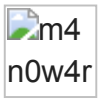
Một sample nhắm vào Bank ở VN

tradahacking.vn/đợt-rồi-tôi-có-đăng-một-status-xin-dạo-trên-fb-may-quá-cũng-có-vài-bạn-nhiệt-tình-gửi-cho-537b19ee3468

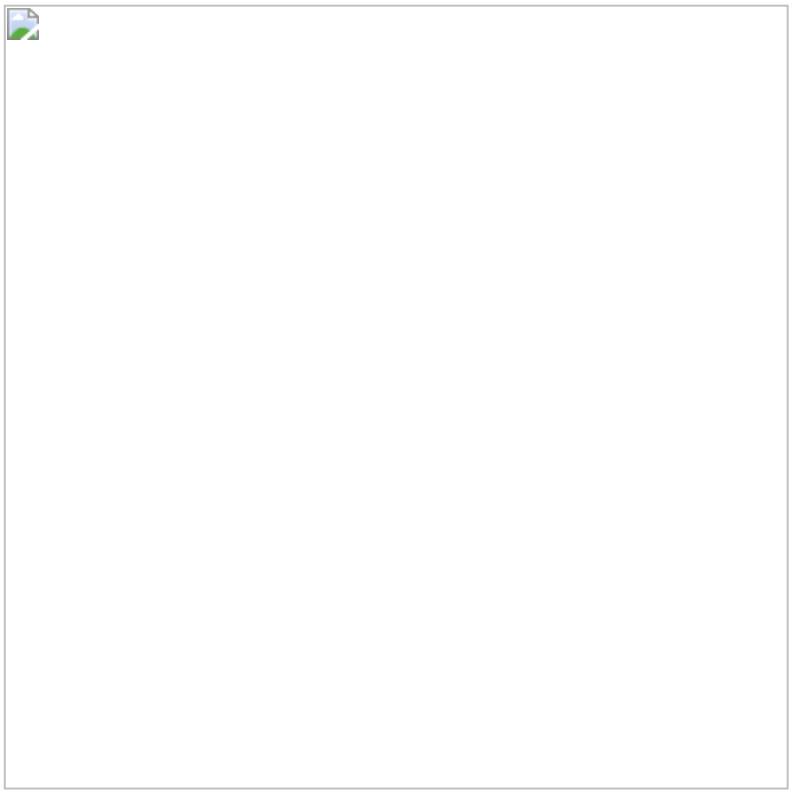
m4n0w4r October 10, 2019



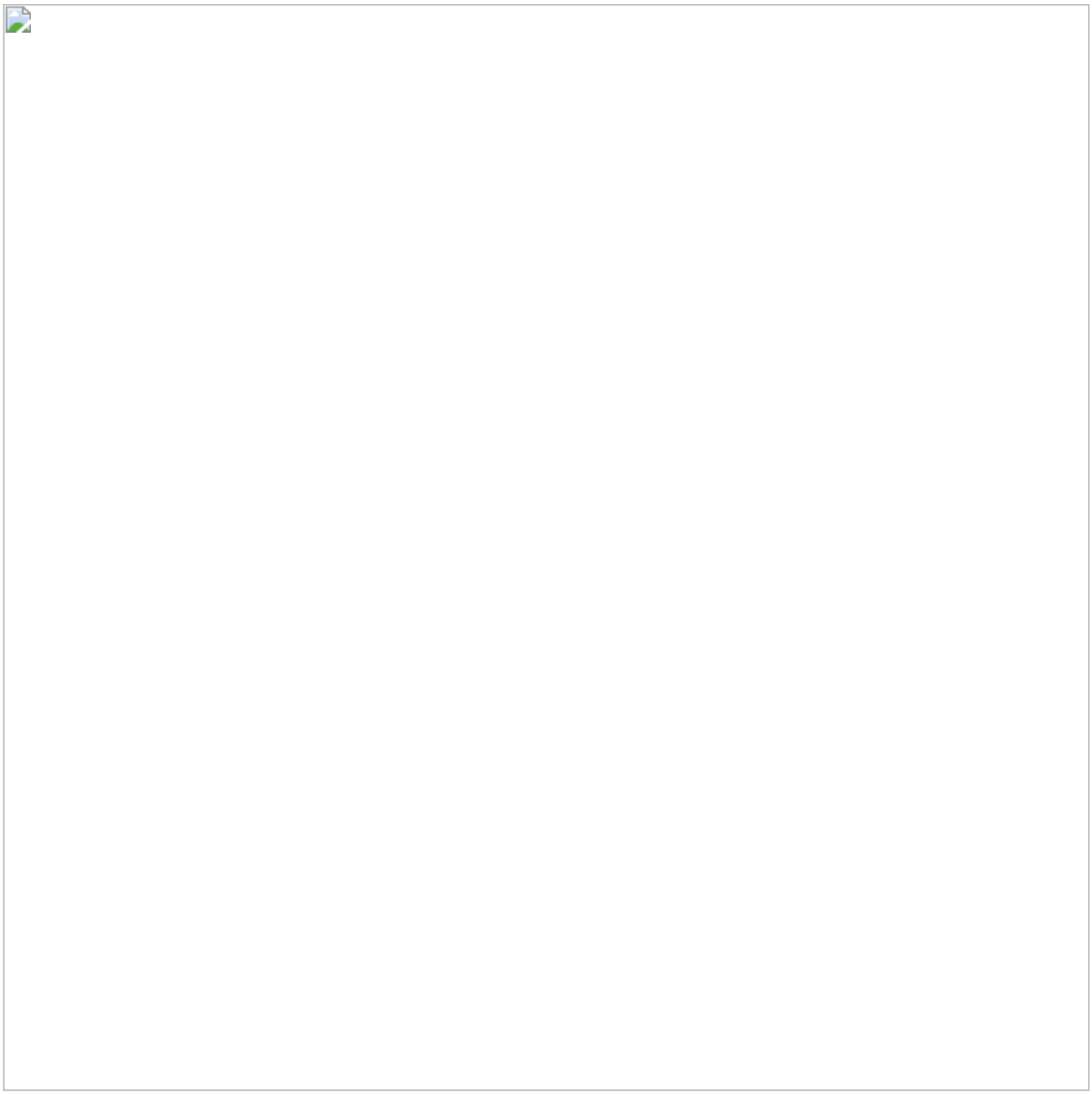
m4n0w4r
Follow
Oct 8, 2019

.

4 min read

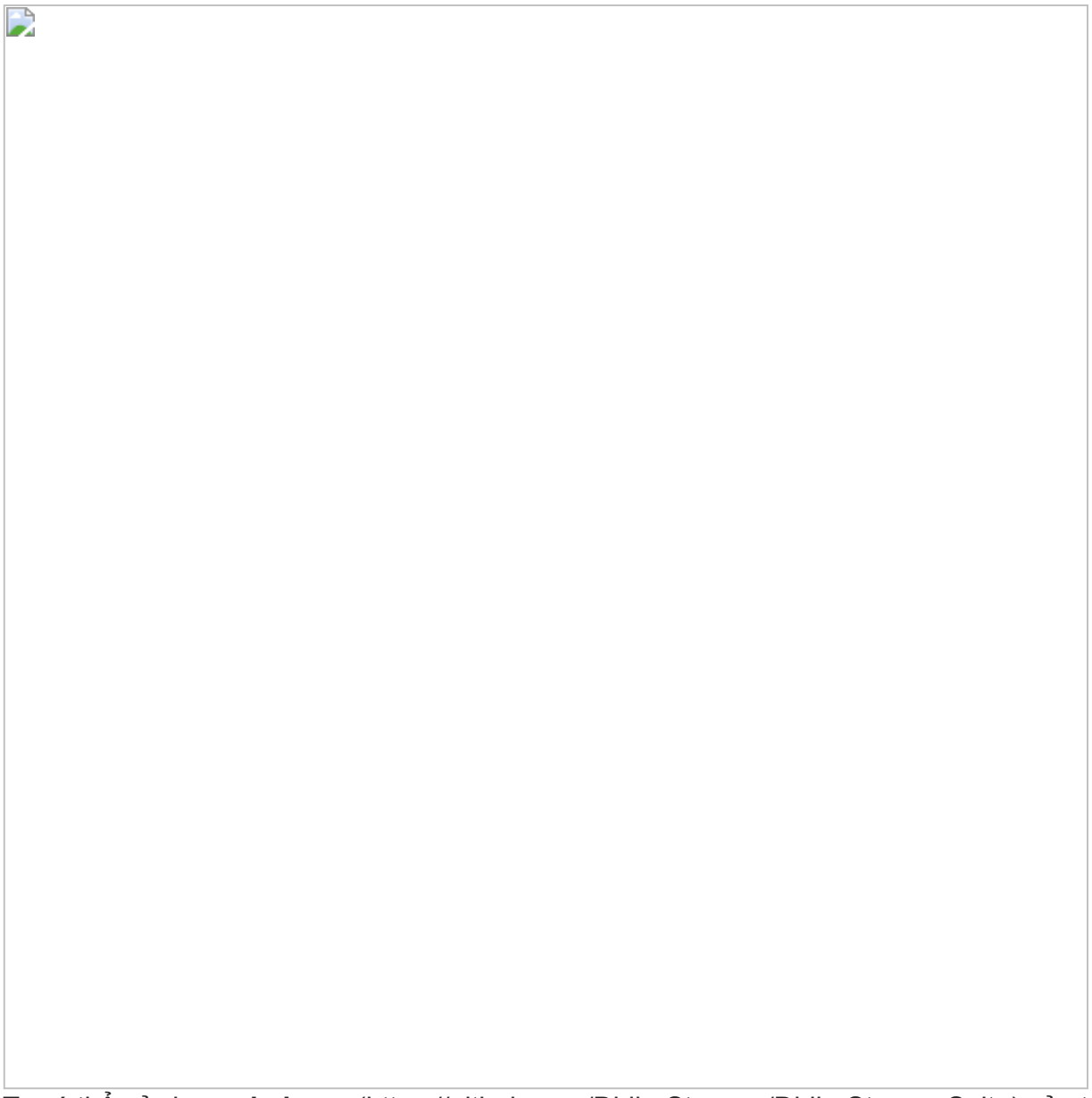


Đợt rồi, sau khi tôi có đăng status xin dạo trên FB, may quá cũng có vài bạn nhiệt tình gửi cho. Mẫu này theo nhận định thì target thẳng vào bộ phân nhân sự của Bank. Sample có tên dạng: **CV_<ten_ung_vien>_ChuyenVienKhachHangCaNhan**

