



BÁO CÁO ĐỒ ÁN 3

KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ



Mục lục

I.	Thông tin nhóm.....	2
II.	Phân công công việc.....	2
III.	Mô tả từng bước thuật toán phát sinh key	3
1.	File Crackme 1.1	3
2.	File Crackme 2.1	5
3.	File Crackme 2.2	7
1.	Tiền xử lý	7
2.	Chương trình 1	9
3.	Chương trình 2	11
4.	Chương trình 3	13
5.	Chương trình keygen.....	15
4.	File crackme 3.1	16
5.	Bài tập trên trang microcorruption.com	21
1.	Level Tutorial.....	21
2.	Level New Orleans.....	22
3.	Sydney.....	22
4.	Hanoi.....	23
5.	Cusco.....	23
6.	Whitehorse	24
IV.	Tài liệu tham khảo	25

I. Thông tin nhóm

- MSSV: 1612348
- MSSV: 1612362
- MSSV: 1612756

Họ tên: Lý Vĩnh Lợi

Họ tên: Trần Văn Lược

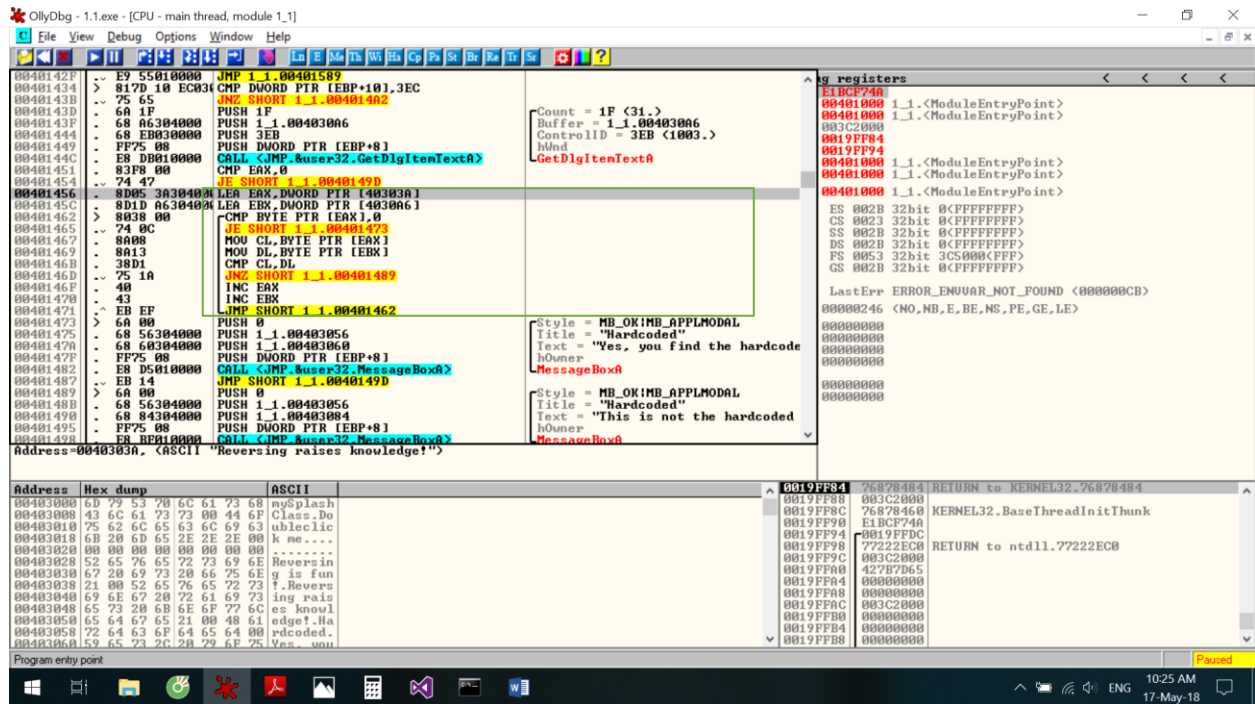
Họ tên: Nguyễn Hữu Trường

II. Phân công công việc

Công việc	Người phụ trách	Mức độ hoàn thành (%)
Crackme 1.1	Trường	100
Crackme 2.1	Trường	100
Crackme 2.2	Lợi	100
Crackme 3.1	Lược	100
Bài tập trên microcorruption	Lợi	100

III. Mô tả từng bước thuật toán phát sinh key

1. File Crackme 1.1



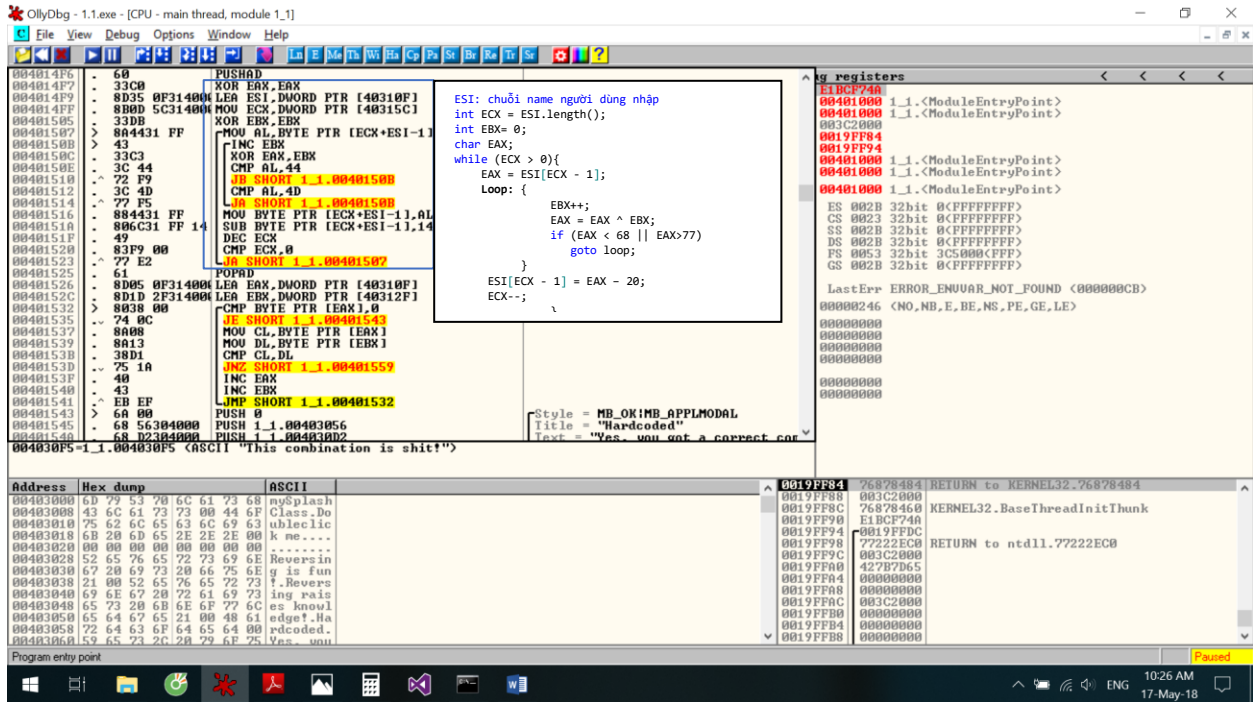
1. Check Hardcoded

EAX chứa chuỗi khóa cố định

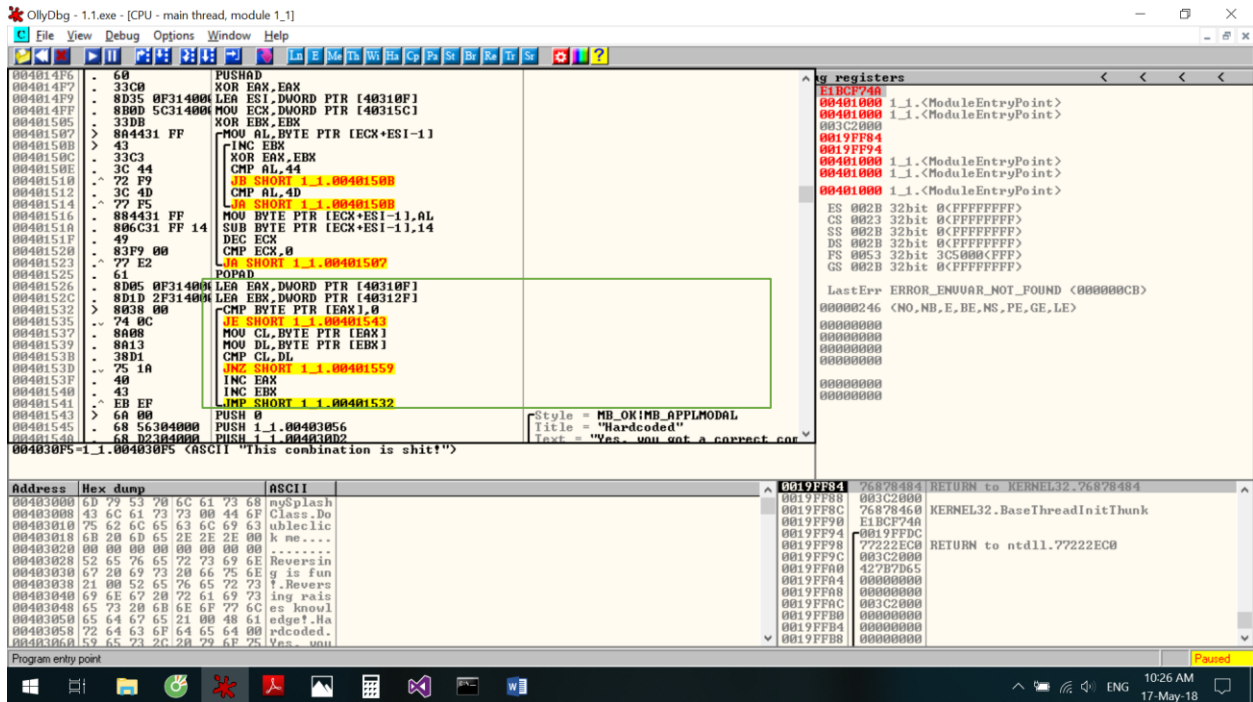
EBX chứa chuỗi hardcoded người dùng nhập vào

Nếu EAX=EBX thì chuỗi hardcoded đúng

Kết quả: EBX = “Reversing raiser knowledge!”



