



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

# BÁO CÁO

# ĐỒ ÁN CRACK PHẦN MỀM

*Bộ môn: Kiến trúc máy tính và hợp ngữ*

*Giáo viên: Nguyễn Thanh Quân, Phạm Tuấn Sơn*

*Nhóm thực hiện: 1612288 – 1612291 – 1612296*

---

Tháng 8, 2016

# Mục lục

Tổng quan.....	3
Thông tin đề án.....	3
Phân công.....	3
Kiến thức cần có để giải quyết.....	3
1.1.....	4
Manh mối.....	4
Đoạn mã phát sinh khóa của chương trình.....	6
Test case.....	6
Phân tích mã assemble.....	6
Tóm tắt thuật toán.....	8
Kết luận.....	9
2.1.....	9
Manh mối.....	9
Phân tích mã assemble.....	13
Tóm tắt thuật toán.....	17
Kết luận.....	18
2.2.....	18
Manh mối.....	18
Phân tích mã assemble.....	21
Tóm tắt thuật toán.....	25
Kết luận.....	25
Tổng kết.....	26
Mức độ hoàn thành.....	26
Kiến thức thu được.....	26
Tham khảo.....	26

# Tổng quan

---

## Thông tin đồ án

**Bộ môn:** Kiến trúc máy tính và hợp ngữ

Giáo viên: Nguyễn Thanh Quân

Giáo viên: Phạm Tuấn Sơn

**Tên: Đồ án 3 - Crack phần mềm**

**Đề:** 4 (1.1.exe, 2.1.exe, De4\_2.2.exe là 2.2.exe)

**Deadline:** 23:55 8/6/2018

Thời gian thực hiện: 22/5/2018 – 8/6/2018

## Phân công

1612288	Nguyễn Khắc Nguyên Khang	1.1
1612291	Nguyễn Thị Ngân Khánh	2.1
1612296	Tạ Ngọc Duy Khoa	2.2

*Công việc gồm: viết báo cáo và tạo keygen (xxx.cpp, xxx.exe) của crackme được phân công.*

## Kiến thức cần có để giải quyết

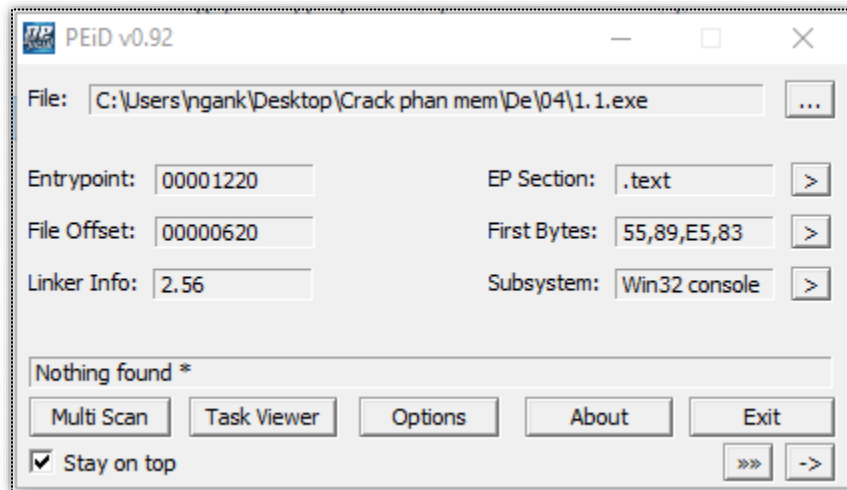
1. Kỹ năng kiểm tra bằng PEiD: kiểm tra file có pack không, kiểm tra có dấu vết của mã hóa trong file thực thi không, unpack nếu cần.
2. Kỹ năng debug trong OllyDbg: đặt breakpoint F2, F9 để bắt đầu debug, F8 để chạy từng lệnh, F7 để đi vào bên trong hàm chạy từng lệnh; tìm chuỗi với Search for > All referenced strings; truy cập đến mọi vùng nhớ bằng Memory map; kỹ năng xử lý địa chỉ của vùng nhớ stack, dump.
3. Đọc hiểu các lệnh hợp ngữ: nhóm lệnh tính toán (ADD, SUB, IMUL, XOR, OR, AND), nhóm lệnh so sánh (TEST, CMP), nhóm lệnh nhảy (JE, JAE, JNE, JNZ...), nhóm lệnh gán (MOV-gán giá trị, LEA-gán địa chỉ).
4. Các yêu cầu cụ thể của 3 crackme:
  - 1.1. Đọc hiểu lệnh hợp ngữ để tìm thuật toán phát sinh key. Thuật toán có input: (string) Regname, output: (int) Regcode.
  - 2.1. Đọc hiểu lệnh hợp ngữ để truy đến file mã nguồn thực sự. Đây là file .bat chứa password.
  - 2.2. Đọc hiểu lệnh hợp ngữ để tìm thuật toán phát sinh key. Thuật toán có input: (string) Firstname, output: (int) Serial.

## 1.1

### Manh mối

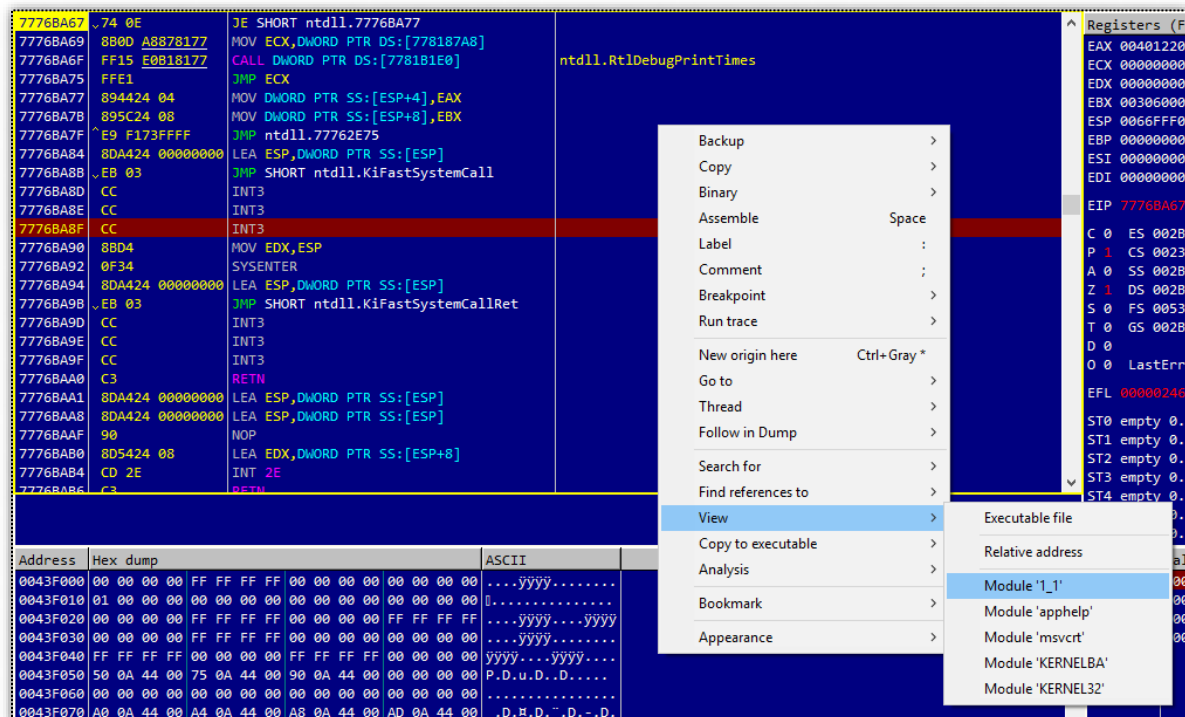
Trước hết, phân tích mã hợp ngữ phải trên file thực thi không bị pack. Pack là thao tác của người lập trình để nén chương trình. Việc này ngăn chặn những nỗ lực đọc hiểu mã nguồn hợp ngữ của các cracker. Nếu chủ quan không kiểm tra xem file bị pack hay không mà trực tiếp phân tích, kết quả phân tích sẽ không chính xác, thậm chí không phân tích được do nhiều dữ kiện quan trọng bị ẩn mất.

Đầu tiên, **Scan with PEiD 1.1.exe** để kiểm tra tệp thực thi này đã pack chưa.

