

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

MÔN HỌC
KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ

BÁO CÁO ĐỒ ÁN 3
CRACK PHẦN MỀM

Thành viên:	18120027:	Nguyễn Thị Thu Hằng
	18120056:	Nguyễn Xuân Mai
	18120066:	Bùi Đoàn Hữu Nhân
	18120085:	Nguyễn Tấn Thìn
	18120090:	Phạm Nguyên Minh Thy

MỤC LỤC

0. Phân công	3
1. Chương trình 1_1	3
a. Manh mối	3
2. Chương trình 1_2	9
a. Manh mối	10
b. Đề xuất thuật toán keygen.....	19
3. Chương trình 1_3	20
a. Manh mối	21
b. Đề xuất thuật toán keygen.....	27
4. Đánh giá mức độ hoàn thành	29
5. Tài liệu tham khảo.....	29

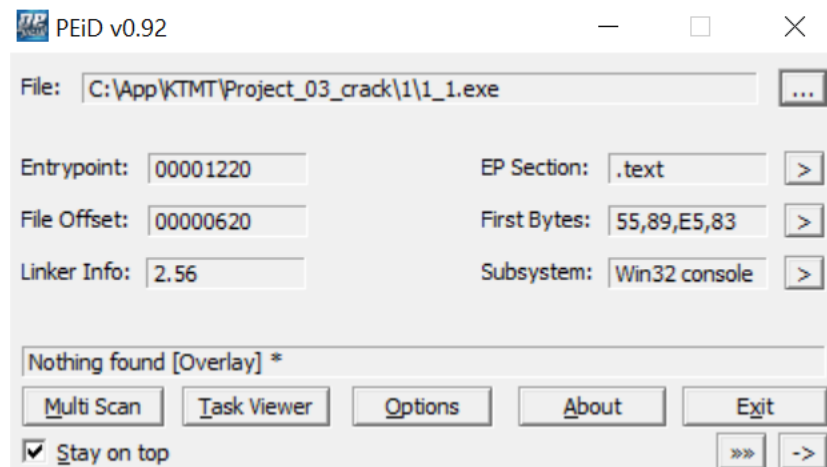
0. Phân công

<i>MSSV</i>	<i>Họ tên</i>	<i>Công việc</i>
18120027	Nguyễn Thị Thu Hằng	File 1_2 Báo cáo 1_2
18120056	Nguyễn Xuân Mai	File 1_3 Báo cáo 1_3, báo cáo tổng hợp
18120066	Bùi Đoàn Hữu Nhân	File 1_2 Báo cáo 1_2
18120085	Nguyễn Tấn Thìn	File 1_1 Báo cáo 1_1
18120090	Phạm Nguyên Minh Thy	File 1_3 Báo cáo 1_3

ĐỀ: $7 + 6 + 6 + 5 + 0 = 24 \bmod 3 = 0 + 1 = 1 \rightarrow \text{đề 1.}$

1. Chương trình 1_1

Trước tiên kiểm tra file bằng phần mềm PEiD.



Nothing found \rightarrow PEiD chưa nhận dạng được.

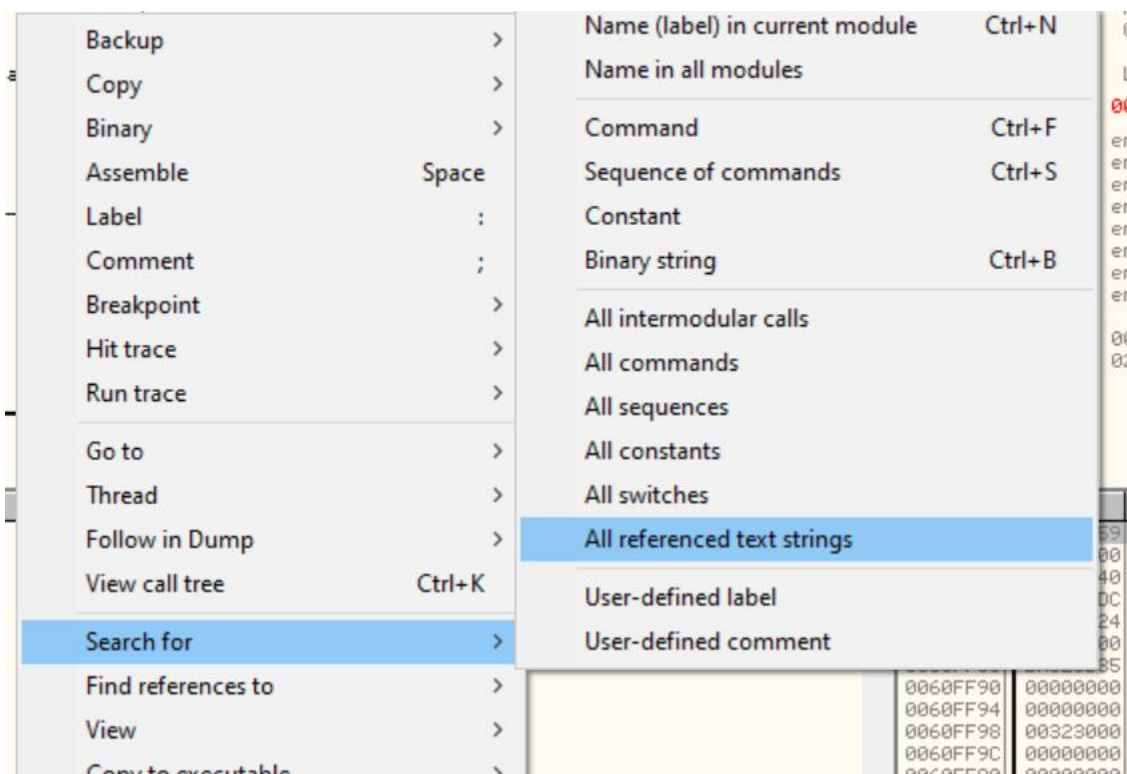
Đừng lo lắng \rightarrow Tiếp tục mở chương trình trên với OllyDbg.

a. Manh mối

Sau khi mở chương trình bằng OllyDbg.

Address	Hex dump	Disassembly	Comment
<ModuleEntr	\$ 55	PUSH EBP	
00401221	. 89E5	MOV EBP,ESP	
00401223	. 83EC 08	SUB ESP,8	
00401226	. C70424 010000	MOV DWORD PTR [ESP],1	
0040122D	. FF15 28514000	CALL DWORD PTR [<&msvort.__set_app_type	msvort.__set_app_type
00401233	. E8 C8FEFFFF	CALL 1_1.00401100	
00401238	. 90	NOP	
00401239	. 8DB426 000000	LEA ESI,DWORD PTR [ESI]	
00401240	. 55	PUSH EBP	
00401241	. 89E5	MOV EBP,ESP	
00401243	. 83EC 08	SUB ESP,8	
00401246	. C70424 020000	MOV DWORD PTR [ESP],2	
0040124D	. FF15 28514000	CALL DWORD PTR [<&msvort.__set_app_type	msvort.__set_app_type
00401253	. E8 A8FEFFFF	CALL 1_1.00401100	
00401258	. 90	NOP	
00401259	. 8DB426 000000	LEA ESI,DWORD PTR [ESI]	
00401260	\$ 55	PUSH EBP	
00401261	. 8B0D 40514000	MOV ECX,DWORD PTR [<&msvort.atexit>]	msvort.atexit
00401267	. 89E5	MOV EBP,ESP	
00401269	. 5D	POP EBP	
0040126A	. FFE1	JMP ECX	
0040126C	. 8D7426 00	LEA ESI,DWORD PTR [ESI]	
00401270	. 55	PUSH EBP	
00401271	. 8B0D 34514000	MOV ECX,DWORD PTR [<&msvort._onexit>]	msvort._onexit
00401277	. 89E5	MOV EBP,ESP	
00401279	. 5D	POP EBP	
0040127A	. FFE1	JMP ECX	
0040127C	. 90	NOP	
0040127D	. 90	NOP	
0040127E	. 90	NOP	
0040127F	. 90	NOP	
00401280	> 55	PUSH EBP	
00401281	. 89E5	MOV EBP,ESP	

Chọn All referenced text strings để tìm ra tất cả các string được reference trong chương trình.



Kết quả thu được như sau:

Đặt breakpoint tại địa chỉ 004013EA (địa chỉ của lệnh so sánh) và thử nhập password với số lượng ký tự lớn hơn 6.

```

C:\App\KTMT\Project_03_crack\1\1_1.exe
8 .oPYo. 8
8 .o8 8
8oPYo. oPYo. 8 .P'8 8 .o .oPYo. odYo.
8 8 8 ' 8.d' 8 8oP' 8oooo8 8' `8
8 8 8 8o' 8 8 `b. 8. 8 8
`YooP' 8 `YooP' 8 `o. `Yooo' 8 8
.....
CrackMe #4 (BruteforceMe)
Password : leeminho

```

Sau khi F8 một số lần ta thấy liền nhảy đến badboy. Nghĩa là nếu số ký tự không phải là 6 thì không cần kiểm tra gì thêm mà nhảy thẳng đến badboy.

```

0040140B > C70424 87324 MOV DWORD PTR [ESP],1_1.00403287
004014E2 . E8 99050000 CALL <JMP.&msvcrt.printf>
004014E7 > E8 04040000 CALL <JMP.&msvcrt._getch>
004014EC . B8 00000000 MOV EAX,0
004014F1 . C9 LEAVE
004014F3 . C9 RET
ASCII "Nope... try again."
printf
_getch

```

Vậy đã xác định được số ký tự trong password phải bằng 6. Bây giờ chúng ta sẽ nhập random bất kỳ 1 password 6 ký tự và tìm ra đáp án cho chương trình này.

