Họ và tên: Ngô Huỳnh Vĩnh Phú

Mssv: N20DCAT043

Lớp: D20CQAT01-N

Lab07-01

Câu 1:

```
loc_401064:
push    esi
push    offset Name      ; "HGL345"
push    0                ; bInitialOwner
push    0                ; lpMutexAttributes
call    ds:CreateMutexA
push    3                ; dwDesiredAccess
push    0                ; lpDatabaseName
push    0                ; lpMachineName
call    ds:OpenSCManagerA
mov     esi, eax
call    ds:GetCurrentProcess
lea     eax, [esp+404h+Filename]
push    3E8h             ; nSize
push    eax              ; lpFilename
push    0                ; hModule
call    ds:GetModuleFileNameA
push    0                ; lpPassword
push    0                ; lpServiceStartName
push    0                ; lpDependencies
push    0                ; lpdwTagId
lea     ecx, [esp+414h+Filename]
push    0                ; lpLoadOrderGroup
push    ecx              ; lpBinaryPathName
push    0                ; dwErrorControl
push    2                ; dwStartType
push    10h              ; dwServiceType
push    2                ; dwDesiredAccess
push    offset DisplayName ; "Malservice"
push    offset DisplayName ; "Malservice"
push    esi              ; hSCManager
call    ds:CreateServiceA
xor     edx, edx
lea     eax, [esp+404h+FileTime]
mov     dword ptr [esp+404h+SystemTime.wYear], edx
lea     ecx, [esp+404h+SystemTime]
mov     dword ptr [esp+404h+SystemTime.wDayOfWeek], edx
push    eax              ; lpFileTime
mov     dword ptr [esp+408h+SystemTime.wHour], edx
```

Chương trình tạo một dịch vụ mang tên Malservice. Chương trình sẽ đặt dwStartType thành 2h (SERVICE_AUTO_START) và dwServiceAccess thành 10h(SERVICE_WIN32_OWN_PROCESS). Ngoài ra, Đường dẫn nhị phân được đặt thành đường dẫn của tệp thực thi của quy trình hiện tại.

Câu 2:

Chương trình này sử dụng Mutex để đảm bảo rằng chỉ một phiên bản của tệp này chạy cùng lúc:

Nếu OpenMutexA trả về giá trị khác NULL thì chương trình sẽ thoát.

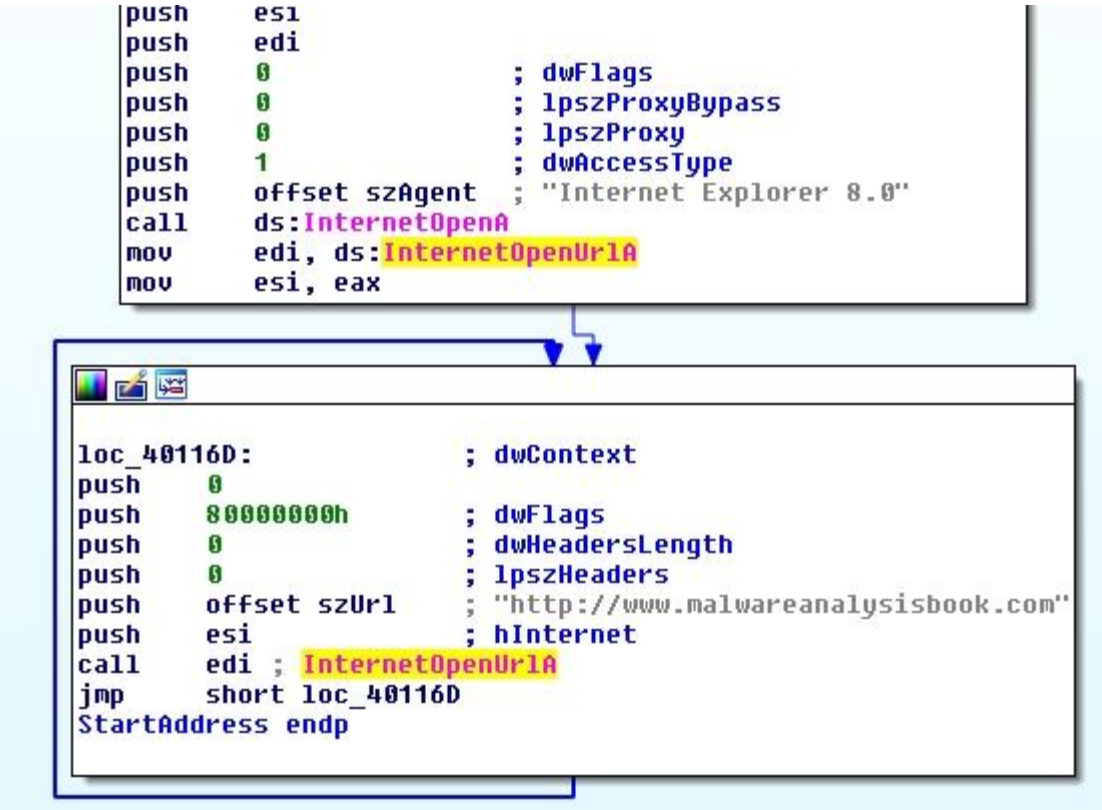
Câu 3:

Mutex HGL345 và dịch vụ Malservice có thể sử dụng làm các chỉ báo host để phát hiện chương trình này.

Câu 4:

Chương trình này có một luồng trỏ đến địa chỉ :

http://www.malwareanalysisbook.com



Câu 5:

Chúng ta có thể thấy một số lệnh gọi hàm API có liên quan đến Ngày và Bộ tính giờ. Như vậy chúng ta có thể thấy rằng sẽ có một thời điểm cụ thể khi Malware sẽ "giải phóng" chức năng thực sự của nó.

