



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: Lê Quốc Hòa, Chung Thùy Linh, Nguyễn Thanh Quân

Nhóm thực hiện: 1712352-1712357-1712405

---

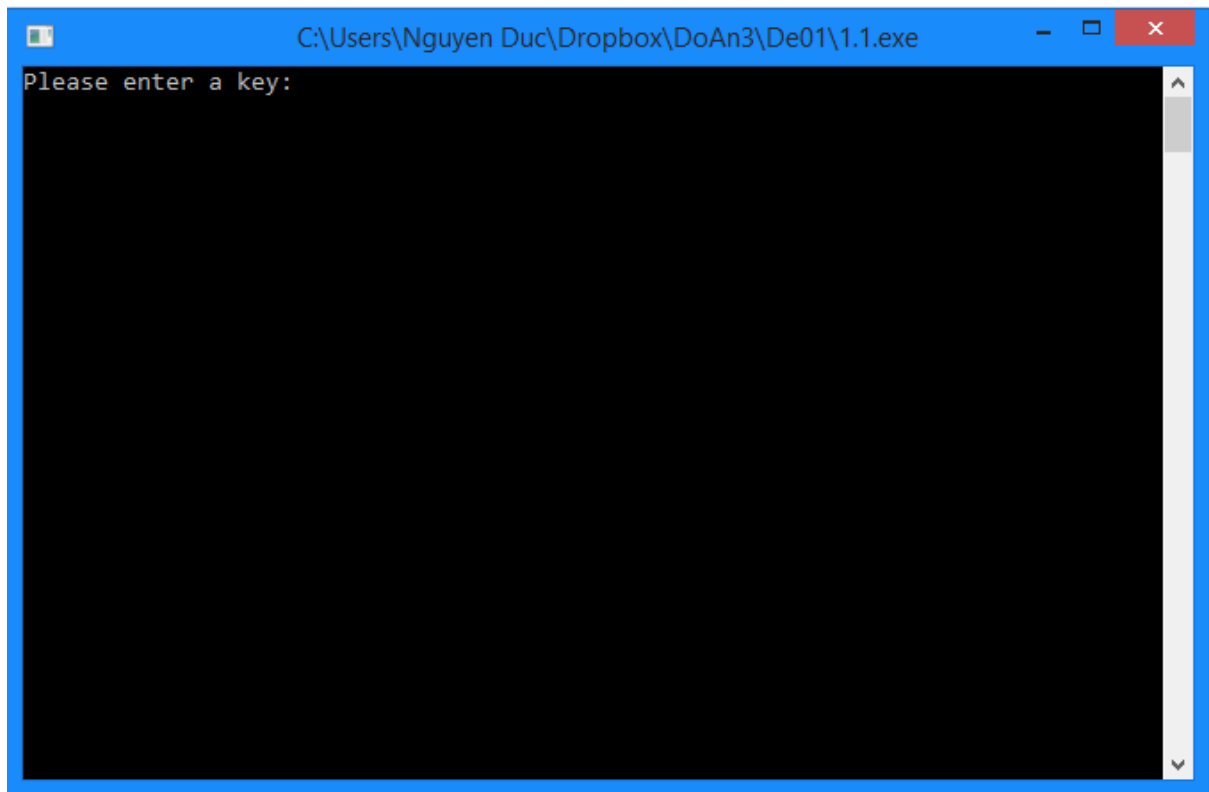
Tháng 6, 2019

**Danh sách thành viên:**

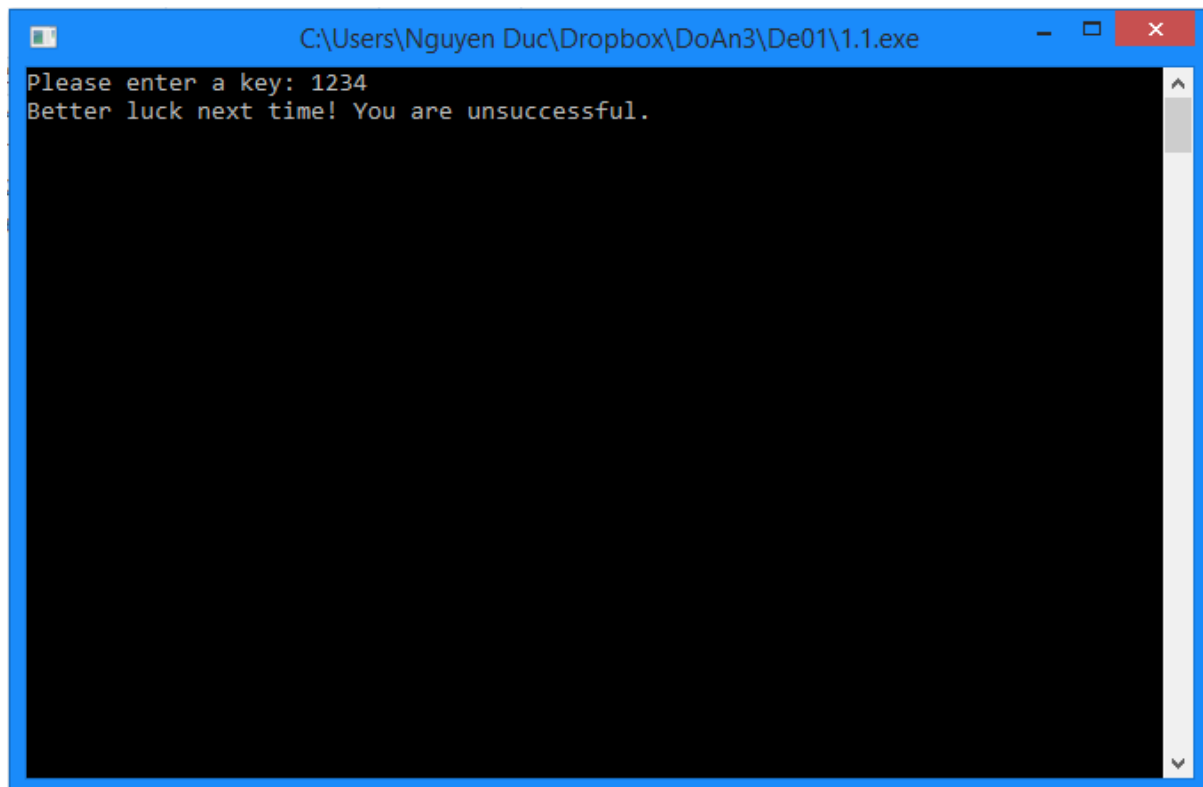
Mã số sinh viên	Tên	Phân công
1712352	Chu Nguyên Đức	1.1, 1.2
1712357	Nguyễn Huỳnh Đức	1.3
1712405	Nguyễn Trường Giang	1.4

# 1.1

Đầu tiên ta chạy thử chương trình



Nhập test case 1234, ta thấy chương trình xuất ra thông báo "Better luck next time! You are unsuccessful."



