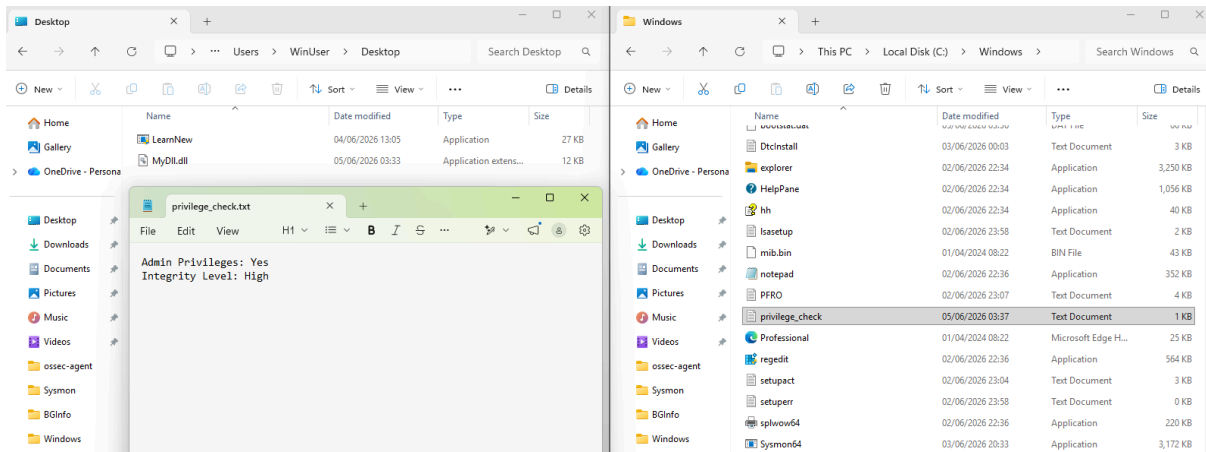


Рис 3.5 Результат успішної атаки DLL Hijacking

Результат успішного виконання атаки дозволяє створити файл у захищеній системній директорії C:\Windows\System32, що вказує на дію, виконану з правами адміністратора. Створений файл (рис 3.5) містить інформацію про привілеї поточного процесу та рівень цілісності (Integrity Level), що визначає ступінь довіри до процесу або об'єкта. Стандартні процеси користувача виконуються на рівні Medium, тоді як процеси з підвищеними привілеями – на рівні High, а системні служби – на рівні System.

3.2.3 Виявлення атаки засобами Wazuh

Після успішної підміни бібліотеки агент Wazuh, що функціонує на цільовій машині, зафіксував відповідні події Sysmon та передав їх на сервер.

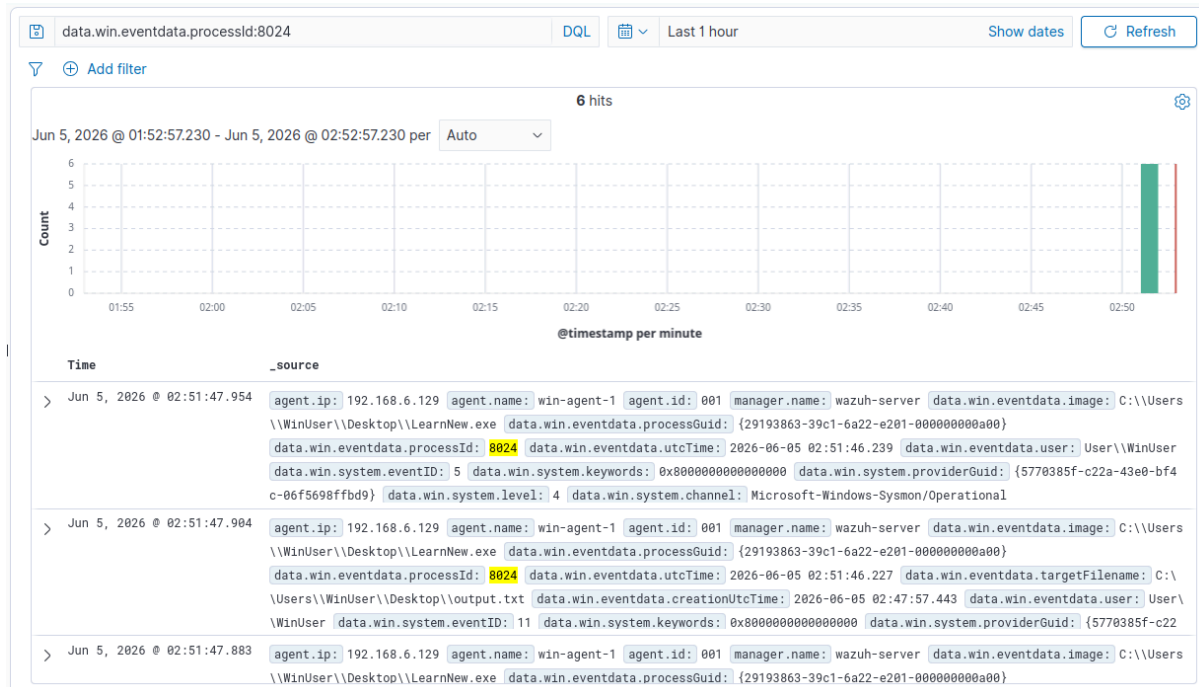


Рис. 3.6 Виявлені події успішної атаки

У результаті успішної атаки DLL Hijacking було зафіксовано 6 подій (рис. 3.6), що відповідають за певну дію процесу `LearnNew.exe`. Знайдені події описують повний життєвий цикл операції завантаження шкідливої бібліотеки: завантаження образу, артефакти створення файлів та пов'язані метадані процесу.

Table JSON

@timestamp	Jun 5, 2026 @ 02:51:47.868
_index	wazuh-archives-4.x-2026.06.05
agent.id	001
agent.ip	192.168.6.129
agent.name	win-agent-1
data.win.eventdata.hashes	MD5=1DF83CD7203A132917FA419A0F5F4ADA, SHA256=B8748F158A8C6F934FFFB5B119C5310184244218591B7BA50B7E2AEEDCD3F294, IMPHASH=9B55697CB4143C30E9C336CD2B582DF9
data.win.eventdata.image	C:\\Users\\WinUser\\Desktop\\LearnNew.exe
data.win.eventdata.imageLoaded	C:\\Users\\WinUser\\Desktop\\MyDll.dll
data.win.eventdata.processGuid	{29193863-39c1-6a22-e201-00000000a00}
data.win.eventdata.processId	8024
data.win.eventdata.signatureStatus	Unavailable
data.win.eventdata.signed	false
data.win.eventdata.user	User\\WinUser
data.win.eventdata.utcTime	2026-06-05 02:51:46.218

Рис 3.7 Подія завантаження образу файлу

Подія Sysmon завантаження образу файлу (рис 3.7), яка записує кожну операцію завантаження бібліотеки з повним шляхом та метаданими підпису, описує процес підміни динамічно пов'язуваних бібліотек.

Таблиця 3.1 Індикатори виявлення події завантаження образу файлу

Поле	Значення	Опис
data.win.eventdata.image	C:\\Users\\WinUser\\Desktop\\LearnNew.exe	Визначає поточний процес, який знаходиться в директорії робочого столу користувача, тоді як легітимне програмне забезпечення зазвичай розташовується в Program Files.
data.win.eventdata.imageLoaded	C:\\Users\\WinUser\\Desktop\\MyDll.dll	Визначає завантажену бібліотеку, яка знаходиться в каталозі, доступному для запису користувачем системи, що нетипово для легітимно файлу.
data.win.eventdata.signatureStatus	Unavailable	Вказує на відсутність цифрового підпису у

		завантаженому файлі, тоді як легітимна бібліотека, надана постачальником, міститиме дійсний цифровий підпис.
data.win.eventdata.signed	false	Підтверджує відсутність дійсного цифрового підпису, що може слугувати індикатором підозрілої активності.
data.win.eventdata.user	User\WinUser	Контекст безпеки користувача, який завантажив виконуваний образ файлу.