2. Tạo serial từ name người dùng nhập vào



3. Check serial do thuật toán tạo ra và serial do người dùng nhập vào

EAX: chuỗi serial do thuật toán tạo ra

EBX: chuỗi serial do người dùng nhập vào

OnlyDbg 2.1.0x [CPU - main thread, module 2]

File View Debug Options Window Help

00401242 - 6A 0C PUSH 0C
 00401244 - 6A 05 PUSH 5
 00401246 - FF75 00 PUSH DWORD PTR [EBP+8]
 00401249 - 00B10000 **CALL JMP_Euser32.SendDigitenMessageA**
 0040124E - 61 POPAD
 0040124F - C9 LEAVE
 00401250 - C2 0400 RET 4
 00401253 BE 8B624000 MOV ESI, 2_1.004062E8
 00401258 BF C8624000 MOV EDI, 2_1.004062C0
 0040125D B0 74624000 MOV EDX, 2_1.00406274
 00401262 B7 04000000 MOV ECK, 4
 00401267 > 85C9 TEST ECK, ECK
 00401269 > 74 14 JE SHORT 2_1.0040127F
 0040126B - 8A06 MOV AL, BYTE PTR [ESI]
 0040126D - 8823 MOV AH, BYTE PTR [EDI]
 0040126F - 32C4 XOR AL, AH
 00401271 - 24 5F AND AL, 5F
 00401273 - 0C 40 OR AL, 40
 00401275 - 34 09 XOR AL, 9
 00401277 - 8B07 MOV BYTE PTR [EDI], AL
 00401279 - 46 INC ESI
 0040127A - 47 INC EDI
 0040127B - 43 DEC ECK
 0040127C - 42 INC EDX
 0040127D - EB E8 JMP SHORT 2_1.00401267
 0040127F > C607 24 MOV BYTE PTR [EDI], 24
 00401282 C3 RET
 00401283 BE 74624000 MOV ESI, 2_1.00406274
 00401288 BF C8624000 MOV EDI, 2_1.004062C0
 0040128D B0 74624000 MOV EDX, 2_1.00406274
 00401292 83C7 05 ADD ESI, 5
 00401295 B7 04000000 MOV ECK, 4
 004062E8 - 2_1.004062E8 <ASCII "enter_name_here">
 ESI - 000F023C
 Local call from 0040120E

Message = WM_SETEXT
 ControlId = 65 (101.)
 hWnd
 SendDigitenMessageA

Registers (FPU)
 EAX 004062E8 ASCII "enter_name_here"
 ECX 758F823A KERNEL32.758F823A
 EDX 00000010
 EIP 00000001
 ESP 0019F890
 EBP 0019F888
 ESI 000F023C
 EDI 80006010
 EIP 00401253 2_1.00401253
 C 0 ES 002B 32bit 0(FFFFFFFF)
 P 1 CS 0023 32bit 0(FFFFFFFF)
 0 0 SS 002B 32bit 0(FFFFFFFF)
 Z 1 DS 002B 32bit 0(FFFFFFFF)
 S 0 FS 0053 32bit 32(0000FFFF)
 T 0 GS 002B 32bit 0(FFFFFFFF)
 D 0
 O 0 LastErr ERROR_SUCCESS (00000000)
 EPL 00000246 <NO, NB, E, BE, NS, PE, GE, LE>
 ST0 empty 0.0
 ST1 empty 32512.0000000000000000
 ST2 empty -12861.248281185041500
 ST3 empty 0.0
 ST4 empty 8653.8041305541992190
 ST5 empty 15.000000000000000000
 ST6 empty 16.000000000000000000
 ST7 empty 16.000000000000000000
 FST 4020 Cond 1 0 0 0 Err 0 1 0 0 0 <EQ>
 FCW 837F Prec NEAR, 64 Mask 1 1 1 1 1

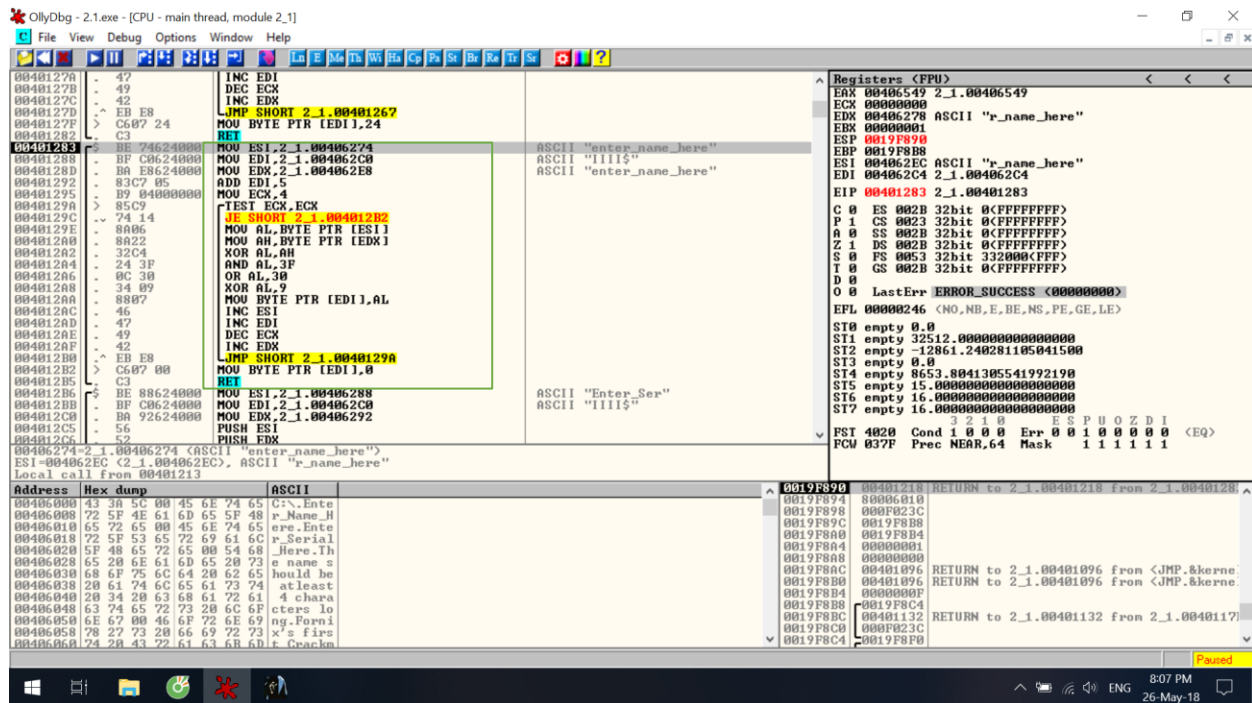
Address	Hex dump	ASCII
00406000	43 38 5C 00 45 6E 74 65	C:\Ente
00406008	72 5F 4E 61 6D 65 5F 48	r_Name_H
00406010	65 72 65 00 45 6E 74 65	re.Ente
00406018	72 5F 53 65 72 69 61 6C	p.Serial
00406020	5F 48 65 72 65 00 54 68	Here.Th
00406028	65 20 6E 61 6D 65 20 73	e name s
00406030	68 6F 75 6C 64 20 62 65	ould be
00406038	20 61 74 6C 65 61 73 74	atleast
00406040	24 20 63 68 61 72 73 64	4 chara
00406048	63 74 65 72 73 20 6C 6F	cters lo
00406050	6E 67 00 46 6F 72 6E 69	ng.Forni
00406058	78 27 73 20 66 69 72 73	x's fire
00406060	74 20 43 72 61 63 6B 6D	t Cracke

0019F890 00401213 RETURN to 2_1.00401213 from 2_1.0040125A
 0019F894 00006010
 0019F89C 0019F888
 0019F8A0 0019F8B8
 0019F8A8 0019F8D8
 0019F8B0 00000001
 0019F8B8 00000000
 0019F8C0 00401096 RETURN to 2_1.00401096 from <JMP.8.kerne
 0019F8C8 00401096 RETURN to 2_1.00401096 from <JMP.8.kerne
 0019F8D0 0019F8D4
 0019F8D8 0019F8C4
 0019F8E0 00401132 RETURN to 2_1.00401132 from 2_1.0040117F
 0019F8C0 000F023C
 0019F8C4 0019F8F0

8:07 PM
26-May-18

EDI và EDX đều trỏ đến chuỗi tên người dùng nhập vào.

⇒ **Chuỗi khóa do hàm trên tạo ra là: “IIIS\$”**

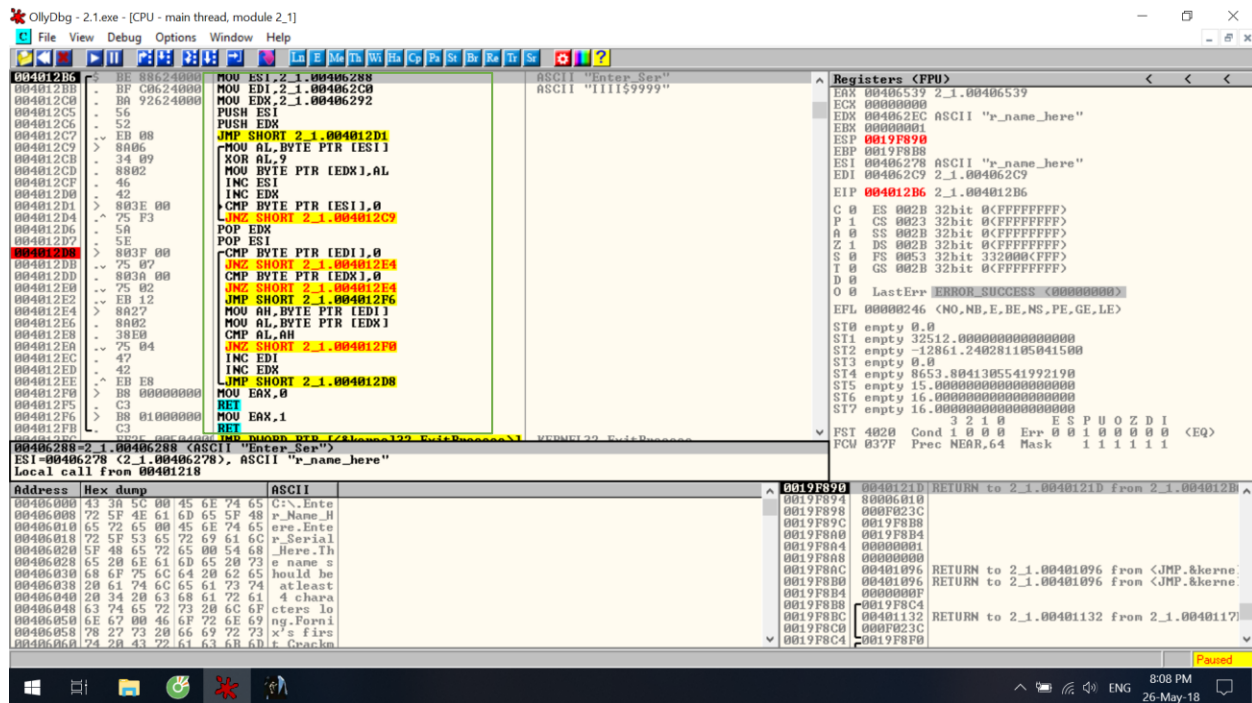


5. Sinh chuỗi khóa từ chuỗi tên do người dùng nhập vào (hàm 2).

EDI và EDX đều trỏ đến chuỗi tên người dùng nhập vào.

