



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP. HCM  
KHOA CÔNG NGHỆ THÔNG TIN

# KIỂM TRÚC MÁY TÍNH & HỢP NGỮ

## BÁO CÁO ĐỒ ÁN 3: CRACK PHẦN MỀM

### ĐỀ 01

**Sinh viên thực hiện:**

**1712914 Phan Nhật Vinh**

**1712919 Lê Văn Vũ**

**1712927 Phạm Thị Tuyết Vy**

**Giáo viên hướng dẫn/ giảng dạy LT:**

**Thầy Lê Quốc Hòa**

**Cô Chung Thùy Linh**

**Thầy Nguyễn Thanh Quân**

**Phân công nhiệm vụ:**

Thành viên		Phân công
MSSV	Họ tên	
1712914	Phan Nhật Vinh	1.2, 1.5
1712919	Lê Văn Vũ	1.3, 1.4, 1.5
1712927	Phạm Thị Tuyết Vy	1.1

*Các thành viên cũng nhau hỗ trợ để hoàn thành đồ án*

**Bảng đánh giá mức độ hoàn thành:**

Câu	Mức độ hoàn thành (%)	Vị trí
1.1	100	Page 1
1.2	100	Page 4
1.3	100	Page 6
1.4	100	Page 12
1.5	100	Page 19

## PHẦN BÀI LÀM

### CÂU 1.1.

Sau khi load chương trình, ta tiến hành chọn vùng cửa sổ chứa code assembly, **chuột phải -> Search for -> All referenced strings**, cửa sổ hiện ra như sau:

0040104E	MOV EBX,-1	(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.█"
0040133E	MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.█"
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'█"
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"

Ta để ý đến dòng **"Congratulation! You are successful"** và **"Better luck next time! You are unsuccessful"**, đây là 2 dòng thông báo được xuất ra màn hình khi người dùng nhập key đúng hoặc sai. Ta thử nhấp đúp chuột vào dòng thông báo đầu tiên:

00401327	. E8 69FFFFFF CALL 1_1.00401290	
0040132E	. 83D0 10404000 CMP DWORD PTR [404010],1	
00401330	. 75 0E JNZ SHORT 1_1.0040133E	
00401330	. C70424 18304 MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful.█"
00401337	. E8 74050000 CALL <JMP.&msvort.printf>	printf
0040133C	. EB 0C JMP SHORT 1_1.0040134A	
0040133E	. C70424 40304 MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.█"
00401345	. E8 66050000 CALL <JMP.&msvort.printf>	printf
0040134A	. E8 D1040000 CALL <JMP.&msvort._getch>	_getch
0040134F	. B8 00000000 MOV EAX,0	
00401354	. C9 LEAVE	
00401355	. C3 RET	

Ta thấy phía trên dòng được bôi đen là dòng nhảy có điều kiện, nhảy đến dòng **0040133E**, là dòng thông báo key nhập vào là sai. Ta xem tiếp dòng phía trên là dòng **CMP**, so sánh giá trị tại địa chỉ **404010** có bằng 1 không, nếu không thì sẽ nhảy đến **0040133E** => Ta xem tiếp phía trên có lời gọi hàm, gọi đến **00401290** => Ta thử đặt BP tại đây và nhảy vào hàm bằng lệnh F7, giá trị nhập thử vào là 123:

00401290	. 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	
004012D7	. 90 NOP	
004012D8	. 55 PUSH EBP	

Đây chính là đoạn code phát sinh key.

Ta giải thích từng dòng lệnh như sau:

- Push giá trị thanh ghi **EBP** vào stack (**0060FF28**).

```
0060FEF8 0060FF28
```

- Chuyển giá trị của thanh ghi **ESP** vào **EBP** (là địa chỉ của phần tử đầu đỉnh stack – vừa push vào ở câu trên).

```
EBP 0060FEF8
```

- Trừ giá trị thanh ghi **ESP** đi 4.

```
ESP 0060FEF4
```

- Chuyển giá trị **0C8** vào địa chỉ **EBP – 4** (đỉnh stack).

```
0060FEF4 000000C8
0060FEF8 0060FF28
```

- Chuyển giá trị trên vào thanh **EDX** (0C8).

```
EDX 000000C8
```

- Chuyển giá trị **EDX** sang **EAX**.

- Shift left **EAX** 2 lần (nhân với  $2^2$ )

```
EAX 00000320
```

- Cộng **EAX** với **EDX**.

```
EAX 000003E8
```

- Chuyển giá trị **EAX** vào **EBP – 4** (đỉnh stack hiện tại)

```
0060FEF4 000003E8
```

- Load địa chỉ của giá trị đỉnh stack (**EBP – 4**) vào **EAX**.

```
EAX 0060FEF4
```

- Xor giá trị tại **EAX** với 64 ( giá trị đỉnh stack là 3E8). Kết quả là 38C.

```
0060FEF4 0000038C
```

- Load địa chỉ của giá trị đỉnh stack (**EBP – 4**) vào **EAX**.

- Thực hiện phép NOT với giá trị này (giá trị đỉnh stack).

```
0060FEF4 FFFFC73
```

- Chuyển giá trị tại địa chỉ 404010 vào **EAX**.

```
EAX 0000007B
```

7B đổi ra thập phân là 123 -> Đây là giá trị người dùng nhập vào.

- So sánh giá trị tại **EAX** với giá trị tại đỉnh stack, nếu bằng thì chuyển giá trị 1 vào địa chỉ 404010, ngược lại là giá trị 0 -> Đây là cờ hiệu xác định xem key người dùng nhập vào có đúng hay không.

- Nhảy ngược về tiếp tục câu lệnh so sánh cờ hiệu như đề cập ở trên, và xuất câu lệnh thông báo cho người dùng.

⇒ Ta xác định được mấu chốt nằm ở giá trị của đỉnh stack hiện tại

```
0060FEF4 FFFFC73
```

=> Đổi ra giá trị thập phân là **4294966387**.

Ta restart lại chương trình và nhập thử:

```
Please enter a key: 4294966387
Congratulation! You are successful.
```

Kết quả chính xác.

**CÂU 1.2.**

Ta cũng thử sau khi load chương trình, ta tiến hành chọn vùng cửa sổ chứa code assembly, **chuột phải -> Search for -> All referenced strings:**

PUSH 0	(Initial CPU selection)
PUSH 1_2.00405063	ASCII "About"
PUSH 1_2.00405000	ASCII "Musical Crackme/ Rules: - No debugging/ - No patching/ - Make a keygen/ wt0vrenr@gmail.com"

Không có chuỗi nào để ta điều tra. Ta tiến hành **Search for -> All intermodular calls:**

```
user32.MessageBoxA
user32.GetDlgItemInt
user32.GetDlgItemInt
```

Ta chú ý đến module **GetDlgItemInt**, thử nhấp vào:

004010EA	JNZ SHORT 1_2.00401105	
004010EC	PUSH 0	Style = MB_OK!MB_APPLMODAL
004010EE	PUSH 1_2.00405063	Title = "About"
004010F3	PUSH 1_2.00405000	Text = "Musical Crackme/ Rules: - No debugging/ - No patching/
004010F8	PUSH DWORD PTR [EBP+8]	hOwner
004010FB	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
00401100	JMP 1_2.004011C2	
00401105	CMF EAX, 3EB	
0040110A	JNZ 1_2.004011A8	
00401110	MOV BYTE PTR [4050F8], 0	
00401117	MOV BYTE PTR [405178], 0	
0040111E	PUSH ECX	
0040111F	PUSH EBX	
00401120	PUSH EAX	
00401121	PUSH 1	IsSigned = TRUE
00401123	PUSH 0	pSuccess = NULL
00401125	PUSH 3EC	ControlID = 3EC (1004.)
0040112A	PUSH DWORD PTR [EBP+8]	hWnd
0040112D	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401132	MOV EBX, EAX	
00401134	XOR EAX, EAX	
00401136	PUSH 1	IsSigned = TRUE
00401138	PUSH 0	pSuccess = NULL
0040113A	PUSH 3ED	ControlID = 3ED (1005.)
0040113F	PUSH DWORD PTR [EBP+8]	hWnd
00401142	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401147	MOV ECX, EAX	
00401149	CALL 1_2.0040109B	

Quan sát kỹ, ta sẽ thấy có 2 lần gọi **GetDlgItemInt**, đặt phía sau phần **MessageBoxA**. Trong phần **MessageBoxA**, ta thấy được phần text giới thiệu chương trình. Từ đó, và dựa vào tên gọi, ta có thể đoán được 2 lần gọi **GetDlgItemInt** là để người dùng nhập vào 2 tham số nào đó, chính xác là ID và phần key.

Để thấy sau lần gọi **GetDlgItemInt** thứ 2, sẽ có dòng lệnh **CALL 1\_2.0040109B**. Rất có thể đây là phần thuật toán phát sinh key, ta thử đặt BP tại dòng này, chạy thử, nhập key và ấn F7:

Ta nhập vào ID **1234**, serial key để trống.

0040109B	ADD EBX, 4C
0040109E	ADD EDI, 3
004010A1	INC EBX
004010A2	ADD EBX, 38B
004010A8	ADD EBX, EBX
004010AA	IMUL EBX, EDI
004010AD	DEC EBX
004010AE	RET

Đây chính là đoạn code phát sinh key.

