



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

## BÁO CÁO

# ĐỒ ÁN CRACK PHẦN MỀM

*Môn học: Kiến trúc máy tính và hợp ngữ*

*Giáo viên: Phạm Tuấn Sơn , Lê Viết Long*

*Nhóm thực hiện: 1752004 – 1752013 - 1752042*

---

Tháng 8, 2018

## Mục lục

Tổng quan .....	3
Thông tin đồ án .....	3
Phân công .....	3
Kiến thức cần có để giải quyết .....	3
1.1 .....	4
Manh mối .....	4
Đoạn mã phát sinh khóa của chương trình .....	6
Test case .....	6
Phân tích mã assemble .....	6
Tóm tắt thuật toán .....	8
Kết luận .....	9
2.1 .....	9
Manh mối .....	9
Phân tích mã assemble .....	13
Tóm tắt thuật toán .....	17
Kết luận .....	18
2.2 .....	18
Manh mối .....	18
Phân tích mã assemble .....	21
Tóm tắt thuật toán .....	25
Kết luận .....	25
Tổng kết .....	26
Mức độ hoàn thành .....	26
Kiến thức thu được .....	26
Tham khảo .....	26

## Tổng quan

---

# Thông tin đồ án

**Bộ môn:** Kiến trúc máy tính và hợp ngữ

Giáo viên: Phạm Tuấn Sơn

Giáo viên: Lê Viết Long

**Tên: Đồ án 3 - Crack phần mềm**

**Đề: 1 (1.1.exe, 1.2.exe, 1.3.exe)**

**Deadline: 12h00 22/08/2018**

Thời gian thực hiện: 15/08/2018 – 22/08/2018

## Phân công

1752004	Võng Khải My	1.1
1752013	Cao Minh Đức	1.2
1752042	Lê Mỹ Phương	1.3

Công việc gồm: viết báo cáo và tạo keygen (xxx.cpp, xxx.exe) của crackme được phân công.

## Kiến thức cần có để giải quyết

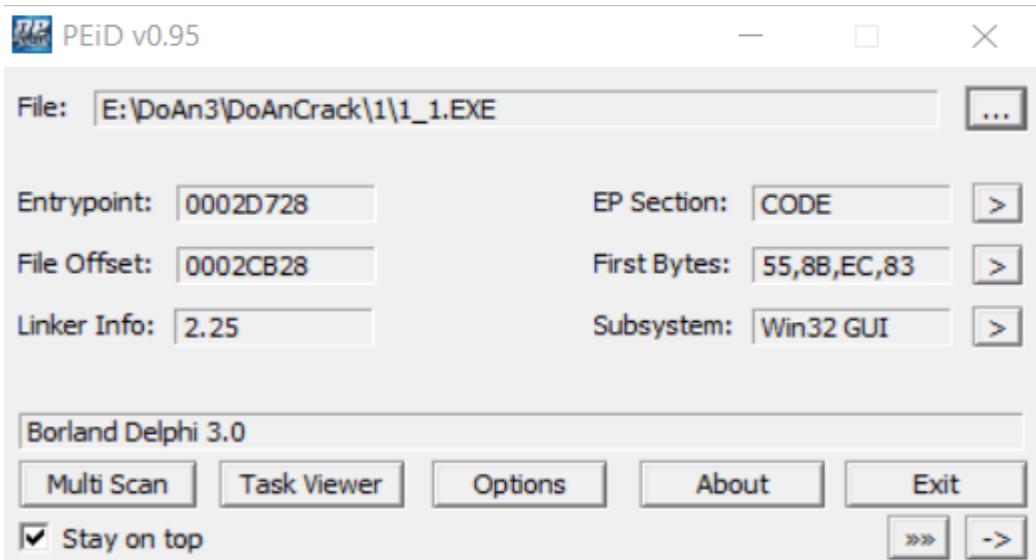
1. Kỹ năng kiểm tra bằng PEiD: kiểm tra file có pack không, kiểm tra có dấu vết của mã hóa trong file thực thi không, unpack nếu cần.
2. Kỹ năng debug trong OllyDbg: đặt breakpoint F2, F9 để bắt đầu debug, F8 để chạy từng lệnh, F7 để đi vào bên trong hàm chạy từng lệnh; tìm chuỗi với Search for > All referenced strings; truy cập đến mọi vùng nhớ bằng Memory map; kỹ năng xử lý địa chỉ của vùng nhớ stack, dump.
3. Đọc hiểu các lệnh hợp ngữ: nhóm lệnh tính toán (ADD, SUB, IMUL, XOR, OR, AND), nhóm lệnh so sánh (TEST, CMP), nhóm lệnh nhảy (JE, JAE, JNE, JNZ...), nhóm lệnh gán (MOV-gán giá trị, LEA-gán địa chỉ).

## 1.1

## Manh mối

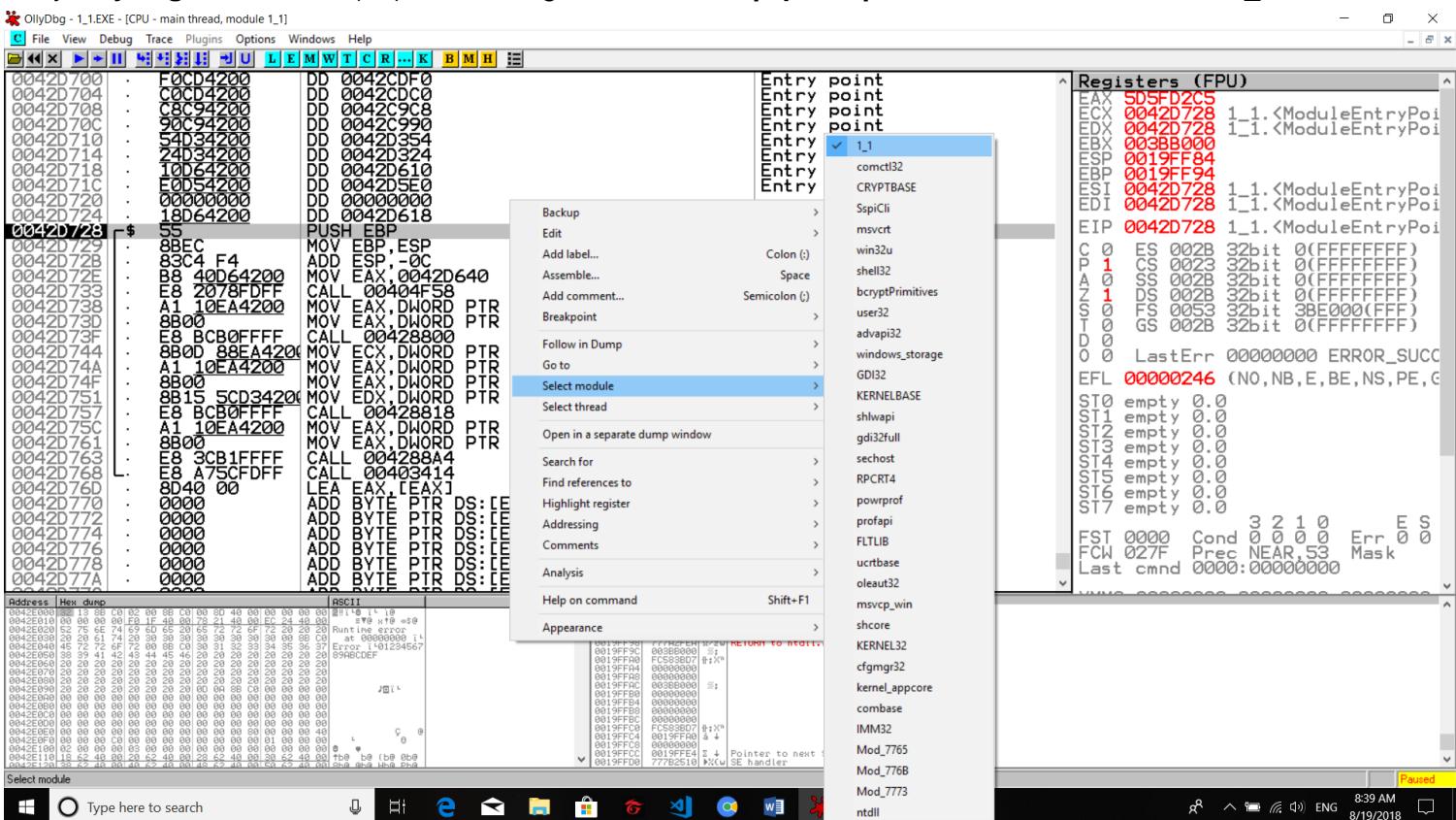
Trước hết, phân tích mã hợp ngữ phải trên file thực thi không bị pack. Pack là thao tác của người lập trình để nén chương trình. Việc này ngăn chặn những nỗ lực đọc hiểu mã nguồn hợp ngữ của các cracker. Nếu chủ quan không kiểm tra xem file bị pack hay không mà trực tiếp phân tích, kết quả phân tích sẽ không chính xác, thậm chí không phân tích được do nhiều dữ kiện quan trọng bị ẩn mất.