Vẫn đặt breakpoint tại địa chỉ 004013EA, bây giờ ta sẽ F8 từng dòng lệnh và xem điều gì đã xảy ra với password vừa nhập 😊).

```

0040138E . C70424 F4314 MOV DWORD PTR [ESP],1_1.00403287
00401395 . E8 E6060000 CALL <JMP.&msvcrt.printf>
0040139A . C70424 1F324 MOV DWORD PTR [ESP],1_1.00403287
004013A1 . E8 DA060000 CALL <JMP.&msvcrt._getch>
004013A6 . C70424 3C324 MOV DWORD PTR [ESP],1_1.00403287
004013AD . E8 CE060000 CALL <JMP.&msvcrt._getch>
004013B2 . 8085 28FFFFFF LEA EAX,DWORD PTR [EAX]
004013B8 . 894424 04 MOV DWORD PTR [EAX],0
004013BC . C70424 4B324 MOV DWORD PTR [ESP],1_1.00403287
004013C3 . E8 A8060000 CALL <JMP.&msvcrt._getch>
004013C8 . 8085 28FFFFFF LEA EAX,DWORD PTR [EAX]
004013CE . 890424 MOV DWORD PTR [EAX],0
004013D1 . E8 8A060000 CALL <JMP.&msvcrt._getch>
004013D6 . 8985 78F0FFF MOV DWORD PTR [EAX],0
004013DC . 8085 28FFFFFF LEA EAX,DWORD PTR [EAX]
004013E2 . 890424 MOV DWORD PTR [EAX],0
004013E5 . E8 76060000 CALL <JMP.&msvcrt._getch>
004013EA . 83F8 06 CMP EAX,6
004013ED . 0F85 BE0000 JNZ 1_1.00403287
004013F3 . 0FB685 28FFF MOVZX EAX,BYTE PTR [EAX]
004013FA . 34 34 XOR AL,34
004013FC . 0FBEC0 MOVSB EAX,EAX
004013FF . 8985 74F0FFF MOV DWORD PTR [EAX],0
00401405 . 0FB685 29FFF MOVZX EAX,BYTE PTR [EAX]
CrackMe #4 (BruteforceMe)
Password : skymtp

```

004013E5	. E8 76060000	CALL <JMP.&msvcrt.strlen>	EAX 00000006
004013E8	. 83F8 06	CMP EAX,6	ECX 0060FE50 ASCII "skyntp"
004013ED	. 0FB685 BE000000	JNZ I_1.004014B1	EDX 7EFF6F73
004013F3	. 0FB685 28FFF	MOVZX EAX,BYTE PTR [EBP-D8]	EBX 00004000
004013FA	. 34 34	XOR AL,34	ESP 0060FC40
004013FC	. 0FBEC0	MOVSX EAX,AL	EBP 0060FF28
004013FF	. 8985 74D0FFF	MOV DWORD PTR [EBP-28C],EAX	ESI 00401220 I_1.<ModuleEntryPoint>
00401405	. 0FB685 29FFF	MOVZX EAX,BYTE PTR [EBP-D7]	EDI 00401220 I_1.<ModuleEntryPoint>
0040140C	. 34 78	XOR AL,78	EIP 004013ED I_1.004013ED
0040140E	. 0FBEC0	MOVSX EAX,AL	C 0 ES 002B 32bit 0(FFFFFFFF)
00401411	. 8985 70D0FFF	MOV DWORD PTR [EBP-290],EAX	P 1 CS 0023 32bit 0(FFFFFFFF)
00401417	. 0FB685 2AFFF	MOVZX EAX,BYTE PTR [EBP-D6]	A 0 SS 002B 32bit 0(FFFFFFFF)
0040141E	. 34 12	XOR AL,12	Z 1 DS 002B 32bit 0(FFFFFFFF)
00401420	. 0FBEC0	MOVSX EAX,AL	S 0 FS 0053 32bit 2B2000(FFF)
00401423	. 8985 6CFDFFF	MOV DWORD PTR [EBP-294],EAX	T 0 GS 002B 32bit 0(FFFFFFFF)
00401429	. 0FB685 2BFFF	MOVSX EAX,BYTE PTR [EBP-D5]	D 0
00401430	. 35 FE000000	XOR EAX,0FE	O 0 LastErr ERROR_FILE_NOT_FOUND (00000002)
00401435	. 8985 68FDFFF	MOV DWORD PTR [EBP-298],EAX	EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
00401438	. 0FB685 2CFFF	MOVSX EAX,BYTE PTR [EBP-D4]	ST0 empty 0.0
00401442	. 35 DB000000	XOR EAX,0DB	ST1 empty 0.0
00401447	. 8985 64FDFFF	MOV DWORD PTR [EBP-29C],EAX	ST2 empty 0.0
0040144D	. 0FB685 2DFFF	MOVZX EAX,BYTE PTR [EBP-D3]	ST3 empty 0.0
00401454	. 34 78	XOR AL,78	
00401456	. 0FBEC0	MOVSX EAX,AL	

Đoạn lệnh trên đã biến đổi password của ta, cùng F8 ở màn hình CPU đồng thời theo dõi màn hình các thanh ghi.

Địa chỉ	Câu lệnh	Ý nghĩa
004013F3	MOVZX EAX, BYTE PTR [EBP-D8]	Lưu trữ ký tự đầu tiên trong password vào thanh ghi EAX
004013FA	XOR AL, 34	Sau đó đem xor với 0x34
00401405	MOVZX EAX, BYTE PTR [EBP-D7]	Lưu trữ ký tự thứ hai trong password vào thanh ghi EAX
0040140C	XOR AL, 78	Sau đó đem xor với 0x78
00401417	MOVZX EAX, BYTE PTR [EBP-D6]	Lưu trữ ký tự thứ ba trong password vào thanh ghi EAX
0040141E	XOR AL, 12	Sau đó đem xor với 0x12
00401429	MOVSX EAX, BYTE PTR [EBP-D5]	Lưu trữ ký tự thứ tư trong password vào thanh ghi EAX
00401430	XOR EAX, 0FE	Sau đó đem xor với 0xFE
0040143B	MOVSX EAX, BYTE PTR [EBP-D4]	Lưu trữ ký tự thứ năm trong password vào thanh ghi EAX
00401442	XOR EAX, 0DB	Sau đó đem xor với 0xDB
0040144D	MOVZX EAX, BYTE PTR [EBP-D3]	Lưu trữ ký tự cuối cùng trong password vào thanh ghi EAX
00401454	XOR AL, 78	Sau đó đem xor với 0x78

Thử với password vừa nhập:

Ký tự	<i>s</i>	<i>k</i>	<i>y</i>	<i>m</i>	<i>t</i>	<i>p</i>
Xor	34	78	12	FE	DB	78
Kết quả	47	13	6B	93	AF	08

004014A3	. 8085 58FEFF	LEA EAX,DWORD PTR [EBP-1A8]	
004014A9	. 890424	MOV DWORD PTR [ESP],EAX	
004014AC	. E8 2F060000	CALL <JMP.&USER32.wsprintfA>	wsprintfA
004014B1	> 8085 88FDFF	LEA EAX,DWORD PTR [EBP-278]	
004014B7	. 8095 58FEFF	LEA EDX,DWORD PTR [EBP-1A8]	
004014BD	. 894424 04	MOV DWORD PTR [ESP+4],EAX	
004014C1	. 891424	MOV DWORD PTR [ESP],EDX	
004014C4	. E8 87050000	CALL <JMP.&msvcrt.strcmp>	strcmp
004014C9	. 85C0	TEST EAX,EAX	

F7 đến địa chỉ 004014B7, nhìn sang màn hình thanh ghi EAX đang chứa giá trị “4D11628EBE1D”

```

EAX 0060FCB0 ASCII "4D11628EBE1D"
ECX A0352814
EDX 0060FD8A
EBX 00004000
ESP 0060FC40
EBP 0060FF28
ESI 00401220 1_1.<ModuleEntryPoint>
EDI 00401220 1_1.<ModuleEntryPoint>
EIP 004014B7 1_1.004014B7

```

Đến địa chỉ tiếp theo thanh ghi EDX thì chứa giá trị “47136B93AF8”

```

Registers (FPU) <
EAX 0060FCB0 ASCII "4D11628EBE1D"
ECX A0352814
EDX 0060FD80 ASCII "47136B93AF8"
EBX 00004000
ESP 0060FC40
EBP 0060FF28
ESI 00401220 1_1.<ModuleEntryPoint>
EDI 00401220 1_1.<ModuleEntryPoint>
EIP 004014BD 1_1.004014BD

```

Nhận ra giá trị tại thanh ghi EDX chính là password mà đã được mã hóa ở trên.

Sau đó chương trình so sánh 2 thanh ghi này với nhau và nhảy về badboy nếu không bằng nhau.

004014CB	. 75 0E	JNZ SHORT 1_1.004014DB	
004014CD	. C70424 5C324	MOV DWORD PTR [ESP],1_1.0040325C	ASCII "That's right! Now write a small tut :)"
004014D4	. E8 A7050000	CALL <JMP.&msvcrt.printf>	printf
004014D9	. EB 0C	JMP SHORT 1_1.004014E7	
004014DB	> C70424 87324	MOV DWORD PTR [ESP],1_1.00403287	ASCII "Nope... try again."
004014E2	. E8 99050000	CALL <JMP.&msvcrt.printf>	printf
004014E7	. E8 04040000	CALL <JMP.&msvcrt._getch>	_getch
004014EC	. B8 00000000	MOV EAX,0	
004014F1	. C9	LEAVE	
004014F3	. C3	RET	

Vậy kết luận được password đúng là nằm ở thanh ghi EDX và đã được mã hóa. Việc cần làm bây giờ là mã hóa ngược lại giá trị tại thanh ghi EDX.

Ta có:

$$a \text{ XOR } b = c \text{ thì } c \text{ XOR } b = a$$

EDX	4D	11	62	8E	BE	1D
Xor	34	78	12	FE	DB	78
Kết quả	79	69	70	70	65	65
Ký tự	<i>y</i>	<i>i</i>	<i>p</i>	<i>p</i>	<i>e</i>	<i>e</i>

Vậy password là yippee. Ta thử lại với chương trình.

```

      8      .oPYo. 8
      8      8 .o8 8
8oPYo. oPYo. 8 .P'8 8 .o .oPYo. odYo.
      8 8 8 ` ' 8.d' 8 8oP' 8oooo8 8' `8
      8 8 8 8o' 8 8 `b. 8. 8 8
`YooP' 8 `YooP' 8 `o. `Yooo' 8 8
:.....:
:.....:
:.....:
CrackMe #4 (BruteforceMe)

Password : yippee

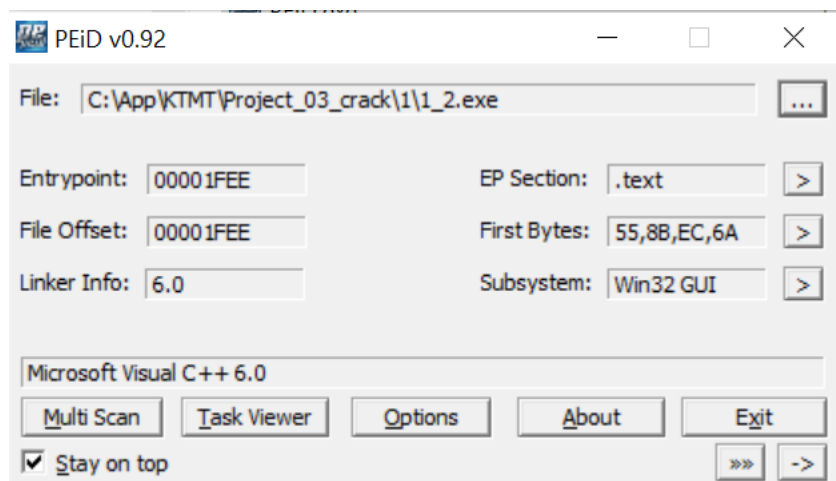
That's right! Now write a small tut :)

```

Thành công!!