Chạy OllyDbg, mở 1.1.exe. Chuột phải chọn Search for > All referenced text strings, ta thấy được chuỗi khi nhập key sai, cùng với chuỗi "Congratulation! You are successful." là khi nhập key đúng

Text strings referenced in 1_1:.text		
Address	Disassembly	Text string
0040121A	DB BC	(Initial CPU selection)
00401302	MOV DWORD PTR [ESP],1_1.00403000	ASCII "Please enter a key: "
00401316	MOV DWORD PTR [ESP],1_1.00403015	ASCII "%d"
00401330	MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful.0"
0040133E	MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.0"
00401517	MOV ECX,1_1.00403094	ASCII "w32_sharedptr->size == sizeof(W32_EH_SHARED)"
00401529	MOV DWORD PTR [ESP],1_1.004030C1	ASCII "%s:%s: failed assertion '%s'0"
00401530	MOV EAX,1_1.004030E0	ASCII ".../gcc/gcc/config/i386/w32-shared-ptr.c"
0040153E	MOV EAX,1_1.0040310C	ASCII "GetAtomNameA (atom, s, sizeof(s)) != 0"

Đi đến đoạn mã chứa chuỗi đó

```

004012E6 8BC0 0F ADD EAX,0F
004012E9 8BC0 0F ADD EAX,0F
004012EC C1E8 04 SHR EAX,4
004012EF C1E0 04 SHL EAX,4
004012F2 8945 FC MOV DWORD PTR [EBP-4],EAX
004012F5 8B45 FC MOV EAX,DWORD PTR [EBP-4]
004012F8 E8 A3040000 CALL 1_1.004017A0
004012FD E8 3E010000 CALL 1_1.00401440
00401302 C70424 00304 MOV DWORD PTR [ESP],1_1.00403000
00401309 E8 A2050000 CALL <JMP.6msvcrt.printf>
0040130E C74424 04 10 MOV DWORD PTR [ESP+4],1_1.00404010
00401316 C70424 15304 MOV DWORD PTR [ESP],1_1.00403015
0040131D E8 7E050000 CALL <JMP.6msvcrt.scanf>
00401322 E8 69FFFFFF CALL 1_1.00401290
00401327 833D 10404000 CMP DWORD PTR [404010],1
0040132E 75 0E JNZ SHORT 1_1.0040133E
00401330 C70424 18304 MOV DWORD PTR [ESP],1_1.00403018
00401337 E8 74050000 CALL <JMP.6msvcrt.printf>
0040133C EB 0C JMP SHORT 1_1.0040134A
0040133E C70424 40304 MOV DWORD PTR [ESP],1_1.00403040
00401345 E8 66050000 CALL <JMP.6msvcrt.printf>
0040134A E8 D1040000 CALL <JMP.6msvcrt._getch>
0040134F B8 00000000 MOV EAX,0
00401354 C9 LEAVE
00401355 C3 RET
00401356 90 NOP
00401357 90 NOP
00401358 90 NOP
00401359 90 NOP
0040135A 90 NOP
0040135B 90 NOP
0040135C 90 NOP

```

ASCII "Please enter a key: "

printf

scanf

ASCII "%d"

scanf

ASCII "Congratulation! You are successful."

printf

ASCII "Better luck next time! You are unsuccessful."

printf

\_getch

00403018=1\_1.00403018 (ASCII "Congratulation! You are successful.")

### Phân tích mã:

00401316 MOV DWORD PTR [ESP],1\_1.00403015

```
0040131D CALL <JMP.&msvcrt.scanf>
```

**Ý nghĩa:** Gọi hàm scanf yêu cầu người dùng nhập key.

00401322 CALL 1\_1.00401290

**Ý nghĩa:** Đi tới dòng lệnh có địa chỉ 00401290


Ta đi tới dòng lệnh tương ứng

*G.P.U* - main thread, module 1_1		
00401290	rs 55	PUSH EBP
00401291	- 89E5	MOV EBP,ESP
00401293	- 83EC 04	SUB ESP,4
00401296	- C745 FC C800	MOV DWORD PTR [EBP-4],0C8
0040129D	- 8B55 FC	MOV EDX,DWORD PTR [EBP-4]
004012A0	- 89D0	MOV EAX,EDX
004012A2	- C1E0 02	SHL EAX,2
004012A5	- 01D0	ADD EAX,EDX
004012A7	- 8945 FC	MOV DWORD PTR [EBP-4],EAX
004012AA	- 8D45 FC	LEA EAX,DWORD PTR [EBP-4]
004012AD	- 8330 64	XOR DWORD PTR [EAX],64
004012B0	- 8D45 FC	LEA EAX,DWORD PTR [EBP-4]
004012B3	- F710	NOT DWORD PTR [EAX]
004012B5	- A1 10404000	MOV EAX,DWORD PTR [404010]
004012BA	- 3B45 FC	CMP EAX,DWORD PTR [EBP-4]
004012BD	~ 75 0C	JNZ SHORT 1_1.004012CB
004012BF	~ C705 10404000	MOV DWORD PTR [404010],1
004012C9	~ EB 0A	JMP SHORT 1_1.004012D5
004012CB	> C705 10404000	MOV DWORD PTR [404010],0
004012D5	> C9	LEAVE
004012D6	- C3	RET

### Phân tích mã:

00401296 MOV DWORD PTR [EBP-4], 0C8

**Ý nghĩa:** Lưu giá trị 0C8<sub>h</sub> (200<sub>d</sub>) vào stack địa chỉ EBP - 4

**Kết quả:** 

0040129D MOV EDX, DWORD PTR [EBP-4]

004012A0 MOV EAX, EDX

**Ý nghĩa:** Lưu giá trị của stack có địa chỉ EBP - 4 vào thanh ghi EDX rồi EAX = EDX

**Kết quả:** 

004012A2 SHL EAX,2

004012A5 ADD EAX,EDX

**Ý nghĩa:** Dịch trái EAX 2 lần rồi EAX = EAX + EDX

**Kết quả:** 

004012A7 MOV DWORD PTR [EBP-4], EAX

**Ý nghĩa:** Lưu EAX vào stack địa chỉ EBP - 4

**Kết quả:** 

004012AA LEA EAX, DWORD PTR [EBP-4]

**Ý nghĩa:** Gán EAX vào địa chỉ EBP - 4 (EBP = 0023FF08)

**Kết quả:** 

004012AD XOR DWORD PTR [EAX], 64

**Ý nghĩa:** Thực hiện phép XOR stack địa chỉ EAX (= EBP - 4) với giá trị 64<sub>h</sub> (= 100<sub>d</sub>)

**Kết quả:**  = (908<sub>d</sub>)

004012B3 NOT DWORD PTR [EAX]

**Ý nghĩa:** Thực hiện phép NOT với stack địa chỉ EAX (= EBP - 4)

**Kết quả:**  = (-909<sub>d</sub>)

004012B5 MOV EAX, DWORD PTR [404010]

**Ý nghĩa:** Gán EAX bằng giá trị vùng nhớ có địa chỉ 404010

**Kết quả:** `!EAX 000004D2` ( $= 1234_d$ )  $\Rightarrow$  EAX đang chứa giá trị mà ta nhập vào

004012BA CMP EAX, DWORD PTR [EBP-4]

004012BD JNZ SHORT 1\_1.004012CB

004012BF MOV DWORD PTR [404010], 1

004012C9 JMP SHORT 1\_1.004012D5

004012CB MOV DWORD PTR [404010], 0

004012D5 LEAVE

004012D6 RET

**Ý nghĩa:** So sánh EAX và giá trị stack địa chỉ  $EBP - 4$  ( $= -909_d$ ). Nếu bằng thì gán giá trị ở vùng nhớ 404010 bằng 1, nếu không bằng thì gán bằng 0. Sau đó quay về dòng lệnh 00401327

