

الفصل الخامس



حماية السيريال

حماية السيريال

تمرين:

البرنامج: crack1.exe

الأدوات: ollydbg

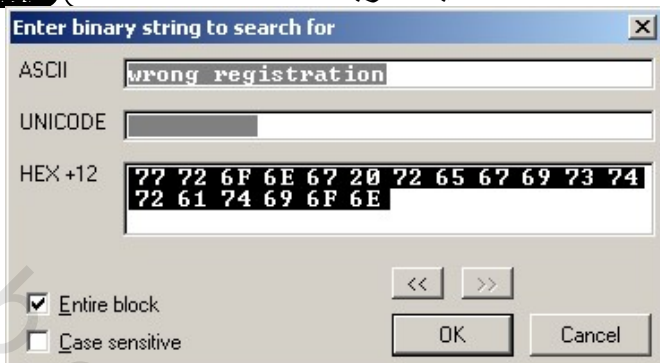
هذا البرنامج يحتوى على مكان لإدخال السريال الصحيح ويعطى رسالة خطأ في حالة عدم إدخال سريال صحيح.



قم الآن بإدخال أي رقم مثلاً 111111111111 واضغط على المفتاح register ستجد البرنامج يظهر رسالة الخطأ "wrong registration try again"



قم بإغلاق البرنامج وتحميله ببرنامج ollydbg واضغط على Ctrl+B وابحث عن نص الرسالة السابقة.



بعد الذهاب إلى مكان الرسالة اذهب لأعلى سطر واحد ستجد أمر JE وهو أمر الاختبار الذي نريده وبعده يحدد البرنامج ما إذا كان السريال صحيح أم لا >

Address	Hex dump	Disassembly	Comment
0045884F	. 64:8920	MOV DWORD PTR FS:[EAX], ESP	
00458852	. 8055 FC	LEA EDI, DWORD PTR SS:[EBP-4]	
00458855	. 8B83 F0200000	MOV EAX, DWORD PTR DS:[EBX+2F0]	
00458858	. E8 84F0FDF	CALL crack1.00437BE4	
00458860	. 8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]	
00458863	. BA C3884500	MOV EDI, crack1.004588C8	ASCII "311-487-8974"
00458865	. E8 438FAFF	CALL crack1.004046B0	
00458867	. 74 19	JE SHORT crack1.004588B8	
0045886F	. B8 E0884500	MOV EAX, crack1.004588E0	ASCII "Wrong registration"
00458874	. E8 4B8EFDF	CALL crack1.004319C4	
00458879	. 3B02	XOR EDI, EDI	
0045887B	. 8B83 F0200000	MOV EAX, DWORD PTR DS:[EBX+2F0]	
0045887E	. E8 8EF0FDF	CALL crack1.00437C14	
00458881	. EB 17	JMP SHORT crack1.0045889F	
00458888	. B8 088C4500	MOV EAX, crack1.00458C08	ASCII "Application successful"
00458890	. E8 328EFDF	CALL crack1.004319C4	
00458894	. 8B83 F0200000	MOV EAX, DWORD PTR DS:[EBX+2F0]	
00458899	. 8B08	MOV ECX, DWORD PTR DS:[EAX]	
0045889C	. FF51 64	CALL DWORD PTR DS:[ECX+64]	

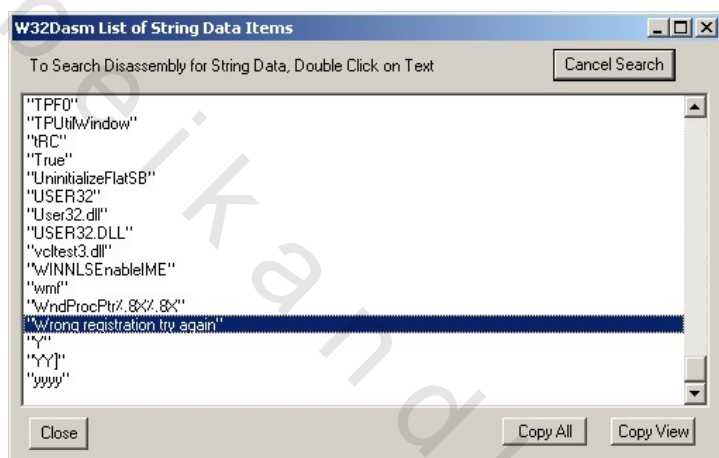
Jump is NOT taken
00458888=crack1.004588B8

ستجد قبل الرسالة نص مهم وهو 311-487-8974 تم تخزينه في المسجل EDX وهو المسجل الذي يتم غالبا التخزين فيه لمقارنة السريال الصحيح فإذا جربت هذا الرقم ستجد انه السريال الصحيح >

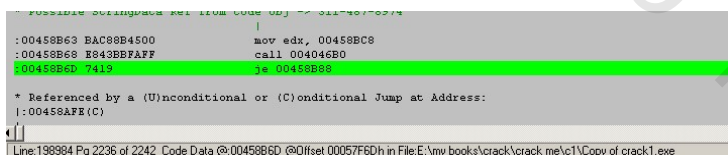
طريقة أخرى:

عادة يكون إيجاد السريال الصحيح ليس بهذه البساطة لذلك يمكن إجراء تعديل سريع للأمر JE وتحويله للأمر JNE حتى يتجاهل البرنامج عملية اختبار السريال .

قم بفتح الملف في برنامج w32dasm ثم ابحث في قسم النصوص عن الرسالة "wrong registration"



اضغط ضغط مزدوج على الرسالة ثم اذهب إلى السطر السابق لها وستجد الأمر القفز JE عند العنوان 57F6D



و الآن اذهب إلى برنامج hiew وافتح البرنامج ثم اضغط F5 واكتب العنوان واضغط enter

```

Hiew: Copy of crack1.exe
Copy of crack1> 4FRO ----- a16 PE.00400000 Hiew 7.21 <c>SEN
57F6d
00400001: 50      dec     bp
00400002: 58      pop     dx
00400003: 0002    push    ax
00400005: 0000    add     [bp+fsi],al
00400007: 0004    add     [bx+fsi],al
00400009: 000F    [si],al
0040000B: 00FF    add     [bx],cl
0040000D: FF00    add     bh,bh
0040000F: FF00    inc     w,[bx+fsi]
00400011: 0000    add     [bx+fsi+0000],bh
00400013: 0000    add     [bx+fsi],al
00400015: 0000    add     [bx+fsi],al
00400017: 004000 add     [bx+fsi+000],al
0040001A: 1A00    shb     al,[bx+fsi]
0040001C: 0000    add     [bx+fsi],al
0040001E: 0000    add     [bx+fsi],al
00400020: 0000    add     [bx+fsi],al
00400022: 0000    add     [bx+fsi],al
00400024: 0000    add     [bx+fsi],al
00400026: 0000    add     [bx+fsi],al
00400028: 0000    add     [bx+fsi],al
0040002A: 0000    add     [bx+fsi],al
0040002C: 0000    add     [bx+fsi],al
1      2      3      4      5      6      7      8      9      10

```

```

Hiew: Copy of crack1.exe
Copy of crack1> 4FRO ----- a32 PE.00450B6D Hiew 7.21 <c>SEN
00450B44: 33C0    xor     eax,eax
00450B46: 55      ebp     push
00450B47: 68B58D4500 push    000458BB5 ---> (1)
00450B4C: 64FF30  d.fs:[eax]
00450B4F: 648920    fs:[eax],esp
00450B52: 8D55FC    lea     edx,[ebp+04]
00450B55: 8B83F0020000 mov     eax,[ebx+000002F0]
00450B5D: E84F0FDFF call    000437E4 ---> (2)
00450B60: 8B45FC    mov     eax,[ebp+04]
00450B63: BAC88D4500 mov     edx,000458BC8 ; 311-487-897
00450B66: F843BBFAFF call    00040460 ---> (4)
00450B6D: 7419    je      000458B88 ---> (5)
00450B6F: B8588D4500 mov     eax,000458BE0 ; Wrong regis
00450B74: E84B8EFDFF call    0004319C4 ---> (7)
00450B79: 33D2    xor     edx,edx
00450B7D: 8B83F0020000 mov     eax,[ebx+000002F0]
00450B81: E88E0FDFF call    000437C14 ---> (8)
00450B86: EB17    jmp     000458B9F ---> (9)
00450B88: B8088D4500 mov     eax,000458C08 ; Application
00450B8D: E8328EFDFF call    0004319C4 ---> (B)
00450B92: 33D2    xor     edx,edx
00450B94: 8B83F0020000 mov     eax,[ebx+000002F0]
00450B9A: 8B08    mov     ecx,[eax]
1 Help 2 PutBk 3 Edit 4 Mode 5 Goto 6 Refer 7 Search 8 Loader 9 Files 10 Quit

```

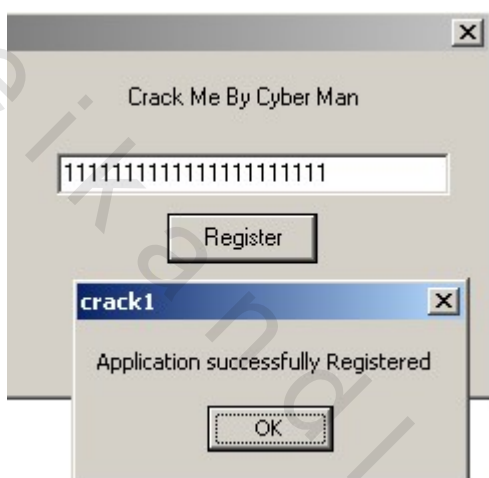
و الآن اضغط المفتاح F3 ثم المفتاح F2 لتعديل أمر الاسمبلي

```

Hiew: Copy of crack1.exe
Copy of crack1> 4FWO EDITMODE a32 PE.00057F6D Hiew 7.21 <c>SEN
00057F44: 33C0    xor     eax,eax
00057F46: 55      ebp     push
00057F47: 68B58D4500 push    000458BB5 ; 'E1'
00057F4C: 64FF30  d.fs:[eax]
00057F4F: 648920    fs:[eax],esp
00057F52: 8D55FC    lea     edx,[ebp+04]
00057F55: 8B83F0020000 mov     eax,[ebx+000002F0]
00057F5B: E884F0FDFF call    000036FE4
00057F60: 8B45FC    mov     eax,[ebp+04]
00057F63: B8388D4500 mov     edx,000458BC8 ; 'E1'
00057F68: E843BBFAFF call    00003A00
00057F6D: 7419    je      000057F88
00057F6F: B8588D4500 mov     eax,000458BE0 ; 'E1'
00057F74: E84B8EFDFF call    000030DC4
00
00
00
00
Assembler
je      000057F88
32-bit opcode
00057F8D: E8328EFDFF call    000030DC4
00057F92: 33D2    xor     edx,edx
00057F94: 8B83F0020000 mov     eax,[ebx+000002F0]
00057F9A: 8B08    mov     ecx,[eax]
1 Help 2 PutBk 3 Edit 4 Mode 5 Goto 6 Refer 7 Search 8 Loader 9 Files 10 Quit

```

حول الأمر السابق من JE 000057F88 إلى JNE 000057F88
اضغط على المفتاح F9 لحفظ التعديلات ثم اغلق برنامج hiew وافتح الآن
البرنامج crack1 وادخل أي أرقام واضغط مفتاح register وستجد أن
البرنامج تم تسجيله بنجاح بسريال خطأ.

