**УДК 004.056.5**

***Мордовцев М.В.,***

*студент ІІ курсу магістратури*
*Харківського національного університету радіоелектроніки*

***Федюшин О.І.,***

*к.т.н. доцент кафедри безпеки інформаційних технологій*
*Харківського національного університету радіоелектроніки*

***Морозов О.Ю.,***

*аспірант кафедри безпеки інформаційних технологій*
*Харківського національного університету радіоелектроніки*

# ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ПРИХОВУВАННЯ ІНФОРМАЦІЇ У PE-ФАЙЛАХ

**Анотація.** У роботі виконано порівняльний аналіз чотирьох методів приховування інформації у PE-файлах: використання службових полів заголовків, простору файлового вирівнювання, таблиці сертифікатів Authenticode та альфа-каналу іконок. Для кожного підходу розглянуто потенційну місткість контейнера, характер модифікації носія та можливості виявлення прихованих даних методами структурного й статистичного аналізу. Запропоновано набір перевірок для виявлення ознак приховування та визначено обмеження їх застосування. Отримані результати можуть бути використані під час розроблення засобів аналізу підозрілих виконуваних файлів і задач цифрової криміналістики.

**Ключові слова:** PE-файли, приховування інформації, стеганографія, структурний аналіз, Authenticode, альфа-канал.

Актуальність дослідження зумовлена зростанням кількості кіберзагроз, у межах яких виконувані файли використовуються не лише як носій програмного коду, а й як контейнер для прихованих конфігураційних даних, команд керування або службової інформації. Особливий інтерес для зловмисників становлять структури формату Portable Executable (PE), які можуть модифікуватися без безпосереднього впливу на виконання програми. Можливості використання структур PE для приховування повідомлень розглянуто D. Shin та співавторами [1]. Авторами було показано, що виконувані файли можуть використовуватися не лише як носії

програмного коду, а й як контейнери для прихованих даних. Наявність таких можливостей створює передумови для застосування PE-файлів як у легітимних задачах цифрового маркування програмного забезпечення, так і в кіберзлочинній діяльності для прихованого зберігання конфігураційних параметрів, команд або інших службових відомостей.

Сучасні кампанії поширення шкідливого програмного забезпечення демонструють використання прихованих даних у службових структурах PE-файлів для ускладнення виявлення під час статичного аналізу. Зокрема, у 2025 році дослідниками компанії G DATA було зафіксовано використання неавтентифікованих атрибутів цифрового підпису Authenticode для зберігання конфігурацій шкідливого програмного забезпечення [2]. Це підтверджує необхідність розроблення підходів до виявлення прихованої інформації не лише в програмному коді, а й у допоміжних структурах виконуваних файлів.

Мета роботи– порівняти чотири методи приховування інформації у PE-файлах та обґрунтувати вимоги до засобу її вбудовування, видобування й виявлення ознак приховування.

У роботі розглянуто чотири області можливого розміщення даних: резервні та службові поля заголовків, простір файлового вирівнювання, таблицю сертифікатів і альфа-канал зображень іконок у ресурсах PE (рис. 1). Для кожного методу розглянуто місткість, характер змін файла, ознаки для аналізу та обмеження перевірок.

Перший метод використовує поля заголовків PE-файлу. Його передумовами є наявність резервних ділянок та невикористання завантажувачем секційних COFF-релокацій і застарілої COFF-інформації про номери рядків у виконуваному образі. Для полів «reserved, must be zero» специфікація передбачає ігнорування вмісту споживачем формату, водночас вимагаючи нульового значення під час формування файла [3]. Для контейнеру, що складається з полів e_res і e_res2 DOS-заголовка,

LoaderFlags та чотирьох полів кожної секції (PointerToRelocations, PointerToLinenumbers, NumberOfRelocations, NumberOfLinenumbers), місткість становить $C_h = 32 + 12N$ байтів, де $N$ – кількість секцій. За $N = 4$ доступно 80 байтів. Перезапис не збільшує розмір файлу, але потребує оцінювання допустимості зміни полів конкретного носія.