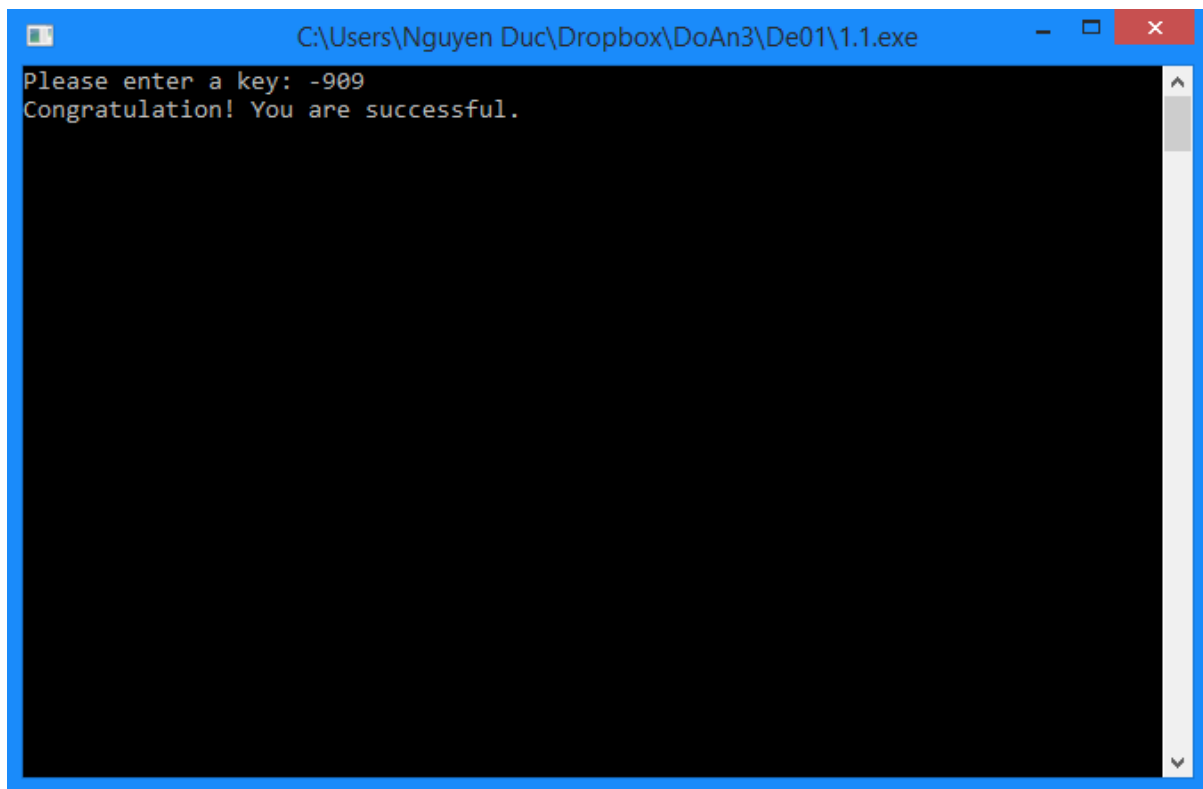
**Kết quả:** `00404010 00 00 00 00`

Quay về dòng lệnh 00401327

```
00401327 | . 833D 10404004 CMP DWORD PTR [404010],1
0040132E | . 75 0E JNZ SHORT 1_1.0040133E
00401330 | . C70424 183044 MOV DWORD PTR [ESP],1_1.00403018 ASCII "Congratulation! You are success
00401337 | . E8 74050000 CALL <JMP.$msvcrt.printf> printf
0040133C | . EB 0C JMP SHORT 1_1.0040134A
0040133E | > C70424 403044 MOV DWORD PTR [ESP],1_1.00403040 ASCII "Better luck next time! You are
00401345 | . E8 66050000 CALL <JMP.$msvcrt.printf> printf
0040134A | > E8 D1040000 CALL <JMP.$msvcrt._getch> _getch
```

**Ý nghĩa:** Nếu giá trị ở vùng nhớ 404010 bằng 1 thì in ra thông báo successful, bằng 0 thì in ra thông báo unsuccessful. Như vậy ta suy ra được key là -909.

Thử lại và thành công.



## 1.2

Đầu tiên ta chạy thử chương trình

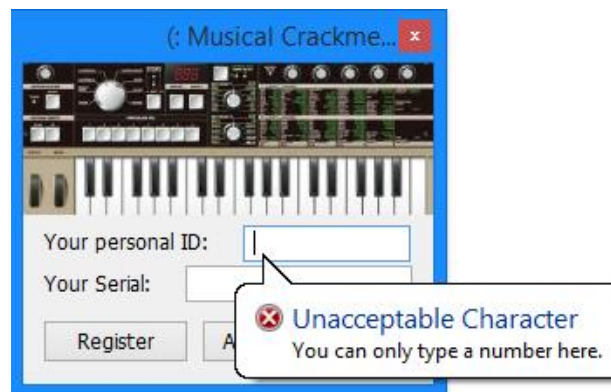


Nhập ID 1712, serial 352 ta thấy không có gì xảy ra





Nhập thử ký tự khác chữ số thì thấy thông báo



Nhập thử số lớn thì thấy ID cho phép nhập tối đa 4 chữ số, còn serial cho nhập tối đa 16 chữ số



Vậy kết luận ID có phạm vi từ 0 đến 9999, còn serial có phạm vi từ 0 đến 9999999999999999, không nhập cũng được.

Chạy OllyDbg, chọn All referenced text strings thì chỉ thấy có chuỗi về About

Text strings referenced in 1_2:text		
Address	Disassembly	Text string
00401011	PUSH 0	(Initial CPU selection)
004010EE	PUSH 1_2.00405063	ASCII "About"
004010F3	PUSH 1_2.00405000	ASCII "Musical Crackme Rules: - No debugging - No

Chuột phải Search for > All intermodular calls thì thấy có nhiều hàm

Found intermodular calls		
Address	Disassembly	Destination
00401002	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
0040100C	CALL <JMP.&comctl32.InitCommonControls>	comctl32.InitCommonControls
00401022	CALL <JMP.&user32.ShowDialogBoxParamA>	user32.ShowDialogBoxParamA
00401029	CALL <JMP.&kernel32.ExitProcess>	(Initial CPU selection)
0040103E	CALL <JMP.&kernel32.FindResourceA>	kernel32.FindResourceA
0040104B	CALL <JMP.&kernel32.SizeofResource>	kernel32.SizeofResource
0040105D	CALL <JMP.&kernel32.LoadResource>	kernel32.LoadResource
00401063	CALL <JMP.&kernel32.LockResource>	kernel32.LockResource
00401075	CALL <JMP.&kernel32.GlobalAlloc>	kernel32.GlobalAlloc
004010CD	CALL <JMP.&user32.SendDlgItemMessageA>	user32.SendDlgItemMessageA
004010FB	CALL <JMP.&user32.MessageBoxA>	user32.MessageBoxA
0040112D	CALL <JMP.&user32.GetDlgItemInt>	user32.GetDlgItemInt
00401142	CALL <JMP.&user32.GetDlgItemInt>	user32.GetDlgItemInt
00401166	CALL <JMP.&user32.GetDlgItem>	user32.GetDlgItem
0040116E	CALL <JMP.&user32.EnableWindow>	user32.EnableWindow
0040117B	CALL <JMP.&user32.GetDlgItem>	user32.GetDlgItem
00401183	CALL <JMP.&user32.EnableWindow>	user32.EnableWindow
00401190	CALL <JMP.&user32.GetDlgItem>	user32.GetDlgItem
00401198	CALL <JMP.&user32.EnableWindow>	user32.EnableWindow
004011BD	CALL <JMP.&user32.EndDialog>	user32.EndDialog
00401241	CALL <JMP.&winmm.waveOutRestart>	winmm.waveOutRestart
00401246	CALL <JMP.&kernel32.ResumeThread>	kernel32.ResumeThread
0040124D	CALL <JMP.&kernel32.SuspendThread>	kernel32.SuspendThread
00401258	CALL <JMP.&winmm.waveOutPause>	winmm.waveOutPause
004012C5	CALL <JMP.&winmm.waveOutReset>	winmm.waveOutReset
004012D0	CALL <JMP.&winmm.waveOutClose>	winmm.waveOutClose
004012DB	CALL <JMP.&kernel32.GlobalFree>	kernel32.GlobalFree
0040141C	CALL <JMP.&winmm.waveOutOpen>	winmm.waveOutOpen
00401683	CALL <JMP.&winmm.waveOutGetPosition>	winmm.waveOutGetPosition
004016D8	CALL <JMP.&kernel32.Sleep>	kernel32.Sleep
004016F3	CALL <JMP.&kernel32.ExitThread>	ntdll.RtlExitUserThread
0040173B	CALL <JMP.&kernel32.GlobalAlloc>	kernel32.GlobalAlloc