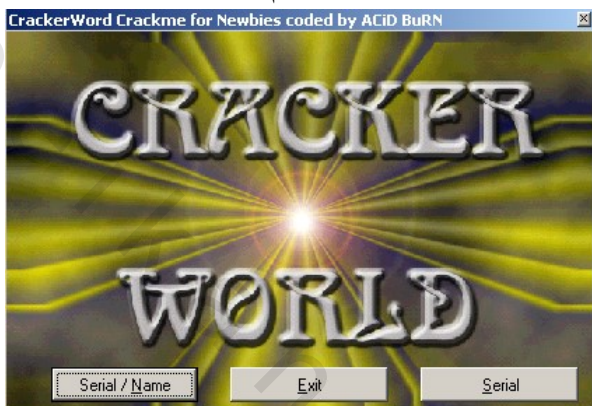


تمرين:

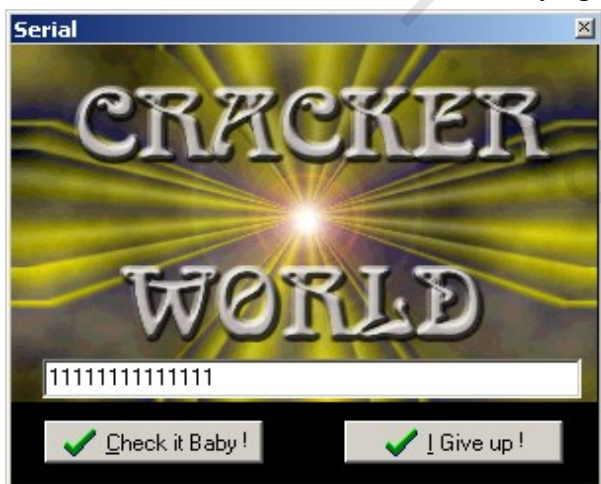
البرنامج: Crack Me

الأدوات: ollydbg

هذا البرنامج يحتوى على تدريب لإدخال السريال الصحيح وآخر لإدخال الاسم والسريال وسنعرف الآن كيفية فك كل منهم .



قم الآن بالضغط على المفتاح serial وحاول إدخال أي رقم ستجد ظهور رسالة خطأ "try again"





قم بتحميل الملف في برنامج ollydbg وابحث عن الرسالة "try again"

CPU - main thread, module crackme				
Address	Hex dump	Disassembly	Comment	
0042FAF3	. E8 60AFFEFF	CALL crackme.0041A958		
0042FAF8	. 8B55 F0	MOV EDX, DWORD PTR SS:[EBP-10]		
0042FAB8	. 8B45 F4	MOV EAX, DWORD PTR SS:[EBP-C]		
0042FAFE	. E8 F93EFDFF	CALL crackme.004039FC		
0042FB03	. 75 1A	JNZ SHORT crackme.0042FB1F		
0042FB05	. 6A 00	PUSH 0		
0042FB07	. B9 CCFB4200	MOV ECX, crackme.0042FB0C	ASCII "Congratz !!"	
0042FB0C	. BA D8FB4200	MOV EDX, crackme.0042FB08	ASCII "Good job dude =)"	
0042FB11	. A1 430A4300	MOV EAX, DWORD PTR DS:[430A48]		
0042FB16	. 8B00	MOV EAX, DWORD PTR DS:[EAX]		
0042FB18	. E8 53A6FFFF	CALL crackme.0042A170		
0042FB1D	. EB 18	JMP SHORT crackme.0042FB37		
0042FB1F	. 6A 00	PUSH 0		
0042FB21	. B9 74FB4200	MOV ECX, crackme.0042FB74	ASCII "Try Again!"	
0042FB26	. BA 08FB4200	MOV EDX, crackme.0042FB08	ASCII "Sorry , The serial	
0042FB2B	. A1 430A4300	MOV EAX, DWORD PTR DS:[430A48]		
0042FB30	. 8B00	MOV EAX, DWORD PTR DS:[EAX]		
0042FB32	. E8 39A6FFFF	CALL crackme.0042A170		
0042FB37	. 33C0	XOR EAX, EAX		
0042FB39	. 5A	POP EDX		

قم الآن بالبحث عن أمر القفز ويكون دائما في السطور السابقة للمكان الحالي
 وقم بوضع نقطة إيقاف على هذا المكان F2
 استمر في البحث عن رسالة الخطأ وقم بوضع نقط إيقاف في كل أمر قفز
 يسبق الرسالة

Address	Hex dump	Disassembly	Comment	Register
0042FAF3	. 75 1A	JNZ SHORT crackme.0042F4F1		EAX: 0000
0042FAD7	. 6A 00	PUSH 0		ECX: 313
0042FAD9	. B9 64F54200	MOV ECX, crackme.0042F564	ASCII "Congratz!"	EDX: 0000
0042FADE	. BA 70F54200	MOV EDX, crackme.0042F570	ASCII "Good Job dude"	EBX: 0000
0042FAE3	. A1 430A4300	MOV EAX, DWORD PTR DS:[430A48]		ESP: 001
0042FAE8	. 8B00	MOV EAX, DWORD PTR DS:[EAX]		EBP: 001
0042FAE4	. E8 51ACFFFF	CALL crackme.0042A170		ESI: 0000
0042FAEF	. EB 18	JMP SHORT crackme.0042F509		EDI: 0000
0042F4F1	. 6A 00	PUSH 0		EBP: 0000

Breakpoints				
Address	Module	Active	Disassembly	Comment
0042F4F1	crackme	Always	JNZ SHORT crackme.0042F4F1	
0042FAF3	crackme	Always	JMP SHORT crackme.0042FA79	
0042FB03	crackme	Always	JNZ SHORT crackme.0042FB1F	

يمكنك رؤية جميع نقاط التوقف في نافذة breakpoints قم الآن بالضغط على المفتاح F9 وحاول أن تدخل أي رقم خطأ وليكن 1111111111 واضغط على مفتاح check it ستجد برنامج ollydbg وقف عن أمر القفز الذي قمنا بوضعه ويمكنك الآن النظر مباشرة في نافذة stack التي تحتوى نصوص موجودة بالذاكرة ويمكنك أن ترى السريال الخطأ الذي قمت بإدخاله 111111111111 وأسطفه النص "hello dude" لسنا بالطبع متأكدين إذا كان هذا هو السريال الصحيح لكن التجربة لن تضر

```

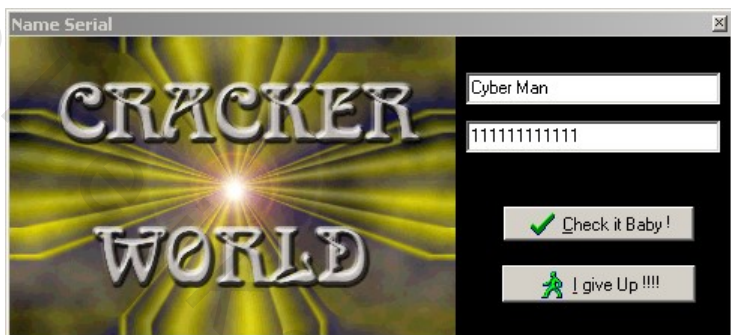
0012F9A8 0012FAB8 Pointer to next SEH record
0012F9AC 0042F52C SE handler
0012F9B0 0012F9C8
0012F9B4 0095975C
0012F9B8 0095A4C8 ASCII "1111111111111111"
0012F9BC 0095A50C ASCII "Hello Dude!"
0012F9C0 0042F550 ASCII "Dude?"
0012F9C4 0042F540 ASCII "Hello"
0012F9C8 0012FA40
0012F9CC 0041B457 RETURN to crackme.0041B457
0012F9D0 0095975C
0012F9D4 0042C6E5 RETURN to crackme.0042C6E5 from crackme.0041B43C
0012F9D8 0095975C
0012F9DC 0042EC26 RETURN to crackme.0042EC26 from crackme.0042C6C4
0012F9E0 0042EBBC crackme.0042EBBC
0012F9E4 0042C7D5 RETURN to crackme.0042C7D5 from crackme.00402CD0
0012F9E8 0012FA30
0012F9EC 0041B31D RETURN to crackme.0041B31D from crackme.00402D10
0012F9F0 0095975C

```

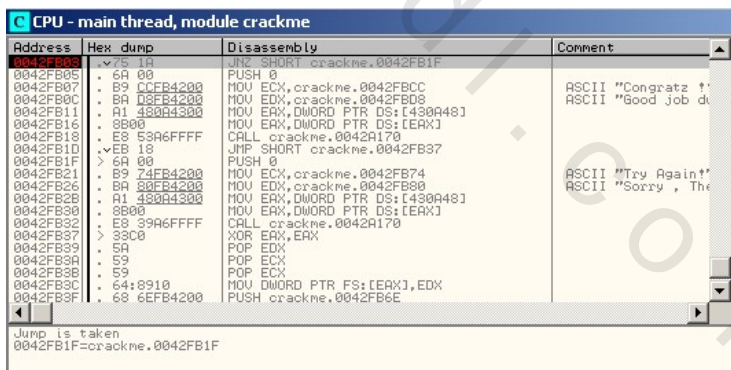
و الآن قم بإدخال النص "Hello Dude!"



و الآن قم بإعادة نفس الخطوات لكن لإدخال الاسم والسريال وحمل الملف بالطبع في برنامج ollydbg ومن مزايا هذا البرنامج انه يقوم بحفظ نقاط الإيقاف السابقة فلا داعي للبحث مره أخرى عن نص الرسالة الخطأ



و ستجد البرنامج وقف عند أمر القفز كما بالشكل



بعد الضغط على المفتاح F9 لاحظ الآن نافذة stack ستجد السريال الصحيح

```

0012F99C 0012FAB8 Pointer to next SEH record
0012F9A0 0042FB67 SE handler
0012F9A4 0012F9C8
0012F9A8 009512C8
0012F9AC 009512C8
0012F9B0 0095A4BC ASCII "5494"
0012F9B4 0095A4A4 ASCII "Cyber Man"
0012F9B8 0095A4EC ASCII "111111111111"
0012F9BC 0095A4D0 ASCII "CW-5494-CRACKED"
0012F9C0 0042FBB8 ASCII "CRACKED"
0012F9C4 0042FBAC ASCII "CW"
0012F9C8 0012FA40
0012F9CC 0041B457 RETURN to crackme.0041B457
0012F9D0 009512C8
0012F9D4 0042C6E5 RETURN to crackme.0042C6E5 from crackme.0041B43C
0012F9D8 009512C8
0012F9DC 0042EC26 RETURN to crackme.0042EC26 from crackme.0042C6C4
0012F9E0 0042EBBC crackme.0042EBBC
0012F9E4 0042C7DE RETURN to crackme.0042C7DE from crackme.0041B43C

```

و النتيجة

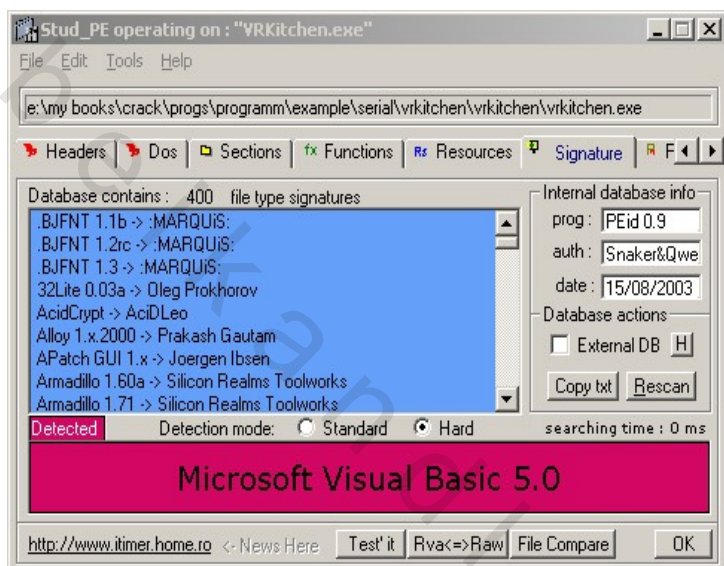


برنامج VRKitchen:

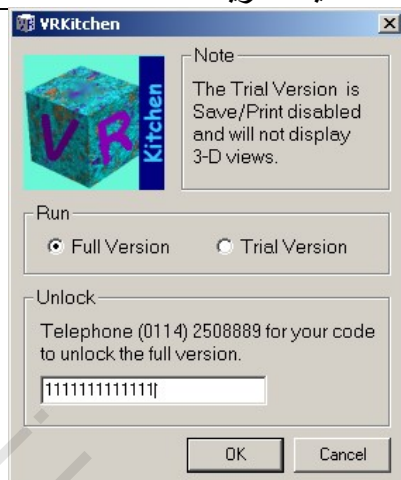
الأدوات: smartcheck

هذا البرنامج مصنوع بلغة visual basic وللتأكد من اللغة يمكنك استخدام

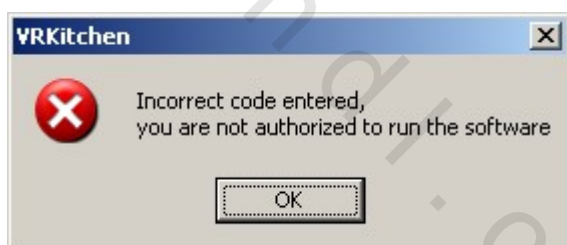
برنامج stud_pe



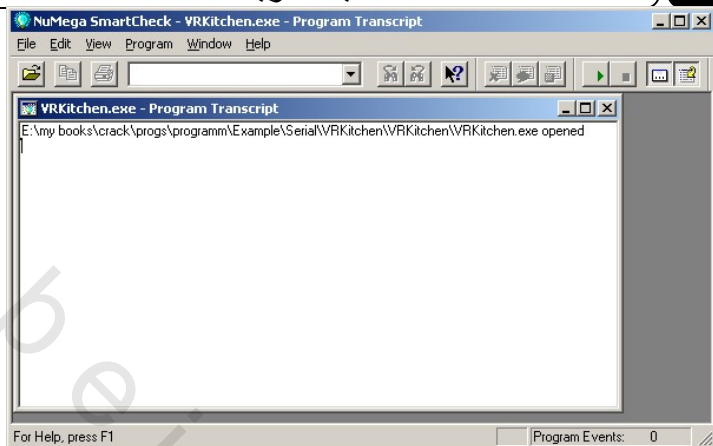
وأفضل أداة للغة فيجوال بيزيك هي smartcheck وهى الأداة التي سنقوم باستخدامها قم الآن بتشغيل البرنامج واختر الاختيار Full Version وقم بإدخال أي سريال.



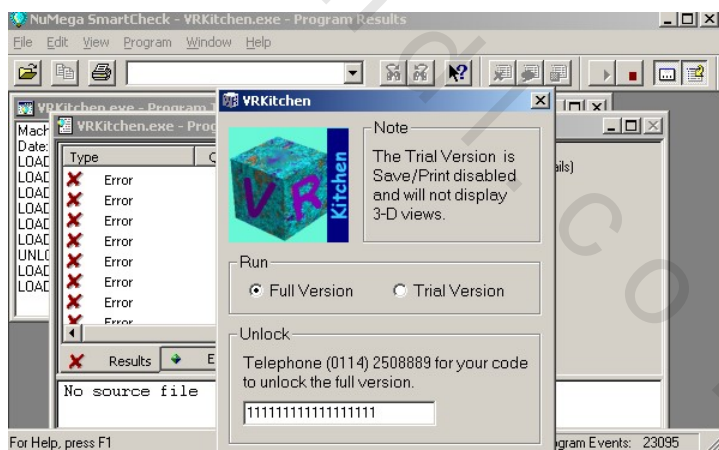
و بالطبع بعد الضغط على مفتاح ok سيظهر رسالة خطأ



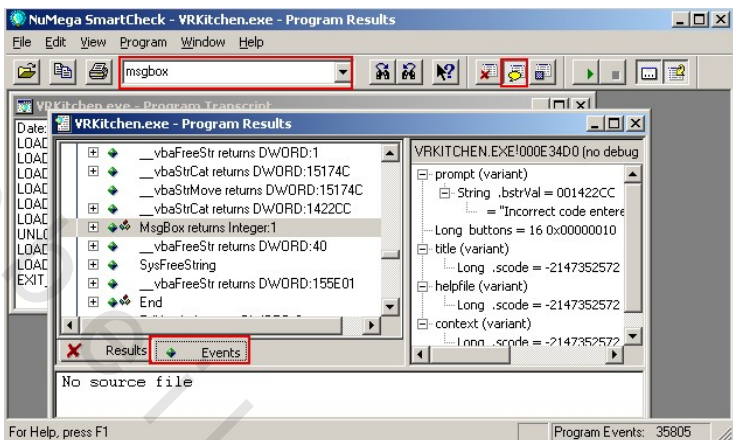
و ما نريده هنا هو معرفة نص رسالة الخطأ قم الآن بحفظ رسالة الخطأ وأغلق البرنامج وافتح برنامج smartcheck وقم بتحميل البرنامج بداخله .



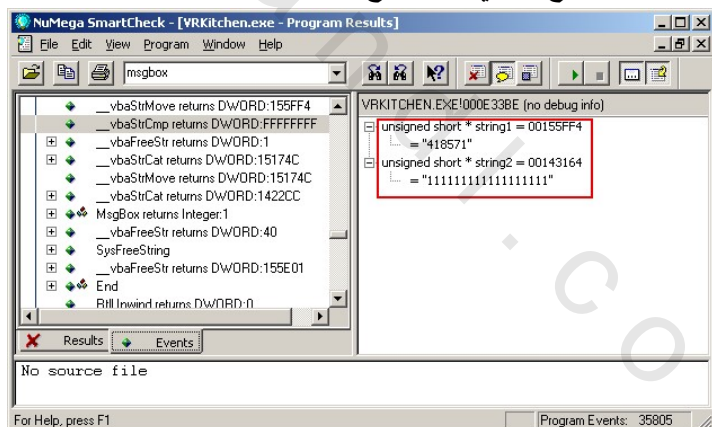
قم الآن بالضغط على المفتاح F5 حتى يتم تشغيل البرنامج وإذا ظهر رسائل قم باختيار suppress ثم قم مره أخرى بكتابة أي سيريال



و الآن تأكد من اختيار صفحة events ثم اختار الأمر View | Show all events ثم اكتب كلمة البحث msgbox وهي المستخدمة في فيجوال بيزيك لإظهار الرسائل.



والآن اصعد لأعلى حتى دالة vbastrcmp وإذا خمنت فهي لمقارنة السريال الذي قمنا بإدخاله مع السريال الصحيح



وكما يتضح من الشكل أن السريال الحقيقي هو 418571

برنامج 3.1 ACDSee:

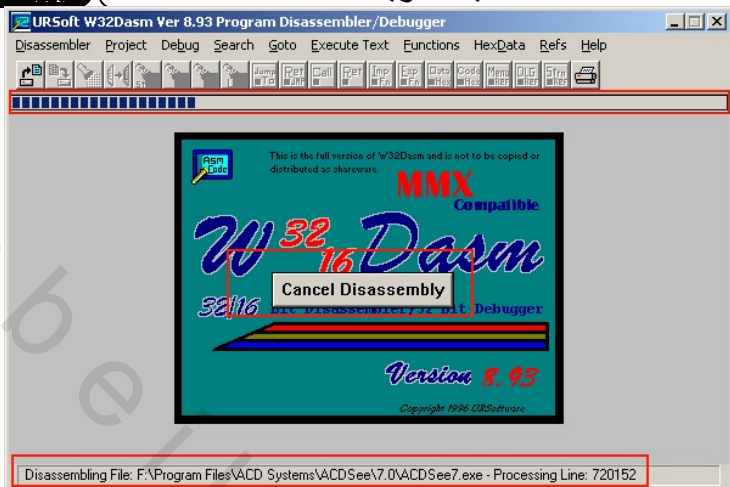
الحماية: الاتصال بالانترنت وفحص معلومات عن جهازك وعدد أيام استخدامك للبرنامج

الأدوات: W32Dasm و HIEW600

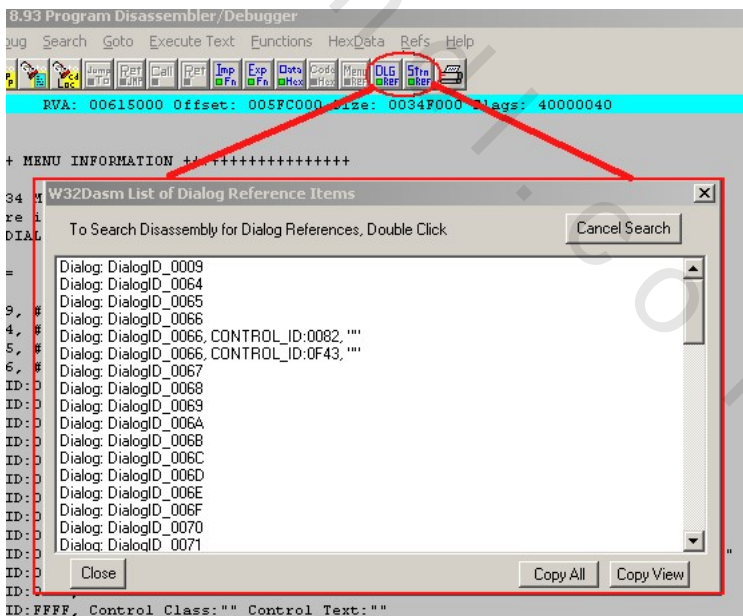
الطريقة: تأكد من عدم اتصالك مع الانترنت وقم بتشغيل البرنامج وسيحاول البرنامج عند التشغيل الاتصال ولكن رسالة خطأ سوف تظهر "error connecting to server" كما يوجد بالبرنامج فحص لعدد أيام التشغيل فإذا تعدى التشغيل 30 يوما يتم إيقاف البرنامج ، قم بفتح برنامج w32dasm واضغط على مفتاح تحميل الملف التنفيذي كما بالشكل:



قم الآن باختيار الملف التنفيذي للبرنامج وهو `acdsee.exe` وسيقوم البرنامج بفك الملف إلى لغة الاسمبلى ومن المزايا الرائعة لبرنامج `w32dasm` هو انه يحتوى على النصوص الخاصة بكل الرسائل التي تظهر عند تشغيل ويمكننا البحث عن الرسالة الخاصة بنا والوصول إلى المكان المسئول عن نداء هذه الرسالة (هذه الطريق لا تناسب جميع البرامج) سيقوم البرنامج بتحميل الملف التنفيذي انتظر حتى تنتهي العملية.



قم بالضغط على مفتاح إظهار الرسائل كما يلي:



"you have %d sponsored days remaining" قم الآن بالبحث عن الرسالة التي تقول
 "you have %d sponsored days remaining" قم الآن بالضغط المزدوج على هذه الرسالة ثم قم بإغلاق دIALOG
 الرسائل وسنجد أن هذه الرسالة تشير إلى السطر التالي:

InTouchClient.?SponsoredMode_GetDaysLeft@InTouchClient@@@QAEHXZ, Ord:0058h

:0043D7F3 CALL DWORD PTR [004D8258]

:0043D7F9 MOV ECX, D, [004F4324]

:0043D7FF PUSH EAX

* Possible Reference to String Resource ID=00263: "You have %d sponsored days remaining before you must connect"

لاحظ هنا أن Intouch هو النظام الخاص ببرنامج acdsee للاتصال بالانترنت وهو يقوم بالنداء على ملف المكتبة intouchclient.dll إذا هذا الملف هو الذي يجب فكه

قم الآن بإعادة الخطوات السابق مع الملف intouchclient.dll وبعد بفتح "acd intouch server could not be contacted" ستجد الرسالة الهامة

:10007A48 MOV ESI, ECX

:10007A4A JE 10007A77 <-- أمر القفز الذي يجب تغييره

:10007A4C MOV EAX, DWORD PTR [ESI+08]

:10007A4F PUSH 1

:10007A51 PUSH EAX

* Reference To: USER32.KillTimer, Ord:0199h

:10007A52 CALL DWORD PTR [100202C8]

:10007A58 MOV ECX, DWORD PTR [ESI+08]

:10007A5B MOV EDX, DWORD PTR [10026E58]

:10007A61 PUSH 30

* Possible Reference to String Resource ID=00196: "ACD

In Touch Service" <-- هل هذا السطر ينكرك بشئ

:10007A63 PUSH C4

* Possible Reference to String Resource ID=00596: "The ACD InTouch server could not be contacted.