2. Chương trình 1_2

Kiểm tra chương trình với PEiD



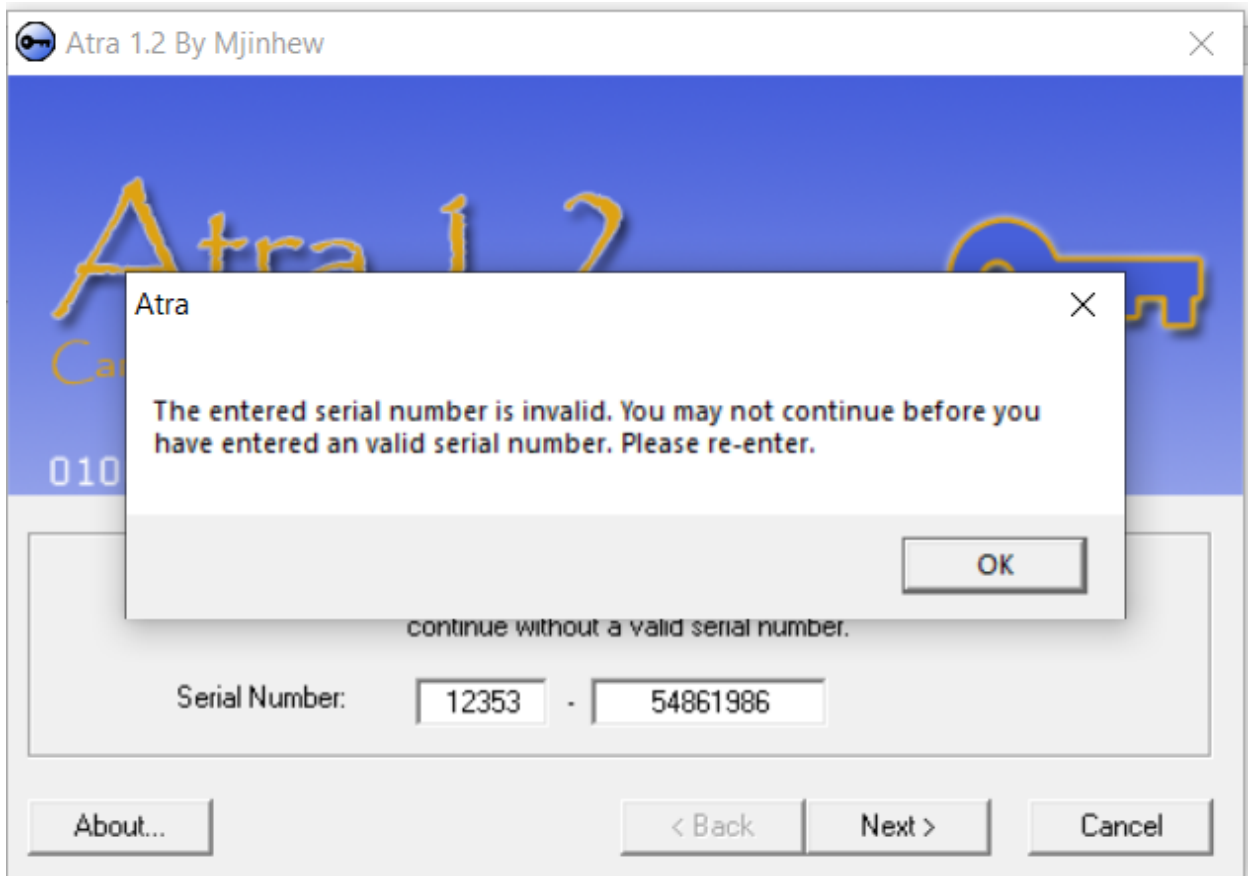
→ Được viết bằng ngôn ngữ C++

a. Manh mối

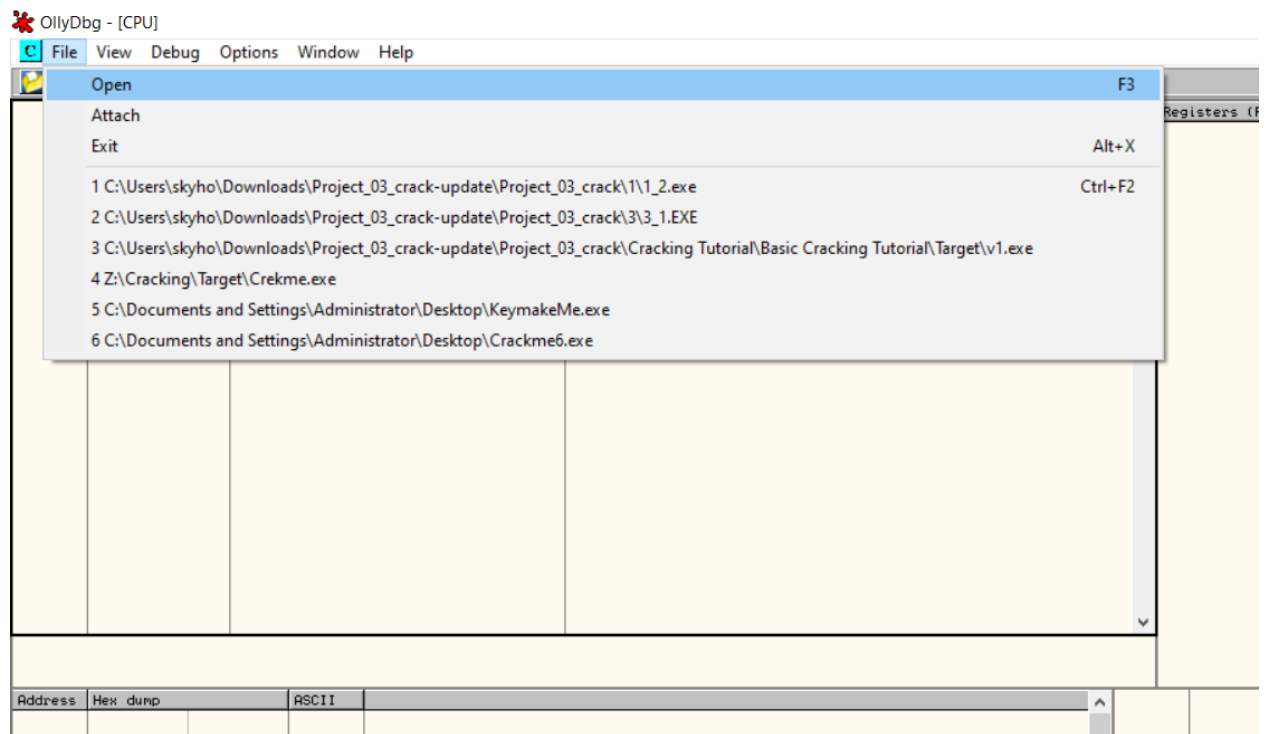
Đầu tiên, mở crackme lên sau đó điền một fake serial vào. Ta tìm được badboy là *“The entered serial number is invalid. You may not continue before you have entered an valid serial number. Please re-enter.”*

Ta để ý thấy được Serial Number được chia làm 2 phần:

- Phần 1: 5 kí tự
- Phần 2: 8 kí tự

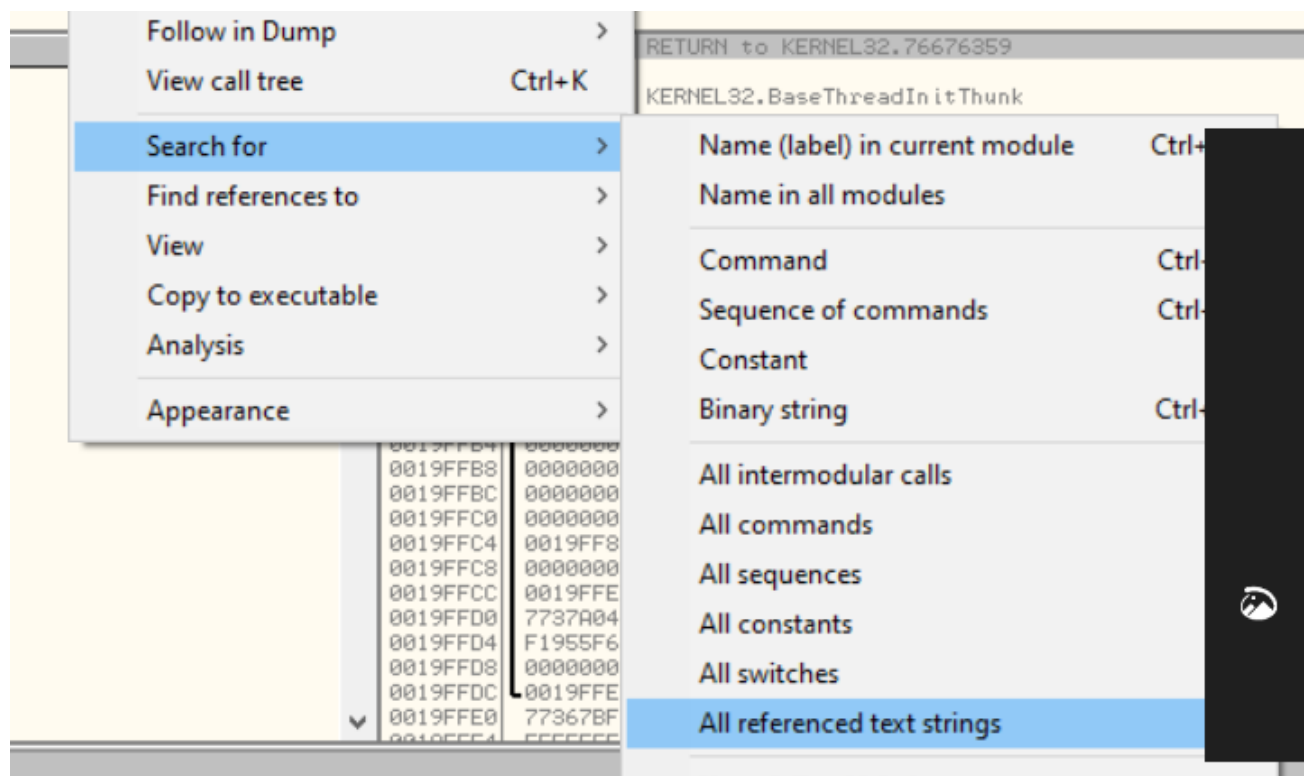


Sau đó, khởi động OllyDbg.exe được đính kèm trong Project_03_crack, nhấn F3 (hoặc File → Open) để mở chương trình 1_2.exe



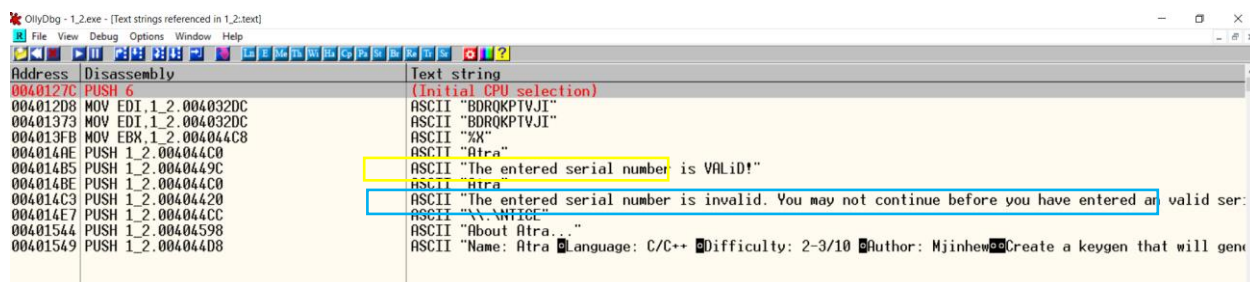
Bắt đầu quá trình tìm kiếm Key thôi nào!!!!

Đầu tiên, nhấp chuột phải vào phần trong CPU → Search for → All referenced text strings.
Mục đích tìm được badboy và goodboy từ đó định hướng tìm kiếm Key.



