

REA – cRaCkErTeAm



Reverse Engineering Association

The basic rules of cracking (to help ya become a good cracker)	5
CrAcKmE tUtOrIaLs	6
CrackMe : \$KORBUT - Anti Patching Technology - Level 2	6
CrackMe : \$KORBUT - Rif Crackme #2 - Level 2.....	9
CrackMe : \$KORBUT - RIF Crackme #3 - Level 4.....	12
CrackMe : \$KORBUT - Rif Crackme #4 - Level 3.....	15
CrackMe : \$KORBUT KeyGen Me #1 - Level 1	18
CrackMe : \$KORBUT Keygenme #1 - Level 1	21
CrackMe : [Flasher] - Delphi CrackME - Level 3.....	22
CrackMe : [v0!d] - crackme01 - Level 1	24
CrackMe : [v0!d] - crackme02 - Level 0	26
CrackMe : abex 2 - Level 1.....	29
CrackMe : abex 3 - Level 1.....	32
CrackMe : abex 4 - Level 1.....	34
CrackMe : abex 5 - Level 1.....	36
CrackMe : acid bytes - cff crackme #4 - Level 0.....	38
CrackMe : analyst - keygenning4newbies #1 - Level 1.....	41
CrackMe : CrackMe by v1ru5 - Level 2	43
CrackMe : exrek - crackme #1 - Level 0.....	44
CrackMe : Fant0m - Crackme 5 - Level 0	48
CrackMe : Fant0m - Crackme 6 - Level 1	51
CrackMe : xyzero - KeygenMe #1 Tangerine - Level 1	54
CrackMe : yoda - lame crackme - Level 0.....	58
CrackMe : Zephyrous - crackme #1 - Level 1	60
CrackMe : Zephyrous - crackme #2 - Level 2	65
CrackMe : Zephyrous - crackme #3 - Level 3	70
CrackMe : Zero - ZeroMakesPiPi - Level 2	76
CrAcKsOfT tUtOrIaLs	78
SoftWare : 1st Desktop Guard 1.5.....	78
SoftWare : 1st SMTP Server 2.5	80
SoftWare : PopUp Inspector 1.5.432.....	82
SoftWare : Hexacolor 3.0.....	85
SoftWare : Space 1.3.4	86
SoftWare : EasyMoney 1.21	88
SoftWare : Ace Money 3.5.6.....	91
SoftWare : Acoustica Mixcraft 1.10.15.....	97
SoftWare : Acoustica MP3 CD Burner 3.01 Build 71	106
SoftWare : Advanced Emailer 2.6.....	109
SoftWare : Alive CD Ripper	111
SoftWare : Alive MP3 CD Burner	114
SoftWare : Alleycode HTML Editor 1.3.....	117
SoftWare : Arial CD Ripper 1.3.5	121
SoftWare : Ashampoo Privacy Protector 1.04 (se)	123
SoftWare : Ashampoo SeeYa! 2.2.0.5 (se)	127
SoftWare : Ashampoo UnInstaller Suite Plus 1.32 (se)	135
SoftWare : AtomicClockService 1.0	138
SoftWare : Audio Edit 3.5	141
SoftWare : Automatic Remasterer Disk 1.7.1.1	142
SoftWare : Becky! Internet Mail 2.10.03	146
SoftWare : BlazingTools Personal Antispy 1.2.0.0.....	149
SoftWare : BugTracker 2.0	152
SoftWare : Child Control 2004 6.889.0	157

SoftWare	: Clever Boxman V2.5	159
SoftWare	: Color Schemer 3.1	162
SoftWare	: Computer Alarm Clock 2.0.....	164
SoftWare	: Cyclone Internet History Killer Pro 3.60	166
SoftWare	: DialogBlocks 1.52.....	172
SoftWare	: DiskArcher Backup Utility 2.01	175
SoftWare	: Dpeg Suite 6.13.....	177
SoftWare	: DU Meter 3.07 Build 192	183
SoftWare	: Dual DVD Copy Gold 3.0	187
SoftWare	: DVD-Cloner 2.32.....	191
SoftWare	: EarMaster Pro 4.0	194
SoftWare	: Easy Desktop Keeper 1.5	197
SoftWare	: Evidence Destructor 2.0.....	199
SoftWare	: ExamXML 3.12	201
SoftWare	: Exe Password 2004 1.1	203
SoftWare	: FairStars Audio Converter 1.4.0.6	206
SoftWare	: File and Folder Protector 1.89	210
SoftWare	: Flash2X EXE Packager 1.0.1.....	214
SoftWare	: FontMap 2.37	217
SoftWare	: FreshDiagnose 6.70.....	219
SoftWare	: HelpBlocks 1.12 for Windows.....	221
SoftWare	: HTML Code Cleaner 3.20.0	224
SoftWare	: HTML Page Guardian v3.0.....	226
SoftWare	: HTML Password v3.1 Pro	227
SoftWare	: IconTOY 3.1	229
SoftWare	: ImTOO Audio Encoder 1.0.1 b 707.....	232
SoftWare	: ImTOO CD Ripper 1.0.12 b 710	235
SoftWare	: ImTOO DVD Ripper 3.0.12 b 331	238
SoftWare	: Installer2go 4.0.6	241
SoftWare	: Mail Bomber 9.1	247
SoftWare	: MP3 Cutter Joiner 1.00	249
SoftWare	: MP3 RM Converter 1.00.....	250
SoftWare	: MP3StickMan 1.1	252
SoftWare	: My Drivers 2005 v3.11 build 2600.....	254
SoftWare	: PDF Information Editor 1.1	256
SoftWare	: BlazingTools Perfect Keylogger 1.6.0.0	259
SoftWare	: Plato DVD Ripper 1.13	261
SoftWare	: Plato DVD to MP3 Ripper.....	266
SoftWare	: Power Edit 2.03.....	268
SoftWare	: Quick Screen Capture 2.2.0	270
SoftWare	: Quick Screenshot Maker 2.1	279
SoftWare	: Real Spy Monitor 1.99	284
SoftWare	: RF1 Player 1.4.0 beta.....	288
SoftWare	: Smart Video Converter 1.5.3	294
SoftWare	: Snooper 1.35.28	298
SoftWare	: Softsilver Transformer 2.5.1	302
SoftWare	: Sparkle Flash Keeper 3.0	307
SoftWare	: Sparkle SWF Desktop 1.0.....	312
SoftWare	: SpeederXP v1.60 full	316
SoftWare	: StoryLines 1.27	318
SoftWare	: TextIndexer 4.27	321
SoftWare	: Time & Chaos 6.0.2.7	322
SoftWare	: Trash it! 1.80.....	324

SoftWare	: Ultra Calendar Reminder 2.3	330
SoftWare	: Visual MP3 CD Burner.....	338
SoftWare	: WAV MP3 Converter 1.00	341
SoftWare	: X DVD Ripper 1.2.1	344
SoftWare	: Xilisoft DVD Audio Ripper 1.0.25 b 804.....	348
SoftWare	: XnView for Windows 1.7.03	351
SoftWare	: Guitar Pro 4.1.0.....	356
SoftWare	: DriveScrubber Professional 2.0a	366
SoftWare	: GetSmile 1.5 Full	370
SoftWare	: Ace DVD BackUp 1.2.11	374
SoftWare	: Applet Cool Text Wizard Version 1.0	383
SoftWare	: Applet Menu Wizard Version 1.0.....	388
SoftWare	: Applet SlidingMenu Wizard Version 1.0	391
SoftWare	: Auto Connect v1.01	394
SoftWare	: CMB Audio Player v2.0.0	401
SoftWare	: Folder View v2.1	405
SoftWare	: Hot Corners v2.21	411
SoftWare	: Password Pro v2.01.....	418
SoftWare	: Printer Express v1.25	420
SoftWare	: Super Cleaner v2.75	426
SoftWare	: SysDate v1.3	433
SoftWare	: Sothink SlidingMenu 2.0	440
SoftWare	: Sothink CoolMenu v3.0	443
SoftWare	: CoffeeCup Firestarter v6.5.....	447
SoftWare	: Quick Notes Plus v5.0 (Build 40)	450
SoftWare	: MagicTweak v2.70	456
SoftWare	: Magic Utilities 2004 v3.10	463
SoftWare	: Privacy Inspector v1.51	478
SoftWare	: AutoZIP Backup v1.1	481
SoftWare	: Batch Image Resizer v2.01	484
SoftWare	: Easy Photo Editor v1.6	493
SoftWare	: Picture Finder Pro v2.6	501
SoftWare	: EnCrypt PDF v2.1	506
SoftWare	: PDF Extract TIFF v1.5	511
SoftWare	: Lite FTP v2.5	514

====REA TEAM INFORMATION=====**517**

The basic rules of cracking (to help ya become a good cracker)

1. Hãy luôn luôn nhớ lấy điều này , bạn không thể bẻ khóa tất cả các phần mềm.Bạn không phải là người giỏi nhất , những kiến thức mà bạn biết được không bao giờ là đủ....
2. Mọi phần mềm đều có thể crack được.
3. Hãy chia sẻ sự hiểu biết của bạn , nếu bạn đã tìm được một vài mẹo nhỏ(thủ thuật) hãy nói cho những người khác để chia sẻ những gì mà bạn biết được , hãy viết các bài hướng dẫn, các thủ thuật , và tất cả những gì bạn có thể làm được để giúp đỡ các thợ hệ cracker tiếp theo về cracking.
4. Hãy đọc thật nhiều các bài hướng dẫn , đây dường như là một quy luật không thể phá bỏ.Trước tiên cũng phải nói rằng chúng ta chưa chắc đã là người giỏi nhất,bởi vì những điều mà các cracker khác biết thì có thể chúng ta lại không biết và ngược lại những điều mà chúng ta biết thì có thể các cracker khác lại không biết . Vì vậy hãy thường xuyên đọc các bài hướng dẫn.
5. Hãy học để code , nếu các bạn có thể biết được một chương trình phức tạp nó hoạt động thế nào và nó được code thế nào thì thật quá dễ dàng để crack nó (Do đó bạn phải biết code vd Code KeyGen chẳng hạn : chính là chúng ta nghiên cứu quá trình tạo Key của chương trình đó ra sao , sau đó dựa vào đó chúng ta viết một Keygen sao cho nhập vào tên bất kì thì chương trình có thể tạo ra được Serial tương ứng với Username nhập vào)
6. Không nên chỉ là những kẻ chỉ biết sử dụng các công cụ luân phiên, hãy tìm lấy một công cụ , học lấy nó và làm chủ nó
7. Hãy biết thêm các công cụ khác , đừng chỉ biết các công cụ thông thường như Windasm, bạn nên biết thêm cả các chương trình dùng để Debug chương trình như Softice (hoặc Olly...)
8. Nếu bạn là thành viên của một tổ chức hay một nhóm nào đó, hoặc thậm chí có thể bạn chỉ là một thành viên thử thì bạn cũng nên gặp mọi người , họ sẽ sẵn sàng giúp bạn. Và bạn cũng có thể giúp họ nữa là đăng khác , Bạn sẽ được biết về các kiểu bảo vệ của những soft khác nhau để từ đó bạn có thể tiếp tục làm việc ngày càng tốt hơn
9. Bạn thân bạn cũng phải thường xuyên được updated, điều này là rất quan trọng . Bạn nên sử dụng các công cụ mới nhất , học những kiểu bảo vệ mới nhất....
10. Tự bản thân bạn phải khám phá mọi thứ, hãy tự thực hành với tất cả các thủ thuật mà bạn đã tìm được kể cả những thủ thuật mà bạn chưa từng đọc. Và cũng đừng quên đem tất cả những thủ thuật mà bạn biết được truyền lại cho những người khác , tự nghiên cứu là phương pháp tốt nhất..
11. Đừng có phá hoại hay can thiệp vào công việc của người khác (hãy tôn trọng lẫn nhau), họ đã phải làm việc rất vất vả . Cũng đừng có phá hoại các file cracks , keygen hay creat serials của những người khác dù là nó chưa được hay lắm.
12. Cuối cùng là các bạn hãy code thật nhiều, đọc thật nhiều , crack thật nhiều và viết thật nhiều các bạn sẽ trở thành một cracker.

