

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN 3**  
**MÔN: KIẾN TRÚC MÁY TÍNH & HỢP NGỮ**

# **CRACKING**

**THỰC HIỆN:**

**18120195 – Vương Thị Ngọc Linh**

**18120247 – Phạm Hồ Ngọc Trâm**

**18120261 – Phạm Hoàng Việt**

**18120284 – Lê Trọng Bằng**

**18120304 – Võ Văn Hoàng Danh**

**Thành phố Hồ Chí Minh – Tháng 6/2020**

---

MỤC LỤC

---

<b>CHƯƠNG 1: GIỚI THIỆU</b>	<b>2</b>
<b>CHƯƠNG 2: NHỮNG PHẦN ĐÃ HOÀN THÀNH VÀ CHƯA HOÀN THÀNH</b>	<b>3</b>
<b>CHƯƠNG 3: PHÂN CHIA CÔNG VIỆC</b>	<b>4</b>
<b>CHƯƠNG 4: CÂU 1</b>	<b>5</b>
<b>CHƯƠNG 5: CÂU 2</b>	<b>10</b>
<b>CHƯƠNG 6: CÂU 3</b>	<b>20</b>
<b>CHƯƠNG 7: THAM KHẢO</b>	<b>24</b>

---

## CHƯƠNG 1: GIỚI THIỆU

---

- Trong phần đồ án này ta sẽ sử dụng những kiến thức về hợp ngữ x86 để crack phần mềm (vì mục đích học tập).
- Đồ án gồm :
  - + Báo cáo mô tả các bước cracking câu 1, 2, 3.
  - + File .cpp và .exe keygen câu 2, 3.
- $[MSSV \text{ các thành viên} + ] \bmod 3 + 1 = 1 = 24 \bmod 3 = 0 + 1 = \text{Đề 1}.$

---

**CHƯƠNG 2: NHỮNG PHẦN ĐÃ HOÀN THÀNH VÀ CHƯA HOÀN THÀNH**

---

Công việc	Mức độ hoàn thành	Ghi chú
Crack câu 1	100%	
Crack câu 2	100%	
Crack câu 3	100%	
Keygen câu 2	100%	
Keygen câu 3	100%	

---

**CHƯƠNG 3: PHÂN CHIA CÔNG VIỆC**

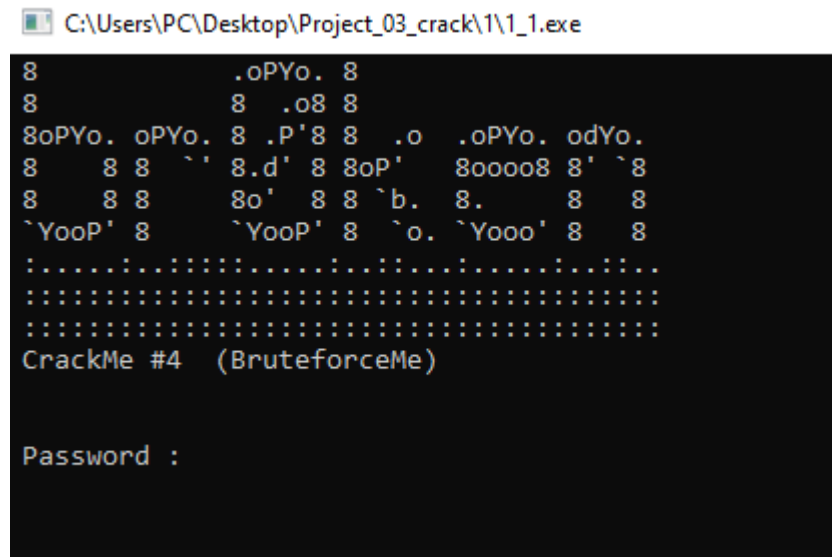
---

Công việc	Người phụ trách	Ghi chú
<u>Crack câu 1</u>	<u>Bằng + Danh</u>	
<u>Crack câu 2</u>	<u>Viết + Danh</u>	
<u>Crack câu 3</u>	<u>Linh</u>	
<u>Keygen câu 2</u>	<u>Viết</u>	
<u>Keygen câu 3</u>	<u>Trâm</u>	
<u>Báo cáo</u>	<u>Linh + Trâm</u>	

## CHƯƠNG 4: CẤU 1

### 4.1 Thông tin cơ bản:

- Khởi động chương trình.



- Password là 1 chuỗi kí tự (chưa biết có bao nhiêu kí tự).
- Khi nhập password sai, chương trình sẽ thông báo “**Nope... try again.**”
- Để tìm được password, ta cần tìm tất cả text string. Click chuột phải -> chọn Search for -> All references text strings.
- Ta tìm thấy thông báo khi nhập sai password, click vào địa chỉ của thông báo này.
- Kiểm tra chương trình, ta thấy thông báo khi nhập đúng password là “**That’s right! Now write a small tut :).**”

00401454	. 34 78	XOR AL,78	
00401456	. 0FBEC0	MOVSX EAX,AL	
00401459	. 8985 60FDFFFF	MOV DWORD PTR [EBP-2A0],EAX	
0040145F	. 8B85 60FDFFFF	MOV EAX,DWORD PTR [EBP-2A0]	
00401465	. 894424 1C	MOV DWORD PTR [ESP+1C],EAX	
00401469	. 8B85 64FDFFFF	MOV EAX,DWORD PTR [EBP-29C]	
0040146F	. 894424 18	MOV DWORD PTR [ESP+18],EAX	
00401473	. 8B85 68FDFFFF	MOV EAX,DWORD PTR [EBP-298]	
00401479	. 894424 14	MOV DWORD PTR [ESP+14],EAX	
0040147D	. 8B85 6CFDFFFF	MOV EAX,DWORD PTR [EBP-294]	
00401483	. 894424 10	MOV DWORD PTR [ESP+10],EAX	
00401487	. 8B85 70FDFFFF	MOV EAX,DWORD PTR [EBP-290]	
0040148D	. 894424 0C	MOV DWORD PTR [ESP+C],EAX	
00401491	. 8B85 74FDFFFF	MOV EAX,DWORD PTR [EBP-28C]	
00401497	. 894424 08	MOV DWORD PTR [ESP+8],EAX	
0040149B	. C74424 04 4E324000	MOV DWORD PTR [ESP+4],1_1.0040324E	ASCII "XXXXXXXXXXXX"
004014A3	. 8D85 58FEFFFF	LEA EAX,DWORD PTR [EBP-1A8]	
004014A9	. 898424	MOV DWORD PTR [ESP],EAX	
004014AC	. E8 2F060000	CALL <JMP.&USER32.wsprintfA>	wsprintfA
004014B1	> 8D85 88FDFFFF	LEA EAX,DWORD PTR [EBP-278]	
004014B7	. 8D95 58FEFFFF	LEA EDI,DWORD PTR [EBP-1A8]	
004014BD	. 894424 04	MOV DWORD PTR [ESP+4],EAX	
004014C1	. 891424	MOV DWORD PTR [ESP],EDI	
004014C4	. E8 87050000	CALL <JMP.&msvcr7.strcmp>	strcmp
004014C9	. 85C0	TEST EAX,EAX	
004014CB	~ 75 0E	JNZ SHORT 1_1.004014DB	
004014CD	. C70424 5C324000	MOV DWORD PTR [ESP],1_1.0040325C	ASCII "That's right! Now write a small tut :)"
004014D4	. E8 A7050000	CALL <JMP.&msvcr7.printf>	printf
004014D9	~ EB 0C	JMP SHORT 1_1.004014E7	
004014DB	> C70424 87324000	MOV DWORD PTR [ESP],1_1.00403287	ASCII "Nope... try again."
004014E2	. E8 99050000	CALL <JMP.&msvcr7.printf>	printf
004014E7	> E8 D4040000	CALL <JMP.&msvcr7._getch>	_getch
004014EC	. B8 00000000	MOV EAX,0	
004014F1	. C9	LEAVE	

