

**Кириченко Олег Вадимович**

студент кафедри інформаційної безпеки та наноелектроніки

Національний університет «Запорізька політехніка», Запоріжжя, Україна

ORCID: 0009-0003-6373-7069

kir.oleg2004@gmail.com

Корольков Роман Юрійович

кандидат технічних наук, доцент,

доцент кафедри інформаційної безпеки та наноелектроніки

Національний університет «Запорізька політехніка», Запоріжжя, Україна

ORCID: 0000-0001-5501-4600

romankor@zp.edu.ua

**ДОСЛІДЖЕННЯ РЕАКЦІЇ МЕХАНІЗМІВ ЗАХИСТУ WINDOWS НА ФАЙЛИ СЦЕНАРІЇВ,
СФОРМОВАНІ ЗАСОБАМИ BLOB API**

Анотація. Сучасні клієнтські вебтехнології дають змогу формувати файли безпосередньо у веббраузері без завантаження готового файла з вебсервера, що актуалізує питання їх подальшої обробки механізмами захисту операційної системи. Метою роботи є експериментальне дослідження метаданих походження сценарного файла, сформованого засобами Blob API, та реакції механізмів захисту Windows на досліджуваних етапах його обробки й виконання. Реалізовано двоетапний експериментальний ланцюжок: HTML-сторінка за допомогою JavaScript, Uint8Array та Blob API формувала .bat-файл у веббраузері, а виконання цього файла ініціювало отримання PowerShell-сценарію засобами curl.exe та його запуск через powershell.exe. У PowerShell-сценарії застосовано обфускацію частини рядкових даних за допомогою XOR та Base64. Експеримент проведено в середовищі Windows 11 24H2 з увімкненими Microsoft Defender Antivirus, Microsoft Defender SmartScreen та Antimalware Scan Interface. Для перевірки наявності Mark of the Web досліджували його представлення у вигляді альтернативного потоку даних NTFS Zone.Identifier. Контрольним варіантом слугував .bat-файл, безпосередньо завантажений із вебресурсу тим самим браузером. У дослідженій конфігурації обидва файли містили потік Zone.Identifier зі значенням ZoneId=3. Водночас продемонстровано, що застосування PowerShell-команди Unblock-File забезпечило видалення потоку Zone.Identifier; під час повторного запуску файла попередження про неможливість перевірити видавця більше не спостерігалося. Експериментальний ланцюжок продовжився отриманням і запуском PowerShell-сценарію та встановленням C2-взаємодії, характерної для сценаріїв віддаленого керування шкідливим програмним забезпеченням. Виконання контрольних команд і передавання результатів відбувалися за активних Microsoft Defender Antivirus та AMSI, а також чотирьох увімкнених правил зменшення поверхні атаки (ASR) у режимі блокування. Практичне значення роботи полягає в обґрунтуванні необхідності комплексного аналізу ланцюжка формування та виконання сценарного коду з незалежним оцінюванням метаданих походження файла та реакції механізмів захисту на різних етапах його обробки.

Ключові слова: Blob API; Mark of the Web; файли сценаріїв; PowerShell; Microsoft Defender; AMSI; обфускація коду; веббезпека.

ВСТУП

Постановка проблеми. Сучасні операційні системи використовують багаторівневі механізми захисту від потенційно небезпечного сценарного коду. У Windows до складових такого багаторівневого захисту належать Microsoft Defender Antivirus, Microsoft Defender SmartScreen та Antimalware Scan Interface (AMSI), тоді як Mark of the Web (MOTW) використовується для збереження відомостей про походження файлів із зовнішніх джерел. Водночас клієнтські вебтехнології, зокрема Blob API, дають змогу формувати файл безпосередньо у веббраузері, змінюючи традиційну модель його отримання: з вебсервера передається HTML- та JavaScript-код, тоді як кінцевий файл формується на стороні клієнта. За такого способу



формування актуальним є дослідження не лише збереження відомостей про походження файла, а й можливості їх програмної модифікації та реакції механізмів захисту Windows на наступних етапах його обробки. Практичний інтерес становлять .bat-файли, які можуть використовуватися для ініціювання отримання та виконання PowerShell-сценаріїв, зокрема шкідливих. Тому дослідження такого способу формування сценарних файлів потребує комплексного аналізу всього ланцюжка – від формування файлу на стороні клієнта та фіксації відомостей про його походження до подальшого виконання й реакції механізмів захисту на окремих етапах його обробки.

Аналіз останніх досліджень і публікацій. Одним зі способів формування та доставки шкідливого вмісту засобами вебтехнологій є техніка HTML Smuggling. К. Kaushik та співавтори [1] проаналізували способи реалізації цієї техніки та протидії їй, а J. R. Dora та L. Hluchý [2] – питання безпеки виконання коду на стороні клієнта із застосуванням цієї техніки.

Для виявлення шкідливих PowerShell-сценаріїв запропоновано механізм MPSD, який використовує AMSI для отримання сценарного вмісту та VirusTotal для його перевірки [3], а також метод Power-ASTNN, що поєднує деобфускацію з нейромережним аналізом абстрактного синтаксичного дерева (AST) [4]. У дослідженні ScriptBlock Smuggling [5] описано ухилення від перевірки через AMSI та підміну даних журналювання шляхом створення розбіжності між перевірюваним і фактично виконуваним кодом. Для виявлення шкідливих PowerShell-команд і сценаріїв також запропоновано підходи, що базуються на глибоких нейронних мережах [6], контекстних представленнях даних AMSI [7], гібридних ознаках [8], оптимізації ознак на основі токенів і AST [9] та моделі з багатошаровим шумозаглушувальним автокодером [10].

Приховування шкідливого сценарного коду ускладнює його аналіз. Для протидії досліджено поєднання деобфускації із семантично орієнтованим виявленням атак [11], відновлення коду на основі AST зі збереженням семантики [12], багатоступеневу деобфускацію засобами PowerDrive [13] та виявлення шкідливого PowerShell-коду за умов змішування й обфускації з використанням архітектури Sentence Transformer і навчання на основі подібності [14]. Використання вбудованих засобів операційної системи в безфайлових атаках узагальнено в огляді [15]. У MITRE ATT&CK HTML Smuggling класифіковано як підтехніку T1027.006, одним зі способів реалізації якої є використання JavaScript Blob [16]. Згідно зі стратегією DET0313, виявлення цієї техніки пов'язують із подіями створення й виконання файлів та аналізом Zone.Identifier, зокрема HostUrl і ReferrerUrl [17]. Практичне значення цієї техніки підкреслює Microsoft, описуючи її застосування у реальних кампаніях доставки шкідливого програмного забезпечення з формуванням вмісту на кінцевому пристрої [18].