Đầu tiên, **Scan with PEiD** 1.1.exe để kiểm tra tệp thực tin này đã pack chưa.



Kết quả là **Borland Delphi 3.0** \*, 1.1.exe không bị pack, nó được viết bằng ngôn ngữ **Borland Delphi**, tiến hành phân tích mã hợp ngữ bằng **OllyDbg**

Chạy OllyDbg, mở 1.1.exe (F3). Nếu không mặc định, chuột phải tại Select Module > Module '1\_1'



Bây giờ, bắt đầu tìm kiếm những chuỗi được 1.1.exe hiển thị. **Chuột phải tại Search for > All referenced text strings**

The screenshot shows the OllyDbg debugger interface with the following details:

- Assembly View:** The left pane displays assembly code for module 1\_1. A context menu is open over the instruction at address 0042D728, which is PUSH EBP. The menu includes options like Backup, Edit, Add label..., Assemble..., Add comment..., Breakpoint, Follow in Dump, Go to, Select module, Select thread, Open in a separate dump window, Search for, Find references to, Highlight register, Addressing, Comments, Analysis, Help on command (Shift+F1), and Appearance.
- Registers View:** The top right pane shows the FPU registers. The CPU register values are:
  - EAX: 5D5FD2C5
  - ECX: 0042D728
  - EDX: 0042D728
  - EBX: 003BB000
  - ESP: 0019F84
  - EBP: 0019F94
  - ESI: 001A0728
- Memory Dump View:** The bottom right pane shows a memory dump of the current CPU state. It includes fields for Address, Hex, and Dump. The dump shows the CPU register values listed above.
- Status Bar:** The bottom status bar indicates the current state: Paused, ENG, 8:40 AM, and 8/19/2018.

Một cửa sổ mới xuất hiện, chứa nhiều chuỗi, nhưng những chuỗi được hiển thị trong hình cần chú ý.

Address	Command	Comments
0042CF4B	ASCII "Label1"	
0042CF58	ASCII "URLLabel1"	
0042CF69	ASCII "FormCreate"	
0042CF74	ASCII "TAboutBox"	
0042CF96	ASCII "TAboutBox"	
0042CFAA	ASCII "About"	
0042D03A	MOV EDX, 0042D1D0	ASCII "Windows 95"
0042D057	MOV EDX, 0042D1E4	ASCII "Windows NT"
0042D06E	MOV EDX, 0042D1F8	ASCII "Windows"
0042D0D5	MOV EAX, 0042D208	ASCII "%s %d.%d (Build %d)"
0042D139	MOV EAX, 0042D224	ASCII "%s %d.%d (Build %d: %s)"
0042D185	MOV EAX, 0042D244	ASCII "%s %d.%d"
0042D1D0	ASCII "Windows 95"	
0042D1E4	ASCII "Windows NT"	
0042D1F8	ASCII "Windows"	
0042D208	ASCII "%s %d.%d (Build %d)"	ASCII "%s %d.%d (Build %d)"
0042D224	ASCII "%s %d.%d (Build %d: %s)"	ASCII "%s %d.%d (Build %d: %s)"
0042D244	ASCII "%s %d.%d"	
0042D2A2	MOV EAX, 0042D2E4	ASCII "#,### KB"
0042D2E4	ASCII "#,### KB###"	
0042D429	ASCII "Edit1"	
0042D435	ASCII "Button1"	
0042D443	ASCII "Label1"	
0042D450	ASCII "Edit2"	
0042D45C	ASCII "Label2"	
0042D469	ASCII "MainMenuItem"	
0042D479	ASCII "Main1"	
0042D485	ASCII "About1"	
0042D492	ASCII "Close1"	
0042D4A1	ASCII "Button1Click"	
0042D4B4	ASCII "Close1Click"	
0042D4C6	ASCII "About1Click"	
0042D4D2	ASCII "TForm1"	
0042D4F6	ASCII "TForm1"	
0042D507	ASCII "crackme"	
0042D537	MOV EDX, 0042D590	ASCII "Benadryl"
0042D543	MOV EDX, 0042D5A4	ASCII "Wrong Code DUDE"
0042D555	MOV EDX, 0042D5BC	ASCII "Thanks you made it"
0042D590	ASCII "Benadryl"	
0042D5A4	ASCII "Wrong Code DUDE"	
0042D5BC	ASCII "Thanks you made it"	

Gợi ý để lại: "Có 2 thông báo, một thông báo thành công "thanks you made it" và một thông báo thất bại "wrong code dude". Vậy thông báo còn lại là gì ??? Tại sao lại có thông báo Benadryl ???"

Tạm kết luận: Có thể Benadryl là kết quả cần tìm.

## Đoạn mã phát sinh khóa của chương trình :

0042D510	55	PUSH EBP	
0042D511	8BEC	MOV EBP, ESP	
0042D513	6A 00	PUSH 0	
0042D515	53	PUSH EBX	
0042D516	8BD8	MOV EBX, EAX	
0042D518	33C0	XOR EAX, EAX	
0042D51A	55	PUSH EBP	
0042D51B	68 7BD54200	PUSH 0042D57B	
0042D520	64:FF30	PUSH DWORD PTR FS:[EAX]	Installs SE handler 42D57B
0042D523	64:8920	MOV DWORD PTR FS:[EAX], ESP	
0042D526	8D55 FC	LEA EDX, [EBP-4]	
0042D529	8B83 DC010000	MOV EAX, DWORD PTR DS:[EBX+1DC]	
0042D52F	E8 54CCFEFF	CALL 0041A188	C1_1.0041A188
0042D534	8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]	ASCII "Benadryl"
0042D537	BA 90D54200	MOV EDX, 0042D590	C1_1.004038D0
0042D53C	E8 8F63FDFF	CALL 004038D0	ASCII "Wrong Code DUDE"
0042D541	v 74 12	JZ SHORT 0042D555	
0042D543	BA A4D54200	MOV EDX, 0042D5A4	
0042D548	8B83 E8010000	MOV EAX, DWORD PTR DS:[EBX+1E8]	
0042D54E	E8 65CCFEFF	CALL 0041A1B8	
0042D553	EB 10	JMP SHORT 0042D565	
0042D555	> BA BCD54200	MOV EDX, 0042D5BC	ASCII "Thanks you made it"
0042D55A	8B83 E8010000	MOV EAX, DWORD PTR DS:[EBX+1E8]	
0042D560	E8 53CCFEFF	CALL 0041A1B8	
0042D565	> 33C0	XOR EAX, EAX	
0042D567	. 5A	POP EDX	
0042D568	. 59	POP ECX	
0042D569	. 59	POP ECX	
0042D56A	. 64:8910	MOV DWORD PTR FS:[EAX], EDX	
0042D56D	> 68 82D54200	PUSH 0042D582	Entry point
0042D572	> 8D45 FC	LEA EAX, [EBP-4]	C1_1.00403544
0042D575	> E8 CA5FFDFF	CALL 00403544	Jump to 42D582
0042D57A	v C3	RETN	