- Ta thấy trong chương trình, ở địa chỉ **004014CB**, có lệnh `JNZ SHORT 1_1.004014DB`. Địa chỉ này chính là địa chỉ xuất ra thông báo khi ta nhập sai password.
- Ngay trên lệnh `JNZ` đó, ta có lời gọi hàm `CALL strcmp()`, nghĩa là chương trình sẽ thực hiện so sánh 2 chuỗi nếu trùng khớp sẽ xuất thông báo password đúng và ngược lại.
- Trên lời gọi hàm `strcmp()` chương trình có thêm vào stack 2 dữ liệu được lấy từ 2 địa chỉ là `[EBP-278]` và `[EBP-1A8]`
- > Ta cần xác định đâu là dữ liệu cố định của chương trình, đâu là dữ liệu mà người dùng nhập vào.
- Tại địa chỉ **004014A3**, chương trình thực hiện dòng lệnh lưu địa chỉ của `[EBP-1A8]` vào thanh ghi `EAX`.
- Sau đó đưa địa chỉ này vào đầu stack `[ESP]` rồi gọi hàm `wsprintf` ở địa chỉ `[EBP-1A8]` để lưu 1 chuỗi sau khi đã được thực hiện 1 số bước xử lý ở các dòng lệnh phía trên.
- > Ta có thể đoán được dữ liệu tại địa chỉ **[EBP-1A8]** là dữ liệu mà người dùng nhập vào và dữ liệu **[EBP-278]** là dữ liệu cố định trong chương trình.

## 4.2 Debug bằng OllyDgb:

- Ta đặt Break point tại địa chỉ **004014B1** và tiến hành debug để tìm ra giá trị cố định mà chương trình đã tạo ra.

004014A3	. 8085 58FEFFFF	LEA EAX,DWORD PTR [EBP-1A8]	
004014A9	. 890424	MOV DWORD PTR [ESP],EAX	
004014AC	. E8 2F060000	CALL <JMP.&USER32.wsprintfA>	<code>wsprintfA</code>
004014B1	> 8085 88FDFFFF	LEA EAX,DWORD PTR [EBP-278]	
004014B7	. 8095 58FEFFFF	LEA EDI,DWORD PTR [EBP-1A8]	
004014BD	. 894424 04	MOV DWORD PTR [ESP+4],EAX	
004014C1	. 891424	MOV DWORD PTR [ESP],EDI	
004014C4	. E8 87050000	CALL <JMP.&msvrt.strcmp>	<code>strcmp</code>
004014C9	. 85C0	TEST EAX,EAX	
004014CB	. 75 0E	JNZ SHORT 1_1.004014DB	
004014CD	. C70424 5C324000	MOV DWORD PTR [ESP],1_1.0040325C	ASCII "That's right! Now write a small tut :)"
004014D4	. E8 A7050000	CALL <JMP.&msvrt.printf>	<code>printf</code>
004014D9	. EB 0C	JMP SHORT 1_1.004014E7	
004014DB	> C70424 87324000	MOV DWORD PTR [ESP],1_1.00403287	ASCII "Nope... try again."
004014E2	. E8 99050000	CALL <JMP.&msvrt.printf>	<code>printf</code>
004014E7	. E8 D4040000	CALL <JMP.&msvrt._getch>	<code>_getch</code>
004014EC	. B8 00000000	MOV EAX,0	
004014F1	. C9	LEAVE	
004014F2	. C3	RET	
004014F3	. 90	NOP	
004014F4	. 90	NOP	
004014F5	. 90	NOP	
004014F6	. 90	NOP	
004014F7	. 90	NOP	

Stack address=0060FCB0, (ASCII "4D11628EBE1D")  
 EAX=00000004  
 Jump from 004013ED

- Xác định được giá trị tại địa chỉ `[EBP-278]` là **"4D11628EBE1D"**
- Ngoài ra, ta thấy dòng "Jump from **004013ED**" nghĩa là dòng này được nhảy đến từ dòng **004013ED** ta tìm đến địa chỉ này.

004013A1	. E8 DA060000	CALL <JMP.&msvcrt.printf>	printf
004013A6	. C70424 3C324000	MOV DWORD PTR [ESP],1_1.0040323C	ASCII "%s" Password : "
004013AD	. E8 CE060000	CALL <JMP.&msvcrt.printf>	printf
004013B2	. 8D85 28FFFFFF	LEA EAX,DWORD PTR [EBP-08]	
004013B8	. 894424 04	MOV DWORD PTR [ESP+4],EAX	
004013BC	. C70424 4B324000	MOV DWORD PTR [ESP],1_1.0040324B	ASCII "%s"
004013C3	. E8 A8060000	CALL <JMP.&msvcrt scanf>	scanf
004013C8	. 8D85 28FFFFFF	LEA EAX,DWORD PTR [EBP-08]	
004013CE	. 890424	MOV DWORD PTR [ESP],EAX	
004013D1	. E8 8A060000	CALL <JMP.&msvcrt.strlen>	strlen
004013D6	. 8985 78FDFFFF	MOV DWORD PTR [EBP-288],EAX	
004013DC	. 8D85 28FFFFFF	LEA EAX,DWORD PTR [EBP-08]	
004013E2	. 890424	MOV DWORD PTR [ESP],EAX	
004013E5	. E8 76060000	CALL <JMP.&msvcrt.strlen>	strlen
004013EA	. 83F8 06	CMP EAX,6	
004013ED	. 0F85 BE000000	JNZ 1_1.004014B1	
004013F3	. 0FB685 28FFFFFF	MOVZX EAX,BYTE PTR [EBP-08]	
004013FA	. 34 34	XOR AL,34	
004013FC	. 0FBEC0	MOVZX EAX,AL	
004013FF	. 8985 74FDFFFF	MOV DWORD PTR [EBP-28C],EAX	
00401405	. 0FB685 28FFFFFF	MOVZX EAX,BYTE PTR [EBP-07]	

- Ta thấy, lệnh “CMP EAX,6”. Mặc khác, sau khi thực hiện lời gọi hàm “scanf”, password người dùng nhập vào sẽ được lưu tại địa chỉ [EBP-D8], tiếp theo chương trình thực hiện lời gọi hàm strlen để đếm độ dài của chuỗi nhập vào rồi lưu độ dài vào thanh ghi EAX.

-> Password là 1 chuỗi có độ dài là 6.

- Lúc này chương trình sẽ so sánh độ dài chuỗi người dùng nhập vào với 6, nếu khác nhau sẽ thông báo password sai, ngược lại thì sẽ tiếp tục xử lý chuỗi này.