Để ý thấy có 2 hàm GetDlgItemInt, đó là 2 hàm cho phép nhập ID và serial. Đi đến dòng lệnh gọi hàm

00401105	> 3D EB030000	CMP EAX,3EB	
0040110A	~ 0F85 98000000	JNZ 1_2.004011A8	
00401110	- C605 F8504000	MOV BYTE PTR [4050F8],0	
00401117	- C605 78514000	MOV BYTE PTR [405178],0	
0040111E	- 51	PUSH ECX	
0040111F	- 53	PUSH EBX	
00401120	- 50	PUSH EAX	
00401121	- 6A 01	PUSH 1	IsSigned = TRUE
00401123	- 6A 00	PUSH 0	pSuccess = NULL
00401125	- 68 EC030000	PUSH 3EC	ControlID = 3EC (1004.)
0040112A	- FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
0040112D	- E8 D8000000	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401132	- 8BD8	MOV EBX,EAX	
00401134	- 33C0	XOR EAX,EAX	
00401136	- 6A 01	PUSH 1	IsSigned = TRUE
00401138	- 6A 00	PUSH 0	pSuccess = NULL
0040113A	- 68 ED030000	PUSH 3ED	ControlID = 3ED (1005.)
0040113F	- FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
00401142	- E8 C3000000	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401147	- 8BC8	MOV ECX,EAX	
00401149	- E8 4DFFFFFF	CALL 1_2.0040109B	
0040114E	- 3BC3	CMP EAX,EBX	
00401150	~ 74 0C	JE SHORT 1_2.0040115E	
00401152	- BB 00000000	MOV EBX,0	
00401157	- BA 00000000	MOV EDX,0	
0040115C	~ EB 53	JMP SHORT 1_2.004011B1	

## Phân tích mã:

0040112D CALL <JMP.&user32.GetDlgItemInt>

**Ý nghĩa:** Gọi hàm nhập ID

**Kết quả:** EAX 000006B0 EAX chứa ID được nhập ( $1712_d = 6B0_h$ )

00401132 MOV EBX,EAX

00401134 XOR EAX,EAX

**Ý nghĩa:** EBX = EAX, EAX = 0. Vậy EBX chứa ID được nhập

00401142 CALL <JMP.&user32.GetDlgItemInt>

**Ý nghĩa:** Gọi hàm nhập Serial

**Kết quả:** EAX 00000160 EAX chứa Serial được nhập ( $352_d = 160_h$ )

00401149 CALL 1\_2.0040109B

**Ý nghĩa:** Chạy tới dòng lệnh 0040109B. Ở đây ta thấy đoạn mã phát sinh serial

Đoạn mã phát sinh serial. Ý nghĩa của từng dòng đã được chú thích trong hình

0040109B	83C3 4C	ADD EBX, 4C	EBX = EBX + 76
0040109E	83C2 03	ADD EDX, 3	EDX = EDX + 3
004010A1	43	INC EBX	EBX++
004010A2	81C3 8B030000	ADD EBX, 38B	EBX = EBX + 907
004010A8	03DB	ADD EBX, EBX	EBX = EBX + EBX
004010AA	0FAFDA	IMUL EBX, EDX	EBX = EBX * EDX
004010AD	4B	DEC EBX	EBX--
004010AE	C3	RET	

Quay về dòng lệnh kế tiếp, ta thấy dòng lệnh so sánh Serial được nhập và Serial phát sinh

0040114E	3BC3	CMP EAX, EBX	
00401150	74 0C	JE SHORT 1_2.0040115E	
00401152	BB 00000000	MOV EBX, 0	
00401157	BA 00000000	MOV EDX, 0	
0040115C	EB 53	JMP SHORT 1_2.004011B1	

## Phân tích đoạn mã phát sinh Serial

Ta đã có được EBX hiện tại đang giữ ID, còn EDX thì bằng 0

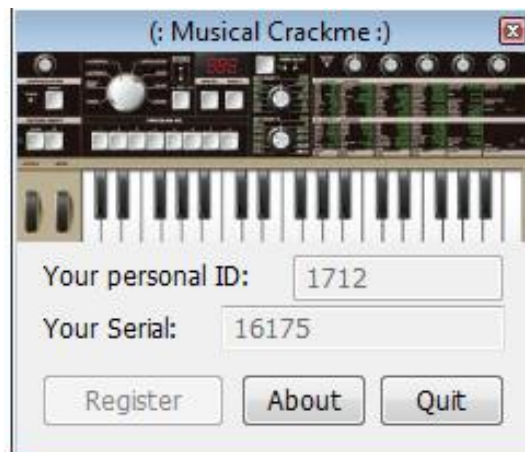
```
ECX 00000160
EDX 00000000
EBX 000006B0
ESP 0013F940
```

Nên ta suy ra được công thức tính serial cần phải nhập:

$$\text{Serial} = 3 * (2 * \text{ID} + 1968) - 1$$

Vậy với ID 1712 thì Serial = 16175

Chạy thử với ID và Serial tính được thì ta thấy chương trình lên nhạc và các khung nhập, Register bị làm mờ, tức là đã thành công.



## 1.3

### Phân tích quá trình debug:

Tìm trong chương trình không thấy đoạn string khi mở chương trình  $\Rightarrow$  Vùng nhớ không thể truy xuất khi chưa chạy chương trình. Tiến hành debug và tìm trong memory map thì thấy xuất hiện.