:10007A68 PUSH 254

:10007A6D PUSH ECX

:10007A6E PUSH EDX

:10007A6F CALL 1000A680

:10007A74 ADD ESP, 14

* Referenced by a (U)nconditional or (C)onditional Jump at

Address: ← هذا السطر هام

:10007A77 MOV EAX, DWORD PTR [ESI+08]

* Possible Reference to Dialog: DialogID_0414

:10007A7A PUSH 414

:10007A7F PUSH 0

:10007A81 PUSH 472

:10007A86 PUSH EAX

كما لاحظت فإن أمر القفز هنا يجب تغييره في العنوان 10007A77 سنقوم

الآن بأخذ هذا العنوان وبالبحت عنه ببرنامج HIEW600 كما بالشكل:

```

K:\CLASS1~1\SoftSet\Utility\CRACKS\CTools\HIEW600\HIEW.EXE
-----
DEXEM.EXE  R  00000000  a16  11408  Hiew 6.00  <c>SEN
00000000: 4D      dec     bp
00000001: 58      pop     dx
00000002: 90      nop
00000003: 0017    add     [bx],dl
00000005: 0001    add     [bx][di],al
00000007: 0008    add     [bx][si],cl
00000009: 009302FF  add     [bp][di][0FF02],dl
0000000D: FFC5    inc     si
0000000F: 0200    add     al,[bx][si]
00000011: 0200    add     al,[bx][si]
00000013: 0000    add     [bx][si],al
00000015: 029F0254  add     bl,[bp][05402]
00000019: 0000    add     [bx][si],al
0000001B: 0032    add     [bp][si],dh
0000001D: 11504B  adc     [bx][si][0004B],dx
00000020: 4C      dec     sp
00000021: 49      dec     cx
00000022: 54      push    sp
00000023: 45      inc     bp
00000024: 20436F  and     [bp][di][0006F],al
00000027: 7072    jo      0000002B
00000029: 2E2031  and     cs:[bx][di],dh
0000002C: 3939    cmp     [bx][di],di
1 Info 2 Edit 3 Edit 4 Goto 5 Refer 6 Search 7 Header 8 Files 9 Unit
  
```

قم بالضغط على F2 ثم اكتب الأمر JMP بدلا من JE قم بحفظ الملف ثم قم بتشغيله مره أخرى نلاحظ انه يقوم بإظهار رسالة قبل أن يحاول الاتصال بالانترنت سنرجع الآن إلى w32dasm ونراجع الكود السابق بدقه أكثر

:10007A77 MOV EAX, DWORD PTR [ESI+08]

* Possible Reference to Dialog: DialogID_0414 <-- هذا الجزء

مستوّل عن إظهار رسالة

:10007A7A PUSH 414

:10007A7F PUSH 0

:10007A81 PUSH 472

:10007A86 PUSH EAX

Name: DialogID_0414, # of Controls=008,

Caption:"InTouch", ClassName:""

قم بفتح ديالوج الرسائل وانظر للنصوص التالية

001 - ControlID:FFFF, Control Class:"STATIC" Control Text:""

002 - ControlID:0449, Control Class:"STATIC" Control Text:"Internet Connection Settings"

003 - ControlID:043E, Control Class:"BUTTON" Control Text:"Access the Internet via a &dial-up connection"

004 - ControlID:043F, Control Class:"BUTTON" Control Text:"Access the Internet via a &LAN"

005 - ControlID:0444, Control Class:"BUTTON" Control Text:"LAN &settings..."

هذه هي النافذة قبل محاوله الاتصال بالسيرفر معنى هذا أن الكود السابق لها هو الذي يقوم بإظهار الرسالة راجع الآن ديالوج الرسائل ثانية:

Name: DialogID_0423, # of Controls=007, Caption:"Setup", ClassName:""

001 - ControlID:0449, Control Class:"STATIC" Control Text:"Sponsored Mode Activated"

002 - ControlID:FFFF, Control Class:"STATIC" Control Text:""

003 - ControlID:FFFF, Control Class:"STATIC" Control Text:"You are now ready to use this software in Sponsored Use Mode."

004 - ControlID:03F2, Control Class:"STATIC" Control Text:"You sponsorship must be renewed in %d days."

005 - ControlID:FFFF, Control Class:"STATIC" Control Text:"Press <Finish> to complete this wizard and start using the software."

006 - ControlID:FFFF, Control Class:"STATIC" Control Text:""

007 - ControlID:03F0, Control Class:"STATIC" Control Text:""

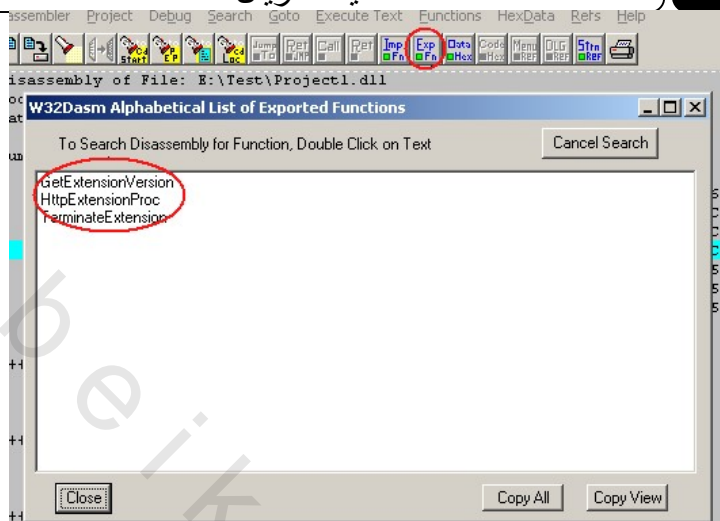
إذا كنت لم تخمن سنقوم الآن بتعديل الكود بدلا من الاتصال بالانترنت سيقوم بتحميل رسالة تظهر نجاح عملية الترخيص وهو رقم 423 والآن لاحظ الكود التالي:

* Possible Reference to Dialog: DialogID_0414 <-- رساله.

:10007A7A PUSH 414 <-- فحص الانترنت

سنقوم الآن بتغيير السطر PUSH 414 لكي يقوم بالذهاب إلى رسالة الترخيص فتصبح PUSH 423

والآن بالنسبة لفحص عدد الأيام قم بفتح دIALOG الملفات المصدرة كما بالشكل:



و لاحظ وجود الدوال التالية:

Addr:10006410 Ord: 89 (0059h) Name:

?SponsoredMode_GetDaysLeft@InTouchClient@@QAEHXZ

Addr:10006450 Ord: 90 (005Ah) Name:

?SponsoredMode_ShowExpiredBox@InTouchClient@@QAEHPAUHWND_@@@Z

Addr:10006430 Ord: 91 (005Bh) Name:

?SponsoredMode_ShowWarningBox@InTouchClient@@QAEHPAUHWND_@@@Z

لاحظ هنا أن الحماية يمكن إيجاده بالبحث عن النص getdaysleft وعند

البحث نجد السطور التالية:

Exported fn():

?SponsoredMode_GetDaysLeft@InTouchClient@@QAEHXZ - Ord:0059h

:10006410 PUSH ESI

:10006411 MOV ESI, ECX

* Reference To:

InTouchClient.?DaysSinceLastConnection@InTouchClient
@@@QAEHXZ

:10006413 CALL 10003860 <-- call get DaysSince function
(returns days since in EAX)

:10006418 MOV ESI, DWORD PTR [ESI+00000414]

:1000641E MOV ECX, EAX <-- المسجل ECX يحتوى عدد الايام

:10006420 MOV EAX, ESI

:10006422 SUB EAX, ECX <-- EAX=30-days

:10006424 JS 1000642A <-- if EAX<=0 then goto make
EAX zero

:10006426 CMP EAX, ESI <-- if EAX is between 1 and 30
continue

:10006428 JLE 1000642C

* Referenced by a (U)nconditional or (C)onditional Jump at
Address:

|:10006424(C)

:1000642A XOR EAX, EAX

من هذه السطور يتضح أنها تقوم بالحصول على آخر يوم تم فيه الاتصال
بالسيرفر وأن المسجل EAX يتم اختبار قيمته فإذا كانت به القيمة محصورة
بين 1 و 30 فيتم استكمال عمل البرنامج والحل الآن أن نقوم بجعل قيمة EAX
تساوى 30 دائماً كما يلي:

من:

CALL 10003860

MOV ESI, DWORD PTR [ESI+00000414]

MOV ECX, EAX

MOV EAX, ESI

SUB EAX, ECX

JS 1000642A

```
CMP EAX, ESI
JLE 1000642C
XOR EAX, EAX
POP ESI
RET
```

إلى:

```
CALL 10003860
MOV ESI, DWORD PTR [ESI+00000414]
MOV ECX, EAX
MOV EAX, ESI
SUB EAX, ECX
MOV EAX, 1E
NOP
NOP
NOP
POP ESI
RET
```

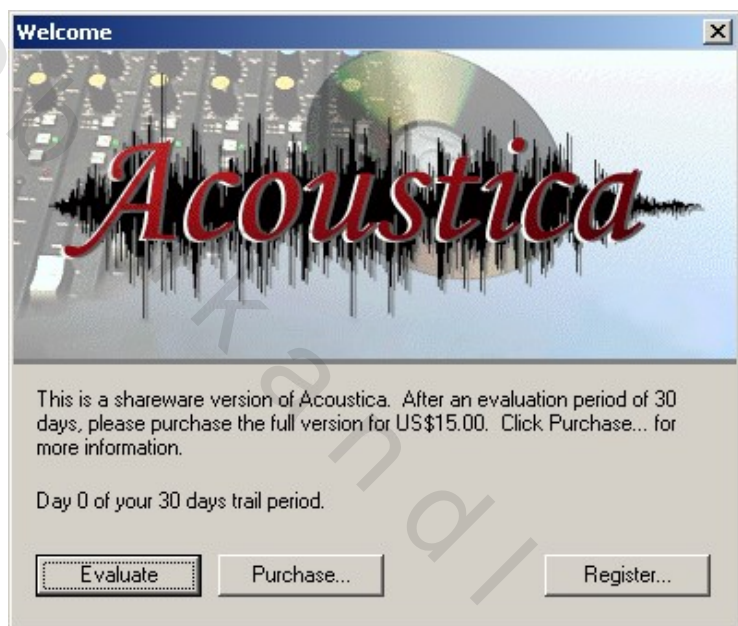
والآن حاول أن تقوم بترخيص البرنامج ستجد أن البرنامج يظهر رسالة نجاح ويعمل بصورة دائمة

برنامج Acoustica:

الأدوات: ollydbg, w32dasm

نوع الحماية: هذا البرنامج يقوم بالعد لمدة 30 يوم ويقل البرنامج بعد ذلك إذا لم

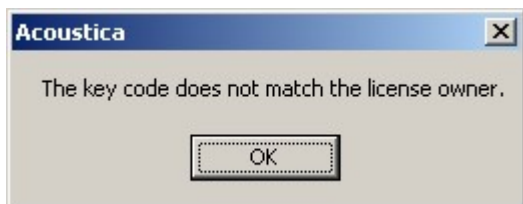
تدخل بيانات تسجيل صحيحة



قم الآن بالضغط على المفتاح register وقم بكتابة أي بيانات



اضغط register حتى يمكننا معرفة رسالة الخطأ

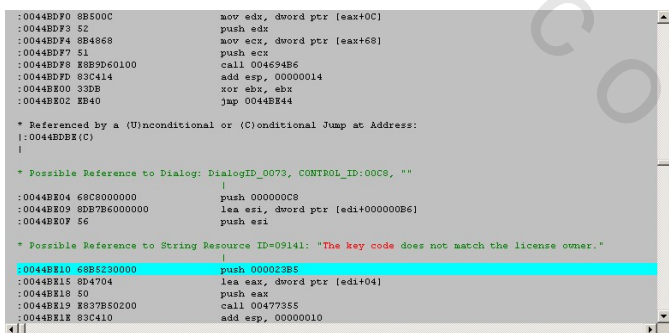


والآن قم بفتح البرنامج ببرنامج w32dasm وابحث عن الرسالة السابقة وهذه الرسالة موجودة مرتين لذلك سنأخذ الأخيرة