Hình 1:

004038D0	\$ 53	PUSH EBX	1_1. 004038D0(guessed void)
004038D1	. 56	PUSH ESI	
004038D2	. 57	PUSH EDI	
004038D3	. 89C6	MOV ESI, EAX	
004038D5	. 89D7	MOV EDI, EDX	
004038D7	. 39D0	CMP EAX, EDX	
004038D9	. v 0F84 8F000000	JE 0040396E	
004038DF	. 85F6	TEST ESI, ESI	
004038E1	. v 74 68	JZ SHORT 0040394B	
004038E3	. 85FF	TEST EDI, EDI	
004038E5	. v 74 6B	JZ SHORT 00403952	
004038E7	. 8B46 FC	MOV ECX, DWORD PTR DS:[ESI-4]	
004038EA	. 8B57 FC	MOV EDX, DWORD PTR DS:[EDI-4]	
004038ED	. 29D0	SUB EAX, EDX	
004038EF	. v 77 02	JA SHORT 004038F3	
004038F1	. 01C2	ADD EDX, EAX	
004038F3	> 52	PUSH EDX	
004038F4	. C1EA 02	SHR EDX, 2	
004038F7	. v 74 26	JZ SHORT 0040391F	
004038F9	> 8B0E	MOV ECX, DWORD PTR DS:[ESI]	
004038FB	. 8B1F	MOV EBX, DWORD PTR DS:[EDI]	
004038FD	. 39D9	CMP ECX, EBX	
004038FF	. v 75 58	JNE SHORT 00403959	
00403901	. 4A	DEC EDX	
00403902	. v 74 15	JZ SHORT 00403919	
00403904	. 8B4E 04	MOV ECX, DWORD PTR DS:[ESI+4]	
00403907	. 8B5F 04	MOV EBX, DWORD PTR DS:[EDI+4]	
0040390A	. 39D9	CMP ECX, EBX	
0040390C	. v 75 4B	JNE SHORT 00403959	
0040390E	. 83C6 08	ADD ESI, 8	
00403911	. 83C7 08	ADD EDI, 8	
00403914	. 4A	DEC EDX	
00403915	. v 75 E2	JNE SHORT 004038F9	
00403917	. v EB 06	JMP SHORT 0040391F	
00403919	> 83C6 04	ADD ESI, 4	
0040391C	. 83C7 04	ADD EDI, 4	
0040391F	> 5A	POP EDX	
00403920	. 83E2 03	AND EDX, 00000003	
00403923	. v 74 22	JZ SHORT 00403947	
00403925	. 8B0E	MOV ECX, DWORD PTR DS:[ESI]	
00403927	. 8B1F	MOV EBX, DWORD PTR DS:[EDI]	
00403929	. 38D9	CMP CL, BL	
0040392B	. v 75 41	JNE SHORT 0040396E	
0040392D	. 4A	DEC EDX	
0040392E	. v 74 17	JZ SHORT 00403947	
00403930	. 38FD	CMP CH, BH	
00403932	. v 75 3A	JNE SHORT 0040396E	
00403934	. 4A	DEC EDX	
00403935	. v 74 10	JZ SHORT 00403947	
00403937	. 81E3 0000FF00	AND EBX, 00FF0000	
0040393D	. 81E1 0000FF00	AND ECX, 00FF0000	
00403943	. 39D9	CMP ECX, EBX	
00403945	. v 75 27	JNE SHORT 0040396E	
00403947	> 01C0	ADD EAX, EAX	
00403949	. v EB 23	JMP SHORT 0040396E	
0040394B	> 8B57 FC	MOV EDX, DWORD PTR DS:[EDI-4]	
0040394E	. 29D0	SUB EAX, EDX	
00403950	. v EB 1C	JMP SHORT 0040396E	
00403952	> 8B46 FC	MOV ECX, DWORD PTR DS:[ESI-4]	
00403955	. 29D0	SUB EAX, EDX	
00403957	. v EB 15	JMP SHORT 0040396E	
00403959	> 5A	POP EDX	
0040395A	. 38D9	CMP CL, BL	
0040395C	. v 75 10	JNE SHORT 0040396E	
0040395E	. 38FD	CMP CH, BH	
00403960	. v 75 0C	JNE SHORT 0040396E	
00403962	. C1E9 10	SHR ECX, 10	
00403965	. C1EB 10	SHR EBX, 10	
00403968	. 38D9	CMP CL, BL	
0040396A	. v 75 02	JNE SHORT 0040396E	
0040396C	. 38FD	CMP CH, BH	
0040396E	> 5F	POP EDI	
0040396F	. 5E	POP ESI	
00403970	. 5B	POP EBX	
00403971	C3	RETN	

Hình 2

Test Case : caominhduc

### Phân tích mã assemble :

0042D534 MOV EAX, DWORD PTR SS:[EBP-4] <= load nội dung stack EBP-4 vào EAX  
Registers (FPU)  
EAX 022B5CD0 ASCII "caominhduc"

Khi đó thanh ghi EAX (022B5CD0) đang chứa chuỗi ("caominhduc")

0042D537 MOV EDX, 0042D590 <= truyền chuỗi "Binadryl" vào EDX  
EDX 0042D590 ASCII "Benadryl"

0042D53C CALL 004038D0 <= Gọi hàm ở địa chỉ 004038D0

004038D0 PUSH EBX

004038D1 PUSH ESI

004038D2 PUSH EDI

004038D3 MOV ESI, EAX <= Đưa giá trị từ thanh ghi EAX vào ESI  
ESI 021D5CD0 ASCII "caominhduc"

004038D5 MOV EDI, EDX <= Đưa giá trị từ thanh ghi EDX vào EDI  
EDI 0042D590 ASCII "Benadryl"

004038D7 CMP EAX, EDX <= so sánh chuỗi nhập vào với "Binadryl", nếu đúng thì xuất ra màn hình "thanks you made it" còn không thì "Wrong code DUDE"

## Tóm tắt thuật toán

- 1-Đọc **Enter Serial#** từ bàn phím người dùng.
- 2-Lưu chuỗi đã được nhập vào stack SS:EBP-4.
- 3-Load chuỗi đã nhập vào thanh ghi EAX từ stack SS: EBP -4.
- 4-Load chuỗi “Binadryl” vào thanh ghi EDX.
- 5-Gọi hàm so sánh .
- 6-Xuất ra màn hình thông báo thành công nếu chuỗi đã nhập trùng với “**Binadryl**”.
- 7-Nếu không trùng thì xuất ra thông báo không trùng.

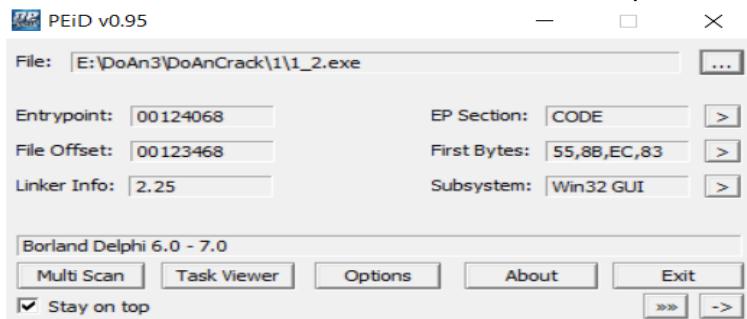
## Kết luận

So sánh chuỗi được nhập vào với **Binadryl**.