00419498	. 8B0D 04E84100	MOV ECX,DWORD PTR [41E8DC]
0041949E	. 8B15 04E84100	MOV EDX,DWORD PTR [41E8D4]
004194A4	. E8 9FB0FEFF	CALL 1_3.00404548
004194A9	. 8B55 BC	MOV EDX,DWORD PTR [EBP-44]
004194AC	. A1 70E84100	MOV EAX,DWORD PTR [41E870]

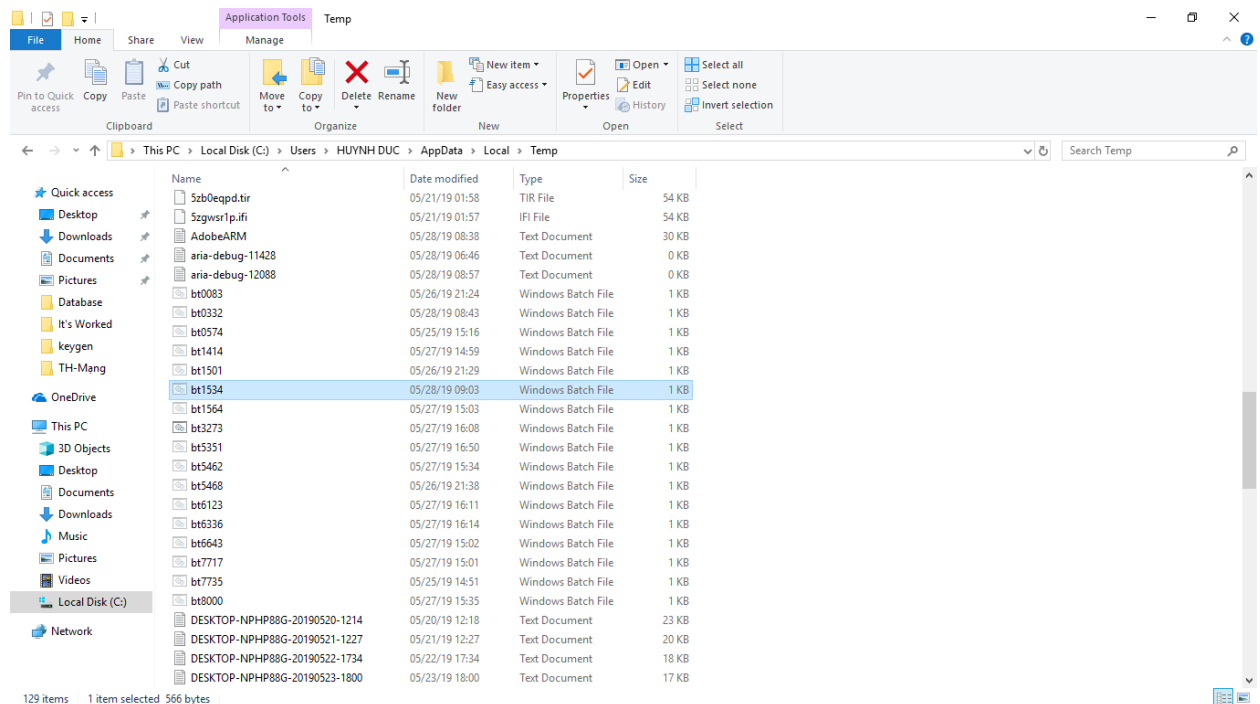
Tiếp tục debug các dòng trên và thấy rằng một đường dẫn xuất hiện.

**Ý nghĩa:** Gán ECX bằng vùng nhớ dump 41E8DC thì thấy ECX = 022E1464 ASCII: “bt1534.bat”

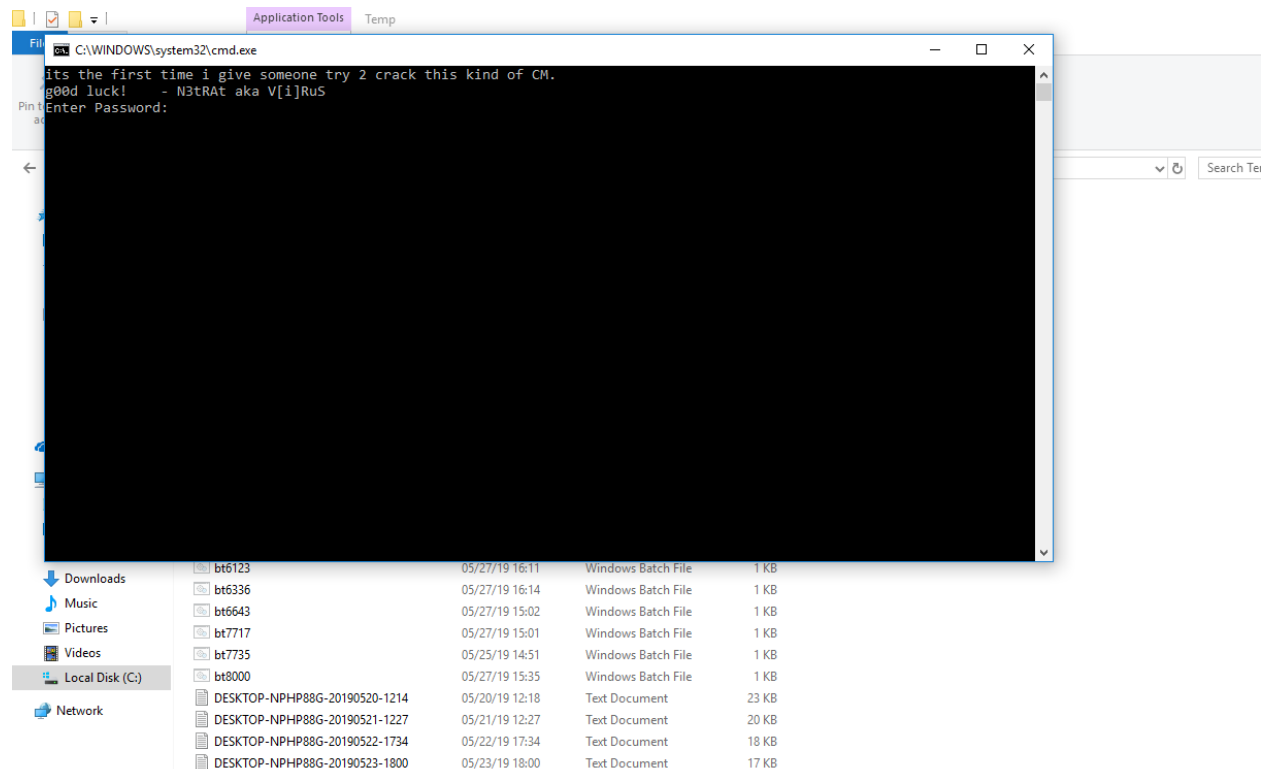
Thanh ghi EDX cũng có ý nghĩa tương tự EDX = “C:\Users\HUYNH DUC\AppData\Local\Temp”

Trả về chuỗi “C:\Users\HUYNH DUC\AppData\Local\Temp\bt1534.bat”. Xong rồi được lưu vào thanh ghi khác và có kết quả:

Tiến hành mở đường dẫn, ta thấy quả thật có file bt1534.bat như trong đường dẫn.



Tiến hành mở file lên, ta thấy xuất hiện như chương trình đang chạy và thấy giống dòng chữ như chương trình chính. Tiến hành nhập thì nó xuất thì nó vẫn như chương trình ta đang xét.



Ở đây có thể phỏng đoán là file .bat này hoạt động tương tự như chương trình mình đang chạy. Tiến hành edit file .bat này thì ta có thể dễ dàng thấy được password của chương trình là gì:

```
@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
```

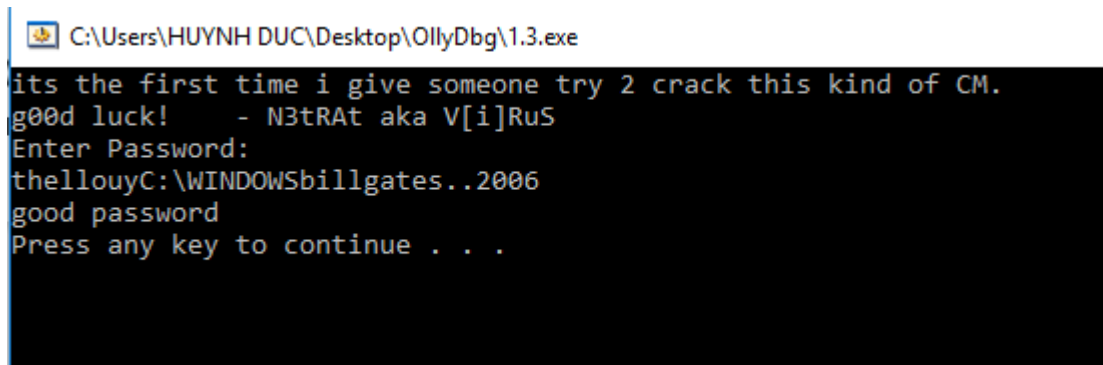
Vậy thuật toán:

1. Tạo đường dẫn.
2. Tạo file bat.
3. Đọc string từ vùng nhớ dump vào file .bat (Vì chỉ thấy chuỗi ký tự trong vùng nhớ dump không xuất hiện trong chương trình).