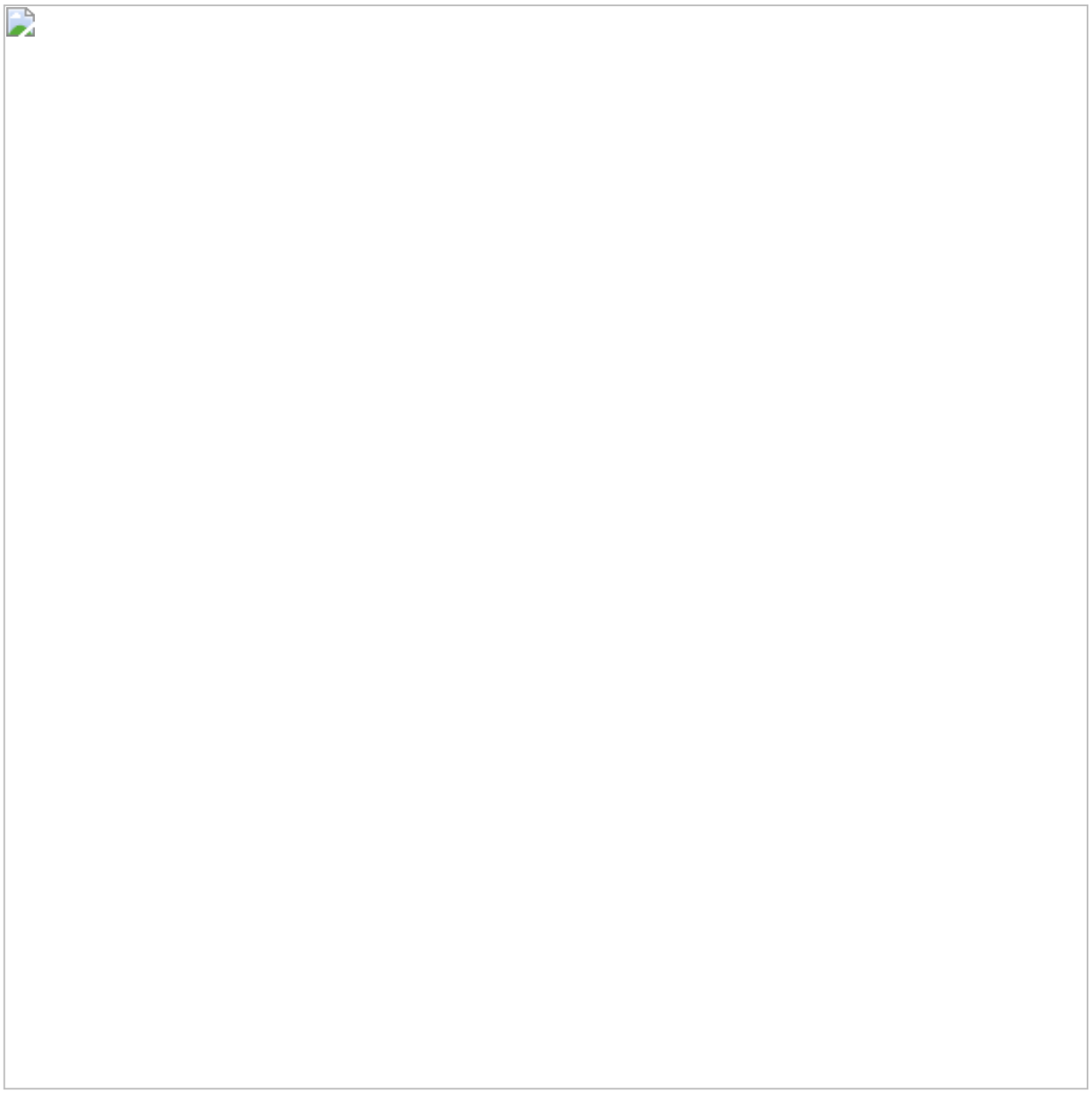


1. Phân tích XML file

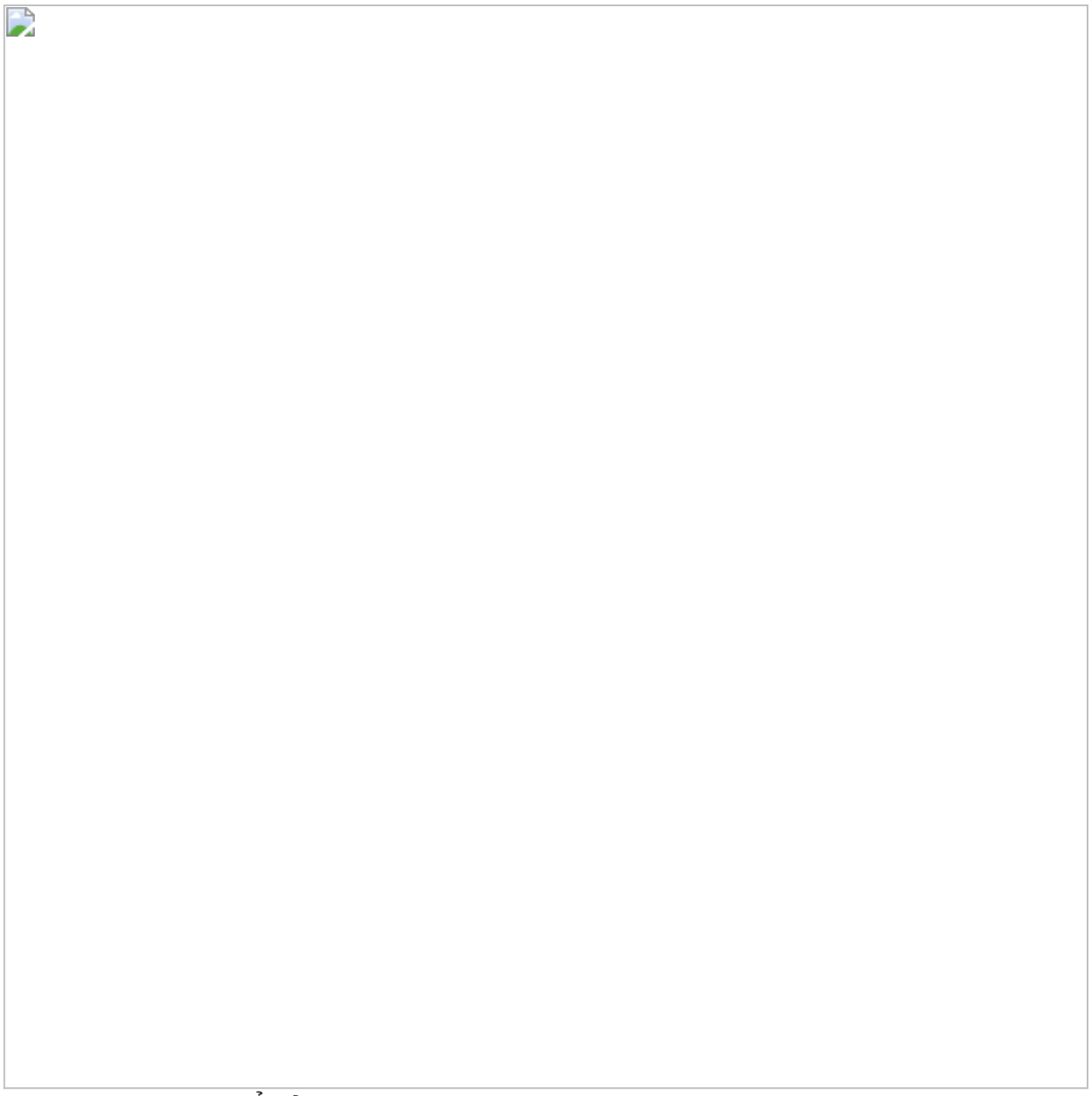
File nhận được có phần mở rộng là **.doc** nhưng khi kiểm tra bằng **HxD** thì thấy đây là một XML file, không phải là OLE file:



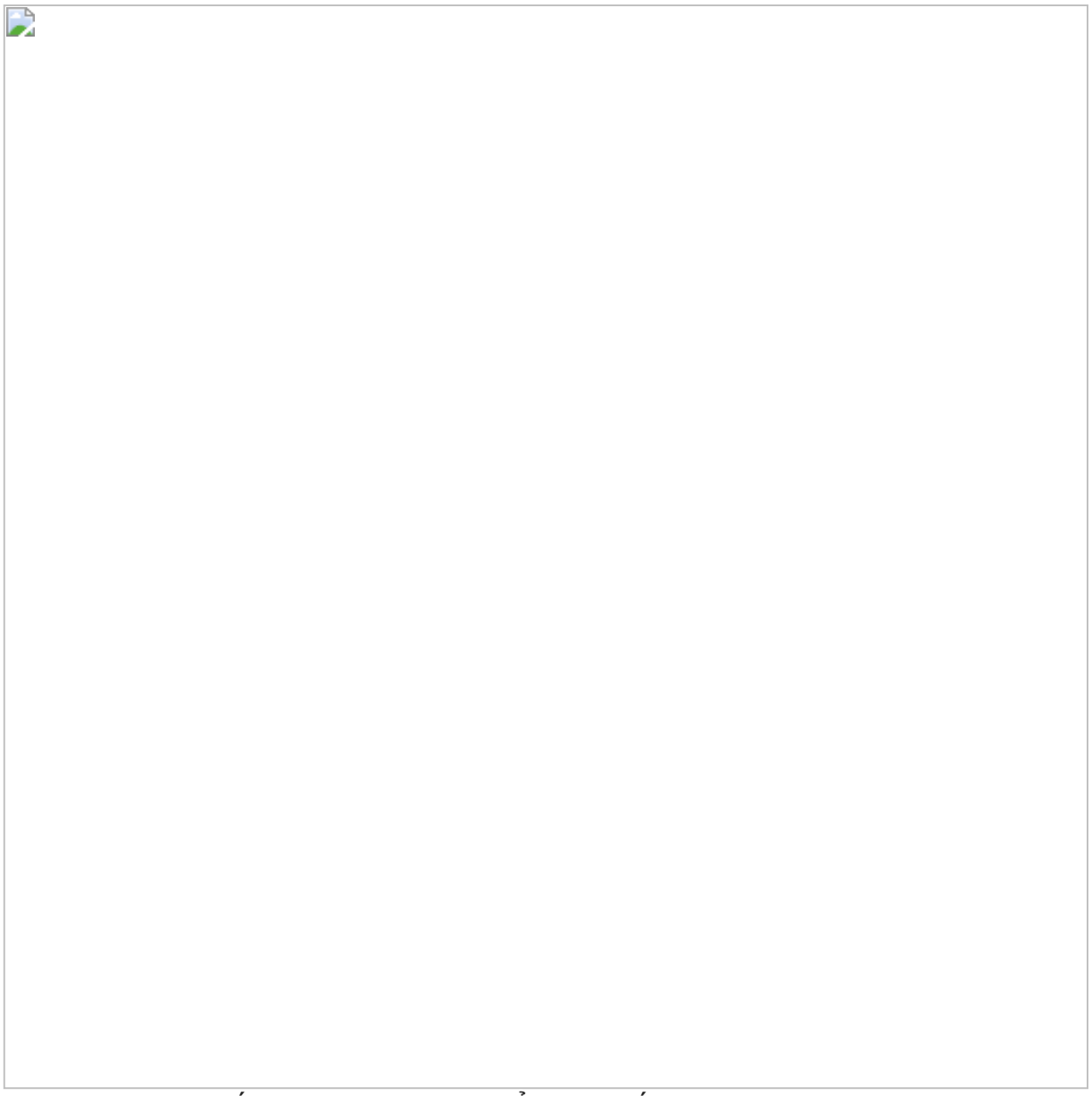
Ta có thể sử dụng **oledump** (https://github.com/DidierStevens/DidierStevensSuite) của tác giả **Didier Stevens** để kiểm tra thông tin liên quan:



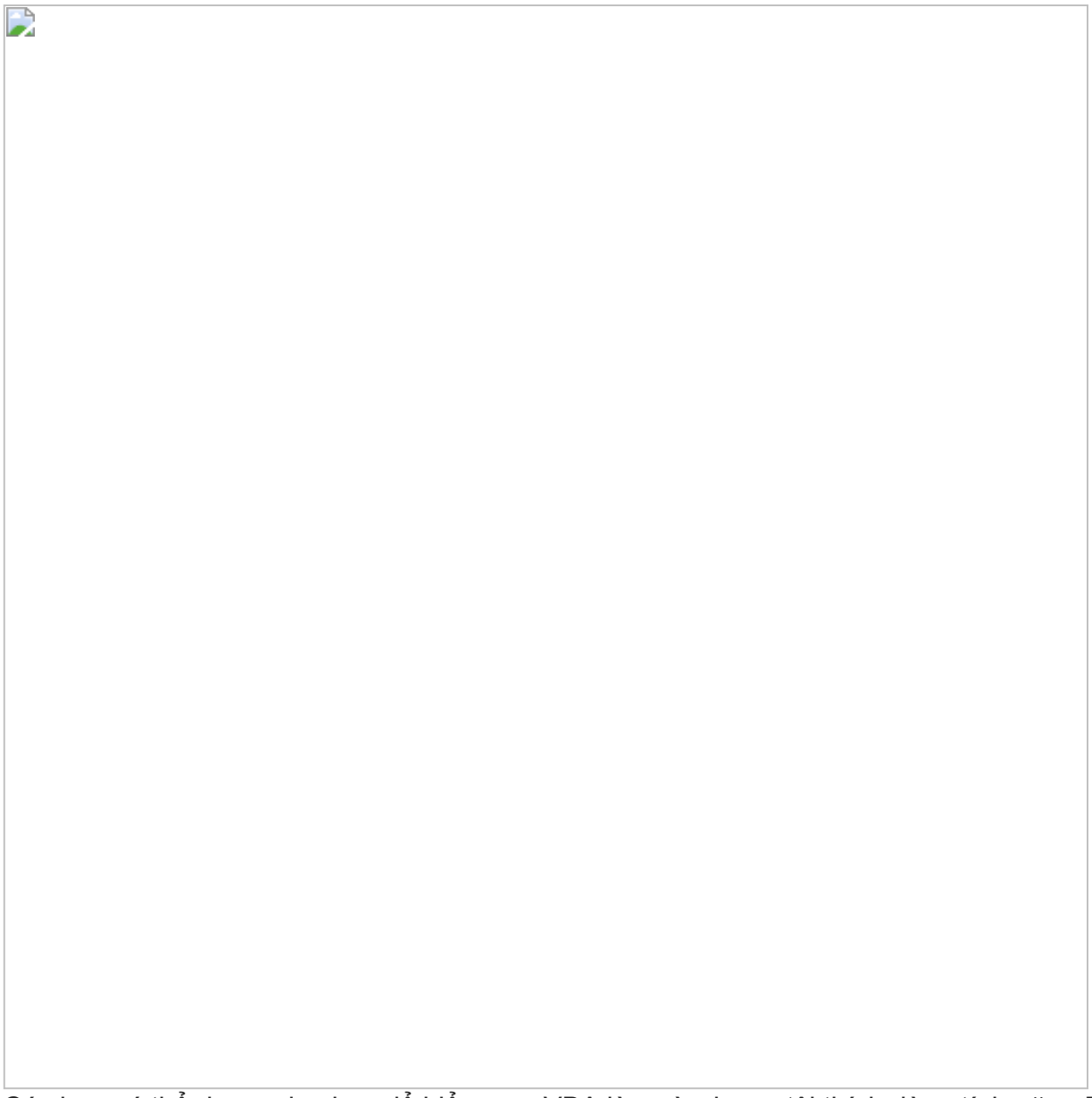
Như trên hình thì file này có hai stream chính là **A: oledata.mso** và **B: editdata.mso**. Trong đó tại stream B, oledump phát hiện stream **B3** là một macro stream. Hoàn toàn có thể dùng oledump để dump ra VBA code:



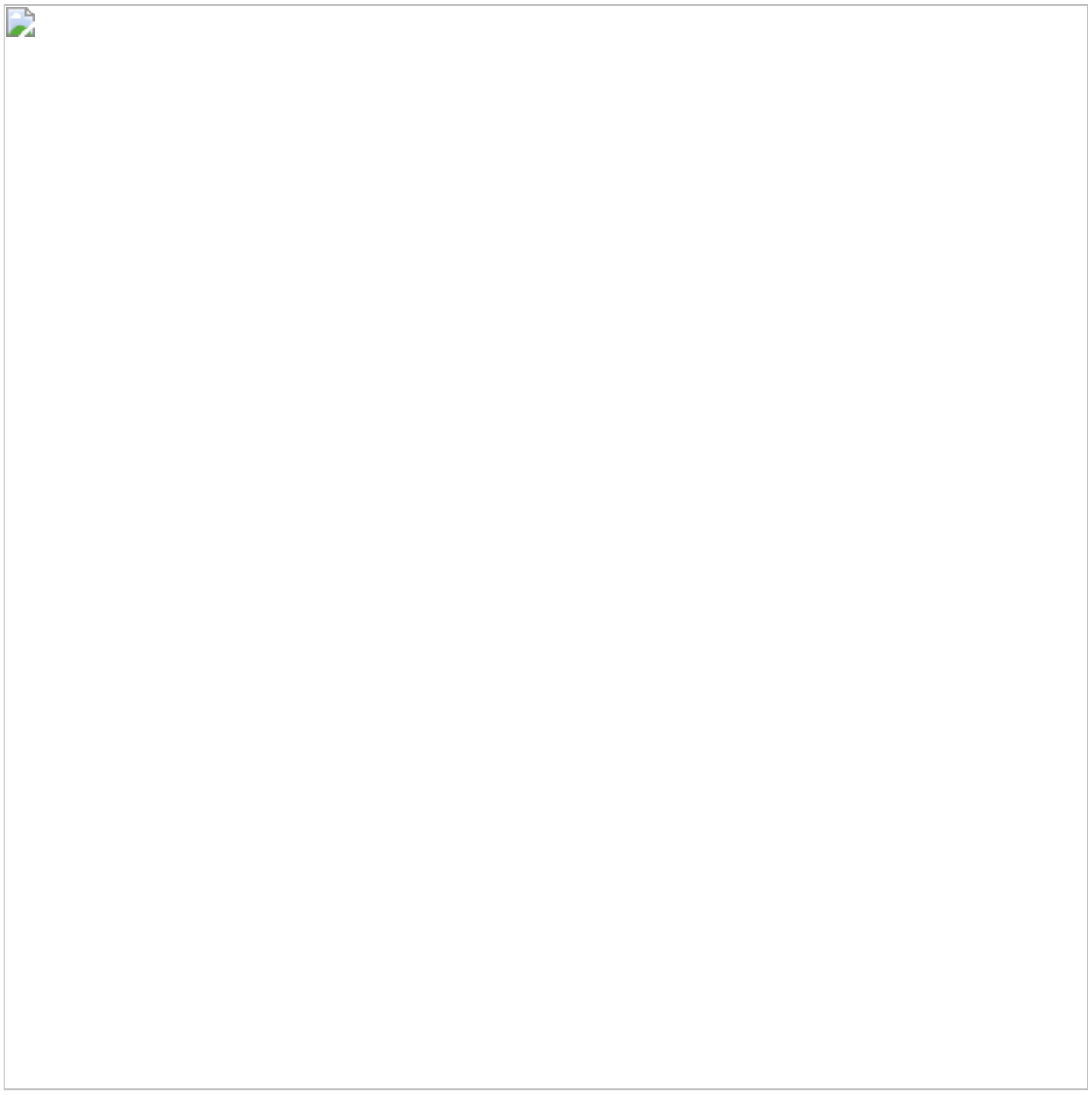
Ngoài ra, cũng có thể sử dụng **olevba** (https://github.com/decalage2/oletools/wiki/olevba) của tác giả **Philippe Lagadec** để dump VBA code:



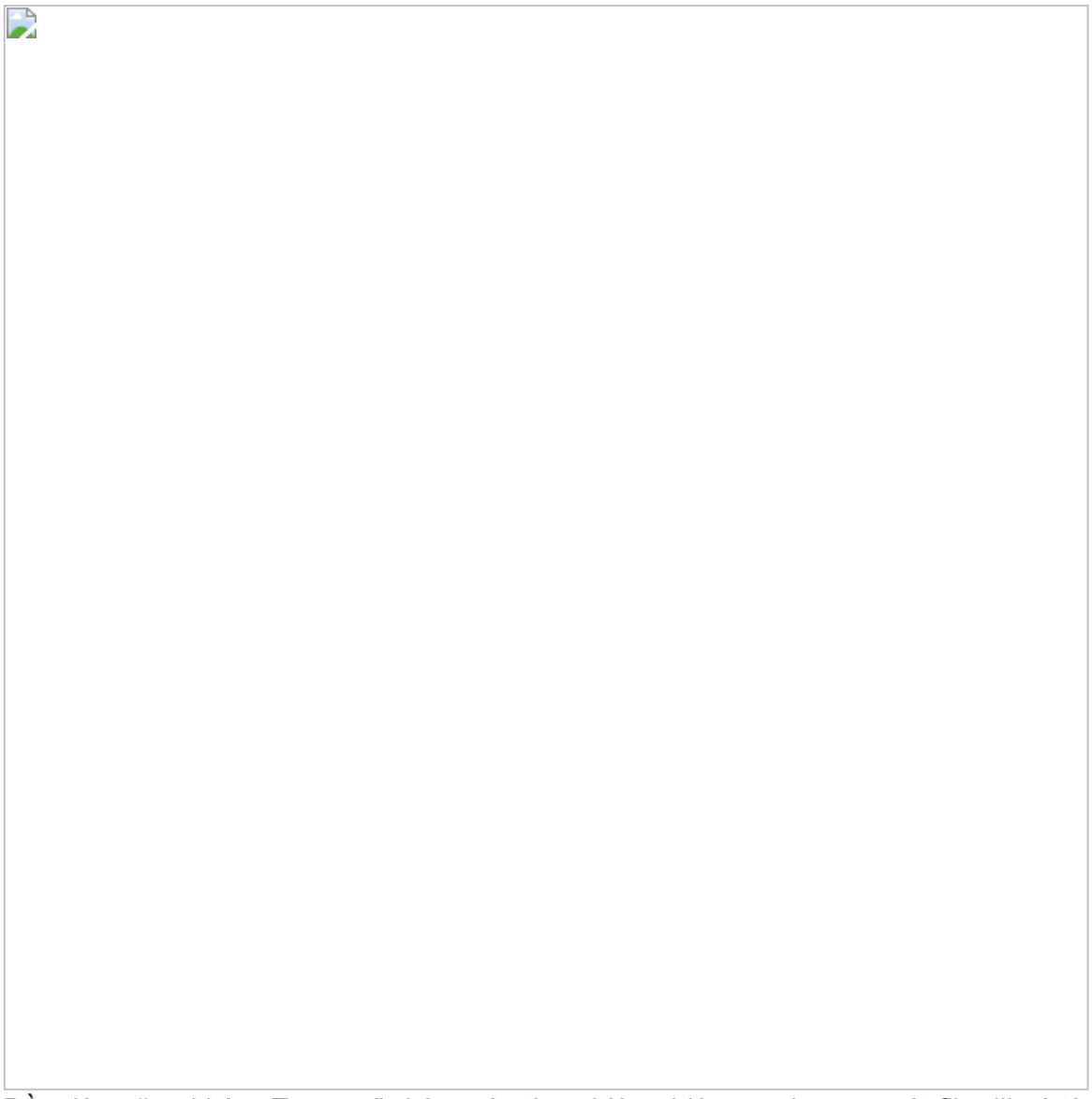
olevba còn cung cấp thêm các thông tin tổng hợp rất hữu ích trong quá trình parse VBA code:



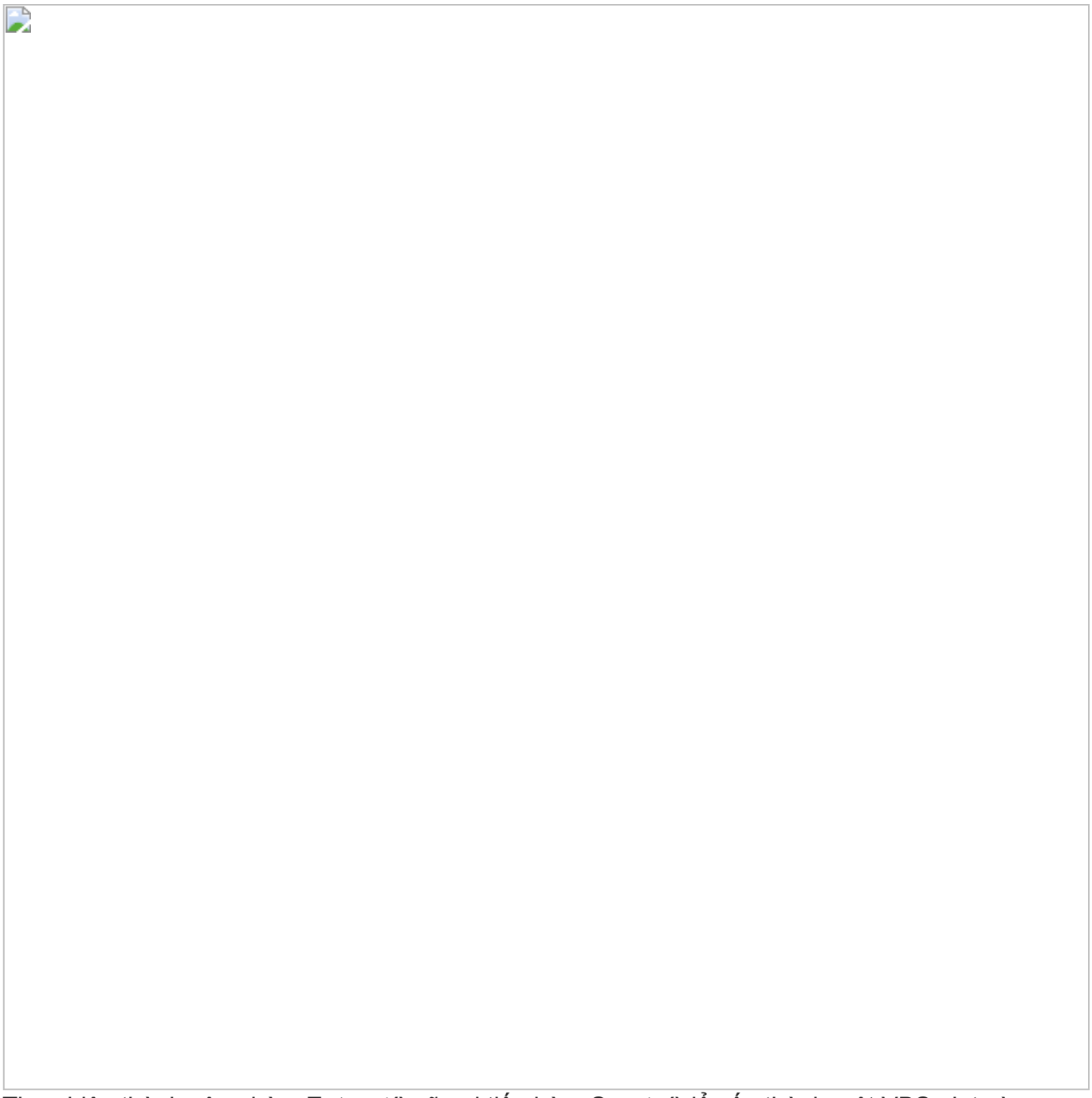
Các bạn có thể đọc code chay để hiểu xem VBA làm gì, nhưng tôi thích dùng tính năng Debug VBA code của Office.



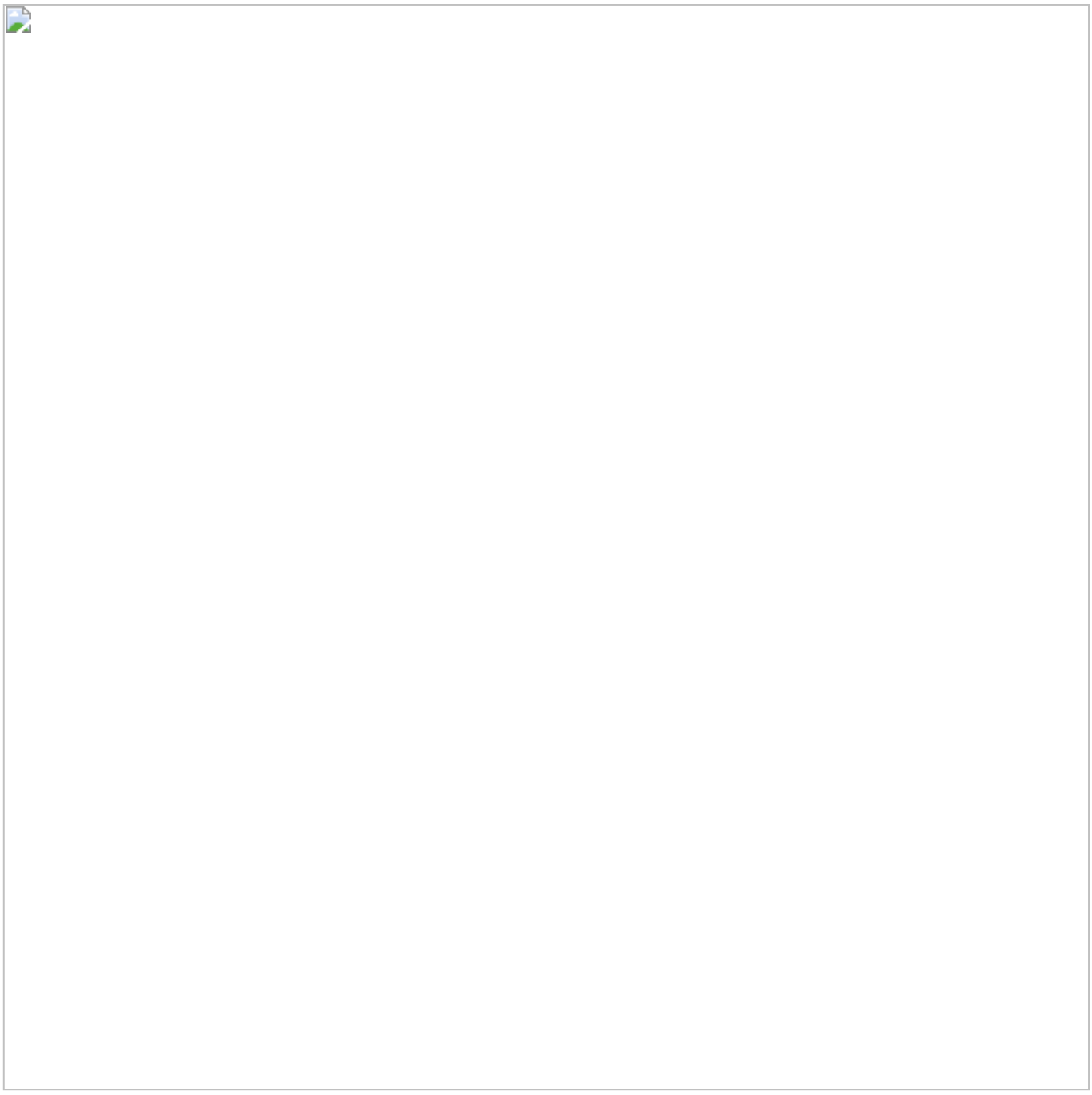
VBA code thực hiện trigger người dùng khi họ đóng tài liệu, nó sẽ gọi tới đoạn code sau:

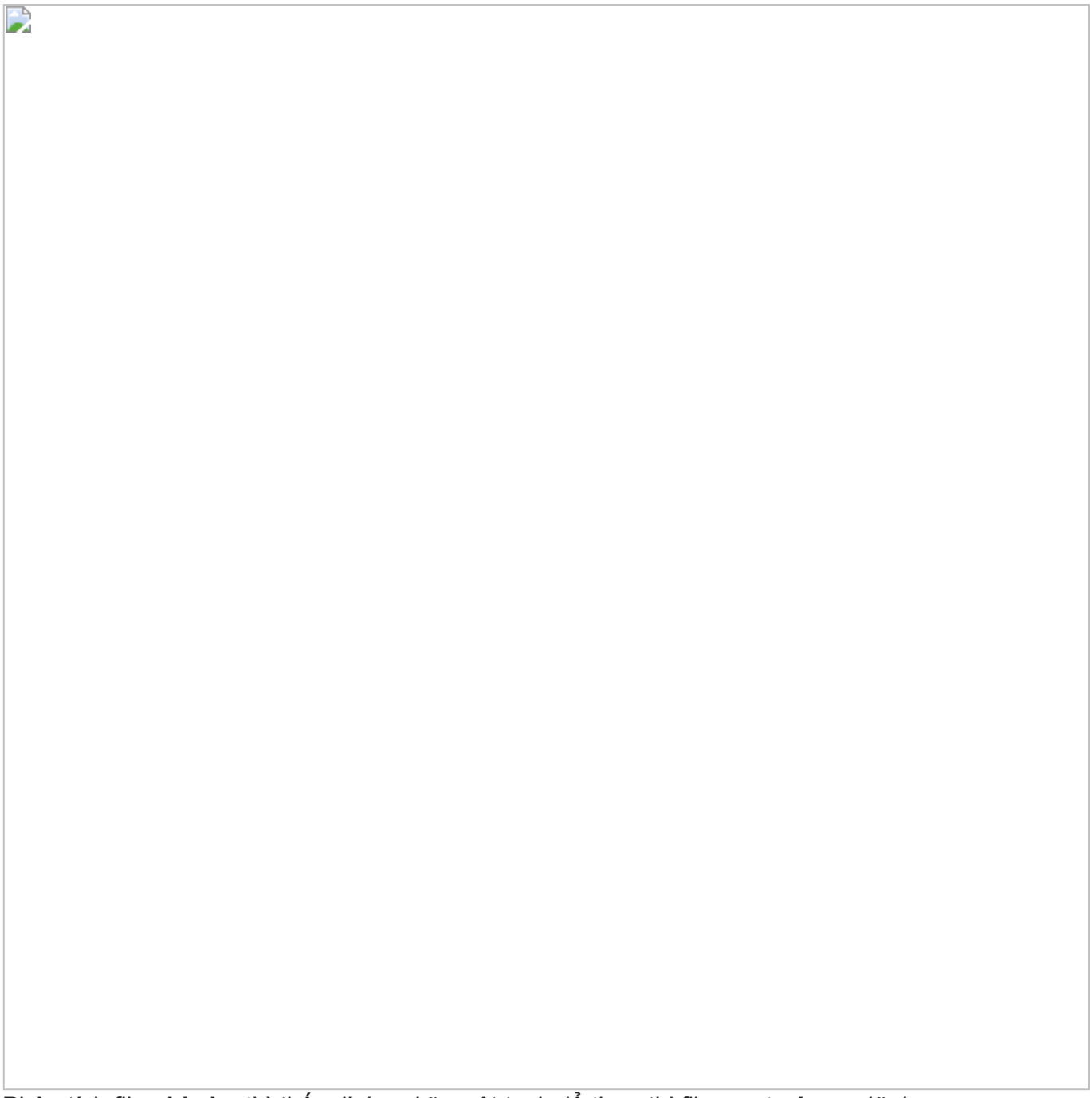


Đầu tiên sẽ gọi hàm **Extract()**, hàm này thực hiện nhiệm vụ drop ra một file dll có tên là **propsys.dll** (*a9483fffb2cc476837d42832df2d79c5*) và copy file **control.exe** (là *Windows Control Panel* của hệ thống) vào thư mục **%LOCALAPPDATA%**:



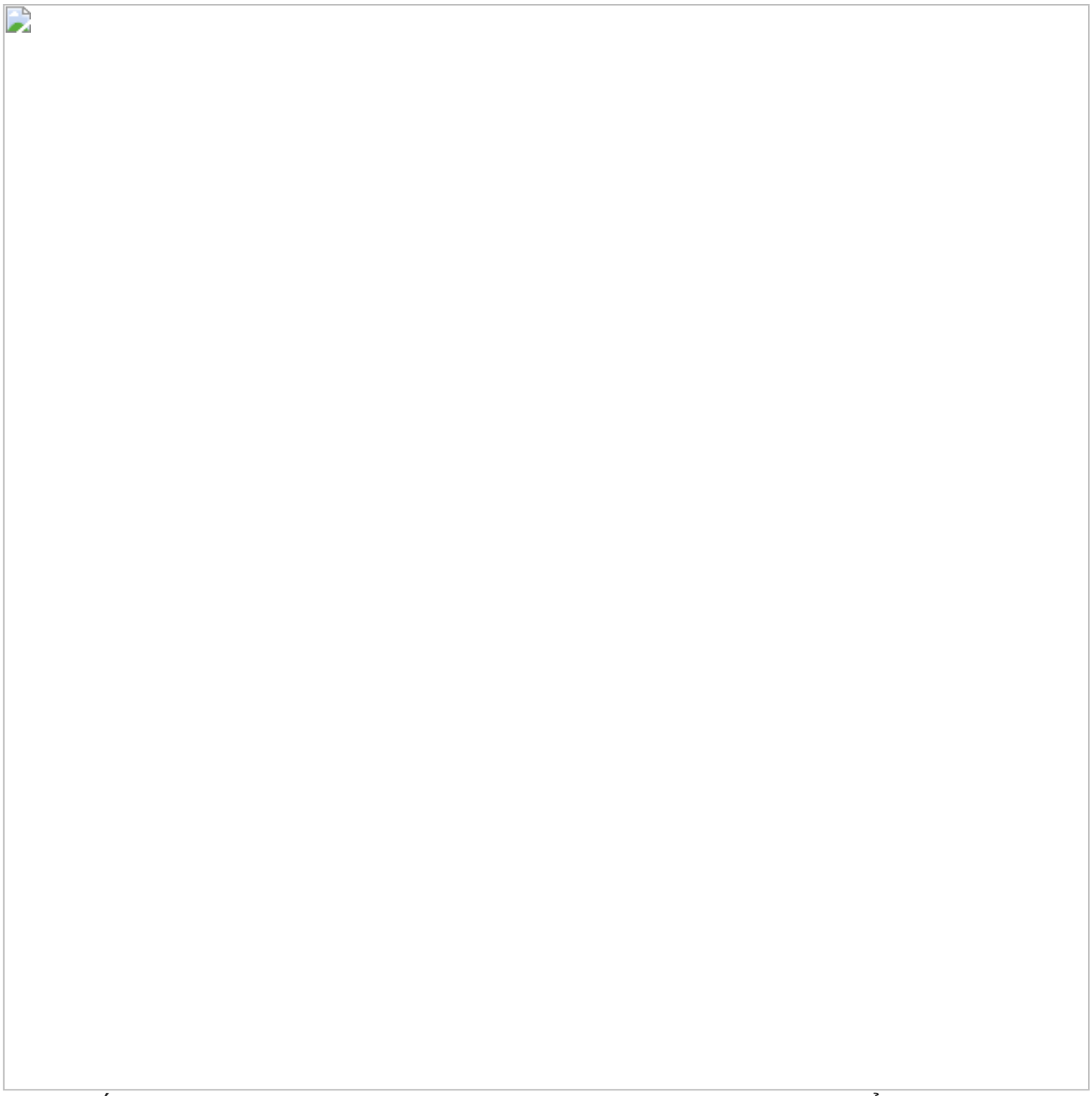
Thực hiện thành công hàm **Extract()** sẽ gọi tiếp hàm **Create()**để cấu thành một VBScript và lưu toàn bộ nội dung của script này vào file **abi.vbs (***612f6862f823a16736b1334daad3e810***)** tại thư mục **%APPDATA%**. Sau đó, sử dụng **cscript** để thực thi file **abi.vbs** vừa tạo:



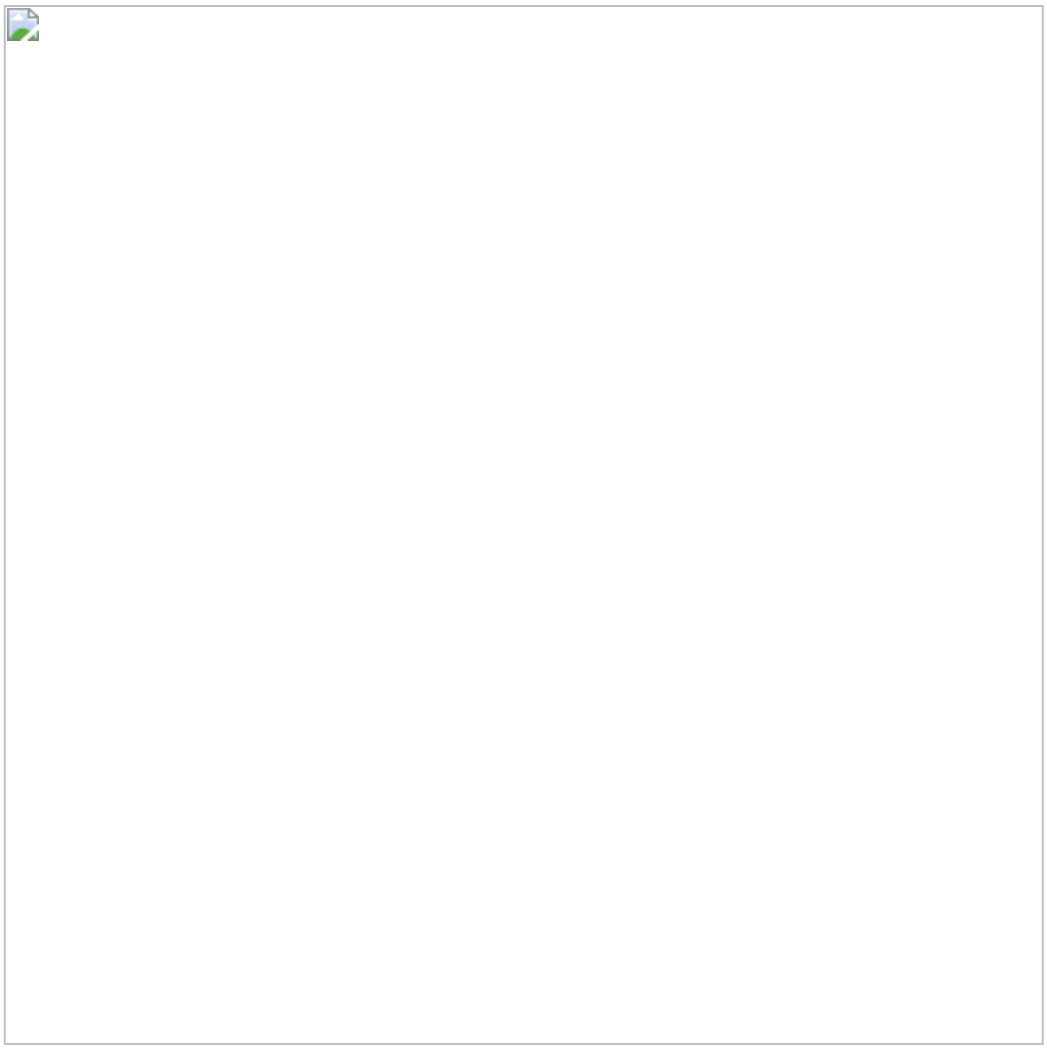


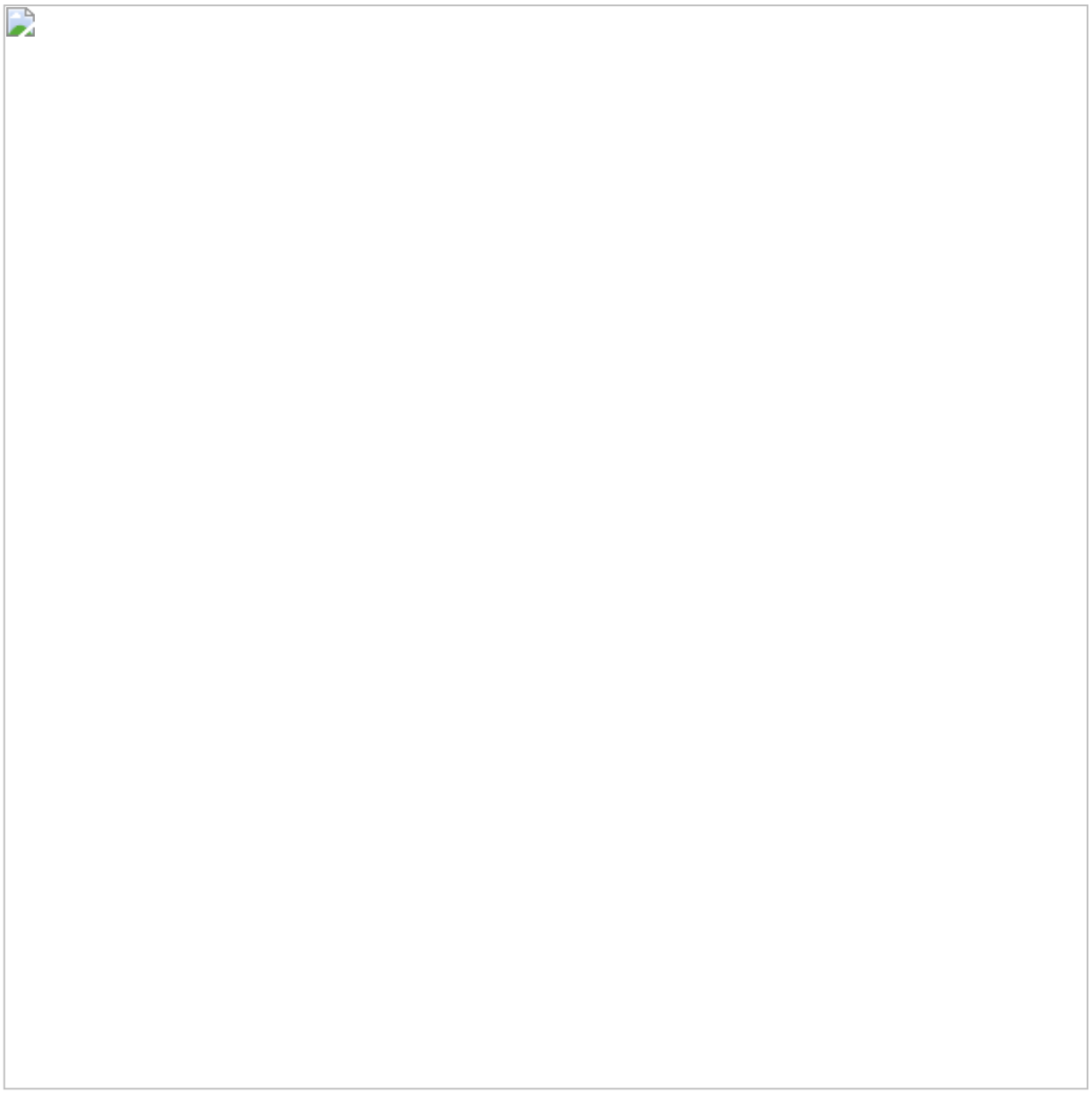
Phân tích file **abi.vbs** thì thấy định nghĩa một task để thực thi file **control.exe** đã được copy vào thư mục **%LOCALAPPDATA%:**

- Tạo thư mục tại
- Khai báo một task mới và cung cấp các thông tin liên quan:

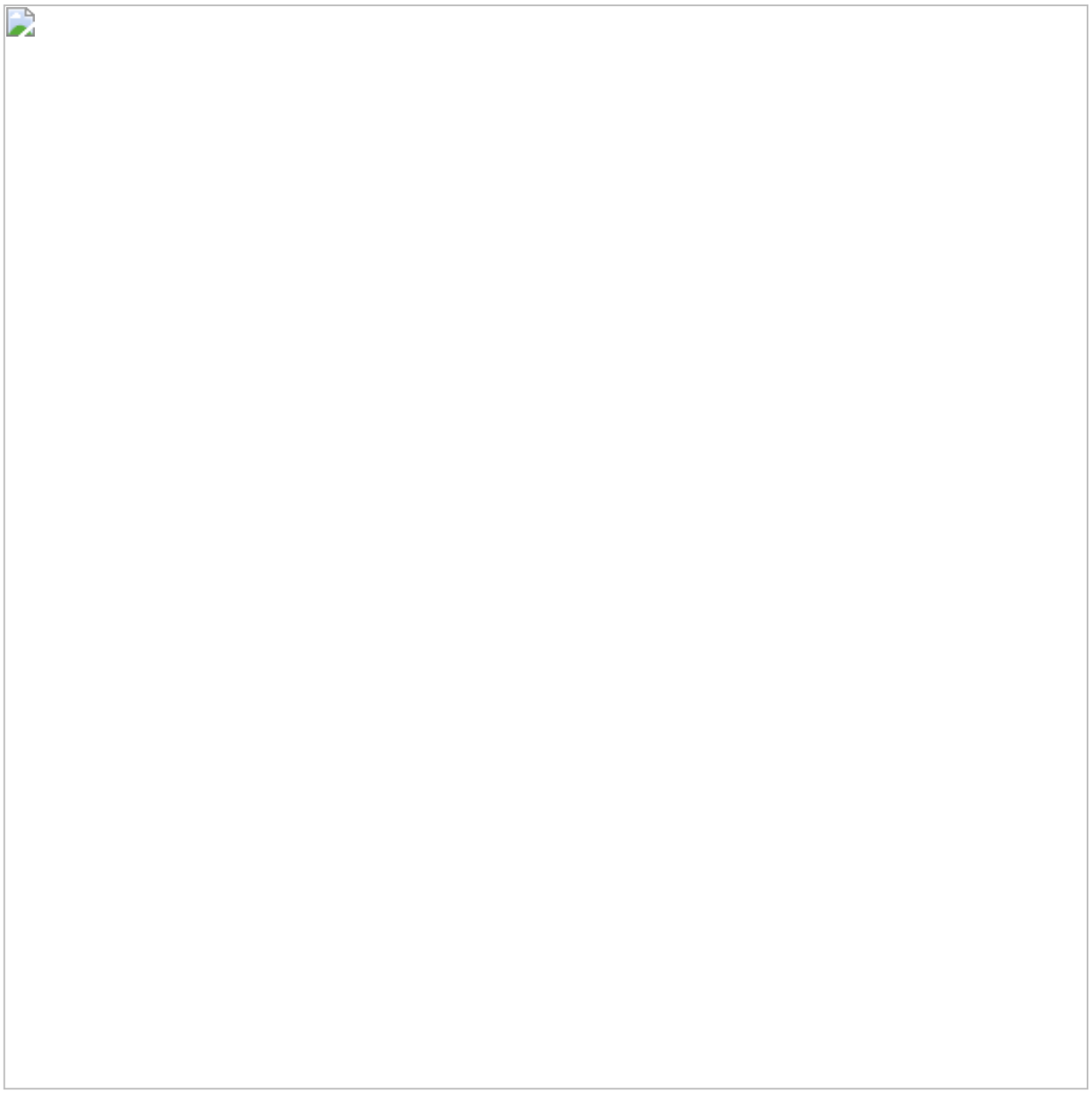


Thiết lập trigger cho việc thực thi và tạo ra một file là có định dạng XML để lưu thông tin về task:



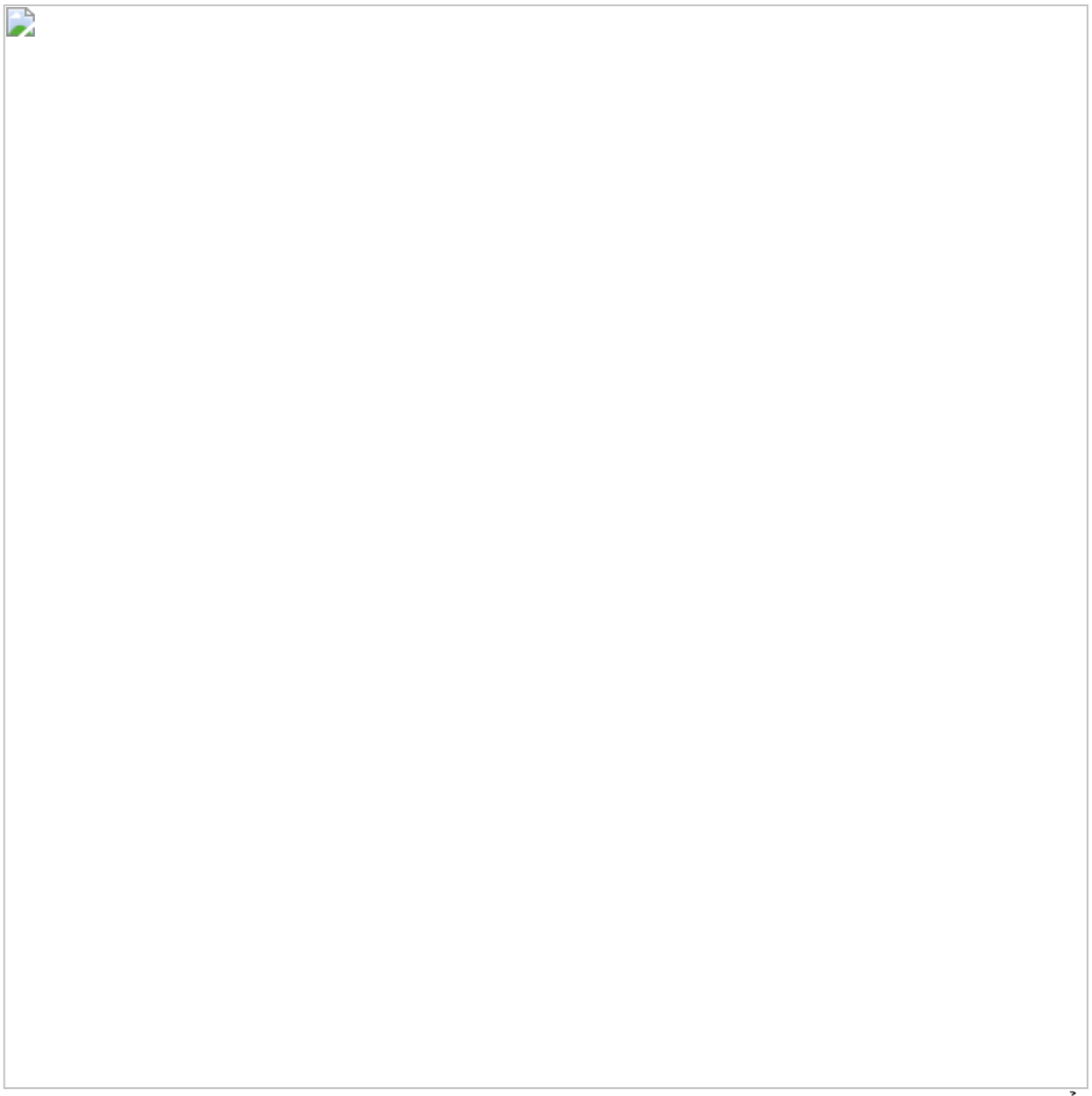


Scheduled task sau khi được tạo thành công sẽ tương tự như sau:

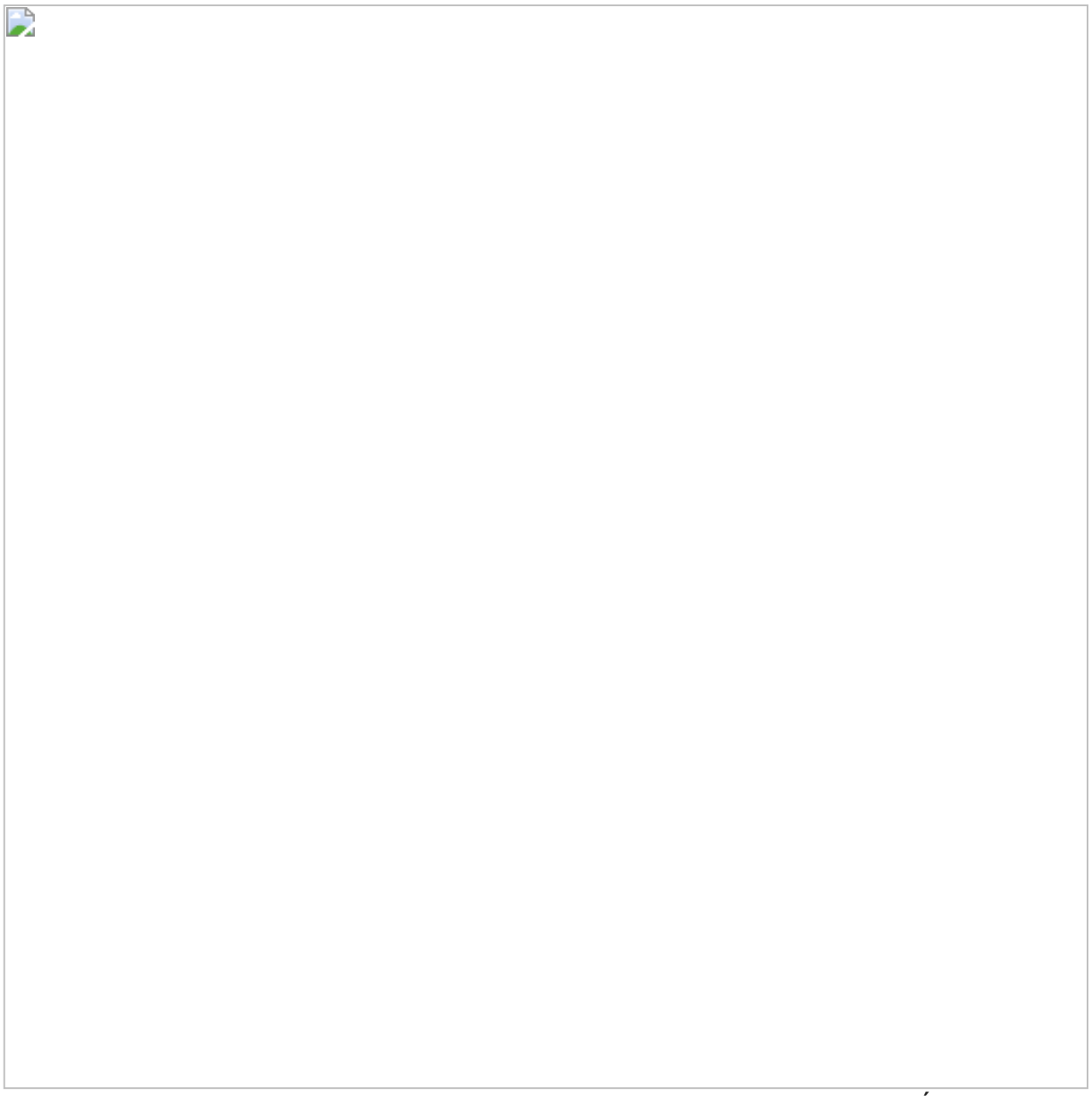


2. Phân tích sơ bộ propsys.dll (32-bit dll)

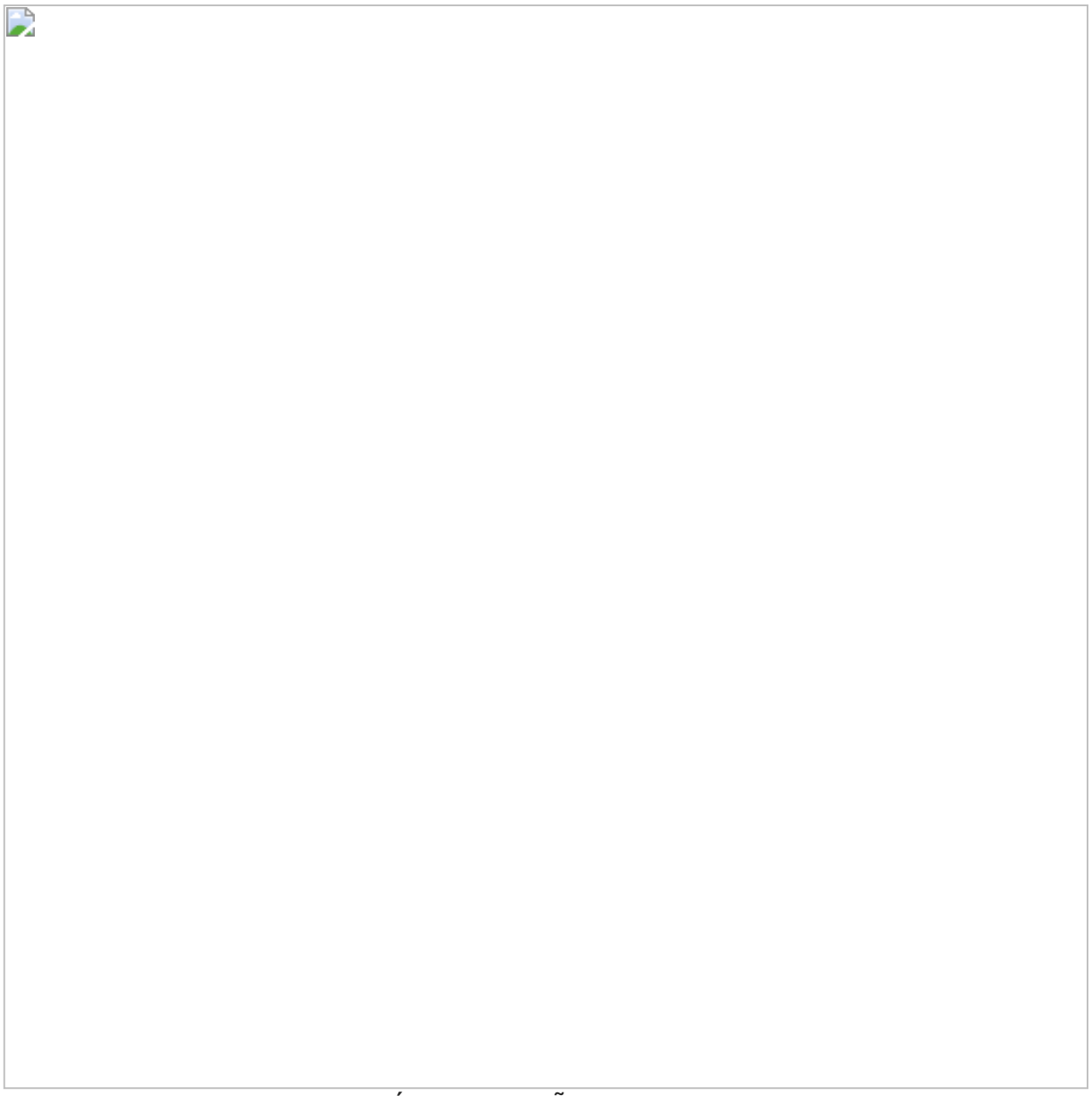
Dll này export một hàm duy nhất là **PSCreateMemoryPropertyStore**:



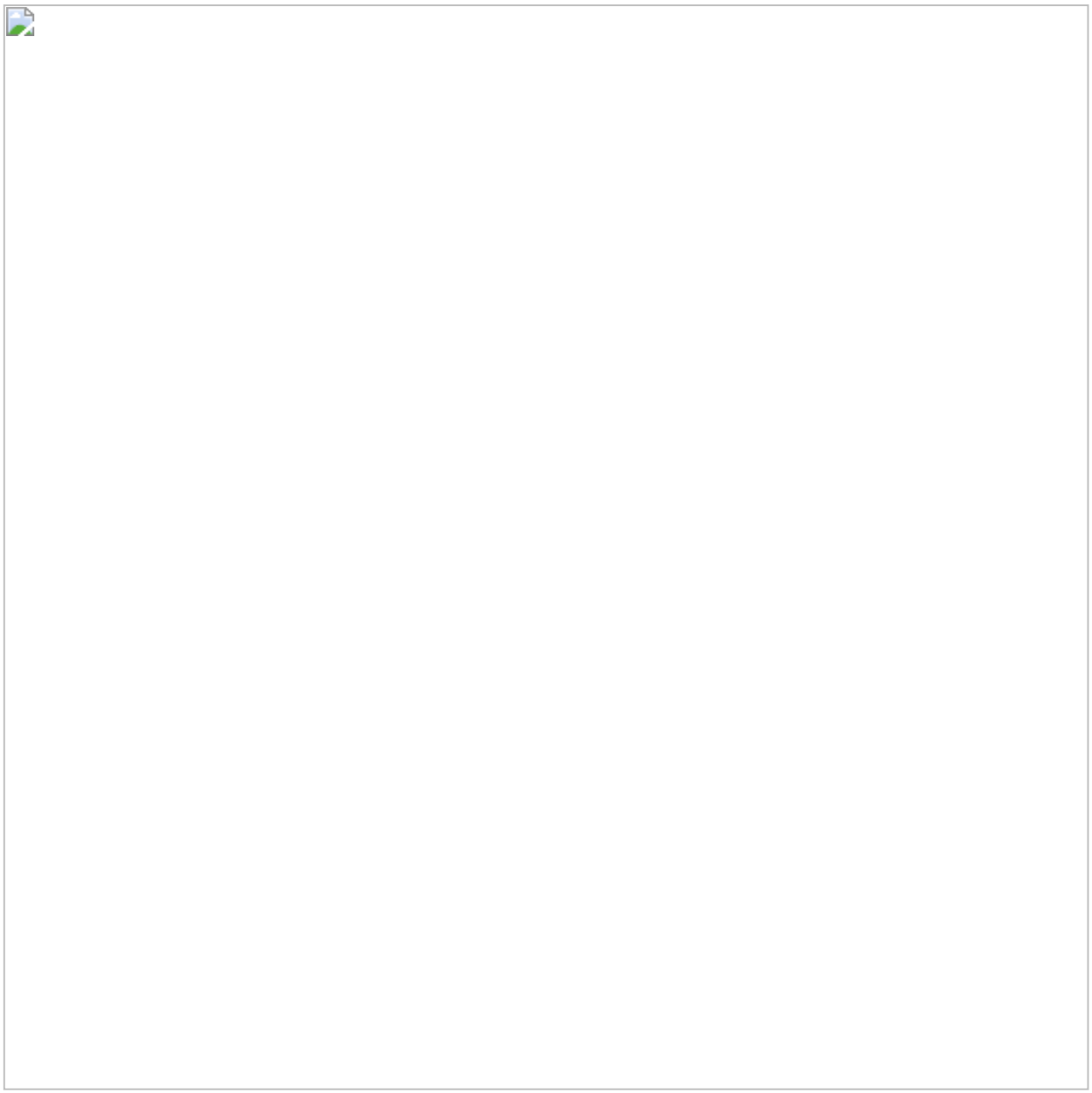
Có Import các hàm từ hai thư viện **kernel32.dll (67 funcs)** và **wininet.dll (4 funcs)**. Để ý các hàm được import từ thư viện **wininet.dll** ta thấy rằng dll này sẽ thực hiện kết nối tới Internet để đọc file nào đó:



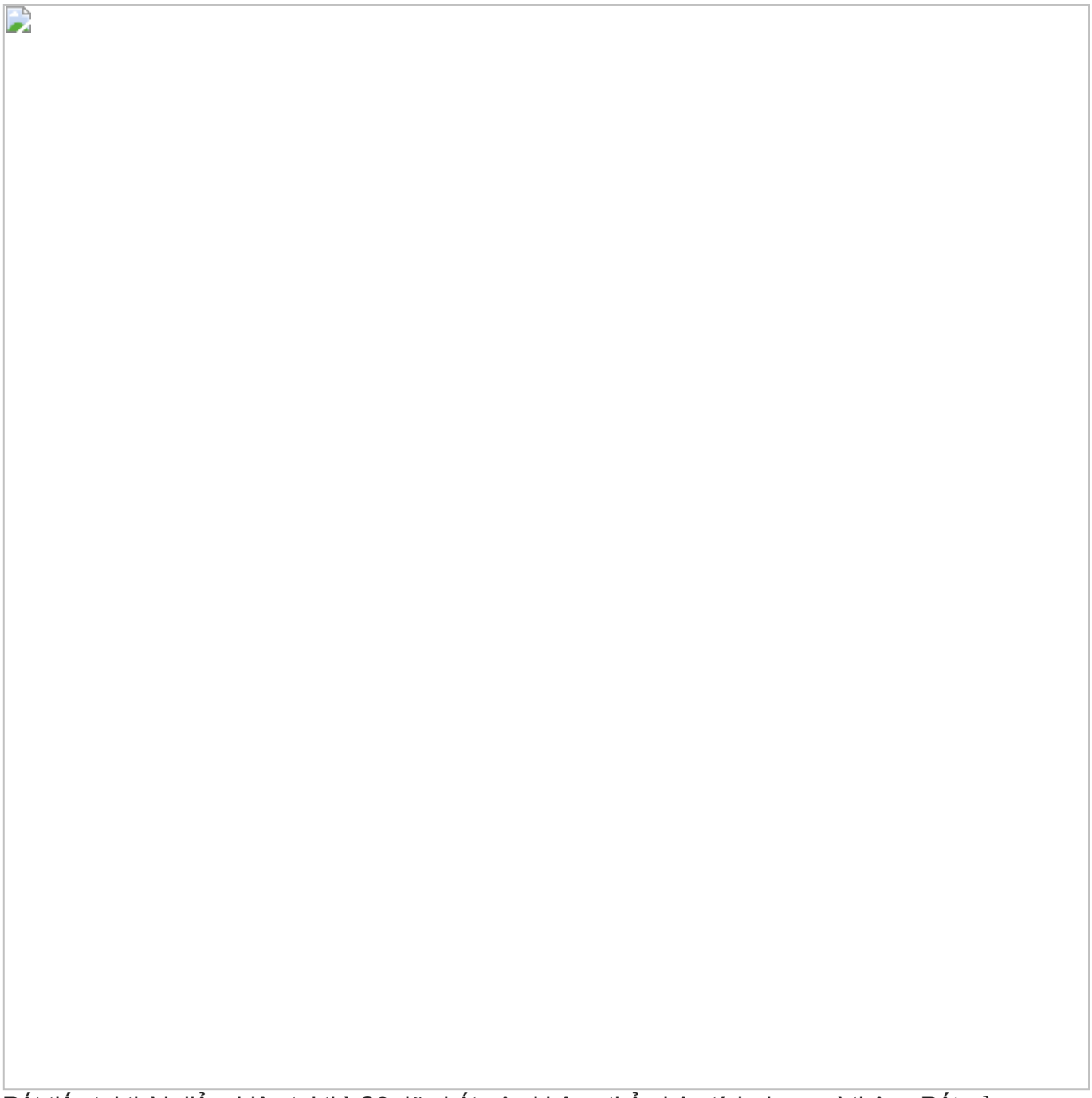
Tập trung vào hàm được export là **PSCreateMemoryPropertyStore**, ta thấy từ hàm này gọi tới các sub function là **sub_10001B90; sub_10001250; sub_10001770; sub_10001390**. Trong đó, đáng chú ý là **sub_10001250** vì nó gọi tới các hàm có truy xuất tới Internet:



sub_10001B90 làm nhiệm vụ cấu thành chuỗi URL (hxxps://sub[.]journeywiki[.]com/nancy.ico) trong memory:



Hàm **sub_10001250** làm nhiệm vụ kết nối tới URL trên với user agent là “**Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0**” để đọc file và lưu vào vùng **buf** có kích thước là **0x400** bytes:



Rất tiếc tại thời điểm hiện tại thì C2 đã chết nên không thể phân tích được gì thêm. Rất cảm ơn các bạn trong friend list đã chia sẻ cho tôi!

Sharing is caring!! :pray:

P/S: Bạn nào có thêm thông tin khác thì vui lòng để lại comment !