## 1.2

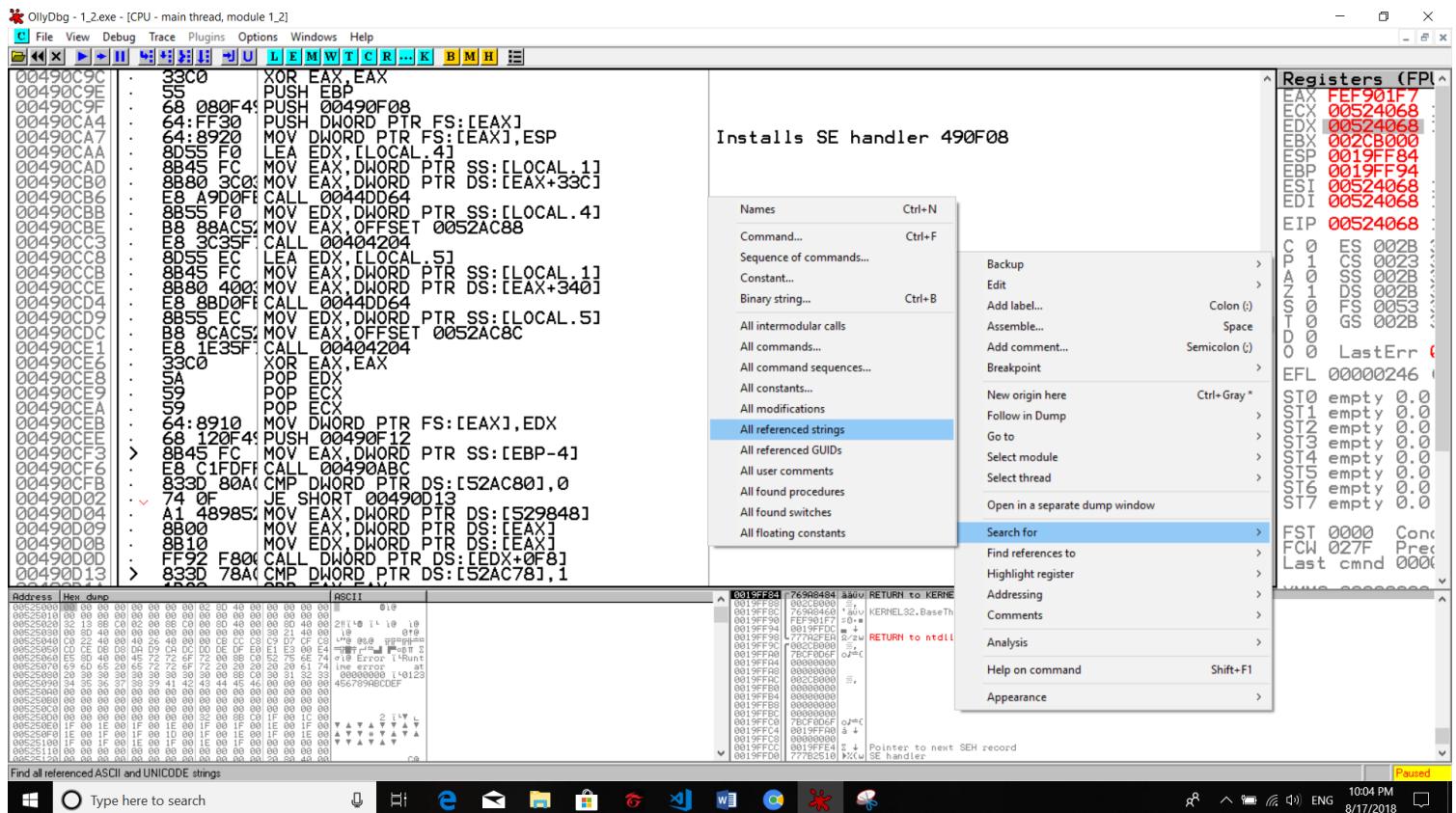
---

Đầu tiên, **Scan with PEiD 1.2.exe** để kiểm tra tệp thực tin này đã pack chưa.



Kết quả là **Borland Delphi 6.0-7.0**\*, nghĩa là 2.1.exe không bị pack, nó được viết bằng code **Borland Delphi**, tiến hành phân tích mã hợp ngữ bằng **OllyDbg** bình thường.

## Manh mồi



## Một cửa sổ mới xuất hiện, tìm đến thông báo in đỏ trong hình.

00490D33	MOV EAX,00490F6C
00490D5F	MOV EDX,00490FA0
00490D74	MOV EDX,00490FC8
00490D89	MOV EDX,00490FD8
00490DB6	MOV EDX,00490E8
00490DE3	MOV EDX,00490FFC
00490E8B	MOV EDX,00491040
00490E9E	MOV EDX,00491064
00490EB8	MOV EDX,00491088
00490ECB	MOV EDX,004910B0
00490F6C	ASCII "FreshUI has been"
00490FA0	ASCII "\Software\FreshD"

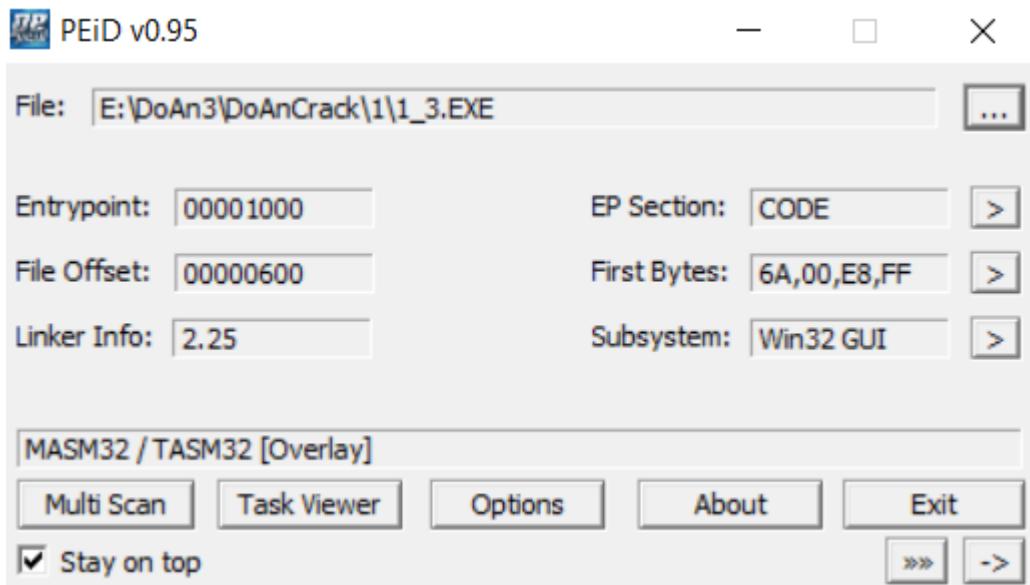
ASCII "FreshUI has been registered successfully."
ASCII "\Software\FreshDevices\FreshUI"
ASCII "Owner"
ASCII "RegCode"
ASCII "Registered"
ASCII "\ndfile\shell\Topic"
ASCII "About - [Business License]"
ASCII "About - [Personal License]"
ASCII "FreshUI - [Business License]"
ASCII "FreshUI - [Personal License]"
ASCII "FreshUI has been registered successfully."
ASCII "\Software\FreshDevices\FreshUI"