Vậy việc lấy key chỉ thông qua file .bat là ta có thể lấy được.

⇒ Pass: pass thellouy<windir>billgates..2006 với windir tùy thuộc vào máy.

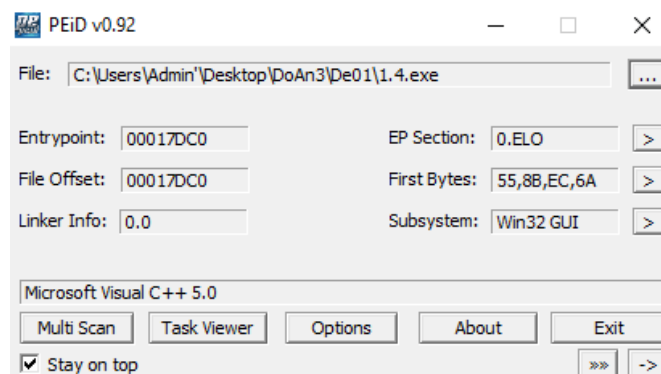
**Kết quả:**



```
C:\Users\HUYNH DUC\Desktop\OllYDbg\1.3.exe
its the first time i give someone try 2 crack this kind of CM.
good luck! - N3tRAt aka V[i]RuS
Enter Password:
thellouyC:\WINDOWSbillgates..2006
good password
Press any key to continue . . .
```

## 1.4

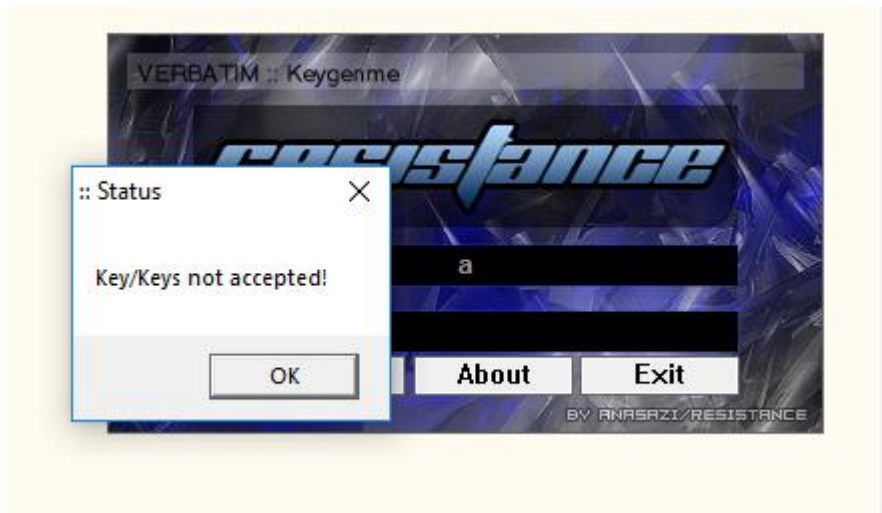
Đầu tiên, scan file 1.4.exe bằng PEiD để xem file có bị pack hay không.



Vậy kết quả là “Microsoft Visual C++ 5.0”, file không bị pack.

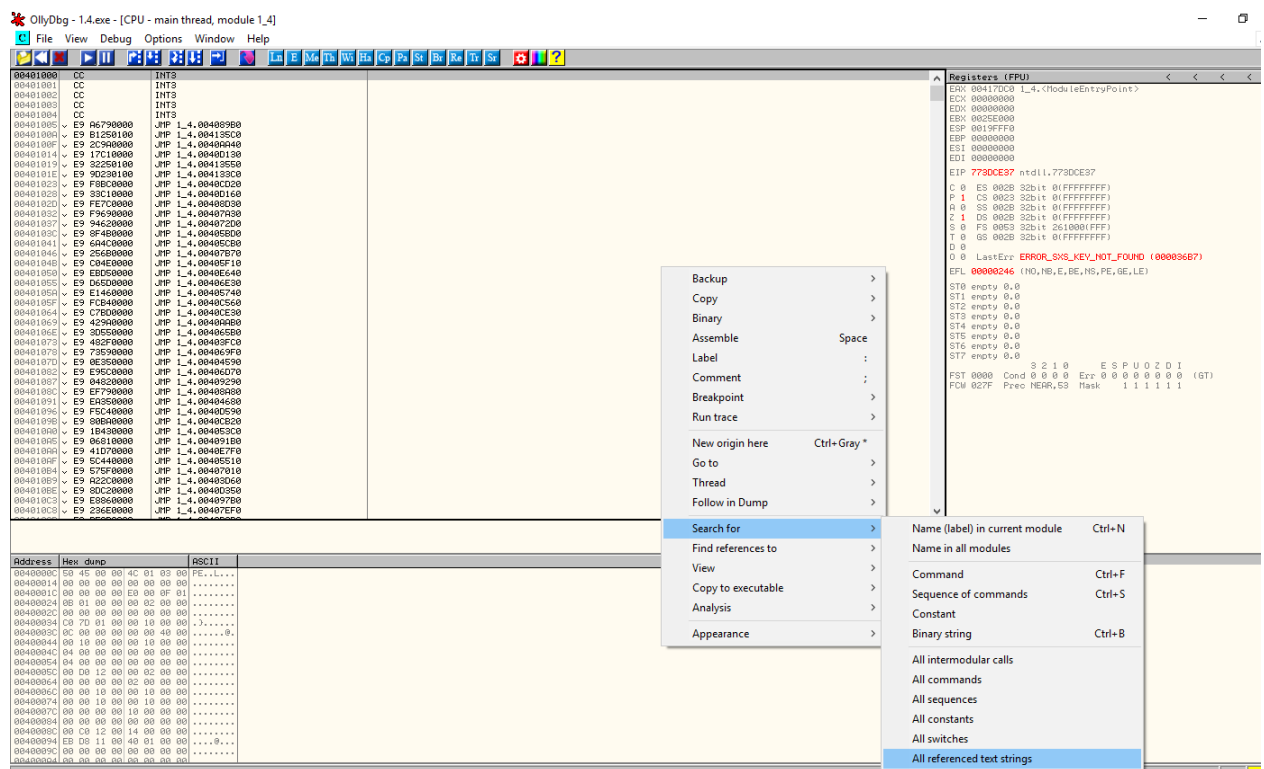
Thực hiện mở file 1.4.exe lên bằng OllyDbg. Rồi chạy thử chương trình (F9) để xuất hiện Nag.





Vậy có một Nag là “Key/Keys not accepted”.

Thực hiện tìm kiếm các chuỗi mà 1.4.exe hiển thị. Chuột phải tại Disassembler Window > Search for > All referenced text strings



Cửa sổ mới xuất hiện với rất nhiều chuỗi. Phát hiện được cả badboy và goodboy.