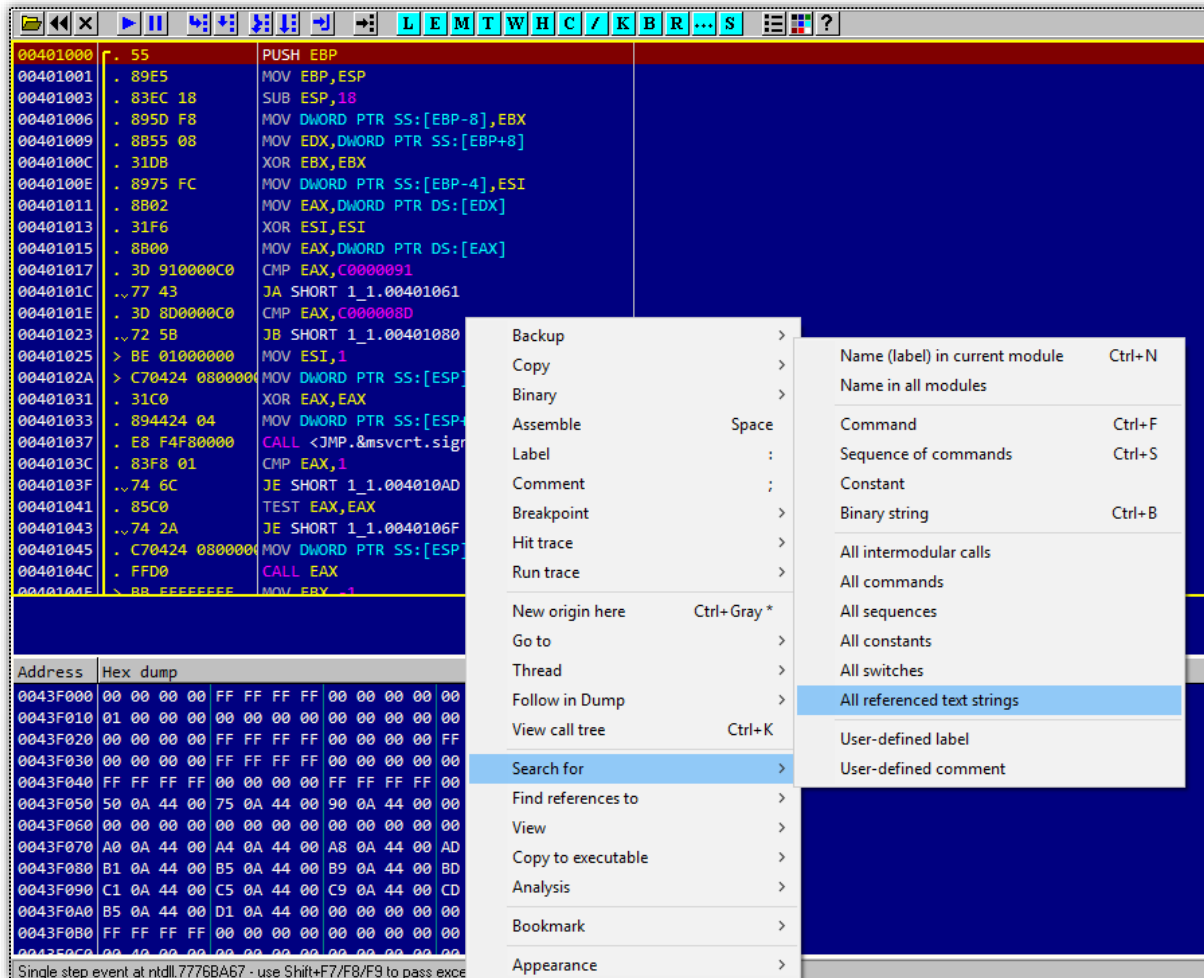


Kết quả là **Nothing found \***, 1.1.exe không bị pack, tiến hành phân tích mã hợp ngữ bằng **OlllyDbg**

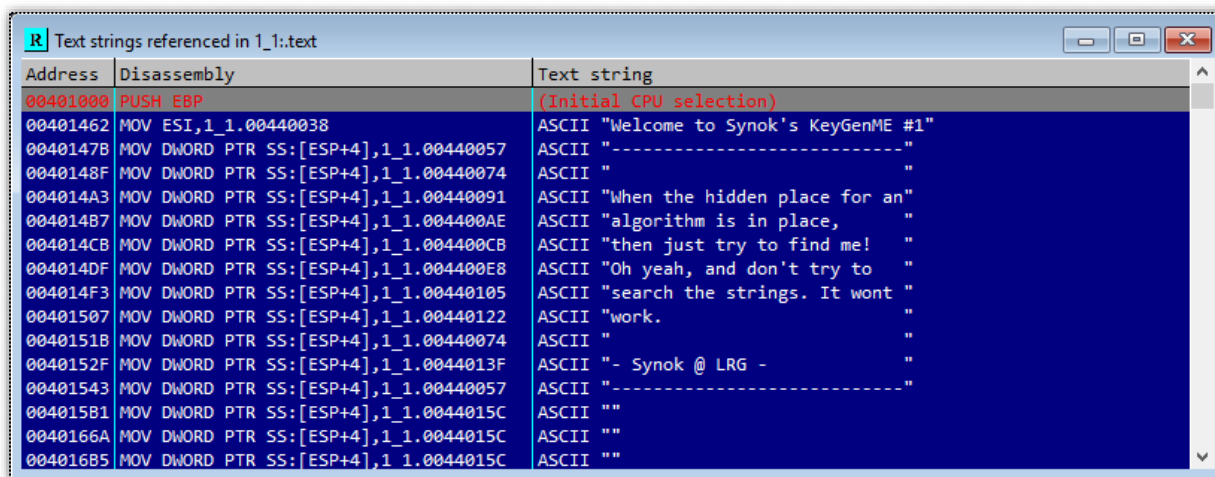
Chạy **OlllyDbg**, mở 1.1.exe (**F3**). Nếu không mặc định, **chuột phải tại Disassembler Window > View > Module '1\_1'**



Bây giờ, bắt đầu tìm kiếm những chuỗi được 1.1.exe hiển thị. **Chuột phải tại Disassembler Window > Search for > All referenced text strings**



Một cửa sổ mới xuất hiện, chứa nhiều chuỗi, nhưng những chuỗi được hiển thị trong hình cần chú ý.



Gợi ý để lại: “When the hidden place for an algorithm is in place, then just try to find me! Oh yeah, and don’t try to search the string. It wont work”.

*Tạm kết luận: key sẽ không có giá trị cố định mà được phát sinh từ thuật toán nào đó.*

## Đoạn mã phát sinh khóa của chương trình

004015EE	. 8D45 D8	LEA EAX,DWORD PTR SS:[EBP-28]
004015F1	. 890424	MOV DWORD PTR SS:[ESP],EAX
004015F4	. E8 97F30000	CALL <JMP.&msvcrt.strlen>
004015F9	. 01C0	ADD EAX,EAX
004015FB	. 8985 64FFFFFF	MOV DWORD PTR SS:[EBP-9C],EAX
00401601	. C74424 04 5F0	MOV DWORD PTR SS:[ESP+4],1_1.0044015F
00401609	. C70424 C03344	MOV DWORD PTR SS:[ESP],1_1.004433C0
00401610	. E8 53AD0300	CALL 1_1.0043C368
00401615	. 8D45 B8	LEA EAX,DWORD PTR SS:[EBP-48]
00401618	. 890424	MOV DWORD PTR SS:[ESP],EAX
0040161B	. E8 60F30000	CALL <JMP.&msvcrt.printf>
00401620	. 8D45 94	LEA EAX,DWORD PTR SS:[EBP-6C]
00401623	. 894424 04	MOV DWORD PTR SS:[ESP+4],EAX
00401627	. C70424 603444	MOV DWORD PTR SS:[ESP],1_1.00443460
0040162E	. E8 FD6D0200	CALL 1_1.00428430
00401633	. 8D45 EF	LEA EAX,DWORD PTR SS:[EBP-11]
00401636	. 890424	MOV DWORD PTR SS:[ESP],EAX
00401639	. E8 42F30000	CALL <JMP.&msvcrt.printf>
0040163E	. 8B85 64FFFFFF	MOV EAX,DWORD PTR SS:[EBP-9C]
00401644	. 01C0	ADD EAX,EAX

## Test case

Reg name .> aloha

Reg code .> 15

## Phân tích mã assemble

```
004015EE    LEA EAX, DWORD PTR SS:[EBP-28]
```

Ý nghĩa: Load địa chỉ stack EBP-28 (0066FF10) vào EAX

Kết quả:

Registers (FPU)	
EAX	0066FF10 ASCII "aloha"

Khi đó, EBP-28 (0066FF10) đang chứa chuỗi Reg name (“aloha”)

Address	Value	ASCII	Comment
EBP-28	686F6C61	aloh	
EBP-24	74100061	a.[t	msvcrt.74100061

```
004015F1    MOV DWORD PTR SS:[ESP], EAX
004015F4    CALL <JMP.&msvcrt.strlen>
```

Ý nghĩa: Truyền chuỗi “aloha” vào đỉnh stack, tiến hành gọi hàm <JMP.&msvcrt.strlen>. Giá trị trả về của hàm là độ dài chuỗi (số nguyên)

Kết quả: EAX lưu giá trị trả về

#### Registers (FPU)

EAX 00000005

004015F9      ADD EAX,EAX

Ý nghĩa:  $EAX = EAX + EAX$

Kết quả:

#### Registers (FPU)

EAX 0000000A

004015FB      MOV DWORD PTR SS:[EBP-9C],EAX

Ý nghĩa: lưu EAX vào stack địa chỉ EBP-9C (0066FE9C)

Kết quả:

Address	Value
EBP-9C	0000000A

00401601      MOV DWORD PTR SS:[ESP+4],1\_1.0044015F  
00401609      MOV DWORD PTR SS:[ESP],1\_1.004433C0  
00401610      CALL 1\_1.0043C368  
00401615      LEA EAX,DWORD PTR SS:[EBP-48]  
00401618      MOV DWORD PTR SS:[ESP],EAX  
0040161B      CALL <JMP.&msvcrt.printf>

Ý nghĩa: In ra màn hình “Reg code. >”

00401620      LEA EAX,DWORD PTR SS:[EBP-6C]  
00401623      MOV DWORD PTR SS:[ESP+4],EAX  
00401627      MOV DWORD PTR SS:[ESP],1\_1.00443460  
0040162E      CALL 1\_1.00428430

Ý nghĩa: Yêu cầu nhập vào key

Kết quả: Lưu key vào stack địa chỉ EBP-6C (0066FECC). Lưu ý: 0000000Fh = 15

Address	Value
EBP-6C	0000000F

...

0040163E      MOV EAX,DWORD PTR SS:[EBP-9C]

Ý nghĩa: Gán EAX bởi giá trị tại địa chỉ stack SS:EBP-9C

Kết quả:

Registers (FPU)

EAX 0000000A

00401644      ADD EAX, EAX

Ý nghĩa:  $EAX = EAX + EAX$

Kết quả:

Registers (FPU)

EAX 00000014

Kết luận: EAX = 00000014h là khóa phát sinh từ chuỗi user name. Sau khi đổi sang hệ thập phân, trong trường hợp này, **khóa là 20**.

## Tóm tắt thuật toán

- 1-Đọc **regname** từ bàn phím người dùng.
- 2-Lưu chuỗi regname vào EAX, lưu EAX vào stack SS:EBP-28.
- 3-Tìm độ dài chuỗi regname: input: chuỗi regname; output(số nguyên): độ dài chuỗi, lưu trong EAX.
- 4- $EAX = EAX + EAX$ .
- 5-Lưu EAX vào stack SS:EBP-9C.
- 6-Đọc **regcode** vào EAX, lưu EAX vào stack SS:EBP-6C.
- 7-Truyền giá trị tại địa chỉ SS:EBP-9C vào EAX.
- 8-Bắt đầu so khớp để tìm goodboy hay badboy. Đoạn mã bên dưới mô phỏng quá trình này.

Input: a string of username  $U$ , an integer of key  $K$ .

Output: GOODBOY or BADBOY.