Cracking is an art , and like any art , it must be taught , practiced , and mastered . This journal was created just for that purpose to help you master the art of cracking .

**** -REA-cRaCkErTeAm -****

CrAcKmE tUtOrIaLs

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	\$KORBUT - Anti Patching Technology - Level 2
Coder :	\$KORBUT
Cracker :	Moonbaby
Type :	Serial
Packed :	N/A
Language :	MASM32 / TASM32
Crack Tool :	OllyDbg 1.09d, PeiD 0.92
Unpack Tool :	N/A
Request :	PATCH

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta không tìm thấy chuỗi thông báo nào cả. Tuy nhiên ta tìm được hàm **GetDlgItemTextA** , vì vậy ta đặt BreakPoint tại hàm này :
00401099 |. E8 AC010000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA

II – Cracking :

- Qúa trình kiểm tra chuỗi Fake Serial được thực hiện như sau :

```

004010A3 |. 8D05 64304000 LEA EAX,DWORD PTR DS:[403064] ; <== Fake Serrial : S[i]
004010A9 |. 8D1D 96304000 LEA EBX,DWORD PTR DS:[403096] ; <== Default String : dStr[i]
004010AF |. 8A4B 7A    MOV CL,BYTE PTR DS:[EBX+7A] ; <== dStr[x] = "z"
004010B2 |. 3808      CMP BYTE PTR DS:[EAX],CL ; <== S[0] == "z"
004010B4 |. 74 02      JE SHORT apt.004010B8 ; <== if Equal go to Next Check
004010B6 |. EB 22      JMP SHORT apt.004010DA ; <== If Not go to NAG
004010B8 |> 8A4B 52    MOV CL,BYTE PTR DS:[EBX+52] ; <== dStr[x] = "R"
004010BB |. 3848 01    CMP BYTE PTR DS:[EAX+1],CL ; <== S[1] == "R"
004010BE |. 74 02      JE SHORT apt.004010C2 ; <== if Equal go to Next Check
004010C0 |. EB 18      JMP SHORT apt.004010DA ; <== If Not go to NAG
004010C2 |> 8A48 01    MOV CL,BYTE PTR DS:[EAX+1] ; <== S[1]
004010C5 |. 0208      ADD CL,BYTE PTR DS:[EAX] ; <== Temp = S[0] + S[1] = 0xCC
004010C7 |. 3848 03    CMP BYTE PTR DS:[EAX+3],CL ; <== S[3] == Temp == 0xCC
004010CA |. 74 02      JE SHORT apt.004010CE ; <== if Equal go to Next Check
004010CC |. EB 0C      JMP SHORT apt.004010DA ; <== If Not go to NAG
004010CE |> 3248 02    XOR CL,BYTE PTR DS:[EAX+2] ; <== Temp = S[2] xor S[3]
004010D1 |. 3848 05    CMP BYTE PTR DS:[EAX+5],CL ; <== S[5] = Temp
004010D4 |. 0F84 CE000000 JE apt.004011A8 ; <== if Equal Congratulation !

```

- Ở đây ta nhận thấy, giá trị của S[3] = 0xCC không thể gõ từ bàn phím được . Và cách bắt buộc là phải PATCH nó . Ở đây ta sửa lệnh hai lệnh :

```
004010CA /74 02    JE SHORT apt.004010CE ; <== if Equal go to Next Check
```

thành

004010CA /75 02 JNZ SHORT apt.004010CE ; <== if Equal go to Next Check
và
 004010D4 /0F84 CE000000 JE apt.004011A8 ; <== if Equal Congratulation !
thành
 004010D4 /0F85 CE000000 JNZ apt.004011A8 ; <== if Equal Congratulation !

- Trong Olly ta Click chuột phải chọn **Copy to Executable → All modification → Copy All**. Click chuột phải vào màn hình mới xuất hiện chọn **Saved File**. Lưu lại file với tên mới . Chạy thử File này, **CRASH**.

- Load chính File này bằng Olly để tìm đoạn gây CRASH. Ta chú ý đến đoạn CODE này :

0040104D . E8 04020000 CALL <JMP.&USER32.SetDlgItemTextA> ; \SetDlgItemTextA
 00401052 . E8 95010000 CALL apt-p.004011EC ; <== Trace Into Here

- Quá trình kiểm tra CrackMe có bị PATCH hay không diễn ra như sau :

00401203 |. 8D35 A3104000 LEA ESI,DWORD PTR DS:[4010A3] ; <== Value of Recent Add

00401209 |. 8D3D DA104000 LEA EDI,DWORD PTR DS:[4010DA]

0040120F |. 8D05 1E304000 LEA EAX,DWORD PTR DS:[40301E] ; <== Value of Modified Original Add

00401215 . 33DB	XOR EBX,EBX
00401217 . EB 14	JMP SHORT apt-p.0040122D
00401219 > 3BF7	/CMP ESI,EDI
0040121B . 74 14	JE SHORT apt-p.00401231
0040121D . 8A16	MOV DL,BYTE PTR DS:[ESI] ; <== Each value of Recent Add.
0040121F . 8A08	MOV CL,BYTE PTR DS:[EAX] ; <== Each value of Modified Original Add.

00401221 . 80F1 90	XOR CL,90 ; <== Convert to Value of Original Add.
00401224 . 38D1	CMP CL,DL ; <== if Crackme Patched
00401226 . 74 01	JE SHORT apt-p.00401229 ; <== Two value at that position is not equal
00401228 . 43	INC EBX ; <== EBX ++

00401229 > FEC1	INC CL
0040122B . 46	INC ESI
0040122C . 40	INC EAX
0040122D > 0BDB	OR EBX,EBX ; <== if EBX !=0
0040122F .^ 74 E8	JE SHORT apt-p.00401219 ; <== End check Modification
00401231 > 0BDB	OR EBX,EBX ; <== Check EBX (0 or 1)
00401233 74 07	JE SHORT apt-p.0040123C ; <== if PATCHED --> CRASH
00401235 . 6A 00	PUSH 0 ; /ExitCode = 0
00401237 . E8 02000000	CALL <JMP.&KERNEL32.ExitProcess> ; \ExitProcess
0040123C > C3	RETN ; <== if not PATCHED, check Serial

- Như vậy, đê sau khi PATCH đoạn kiểm tra Serial mà CrackMe không bị CRASH ta cần PATCH thêm một điểm nữa, đó là quá trình kiểm tra CrackMe có bị PATCH hay không . Ta thay :

00401233 /74 07 JE SHORT apt-p.0040123C ; <== if PATCHED --> CRASH

thành

00401233 /75 07 JNZ SHORT apt-p.0040123C ; <== if PATCHED --> check Serial

- Chạy thử CrackMe, không còn bị CRASH.

- Như vậy với cách PATCH ở trên, thì chuỗi Serial yêu cầu nhập phải là “zRxxxx”, có chiều dài tối thiểu là 6 ký tự . Nếu :

/+/- CrackMe thoả với mọi chuỗi Serial có chiều dài tối thiểu 6 ký tự thì ta thay toàn bộ JE thành JMP (tránh tình trạng vô tình hai ký tự so sánh giống nhau).

chuyển

004010B4 /74 02 JE SHORT apt.004010B8 ; <== if Equal go to Next Check

thành

004010B4	/EB 02	JMP SHORT apt.004010B8	; <== Next Check without check
<i>chuyển</i>			
004010BE	./74 02	JE SHORT apt.004010C2	; <== if Equal go to Next Check
<i>thành</i>			
004010BE	/EB 02	JMP SHORT apt.004010C2	; <== Next Check without check
<i>chuyển</i>			
004010CA	/74 02	JE SHORT apt.004010CE	; <== if Equal go to Next Check
<i>thành</i>			
004010CA	/EB 02	JMP SHORT apt.004010CE	; <== Next Check without check
<i>chuyển</i>			
004010D4	/0F84 CE000000 JE	apt.004011A8	; <== if Equal Congratulation !
<i>thành</i>			
004010D4	/E9 CF000000 JMP	apt.004011A8	; <== Always Congratulation !
<i>và đoạn kiểm tra sự thay đổi nếu bị PATCH</i>			
00401233	/74 07	JE SHORT apt-p.0040123C	; <== if PATCHED --> CRASH
<i>thành</i>			
00401233	/75 07	JNZ SHORT apt-p.0040123C	; <== if PATCHED --> check Serial
<i>hay thành</i>			
00401233	/EB 07	JMP SHORT apt.0040123C	; <== Always Go to check Serial
/+/- Còn nếu muốn chuỗi Serial đúng với mọi chiều dài chuỗi, không cần kiểm tra ta làm như sau			
<i>chuyển</i>			
004010A1	/72 37	JB SHORT apt.004010DA	; <== go to NAG if Len.S less than 6 charts
<i>thành</i>			
004010A1	/EB 31	JMP SHORT apt.004010D4	; <== Always go to Last Check
<i>chuyển</i>			
004010D4	/0F84 CE000000 JE	apt.004011A8	; <== if Equal Congratulation !
<i>thành</i>			
004010D4	/E9 CF000000 JMP	apt.004011A8	; <== Always Congratulation !
<i>và đoạn kiểm tra sự thay đổi nếu bị PATCH</i>			
00401233	/74 07	JE SHORT apt-p.0040123C	; <== if PATCHED --> CRASH
<i>thành</i>			
00401233	/75 07	JNZ SHORT apt-p.0040123C	; <== if PATCHED --> check Serial
<i>hay thành</i>			
00401233	/EB 07	JMP SHORT apt.0040123C	; <== Always Go to check Serial

/*/*/*/ - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : N / A

User : VHT-cRaCkErS

Serial : N / A

III – KeyGen :

N/A

IV – SourceCode (VC++) :

N/A

V – End of Tut :

- Finished - 17/06/2004

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	\$KORBUT - Rif Crackme #2 - Level 2
Coder :	\$KORBUT
Cracker :	Moonbaby
Type :	Name / Serial
Packed :	N/A
Language :	MASM32 / TASM32
Crack Tool :	OllyDbg 1.09d, PeiD 0.92
Unpack Tool :	N/A
Request :	Correct Serial

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "- CONGRATULATIONS OLD RASCAL -" ở địa chỉ :
00401535 |. 68 EB314000 PUSH crackme2.004031EB ; /Text = "- CONGRATULATIONS OLD RASCAL -"
- Truy ngược lên ta thấy hàm **GetDlgItemTextA** , vì vậy ta đặt BreakPoint tại hàm này :
004014A0 |. E8 DD000000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA

II – Cracking :

- Nhấn F9 để chạy CrackMe. Nhập User và Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. Quá trình kiểm tra chiều dài chuỗi U và Fake Serial nhập nằm ngay sau BreakPoint :


```
004014A0 |. E8 DD000000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA
004014A5 |. 83F8 05     CMP EAX,5          ; <== Name atleast 5 charts
004014A8 |. 73 05     JNB SHORT crackme2.004014AF
```

-
- Quá trình mã hoá của CrackMe này rất đơn giản, như được diễn giải :
- ```
004014CF |. A0 10324000 MOV AL,BYTE PTR DS:[403210] ; <== U[0]
004014D4 |. 34 3C XOR AL,3C ; <== Temp = U[0] xor 0x3C
004014D6 |. 34 09 XOR AL,9 ; <== Temp = Temp xor 0x9
004014D8 |. 50 PUSH EAX ; <== Temp
004014D9 |. 0FBEC0 MOVSX EAX,AL ; <== Temp
004014DC |. 03F8 ADD EDI,EAX ; <== Temp
004014DE |. A0 88324000 MOV AL,BYTE PTR DS:[403288] ; <== S[0]
004014E3 |. 34 09 XOR AL,9 ; <== Temp_01 = S[0] xor 0x9
004014E5 |. 5B POP EBX ; <== Temp
004014E6 |. 32C3 XOR AL,BL ; <== Temp_01 = Temp_01 xor Temp
004014E8 |. 0FBEC0 MOVSX EAX,AL ; <== Temp_01
004014EB |. 03F8 ADD EDI,EAX ; <== Temp_01 = Temp_01 + Temp
```

```

004014ED |. A0 12324000 MOV AL,BYTE PTR DS:[403212] ; <== U[2]
004014F2 |. 8A1D 8A324000 MOV BL,BYTE PTR DS:[40328A]; <== S[2]
004014F8 |. 02C3 ADD AL,BL ; <== Temp = U[2] + S[2]
004014FA |. 0FBEC0 MOVSX EAX,AL ; <== Temp
004014FD |. 0FBEDB MOVSX EBX,BL ; <== S[2]
00401500 |. 03C3 ADD EAX,EBX ; <== Temp = Temp + S[2]
00401502 |. 03F8 ADD EDI,EAX ; <== Temp_01 = Temp_01 + Temp
00401504 |. A0 8C324000 MOV AL,BYTE PTR DS:[40328C] ; <== S[4]
00401509 |. 8A1D 14324000 MOV BL,BYTE PTR DS:[403214] ; <== U[4]
0040150F |. 0FBEC0 MOVSX EAX,AL ; <== S[4]
00401512 |. 0FBEDB MOVSX EBX,BL ; <== U[4]
00401515 |. 03C3 ADD EAX,EBX ; <== Temp = S[4] + U[4]
00401517 |. 03F8 ADD EDI,EAX ; <== Temp_01 = Temp_01 + Temp
00401519 |. 81F7 90000000 XOR EDI,90 ; <== Temp_01 = Temp_01 xor 0x90
0040151F |. 81FF 4F010000 CMP EDI,14F ; <== if (Temp_01 == 0x14F)
00401525 |. 74 05 JE SHORT crackme2.0040152C ; <== Congratulations !!!

```

- Tuy nhiên ở đây có một vấn đề có thể ảnh hưởng đến quá trình mã hóa . Câu lệnh chính :

0040150F |. 0FBEC0 MOVSX EAX,AL ; <== S[4]

- Ở đây, ban đầu EAX = 00000000, khi chuyển ký tự thứ tư vào AL thì EAX = 000000XX. Tuy nhiên khi chuyển MOVSX EAX,AL thì EAX = FFFFFFFXX. Giá trị này khi cộng với EBX = YY cho ta hai kết quả khác nhau :

/Option 1/- **Nếu XX + YY <= FF** thì giá trị của EAX = FFFFFFFZ . Kết quả này khi được cộng với giá trị của EDI = AA thì kết quả luôn luôn là BB ( cho dù thực tế ZZ + AA = BBB ) . Như vậy, trong quá trình tính toán, nếu ta không quan tâm đến sự chuyển đổi từ AL sang EAX thì kết quả EDI phải được tính bằng **EDI = EDI AND 0xFF**

/Option 2/- **Nếu XX + YY > FF** thì giá trị thực phải là 1000000ZZ, như vậy là bị tràn EAX. Và lúc này EAX = 000000ZZ . Kết quả này khi cộng với giá trị của EDI sẽ không bị không chế mà cho ra giá trị thực ( BB hay BBB ) . Như vậy, cần loại bỏ FFFFFF trước khi tính toán có nghĩa **EAX = EAX AND 0xFF**

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                |    |       |
|---------------------|----------------|----|-------|
| User : REA-cRaCkErS | Serial : IE*z~ | or | iR(F) |
| User : VHT-cRaCkErS | Serial : 1c tr | or | 9\}-x |

### III – KeyGen :

/Section 1/- Chiều dài của S tối thiểu là 5 ký tự .

/Section 2/- Kiểm tra giá trị **Temp = S[4] + U[4]** với **0xFF**

/Terrible Note/- KeyGen bị CRASH với User : ^%#%@

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
char reaRandomString[255]++;
int LenUser=0;
int i=0, Result=0;
int Temp=0, TempSerial=0;

```

```

LenUser=GetDlgItemText(IDC_Name, reaName, 64);
srand((unsigned)time(NULL));
i=0;

```

```

while (i < 96)
{
 reaRandomString[i] = (char) (32+i);
 i++;
}

if (LenUser < 5)
{
 MessageBox("=-*=-*=- Your name atleast 5 charts =*=-*=-","Hey !! Please input
your name again !! ");
}
else
{
 while (Result == 0)
 {
 i=0;
 while (i < 5)
 {
 reaSerial[i] = reaRandomString[rand() % 96];
 i++;
 }

 Temp = (reaName[0] ^ 0x3C) ^ 0x9;
 TempSerial = ((reaSerial[0] ^ 0x9) ^ Temp) + Temp;
 Temp = reaName[2] + reaSerial[2] * 2;
 if (Temp <= 0xFF)
 {
 TempSerial = (TempSerial + Temp) & 0xFF;
 }
 else
 {
 Temp = Temp & 0xFF;
 TempSerial = TempSerial + Temp;
 }
 Temp = reaSerial[4] + reaName[4];
 TempSerial = (TempSerial + Temp) ^ 0x90;

 if (TempSerial == 0x14F)
 {
 Result = 1;
 SetDlgItemText(IDC_Serial,reaSerial);
 }
 else
 {
 Result = 0;
 }
 }
}

```

**V – End of Tut :**

- Finished – ***17/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | \$KORBUT - RIF Crackme #3 - Level 4                 |
| Coder :       | \$KORBUT                                            |
| Cracker :     | Moonbaby                                            |
| Type :        | Number / String                                     |
| Packed :      | N/A                                                 |
| Language :    | MASM32 / TASM32                                     |
| Crack Tool :  | OllyDbg 1.09d, PeiD 0.92                            |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Number                                      |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "- CONGRATULATIONS OLD RASCAL -" ở địa chỉ :  
004015B7 |. 68 A7314000 PUSH crackme3.004031A7 ; /Text = "- CONGRATULATIONS  
OLD RASCAL -"
- Truy ngược lên ta thấy hàm **GetDlgItemInt** , vì vậy ta đặt BreakPoint tại hàm này :  
0040149D |. E8 62010000 CALL <JMP.&USER32.GetDlgItemInt> ; \GetDlgItemInt

### II – Cracking :

- Chiều dài chuỗi MagicWord được quy định :  
004014B3 |. E8 52010000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA  
004014B8 |. 83F8 0A CMP EAX,0A ; <== Length of String must be 10 charts
- CrackMe này lấy dữ liệu ở ô MagicNumber bằng cách sử dụng hàm **GetDlgItemInt** . Điều này có nghĩa số nhập vào ( 0 – 9 ) sẽ được chuyển sang giá trị ở dạng HEX (**MgValue**) . Ở đây ta có 3 trường hợp :
  - /+/- Nếu là số XXXXX thì số này được chuyển sang giá trị ở dạng HEX
  - /+/- Nếu chuỗi có dạng XXXXXY ( với X là số, Y là ký tự khác số ) số sẽ được chuyển sang giá trị ở dạng lấy từ số XXXXX ( chấp nhận dạng số )
  - /+/- Nếu chuỗi có dạng YXXXXX thì không được coi là số, như vậy giá trị trả về được lưu trữ là 0 ( NULL ) .
- Hay nói một cách khác, ô nhập này có hai dữ liệu đầu vào (1) có giá trị, hoặc (2) không có giá trị (NULL).
- CrackMe này được chia làm 11 lần kiểm tra . Trong đó có 10 lần kiểm tra các ký tự của chuỗi nhập vào . Lần thứ 11 kiểm tra giá trị tính toán .
- 10 lần kiểm tra này gần giống nhau, và rất đơn giản :

|             |                                             |                                            |
|-------------|---------------------------------------------|--------------------------------------------|
| 004014C8  . | 8D05 CC314000 LEA EAX,DWORD PTR DS:[4031CC] | ; <== U                                    |
| 004014CE  . | 8A18 MOV BL,BYTE PTR DS:[EAX]               | ; <== U[0]                                 |
| 004014D0  . | 33DF XOR EBX,EDI                            | ; <== Temp_01 = U[0] xor MgValue           |
| 004014D2  . | 8A48 01 MOV CL,BYTE PTR DS:[EAX+1]          | ; <== U[1]                                 |
| 004014D5  . | 33CF XOR ECX,EDI                            | ; <== Temp_02 = U[1] xor MgValue           |
| 004014D7  . | 03F1 ADD ESI,ECX                            | ; <== Temp_02                              |
| 004014D9  . | 38CB CMP BL,CL                              | ; <== if ((EBX and 0xFF) < (ECX and 0xFF)) |

004014DB 72 05 JB SHORT crackme3.004014E2 ; <== Continue

- Toàn bộ 10 quá trình so sánh này dựa vào đặc tính của lệnh **XOR** . Vì thế có thể xem 10 quá trình so sánh các giá trị được XOR với nhau như 10 quá trình so sánh từng ký tự với nhau . Vì thế ta có thể xem hai quá trình được diễn giải bên dưới đây là như nhau :

/With XOR/- So sánh các giá trị của từng ký tự sau khi XOR :

```
reaSerial[0] ^ MgValue < reaSerial[1] ^ MgValue && reaSerial[1] ^ MgValue > reaSerial[2] ^ MgValue
&& reaSerial[2] ^ MgValue < reaSerial[3] ^ MgValue && reaSerial[3] ^ MgValue > reaSerial[4] ^
MgValue && reaSerial[4] ^ MgValue < reaSerial[5] ^ MgValue && reaSerial[5] ^ MgValue <
reaSerial[6] ^ MgValue && reaSerial[6] ^ MgValue < reaSerial[7] ^ MgValue && reaSerial[7] ^
MgValue > reaSerial[8] ^ MgValue && reaSerial[8] ^ MgValue < reaSerial[9] ^ MgValue &&
reaSerial[9] ^ MgValue > reaSerial[0] ^ MgValue
```

/Without XOR/- So sánh từng ký tự với nhau :