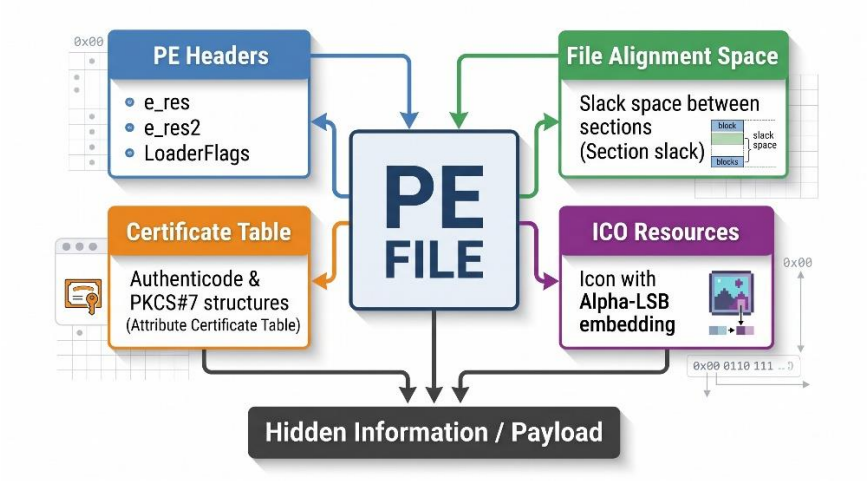


Рис. 1 – Області можливого приховування інформації у структурі PE-файлу

Виявлення у заголовках базується на підході структурного аналізу формату PE, описаному K. Hahn та реалізованому в засобі PortEx [7], у межах якого перевіряються сигнатури, межі структур і значення службових полів на відповідність вимогам специфікації. Ненульові LoaderFlags, PointerToRelocations і NumberOfRelocations фіксуються як відхилення від вимог до образу; для полів номерів рядків враховується рекомендація нульового значення [3]. Ненульові e_res і e_res2 є евристичною знахідкою. Такий контроль виявляє аномалію, але не встановлює її причину.

Другий метод використовує простір файлового вирівнювання. Округлення розміру секції у файлі до кратного значення, заданого полем вирівнювання файлових даних (FileAlignment), може утворювати додаткові байти за межами розміру секції в пам’яті (VirtualSize) [3]. Якщо файловий

зсув початку секції (PointerToRawData) позначити $P$, розмір її даних у файлі (SizeOfRawData) – $R$, а розмір секції в пам'яті (VirtualSize) – V, то за $R > V > 0$ кандидатна ділянка має межі $[P + V;\ P + R)$. Також розглядається проміжок між кінцем таблиці секцій і кінцем області заголовків, розмір якої задається полем SizeOfHeaders. Ці байти придатні для запису за умови відсутності потрібних програмі даних і посилань інших структур. Ємність $C_i = \sum s_i$ визначається сумою довжин відібраних неперекривних ділянок, придатних для перезапису.

Виявлення у просторі вирівнювання ґрунтується на відомому принципі пошуку ненульового заповнення, застосованому в PEberus [8]. Після контролю меж і перекриттів для кожної ділянки довжиною $s_i > 0$ визначаються кількість ненульових байтів $c_i$ та частка $d_i = c_i/s_i$. У запропонованому наборі перевірок правило $c_i > 0$ реєструє кожну ділянку окремо; це адаптація принципу, а не повне відтворення підсумкового рішення PEberus для файла.

Третій метод використовує таблицю сертифікатів, виключену з обчислення хешу PE-образу [3], та неавтентифіковані атрибути, не захищені підписом. Це дозволяє додавати дані без зміни виконуваних інструкцій [4]. Розглядаються байти після об'єкта PKCS#7 у WIN_CERTIFICATE та дані всередині неавтентифікованого атрибута. Ємність контейнеру визначається доступним або створеним обсягом у конкретному варіанті структури; файл може збільшуватися. Незмінність хешу образу не гарантує успішної перевірки Authenticode: зокрема, Microsoft описує посилену перевірку сторонніх даних у WIN_CERTIFICATE [5].

Виявлення у таблиці сертифікатів спирається на відомі перевірки доповнення Authenticode [4, 5]. За файловим зсувом каталогу контролюються межі записів WIN_CERTIFICATE, їхня довжина (dwLength) та 8-байтове вирівнювання [3]. Кінець PKCS#7 зіставляється з

межами запису, а сторонні ненульові байти відокремлюються від нульового вирівнювання. Для другого способу розміщення потрібен розбір атрибутів SignerInfo: зіставлення типів, довжин і структури значень, як показано в [2, 4].

Четвертий метод використовує молодший біт альфа-каналу 32-бітних іконок у ресурсах PE. Заміна цього біта змінює прозорість щонайбільше на одиницю, зберігаючи RGB-компоненти: ємність утворюється за рахунок малої зміни зображення. Альфа-LSB у ICO розглянуто D. Noever та F. McKee [6]. За одного біта на піксель $C_a = [K/8]$ байтів, де K – кількість придатних пікселів. Критерій $\alpha \geq 2$ зберігається після заміни молодшого біта й забезпечує відтворюваний відбір. За $K = 4096$ доступно 512 байтів. PNG-ресурси потребують декодування й повторного кодування; їх розмір може змінитися.

Виявлення в іконках розглядається як задача побудови евристичного детектора. Після видобування зображень і перевірки придатності альфа-каналу можливими ознаками є співвідношення частот значень 254 і 255, розподіл молодших бітів та поява значень 254 усередині майже непрозорих областей. Ці ознаки випливають із механізму заміни біта й розглядаються як напрям подальшого дослідження, а не як відтворення готового опублікованого детектора. Пороги та ефективність такого підходу необхідно встановити експериментально на окремих вибірках для калібрування й оцінювання.