DWORD PTR SS:[EBP-9C] = 2 \*  $U.length$

DWORD PTR SS:[EBP-6C] =  $K$

EAX = DWORD PTR SS:[EBP-9C]

EAX = EAX + EAX

if EAX !=  $K$  jump BADBOY

EAX = 'Cracked! :)'

print

BADBOY:

EAX = DWORD PTR SS:[EBP-9C]

EAX = EAX + EAX

if EAX =  $K$  jump return

EAX = 'Try again! :('

print

return

Từ đoạn đọc regname từ bàn phím người dùng đã không có sự dịch stack, nên dù mô phỏng bằng địa chỉ stack tương đối, nhưng kết quả vẫn đảm bảo.



## Kết luận

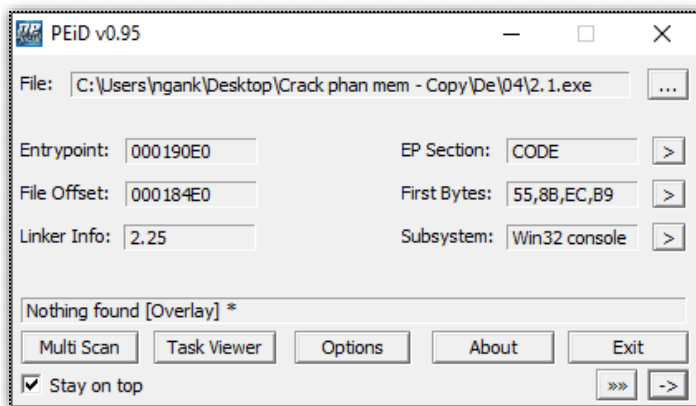
Reg name. >: string U

Reg code. >: int K

Với keygen 1\_1.exe, input: U, output: K

## 2.1

Đầu tiên, **Scan with PEiD 2.1.exe** để kiểm tra tệp thực tin này đã pack chưa.



Kết quả là **Nothing found [Overlay] \***, nghĩa là 2.1.exe không bị pack, tiến hành phân tích mã hợp ngữ bằng **OlllyDbg** bình thường.

## Manh mồi

Chuỗi in ra trong console không xuất hiện trong vùng nhớ chương trình, đúng hơn là vùng nhớ không thể truy xuất khi chưa chạy chương trình.

```
004195F3 . E8 98CBFEFF CALL <JMP.&kernel32.WaitForSingleObject;LWaitForSingleObject>
```

Sau khi chạy dòng lệnh 004195F3, màn hình console xuất một lượt như sau:

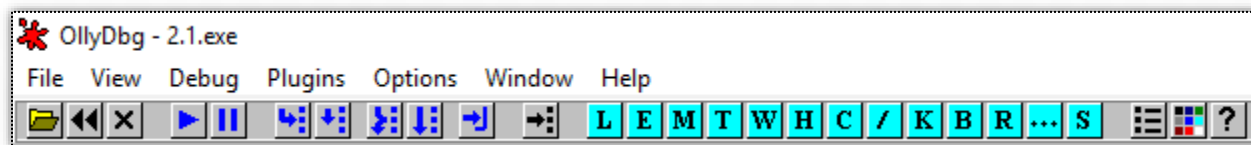
```
its the first time i give someone try 2 crack this kind of CM.
g00d luck! - N3tRAt aka V[i]RuS
Enter Password:
```

Nhập từ bàn phím password thử, kết quả là:

```
its the first time i give someone try 2 crack this kind of CM.
g00d luck! - N3tRAt aka V[i]RuS
Enter Password:
12
bad password
Press any key to continue . . .
```

Bằng cách **chuột phải tại Disassembler Window > Search for > All referenced text strings**, các string được tìm thấy [00401006:004197FC] không hề chứa những string được hiển thị từ console. [1]

Tìm đến **Memory map** của chương trình bằng cách chọn **M** trên toolbar



Một cửa sổ được hiện ra gồm rất nhiều dòng dữ liệu, nội dung sơ bộ: địa chỉ bắt đầu của 1 vùng nhớ và đối tượng quản lý vùng nhớ đó...

Address	Size	Owner	Section	Contains	Type	Access	Initial access
00011000	00001000				Priv 00021004	RW	RW
00028000	00001000				Priv 00021004	RW	RW
00030000	00010000				Map 00041004	RW	RW
00040000	00019000				Map 00041002	R	R
00095000	00008000				Priv 00021104	RW Guarded	RW
0019A000	00002000				Priv 00021104	RW Guarded	RW
0019C000	00004000			stack of ma	Priv 00021104	RW Guarded	RW
001A0000	00004000				Map 00041002	R	R
001B0000	00001000				Priv 00021004	RW	RW
001C1000	00001000				Priv 00021004	RW	RW
001E0000	00001000				Priv 00021004	RW	RW
001F0000	00004000				Map 00041002	R	R
003EE000	00004000				Priv 00021004	RW	RW
003F2000	00003000			data block c	Priv 00021004	RW	RW
003F5000	00003000			data block c	Priv 00021004	RW	RW
003F8000	00003000			data block c	Priv 00021004	RW	RW
003FB000	00004000			data block c	Priv 00021004	RW	RW
00400000	00001000	2_1		PE header	Imag 01001002	R	RWE
00401000	00019000	2_1	CODE	code	Imag 01001002	R	RWE
0041A000	00002000	2_1	DATA	data	Imag 01001002	R	RWE
0041C000	00003000	2_1	BSS		Imag 01001002	R	RWE
0041F000	00001000	2 1	.idata	imports	Imag 01001002	R	RWE

Kiên nhẫn tìm kiếm từng vùng, phát hiện vùng nhớ này có điều đặc biệt:

Address	Size	Owner	Section	Contains	Type	Access	Initial access
02220000	00004000				Priv 00021004	RW	

Chuỗi hiện thị ở console đã được tìm thấy:

```

02220D00 00 0D 22 02 00 0D 22 02 3C 02 00 00 45 43 48 4F ..".." < ..ECHO
02220D10 20 4F 46 46 0D 0A 63 6C 73 0D 0A 52 45 4D 20 74 OFF..cls..REM t
02220D20 69 74 6C 65 20 63 72 61 63 6B 6D 65 20 2D 20 42 itle crackme - B
02220D30 61 74 63 68 20 6F 72 20 6E 6F 74 3F 0D 0A 73 65 atch or not?..se
02220D40 74 20 72 3D 6F 0D 0A 73 65 74 20 6F 3D 74 0D 0A t r=o..set o=t..
02220D50 73 65 74 20 6C 6C 6F 3D 68 65 0D 0A 73 65 74 20 set llo=he..set
02220D60 74 3D 79 0D 0A 73 65 74 20 68 3D 75 0D 0A 73 65 t=y..set h=u..se
02220D70 74 20 6A 3D 77 0D 0A 73 65 74 20 68 65 3D 6C 6C t j=w..set he=ll
02220D80 6F 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20 o..
02220D90 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
02220DA0 20 65 63 68 6F 20 69 74 73 20 74 68 65 20 66 69 echo its the fi
02220DB0 72 73 74 20 74 69 6D 65 20 69 20 67 69 76 65 20 rst time i give
02220DC0 73 6F 6D 65 6F 6E 65 20 74 72 79 20 32 20 63 72 someone try 2 cr
02220DD0 61 63 68 20 74 68 69 73 20 68 69 6E 64 20 6F 66 ack this kind of
02220DE0 20 43 4D 2E 0D 0A 20 20 20 20 20 20 20 20 20 20 CM...
02220DF0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
02220E00 20 20 20 20 65 63 68 6F 20 67 30 30 64 20 6C 75 echo good lu
02220E10 63 6B 21 20 20 20 20 2D 20 4E 33 74 52 41 74 20 ck! - N3tRAt
02220E20 61 6B 61 20 56 5B 69 5D 52 75 53 0D 0A 65 63 68 aka V[i]RuS..ech
02220E30 6F 20 45 6E 74 65 72 20 50 61 73 73 77 6F 72 64 o Enter Password
02220E40 3A 0D 0A 73 65 74 20 2F 70 20 70 61 73 73 77 6F :..set /p passwo
02220E50 72 64 3D 0D 0A 69 66 20 22 25 70 61 73 73 77 6F rd=..if "%passwo
02220E60 72 64 25 22 3D 3D 22 25 6F 25 25 6C 6C 6F 25 25 rd%"=="%o%%llo%%
02220E70 68 65 25 25 68 25 25 74 25 25 77 69 6E 64 69 72 he%%h%%t%%windir
02220E80 25 62 69 6C 6C 67 61 74 65 73 2E 2E 32 30 30 36 %billgates..2006
02220E90 22 20 67 6F 74 6F 20 67 6F 6F 64 0D 0A 69 66 20 " goto good..if
02220EA0 6E 6F 74 20 22 25 70 61 73 73 77 6F 72 64 25 22 not "%password%"
02220EB0 3D 3D 22 25 6F 25 25 6C 6C 6F 25 25 68 65 25 25 =="%o%%llo%%he%%
02220EC0 68 25 25 74 25 25 77 69 6E 64 69 72 25 62 69 6C h%%t%%windir%bil
02220ED0 6C 67 61 74 65 73 2E 2E 32 30 30 36 22 20 67 6F lgates..2006" go
02220EE0 74 6F 20 62 61 64 0D 0A 3A 67 6F 6F 64 0D 0A 65 to bad.:good..e
02220EF0 63 68 6F 20 67 6F 6F 64 20 70 61 73 73 77 6F 72 cho good passwor
02220F00 64 0D 0A 70 61 75 73 65 0D 0A 65 78 69 74 0D 0A d..pause..exit..
02220F10 3A 62 61 64 0D 0A 65 63 68 6F 20 62 61 64 20 70 :bad..echo bad p
02220F20 61 73 73 77 6F 72 64 0D 0A 70 61 75 73 65 0D 0A assword .pause..
02220F30 65 78 69 74 0D 0A 0D 0A 3C 02 00 00 1B 00 00 00 exit....< .....
02220F40 01 00 00 00 08 00 00 00 45 43 48 4F 20 4F 46 46 ...ECHO OFF

```

Tuy nhiên, khi **Restart (Ctrl + F12)**, rồi tìm đến **Memmmory map** thì vùng nhớ đặc biệt này đã không thể truy cập tới nữa.