Зазначені дослідження обґрунтовують аналіз загроз на послідовних етапах доставки й виконання сценарного коду. У цій роботі розглянуто експериментальний ланцюжок, що моделює отримання та виконання сценарію дистанційного керування, характерного для шкідливого програмного забезпечення. Дослідження охоплює порівняння метаданих походження сформованого засобами Blob API та безпосередньо завантаженого .bat-файлів, а також фіксацію системних попереджень і реакції захисту під час подальшого виконання файлу, сформованого засобами Blob API.

Мета статті. Метою роботи є експериментальне дослідження метаданих походження сценарного файлу, сформованого засобами Blob API, та реакції механізмів захисту Windows під час його обробки й виконання в ланцюжку, що моделює отримання та запуск сценарію дистанційного керування, характерного для шкідливого програмного забезпечення. Дослідження охоплює порівняння метаданих походження сформованого та безпосередньо завантаженого файлів, фіксацію системних попереджень і наявності або відсутності блокування, перевірку стану Zone.Identifier після застосування Unblock-File та наскрізному спостереженні реакції механізмів захисту Windows на послідовних етапах виконання PowerShell-сценарію до встановлення C2-взаємодії.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Формування файлів на стороні клієнта засобами Blob API

Беззастосунки можуть формувати файли безпосередньо на стороні клієнта засобами JavaScript. Blob API дає змогу представити послідовність байтів у вигляді об'єкта Blob, створити для нього Blob URL та ініціювати збереження сформованого вмісту засобами браузера [19]. За такого підходу вебсервер передає HTML- та JavaScript-код, тоді як кінцевий файл формується на стороні клієнта.

Клієнське формування файлів є одним із механізмів реалізації техніки HTML Smuggling, яку в базі знань MITRE ATT&CK класифіковано як підтехніку T1027.006 [16].

Принциповою особливістю такого способу є розмежування мережевого передавання вебконтенту та локального формування кінцевого файлу. У цій роботі Blob API розглядається не як засіб обходу



механізмів захисту, а як засіб формування сценарного файлу на початковому етапі досліджуваного ланцюжка.

Mark of the Web та Zone.Identifier

Mark of the Web використовується у Windows для збереження відомостей про походження файлів із зовнішніх джерел; у файловій системі NTFS відповідні відомості можуть зберігатися в альтернативному потоці даних Zone.Identifier [20, 21].

Відповідно до специфікації Microsoft, потік Zone.Identifier може містити секцію [ZoneTransfer] із параметром Zoned; значення Zoned=3 відповідає зоні Internet [21]. Такі метадані характеризують походження файлу і можуть ураховуватися механізмами Windows під час його подальшої обробки. Водночас наявність таких метаданих сама по собі не свідчить про шкідливість файлу.

Microsoft Defender SmartScreen і Microsoft Defender Antivirus

Microsoft Defender SmartScreen і Microsoft Defender Antivirus (MDA) реалізують різні механізми захисту Windows. SmartScreen перевіряє вебресурси та завантажені файли для виявлення потенційних загроз, використовуючи, зокрема, відомості про їхню репутацію [22]. Водночас попередження SmartScreen не є безпосереднім підтвердженням наявності шкідливого коду. Залежно від результату репутаційної перевірки SmartScreen може відображати користувачеві попередження щодо запуску файлу [22].

Microsoft Defender Antivirus забезпечує антивірусний захист у реальному часі та використовує, зокрема, поведінковий і хмарний аналіз для виявлення загроз [23]. Відсутність блокування під час репутаційної перевірки не виключає подальшого аналізу файлу або пов'язаних із його виконанням процесів засобами MDA.

Antimalware Scan Interface та контроль PowerShell-сценаріїв

AMSI є стандартним інтерфейсом Windows, який забезпечує взаємодію застосунків і служб із засобами антивірусного захисту. Через цей інтерфейс застосунок може передавати вміст для аналізу антивірусному засобу, інтегрованому з AMSI. У Windows PowerShell така інтеграція забезпечує перевірку сценарного вмісту перед його виконанням [24]. Отже, відсутність блокування початкового .bat-файлу не виключає подальшої антивірусної перевірки PowerShell-сценарію через AMSI.

Від антивірусної перевірки через AMSI слід відрізнити дію політики виконання PowerShell (Execution Policy), яка регулює умови запуску сценаріїв. Параметр -ExecutionPolicy Bypass задає для сеансу політику Bypass, якщо її не перевизначено груповою політикою. За її дії засоби Execution Policy не блокують виконання сценаріїв і не відображають відповідних попереджень або запитів; водночас Microsoft не визначає Execution Policy як межу безпеки [25]. У цьому дослідженні зазначений параметр характеризує спосіб запуску PowerShell-сценарію, а його застосування не означає вимкнення AMSI або Microsoft Defender Antivirus.

МЕТОДИКА ДОСЛІДЖЕННЯ

Для експериментального дослідження сформовано двоетапний ланцюжок формування та виконання сценарного коду. На першому етапі .bat-файл формувався у веббраузері засобами HTML і JavaScript. Для вмісту файлу формувався об'єкт Uint8Array, елементи якого заповнювалися числовими значеннями символів за допомогою методу charCodeAt(); на основі отриманого масиву створювався об'єкт Blob із MIME-типом application/octet-stream. Для створеного об'єкта Blob генерувався Blob URL за допомогою методу URL.createObjectURL(), після чого програмно ініціювалося збереження файлу засобами браузера [19]. Таким чином, вебресурс передавав HTML- та JavaScript-код, тоді як кінцевий .bat-файл формувався на стороні клієнта. Фрагмент реалізації наведено в лістингу 1 (ініціалізацію змінної content у наведеному фрагменті опущено).