Đoạn code chính: $EDI = (((ESI \wedge EDX) \& 63) \mid 48) \mid 9(\text{dec}) = 48 \mid 9 = '9'$

⇒ Chuỗi khóa do 2 hàm trên tạo ra là: “IIII\$9999”



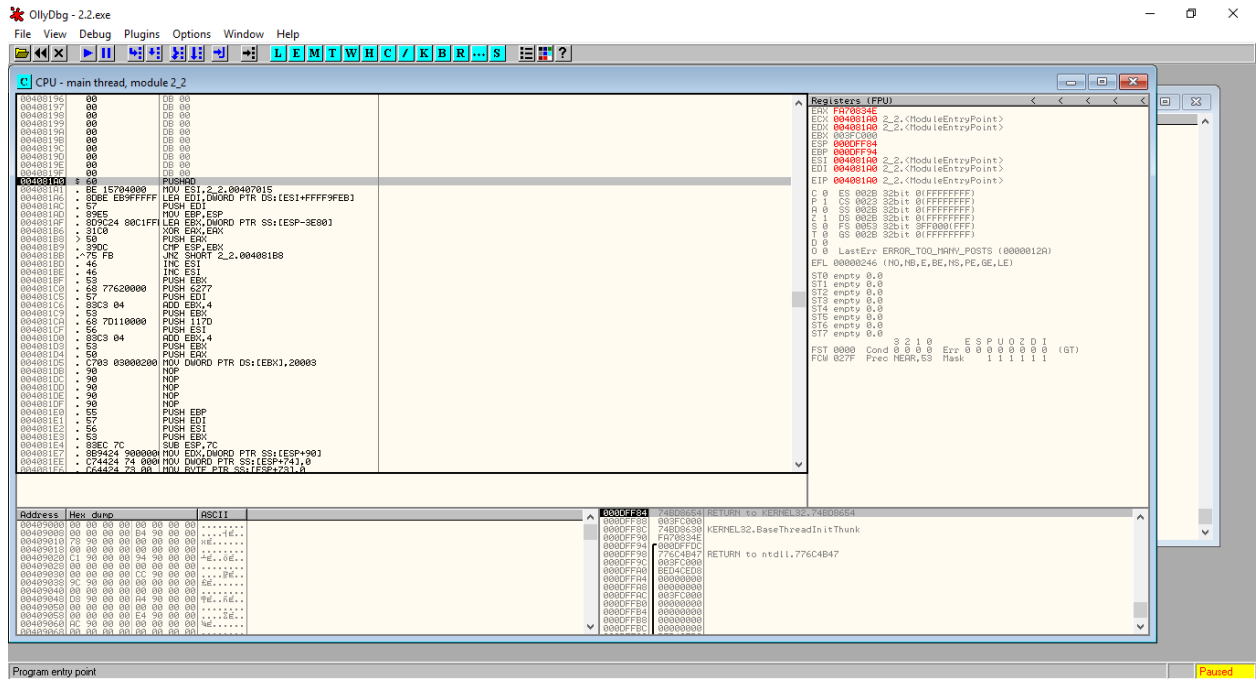
6. Mã hóa tối đa 9 kí tự đầu serial và so sánh chuỗi mã hóa với chuỗi khóa do 2 hàm sinh khóa tạo ra.

- Mã hóa tối đa 9 kí tự đầu serial bằng cách dùng toán tử XOR: XOR AL, 9
 - So sánh chuỗi mã hóa với chuỗi khóa, nếu đúng thì serial đúng, nếu khác thì serial sai
- ⇒ Serial “@@@@-0000” luôn đúng với mọi chuỗi tên người dùng nhập vào.

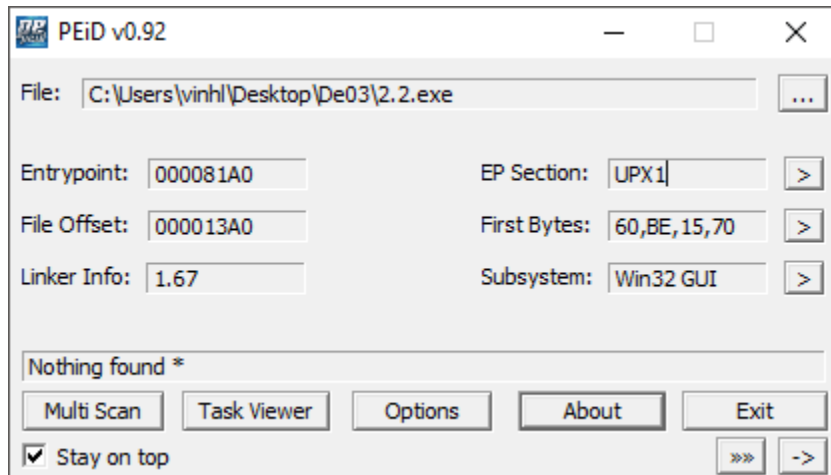
3. File Crackme 2.2

1. Tiền xử lý

- File 2.2.exe ban đầu đã được nén, vì sau khi run trong OllyDbg không xuất hiện thông tin của từng dòng lệnh



- Sau đó đưa vào PEID để kiểm tra



- Sử dụng UPX [1] để giải nén chương trình

```

C:\Windows\System32\cmd.exe

C:\Users\vinhl\Desktop\upx394w>upx -d -oUnpacked.exe C:\Users\vinhl\Desktop\De03\2.2.exe
Ultimate Packer for executables
Copyright (C) 1996 - 2017
UPX 3.94w Markus Oberhumer, Laszlo Molnar & John Reiser May 12th 2017

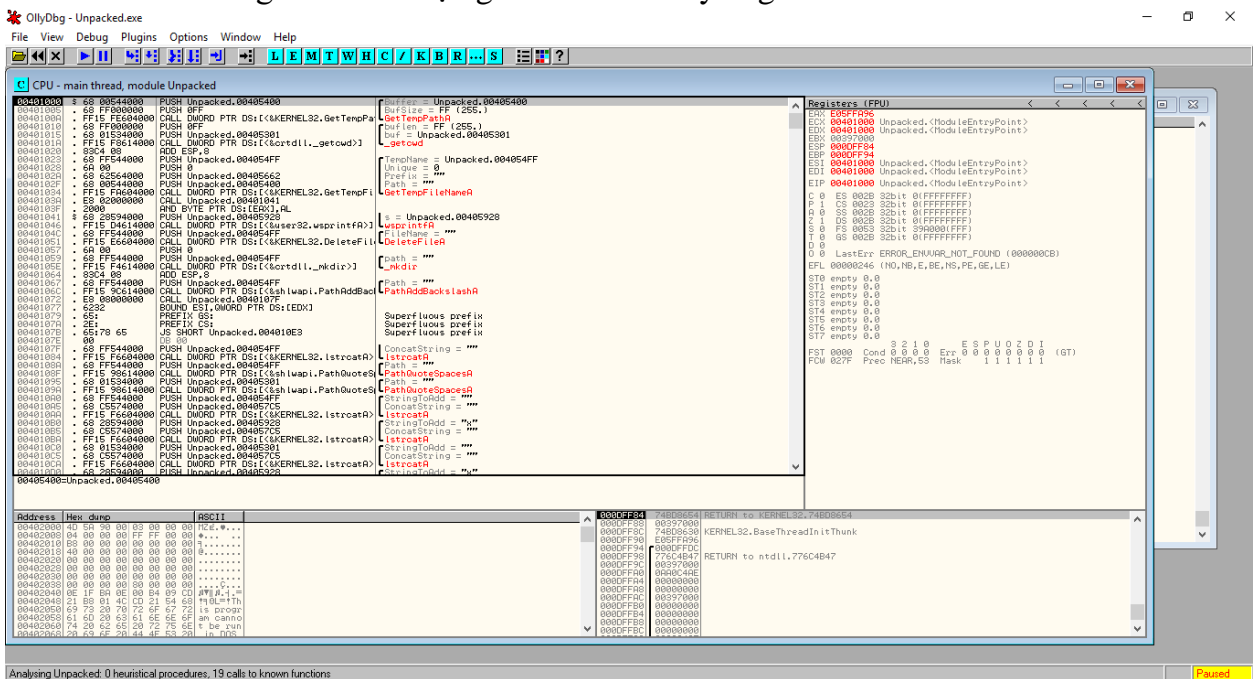
-----
File size      Ratio      Format      Name
-----
16896 <-      8704      51.52%     win32/pe     Unpacked.exe

Unpacked 1 file.

C:\Users\vinhl\Desktop\upx394w>

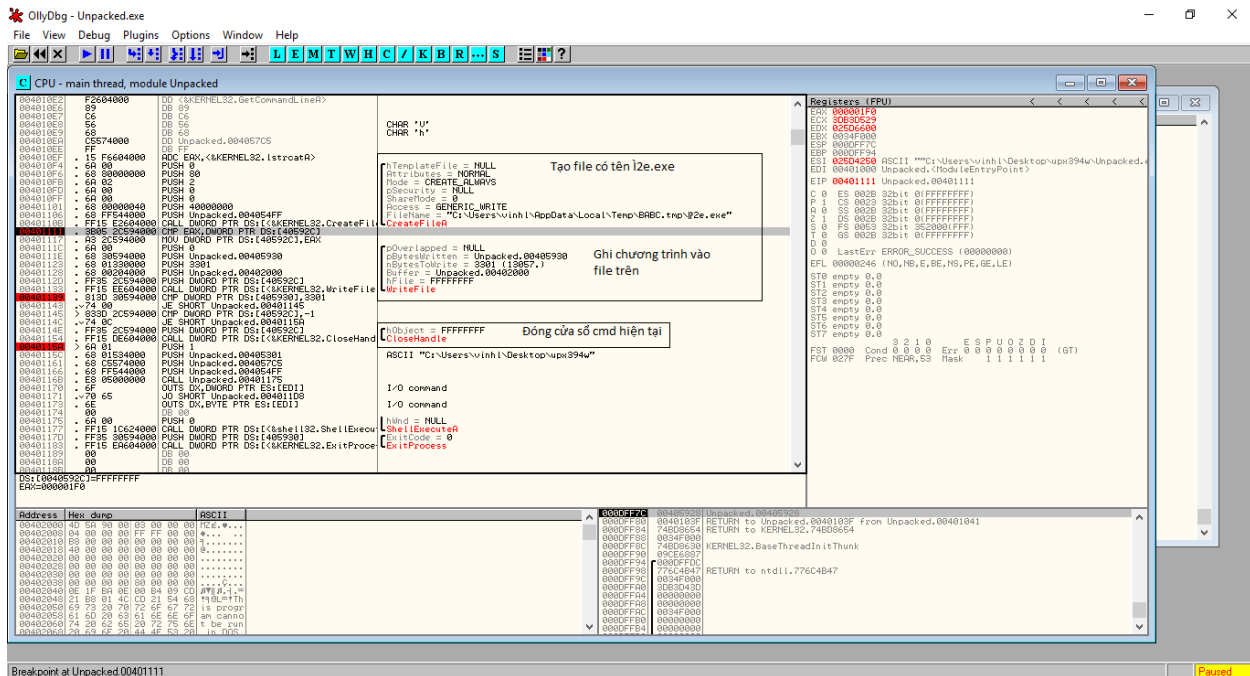
```