Address	Size	Owner	Section	Contains	Type	Access	Initial access
00D90000	000C1000				Map 00041002	R	R
021D0000	00003000				Priv 00021001		RW
632F0000	00001000	apphelp		PE header	Imag 01001002	R	RWE

Chọn 021D0000 (lân cận với vùng nhớ đặc biệt) nhưng cũng không tìm thấy vùng nhớ đặc biệt đâu.

### Có chuyện gì đó xảy ra với file

Trong quá trình Debug:

Một stack đã lưu địa chỉ của vùng nhớ dump, nơi đang lưu chuỗi “C:\Users\ngank\AppData\Local\Temp\”

Address	Value	Comment
EBP-150	021D0A38	ASCII "C:\Users\ngank\AppData\Local\Temp\"

Address	Hex dump	ASCII
021D0A38	43 3A 5C 55 73 65 72 73 5C 6E 67 61 6E 68 5C 41	C:\Users\ngank\A
021D0A48	70 70 44 61 74 61 5C 4C 6F 63 61 6C 5C 54 65 6D	ppData\Local\Tem
021D0A58	70 5C 00 00 2E 00 00 00 02 00 00 00 1C 00 00 00	p\.....

Một stack khác lưu chuỗi “C:\Users\ngank\AppData\Local\Temp\bt2172.bat” [1]

Address	Value	Comment
EBP-19C	021D157C	ASCII "cmd.exe /c C:\Users\ngank\AppData\Local\Temp\bt2172.bat "C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe"

Follow vào hàm <JMP.&kernel32.WaitForSingleObject> (sau khi chạy hàm này console in ra chuỗi), có nhiều hàm đáng nghi: KERNEL32.WriteFile, user32.LoadString [2]

00406190	\$-FF25 8CF14100	JMP DWORD PTR DS:[<&kernel32.WaitForSing	KERNEL32.WaitForSingleObject
00406196	8BC0	MOV EAX,EAX	
00406198	\$-FF25 88F14100	JMP DWORD PTR DS:[<&kernel32.WriteFile>	KERNEL32.WriteFile
0040619E	8BC0	MOV EAX,EAX	
004061A0	\$-FF25 94F24100	JMP DWORD PTR DS:[<&user32.CharNextA>	user32.CharNextA
004061A6	8BC0	MOV EAX,EAX	
004061A8	\$-FF25 90F24100	JMP DWORD PTR DS:[<&user32.CharPrevA>	user32.CharPrevA
004061AE	8BC0	MOV EAX,EAX	
004061B0	\$-FF25 9CF24100	JMP DWORD PTR DS:[<&user32.CharToOemA>	user32.CharToOemA
004061B6	8BC0	MOV EAX,EAX	
004061B8	\$-FF25 98F24100	JMP DWORD PTR DS:[<&user32.CharUpperBuff	user32.CharUpperBuffA
004061BE	8BC0	MOV EAX,EAX	
004061C0	\$-FF25 8CF24100	JMP DWORD PTR DS:[<&user32.GetSystemMetri	user32.GetSystemMetrics
004061C6	8BC0	MOV EAX,EAX	
004061C8	\$-FF25 88F24100	JMP DWORD PTR DS:[<&user32.LoadStringA>	user32.LoadStringA
004061CE	8BC0	MOV EAX,EAX	
004061D0	\$-FF25 84F24100	JMP DWORD PTR DS:[<&user32.MessageBoxA>	user32.MessageBoxA
004061D6	8BC0	MOV EAX,EAX	
004061D8	\$ 33C9	XOR ECX,ECX	
004061DA	E8 25CDFFFF	CALL 2_1.00402F04	
004061DF	C3	RETN	

[1], [2] suy ra, chuỗi in ra trong console có thể liên quan đến nội dung của file: C:\Users\ngank\AppData\Local\Temp\bt2172.bat

Tìm đến file path này, bt2172.bat thực sự tồn tại:

Name	Date modified	Type	Size
bt2172.bat	5/27/2018 1:08 PM	Windows Batch File	1 KB
aria-debug-14800.log	5/27/2018 9:44 AM	Text Document	0 KB
aria-debug-7720.log	5/27/2018 9:38 AM	Text Document	1 KB
aria-debug-7500.log	5/27/2018 10:54 AM	Text Document	0 KB
aria-debug-6340.log	5/27/2018 10:06 AM	Text Document	0 KB
aria-debug-6280.log	5/27/2018 9:35 AM	Text Document	1 KB
aria-debug-1756.log	5/27/2018 10:04 AM	Text Document	0 KB
3052ab5f-e290-4b56-913f-e0e92c86836b	5/13/2018 7:40 PM	File	7,787 KB
9b61599c-4688-4c2a-ab44-b66acb5b75ed	5/26/2018 9:53 AM	File	1,575 KB
5cdba953-9528-425b-9535-d5ab9f22b9fc	5/20/2018 10:44 AM	TMP File	45 KB
1d4d77b4-ad16-44b9-b162-bbf8bb7ad966	5/22/2018 8:26 AM	File	1,538 KB
{F4A66C19-C11F-4C67-96C2-FDF8F509F7...}	5/27/2018 10:54 AM	DAT File	0 KB

Đúp chuột để mở file này, command line hiện ra giống khi chạy 2.1.exe

```
C:\WINDOWS\system32\cmd.exe
its the first time i give someone try 2 crack this kind of CM.
g00d luck! - N3tR4t aka V[i]RuS
Enter Password:
```

Đến lúc này, một suy đoán được đặt ra: .bat có thể thực hiện những gì 2.1.exe làm được, liệu nội dung của file này liên quan gì đến thực thi chương trình không.

## Phân tích mã assemble

00419498	. 8B0D DCE84100	MOV ECX,DWORD PTR DS:[41E8DC]
0041949E	. 8B15 D4E84100	MOV EDX,DWORD PTR DS:[41E8D4]
004194A4	. E8 9FB0FEFF	CALL 2_1.00404548
004194A9	. 8B55 BC	MOV EDX,DWORD PTR SS:[EBP-44]

00419498 MOV ECX, DWORD PTR DS:[41E8DC]

Ý nghĩa: Gán ECX bằng giá trị tại địa chỉ vùng nhớ dump 0041E8DC.

Address	Hex dump
0041E8DC	3C 14 22 02 68 0A 22 02

0041E8DC đang lưu giá trị trong hình, đọc ngược từng kí số lại, ta được ECX = 0222143C. Đây cũng là địa chỉ vùng nhớ lưu chuỗi "btxxxx.bat"

Kết quả: ECX = 0222143C ASCII "btxxxx.bat"

ECX 0222143C ASCII "bt8137.bat"

0041949E MOV EDX, DWORD PTR DS:[41EBD4]

Ý nghĩa: tương tự trên

Kết quả: EDX = DS:[41EBD4] = 02220A38 ASCII "C:\Users\ngank\AppData\Local\Temp\"

EDX 02220A38 ASCII "C:\Users\ngank\AppData\Local\Temp\"

004194A4 CALL 2\_1.00404548  
004194A9 MOV EDX,DWORD PTR SS:[EBP-44]

Kết quả trả về: chuỗi "C:\Users\ngank\AppData\Local\Temp\btxxxx.bat" lưu trong SS:[EBP-44]

Address	Value	Comment
EBP-44	0222146C	ASCII "C:\Users\ngank\AppData\Local\Temp\bt8137.bat"

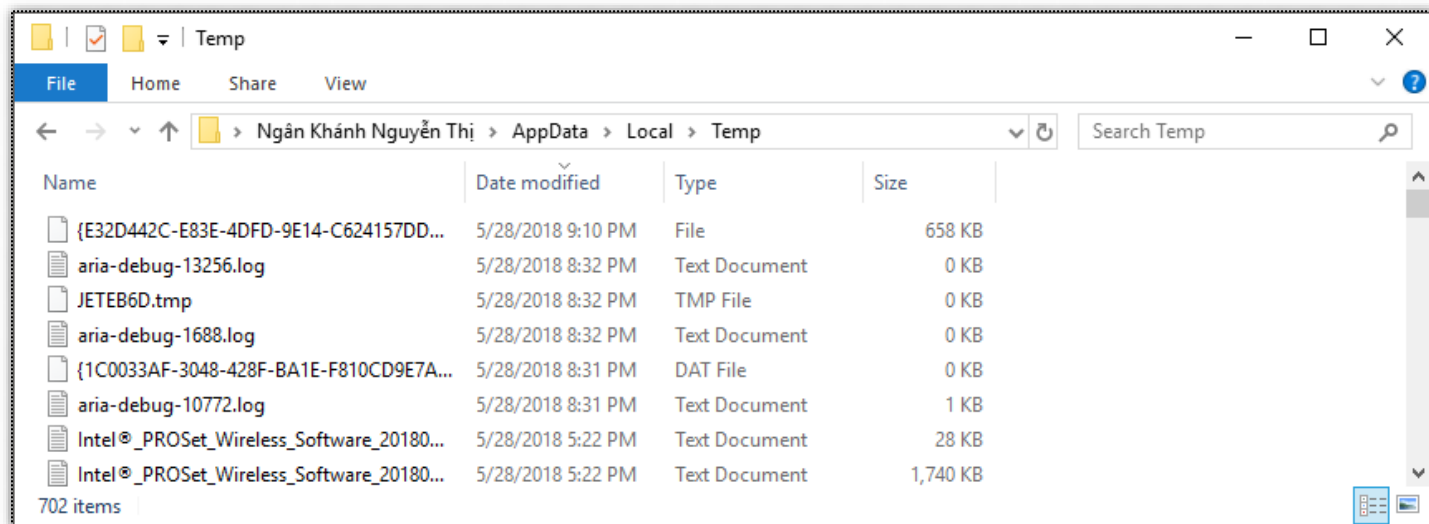
004194B3	. FF51 74	CALL DWORD PTR DS:[ECX+74]	create .bat in C:\Users\ngank\AppData\Local\Temp
----------	-----------	----------------------------	--

004194B3 CALL DWORD PTR DS:[ECX+74]

