**Introduction**

Một lỗ hổng leo thang đặc quyền trong Windows được báo cáo với Microsoft vào ngày 9 tháng 9 năm 2022 bởi Andrea Pierini và Antonio Cocomazzi. Lỗ hổng cho phép kẻ tấn công có đặc quyền thấp trên máy chủ đọc/ghi bất kỳ tệp nào với đặc quyền của SYSTEM.

Microsoft đã phát hành 1 bản vá lỗ hổng vào tháng 1 năm 2023 và 1 PoC đã được phát hành sau đó vào 10 tháng hai năm 2023. Lỗ hổng được đặt là CVE-2023-21746.

Trong khi lỗ hổng không cho phép thực thi lệnh với đặc quyền của SYSTEM, nhưng có nhiều cách để làm được điều đó. May mắn thay, vào 13 tháng 2, một kỹ thuật leo thang đặc quyền khác được công bố bởi BlackArrowSec tận dụng dịch vụ StorSvc, cho phép kẻ tấn công thực thi lệnh với đặc quyền của hệ thống khi họ có thể ghi 1 file DLL đến 1 thư mục bất kỳ trong PATH.

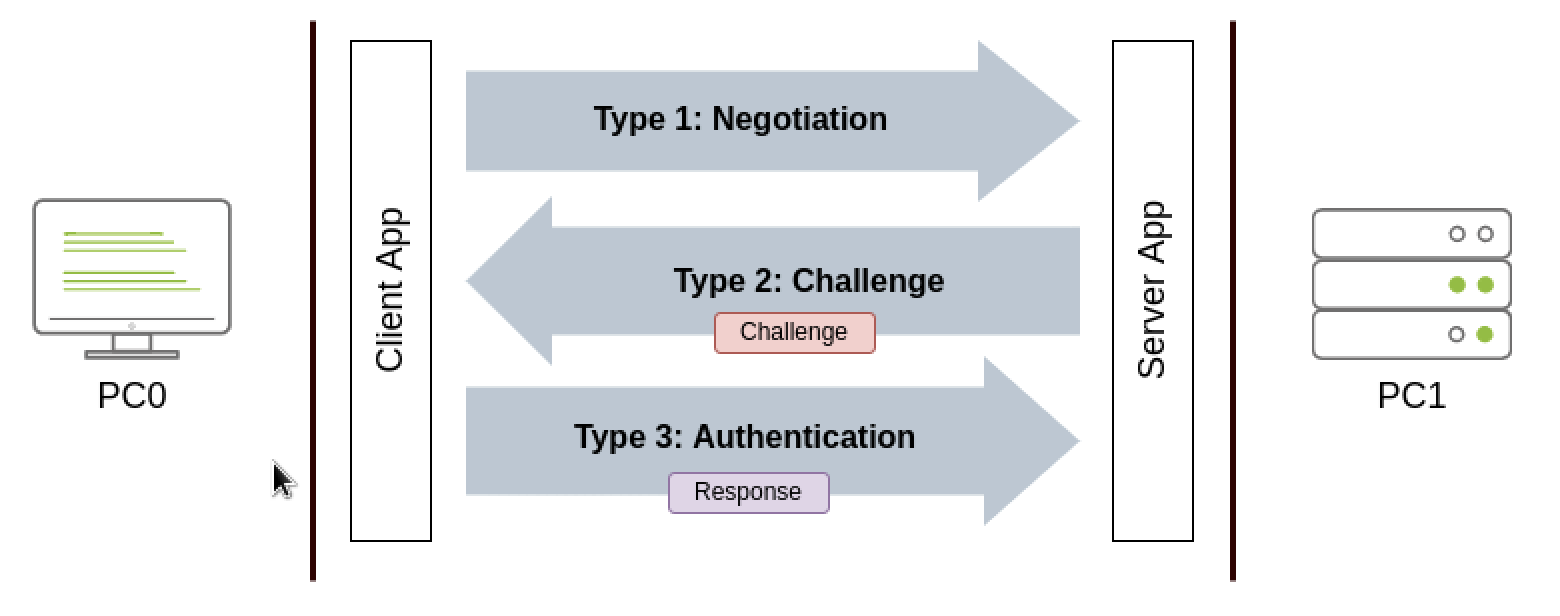
Trong phòng này, chúng ta sẽ áp dụng hai lỗ hổng và kết hợp chúng để thực thi lệnh bất kỳ trên hệ thống.

**NTLM Authentication Refresher**

**NTLM Authentication**

Trường hợp NTLM xác thực thông thường liên quan đến 1 người dùng cố gắng xác thực đến 1 máy chủ từ xa. Quá trình xác thực bao gồm 3 gói tin:

* **Type 1 Message:** máy khách gửi 1 gói tin để đàm phán các điều khoản của quá trình xác thực. Gói tin chứa tên của máy khách và tên miền. Máy chủ nhận gói tin và có thể kiểm tra xem xác thực đã được bắt đầu từ 1 máy ảo khác chưa.
* **Type 2 Message:** Máy chủ phản hồi máy khách với 1 challenge. “Challenge” là 1 con số ngẫu nhiên dùng để xác thực máy khách mà không cần phải truyền thông tin đăng nhập qua mạng.
* **Type 3 Message:** Máy khách dùng challenge nhận được từ Type 2 Message và kết hợp nó với password hash để tạo 1 phản hồi đến challenge. Phản hồi được gừi đến máy chủ là 1 phần của Type 3 Message. Bằng cách này, máy chủ có thể kiểm tra xem máy khách có biết password hash chính xác của người dùng không mà không cần truyền nó qua mạng.