Зіставлення виконано за місткістю й характером модифікацій носія (табл. 1), а також правилами виявлення та межами їх інтерпретації (табл. 2). Місткість оцінюється за кількістю байтів, доступних для вбудовування у визначеному контейнері. Для альфа-LSB кількість доступних бітів перераховується в байти; формат повідомлення та його службові дані не враховуються.

Таблиця 1 – Ємність і характер модифікацій носія

| Метод | Місткість, байт | Зміни носія |
|---|---|---|
| Поля заголовків | $32 + 12N$ для обраного набору | Перезапис полів; довжина файла стала |
| Простір вирівнювання | $\sum s_i$ | Перезапис наявних ділянок; довжина стала |
| Таблиця сертифікатів | Залежить від способу доповнення | Можливе збільшення файла та зміна довжин структур |
| Альфа-канал іконок | $[K/8]$ | Зміна прозорості; розмір PNG може змінитися |

Таблиця 2 – Перевірки областей та обмеження виявлення

| Область | Правило перевірки | Обмеження |
|---|---|---|
| Заголовки | Контроль нульових значень; евристики резервних полів | Відхилення не встановлює причину зміни |
| Вирівнювання | Ненульові байти $c_i$ та їх частка $d_i$ | Можливе нетипове початкове заповнення |
| Сертифікати | Межі структур; типи й значення атрибутів | Допустимий атрибут також може містити повідомлення |
| Іконки | Розподіл прозорості та просторові ознаки | Потрібні калібрування й оцінювання на окремій вибірці |

Вибір контейнера залежить від умов використання. Для індивідуальної мітки важливі відтворюване видобування та збереження після перетворень файла, для аналізу підозрілих зразків – охоплення областей і пояснення знахідок.

Обґрунтований набір охоплює нормативні й евристичні перевірки заголовків, ненульове заповнення вирівнювання, структурний аналіз сертифікатної таблиці та обчислення статистичних ознак іконок. Звіт має містити координати ділянки, значення ознаки, правило спрацювання й статус повноти аналізу. Видобування відомого повідомлення є окремою процедурою. Аномалія потребує подальшого дослідження; відсутність знахідок стосується лише перевірених областей.

За результатами порівняння доцільно передбачити чотири модулі аналізатора та спільний формат звіту. Для кожної знахідки необхідно вказувати структуру, файловий зсув, довжину ділянки, виміряне значення

ознаки й підставу спрацювання. Невиконані перевірки мають позначатися окремо із зазначенням причини. Видобування повідомлення відомого формату слід відокремлювати від пошуку довільного прихованого вмісту. Знайдена аномалія є підставою для подальшого дослідження, але сама по собі не доводить шкідливості файла.

Отже, порівняння встановлює зв’язок між областю вбудовування, способом оцінювання місткості та відповідними перевірками. Результатом є обґрунтування поєднання нормативних, евристичних і статистичних перевірок у програмному засобі. Для маркування примірників потрібне оцінювання впливу запису та збереження мітки, для аналізу підозрілих файлів – локалізація ознак приховування з поясненням їх змісту. Практична місткість, сумісність модифікованих файлів і точність виявлення залишаються завданнями подальшого експериментального дослідження.

## Список використаних джерел


1. Data Hiding in Windows Executable Files / D. Shin, Y. Kim, K. Byun, S. Lee. *Proceedings of the 6th Australian Digital Forensics Conference*. Perth, 2008. P. 153–159. DOI: https://doi.org/10.4225/75/57b2748e40cc0.
2. Go L., Hahn K. ConnectUnwise: Threat actors abuse ConnectWise as builder for signed malware. *G DATA Security Blog*. 23.06.2025. URL: https://blog.gdatasoftware.com/2025/06/38218-connectwise-abuse-malware (дата звернення: 31.10.2026).
3. Microsoft. PE Format. *Microsoft Learn*. URL: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (дата звернення: 31.10.2026).
4. Jones K. Authenticode Stuffing Tricks. 15.04.2016. URL: https://vcsjones.dev/authenticode-stuffing-tricks/ (дата звернення: 31.10.2026).
5. Microsoft. Microsoft Security Advisory 2915720: Changes in Windows Authenticode Signature Verification. 10.12.2013. Дата оновлення: 29.07.2014. URL: https://learn.microsoft.com/en-us/security-updates/SecurityAdvisories/2014/2915720.
6. Noever D., McKee F. Favicon Trojans: Executable Steganography Via Ico Alpha Channel Exploitation. *arXiv*. 2025. arXiv:2507.09074. DOI: https://doi.org/10.48550/arXiv.2507.09074.
7. Hahn K. Robust Static Analysis of Portable Executable Malware : master’s

thesis in computer science. Leipzig : HTWK Leipzig, 2014. URL: https://papers.put.as/papers/malware/2014/masterthesiskatja.pdf (дата звернення: 31.10.2026).
8. Against All Odds: Winning the Defense Challenge in an Evasion Competition with Diversification / E. Quiring, L. Pirch, M. Reimsbach, D. Arp, K. Rieck. *arXiv*. 2020. arXiv:2010.09569. DOI: https://doi.org/10.48550/arXiv.2010.09569.