Sau đó sẽ xuất hiện ra một hộp thoại có tên [OllyDbg – 1_2.exe – [Text strings referenced in 1_2::text]], chứa toàn bộ các dòng code có chứa nội dung là toàn bộ text string trong 1_2.exe. Ta thấy được:

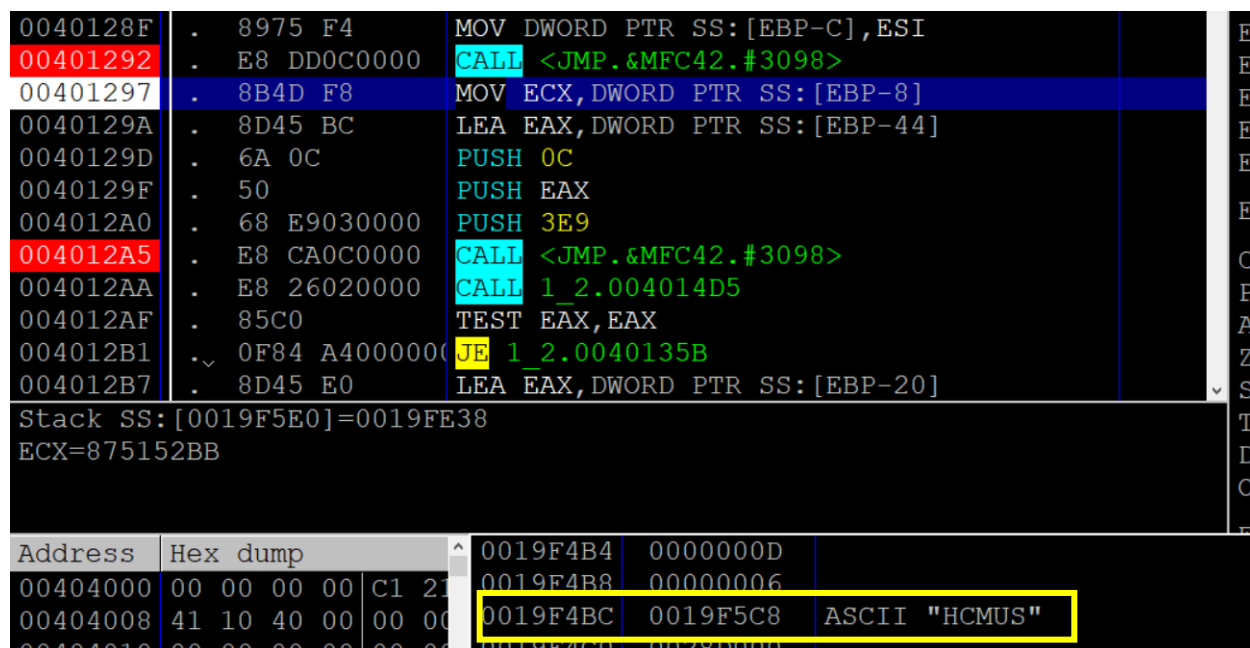
- **Badboy** : *“The entered serial number is invalid. You may not continue before you have entered an valid serial number. Please re-enter”*
- **Goodboy**: *“The entered serial numver is VALiD!”*



Nhấp đúp chuột và dòng lệnh có chữ goodboy, ta sẽ được như hình dưới. Quan sát các câu lệnh xung quanh goodboy và badboy trong hình ta dễ dàng thấy được có một lệnh JNZ SHORT 1_2.004014C3 nằm trên dòng lệnh chứa goodboy.

Ta có thể đặt giả thuyết 1_2.004014C3 là một hàm, và thử tìm xem nó có chứa Key hay không. Và theo thực tế là KHÔNG. Vì thế việc tiếp theo chúng ta phải lần theo chuỗi Serial Number của users nhập vào.

Thử đặt breakpoint tại các vị trí có hàm xử lí lấy chuỗi ta thu được địa chỉ lưu chuỗi Serial Number Input.



Part 1 được lấy vào tại dòng 00401292 với địa chỉ không cố định

0040129F	. 50	PUSH EAX	
004012A0	. 68 E9030000	PUSH 3E9	
004012A5	. E8 CA0C0000	CALL <JMP.&MFC42.#3098>	
004012AA	. E8 26020000	CALL 1_2.004014D5	
004012AF	. 85C0	TEST EAX,EAX	
004012B1	. 0F84 A4000000	JE 1_2.0040135B	
004012B7	. 8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
004014D5=1_2.004014D5			

Address	Hex dump		
00404000	00 00 00 00 C1 21	0019F57C	0000000C
00404008	41 10 40 00 00 00	0019F580	0019FE38
00404010	00 00 00 00 00 00	0019F584	0019F5E8
00404018	00 00 00 00 00 00	0019F588	004012AA
00404020	00 00 00 00 96 30	0019F58C	000003E9
		0019F590	0019F5A4
			ASCII "CQ18CTT1"

Part 2 được lấy vào tại dòng **004012A5** với địa chỉ không cố định

0040128C	MOV DWORD PTR SS:[EBP-14],ESI	
0040128F	MOV DWORD PTR SS:[EBP-C],ESI	
00401292	CALL <JMP.&MFC42.#3098>	1
00401297	MOV ECX,DWORD PTR SS:[EBP-8]	
0040129A	LEA EAX,DWORD PTR SS:[EBP-44]	
0040129D	PUSH 0C	
0040129F	PUSH EAX	
004012A0	PUSH 3E9	
004012A5	CALL <JMP.&MFC42.#3098>	2
004012AA	CALL 1_2.004014D5	
004012AF	TEST EAX,EAX	
004012B1	JE 1_2.0040135B	
004012B7	LEA EAX,DWORD PTR SS:[EBP-20]	

Tổng quát lại tại dòng 1 lưu địa chỉ phần 1, dòng 2 lưu địa chỉ phần 2

Tiến hành truy vết Key từ badboy. Ta dễ dàng thấy được badboy được nhảy từ các địa chỉ **00401325, 0040136A, 004013A8, 004013D1, 004013DB** tức ta có thể khi xét KEY không thỏa sẽ nhảy đến xuất ra **badboy**

004014BC	> 6A 00	PUSH 0	
004014BE	. 68 C0444000	PUSH 1_2.004044C0	ASCII "Atra"
004014C3	> 68 20444000	PUSH 1_2.00404420	ASCII "The entered serial"
004014C8	> 8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]	
004014CB	. E8 9E0A0000	CALL <JMP.&MFC42.#4224>	
004014D0	> 5F	POP EDI	
Jumps from 00401325, 0040136A, 004013A8, 004013D1, 004013DB			

Ta lần lượt xét các lệnh nhảy đó thuộc hàm nào? và Điều kiện của serial number là gì? Đầu tiên xét **00401325**, đây là đoạn code kiểm tra nếu độ dài chuỗi serial_part_1 = 0 thì báo lỗi.

```

004012F5 > 57          PUSH EDI
004012F6 . 33F6        XOR ESI,ESI
004012F8 . E8 DF0C0000 CALL <JMP.&MSVCRT.strlen>
004012FD . 85C0        TEST EAX,EAX
004012FF . 59          POP ECX
00401300 . 76 35       JBE SHORT 1_2.00401337
00401302 > 8A03        MOV AL,BYTE PTR DS:[EBX]
00401304 . 3A86 DC324000 CMP AL,BYTE PTR DS:[ESI+40]
0040130A . 75 05       JNZ SHORT 1_2.00401311
0040130C . FF45 F0     INC DWORD PTR SS:[EBP-10]
0040130F . EB 1A       JMP SHORT 1_2.0040132B
00401311 > 3A86 D0324000 CMP AL,BYTE PTR DS:[ESI+40]
00401317 . 75 05       JNZ SHORT 1_2.0040131E
00401319 . FF45 EC     INC DWORD PTR SS:[EBP-14]
0040131C . EB 0D       JMP SHORT 1_2.0040132B
0040131E > FF45 F4     INC DWORD PTR SS:[EBP-C]
00401321 . 837D F4 2F  CMP DWORD PTR SS:[EBP-C],2
00401325 . 0F87 91010000 JA 1_2.004014BC
00401328 > 57          PUSH EDI

```

Ta xét đến **004013D1** và **004013DB**, thì có được dòng code này:

```

004013A1 > FF45 F4     INC DWORD PTR SS:[EBP-C]
004013A4 . 837D F4 2F  CMP DWORD PTR SS:[EBP-C],2F
004013A8 . 0F87 0E010000 JA 1_2.004014BC
004013AE > 57          PUSH EDI
004013AF . 46          INC ESI
004013B0 . E8 270C0000 CALL <JMP.&MSVCRT.strlen>
004013B5 . 3BF0        CMP ESI,EAX
004013B7 . 59          POP ECX
004013B8 . 72 CB       JB SHORT 1_2.00401385
004013BA > FF45 FC     INC DWORD PTR SS:[EBP-4]
004013BD . 8D45 E0     LEA EAX,DWORD PTR SS:[EBP-20]
004013C0 . 50          PUSH EAX
004013C1 . 43          INC EBX
004013C2 . E8 150C0000 CALL <JMP.&MSVCRT.strlen>
004013C7 . 3945 FC     CMP DWORD PTR SS:[EBP-4],EAX
004013CA . 59          POP ECX
004013CB . 72 AB       JB SHORT 1_2.00401378
004013CD . 837D F0 03  CMP DWORD PTR SS:[EBP-10],3
004013D1 . 0F85 E5000000 JNZ 1_2.004014BC
004013D7 . 837D EC 02  CMP DWORD PTR SS:[EBP-14],2
004013DB . 0F85 DB000000 JNZ 1_2.004014BC

```

Đoạn code này có tác dụng kiểm tra xem [EBP-10] có bằng 3 không, với [EBP-10] là số ký tự của serial_part_1 mà xuất hiện trong chuỗi “BDRQKPTVJI”. Kiểm tra tương tự tại lệnh **004013DB** ta thấy đoạn code ở đây so sánh [EBP-14] với 2. với [EBP-14] là số ký tự của part 1 mà xuất hiện trong chuỗi “0123456789”.