Đầu tiên, chúng ta thấy SystemTimetoFileTime đang được sử dụng để chuyển đổi thời gian hệ thống sang định dạng thời gian tệp (01/01/2100 00:00:00):

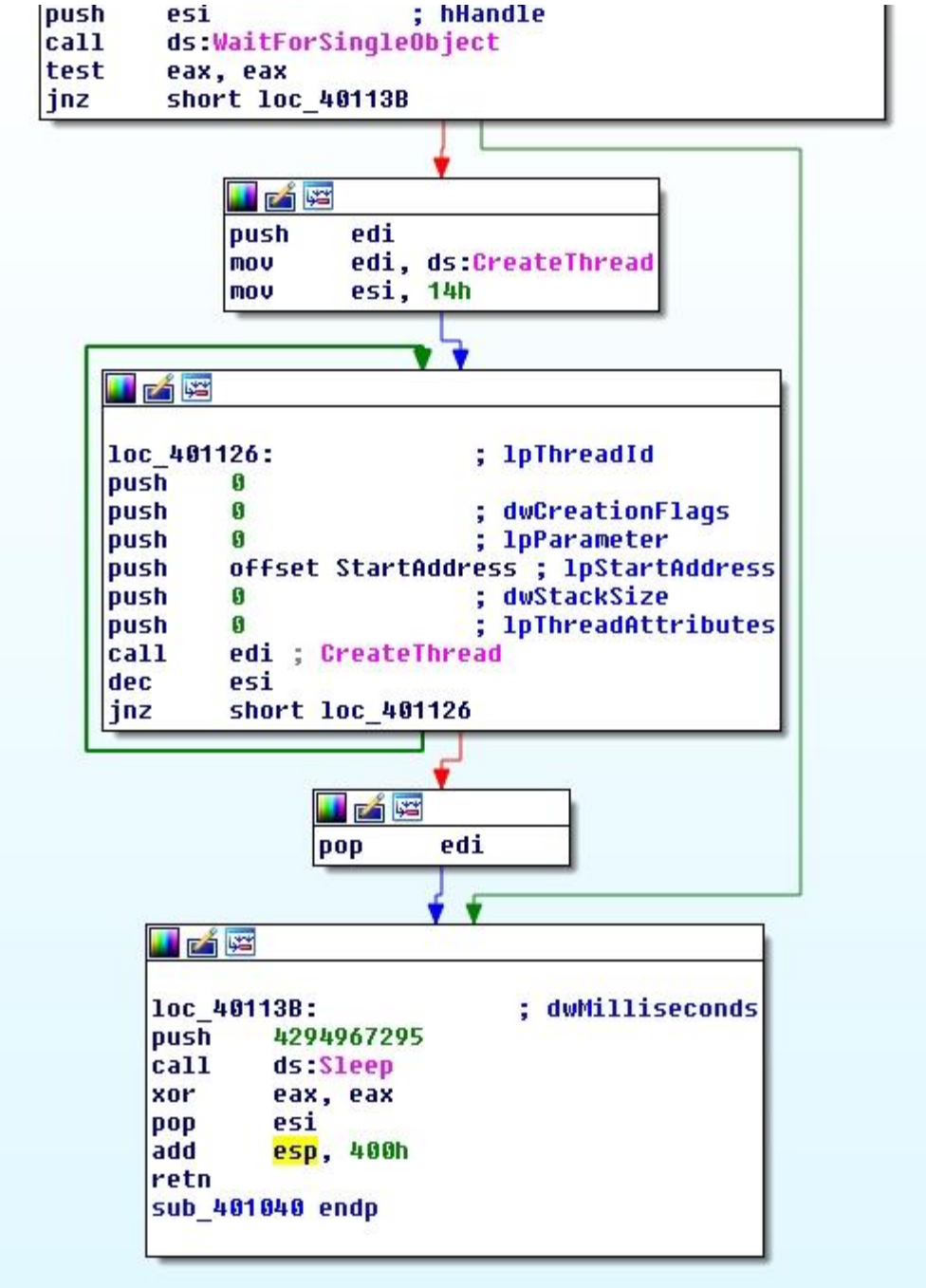
```
xor     edx, edx
lea     eax, [esp+404h+FileTime]
mov     dword ptr [esp+404h+SystemTime.wYear], edx
lea     ecx, [esp+404h+SystemTime]
mov     dword ptr [esp+404h+SystemTime.wDayOfWeek], edx
push    eax             ; lpFileTime
mov     dword ptr [esp+408h+SystemTime.wHour], edx
push    ecx             ; lpSystemTime
mov     dword ptr [esp+40Ch+SystemTime.wSecond], edx
mov     [esp+40Ch+SystemTime.wYear], 2100
call    ds:SystemTimeToFileTime
```

Sau đó, phần mềm độc hại sử dụng CreateWaitableTimerA để tạo đối tượng hẹn giờ. Sau đó, đối tượng hẹn giờ được đặt để kích hoạt vào ngày đã thấy trước đó (SetWaitableTimer). Cuối cùng, WaitForSingleObject được sử dụng. Hàm này đợi