Детальний аналіз події завантаження шкідливої бібліотеки дозволив виділити ключові поля, кожне з яких може слугувати індикатором для виявлення подібних атак (табл. 3.1). Наступна подія Sysmon модифікацій файлової системи підтверджує доставку корисного навантаження, збору облікових даних або встановленню персистентності. Поєднання подій завантаження бібліотеки та подальшої модифікацій файлової системи, що походять від одного батьківського процесу, утворюють поведінкову модель, яка є ознакою успішної атаки DLL Hijacking.

3.2.4 Алгоритм атаки підміни динамічно пов'язуваних бібліотек

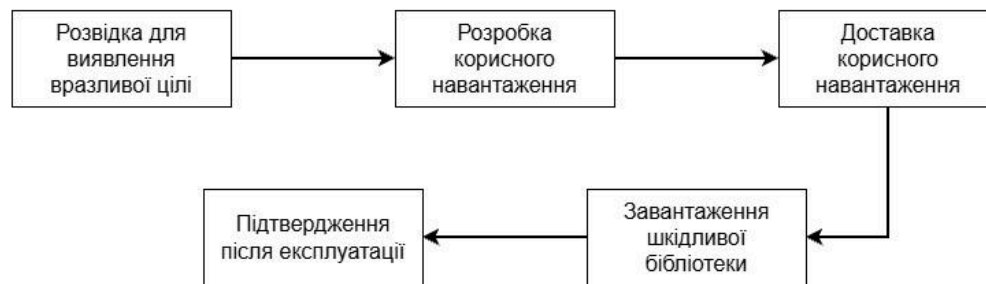


Рис. 3.8 Алгоритм атаки DLL Hijacking

На основі емпіричних даних, зібраних під час експерименту, та подальшого аналізу пов'язаних подій було розроблено наступний покроковий алгоритм атаки (рис. 3.8), який охоплює повний ланцюжок ескалації привілеїв від початкової розвідки до доставки корисного навантаження:

- Розвідка для виявлення вразливої цілі, що охоплює систематичну ідентифікацію цільового програмного забезпечення, яке завантажує одну або кілька динамічних бібліотек зі шляхів, що доступні обліковому запису користувача з низькими привілеями.
- Розробка корисного навантаження, що охоплює створення шкідливої динамічної бібліотеки для відтворення контрольованої поведінки легітимної цільової бібліотеки, вбудовуючи шкідливий код. Розроблена бібліотека після завантаження в цільовий процес забезпечує виконання корисного навантаження в контексті процесу.
- Доставка корисного навантаження, що полягає в розміщенні створеної бібліотеки у визначеному вразливому каталозі з вищим пріоритетом у порядку пошуку, ніж шлях до легітимної бібліотеки.
- Завантаження шкідливої бібліотеки, що охоплює запуск цільового програмного забезпечення та завантаження зміненої бібліотеки для виконання вбудованого корисного навантаження в контексті безпеки цільового процесу.

- Підтвердження після експлуатації, що охоплює перевірку виконання корисного навантаження з очікуваним рівнем привілеїв.

3.2.5 Розробка правил виявлення атаки підміни динамічно пов'язуваних бібліотек.

```
<group name="local, dll_hijacking,">

  <rule id="101101" level="10">
    <if_group>sysmon_event?</if_group>
    <field name="win.eventdata.imageLoaded" type="pcre2">.*\\Users\\.*\\.dll</field>
    <field name="win.eventdata.signed">false</field>
    <description>Potential DLL Hijacking: Unsigned DLL loaded.</description>
    <mitre>
      <id>T1574.0001</id>
    </mitre>
  </rule>

</group>
```

Рис. 3.9 Правило виявлення атаки DLL Hijacking

Для автоматизованого виявлення подібних подій було розроблено правило Wazuh (рис. 3.9), що спрацьовує при виконанні двох обов'язкових умов, які перевіряють існування цифрового підпису для динамічно пов'язуваної бібліотеки за шляхом, що відповідає регулярному виразу `.*\\Users\\.*\\.dll`. Такий підхід дозволяє виявляти завантаження невідомих бібліотек із нестандартних розташувань, що є характерною ознакою DLL Hijacking.