Giải thích code:

0040135B | LEA EAX,DWORD PTR SS:[EBP-20]

0040135E	MOV DWORD PTR SS:[EBP-4],ESI	
00401361	PUSH EAX	// serial_part_1
00401362	CALL <JMP.&MSVCRT.strlen>	// Độ dài serial_part_1
00401367	TEST EAX,EAX	
00401369	POP ECX	
0040136A	JBE Atra.004014BC	
00401370	LEA EBX,DWORD PTR SS:[EBP-20]	
00401373	MOV EDI,Atra.004032DC	// ASCII "BDRQKPTVJI"
00401378	PUSH EDI	// "BDRQKPTVJI"
00401379	XOR ESI,ESI	// ESI=0
0040137B	CALL <JMP.&MSVCRT.strlen>	// Độ dài chuỗi "BDRQKPTVJI"
00401380	TEST EAX,EAX	
00401382	POP ECX	
00401383	JBE SHORT Atra.004013BA	
00401385	MOV AL,BYTE PTR DS:[EBX]	// [4032DC] = "BDRQKPTVJI"
00401387	CMP AL,BYTE PTR DS:[ESI+4032DC]	// serial_part_1[i] =[j+4032DC]
0040138D	JNZ SHORT Atra.00401394	// alpha++
0040138F	INC DWORD PTR SS:[EBP-10]	
00401392	JMP SHORT Atra.004013AE	// [4032D0] = "0123456789"
00401394	CMP AL,BYTE PTR DS:[ESI+4032D0]	// serial_part_1[i] =[j+4032DC]
0040139A	JNZ SHORT Atra.004013A1	// number++
0040139C	INC DWORD PTR SS:[EBP-14]	
0040139F	JMP SHORT Atra.004013AE	
004013A1	INC DWORD PTR SS:[EBP-C]	
004013A4	CMP DWORD PTR SS:[EBP-C],2F	
004013A8	JA Atra.004014BC	
004013AE	PUSH EDI	// "BDRQKPTVJI"
004013AF	INC ESI	// ESI++
004013B0	CALL <JMP.&MSVCRT.strlen>	// Độ dài chuỗi "BDRQKPTVJI"
004013B5	CMP ESI,EAX	// Nếu j < length("BDRQKPTVJI")
004013B7	POP ECX	// Tiếp tục vòng lặp
004013B8	JB SHORT Atra.00401385	
004013BA	INC DWORD PTR SS:[EBP-4]	// i++
004013BD	LEA EAX,DWORD PTR SS:[EBP-20]	
004013C0	PUSH EAX	// serial_part_1
004013C1	INC EBX	


```

004013C2 | CALL <JMP.&MSVCRT.strlen>           // length(serial_part_1)
004013C7 | CMP DWORD PTR SS:[EBP-4],EAX         // Nếu i < length(serial_part_1)
004013CA | POP ECX                             // Tiếp tục vòng lặp
004013CB | JB SHORT Atra.00401378
004013CD | CMP DWORD PTR SS:[EBP-10],3          // Nếu alpha = 3 thì tiếp tục
004013D1 | JNZ Atra.004014BC
004013D7 | CMP DWORD PTR SS:[EBP-14],2          // Nếu number = 2 thì tiếp tục
004013DB | JNZ Atra.004014BC

```

Ta có thể nhận xét về điều kiện để được chấp nhận của part 1:

Part 1 có tổng cộng 5 ký tự.
 Trong đó 3 ký tự phải nằm trong chuỗi **“BDRQKPTVJI”**. 2 ký tự phải nằm trong chuỗi **“0123456789”**.

- Part 1 có tổng cộng 5 ký tự
- Trong đó 3 ký tự phải nằm trong chuỗi **“BDRQKPTVJI”**. 2 ký tự phải nằm trong chuỗi **“0123456789”**.

Vậy nếu không đến Badboy thì chương trình sẽ làm gì tiếp theo???

Ta có thể dưới lệnh nhảy đến badboy còn rất nhiều câu lệnh, vậy nếu không nhảy đến badboy thì chương trình sẽ tiếp tục xử lí serial_part1.

004013E1	. 8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
004013E4	. 50	PUSH EAX	
004013E5	. E8 F20B0000	CALL <JMP.&MSVCRT.strlen>	strlen
004013EA	. 50	PUSH EAX	
004013EB	. 8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
004013EE	. 50	PUSH EAX	
004013EF	. E8 6A090000	CALL 1_2.00401D5E	

Giải thích code:

```

004013E1 | LEA EAX,DWORD PTR SS:[EBP-20]
004013E4 | PUSH EAX                               // serial_part_1
004013E5 | CALL <JMP.&MSVCRT.strlen>             // length(serial_part_1)
004013EA | PUSH EAX
004013EB | LEA EAX,DWORD PTR SS:[EBP-20]
004013EE | PUSH EAX
004013EF | CALL Atra.00401D5E                    // Gọi thuật toán Hash

```

Đoạn code từ dòng **00401D5E** thực chất là hash chuỗi part 1 với thuật toán phức tạp và kết thúc tại dòng **00401D99**. Kết quả thu được là một số sau khi xor output của thuật toán hash với nhau.

00401D5E	\$ 55	PUSH EBP
00401D5F	. 8BEC	MOV EBP,ESP
00401D61	. 83EC 68	SUB ESP,68
00401D64	. 8D45 98	LEA EAX,DWORD PTR SS:[EBP-68]
00401D67	. 50	PUSH EAX
00401D68	. E8 3AF8FFFF	CALL 1_2.004015A7
00401D6D	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00401D70	. 8D45 98	LEA EAX,DWORD PTR SS:[EBP-68]
00401D73	. FF75 08	PUSH DWORD PTR SS:[EBP+8]
00401D76	. 50	PUSH EAX
00401D77	. E8 53F8FFFF	CALL 1_2.004015CF
00401D7C	. 8D45 F0	LEA EAX,DWORD PTR SS:[EBP-10]
00401D7F	. 50	PUSH EAX
00401D80	. 8D45 98	LEA EAX,DWORD PTR SS:[EBP-68]
00401D83	. 50	PUSH EAX
00401D84	. E8 2CFFFFFF	CALL 1_2.00401CB5
00401D89	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
00401D8C	. 83C4 18	ADD ESP,18
00401D8F	. 3345 F8	XOR EAX,DWORD PTR SS:[EBP-8]
00401D92	. 3345 F4	XOR EAX,DWORD PTR SS:[EBP-C]
00401D95	. 3345 F0	XOR EAX,DWORD PTR SS:[EBP-10]
00401D98	. C9	LEAVE
00401D99	. C3	RETN

Tiếp theo giải thích tiếp đoạn code tiếp theo coi serial_part_1 bị biến đổi như thế nào

```

004013F4 | MOV EDI,DWORD PTR DS:[<&MSVCRT.sprintf>]    //msvcrt.sprintf
004013FA | PUSH EAX                                     ; /<%X>
004013FB | BB C8444000      MOV EBX,Atra.004044C8      ; |ASCII "%X"
00401400 | LEA EAX,DWORD PTR SS:[EBP-2C]
00401403 | PUSH EBX                                     // format => "%X"
00401404 | PUSH EAX                                     //s
00401405 | CALL EDI      // Định dạng số nguyên hệ thập lục phân không dấu (các chữ cái
hoa)
00401407 | AND DWORD PTR SS:[EBP-4],0
0040140B | LEA EAX,DWORD PTR SS:[EBP-2C]
0040140E | PUSH EAX                                     // serial_part_1
0040140F | CALL <JMP.&MSVCRT.strlen>                   // length(serial_part_1)
00401414 | ADD ESP,1C
00401417 | TEST EAX,EAX
00401419 | JBE SHORT Atra.00401444
0040141B | MOV EAX,DWORD PTR SS:[EBP-4]
0040141E | PUSH EAX
0040141F | LEA ESI,DWORD PTR SS:[EBP+EAX-2C]
00401423 | MOVSX EAX,BYTE PTR SS:[EBP+EAX-2C]

```

```

00401428 | PUSH EAX
00401429 | CALL Atra.00401247      // Table = (A6H, 16H, AFH, FDH, D4H, 07H, 10H,
F6H)
0040142E | INC DWORD PTR SS:[EBP-4]
00401431 | MOV BYTE PTR DS:[ESI],AL      // S1[i] = S1[i] xor Table1[j]
00401433 | LEA EAX,DWORD PTR SS:[EBP-2C]
00401436 | PUSH EAX                    // serial_part_1
00401437 | CALL <JMP.&MSVCRT.strlen>      // serial_part_1
0040143C | ADD ESP,0C
0040143F | CMP DWORD PTR SS:[EBP-4],EAX  // Nếu i < length(serial_part_1)
00401442 | JB SHORT Atra.0040141B        // Tiếp tục lặp
00401444 | AND DWORD PTR SS:[EBP-4],0
00401448 | LEA EAX,DWORD PTR SS:[EBP-2C]
0040144B | PUSH EAX                    // serial_part_1
0040144C | CALL <JMP.&MSVCRT.strlen>      // serial_part_1
00401451 | TEST EAX,EAX
00401453 | POP ECX
00401454 | JBE SHORT Atra.0040147F
00401456 | MOV EAX,DWORD PTR SS:[EBP-4]
00401459 | PUSH EAX
0040145A | LEA ESI,DWORD PTR SS:[EBP+EAX-2C]
0040145E | MOVSX EAX,BYTE PTR SS:[EBP+EAX-2C]
00401463 | PUSH EAX
00401464 | CALL Atra.00401261
00401469 | INC DWORD PTR SS:[EBP-4]
0040146C | MOV BYTE PTR DS:[ESI],AL      // serial_part_1[i] = (serial_part_1[i] shl
i) or serial_part_1[i]
0040146E | LEA EAX,DWORD PTR SS:[EBP-2C]
00401471 | PUSH EAX                    // serial_part_1
00401472 | CALL <JMP.&MSVCRT.strlen>      // length(serial_part_1)
00401477 | ADD ESP,0C
0040147A | CMP DWORD PTR SS:[EBP-4],EAX  // Nếu i < length(serial_part_1)
0040147D | JB SHORT Atra.00401456        // Tiếp tục lặp
0040147F | LEA EAX,DWORD PTR SS:[EBP-2C]
00401482 | PUSH EAX                    // serial_part_1
00401483 | CALL <JMP.&MSVCRT.strlen>      // length(serial_part_1)
00401488 | PUSH EAX
00401489 | LEA EAX,DWORD PTR SS:[EBP-2C]
0040148C | PUSH EAX
0040148D | CALL Atra.004010C0      // Đoạn code từ dòng 004010C0 thực chất là hash
chuỗi S phía trên với thuật toán CRC32.
00401492 | PUSH EAX

```

```

00401493 | LEA EAX,DWORD PTR SS:[EBP-38]
00401496 | PUSH EBX
00401497 | PUSH EAX
00401498 | CALL EDI                                     // serial_part_2 là tmp biểu diễn theo hệ
thập lục phân
0040149A | LEA EAX,DWORD PTR SS:[EBP-38]
0040149D | PUSH EAX                                     // serial_part_2
0040149E | LEA EAX,DWORD PTR SS:[EBP-44]
004014A1 | PUSH EAX                                     // serial_part_2 user nhập vào
004014A2 | CALL <JMP.&MSVCRT.strcmp>                   // So sánh 2 serial_part_2 với nhau
004014A7 | ADD ESP,20
004014AA | TEST EAX,EAX
004014AC | PUSH 0
004014AE | PUSH Atra.004044C0                           // ASCII "Atra"
004014B3 | JNZ SHORT Atra.004014C3
004014B5 | PUSH Atra.0040449C                           // ASCII "The entered serial number is
VALiD!"
004014BA | JMP SHORT Atra.004014C8
004014BC | PUSH 0
004014BE | PUSH Atra.004044C0                           // ASCII "Atra"
004014C3 | PUSH Atra.00404420                           // ASCII "The entered serial number is
invalid. You may not continue before you have entered an valid serial number. Please re-
enter."
004014C8 | MOV ECX,DWORD PTR SS:[EBP-8]
004014CB | CALL <JMP.&MFC42.#4224>