**NTLM Local Authentication**

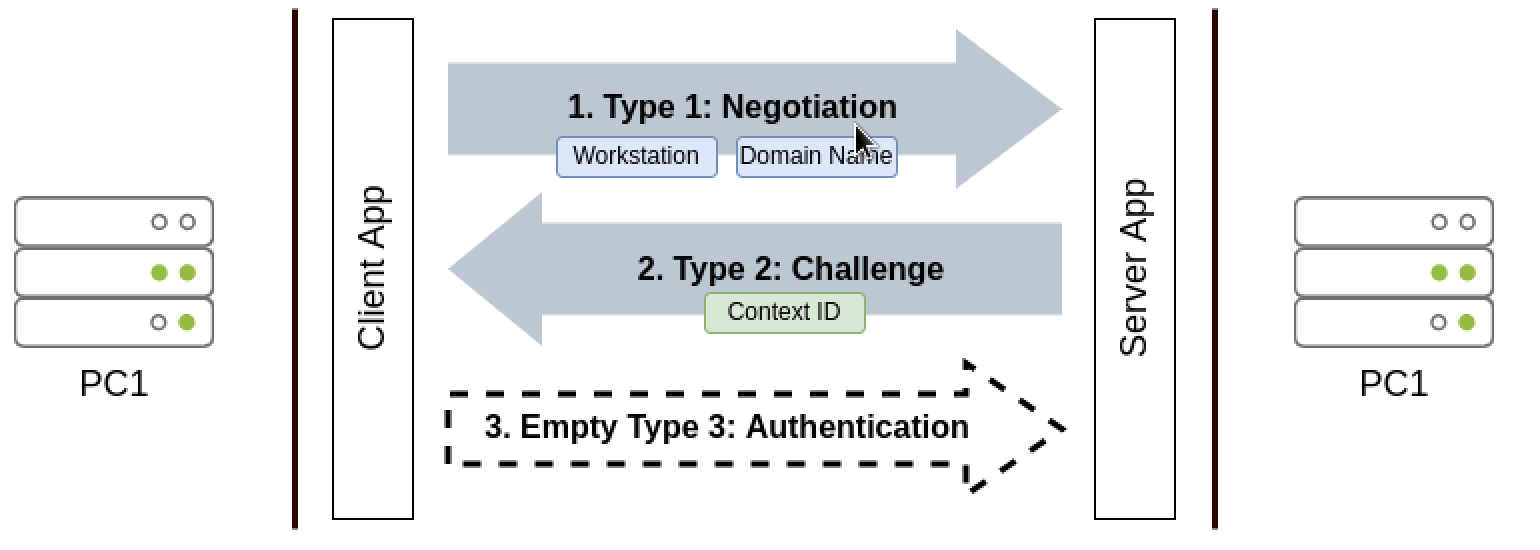
NTLM local authentication được dùng khi người dùng cố gắng đăng nhập 1 dịch vụ đang chạy trên cùng 1 máy ảo. Vì máy khách và máy chủ cùng nằm trên 1 máy ảo, quá trình challenge-response không cần thiết. Thay vào đó, xác thực được thực hiện khác đi bằng cách thiết lập 1 Security Context. Trong khi chúng ta không đào sâu chi tiết về cái được chứa trong Security Context, nghĩ nó giống như 1 bộ các tham số bảo mật liên quan đến 1 kết nối bao gồm session key và người dùng mà đặc quyền của họ được dùng cho kết nối đó.

Quá trình xác thực sẽ giống 3 tin nhắn trước đó, nhưng thông tin để xác thực sẽ thay đổi như sau;

Type 1 Message: máy khách gửi tin nhắn này để bắt đầu kết nối. Nó được dùng để đàm phán các tham số xác thực như trước đó nhưng cũng chứa tên máy ảo và tên miền. Máy chủ có thể kiểm tra tên máy ảo và tên miền và quá trình local authentication sẽ bắt đầu nếu thông tin trên khớp với nhau

Type 2 Message: máy chủ tạo 1 Security Context và gửi lại ID của nó cho máy khách trong tin nhắn này. Máy khách sau đó có thể dùng ID của Security Context để ánh xạ nó với kết nối.

Type 3 Message: nếu máy khách ánh xạ thành công với Security Context ID đang có, 1 Type 3 Message rỗng sẽ gửi lại máy chủ để chứng minh rằng quá trình local authentication đã thành công.