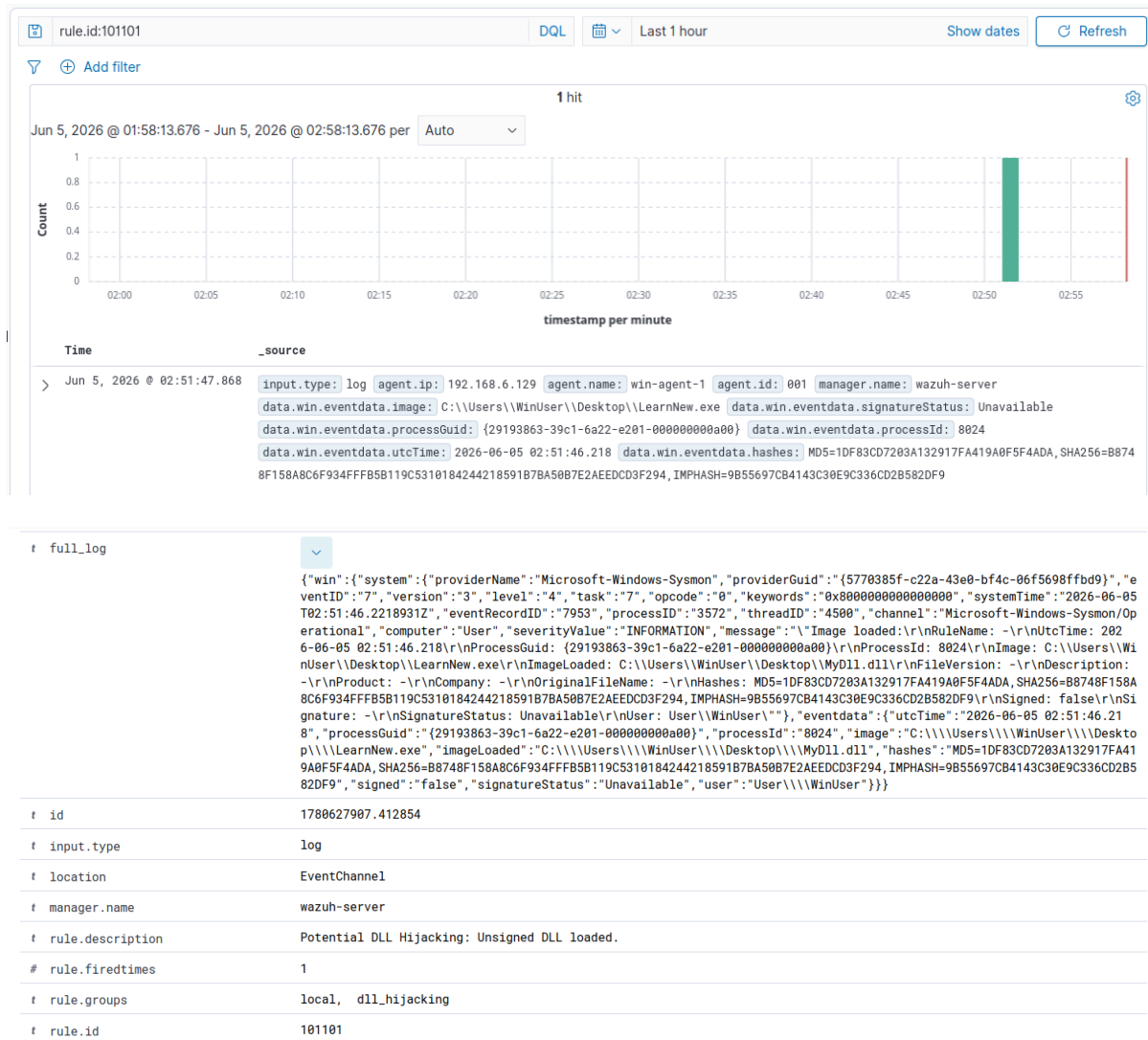


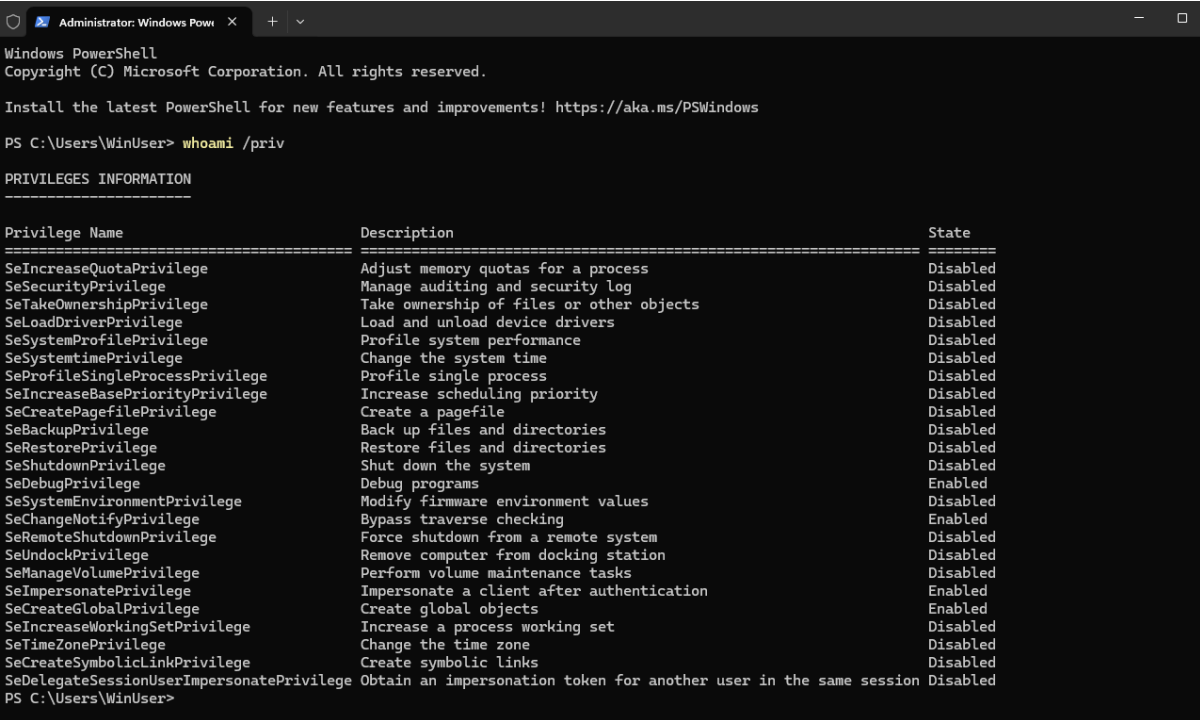
Рис. 3.10 Подія виявлення атаки DLL Hijacking

Після успішної атаки DLL Hijacking системою Wazuh було виявлено подію (рис. 3.10), що підтверджує коректну роботу розробленого механізму виявлення та здатність своєчасно сигналізувати про потенційну атаку підміни динамічно пов'язуваних бібліотек.

3.3 Атака крадіжки маркера безпеки

3.3.1 Загальна характеристика атаки

Атака Token Impersonation/Theft базується на механізмі маркерів безпеки, що є ключовим елементом моделі управління доступом операційної системи Windows. Кожен процес і потік в операційній системі Windows пов'язаний з маркерами безпеки, поділяються на первинні токени, які призначаються процесам під час створення та визначають контекст безпеки для всіх операцій, що виконуються цим процесом, та токени уособлення, які дозволяють потоку тимчасово прийняти ідентифікацію іншого принципала безпеки без створення нового процесу.



```

Administrator: Windows Powe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\WinUser> whoami /priv

PRIVILEGES INFORMATION
=====
Privilege Name        Description                                     State
-----
SeIncreaseQuotaPrivilege Adjust memory quotas for a process              Disabled
SeSecurityPrivilege    Manage auditing and security log                 Disabled
SeTakeOwnershipPrivilege Take ownership of files or other objects         Disabled
SeLoadDriverPrivilege Load and unload device drivers                  Disabled
SeSystemProfilePrivilege Profile system performance                      Disabled
SeSystemtimePrivilege Change the system time                          Disabled
SeProfileSingleProcessPrivilege Profile single process                         Disabled
SeIncreaseBasePriorityPrivilege Increase scheduling priority                   Disabled
SeCreatePagefilePrivilege Create a pagefile                              Disabled
SeBackupPrivilege      Back up files and directories                   Disabled
SeRestorePrivilege     Restore files and directories                   Disabled
SeShutdownPrivilege    Shut down the system                           Disabled
SeDebugPrivilege       Debug programs                                 Enabled
SeSystemEnvironmentPrivilege Modify firmware environment values              Disabled
SeChangeNotifyPrivilege Bypass traverse checking                       Enabled
SeRemoteShutdownPrivilege Force shutdown from a remote system             Disabled
SeUndockPrivilege       Remove computer from docking station            Disabled
SeManageVolumePrivilege Perform volume maintenance tasks                Disabled
SeImpersonatePrivilege Impersonate a client after authentication        Enabled
SeCreateGlobalPrivilege Create global objects                           Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                   Disabled
SeTimeZonePrivilege    Change the time zone                           Disabled
SeCreateSymbolicLinkPrivilege Create symbolic links                           Disabled
SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another user in the same session Disabled
PS C:\Users\WinUser>

```

Рис. 3.11 Набір привілеїв Windows

Windows визначає широкий набір привілеїв, що надаються обліковим записам і визначають можливість виконання певних системних операцій незалежно від дозволів на об'єкти файлової системи чи реєстру. Серед привілеїв (рис. 3.11), які є критично важливими для безпеки операційної системи, виокремлюються наступні:

- **SeImpersonatePrivilege**, який надає процесу можливість виконувати довільні дії в контексті безпеки іншого суб'єкта після його

автентифікації, що є необхідною умовою для багатьох методів маніпулювання токенами.

- SeAssignPrimaryTokenPrivilege, який надає можливість замінити основний токен процесу на рівні ядра, дозволяючи потоку запускати дочірній процес у довільному контексті безпеки.
- SeDebugPrivilege, який дозволяє власнику відкривати будь-який об'єкт процесу з правами повного доступу, минаючи звичайні перевірки доступу на рівні об'єктів.
- SeCreateTokenPrivilege, який надає найпряміший контроль, дозволяючи створювати довільний токен безпеки.

Сутність атаки полягає у дублюванні маркера безпеки привілейованого процесу та використанні отриманого дубліката для запуску нового процесу з правами системного облікового запису, що дозволяє ефективно підвищити привілеї до системного рівня SYSTEM. Оскільки дана атака здійснюється за допомогою легітимних викликів Windows API, вона не створює артефактів впровадження виконуваних файлів, що дозволяє унеможливити виявлення антивірусними засобами на основі сигнатур, які не виконують поведінковий аналіз послідовностей викликів API.