- Sau đó đưa chương trình đã được giải nén vào OllyDbg và run

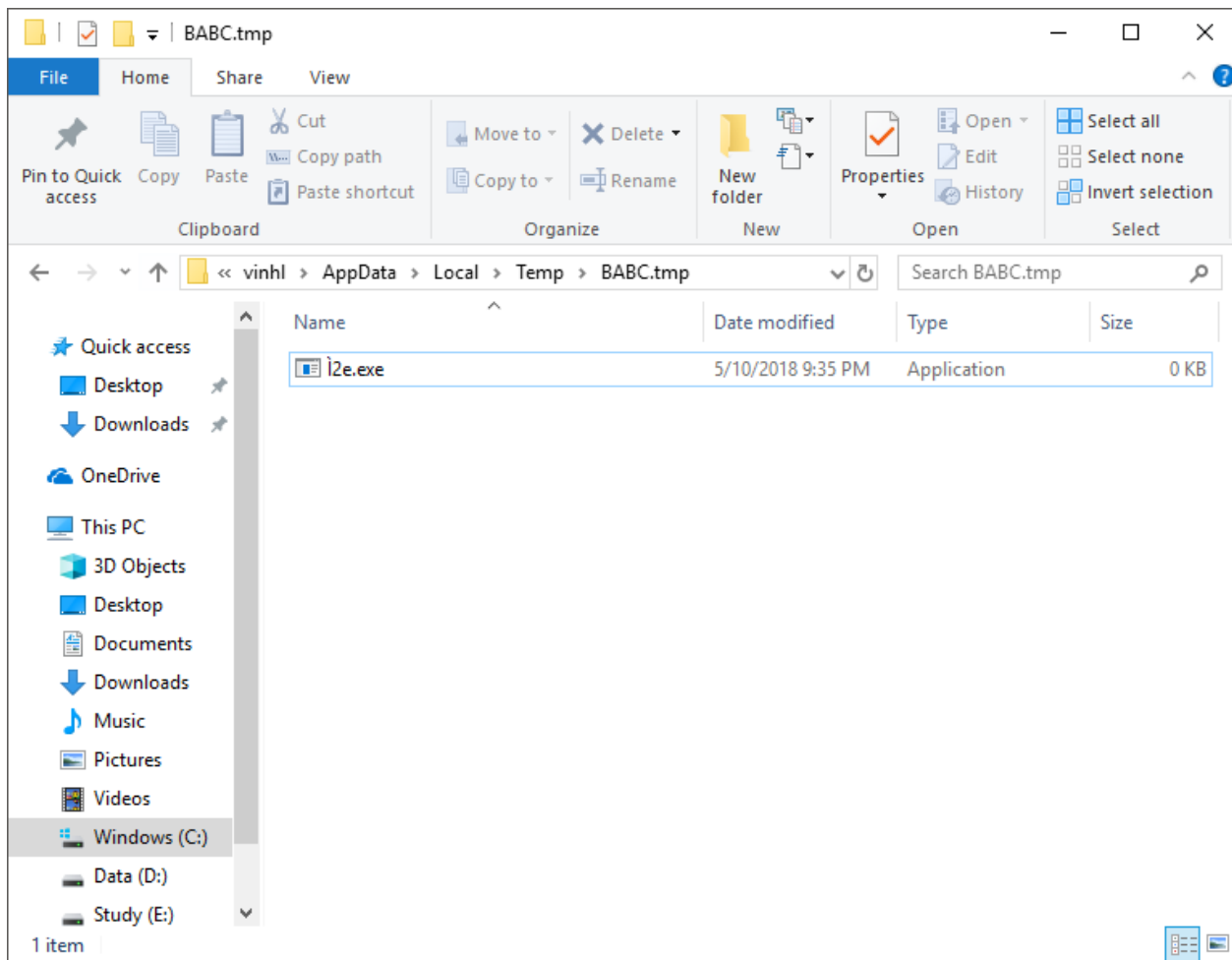


2. Chương trình 1

Để nhanh chóng, đặt các breakpoint vào sau các dòng lệnh được gom nhóm và chạy (F9), ta sẽ thấy sự thay đổi của đường dẫn



10

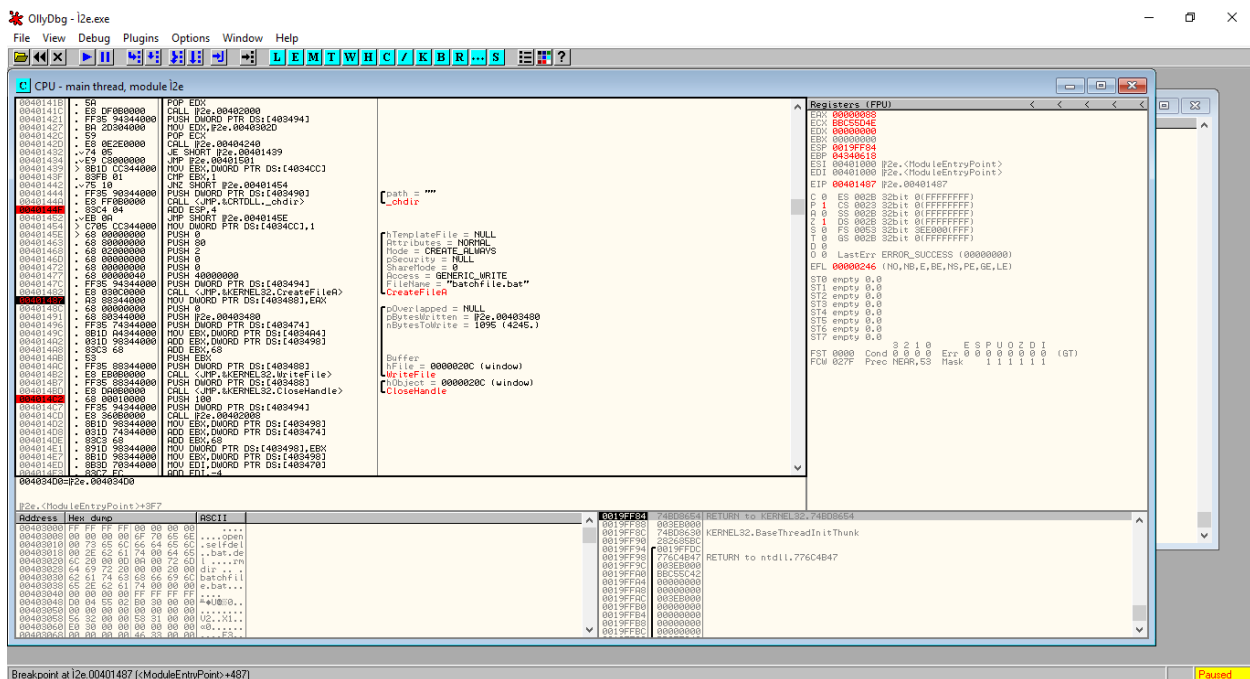
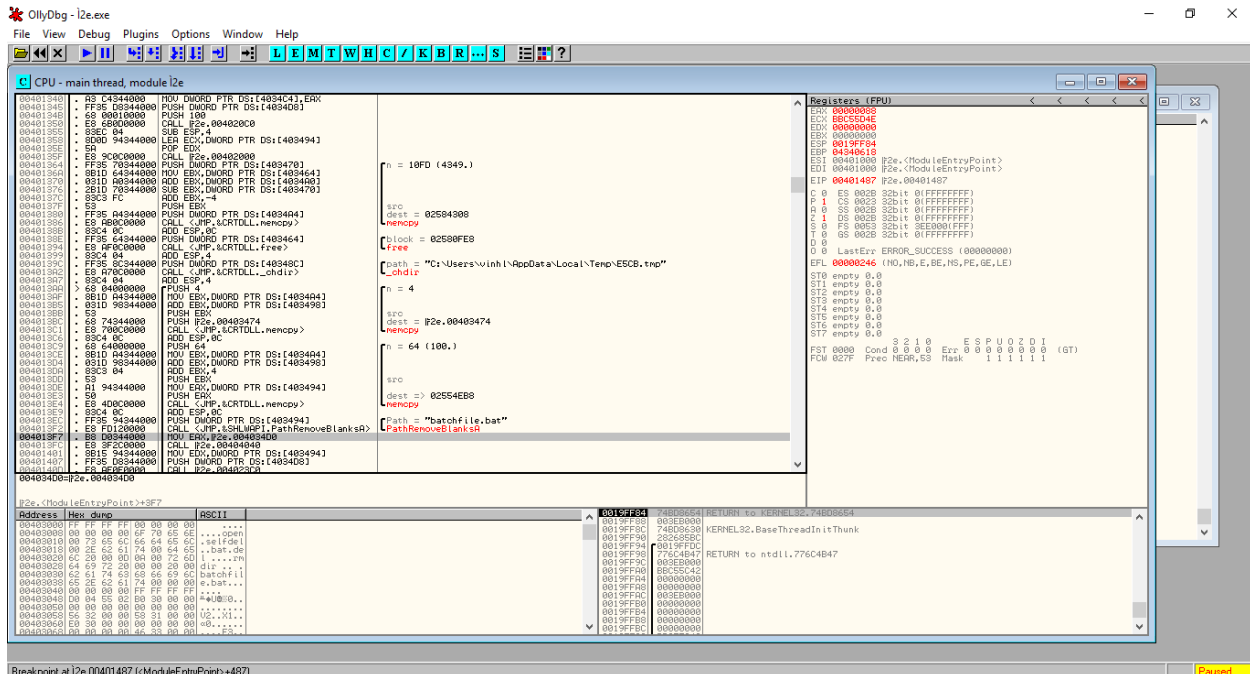


Để khi chương trình chính kết thúc file trên vẫn còn, copy file trên ra một thư mục (chỉ copy được sau khi chạy chương trình đến breakpoint tại dòng 0040115A)

3. Chương trình 2

Dùng OllyDbg mở chương trình được sinh ra ở trên

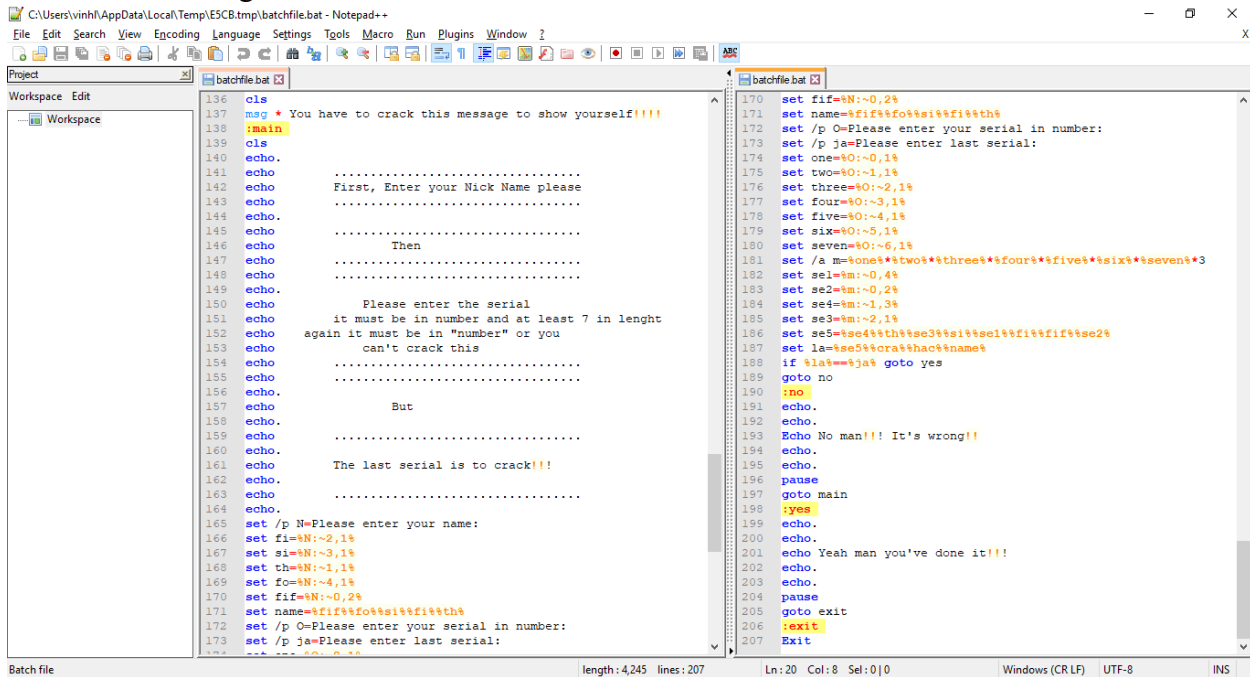
Tương tự ta đặt các breakpoint sau các nhóm lệnh quan trọng (WriteFile, CloseHandle...)