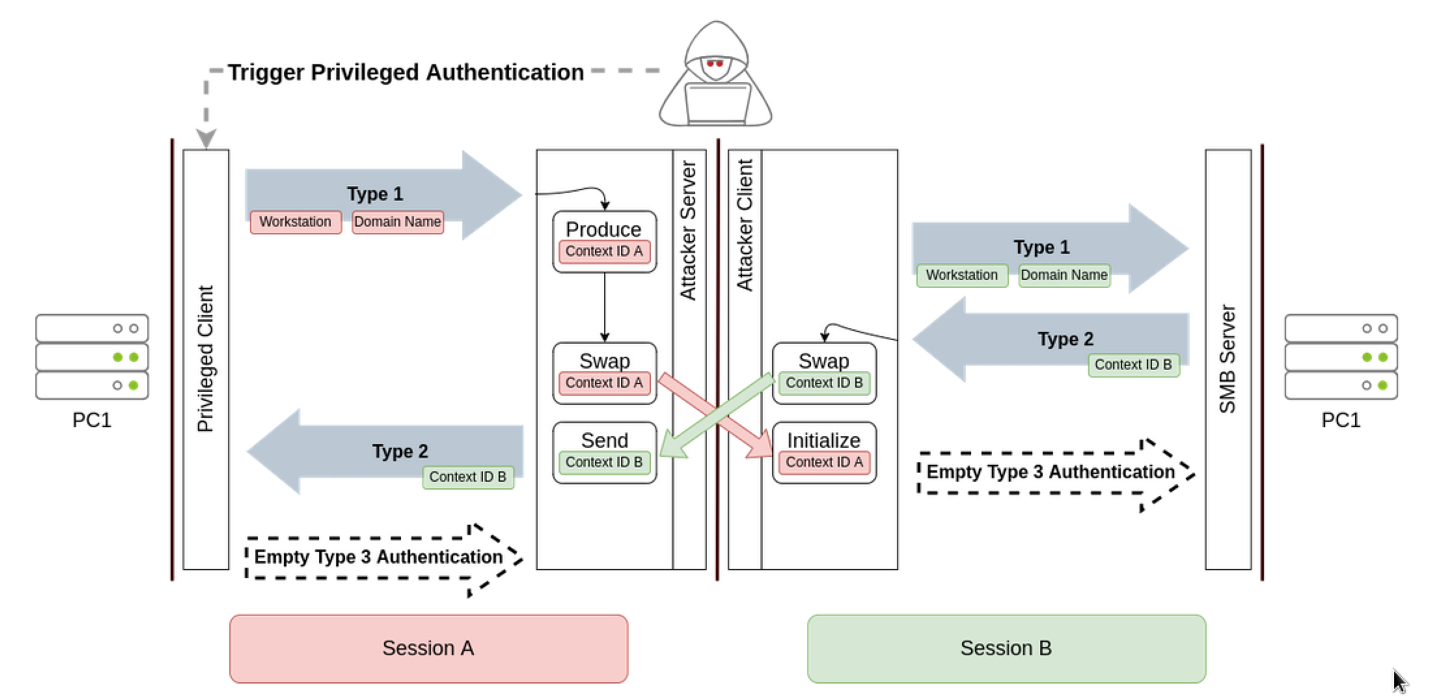
Vì tất cả các bước xảy ra trong cùng 1 máy ảo, không cần thiết dùng phương thức challenge-response như trước đó. Máy ảo có thể xác thực Security Context ID cho các ứng dụng máy khách và máy chủ.

**LocalPotato**

LocalPotato PoC lợi dụng lỗ hổng trong trường hợp xác thực NTLM đặc biệt, gọi là NTLM local authentication để lừa 1 tiến trình có đặc quyền xác thực phiên mà kẻ tấn công bắt đầu dựa vào máy chủ SMB. Kết quả là, kẻ tấn công có được kết nối cấp cho anh ta quyền truy cập vào bất cứ shares nào với đặc quyền của tiến trình trên, bao gồm các Share nhạy cảm như C$ hay ADMIN$.

Quá trình khai thác như sau:

1. Kẻ tấn công sẽ lừa 1 tiến trình có đặc quyền kết nối đến 1 máy chủ rogue. Điều này hoạt động tương tự như các khai thác Potato trước, 1 người dùng không có đặc quyền buộc hệ điều hành tạo 1 kết nối bằng tiến trình có đặc quyền (SYSTEM).
2. Máy chủ rogue sẽ khởi tạo 1 Security Context A cho kết nối có đặc quyền nhưng sẽ không gửi lại ngay lập tức. Thay vào đó, kẻ tấn công sẽ khởi động 1 rogue client, khởi tạo 1 kết nối đến máy chủ SMB với thông tin đăng nhập không có đặc quyền. Máy khách sẽ gửi Type1 Message để khởi tạo kết nối và máy chủ sẽ phản hồi bằng cách gửi Type2 Message với ID của Security Context B.
3. Kẻ tấn công sẽ tráo đổi Context Ids từ cả hai kết nối để mà tiến trình có đặc quyền nhận được bối cảnh của kết nối SMB server thay vì của chính nó. Kết quả là, máy khách của kẻ tấn công có thể truy cập bất cứ Share nào với đặc quyền của SYSTEM.



Có kết nối đặc quyền đến SMB shares, kẻ tấn công có thể đọc hoặc ghi tệp trong hệ thống đích ở bất kỳ vị trí nào. Điều này không cho phép ta thực thi lệnh trong máy ảo, chúng ta sẽ kết hợp nó với 1 kỹ thuật khác để đạt được mục đích đó.

Lưu ý là lỗ hổng ở trong giao thức NTLM thay vì máy chủ SMB, vì vậy cách khai thác này có thể áp dụng cho bất kỳ dịch vụ nào xác thực qua NTLM. Trong thực tế, bạn phải xử lí một số lớp phòng thủ khi chọn giao thức để tấn công. PoC khai thác máy chủ SMB để tránh một số biện pháp bảo vệ bổ sung áp dụng cho các giao thức khác, thậm chí phải thực thi các bypass để khai thác hoạt động được trên máy chủ SMB. Chúng ta sẽ không đi sâu vào kĩ thuật đó trong lab, bạn có thể đọc thêm trong bài viết gốc của tác giả.