004013DC	. 8D85 28FFFFFF	LEA EAX,DWORD PTR [EBP-D8]	
004013E2	. 890424	MOV DWORD PTR [ESP],EAX	
004013E5	. E8 76060000	CALL <JMP.&msvort.strlen>	strlen
004013EA	. 83F8 06	CMP EAX,6	
004013ED	~ 0F85 BE000000	JNZ 1_1.004014B1	
004013F3	. 0FB685 28FFFFFF	MOUZ% EAX,BYTE PTR [EBP-D8]	
004013FA	. 34 34	XOR AL,34	
004013FC	. 0FBEC0	MOVSX EAX,AL	
004013FF	. 8985 74FDFFFF	MOV DWORD PTR [EBP-28C],EAX	
00401405	. 0FB685 29FFFFFF	MOUZ% EAX,BYTE PTR [EBP-D7]	
0040140C	. 34 78	XOR AL,78	
0040140E	. 0FBEC0	MOVSX EAX,AL	
00401411	. 8985 70FDFFFF	MOV DWORD PTR [EBP-290],EAX	
00401417	. 0FB685 2AFFFFFF	MOUZ% EAX,BYTE PTR [EBP-D6]	
0040141E	. 34 12	XOR AL,12	
00401420	. 0FBEC0	MOVSX EAX,AL	
00401423	. 8985 6CFDFFFF	MOV DWORD PTR [EBP-294],EAX	
00401429	. 0FB685 2BFFFFFF	MOVSX EAX,BYTE PTR [EBP-D5]	
00401430	. 35 FE000000	XOR EAX,0FE	
00401435	. 8985 68FDFFFF	MOV DWORD PTR [EBP-298],EAX	
0040143B	. 0FB685 2CFFFFFF	MOVSX EAX,BYTE PTR [EBP-D4]	
00401442	. 35 DB000000	XOR EAX,0DB	
00401447	. 8985 64FDFFFF	MOV DWORD PTR [EBP-29C],EAX	
0040144D	. 0FB685 2DFFFFFF	MOUZ% EAX,BYTE PTR [EBP-D3]	
00401454	. 34 78	XOR AL,78	
00401456	. 0FBEC0	MOVSX EAX,AL	
00401459	. 8985 60FDFFFF	MOV DWORD PTR [EBP-2A0],EAX	
0040145F	. 8B85 60FDFFFF	MOV EAX,DWORD PTR [EBP-2A0]	
00401465	. 894424 1C	MOV DWORD PTR [ESP+1C],EAX	
00401469	. 8B85 64FDFFFF	MOV EAX,DWORD PTR [EBP-29C]	
0040146F	. 894424 18	MOV DWORD PTR [ESP+18],EAX	
00401473	. 8B85 68FDFFFF	MOV EAX,DWORD PTR [EBP-298]	
00401479	. 894424 14	MOV DWORD PTR [ESP+14],EAX	
0040147D	. 8B85 6CFDFFFF	MOV EAX,DWORD PTR [EBP-294]	
00401483	. 894424 10	MOV DWORD PTR [ESP+10],EAX	
00401487	. 8B85 70FDFFFF	MOV EAX,DWORD PTR [EBP-290]	
0040148D	. 894424 0C	MOV DWORD PTR [ESP+C],EAX	
00401491	. 8B85 74FDFFFF	MOV EAX,DWORD PTR [EBP-28C]	
00401497	. 894424 08	MOV DWORD PTR [ESP+8],EAX	
0040149B	. C74424 04 4E324000	MOV DWORD PTR [ESP+4],1_1.0040324E	ASCII "XXXXXXXXXX"
004014A3	. 8D85 58FEFFFF	LEA EAX,DWORD PTR [EBP-1A8]	
004014A9	. 890424	MOV DWORD PTR [ESP],EAX	
004014AC	. E8 2F060000	CALL <JMP.&USER32.wsprintfA>	wsprintfA
004014B1	> 8D85 88FDFFFF	LEA EAX,DWORD PTR [EBP-278]	
004014B7	. 8D95 58FEFFFF	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.&msvort.strcmp>	strcmp
004014C9	. 85C0	TEST EAX,EAX	
004014CB	~ 75 0E	JNZ SHORT 1_1.004014DB	
004014CD	. C70424 5C324000	MOV DWORD PTR [ESP],1_1.0040325C	ASCII "That's right! Now "
004014D4	. E8 A7050000	CALL <JMP.&msvort.printf>	printf
004014D9	~ EB 0C	JMP SHORT 1_1.004014E7	
004014DB	> C70424 87324000	MOV DWORD PTR [ESP],1_1.00403287	ASCII "Nope... try again."
004014E2	. E8 99050000	CALL <JMP.&msvort.printf>	printf
004014E7	> E8 D4040000	CALL <JMP.&msvort._getch>	_getch

- Sau khi kiểm tra độ dài chuỗi đã thỏa mãn điều kiện, chương trình tiếp tục thực hiện 1 số lệnh bên dưới.

- Tại địa chỉ **004013F3**, chương trình lấy 1 byte tại địa chỉ **[EBP-D8]**, tức là lấy ký tự đầu tiên của chuỗi người dùng nhập vào lưu vào 8 bit thấp (AL) của thanh ghi EAX, đồng thời các bit trái còn lại được điền là những số 0.

- Theo quy tắc xor, **A xor B = C -> A = C xor B**.

- A là password cần tìm để nhập đúng nên để tìm A ta lấy C (các giá trị lưu trong AL) xor lần lượt với 34, 78, 12, 0FE, 0DB, 78

-Sau khi xor, chương trình lưu giá trị này tại địa chỉ **[EBP-28C]**.

	Hash[0]	Hash[1]	Hash[2]	Hash[3]	Hash[4]	Hash[5]
RES	4D	11	62	8E	BE	1D
XOR	34	78	12	0FE	0DB	78
HEX	79	69	70	70	65	65
ASCII	y	i	p	p	e	e

-> Password là yippee

```
C:\Users\N\OneDrive\Desktop\Project_02_Crack\11_12_2022
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 :)
```

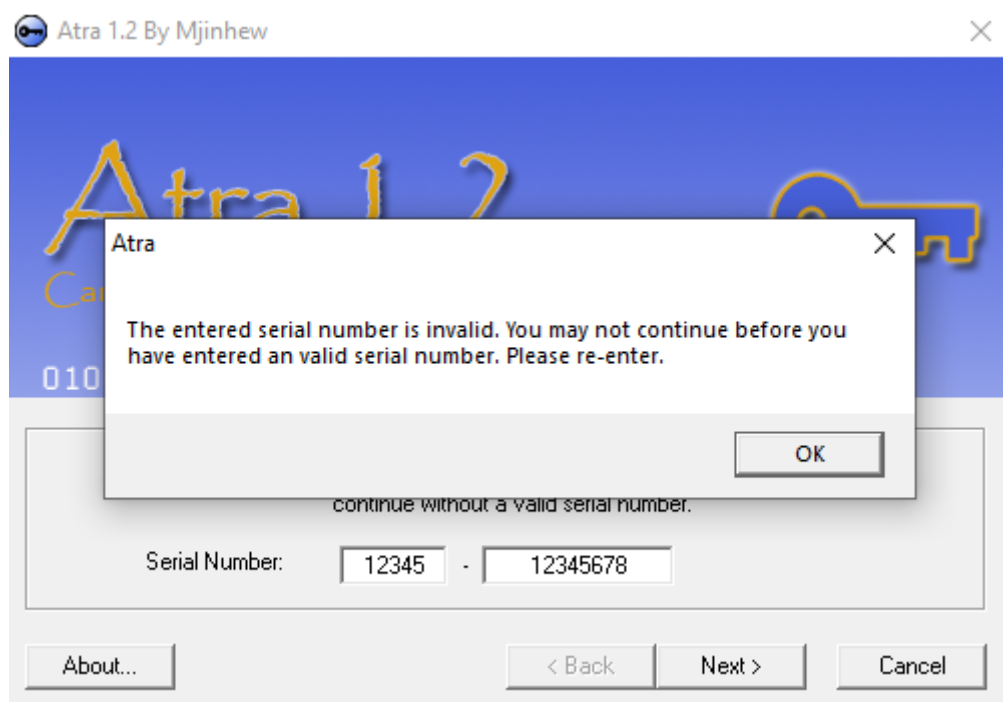
## CHƯƠNG 5: CÂU 2

### 5.1 Thông tin cơ bản:

Người dùng nhập vào một chuỗi Serial gồm 2 phần, phần 1 – 5 ký tự, phần 2 – 8 ký tự (như hình dưới)