و بعد ذلك ستجد أن المكان الذي تقف به يوجد به الأمر

PUSH 000023B5



قم بتسجيل العنوان لهذا السطر في أي ورقه ثم اقل البرنامج وحمل الملف ببرنامج ollydbg وابحث عن العنوان 44BE10 بالضغط على المفاتيح

Ctrl+G

CPU - main thread, module acoustic			
Address	Hex dump	Disassembly	Comment
0044BDF4	. 8B48 68	MOV ECX, DWORD PTR DS:[EAX+68]	
0044BDF7	. 51	PUSH ECX	Arg1
0044BDF8	E9 B9D60100	CALL acoustic.004694B6	acoustic.004694B6
0044BDFD	83C4 14	ADD ESP, 14	
0044BE00	. 330B	XOR EBX, EBX	
0044BE02	. EB 40	JMP SHORT acoustic.0044BE44	
0044BE04	> 68 C8000000	PUSH 0C8	Arg4 = 000000C8
0044BE09	> 80B7 B6000000	LEA ESI, DWORD PTR DS:[EDI+B6]	
0044BE0F	. 56	PUSH ESI	Arg3
0044BE10	. 68 B5230000	PUSH 23B5	Arg2 = 000023B5
0044BE15	. 8D47 04	LEA EAX, DWORD PTR DS:[EDI+4]	
0044BE18	. 50	PUSH EAX	Arg1
0044BE19	E8 37B50200	CALL acoustic.00477355	acoustic.00477355
0044BE1E	83C4 10	ADD ESP, 10	
0044BE21	. 8B06	MOV EDI, ESI	
0044BE23	8B4F 66	MOV ECX, DWORD PTR DS:[EDI+66]	
0044BE26	. 8B01	MOV EAX, DWORD PTR DS:[ECX]	
0044BE28	. 6A 00	PUSH 0	Arg5 = 00000000
0044BE2A	. 68 42FE4900	PUSH acoustic.0049FE42	Arg4 = 0049FE42
0044BE2F	. 52	PUSH EDI	Arg3

والآن قم بوضع نقطة إيقاف على السطر الحالي F2 ثم قم بتشغيل البرنامج بالضغط على المفتاح F9 وعند كتابة بيانات خطأ مره أخرى ستجد أن برنامج ollydbg توقف عند هذا السطر ويظهر السريال الحقيقي في نافذة stack (قم بالتمرير لأعلى قليلا حتى ترى السريال الحقيقي)

0012F524	000000C01	
0012F528	000000003	
0012F52C	00000001A	
0012F530	0012F314	UNICODE "11111111111111111111111111111111"
0012F534	0012F5B0	ASCII "11111111111111111111111111111111"
0012F538	FFFFFFFF	
0012F53C	0012F607	
0012F540	00011224	
0012F544	00011224	
0012F548	0044B292	acoustic.0044B292
0012F54C	0012F560	ASCII "AC220-70180"
0012F550	0012F5B0	ASCII "11111111111111111111111111111111"
0012F554	0012FD98	
0012F558	0012F7EC	
0012F55C	00A52101	
0012F560	32324341	
0012F564	30372D30	
0012F568	00303831	
0012F56C	77D41940	USER32.77D41940

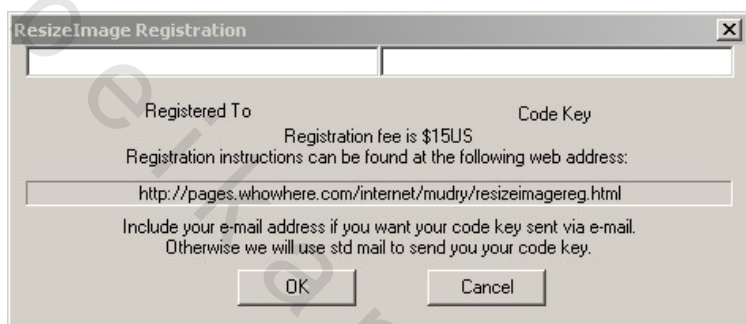
والآن قم بتجربة السريال AC220-70180 ولاحظ أن السريال يتغير بتغير الاسم المستخدم للتريخيص.

كيفية صنع برنامج مولد للسريال KeyGen:

البرنامج: resizeimage

الأدوات: softice, programming language

قم أولاً بتشغيل برنامج softice ثم قم بتشغيل البرنامج resizeimage فتظهر نافذة بها مكان لإدخال الاسم والسريال .



حاول الآن إدخال أي اسم وأي رقم يمكن تذكرهم بسهولة مثلاً

Cyber Man

111111111111111111

ارجع الآن إلى برنامج softice عن طريق المفاتيح Ctrl+D ثم قم بوضع نقطة الإيقاف المعروفة.

BPX GetWindowTextA

ثم اضغط Ctrl+D مرة أخرى للعودة للبرنامج قم الآن بالضغط على المفتاح Ok فتظهر شاشة softice مع الرسالة التالية:

Break due to BPX USER32!GetWindowTextA

USER32!GetWindowTextA

و الآن قم بالضغط على المفاتيح Ctrl+D حتى ترجع برنامج softice مرة أخرى وسنقوم الآن بتتبع الكود وستجد وجود نداء آخر للدالة

GetWindowTextA وحتى لا تجد نفسك خارج مكان الكود قم بالضغط مره أخرى على المفاتيح Ctrl+D

اضغط الآن على المفتاح F11 حتى ترجع للكود الذي قام بالنداء على الدالة GetWindowTextA

و سنرى الآن لاحظ أننا وضعنا علامات xxxx بديل عن عنوان الذاكرة بالنظام السداسي لأنها تختف من كمبيوتر لآخر :

```
xxxx:00436280 CALL [USER32!GetWindowTextA] <----
our breakpoint (3rd)
```

```
xxxx:00436286 push FF
xxxx:00436288 mov ecx, [ebp+10]
xxxx:0043628B call 00433DA5
xxxx:00436290 jmp 0043629D
```

و يمكننا الآن أن نستمر في تتبع البرنامج بالمفتاح F10 حتى أمر النداء call في العنوان 0043628B وبعد ذلك يتم القفز إلى العنوان 0043629D

```
xxxx:0043629D pop edi <----- we jump here
xxxx:0043629E pop esi
xxxx:0043629F pop ebp
xxxx:004362A0 ret 000C
```

استمر باستخدام المفتاح F10 حتى تجد نفسك في المكان التالي

```
xxxx:00406550 call 00436251
xxxx:00406555 pop edi <----- we are back here, from
the CALL above
xxxx:00406556 pop esi
xxxx:00406557 ret 0004
```

و بالاستمرار يتم القفز إلى الكود التالي بسبب النداء السابق

```
xxxx:00432974 call [eax+0000008C]
xxxx:0043297A mov [ebp-18], 00000001 <---- we arrive
here, back from the CALL
```

...

... عدة سطور غير هامه هنا ...

...
 xxxx:004329A3 ret 0004
 استمر في تتبع الأمر call فعادة يأخذ التتبع ببرنامج softice بعض الوقت
 xxxx:00406580 call 0043292C
 xxxx:00406585 mov edi, [ebx] <----- we are here this
 time...
 اضغط الآن المفتاح F10 حتى ننفذ السطر 406585 ثم قم بتفريغ محتويات
 المسجل EDI بالأمر التالي

D EDI

و يجب أن تشاهد الاسم Cyber Man في نافذة Data مما يدل على اننا في
 المكان المناسب

```
xxxx:00406587 mov ecx, FFFFFFFF
xxxx:0040658C sub eax, eax
xxxx:0040658E repnz
xxxx:0040658F scasb
xxxx:00406590 not ecx
xxxx:00406592 dec ecx
```

الكود السابق يقوم بعدد الحروف الخاصة بالاسم والذي تم تسجيله بالمسجل
 ECX

```
xxxx:00406593 je 004066A4
```

و يمكننا أن نعبر عن فحص عدد الحروف بالأمر التالي

If ecx=0 (لا يوجد اسم تم إدخاله) then jump to the line 4066A4, if
 not, continue:

```
xxxx:00406599 mov edi, [esi+5C]
```

و الآن قم بالضغط على المفتاح F10 واكتب الأمر

D EDI

ستجد الآن السيريال الخطأ الذي قمنا بإدخاله مما يدل على أن هذا الجزء
 يحتوي على المعادلات الخاصة بحساب السيريال الصحيح

```
xxxx:0040659C mov ecx, FFFFFFFF
xxxx:004065A1 sub eax, eax
```

```
xxxx:004065A3 repnz
xxxx:004065A4 scasb
xxxx:004065A5 not ecx
xxxx:004065A7 dec ecx
xxxx:004065A8 je 004066A4
```

مره أخرى يتم التأكد أن حقل السيريال غير خالي

```
xxxx:004065AE push 00000000
xxxx:004065B0 mov ecx, ebx
xxxx:004065B2 call 00433D50
```

النداء الأخير غير هام فقم بالضغط على المفتاح F10

```
xxxx:004065B7 push eax
xxxx:004065B8 mov ecx, esi
xxxx:004065BA call 004066E0 <----- interesting :>
```

سنقوم الآن بالدخول إلى هذا الكود قم بالضغط على المفتاح F8 حتى ترى الكود التالي

```
xxxx:004066E0 sub esp, 0000002C
xxxx:004066E3 mov [esp], 00000005 *
xxxx:004066EB mov [esp+04], 00000003 *
xxxx:004066F3 mov [esp+08], 00000007 *
xxxx:004066FB mov [esp+0C], 00000001 *
```

لاحظ الآن من السطور السابقة أن البرنامج يقوم بتسجيل بعض القيم الثابتة وهي 5,3,7,1

```
xxxx:00406703 push ebx
xxxx:00406704 push esi
xxxx:00406705 push edi
```

سطر آخر هام

```
xxxx:00406706 mov ebx, 12345678 *
```

هذا السطر يقوم بتخزين الرقم السداسي 12345678 في المسجل EBX

```
xxxx:0040670B mov [esp+20], 00000002 *
```

حماية السريال

```
xxxx:00406713  mov [esp+24], 00000006  *
xxxx:0040671B  push ebp
xxxx:0040671C  push 00000009
xxxx:0040671E  mov [esp+30], 00000004  *
```

لاحظ مره أخرى تخزين أرقام ثابتة هي 2,6,4 غالبا سنقابل هذه الأرقام مره

أخرى لذلك تذكرهم جيدا

```
xxxx:00406726  xor ebp, ebp  *
xxxx:00406728  mov [esp+24], ebp  *
```

الآن يتم تصفير المسجل ebp وتخزينه والأرقام 5,3,7,1,0,2,6,4 قد يتم استخدامهم لاحقا ، ويمكننا أن نرى الترتيب الخاص بهذه الأرقام إذا كتبنا الأمر ED ESP+24 والضغط على مفتاح السهم لأعلى فيمكنك أن ترى في نافذة

data المعلومات التالية:

```
xxxx:0068E874 00000005 00000003 00000007 00000001
xxxx:0068E884 00000000 00000002 00000006 00000004
```

و الآن لنرى السطر التالي والذي سنجده في هذه الحالة الأمر CALL

```
xxxx:0040672C  call 0041DFB0
```

و لاتهمنا هذه النداءات الآن لذلك اضغط المفتاح F10 لتقادى هذه لسطور حتى تصل إلى السطور التالية:

```
xxxx:00406731  mov edx, dword ptr [esp+44]
```

و الآن لنرى محتويات المسجل EDX بكتابة الأمر D EDX ستري الآن الاسم الذي قمت بإدخاله مثلا "Cyber Man" فإذا افترضنا أن السريال يتم بناؤه من الاسم الذي قمنا بإدخاله فإن هذه السطور هامه جدا

```
xxxx:00406735  add esp, 00000004
xxxx:00406738  mov esi, eax
xxxx:0040673A  mov edi, edx
```

```
xxxx:0040673C  mov ecx, FFFFFFFF
xxxx:00406741  sub eax, eax
xxxx:00406743  repnz
xxxx:00406744  scasd
xxxx:00406745  not ecx
xxxx:00406747  dec ecx
xxxx:00406748  je 00406764
```