Ta tiến hành phân tích đoạn code này:

- Cộng **EBX** với **4C**.

EBX 000004D2 Quan sát kỹ, ta thấy **EBX** có giá trị **4D2**, chuyển sang thập phân là **1234**.

Cộng EBX với 4C, ta được:

EBX 0000051E

- Cộng **EDX** với **3**, lúc này **EDX** mang giá trị 0.

EDX 00000003

- Tăng **EBX** lên 1.

EBX 0000051F

- Cộng **EBX** với **38B**.

EBX 000008AA

- Cộng **EBX** với **EBX**.

EBX 00001154

- Nhân **EBX** với **EDX** ( số 3)

EBX 000033FC

- Trừ **EBX** đi 1 -> Đây là kết quả sau cùng, đổi ra hệ thập phân là **13307** là serial key ứng với ID 1234.

EBX 000033FB

- Return.

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

- Ta lại so sánh **EAX** với **EBX**

EAX 00000000

**EAX** có giá trị là 0 -> Đây là giá trị serial key mà người dùng nhập vào, ở đây ta không nhập gì cả nên giá trị của nó là 0.

- Nếu **EAX = EBX** -> Key đúng, ngược lại thì sai.

Thử restart lại chương trình, nhập ID là 1234 , key là 13307.



**Kết quả chính xác.**



**CÂU 1.3.**

- Mở file 1.3.exe, trên màn hình xuất hiện:

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

- Thử nhập 1 password bất kì:

```
its the first time i give someone try 2 crack this kind of CM.
g00d luck!    - N3tRat aka V[i]RuS
Enter Password:
vanvu
bad password
Press any key to continue . . .
```

- Tìm kiếm các text string thì không thấy bất cứ text string nào có nội dung như ở trên màn hình console???

Address	Disassembly	Text string
0041697C	ASCII "Top",0	
00416EF8	DD 1_3.00416F44	ASCII 0C,"ECorruptFile"
00416F18	DD 1_3.00416F44	ASCII 0C,"ECorruptFile"
00416F45	ASCII "ECorruptFile"	
00416F5A	ASCII "TOnAskForKey"	
00416F6A	ASCII "sender"	
00416F71	ASCII "TObject"	
00416F79	ASCII "String"	
00416FA8	DD 1_3.00417016	ASCII 0A,"THKStreams"
00417017	ASCII "THKStreams"	
0041702A	ASCII "THKStreams"	
0041703F	ASCII "HKStreamCol"	
00417067	ASCII "Compressed"	
0041708C	ASCII "Encrypted"	
004170B0	ASCII "Key"	
004170CE	ASCII "OnAskForKey"	
004170F4	ASCII "OnCorrupt"	
00417226	MOV EAX,1_3.00417248	ASCII "File is corrupt."
00417248	ASCII "File is corrupt."	
00417258	ASCII 0	
00417C98	MOV EAX,1_3.00417EB8	ASCII "Compressed file is corrupt"
00417EB8	ASCII "Compressed file "	
00417EC8	ASCII "is corrupt",0	
00418958	ASCII "\",0	
00418CDE	MOV EAX,1_3.00418D90	ASCII "TMP"
00418CFA	MOV EAX,1_3.00418D9C	ASCII "TEMP"
00418D20	MOV EAX,1_3.00418DAC	ASCII "USERPROFILE"
00418D90	ASCII "TMP",0	
00418D9C	ASCII "TEMP",0	
00418DAC	ASCII "USERPROFILE",0	
004190E0	PUSH EBP	(Initial CPU selection)
004191E8	MOV EAX,1_3.00419734	ASCII "MYFILES"
00419228	MOV EDX,1_3.00419744	ASCII "Quick Batch File Compiler"
004192DC	MOV EDX,1_3.00419768	ASCII "BAT"
004192F1	MOV EDX,1_3.00419774	ASCII "FILES"
00419417	PUSH 1_3.00419790	ASCII "bt"
00419470	PUSH 1_3.0041979C	ASCII ".bat"
00419484	MOV EAX,1_3.004197AC	ASCII "@shift 1"
00419517	MOV EAX,1_3.004197D0	ASCII "^&"
00419571	PUSH 1_3.004197E8	ASCII "cmd.exe /c "
00419599	PUSH 1_3.004197FC	ASCII "command.com /c "
00419734	ASCII "MYFILES",0	
00419744	ASCII "Quick Batch File"	
00419754	ASCII " Compiler",0	
00419768	ASCII "BAT",0	
00419774	ASCII "FILES",0	
00419784	ASCII "\",0	
00419790	ASCII "bt",0	
0041979C	ASCII ".bat",0	
004197AC	ASCII "@shift 1",0	
004197C0	ASCII " ",0	
004197D0	ASCII "^&",0	
004197DC	ASCII "&",0	
004197E8	ASCII "cmd.exe /c ",0	
004197FC	ASCII "command.com /c ",0	

- Tìm đến Memory Map thì ta thấy nó có sự khác biệt trước và sau khi ấn F9:

[illegible]

==> Ở phần mapped as xuất hiện dòng chữ mới (phần đóng khung màu xanh phía trên) tại địa chỉ 022D0000. Nhấn đúp chuột vào địa chỉ 021D0000. Ta phát hiện được nhiều manh mối:

+ ) xuất hiện ASCII có nội dung giống phần hiển thị trên màn hình console:

021D10B0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	65	63	
021D10C0	68	6F	20	69	74	73	20	74	68	65	20	66	69	72	73	74	ho its the first
021D10D0	20	74	69	6D	65	20	69	20	67	69	76	65	20	73	6F	6D	time i give som
021D10E0	65	6F	6E	65	20	74	72	79	20	32	20	63	72	61	63	6B	one try 2 crack
021D10F0	20	74	68	69	73	20	6B	69	6E	64	20	6F	66	20	43	4D	this kind of CM
021D1100	2E	00	00	00	56	00	00	00	01	00	00	00	45	00	00	00	....U.....E...
021D1110	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
021D1120	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
021D00C0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
021D0DD0	20	20	20	20	65	63	68	6F	20	67	30	30	64	20	60	75	echo g00d lu
021D0DE0	63	6B	21	20	20	20	20	20	20	4E	33	74	52	41	74	20	ok! - N3tR4t
021D0DF0	61	6B	61	20	56	5B	69	5D	52	75	53	0D	0A	65	63	68	aka U[]R\$\$.ech
021D0E00	6F	20	45	6E	74	65	72	20	50	61	73	73	77	6F	72	64	o Enter Passwor
021D0E10	3A	0D	0A	73	65	74	20	2F	70	20	70	61	73	73	77	6F	:.set /p passwo
021D0E20	72	64	30	0D	0A	69	66	20	22	25	70	61	73	73	77	6F	rd=..if ""passwo
021D0E30	72	64	25	22	30	20	22	25	6F	25	25	6C	6F	25	25	25	rd%"=="%o%lllo%
021D0E40	68	65	25	68	25	25	74	25	25	77	69	6E	64	69	72	60	he%h%t%windir%
021D0E50	25	62	69	6C	6C	67	61	74	65	73	2E	2E	32	30	36	72	%llgates..2006
021D0E60	22	20	67	6F	74	6F	20	67	6F	6F	64	0D	0A	69	66	20	" goto good..if
021D0E70	6E	6F	74	20	22	25	70	61	73	73	77	6F	72	64	25	22	not ""password%"
021D0E80	3D	30	22	25	6F	25	25	6C	6F	25	25	68	65	25	25	25	=="%o%lllo%he%"
021D0E90	68	25	25	74	25	25	77	69	6E	64	69	72	25	62	69	6D	h%t%t%windir%bil
021D0EA0	6C	67	61	74	65	73	2E	2E	32	30	30	36	22	20	67	6F	lgates..2006" go
021D0EB0	74	6F	20	62	61	64	0D	0A	3A	67	6F	6F	64	0D	0A	65	to bad...good..e
021D0EC0	63	68	6F	20	67	6F	6F	64	20	70	61	73	77	6F	72	60	cho good passwor
021D0ED0	64	0D	0A	70	61	75	73	65	0D	0A	65	78	69	74	0D	0F	d..pause..exit..
021D0EE0	3A	62	61	64	0D	0A	65	63	68	6F	20	62	61	64	20	70	:bad..echo bad p
021D0EF0	61	73	73	77	6F	72	64	0D	0A	70	61	75	73	65	0D	0F	assword..pause..
021D0F00	65	78	69	74	0D	0A	0D	0A	3C	02	00	00	1B	00	00	00	exit.....(.....
021D0F10	01	00	00	00	08	00	00	00	45	43	48	4F	20	4F	46	46	.....ECHO OFF
021D0F20	00	00	00	00	26	00	00	00	01	00	00	00	14	00	00	00	.....&.....
021D0F30	65	63	68	6F	20	45	6E	74	65	72	20	50	61	73	73	77	echo Enter Passw
021D0F40	6F	72	64	3A	0D	00	00	00	12	00	00	00	01	00	00	00	ord:.....Rat
021D0F50	03	00	00	00	63	6C	73	00	16	00	00	00	01	00	00	00	...le

+) Xuất hiện đường dẫn: tìm đến đường dẫn này thì thấy tồn tại.

