УДК 004.056:004.89:623.746.4-519

Туз Михайло Олександрович

студент VI курсу Харківського національного університету радіоелектроніки

Нехороших Дмитро Михайлович

асистент викладача кафедри безпеки інформаційних технологій Харківського національного університету радіоелектроніки

ДОСЛІДЖЕННЯ АРТЕФАКТІВ ПЕРЕВІРКИ СЕРЕДОВИЩА ВИКОНАННЯ ШКІДЛИВОГО ПЗ ТИПУ STEALER

Анотація. У роботі було досліджено артефакти перевірки середовища виконання шкідливого програмного забезпечення типу Stealer. Проаналізовано використання функцій WinAPI для отримання ідентифікаційних параметрів системи та подальшого порівняння отриманих значень. Було встановлено, що аналіз послідовностей викликів WinAPI та пов'язаних із ними умовних переходів дає змогу виявляти механізми перевірки середовища виконання й відтворювати відповідні етапи логіки роботи шкідливого програмного забезпечення.

Ключові слова: цифрова криміналістика, DFIR, Stealer, WinAPI, артефакти виконання, перевірка середовища.

На сьогодні кількість кібератак, спрямованих проти України досить швидко зростає, де крадіжка даних займає 37% (рис. 1). Зловмисники вкрай частіше використовують нові тактики, методи та процедури (TTPs) для компрометації кінцевих пристроїв. Одним із завдань шкідливого програмного забезпечення після реалізації початкового вектору компрометації (*англ. Initial Access*) є виявлення та розмежування контрольованого середовища виконання (пісочниць, віртуальних машин тощо) і реальних систем користувачів. Для цього можуть застосовуватися механізми перевірки системних параметрів, аналізу характеристик операційного середовища та обфускації коду (MITRE ATTACK – T1497

Virtualization/Sandbox Evasion).

Водночас з цим, необхідно зазначити, що команди DFIR фокусуються саме на дослідженні зразків, тобто на успішному виявленні відповідних артефактів та відновлення логіки виконання. Чому це є важливим? Тому що це надає змогу зрозуміти цілісно поведінку ШПЗ, та, відповідно до цього, вжити необхідних заходів захисту, які у майбутньому зможуть зменшити як фінансові так і репутаційні збитки компанії. Таким чином, дослідження артефактів перевірки середовища виконання шкідливого програмного забезпечення типу Stealer є актуальним завданням.



Рисунок 1 – Кількість кібератак за 2025 рік [2]

Для чого ШПЗ перевіряє середовище (*англ. Environment*) перед виконанням основного корисного навантаження (*англ. Payload*)?

DFIR спеціалістам необхідно розуміти, що якщо ми говоримо про діяльність APT-угруповань та інших зловмисних організацій, процес розробки ШПЗ це досить трудомісткий процес, який потребує залучення фахівців, технічних засобів та значних фінансових ресурсів. Згідно з Cyber Kill Chain (структурою атаки зі сторони зловмисника), фактичний етап перевірки середовища виконання логічно розташовується між етапами Exploitation та Installation (рис. 2). Тож саме на цьому етапі зловмиснику важливо переконатися, що ШПЗ виконується саме в цільовій системі, а не в

контрольованому середовищі аналізу. В майбутньому, для зловмисника це дозволяє зберегти функціональність ШПЗ від передчасного виявлення та не допустити розкриття його можливостей, на розробку яких було витрачено значні ресурси.



Рисунок 2 – модель Cyber Kill Chain

Методика аналізу. Для дослідження механізмів перевірки середовища виконання було використано програмне забезпечення IDA. Необхідно зазначити, що Stealer це кінцевий зразок ШПЗ, який був доставлений на заражений кінцевий пристрій через використання пакування в секції PE-файлу та попереднього шифрування за допомогою алгоритму RC4. Після вилучення та дешифрування другого етапу отриманий виконуваний файл було завантажено в IDA для подальшого статичного аналізу (рис. 3).