من السطور السابقة نجد انه يتم اختبار طول الاسم المدخل فإذا لم يكن يساوى صفرا يتم استكمال تنفيذ الإجراء

```
xxxx:0040674A  inc ebp
```

الآن المسجل EBP يحتوى القيمة 1 غالبا لاستخدامه كعداد وسنتأكد من ذلك فيما بعد

```
xxxx:0040674B  mov edi, edx
xxxx:0040674D  movsx eax, byte ptr [edx+ebp-01] *
```

السطر الأخير هام جدا:

كما رأينا من قبل أن المسجل EBP يساوى 1 والآن يتم تحميل البايت الخاص بـ EAX في EDX+EBP-01 بسبب التعليمه MOVSX والتي ستساوى الآن المعادلة التالية:

$$EAX = EDX + 1 - 01 = EDX$$

و حيث أن الاسم الذي قمنا بإدخاله تم تخزينه في المسجل EDX فيبدو أن روتين حساب السريال يبدأ من هنا وسيحمل المسجل EAX القيمة السداسية المكافئة لأول حرف من الاسم "C" وهي 43

```
xxxx:00406752  add ebx, eax
```

و الآن إذا تذكرت السطر الذي يوجد به الأمر التالي:

$$EBX = 12345678$$

فيكون لدينا المعادلة التالية:

$$EBX = EBX + EAX = 12345678 + 43$$

و الآن المسجل EBX يخزن ناتج المعادلة السابقة وسنرى فيما بعد كيف يتم استخدام هذه القيمة.

```
xxxx:00406754  mov ecx, FFFFFFFF
xxxx:00406759  sub eax, eax
xxxx:0040675B  repnz
xxxx:0040675C  scasb
xxxx:0040675D  not ecx
xxxx:0040675F  dec ecx
xxxx:00406760  cmp ecx, ebp
xxxx:00406762  ja 0040674A
```

يقوم البرنامج مره أخرى بحساب طول النص ويقارنه بقيمة المسجل EBP والتي تساوى 1 وسيتم الدخول في حلقه طالما $EBP < ECX$ حتى العنوان 0040674A و لنرى الآن كيفية تنفيذ الحلقة الثانية من التكرار

```
ebp=2 (INC EBP)
eax=79 (hex value for y)
ebx=12345678+48+79
```

الحلقة الثالثة

```
ebp=3 (INC EBP)
eax=62 (hex value for b)
ebx=12345678+48+61+62
```

...

وهكذا حتى الحلقة الأخيرة ولاحظ أن يتم وضع أساس يساوى طور الاسم المدخل ثم الأرقام المكافئة لحروف الاسم

basis + C y b e r M a n

والآن بإضافة الحروف السابقة يصبح عندنا المعادلة التالية

$$\begin{aligned} \text{EBX} &= 12345678+43+79+62+65+72+4\text{D}+61+6\text{E} \\ &= 12345989 \end{aligned}$$

لا تتسنى إجراء العملية بالنظام السداسي وليس العشري والآن لنرى ما يحدث بعد ذلك

xxxx:00406762 ja 0040674A

نتيجة للسطر السابق عندما $\text{ECX} = \text{EBP}$ لا يتم التكرار ويتم استكمال تنفيذ باقي البرنامج

xxxx:00406764 lea eax, dword ptr [esp+30]

xxxx:00406768 push ebx

xxxx:00406769 push 00442854

xxxx:0040676E push eax

xxxx:0040676F call 0041D2F0

لاحظ في السطر 406768 يتم تخزين قيمة EBX حتى يمكن استخدامها مرة أخرى ويمكنك تمرير النداء في السطر 40676F باستخدام المفتاح F10 حتى نصل للسطور التالية:

xxxx:00406774 add esp, 0000000C

xxxx:00406777 xor eax, eax

السطور التالية تحوى حلقة تكراريه أخرى من العنوان 406779 إلى العنوان 406789 وهذا الجزء هام جدا لأنه يقوم بإنشاء المفتاح الذي نريده لتسجيل البرنامج وفيما نستعرض هذه السطور:

xxxx:00406779 mov ecx, dword ptr [esp+4*eax+10]

هذا السطر يقوم بتخزين القيمة في العنوان $\text{ESP}+10$ في المسجل ECX (لان EAX تساوى صفر) وبكتابة امر التفرغ ECX D سيظهر لك القيمة 5 (هل تذكر نموذج الأرقام التي قابلناها سابقا 5,3,7,1,0,4,6,2)

xxxx:0040677D inc eax

الآن المسجل EAX يحتوى القيمة 1

```
xxxx:0040677E cmp eax, 00000008
```

إذا كانت EAX تساوى 8 فإننا في آخر حلقة

```
xxxx:00406781 mov cl, byte ptr [esp+ecx+30]
```

```
xxxx:00406785 mov byte ptr [eax+esi-01], cl
```

```
xxxx:00406789 jb 00406779 <----- Loop while
```

```
EAX < 8
```

السطرين السابقين هما من يقومان بتوليد السريال

أول حلقة:

ECX=5

EAX=1

CL= 9 (the number that is at [ESP+30] + 5)

store CL in [EAX+ESI-1] (=ESI)

الحلقة الثانية:

ECX=3

EAX=2

CL= 4 (the number that is at [ESP+30] + 3)

store CL in [EAX+ESI-1] (=ESI+1)

الحلقة الثالثة:

ECX=7

EAX=3

CL= 7 (the number that is at [ESP+30] + 7)

store CL in [EAX+ESI-1] (=ESI+2)

...

آخر حلقة:

ECX=2

EAX=8

CL= 3 (the number that is at [ESP+30] + 2)

store CL in [EAX+ESI-1] (=ESI+7)

من الرقم السابق 12345989 الذي قمنا باستخراجه من الاسم ومع نموذج الأرقام 5,3,7,1,0,4,6,2 نستخلص أن البرنامج يقوم بتحريك كل رقم إلى آخر مكان عن طريق الحلقة التكرارية ويتم ذلك كما يلي:

- get the number at position ESP+30+5, store it at ESI
(=9)
- get the number at position ESP+30+3, store it at ESI+1
(=4)
- get the number at position ESP+30+7, store it at ESI+2
(=7)
- get the number at position ESP+30+1, store it at ESI+3
(=2)
- get the number at position ESP+30+0, store it at ESI+4
(=1)
- get the number at position ESP+30+2, store it at ESI+5
(=3)
- get the number at position ESP+30+6, store it at ESI+6
(=A)
- get the number at position ESP+30+4, store it at ESI+7
(=5)

و هكذا يكون السيريال الخاص بالاسم هو 947213A5 وبهذه الطريقة نكون عرفنا كيفية حساب هذا السيريال

و هذا ينطبق على أي اسم نقوم بإدخاله وسنقوم الآن بكتابة البرنامج اللازم لتوليد أي سيريال صحيح بناء على أي اسم وسوف نقوم باستخدام لغة QBasic لأنها أسهل في الفهم و التحويل لأي لغة

---[بداية البرنامج]-----

```

1) | CLS
2) | LOCATE 2, 1
3) | COLOR 3, 0: PRINT TAB(24); "Resize Image v3.3
«Serial Maker»"
4) | COLOR 7, 0: PRINT TAB(30); "by "; : COLOR 15, 0:
PRINT "HarvestR CIA '98"; :
5) | LOCATE 5, 15: PRINT "Enter you Name : "; : COLOR
15, 0: LINE INPUT ; name$
6) | TmpVal# = 305419896
7) | FOR i = 1 TO LEN(name$)
8) | TmpVal# = TmpVal# + ASC(MID$(name$, i, 1))
9) | NEXT i
10) | TmpVal$ = HEX$(TmpVal#)
11) | FOR i = 0 TO 7
12) | code$(i) = MID$(TmpVal$, i, 1)
13) | NEXT i
14) | serial$ = code$(5) + code$(3) + code$(7) + code$(1) +
code$(0) + code$(2) + code$(6) + code$(4)
15) | LOCATE 6, 15: COLOR 7, 0: PRINT "Your serial is :
"; : COLOR 7, 4: PRINT " ";

```

---[شرح الكود]-----

--

من السطر 1 إلى 5 : تظهر رسالة تطلب من المستخدم إدخال اسمه
السطر 6: يتم الإعلان عن المتغير tmpval# ووضع القيمة 305419896 به
وهي القيمة المكافئة للرقم السداسي 12345678
السطر من 7 إلى 9 : يتم إضافة كود ASCII لكل حرف إلى المتغير
tmpval#

السطر 10: تحويل القيمة العشرية للمتغير tmpval# إلى سداسيه ووضعها في نص

السطر من 11 إلى 13: يتم إنشاء متغير نص code\$(i) لكل رقم من نموذج الرقم السابق

السطر 14: يتم توليد السيريال النهائي عن طريق تحريك كل رقم للمكان المناسب طبقا للنموذج الموجود للأرقام (5,3,7,1,0,2,6,4)

السطر 15: عرض السيريال الصحيح

و الآن يمكننا تسجيل البرنامج 3.3 ResizeImage باى اسم تريده

لاحظ أن بيانات التسجيل يتم تخزينها في الملف resizeimage.ini في الدليل الخاص بالويندوز

-----[تمرين]-----
--

على سبيل التمرين حاول الآن صنع برنامج توليد للسيريال للبرنامج الثاني WallPaper وهو من صنع نفس الشركة ويستخدم البرنامج نفس الطريقة لتوليد السيريال مع اختلافات بسيطة.

حماية الأجهزة المادية:

تختلف طرق حماية البرامج باستخدام أجهزة مادية ولكن الفكرة العامة واحدة فلا يعمل البرنامج إذا لم يجد الجهاز المادي HardWare متصلا بالكمبيوتر وأشهر طرق الحماية هي باستخدام الاسطوانات المدمجة وقوابس الدونجل وستعرض لكل نوع منهم.

حماية Dongle:

أصبح هذا النوع من الحماية من أكثر طرق الحماية استخداما على مستوى العالم وأصبحت البرامج التي تستخدم هذا النوع من الحماية كالأسطورة بحيث لا يمكن التوصل أبدا إلى طريقة لفكها وسنقوم الآن بإثبات الفشل الذريع لهذا النوع من الحماية كما أثبتته الكثير من الكراكرز على مستوى العالم وليس فقط هذا النوع من الحماية يمكن كسره بسهولة لكنه أيضا مكلف بالنسبة للبرمجيات الصغيرة ولا يستحق صرف قرش واحد عليه وسنعرف فيما يلي لماذا (لقد استخدمت يوما دونجل لحماية برامج شركة ما) لن نقوم هنا بتحديد برنامج نستهدفه للكسر لكن سنتعلم الطريقة العامة لكسر هذا النوع من الحماية وستجد الأمر ممتع حتى أكثر من استخدام برنامج طالما أردت الحصول عليه.

وأجهزة الدونجل إذا لم تكن تعرف هي جهاز بلاستيك صغير يتم عادة توصيله الميناء LPT أو COM الخاص بالكمبيوتر ولا يتعارضوا مع أي طابعه متصلة بنفس الميناء .

ويوجد بداخل الدونجل ذاكرة خاصة تسمى EEPROM أو ASIC والتي يمكننا أن نعتبرها مثل الصندوق الأسود ومنها يتم استقبال وإرسال البيانات من البرنامج إلى الدونجل عن طريق برنامج الدونجل (عادة ملف dll) والذي هو عبارة عن أوامر API ويتم بيعه مع الدونجل .