3.3.2 Практична реалізація атаки Token Impersonation/Theft

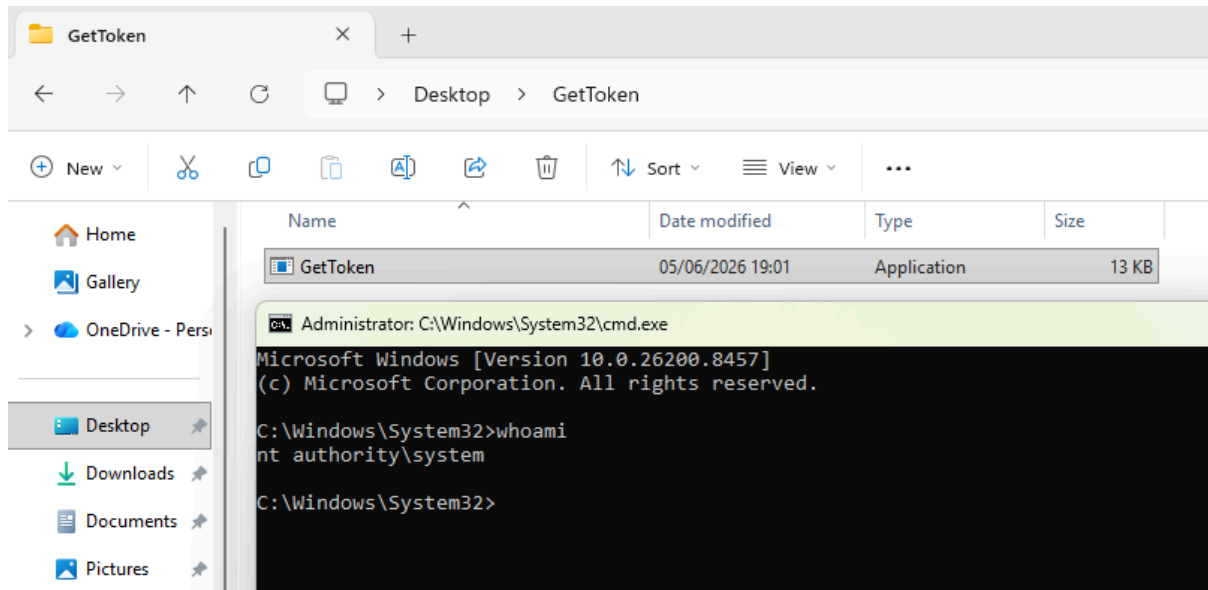


Рис. 3.12 Результат успішної атаки Token Impersonation/Theft

Для реалізації атаки було створено тестову програму GetToken.exe, що реалізує виявлення цільового привілейованого процесу winlogon.exe, відкриття зазначеного процесу за допомогою OpenProcess, отримання та дублювання його маркера безпеки, використовуючи функцію DuplicateTokenEx, а також запуск нового процесу командного рядка з привілеями системного рівня (рис. 3.12), використовуючи продубльований маркер.

Практичне значення техніки Token Impersonation/Theft полягає в можливості негайної ескалації привілеїв зловмисником, який отримав доступ до цільової системи, до системного рівня. На відміну від інших методів ескалації привілеїв, які можуть вимагати використання вразливостей програмного забезпечення або маніпулювання ресурсами файлової системи, дана атака використовує легітимні функції Windows API, що суттєво ускладнює її виявлення.

3.3.3 Виявлення атаки засобами Wazuh

Після успішної крадіжки маркера безпеки агент Wazuh, що працював на цільовій машині, зафіксував серію подій Sysmon та переслав їх на сервер Wazuh для подальшого аналізу.

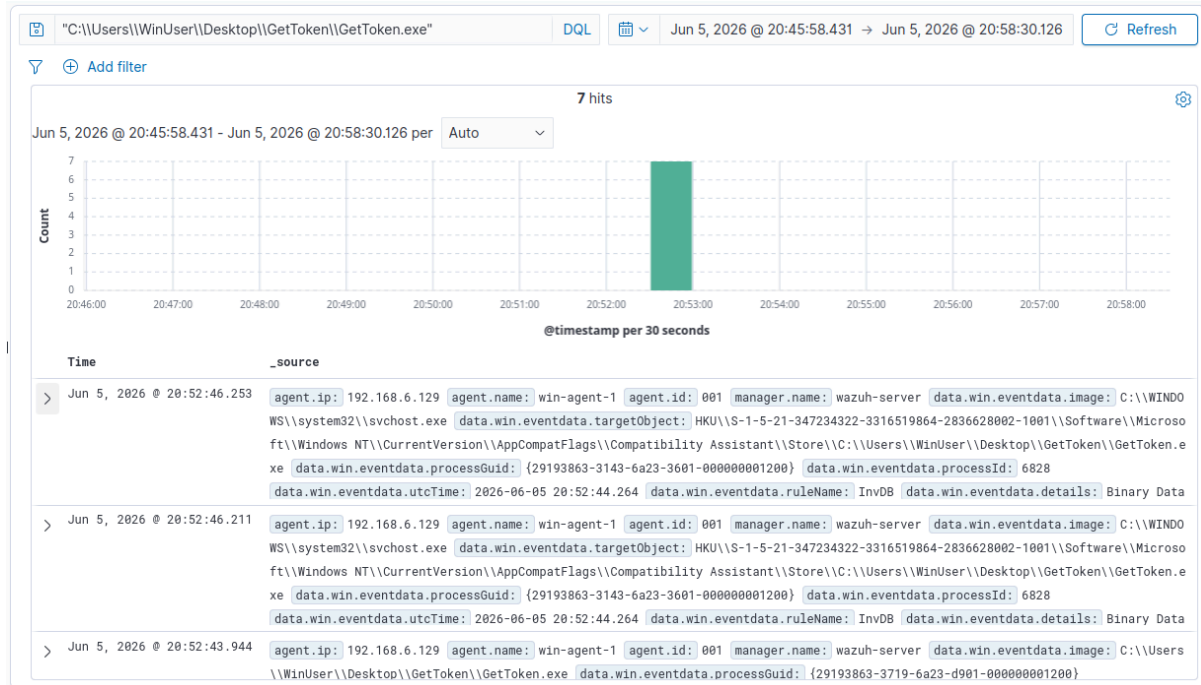


Рис. 3.13 Виявлені події успішної атаки

У результаті успішної атаки Token Impersonation/Theft було зафіксовано 7 подій (рис. 3.13), які разом описують послідовність атаки від початкового несанкціонованого доступу до створення дочірнього процесу з привілеями системного рівня.

@timestamp	Jun 5, 2026 @ 20:52:43.920
_index	wazuh-archives-4.x-2026.06.05
agent.id	001
agent.ip	192.168.6.129
agent.name	win-agent-1
data.win.eventdata.callTrace	C:\\WINDOWS\\SYSTEM32\\ntdll.dll+162164 C:\\WINDOWS\\System32\\KERNELBASE.dll+360c6 C:\\Users\\WinUser\\Desktop\\GetToken\\GetToken.exe+115a C:\\Users\\WinUser\\Desktop\\GetToken\\GetToken.exe+14a0 C:\\WINDOWS\\System32\\KERNEL32.DLL+2e957 C:\\WINDOWS\\SYSTEM32\\ntdll.dll+427c
data.win.eventdata.grantedAccess	0x1400
data.win.eventdata.sourceImage	C:\\Users\\WinUser\\Desktop\\GetToken\\GetToken.exe
data.win.eventdata.sourceProcessGUID	{29193863-3719-6a23-d901-000000001200}
data.win.eventdata.sourceProcessId	6624
data.win.eventdata.sourceThreadId	2692
data.win.eventdata.sourceUser	User\\WinUser
data.win.eventdata.targetImage	C:\\WINDOWS\\system32\\winlogon.exe
data.win.eventdata.targetProcessGUID	{29193863-309e-6a23-0a00-000000001200}
data.win.eventdata.targetProcessId	728
data.win.eventdata.targetUser	NT AUTHORITY\\SYSTEM

Рис. 3.14 Подія доступу до маркера системного процесу

Детальний аналіз отриманої інформації виявляє два різні типи подій, які разом охоплюють повний обсяг атаки. Перша відповідна подія ідентифікує спробу несанкціонованого доступу до маркера привілейованого системного процесу winlogon.exe (рис. 3.14).