Лістинг 1

Фрагмент JavaScript-коду формування файлу засобами Blob API

```
// content містить попередньо заданий текст досліджуваного .bat-файла.
var bytes = new Uint8Array(content.length);
for (var i = 0; i < content.length; i++) {
    bytes[i] = content.charCodeAt(i);
}
var blob = new Blob([bytes], {
    type: 'application/octet-stream'
});
var a = document.createElement('a');
```

```
a.href = URL.createObjectURL(blob);
a.download = 'document.bat';
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
```

Вміст змінної content включав логіку отримання та запуску PowerShell-сценарію, а також виклик Unblock-File для видалення потоку Zone.Identifier поточного .bat-файла після його запуску. Наявність потоку Zone.Identifier до запуску та його стан після виконання цієї операції фіксувалися як окремі контрольні ознаки.

На другому етапі збережений .bat-файл виконувався через cmd.exe, що запускало ланцюжок команд: спочатку отримання PowerShell-сценарію засобами curl.exe, а потім його виконання у середовищі Windows PowerShell 5.1. Частина рядкових даних сценарію була обфускована із застосуванням XOR і Base64. PowerShell-сценарій запускався з параметрами -NoProfile -ExecutionPolicy Bypass -File за активних Microsoft Defender Antivirus та AMSI.

Загальну структуру експериментального ланцюжка та контрольні точки наведено на рис. 1.

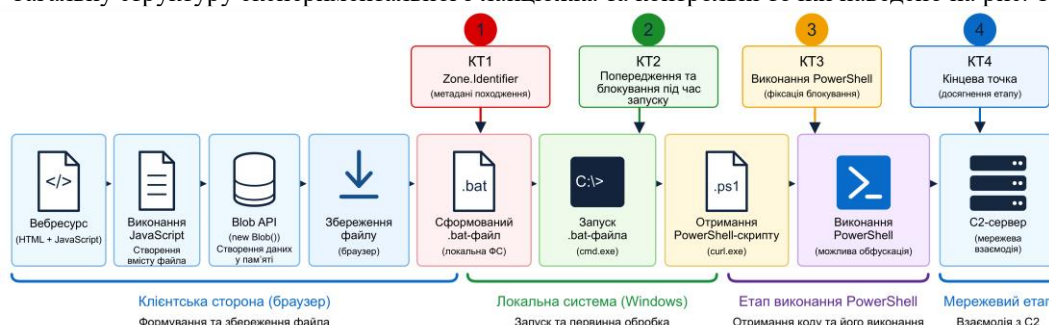


Рис. 1. Схема експериментального ланцюжка та контрольні точки дослідження

У контрольних точках (КТ) фіксувалися: наявність Zone.Identifier після збереження .bat-файла; системні попередження та випадки блокування під час збереження й запуску файла; результат програмного видалення Zone.Identifier; отримання й виконання PowerShell-сценарію за активних Microsoft Defender Antivirus та AMSI; досягнення передбаченої експериментом кінцевої контрольної точки C2-взаємодії.

Експеримент проводився в середовищі Windows 11 24H2; досліджувані файли зберігалися на диску з файловою системою NTFS. Формування експериментального та завантаження контрольного файла виконувалися в Microsoft Edge. Для другого етапу використовувалися Windows PowerShell 5.1 та curl.exe. У конфігурації Microsoft Defender Antivirus було ввімкнено чотири правила зменшення поверхні атаки (Attack Surface Reduction, ASR) у режимі блокування. Конфігурацію програмного середовища і стан механізмів захисту наведено в табл. 1.

Таблиця 1

Конфігурація експериментального середовища

Параметр	Зафіксоване значення
Операційна система	Windows 11, версія 24H2, збірка 26100.9457, x64
Файлова система	NTFS
Веббраузер	Microsoft Edge 153.0.4234.32
Windows PowerShell	5.1.26100.9444
curl.exe	8.21.0
Microsoft Defender Antivirus	4.18.26080.4; активний
Antimalware Engine	1.1.26080.3
Security Intelligence	1.459.283.0
Останнє оновлення Security Intelligence	19.09.2026 03:43:12
RealTimeProtectionEnabled	True
BehaviorMonitorEnabled	True
IoavProtectionEnabled	True



OnAccessProtectionEnabled	True
Microsoft Defender SmartScreen (Explorer)	SmartScreenEnabled=Warn
Хмарний захист	MAPSReporting=2 (Advanced)
Автоматичне надсилання зразків	SubmitSamplesConsent=1 (Send safe samples automatically)
Правила ASR	4 правила ASR у режимі Block (1): Executable prevalence/age/trust; LSASS credential theft; PsExec/WMI process creation; Office child processes.

Для контрольного порівняння з вебресурсу тим самим браузером завантажувався аналогічний .bat-файл з подальшою фіксацією наявності Zone.Identifier в обох випадках.

Порядок проведення експерименту та критерії фіксації результатів

Алгоритм проведення експериментального дослідження наведено на рис. 2.