وما يفعله المطور هو النداء على دالة API تقوم بإرجاع قيمه معينه تبين وجود الدونجل الصحيح أم لا

ويوجد أنواع كثيرة من الدونجل منها أشهرها:

✦ دونجل Sentinel والذي بنفسه له عائله خاصة به ومن صنع شركة

Rainbow Technoogies

✦ دونجل Hasp من شركة Aladin

✦ دونجل Fest

✦ دونجل Actikey

و الكثير غيرها والأدوات التي سوف نقوم باستخدامها هي:

SoftIce, W32Dasm, HIEW

والخطوة الأولى هي التعرف على نوع الدونجل الذي نتعامل معه ويتم ذلك بسهولة عن طريق فتح ملفات dll و vxd والبحث فيها عن أي نص له علاقة بالأنواع السابقة.

الخطوة الثانية هي التعرف على أي معلومات عن هذا الدونجل عن طريق الانترنت حيث يمكنك الحصول على دليل الاستخدام الخاص بهذا الدونجل ويجب عليك فهم كيفية عمل هذا الدونجل فهما كاملا .

الفكرة هنا أننا نريد كسر البرنامج وليس الدونجل نفسه فحتى طرق التشفير القوية هي عبارة عن دوال API بسيطة تقوم بقراءة سيريال وأحيانا يتم الكتابة في ذاكرة الدونجل .

الخطوة الثالثة: يمكنك أيضا الحصول على نسخه مجانية من بعض الشركات .
الخطوة الرابعة: والآن بعد عرفت بعض المعلومات عن الدونجل والبرنامج المراد كسره والآن وقت التفكير في طريقة استخدام الدونجل لحماية البرنامج وعادة يتم الأمر كما يلي:

call DONGLE

or ax, ax

jz gooduser

baduser:

وهذا بمنتهى البساطة فلا فكره إطلاقاً أصعب من هذا لاستخدام الدونجل وعادة يكون اضعف جزء في الدونجل هو برنامج الاستخدام نفسه.

Application <----> Dongle API (ملف DLL) <----> Dongle

على عكس ما سبق فإن مبرمجي شركات الدونجل يقوموا بحماية البيانات المخزنة بالدونجل جيداً عن طريق التشفير وعدم إمكانية استخدام برامج الديبجر وغيره لذلك يجب ترك هذا الجزء كما هو .

والمهمة الآن هي إيجاد مكان النداء الخاص بالدونجل واستخدام W32dasm للبحث عن دالة API والتي عادة تكون باسم يعبر عن وظيفتها مثلاً

QuickCheckIfDonglePresent()

GetDongleSerialNumber()

ReadDongleByte(addr) - or word, or dword

WriteDongleByte(addr, value)

دوال الدونجل تكون منظمه فحاول أن تجد معاملات هذه الدوال ومعرفة النتيجة التي تقوم بإرجاعها ويمكننا بعد ذلك معرفة ما هي النتيجة التي يتوقعها البرنامج عن طريق الأمر cmp ax

الخطوة الخامسة: قم بوضع التعليمة CC في المكان الذي وجدت به النداء على دالة API ثم قم بتشغيل برنامج SoftIce مع وضع نقطة التوقف كما يلي:

BPINT3

استبدل الآن INT3 بالبايت الأصلي ثم ضع نقطة توقف كما يلي:

BPX (address)

قم الآن بمتابعة الكود وقد يمكنك سريعاً التوصل إلى روتين فحص الدونجل ومعرفة اختبار وجود الدونجل وعكسه

حاول أيضا إرجاع قيمه لنداء الفحص قد تكون هذه القيمة هي اللازمة للاستمرار في تشغيل البرنامج وأحيانا أيضا يتم تخزين السريال الصحيح في الدونجل كما هو فحاول البحث عنه ببرنامج W32Dasm مثال:

```
cmp eax, some_value1
je is_model_1
cmp eax, some_value2
je is_model_2
...
```

من المقارنة السابقة قد يتم فحص أكثر من حاله للدونجل مثل تشغيل أو إغلاق خصائص معينه بالبرنامج مثال:

```
test eax, mask_1
jz lab1
call enable_option_1
lab1:
...
```

مثال:

و قد يتم تخزين السريال في المسجل ebx كما يلي:

```
mov ecx, 12
lab1:
cmp word ptr [ebx], 1
jne lab2
call enable_option(cx)
lab2:
inc ebx
loop lab1
```

و إذا كان الدونجل الأصلي متاح لك فيمكنك قراءة كل القيم التي يتم إرجاعها وإعطاء هذه القيم للدوال بالبرنامج ومعرفة أي منها هي التي يريدها البرنامج للعمل
مثال:

الدونجل من نوع Actikey ومن قراءة دليل الاستخدام عرفنا أن الملف المسئول عن دوال الدونجل هو CCNMMNT.DLL
نقوم بفتح الملف ببرنامج W32Dasm اذهب للمقطع .idata. وستجد مرجع للملف CCNMMNT.DLL به فإذهب إلى هذا العنوان وستجد المرجع كما يلي:

```
00594510 ; Imports from CCNMMNT.dll
00594510
00594510          extrn CCNMM:dword      ; DATA
XREF: j_CCNMM r
```

بعد ذلك

```
00408770 ;          Sub routine
00408770
00408770 j_CCNMM      proc near          ; CODE
XREF: key_io+6F p
00408770          jmp     ds:CCNMM      ; jump to
Dongle DLL
00408770 j_CCNMM      endp
```

وإذا تحركت لأعلى ستجد نفسك في الجزء الذي يعبر عن وجود دونجل صحيح وبعد ذلك يجب أن ترى الكود التالي:

```
0040BA10 key_io      proc near          ; CODE XREF:
main_key_check+8
```

```
0040BA10 ;
main_key_check+18C p
0040BA10 ; key_check2+45 p
```

سنجد مرجع واحد فقط لان key_check2 تم النداء عليها من

main_key_check ومازلنا داخل دالة API والآن فلنستمر لأعلى:

```
0040B840 ; Subroutine
0040B840
0040B840 main_key_check proc near ; CODE
XREF: key_fun_00+4D p
0040B840 ;
keyfn_18_sernum+68 p
0040B840 ; keyfn_02_write+5B p
0040B840 ; keyfn_03_read+56 p
0040B840 ; key_fun_1b+50 p
0040B840 ; key_fun_1c+54 p
0040B840 ; key_fun_1d+5C p
```

و يمكننا من اسم الدوال معرفة وظيفة كل منهم وتتبع كل إجراء كما يلي:

```
0040A250 keyfn_18_sernum proc near ; CODE
XREF: (several)
0040A250
0040A250 arg_0 = dword ptr 8
0040A250
0040A250 push esi
0040A251 call key_fun_00
0040A256 cmp ax, 1
0040A25A jnz short skc110
0040A25C mov ax, 1 ; 1 - error?
0040A260 pop esi
0040A261 retn
```

0040A262 skc110:

```

0040A262      mov     word ptr ds:key_fun, 18h ;
function 18!!!
0040A26B      xor     eax, eax
0040A26D      mov     esi, [esp+4+arg_0]
0040A271      mov     word ptr ds:key_result_1,
magic1
0040A27A      mov     word ptr ds:key_par3, ax
0040A280      push    offset key_par4
0040A285      mov     word ptr ds:key_par4, ax
0040A28B      push    offset key_par3
0040A290      mov     word ptr ds:key_par1, magic2
0040A299      mov     word ptr ds:key_par2, magic3
0040A2A2      push    offset key_par2
0040A2A7      mov     [esi], eax
0040A2A9      push    offset key_par1
0040A2AE      push    offset key_result_1
0040A2B3      push    offset key_fun
0040A2B8      call    main_key_check      ; here our
call
0040A2BD      mov     word ptr ds:main_result, ax
0040A2C3      add     esp, 18h

```

و الآن سنجد بعض المقارنات (any thing) cmp ax, وستعرف قيمه التي يتوقعها البرنامج من الدونجل ويمكنك استخدام برنامج HIEW لتجربة هذه القيم وستجد أن البرنامج يعمل كاملا
الاستنتاج:

☞ كسر حماية الدونجل ليس أصعب من كسر حماية السيريال وبرامج shareware عادة يكون كسرها أصعب فأنت تحتاج إلى معرفه جيده

بالدونجل وبرنامج للتحويل مثل W32Dasm حتى تجد عناوين النداء

لدوال API وبرنامج softice بالتأكد وتصبح على الطريق الصحيح

❶ لا تبدأ بوضع نقاط توقف باستخدام BPIO حاول أن تجد اضعف

اتصال يمكن كسره بين البرنامج ودوال الدونجل.

❷ لا يتم استخدام كل دوال الدونجل فجرب أولاً أبسط الدوال.

❸ اختبر تشغيل البرنامج بتركه يعمل لمدة دقيقة أو أكثر فعادة يتم اختبار

الدونجل مره أخرى كل مده من الوقت كما اختبر كل وظائف البرنامج

❹ احترس من اختبار الكسر بوجود دونجل في الكمبيوتر فقد يؤدي هذا

إلى تلف الدونجل بسبب إجراء تدمير ذاتي 😊.

حماية الاسطوانات المدمجة:

بالنسبة لهذا النوع من الحماية فهي كلها تشترك في فحص بيانات معينه من الاسطوانة مثل عنوان معين أو عدد المقاطع بها أو سيريال مخفي بها وهكذا وإذا لم يجد البرنامج الاسطوانة فانه يقوم بعرض رسالة ثم يخرج من البرنامج

أكثر البرامج استخداما لهذا النوع من الحماية هي الألعاب كما يتم أيضا وضع حماية إضافية على هذه الاسطوانات تمنع نسخ الاسطوانة بالكامل عن طريق برامج مثل Alcohol أو Blind Write وفيما يلي نتعرف كيف نتغلب على هذه الوسائل.

واحد أهم أنواع الحماية هي حماية CD Lock وهي تحمي الاسطوانات من إمكانية عمل نسخه مطابقة منها وسأخذ مثال على هذا النوع لعبة "Thief Dark Project"

و الآن السؤال كيف نتعرف على نوع الحماية المستخدمة ؟ قم بفحص الاسطوانة وشاهد الملفات المستخدمة بها فإذا وجدت أربع ملفات لهم الامتداد .afp وحجم الملف اكبر من 600 MB فهذه حماية CD Lock بالتأكيد وقد تستخدم هذه النوع من الحماية تقنيه تسمى Overtime Protection ولكنها ليست موجودة في هذه اللعبة ولكننا سنقوم بتعلم فكها على أية حال ، كما سنقوم الآن بالبحث عن هذه الملفات باستخدام برنامج W32Dasm ويجب أن تشاهد الكود التالي:

* Referenced by a CALL at Addresses:

|:0050D80E , :0050DA6F , :0050DA8E

:00526100 8A442404 mov al, byte ptr [esp+04]

:00526104 53 push ebx

:00526105 56 push esi

:00526106 57

push edi

* Possible StringData Ref from Data Obj ->"rb"

:00526107 6838056100

push 00610538

* Possible StringData Ref from Data Obj ->"d:\KXEP.AFP"

:0052610C 6878046100

push 00610478

:00526111 A278046100

mov byte ptr [00610478], al

-> cdletter:\file.afp

:00526116 A288046100

mov byte ptr [00610488], al

-> ""

:0052611B A298046100

mov byte ptr [00610498],

al -> ""

:00526120 A2A8046100

mov byte ptr [006104A8],

al -> ""

:00526125 E886550700

call 0059B6B0

:0052612A 8BF0

mov esi, eax

:0052612C 83C408

add esp, 00000008

:0052612F 85F6

test esi, esi

:00526131 7504

jne 00526137

:00526133 33FF

xor edi, edi

:00526135 EB2C

jmp 00526163

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526131(C)

:00526137 6A00

push 00000000

:00526139 6800508229

push 29825000

:0052613E 56

push esi

:0052613F E8CC710700

call 0059D310

```
:00526144 83C40C      add esp, 0000000C
:00526147 56             push esi
:00526148 E8F35F0700    call 0059C140
:0052614D 83C404      add esp, 00000004
:00526150 33C9        xor ecx, ecx
:00526152 83F834      cmp eax, 00000034
:00526155 0F94C1      sete cl
:00526158 56             push esi
:00526159 8BF9        mov edi, ecx
:0052615B E890530700    call 0059B4F0
:00526160 83C404      add esp, 00000004
```