004032ED	PUSH	1_4.00478048	ASCII "RES-REGGED.txt"
004032FE	PUSH	1_4.00478054	ASCII "RES-VALIDATE.dia"
004032E1	PUSH	1_4.0047805C	ASCII "EDIT"
004032D3	PUSH	1_4.0047805C	ASCII "EDIT"
00403480	PUSH	1_4.0047807C	ASCII "validate"
00403400	PUSH	1_4.00478074	ASCII "BUTTON"
00403440	PUSH	1_4.0047806C	ASCII "boom"
0040344F	PUSH	1_4.00478074	ASCII "BUTTON"
0040340C	PUSH	1_4.00478064	ASCII "Exit"
00403491	PUSH	1_4.00478074	ASCII "BUTTON"
004035E9	PUSH	1_4.0047818C	ASCII "VERBATIM is Keygenne"
00403629	PUSH	1_4.00478164	ASCII "Enter a name and press validate!"
00403736	PUSH	1_4.00478160	ASCII "OK"
004037C9	PUSH	1_4.0047815C	ASCII "OK"
004037CE	PUSH	1_4.00478148	ASCII "Serial Accepted"
004037E8	PUSH	1_4.0047813C	ASCII "": Status"
004037F2	PUSH	1_4.00478128	ASCII "Key/Keys not accepted!"
0040380F	PUSH	1_4.00478110	ASCII "": VERBATIM"
00403814	PUSH	1_4.00478088	ASCII "Coded by Anasazi/RESISTANCE Greetz fly out to Shub, hmw0101, tHE wUTbELE, @s7K, Potassium, Iena51 and Ank89"
0040387F	PUSH	1_4.00478254	ASCII "missing locale facet"
0040393F	PUSH	1_4.00478254	ASCII "missing locale facet"
00403940	PUSH	1_4.00478254	ASCII "missing locale facet"
004039CD	PUSH	1_4.00478278	ASCII "0123456789abdefghicdef"
00403979	PUSH	1_4.00478254	ASCII "missing locale facet"
0040396D	MOV	EDI, 1_4.00478294	ASCII "false"
0040396D	MOV	EDI, 1_4.0047829C	ASCII "true"
0040397F	ASCII	"-c!"	
00403E00	ASCII	"-c!"	
00403E07	ASCII	"-c!"	
00403E19	ASCII	"-c!"	
00412264	MOV	DWORD PTR [EBP-110], 1_4.00478920	ASCII "Extended Modles: "
0041188C	PUSH	1_4.00478420	ASCII ".CrCheckMemory!"
004118C1	PUSH	1_4.00478414	ASCII "doghead."
00411924	PUSH	1_4.00478390	ASCII "Client hook allocation failure at file %hs line %d."
00411947	PUSH	1_4.004783B8	ASCII "Client hook allocation failure."
0041194C	PUSH	1_4.004783B4	ASCII "OK"
0041196B	PUSH	1_4.00478390	ASCII "Invalid allocation size: %u bytes."
004119FA	PUSH	1_4.0047835C	ASCII "Error: memory allocation: bad memory block type."
004119FF	PUSH	1_4.004783B4	ASCII "OK"
00411C3B	PUSH	1_4.00478420	ASCII ".CrCheckMemory!"
00411C77	PUSH	1_4.00478414	ASCII "doghead."
00411D0C	PUSH	1_4.0047839C	ASCII "Client hook re-allocation failure at file %hs line %d."
00411D07	PUSH	1_4.00478378	ASCII "Client hook re-allocation failure."
00411D04	PUSH	1_4.004783B4	ASCII "OK"
00411D06	PUSH	1_4.00478548	ASCII "Allocation too large or negative: %u bytes."
00411D0B	PUSH	1_4.0047835C	ASCII "Error: memory allocation: bad memory block type."
00411D0E	PUSH	1_4.00478354	ASCII "OK"

Nhấp đúp chuột vào goodboy để trở lại chỗ nó xuất hiện ở Dissassemble Windows

00403736	•	50	PUSH EAX	
0040379D	•	E8 8E270100	CALL 1_4.00415F30	
004037A2	•	83C4 08	ADD ESP,8	
004037B5	•	8945 08	MOV DWORD PTR [EBP-58],EAX	
004037B9	•	E9 90CFFFFF	CALL 1_4.00401456	
004037BD	•	833D F4624900	CMPL DWORD PTR [4962F4],1	
004037B4	•	75 30	JNZ SHORT 1_4.004037E6	
004037B6	•	833D F8624900	CMPL DWORD PTR [4962F8],1	
004037BD	•	75 27	JNZ SHORT 1_4.004037E6	
004037BF	•	837D A8 00	CMPL DWORD PTR [EBP-58],0	
004037C2	•	75 21	JNZ SHORT 1_4.004037E6	
004037C5	•	8BF4	MOV ESI,ESP	
004037C7	•	6A 00	PUSH 0	
004037C9	•	68 5CB14700	PUSH 1_4.0047B15C	Style = MB_OK MB_APPLMODAL
004037CE	•	68 48B14700	PUSH 1_4.0047B148	Title = "nFO"
004037D3	•	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	Text = "Serial Accepted"
004037D6	•	51	PUSH ECX	hOwner
004037D7	•	FF15 90144A00	CALL DWORD PTR [4A1490]	MessageBoxA
004037DD	•	3BF4	CMPL ESI,ESP	
004037DF	•	E8 5C1E0100	CALL 1_4.00415640	
004037E4	•	EB 1F	JMP SHORT 1_4.00403805	
004037E6	>	8BF4	MOV ESI,ESP	
004037E9	•	6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
004037EA	•	68 3CB14700	PUSH 1_4.0047B13C	Title = "": Status"
004037EF	•	68 20B14700	PUSH 1_4.0047B120	Text = "Key/Keys not accepted!"
004037F4	•	8B55 08	MOV EDI,DWORD PTR [EBP+8]	hOwner
004037F7	•	52	PUSH EDI	MessageBoxA
004037F8	•	FF15 90144A00	CALL DWORD PTR [4A1490]	
004037FE	•	3BF4	CMPL ESI,ESP	
00403800	•	E8 3B1E0100	CALL 1_4.00415640	
00403805	>	837D 10 67	CMPL DWORD PTR [EBP+10],67	

Thử kéo ngược lên phía trên để tìm thuật toán ra được serial. Và nhận thấy xuất hiện 2 **GetDlgItemTextA**. Một cái dùng để đọc Name, cái còn lại để đọc Serial.