```
reaSerial[0] < reaSerial[1] && reaSerial[1] > reaSerial[2] && reaSerial[2] < reaSerial[3] && reaSerial[3] >
reaSerial[4] && reaSerial[4] < reaSerial[5] && reaSerial[5] < reaSerial[6] && reaSerial[6] <
reaSerial[7] && reaSerial[7] > reaSerial[8] && reaSerial[8] < reaSerial[9] && reaSerial[9] > reaSerial[0]
```

- Lần kiểm tra thứ 11 là lấy tổng các giá trị của từng ký tự được XOR so sánh với giá trị (0x354 ^ 0x59) .

0040159E |> \83F6 59 XOR ESI,59 ; <== Temp = Temp xor 0x59

004015A1 |. 81FE 54030000 CMP ESI,354 ; <== if (Temp == 0x354)

004015A7 |. 74 05 JE SHORT crackme3.004015AE ; <== Congratulation !

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                |            |    |            |
|---------------------|----------------|------------|----|------------|
| User : REA-cRaCkErS | Serial : N / A |            |    |            |
| User : VHT-cRaCkErS | Serial : N / A |            |    |            |
| Serial : BuO~/18M&~ | or             | 6U(oIKPs@T | or | A^+D/<U{F~ |

### III – KeyGen :

/Note /- Không có cách nào khác hơn là dùng BruteForce để tìm chuỗi thoả yêu cầu . Tuy nhiên, để thoả mãn hai điều kiện trên, với chuỗi gồm 10 ký tự là một điều hầu như là bất khả thi với các cách giải thông thường .

/Terrible Note/- KeyGen chỉ đúng trong các trường hợp sau :

1- Chuỗi nhập được bắt đầu bằng một ký tự khác số (0-9) hoặc chuỗi ký tự .

2- Không nhập gì cả.

### IV – SourceCode ( VC++ ) :

```
char reaSerial[64]="";
char reaRandomString[256]="";
int i=0;
int TempValue=0,Result=0;
srand((unsigned)time(NULL));
i=0;
while (i < 96)
{
 reaRandomString[i] = (char) (32+i);
 i++;
}
```

```

Result=0;
while (Result == 0)
{
 i=0;
 while (i < 10)
 {
 reaSerial[i] = reaRandomString[rand() % 96];
 reaSerial[i+1] = NULL;
 i++;
 }

 if(reaSerial[0] < reaSerial[1] && reaSerial[1] > reaSerial[2] && reaSerial[2] <
 reaSerial[3] && reaSerial[3] > reaSerial[4] && reaSerial[4] < reaSerial[5] && reaSerial[5] < reaSerial[6]
 && reaSerial[6] < reaSerial[7] && reaSerial[7] > reaSerial[8] && reaSerial[8] < reaSerial[9] &&
 reaSerial[9] > reaSerial[0])
 {
 i=0;
 TempValue=0;
 while (i < 10)
 {
 TempValue = TempValue + reaSerial[i];
 i++;
 }
 if (TempValue == (0x354 ^ 0x59))
 {
 Result = 1;
 SetDlgItemText(IDC_Serial,reaSerial);
 }
 else
 {
 Result = 0;
 }
 }
 else
 {
 Result = 0;
 }
}

```

**V – End of Tut :**

- Finished – ***17/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                           |
|---------------|-----------------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a>       |
| CrackMe :     | \$KORBUT - Rif Crackme #4 - Level 3                       |
| Coder :       | \$KORBUT                                                  |
| Cracker :     | Moonbaby                                                  |
| Type :        | Nag – Name / Serial                                       |
| Packed :      | N/A                                                       |
| Language :    | MASM32 / TASM32                                           |
| Crack Tool :  | OllyDbg 1.09d, PeID 0.92                                  |
| Unpack Tool : | N/A                                                       |
| Request :     | Remove nag - Correct Serial / KeyGen for Original CrackMe |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Hey ! Ya crack'd me !!!" ở địa chỉ :  
00401630 . 68 70314000 PUSH rifme4-n.00403170 ; |Text = "Hey ! Ya crack'd me !!!"
- Truy ngược lên trên ta thấy hàm **GetDlgItemTextA** , vì vậy ta đặt BreakPoint tại hàm này :  
0040156A . E8 27010000 CALL <JMP.&USER32.GetDlgItemTextA> ; |GetDlgItemTextA

### II – Cracking :

- Để không xuất hiện NAG khi Crack, ta tiến hành loại bỏ NAG . Tìm thông báo NAG ở địa chỉ : 004015A6 ]. 68 A7314000 PUSH rifme4.004031A7 ; |Text = "I'm the worst nag you never seen Remove me and code the keygen See the rewls before patching !!!"

- Truy ngược lên chút ta có được toàn bộ đoạn CODE này như sau :

```

0040158C]. E8 05010000 CALL <JMP.&USER32.GetDlgItemTextA> ; |GetDlgItemTextA
00401591]. 0BC0 OR EAX,EAX ; <== Serial atleast 1 chart
00401593 75 0A JNZ SHORT rifme4.0040159F ; <== Change this JNZ to jump out of NAG
00401595]. 68 52314000 PUSH rifme4.00403152 ; ASCII "Ye must enter a serial ! Baka"
0040159A]. E9 A2000000 JMP rifme4.00401641
0040159F > 6A 10 PUSH 10 ; /Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
004015A1]. 68 0A324000 PUSH rifme4.0040320A ; |Title = "Nag, nag, nag, nag, nag..."
004015A6]. 68 A7314000 PUSH rifme4.004031A7 ; |Text = "I'm the worst nag you never seen Remove me and code the keygen See the rewls before patching !!!"
004015AB]. FF75 08 PUSH [ARG.1] ; |hOwner
004015AE]. E8 07010000 CALL <JMP.&USER32.MessageBoxA> ; |MessageBoxA
004015B3]. B9 59000000 MOV ECX,59

```

- Đơn giản là ta cần chuyển lệnh nhảy sau khi đã kiểm tra chiều dài chuỗi Serial nhập qua đoạn xuất hiện NAG . Tức là chuyển  
00401593 75 0A JNZ SHORT rifme4.0040159F ; <== Change this JNZ to jump out of NAG **thành**

00401593 /75 1E JNZ SHORT rifme4.004015B3 ; <== Change this JNZ to jump out of NAG  
 - Trong Olly ta Click chuột phải chọn **Copy to Executable** → **All modification** → **Copy All**. Click chuột phải vào màn hình mới xuất hiện chọn **Saved File**. Lưu lại file với tên mới. Sau đó sử dụng File mới để tiến hành Cracking.

- Quá trình kiểm tra chiều dài chuỗi U và S nhập :

```
0040156A . E8 27010000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA
0040156F . 0BC0 OR EAX,EAX ; <== Name atleast 1 chart
00401571 . 75 0A JNZ SHORT rifme4-n.0040157D
00401573 . 68 37314000 PUSH rifme4-n.00403137 ; ASCII "Ye must enter a name ! Aho"
.....
0040158C . E8 05010000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA
00401591 . 0BC0 OR EAX,EAX ; <== Serial atleast 1 chart
00401593 . 75 1E JNZ SHORT rifme4-n.004015B3
00401595 . 68 52314000 PUSH rifme4-n.00403152 ; ASCII "Ye must enter a serial ! Baka"
```

- Hai đoạn CODE :

```
004015BC > 0FBE81 D23240>MOVsx EAX,BYTE PTR DS:[ECX+4032D2]
004015C3 . |03D8 ADD EBX,EAX
004015C5 . |33D9 XOR EBX,ECX
004015C7 . |49 DEC ECX
004015C8 > |0BC9 OR ECX,ECX
004015CA . ^\75 F0 JNZ SHORT rifme4-n.004015BC
004015CC . EB 0C JMP SHORT rifme4-n.004015DA
004015CE > 0FBE81 D23240>MOVsx EAX,BYTE PTR DS:[ECX+4032D2]
004015D5 . 33D8 XOR EBX,EAX
004015D7 . 33D9 XOR EBX,ECX
004015D9 . 41 INC ECX
004015DA > 83F9 59 CMP ECX,59
004015DD . ^ 75 EF JNZ SHORT rifme4-n.004015CE
```

- Đây là một quá trình tạo ra số mặc định. Số này là **0xFFFFF768**. Quá trình mã hoá chuỗi U nhập thực nằm ở đoạn cuối cùng :

```
004015E3 > 0FBE81 3C3240>MOVsx EAX,BYTE PTR DS:[ECX+40323C] ; <== U[i]
004015EA . |2BD8 SUB EBX,EAX ; <== Temp = Temp - U[i]
004015EC . |03D9 ADD EBX,ECX ; <== Temp = Temp + i
004015EE . |41 INC ECX ; <== i++
004015EF > |0BC0 OR EAX,EAX ; <== if (i < Len.U + 1)
004015F1 . ^\75 F0 JNZ SHORT rifme4-n.004015E3 ; <== Continue Loop
004015F3 . 81F3 A5BF8E02 XOR EBX,28EBFA5 ; <== Temp = Temp xor 0x28EBFA5
004015F9 . 83C1 0A ADD ECX,0A ; <== TempLen = (Len.U + 1) and 0xA
004015FC . 33CB XOR ECX,EBX ; <== TempLen = TempLen xor Temp
004015FE . 51 PUSH ECX ; /<%IX> <== TempLen
004015FF . 53 PUSH EBX ; |<%lu> <== Temp
00401600 . 68 25324000 PUSH rifme4-n.00403225 ; |Format = "%lu-1789-%1X-R!F"
00401605 . 68 6E324000 PUSH rifme4-n.0040326E ; |s = rifme4-n.0040326E
0040160A . E8 5D000000 CALL <JMP.&USER32.wsprintfA> ; \wsprintfA
```

- Tuy nhiên, bạn hãy đọc kỹ lại yêu cầu của CODER “ **The KeyGen must work with the original** “ . Nghĩa là tạo KeyGen cho CrackMe khi chúng ta chưa PATCH NAG. Thử lại kết quả tìm được với Original CrackMe Kết quả “ **Keep on trying, young rascal !** ” . Đây chính là điểm hay của CrackMe này. Hay nói cách khác đây là một quá trình **ANTI PATCH**.

- Load CrackMe chưa bị PATCH, ta sẽ tìm được giá trị mặc định lúc này **0xFFFFF800** . Thay đổi và

làm lại, kết quả sẽ cho chúng ta "*Hey ! Ya crack'd me !!!*"

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                                       |
|---------------------|---------------------------------------|
| User : REA-cRaCkErS | Serial : 4252060638-1789-FD714BC9-R!F |
| User : VHT-cRaCkErS | Serial : 4252060612-1789-FD714BD3-R!F |

### III – KeyGen :

|           |                    |   |                                   |
|-----------|--------------------|---|-----------------------------------|
| /Note 1/- | Khi chưa PATCH NAG | : | Default Value = <b>0xFFFFF800</b> |
| /Note 2/- | Khi đã PATCH NAG   | : | Default Value = <b>0xFFFFF768</b> |

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
int LenUser=0;
int i=0;
int Temp=0, TempCheck=0;

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 1)
{
 MessageBox("=-*=-*=- Your name atleast 1 chart =*=-*=-","Hey !! Please input
your name again !! ");
}
else
{
 Temp = 0xFFFFF800; // Without Nag 0xFFFFF768
 i=0;
 while (i < (LenUser + 1))
 {
 Temp = (Temp - reaName[i]) + i;
 i++;
 }

 Temp = Temp ^ 0x28EBFA5;
 TempCheck = i + 0xA;
 TempCheck = TempCheck ^ Temp;

 wsprintf(reaserial,"%lu-1789-%IX-R!F",Temp,TempCheck);

 SetDlgItemText(IDC_Serial,reaserial);
}