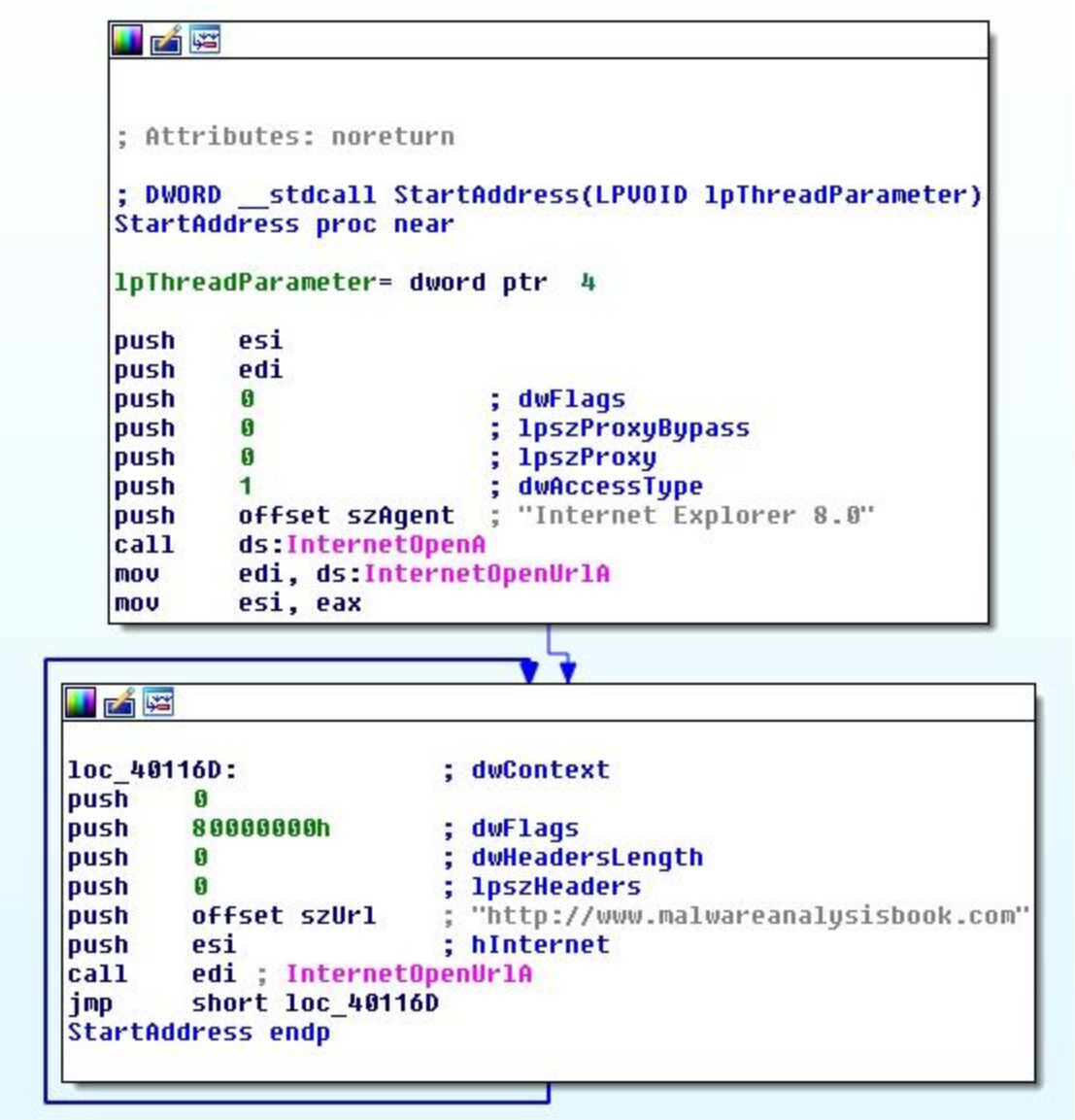
cho đến khi đối tượng được chỉ định ở trạng thái được báo hiệu hoặc hết thời gian chờ. Trong trường hợp này, nó sẽ đợi 01/01/2100 00:00:00 hoặc sau khi hết thời gian chờ (4294967295 mili giây = ~49,71 ngày):

```
call    ds:SystemTimeToFileTime
push    0               ; lpTimerName
push    0               ; bManualReset
push    0               ; lpTimerAttributes
call    ds:CreateWaitableTimerA
push    0               ; fResume
push    0               ; lpArgToCompletionRoutine
push    0               ; pfnCompletionRoutine
lea     edx, [esp+410h+FileTime]
mov     esi, eax
push    0               ; lPeriod
push    edx             ; lpDueTime
push    esi             ; hTimer
call    ds:SetWaitableTimer
push    4294967295      ; dwMilliseconds
push    esi             ; hHandle
call    ds:WaitForSingleObject
```

Nếu hết khoảng thời gian chờ, Phần mềm độc hại sẽ ngủ thêm 49,71 ngày nữa rồi thoát ra. Nếu đến ngày 01/01/2100 00:00:00 thì Phần mềm độc hại sẽ tạo 20 Threads:



Các Threads này được đặt để bắt đầu ở 0x401150 (offset StartAddress). Sau đó, mỗi Threads này sẽ yêu cầu lặp đi lặp lại http://www.malwareanalysisbook.com (vòng lặp vô hạn)



Vì vậy, chúng tôi có thể kết luận rằng mục tiêu của chương trình này rất có thể là gây ra cuộc tấn công DOS tới wed http://www.malwareanalysisbook.com.

Câu 6:

Phần mềm độc hại cuối cùng sẽ ngừng thực thi trước năm 2100, sau khi nó chạy được ít nhất ~100 ngày (~49,71 ngày đối với lệnh gọi WaitForSingleObject + ~49,71 ngày đối với lệnh gọi Ngủ). Tuy nhiên, Phần mềm độc hại đã có được tính ổn định thông qua Dịch vụ đã tạo (Malservice) và sẽ bắt đầu chạy lại sau khi Khởi động lại.

Sau khi đến ngày 01/01/2100 00:00:00, nó sẽ không bao giờ ngừng chạy (một cách duyên dáng) vì các Threads đang thực hiện một vòng lặp vô hạn.