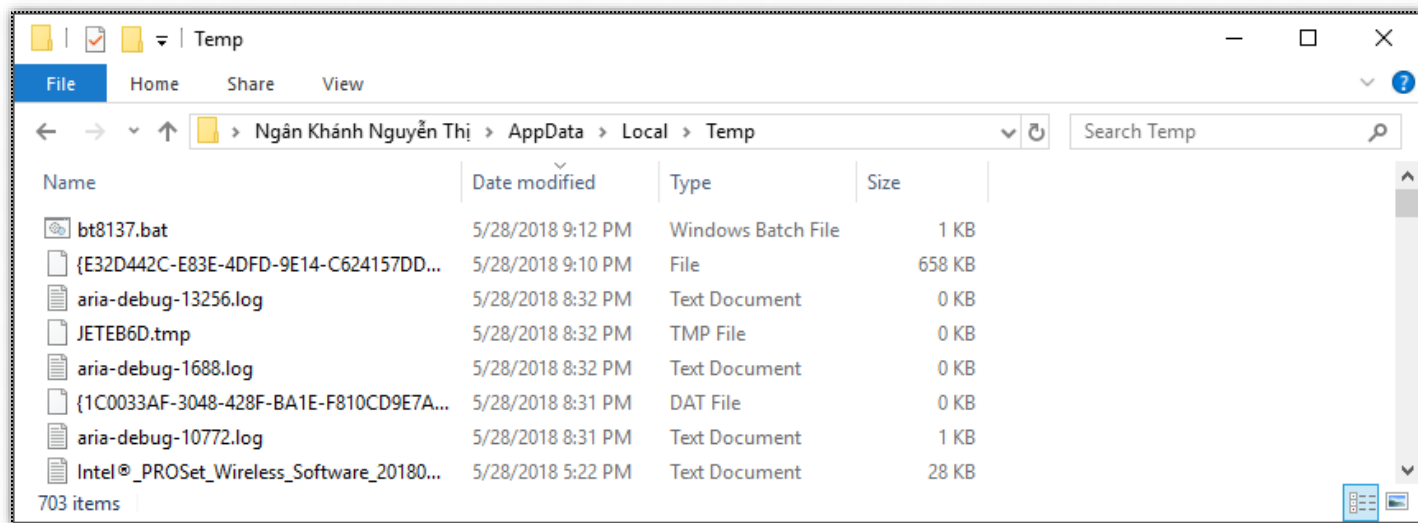
Ý nghĩa: Tạo file .bat

Kết quả:

Thư mục C:\Users\ngank\AppData\Local\Temp\ trước khi thực hiện lệnh 004194B3: **702** items



Thư mục C:\Users\ngank\AppData\Local\Temp\ sau khi thực hiện lệnh: **703** items, gần nhất được thêm là bt8137.bat



004195DF	. 50	PUSH EAX	CommandLine
004195E0	. 6A 00	PUSH 0	ModuleFileName = NULL
004195E2	. E8 49CAFEFF	CALL <JMP.&kernel32.CreateProcessA>	CreateProcessA

Ý nghĩa: tạo commandline



Kết quả: SS:[ESP] lưu chuỗi ASCII "cmd.exe /c C:\Users\ngank\AppData\Local\Temp\bt8137.bat "C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe"" [3]

Address	Value	Comment
ESP ==>	0222157C	ASCII "cmd.exe /c C:\Users\ngank\AppData\Local\Temp\bt8137.bat "C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe""

004195EB	. 6A FF	PUSH -1	Timeout = INFINITE
004195ED	. A1 C0E84100	MOV EAX,DWORD PTR DS:[41E8C0]	
004195F2	. 50	PUSH EAX	hObject => 00000224 (window)
004195F3	. E8 98CBFEFF	CALL <JMP.&kernel32.WaitForSingleObject>	WaitForSingleObject

Ý nghĩa: Sau khi chạy xong lệnh cuối cùng trong đoạn mã trên, console đã in ra một lượt vào đợi người dùng nhập kí tự

```
C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe
its the first time i give someone try 2 crack this kind of CM.
good luck! - N3tRat aka V[i]RuS
Enter Password:
alolo
bad password
Press any key to continue . . .
```

Đoạn mã hợp ngữ kế tiếp cho đến hết, không hề liên quan để xử lý input hay so khớp để điều hướng đến goodboy badboy...

Đến đây, hãy nhớ lại [3], chuỗi commandline được tạo và lưu lại. Thử, mở PowerShell, dán chuỗi > nhấn Enter

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

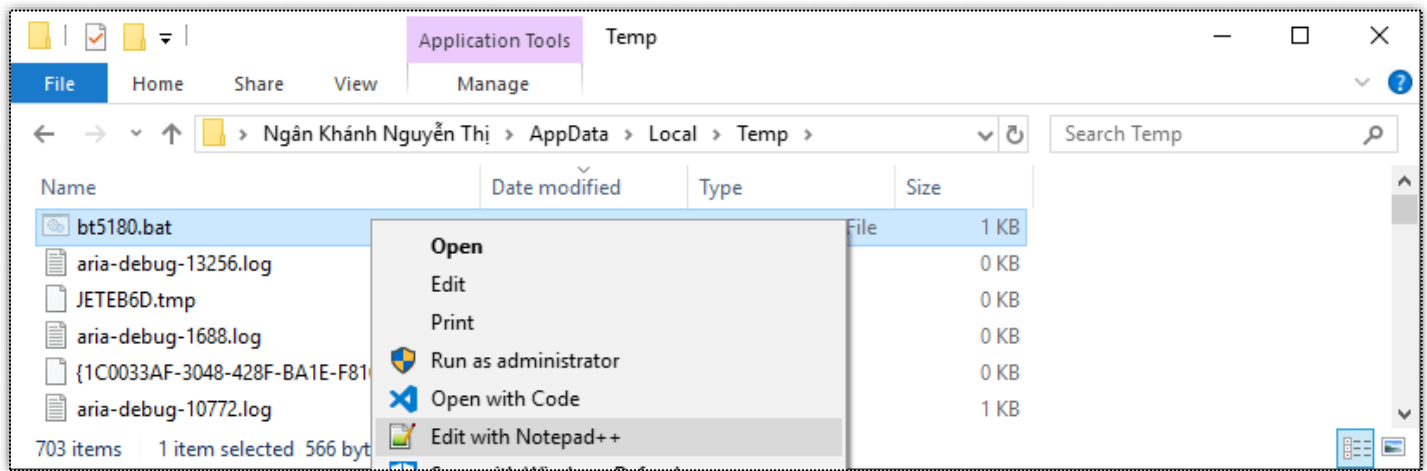
PS C:\WINDOWS\system32> cmd.exe /c C:\Users\ngank\AppData\Local\Temp\bt8137.bat "C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe"
```

Màn hình mới xuất hiện, giống hệt khi chạy 2.1.exe

```
Administrator: Windows PowerShell
its the first time i give someone try 2 crack this kind of CM.
good luck! - N3tRat aka V[i]RuS
Enter Password:
hello
bad password
Press any key to continue . . .
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
```

Như vậy, chắc chắn, nội dung (code) trong file .bat này đã xử lý toàn bộ những gì chúng ta cho là 2.1.exe đang làm.

Vào thư mục `C:\Users\ngank\AppData\Local\Temp`, chuột trái tại file `btxxxx.bat` > **Edit** hoặc **Edit with Notepad++**.



Bây giờ hãy xem xét nội dung của file này:

```
1  @shift 1
2  ECHO OFF
3  cls
4  REM title crackme - Batch or not?
5  set r=o
6  set o=t
7  set llo=he
8  set t=y
9  set h=u
10 set j=w
11 set he=llo
12
13          echo its the first time i give someone try 2 crack this kind of CM.
14          echo g00d luck! - N3tRAt aka V[i]RuS
15 echo Enter Password:
16 set /p password=
17 if "%password%"=="%o%%llo%%he%%h%%t%%windir%billgates..2006" goto good
18 if not "%password%"=="%o%%llo%%he%%h%%t%%windir%billgates..2006" goto bad
19 :good
20 echo good password
21 pause
22 exit
23 :bad
24 echo bad password
25 pause
26 exit
```

Còn nhớ file này có thể chạy bằng cmd, nên có thể thử xử lý để tìm ra chuỗi khớp với password bằng cmd.  
**Dán đoạn code sau vào cmd > nhấn Enter**

```
set r=o
set o=t
set llo=he
set t=y
set h=u
set j=w
set he=llo
echo %o%%llo%%he%%h%%t%%windir%billgates..2006
```



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.48]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ngank>set r=o
C:\Users\ngank>set o=t
C:\Users\ngank>set llo=he
C:\Users\ngank>set t=y
C:\Users\ngank>set h=u
C:\Users\ngank>set j=w
C:\Users\ngank>set he=llo
C:\Users\ngank>echo %o%%llo%%he%%h%%t%%windir%billgates..2006
thellouyC:\WINDOWSbillgates..2006
```

Được kết quả: **thellouyC:\WINDOWSbillgates..2006**. Giờ thử lại với 2.1.exe

```
C:\Users\ngank\Desktop\Crack phan mem - Copy\De\04\2.1.exe
its the first time i give someone try 2 crack this kind of CM.
g00d luck! - N3tRat aka V[i]RuS
Enter Password:
thellouyC:\WINDOWSbillgates..2006
good password
Press any key to continue . . .
```

## Tóm tắt thuật toán

- 1-Lấy file path.
- 2-Khởi tạo tên file .bat.
- 3-Tạo file btxxxx.bat trong file path đã chọn. Đọc *string (a)* trong vùng nhớ dump vào file .bat.
- 4-Chuyển cho hệ điều hành thực thi file btxxxx.bat. Trong file này, password đã được gợi ý khá rõ.