021D0A10	00 07 00 02 00 00 00 00	16 00 00 00 04 08 41 00	.....0
021D0A20	00 00 00 00 00 00 00 00	00 00 00 00 32 00 00 00	.....2...
021D0A30	00 00 00 00 21 00 00 00	43 3A 5C 55 73 65 72 73	.....!...C:\Users
021D0A40	5C 44 45 4C 4C 5C 41 70	70 44 61 74 61 5C 4C 6F	...DELL\AppDataLo
021D0A50	63 61 6C 5C 54 65 6D 70	5C 00 00 00 2A 00 00 00	cal\Temp\...*....
021D0A60	02 00 00 00 18 00 00 00	43 3A 5C 55 73 65 72 73	.....C:\Users
021D0A70	5C 44 45 4C 4C 5C 41 70	70 44 61 74 61 5C 4C 6F	...DELL\AppDataLo
021D0A80	63 61 6C 00 52 00 00 00	D4 6F 41 00 00 00 00 00	cal.R....oR....
021D0A90	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
021D0AA0	00 00 00 00 00 00 00 00	00 00 00 00 01 00 00 00	.....

- Trong quá trình debug, khi chạy đến ddiacj chỉ: 00419498, thì thấy ECX đang lưu ASCII "btxxxx.bat"

```

EAX 0019FF2C
ECX 0214140C ASCII "bt5224.bat"
EDX 00410AC4 1_3.00410AC4
EBX 00410AC4 1_3.00410AC4
ESP 0019FDF8
EBP 0019FF70
ESI 004190E0 1_3.<ModuleEntryPoint>
EDI 004190E0 1_3.<ModuleEntryPoint>

```

==> Phân tích mã hợp ngữ:

```

00419498 . 8B0D DCE84100 MOV ECX,DWORD PTR [41E8DC]
0041949E . 8B15 D4E84100 MOV EDX,DWORD PTR [41E8D4]
004194A4 . E8 9FB0FEFF CALL 1_3.00404548

```

00419498 . 8B0D DCE84100 MOV ECX,DWORD PTR [41E8DC] : gán ECX bằng giá trị tại vùng nhớ dump 41E8DC (nơi đang lưu chuỗi ASCII "btxxxx.bat")

- Khi run đến address 0091949E, EDX đang chứa ASCII "C:\Users\DELL\AppData\Local\Temp\"

```

EAX 0019FF2C
ECX 0214140C ASCII "bt5224.bat"
EDX 02140A38 ASCII "C:\Users\DELL\AppData\Local\Temp\"
EBX 00410AC4 1_3.00410AC4
ESP 0019FDF8
EBP 0019FF70
ESI 004190E0 1_3.<ModuleEntryPoint>
EDI 004190E0 1_3.<ModuleEntryPoint>

```

==>Phân tích mã hợp ngữ:

```

00419498 . 8B0D DCE84100 MOV ECX,DWORD PTR [41E8DC]
0041949E . 8B15 D4E84100 MOV EDX,DWORD PTR [41E8D4]
004194A4 . E8 9FB0FEFF CALL 1_3.00404548

```

0041949E . 8B15 D4E84100 MOV EDX,DWORD PTR [41E8D4]: gán EDX bằng giá trị tại vùng nhớ dump 41E8DC (nơi đang lưu chuỗi "C:\Users\DELL\AppData\Local\Temp\")

- Khi run đến address 004194A9], EDX đang chứa ASCII "C:\Users\DELL\AppData\Local\Temp\bt5224.bat"

```

EAX 02140B68
ECX 00000000
EDX 0214143C ASCII "C:\Users\DELL\AppData\Local\Temp\bt5224.bat"
EBX 00410AC4 1_3.00410AC4
ESP 0019FDF8
EBP 0019FF70
ESI 004190E0 1_3.<ModuleEntryPoint>
EDI 004190E0 1_3.<ModuleEntryPoint>
EIP 004194B1 1_3.004194B1

```

==> Phân tích mã hợp ngữ như trên.

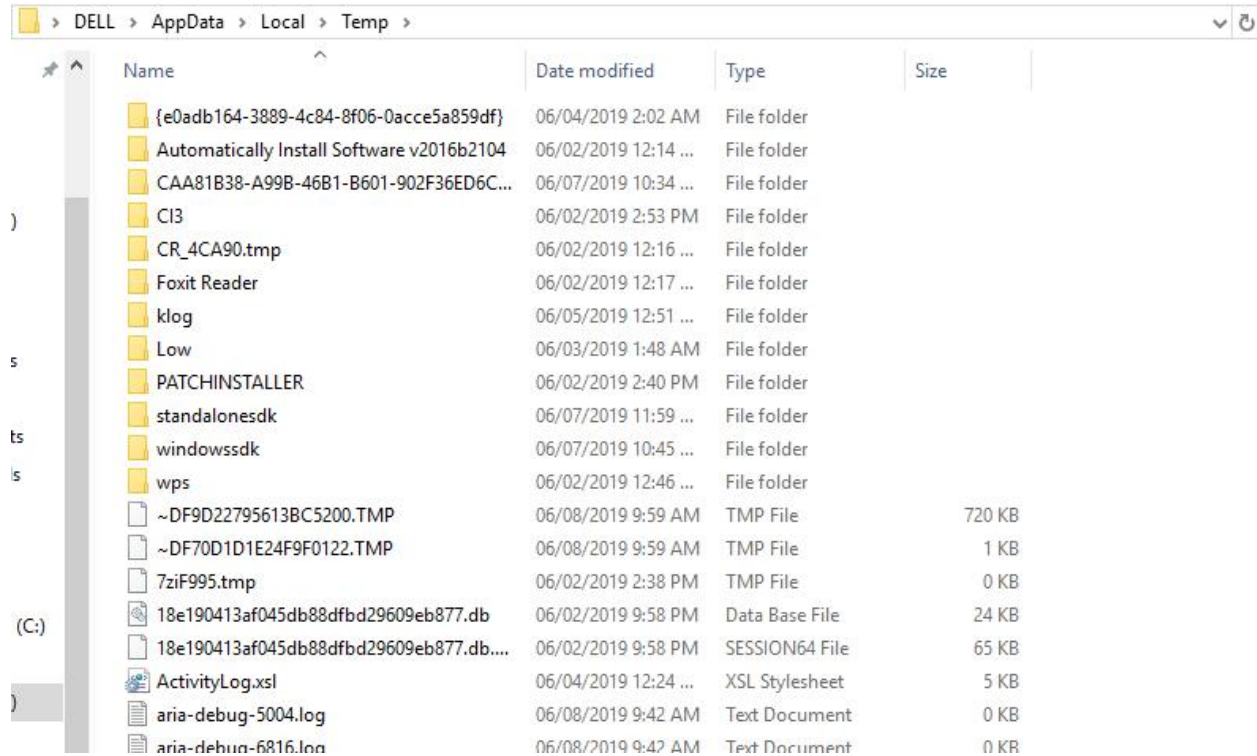
```

004194B1 . 8B08          MOV ECX,DWORD PTR [EAX]
004194B3 . FF51 74       CALL DWORD PTR [ECX+74] Tạo file .bat
004194B6 . 8045 B8       LEA EAX,DWORD PTR [EBP-48]

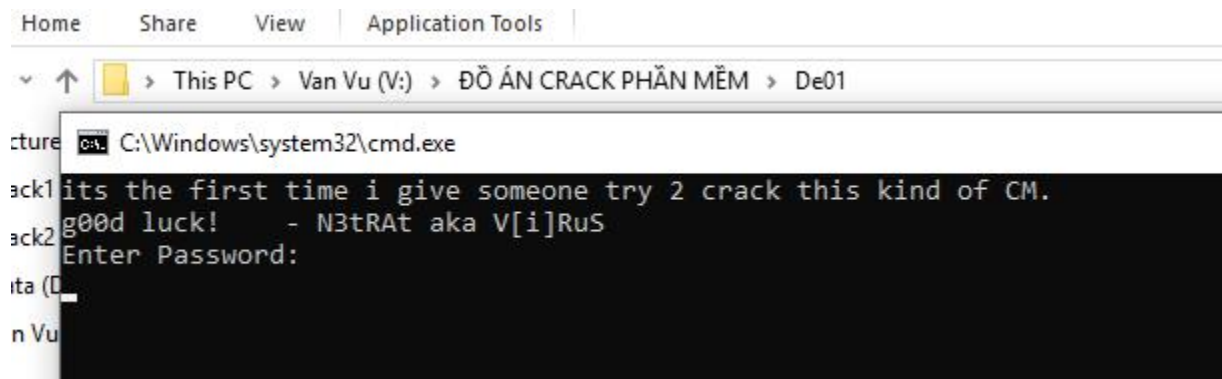
```



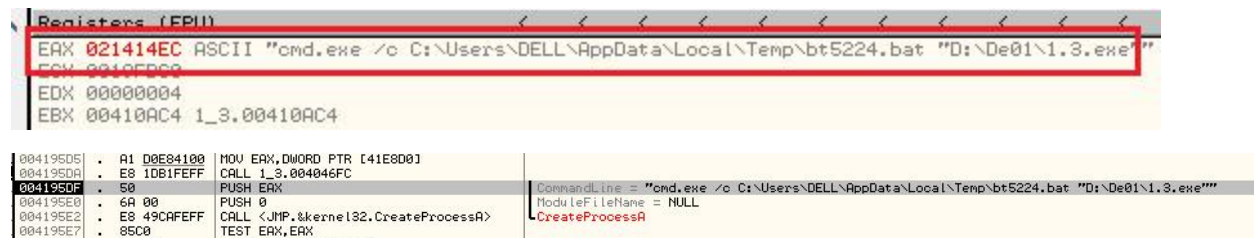
- Tìm đến đường dẫn C:\Users\DELL\AppData\Local\Temp\



==> Sau đó tìm đến C:\Users\DELL\AppData\Local\Temp\bt5224.bat thì xuất hiện cmd.exe có nội dung như trên màn hình console khi mở file 1.3.exe ==> password chắc chắn có liên quan đến file .bat này.