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526135(U)

```
|
:00526163 33DB        xor ebx, ebx
```

* Possible StringData Ref from Data Obj ->"rb"

```
|
:00526165 683C056100  push 0061053C
:0052616A 85FF        test edi, edi
```

* Possible StringData Ref from Data Obj ->"d:\KZXP.AFP"

```
|
:0052616C 6888046100  push 00610488
:00526171 0F95C3      setne bl
:00526174 E837550700    call 0059B6B0
:00526179 8BF0        mov esi, eax
:0052617B 83C408      add esp, 00000008
:0052617E 85F6        test esi, esi
:00526180 7504        jne 00526186
:00526182 33FF        xor edi, edi
```

:00526184 EB2C

jmp 005261B2

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526180(C)

:00526186 6A00	push 00000000
:00526188 6800E88229	push 2982E800
:0052618D 56	push esi
:0052618E E87D710700	call 0059D310
:00526193 83C40C	add esp, 0000000C
:00526196 56	push esi
:00526197 E8A45F0700	call 0059C140
:0052619C 83C404	add esp, 00000004
:0052619F 33D2	xor edx, edx
:005261A1 83F857	cmp eax, 00000057
:005261A4 0F94C2	sete dl
:005261A7 56	push esi
:005261A8 8BFA	mov edi, edx
:005261AA E841530700	call 0059B4F0
:005261AF 83C404	add esp, 00000004

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526184(U)

:005261B2 85DB	test ebx, ebx
:005261B4 740B	je 005261C1
:005261B6 85FF	test edi, edi
:005261B8 7407	je 005261C1
:005261BA BB01000000	mov ebx, 00000001
:005261BF EB02	jmp 005261C3

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:005261B4(C), :005261B8(C)

|:005261C1 33DB xor ebx, ebx

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:005261BF(U)

* Possible StringData Ref from Data Obj ->"rb"

|:005261C3 6840056100 push 00610540

* Possible StringData Ref from Data Obj ->"d:\TFJB.AFP"

|:005261C8 6898046100 push 00610498
 |:005261CD E8DE540700 call 0059B6B0
 |:005261D2 8BF0 mov esi, eax
 |:005261D4 83C408 add esp, 00000008
 |:005261D7 85F6 test esi, esi
 |:005261D9 7504 jne 005261DF
 |:005261DB 33FF xor edi, edi
 |:005261DD EB2C jmp 0052620B

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:005261D9(C)

|:005261DF 6A00 push 00000000
 |:005261E1 6800F88329 push 2983F800
 |:005261E6 56 push esi

```
:005261E7 E824710700    call 0059D310
:005261EC 83C40C      add esp, 0000000C
:005261EF 56          push esi
:005261F0 E84B5F0700    call 0059C140
:005261F5 83C404      add esp, 00000004
:005261F8 33C9        xor ecx, ecx
:005261FA 83F835      cmp eax, 00000035
:005261FD 0F94C1      sete cl
:00526200 56          push esi
:00526201 8BF9        mov edi, ecx
:00526203 E8E8520700    call 0059B4F0
:00526208 83C404      add esp, 00000004
```

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

```
:005261DD(U)
|
:0052620B 85DB        test ebx, ebx
:0052620D 740B        je 0052621A
:0052620F 85FF        test edi, edi
:00526211 7407        je 0052621A
:00526213 BB01000000    mov ebx, 00000001
:00526218 EB02        jmp 0052621C
```

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:

```
:0052620D(C), :00526211(C)
|
:0052621A 33DB        xor ebx, ebx
```

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

```
:00526218(U)
|
```

* Possible StringData Ref from Data Obj ->"rb"

:0052621C 6844056100 push 00610544

* Possible StringData Ref from Data Obj ->"d:\IGYQ.AFP"

```

:00526221 68A8046100                      push 006104A8
:00526226 E885540700                      call 0059B6B0
:0052622B 8BF0                                  mov esi, eax
:0052622D 83C408                              add esp, 00000008
:00526230 85F6                                  test esi, esi
:00526232 7504                                  jne 00526238
:00526234 33FF                                  xor edi, edi
:00526236 EB2C                                  jmp 00526264
    
```

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

:00526232(C)

```

:00526238 6A00                                  push 00000000
:0052623A 6800688129                              push 29816800
:0052623F 56                                      push esi
:00526240 E8CB700700                              call 0059D310
:00526245 83C40C                                  add esp, 0000000C
:00526248 56                                      push esi
:00526249 E8F25E0700                              call 0059C140
:0052624E 83C404                                  add esp, 00000004
:00526251 33D2                                  xor edx, edx
:00526253 83F874                                  cmp eax, 00000074
:00526256 0F94C2                                  sete dl
:00526259 56                                      push esi
:0052625A 8BFA                                  mov edi, edx
    
```

:0052625C E88F520700

call 0059B4F0

:00526261 83C404

add esp, 00000004

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526236(U)

:00526264 85DB

test ebx, ebx

:00526266 740D

je 00526275

:00526268 85FF

test edi, edi

:0052626A 7409

je 00526275

:0052626C B801000000

mov eax, 00000001

:00526271 5F

pop edi

:00526272 5E

pop esi

:00526273 5B

pop ebx

:00526274 C3

ret

* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:

|:00526266(C), :0052626A(C)

:00526275 5F

pop edi

:00526276 5E

pop esi

:00526277 33C0

xor eax, eax

:00526279 5B

pop ebx

:0052627A C3

ret

ويمكننا أن نقوم بفك هذه الحماية بأكثر م طريقة بناء على ما سبق:

أول طريقة:

حماية السريال

يمكننا كسر الحماية عن طريق التعليمة NOP أو 9090 في النظام السداسي وهذه التعليمة تعنى أن يكمل البرنامج تنفيذ باقي الأسطر التالية وسنقوم بوضعها بدلا من أمر القفز التالي

```
:00526264 85DB      test ebx, ebx
:00526266 740D      je 00526275  -> NOP ضع
:00526268 85FF      test edi, edi
:0052626A 7409      je 00526275  -> Nop ضع
:0052626C B801000000 mov eax, 00000001
```

وهذه الوسيلة البسيطة لا تجدي مع كل الألعاب لذلك سنقوم الآن بدراسة الطريق التي تستخدمها هذا النوع من الحماية وإلغاؤها تماما . والآن إذا نظرت إلى ملفات .afp. ستجد أن هذه الحماية توجد بها نفس الكود التالي:

```
:00526125 E886550700 call 0059B6B0 ->> file.afp
:0052612A 8BF0      mov esi, eax  ->> فحص إذا كان
اللف موجودا
:0052612C 83C408      add esp, 00000008
:0052612F 85F6      test esi, esi
:00526131 7504      jne 00526137
:00526133 33FF      xor edi, edi
:00526135 EB2C      jmp 00526163
```

* Referenced by a (U)nconditional or (C)onditional Jump at Address:

|:00526131(C)

```
:00526137 6A00      push 00000000
:00526139 6800508229 push 29825000
:0052613E 56      push esi
```

```

:0052613F E8CC710700    call 0059D310 ->>
SetFilePointer
:00526144 83C40C            add esp, 0000000C
:00526147 56                push esi
:00526148 E8F35F0700    call 0059C140 ->> ReadFile
:0052614D 83C404            add esp, 00000004
:00526150 33C9              xor ecx, ecx
:00526152 83F834            cmp eax, 00000034 ->> مقارنه
    
```

مما سبق يمكننا فهم طريقة الحماية المستخدمة فهي أولاً تختبر إذا كان الملف موجوداً أم لا ثم تقوم بتحميل بايت واحد من الملف إلى الذاكرة ثم تقوم بمقارنته عن طريق التعليمة CMP ولكي نستطيع إلغاء هذه الحماية يجب أن ننشئ أربع ملفات .afp بهم البايت الذي يتم مقارنته وسنجد في هذه اللعبة البايتات التالية مطلوبة:

```

- KXEP.AFP  BYTE 34
- KZXP.AFP  BYTE 57
- TFJB.AFP  BYTE 35
- IGYQ.AFP  BYTE 74
    
```

و لكننا لم ننتهي بعد فإذا جربت تشغيل اللعبة بعد هذه التعديلات فسترى رسالة خطأ لماذا؟ لأن البرنامج يبدأ بقراءة ملف ويتم إرجاع القيمة 1- لذلك يجب تغيير مؤشر قراءة الملف كما يلي:

```

:00526137 6A00            push 00000000
:00526139 6800508229      push 29825000 ->> Offset !!!!
:0052613E 56                push esi
:0052613F E8CC710700    call 0059D310 ->>
SetFilePointer
    
```

قم بتغيير السطر 00526139 إلى

PUSH 6800000000 (000000000 in decimal)

و يجب تغيير هذا المؤشر عند قراءة أي من الملفات الأربعة

ملحوظة: قد لا يقتصر الأمر على هذا الحد فقد تتضمن الحماية فحص لملفات الموسيقى الخاصة باللعبة أو تشفير ملفات afp لذلك يجب تجربة الكراك الذي قمت به جيدا فإذا لم يعمل بعد الدخول في اللعبة فيجب فحص إمكانية التشفير وفك هذا التشفير (يمكنك معرفة المزيد عن طرق التشفير من فصل حماية التشفير)

من أمثلة الألعاب التي تستخدم هذا النوع من الحماية هي (مستوى الصعوبة بين أقواس)

Gangsters: (hard - multi verif+cd check encrypted)
Tomb Raider 3: (Normal)
Thief - Dark Project : (easy)
Commando: (easy)

طريقة عامة لفك معظم أنواع حماية الاسطوانة:

❶ قم بتشغيل اللعبة بدون الاسطوانة واكتب أو تذكر الرسالة التي تظهر وتطلب وجود الاسطوانة الأصلية.

❷ قم الآن بتشغيل برنامج D23Dasm وافتح ملف اللعبة.

❸ قم الآن بفتح نافذة النصوص String Data References وقم بالبحث عن الرسالة في هذه النافذة وعندما تجدها قم بالضغط المزدوج عليها ثم أغلق النافذة.

❹ يجب أن تكون الآن في المكان الذي يطلب الاسطوانة ابحث عن أوامر الCall أو أوامر القفز مثل JE أو JNE قبل مكان الرسالة وقم بتسجيل العنوان Offset الخاص بكل منها.

❺ والآن اذهب إلى برنامج HIEW من خلال هذا البرنامج قم بفتح الملف السابق وابحث عن العنوان عن طريق F5 .

لاحظ أن كل رقمين في النظام السداسي يعادل بايت واحد فمثلا الرقم

E92BF9BF74

يمثل البايتات التالية:

E9-2B-F9-BF-74

وبالتالي يمثل هذا الرقم 5 بايت

اضغط F3 لتعديل الملف وأملأ البايتات الخمسة بالأرقام 90 90 90 90 90 التي تعبر التعليم NOP اضغط الآن F10 ويجب أن تعمل اللعبة الآن بدون الحاجة إلى الاسطوانة.

لاحظ أننا كما ذكرنا سابقا هناك بعض الحماية الآن تعتمد على التشفير لذلك يجب فك Unpack الملف الحقيقي أولا ثم إتباع الطرق التقليدية لإلغاء الحماية.

كلمة أخيره:

لا يمكنك أبدا تعلم الكراك عن طريق الالتزام
 بخطوات محدده بعينها ولكن يجب عليك أن تفهم تعليمات
 الاسمبلى جيدا ويجب أيضا أن تعرف إحدى لغات البرمجة
 القوية مثل C++ Builder أو Delphi حتى يمكنك أن
 تتخيل نفسك مكان المبرمج وما يجب أن يتم عمله وإذا
 أردت أن تصبح كراكر محترف يجب أن يكون لديك صبر
 طويل — كما يجب أن تجتهد للبحث والدراسة عن
 طرق الحماية الجديدة فكل بضعة شهور قليله تقريبا تظهر
 إحدى الشركات العالمية بنوع جديد من وسائل التشفير.