Таблиця 3.2 Індикатори виявлення події доступу до маркера процесу

Поле	Значення	Опис
data.win.eventdata.sourceImage	C:\\Users\\WinUser\\Desktop\\GetToken\\GetToken.exe	Визначає поточний процес, що ініціює несанкціонований доступ, відкриваючи дескриптор процесу з системним рівнем привілеїв.
data.win.eventdata.targetImage	C:\\WINDOWS\\system32\\winlogon.exe	Ідентифікує цільовий процес.
data.win.eventdata.grantedAccess	0x1400	Маска доступу, що надає можливість зчитати токен доступу цільового процесу.
data.win.eventdata.sourceUser	User\\WinUser	Визначає контекст безпеки облікового запису користувача, що ініціює доступ.
data.win.eventdata.targetUser	NT AUTHORITY\\SYSTEM	Визначає контекст безпеки цільового процесу.

Дослідження першої події дозволило виділити основні поля, кожне з яких може слугувати індикатором для виявлення технік Token Impersonation/Theft (табл. 3.2). Поле sourceImage визначає процес-ініціатор GetToken.exe, який працює в контексті безпеки непривілейованого користувача, тоді як targetImage описує цільовий системний процес winlogon.exe. Артефактом можливої атаки крадіжки маркера безпеки також може бути поле grantedAccess, що містить значення 0x1400, яке поєднує

мінімальний набір прав PROCESS_QUERY_INFORMATION та PROCESS_QUERY_LIMITED_INFORMATION, необхідних для отримання токена цільового процесу.

# data.win.eventdata.description	Windows Command Processor
# data.win.eventdata.fileVersion	10.0.26100.8328 (WinBuild.160101.0800)
# data.win.eventdata.hashes	MD5=CED4A0B4CBF72E2520E0A2CCFF79370, SHA256=D5697FEF6995E992B9232A2B19665A297743427316C7225A5B772F0832F20FCA, IMPHASH=B0F049C014592B156EB1FA857E99CEB9
# data.win.eventdata.image	C:\Windows\System32\cmd.exe
# data.win.eventdata.integrityLevel	System
# data.win.eventdata.logonGuid	{29193863-30a5-6a23-e703-000000000000}
# data.win.eventdata.logonId	0x3e7
# data.win.eventdata.originalFileName	Cmd.Exe
# data.win.eventdata.parentCommandLine	"C:\Users\WinUser\Desktop\GetToken\GetToken.exe"
# data.win.eventdata.parentImage	C:\Users\WinUser\Desktop\GetToken\GetToken.exe
# data.win.eventdata.parentProcessGuid	{29193863-3719-6a23-d901-000000001200}
# data.win.eventdata.parentProcessId	6624
# data.win.eventdata.parentUser	User\WinUser
# data.win.eventdata.processGuid	{29193863-371a-6a23-db01-000000001200}
# data.win.eventdata.processId	8112
# data.win.eventdata.product	Microsoft Windows® Operating System
# data.win.eventdata.sessionId	1
# data.win.eventdata.user	NT AUTHORITY\SYSTEM

Рис 3.15 Подія створення нового процесу

Друга подія фіксує створення нового процесу cmd.exe, що відображено в полі image, з привілеями системного рівня (рис. 3.15).

Таблиця 3.3 Індикатори виявлення події створення нового процесу

Поле	Значення	Опис
data.win.eventdata.image	C:\Windows\System32\cmd.exe	Вказує на породжений процес з привілеями системного рівня.
data.win.eventdata.integrityLevel	System	Визначає найвищий рівень довіри до створеного процесу.
data.win.eventdata.user	NT AUTHORITY\SYSTEM	Визначає контекст безпеки цільового процесу.
data.win.eventdata.parentImage	C:\Users\WinUser\Desktop\GetToken\GetToken.exe	Визначає батьківський процес, що ініціює несанкціонований доступ.
data.win.eventdata.parentUser	User\WinUser	Визначає контекст безпеки облікового

		запису користувача, що ініціює доступ до привілейованого процесу.
--	--	---

Детальний аналіз події створення нового процесу дозволив виділити ключові поля, що можуть слугувати індикаторами для виявлення подібних атак (табл. 3.3). Поле `parentImage` підтверджує, що цей процес з підвищеними правами був створений безпосередньо `GetToken.exe`, який працює від імені стандартного користувача `User\WinUser`. Ланцюжок подій несанкціонованого доступу до маркера привілейованого системного процесу та створення нового процесу однозначно відображає успішне підвищення привілеїв шляхом крадіжки токена безпеки.

3.3.4 Алгоритм атаки крадіжки маркера безпеки

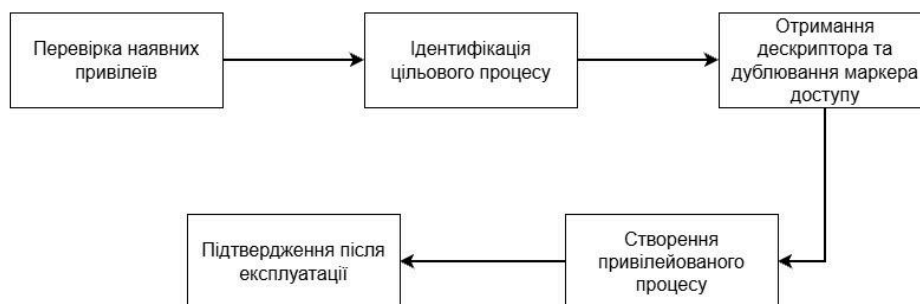


Рис. 3.16 Алгоритм атаки Token Impersonation/Theft

На основі отриманої інформації, зібраної з, та подальшого її аналізу було створено формалізований алгоритм (рис 3.16), який охоплює повний ланцюжок ескалації привілеїв від попередньої перевірки вимог до доставки корисного навантаження:

- Перевірка наявних привілеїв, що охоплює аудит контексту безпеки скомпрометованого процесу для перевірки наявності та статусу певних привілеїв системного рівня, необхідних для проведення подальшої атаки крадіжки маркера безпеки.

- Ідентифікація цільового процесу, охоплює знаходження активних системних компонентів та пов'язаних з ними ідентифікаторів процесів з метою визначення цільового процесу з привілеями підвищеного рівня.
- Отримання дескриптора та дублювання маркера доступу, що охоплює виклик низькорівневих функцій Windows API для створення копії токена доступу цільового процесу, зберігаючи початковий набір привілеїв та рівень цілісності.
- Створення привілейованого процесу, що охоплює ініціалізацію нового дочірнього процесу або впровадження корисного навантаження в контексті безпеки копії токена доступу цільового процесу.
- Підтвердження після експлуатації, що охоплює перевірку виконання корисного навантаження з очікуваним рівнем привілеїв.

3.2.5 Розробка правил виявлення атаки крадіжки маркера безпеки

```
<group name="local, token_stealing,">
  <rule id="101122" level="8">
    <if_group>windows</if_group>
    <field name="win.system.eventID" type="pcre2">^10$</field>
    <field name="win.eventdata.targetImage" type="pcre2">(?!)(winlogon|lsass|services|spoolsv|suchost)\.exe$</field>
    <field name="win.eventdata.grantedAccess" type="pcre2">^0x1400$|^0x1000$</field>
    <field name="win.eventdata.sourceUser" negate="yes" type="pcre2">^NT AUTHORITY\\(SYSTEM|LOCAL SERVICE|NETWORK SERVICE)$</field>
    <description>TokenTheft: A user process has gained unusual access to a system process</description>
  </rule>

  <rule id="101133" level="15" timeframe="5">
    <if_group>windows</if_group>
    <if_matched_sid>101122</if_matched_sid>
    <field name="win.system.eventID" type="pcre2">^1$</field>
    <field name="win.eventdata.user" type="pcre2">(?!)^NT AUTHORITY\\(SYSTEM|LOCAL SERVICE|NETWORK SERVICE)$</field>
    <field name="win.eventdata.parentUser" negate="yes" type="pcre2">^NT AUTHORITY\\(SYSTEM|LOCAL SERVICE|NETWORK SERVICE)$</field>
    <same_field>win.system.computer</same_field>
    <description>Possible Privilege Escalation: Token Impersonation</description>
    <mitre>
      <id>T1134.001</id>
    </mitre>
  </rule>
</group>
```

Рис. 3.17 Правило виявлення атаки Token Impersonation/Theft

Для автоматизованого виявлення подібних атак було розроблено два взаємодоповнюючих правила Wazuh (рис. 3.17).