- Tiếp tục debug, khi run đến địa chỉ 004195D5, EDX chứa ASCII như hình dưới:



==> 004195DF . 50 PUSH EAX → Tạo commandline

- Khi run đến 004195F3 thì bị dừng lại, trên màn hình xuất hiện:

its the first time i give someone try 2 crack this kind of CM.

g00d luck! - N3tRat aka V[i]RuS

Enter Password:

=> Ý nghĩa dòng lệnh tại đại chỉ 004195F3: đợi người dùng nhập password.

004195E7	. 85C0	TEST EAX,EAX	
004195E9	74 33	JE SHORT 1_3.0041961E	
004195EB	6A FF	PUSH -1	
004195ED	A1 C0E84100	MOV EAX,DWORD PTR [41E8C0]	Timeout = INFINITE
004195F2	50	PUSH EAX	hObject => 00000240
004195F3	E8 98CBFEFF	CALL <JMP.&kernel32.WaitForSingleObject	WaitForSingleObject
004195F8	68 E8E84100	PUSH 1_3.0041E8E8	pExitCode = 1_3.0041E8E8

\*Ta bắt đầu đi tìm password:

- Mở Command Prompt lên, nhập: cmd.exe /c C:\Users\DELL\AppData\Local\Temp\bt5224.bat "D:\De01\1.3.exe"

```

CA: Command Prompt
Microsoft Windows [Version 10.0.17763.529]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\DELL>cmd.exe /c C:\Users\DELL\AppData\Local\Temp\bt5224.bat "D:\De01\1.3.exe"

```

=> Trên màn hình hiển thị nội dung giống khi ta mở file 1.3.exe lên:

```

CA: Command Prompt - cmd.exe /c C:\Users\DELL\AppData\Local\Temp\bt5224.bat "D:\De01\1.3.exe"
its the first time i give someone try 2 crack this kind of CM.
g00d luck! - N3tRat aka V[i]RuS
Enter Password:
vanvu
bad password
Press any key to continue . . .

```

=> Đến đây, ta khẳng định nội dung chứa trong file btxxxx.bat chắc chắn chứa manh mối để ta tìm password. Ta tìm đến file btxxxx.bat, sau đó nhấp chuột phải chọn edit thì nội dung (**string pass**)<sup>(\*)</sup> hiển thị như sau: (*(\*) là tên em tự đặt*)

```

@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

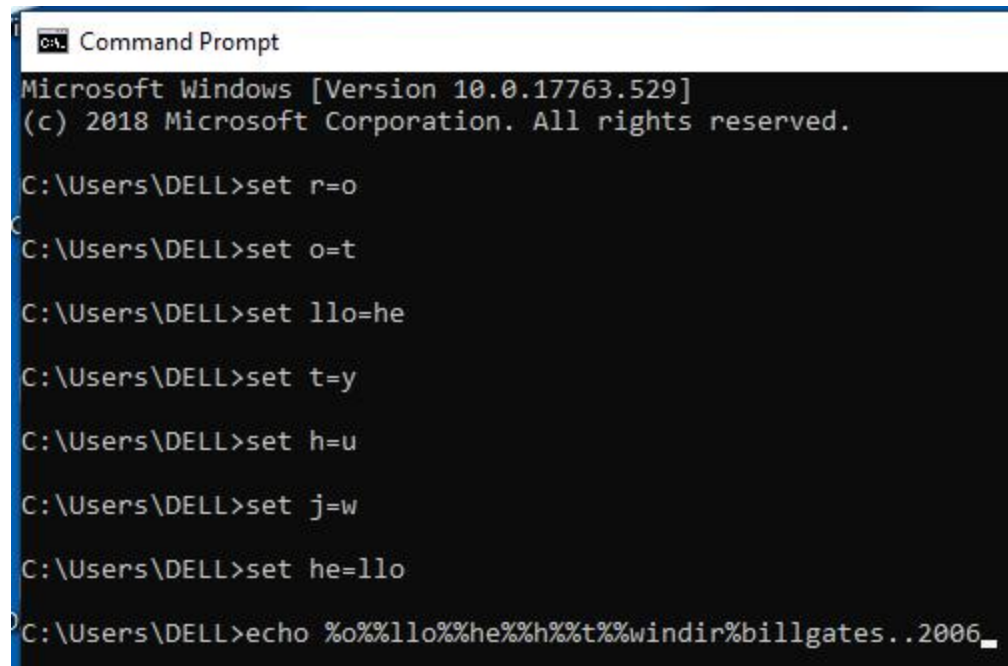
pause

exit

```

==> Ta để ý dòng **if "%password%"=="%o%%llo%%he%%h%%t%%windir%billgates..2006" goto good**

==> Nhập vào Command Prompt nội dung:



```

C:\Users\DELL>set r=o
C:\Users\DELL>set o=t
C:\Users\DELL>set llo=he
C:\Users\DELL>set t=y
C:\Users\DELL>set h=u
C:\Users\DELL>set j=w
C:\Users\DELL>set he=llo
C:\Users\DELL>echo %o%%llo%%he%%h%%t%%windir%billgates..2006_

```

=> Nhấn enter, thì trên màn hình trả về kết quả: `thellouyC:\Windowsbillgates..2006`

=> Nhấn dòng chữ này vào password thì kết quả là **good password**:

```
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 . . .
```

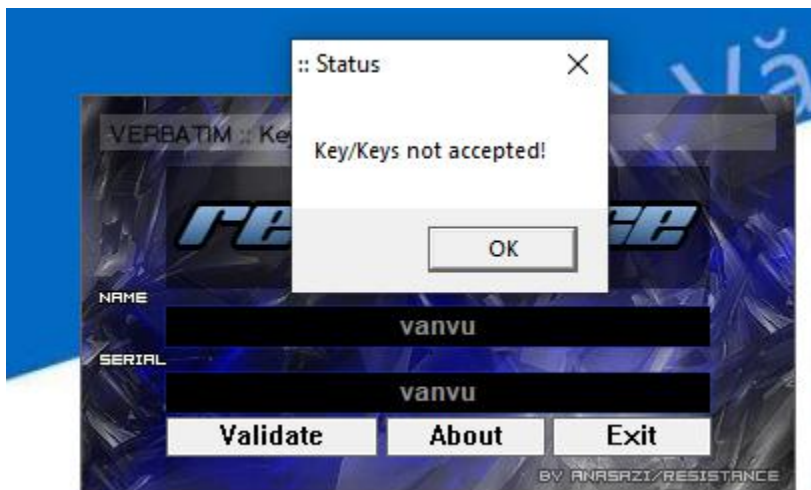
\***Thuật toán tạo pass**: lấy path → tạo file btxxxx.bat trong path → đọc **string pass** trong vùng nhớ dump vào file btxxxx.bat → chuyển cho Window thực thi file này.

\* **Kết luận password**: `thellouy<windir>Windowsbillgates..2006`

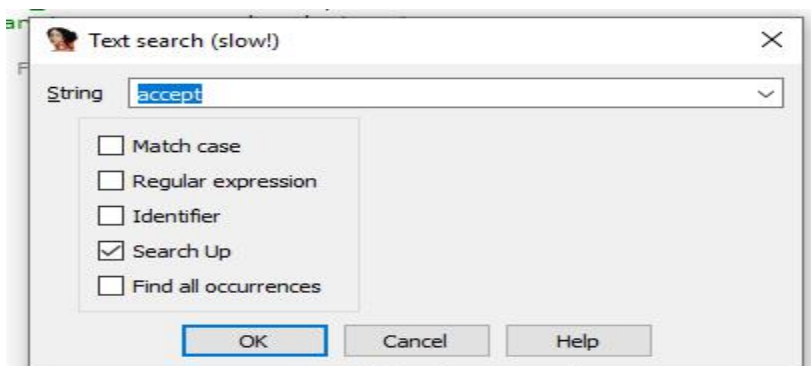
trong đó **<windir>** là nơi chứa hệ điều hành của máy (thường là: C:\)

### CÂU 1.4.

- Trước tiên ta nhập NAME và SERIAL bất kì, xuất hiện box chữ dòng chữ 'Key/Keys not accept!'