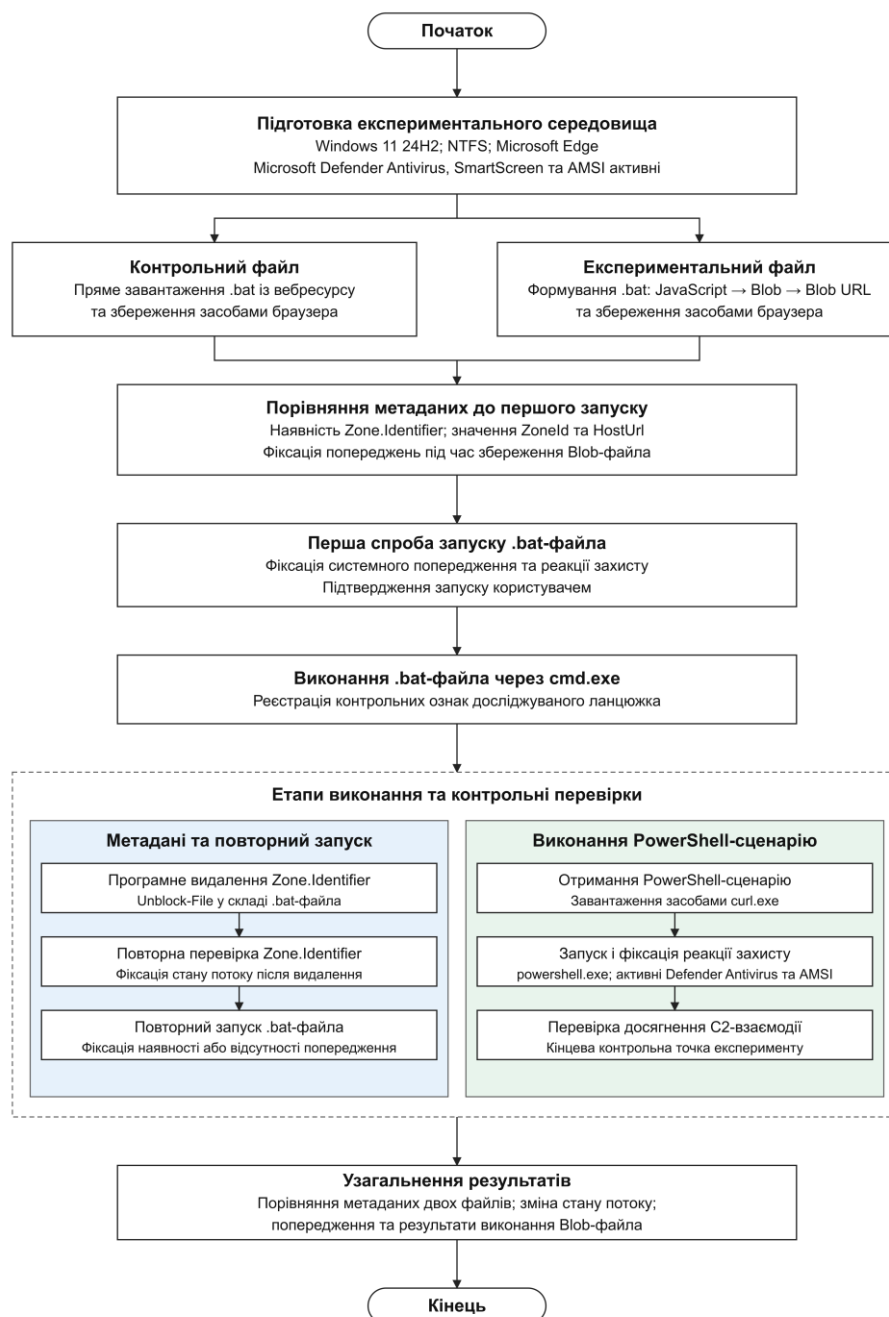


Рис. 2. Алгоритм проведення експериментального дослідження



Для експериментального та контрольного файлів перевірялася наявність альтернативного потоку NTFS Zone.Identifier; у разі його виявлення фіксувалися значення ZoneId та інші доступні відомості. Для файла, сформованого засобами Blob API, окремо фіксувалися наявність системних попереджень або блокування під час збереження та запуску, стан Zone.Identifier після виконання операції його програмного видалення, факт отримання й виконання PowerShell-сценарію та досягнення передбаченої експериментом C2-взаємодії.

Критерієм досягнення кінцевої контрольної точки (КТ4) було отримання через Telegram результатів виконання контрольних команд, надісланих експериментальній системі через Telegram-бота. Для перевірки використовували команди whoami, ipconfig і systeminfo. Фіксували послідовність надсилання команд, отримання файлів із результатами та відповідність їхнього вмісту надісланим командам.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Результати перевірки Zone.Identifier

Перевірку наявності та вмісту потоку Zone.Identifier проводили для обох файлів одразу після їх збереження у файлової системі NTFS. В обох випадках виявлено потік Zone.Identifier зі значенням ZoneId=3. Водночас значення поля HostUrl відрізнялися: для безпосередньо завантаженого файла воно містило повну URL-адресу файла, а для сформованого засобами Blob API – адресу <https://gist.github.com/> без шляху до конкретного ресурсу (рис. 3). Отже, збіг значення ZoneId не супроводжувався тотожністю всього вмісту метаданих походження. Оскільки файли отримували з різних веб-адрес, наведене порівняння не дає змоги ізольовано визначити вплив способу їх формування на значення HostUrl.

```
PS C:\> Write-Host "=== DIRECT: BEFORE EXECUTION ==="
=== DIRECT: BEFORE EXECUTION ===
PS C:\> Get-Item "C:\MOTW-Test\Direct\document.bat" -Stream Zone.Identifier

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\MOTW-Test\Direct\document.bat:Zone.Identifier
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\MOTW-Test\Direct
PSChildName  : document.bat:Zone.Identifier
PSDrive      : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName     : C:\MOTW-Test\Direct\document.bat
Stream       : Zone.Identifier
Length       : 130

PS C:\> Get-Content "C:\MOTW-Test\Direct\document.bat" -Stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
HostUrl=https://gist.githubusercontent.com/[redacted]/raw/document.bat
PS C:\>
PS C:\> Write-Host "`n=== BLOB: BEFORE EXECUTION ==="

=== BLOB: BEFORE EXECUTION ===
PS C:\> Get-Item "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\MOTW-Test\Blob\document.bat:Zone.Identifier
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\MOTW-Test\Blob
PSChildName  : document.bat:Zone.Identifier
PSDrive      : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName     : C:\MOTW-Test\Blob\document.bat
Stream       : Zone.Identifier
Length       : 60

PS C:\> Get-Content "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
HostUrl=https://gist.github.com/
PS C:\> |
```

Рис. 3. Результати перевірки потоку Zone.Identifier для безпосередньо завантаженого та сформованого засобами Blob API .bat-файлів

Реакція механізмів захисту під час запуску .bat-файла

Подальше дослідження виконувалося для .bat-файла, сформованого засобами Blob API. Під час першого запуску .bat-файла було зафіксовано системне попередження безпеки Windows про неможливість перевірити видавця (рис. 4). Після підтвердження запуску користувачем файл було виконано через cmd.exe. Після виконання передбаченої у .bat-файлі операції Unblock-File повторна перевірка показала відсутність альтернативного потоку Zone.Identifier (рис. 5). Під час повторного запуску цього самого файла зазначене системне попередження більше не спостерігалося.