Nếu nhập sai thì sẽ hiện ra thông báo: **“The entered serial number is invalid. You may not continue before you have entered an valid serial number. Please re-enter.”**



## 5.2 Debug chương trình bằng OllyDbg:

Tìm các text strings trong chương trình (chuột phải → Search for → All reference text strings)

Ta thấy thông báo nhập đúng là **“The entered serial number is VALiD!”**

Address	Disassembly	Text string
00401208	MOV EDI, 12.004032DC	RSCII "BDRKPTUJ!"
00401278	MOV EDI, 12.004032DC	RSCII "BDRKPTUJ!"
004013FB	MOV EBX, 12.004044C8	RSCII "20"
0040148E	PUSH 12.004044C0	RSCII "Atra"
0040148E	PUSH 12.0040449C	RSCII "The entered serial number is VALiD!"
0040148E	PUSH 12.004044C0	RSCII "Atra"
004014C3	PUSH 12.004044C0	RSCII "The entered serial number is invalid. You may not continue before you have entered a valid serial number. Please re-enter."
004014E7	PUSH 12.004044CC	RSCII "\\\NTICE"
00401544	PUSH 12.004044D8	RSCII "Atra"
00401549	PUSH 12.004044D8	RSCII "Name: Atra @Language: C/C++ @Difficulty: 2-3/10 @Author: Mjinheve@Create a keygen that will generate some random serials.@Coded in MS VC++ 6.0, GFX made in Photoshop 2.0"
00401FEE	PUSH EBP	(Initial CPU selection)

Đặt break-point tại dòng **00401292** và **004012A5**. F9 để Run -> F8 :

0040128F	. 8975 F4	MOV DWORD PTR [EBP-C],ESI
00401292	. E8 D0C00000	CALL <JMP.&MFC42.#3098>
00401297	. 8B4D F8	MOV ECX,DWORD PTR [EBP-8]
0040129A	. 8D45 BC	LEA EAX,DWORD PTR [EBP-44]
0040129D	. 6A 0C	PUSH 0C
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 [EBP-20]
004012BA	. 50	PUSH EAX
004012BB	. E8 1C0D0000	CALL <JMP.&MSVCRT.strlen>
004012C0	. 85C0	TEST EAX,EAX
004012C2	. 59	POP ECX
004012C3	✓ 0F86 92000000	JBE 1_2.0040135B
004012C9	. 8A45 E0	MOV AL,BYTE PTR [EBP-20]
004012CC	B1 03	MOV CL,3
Stack SS:[0019F504]=0019FE38		
ECX=AS00CE3		

Chuỗi nhập vào được lưu tại:

0019F4AC	00401297	..@.	1_2.00401297
0019F4B0	000003E8	....	
0019F4B4	0019F4EC	....	ASCII "12345"
0019F4B8	00000006	....	
0019F4BC	0019FE38	8...	
0019F4C0	62CC3FD8	..?.b	MFC42.#4234
0019F4C4	00000001	....	
0019F4C8	000014A0	....	
0019F4CC	00000000	....	
0019F4D0	0019F500	....	
0019F4D4	0019F504	....	
0019F4D8	00000000	....	
0019F4DC	00000000	....	
0019F4E0	00000000	....	
0019F4E4	0019F560	'...	
0019F4E8	65010F32	2..e	
0019F4EC	34333231	1234	
0019F4F0	74E90035	5..t	gdi32ful.74E90035
0019F4F4	0019F560	'...	
0019F4F8	00000000	....	

Tại vị trí in ra “badboy”, ta thấy nó được jump từ các dòng: **00401325, 0040136A, 004013A8, 004013D1, 004013DB.**

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 number is invalid. You may not continue I
004014C8	> 8B4D F8	MOV ECX,DWORD PTR [EBP-8]	
004014CB	. E8 9E0A0000	CALL <JMP.&MFC42.#4224>	
004014D0	> 5F	POP EDI	
004014D1	. 5E	POP ESI	
004014D6	. 5D	POP EBX	
Jumps from 00401325, 0040136A, 004013A8, 004013D1, 004013DB			

- Dòng **00401325**:

0040131E	>	FF45 F4	INC DWORD PTR [EBP-C]	
00401321	.	837D F4 2F	CMP DWORD PTR [EBP-C],2F	
00401325	✓	0F87 9101000	JA 1_2.004014BC	
0040132B	>	57	PUSH EDI	[ <sup>s</sup> strlen
0040132C	.	46	INC ESI	
0040132D	.	E8 AA0C0000	CALL <JMP.&MSUCRT.strlen>	
00401332	.	3BF0	CMP ESI,EAX	
00401334	.	59	POP ECX	
00401335	^	72 CB	JB SHORT 1_2.00401302	
00401337	>	EE45 EC	INC DWORD PTR [EBP-4]	
004014BC=1_2.004014BC				

- Dòng 0040136A:

00401367	• 85C0	TEST EAX,EAX	
00401369	• 59	POP ECX	
0040136A	✓ 0F86 4C010000	JBE 1_2.004014BC	
00401370	• 8D5D E0	LEA EBX,DWORD PTR [EBP-20]	
00401373	• BF DC324000	MOV EDI,1_2.004032DC	ASCII "BDRQKPTUJI"
00401378	> 57	PUSH EDI	[ <sup>s</sup> strlen
00401379	• 33F6	XOR ESI,ESI	
0040137B	• E8 5C0C0000	CALL <JMP.&MSUCRT.strlen>	
00401380	• 85C0	TEST EAX,EAX	
004014BC=1_2.004014BC			

- Dòng 004013A8:

004013A1	.	837D F4 2F	CMP DWORD PTR [EBP-C],2F	
004013A8	✓	0F87 0E010000	JA 1_2.004014BC	
004013AE	>	57	PUSH EDI	[ <sup>s</sup> strlen
004013AF	.	46	INC ESI	
004013B0	.	E8 270C0000	CALL <JMP.&MSUCRT.strlen>	
004013B5	.	3BF0	CMP ESI,EAX	
004013B7	.	59	POP ECX	
004013B8	^	72 CB	JB SHORT 1_2.00401385	
004014BC=1_2.004014BC				

- Dòng 004013D1:

004013CD	.	837D F0 03	CMP DWORD PTR [EBP-10],3	
004013D1	✓	0F85 E500000	JNZ 1_2.004014BC	
004013D7	.	837D EC 02	CMP DWORD PTR [EBP-14],2	
004013DB	✓	0F85 DB00000	JNZ 1_2.004014BC	
004013E1	.	8D45 E0	LEA EAX,DWORD PTR [EBP-20]	
004013E4	.	50	PUSH EAX	[ <sup>s</sup> strlen
004013E5	.	E8 F20B0000	CALL <JMP.&MSUCRT.strlen>	
004013E9	.	50	PUSH EAX	
004014BC=1_2.004014BC				