```

b. Đề xuất thuật toán keygen

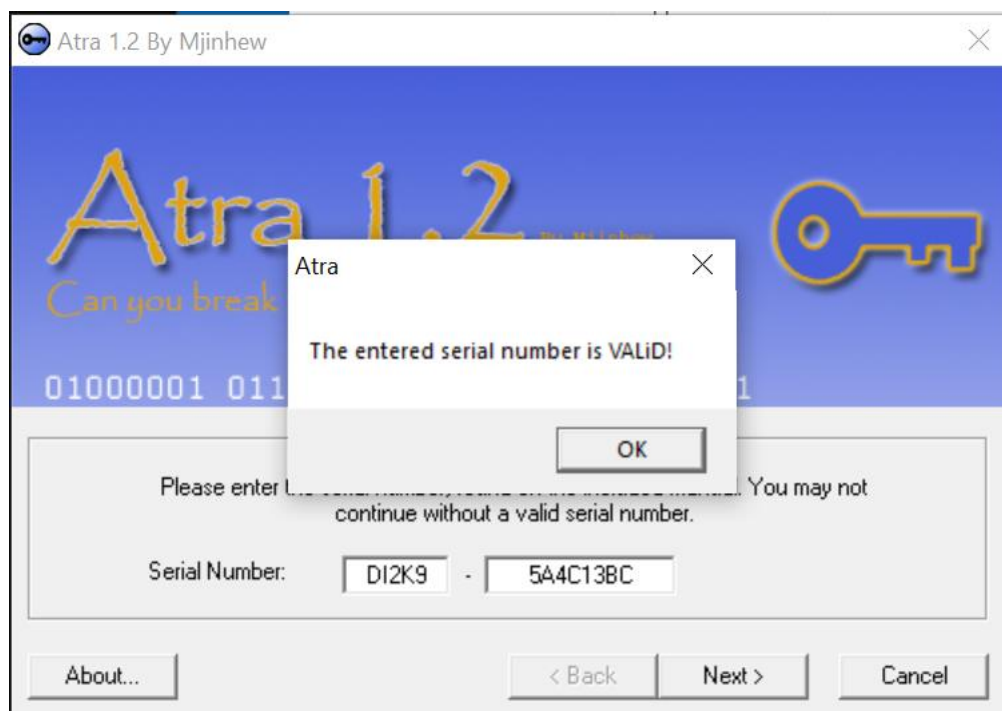
Đề xuất ý tưởng viết keygen.exe. Dựa trên thuật toán đề xuất, tiến hành viết keygen như sau:

- Yêu cầu nhập vào part 1 đúng như điều kiện của chương trình.
- Vì thuật toán hash lần 1 quá phức tạp, tiến hành convert từ vị từ asm (push, lea, mov, shl) thành toán tử có sẵn trong ngôn ngữ C. Chèn thẳng đoạn code hash part 1 đã được convert vào trong code C của chương trình.
- Các bước còn lại làm tương tự chương trình để sinh ra part 2 từ part 1.

Thử lại sau khi viết chương trình keygen:

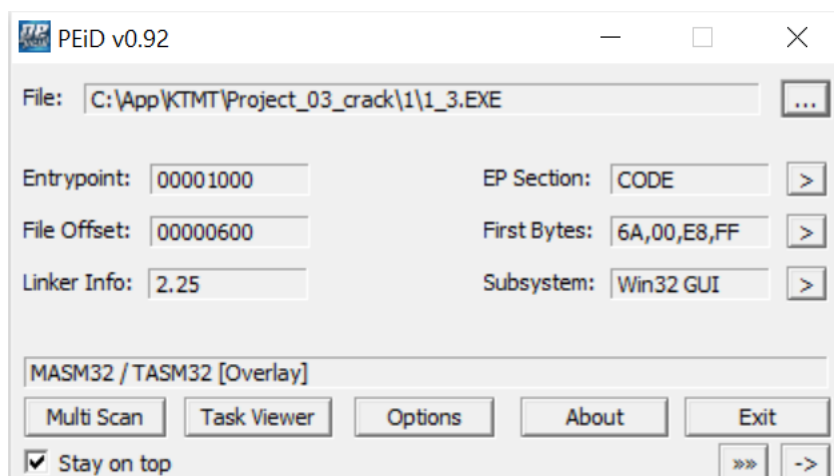
```
D:\Xuân Mai\HCMUS\4\KTMT\Đồ Án\Đồ án 3\1.3\KEYGEN_XUATEXE\KEYGEN.exe

Luu y: Phan 1 gom 5 ki tu va phai 3 ki tu thuoc chuoai BDRQKPTVJI va 2 ki tu so thuoc chuoai 0123456789
SERIAL PART 1: DI2K9
SERIAL PART 2: 5A4C13BC
Press any key to continue . . .
```



Thành công!

3. Chương trình 1_3



Kiểm tra bằng PEiD → Viết bằng ASM32.

Sau đó mở Olly Dbg để dò password.

a. Manh mối

Màn hình hiển thị sau khi mở file 1_3.exe

Address	Hex dump	Disassembly	Comment
00401327	. 5B	POP EBX	
00401328	. C9	LEAVE	
00401329	. C2 1000	RET 10	
0040132C	> 817D 10 F203	CMP DWORD PTR [EBP+10],3F2	
00401333	~ 75 11	JNZ SHORT 1_3.00401346	
00401335	> 6A 00	PUSH 0	Result = 0
00401337	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
0040133A	. E8 73010000	CALL <JMP.&USER32.EndDialog>	EndDialog
0040133F	. B8 01000000	MOV EAX,1	
00401344	^ EB DF	JMP SHORT 1_3.00401325	
00401346	> B8 00000000	MOV EAX,0	
0040134B	^ EB D8	JMP SHORT 1_3.00401325	
0040134D	~ 6A 30	PUSH 30	
0040134F	. 68 29214000	PUSH 1_3.00402129	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
00401354	. 68 34214000	PUSH 1_3.00402134	Title = "Good work!"
00401359	. FF75 08	PUSH DWORD PTR [EBP+8]	Text = "Great work! Now try the next CrackMe!"
0040135C	. E8 D9000000	CALL <JMP.&USER32.MessageBoxA>	hOwner
00401361	. C3	RET	MessageBoxA
00401362	~ 6A 00	PUSH 0	
00401364	. E8 AD000000	CALL <JMP.&USER32.MessageBeep>	BeepType = MB_OK
00401369	. 6A 30	PUSH 30	MessageBeep
0040136B	. 68 60214000	PUSH 1_3.00402160	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
00401370	. 68 69214000	PUSH 1_3.00402169	Title = "No luck!"
00401375	. FF75 08	PUSH DWORD PTR [EBP+8]	Text = "No luck there, mate!"
00401378	. E8 BD000000	CALL <JMP.&USER32.MessageBoxA>	hOwner
0040137D	. C3	RET	MessageBoxA
0040137E	~ 8B7424 04	MOV ESI,DWORD PTR [ESP+4]	
00401382	. 56	PUSH ESI	
00401383	. 8A06	MOV AL,BYTE PTR [ESI]	
00401385	. 84C0	TEST AL,AL	
00401387	~ 74 13	JE SHORT 1_3.0040139C	
00401389	. 3C 41	CMP AL,41	
0040138B	~ 72 1F	JB SHORT 1_3.004013AC	

Backup

Copy

Binary

Assemble

Label

Comment

Breakpoint

Hit trace

Run trace

New origin here

Go to

Thread

Follow in Dump

View call tree

Search for

Find references to

View

Copy to executable

Analysis

Appearance

Name (label) in current module

Name in all modules

Command

Sequence of commands

Constant

Binary string

All intermodular calls

All commands

All sequences

All constants

All switches

All referenced text strings

User-defined label

User-defined comment

EAX 0019FFCC

ECX 00401000 1_3.<ModuleEntry

EDX 00401000 1_3.<ModuleEntry

EBX 00225000

Address	Value	Comment
0019FF74	769A6359	RETURN to KERNEL32.769A
0019FF78	00225000	
0019FF7C	769A6340	KERNEL32.BaseThreadInit
0019FF80	0019FFDC	
0019FF84	776E7C24	RETURN to ntdll.776E7C2
0019FF88	00225000	
0019FF8C	9A8B0131	
0019FF90	00000000	

Chọn All referenced text strings để tìm ra tất cả các string được reference trong chương trình, kết quả thu được như sau:

Address	Disassembly	Text string
00401000	PUSH 0	(Initial CPU selection)
0040100E	PUSH 1_3.004020F4	ASCII "No need to disasm the code!"
00401077	MOV DWORD PTR [402084],1_3.00402110	ASCII "MENU"
00401081	MOV DWORD PTR [402088],1_3.004020F4	ASCII "No need to disasm the code!"
004010B7	PUSH 1_3.004020E7	ASCII "CrackMe v1.0"
004010BC	PUSH 1_3.004020F4	ASCII "No need to disasm the code!"
004011F7	PUSH 1_3.0040211F	ASCII "DLG_ABOUT"
00401213	PUSH 1_3.00402115	ASCII "DLG_REGIS"
0040134F	PUSH 1_3.00402129	ASCII "Good work!"
00401354	PUSH 1_3.00402134	ASCII "Great work, mate! Now try the next CrackMe!"
0040136B	PUSH 1_3.00402160	ASCII "No luck!"
00401370	PUSH 1_3.00402169	ASCII "No luck there, mate!"
004013AF	PUSH 1_3.00402160	ASCII "No luck!"
004013B4	PUSH 1_3.00402169	ASCII "No luck there, mate!"

Để thấy chương trình tồn tại 1 goodboy và 2 badboy.

Goodboy in ra màn hình **"Great work, mate! Now try the next CrackMe"**.

Badboy in ra màn hình **"No luck there, mate!"**.

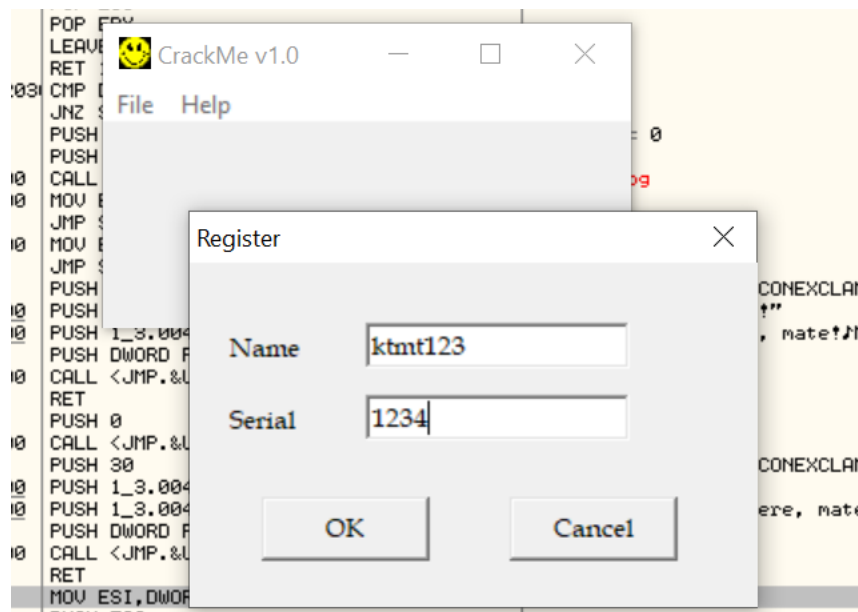
0040137E	8B7424 04	MOV ESI,DWORD PTR [ESP+4]	
00401382	56	PUSH ESI	
00401383	8A06	MOV AL,BYTE PTR [ESI]	
00401385	84C0	TEST AL,AL	
00401387	74 13	JE SHORT 1_3.0040139C	
00401389	3C 41	CMP AL,41	
0040138B	72 1F	JB SHORT 1_3.004013AC	
0040138D	3C 5A	CMP AL,5A	
0040138F	73 03	JNB SHORT 1_3.00401394	
00401391	46	INC ESI	
00401392	EB EF	JMP SHORT 1_3.00401383	
00401394	E8 39000000	CALL 1_3.004013D2	
00401399	46	INC ESI	
0040139A	EB E7	JMP SHORT 1_3.00401383	
0040139C	5E	POP ESI	
0040139D	E8 20000000	CALL 1_3.004013C2	
004013A2	81F7 78560000	XOR EDI,5678	
004013A8	8BC7	MOV EAX,EDI	
004013AA	EB 15	JMP SHORT 1_3.004013C1	
004013AC	5E	POP ESI	
004013AD	6A 30	PUSH 30	
004013AF	68 60214000	PUSH 1_3.00402160	
004013B4	68 69214000	PUSH 1_3.00402169	
004013B9	FF75 08	PUSH DWORD PTR [EBP+8]	
004013BC	E8 79000000	CALL <JMP.&USER32.MessageBoxA>	Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL Title = "No luck!" Text = "No luck there, mate!" hOwner MessageBoxA
004013C1	C3	RET	

Xét 1 badboy riêng biệt, thấy nó nằm ở cuối 1 hàm có địa chỉ 0040137E.