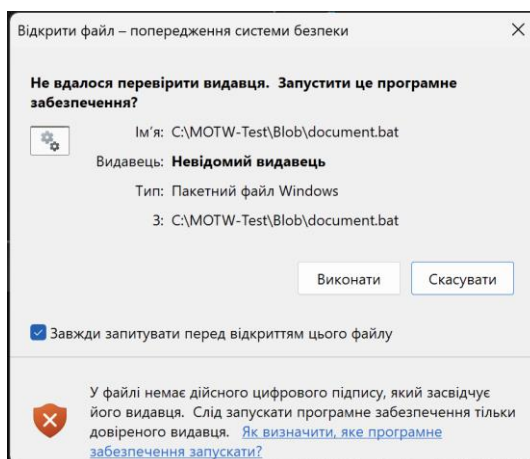


Рис. 4. Системне попередження безпеки Windows під час першого запуску .bat-файла, сформованого засобами Blob API

```
=== BLOB: AFTER EXECUTION ===
PS C:\>
PS C:\> Get-Item "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier
Get-Item : Could not open the alternate data stream 'Zone.Identifier' of the file 'C:\MOTW-Test\Blob\document.bat'.
At line:1 char:1
+ Get-Item "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\MOTW-Test\Blob\document.bat:String) [Get-Item], FileNotFoundException
+ FullyQualifiedErrorId : AlternateDataStreamNotFound,Microsoft.PowerShell.Commands.GetItemCommand

PS C:\>
PS C:\> Get-Content "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier
Get-Content : Could not open the alternate data stream 'Zone.Identifier' of the file 'C:\MOTW-Test\Blob\document.bat'.
At line:1 char:1
+ Get-Content "C:\MOTW-Test\Blob\document.bat" -Stream Zone.Identifier
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\MOTW-Test\Blob\document.bat:String) [Get-Content], FileNotFoundException
+ FullyQualifiedErrorId : GetContentReaderFileNotFoundError,Microsoft.PowerShell.Commands.GetContentCommand

PS C:\> |
```

Рис. 5. Результат повторної перевірки потоку Zone.Identifier після виконання .bat-файла

Результати отримання та виконання PowerShell-сценарію

Під час виконання .bat-файла PowerShell-сценарій було отримано засобами curl.exe та запущено через powershell.exe із параметрами -NoProfile -ExecutionPolicy Bypass -File. Microsoft Defender Antivirus та AMSI залишалися активними.

У дослідженій конфігурації не зафіксовано реакції механізмів захисту, яка б припинила виконання на цьому етапі. Для перевірки досягнення кінцевої контрольної точки C2-взаємодії через Telegram-бота було надіслано контрольні команди whoami, ipconfig і systeminfo. У відповідь отримано текстові файли з результатами, що містили відповідно ім'я облікового запису, параметри мережевої конфігурації та відомості про операційну систему (рис. 6). Отримання результатів, які відповідали надісланим командам, засвідчило двосторонній обмін командами та результатами їх виконання через Telegram у межах експерименту.

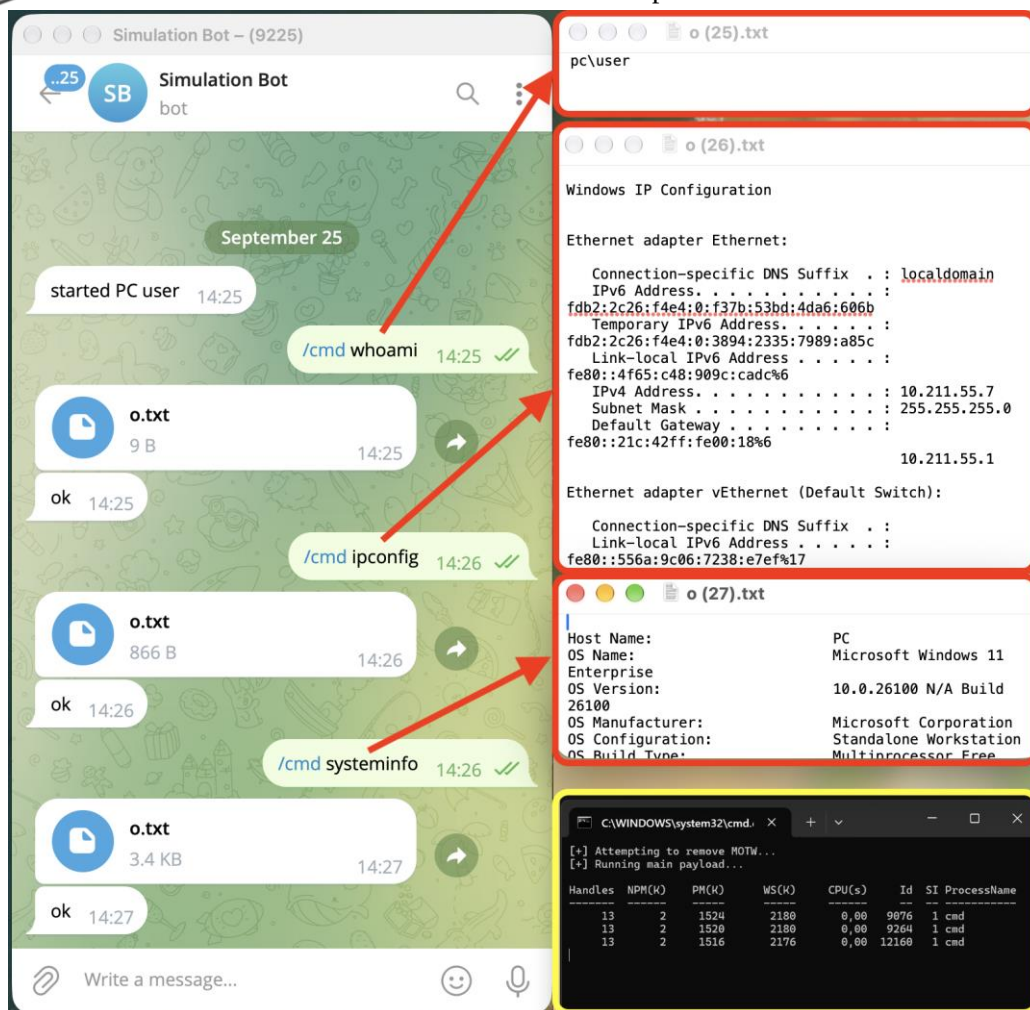


Рис. 6. Обмін контрольними командами та результатами їх виконання через Telegram-бота в межах експерименту

Аналіз та обговорення результатів

Отримані результати показують, що в дослідженій конфігурації Windows 11 24H2 і Microsoft Edge обидва способи отримання .bat-файла супроводжувалися збереженням потоку Zone.Identifier зі значенням ZoneId=3. Отже, формування файла засобами Blob API у цьому експерименті не перешкоджало збереженню метаданих його мережевого походження. Контрольне порівняння охоплювало наявність потоку та значення ZoneId; подальше виконання досліджувалося лише для файла, сформованого засобами Blob API.