## 1.3

---

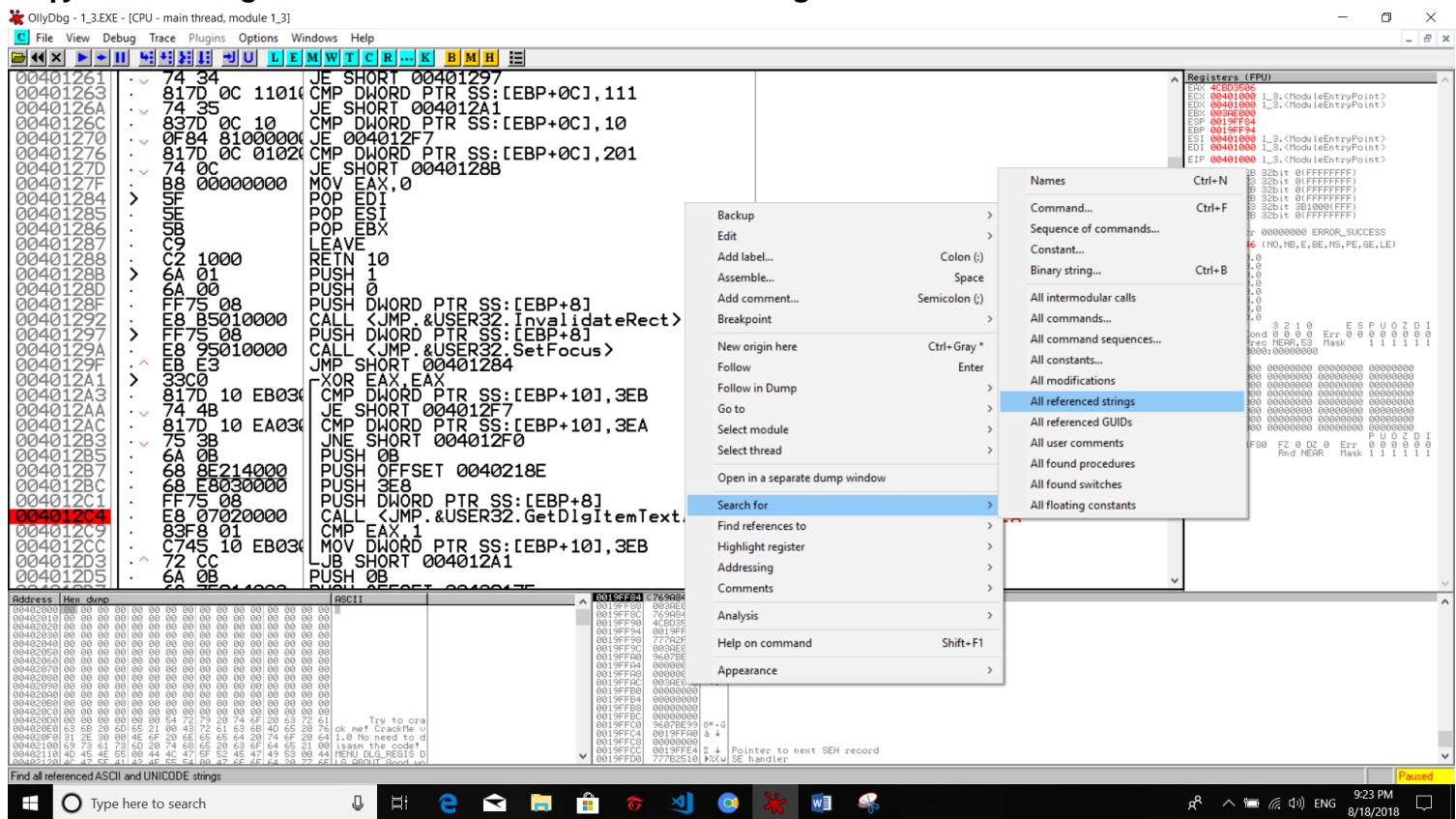
### Manh mồi

Đầu tiên, kiểm tra file thực thi có bị pack hay không bằng **PEiD**



Kết quả là **MASM32/ TASM32\***, nghĩa là 1.3.exe không bị pack mà nó viết bằng ngôn ngữ MASM32/ TASM32. Tiến hành phân tích mã hợp ngữ với **OllyDbg** bình thường.

## Chạy thử chương trình -> search for all referenced strings



Ta chú ý đến chuỗi được đánh dấu trong hình

Address	Command	Comments
0040100E	PUSH OFFSET 004020F4	ASCII "No need to disasm the code!"
00401077	MOV DWORD PTR DS:[402084],OFFSET 004021	ASCII "MENU"
00401081	MOV DWORD PTR DS:[402088],OFFSET 004020	ASCII "No need to disasm the code!"
00401087	PUSH OFFSET 004020E7	ASCII "CrackMe v1.0"
004010BC	PUSH OFFSET 004020F4	ASCII "No need to disasm the code!"
004011F7	PUSH OFFSET 0040211F	ASCII "DLG_ABOUT"
00401213	PUSH OFFSET 00402115	ASCII "DLG_REGIS"
0040134F	PUSH OFFSET 00402129	ASCII "Good work!"
<b>00401354</b>	<b>PUSH OFFSET 00402134</b>	<b>ASCII "Great work, mate!! Now try the next CrackMe!"</b>
0040136B	PUSH OFFSET 00402160	ASCII "No luck!"
00401370	PUSH OFFSET 00402169	ASCII "No luck there, mate!"
004013AF	PUSH OFFSET 00402160	ASCII "No luck!"
004013B4	PUSH OFFSET 00402169	ASCII "No luck there, mate!"

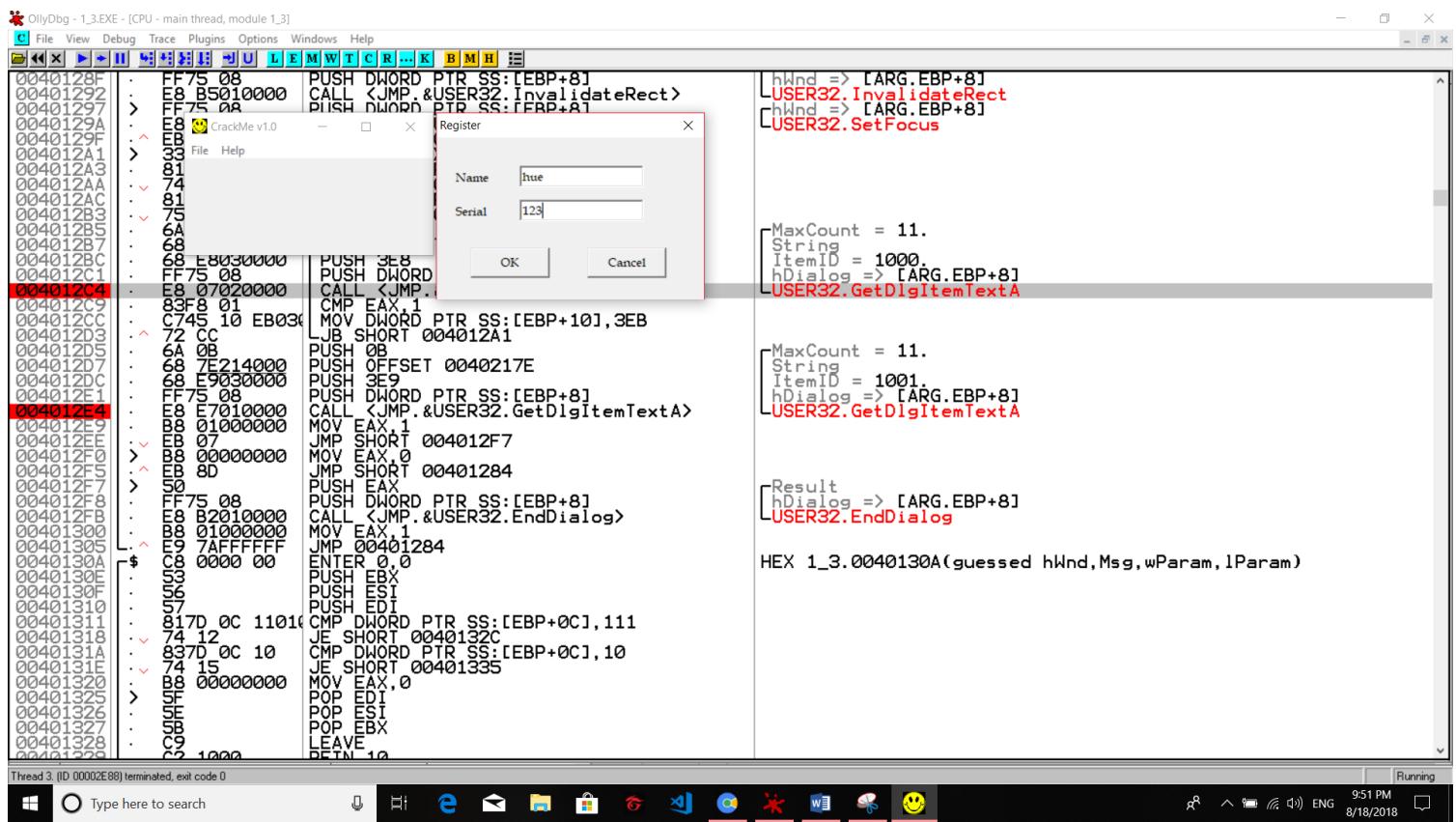
Go to vào đây, bắt đầu tìm kiếm manh mối, ta thấy rằng không có manh mối ở địa chỉ mà ta go to vào, nó chỉ là xuất ra màn hình thông báo bạn đã crack thành công, tiếp tục kéo lên trên tìm kiếm manh mối.