0040137E	8B7424 04	MOV ESI,DWORD PTR [ESP+4]	
00401382	56	PUSH ESI	
00401383	8A06	MOV AL,BYTE PTR [ESI]	
00401385	84C0	TEST AL,AL	
00401387	74 13	JE SHORT 1_3.0040139C	
Local call from 0040122D			

Hàm này lại được gọi từ 0040122D.

Ta đặt breakpoint tại vị trí 0040137E.



Nhập thử 1 name và serial bất kỳ như hình trên.

00401228	. 68 8E214000	PUSH 1_3.0040218E	ASCII "ktmt123"
0040122D	. E8 4C010000	CALL 1_3.0040137E	
00401232	. 50	PUSH EAX	
00401233	. 68 7E214000	PUSH 1_3.0040217E	ASCII "1234"
00401238	. E8 9B010000	CALL 1_3.004013D8	
0040123D	. 8B04 04	MOV ESP, 4	

Quay trở lại địa chỉ 0040122D, ta thấy name được lưu ở 0040218E và serial được lưu ở 0040217E.

Sau khi push name vào, chương trình liên gọi hàm tại vị trí 0040137E.

Vậy badboy đầu tiên dùng vào việc check name (hàm ở vị trí 0040137E).

Ta sẽ xét cụ thể chi tiết hàm tại địa chỉ 0040137E.

0040137E	. \$ 8B7424 04	MOV ESI, DWORD PTR [ESP+4]	1_3.0040218E
00401382	. 56	PUSH ESI	
00401383	> 8A06	MOV AL, BYTE PTR [ESI]	
00401385	. 84C0	TEST AL, AL	
00401387	~ 74 13	JE SHORT 1_3.0040139C	
00401389	. 3C 41	CMP AL, 41	
0040138B	~ 72 1F	JB SHORT 1_3.004013AC	
0040138D	. 3C 5A	CMP AL, 5A	
0040138F	~ 73 03	JNB SHORT 1_3.00401394	
00401391	. 46	INC ESI	
00401392	^ EB EF	JMP SHORT 1_3.00401383	
00401394	> E8 39000000	CALL 1_3.004013D2	
00401399	. 46	INC ESI	
0040139A	^ EB E7	JMP SHORT 1_3.00401383	
0040139C	> 5E	POP ESI	
0040139D	. E8 20000000	CALL 1_3.004013C2	
004013A2	. 81F7 78560000	XOR EDI, 5678	
004013A8	. 8BC7	MOV EAX, EDI	
004013AA	~ EB 15	JMP SHORT 1_3.004013C1	
004013AC	> 5E	POP ESI	
004013AD	. 6A 30	PUSH 30	
004013AF	. 68 60214000	PUSH 1_3.00402160	
004013B4	. 68 69214000	PUSH 1_3.00402169	
004013B9	. FF75 08	PUSH DWORD PTR [EBP+8]	
004013BC	. E8 79000000	CALL <JMP.&USER32.MessageBoxA>	
004013C1	> C3	RET	

```

Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
Title = "No luck!"
Text = "No luck there, mate!"
hOwner
MessageBoxA

```

Xét hàm con tại vị trí 00401383 → 0040139A. Hàm này kiểm tra nếu không phải chữ cái ('a' → 'z') và ('A' → 'Z') thì in ra badboy.

Địa chỉ	Câu lệnh	Ý nghĩa
00401383	MOV AL, BYTE PTR [ESI]	Chuyển từng ký tự từ đầu đến cuối của name vào thanh ghi AL (8 byte thấp của thanh ghi EAX).
00401385	TEST AL, AL	Kiểm tra xem AL có bằng không không (kiểm tra kết thúc chuỗi).
00401387	JE SHORT 1_3.0040139C	Nếu đã xét đến ký tự kết thúc chuỗi thì nhảy đến vị trí 0040139C.
00401389	CMP AL, 41 JB SHORT 1_3.004013AC	So sánh ký tự đó với ký tự 'A' (ascii = 65 ₁₀ = 41 ₁₆). Nếu bé hơn thì nhảy đến vị trí 004013AC và in ra màn hình badboy.
0040138D 0040138F	CMP AL, 5A JNB SHORT 1_3.00401394	So sánh ký tự đó với ký tự 'Z' (ascii = 90 ₁₀ = 5A ₁₆). Nếu lớn hơn hoặc bằng thì nhảy đến vị trí 00401394, sau đó gọi hàm tại vị trí 004013D2, tại hàm này làm nhiệm vụ trừ đi cho 32 ₁₀ = 20 ₁₆ (nghĩa là chuyển từ ký tự in thường thành in hoa) sau đó tiếp tục tăng ký tự và nhảy ngược lại đầu hàm tại vị trí 00401383.
00401391 00401392	INC ESI JMP SHORT 1_3.00401383	Nếu như đã là chữ in hoa thì cứ tăng tiếp tục ký tự và nhảy ngược lại đầu hàm tại vị trí 00401383.

Sau khi check đúng name (không có ký tự đặc biệt) tiếp tục nhảy đến địa chỉ 004013C2. Lúc này ESI đang lưu name với các ký tự đã được upcase.

0040139C	> 5E	POP ESI
0040139D	. E8 20000000	CALL 1_3.004013C2

004013C2	33FF	XOR EDI,EDI
004013C4	. 33DB	XOR EBX,EBX
004013C6	> 8A1E	MOV BL,BYTE PTR [ESI]
004013C8	. 840B	TEST BL,BL
004013CA	~ 74 05	JE SHORT 1_3.004013D1
004013CC	. 03FB	ADD EDI,EBX
004013CE	. 46	INC ESI
004013CF	^ EB F5	JMP SHORT 1_3.004013C6
004013D1	> C3	RET

Tại hàm này:

Địa chỉ	Câu lệnh	Ý nghĩa
004013C2 004013C4	XOR EDI, EDI XOR EBX, EBX	EDI = 0 EBX = 0
004013C6	MOV BL, BYTE PTR [ESI]	Lưu từng ký tự của name
004013C8	TEST BL, BL JE SHORT 1_3.004013D1	Nếu xét đến ký tự kết thúc thì kết thúc hàm và nhảy về địa chỉ 004013A2
004013CC	ADD EDI, EBX	EDI = EDI + EBX
004013CE 004013CF	INC ESI JMP SHORT 1_3.004013C6	Tăng ký tự và nhảy trở về địa chỉ 004013C6

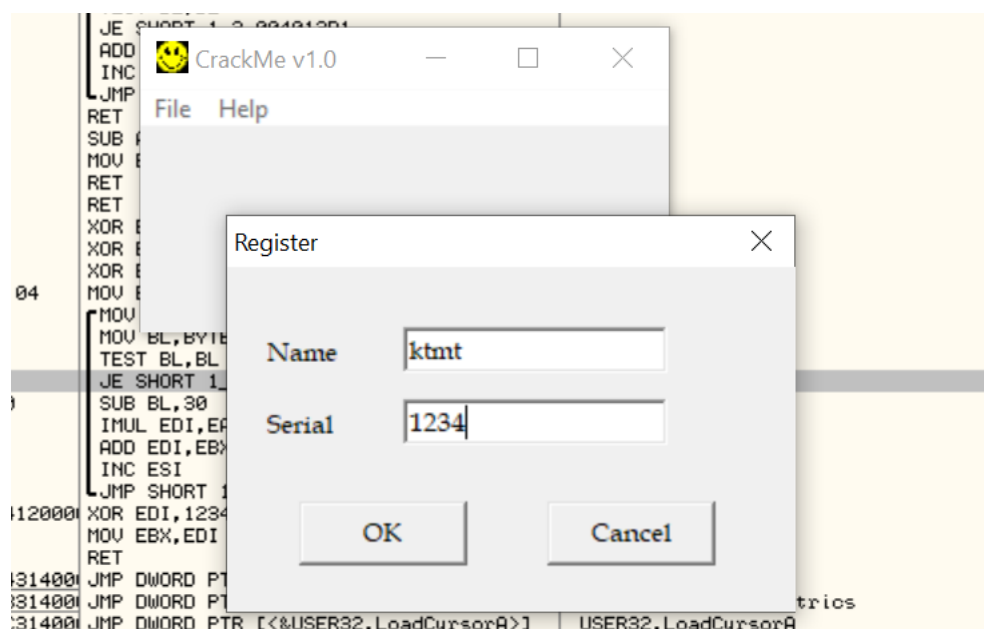
Sau khi kết thúc hàm và nhảy về địa chỉ 004013A2, lấy giá trị EDI xor với 0x5678, lưu giá trị tại EAX, nhảy về địa chỉ 0040122D.

F7 3 lần, tại dòng 00401233, chương trình push serial và gọi hàm đến địa chỉ 004013D8.

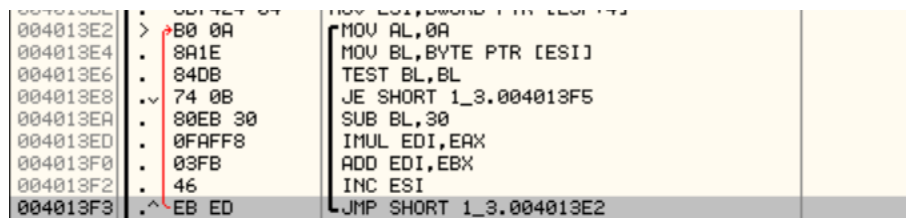
004013D8	33C0	XOR EAX,EAX
004013DA	. 33FF	XOR EDI,EDI
004013DC	. 33DB	XOR EBX,EBX
004013DE	. 8B7424 04	MOV ESI,DWORD PTR [ESP+4]
004013E2	> B0 0A	MOV AL,0A
004013E4	. 8A1E	MOV BL,BYTE PTR [ESI]
004013E6	. 84DB	TEST BL,BL
004013E8	~ 74 0B	JE SHORT 1_3.004013F5
004013EA	. 80EB 30	SUB BL,30
004013ED	. 0FAFF8	IMUL EDI,EAX
004013F0	. 03FB	ADD EDI,EBX
004013F2	. 46	INC ESI
004013F3	^ EB ED	JMP SHORT 1_3.004013E2
004013F5	> 81F7 34120000	XOR EDI,1234
004013FB	. 8BDF	MOV EBX,EDI
004013FD	. C3	RET

Đặt breakpoint tại địa chỉ này.