Перше правило спрацьовує при виявленні події доступу до процесу, де цільовий процес відповідає критично важливим для системи виконуваним файлам, зокрема winlogon.exe, lsass.exe, services.exe,

spoolsv.exe та svchost.exe, значення grantedAccess, яке містить маску доступу, необхідну для зчитування токена цільового процесу, відповідає 0x1400 або 0x1000, а обліковий запис sourceUser не належить до набору легітимних системних облікових записів. Таке поєднання дозволяє ефективно виявляти спроби несанкціонованого доступу до привілейованих процесів із метою читання їх маркерів безпеки.

Друге правило функціонує як корельований індикатор ескалації, який спрацьовує протягом 5 секунд після події, виявленої попереднім правилом, та додатково вимагає, щоб створення дочірнього процесу було з привілеями системного рівня, тоді як батьківський обліковий запис залишався звичайним обліковим записом. Такий підхід дозволяє ефективно відрізнити легітимну активність системи від фактичної крадіжки токенів, оскільки послідовний шаблон доступу до системного процесу з подальшим породженням нового з привілеями системного рівня є індикатором успішного підвищення привілеїв.

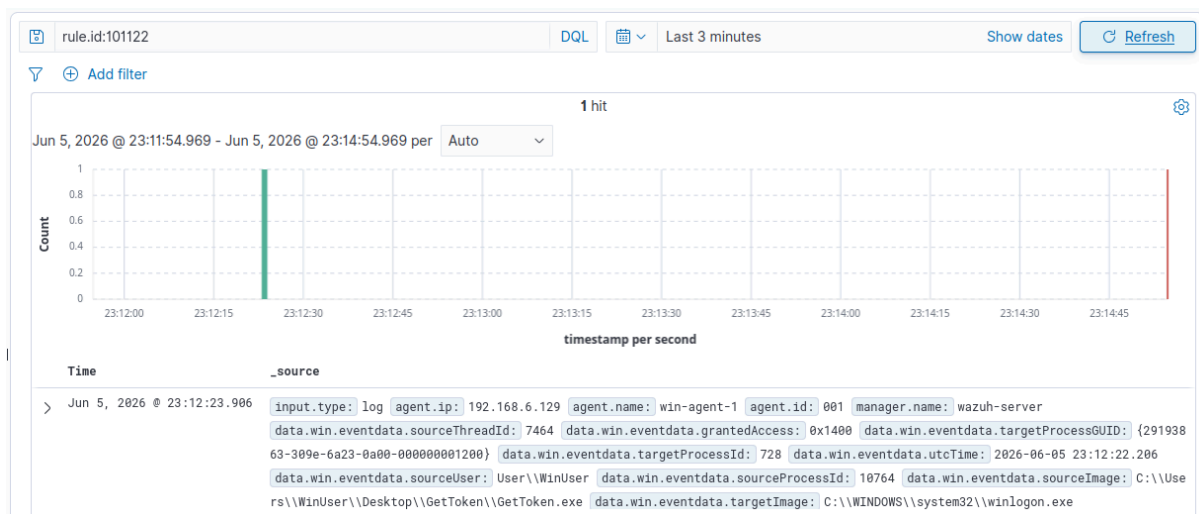


Рис. 3.18 Подія виявлення доступу до маркера системного процесу

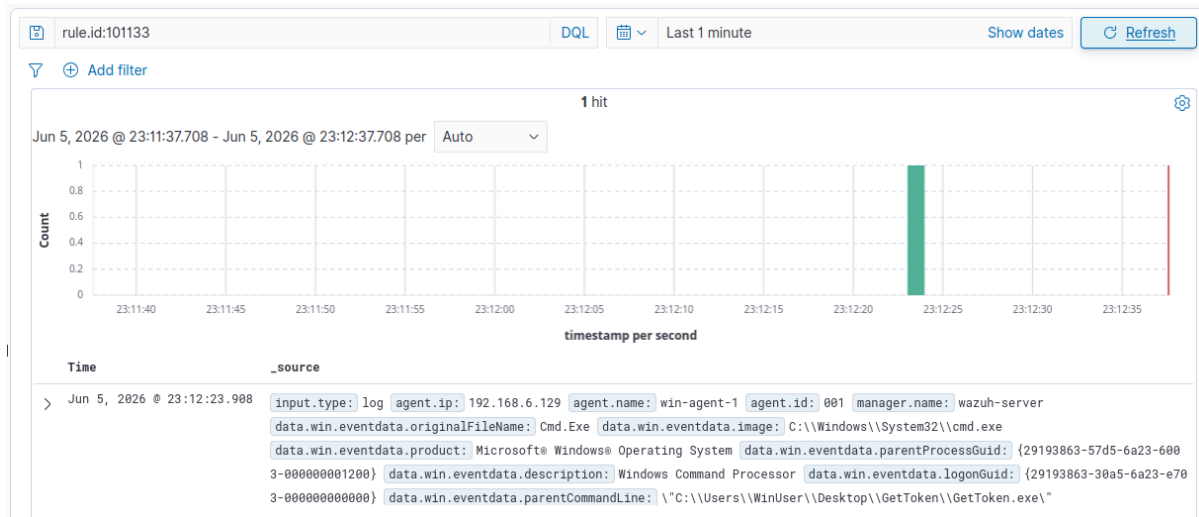


Рис. 3.19 Подія виявлення атаки Token Impersonation/Theft

Після успішної атаки Token Impersonation/Theft системою Wazuh було виявлено спроби несанкціонованого доступу до привілейованих процесів (рис 3.18) та спроби створення нового процесу з підвищеними привілеями (рис 3.19), що підтверджує коректну роботу механізму виявлення подій та здатність своєчасно сигналізувати про потенційну атаку крадіжки маркера безпеки.

Висновки до розділу 3

У третьому розділі було представлено практичне дослідження двох методів ескалації привілеїв, що широко використовуються в реальних кібератаках, реалізованих у контрольованому лабораторному середовищі, що складається з кінцевої точки Windows 11, що контролюється платформою Wazuh SIEM з увімкненою телеметрією Sysmon.

Дослідження підміни динамічно пов'язуваних бібліотек продемонструвало, що атаку можна здійснити шляхом розміщення шкідливої бібліотеки в каталозі, доступному для запису користувачем, який перевіряється раніше в порядку пошуку операційної системи, ніж шлях до легітимної бібліотеки. Після успішної заміни агент Wazuh зафіксував події Sysmon, що охоплюють повний життєвий цикл атаки та дозволили побудувати її алгоритм. На основі побудованого алгоритму було розроблено та перевірено правило виявлення Wazuh, засноване на кореляції ознак відсутності дійсного цифрового підпису та нестандартного шляху виконуваного файлу.

Дослідження техніки крадіжки маркера доступу продемонструвало операційно значущий вектор загрози, за якого стандартний користувацький процес отримав дескриптор привілейованого системного процесу, дублював власний токен безпеки та породив дочірній процес cmd.exe, що працює з системним рівнем привілеїв. Проведений аналіз ідентифікував два ключові типи подій, що дозволило побудувати алгоритм атаки та створити кореляційні правила для виявлення прихованої загрози.

Таким чином, результати практичної частини підтвердили, що атаки ескалації привілеїв різних категорій створюють якісно різні телеметричні сигнатури, які можна надійно виявити за допомогою спеціально розроблених правил кореляції SIEM. Формалізовані алгоритми атак, розроблені для обох методів, забезпечують багаторазову аналітичну основу для розширення охоплення виявлення на подібні моделі атак в операційних середовищах.