Під час першого запуску сформованого файла система відобразила попередження про неможливість перевірити видавця; виконання продовжилось після підтвердження користувачем. Після застосування Unblock-File потік Zone.Identifier більше не виявлявся, а під час повторного запуску зазначене попередження не спостерігалось. Оскільки видалення потоку відбулося після дозволеного користувачем першого запуску, цей результат не свідчить про уникнення початкового попередження.

На наступному етапі зафіксовано отримання й виконання PowerShell-сценарію та двосторонній обмін контрольними командами й результатами через Telegram-бота. У дослідженій конфігурації, що включала чотири правила ASR у режимі блокування, у межах застосованого способу спостереження не зафіксовано блокування, яке припинило б цей ланцюжок.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У роботі експериментально досліджено реакцію механізмів захисту Windows на .bat-файл, сформований у веббраузері засобами Blob API, на послідовних етапах його обробки та виконання.



У дослідженій конфігурації Windows 11 24H2 і Microsoft Edge обидва .bat-файли – сформований засобами Blob API та безпосередньо завантажений контрольний – містили потік Zone.Identifier зі значенням ZoneId=3. Таким чином, було встановлено, що у дослідженій конфігурації формування файла засобами Blob API не перешкоджало збереженню позначки належності до зони Internet.

Після застосування Unblock-File потік Zone.Identifier більше не виявлявся, а під час повторного запуску файла попередження про неможливість перевірити видавця не спостерігалось. Також зафіксовано отримання й виконання PowerShell-сценарію та двосторонню C2-взаємодію через Telegram-бота. У дослідженій конфігурації з увімкненими Microsoft Defender Antivirus, інтеграцією PowerShell з AMSI та чотирма правилами ASR у режимі блокування не зафіксовано блокування, яке припинило б виконання експериментального ланцюжка.

Подальші дослідження доцільно спрямувати на окреме оцінювання впливу способу формування файла, параметрів запуску PowerShell та обфускації сценарного коду на реакцію механізмів захисту. Перспективним є розширення експериментального середовища засобами AppLocker, Windows Defender Application Control та EDR-рішеннями для порівняння їхньої реакції на досліджувані ланцюжки у корпоративних конфігураціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kaushik, K., Sharma, G., Narooka, P., Chhabra, G., & Vishnoi, A. (2023). HTML smuggling: Attack and mitigation. In *2023 16th International Conference on Security of Information and Networks (SIN)* (pp. 1–5). IEEE. <https://doi.org/10.1109/SIN60469.2023.10474837>
2. Dora, J. R., & Hluchý, L. (2025). Security review of a client-side code execution with HTML smuggling attack. In *2025 IEEE 12th International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)* (pp. 131–136). IEEE. <https://doi.org/10.1109/ICCC64928.2025.10999118>
3. Wu, M.-H., Hsu, F.-H., Hunag, J.-H., Wang, K., Liu, Y.-Y., Chen, J.-X., Wang, H.-J., & Yang, H.-T. (2024). MPSD: A robust defense mechanism against malicious PowerShell scripts in Windows systems. *Electronics*, 13(18), Article 3717. <https://doi.org/10.3390/electronics13183717>
4. Zhang, S., Li, S., Lu, J., & Yang, W. (2025). Power-ASTNN: A deobfuscation and AST neural network enabled effective detection method for malicious PowerShell scripts. *Computers & Security*, 154, Article 104441. <https://doi.org/10.1016/j.cose.2025.104441>
5. Rose, A. J., Graham, S. R., Schubert Kabban, C. M., Krasnov, J. J., & Henry, W. C. (2024). ScriptBlock smuggling: Uncovering stealthy evasion techniques in PowerShell and .NET environments. *Journal of Cybersecurity and Privacy*, 4(2), 153–166. <https://doi.org/10.3390/jcp4020008>
6. Hendler, D., Kels, S., & Rubin, A. (2018). Detecting malicious PowerShell commands using deep neural networks. In *Proceedings of the 2018 Asia Conference on Computer and Communications Security* (pp. 187–197). Association for Computing Machinery. <https://doi.org/10.1145/3196494.3196511>
7. Hendler, D., Kels, S., & Rubin, A. (2020). AMSI-based detection of malicious PowerShell code using contextual embeddings. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20)* (pp. 679–693). Association for Computing Machinery. <https://doi.org/10.1145/3320269.3384742>
8. Fang, Y., Zhou, X., & Huang, C. (2021). Effective method for detecting malicious PowerShell scripts based on hybrid features. *Neurocomputing*, 448, 30–39. <https://doi.org/10.1016/j.neucom.2021.03.117>
9. Song, J., Kim, J., Choi, S., Kim, J., & Kim, I. (2021). Evaluations of AI-based malicious PowerShell detection with feature optimizations. *ETRI Journal*, 43(3), 549–560. <https://doi.org/10.4218/etrij.2020-0215>
10. Alahmadi, A., Alkhraan, N., & BinSaeedan, W. (2022). MPSAutodetect: A malicious PowerShell script detection model based on stacked denoising auto-encoder. *Computers & Security*, 116, Article 102658. <https://doi.org/10.1016/j.cose.2022.102658>
11. Xiong, C., Li, Z., Chen, Y., Zhu, T., Wang, J., Yang, H., & Ruan, W. (2022). Generic, efficient, and effective deobfuscation and semantic-aware attack detection for PowerShell scripts. *Frontiers of Information Technology & Electronic Engineering*, 23(3), 361–381. <https://doi.org/10.1631/FITEE.2000436>
12. Chai, H., Ying, L., Duan, H., & Zha, D. (2022). Invoke-Deobfuscation: AST-based and semantics-preserving deobfuscation for PowerShell scripts. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 295–306). IEEE. <https://doi.org/10.1109/DSN53405.2022.00039>
13. Ugarte, D., Maiorca, D., Cara, F., & Giacinto, G. (2019). PowerDrive: Accurate de-obfuscation and analysis of PowerShell malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th*