00401385	> 8A03	MOV AL, BYTE PTR [EBX]
00401387	. 3A86 DC32400	CMP AL, BYTE PTR [ESI+4032DC]
0040138D	. 75 05	JNZ SHORT 1_2.00401394
0040138F	. FF45 F0	INC DWORD PTR [EBP-10]
00401392	. EB 1A	JMP SHORT 1_2.004013AE
00401394	> 3A86 D032400	CMP AL, BYTE PTR [ESI+4032D0]
0040139A	. 75 05	JNZ SHORT 1_2.004013A1
0040139C	. FF45 EC	INC DWORD PTR [EBP-14]
0040139F	. EB 0D	JMP SHORT 1_2.004013AE
004013A1	> FF45 F4	INC DWORD PTR [EBP-C]
004013A4	. 837D F4 2F	CMP DWORD PTR [EBP-C], 2F
004013A8	. 0F87 0E01000	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

Tại dòng **00401373** ta thấy có một chuỗi ASCII được quy định sẵn là **“BDRQKPTVJI”**. Đoạn code từ dòng **00401385** đến dòng **004013B8** đếm số ký tự trong chuỗi part 1 mà người dùng nhập vào mà có xuất hiện trong chuỗi **“BDRQKPTVJI”** và lưu vào **[EBP-10]**, sau đó kiểm tra số ký tự trong chuỗi part 1 có xuất hiện trong chuỗi **“0123456789”** và lưu vào **[EBP-14]**.

Đoạn code tại dòng **004013CD** đếm xem **[EBP-10]** có bằng 3 không, nếu không nhảy đến **“badboy”**. Tương tự code tại dòng **004013D7** kiểm tra **[EBP-14]** có bằng 2 không, nếu không thì nhảy đến **“badboy”**.

004013CD	. 72 AB	JB SHORT 1_2.00401378
004013CD	. 837D F0 03	CMP DWORD PTR [EBP-10], 3
004013D1	. 0F85 E500000	JNZ 1_2.004014BC
004013D7	. 837D EC 02	CMP DWORD PTR [EBP-14], 2
004013DB	. 0F85 DB00000	JNZ 1_2.004014BC
004013E1	. 8D45 E0	LEA EAX, DWORD PTR [EBP-20]

Từ đó có thể đưa ra kết luận: **Part 1** có 5 ký tự với 3 ký tự từ chuỗi **“BDRQKPTVJI”** và 2 ký tự từ chuỗi **“0123456789”**.

### 5.2.1 Xử lý part 1:

Sau khi thực hiện việc so sánh part 1 với các chuỗi ở **[EBP-10]** và **[EBP-14]**, và đều đúng thì chương trình tiếp tục thực hiện cho đến dòng **004013EF** và jump đến **00401D5E**.

00401D5E	. 55	PUSH EBP	
00401D5F	. 8BEC	MOV EBP,ESP	
00401D61	. 83EC 68	SUB ESP,68	
00401D64	. 8D45 98	LEA EAX,DWORD PTR [EBP-68]	
00401D67	. 50	PUSH EAX	
00401D68	. E8 3AF8FFFF	CALL 1_2.004015A7	
00401D6D	. FF75 0C	PUSH DWORD PTR [EBP+C]	
00401D70	. 8D45 98	LEA EAX,DWORD PTR [EBP-68]	
00401D73	. FF75 08	PUSH DWORD PTR [EBP+8]	
00401D76	. 50	PUSH EAX	
00401D77	. E8 53F8FFFF	CALL 1_2.004015CF	
00401D7C	. 8D45 F0	LEA EAX,DWORD PTR [EBP-10]	
00401D7F	. 50	PUSH EAX	
00401D80	. 8D45 98	LEA EAX,DWORD PTR [EBP-68]	
00401D83	. 50	PUSH EAX	
00401D84	. E8 2CF8FFFF	CALL 1_2.00401CB5	
00401D89	. 8B45 FC	MOV EAX,DWORD PTR [EBP-4]	
00401D8C	. 83C4 18	ADD ESP,18	
00401D8F	. 3345 F8	XOR EAX,DWORD PTR [EBP-8]	
00401D92	. 3345 F4	XOR EAX,DWORD PTR [EBP-C]	
00401D95	. 3345 F0	XOR EAX,DWORD PTR [EBP-10]	
00401D98	. C9	LEAVE	
00401D99	. C3	RET	

Sau khi thực hiện các lệnh trong đoạn từ **00401D5E** đến **00401D99**, **RET** quay về dòng **004013FF**.

Đoạn code từ **004013F4** đến **00401405** là tiến hành chuyển kết quả thu được thành hệ hexa ở dạng string.

004013F4	. 8B3D B031400	MOV EDI,DWORD PTR [<MSUCRT.sprintf>]	MSUCRT.sprintf
004013FA	. 50	PUSH EAX	<XX>
004013FB	. BB C8444000	MOV EBX,1_2.004044C8	ASCII "%X"
00401400	. 8D45 D4	LEA EAX,DWORD PTR [EBP-2C]	format => "%X"
00401403	. 53	PUSH EBX	s
00401404	. 50	PUSH EAX	sprintf
00401405	. FFD7	CALL EDI	
00401407	. 8365 FC 00	AND DWORD PTR [EBP-4],0	
0040140B	. 8D45 D4	LEA EAX,DWORD PTR [EBP-2C]	
0040140E	. 50	PUSH EAX	s
0040140F	. E8 C80B0000	CALL <JMP.&MSUCRT.strlen>	strlen
00401414	. 83C4 1C	ADD ESP,1C	
00401417	. 85C0	TEST EAX,EAX	
00401419	. 76 29	JBE SHORT 1_2.00401444	

Dòng **00401419**, jump đến dòng **00401444** nếu *below or equal*.

Đoạn code sau khi jump đến dòng **00401444**:

00401444	> 8365 FC 00	AND DWORD PTR [EBP-4],0	
00401448	. 8D45 D4	LEA EAX,DWORD PTR [EBP-2C]	
0040144B	. 50	PUSH EAX	s
0040144C	. E8 8B0B0000	CALL <JMP.&MSUCRT.strlen>	strlen
00401451	. 85C0	TEST EAX,EAX	
00401453	. 59	POP ECX	
00401454	. 76 29	JBE SHORT 1_2.0040147F	

Nếu không thì tiếp tục thực hiện đoạn code từ **0040141B** đến **00401442**. Chú ý, ở dòng **00401429**, có lệnh **CALL 1\_2.00401247** → Đi đến dòng **00401247**:

00401419	. 76 29	JBE SHORT 1_2.00401444	
0040141B	> 8B45 FC	MOV EAX,DWORD PTR [EBP-4]	
0040141E	. 50	PUSH EAX	
0040141F	. 8D7405 D4	LEA ESI,DWORD PTR [EBP+EAX-2C]	
00401423	. 0FB4405 D4	MOVSX EAX,BYTE PTR [EBP+EAX-2C]	
00401428	. 50	PUSH EAX	
00401429	. E8 19FEFFFF	CALL 1_2.00401247	
0040142F	. FF45 FC	INC DWORD PTR [EBP-4]	
00401431	. 8B06	MOV BYTE PTR [ESI],AL	
00401433	. 8D45 D4	LEA EAX,DWORD PTR [EBP-2C]	
00401436	. 50	PUSH EAX	s
00401437	. E8 A00B0000	CALL <JMP.&MSUCRT.strlen>	strlen
0040143C	. 83C4 0C	ADD ESP,0C	
0040143F	. 3945 FC	CMPL DWORD PTR [EBP-4],EAX	
00401442	. 72 D7	JB SHORT 1_2.0040141B	



- Dòng **00401247**: “**MOV EAX, DWORD PTR [ESP + 8]**” → **EAX** = 32 bit trên cùng của stack.
- Dòng **0040124B**: “**MOV ECX, EAX**” → copy giá trị từ thanh ghi *EAX* vào thanh ghi *ECX*.
- Dòng **0040124D**: “**SHL ECX, 2**” → dịch trái 2 lần dãy bit trong **ECX**, tương đương với  $ECX_{16} \times 16^2$ .
- Dòng **00401250**: “**MOV EAX, DWORD PTR [ECX + 4032E8]**” → lưu giá trị từ thanh ghi có địa chỉ  $ECX+4032E8$  vào thanh ghi *EAX*.