```

### V – End of Tut :

- Finished – **17/06/2004**

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | \$KORBUT KeyGen Me #1 - Level 1                     |
| Coder :       | \$KORBUT                                            |
| Cracker :     | Moonbaby                                            |
| Type :        | Name / Serial                                       |
| Packed :      | N/A                                                 |
| Language :    | MASM32 / TASM32                                     |
| Crack Tool :  | OllyDbg 1.09d, PeiD 0.92                            |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Serial / KeyGen                             |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Hum, c'est pas bon !" ở địa chỉ :  
004010E5 . 68 70304000 PUSH skor\_SN.00403070 ; |Text = "Hum, c'est pas bon !"  
- Truy ngược lên ta thấy hàm **GetDlgItemTextA** , vì vậy ta đặt BreakPoint tại hàm này :  
00401024 . E8 F1050000 CALL <JMP.&user32.GetDlgItemTextA> ; \GetDlgItemTextA

### II – Cracking :

- Nhấn F9 để chạy CrackMe. Nhập User và Fake Serial vào. CrackMe dừng lại tại điểm đặt BP.  
00401024 . E8 F1050000 CALL <JMP.&user32.GetDlgItemTextA> ; \GetDlgItemTextA  
00401029 . 0BC0 OR EAX,EAX ; <== Name atleast 1 chart  
0040102B . 75 0A JNZ SHORT skor\_SN.00401037 ; <== Else Break
- Quá trình mã hoá chuỗi U diễn ra rất đơn giản :  
0040103D >/8B1D B8304000 MOV EBX,DWORD PTR DS:[4030B8] ; <== Only 4 first charts of U  
00401043 . |0FBE90 4E3140>MOVSX EDX,BYTE PTR DS:[EAX+40314E] ; <== Value at this add.  
0040104A . |03DA ADD EBX,EDX ; <== Temp = DWORD (U) + Value  
0040104C . |C1CB 05 ROR EBX,5 ; <== Temp = Temp ror 0x5  
0040104F . |83F3 20 XOR EBX,20 ; <== Temp = Temp xor 0x20  
00401052 . |8BF3 MOV ESI,EBX ; <== Temp  
00401054 . |0FCE BSWAP ESI ; <== Convert value of Temp  
00401056 . |C1CE EB ROR ESI,0EB ; <== Temp = Temp ror 0x0EB  
00401059 . |40 INC EAX ; <== i++;  
0040105A . |49 DEC ECX ; <== LenU --;  
0040105B .^|75 E0 JNZ SHORT skor\_SN.0040103D ; <== Loop until LenU == 0x0

- Tuy nhiên ở đây có vài điểm cần chú ý :

/Note 1/- Chỉ có 4 ký tự đầu tiên của U tham gia vào quá trình mã hoá (DWORD)

/Note 2/- Địa chỉ DS:[EAX+40314E] không phải lúc nào cũng là NULL. Nếu ta bỏ đi giá trị EAX mà truy ngược về giá trị ở địa chỉ 40314E thì ta biết được địa chỉ này chứa ComputerName . Tuy nhiên do tùy thuộc vào EAX ( chiều dài của U nhập ) mà địa chỉ DS:[EAX+40314E] có thể chỉ đến một giá trị nằm ngoài khoảng chiều dài của ComputerName . Ví dụ, nếu ComputerName có chiều dài là 4 ký tự, LenU có chiều dài là 5 ký tự thì địa chỉ lúc này sẽ chỉ đến giá trị nằm sau chuỗi ComputerName, tức là

giá trị NULL. Nếu LenU có chiều dài là 2 thì giá trị đầu tiên của địa chỉ DS:[EAX+40314E] sẽ là DS:[2 + 40314E] tức là chỉ đến ký tự thứ III của chuỗi ComputerName. Chính vì thế, nếu LenU lớn hơn chiều dài của ComputerName thì chuỗi ComputerName không tham gia vào quá trình mã hoá.

/Note 3/- Lệnh **BSWAP ESI** có tác dụng đảo ngược giá trị. Ví dụ giá trị ban đầu của ESI=ABCDEF thì sau khi thực hiện lệnh **BSWAP ESI** thì giá trị của ESI = EFCDAB

- Quá trình xuất chuỗi được tiến hành theo định dạng ".(%lu)." :

|                                                     |                           |
|-----------------------------------------------------|---------------------------|
| 00401098 . 68 80314000 PUSH skor_SN.00403180        | ;  Format = ".(%lu)." ;   |
| 0040109D . 68 EA304000 PUSH skor_SN.004030EA        | ;  s = skor_SN.004030EA ; |
| 004010A2 . E8 55050000 CALL <JMP.&user32.wsprintfA> | \wsprintfA                |

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                         |                     |
|---------------------|-------------------------|---------------------|
| User : REA-cRaCkErS | Serial : .(1377911117). | ComputerName (Long) |
| User : VHT-cRaCkErS | Serial : .(1445745741). | ComputerName (Long) |

### III – KeyGen :

/Section 1/- Xác định ComputerName và chuyển sang dạng UpperCase

/Section 2/- Kiểm tra chiều dài của Len.U và Len. ComputerName để lựa chọn các tính thiirc hợp.

/Section 2/- Chuyển giá trị tính toán theo định dạng ".(%lu)."

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
char reaCompName[255]="";
int LenUser=0;
int i=0,Temp=0, TempSerial;
unsigned long reaSize=255;

LenUser=GetDlgItemText(IDC_Name,reaName,64);

if (LenUser < 1)
{
 MessageBox("=-*=-*=- Your name atleast 1 chart =*=-*=-","Hey !! Please input
your name again !! ");
}
else
{
 GetComputerName(reaCompName,&reaSize);
 CharUpper(reaCompName);
 _asm
 {
 XOR ESI,ESI
 XOR EBX,EBX
 MOV ECX,EAX
 }
 if (LenUser < lstrlen(reaCompName))
 {
 i=0;
 while (i < LenUser)
 {

```

```

 _asm
 {
 MOV EBX,DWORD PTR reaName
 MOV Temp,EBX
 }

 Temp = Temp + reaCompName[LenUser + i];
 _asm
 {
 MOV EBX,Temp
 ROR EBX,0x5
 XOR EBX,0x20
 MOV ESI,EBX
 BSWAP ESI
 ROR ESI,0x0EB
 MOV TempSerial,ESI
 }
 i++;
}
else
{
 i=0;
 while (i < LenUser)
 {
 _asm
 {
 MOV EBX,DWORD PTR reaName
 ROR EBX,0x5
 XOR EBX,0x20
 MOV ESI,EBX
 BSWAP ESI
 ROR ESI,0x0EB
 MOV TempSerial,ESI
 }
 i++;
 }
}

wsprintf(reaSerial,".(%lu).",TempSerial);
SetDlgItemText(IDC_Serial,reaSerial);
}

```

**V – End of Tut :**

- Finished – ***16/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | \$KORBUT Keygenme #1 - Level 1                      |
| Coder :       | \$KORBUT                                            |
| Cracker :     | Moonbaby                                            |
| Type :        | Name / Serial                                       |
| Packed :      | N/A                                                 |
| Language :    | Nothing found *                                     |
| Crack Tool :  | OllyDbg 1.09d, PeID 0.92                            |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Serial / KeyGen                             |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK nhưng không cho ta biết CrackMe được được viết bằng ngôn ngữ gì **Nothing found \***
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Non, c'est pas bon, allez on recommence" ở địa chỉ :  
004010CB |. 68 3D304000 PUSH keyme1.0040303D ; |Text = "Non, c'est pas bon, allez on recommence"
- Truy ngược lên trên ta thấy hàm **GetDlgItemTextA** , vì vậy ta đặt BreakPoint tại hàm này :  
00401068 |. E8 93000000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA

### II – Cracking :

- Nhấn F9 để chạy CrackMe. Nhập User và Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. CrackMe chỉ yêu cầu tối thiểu một ký tự cho mỗi oo nhập :
- ```
00401068 |. E8 93000000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA
0040106D |. 83F8 00     CMP EAX,0          ; <== Name atleast 1 chart
00401070 |. 74 6A     JE SHORT keyme1.004010DC
```
-

- ```
00401081 |. E8 7A000000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA
00401086 |. 83F8 00 CMP EAX,0 ; <== Serial atleast 1 chart
00401089 |. 74 51 JE SHORT keyme1.004010DC
```

- Quá trình mã hoá chuỗi U diễn ra rất đơn giản :

- ```
00401094 |>/8A99 AC304000 /MOV BL,BYTE PTR DS:[ECX+4030AC]      ; <== U[i]
0040109A |. 8A91 DE304000 |MOV DL,BYTE PTR DS:[ECX+4030DE]      ; <== S[i]
004010A0 |. |FEC3     |INC BL           ; <== U[i] + 1
004010A2 |. |38DA     |CMP DL,BL        ; <== cmp (U[i] + 1) & S[i]
004010A4 |. |75 1E     |JNZ SHORT keyme1.004010C4      ; <== if Not Equal jmp Nag
004010A6 |. |48       |DEC EAX         ; <== LenU --
004010A7 |. |74 03     |JE SHORT keyme1.004010AC      ; <== Out loop if LenU == 0
004010A9 |. |41       |INC ECX         ; <== i++
004010AA |.^|EB E8     |JMP SHORT keyme1.00401094      ; <== Continue Loop
```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS
User : VHT-cRaCkErS

Serial : SFB.dSbDlFsT
Serial : WIU.dSbDlFsT

III – KeyGen :

/Section 1/- $S[i] = U[i] + 1$

IV – SourceCode (VC++) :

```
char reaName[64]="";
char reaSerial[64]="";
int LenUser=0;
int i=0;
```

```
LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 1 )
{
    MessageBox("==*==*== Your name atleast 1 chart ==*==*==","Hey !! Please input
your name again !! ");
}
else
{
    i=0;
    while ( i < LenUser)
    {
        reaSerial[i] = reaName[i] + 1;
        i++;
    }
}