- Mở file 1.4.exe trong IDA rồi search text 'accept' để tìm ra manh mối:





→ Dòng chữ "Key/Keys not accept!" được tìm thấy, nó nằm trong loc\_4037E6 (đây chính là hàm badboy)

```

0.ELO:004037E6 ;
0.ELO:004037E6
0.ELO:004037E6 loc_4037E6: ; CODE XREF: sub_403500+2B4↑j
0.ELO:004037E6 ; sub_403500+2BD↑j ...
0.ELO:004037E6 mov esi, esp
0.ELO:004037E8 push 0 ; uType
0.ELO:004037EA push offset aStatus ; ":: Status"
0.ELO:004037EF push offset aKeyKeysNotAcce ; "Key/Keys not accepted!"
0.ELO:004037F4 mov edx, [ebp+hWnd]
0.ELO:004037F7 push edx
0.ELO:004037F7 ; CHAR aKeyKeysNotAcce[]
0.ELO:004037F7 aKeyKeysNotAcce db 'Key/Keys not accepted!',0
0.ELO:004037F7 ; DATA XREF: sub_403500+2EF↑o
0.ELO:004037F7

```

- Nhìn vào hàm loc\_4037E6 (badboy) thì hàm này ở được gọi đến ở hàm phía trên, ta tìm thấy dòng chữ "Serial Accepted" ở địa chỉ 004037CE

```

0.ELO:00403799 lea eax, [ebp+var_28]
0.ELO:0040379C push eax
0.ELO:0040379D call sub_415F30
0.ELO:004037A2 add esp, 8
0.ELO:004037A5 mov [ebp+var_58], eax
0.ELO:004037A8 call sub_401456
0.ELO:004037AD cmp dword_4962F4, 1
0.ELO:004037B4 jnz short loc_4037E6
0.ELO:004037B6 cmp dword_4962F8, 1
0.ELO:004037BD jnz short loc_4037E6
0.ELO:004037BF cmp [ebp+var_58], 0
0.ELO:004037C3 jnz short loc_4037E6
0.ELO:004037C5 mov esi, esp
0.ELO:004037C7 push 0 ; uType
0.ELO:004037C9 push offset Caption ; lpCaption
0.ELO:004037CE push offset Text ; "Serial Accepted"
0.ELO:004037D3 mov ecx, [ebp+hWnd]
0.ELO:004037D6 push ecx ; hWnd
0.ELO:004037D7 call MessageBoxA
0.ELO:004037DD cmp esi, esp
0.ELO:004037DF call sub_415640
0.ELO:004037E4 jmp short loc_403805
0.ELO:004037E6 ;
0.ELO:004037E6 loc_4037E6: ; CODE XREF: sub_403500+2B4↑j
0.ELO:004037E6 ; sub_403500+2BD↑j ...
0.ELO:004037F6

```

Ta thấy có 2 DWORD: dword\_4962F8 ở địa chỉ 004037B4 và dword\_4962F4 ở địa chỉ 004037B6 được so sánh với 1; 1 con trỏ ebp+var\_58 ở 004037BF được so sánh với 0.

- Tìm đến hàm sub\_401456 (add: 004037A8) ở phía trên rồi tìm đến sub\_402E80

```

0.ELO:004037A2 add esp, 8
0.ELO:004037A5 mov [ebp+var_58], eax
0.ELO:004037A8 call sub_401456
0.ELO:004037AD cmp dword_4962F4, 1
0.ELO:004037B4 jnz short loc_4037E6
0.ELO:004037B6 cmp dword_4962F8, 1
0.ELO:004037BD jnz short loc_4037E6
0.ELO:004037BF cmp [ebp+var_58], 0
0.ELO:004037C3 jnz short loc_4037E6
0.ELO:004037C5 mov esi, esp
0.ELO:004037C7 push 0
0.ELO:004037C9 push offset Caption
0.ELO:004037CE push offset Text
0.ELO:004037D3 mov ecx, [ebp+hWnd]
0.ELO:004037D6 push ecx
0.ELO:004037D7 call MessageBoxA
0.ELO:004037DD cmp esi, esp
0.ELO:004037DF call sub_415640
0.ELO:004037E4 jmp short loc_403805
0.ELO:004037E6 ;
0.ELO:004037E6 loc_4037E6: ; CODE XREF: sub_403500+2B4↑j
0.ELO:004037E6 ; sub_403500+2BD↑j ...
0.ELO:004037F6

```

→ Ta thấy có dòng chữ "RES-REGGED.txt" xuất hiện:

```

0.ELO:00402EB1      rep stosd
0.ELO:00402EB3      lea     eax, [ebp+var_144]
0.ELO:00402EB9      push    eax
0.ELO:00402EBA      lea     ecx, [ebp+var_1C]
0.ELO:00402EBD      call   sub_401564
0.ELO:00402EC2      mov     [ebp+var_4], 0
0.ELO:00402EC9      push    1
0.ELO:00402ECB      lea     ecx, [ebp+var_B0]
0.ELO:00402ED1      call   sub_401131
0.ELO:00402ED6      mov     byte ptr [ebp+var_4], 1
0.ELO:00402EDA      push    1
0.ELO:00402EDC      lea     ecx, [ebp+var_140]
0.ELO:00402EE2      call   sub_401131
0.ELO:00402EE7      mov     byte ptr [ebp+var_4], 2
0.ELO:00402EEB      push    1
0.ELO:00402EED      push    offset aResReggedTxt ; "RES-REGGED.txt"
0.ELO:00402EF2      lea     ecx, [ebp+var_B0]
0.ELO:00402EF8      call   sub_401168
0.ELO:00402EFD      mov     ecx, [ebp+var_B0]
0.ELO:00402F03      mov     edx, [ecx+4]
0.ELO:00402F06      lea     ecx, [ebp+edx+var_B0]
0.ELO:00402F0D      call   sub_401226
0.ELO:00402F12      and     eax, 0FFh
0.ELO:00402F17      test    eax, eax
0.ELO:00402F19      jz      short loc_402F27
0.ELO:00402F1B      mov     dword_4962F0, 0

```

Phân tích:

00402EED            push   offset aResReggedTxt ; ASCII"RES-REGGED.txt"

00402EF2            lea    ecx, [ebp+var\_B0]            ; Check RES-REGGED.txt

- Di chuyển xuống dưới, ta thấy:

```

0.ELO:00402F5C      mov     eax, [edx+4]
0.ELO:00402F5F      lea     ecx, [ebp+eax+var_B0]
0.ELO:00402F66      mov     [ebp+var_148], ecx
0.ELO:00402F6C      loc_402F6C:      ; CODE XREF: sub_402E80+D4↑j
0.ELO:00402F6C      mov     ecx, [ebp+var_148]
0.ELO:00402F72      call   sub_40148D
0.ELO:00402F77      test    eax, eax
0.ELO:00402F79      jz      short loc_402FF1
0.ELO:00402F7B      cmp     [ebp+var_20], 1
0.ELO:00402F7F      jl      short loc_402FDD     >1
0.ELO:00402F81      cmp     [ebp+var_20], 5
0.ELO:00402F85      jg      short loc_402FDD     <5
0.ELO:00402F87      call   sub_4015D7     Check xong
0.ELO:00402F8C      push    1
0.ELO:00402F8E      push    offset aResValidateDiz ; "RES-VALIDATE.diz"
0.ELO:00402F93      lea     ecx, [ebp+var_140]     check RES-VALIDATE.diz
0.ELO:00402F99      call   sub_401168
0.ELO:00402F9E      mov     edx, [ebp+var_140]
0.ELO:00402FA4      mov     eax, [edx+4]
0.ELO:00402FA7      lea     ecx, [ebp+eax+var_140]
0.ELO:00402FAE      call   sub_401226
0.ELO:00402FB3      and     eax, 0FFh
0.ELO:00402FB8      test    eax, eax
0.ELO:00402FBA      jz      short loc_402FC8
0.ELO:00402FBC      mov     dword_4962F8, 0
0.ELO:00402FC6      jmp     short loc_402FD2
0.ELO:00402FC8 : -----