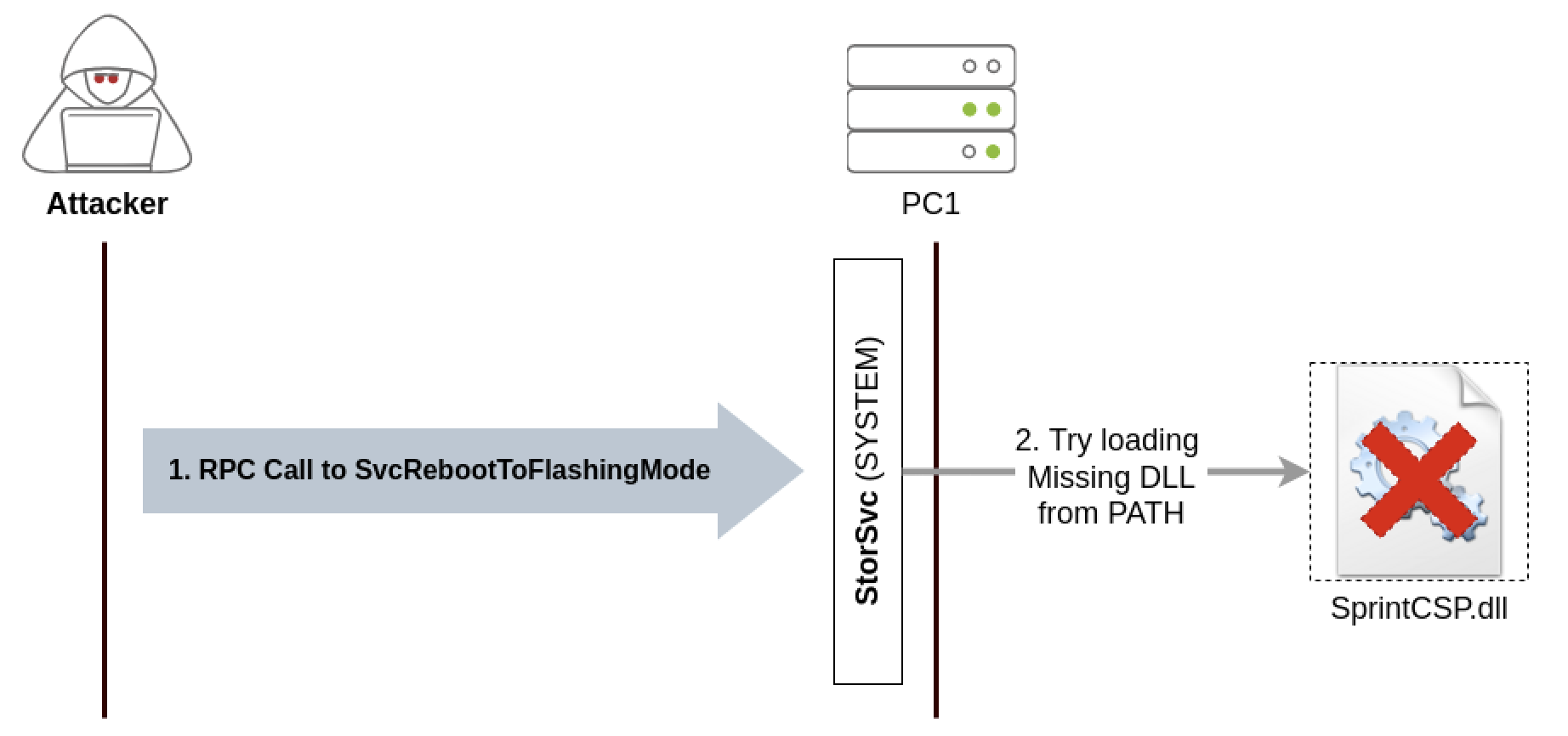
**Lợi dụng StorSvc để thực thi lệnh**

Chúng ta đã dùng LocalPotato để ghi tệp bất kỳ trong hệ thống đích. Để shell có đặc quyền, chúng ta cần tìm cách để thực thi lệnh.

Một kỹ thuật leo thang đặc quyền được tìm thấy, kẻ tấn công có thể chiếm quyền điều khiển DLL bị thiếu để chạy các lệnh tùy ý với đặc quyền của SYSTEM. Vấn đề duy nhất là phải ghi 1 DLL vào trong biến PATH của hệ thống để thực thi. Mặc định, Windows PATH chứa các thư mục chỉ các tài khoản có đặc quyền mới được ghi. Mặc dù có thể tìm thấy các máy trong đó việc cài đặt các ứng dụng cụ thể đã thay đổi biến PATH và khiến máy dễ bị tấn công, cách tấn công chỉ áp dụng cho các tình huống cụ thể. Kết hợp cách tấn công này với LocalPotato cho phép chúng ta khắc phục hạn chế này và khai thác thành công.

**StorSvc và DLL Hijacking**

Kẻ tấn công có thể gửi 1 RPC call đến phương thức SvcRebooToFlashingMode được cung cấp bởi dịch vụ StorSvc, do đó sẽ kích hoạt được nỗ lực tải 1 DLL bị thiếu gọi là SprintCSP.dll



Nếu bạn không quen với RPC, hãy nghĩ nó giống như 1 API hiển thị các chức năng để chúng ta có thể gọi từ xa. Trong trường hợp này, dịch vụ StorSvc hiển thị phương thức SvcRebootToFlashingMode, bất cứ ai truy cập máy ảo có thể gọi được.

Vì StorSvc chạy với đặc quyền của SYSTEM, việc tạo SprintCSP.dll ở đâu đó trong PATH sẽ tải nó bất cứ khi nào lệnh gọi tới SvcRebootToFlashingMode được thực hiện.