SetDlgItemText(IDC_Serial,reaSerial);
}
```

V – End of Tut :

- Finished – ***16/06/2004***

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	[Flasher] - Delphi CrackME - Level 3
Coder :	[Flasher]
Cracker :	Moonbaby
Type :	Serial
Packed :	N/A
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PeID 0.92
Unpack Tool :	N/A
Request :	Correct Serial

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **Borland Delphi 4.0 - 5.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo "Incorrect..." và tìm được ở địa chỉ :
00444554 . BA 58464400 MOV EDX,Crackme_.00444658 ; ASCII "Incorrect..."
- Truy ngược lên trên và đặt BreakPoint tại :
00444480 . E8 9FE3FBFF CALL Crackme_.00402824 ; <== Set BreakPoint here

II – Cracking :

- CrackMe này chỉ yêu cầu tìm đúng Serial . Mặc dù đoạn Code thực hiện để có được Serial thực rất dài dòng, tuy nhiên CODER lại cho ta đoán so sánh ở đây :

00444526 . 8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18]	; <== Fake Serial
00444529 . 8B55 F0 MOV EDX,DWORD PTR SS:[EBP-10]	; <== Real Serial
0044452C . E8 6BF7FBFF CALL Crackme_.00403C9C	; <== Check Serial
00444531 . 75 1B JNZ SHORT Crackme_.0044454E	; <== If Corect REGISTRED
- Quả thực là quá đơn giản và không đúng với LEVEL mà CODER đã đề cập, cũng như trong phần giới thiệu “ it was easy to made but i'm not sure that it will be so easy to crack ” . Tôi đã đọc lại toàn bộ các hàm API mà CrackMe sử dụng nhưng không hề thấy hàm tạo chuỗi ngẫu nhiên hay sử dụng thời gian hệ thống hay bắt cứ gì có thể làm chuỗi Serial sinh ra ngẫu nhiên .
- Thủ lại với nhiều máy tính khác nhau cũng cho ra một số Serial tương tự . Có lẽ cái khó ở đây là CODER muốn chúng ta nghiên cứu cách thức tạo Serial .

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : N / A
User : VHT-cRaCkErS	Serial : N / A
Serial : 66511818-630035424	

III – KeyGen :

N / A

IV – SourceCode (VC++) :

N / A

V – End of Tut :

- Finished – **18/06/2004**

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	[v0!d] - crackme01 - Level 1
Coder :	[v0!d]
Cracker :	Moonbaby
Type :	Name / Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PeiD 0.92
Unpack Tool :	N/A
Request :	Correct Serial / KeyGen

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng Microsoft Visual C++ 6.0
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "GOOD JOB! - CRACKED!" ở địa chỉ : 00401271 . 68 80604000 PUSH vcrkme01.00406080 ; |Title = "GOOD JOB! - CRACKED!"
- Truy ngược lên trên và nhận thấy CrackMe sử dụng hàm **GetDlgItemTextA** nên ta đặt BreakPoint tại : 00401233 . FFD6 CALL ESI ; |GetDlgItemTextA

II – Cracking :

- Quan sát đoạn CODE chính của CrackMe ta tìm được đoạn CODE cần thiết chuyển đến quá trình hoá chuỗi U nhập :

```
00401257 . E8 A4FDFFFF CALL vcrkme01.00401000      ; <== Get & check serial
0040125C . 83C4 08 ADD ESP,8                         ; <== EAX : return checking value
0040125F . 83F8 01 CMP EAX,1                          ; <== EAX = 1 : Correct
00401262 . A3 646C4000 MOV DWORD PTR DS:[406C64],EAX ; <== then
00401267 . 75 65 JNZ SHORT vcrkme01.004012CE       ; <== CRACKED !!
```

- Dùng F7 trace into lệnh CALL này ta đến quá trình mã hoá chuỗi U thành Serial và kiểm tra chuỗi Serial. CrackMe nà chia làm nhiều bước kiểm tra . Trước hết kiểm tra ký tự đầu tiên của chuỗi Serial thực phải là ký tự đầu tiên của chuỗi U nhập :

```
00401010 |. 8A06      MOV AL,BYTE PTR DS:[ESI]          ; <== U[0]
00401012 |. 3AC1      CMP AL,CL                        ; <== U[0] == S[0]
00401014 |. 0F85 69010000 JNZ vcrkme01.00401183       ; <== If not END
```

- Chiều dài của User tối thiểu là 5 ký tự :

```
00401026 |. 83F9 05  CMP ECX,5                         ; <== Length U atleast 5 charts
00401029 |. 0F82 54010000 JB vcrkme01.00401183
```

- Ký tự thứ hai phải là “-“ (0x2D) :

```
0040102F |. 807B 01 2D  CMP BYTE PTR DS:[EBX+1],2D    ; <== If (S[1] == "-")
00401033 |. 0F85 4A010000 JNZ vcrkme01.00401183       ; <== Continue
```

- Sau đó là đến quá trình mã hoá vào tạo đoạn Serial thứ nhất :

```

00401049 > /0FBE0C32    /MOVSX ECX,BYTE PTR DS:[EDX+ESI] ; <== U[i]
0040104D |. |03E9      |ADD EBP,ECX                      ; <== Temp = Temp + U[i]
0040104F |. |8BFE      |MOV EDI,ESI                      ; <== [ from START ]
00401051 |. |83C9 FF  |OR ECX,FFFFFFFF                ; <== [ ]
00401054 |. |33C0      |XOR EAX,EAX                      ; <== [ ]
00401056 |. |42        |INC EDX                         ; <== [ Only get length of U ]
00401057 |. |F2:AE     |REPNE SCAS BYTE PTR ES:[EDI]       ; <== [ ]
00401059 |. |F7D1      |NOT ECX                         ; <== [ ]
0040105B |. |49        |DEC ECX                          ; <== [ to END ]
0040105C |. |3BD1      |CMP EDX,ECX                      ; <== while ( i < Len.U )
0040105E |.^|72 E9    |JB SHORT vcrkme01.00401049      ; <== Continue Loop
00401060 > 81C5 64600000 ADD EBP,6064                  ; <== Temp = Temp + 0x6064
00401066 |. 55        PUSH EBP                        ; <== Temp
00401067 |. 68 34604000 PUSH vcrkme01.00406034      ; ASCII "%lu"
0040106C |. 68 306B4000 PUSH vcrkme01.00406B30      ; <== Real S[2]
00401071 |. E8 B6030000 CALL vcrkme01.0040142C      ; <== Get DEC number

```

- Ký tự cuối cùng của chuỗi U sẽ được chuyển sang dạng UpperCase và được xem là ký tự thứ ba của chuỗi Serial thực :

```

00401091 |. OFBE4431 FF  MOVSX EAX,BYTE PTR DS:[ECX+ESI-1]      ; <== U[Len.U]
00401096 |. 50          PUSH EAX
00401097 |. E8 C4020000 CALL vcrkme01.00401360                  ; <== Convert to Upper Case
0040109C |. A2 466B4000 MOV BYTE PTR DS:[406B46],AL            ; <== Real S[2]

```

- Đến giai đoạn này chuỗi Serial thực có dạng “A-Bxxxx” . Giá trị Temp ở trên được cộng thêm với giá trị mặc định và cũng được chuyển về dạng DEC cho ta đoạn mã hoá thứ hai của chuỗi Serial thực :

```
004010B1 |. 81C5 64600000 ADD EBP,6064                  ; <== Temp_01 = TEmp_01 + 0x6064
```

- Chuỗi mới này được gắn vào chuỗi ở trên :

```
004010D3 |. 68 34604000 PUSH vcrkme01.00406034      ; ASCII "%lu"
```

```
.....
```

```
00401101 |. F3:A5      REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS>      ; <== "-"
```

- Chuỗi Serial thực sẽ có dạng “A-Bxxxx-yyyy”

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : R-S25655-50331
User : VHT-cRaCkErS	Serial : V-S25681-50357

III – KeyGen :

```

/Section 1/-  S[0] == U[0]  S[1] == "-"  S[2] == CharUpper(U[Len.U])
/Section 2/-  Temp_01 = (Temp_01 + U[i]) + 0x6064
/Section 3/-  Temp_02 = Temp_01 + 0x6064

```

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};
char reaTemp[64]={0};
int LenUser=0;
int i=0;
int Temp=0;

```

```

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 5 )
{
}
MessageBox("Your name atleast 5 charts ","Hey !! Please input your name again !! ");
}
else
{
    reaSerial[0] = reaName[0];
    reaSerial[1] = 0x2D;
    i=0;
    Temp=0;
    while ( i < LenUser)
    {
        Temp = Temp + reaName[i];
        i++;
    }
    Temp = Temp + 0x6064;

    wsprintf(reTemp,"%i-%i",Temp,(Temp+0x6064));
    CharUpper(reName);
    reaSerial[2] = reaName[LenUser-1];
    lstrcat(reSerial,reTemp);
    SetDlgItemText(IDC_Serial,reSerial);
}

```

V – End of Tut :

- Finished – ***21/06/2004***

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	[v0!d] - crackme02 - Level 0
Coder :	[v0!d]
Cracker :	Moonbaby
Type :	Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PeiD 0.92
Unpack Tool :	N/A
Request :	Correct Serial

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng Microsoft Visual C++ 6.0
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "GOOD JOB! - CRACKED!" ở địa chỉ :

0040118D . 68 60514000 PUSH vcrkme02.00405160 ; |Title = "GOOD JOB! - CRACKED!"
 - Truy ngược lên trên và nhận thấy CrackMe sử dụng hàm **GetDlgItemTextA** nên ta đặt BreakPoint tại :
 00401169 . FF15 A4404000 CALL DWORD PTR DS:[<&USER32.GetDlgItemTe>];
 \GetDlgItemTextA

II – Cracking :

- Quan sát đoạn CODE chính của CrackMe ta tìm được đoạn CODE cần thiết chuyển đến quá trình hoá chuỗi S nhập :

00401174 . E8 87FFFF CALL vcrkme02.00401000 ; <== Get & Compare Serial
 00401179 . 83C4 04 ADD ESP,4

0040117C . A3 A0564000 MOV DWORD PTR DS:[4056A0],EAX ; <== Get value of EAX

00401181 . 85C0 TEST EAX,EAX ; <== if (EAX !=0)

00401183 . 74 37 JE SHORT vcrkme02.004011BC ; <== Congratulation !!