ВИСНОВКИ

У дипломній роботі було проведено дослідження технік підвищення привілеїв та побудовано методи протидії, що дозволило отримати такі результати:

- Аналіз архітектури безпеки Windows та моделі контролю доступу встановив, що захист операційної системи не забезпечується єдиним механізмом, а виникає в результаті узгодженої взаємодії монітора безпеки, LSASS, SAM та пов'язаних підсистем у режимах ядра та користувача. Також було продемонстровано модель контролю доступу, яка гібридною системою та поєднує дискреційні елементи керування, обов'язкові рівні цілісності та перевизначення на основі привілеїв, складність яких створює як архітектурну гнучкість, так і постійний набір невідповідностей, що можуть бути використані.
- Аналіз механізмів управління привілеями програмного забезпечення на рівні API Windows показав, що критично важливі для безпеки функції, зосереджені в бібліотеці `advapi32.dll`, містять слабкі місця. Також було визначено, що ці недоліки є скоріше архітектурними, ніж конфігураційними, тобто їх неможливо вирішити лише за допомогою коригування політик, а натомість створюють постійну поверхню атаки, яку можна компенсувати лише на рівні виявлення.
- Дослідження та порівняння існуючих методів ескалації привілеїв продемонстрували, що поверхня атаки, доступна зловмиснику в середовищі Windows, є широкою та неоднорідною. Також було визначено, що відмінність між вертикальним та горизонтальним напрямками ескалації має операційне значення, оскільки розширені ланцюги атак у корпоративному середовищі поєднують обидва для досягнення повного адміністративного контролю, а це означає, що охоплення виявлення має враховувати як локальні межі привілеїв, так і горизонтальне переміщення облікових даних.

- Розробка методів виявлення на основі аналізу журналу подій безпеки дала конкретні, емпірично підтверджені результати. Правила кореляції були розроблені та протестовані на платформі Wazuh SIEM з використанням телеметрії Sysmon, зібраної з контрольованої кінцевої точки Windows. Усі механізми були перевірені експериментально та додатково були розроблені алгоритми атаки, що забезпечує структуровану аналітичну основу для розширення охоплення виявлення на пов'язані шаблони ескалації привілеїв в операційних середовищах.

Сукупний результат роботи підтверджує, що ескалація привілеїв у середовищах Windows структурно забезпечується обмеженнями, властивими власній системі аудиту та контролю доступу, проте залишається надійно виявленою за допомогою поведінкової телеметрії, коли платформа SIEM належним чином налаштована з моніторингом кінцевих точок. Дослідження має практичне застосування, оскільки розроблені правила виявлення можуть бути розгорнуті в реальній інфраструктурі моніторингу безпеки, тоді як аналітична структура, створена для досліджуваних методів, забезпечує методологію багаторазового використання для розробки охоплення виявлення в ширшому спектрі методів ескалації привілеїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Bassil Y. Windows and Linux Operating Systems from a Security Perspective. arXiv. URL: <https://arxiv.org/pdf/1204.0197>
2. Russinovich M., Solomon D., Ionescu A. Windows Internals Part 1, 6th Edition. URL: https://repo.zenk-security.com/Linux%20et%20systemes%20d.exploitations/Windows%20Internals%20Part%201_6th%20Edition.pdf
3. Security Principals. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-principals>
4. Local Security Authority Subsystem Service. Grokipedia. URL: https://grokipedia.com/page/Local_Security_Authority_Subsystem_Service
5. Windows Authentication Architecture. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-architecture>
6. What Is a Security Account Manager (SAM) Database? JumpCloud. URL: <https://jumpcloud.com/it-index/what-is-a-security-account-manager-sam-database>
7. Windows Kernel-Mode Security Reference Monitor. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-security-reference-monitor>
8. Reference Monitor. Grokipedia. URL: https://grokipedia.com/page/Reference_monitor
9. Security Identifier. Grokipedia. URL: https://grokipedia.com/page/Security_Identifier
10. Security Identifiers. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers>
11. Access Tokens. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows/win32/secauthz/access-tokens>

12. Снігур , А., & Гальчинський , Л. (2024). Виявлення персистенності загроз типу АРТ в системах Windows. Collection of Scientific Papers «SCIENTIA», (November 22, 2024; Athens, Greece), 157–165.

Retrieved from <https://previous.scientia.report/index.php/archive/article/view/2231>

13. Windows Security Internals: A Deep Dive into Windows Authentication.

URL:

https://ia801904.us.archive.org/6/items/windows-security-internals-a-deep-dive-into-windows-authentication/Windows_Security_Internals_A_Deep_Dive_into_Windows_Authentication.pdf

14. What Is Discretionary Access Control (DAC)? TechPrescient. URL:

<https://www.techprescient.com/blogs/discretionary-access-control-dac/>

15. Gallagher K. et al. Assessment of Access Control Systems. NIST. URL:

<https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7316.pdf>

16. Crowell A. Windows Access Control Demystified. University of Maryland.

URL:

https://www.cs.umd.edu/sites/default/files/scholarly_papers/ACrowell_1.pdf

17. Access Control Lists. Microsoft Learn. URL:

<https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-lists>

18. Security Descriptors. Microsoft Learn. URL:

<https://learn.microsoft.com/en-us/windows/win32/secauthz/security-descriptors>

19. Security Descriptor. Microsoft Open Specifications. URL:

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-azod/ec52bde3-9c86-4484-9080-e72148a2d53b

20. Ligh M. et al. The Art of Memory Forensics. URL:

[https://repo.zenk-security.com/Forensic/The%20Art%20of%20Memory%20Forensics%20-%20Detecting%20Malware%20and%20Threats%20in%20Windows,%20Linux,%20and%20Mac%20Memory%20\(2014\).pdf](https://repo.zenk-security.com/Forensic/The%20Art%20of%20Memory%20Forensics%20-%20Detecting%20Malware%20and%20Threats%20in%20Windows,%20Linux,%20and%20Mac%20Memory%20(2014).pdf)

21. What Is Privilege Escalation? IBM. URL:

<https://www.ibm.com/think/topics/privilege-escalation>

22. Cybersecurity – Attack and Defense Strategies. URL:

<https://www.tsoungui.fr/ebooks/CYBER-Security.pdf>

23. Privilege Escalation on Windows. Delinea. URL:
<https://delinea.com/blog/windows-privilege-escalation>
24. Privilege Escalation, Tactic TA0004. MITRE ATT&CK. URL:
<https://attack.mitre.org/tactics/TA0004/>
25. Гальчинський Л. Ю., Ващаєв Т.В. Систематизація технік privilege escalation та методологічні підходи до їх виявлення в операційних системах windows: матеріали Міжнар. наук.-практ. конф, м. Чикаго, 2026.
<https://sci-conf.com.ua/wp-content/uploads/2026/05/INTERNATIONAL-EXPERIENCE-IN-SCIENTIFIC-RESEARCH-14-16.05.26.pdf>
26. What Is Security Event Management (SEM)? Palo Alto Networks. URL:
<https://www.paloaltonetworks.com/cyberpedia/what-is-security-event-management-sem>
27. How Attackers Abuse Access Token Manipulation (ATT&CK T1134). Elastic. URL:
<https://www.elastic.co/blog/how-attackers-abuse-access-token-manipulation>
28. What Is SIEM Architecture? NetWitness. URL:
<https://www.netwitness.com/cyber-glossary/siem-architecture/>
29. Data Normalization Explained. Splunk. URL:
https://www.splunk.com/en_us/blog/learn/data-normalization.html
30. Data Aggregation: How It Works. Splunk. URL:
https://www.splunk.com/en_us/blog/learn/data-aggregation.html
31. SIEM Correlation Rules. Cymulate. URL:
<https://cymulate.com/cybersecurity-glossary/siem-correlation-rules/>