00403698	•	6A 14	PUSH 14	
0040369A	•	8D45 EC	LEA EAX,DWORD PTR [EBP-14]	Count = 14 (20.)
0040369D	•	50	PUSH EAX	Buffer
0040369E	•	6A 6A	PUSH 6A	ControlID = 6A (106.)
004036A0	•	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	
004036A3	•	51	PUSH ECX	hMnd
004036A4	•	FF15 90144A00	CALL DWORD PTR [4A1490]	GetDlgItemTextA
004036AA	•	3BF4	CMPL ESI,ESP	
004036AC	•	E8 9F1F0100	CALL 1_4.00415640	
004036B1	•	8D55 EC	LEA EDI,DWORD PTR [EBP-14]	
004036B4	•	52	PUSH EDI	
004036B5	•	E8 86290100	CALL 1_4.00416040	
004036BA	•	83C4 04	ADD ESP,4	
004036BD	•	8BF4	MOV ESI,ESP	
004036BF	•	6A 14	PUSH 14	Count = 14 (20.)
004036C1	•	8D45 D8	LEA EAX,DWORD PTR [EBP-28]	Buffer
004036C4	•	50	PUSH EAX	ControlID = 6B (107.)
004036C5	•	6A 6B	PUSH 6B	
004036C7	•	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	
004036CA	•	51	PUSH ECX	hMnd
004036CB	•	FF15 90144A00	CALL DWORD PTR [4A1490]	GetDlgItemTextA
004036D1	•	3BF4	CMPL ESI,ESP	
004036D3	•	E8 681F0100	CALL 1_4.00415640	
004036D8	•	8D55 D8	LEA EDI,DWORD PTR [EBP-28]	
004036DB	•	52	PUSH EDI	
004036DC	•	E8 5F290100	CALL 1_4.00416040	
004036E1	•	83C4 04	ADD ESP,4	
004036E4	•	8D45 EC	LEA EAX,DWORD PTR [EBP-14]	
004036E7	•	50	PUSH EAX	
004036E8	•	E8 D3280100	CALL 1_4.00415F30	
004036ED	•	83C4 04	ADD ESP,4	
004036F0	•	8945 08	MOV DWORD PTR [EBP-40],EAX	
004036F3	•	8B4D 08	MOV ECX,DWORD PTR [EBP-48]	
004036F6	•	334D B8	XOR ECX,DWORD PTR [EBP-48]	
004036F9	•	894D B8	MOV DWORD PTR [EBP-48],ECX	

Ngay phía dưới chính là thuật toán để ra serial của crackme.

00403601	. 3B F4	UTP ESI,ESP	
00403603	. E9 81F0100	CALL 1_4.00415640	
00403608	. 8D55 D8	LEA EDX,DWORD PTR [EBP-28]	edx = name
0040360B	. 52	PUSH EDX	
0040360C	. E8 5F290100	CALL 1_4.00416040	reverse name
00403611	. 83C4 04	ADD ESP,4	
00403614	. 8D45 EC	LEA EAX,DWORD PTR [EBP-14]	eax = reverse name
00403617	. 50	PUSH EAX	
00403618	. E8 D3280100	CALL 1_4.00415FC0	
0040361D	. 83C4 04	ADD ESP,4	
0040361F	. 8945 C0	MOV DWORD PTR [EBP-40],EAX	
00403623	. 8B4D B8	MOV ECX,DWORD PTR [EBP-48]	ecx = reverse name
00403626	. 334D B8	XOR ECX,DWORD PTR [EBP-48]	
00403629	. 894D B8	MOV DWORD PTR [EBP-48],ECX	
0040362C	. C745 AC 0000	MOV DWORD PTR [EBP-54],0	
00403703	. EB 09	JMP SHORT 1_4.0040370E	
00403705	> 8B55 AC	MOV EDX,DWORD PTR [EBP-54]	
00403708	. 83C2 01	ADD EDX,1	
0040370B	. 8955 AC	MOV DWORD PTR [EBP-54],EDX	
0040370E	> 8B45 AC	MOV EAX,DWORD PTR [EBP-54]	
00403711	. 3B45 C0	CMPL EAX,DWORD PTR [EBP-40]	
00403714	. 7D 1F	JGE SHORT 1_4.00403735	
00403716	. 8B4D AC	MOV ECX,DWORD PTR [EBP-54]	
00403719	. 0FBEE50D EC	MOVSX EDX,BYTE PTR [EBP+ECX-14]	
0040371E	. 8955 BC	MOV DWORD PTR [EBP-44],EDX	
00403721	. 8B45 BC	MOV EAX,DWORD PTR [EBP-44]	
00403724	. 83EB 20	SUB EAX,20	
00403727	. 8945 BC	MOV DWORD PTR [EBP-44],EAX	
0040372A	. 8B4D B8	MOV ECX,DWORD PTR [EBP-48]	
0040372D	. 2B4D BC	SUB ECX,DWORD PTR [EBP-44]	
00403730	. 894D B8	MOV DWORD PTR [EBP-48],ECX	
00403733	. EB D0	JMP SHORT 1_4.00403705	
00403735	> 8B F4	MOV ESI,ESP	

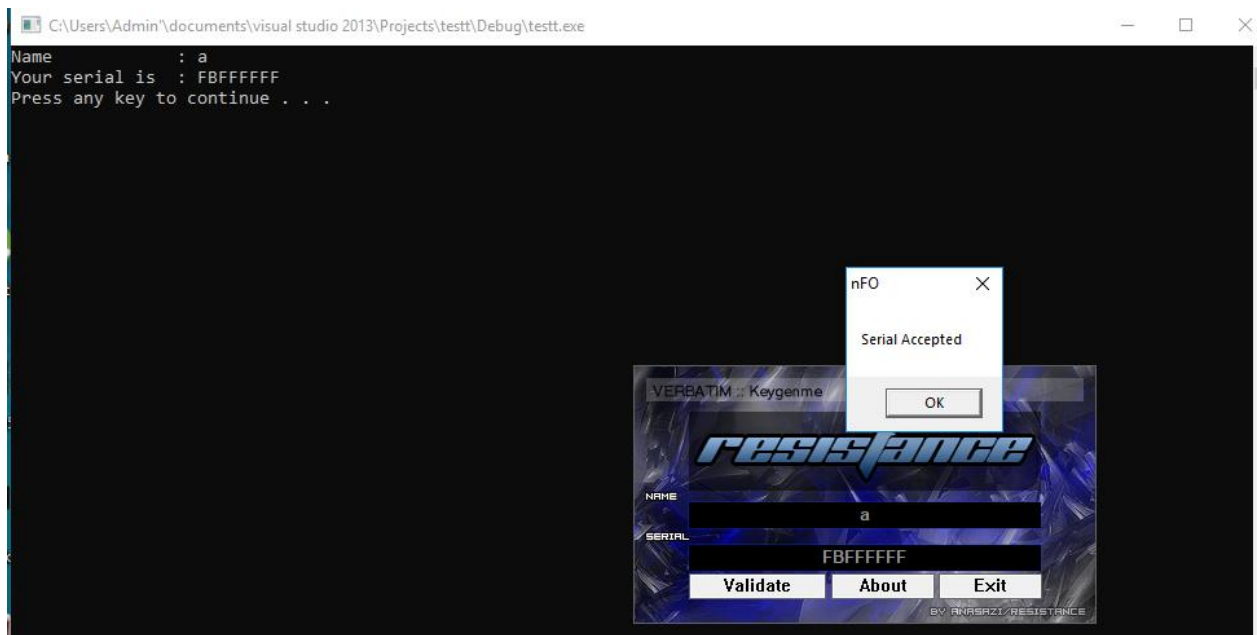
Phần bôi đen chính là công đoạn mà chương trình sau khi có được Name nhập vào sẽ tính toán ra serial hợp lệ.

Tổng quát về cả quá trình để ra được serial như sau:

- Ban đầu, từ chuỗi Name nhập vào, chương trình sẽ đảo ngược chuỗi Name. Sau đó, dùng vòng lặp với biến đếm là thanh ghi EDX, lấy ra từng kí tự của chuỗi name đã bị đảo ngược gán vào EAX rồi trừ cho 20 ở hệ Hexa và lưu lại kết quả. Sau đó lấy ECX trừ cho kết quả đó.

- Sau khi chạy xong vòng lặp, có được kết quả ở Hexa kiểu 8 bits, đảo ngược lại kết quả đó để ra được serial của crackme.

Và serial của Name “a” là : FBFFFFFF



## **Tổng kết**

Mức độ hoàn thành: 80%

Tham khảo: Google