00401247	8B4424 08	MOV EAX,DWORD PTR [ESP+8]
0040124B	. 8BC8	MOV ECX,EAX
0040124D	. C1E1 02	SHL ECX,2
00401250	. 8B81 E8324000	MOV EAX,DWORD PTR [ECX+4032E8]
00401256	. 2B81 20404000	SUB EAX,DWORD PTR [ECX+404020]
0040125C	. 334424 04	XOR EAX,DWORD PTR [ESP+4]
00401260	. C3	RET

### 5.2.2 Xử lý part 2:

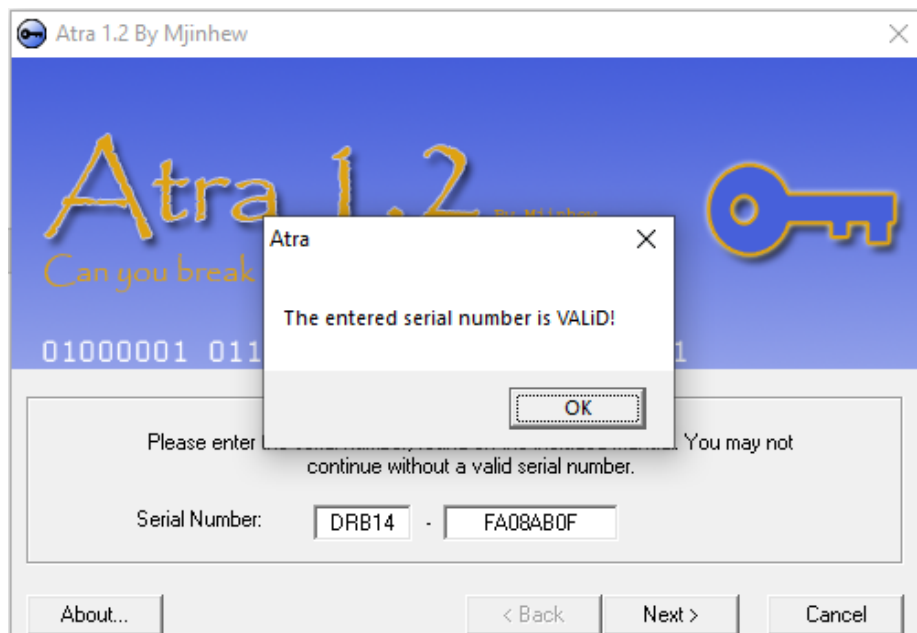
Với part 1 đã có định dạng, ta tiến hành nhập bất kì 1 part 1 nào đó, ở đây chúng ta nhập: “**DRB14**” và part 2 là “**12345678**”.



Đặt Break point ở dòng **0040149A**, kết quả thu được là:

0019F4A4	0019F4D4	....
0019F4A8	004044C8	.D@.
0019F4AC	FA08AB0F	....
0019F4B0	0019F4E0	....
0019F4B4	00000008	....
0019F4B8	0019F4E0	....
0019F4BC	0019FE38	8...
0019F4C0	6BF93FD8	.?.k
0019F4C4	00000001	....
0019F4C8	34333231	1234
0019F4CC	38373635	5678
0019F4D0	0019F500	....
0019F4D4	38304146	FA08
0019F4D8	46304241	AB0F
0019F4DC	00000000	....
0019F4E0	F9FAFC95	....
0019F4E4	C761E5E2	..a.
0019F4E8	60010900	...'

Ta có thể thấy ở dòng số **0019F4D4**: xuất hiện string **FA08AB0F**. Thử lại string này với vai trò là part 2, part 1 giữ nguyên.



Vậy đây là part 2 cần tìm, đoạn code phía trên dòng đặt break-point đã biến đổi chuỗi part 1 thành part 2 bằng 1 thuật toán nào đó.

Từ dòng **004015CF** đến **0040166C**: Xử lý chuỗi part 1 thành 1 chuỗi mới có 16bit hex.

Hình [1]

004015CF	55	PUSH EBP	
004015D0	8BEC	MOV EBP,ESP	
004015D2	53	PUSH EBX	
004015D3	56	PUSH ESI	
004015D4	8B75 08	MOV ESI,DWORD PTR [EBP+8]	
004015D7	57	PUSH EDI	
004015D8	8B7D 10	MOV EDI,DWORD PTR [EBP+10]	
004015DB	8B4E 10	MOV ECX,DWORD PTR [ESI+10]	
004015DE	8BD7	MOV EDX,EDI	
004015E0	8BC1	MOV EAX,ECX	
004015E2	C1E8 03	SHR EAX,3	
004015E5	8D0CF9	LEA ECX,DWORD PTR [ECX+EDI*8]	
004015E8	83E0 3F	AND EAX,3F	
004015EB	C1E2 03	SHL EDX,3	
004015EE	3BCA	CMP ECX,EDX	
004015F0	894E 10	MOV DWORD PTR [ESI+10],ECX	
004015F3	73 03	JNB SHORT 1_2.004015F8	
004015F5	FF46 14	INC DWORD PTR [ESI+14]	
004015F8	6A 40	PUSH 40	
004015FA	8BCF	MOV ECX,EDI	
004015FC	5B	POP EBX	
004015FD	C1E9 1D	SHR ECX,1D	
00401600	014E 14	ADD DWORD PTR [ESI+14],ECX	
00401603	2BD8	SUB EBX,EAX	
00401605	3BFB	CMP EDI,EBX	
00401607	72 42	JB SHORT 1_2.0040164B	
00401609	53	PUSH EBX	
0040160A	8D4430 18	LEA EAX,DWORD PTR [EAX+ESI+18]	
0040160E	FF75 0C	PUSH DWORD PTR [EBP+C]	
00401611	50	PUSH EAX	
00401612	E8 CB090000	CALL <JMP.&MSUCRT.memcpy>	
00401617	8D46 18	LEA EAX,DWORD PTR [ESI+18]	
0040161A	50	PUSH EAX	
0040161B	56	PUSH ESI	
0040161C	E8 4C000000	CALL 1_2.0040166D	
00401621	83C4 14	ADD ESP,14	
00401624	895D 08	MOV DWORD PTR [EBP+8],EBX	
00401627	83C3 3F	ADD EBX,3F	
0040162A	3BDF	CMP EBX,EDI	
0040162C	73 19	JNB SHORT 1_2.00401647	
0040162E	8B45 0C	MOV EAX,DWORD PTR [EBP+C]	
00401631	8D4418 C1	LEA EAX,DWORD PTR [EAX+EBX-3F]	
00401635	50	PUSH EAX	
00401636	56	PUSH ESI	
00401637	E8 31000000	CALL 1_2.0040166D	
0040163C	8345 08 40	ADD DWORD PTR [EBP+8],40	
00401640	59	POP ECX	
00401641	59	POP ECX	
00401642	83C3 40	ADD EBX,40	
00401645	EB E3	JMP SHORT 1_2.0040162A	
00401647	33C0	XOR EAX,EAX	
00401649	EB 04	JMP SHORT 1_2.0040164F	
0040164B	8365 08 00	AND DWORD PTR [EBP+8],0	
0040164F	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	
00401652	8B55 0C	MOV EDX,DWORD PTR [EBP+C]	
00401655	2BF9	SUB EDI,ECX	
00401657	03CA	ADD ECX,EDX	
00401659	57	PUSH EDI	
0040165A	8D4430 18	LEA EAX,DWORD PTR [EAX+ESI+18]	
0040165E	51	PUSH ECX	
0040165F	50	PUSH EAX	
00401660	E8 7D090000	CALL <JMP.&MSUCRT.memcpy>	
00401665	83C4 0C	ADD ESP,0C	
00401668	5F	POP EDI	
00401669	5E	POP ESI	
0040166A	5B	POP EBX	
0040166B	5D	POP EBP	
0040166C	C3	RET	