```

- Đến đây, ta chưa nói được gì (nhưng thấy được rằng file RES-REGGED.txt sẽ chứa 1 trong các giá trị: 1, 2, 3, 4, 5), tiếp tục di chuyển xuống dưới, tìm thấy thuật toán của SERIAL ở loc\_40368C tại 0040368C:

```

0.ELO:0040369E      push     6Ah          ; nIDDlgItem
0.ELO:004036A0      mov      ecx, [ebp+hWnd]
0.ELO:004036A3      push     ecx          ; hDlg
0.ELO:004036A4      call     GetDlgItemTextA
0.ELO:004036AA      cmp      esi, esp
0.ELO:004036AC      call     sub_415640
0.ELO:004036B1      lea      edx, [ebp+String]
0.ELO:004036B4      push     edx
0.ELO:004036B5      call     sub_416040
0.ELO:004036BA      add      esp, 4
0.ELO:004036BD      mov      esi, esp
0.ELO:004036BF      push     14h          ; cchMax
0.ELO:004036C1      lea      eax, [ebp+var_28]
0.ELO:004036C4      push     eax          ; lpString
0.ELO:004036C5      push     68h          ; nIDDlgItem
0.ELO:004036C7      mov      ecx, [ebp+hWnd]
0.ELO:004036CA      push     ecx          ; hDlg
0.ELO:004036CB      call     GetDlgItemTextA
0.ELO:004036D1      cmp      esi, esp
0.ELO:004036D3      call     sub_415640
0.ELO:004036D8      lea      edx, [ebp+var_28]
0.ELO:004036DB      push     edx
0.ELO:004036DC      call     sub_416040
0.ELO:004036E1      add      esp, 4
0.ELO:004036E4      lea      eax, [ebp+String]
0.ELO:004036E7      push     eax
0.ELO:004036E8      call     sub_415FC0
0.ELO:004036ED      add      esp, 4
0.ELO:004036F0      mov      [ebp+var_40], eax
0.ELO:004036F3      mov      ecx, [ebp+var_48]
0.ELO:004036F6      xor      ecx, [ebp+var_48]
000036B4 0000000000004036B4: sub_403500+1B4 (Synchronized with Hex View-1)

```

→ Phân tích mã hợp ngữ:

```

004036A0      mov      ecx, [ebp+hWnd]

004036A3      push     ecx          ; hDlg

004036A4      call     GetDlgItemTextA

004036AA      cmp      esi, esp     ; EAX = độ dài của Name

004036AC      call     sub_415640

004036B1      lea      edx, [ebp+String]

004036B4      push     edx          ; EDX = Name

004036B5      call     sub_416040

004036BA      add      esp, 4       ; EAX = reverse Name

004036BD      mov      esi, esp

004036BF      push     14h          ; cchMax

```



```

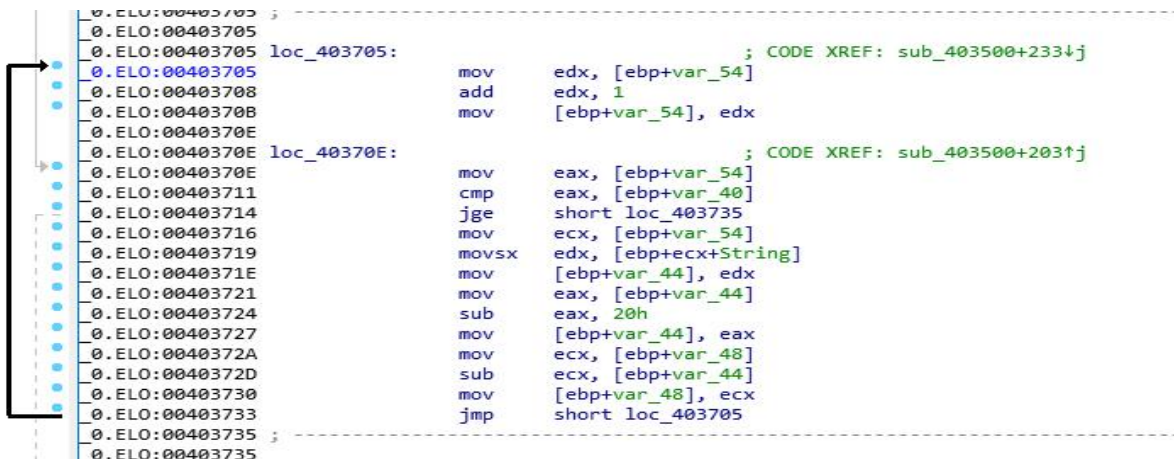
004036C1    lea  eax, [ebp+var_28]
004036C4    push eax          ; lpString
004036C5    push 6Bh          ; nIDDlgItem
004036C7    mov  ecx, [ebp+hWnd]
004036CA    push ecx          ; hDlg
004036CB    call GetDlgItemTextA
004036D1    cmp  esi, esp      ; EAX = độ dài của Serial
004036D3    call sub_415640
004036D8    lea  edx, [ebp+var_28]
004036DB    push edx
004036DC    call sub_416040
004036E1    add  esp, 4
004036E4    lea  eax, [ebp+String]
004036E7    push eax
004036E8    call sub_415FC0
004036ED    add  esp, 4        ; lấy độ dài của Name từ EAX
004036F0    mov  [ebp+var_40], eax

```

→ **sub\_416040**: lấy 1 chuỗi và trả về chuỗi đảo ngược.

**sub\_415FC0**: lấy 1 chuỗi và trả về độ dài chuỗi.

- Di chuyển xuống dưới, tìm được thuật toán tạo ra serial:



```

0.ELO:00403705 ;
0.ELO:00403705 loc_403705: mov     edx, [ebp+var_54] ; CODE XREF: sub_403500+233↓j
0.ELO:00403708 add     edx, 1
0.ELO:00403708 mov     [ebp+var_54], edx
0.ELO:0040370E loc_40370E: mov     eax, [ebp+var_54] ; CODE XREF: sub_403500+203↑j
0.ELO:0040370E cmp     eax, [ebp+var_40]
0.ELO:00403711 jge     short loc_403735
0.ELO:00403714 mov     ecx, [ebp+var_54]
0.ELO:00403719 movsx   edx, [ebp+ecx+String]
0.ELO:0040371E mov     [ebp+var_44], edx
0.ELO:00403721 mov     eax, [ebp+var_44]
0.ELO:00403724 sub     eax, 20h
0.ELO:00403727 mov     [ebp+var_44], eax
0.ELO:0040372A mov     ecx, [ebp+var_48]
0.ELO:0040372D sub     ecx, [ebp+var_44]
0.ELO:00403730 mov     [ebp+var_48], ecx
0.ELO:00403733 jmp     short loc_403705
0.ELO:00403735 ;
0.ELO:00403735

```



→ Phân tích mã hợp ngữ:

```

00403705 loc_403705:                                ; CODE XREF: sub_403500+233↓j
00403705      mov     edx, [ebp+var_54]
00403708      add     edx, 1
0040370B      mov     [ebp+var_54], edx
0040370E
0040370E loc_40370E:                                ; CODE XREF: sub_403500+203↑j
0040370E      mov     eax, [ebp+var_54]
00403711      cmp     eax, [ebp+var_40] ; EAX = count
00403714      jge     short loc_403735 ; nhảy đến loc_403735 nếu count = độ dài Name
00403716      mov     ecx, [ebp+var_54] ; ECX = count
00403719      movsx   edx, [ebp+ecx+String] ; đặt kí tự char hiện tại vào EDX
0040371E      mov     [ebp+var_44], edx ; đặt kí tự char hiện tại vào var_44
00403721      mov     eax, [ebp+var_44] ; var_44=EAX
00403724      sub     eax, 20h          ; EAX-20h (tức là 0x20, có giá trị 32 trong hệ 10)
00403727      mov     [ebp+var_44], eax ; var_44 = giá trị mới (EAX)
0040372A      mov     ecx, [ebp+var_48] ; ECX=var_48
0040372D      sub     ecx, [ebp+var_44] ; ECX-var44
00403730      mov     [ebp+var_48], ecx
00403733      jmp     short loc_403705