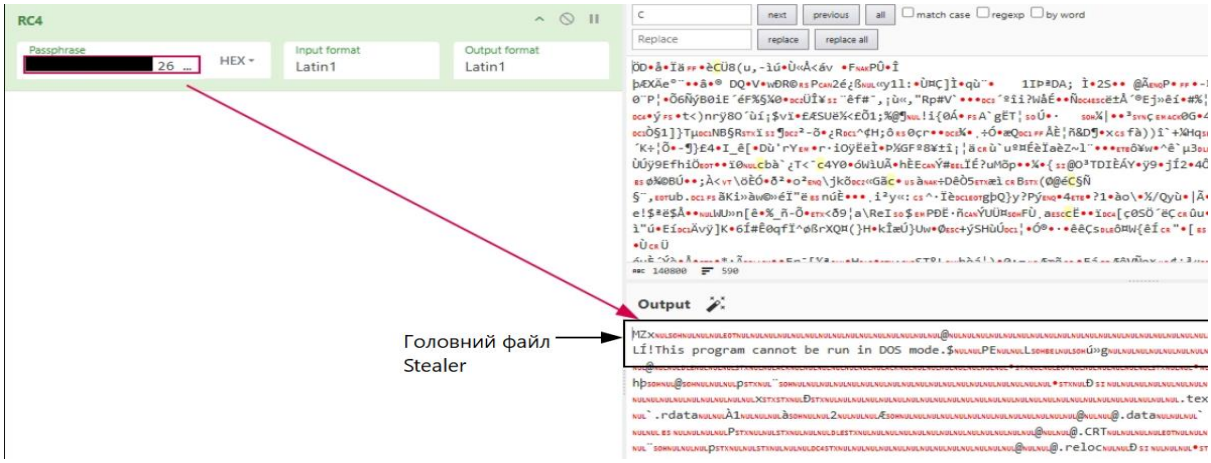


Рисунок 3 – отримання головного файлу Stealer

Після успішного завантаження головного файлу Stealer в програму IDA була помічена функція «start» за адресою «0x417210». Дана функція є головною і саме з неї починається робота та перевірки в Stealer.



Рисунок 4 – головна функція «start»

При дослідженні даної функції було помічено достатньо велику кількість перевірок за допомогою WinAPI функцій «GetComputerNameW», «GetUserNameW», «GetFileType», «GetModuleFileNameA», «GetTempPathW», які фактично виконують свою роботу перед запуском наступної функції «sub_416050».

При більш детальному дослідженні було встановлено, що функції «GetUserNameW» та «GetComputerNameW» використовуються конкретно для отримання імені поточного користувача та назви комп'ютера. Результати виконання цих функцій зберігаються у буферах lpBuffer_2 та lpBuffer (рис. 5).

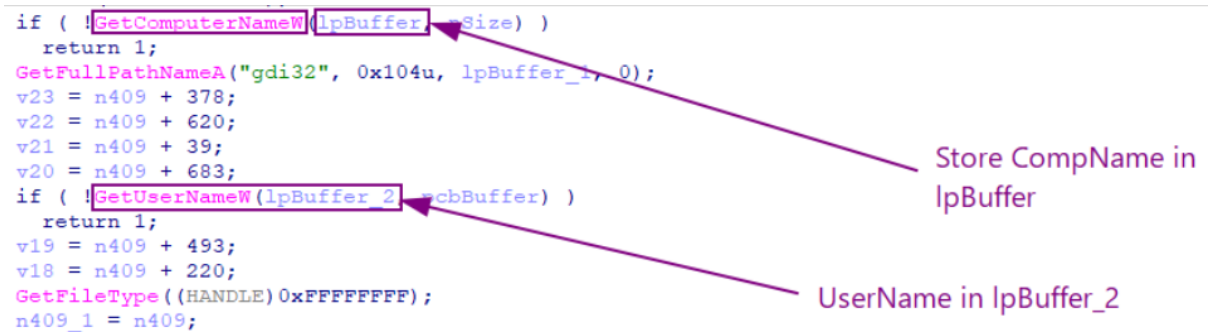


Рисунок 5 – успішне зберігання значень імені користувача та комп'ютера в буферах

Наступним етапом було досліджено умови порівняння отриманих даних. Тобто значення lpBuffer_2[0], lpBuffer_2[5] та lpBuffer_2[6]

порівнюються з десятковими значеннями 74, 111 та 101, які відповідають символам J, o та e у таблиці ASCII. Тобто, можна встановити, що ШПЗ перевіряє окремі позиції імені користувача, очікуючи структуру J****oe.