**Các bước xử lý như sau:**

- Ban đầu có part 1 là string, tạm gọi là s1. Từ s1 này, qua thuật toán ở hình[1] thì s1 sẽ được chuyển sang dạng hex và gắn vào đầu của 1 chuỗi hex mới (s có 64 bit) với dạng.
- + 5 phần tử đầu của s là s1, phần tử thứ 6 có mã là 0x80, phần tử thứ 57 có mã là 0x28.
- + Thực hiện biến chuỗi 64 phần tử thành chuỗi có 16 phần tử nhưng kiểu dữ liệu tăng từ 8bit lên 32bit.

00401419	.v 76 29	JBE SHORT 1_2.00401444	
0040141B	> 8B45 FC	MOV EAX,DWORD PTR [EBP-4]	
0040141E	. 50	PUSH EAX	
0040141F	. 8D7405 D4	LEA ESI,DWORD PTR [EBP+EAX-2C]	
00401423	. 0FBE4405 D4	MOVSX EAX,BYTE PTR [EBP+EAX-2C]	
00401428	. 50	PUSH EAX	
00401429	. E8 19FEFFFF	CALL 1_2.00401247	
0040142E	. FF45 FC	INC DWORD PTR [EBP-4]	
00401431	. 8B06	MOV BYTE PTR [ESI],AL	
00401433	. 8D45 D4	LEA EAX,DWORD PTR [EBP-2C]	
00401436	. 50	PUSH EAX	
00401437	. E8 A00B0000	CALL <JMP.&MSUCRT.strlen>	[s strlen
0040143C	. 83C4 0C	ADD ESP,0C	
0040143F	. 3945 FC	CMP DWORD PTR [EBP-4],EAX	
00401442	.^ 72 D7	JB SHORT 1_2.0040141B	

00401247	\$. 8B4424 08	MOV EAX,DWORD PTR [ESP+8]	
0040124B	. 8BC8	MOV ECX,EAX	
0040124D	. C1E1 02	SHL ECX,2	
00401250	. 8B81 E8324000	MOV EAX,DWORD PTR [ECX+4032E8]	
00401256	. 2B81 20404000	SUB EAX,DWORD PTR [ECX+404020]	
0040125C	. 334424 04	XOR EAX,DWORD PTR [ESP+4]	
00401260	. C3	RET	

+ Hai đoạn mã ở 2 hình trên sẽ thực hiện chuyển s về 1 chuỗi khác, như sau:

- o Gọi i là index của giá trị ascii trong s.
- o Chuỗi s được lưu ở thanh ghi ESI.
- o EAX = i = ECX
- o ESP = s[i]
- o ECX = ECX << i

EAX = giá trị tại thanh ghi có địa chỉ [ECX + 4032E8]

EAX = EAX – giá trị tại thanh ghi có địa chỉ [ECX + 404020]

EAX = EAX xor ESP

- o Lấy 2 bit cuối của EAX rồi lưu vào s[i]
- o Nếu EAX < 8 thì thực hiện lại.

00401261	\$. 8B4424 04	MOV EAX,DWORD PTR [ESP+4]	
00401265	. 8B4C24 08	MOV ECX,DWORD PTR [ESP+8]	
00401269	. D3E0	SHL EAX,CL	
0040126B	. 0B4424 04	OR EAX,DWORD PTR [ESP+4]	
0040126F	. C3	RET	

+ Tiếp theo, đoạn mã 8 ký tự đó sẽ được thực hiện theo nguyên tắc như hình trên:

- o Gọi p là mảng ký tự, I là vị trí của ký tự trong mảng
- o i = 0
- o p[i] = (p[i] << i) | p[i]
- o i++, nếu i = 8 thì thoát

+ Tiếp đó, đoạn mã từ dòng 004010C0 đến 004010FA:

004010C0	56	PUSH ESI
004010C1	33D2	XOR EDX,EDX
004010C3	83CE FF	OR ESI,FFFFFFFF
004010C6	395424 0C	CMP DWORD PTR [ESP+C],EDX
004010CA	7E 29	JLE SHORT 1_2.004010F5
004010CC	57	PUSH EDI
004010CD	B9 FF000000	MOV ECX,0FF
004010D2	8B4424 0C	MOV EAX,DWORD PTR [ESP+C]
004010D6	8BFE	MOV EDI,ESI
004010D8	23F9	AND EDI,ECX
004010DA	8A0402	MOV AL,BYTE PTR [EDX+EAX]
004010DD	23C1	AND EAX,ECX
004010DF	33C7	XOR EAX,EDI
004010E1	C1EE 08	SHR ESI,8
004010E4	8B0485 204040	MOV EAX,DWORD PTR [EAX*4+404020]
004010EB	33F0	XOR ESI,EAX
004010ED	42	INC EDX
004010EE	3B5424 10	CMP EDX,DWORD PTR [ESP+10]
004010F2	7C DE	JL SHORT 1_2.004010D2
004010F4	5F	POP EDI
004010F5	8BC6	MOV EAX,ESI
004010F7	5E	POP ESI
004010F8	F7D0	NOT EAX
004010FA	C3	RET

- Thực hiện chuyển đổi lần cuối, với nhiều thao tác
- 8 lần thực hiện

+ Sau khi thực hiện xong thì ta sẽ nhận được kết quả là 1 string 8byte.

- Chương trình sẽ so sánh chuỗi *s* này với part 2 do người dùng nhập. Nếu đúng thì sẽ hiện ra thông báo đúng hoặc ngược lại.

### 5.3 Key gen 1 2:

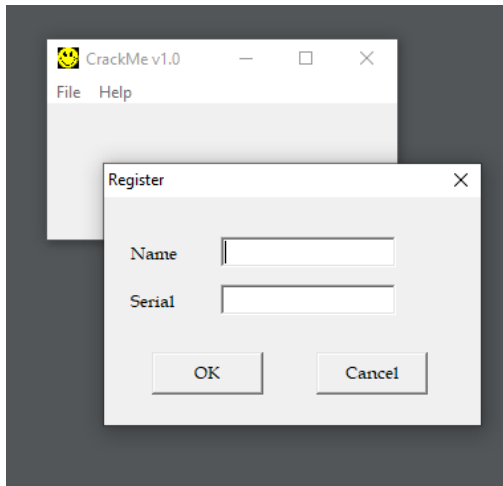
Nhận ra các phép biến đổi chuỗi để tạo thành key như được nhắc đến phía trên, ta viết được chương trình keygen, chương trình này thực hiện 3 thao tác:

- Đọc dữ liệu từ vùng nhớ
- Yêu cầu người dùng nhập serial number 1 thỏa yêu cầu
- Thực hiện biến đổi và xuất full key