- International Conference, DIMVA 2019* (pp. 240–259). Springer. https://doi.org/10.1007/978-3-030-22038-9_12
14. Fu, Z., Song, L., Ding, S. H. H., Alaca, F., & Acharya, S. (2025). 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*, 28(4), Article 55, 1–23. <https://doi.org/10.1145/3771542>
 15. Sudhakar, & Kumar, S. (2020). An emerging threat fileless malware: A survey and research challenges. *Cybersecurity*, 3, Article 1. <https://doi.org/10.1186/s42400-019-0043-x>
 16. MITRE. (n.d.). *Obfuscated files or information: HTML smuggling (T1027.006)*. MITRE ATT&CK. Retrieved September 19, 2026, from <https://attack.mitre.org/techniques/T1027/006/>
 17. MITRE. (n.d.). *Detection strategy for HTML smuggling via JavaScript Blob + dynamic file drop (DET0313)*. MITRE ATT&CK. Retrieved September 19, 2026, from <https://attack.mitre.org/detectionstrategies/DET0313/>
 18. Microsoft Threat Intelligence. (2021, November 11). *HTML smuggling surges: Highly evasive loader technique increasingly used in banking malware, targeted attacks*. Microsoft Security Blog. <https://www.microsoft.com/en-us/security/blog/2021/11/11/html-smuggling-surges-highly-evasive-loader-technique-increasingly-used-in-banking-malware-targeted-attacks>
 19. World Wide Web Consortium. (2026, September 12). *File API* (W3C Working Draft). <https://www.w3.org/TR/2026/WD-FileAPI-20260912/>
 20. Byeon, H. (2026). Trust propagation failure in the recursive-execution path of nested archives. *International Journal of Safety and Security Engineering*, 16(3), 503–508. <https://doi.org/10.18280/ijssse.160303>
 21. Microsoft. (n.d.). *[MS-FSCC]: Zone.Identifier stream name*. Microsoft Learn. Retrieved September 19, 2026, from https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/6e3f7352-d11c-4d76-8c39-2516a9df36e8
 22. Microsoft. (n.d.). *Microsoft Defender SmartScreen overview*. Microsoft Learn. Retrieved September 19, 2026, from <https://learn.microsoft.com/en-us/windows/security/operating-system-security/virus-and-threat-protection/microsoft-defender-smartscreen/>
 23. Microsoft. (n.d.). *Microsoft Defender Antivirus in Windows overview*. Microsoft Learn. Retrieved September 20, 2026, from <https://learn.microsoft.com/en-us/defender-endpoint/microsoft-defender-antivirus-windows>
 24. Microsoft. (n.d.). *Antimalware Scan Interface (AMSI)*. Microsoft Learn. Retrieved September 19, 2026, from <https://learn.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal>
 25. Microsoft. (n.d.). *About execution policies*. Microsoft Learn. Retrieved September 20, 2026, from https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-5.1

**Oleh V. Kyrychenko**

Student, Department of Information Security and Nanoelectronics
Zaporizhzhia Polytechnic National University, Zaporizhzhia, Ukraine
ORCID: 0009-0003-6373-7069
kir.oleg2004@gmail.com

Roman Yu. Korolkov

PhD in Engineering, Associate Professor,
Associate Professor of the Department of Information Security and Nanoelectronics
Zaporizhzhia Polytechnic National University, Zaporizhzhia, Ukraine
ORCID: 0000-0001-5501-4600
romankor@zp.edu.ua

**RESEARCH ON THE RESPONSE OF WINDOWS SECURITY MECHANISMS TO SCRIPT FILES
CREATED USING THE BLOB API**

Abstract. Modern client-side web technologies enable files to be generated directly in a web browser without downloading a complete file from a web server, raising questions about their subsequent handling by operating system security mechanisms. This study aims to experimentally investigate the origin metadata of a script file generated using the Blob API and the response of Windows security mechanisms at the examined stages of its processing and execution. A two-stage experimental chain was implemented: an HTML page used JavaScript, Uint8Array, and the Blob API to generate a .bat file in the browser, and execution of this file initiated the retrieval of a PowerShell script using curl.exe and its execution through powershell.exe. Some string data in the PowerShell script were obfuscated using XOR and Base64. The experiment was conducted in a Windows 11 24H2 environment with Microsoft Defender Antivirus, Microsoft Defender SmartScreen, and the Antimalware Scan Interface (AMSI) enabled. The presence of Mark of the Web was assessed by examining its representation as the NTFS Zone.Identifier alternate data stream. A .bat file downloaded directly from a web resource using the same browser served as the control. In the tested configuration, both files contained a Zone.Identifier stream with ZoneId=3. The experiment also demonstrated that the PowerShell Unblock-File cmdlet removed the Zone.Identifier stream; the warning that the publisher could not be verified was no longer observed when the file was launched again. The experimental chain continued with the retrieval and execution of the PowerShell script and the establishment of command-and-control (C2) communication characteristic of malware remote-control scenarios. Test commands were executed and their results transmitted while Microsoft Defender Antivirus and AMSI remained active and four attack surface reduction (ASR) rules were enabled in block mode. The practical significance of the study lies in demonstrating the need for a comprehensive analysis of the script generation and execution chain, with separate assessment of file-origin metadata and the response of security mechanisms at different processing stages.

Keywords: Blob API; Mark of the Web; script files; PowerShell; Microsoft Defender; AMSI; code obfuscation; web security.