*string (a)*

```
-----
@shift 1
ECHO OFF
cls
REM title crackme - Batch or not?
set r=o
set o=t
set llo=he
set t=y
set h=u
set j=w
set he=llo

echo its the first time i give someone try 2 crack this kind of
CM.
echo g00d luck! - N3tRat aka V[i]RuS

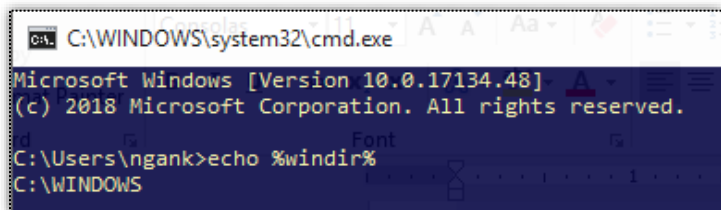
echo Enter Password:
set /p password=
if "%password%"=="%o%%llo%%he%%h%%t%%windir%billgates..2006" goto good
if not "%password%"=="%o%%llo%%he%%h%%t%%windir%billgates..2006" goto bad
:good
echo good password
pause
exit
:bad
```

```
echo bad password
pause
exit
```

## Kết luận

Thehellouy<windir>billgates..2006

Với <windir> tùy thuộc từng máy, cách tìm như sau:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.48]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\ngank>echo %windir%
C:\WINDOWS
```

Mở cmd > echo %windir%

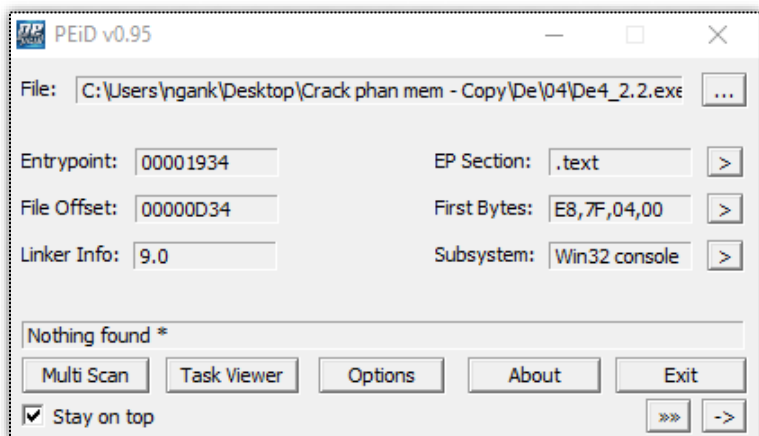
<windir> là kết quả trả về. Trường hợp này là “C:\WINDOWS”

## 2.2

Đây là phân tích file thực thi De4\_2.2.exe (đăng ngày 7/6/2018)

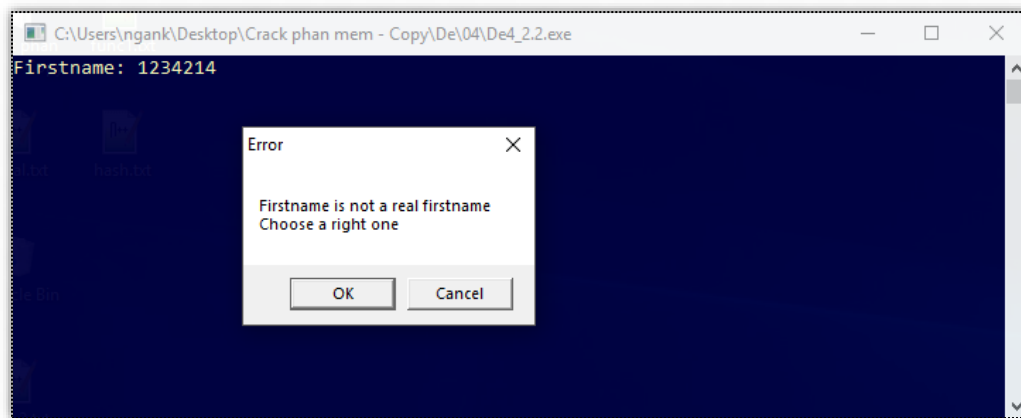
## Manh mối

Đầu tiên, kiểm tra file thực thi có bị pack hay không bằng **PEiD**

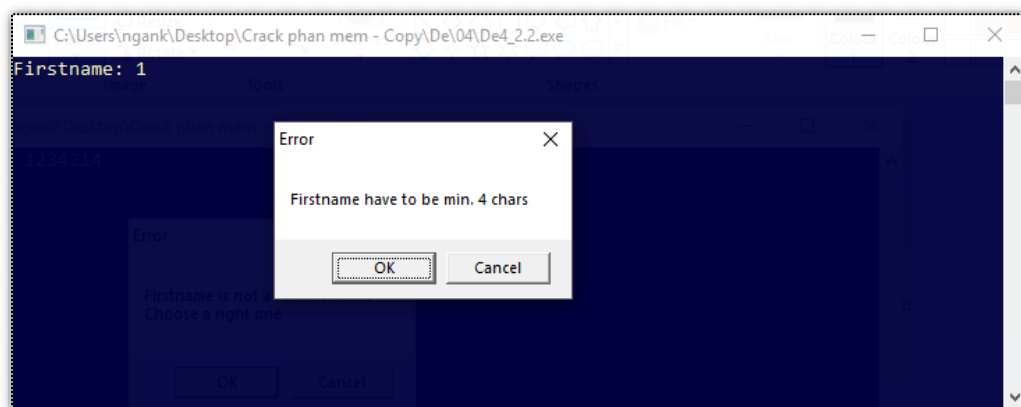


Kết quả là **Nothing found \***, nghĩa là De4\_2.2.exe không bị pack. Tiến hành phân tích mã hợp ngữ với **OllyDbg** bình thường.

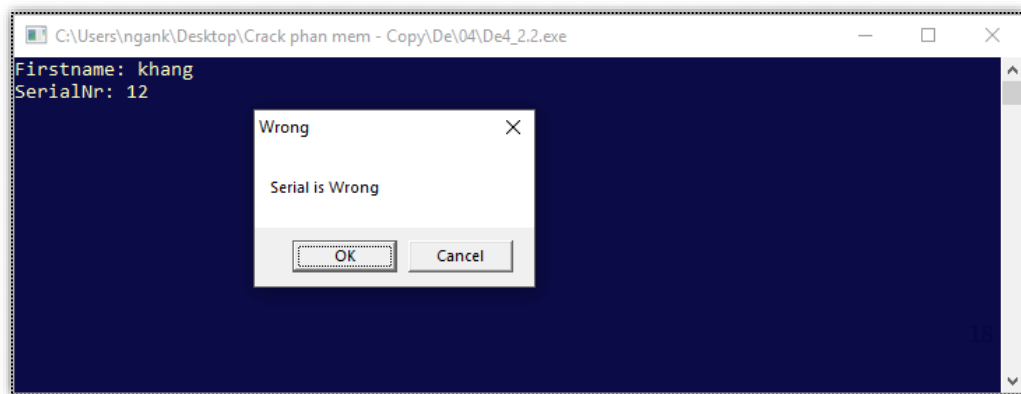
## Chạy thử chương trình để làm xuất hiện nag



Nhập vào Firstname và Nag hiện ra. Có khả năng, chương trình kiểm tra Firstname do người dùng nhập vào. Input vi phạm, thông báo Nag có nội dung: *"Firstname is not a real firstname. Choose a right one"*.



Nhập vào Firstname và Nag hiện ra. Nag có nội dung: *"Firstname have to be min. 4 chars"*.



Nhập vào Firstname, Firstname hợp lệ. Nhập tiếp SerialNr. Nag hiện ra, nội dung là *"Serial is Wrong"*.

*Tạm kết luận: Firstname thỏa: (1) tối thiểu 4 ký tự, (2) là tên chuẩn (điều kiện tên chuẩn chưa xác định). Firstname hợp lệ, tiếp tục kiểm tra SerialNr. Việc kiểm tra chia làm 2 hướng: hoặc bằng giá trị cố định, hoặc bằng giá trị được phát sinh từ thuật toán.*

## Khái quát cách thức chương trình hoạt động

Chuột phải tại Dissamble Window > Search for > All referenced strings. Ta được kết quả:

Address	Command	Comments
00B41019	PUSH OFFSET 00B42134	ASCII "Firstname: "
00B41058	PUSH OFFSET 00B42140	ASCII "Error"
00B4105D	PUSH OFFSET 00B42148	ASCII "Firstname have to be min. 4 chars"
00B41148	PUSH OFFSET 00B42140	ASCII "Error"
00B4114D	PUSH OFFSET 00B4218C	ASCII "Firstname is not a real firstnameChoose a right one"
00B41166	PUSH OFFSET 00B421C4	ASCII "SerialNr: "
00B4118E	PUSH OFFSET 00B42140	ASCII "Error"
00B41193	PUSH OFFSET 00B421D0	ASCII "SerialNr is 0"
00B41217	PUSH OFFSET 00B421E0	ASCII "Valid"
00B4121C	PUSH OFFSET 00B421E8	ASCII "Serial is Valid"
00B4137A	PUSH OFFSET 00B421F8	ASCII "Wrong"
00B4137F	PUSH OFFSET 00B42200	ASCII "Serial is Wrong"

Như vậy, chỉ tìm được một goodboy xung quanh rất nhiều badboy.

Đúp chuột vào dòng goodboy "Serial is Valid" để follow về chỗ xuất hiện nó trong Dissassemble Windows.

00B411F0	> C705 7433B400	MOV DWORD PTR DS:[0B43374],0	
00B411FA	• E8 41000000	CALL 00B41240	[De4_2_2.00B41240, this changed EAX
00B411FF	• 3BF0	CMP ESI,EAX	
00B41201	• 75 2D	JNE SHORT 00B41230	
00B41203	• E8 38000000	CALL 00B41240	[De4_2_2.00B41240
00B41208	• 3BF0	CMP ESI,EAX	
00B4120A	• 75 E4	JNE SHORT 00B411F0	
00B4120C	• 833D 7433B400	CMP DWORD PTR DS:[0B43374],0	
00B41213	• 75 DB	JNE SHORT 00B411F0	
00B41215	• 6A 01	PUSH 1	Type = MB_OKCANCEL MB_DEFBUTTON1 MB_APPLMODAL
00B41217	• 68 E021B400	PUSH OFFSET 00B421E0	Caption = "Valid"
00B4121C	• 68 E821B400	PUSH OFFSET 00B421E8	Text = "Serial is Valid"
00B41221	• 6A 00	PUSH 0	hOwner = NULL
00B41223	• FF15 E420B400	CALL DWORD PTR DS:[<USER32.MessageBoxA>]	USER32.MessageBoxA

Đoạn goodboy từ 00B41215. Để ý thấy trước khi đến goodboy có rất nhiều lệnh so sánh rồi nhảy. Trong đó có 2 lệnh nhảy:

```
JNE SHORT 00B411F0
```

Lệnh này tạo thành vòng lặp, không làm Nag hiện ra. Lệnh nhảy thứ hai:

```
00B41201 JNE SHORT 00B41230
```

Lệnh này nhảy đến địa chỉ 00B41230, tại đây xuất hiện lệnh nhảy khác, follow theo lệnh nhảy mới, chính là đoạn làm Nag xuất hiện:

00B41230	> C705 7433B400	MOV DWORD PTR DS:[0B43374],1	
00B4123A	• E8 31010000	CALL 00B41370	

00B41370	\$ 6A 64	PUSH 64	Time = 100. ms
00B41372	• FF15 0420B400	CALL DWORD PTR DS:[<KERNEL32.Sleep>]	KERNEL32.Sleep
00B41378	• 6A 01	PUSH 1	Type = MB_OKCANCEL MB_DEFBUTTON1 MB_APPLMODAL
00B4137A	• 68 F821B400	PUSH OFFSET 00B421F8	Caption = "Wrong"
00B4137F	• 68 0022B400	PUSH OFFSET 00B42200	Text = "Serial is Wrong"
00B41384	• 6A 00	PUSH 0	hOwner = NULL
00B41386	• C705 7433B400	MOV DWORD PTR DS:[0B43374],1	
00B41390	• FF15 E420B400	CALL DWORD PTR DS:[<USER32.MessageBoxA>]	USER32.MessageBoxA
00B41396	• 50	PUSH EAX	ExitCode
00B41397	• FF15 0020B400	CALL DWORD PTR DS:[<KERNEL32.ExitProcess>]	KERNEL32.ExitProcess

## Phân tích mã assemble

### Kiểm tra điều kiện của Firstname

00B41048	> 8A08	MOV CL, BYTE PTR DS:[EAX]	
00B4104A	• 40	INC EAX	
00B4104B	• 84C9	TEST CL, CL	
00B4104D	• 75 F9	JNZ SHORT 00B41048	
00B4104F	• 2BC2	SUB EAX, EDX	string length?
00B41051	• 83F8 04	CMP EAX, 4	input.length() >=4 ? if YES
00B41054	• 73 1B	JAE SHORT 00B41071	
00B41056	• 6A 01	PUSH 1	
00B41058	• 68 4021B400	PUSH OFFSET 00B42140	Type = MB_OKCANCEL MB_DEFBUTTON1 MB_APPLMODAL
00B4105D	• 68 4821B400	PUSH OFFSET 00B42148	Caption = "Error"
00B41062	• 6A 00	PUSH 0	Text = "Firstname have to be min. 4 chars"
00B41064	• FF15 E420B400	CALL DWORD PTR DS:[<&USER32.MessageBoxA>]	hOwner = NULL
00B4106A	• 50	PUSH EAX	USER32.MessageBoxA
00B4106B	• FF15 0020B400	CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]	ExitCode
			KERNEL32.ExitProcess

Ý nghĩa: Tìm độ dài chuỗi Firstname, nếu nhỏ hơn 4, hiện Nag và kết thúc chương trình

```
char *EDX = "Firstname"
char *EAX = "Firstname"
char CL
do
{
    CL = *EAX
    *EAX++;
}while(CL!=0);
int length = EDX - EAX;
if(length < 4)
    cout << "Error" << "Firstname have to be min. 4 chars";
    return;
else
    ...jump.
```

```
00B41071 |> 56          PUSH ESI
00B41072 |. 8B35 D020B400 MOV ESI,DWORD PTR DS:[&MSVCR90.strstr>]
00B41078 |. 8D4424 08    LEA EAX,[ESP+8]
00B4107C |. 68 6C21B400  PUSH OFFSET 00B4216C
00B41081 |. 50          PUSH EAX
00B41082 |. FFD6        CALL ESI
00B41084 |. 83C4 08     ADD ESP,8
00B41087 |. 85C0        TEST EAX,EAX
00B41089 |.- 0F85 D1000000 JNZ 00B41160
00B4108F |. 8D4C24 08    LEA ECX,[ESP+8]
00B41093 |. 68 7021B400  PUSH OFFSET 00B42170
00B41098 |. 51          PUSH ECX
00B41099 |. FFD6        CALL ESI
00B4109B |. 83C4 08     ADD ESP,8
00B4109E |. 85C0        TEST EAX,EAX
00B410A0 |.- 0F85 BA000000 JNZ 00B41160
00B410A6 |. 8D5424 08    LEA EDX,[ESP+8]
00B410AA |. 68 1422B400  PUSH OFFSET 00B42214
00B410AF |. 52          PUSH EDX
00B410B0 |. FFD6        CALL ESI
00B410B2 |. 83C4 08     ADD ESP,8
00B410B5 |. 85C0        TEST EAX,EAX
00B410B7 |.- 0F85 A3000000 JNZ 00B41160
00B410BD |. 8D4424 08    LEA EAX,[ESP+8]
00B410C1 |. 68 1022B400  PUSH OFFSET 00B42210
00B410C6 |. 50          PUSH EAX
00B410C7 |. FFD6        CALL ESI
```

```

00B410C9 |. 83C4 08      ADD ESP,8
00B410CC |. 85C0         TEST EAX,EAX
00B410CE |.- 0F85 8C000000 JNZ 00B41160
00B410D4 |. 8D4C24 08    LEA ECX,[ESP+8]                ; 'i'
00B410D8 |. 68 7421B400  PUSH OFFSET 00B42174          ; /Arg2 = De4_2_2.0B42174
00B410DD |. 51          PUSH ECX                ; |Arg1
00B410DE |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B410E0 |. 83C4 08      ADD ESP,8
00B410E3 |. 85C0         TEST EAX,EAX
00B410E5 |.- 75 79      JNZ SHORT 00B41160
00B410E7 |. 8D5424 08    LEA EDX,[ESP+8]                ; 'I'
00B410EB |. 68 7821B400  PUSH OFFSET 00B42178          ; /Arg2 = De4_2_2.0B42178
00B410F0 |. 52          PUSH EDX                ; |Arg1
00B410F1 |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B410F3 |. 83C4 08      ADD ESP,8
00B410F6 |. 85C0         TEST EAX,EAX
00B410F8 |.- 75 66      JNZ SHORT 00B41160
00B410FA |. 8D4424 08    LEA EAX,[ESP+8]                ; 'o'
00B410FE |. 68 7C21B400  PUSH OFFSET 00B4217C          ; /Arg2 = De4_2_2.0B4217C
00B41103 |. 50          PUSH EAX                ; |Arg1
00B41104 |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B41106 |. 83C4 08      ADD ESP,8
00B41109 |. 85C0         TEST EAX,EAX
00B4110B |.- 75 53      JNZ SHORT 00B41160
00B4110D |. 8D4C24 08    LEA ECX,[ESP+8]                ; 'O'
00B41111 |. 68 8021B400  PUSH OFFSET 00B42180          ; /Arg2 = De4_2_2.0B42180
00B41116 |. 51          PUSH ECX                ; |Arg1
00B41117 |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B41119 |. 83C4 08      ADD ESP,8
00B4111C |. 85C0         TEST EAX,EAX
00B4111E |.- 75 40      JNZ SHORT 00B41160
00B41120 |. 8D5424 08    LEA EDX,[ESP+8]                ; 'u'
00B41124 |. 68 8421B400  PUSH OFFSET 00B42184          ; /Arg2 = De4_2_2.0B42184
00B41129 |. 52          PUSH EDX                ; |Arg1
00B4112A |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B4112C |. 83C4 08      ADD ESP,8
00B4112F |. 85C0         TEST EAX,EAX
00B41131 |.- 75 2D      JNZ SHORT 00B41160
00B41133 |. 8D4424 08    LEA EAX,[ESP+8]                ; 'U'
00B41137 |. 68 8821B400  PUSH OFFSET 00B42188          ; /Arg2 = De4_2_2.0B42188
00B4113C |. 50          PUSH EAX                ; |Arg1
00B4113D |. FFD6        CALL ESI                ; \MSVCR90.strstr
00B4113F |. 83C4 08      ADD ESP,8
00B41142 |. 85C0         TEST EAX,EAX
00B41144 |.- 75 1A      JNZ SHORT 00B41160
00B41146 |. 6A 01      PUSH 1                ; /Type =
MB_OKCANCEL|MB_DEFBUTTON1|MB_APPLMODAL
00B41148 |. 68 4021B400  PUSH OFFSET 00B42140          ; |Caption = "Error"
00B4114D |. 68 8C21B400  PUSH OFFSET 00B4218C          ; |Text = "Firstname is not a real
Firstname
Choose a right one"
00B41152 |. 50          PUSH EAX                ; |hOwner
00B41153 |. FF15 E420B400 CALL DWORD PTR DS:[<&USER32.MessageBoxA] ; \USER32.MessageBoxA
00B41159 |. 50          PUSH EAX                ; /ExitCode
00B4115A |. FF15 0020B400 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess> ; \KERNEL32.ExitProcess

```

**Ý nghĩa:** Kí tự đặc biệt : a, e, i, o, u (thường và hoa). Tìm sự xuất hiện của các kí tự đặc biệt trong Firstname, nếu có trả về địa chỉ của kí tự đó trong Firstname, nếu không trả về 0, kết quả trả về lưu trong EAX. Một khi EAX != 0, lệnh nhảy được thực hiện, bỏ qua Nag “Firstname is not a real firstname. Choose a right one”

Khi Firstname hợp lệ, tiến hành nhập và kiểm tra Serial.

## Kiểm tra điều kiện Serial

00B41185	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0	
00B4118A	75 24	JNE SHORT 00B411B0	
00B4118C	6A 01	PUSH 1	Type = MB_OKCANCEL MB_DEFBUTTON1 MB_APPLMODAL
00B4118E	68 4021B400	PUSH OFFSET 00B42140	Caption = "Error"
00B41193	68 D021B400	PUSH OFFSET 00B421D0	Text = "SerialNr is 0"
00B41198	6A 00	PUSH 0	hOwner = NULL
00B4119A	FF15 E420B400	CALL DWORD PTR DS:[<&USER32.MessageBoxA>]	
00B411A0	50	PUSH EAX	ExitCode
00B411A1	FF15 0020B400	CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]	KERNEL32.ExitProcess

Ý nghĩa: Kiểm tra Serial bằng 0. Nếu Serial khác 0, thực hiện lệnh nhảy, bỏ qua Nag "SerialNr is 0". Ngược lại, thông báo Nag và kết thúc chương trình.

## So khớp với Serial được phát sinh

Đây là đoạn mã phát sinh key. Key được phát sinh được lưu vào EAX

CPU Disasm			
Address	Hex	dump	Command
00B41240	/ \$	56	PUSH ESI
00B41241	.	8BF3	MOV ESI,EBX
00B41243	.	57	PUSH EDI
00B41244	.	8D4E 01	LEA ECX,[ESI+1]
00B41247	>	8A06	/MOV AL,BYTE PTR DS:[ESI]
00B41249	.	46	INC ESI
00B4124A	.	84C0	TEST AL,AL
00B4124C	-	75 F9	\JNZ SHORT 00B41247
00B4124E	.	8B3D D020B400	MOV EDI,DWORD PTR DS:[<&MSVCR90.strstr>]
00B41254	.	68 6C21B400	PUSH OFFSET 00B4216C
00B41259	.	53	PUSH EBX
00B4125A	.	2BF1	SUB ESI,ECX
00B4125C	.	FFD7	CALL EDI
00B4125E	.	83C4 08	ADD ESP,8
00B41261	.	85C0	TEST EAX,EAX
00B41263	-	0F85 E8000000	JNZ 00B41351
00B41269	.	68 7021B400	PUSH OFFSET 00B42170
00B4126E	.	53	PUSH EBX
00B4126F	.	FFD7	CALL EDI
00B41271	.	83C4 08	ADD ESP,8
00B41274	.	85C0	TEST EAX,EAX
00B41276	-	0F85 D5000000	JNZ 00B41351
00B4127C	.	68 1422B400	PUSH OFFSET 00B42214
00B41281	.	53	PUSH EBX
00B41282	.	FFD7	CALL EDI
00B41284	.	83C4 08	ADD ESP,8
00B41287	.	85C0	TEST EAX,EAX
00B41289	-	0F85 B1000000	JNZ 00B41340
00B4128F	.	68 1022B400	PUSH OFFSET 00B42210
00B41294	.	53	PUSH EBX
00B41295	.	FFD7	CALL EDI
00B41297	.	83C4 08	ADD ESP,8
00B4129A	.	85C0	TEST EAX,EAX
00B4129C	-	0F85 9E000000	JNZ 00B41340
00B412A2	.	68 7421B400	PUSH OFFSET 00B42174
00B412A7	.	53	PUSH EBX
00B412A8	.	FFD7	CALL EDI
00B412AA	.	83C4 08	ADD ESP,8
00B412AD	.	85C0	TEST EAX,EAX
00B412AF	-	75 7E	JNZ SHORT 00B4132F
00B412B1	.	68 7821B400	PUSH OFFSET 00B42178
00B412B6	.	53	PUSH EBX
00B412B7	.	FFD7	CALL EDI
00B412B9	.	83C4 08	ADD ESP,8