Lab07-02

Câu 1

Phần mềm độc hại bắt đầu bằng cách sử dụng OleInitialize. Hàm này được sử dụng để khởi tạo thư viện COM. Sau đó CoCreateInstance được sử dụng. Hàm này tạo một đối tượng chưa được khởi tạo duy nhất của lớp được liên kết với một CLSID được chỉ định.

Cú pháp của hàm này như sau:

```
lea     eax, [esp+24h+ppv]
push    eax             ; ppv
push    offset riid     ; riid
push    4               ; dwClsContext
push    0               ; pUnkOuter
push    offset rclsid   ; rclsid
call    ds:CoCreateInstance
```

Chúng ta cần kiểm tra riid và rclsid.

Để có được những giá trị này, tôi đã sử dụng Plugin IDA để trả về mã định danh của riid và rclsid. Kết quả thực hiện có thể được nhìn thấy trong hình ảnh sau:

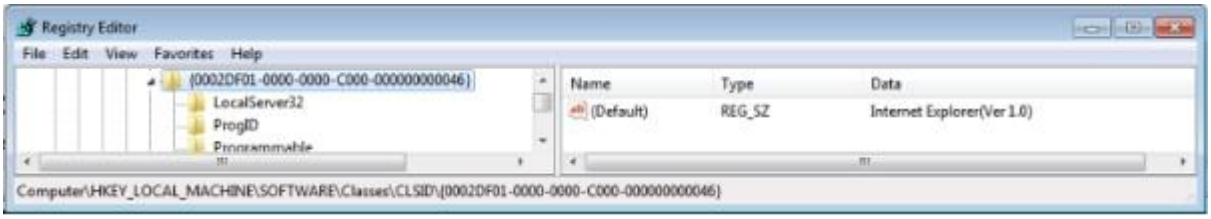
```
IDC>MakeGuid(riid)
{D30C1661-CDAF-11D0-8A3E-00C04FC9E26E}
          0.        0h            0o 00000000000000000000000000000000b '....'
IDC>MakeGuid(rclsid)
{0002DF01-0000-0000-C000-000000000046}
          0.        0h            0o 00000000000000000000000000000000b '....'
```

Chúng ta có thể sử dụng Regedit để kiểm tra thông tin về mã mà Máy chủ COM sẽ thực thi. Chúng ta có thể sử dụng các phím sau:

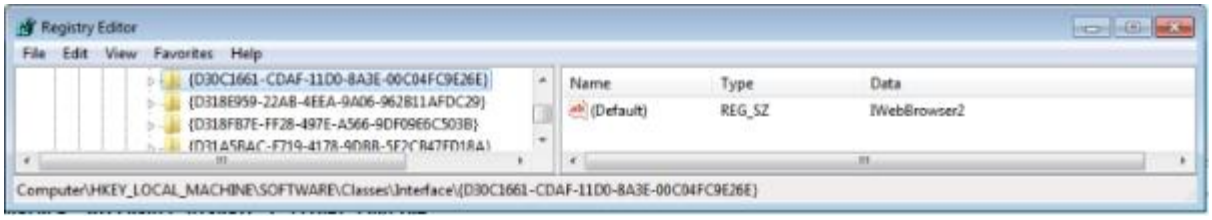
HKLM\SOFTWARE\Classes\CLSID\{clsid} (để kiểm tra CLSID)

HKLM\SOFTWARE\Classes\Interface\{riid} (để kiểm tra IID)

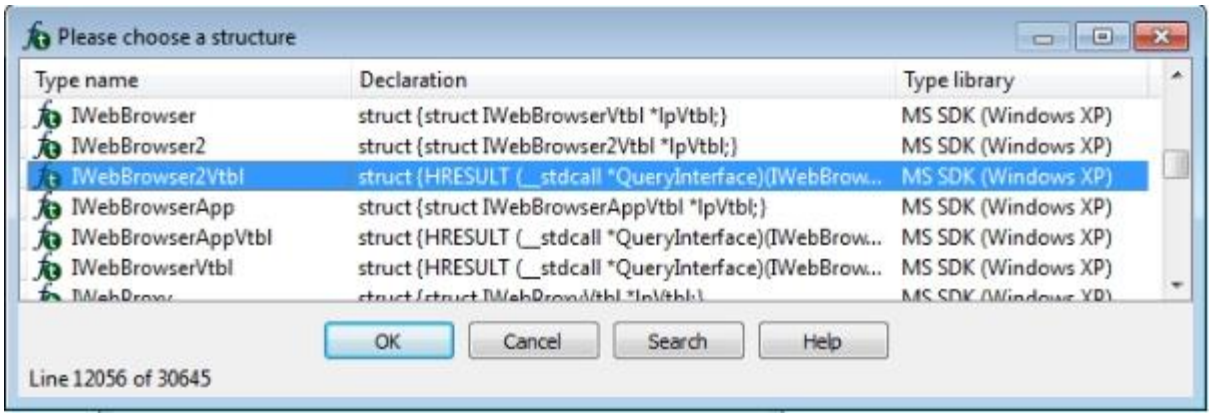
CLSID là Internet Explorer:



Giao diện được tải là IWebBrowser2:



DA lưu trữ các offset và Cấu trúc cho các giao diện chung. Trong trường hợp này, hãy thêm cấu trúc có tên InterfaceNameVtbl. Để làm được điều đó, chúng ta cần chuyển đến Tab Structures > Nhấn phím "Insert" > Add Standard Structure và chọn cấu trúc IWebBrowser2Vtbl:



Quay trở lại phần Disassembly chúng ta có thể thấy IDA đã thêm chú thích vào lệnh gọi:

```
lea     ecx, [esp+24h+pvarg]
push    esi
push    ecx             ; pvarg
call    ds:VariantInit
push    offset psz      ; "http://www.malwareanalysisbook.com/ad.h"...
mov     word ptr [esp+2Ch+Flags.anonymous_0], 3
mov     dword ptr [esp+2Ch+Flags.anonymous_0+8], 1
call    ds:SysAllocString
lea     ecx, [esp+28h+pvarg]
mov     esi, eax
mov     eax, [esp+28h+ppv]
push    ecx             ; Headers
lea     ecx, [esp+2Ch+pvarg]
mov     edx, [eax]
push    ecx             ; PostData
lea     ecx, [esp+30h+pvarg]
push    ecx             ; TargetFrameName
lea     ecx, [esp+34h+Flags]
push    ecx             ; Flags
push    esi             ; URL
push    eax             ; This
call    [edx+IWebBrowser2Vtbl.Navigate]
push    esi             ; bstrString
call    ds:SysFreeString
pop     esi
```

Về cơ bản, tất cả những gì Phần mềm độc hại thực hiện là một yêu cầu đơn giản tới http://www.malwareanalysisbook.com/ad.html.

Do đó, Phần mềm độc hại không đạt được tính ổn định duy trì kết nối tới hệ điều hành.

Câu 2:

Mục đích của chương trình này là mở URL sau bằng Internet Explorer:

http://www.malwareanalysisbook.com/ad.html

Câu 3:

Phần mềm độc hại kết thúc việc thực thi ngay sau khi yêu cầu được thực hiện.

Lab 07-03

Câu 1:

Chương trình thực hiện việc mapping C:\\Windows\\System32\\Kernel32.dll và Lab07-03.dll, sau khi thực hiện một loạt các hành vi, nó tiến hành sửa một hoặc cả hai file trên. Cuối cùng, file: C:\\windows\\system32\\kerne132.dll được tạo ra. Bằng phương thức này, mã độc đảm bảo hoạt động ngay cả khi chương trình khởi động lại.

Câu 2:

Việc tạo ra file mới: C:\\windows\\system32\\kerne132.dll.

Nội dung trong file chứa chuỗi: WARNING_THIS_WILL_DESTROY_YOUR_MACHINE và một mutex: SADFHUHF

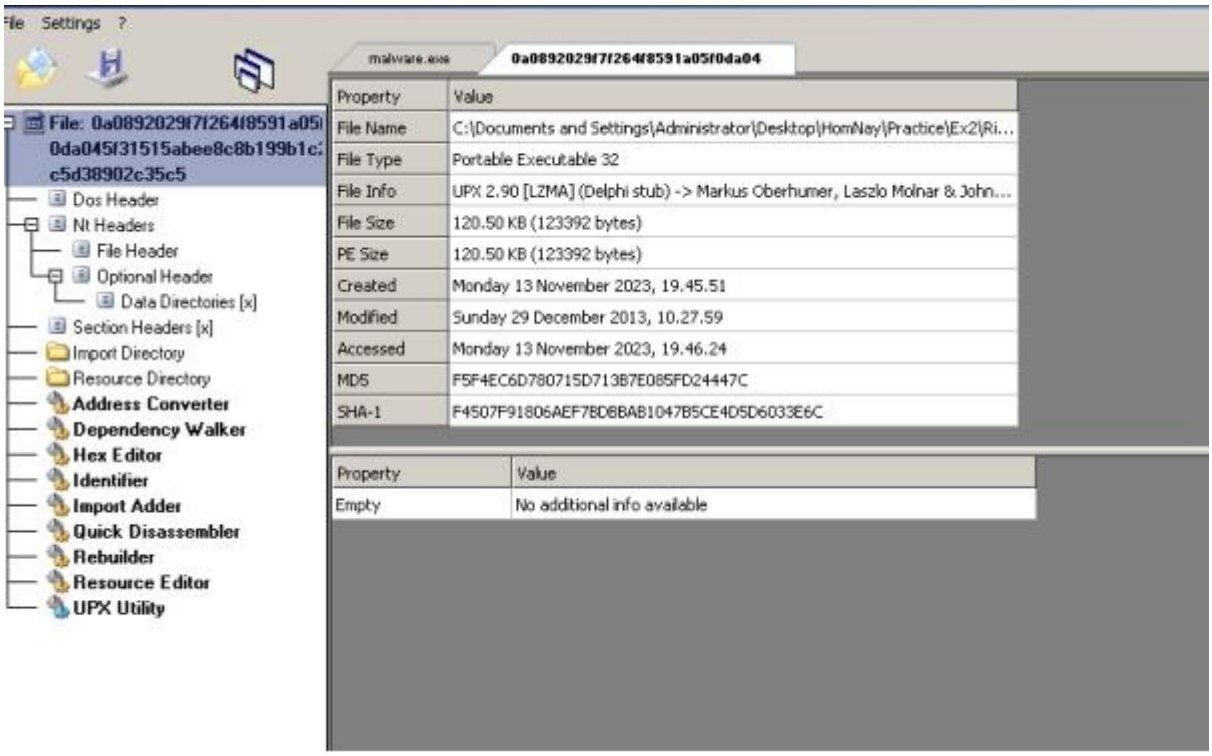
Câu 3:

Chương trình: Lab07-03.exe sử dụng để thực thi: Lab07-03.dll thành một backdoor hoạt động liên tục trên máy bị lây nhiễm. DLL này kết nối tới địa chỉ C&C: 127.26.152.13 và cho phép nhận lệnh thực thi: sleep và exec.