Аналогічним чином перевіряється назва комп'ютера. lpBuffer[0] порівнюється зі значенням 72 (символ H), а lpBuffer[3] – зі значенням 57 (символ 9).

Крім того, перевіряється nSize[0], що містить довжину назви комп'ютера, отриману через GetComputerNameW. Усі зазначені умови об'єднані логічним оператором OR, тому невідповідність хоча б одній із них призводить до виклику функції sub_416050 (рис. 6).

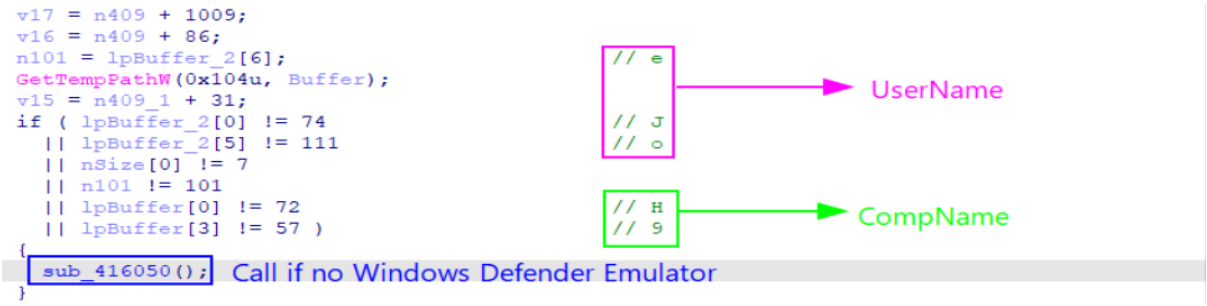


Рисунок 6 – перевірка імені користувача та комп’ютера перед викликом подальшої функції «sub_416050»

Як результат, були отримані часткові значення імені комп’ютера – H9**** та імені користувача – J****oe. Але, враховуючи неповноту отриманих значень, можна було припустити, що перевірка стосується виявлення середовища Windows Defender, проте точно стверджувати це було неможливо, тож для подальшого підтвердження було вирішено провести додаткове дослідження джерел та звітів щодо інших зразків ШПЗ. Після дослідження було отримано репорт, у якому ШПЗ використовувало перевірку за повними значеннями, які були «зашиті» в коді. Дані значення мали вигляд імені користувача «JohnDoe» та комп’ютера «HAL9TH» [1]. Зіставлення цих даних із результатами власного аналізу дозволило

підтвердити, що перевірки орієнтовані саме на середовище Windows Defender Emulator.