Додаток А

```
#include <windows.h>
#include <fstream>
#include <iostream>

typedef const char* (*GetTextFunc)();

int main()
{
    const char* text = "A dynamic-link library (DLL) is a shared library in the
Microsoft Windows or OS/2 operating system. A DLL can contain executable
code (functions), data, and resources.";

    HMODULE hDll = LoadLibraryA("MyDll.dll");

    if (hDll)
    {
        GetTextFunc GetText = (GetTextFunc)GetProcAddress(hDll, "GetText");

        if (GetText)
        {
            text = GetText();
        }
    }

    std::ofstream file("output.txt");

    if (file.is_open())
    {
        file << text;
```

```
        file.close();  
    }  
  
    if (hDll)  
    {  
        FreeLibrary(hDll);  
    }  
  
    return 0;  
}
```

Додаток Б

```
#include <windows.h>
#include <fstream>
#include <string>
#include "pch.h"

using namespace std;

bool IsElevated()
{
    HANDLE token;
    if (OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &token))
    {
        TOKEN_ELEVATION elevation;
        DWORD size;

        GetTokenInformation(token, TokenElevation, &elevation,
sizeof(elevation), &size);

        CloseHandle(token);

        return elevation.TokenIsElevated;
    }
    return false;
}

const char* GetIntegrityLevelName()
{
    HANDLE hToken = NULL;
```

```

if (OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &hToken))
{
    DWORD size = 0;

    if (!GetTokenInformation(hToken, TokenIntegrityLevel, NULL, 0, &size)
    && GetLastError() == ERROR_INSUFFICIENT_BUFFER)
    {
        PTOKEN_MANDATORY_LABEL pTIL =
(PTOKEN_MANDATORY_LABEL)LocalAlloc(LPTR, size);
        if (pTIL != NULL)
        {
            if (GetTokenInformation(hToken, TokenIntegrityLevel, pTIL, size,
&size))
            {
                UCHAR subAuthorityCount =
*GetSidSubAuthorityCount(pTIL->Label.Sid);
                DWORD dwIntegrityLevel =
*GetSidSubAuthority(pTIL->Label.Sid, subAuthorityCount - 1);

                const char* result = "Unknown";
                if (dwIntegrityLevel >=
SECURITY_MANDATORY_SYSTEM_RID)
                    result = "System";
                else if (dwIntegrityLevel >=
SECURITY_MANDATORY_HIGH_RID)
                    result = "High";
                else if (dwIntegrityLevel >=
SECURITY_MANDATORY_MEDIUM_RID)
                    result = "Medium";
            }
        }
    }
}

```

```

        else if (dwIntegrityLevel >=
SECURITY_MANDATORY_LOW_RID)
            result = "Low";

```

```

        LocalFree(pTIL);
        CloseHandle(hToken);

```

```

        return result;
    }
    LocalFree(pTIL);
}
}
CloseHandle(hToken);
}
return "Unknown";
}

```

```

void AdminCheck()
{
    BOOL isAdmin = IsElevated();
    const char* filePath;

    if (isAdmin)
    {
        filePath = "C:\\Windows\\privilege_check.txt";
    }
    else
    {
        char currentDir[MAX_PATH];
        GetCurrentDirectoryA(MAX_PATH, currentDir);
    }
}

```

```

    filePath = "privilege_test.txt";
}

HANDLE hFile = CreateFileA(filePath, GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

if (hFile != INVALID_HANDLE_VALUE)
{
    DWORD writtenData;
    const char* adminText = isAdmin ? "Admin Privileges: Yes\r\n" : "Admin
Privileges: No\r\n";
    const char* integrityLevel = "Integrity Level: ";
    const char* ilText = GetIntegrityLevelName();

    WriteFile(hFile, adminText, (DWORD)strlen(adminText), &writtenData,
NULL);
    WriteFile(hFile, integrityLevel, (DWORD)strlen(integrityLevel),
&writtenData, NULL);
    WriteFile(hFile, ilText, (DWORD)strlen(ilText), &writtenData, NULL);
    CloseHandle(hFile);
}
}

BOOL APIENTRY DllMain( HMODULE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:

```

```
    DisableThreadLibraryCalls(hModule);  
    AdminCheck();  
case DLL_PROCESS_DETACH:  
    break;  
}  
return TRUE;  
}
```

Додаток В

```
#include <windows.h>
```

```
#include <stdio.h>
```

```
#include <tlhelp32.h>
```

```
#include <wchar.h>
```

```
bool EnableDebugPrivilege()
```

```
{
```

```
    HANDLE hToken;
```

```
    TOKEN_PRIVILEGES tp;
```

```
    LUID luid;
```

```
    if (!OpenProcessToken(GetCurrentProcess(),  
TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken)) {  
        return false;  
    }
```

```
    if (!LookupPrivilegeValue(NULL, SE_DEBUG_NAME, &luid)) {  
        CloseHandle(hToken);  
        return false;  
    }
```

```
    tp.PrivilegeCount = 1;
```

```
    tp.Privileges[0].Luid = luid;
```

```
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
```

```
    if (!AdjustTokenPrivileges(hToken, FALSE, &tp,  
sizeof(TOKEN_PRIVILEGES), NULL, NULL)) {  
        CloseHandle(hToken);  
        return false;  
    }
```

```

    }

    CloseHandle(hToken);
    return (GetLastError() != ERROR_NOT_ALL_ASSIGNED);
}

DWORD FindSystemProcessID(const wchar_t* processName)
{
    PROCESSENTRY32W pe32;
    pe32.dwSize = sizeof(PROCESSENTRY32W);

    HANDLE hSnapshot =
    CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (hSnapshot == INVALID_HANDLE_VALUE)
        return 0;

    if (Process32FirstW(hSnapshot, &pe32))
    {
        do
        {
            if (_wcsicmp(pe32.szExeFile, processName) == 0)
            {
                DWORD pid = pe32.th32ProcessID;
                CloseHandle(hSnapshot);
                return pid;
            }
        } while (Process32NextW(hSnapshot, &pe32));
    }

    CloseHandle(hSnapshot);
}

```



```
        NULL,  
        &si,  
        &pi  
    );  
    CloseHandle(hDupToken);  
}  
CloseHandle(hToken);  
}  
CloseHandle(hProcess);  
}  
}  
}
```

Додаток Г

```
<group name="local, dll_hijacking,">
  <rule id="101101" level="10">
    <if_group>sysmon_event7</if_group>
    <field name="win.eventdata.imageLoaded"
type="pcre2">.*\\Users\\.*.dll</field>
    <field name="win.eventdata.signed">>false</field>
    <description>Potential DLL Hijacking: Unsigned DLL loaded.</description>
    <mitre>
      <id>T1574.0001</id>
    </mitre>
  </rule>
</group>
```

```
<group name="local, token_stealing,">
  <rule id="101122" level="8">
    <if_group>windows</if_group>
    <field name="win.system.eventID" type="pcre2">^10$</field>
    <field name="win.eventdata.targetImage"
type="pcre2">(?!i)\\+(winlogon|lsass|services|spoolsv|suchost.exe$</field>
    <field name="win.eventdata.grantedAccess"
type="pcre2">^0x1400$1^0x1000$</field>
    <field name="win.eventdata.sourceUser" negate="yes" type="pcre2">^NT
AUTHORITY (SYSTEM|LOCAL SERVICE|NETWORK SERVICE)$</field>
    <description>TokenTheft: A user process has gained unusual access to a
system process</description>
  </rule>
```

```
<rule id="101133" level="15" timeframe="5">
  <if_group>windows</if_group>
```

```

<if_matched_sid>101122</if_matched_sid>
<field name="win.system.event ID" type="pcre2">^1$</field>
<field name="win.eventdata.user" type="pcre2">(?i)^NT
AUTHORITY+SYSTEM$</field>
<field name="win.eventdata.parentUser" negate="yes" type="pcre2">^NT
AUTHORITY (SYSTEM|LOCAL SERVICE |NETWORK
SERVICE)$</field>
<same_field>win.system.computer</same_field>
<description>Possible Privilege Escalation: Token
Impersonation</description>
<mitre>
  <id>T1134.001</id>
</mitre>
</rule>
</group>

```