Sau đó, vào đường dẫn trong thư mục Temp của User tương tự như trên, ta tìm được file batchfile.bat. Ta copy file này ra một thư mục khác (Lưu ý chỉ copy được sau khi chạy đến breakpoint tại dòng 004014C2)

Đây là file nguồn chính của chương trình được viết bằng BashScript của Windows, nên có thể mở lên rõ ràng bằng trình soạn thảo thông thường (ví dụ Notepad, Notepad++,...)

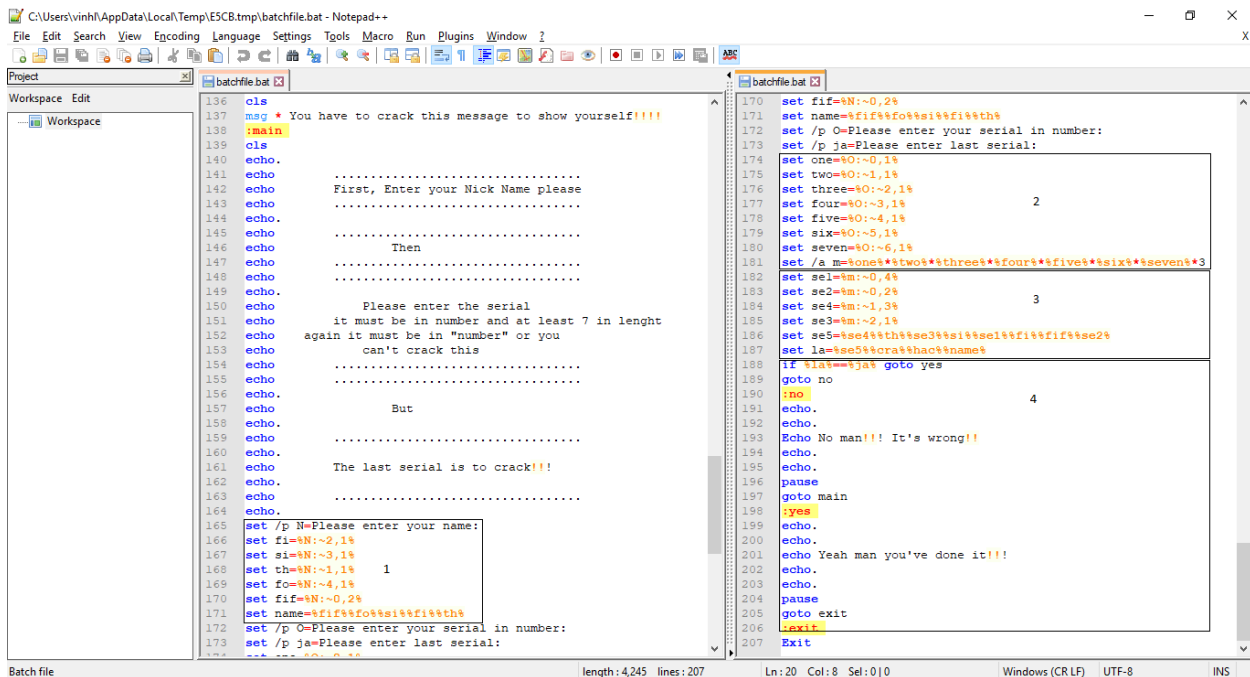
4. Chương trình 3



```
136 cls
137 msg * You have to crack this message to show yourself!!!!
138 :main
139 cls
140 echo.
141 echo.
142 echo. First, Enter your Nick Name please
143 echo. ....
144 echo.
145 echo. ....
146 echo. Then
147 echo. ....
148 echo.
149 echo. ....
150 echo. Please enter the serial
151 echo. it must be in number and at least 7 in lenght
152 echo. again it must be in "number" or you
153 echo. can't crack this
154 echo. ....
155 echo.
156 echo. But
157 echo. ....
158 echo.
159 echo. ....
160 echo. The last serial is to crack!!!
161 echo.
162 echo. ....
163 echo.
164 echo. ....
165 set /p N=Please enter your name:
166 set fi=%N:~2,1%
167 set si=%N:~3,1%
168 set th=%N:~4,1%
169 set fo=%N:~5,1%
170 set fif=%N:~6,2%
171 set name=%fif%%fo%%si%%fi%%th%
172 set /p O=Please enter your serial in number:
173 set /p ja=Please enter last serial:
174
175 set one=%O:~0,1%
176 set two=%O:~1,1%
177 set three=%O:~2,1%
178 set four=%O:~3,1%
179 set five=%O:~4,1%
180 set six=%O:~5,1%
181 set seven=%O:~6,1%
182 set /a m=%one%*two%*three%*four%*five%*six%*seven%*3
183 set se1=%m:~0,4%
184 set se2=%m:~0,2%
185 set se4=%m:~1,3%
186 set se3=%m:~2,1%
187 set se5=%se4%%th%%se3%%si%%se1%%fi%%fif%%se2%
188 set la=%se5%crack%%name%
189 if %la%==%ja% goto yes
190 goto no
191 :no
192 echo.
193 Echo No man!!! It's wrong!!
194 echo.
195 echo.
196 pause
197 goto main
198 :yes
199 echo.
200 echo. Yeah man you've done it!!!
201 echo.
202 echo.
203 pause
204 goto exit
205 :exit
206 Exit
207
```

Theo định nghĩa các lệnh Bash của Windows

- echo: Xuất ra màn hình
- set: Đặt <tên biến>=<giá trị>
- Biến sau khi đặt bằng set, chỗ nào cần sử dụng ta ghi %<tên biến>%
- Có thể có một số thao tác xử lý (đặc biệt là chuỗi)



```
136 cls
137 msg * You have to crack this message to show yourself!!!!
138 :main
139 cls
140 echo.
141 echo.
142 echo. First, Enter your Nick Name please
143 echo. ....
144 echo.
145 echo. ....
146 echo. Then
147 echo. ....
148 echo.
149 echo. ....
150 echo. Please enter the serial
151 echo. it must be in number and at least 7 in lenght
152 echo. again it must be in "number" or you
153 echo. can't crack this
154 echo. ....
155 echo.
156 echo. But
157 echo. ....
158 echo.
159 echo. ....
160 echo. The last serial is to crack!!!
161 echo.
162 echo. ....
163 echo.
164 echo. ....
165 set /p N=Please enter your name:
166 set fi=%N:~2,1%
167 set si=%N:~3,1%
168 set th=%N:~4,1%
169 set fo=%N:~5,1%
170 set fif=%N:~6,2%
171 set name=%fif%%fo%%si%%fi%%th%
172 set /p O=Please enter your serial in number:
173 set /p ja=Please enter last serial:
174
175 set one=%O:~0,1%
176 set two=%O:~1,1%
177 set three=%O:~2,1%
178 set four=%O:~3,1%
179 set five=%O:~4,1%
180 set six=%O:~5,1%
181 set seven=%O:~6,1%
182 set /a m=%one%*two%*three%*four%*five%*six%*seven%*3
183 set se1=%m:~0,4%
184 set se2=%m:~0,2%
185 set se4=%m:~1,3%
186 set se3=%m:~2,1%
187 set se5=%se4%%th%%se3%%si%%se1%%fi%%fif%%se2%
188 set la=%se5%crack%%name%
189 if %la%==%ja% goto yes
190 goto no
191 :no
192 echo.
193 Echo No man!!! It's wrong!!
194 echo.
195 echo.
196 pause
197 goto main
198 :yes
199 echo.
200 echo. Yeah man you've done it!!!
201 echo.
202 echo.
203 pause
204 goto exit
205 :exit
206 Exit
207
```

Dò ng	Giải thích	Ví dụ
165	Nhập username từ bàn phím và lưu vào biến N	Nếu nhập là “username” thì biến N sẽ có giá trị là “username” (không có dấu “”)
166	Đặt biến fi = %N:~2,1% nghĩa là bỏ qua 2 kí tự từ trái sang, lấy 1 kí tự trong biến N	fi = e
167	Đặt biến si = %N:~3,1% nghĩa là bỏ qua 3 kí tự từ trái sang, lấy 1 kí tự trong biến N	si = r
168	Đặt biến th = %N:~1,1% nghĩa là bỏ qua 1 kí tự từ trái sang, lấy 1 kí tự trong biến N	th = s
169	Đặt biến fo = %N:~4,1% nghĩa là bỏ qua 4 kí tự từ trái sang, lấy 1 kí tự trong biến N	fo = n
170	Đặt biến fif = %N:~0,2% nghĩa là bỏ qua 0 kí tự từ trái sang, lấy 2 kí tự trong biến N	fif = us
171	Đặt biến name=%fif%%fo%%si%%fi%%th% nghĩa là nối các chuỗi trên theo thứ tự tương ứng và lưu vào biến name	name = usnres
	<i>Lưu ý: Nếu username nhỏ hơn 5 kí tự thì kết quả truy xuất vượt ngoài phạm vi (biến fo chẳng hạn) sẽ trả về là chuỗi rỗng</i>	
172	Nhập serial và lưu vào biến O	123456789
173	Nhập key và lưu vào biến ja	<Key này chỉ có giá trị so sánh, chúng ta cần sinh ra key này từ thuật toán>
174 – 180	Tương tự trên nhưng lấy từng kí tự một của chuỗi serial (biến O) set one=%O:~0,1% set two=%O:~1,1% set three=%O:~2,1% set four=%O:~3,1% set five=%O:~4,1% set six=%O:~5,1% set seven=%O:~6,1%	One = 1 Two = 2 Three = 3 Four = 4 Five = 5 Six = 6 Seven = 7
181	set /a m=%one%%two%%three%%four%%five% %%six%%seven%*3 lưu vào biến m giá trị của các số trên (one, two, ... seven) theo công thức m=one*two*three*four*five*six*seven*3	M = 1*2*3*4*5*6*7*3 = 15120
182	set se1=%m:~0,4% Lấy 4 kí tự đầu của biến m	Se1=1512

183	set se2=%m:~0,2% Lấy 2 kí tự đầu của biến m	Se2=15
184	set se4=%m:~1,3% Bỏ qua kí tự đầu, lấy 3 kí tự tiếp theo	Se4=512
185	set se3=%m:~2,1% Bỏ qua 2 kí tự đầu, lấy 1 kí tự tiếp theo	Se3=1
186	set se5=%se4%%th%%se3%%si%%se1%%fi%%fif %%se2% Nối các chuỗi lại theo công thức	Se5=512s1r1512eus15
187	set la=%se5%%cra%%hac%%name% Nối chuỗi theo công thức	La=512s1r1512eus15admirable154usnres
188	So sánh chuỗi serial nhập vào (lưu tại biến ja) và chuỗi key được sinh ra (lưu tại biến la), in thông báo và kết thúc	

```

.....
Then
.....

Please enter the serial
it must be in number and at least 7 in lenght
again it must be in "number" or you
can't crack this
.....

But
.....

The last serial is to crack!!!
.....

Please enter your name: username
Please enter your serial in number: 123456789
Please enter last serial:512s1r1512eus15admirable154usnres

Yeah man you've done it!!!

Press any key to continue . . .