Ta nhập một name và serial khác với name phù hợp. Chương trình dừng lại tại breakpoint, ta debug bằng F7 theo từng dòng.



Nhảy vào hàm con tại địa chỉ 004013E2.



Địa chỉ	Câu lệnh	Ý nghĩa
004013D8 004013DA 004013DC	XOR EAX, EAX XOR EDI, EDI XOR EBX, EBX	Set EAX = 0 Set EDI = 0 Set EBX = 0
004013E2	MOV AL, 0A	AL = A (hệ 16)
004013E4	MOV BL, BYTE PTR [ESI]	BL (8 bits thấp của EBX) là 1 ký tự đang xét của chuỗi serial nhập vào.
004013E6 004013E8	TEST BL, BL JE SHORT 1_3.004013F5	Kiểm tra nếu đến cuối chuỗi rồi thì nhảy đến vị trí 004013F5 và kết thúc hàm.
004013EA	SUB BL, 30	Trừ BL cho $30_{16} = 48_{10}$
004013ED 004013F0	IMUL EDI, EAX ADD EDI, EBX	Nhân thanh ghi EDI cho EAX. Sau đó cộng với thanh ghi EBX.
004013F2 004013F3	INC ESI JMP SHORT 1_3.004013E2	Tăng ký tự thêm 1 và nhảy trở lại hàm tại địa chỉ 004013E2.

004013F5	>	81F7 3412000	XOR EDI,1234
004013FB	.	8BDF	MOV EBX,EDI
004013FD	.	C3	RET

Địa chỉ	Câu lệnh	Ý nghĩa
004013F5	XOR EDI, 1234	Sau khi xét hết ký tự, lấy EDI xor với 0x1234.
004013FB	MOV EBX, EDI	Di chuyển kết quả vừa xor được vào thanh ghi EBX và kết thúc hàm. Quay lại địa chỉ 0040123D.

Sau khi kết thúc hàm, quay trở lại địa chỉ 00401238.

00401241	.	3BC3	CMP EAX,EBX
00401243	~v	74 07	JE SHORT 1_3.0040124C
00401245	.	E8 18010000	CALL 1_3.00401362
0040124A	^	EB 9A	JMP SHORT 1_3.004011E6
0040124C	>	E8 FC000000	CALL 1_3.0040134D
00401251	^	EB 93	JMP SHORT 1_3.004011E6

So sánh 2 thanh ghi EAX và EBX. Nếu bằng nhau thì nhảy đến goodboy tại địa chỉ 0040134D

0040134D	\$. 6A 30	PUSH 30	{ Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL Title = "Good work!" Text = "Great work, mate!#Now try the next CrackMe!" hOwner MessageBoxA }
0040134F	. 68 29214000	PUSH 1_3.00402129	
00401354	. 68 34214000	PUSH 1_3.00402134	
00401359	. FF75 08	PUSH DWORD PTR [EBP+8]	
0040135C	. E8 D9000000	CALL <JMP.&USER32.MessageBoxA>	
00401361	. C3	RET	

Nếu khác nhau, nhảy đến badboy tại địa chỉ 00401362

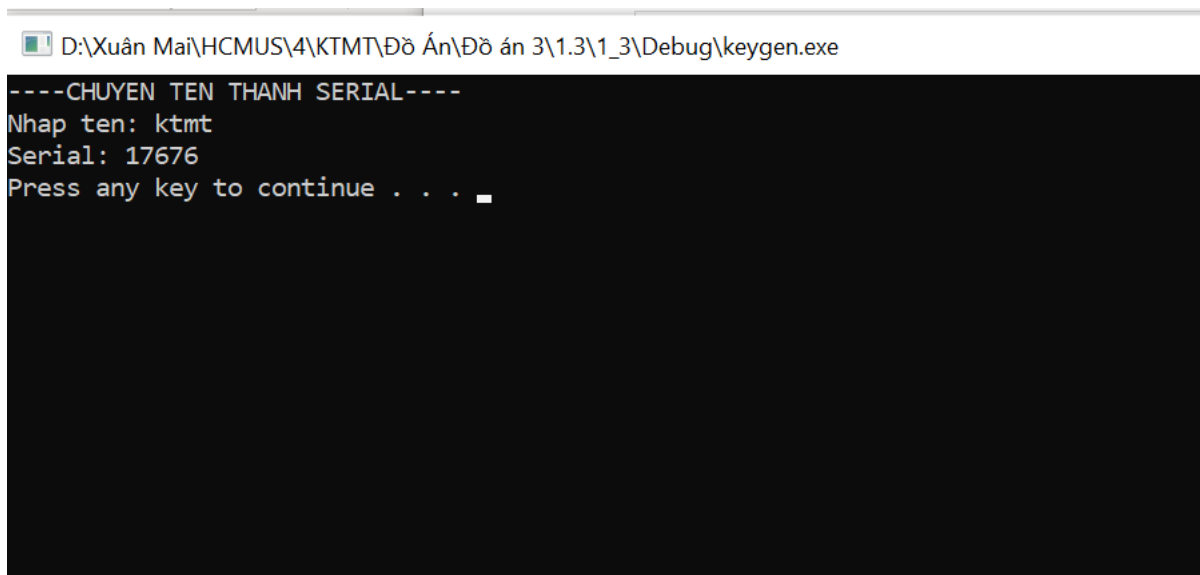
00401362	\$. 6A 00	PUSH 0	{ BeepType = MB_OK MessageBoxA Style = MB_OK!MB_ICONEXCLAMATION!MB_APPLMODAL Title = "No luck!" Text = "No luck there, mate!" hOwner MessageBoxA }
00401364	. E8 AD000000	CALL <JMP.&USER32.MessageBoxBeep>	
00401369	. 6A 30	PUSH 30	
0040136B	. 68 60214000	PUSH 1_3.00402160	
00401370	. 68 69214000	PUSH 1_3.00402169	
00401375	. FF75 08	PUSH DWORD PTR [EBP+8]	
00401378	. E8 BD000000	CALL <JMP.&USER32.MessageBoxA>	
0040137D	. C3	RET	

b. Đề xuất thuật toán keygen

Dựa theo chương trình đọc được từ đoạn mã trên trong phần mềm Olly Dbg, ta thấy tương ứng 1 name sẽ có 1 serial riêng biệt. Serial có thể được tính toán từ name khi nhập vào bằng một số phép tính như sau:

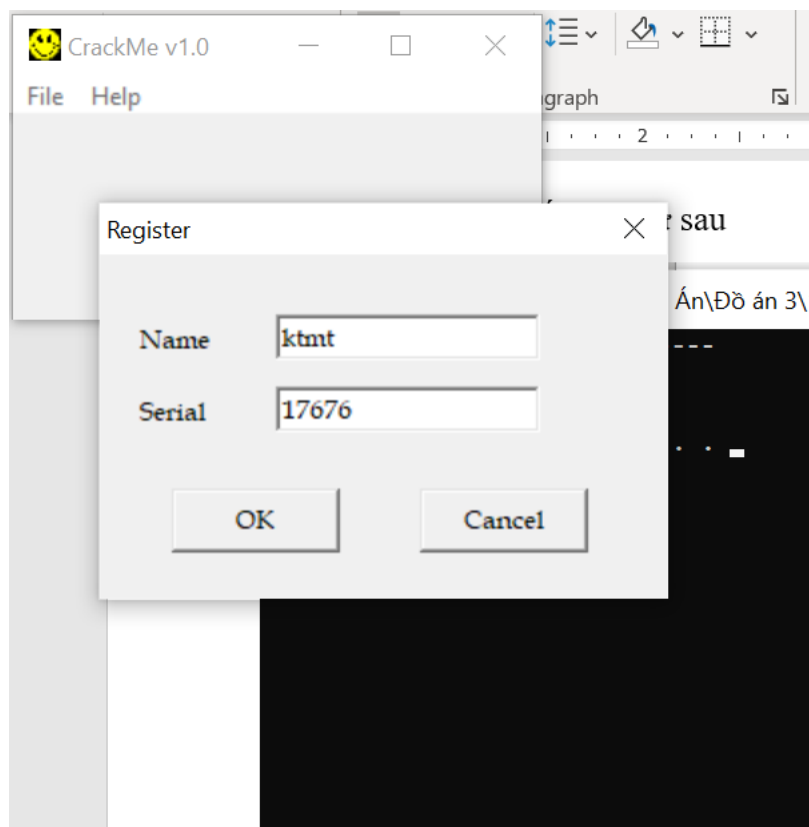
- Kiểm tra nếu chỉ chứa các chữ cái thì upcase toàn bộ (trừ các chữ đã là chữ in hoa sẵn), ngược lại thì phải nhập vào tên khác.
- Sau đó tính tổng các ký tự trong name.
- XOR kết quả với 0x5678.
- XOR tiếp tục kết quả với 0x1234.
- Kết quả thu được chính là serial dạng số.

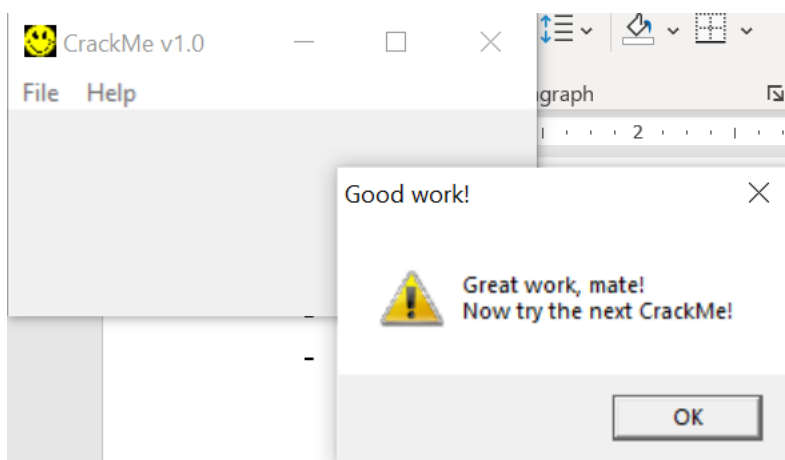
Viết chương trình sinh keygen và thử với 1 name bất kỳ như sau:



```
D:\Xuân Mai\HCMUS\4\KTMT\Đồ Án\Đồ án 3\1.3\1_3\Debug\keygen.exe
----CHUYEN TEN THANH SERIAL----
Nhap ten: ktmt
Serial: 17676
Press any key to continue . . .
```

Sau đó test lại bằng cách đăng nhập vào file đề.





Thành công.

4. Đánh giá mức độ hoàn thành

Chương trình	Mức độ	Note
1_1	100%	
1_2	100%	Hoàn thành keygen
1_3	100%	Hoàn thành keygen

5. Tài liệu tham khảo

- Cracking Tutorial file in Moodle
- <https://whitehat.vn/threads/re2-huong-dan-su-dung-ollydbg.883/>