```

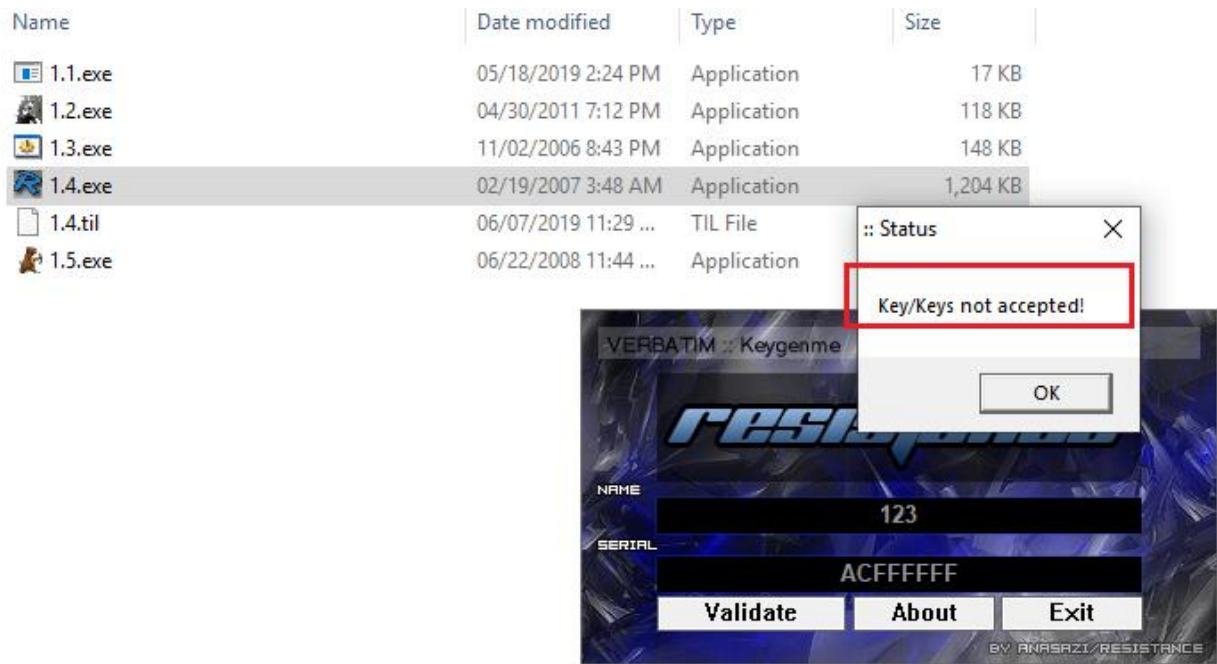
→ Thuật toán tạo ra serial được biểu diễn bằng đoạn code:

```

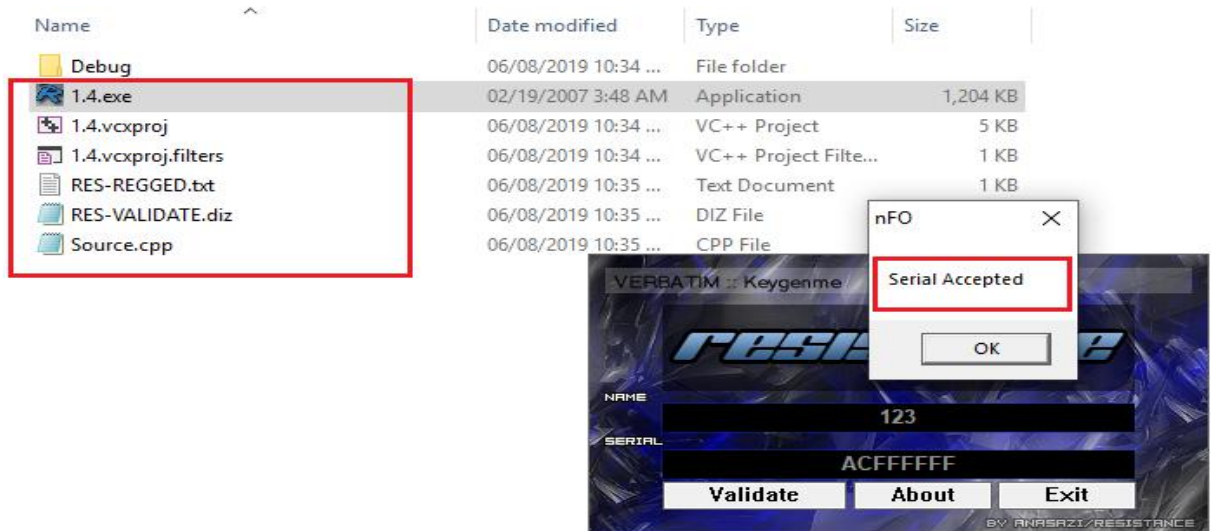
for(int i=0;i<strlen(Name);i++)
    var_48 -= (NameRev[i] - 0x20); //NameRev[i] - 0x20 = var_44
//mov     eax, [ebp + var_44]; var_44 = EAX
//sub     eax, 20h; EAX - 20h
//mov[ebp + var_44], eax
//mov     ecx, [ebp + var_48]; ECX = var_48
//sub     ecx, [ebp + var_44]; ECX - var44

```

Bằng, thuật toán trên, ta chọn **NAME = 123**, tìm được **SERIAL = ACFFFFFF**. Nhưng khi nhập vào thì kết quả không chính xác → Serial vẫn chưa được thiết lập???



- Như ở trên, có 2 file là RES-REGGED.txt và RES-VALIDATE.diz được tạo ra. Qua nhiều lần thử, nhận thấy serial được đặt trong call chứa 2 file này. Bây giờ, ta đặt file 1.4 exe trong project được code bằng C khi đã debug → **KẾT QUẢ CHÍNH XÁC!**



**CÂU 1.5.**

Ta thử xem cách **Search for -> All referenced string:**

00401400	PUSH 1_5.00401405	ASCII "Error"
004014E2	PUSH 1_5.004014AE	ASCII "Your name must be at least one byte!"
00401643	PUSH 1_5.00403079	ASCII " Welcome to a Crackme "
0040165F	PUSH 1_5.00403090	ASCII " Ribbere 1.42 "
0040167B	PUSH 1_5.004030C4	ASCII " your host: Bswap "
00401697	PUSH 1_5.004030EC	ASCII " Written in 100% ASM "
004016B3	PUSH 1_5.0040310E	ASCII " Some greetings go to: "
004016CF	PUSH 1_5.00403130	ASCII " All "
004016EB	PUSH 1_5.00403165	ASCII " Members "
00401707	PUSH 1_5.0040319A	ASCII " of "
00401723	PUSH 1_5.004031C6	ASCII " the "
00401758	PUSH 1_5.004031F2	ASCII " crackmes.de TEAM "
00401780	PUSH 1_5.00403216	ASCII " Tools used "
004017A9	PUSH 1_5.00403242	ASCII " RadASM and Masm 32 V10 "
004017C5	PUSH 1_5.00403269	ASCII " Made in Holland "

Để ý dòng **"Your name must be at least one byte!"**, ta đoán đây là dòng thông báo khi người dùng không nhập gì và bấm register. Tiến hành nhấp vào và quan sát:

00401400	> 6A 40	PUSH 40	Style = MB_OK MB_ICONASTERISK MB_APPLMODAL
0040140D	. 68 D5144000	PUSH 1_5.00401405	Title = "Error"
004014E2	. 68 AE144000	PUSH 1_5.004014AE	Text = "Your name must be at least one byte!"
004014E7	. 6A 00	PUSH 0	hOwner = NULL
004014E9	. E8 3A080000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004014EE	. 6A 00	PUSH 0	Enable = FALSE
004014F0	. FF35 60304000	PUSH DWORD PTR [403060]	hWnd = NULL
004014F6	. E8 D3070000	CALL <JMP.&user32.EnableWindow>	EnableWindow
004014FB	. E9 99000000	JMP 1_5.00401599	
00401500	> 50	PUSH EAX	
00401501	. E8 55040000	CALL 1_5.0040195B	

Sau khi kiểm tra người dùng có nhập hay không, sẽ có lệnh gọi đến **0040195B**, ta thử đặt BP và nhảy vào hàm tại đây:

Ta nhập vào **123** và quan sát kết quả.

0040195B	. 55	PUSH EBP	
0040195C	. 8BEC	MOV EBP,ESP	
0040195E	. BE 1C334000	MOV ESI,1_5.0040331C	ASCII "123"
00401963	. BF 3B334000	MOV EDI,1_5.0040333B	
00401968	. B9 10000000	MOV ECX,10	
0040196D	> 0FB606	MOVZX EAX,BYTE PTR [ESI]	
00401970	. 51	PUSH ECX	
00401971	. 50	PUSH EAX	
00401972	. E8 08000000	CALL 1_5.0040197F	
00401977	. 8907	MOV DWORD PTR [EDI],EAX	
00401979	. 47	INC EDI	
0040197A	. 46	INC ESI	
0040197B	. E2 F0	LOOPD SHORT 1_5.0040196D	
0040197D	. C9	LEAVE	
0040197E	. C3	RET	

Các bước chuẩn bị: Lưu giá trị nhập vào thanh **ESI** là **"123"**, set giá trị 10 cho **ECX**, chuyển giá trị **ESP** (địa chỉ đỉnh stack) vào **EBP**.

Bắt đầu tại dòng **0040196D**, thực hiện chuyển 1 byte từ vùng nhớ trở tới bởi **ESI** vào **EAX**, **EAX** lúc này mang giá trị 31 -> Đây là dạng hệ 16 của kí tự số 1 trong bảng mã ASCII, tức là kí tự đầu tiên trong chuỗi key mà ta đã nhập vào (123). Sau đó push **ECX** và **EAX** vào stack.

Tại địa chỉ 401972, tiếp tục ta có lệnh gọi đến **0040197F**, lần lượt nhấn F8 cho đến khi đến dòng **004019C5**, đây là nơi bắt đầu quá trình mã hóa key nhập vào.

004019C5	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]	
004019C8	. 03C1	ADD EAX,ECX	
004019CA	. 83F8 21	CMP EAX,21	
004019CD	. 73 03	JNB SHORT 1_5.004019D2	
004019CF	. 83C0 21	ADD EAX,21	
004019D2	. 83F8 7B	CMP EAX,7B	
004019D5	. 7E 02	JLE SHORT 1_5.004019D9	
004019D7	. D1E8	SHR EAX,1	
004019D9	. 8945 08	MOV DWORD PTR [EBP+8],EAX	
004019DC	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]	
004019DF	. C9	LEAVE	
004019E0	. C3	RET	

Ta phân tích các dòng code:

- Chuyển giá trị tại địa chỉ **EBP + 8** vào **EAX**

EAX: 00000031

- Cộng **EAX** với **ECX** (**ECX** được khởi tạo với giá trị là 10 hệ 6, tức 16 ở hệ 10).