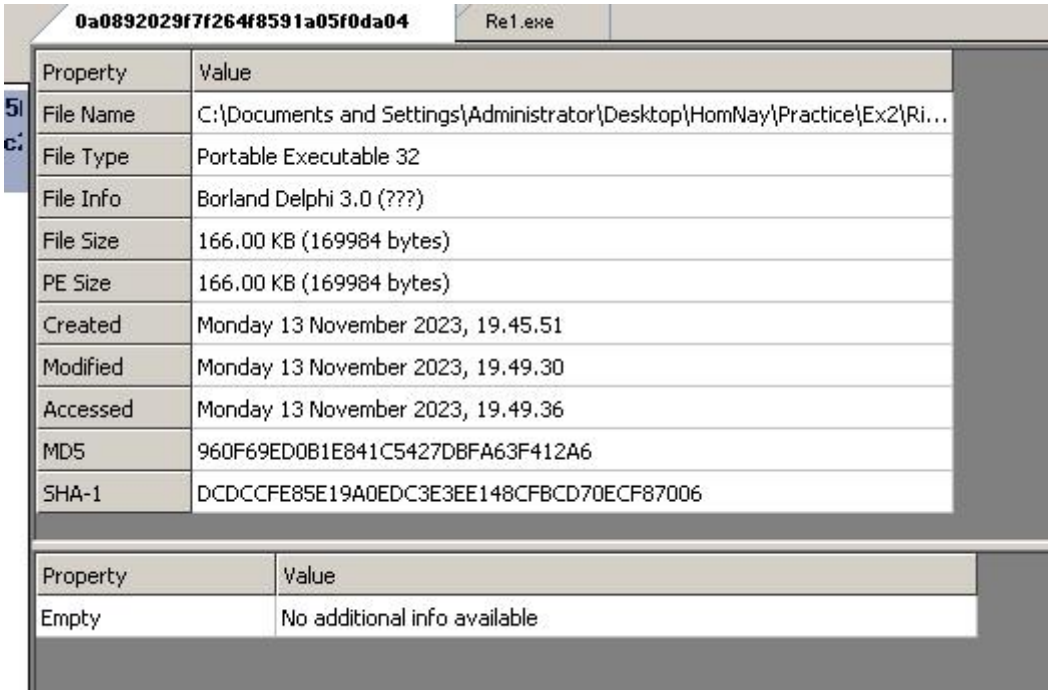
Câu 4:

Phương pháp đơn giản nhất là xóa các file: Lab07-03.exe, Lab07-03.dll, kerne132.dll. Tuy nhiên, trước đó cần tìm và chỉnh sửa các chương trình gọi tới DLL này để thực hiện gọi về DLL chuẩn: kernel32.dll.

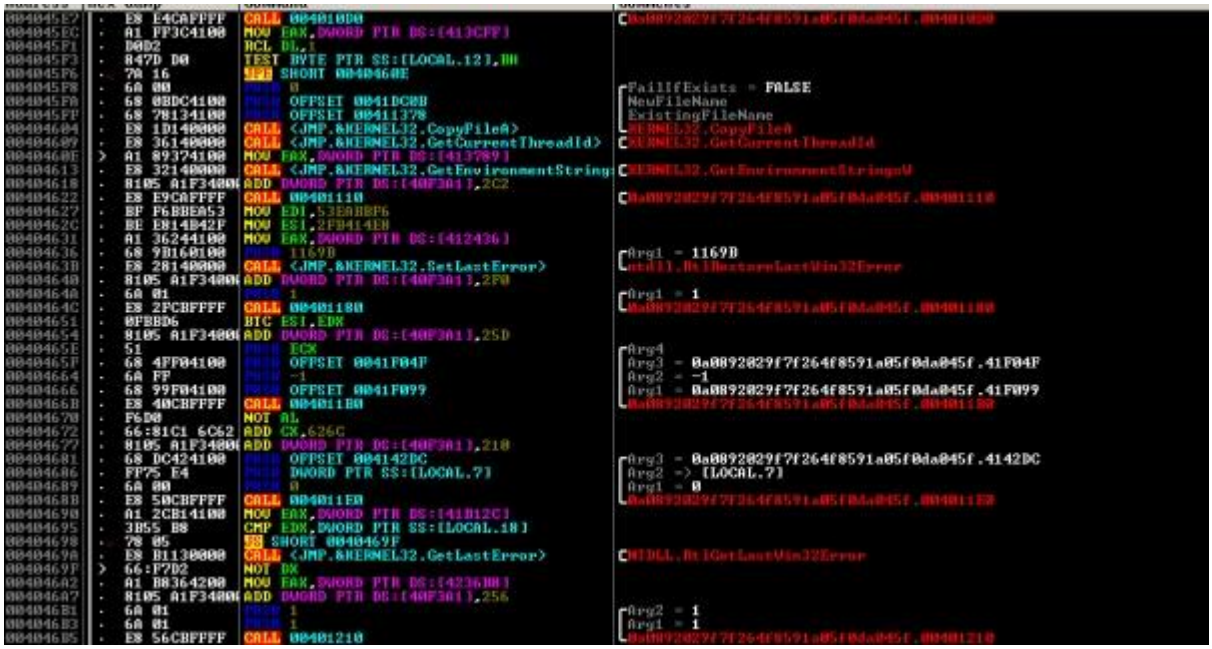
Ex2:



Sau khi mở file thì ta thấy file bị packed bởi UPX2.9. bây giờ ta sẽ unpack nó để phân tích.