REFERENCES

1. Kaushik, K., Sharma, G., Narooka, P., Chhabra, G., & Vishnoi, A. (2023). HTML smuggling: Attack and mitigation. In *2023 16th International Conference on Security of Information and Networks (SIN)* (pp. 1–5). IEEE. <https://doi.org/10.1109/SIN60469.2023.10474837>
2. Dora, J. R., & Hluchý, L. (2025). Security review of a client-side code execution with HTML smuggling attack. In *2025 IEEE 12th International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)* (pp. 131–136). IEEE. <https://doi.org/10.1109/ICCC64928.2025.10999118>
3. Wu, M.-H., Hsu, F.-H., Hunag, J.-H., Wang, K., Liu, Y.-Y., Chen, J.-X., Wang, H.-J., & Yang, H.-T. (2024). MPSD: A robust defense mechanism against malicious PowerShell scripts in Windows systems. *Electronics*, 13(18), Article 3717. <https://doi.org/10.3390/electronics13183717>
4. Zhang, S., Li, S., Lu, J., & Yang, W. (2025). Power-ASTNN: A deobfuscation and AST neural network enabled effective detection method for malicious PowerShell scripts. *Computers & Security*, 154, Article 104441. <https://doi.org/10.1016/j.cose.2025.104441>



5. Rose, A. J., Graham, S. R., Schubert Kabban, C. M., Krasnov, J. J., & Henry, W. C. (2024). ScriptBlock smuggling: Uncovering stealthy evasion techniques in PowerShell and .NET environments. *Journal of Cybersecurity and Privacy*, 4(2), 153–166. <https://doi.org/10.3390/jcp4020008>
6. Hendler, D., Kels, S., & Rubin, A. (2018). Detecting malicious PowerShell commands using deep neural networks. In *Proceedings of the 2018 Asia Conference on Computer and Communications Security* (pp. 187–197). Association for Computing Machinery. <https://doi.org/10.1145/3196494.3196511>
7. Hendler, D., Kels, S., & Rubin, A. (2020). AMSI-based detection of malicious PowerShell code using contextual embeddings. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20)* (pp. 679–693). Association for Computing Machinery. <https://doi.org/10.1145/3320269.3384742>
8. Fang, Y., Zhou, X., & Huang, C. (2021). Effective method for detecting malicious PowerShell scripts based on hybrid features. *Neurocomputing*, 448, 30–39. <https://doi.org/10.1016/j.neucom.2021.03.117>
9. Song, J., Kim, J., Choi, S., Kim, J., & Kim, I. (2021). Evaluations of AI-based malicious PowerShell detection with feature optimizations. *ETRI Journal*, 43(3), 549–560. <https://doi.org/10.4218/etrij.2020-0215>
10. Alahmadi, A., Alkhraan, N., & BinSaeedan, W. (2022). MPSAutodetect: A malicious PowerShell script detection model based on stacked denoising auto-encoder. *Computers & Security*, 116, Article 102658. <https://doi.org/10.1016/j.cose.2022.102658>
11. Xiong, C., Li, Z., Chen, Y., Zhu, T., Wang, J., Yang, H., & Ruan, W. (2022). Generic, efficient, and effective deobfuscation and semantic-aware attack detection for PowerShell scripts. *Frontiers of Information Technology & Electronic Engineering*, 23(3), 361–381. <https://doi.org/10.1631/FITEE.2000436>
12. Chai, H., Ying, L., Duan, H., & Zha, D. (2022). Invoke-Deobfuscation: AST-based and semantics-preserving deobfuscation for PowerShell scripts. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 295–306). IEEE. <https://doi.org/10.1109/DSN53405.2022.00039>
13. Ugarte, D., Maiorca, D., Cara, F., & Giacinto, G. (2019). PowerDrive: Accurate de-obfuscation and analysis of PowerShell malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019* (pp. 240–259). Springer. https://doi.org/10.1007/978-3-030-22038-9_12
14. Fu, Z., Song, L., Ding, S. H. H., Alaca, F., & Acharya, S. (2025). 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*, 28(4), Article 55, 1–23. <https://doi.org/10.1145/3771542>
15. Sudhakar, & Kumar, S. (2020). An emerging threat fileless malware: A survey and research challenges. *Cybersecurity*, 3, Article 1. <https://doi.org/10.1186/s42400-019-0043-x>
16. MITRE. (n.d.). *Obfuscated files or information: HTML smuggling (T1027.006)*. MITRE ATT&CK. Retrieved September 19, 2026, from <https://attack.mitre.org/techniques/T1027/006/>
17. MITRE. (n.d.). *Detection strategy for HTML smuggling via JavaScript Blob + dynamic file drop (DET0313)*. MITRE ATT&CK. Retrieved September 19, 2026, from <https://attack.mitre.org/detectionstrategies/DET0313/>
18. Microsoft Threat Intelligence. (2021, November 11). *HTML smuggling surges: Highly evasive loader technique increasingly used in banking malware, targeted attacks*. Microsoft Security Blog. <https://www.microsoft.com/en-us/security/blog/2021/11/11/html-smuggling-surges-highly-evasive-loader-technique-increasingly-used-in-banking-malware-targeted-attacks>
19. World Wide Web Consortium. (2026, September 12). *File API (W3C Working Draft)*. <https://www.w3.org/TR/2026/WD-FileAPI-20260912/>
20. Byeon, H. (2026). Trust propagation failure in the recursive-execution path of nested archives. *International Journal of Safety and Security Engineering*, 16(3), 503–508. <https://doi.org/10.18280/ijss.160303>
21. Microsoft. (n.d.). *[MS-FSCC]: Zone.Identifier stream name*. Microsoft Learn. Retrieved September 19, 2026, from https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/6e3f7352-d11c-4d76-8c39-2516a9df36e8
22. Microsoft. (n.d.). *Microsoft Defender SmartScreen overview*. Microsoft Learn. Retrieved September 19, 2026, from <https://learn.microsoft.com/en-us/windows/security/operating-system-security/virus-and-threat-protection/microsoft-defender-smartscreen/>
23. Microsoft. (n.d.). *Microsoft Defender Antivirus in Windows overview*. Microsoft Learn. Retrieved September 20, 2026, from <https://learn.microsoft.com/en-us/defender-endpoint/microsoft-defender-antivirus-windows>



24. Microsoft. (n.d.). *Antimalware Scan Interface (AMSI)*. Microsoft Learn. Retrieved September 19, 2026, from <https://learn.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal>
25. Microsoft. (n.d.). *About execution policies*. Microsoft Learn. Retrieved September 20, 2026, from https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-5.1

Отримано редакцією журналу / Received: 22.06.26

Прорецензовано / Revised: 30.06.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.