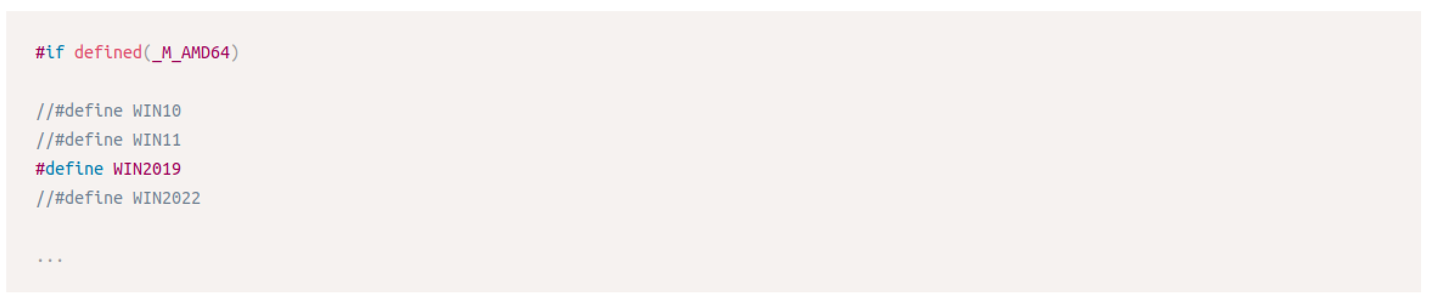
**Phân tích khai thác**

Để dùng khai thác này, đầu tiên bạn cần biên dịch hai tập tin sau:

* SprintCSP.dll: Đây là DLL bị thiếu mà chúng ta sẽ chiếm quyền điều khiển. Code trong khai thác sẽ chạy lệnh whoami và đầu ra nằm ở C:\Program Data\whoamiall.txt. Chúng ta sẽ cần thay đổi lệnh để chạy reverse shell.
* RpcClient.exe: Chương trình này sẽ thực hiện RPC call đến phương thức SvcRebootToFlashingMode. Dựa vào phiên bản của Windows mà bạn nhắm đến, bạn có thể cần điều chỉnh khai thác một chút vì các phiên bản khác nhau của Windows dùng các mã định danh của giao diện khác nhau để gọi SvcRebootToFlashingMode

Bạn có thể tìm thấy 2 tập tin này ở C:\tools\LPE via StorSvc\

Bắt đầu xử lí với RpcClient.exe. Chúng ta sẽ cần thay đổi khai thác dựa vào phiên bản của Windows. Để làm điều đó, chúng ta sẽ cần thay đổi dòng đầu tiên của C:\tools\LPE via StorSvc\RpcClient\RpcClient\storsvc\_c.c để chọn đúng hệ điều hành. Vì máy ảo chạy Windows Server 2019, chúng ta sẽ chỉnh sửa tập tin như sau:

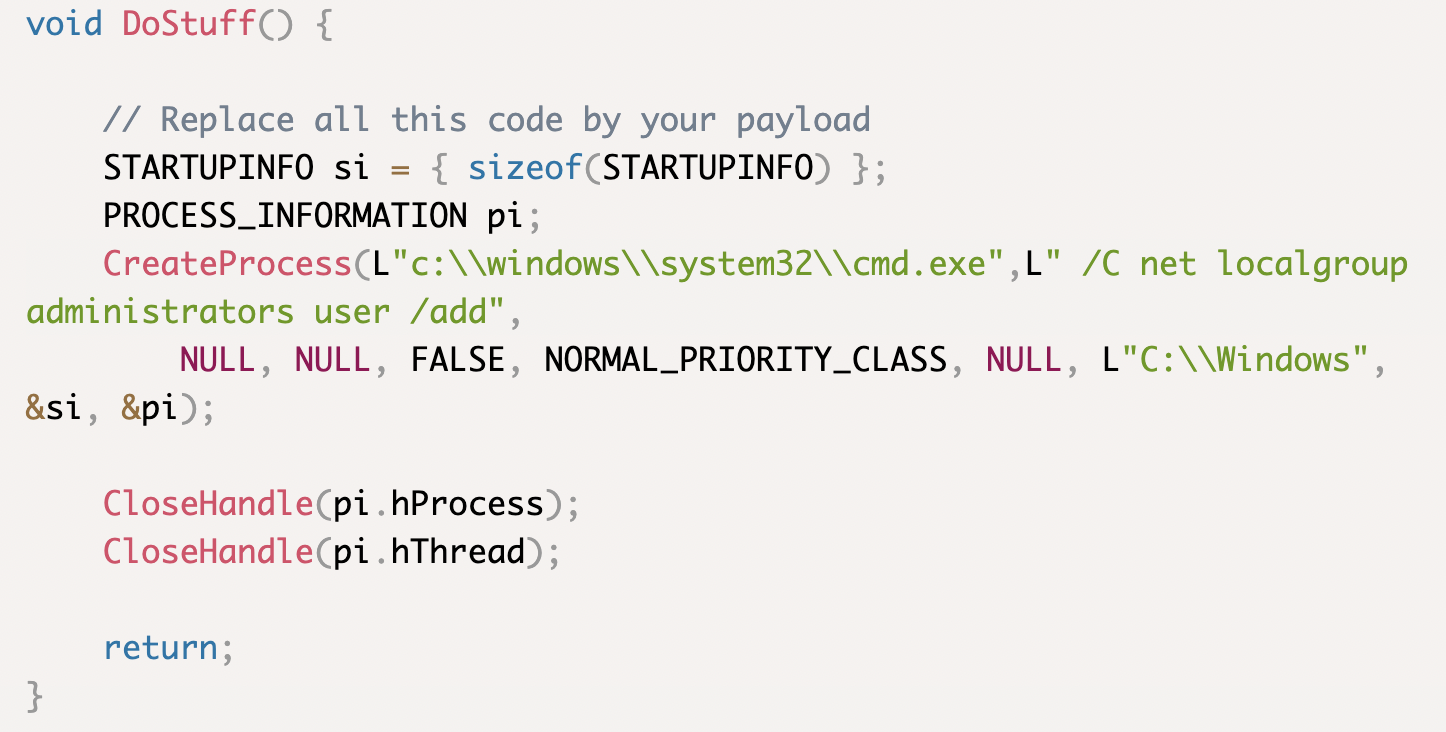


Khai thác sẽ dùng đúng RPC cho Windows 2019. Hãy mở command prompt của lập trình viên bằng shortcut trên desktop. Chúng ta sẽ xây dựng project bằng cách chạy lệnh sau:



Tệp thực thi được tìm thấy trên desktop.

Biên dịch SprintCSP.dll, chúng ta điều chỉnh hàm Dostuff() trong C:\tools\LPE via StorSvc\SprintCSP\SprintCSP\main.c để nó cấp quyền truy cập đến máy ảo. Để đơn giản, chúng ta sẽ làm DLL thêm người dùng hiện tại vào nhóm Administrators. Đây là tập lệnh với câu lệnh được thay thế:



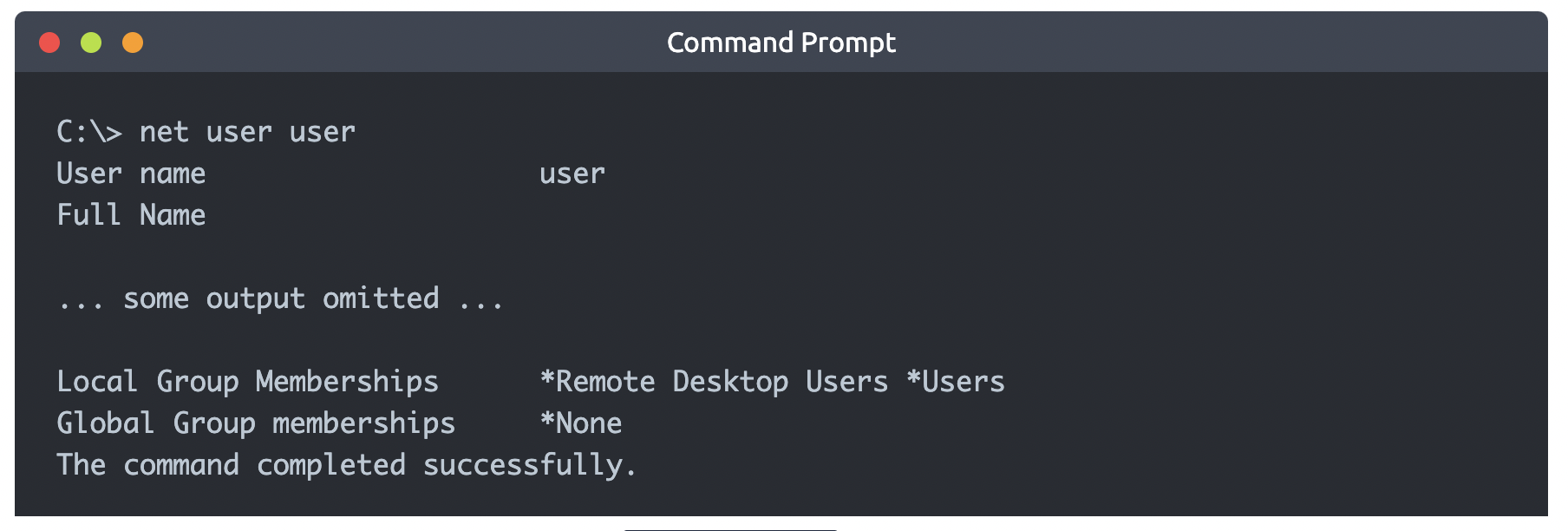
Chúng ta sẽ biên dịch DLL và di chuyển nó đến desktop.