The screenshot shows the OllyDbg debugger interface with the assembly window open. The assembly code is displayed in the left pane, and the right pane shows the results of various API calls. Several callouts in red highlight specific function calls and their parameters:

- USER32.EndDialog**: Result = hDialog => [ARG.EBP+8]
- USER32.MessageBoxA**: Type = MB\_OKIMB\_ICONEXCLAMATION|MB\_DEFBUTTON1|MB\_APPLMODAL  
Caption = "Good work!"  
Text = "Great work, mate! Now try the next CrackMe!"  
hOwner => [ARG.EBP+8]
- USER32.MessageBoxA**: Type = MB\_OK  
Caption = "No luck!"  
Text = "No luck there, mate!"  
hOwner => [ARG.EBP+8]
- USER32.MessageBeep**: Type = MB\_OKIMB\_ICONEXCLAMATION|MB\_DEFBUTTON1|MB\_APPLMODAL  
Caption = "No luck!"  
Text = "No luck there, mate!"  
hOwner => [ARG.EBP+8]

The assembly code itself includes several calls to `MessageBoxA` and `MessageBeep`, which correspond to the highlighted callouts. The code is part of the `1_3` module, specifically at address `0040130A`.

Chạy thử chương trình, NAME: hue ; Serial: 123



Kéo lên trên, ta thấy xuất hiện chuỗi NAME và SERIAL mà ta nhập vào. Vậy chuyện gì đã xảy ra, ta tiến hành đặt

Breakpoint ở 00401223, 004012C4, 004012E4

OllyDbg - 1\_3.EXE - [CPU - main thread, module 1\_3]

```

00401209 > 6A 00 PUSH 0
0040120B . 68 53124000 PUSH 00401253
00401210 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
00401213 . 68 15214000 PUSH OFFSET 00402115
00401218 . FF35 CA204000 PUSH DWORD PTR DS:[4020CA]
0040121E . E8 7D020000 CALL <JMP. &USER32.DialogBoxParamA>
00401223 00401223 > 83F8 00 CMP EAX, 0
00401226 . ^ 74 BE JE SHORT 004011E6
00401228 . 68 8E214000 PUSH OFFSET 0040218E
0040122D . E8 4C010000 CALL 0040137E
00401232 . 50 PUSH EAX
00401233 . 68 7E214000 PUSH OFFSET 0040217E
00401238 . E8 9B010000 CALL 004013D8
0040123D . E8 83C4 04 ADD ESP, 4
00401240 . 58 POP EAX
00401241 . 3BC3 CMP EAX, EBX
00401243 . ^ 74 07 JE SHORT 0040124C
00401245 . E8 18010000 CALL 00401362
0040124A . EB 9A JMP SHORT 004011E6
0040124C > E8 FC000000 CALL 0040134D
00401251 . EB 93 JMP SHORT 004011E6
00401253 $ C8 0000 00 ENTER 0, 0
00401257 . 53 PUSH EBX
00401258 . 56 PUSH ESI
00401259 . 57 PUSH EDI
0040125A . 817D 0C 100100 CMP DWORD PTR SS:[EBP+0C], 110
00401261 . ^ 74 34 JE SHORT 00401297
00401263 . 817D 0C 110100 CMP DWORD PTR SS:[EBP+0C], 111
0040126A . ^ 74 35 JE SHORT 004012A1
0040126C . 837D 0C 10 CMP DWORD PTR SS:[EBP+0C], 10
00401270 . 0F84 81000000 JE 004012F7
00401276 . 817D 0C 010200 CMP DWORD PTR SS:[EBP+0C], 201
0040127D . ^ 74 0C JE SHORT 0040128B
0040127F . B8 00000000 MOV EAX, 0
00401284 > SF POP EDI
00401285 . SE POP ESI
00401286 . SB POP EBX
00401287 . C9 LEAVE
00401288 > C2 1000 RETN 10
0040128D . 6A 01 PUSH 1
0040128D . 6A 00 PUSH 0
0040128F . FF75 08 PUSH DWORD PTR SS:[EBP+8]
00401292 . E8 85010000 CALL <JMP. &USER32.InvalidateRect>
00401297 > FF75 08 PUSH DWORD PTR SS:[EBP+8]
0040129A . E8 95010000 CALL <JMP. &USER32.SetFocus>
0040129E . EB F3 IMP SHORT 00401294

```

InitParam = 0  
DialogProc = 1\_3.401253  
hParent => [ARG.EBP+8]  
TemplateName = "DLG\_REGIS"  
hInst = 00400000 ('I\_3')  
USER32.DialogBoxParamA

ASCII "hue"  
ASCII "123"

HEX 1\_3.00401253(guessed hWnd,Msg,wParam,lParam)

Breakpoint at 1\_3.00401223

OllyDbg - 1\_3.EXE - [CPU - main thread, module 1\_3]

```

0040127D . ^ 74 0C JE SHORT 0040128B
0040127F . B8 00000000 MOV EAX, 0
00401284 > SF POP EDI
00401285 . SE POP ESI
00401286 . SB POP EBX
00401287 . C9 LEAVE
00401288 > C2 1000 RETN 10
0040128D . 6A 01 PUSH 1
0040128D . 6A 00 PUSH 0
0040128F . FF75 08 PUSH DWORD PTR SS:[EBP+8]
00401292 > E8 85010000 CALL <JMP. &USER32.InvalidateRect>
00401297 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
0040129A . E8 E9010000 CALL <JMP. &USER32.SetFocus>
0040129E > EB E9010000 JMP SHORT 00401284
> 53C9 XOR EAX, EAX
004012A1 . 817D 10 EB0300 CMP DWORD PTR SS:[EBP+10], 3EB
004012AA . ^ 74 4B JE SHORT 004012F7
004012AC . 817D 10 EA0300 CMP DWORD PTR SS:[EBP+10], 3EA
004012B3 . ^ 75 2B JNE SHORT 004012F0
004012B5 . 6A 0B PUSH 0B
004012B7 . 68 8E214000 PUSH OFFSET 0040218E
004012C1 . 68 E8030000 PUSH 3EB
004012C4 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
004012C9 . E8 07020000 CALL <JMP. &USER32.GetDlgItemTextA>
004012CC . 83F8 01 CMP EAX, 1
004012D3 > C745 10 EB0300 MOV DWORD PTR SS:[EBP+10], 3EB
004012D5 . 6A 0B PUSH 0B
004012D7 . 68 7E214000 PUSH OFFSET 0040217E
004012DC . 68 E9030000 PUSH 3E9
004012E1 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
004012E9 . E8 E7010000 CALL <JMP. &USER32.GetDlgItemTextA>
004012EE . B8 01000000 MOV EAX, 1
004012F0 . EB 07 JMP SHORT 004012F7
004012F5 > EB 00000000 MOV EAX, 0
004012F7 . EB 8D JMP SHORT 00401284
004012F8 > 50 PUSH EAX
004012FB . FF75 08 PUSH DWORD PTR SS:[EBP+8]
00401300 . E8 B2010000 CALL <JMP. &USER32.EndDialog>
00401305 . B8 01000000 MOV EAX, 1
0040130A > E9 7AFFFFFF JMP 00401284
0040130A $ C8 0000 00 ENTER 0, 0
0040130F . 53 PUSH EBX
00401310 . 56 PUSH ESI
00401310 . 57 PUSH EDI
00401311 . 817D 0C 110100 CMP DWORD PTR SS:[EBP+0C], 111