EAX: 00000041

- So sánh **EAX** với **21**, nếu **EAX >= 21** thì nhảy đến **004019D2**. Ở đó, tiếp tục so sánh với **7B**, nếu **EAX <= 7B** thì tiếp tục nhảy đến **004019D9**. Ở đây, ta shift right **EAX** 1 lần, rồi lưu giá trị **EAX** vào địa chỉ **EBP + 8**, sau đó lưu ngược giá trị tại **EBP + 8** lại vào **EAX**.
- Thoát khỏi hàm, trở lại đoạn code phía trên.

0040195B	. 55	PUSH EBP	
0040195C	. 8BEC	MOV EBP,ESP	
0040195E	. BE 1C334000	MOV ESI,1_5.0040331C	ASCII "123"
00401963	. BF 3B334000	MOV EDI,1_5.0040333B	
00401968	. B9 10000000	MOV ECX,10	
0040196D	. 0FB606	MOVZX EAX,BYTE PTR [ESI]	
00401970	. 51	PUSH ECX	
00401971	. 50	PUSH EAX	
00401972	. E8 08000000	CALL 1_5.0040197F	
00401977	. 8907	MOV DWORD PTR [EDI],EAX	
00401979	. 47	INC EDI	
0040197A	. 46	INC ESI	
0040197B	. E2 F0	LOOPD SHORT 1_5.0040196D	
0040197D	. C9	LEAVE	
0040197E	. C3	RET	

- Chuyển giá trị **EAX** vào vùng nhớ **EDI**.
- Sau đó tăng **EDI** và **ESI** lên 1 đơn vị, rồi tiếp tục lặp lại từ dòng **0040196D** (kết thúc 1 lần lặp là **ECX** trừ đi 1 đơn vị)
- Như vậy, ta lại tiếp tục đưa byte tiếp theo trong chuỗi ký tự nhập vào để xử lý mã hóa cho đến **ECX = 0**. Các ký tự được mã hóa của mỗi lần lặp được push vào stack.
- Sau khi lặp xong 16 lần, ta trở về đoạn code:

00401907	. E8 55040000	CALL 1_5.0040195B	
0040190E	. A1 3D334000	MOV EAX,DWORD PTR [40333D]	
0040190B	. 8B1D 41334000	MOV EBX,DWORD PTR [403341]	
00401911	. A3 E4324000	MOV DWORD PTR [4032E4],EAX	
00401916	. 891D E8324000	MOV DWORD PTR [4032E8],EBX	
0040191C	. E8 C0040000	CALL 1_5.004019E1	
00401921	. 33C0	XOR EAX,EAX	
00401923	. 33DB	XOR EBX,EBX	

- Chuyển 1 word từ địa chỉ **40333D** vào thanh **EAX**.
- Chuyển 1 word từ địa chỉ **403341** vào thanh **EBX**.
- Sau đó lần lượt chuyển 2 giá trị trên vào địa chỉ **4032E4** và **4032E8**.
- Nhảy đến lệnh **004019E1**, ta tiến hành chạy từng bước.



004019E1	8D3D E4324000	LEA EDI,DWORD PTR [4032E4]
004019E7	6A 00	PUSH 0
004019E9	6A 00	PUSH 0
004019EB	50	PUSH EAX
004019EC	DF2C24	FILD QWORD PTR [ESP]
004019EF	DF3424	FBSTP TBYTE PTR [ESP]
004019F2	59	POP ECX
004019F3	58	POP EAX
004019F4	8BD1	MOV EDX,ECX
004019F6	8BD8	MOV EBX,EAX
004019F8	C1E9 04	SHR ECX,4
004019FB	C1E8 04	SHR EAX,4
004019FE	83E3 0F	AND EBX,0F
00401A01	81E2 0F0F0F0F	AND EDX,0F0F0F0F
00401A07	81E1 0F0F0F0F	AND ECX,0F0F0F0F
00401A0D	81C2 30303030	ADD EDX,30303030
00401A13	81C1 30303030	ADD ECX,30303030
00401A19	83C0 30	ADD EAX,30
00401A1C	83C3 30	ADD EBX,30
00401A1F	8807	MOV BYTE PTR [EDI],AL
00401A21	885F 01	MOV BYTE PTR [EDI+1],BL
00401A24	884F 08	MOV BYTE PTR [EDI+8],CL
00401A27	8857 09	MOV BYTE PTR [EDI+9],DL
00401A2A	886F 06	MOV BYTE PTR [EDI+6],CH
00401A2D	8877 07	MOV BYTE PTR [EDI+7],DH
00401A30	0FC9	BSWAP ECX
00401A32	0FCA	BSWAP EDX
00401A34	884F 02	MOV BYTE PTR [EDI+2],CL
00401A37	8857 03	MOV BYTE PTR [EDI+3],DL
00401A3A	886F 04	MOV BYTE PTR [EDI+4],CH
00401A3D	8877 05	MOV BYTE PTR [EDI+5],DH
00401A40	58	POP EAX
00401A41	C3	RET

==> Phân tích mã hợp ngữ:

#### #Đầu tiên số được chuyển đổi thành giá trị dấu phẩy động

004019E1 /\$ 8D3D E4324000 LEA EDI,DWORD PTR [4032E4]

004019E7 |. 6A 00      PUSH 0

004019E9 |. 6A 00      PUSH 0

004019EB |. 50          PUSH EAX

004019EC |. DF2C24      FILD QWORD PTR [ESP]

#### # chuyển đổi thành số nguyên BCD được đóng gói 18 chữ số

004019EF |. DF3424      FBSTP TBYTE PTR [ESP]

#### #sau đó 4 byte đầu tiên của giá trị bcd được pack được và vào cả ECX và EDX

004019F2 |. 59          POP ECX

004019F3 |. 58          POP EAX

004019F4 |. 8BD1        MOV EDX,ECX

004019F6 |. 8BD8        MOV EBX,EAX

#### # ECX được dịch phải 4 lần

004019F8 |. C1E9 04      SHR ECX,4

004019FB |. C1E8 04      SHR EAX,4

#### #cl và dl đưọc thực hiện phép AND

004019FE |. 83E3 0F      AND EBX,0F

00401A01 |. 81E2 0F0F0F0F AND EDX,0F0F0F0F

00401A07 |. 81E1 0F0F0F0F AND ECX,0F0F0F0F

**#cl và dl được cộng thêm 0x30**

00401A0D |. 81C2 30303030 ADD EDX,30303030

00401A13 |. 81C1 30303030 ADD ECX,30303030

00401A19 |. 83C0 30     ADD EAX,30

00401A1C |. 83C3 30     ADD EBX,30

**#sau đó CL được lưu trong địa chỉ được chỉ EDI + 8 và dl trong EDI + 9**

*#code trong C:*

```
mov[EDIPlus8], cl;
```

```
mov[EDIPlus9], dl;
```

00401A1F |. 8807     MOV BYTE PTR [EDI],AL

00401A21 |. 885F 01   MOV BYTE PTR [EDI+1],BL

00401A24 |. 884F 08   MOV BYTE PTR [EDI+8],CL

00401A27 |. 8857 09   MOV BYTE PTR [EDI+9],DL

00401A2A |. 886F 06   MOV BYTE PTR [EDI+6],CH

00401A2D |. 8877 07   MOV BYTE PTR [EDI+7],DH

00401A30 |. 0FC9     BSWAP ECX

00401A32 |. 0FCA     BSWAP EDX

00401A34 |. 884F 02   MOV BYTE PTR [EDI+2],CL

00401A37 |. 8857 03   MOV BYTE PTR [EDI+3],DL

00401A3A |. 886F 04   MOV BYTE PTR [EDI+4],CH

00401A3D |. 8877 05   MOV BYTE PTR [EDI+5],DH

00401A40 |. 58     POP EAX

00401A41 |. C3     RET

- Lưu ý rằng: hai byte đầu tiên của bộ đệm được chuyển đổi được chuyển vào AX và các giá trị được tính từ trên vào BX:

```
key[0] = EDIPlus8;
```

```
key[1] = EDIPlus9;
```

0040151C	. E8 C0040000	CALL 1_5.004019E1
00401521	. 33C0	XOR EAX,EAX
00401523	. 33DB	XOR EBX,EBX
00401525	. 66:A1 3B3340	MOV AX,WORD PTR [40333B]
0040152B	. 66:8B5F 08	MOV BX,WORD PTR [EDI+8]

- Sau đó, các giá trị được trừ và ax được sửa đổi thêm:

```
key[0] -= 16;
```

```
key[1] -= 15;
```

0040152F	. 66 2BC3	SUB AX,BX
00401532	. 35 3F1B0000	XOR EAX,1B3F
00401537	. 2D 23010000	SUB EAX,123
0040153C	.v EB 03	JMP SHORT 1_5.00401541

- Sau đó, nhảy đến các hàm tại add 00401599 và 0040154A để in ra các thông báo.

00401541	> 0BC0	OR EAX,EAX
00401543	.v 75 54	JNZ SHORT 1_5.00401599
00401545	.v EB 03	JMP SHORT 1_5.0040154A

\* Thuật toán phát sinh key được cài đặt rõ ràng trong keygen.

==> Ta nhập một key tạo được: **==@eWB37G\$SF** ==> CHO KẾT QUẢ : **(Registered), post your solution please** có nghĩa là Key đã đăng kí, vui lòng gửi bài giải.



\_\_ HẾT \_\_