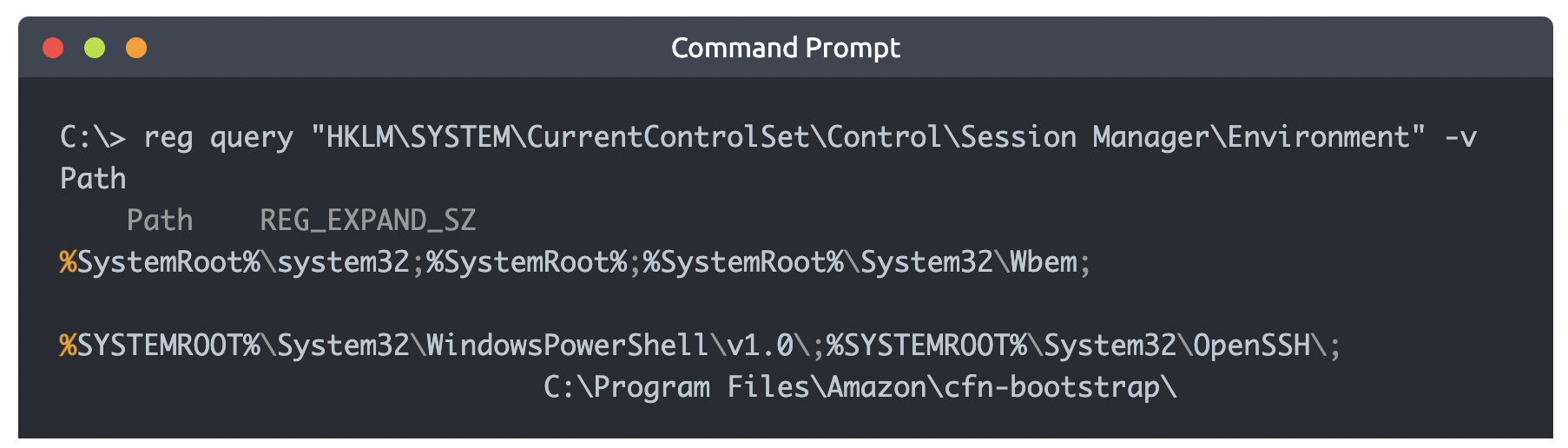
**Elevating our Privileges**

Chúng ta sẽ khởi động khai thác. Hãy chắc chắn bạn có khai thác LocalPotato.exe, RpcClient.exe và SprintCSP.dll trên desktop trước khi tiếp tục.

Kiểm tra người dùng hiện tại không trong nhóm Administrators:

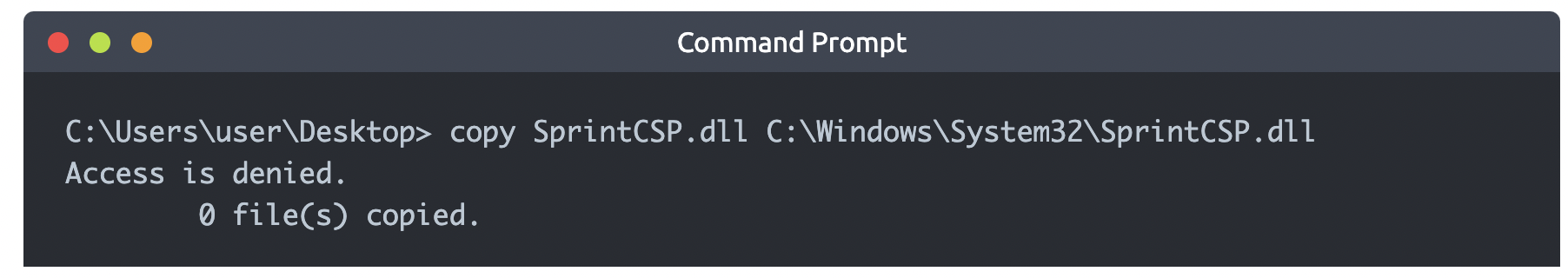


Để thành công khai thác StorSvc, chúng ta cần sao chép SprintCSP.dll đến thư mục bất kỳ trong PATH. Kiểm tra PATH bằng lệnh sau:

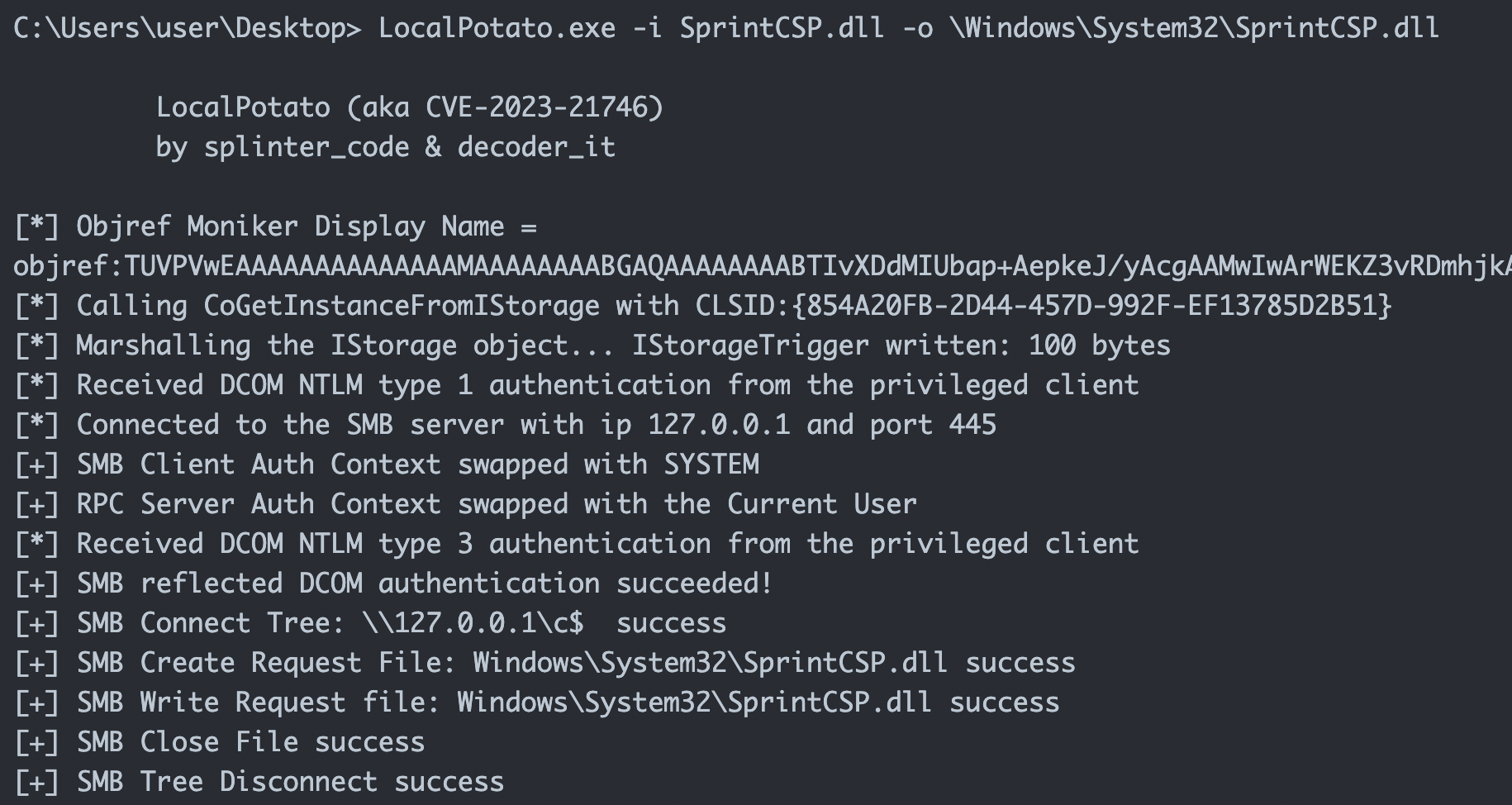


Chúng ta sẽ nhắm đến thư muc %SystemRoot%\system32, mở rộng là C:\windows\system32

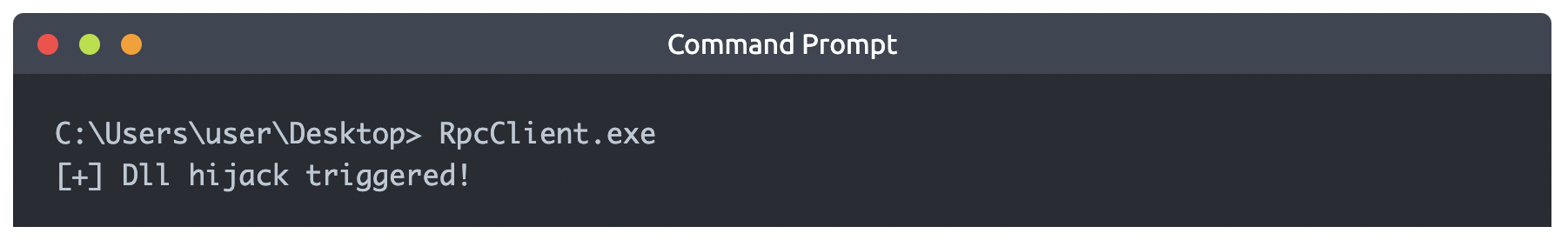
Để chắc chắn, chúng ta sao chép DLL trong system32, nhưng người dùng hiện tại không có đủ đặc quyền để làm:



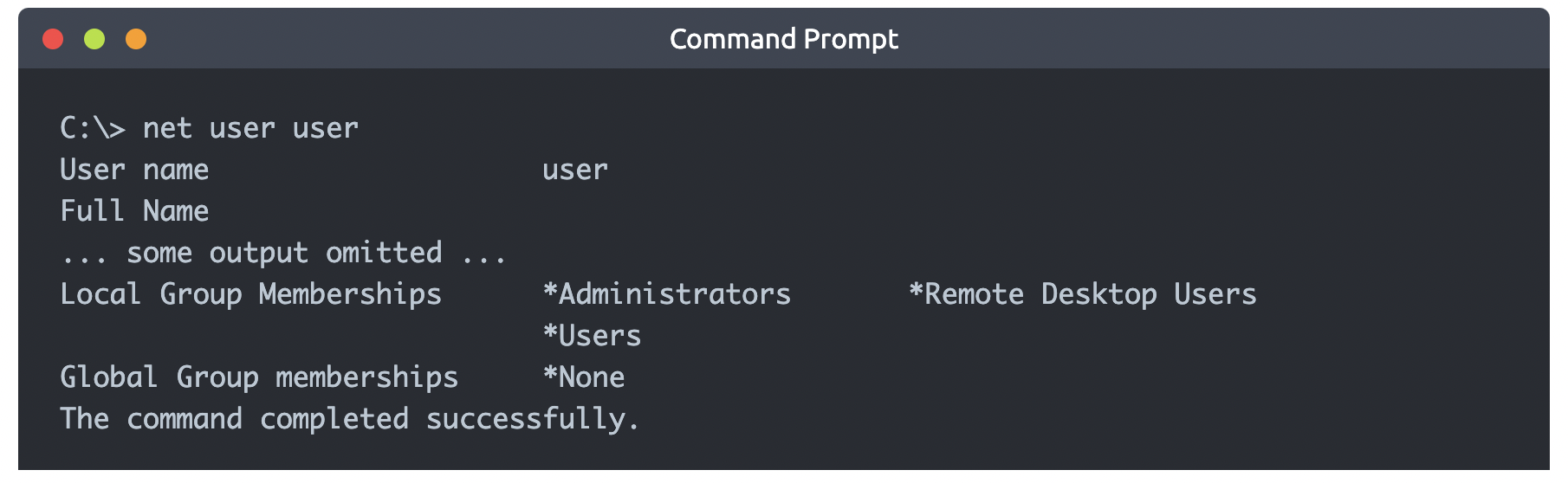
Bằng cách dùng LocalPotato, chúng ta có thể sao chép SprintCSP.dll vào system32:



Chúng ta có thể chạy RpcClient.exe để gọi đến hàm SvcRebootToFlashingMode, thực thi payload trong DLL.



Để xác thực khai thác như mong đợi, chúng ta có thể kiểm tra người dùng có thuộc nhóm Administrators:



Để mở command prompt với đặc quyền của quản trị viên, bạn bấm chuột phải và Run as administrator. Nhập thông tin đăng nhập là user-Password123