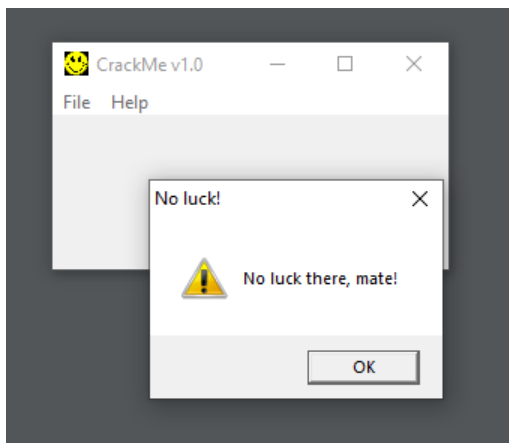
## CHƯƠNG 6: CÂU 3

### 6.1 Thông tin cơ bản:

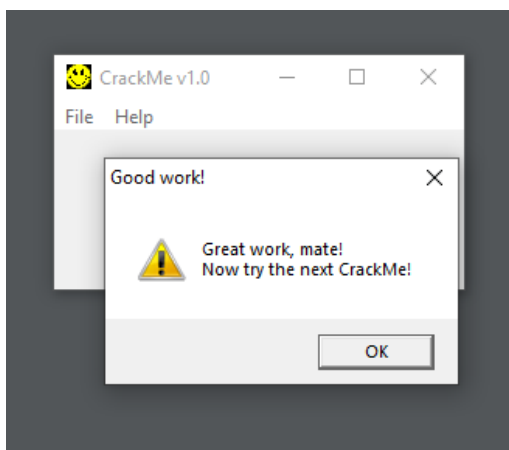
-Giao diện: Giao diện chương trình có nơi nhập Name và Serial đi kèm trong phần Register.



- Nếu nhập sai.



- Nếu nhập đúng.



## 6.2 Debug bằng OllyDbg:

- Xét đoạn code in ra badboy:

00401369	6A 30	PUSH 30	Style = MB_OK;MB_ICONEXCLAMATION;MB_APPLMODAL
0040136B	68 60214000	PUSH 1_3.00402160	Title = "No luck!"
00401370	68 69214000	PUSH 1_3.00402169	Text = "No luck there, mate!"
00401375	FF75 08	PUSH DWORD PTR [EBP+8]	hOwner
00401378	E8 BD000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA

Và

004013A0	. 6A 30	PUSH 30	[Style = MB_OK;MB_ICONEXCLAMATION;MB_APPLMODAL Title = "No luck!" Text = "No luck there, mate!" hOwner MessageBoxA
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.MessageBox>	
004013C1	. C3	RET	

-Kiểm tra vị trí của badboy, ta thấy dòng code print badboy window ở vị trí **00401362** được gọi từ dòng **00401245** và vị trí **004013AC** được jump từ dòng **0040138B**.

-Ta tiến hành đặt breakpoint ở trước các vị trí call goodboy và badboy ở điều kiện đủ là dòng **00401245**, **0040138B**.

-Thử debug với username là abc và password là 123:

00401228	. 68 8E214000	PUSH 1_3.0040218E	ASCII "ABC"
0040122D	. E8 4C010000	CALL 1_3.0040137E	
00401232	. 50	PUSH EAX	
00401233	. 68 7E214000	PUSH 1_3.0040217E	ASCII "123"
00401238	. E8 9B010000	CALL 1_3.004013D8	
0040123D	. 83C4 04	ADD ESP,4	
00401240	. 58	POP EAX	

-> Name input được lưu ở 0040218E, Serial input được lưu ở 0040217E và dòng 0040137E là vị trí xử lý Name, dòng 004013D8 là vị trí xử lý Serial.

00401228	. 68 8E214000	PUSH 1_3.0040218E	ASCII "ABC"
0040122D	. E8 4C010000	CALL 1_3.0040137E	
00401232	. 50	PUSH EAX	
00401233	. 68 7E214000	PUSH 1_3.0040217E	ASCII "123"
00401238	. E8 9B010000	CALL 1_3.004013D8	
0040123D	. 83C4 04	ADD ESP,4	
00401240	. 58	POP EAX	
00401241	. 3BC3	CMP EAX,EBX	
00401243	. 74 07	JE SHORT 1_3.0040124C	
00401245	. E8 10010000	CALL 1_3.00401362	
0040124A	. EB 9A	JMP SHORT 1_3.004011E6	
0040124C	. E8 FC000000	CALL 1_3.00401340	
00401251	. EB 93	JMP SHORT 1_3.004011E6	

-> Sau khi xử lý, chương trình so sánh 2 chuỗi vừa thua được. Nếu giống nhau thì *jump* đến dòng **0040124C** -> **0040134D** (in ra goodboy ). Nếu khác nhau thì *jump* đến dòng **00401362** để in ra badboy...

- Truy vết username tại dòng **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	
00401389	. 3C 41	CMPL AL,41	
0040138B	. 72 1F	JB SHORT 1_3.004013AC	
0040138D	. 3C 5A	CMPL 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	

- Ta thấy từ **00401383** đến **0040139A**: tiến hành kiểm tra các ký tự nhập vào có mã ASCII bé hơn 65 hay không. Nếu có thì nhảy đến badboy thứ 2 **004013AC**

- Đoạn code ở dòng **004013C2** tiến hành cộng tất cả các ký tự đã được kiểm tra và upcase hoặc trừ đi 20 trong mã ASCII của nó, ở phía trước lại với nhau.

```

33FF XOR EDI,EDI
33DB XOR EBX,EBX
> 8A1E MOV BL,BYTE PTR [ESI]
84DB TEST BL,BL
v 74 05 JE SHORT 1_3.004013D1
03FB ADD EDI,EBX
46 INC ESI
^ EB F5 JMP SHORT 1_3.004013C6
> C3 RET

```

- Cuối cùng tiến hành XOR biến vừa thu được với 0x5678 để thu được kết quả sau cùng và trở về vị trí trước đó để xử lý serial key ở dòng **004013A2**

- **Xử lý Serial:**

Ta tiến hành kiểm tra tại vị trí 004013D8 và thấy được đoạn code xử lý serial như sau:

```

004013D7 . C3 RET
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 v 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
004013FF . FF25 84314000 JMP DWORD PTR [<&MSFR32.KillTimer>]

```

➔ Trong đó đoạn code từ 004013E2 đến 004013F5 thực hiện convert chuỗi serial key thành số, sau đó XOR với **0x1234** ở dòng **004013F5**.

### 6.3 Viết keygen:

Ta rõ ràng có thể thấy tương ứng 1 name chỉ có 1 serial và name phải là chữ cái. Và serial có thể được generate từ name bằng cách lợi dụng tính chất của phép XOR. Thay vì đem serial đem xor với 0x1234. Ta tiến hành như sau:

- Kiểm tra các ký tự của name có mã ASCII < 41<sub>16</sub> hay không, nếu có thì báo lỗi và yêu cầu nhập lại. Nếu đúng thì lấy mã ASCII của ký tự đó trừ đi 20.
- Đem cộng giá trị ascii các ký tự của name lại với nhau
- Đem xor kết quả với 0x5678 .
- Đem xor kết quả tiếp với 0x1234 .
- Kết quả thu được chính là serial ở dạng số.



---

## CHƯƠNG 7: THAM KHẢO

---

Slide bài giảng của thầy Viet Long về các cracking tutorial.

Tham khảo thêm kiến thức và fix bug ở các trang

[www.stackoverflow.com](http://www.stackoverflow.com)

<https://daynhauhoc.com>

<http://www.cs.virginia.edu/>

<https://www.eecg.utoronto.ca/>

<https://doc.dpdk.org/>