```

Erase = TRUE  
Rect = NULL  
hWnd => [ARG.EBP+8]  
USER32.InvalidateRect  
hWnd => [ARG.EBP+8]  
USER32.SetFocus

MaxCount = 11;  
String = "hue";  
ItemID = 1000;  
hDialog => [ARG.EBP+8]  
USER32.GetDlgItemTextA

MaxCount = 11;  
String = "123";  
ItemID = 1001;  
hDialog => [ARG.EBP+8]  
USER32.GetDlgItemTextA

Result  
hDialog => [ARG.EBP+8]  
USER32.EndDialog

HEX 1\_3.0040130A(guessed hWnd,Msg,wParam,lParam)

Breakpoint at 1\_3.00401223

Ta tiến hành Debug ở địa chỉ **00401223** trước

00401223	83F8 00	CMP EAX, 0	
00401226	.^ 74 BE	JE SHORT 004011E6	ASCII "hue"
00401228	. 68 8E214000	PUSH OFFSET 0040218E	
0040122D	. E8 4C010000	CALL 0040137E	
00401232	. 50	PUSH EAX	
00401233	. 68 7E214000	PUSH OFFSET 0040217E	ASCII "123"
00401238	. E8 9B010000	CALL 004013D8	
0040123D	. 83C4 04	ADD ESP, 4	
00401240	. 58	POP EAX	
00401241	. 3BC3	CMP EAX, EBX	
00401243	.^ 74 07	JE SHORT 0040124C	
00401245	. E8 18010000	CALL 00401362	
0040124A	.^ E8 9A	JMP SHORT 004011E6	
0040124C	> ^ E8 FC000000	CALL 0040134D	
00401251	.^ E8 93	JMP SHORT 004011E6	
00401253	\$ C8 0000 00	ENTER 0, 0	HEX 1_3.00401253(guessed hWnd,Msg,wParam,lParam)
00401257	. 53	PUSH EBX	
00401258	. 56	PUSH ESI	
00401259	. 57	PUSH EDI	
0040125A	. 817D 0C 10010	CMP DWORD PTR SS:[EBP+0C], 110	
00401261	.^ 74 34	JE SHORT 00401297	
00401263	. 817D 0C 11010	CMP DWORD PTR SS:[EBP+0C], 111	
0040126A	.^ 74 35	JE SHORT 004012A1	
0040126C	. 837D 0C 10	CMP DWORD PTR SS:[EBP+0C], 10	
00401270	.^ 0F84 81000000	JE 004012F7	
00401276	. 817D 0C 01020	CMP DWORD PTR SS:[EBP+0C], 201	
0040127D	.^ 74 0C	JE SHORT 0040128B	
0040127F	. B8 00000000	MOV EAX, 0	
00401284	> 5F	POP EDI	
00401285	. 5E	POP ESI	
00401286	. 5B	POP EBX	
00401287	. C9	LEAVE	
00401288	. C2 1000	RETN 10	
00401289	> 6A 01	PUSH 1	
0040128D	. 6A 00	PUSH 0	
0040128F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	Erase = TRUE
00401292	. E8 B5010000	CALL <JMP.&USER32.InvalidateRect>	Rect = NULL
00401297	> FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd => [ARG.EBP+8]
0040129A	. E8 95010000	CALL <JMP.&USER32.SetFocus>	USER32.InvalidateRect
0040129F	.^ EB E3	JMP SHORT 00401284	hWnd => [ARG.EBP+8]
004012A1	> 33C0	XOR EAX, EAX	USER32.SetFocus
004012A3	. 817D 10 EB030	CMP DWORD PTR SS:[EBP+10], 3EB	
004012AA	.^ 74 4B	JE SHORT 004012F7	
004012AC	. 817D 10 EA030	CMP DWORD PTR SS:[EBP+10], 3EA	
004012B3	.^ 75 3B	JNE SHORT 004012F0	
004012B5	. FA 00	PUSH 0B	

00401223	CMP EAX, 0
00401226	JE SHORT 004011E6

*Lệnh nhảy này sẽ không được thực hiện vì EAX đang mang giá trị là 1*

Registers (FPU)  
EAX 00000001

00401228	:	68 8E214000	PUSH OFFSET 0040218E	ASCII "hue"
0040122D	:	E8 4C010000	CALL 0040137E	
00401232	:	50	PUSH EAX	

00401228 PUSH OFFSET 0040218E <= Đây là chuỗi NAME mà chúng ta vừa nhập lúc nãy  
0040122D CALL 0040137E <= gọi hàm ở địa chỉ 0040137E, chúng ta sẽ follow cái hàm này

Hàm ở địa chỉ 0040137E

0040137E	- \$	8B7424 04	MOV ESI, DWORD PTR SS:[ARG.1]
00401382	.	56	PUSH ESI
00401383	>	8A06	MOV AL, BYTE PTR DS:[ESI]
00401385	.	84C0	TEST AL, AL
00401387	.	74 13	JZ SHORT 0040139C
00401389	.	3C 41	CMP AL, 41
0040138B	.	72 1F	JB SHORT 004013AC
0040138D	.	3C 5A	CMP AL, 5A
0040138F	.	73 03	JAE SHORT 00401394
00401391	.	46	INC ESI
00401392	.	EB EF	JMP SHORT 00401383
00401394	>	E8 39000000	CALL 004013D2
00401399	.	46	INC ESI
0040139A	.	EB E7	JMP SHORT 00401383
0040139C	>	5E	POP ESI
0040139D	.	E8 20000000	CALL 004013C2
004013A2	.	81F7 78560000	XOR EDI, 00005678
004013A8	.	8BC7	MOV EAX, EDI
004013AA	.	EB 15	JMP SHORT 004013C1
004013AC	>	5E	POP ESI
004013AD	.	6A 30	PUSH 30
004013AF	.	68 60214000	PUSH OFFSET 00402160
004013B4	.	68 69214000	PUSH OFFSET 00402169
004013B9	.	FF75 08	PUSH DWORD PTR SS:[EBP+8]
004013BC	.	E8 79000000	CALL <JMP.&USER32.MessageBoxA>
004013C1	>	C3	RETN
004013C2	\$	33FF	XOR EDI, EDI
004013C4	.	33DB	XOR EBX, EBX
004013C6	>	8A1E	MOV BL, BYTE PTR DS:[ESI]
004013C8	.	84DB	TEST BL, BL
004013CA	.	74 05	JZ SHORT 004013D1
004013CC	.	03FB	ADD EDI, EBX
004013CE	.	46	INC ESI
004013CF	.	EB F5	JMP SHORT 004013C6
004013D1	>	C3	RETN
004013D7	.	C3	RETN
004013D8	\$	33C0	XOR EAX, EAX
004013DA	.	33FF	XOR EDI, EDI
004013DC	.	33DB	XOR EBX, EBX
004013DE	.	8B7424 04	MOV ESI, DWORD PTR SS:[ARG.1]
004013E2	>	B0 0A	MOV AL, 0A
004013E4	.	8A1E	MOV BL, BYTE PTR DS:[ESI]
004013E6	.	84DB	TEST BL, BL
004013E8	.	74 0B	JZ SHORT 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 004013E2
004013F5	>	81F7 34120000	XOR EDI, 00001234
004013FB	.	8BDF	MOV EBX, EDI
004013FD	.	C3	RETN