- Dùng F7 trace vào lệnh CALL này ta đến quá trình mã hoá và kiểm tra chuỗi Serial . Đầu tiên, CrackMe kiểm tra chiều dài chuỗi Serial :

00401014 |. 83F9 10 CMP ECX,10 ; <== S must be 16 charts

00401017 |. 0F85 93000000 JNZ vcrkme02.004010B0

- Sau đó, CrackMe kiểm tra số lượng “Số” (0-9) có trong chuỗi Serial :

0040101D |. 8A5D 00 MOV BL,BYTE PTR SS:[EBP] ; <== S[0]

00401020 |. 33F6 XOR ESI,ESI

00401022 |. B2 30 MOV DL,30

00401024 > 84DB /TEST BL,BL ; <== [from START]

00401026 |. 74 16 JE SHORT vcrkme02.0040103E ; <== []

00401028 |. 8AC3 |MOV AL,BL ; <== []

0040102A |. 8BCD |MOV ECX,EBP ; <== [This section mean]

0040102C > 3AD0 |/CMP DL,AL ; <== [The Serial can only have]

0040102E |. 75 01 ||JNZ SHORT vcrkme02.00401031 ; <== [Maximum 6 numbers]

00401030 |. 46 ||INC ESI ; <== []

00401031 > 83FE 06 ||CMP ESI,6 ; <== []

00401034 |. 7D 7A ||JGE SHORT vcrkme02.004010B0 ; <== []

00401036 |. 8A41 01 ||MOV AL,BYTE PTR DS:[ECX+1] ; <== []

00401039 |. 41 ||INC ECX ; <== []

0040103A |. 84C0 ||TEST AL,AL ; <== []

0040103C |.^ 75 EE ||JNZ SHORT vcrkme02.0040102C ; <== []

0040103E > 33F6 |XOR ESI,ESI ; <== []

00401040 |. FEC2 |INC DL ; <== []

00401042 |. 80FA 39 |CMP DL,39 ; <== []

00401045 |.^ 7E DD |JLE SHORT vcrkme02.00401024 ; <== [to END]

- Sau quá trình này, CrackMe bắt đầu vào quá trình kiểm tra chính. Quá trình này được chia làm ba giai đoạn và được diễn giải :

00401047 |. 0FBE45 03 MOVSX EAX,BYTE PTR SS:[EBP+3] ; <== S[3]

0040104B |. 0FBE4D 02 MOVSX ECX,BYTE PTR SS:[EBP+2] ; <== S[2]

0040104F |. 0FBE55 01 MOVSX EDX,BYTE PTR SS:[EBP+1] ; <== S[1]

00401053 |. 03C1 ADD EAX,ECX ; <== Temp = S[3] + S[2]

00401055 |. 0FBECB MOVsx ECX,BL ; <== S[0]

00401058 |. 03C2 ADD EAX,EDX ; <== Temp = Temp + S[1]

0040105A |. 8D9408 40FFFF>LEA EDX,DWORD PTR DS:[EAX+ECX-C0] ; <== Temp = Temp + S[0]

00401061 |. 83FA 16 CMP EDX,16 ; <== if (Temp == 0x16)

00401064 75 4A JNZ SHORT vcrkme02.004010B0 ; <== Continue check

00401066 |. 0FBE45 0D MOVSX EAX,BYTE PTR SS:[EBP+D] ; <== S[13]

```

0040106A |. 0FBE4D 0A    MOVSX ECX,BYTE PTR SS:[EBP+A] ; <== S[10]
0040106E |. 0FBE55 07    MOVSX EDX,BYTE PTR SS:[EBP+7] ; <== S[7]
00401072 |. 03C1        ADD EAX,ECX ; <== Temp = S[13] + S[10]
00401074 |. 0FBE4D 04    MOVSX ECX,BYTE PTR SS:[EBP+4] ; <== S[4]
00401078 |. 03C2        ADD EAX,EDX ; <== Temp = Temp + S[7]
0040107A |. 8D9408 40FFFF>LEA EDX,DWORD PTR DS:[EAX+ECX-C0] ; <== Temp = Temp +
S[4]
00401081 |. 83FA 1E      CMP EDX,1E ; <== if (Temp == 0x1E)
00401084 75 2A        JNZ SHORT vcrkme02.004010B0 ; <== Continue check
00401086 |. 0FBE45 0F    MOVSX EAX,BYTE PTR SS:[EBP+F] ; <== S[15]
0040108A |. 0FBE4D 0C    MOVSX ECX,BYTE PTR SS:[EBP+C] ; <== S[12]
0040108E |. 0FBE55 09    MOVSX EDX,BYTE PTR SS:[EBP+9] ; <== S[9]
00401092 |. 03C1        ADD EAX,ECX ; <== Temp = S[15] + S[12]
00401094 |. 0FBE4D 06    MOVSX ECX,BYTE PTR SS:[EBP+6] ; <== S[6]
00401098 |. 03C2        ADD EAX,EDX ; <== Temp = Temp + S[9]
0040109A |. 8D9408 40FFFF>LEA EDX,DWORD PTR DS:[EAX+ECX-C0] ; <== Temp=Temp + S[6]
004010A1 |. 83FA 09      CMP EDX,9 ; <== if (Temp == 0x9)
004010A4 75 0A        JNZ SHORT vcrkme02.004010B0 ; <== Congratulation !

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS
User : VHT-cRaCkErS

Serial : N / A
Serial : N / A

III – KeyGen :

N / A

IV – SourceCode (VC++) :

```
char reaSerial[64]={0};
```

```

char reaRandomString[256]={0};
int i=0;
int Temp_01=0,Temp_02=0,Temp_03=0;
int Result=0;
    srand( (unsigned)time( NULL ) );
    i=0;
    while ( i < 20 )
    {
        reaRandomString[i] = (char) (i);
        i++;
    }
    while (lstrlen(reaSerial) != 16)
    {
        i=0;
        while ( i < 16 )
        {
            reaSerial[i] = reaRandomString[rand() % 29];
            i++;
        }
    }
    Result=0;
    while (Result == 0)

```

```

{
    i=0;
    Temp_01=0;
    while ( i < 4)
    {
        Temp_01 = Temp_01 + reaSerial[i];
        i++;
    }
    Temp_02=0;
    Temp_02 = reaSerial[4] + reaSerial[7] + reaSerial[0xA] + reaSerial[0xD];
    Temp_03=0;
    Temp_03 = reaSerial[6] + reaSerial[9] + reaSerial[0xC] + reaSerial[0xF];

    if (Temp_01 == 0x9 && Temp_02 == 0x1E && Temp_03 == 0x9)
    {
        Result++;
    }
    else
    {
        Result=0;
    }
}
SetDlgItemText(IDC_Serial,reaSerial);

```

V – End of Tut :

- Finished – ***21/06/2004***

Reverse Engineering Association

CrackMe

Homepage	:	http://crackmes.de
CrackMe	:	abex 2 - Level 1
Coder	:	abex
Cracker	:	Moonbaby
Type	:	Name / Serial
Packed	:	N/A
Language	:	Microsoft Visual Basic 5.0 / 6.0
Crack Tool	:	OllyDbg 1.09d, PeID 0.92
Unpack Tool	:	N/A
Request	:	Correct Serial / KeyGen

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **Microsoft Visual Basic 5.0 / 6.0**

- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "Nope, this serial is wrong!" ở địa chỉ :

00403476 . C785 2CFFFFFF>MOV DWORD PTR SS:[EBP-D4],abexcrac.00402>; UNICODE
"Nope, this serial is wrong!"

- Truy ngược lên trên và đặt BreakPoint tại :

00402F78 . FF92 08030000 CALL DWORD PTR DS:[EDX+308] ; <== Set BreakPoint here

II – Cracking :

- Chiều dài chuỗi U nhập tối thiểu là 4 ký tự :

0040307E . C785 2CFFFFFF>MOV DWORD PTR SS:[EBP-D4],abexcrac.00402>; UNICODE
"Please enter at least 4 chars as name!"

- Quá trình mã hoá chuỗi U được diễn ra như sau :

```
00403197 >/85C0      TEST EAX,EAX
00403199 . |OF84 06010000 JE abexcrac.004032A5
0040319F . |8D95 64FFFFFF LEA EDX,DWORD PTR SS:[EBP-9C]
004031A5 . |8D45 DC    LEA EAX,DWORD PTR SS:[EBP-24]
004031A8 . |52        PUSH EDX
004031A9 . |50        PUSH EAX
004031AA . |C785 6CFFFFFF>MOV DWORD PTR SS:[EBP-94],1
004031B4 . |89BD 64FFFFFF MOV DWORD PTR SS:[EBP-9C],EDI
004031BA . |FF15 A8104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaI4Var>;  
MSVBVM60.__vbaI4Var
004031C0 . |8D4D 8C    LEA ECX,DWORD PTR SS:[EBP-74]
004031C3 . |50        PUSH EAX
004031C4 . |8D95 54FFFFFF LEA EDX,DWORD PTR SS:[EBP-AC]
004031CA . |51        PUSH ECX
004031CB . |52        PUSH EDX
004031CC . |FFD3      CALL EBX
004031CE . |8D95 54FFFFFF LEA EDX,DWORD PTR SS:[EBP-AC]
004031D4 . |8D4D AC    LEA ECX,DWORD PTR SS:[EBP-54]
004031D7 . |FFD6      CALL ESI
004031D9 . |8D8D 64FFFFFF LEA ECX,DWORD PTR SS:[EBP-9C]
004031DF . |FF15 0C104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeV>;  
MSVBVM60.__vbaFreeVar
004031E5 . |8D45 AC    LEA EAX,DWORD PTR SS:[EBP-54]
004031E8 . |8D8D 78FFFFFF LEA ECX,DWORD PTR SS:[EBP-88]
004031EE . |50        PUSH EAX
004031EF . |51        PUSH ECX
004031F0 . |FF15 80104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrVa>;  
MSVBVM60.__vbaStrVarVal
004031F6 . |50        PUSH EAX
004031F7 . |FF15 1C104000 CALL DWORD PTR DS:[<&MSVBVM60.#516>] ;  
MSVBVM60.rtcAnsiValueBstr
004031FD . |8D95 24FFFFFF LEA EDX,DWORD PTR SS:[EBP-DC]
00403203 . |8D4D AC    LEA ECX,DWORD PTR SS:[EBP-54]
00403206 . |66:8985 2CFFF>MOV WORD PTR SS:[EBP-D4],AX
0040320D . |89BD 24FFFFFF MOV DWORD PTR SS:[EBP-DC],EDI
00403213 . |FFD6      CALL ESI
00403215 . |8D8D 78FFFFFF LEA ECX,DWORD PTR SS:[EBP-88]
0040321B . |FF15 CC104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeS>;  
MSVBVM60.__vbaFreeStr
```

```

00403221 . |8D55 AC    LEA EDX,DWORD PTR SS:[EBP-54]
00403224 . |8D85 24FFFFFF LEA EAX,DWORD PTR SS:[EBP-DC]
0040322A . |52      PUSH EDX
0040322B . |8D8D 64FFFFFF LEA ECX,DWORD PTR SS:[EBP-9C]
00403231 . |50      PUSH EAX
00403232 . |51      PUSH ECX
00403233 . |C785 2CFFFFFF>MOV DWORD PTR SS:[EBP-D4],64      ; <== S[i] = U[i] + 0x64
0040323D . |89BD 24FFFFFF MOV DWORD PTR SS:[EBP-DC],EDI
00403243 . |FF15 AC104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarAd>;
MSVBVM60.__vbaVarAdd
00403249 . |8BD0      MOV EDX,EAX
0040324B . |8D4D AC    LEA ECX,DWORD PTR SS:[EBP-54]
0040324E . |FFD6      CALL ESI
00403250 . |8D55 AC    LEA EDX,DWORD PTR SS:[EBP-54]
00403253 . |8D85 64FFFFFF LEA EAX,DWORD PTR SS:[EBP-9C]
00403259 . |52      PUSH EDX
0040325A . |50      PUSH EAX
0040325B . |FF15 94104000 CALL DWORD PTR DS:[<&MSVBVM60.#573>] ; 
MSVBVM60.rtcHexVarFromVar
00403261 . |8D95 64FFFFFF LEA EDX,DWORD PTR SS:[EBP-9C]
00403267 . |8D4D AC    LEA ECX,DWORD PTR SS:[EBP-54]
0040326A . |FFD6      CALL ESI
0040326C . |8D4D BC    LEA ECX,DWORD PTR SS:[EBP-44]
0040326F . |8D55 AC    LEA EDX,DWORD PTR SS:[EBP-54]
00403272 . |51      PUSH ECX
00403273 . |8D85 64FFFFFF LEA EAX,DWORD PTR SS:[EBP-9C]
00403279 . |52      PUSH EDX
0040327A . |50      PUSH EAX
0040327B . |FF15 84104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarCa>;
MSVBVM60.__vbaVarCat
00403281 . |8BD0      MOV EDX,EAX
00403283 . |8D4D BC    LEA ECX,DWORD PTR SS:[EBP-44]
00403286 . |FFD6      CALL ESI
00403288 . |8D8D BCFFFFF LEA ECX,DWORD PTR SS:[EBP-144]
0040328E . |8D95 CCFFFFF LEA EDX,DWORD PTR SS:[EBP-134]
00403294 . |51      PUSH ECX
00403295 . |8D45 DC    LEA EAX,DWORD PTR SS:[EBP-24]
00403298 . |52      PUSH EDX
00403299 . |50      PUSH EAX
0040329A . |FF15 C0104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarFo>;
MSVBVM60.__vbaVarForNext

```

- Vòng lặp này chỉ diễn ra với chiều dài của chuỗi Serial tạo thành nhỏ hơn hay bằng 8 ký tự. Nói cách khác, chiều dài của S là 8 ký tự.

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : B6A9A591
User : VHT-cRaCkErS	Serial : BAACB891

III – KeyGen :

/Section /- S[i] = U[i] + 0x64

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};
char reaTemp[64]={0};
int LenUser=0;
int i=0, Temp=0;
LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 4 )
{
    MessageBox("----- Your name atleast 4 charts ----- ","Hey !!
Please input your name again !! ");
}
else
{
    i=0;          Temp = 0;
    while ( strlen(reaSerial) < 8)
    {
        Temp = reaName[i] + 0x64;
        wsprintf(reaTemp,"%X",Temp);
        lstrcat(reaSerial,reaTemp);
        i++;
    }
    SetDlgItemText(IDC_Serial,reaSerial);
}