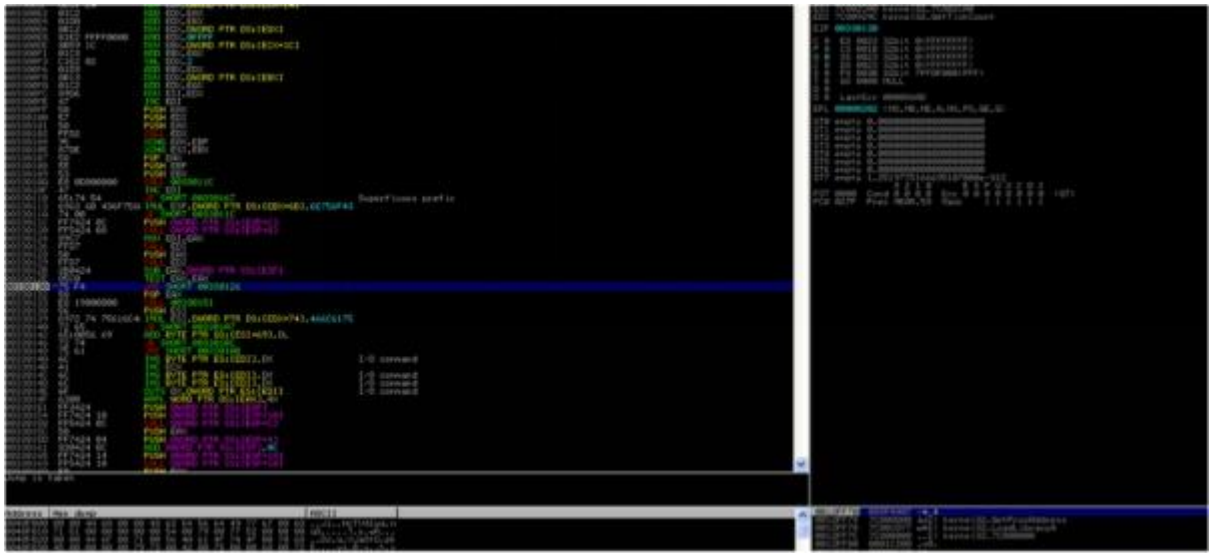
Đã unpack thành công



Trong file có rất nhiều hàm CopyFileA cố gắng sao chép một số tệp ngẫu nhiên vào vị trí ngẫu nhiên và việc này được thực hiện nhiều lần trong một vòng lặp rất lớn. Đây là mã rác được sử dụng để có thể làm nản long người làm reverse.

Bên trong mã rác này, phần mềm độc hại triển khai một kỹ thuật chống gỡ lỗi rất mạnh mẽ. Phần mềm độc hại gọi API “kerne32.CloseHandle” với các giá trị ngẫu nhiên là “hObject”

Tuy nhiên, ngay cả sau khi bỏ qua kỹ thuật chống gỡ lỗi này, nếu bạn cho phép phần mềm độc hại chạy, nó sẽ được thực thi và chấm dứt với mã thoát 0 mà không làm gì hoặc sẽ dừng với ngoại lệ "Vi phạm truy cập", tùy thuộc vào thời gian trôi qua kể từ khi chương trình được thực thi. Điều này là do kỹ thuật chống gỡ lỗi được phần mềm độc hại triển khai bằng cách sử dụng ‘kernel32.GetTickCount’API



Lệnh tại 0x00330126 sẽ gọi kernel32.GetTickCount và PUSH giá trị đó trên ngăn xếp. Nó lại thực hiện lệnh gọi tương tự, trừ giá trị đó khỏi giá trị thu được trước đó và kiểm tra xem nó có bằng 0 hay không. Nó tiếp tục vòng lặp này cho đến khi nó nhận được phép trừ của hai giá trị này bằng 0. Mỗi khi vòng lặp này được thực thi, giá trị của kernel32.GetTickCount sẽ được đẩy lên ngăn xếp. Sau khi thoát khỏi vòng lặp này, GỌI 00330151 được thực hiện. Hàm này tạo CALL DWORD PTR SS:[ESP+C], lý tưởng nhất là kernel32.GetProcAdddress. Tuy nhiên, nếu bạn đang gỡ lỗi phần mềm độc hại thì ngăn xếp có thể có các giá trị được đẩy lên ngăn xếp do vòng lặp 'GetTickCount' trước đó và do đó gây ra Vi phạm quyền truy cập. Để bỏ qua kỹ thuật gỡ lỗi này, bạn cần điều chỉnh giá trị ESP sao cho [ESP+C] trỏ đến kernel32.GetProcAddress.

Phần mềm độc hại đang được phân tích được tạo bằng cách sử dụng CrimeWare Kit có sẵn trên thị trường ngầm có tên CRUM Cryptor Polymorphic của Sunzer Flint (Hình 5). Đây là chương trình được các tác giả phần mềm độc hại sử dụng để mã hóa phần mềm độc hại thông qua khóa ngẫu nhiên 256 byte và cũng có tính đa hình.

Hai kỹ thuật chống gỡ lỗi cuối cùng được phần mềm độc hại triển khai trước khi nó tự giải mã, được thực hiện bằng cách truy cập Khối môi trường quy trình (PEB) của quy trình hiện tại. Kỹ thuật đầu tiên là kiểm tra xem byte ở offset 0x02(IsDebugged) trong PEB có được đặt hay không. Nếu một chương trình đang được gỡ lỗi, byte này được đặt thành 1, còn lại là 0. Kỹ thuật chống gỡ lỗi khác là kiểm tra NtGlobalFlags ở offset 0x68 trong PEB. Nếu quá trình này được gỡ lỗi, một số cờ kiểm soát các thủ tục thao tác vùng nhớ heap trong ntdll sẽ được đặt. Tính năng chống gỡ lỗi này có thể được bỏ qua bằng cách đặt lại trường NtGlobalFlags