```

5. Chương trình keygen

Dựa vào các thao tác trên, ta có thể viết được chương trình keygen:


```
C:\WINDOWS\system32\cmd.exe
Enter username:
abc
Enter serial (must be number and 7 digits or more):
217438
Enter serial (must be number and 7 digits or more):
2174152
Key: 680b81680cab16admirable154abcb
Press any key to continue . . .

Crackme2.exe

.....
                Then
.....

        Please enter the serial
it must be in number and at least 7 in lenght
again it must be in "number" or you
        can't crack this
.....

                But
.....

The last serial is to crack!!!
.....

Please enter your name: abc
Please enter your serial in number: 2174152
Please enter last serial:680b81680cab16admirable154abcb

Yeah man you've done it!!!

Press any key to continue . . .
```

4. File crackme 3.1

Toàn bộ biến, thanh ghi, địa chỉ hàm được tham chiếu từ lần chạy chương trình lúc thực hiện báo cáo. Địa chỉ của hàm được xem như tên hàm.

- a. Bằng cách dung tính năng search for all text string của Olly, ta tìm được Goodboy của file exe này. Qua đó, ta dễ thấy được hàm dưới đây được dùng để phát sinh key và kiểm tra tính chính xác dự vào Username mà người dùng nhập vào:

0040239E	. 55	PUSH EBP	
0040239F	. 8BEC	MOV EBP,ESP	
004023A1	. 83EC 74	SUB ESP,74	
004023A4	. 894D 8C	MOV DWORD PTR [EBP-74],ECX	
004023A7	. C745 A4 0000	MOV DWORD PTR [EBP-5C],0	
004023AE	. C745 FC 0000	MOV DWORD PTR [EBP-4],0	
004023B5	. 6A 11	PUSH 11	
004023B7	. 8D45 90	LEA EAX,DWORD PTR [EBP-70]	
004023BA	. 50	PUSH EAX	
004023BB	. 68 E8030000	PUSH 3E8	
004023C0	. 8B4D 8C	MOV ECX,DWORD PTR [EBP-74]	
004023C3	. E8 64370000	CALL <JMP.&MFC42.#3098>	
004023C8	. 8945 A4	MOV DWORD PTR [EBP-5C],EAX	
004023CB	. 6A 19	PUSH 19	
004023CD	. 8D4D DC	LEA ECX,DWORD PTR [EBP-24]	
004023D0	. 51	PUSH ECX	
004023D1	. 68 E9030000	PUSH 3E9	
004023D6	. 8B4D 8C	MOV ECX,DWORD PTR [EBP-74]	
004023D9	. E8 4E370000	CALL <JMP.&MFC42.#3098>	
004023DE	. 8945 FC	MOV DWORD PTR [EBP-4],EAX	
004023E1	. 837D A4 06	CMP DWORD PTR [EBP-5C],6	
004023E5	.. 7E 65	JLE SHORT 3_1.0040244C	
004023E7	. 837D FC 04	CMP DWORD PTR [EBP-4],4	
004023EB	.. 7E 5F	JLE SHORT 3_1.0040244C	
004023ED	. 8D55 A8	LEA EDX,DWORD PTR [EBP-58]	
004023F0	. 52	PUSH EDX	
004023F1	. 8D45 DC	LEA EAX,DWORD PTR [EBP-24]	
004023F4	. 50	PUSH EAX	
004023F5	. E8 D6370000	CALL <JMP.&MSUCRT.strlen>	
004023FA	. 83C4 04	ADD ESP,4	
004023FD	. 50	PUSH EAX	
004023FE	. 8D4D DC	LEA ECX,DWORD PTR [EBP-24]	
00402401	. 51	PUSH ECX	
00402402	. E8 49F0FFFF	CALL 3_1.00401450	
00402407	. 83C4 0C	ADD ESP,0C	
0040240A	. C645 B8 00	MOV BYTE PTR [EBP-48],0	
0040240E	. 8D55 C8	LEA EDX,DWORD PTR [EBP-38]	
00402411	. 52	PUSH EDX	
00402412	. 8D45 A8	LEA EAX,DWORD PTR [EBP-58]	
00402415	. 50	PUSH EAX	
00402416	. 8D4D 90	LEA ECX,DWORD PTR [EBP-70]	
00402419	. 51	PUSH ECX	
0040241A	. E8 EAF5FFFF	CALL 3_1.00401A09	
0040241F	. 83C4 0C	ADD ESP,0C	
00402422	. 6A 10	PUSH 10	
00402424	. 8D55 C8	LEA EDX,DWORD PTR [EBP-38]	
00402427	. 52	PUSH EDX	
00402428	. 8D45 90	LEA EAX,DWORD PTR [EBP-70]	
0040242B	. 50	PUSH EAX	
0040242C	. E8 B7370000	CALL <JMP.&MSUCRT.memcmp>	
00402431	. 83C4 0C	ADD ESP,0C	
00402434	. 85C0	TEST EAX,EAX	
00402436	.. 75 14	JNZ SHORT 3_1.0040244C	
00402438	. 6A 00	PUSH 0	
0040243A	. 68 28804000	PUSH 3_1.00408028	
0040243F	. 68 34804000	PUSH 3_1.00408034	
00402444	. 8B4D 8C	MOV ECX,DWORD PTR [EBP-74]	
00402447	. E8 DA360000	CALL <JMP.&MFC42.#4224>	
0040244C	> 8BE5	MOV ESP,EBP	
0040244E	. 5D	POP EBP	
0040244F	. C3	RET	

s
strlen
Arg2
Arg1
3_1.00401450

Arg3
Arg2
Arg1
3_1.00401A09

n = 10 (16.)
s2
s1
memcmp

ASCII "U did it!!!"
ASCII "Good Work Cryptologist!!!"

- b. Địa chỉ của password được lưu vào địa chỉ [EBP – 24] và của username được lưu vào [EBP – 58]. Chương trình sẽ kiểm tra độ dài của password (> 4) và username (> 6), nếu không thỏa mãn => Đăng nhập thất bại.

004023E1	. 837D A4 06	CMP DWORD PTR [EBP-5C],6
004023E5	.. 7E 65	JLE SHORT 3_1.0040244C
004023E7	. 837D FC 04	CMP DWORD PTR [EBP-4],4
004023EB	.. 7E 5F	JLE SHORT 3_1.0040244C

- c. Lấy độ dài password và lưu vào thanh ghi EAX

004023ED	. 8D55 A8	LEA EDX,DWORD PTR [EBP-58]	
004023F0	. 52	PUSH EDX	
004023F1	. 8D45 DC	LEA EAX,DWORD PTR [EBP-24]	
004023F4	. 50	PUSH EAX	
004023F5	. E8 D6370000	CALL <JMP.&MSUCRT.strlen>	Arg2 strlen

d. Gọi hàm 00401450 với hai tham số là password và độ dài.

004023FD	. 50	PUSH EAX	Arg2
004023FE	. 8D4D DC	LEA ECX,DWORD PTR [EBP-24]	
00402401	. 51	PUSH ECX	Arg1
00402402	. E8 49F0FFFF	CALL 3_1.00401450	3_1.00401450

Chi tiết như sau:

- Lấy mỗi ký tự trong password ra và thực hiện các phép so sánh, phép toán ADD, SUB,... mặc định để chuyển sang 1 ký tự mới.

00401450	. 55	PUSH EBP
00401451	. 8BEC	MOV EBP,ESP
00401453	. 83EC 24	SUB ESP,24
00401456	. C745 E4 0000	MOV DWORD PTR [EBP-1C],0
0040145D	. C745 F4 0000	MOV DWORD PTR [EBP-C],0
00401464	. C745 F8 0000	MOV DWORD PTR [EBP-8],0
0040146B	. C745 E8 0000	MOV DWORD PTR [EBP-18],0
00401472	. EB 09	JMP SHORT 3_1.0040147D
00401474	. 8B45 E8	MOV EAX,DWORD PTR [EBP-18]
00401477	. 83C0 01	ADD EAX,1
0040147A	. 8945 E8	MOV DWORD PTR [EBP-18],EAX
0040147D	. 8B4D E8	MOV ECX,DWORD PTR [EBP-18]
00401480	. 3B4D 0C	CMP ECX,DWORD PTR [EBP+C]
00401483	. 0F8D FA000000	JGE 3_1.00401583
00401489	. 8B55 08	MOV EDX,DWORD PTR [EBP+8]
0040148C	. 0355 E8	ADD EDX,DWORD PTR [EBP-18]
0040148F	. 8A02	MOV AL,BYTE PTR [EDX]
00401491	. 8B45 F0	MOV BYTE PTR [EBP-10],AL
00401494	. 8B4D F0	MOV ECX,DWORD PTR [EBP-10]
00401497	. 81E1 FF000000	AND ECX,0FF
0040149D	. 83F9 2B	CMP ECX,2B
004014A0	. 7C 22	JL SHORT 3_1.004014C4
004014A2	. 8B55 F0	MOV EDX,DWORD PTR [EBP-10]
004014A5	. 81E2 FF000000	AND EDX,0FF
004014AB	. 83FA 7A	CMP EDX,7A
004014AE	. 7F 14	JG SHORT 3_1.004014C4
004014B0	. 8B45 F0	MOV EAX,DWORD PTR [EBP-10]
004014B3	. 25 FF000000	AND EAX,0FF
004014B8	. 0FB888 A56340	MOVSX ECX,BYTE PTR [EAX+4063A5]
004014BF	. 894D E0	MOV DWORD PTR [EBP-20],ECX
004014C2	. EB 07	JMP SHORT 3_1.004014CB
004014C4	. C745 E0 0000	MOV DWORD PTR [EBP-20],0
004014CB	. 8A55 E0	MOV DL,BYTE PTR [EBP-20]
004014CE	. 8B55 F0	MOV BYTE PTR [EBP-10],DL
004014D1	. 8B45 F0	MOV EAX,DWORD PTR [EBP-10]
004014D4	. 25 FF000000	AND EAX,0FF
004014D9	. 85C0	TEST EAX,EAX
004014DB	. 74 2C	JE SHORT 3_1.00401509
004014DD	. 8B4D F0	MOV ECX,DWORD PTR [EBP-10]
004014E0	. 81E1 FF000000	AND ECX,0FF
004014E6	. 83F9 24	CMP ECX,24
004014E9	. 75 09	JNZ SHORT 3_1.004014F4
004014EB	. C745 DC 0000	MOV DWORD PTR [EBP-24],0
004014F2	. EB 0F	JMP SHORT 3_1.00401503
004014F4	. 8B55 F0	MOV EDX,DWORD PTR [EBP-10]
004014F7	. 81E2 FF000000	AND EDX,0FF
004014FD	. 83EA 3D	SUB EDX,3D
00401500	. 8955 DC	MOV DWORD PTR [EBP-24],EDX
00401503	. 8A45 DC	MOV AL,BYTE PTR [EBP-24]
00401506	. 8B45 F0	MOV BYTE PTR [EBP-10],AL
00401509	. 8B4D F0	MOV ECX,DWORD PTR [EBP-10]
0040150C	. 81E1 FF000000	AND ECX,0FF
00401512	. 83E9 01	SUB ECX,1
00401515	. 8B55 E4	MOV EDX,DWORD PTR [EBP-1C]
00401518	. 8B4C15 EC	MOV BYTE PTR [EBP+EDX-14],CL

- Cứ mỗi 4 ký tự sẽ thực hiện tiếp một loạt các phép toán để sinh ra 3 ký tự khác (ở hàm **004013F0**) và lưu nối tiếp nhau cho đến khi xét hết. Các ký tự dư ra sẽ không xét.