<b>0040137E</b>	<b>MOVE ESI, DWORD PTR SS:[ARG.1]</b> <= move chuỗi NAME vào ESI
00401382	PUSH ESI
00401383	MOV AL, BYTE PTR DS:[ESI] <= AL = Mã ASCII từng ký tự trong chuỗi name
00401385	TEST AL,AL <= kiểm tra xem AL có bằng 0 hay không
00401387	JZ SHORT 0040139C <= nếu AL = 0 thì nhảy tới lệnh ở địa chỉ 0040139C
00401389	CMP AL, 41 <= so sánh AL với ký tự 'A' mang mã 41 trong ASCII
0040138B	JB SHORT 004013AC <= nếu nhỏ hơn 41 thì nhảy tới lệnh ở địa chỉ 004013AC
0040138D	CMP AL, 5A <= nếu không nhỏ hơn 41 thì so sánh với 5A chính là ký tự 'Z' trong mã ASCII
0040138F	JAE SHORT 00401394 <= nhảy khi lớn hơn hoặc bằng 5A tới địa chỉ 00401394, tại địa chỉ này gọi hàm ở địa chỉ 004013D2, hàm này lấy AL – 0x20, có nghĩa là chuyển ký tự AL từ chữ thường thành chữ hoa
Hàm trên sẽ lặp lại cho	đến khi chuỗi NAME được chuyển thành ký tự hoa hết, sau đó gọi hàm ở địa chỉ 004013C2
<b>004013C2</b>	XOR EDI, EDI
004013C4	XOR EBX, EBX
004013C6	MOV BL, BYTE PTR DS:[ESI] <= BL = mã ASCII từng ký tự trong chuỗi NAME ( đã dc chuyển thành chữ hoa)
004013C8	TEST BL, BL <= test xem BL có bằng 0 hay không
004013CA	JZ SHORT 004013D1 <= nếu bằng 0 thì nhảy đến lệnh ở địa chỉ 004013D1
004013CC	ADD EDI, EBX <= cộng lại , hàm này sẽ chạy cho tới khi cộng hết tổng của các ký tự trong chuỗi NAME rồi nhảy về địa chỉ 004013A2
<b>004013A2</b>	XOR EDI, 00005678 <= tổng ASCII các ký tự trong chuỗi NAME chuyển sang chữ hoa XOR với 0x56578h
<b>004013D8</b>	<b>XOR EAX, EAX</b>
004013DA	XOR EDI, EDI
004013DC	XOR EBX, EBX
004013DE	MOV ESI, DWORD PTR SS:[ARG.1] <= ESI chứa SERIAL
004013E2	MOV AL, 0A <= gán AL =0Ah
004013E4	MOV BL, BYTE PTR SS:[ARG.1] <= BL = mã ASCII từng ký tự trong SERIAL
004013E6	TEST BL,BL <= test xem BL có bằng 0 hay không
004013E8	JZ SHORT 004013F5 <= nếu bằng 0 thì nhảy đến lệnh ở 004013F5
004013EA	SUB BL, 30 <= từng ký tự trong SERIAL chuyển sang mã ASCII – 0x30. Trong ASCII 0x30 chính là ký tự '0'. Suy ra đang biến từng ký tự trong chuỗi SERIAL về số hệ thập phân.
004013ED	IMUL EDI, EAX <= EDI = EDI * 0A

004013F0	ADD EDI,EBX <= EDI = (EDI * 0A ) + (từng ký tự - 0x30)
Hàm này sẽ lặp lại	cho tới hết các ký tự trong chuỗi SERIAL
Xong khi làm xong thì	Sẽ nhảy tới địa chỉ
004013F5	XOR EDI, 00001234 <= <b>đem kết quả ở trên XOR 0x1234</b>

*Sau đó chương trình sẽ So sánh 2 kết quả trên, nếu trùng nhau thì thông báo thành công, nếu không thì thông báo thất bại*

## Tóm tắt thuật toán

- NAME :

```

1  result = 0
2  foreach char in name:
3      if char < 0x41:
4          ERROR(Invalid Name)
5      else:
6          if char > 0x5A:
7              char = char - 0x20
8          result = result + char
9  result = result XOR 0x5678
10 return result

```

- SERIAL:

```

1  result = 0
2  foreach char in serial:
3      result = result * 0xA
4      result = result + (char - 0x30)
5  result = result XOR 0x1234
6  return result

```

- Sau khi biến đổi xong, ta đem so sánh 2 kết quả này với nhau, nếu trùng thì thành công, ngược lại thì thất bại.

- Ta còn có thể kết luận như sau:  
Tổng ASCII các ký tự của chuỗi NAME chuyển sang chữ hoa (hệ 16) XOR 0x5678h = SERIAL thập phân XOR 0x1234h
- Một tính chất ta cần nhớ để viết KEYGEN : A XOR B = C => C XOR B = A

## Tổng kết

---

### Mức độ hoàn thành

1.1	100%
2.1	
2.2	100%

### Kiến thức thu được

Học cách sử dụng PEiD, OllyDbg, cách đọc mã hợp ngữ.

Mỗi crackme sử dụng một cách phát sinh key riêng, làm nhiều thì kinh nghiệm càng cao. Trong đó có những kĩ thuật thú vị như: tạo file rồi chuyển quyền cho hệ điều hành thực thi file đó (như 2.1), hay thuật toán phát sinh key dựa trên input của người dùng (như 1.1, 2.2). Ngoài ra, bài mà chúng tôi mất thời gian nhất, nhưng cũng gây kích thích nhất là 2.2 cũ. Thuật toán phát sinh key của bài khá phức tạp: key gồm 12 ký tự, mã hóa lần 1: mã hóa 4 ký tự đầu thành 16 ký số hexa, mã hóa lần 2: dùng 16 ký số hexa cùng với bảng băm của thuật toán mã hóa CRC32...

## Tham khảo

---

1. P.E Onimusha, loạt bài **Basic Cracking Tutorial**
2. \_kienmanowar\_, loạt bài **Ollydbg\_tut**
3. Mirror của crackmes.de: <http://crackmes.cf/users/>
4. Các link trong Link tham khảo thêm trên moodle:
  - 4.1. Introduce x86 32 bit: <http://www.cs.virginia.edu/~evans/cs216/guides/x86.html> 4.2.  
Introduce x86 64 bit  
[https://wiki.cdot.senecacollege.ca/wiki/X86\\_64\\_Register\\_and\\_Instruction\\_Quick\\_Start](https://wiki.cdot.senecacollege.ca/wiki/X86_64_Register_and_Instruction_Quick_Start)
  - 4.3. Understanding C by learning assembly: <https://www.recurse.com/blog/7-understanding-c-by-learning-assembly> <https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/> (32 bits)  
<https://eli.thegreenplace.net/2011/09/06/stack-frame-layout-on-x86-64> (64 bits)
  - 4.4. Intel and AT&T Syntax: <http://www.imada.sdu.dk/Courses/DM18/Litteratur/IntelnATT.htm>
  - 4.5. JUMP quick reference: <http://unixwiz.net/techtips/x86-jumps.html>
5. <https://stackoverflow.com/>