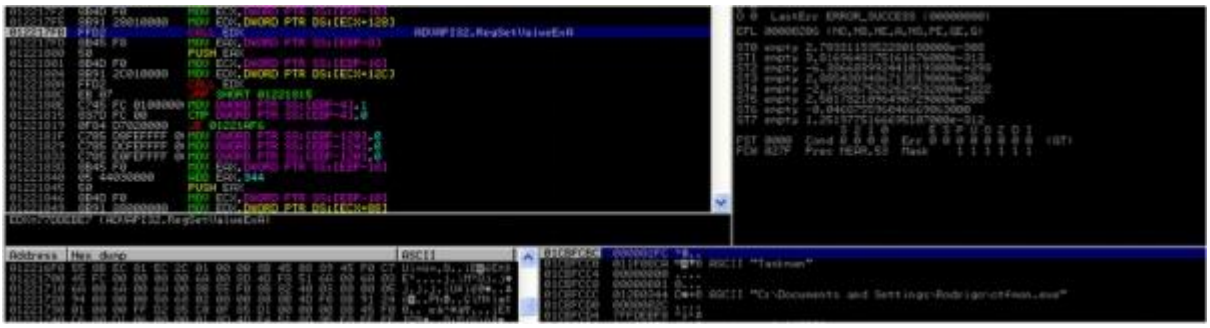
Sau khi chúng tôi bỏ qua tất cả kỹ thuật chống gỡ lỗi này, phần mềm độc hại sẽ bắt đầu nhập thư viện khác mà nó yêu cầu bằng API kernel32.LoadLibraryA.

Sau đó, phần mềm độc hại sẽ cố gắng tìm xem quy trình “explorer.exe” có đang chạy trên hệ thống hay không và xử lý quy trình này thông qua kernel32.OpenProcess API

Sau đó, phần mềm độc hại dự trữ một vùng bộ nhớ trong không gian địa chỉ ảo của quy trình “explorer.exe” bằng API kernel32.VirutalAllocEx và tạo một luồng trong quy trình explorer.exe thông qua API kernel32.CreateRemoteThread (Hình 8). Khi luồng từ xa được tạo trong quy trình “explorer.exe”, phần mềm độc hại sẽ tự kết thúc với mã thoát 0

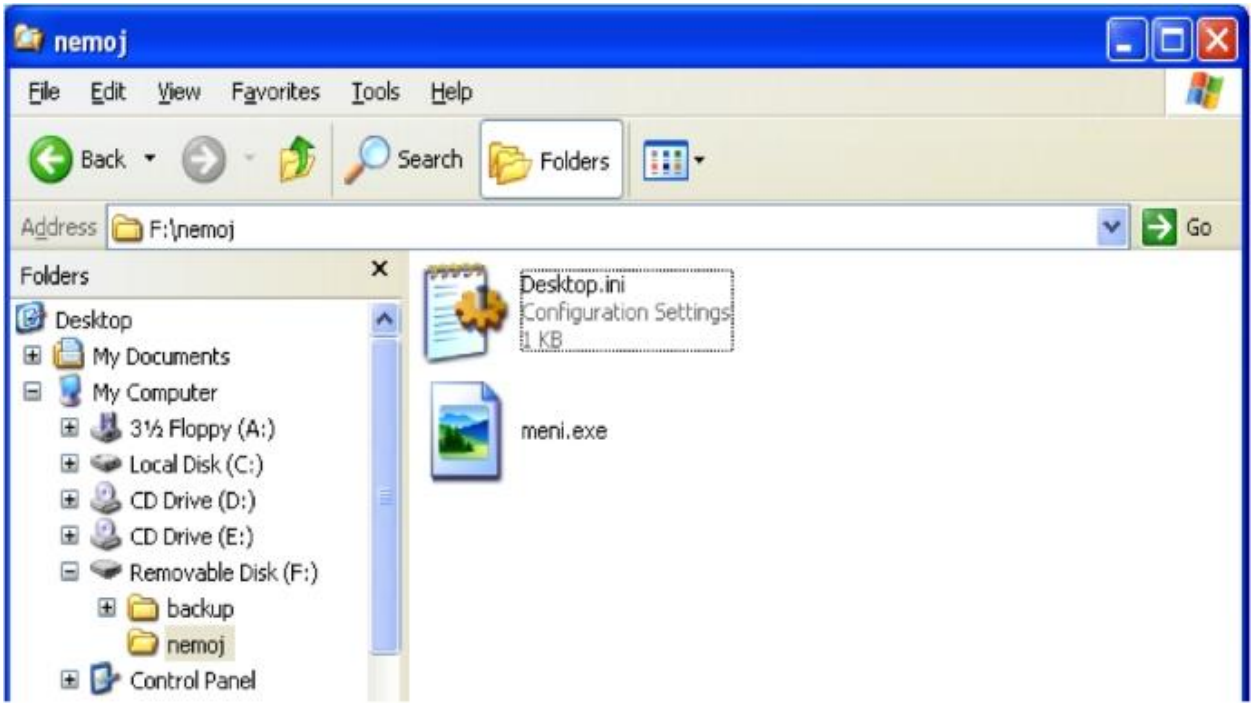
Sau khi luồng mới này được tạo trong quy trình thám hiểm, tệp phần mềm độc hại gốc sẽ được sao chép vào vị trí "%USERPROFILE%\\ctfmon.exe" (Hình 9) và đặt thuộc tính tệp thành hệ thống, chỉ đọc và ẩn.

Sau khi tạo tệp thực thi, phần mềm độc hại sẽ tạo khóa "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman": "%USERPROFILE%\ctfmon.exe" (Hình 10). Khóa này đảm bảo rằng mỗi khi quá trình explorer.exe được tạo, phần mềm độc hại sẽ được thực thi.



Phần mềm độc hại tạo NamedPipe mà sau này có thể được sử dụng để liên lạc giữa các quá trình

Sau đó, phần mềm độc hại cố gắng liên lạc với chủ nhân của nó tại "tinaivanovic.sexy-serbain-girls.info"



Phần mềm độc hại được biết là lây lan qua ổ USB. Khi kết nối thẻ USB với máy chủ bị nhiễm, phần mềm độc hại sẽ thả một bản sao của chính nó vào "[RemovableDrive]\\nemoj\\meni.exe" và tạo tệp autorun.inf