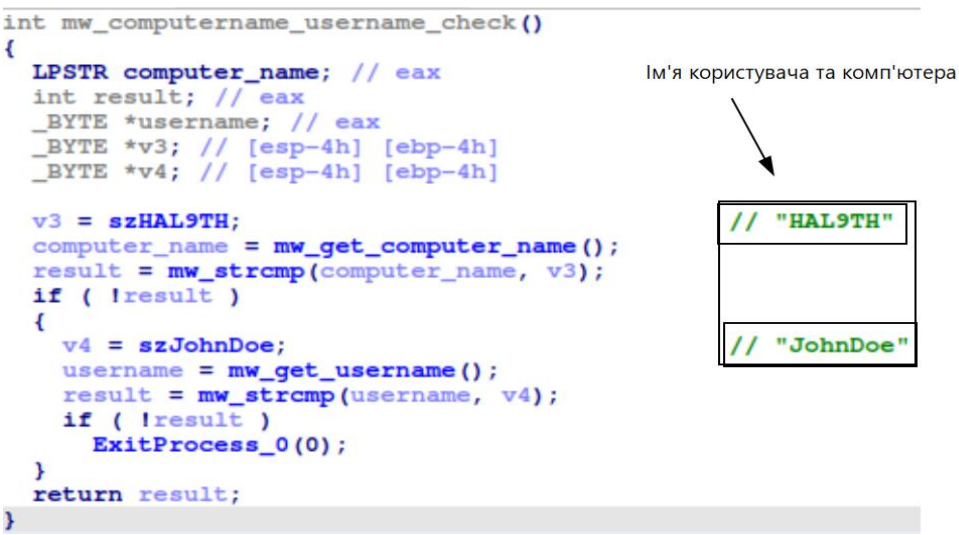


Рисунок 7 – отримання прикладу повної перевірки значень з іншого джерела [1]

Таким чином, ШПЗ виконувало перевірку середовища виконання Windows Defender Emulator. Ситуація в тому, що коли PE-файл завантажується на кінцевий пристрій, Windows Defender перевіряє статично файл, проте, звісно, цього не достатньо, тому він запускає x86/x64 інструкції та симулює роботу базових WinAPI-функцій ядра. Зловмисник обходить дану перевірку, тому що основне корисне навантаження не виконується, через успішне отримання імені користувача та комп’ютера, яке відповідає механізму захисту Windows Defender Emulator.

На рисунку нижче показаний приклад зловмисного коду, який зберігається в головній функції Stealer «sub_416050», при подальшій роботі.

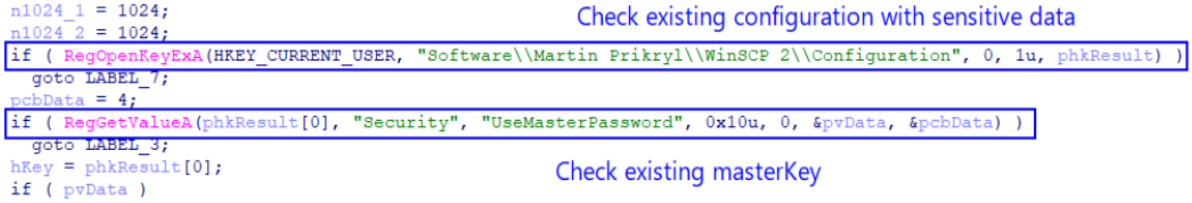


Рисунок 8 – Спроба отримати чутливі дані клієнта «WinSCP2»

Висновки. У результаті дослідження було встановлено, що ШПЗ типу Stealer використовує WinAPI-функції GetUserNameW та GetComputerNameW для параметрів імені користувача та комп’ютера для перевірки середовища виконання. Виявлено, що поєднання таких викликів із порівнянням отриманих значень та умовними переходами може свідчити про спроби виявлення контрольованих середовищ, зокрема Windows Defender Emulator. Отже, аналіз послідовностей викликів WinAPI та пов'язаних із ними умов виконання може бути використаний фахівцями DFIR для виявлення механізмів антианалізу та відновлення логіки роботи ШПЗ.

Список використаних джерел

1. MintsLoader: StealC and BOINC Delivery : website / eSentire Threat Response Unit. 2025. URL: https://www.esentire.com/blog/mintsloader-stealc-and-boinc-delivery (дата звернення: 25.10.2026).
2. СБУ за 2025 рік відбила понад 3 тисячі російських кібератак : веб-сайт / NV.ua. 2026. URL: https://nv.ua/ukr/ukraine/events/sbu-za-2025-rik-vidbila-ponad-3-tisyachi-rosiyskih-kiberatak-50578657.html (дата звернення: 27.10.2026).
3. Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile : NIST Special Publication 800-61, Rev. 3 / National Institute of Standards and Technology. Gaithersburg : NIST, 2025. URL: https://csrc.nist.gov/pubs/sp/800/61/r3/final (дата звернення: 29.10.2026).
4. Sikorski M., Honig A. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. San Francisco : No Starch Press, 2012. 648 p. ISBN 978-1-59327-290-6.