0040151C	. 8B45 E4	MOV EAX,DWORD PTR [EBP-1C]	
0040151F	. 83C0 01	ADD EAX,1	
00401522	. 8945 E4	MOV DWORD PTR [EBP-1C],EAX	
00401525	. 837D E4 04	CMP DWORD PTR [EBP-1C],4	
00401529	~ 75 53	JNZ SHORT 3_1.0040157E	
0040152B	. 8D4D FC	LEA ECX,DWORD PTR [EBP-4]	
0040152E	. 51	PUSH ECX	Arg2
0040152F	. 8D55 EC	LEA EDX,DWORD PTR [EBP-14]	
00401532	. 52	PUSH EDX	Arg1
00401533	. E8 B8FEFFFF	CALL 3_1.004013F0	3_1.004013F0
00401538	. 83C4 08	ADD ESP,8	
0040153B	. C745 E4 0000	MOV DWORD PTR [EBP-1C],0	
00401542	. 8B45 10	MOV EAX,DWORD PTR [EBP+10]	
00401545	. 0345 F8	ADD EAX,DWORD PTR [EBP-8]	
00401548	. 8A4D FC	MOV CL,BYTE PTR [EBP-4]	
0040154B	. 8B08	MOV BYTE PTR [EAX],CL	
0040154D	. 8B55 F8	MOV EDX,DWORD PTR [EBP-8]	
00401550	. 83C2 01	ADD EDX,1	
00401553	. 8955 F8	MOV DWORD PTR [EBP-8],EDX	
00401556	. 8B45 10	MOV EAX,DWORD PTR [EBP+10]	
00401559	. 0345 F8	ADD EAX,DWORD PTR [EBP-8]	
0040155C	. 8A4D FD	MOV CL,BYTE PTR [EBP-3]	
0040155F	. 8B08	MOV BYTE PTR [EAX],CL	
00401561	. 8B55 F8	MOV EDX,DWORD PTR [EBP-8]	
00401564	. 83C2 01	ADD EDX,1	
00401567	. 8955 F8	MOV DWORD PTR [EBP-8],EDX	
0040156A	. 8B45 10	MOV EAX,DWORD PTR [EBP+10]	
0040156D	. 0345 F8	ADD EAX,DWORD PTR [EBP-8]	
00401570	. 8A4D FE	MOV CL,BYTE PTR [EBP-2]	
00401573	. 8B08	MOV BYTE PTR [EAX],CL	
00401575	. 8B55 F8	MOV EDX,DWORD PTR [EBP-8]	
00401578	. 83C2 01	ADD EDX,1	
0040157B	. 8955 F8	MOV DWORD PTR [EBP-8],EDX	
0040157E	> E9 F1FEFFFF	JMP 3_1.00401474	
00401583	> 8B45 F8	MOV EAX,DWORD PTR [EBP-8]	
00401586	. 8BE5	MOV ESP,EBP	
00401588	. 5D	POP EBP	
00401589	. C3	RET	

- e. Gọi hàm **00401A09** với 3 tham số gồm: địa chỉ dãy ký tự phát sinh sau bước d, địa chỉ username và địa chỉ một vùng nhớ trống để lưu kết quả.

0040240E	. 8D55 C8	LEA EDX,DWORD PTR [EBP-38]		Registers (FPU)
00402411	. 52	PUSH EDX	Arg3	EAX 0018F7BC
00402412	. 8D45 A8	LEA EAX,DWORD PTR [EBP-58]		ECX 0018F7A4 ASCII "abodefg"
00402415	. 50	PUSH EAX	Arg2	EDX 0018F7DC
00402416	. 8D4D 90	LEA ECX,DWORD PTR [EBP-70]		EBX 00000111
00402419	. 51	PUSH ECX	Arg1	ESP 0018F794
0040241A	. E8 EAF5FFFF	CALL 3_1.00401A09	3_1.00401A09	EBP 0018F814

Chi tiết hàm:

00401A09	55	PUSH EBP	
00401A0A	8BEC	MOV EBP,ESP	
00401A0C	81EC 00020000	SUB ESP,200	
00401A12	8B45 08	MOV EAX,DWORD PTR [EBP+8]	
00401A15	50	PUSH EAX	
00401A16	E8 B5410000	CALL <JMP.&MSVCRT.strlen>	s strlen
00401A18	83C4 04	ADD ESP,4	
00401A1E	50	PUSH EAX	
00401A1F	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	n
00401A22	51	PUSH ECX	src
00401A23	8D95 00FFFFFF	LEA EDX,DWORD PTR [EBP-100]	
00401A29	52	PUSH EDX	dest
00401A2A	E8 A7410000	CALL <JMP.&MSVCRT.memcpy>	memcpy
00401A2F	83C4 0C	ADD ESP,0C	
00401A32	8B45 08	MOV EAX,DWORD PTR [EBP+8]	
00401A35	50	PUSH EAX	
00401A36	E8 95410000	CALL <JMP.&MSVCRT.strlen>	s strlen
00401A38	83C4 04	ADD ESP,4	
00401A3E	C68405 00FFFF	MOV BYTE PTR [EBP+EAX-100],0	
00401A46	8D8D 00FFFFFF	LEA ECX,DWORD PTR [EBP-200]	
00401A4C	51	PUSH ECX	Arg1
00401A4D	E8 38FBFFFF	CALL 3_1.0040158A	3_1.0040158A
00401A52	83C4 04	ADD ESP,4	
00401A55	8B55 08	MOV EDX,DWORD PTR [EBP+8]	
00401A58	52	PUSH EDX	s strlen
00401A59	E8 72410000	CALL <JMP.&MSVCRT.strlen>	
00401A5E	83C4 04	ADD ESP,4	
00401A61	50	PUSH EAX	Arg2
00401A62	8B45 08	MOV EAX,DWORD PTR [EBP+8]	
00401A65	50	PUSH EAX	Arg1
00401A66	E8 4FFBFFFF	CALL 3_1.004015BA	3_1.004015BA
00401A6B	83C4 08	ADD ESP,8	
00401A6E	8B4D 08	MOV ECX,DWORD PTR [EBP+8]	
00401A71	C641 10 00	MOV BYTE PTR [ECX+10],0	
00401A75	8D95 00FFFFFF	LEA EDX,DWORD PTR [EBP-100]	
00401A7B	52	PUSH EDX	Arg2
00401A7C	8D85 00FFFFFF	LEA EAX,DWORD PTR [EBP-200]	
00401A82	50	PUSH EAX	Arg1
00401A83	E8 76FBFFFF	CALL 3_1.004015FE	3_1.004015FE
00401A88	83C4 08	ADD ESP,8	
00401A8B	8D8D 00FFFFFF	LEA ECX,DWORD PTR [EBP-200]	
00401A91	51	PUSH ECX	Arg1
00401A92	E8 0CFCFFFF	CALL 3_1.004016A3	3_1.004016A3
00401A97	83C4 04	ADD ESP,4	
00401A9A	8D95 00FFFFFF	LEA EDX,DWORD PTR [EBP-200]	
00401AA0	52	PUSH EDX	Arg3
00401AA1	8B45 10	MOV EAX,DWORD PTR [EBP+10]	
00401AA4	50	PUSH EAX	Arg2
00401AA5	8B4D 0C	MOV ECX,DWORD PTR [EBP+C]	
00401AA8	51	PUSH ECX	Arg1
00401AA9	E8 0DFEFFFF	CALL 3_1.004018BB	3_1.004018BB
00401AAE	83C4 0C	ADD ESP,0C	
00401AB1	8D95 00FFFFFF	LEA EDX,DWORD PTR [EBP-200]	
00401AB7	52	PUSH EDX	Arg3
00401AB8	8B45 10	MOV EAX,DWORD PTR [EBP+10]	
00401ABB	83C0 08	ADD EAX,8	
00401ABE	50	PUSH EAX	Arg2
00401ABF	8B4D 0C	MOV ECX,DWORD PTR [EBP+C]	
00401AC2	83C1 08	ADD ECX,8	
00401AC5	51	PUSH ECX	Arg1
00401AC6	E8 F0FDFFFF	CALL 3_1.004018BB	3_1.004018BB
00401AC8	83C4 0C	ADD ESP,0C	
00401ACE	8B55 10	MOV EDX,DWORD PTR [EBP+10]	
00401AD1	C642 10 00	MOV BYTE PTR [EDX+10],0	
00401AD5	33C0	XOR EAX,EAX	
00401AD7	8BE5	MOV ESP,EBP	
00401AD9	5D	POP EBP	
00401ADA	C3	RET	

Ta quan tâm đến 5 hàm con được gọi trong hàm này:

- **0040158A**: Phát sinh một dãy 256 ký tự với giá trị tăng dần từ 0 đến 255.

- **004015BA**: Dựa vào username mà người dùng nhập vào và qua một thuật toán được cài mặc định phát sinh dãy 16 ký tự mới.
- **004015FE**: Hoán vị các phần tử trong dãy 256 ký tự sinh ra ở hàm **0040158A** qua một thuật toán mặc định.
- **004016A3**: Lấy từ dãy 256 ký tự đó sau khi gọi hàm **004015FE** ra 8 bộ ký tự, mỗi bộ 4 ký tự lưu vào một vùng nhớ (tạm gọi là T). Ký tự tại vị trí nào được lấy ra được quy định từ đầu trong mã nguồn.
- **004018BB**: Lấy 2 bộ ký tự (mỗi bộ 4 ký tự). Qua một thuật toán mặc định, xor với giá trị tại T và lưu vào stack theo thứ tự ngược lại. Qua hai lần gọi hàm, 16 ký tự được sinh ra ở bước d sẽ được chuyển thành 16 ký tự khác, mỗi lần 2 bộ (8 ký tự). Nếu ở bước d sinh ra không đủ 16 ký tự thì đăng nhập sẽ thất bại, nếu dư thì không ảnh hưởng.

f. So sánh giá trị của 16 ký tự sinh ra từ hàm **004015BA** với 16 ký tự sinh ra từ hai lần gọi hàm **004018BB**. Nếu bằng nhau => Đăng nhập thành công.