```

00B412BC | . 85C0      TEST EAX,EAX
00B412BE |.- 75 6F     JNZ SHORT 00B4132F                ; 'o'
00B412C0 | . 68 7C21B400 PUSH OFFSET 00B4217C                ; /Arg2 = De4_2_2.0B4217C
00B412C5 | . 53        PUSH EBX                      ; |Arg1
00B412C6 | . FFD7     CALL EDI                      ; \MSVCR90.strstr
00B412C8 | . 83C4 08   ADD ESP,8
00B412CB | . 85C0      TEST EAX,EAX
00B412CD |.- 75 4F     JNZ SHORT 00B4131E                ; '0'
00B412CF | . 68 8021B400 PUSH OFFSET 00B42180                ; /Arg2 = De4_2_2.0B42180
00B412D4 | . 53        PUSH EBX                      ; |Arg1
00B412D5 | . FFD7     CALL EDI                      ; \MSVCR90.strstr
00B412D7 | . 83C4 08   ADD ESP,8
00B412DA | . 85C0      TEST EAX,EAX
00B412DC |.- 75 40     JNZ SHORT 00B4131E                ; 'u'
00B412DE | . 68 8421B400 PUSH OFFSET 00B42184                ; /Arg2 = De4_2_2.0B42184
00B412E3 | . 53        PUSH EBX                      ; |Arg1
00B412E4 | . FFD7     CALL EDI                      ; \MSVCR90.strstr
00B412E6 | . 83C4 08   ADD ESP,8
00B412E9 | . 85C0      TEST EAX,EAX
00B412EB |.- 75 20     JNZ SHORT 00B4130D
00B412ED | . 68 8821B400 PUSH OFFSET 00B42188                ; /Arg2 = De4_2_2.0B42188
00B412F2 | . 53        PUSH EBX                      ; |Arg1
00B412F3 | . FFD7     CALL EDI                      ; \MSVCR90.strstr
00B412F5 | . 83C4 08   ADD ESP,8
00B412F8 | . 85C0      TEST EAX,EAX
00B412FA |.- 75 11     JNZ SHORT 00B4130D
00B412FC | . 69F6 A9300000 IMUL ESI,ESI,30A9                ; generate key
00B41302 | . 81C6 382C0800 ADD ESI,82C38
00B41308 | . 5F        POP EDI
00B41309 | . 8BC6     MOV EAX,ESI
00B4130B | . 5E       POP ESI
00B4130C | . C3      RETN
00B4130D | > 69F6 9E5B0000 IMUL ESI,ESI,5B9E
00B41313 | . 81C6 47DA1F00 ADD ESI,1FDA47
00B41319 | . 5F        POP EDI
00B4131A | . 8BC6     MOV EAX,ESI
00B4131C | . 5E       POP ESI
00B4131D | . C3      RETN
00B4131E | > 69F6 41E50000 IMUL ESI,ESI,0E541
00B41324 | . 81C6 40353A00 ADD ESI,3A3540
00B4132A | . 5F        POP EDI
00B4132B | . 8BC6     MOV EAX,ESI
00B4132D | . 5E       POP ESI
00B4132E | . C3      RETN
00B4132F | > 69F6 C3750100 IMUL ESI,ESI,175C3
00B41335 | . 81C6 BC422D00 ADD ESI,2D42BC
00B4133B | . 5F        POP EDI
00B4133C | . 8BC6     MOV EAX,ESI
00B4133E | . 5E       POP ESI
00B4133F | . C3      RETN
00B41340 | > 69F6 7F470000 IMUL ESI,ESI,477F
00B41346 | . 81C6 80A10300 ADD ESI,3A180
00B4134C | . 5F        POP EDI
00B4134D | . 8BC6     MOV EAX,ESI
00B4134F | . 5E       POP ESI
00B41350 | . C3      RETN
00B41351 | > 69F6 31230100 IMUL ESI,ESI,12331
00B41357 | . 81C6 E05E6600 ADD ESI,665EE0
00B4135D | . 5F        POP EDI
00B4135E | . 8BC6     MOV EAX,ESI
00B41360 | . 5E       POP ESI
00B41361 | \. C3     RETN

```



Sau đây là đoạn mã C++ mô phỏng mã assemble ở trên

*1-duyệt string Firstname, tìm vị trí đầu tiên xuất hiện một trong các kí tự đặc biệt*  
*2-với mỗi kí tự đặc biệt, có const int a, const int b*  
*3-kết quả: Firstname.length \* a + b*

---

```
string Firstname;
char OFFSET[] = { 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U' };

int i;
for(i = 0; i < 10; i++)
{
    if(Firstname.find_first_of(OFFSET[i]) != string::npos)
        break;
}
int a, b;
switch(i)
{
    case 0:
    case 1:
        a = 74545, b = 6708960;
        break;
    case 2:
    case 3:
        a = 18303, b = 237952;
        break;
    case 4:
    case 5:
        a = 95683, b = 2966204;
        break;
    case 6:
    case 7:
        a = 95683, b = 2966204;
        break;
    case 8:
    case 9:
        a = 23454, b = 2087495;
        break;
}
return a*Firstname.size() + b;
```

## Tóm tắt thuật toán

1-Kiểm tra Firstname: thỏa (1) tối thiểu 4 kí tự, (2) chứa 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'.

2-Kiểm tra Serial: thỏa (1) khác 0, (2) bằng key do chương trình phát sinh (\*đã nêu trên).

## Kết luận

Firstname: tối thiểu 4 kí tự; chứa 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'.

Serial: 2\_2.exe (keygen)

# Tổng kết

---

## Mức độ hoàn thành

1.1	100%
2.1	100%
2.2	100%

## Kiến thức thu được

Học cách sử dụng PEiD, OllyDbg, cách đọc mã hợp ngữ.

Mỗi crackme sử dụng một cách phát sinh key riêng, làm nhiều thì kinh nghiệm càng cao. Trong đó có những kĩ thuật thú vị như: tạo file rồi chuyển quyền cho hệ điều hành thực thi file đó (như 2.1), hay thuật toán phát sinh key dựa trên input của người dùng (như 1.1, 2.2). Ngoài ra, bài mà chúng tôi mất thời gian nhất, nhưng cũng gây kích thích nhất là 2.2 cũ. Thuật toán phát sinh key của bài khá phức tạp: key gồm 12 kí tự, mã hóa lần 1: mã hóa 4 kí tự đầu thành 16 kí số hexa, mã hóa lần 2: dùng 16 kí số hexa cùng với bảng băm của thuật toán mã hóa CRC32...

## Tham khảo

---

1. *P.E Onimusha*, loạt bài **Basic Cracking Tutorial**
2. *\_kienmanowar\_*, loạt bài **Ollydbg\_tut**
3. Mirror của crackmes.de: <http://crackmes.cf/users/>
4. Các link trong Link tham khảo thêm trên moodle:
  - 4.1. Introduce x86 32 bit: <http://www.cs.virginia.edu/~evans/cs216/guides/x86.html>
  - 4.2. Introduce x86 64 bit  
[https://wiki.cdott.senecacollege.ca/wiki/X86\\_64\\_Register\\_and\\_Instruction\\_Quick\\_Start](https://wiki.cdott.senecacollege.ca/wiki/X86_64_Register_and_Instruction_Quick_Start)
  - 4.3. Understanding C by learning assembly:  
<https://www.recurse.com/blog/7-understanding-c-by-learning-assembly>  
<https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/> (32 bits)  
<https://eli.thegreenplace.net/2011/09/06/stack-frame-layout-on-x86-64> (64 bits)
  - 4.4. Intel and AT&T Syntax: <http://www.imada.sdu.dk/Courses/DM18/Litteratur/IntelATT.htm>
  - 4.5. JUMP quick reference: <http://unixwiz.net/techtips/x86-jumps.html>
5. <https://stackoverflow.com/>