00402422	. 6A 10	PUSH 10	<div><div>n = 10 (16.)</div><div>s2</div><div>s1</div><div>memcmp</div><div>ASCII "U did it!!!"</div><div>ASCII "Good Work Cryptologist!!!"</div></div>
00402424	. 8D55 C8	LEA EDX,DWORD PTR [EBP-38]	
00402427	. 52	PUSH EDX	
00402428	. 8D45 90	LEA EAX,DWORD PTR [EBP-70]	
0040242B	. 50	PUSH EAX	
0040242C	. E8 B7370000	CALL <JMP.&MSUCRT.memcmp>	
00402431	. 83C4 0C	ADD ESP,0C	
00402434	. 85C0	TEST EAX,EAX	
00402436	~ 75 14	JNZ SHORT 3_1.0040244C	
00402438	. 6A 00	PUSH 0	
0040243A	. 68 28804000	PUSH 3_1.00408028	
0040243F	. 68 34804000	PUSH 3_1.00408034	
00402444	. 8B4D 8C	MOV ECX,DWORD PTR [EBP-74]	

5. Bài tập trên trang microcorruption.com

1. Level Tutorial

4484:	6e4f	mov.b @r15, r14
4486:	1f53	inc r15
4488:	1c53	inc r12
448a:	0e93	tst r14
448c:	fb23	jnz #0x4484 <check_password+0x0>
448e:	3c90 0900	cmp #0x9, r12
4492:	0224	jeq #0x4498 <check_password+0x14>
4494:	0f43	clr r15
4496:	3041	ret
4498:	1f43	mov #0x1, r15
449a:	3041	ret

Từ dòng 4484 đến 448C: Duyệt từng byte từ người dùng nhập vào (lưu trong thanh ghi r15) vào thanh ghi r14 cho đến khi byte đó có giá trị bằng 0x00 (lúc này zero-flag được bật, jnz tại dòng 448C sẽ không thực hiện). Lúc này so sánh r12 (có ý nghĩa là số kí byte người dùng nhập vào) tính cả byte 0x00 (do inc trước rồi mới so sánh với 0x9). Sau lệnh cmp thì zero-flag = 0, gán r15 = 0x1 và nhảy ra hàm main, do r15 = 0x1 nên unlock thành công

➔ Key: bất cứ chuỗi byte nào đủ 9 byte (kết thúc bằng 0x00), ví dụ

- 11 11 11 11 11 11 11 11 00
- 22 22 22 22 22 22 22 22 00

2. Level New Orleans

```
44bc: 0e43      clr r14
44be: 0d4f      mov r15, r13
44c0: 0d5e      addr14, r13
44c2: ee9d 0024  cmp.b     @r13, 0x2400(r14)
44c6: 0520      jne #0x44d2 <check_password+0x16>
44c8: 1e53      inc r14
44ca: 3e92      cmp #0x8, r14
44cc: f823      jne #0x44be <check_password+0x2>
44ce: 1f43      mov #0x1, r15
44d0: 3041      ret
44d2: 0f43      clr r15
44d4: 3041      ret
```

Key được lưu tại vùng nhớ 0x2400 (thấy được khi chạy debug):

```
2400: 584d 6e55 3575 4e00 0000 0000 0000 0000 XMnU5uN.....
```

Key do người dùng nhập được lưu tại vùng nhớ trỏ bởi r15, đoạn code duyệt từng kí tự (biến đếm là r14), so sánh từ byte được trỏ bởi r13 (r13, r15 hiện tại chỉ là địa chỉ ô nhớ, @r13 truy xuất đến giá trị tại ô nhớ đó), so sánh từng byte người dùng nhập với từng byte tương ứng tại vùng nhớ 0x2400. Nếu có 1 byte nào sai sẽ gán r15 (kết quả) = 0 và fail. Còn không, sẽ so sánh đến qua byte tại 0x2407 thì kết thúc (mặc dù sau đó có thể còn byte khác)

➔ Key: Bất kì chuỗi byte nào bắt đầu bằng 8 byte: 584d 6e55 3575 4e00, ví dụ

- 584d 6e55 3575 4e00
- 584d 6e55 3575 4e00 1111 1111 1111

3. Sydney

```
448a: bf90 3626 0000  cmp #0x2636, 0x0(r15)
4490: 0d20      jnz $+0x1c
4492: bf90 6465 0200  cmp #0x6564, 0x2(r15)
4498: 0920      jnz $+0x14
449a: bf90 6b77 0400  cmp #0x776b, 0x4(r15)
44a0: 0520      jne #0x44ac <check_password+0x22>
44a2: 1e43      mov #0x1, r14
44a4: bf90 4523 0600  cmp #0x2345, 0x6(r15)
44aa: 0124      jeq #0x44ae <check_password+0x24>
44ac: 0e43      clr r14
44ae: 0f4e      mov r14, r15
44b0: 3041      ret
```

Lần này so sánh từng cặp byte (1 word) do người dùng nhập lưu tại vùng nhớ trỏ bởi r15

Do byte được lưu theo Little Endian nên key sẽ là:

➔ Key: 3626 6465 6b77 4523

4. Hanoi

```
4544: b012 5444      call      #0x4454 <test_password_valid>
4548: 0f93          tst r15
454a: 0324          jz $+0x8
454c: f240 2200 1024 mov.b     #0x22, &0x2410
4552: 3f40 d344      mov #0x44d3 "Testing if password is valid.", r15
4556: b012 de45      call     #0x45de <puts>
455a: f290 1500 1024 cmp.b     #0x15, &0x2410
4560: 0720          jne #0x4570 <login+0x50>
4562: 3f40 f144      mov #0x44f1 "Access granted.", r15
4566: b012 de45      call     #0x45de <puts>
456a: b012 4844      call     #0x4448 <unlock_door>
```

Nhìn vào dòng 455A thấy so sánh byte tại ô nhớ 2410 với 0x15. Khi chạy debug ta thấy vùng nhớ từ ô 2400 đến 2410 dùng để chứa key do người dùng nhập (tối đa 16 bytes). Do đó ta có tình nhập dư 1 byte (và byte này có giá trị 0x15) thì key sẽ được chấp nhận

➔ Key: Chỉ cần có độ dài 17 bytes và bytes cuối cùng là 0x15 thì hợp lệ, ví dụ:

- 1111 1111 1111 1111 1111 1111 1111 1111 15
- 2222 2222 3333 3333 4444 4444 5555 5555 15

5. Cusco

Giả sử ban đầu ta nhập “1234567890123456” (tương đương hexa: 3132 3334 3536 3738 3930 3132 3334 3536) thì trong bộ nhớ sau khi nhập có dạng:

```
43d0: 0000 0000 0000 0000 0000 0000 5645 0100 .....VE..
43e0: 5645 0000 a045 0200 ee43 3000 1e45 3132 VE...E...C0..E12
43f0: 3334 3536 3738 3930 3132 3334 3536 0044 34567890123456.D
4400: 3140 0044 1542 5c01 75f3 35d0 085a 3f40 1@.D.B\.u.5..Z?@
```

Ta để ý ở byte 43fe là 0044 3140 (tức là 0x4400), đây là dòng mà nếu ta chạy tiếp câu lệnh ret (đã đặt breakpoint tại đây) thì thanh ghi pc sẽ quay về địa chỉ này

```
4532: 3f40 e144      mov #0x44e1 "That password is not correct.", r15
4536: b012 a645      call     #0x45a6 <puts>
453a: 3150 1000      add #0x10, sp
453e: 3041          ret
```

Tức là sẽ nhảy vào hàm <__init_stack>

```
4400 <__init_stack>
4400: 3140 0044      mov #0x4400, sp
```

Và chạy tiếp thì chương trình sẽ yêu cầu nhập lại key. Đây chính là chỗ ta có thể hack dựa vào cơ chế tràn bộ nhớ, tức là nhập quá 16 bytes thì các byte sau sẽ ghi đè lên ô nhớ này

Như vậy, ta chọn giá trị là ee43 thì sau lệnh ret, thanh ghi pc sẽ đến ô nhớ 0x43ee, tức là ô nhớ lưu 16 bytes đầu của key mình nhập vào, mục đích để điều chỉnh thanh ghi pc thông qua key mình nhập

Như vậy nếu nhập key là 1111 1111 1111 1111 1111 1111 1111 1111 ee43 thì sau lệnh ret, thanh ghi pc sẽ nhảy đến ô có giá trị 11 khoanh đỏ dưới đây:

```
43d0: 0000 0000 0000 0000 0000 0000 5645 0100 .....VE..
43e0: 5645 0300 ca45 0000 0a00 0000 3a45 1111 VE...E.....:E..
```



```

43f0:  1111 1111 1111 1111 1111 1111 1111 ee43  ....C
4400:  0040 0044 1542 5c01 75f3 35d0 085a 3f40  .@.D.B\..u.5..Z?@

```

Quay lại thanh ghi pc: lúc này nếu ô nhớ có giá trị là 0044 3140 thì thanh ghi pc sẽ nhảy đến dòng 4400 tức là hàm <__init_stack>

```

4400 <__init_stack>
4400:  3140 0044      mov #0x4400, sp

```

Do đó nếu ta thay giá trị này để thanh ghi nhảy đến hàm <unlock_door> thì ta cần 4 bytes đầu có giá trị là b012 4644 (b012 là lệnh nhảy, 4644 là dòng muốn nhảy đến, tương đương lệnh call #0x4446, tức là hàm <unlock_door>)

```

4446 <unlock_door>
4446:  3012 7f00      push      #0x7f
444a:  b012 4245      call     #0x4542 <INT>
444e:  2153          incd     sp
4450:  3041          ret

```

➔ Key: Bắt đầu bằng 4 bytes b012 4644 và kết thúc bằng 2 bytes ee43, các bytes giữa không quan trọng, tổng số bytes là 18 (hack qua lỗi tràn bộ nhớ), ví dụ:

- B012 4644 1111 1111 1111 1111 1111 1111 ee43
- B012 4644 2222 2222 2222 2222 3333 3333 ee43

6. Whitehorse

Tương tự như Cusco, giả sử ban đầu ta nhập “1234567890123456” (tương đương hexa: 3132 3334 3536 3738 3930 3132 3334 3536) thì trong bộ nhớ sau khi nhập có dạng:

```

3cb0:  0000 4645 0000 9045 0200 c03c 3000 1245  ..FE...E...<0..E
3cc0:  3132 3334 3536 3738 3930 3132 3334 3536  1234567890123456
3cd0:  0044 0000 0000 0000 0000 0000 0000 0000  .D.....

```

Để ý ta thấy byte 3cd0 có giá trị là 0044, và tương tự Cusco, đây là địa chỉ của <__init_stack> là chạy lại chương trình. Do đó ta overwrite lại vùng nhớ này bằng cách điền tràn số, giá trị cần chen vào là c03c để đưa thanh ghi pc sau lệnh ret bên dưới trở vào vùng nhớ chứa key

```

4522:  3f40 d544      mov #0x44d5 "That password is not correct.", r15
4526:  b012 9645      call     #0x4596 <puts>
452a:  3150 1000      add #0x10, sp
452e:  3041          ret

```

Tức là key lúc này có dạng 3132 3334 3536 3738 3930 3132 3334 3536 c03c

Nhìn qua hàm <conditional_unlock_door>, ta thấy khác ở dòng 445c, ở Cusco, giá trị push vào là 0x7f.

```

445a:  0f12          push     r15
445c:  3012 7e00      push     #0x7e
4460:  b012 3245      call     #0x4532 <INT>
4464:  5f44 fcff      mov.b    -0x4(r4), r15

```

Tra cứu tài liệu manual [2] thì thấy:

- INT 0x7E. Interface with the HSM-2. Trigger the deadbolt unlock if the password is correct. Takes one argument: the password to test.
- INT 0x7F. Interface with deadbolt to trigger an unlock if the password is correct. Takes no arguments.

Như vậy nếu ta gọi <INT> 0x7F như trước có khả năng sẽ bypass được password (do hàm kiểm tra là <INT> 0x7E)

```
3012 7f00    push    #0x7f
b012 3245    call    #0x4532 <INT>
```

Do đó byte cần truyền vào để thanh pc chạy được 2 dòng trên là 3012 7f00 b012 3245

- ➔ Key: Có dạng 3012 7f00 b012 3245 xxxx xxxx xxxx xxxx c03c, ví dụ
- 3012 7f00 b012 3245 1111 1111 1111 1111 c03c
 - 3012 7f00 b012 3245 2222 3333 4444 5555 c03c

IV. Tài liệu tham khảo

- [1] L. M. & J. F. R. Markus F.X.J. Oberhumer, "https://upx.github.io/," The UPX Team, [Online]. Available: <https://upx.github.io/>. [Accessed 10 05 2018].
- [2] "Microcorruption," [Online]. Available: <https://microcorruption.com/manual.pdf>. [Accessed 18 May 2018].