```

V – End of Tut :

- Finished – ***28/06/2004***

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	abex 3 - Level 1
Coder :	abex
Cracker :	Moonbaby
Type :	KeyFile
Packed :	N/A
Language :	MASM32 / TASM32
Crack Tool :	OllyDbg 1.09d, PeiD 0.92
Unpack Tool :	N/A
Request :	Correct KeyFile

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32**

- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "Hmmmmm, I can't find the file!" ở địa chỉ :

0040107C |. 68 5A204000 PUSH abexcrac.0040205A ;||Text = "Hmmmmm, I can't find the file!"

- Truy ngược lên trên và ta thấy đoạn CODE :

```

00401013 |. 6A 00      PUSH 0          ;|hTemplateFile = NULL
00401015 |. 68 80000000  PUSH 80     ;|Attributes = NORMAL
0040101A |. 6A 03      PUSH 3          ;|Mode = OPEN_EXISTING
0040101C |. 6A 00      PUSH 0          ;|pSecurity = NULL
0040101E |. 6A 00      PUSH 0          ;|ShareMode = 0
00401020 |. 68 00000080  PUSH 80000000 ;|Access = GENERIC_READ
00401025 |. 68 B9204000  PUSH abexcrac.004020B9 ;|FileName = "abex.l2c"
0040102A |. E8 5E000000  CALL <JMP.&KERNEL32.CreateFileA> ;|CreateFileA

```

- Như vậy CrackMe này cần KeyFile : "abex.l2c". Ta đặt BreakPoint tại hàm tạo File này.

II – Cracking :

- Quá trình kiểm tra File được diễn ra như sau :

```

0040102F |. A3 CA204000  MOV DWORD PTR DS:[4020CA],EAX ;<== Have File or Not ?
00401034 |. 83F8 FF      CMP EAX,-1           ;<== If have file
00401037 |. 74 3C      JE SHORT abexcrac.00401075 ;<== Continue next check
00401039 |. 6A 00      PUSH 0          ;|pFileSizeHigh = NULL
0040103B |. FF35 CA204000 PUSH DWORD PTR DS:[4020CA] ;|hFile = NULL
00401041 |. E8 4D000000  CALL <JMP.&KERNEL32.GetFileSize> ;|GetFileSize
00401046 |. 83F8 12      CMP EAX,12        ;<== Size of KeyFile must be 18 bytes
00401049 |. 75 15      JNZ SHORT abexcrac.00401060 ;<== If correct ! Congrat !!!
0040104B |. 6A 00      PUSH 0          ;|Style =
MB_OK|MB_APPLMODAL
0040104D |. 68 35204000  PUSH abexcrac.00402035 ;|Title = "Well done!"
00401052 |. 68 40204000  PUSH abexcrac.00402040 ;|Text = "Yep, keyfile found!"
00401057 |. 6A 00      PUSH 0          ;|hOwner = NULL
00401059 |. E8 41000000  CALL <JMP.&USER32.MessageBoxA> ;|MessageBoxA

```

- Như vậy, chỉ cần nhập vào file "abex.l2c" chuỗi dài 18 ký tự là File được chấp nhận.

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : N/A
User : VHT-cRaCkErS	Serial : N/A
File : abeex.l2c	Serial : input 18 charts

III – KeyGen :

N/A

IV – SourceCode (VC++) :

N/A

V – End of Tut :

- Finished – 28/06/2004

Reverse Engineering Association

CrackMe

Homepage :	http://crackmes.de
CrackMe :	abex 4 - Level 1
Coder :	abex
Cracker :	Moonbaby
Type :	Serial
Packed :	N/A
Language :	Microsoft Visual Basic 5.0 / 6.0
Crack Tool :	OllyDbg 1.09d, PeID 0.92
Unpack Tool :	N/A
Request :	Correct Serial / KeyGen

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **Microsoft Visual Basic 5.0 / 6.0**
 - Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "Very good, you got it!" ở địa chỉ : 00401F47 . C785 04FFFFFF>MOV DWORD PTR SS:[EBP-FC],abexcrac.00401>; UNICODE "Very good, you got it!"
 - Tuy nhiên, ta nhận thấy đoạn CODE này bị cô lập. Điều này có nghĩ đây là một FUNCTION được gọi từ một hàm CALL khác . Không thể dùng cách đặt BreakPoint ở trong Function này được .
 - Ta sẽ đi tìm các hàm API được sử dụng trong CrackMe này . Ta thấy có các hàm đáng quan tâm sau : MSVBVM60 rtcGetPresentDate, MSVBVM60 rtcGetYear, MSVBVM60 rtcGetHourOfDay, MSVBVM60 __vbaVarAdd, MSVBVM60 __vbaVarMul
 - Điều này hướng ta suy nghĩ đến CrackMe sẽ sử dụng các thông số về ngày, tháng năm, giờ ... để thực hiện các phép toán cộng và nhân . Vì thế ta sẽ truy đến địa chỉ lấy các thông số này . Khi truy đến đây, nhận thấy tất cả các hàm liệt kê trên cùng nằm trên một Function . Như vậy đây là đoạn CODE chính ta cần quan tâm . Đặt BreakPoint tại :
- 00402120 . FF51 04 CALL DWORD PTR DS:[ECX+4] ; MSVBVM60.Zombie_AddRef

II – Cracking :

- Quá trình mã hóa này được diễn ra như sau :
- ```

00402123 . 8B1D B0104000 MOV EBX,DWORD PTR DS:<&MSVBVM60.#546> ;
MSVBVM60 rtcGetPresentDate
00402129 . 8D55 D8 LEA EDX,DWORD PTR SS:[EBP-28]
0040212C . 33F6 XOR ESI,ESI
0040212E . 52 PUSH EDX
0040212F . 8975 E8 MOV DWORD PTR SS:[EBP-18],ESI
00402132 . 8975 D8 MOV DWORD PTR SS:[EBP-28],ESI
00402135 . 8975 C8 MOV DWORD PTR SS:[EBP-38],ESI
00402138 . 8975 B8 MOV DWORD PTR SS:[EBP-48],ESI
0040213B . 8975 A8 MOV DWORD PTR SS:[EBP-58],ESI
0040213E . 8975 98 MOV DWORD PTR SS:[EBP-68],ESI
00402141 . 8975 88 MOV DWORD PTR SS:[EBP-78],ESI
00402144 . 89B5 78FFFFFF MOV DWORD PTR SS:[EBP-88],ESI

```

```

0040214A . 89B5 68FFFFFF MOV DWORD PTR SS:[EBP-98],ESI
00402150 . 89B5 58FFFFFF MOV DWORD PTR SS:[EBP-A8],ESI
00402156 . FFD3 CALL EBX ; <&MSVBVM60.#546>
00402158 . 8D45 D8 LEA EAX,DWORD PTR SS:[EBP-28]
0040215B . 8D4D C8 LEA ECX,DWORD PTR SS:[EBP-38]
0040215E . 50 PUSH EAX
0040215F . 51 PUSH ECX
00402160 . FF15 A4104000 CALL DWORD PTR DS:[<&MSVBVM60.#543>] ;
MSVBVM60 rtcGetHourOfDay
00402166 . 8D55 98 LEA EDX,DWORD PTR SS:[EBP-68]
00402169 . B8 02000000 MOV EAX,2
0040216E . 52 PUSH EDX
0040216F . C785 70FFFFFF>MOV DWORD PTR SS:[EBP-90],5
00402179 . 8985 68FFFFFF MOV DWORD PTR SS:[EBP-98],EAX
0040217F . C785 60FFFFFF>MOV DWORD PTR SS:[EBP-A0],3E8
00402189 . 8985 58FFFFFF MOV DWORD PTR SS:[EBP-A8],EAX
0040218F . FFD3 CALL EBX
00402191 . 8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
00402194 . 8D4D 88 LEA ECX,DWORD PTR SS:[EBP-78]
00402197 . 50 PUSH EAX
00402198 . 51 PUSH ECX
00402199 . FF15 20104000 CALL DWORD PTR DS:[<&MSVBVM60.#553>] ;
MSVBVM60 rtcGetYear
0040219F . 8B1D 5C104000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaVa>];
MSVBVM60 __vbaVarMul
004021A5 . 8D55 C8 LEA EDX,DWORD PTR SS:[EBP-38]
004021A8 . 8D85 68FFFFFF LEA EAX,DWORD PTR SS:[EBP-98]
004021AE . 52 PUSH EDX
004021AF . 8D4D B8 LEA ECX,DWORD PTR SS:[EBP-48]
004021B2 . 50 PUSH EAX
004021B3 . 51 PUSH ECX
004021B4 . FFD3 CALL EBX ; <&MSVBVM60.__vbaVarMul>
004021B6 . 50 PUSH EAX
004021B7 . 8D95 58FFFFFF LEA EDX,DWORD PTR SS:[EBP-A8]
004021BD . 8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
004021C0 . 52 PUSH EDX
004021C1 . 50 PUSH EAX
004021C2 . FF15 94104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarAd>];
MSVBVM60 __vbaVarAdd
004021C8 . 8D4D 88 LEA ECX,DWORD PTR SS:[EBP-78]
004021CB . 50 PUSH EAX
004021CC . 8D95 78FFFFFF LEA EDX,DWORD PTR SS:[EBP-88]
004021D2 . 51 PUSH ECX
004021D3 . 52 PUSH EDX
004021D4 . FFD3 CALL EBX ; <&MSVBVM60.__vbaVarMul>
```

- Quá trình này được viết lại như sau :

(( HourOfDay \* 0x5 ) + 0x3E8 ) \* Year

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |              |
|---------------------|--------------|
| User : REA-cRaCkErS | Serial : N/A |
| User : VHT-cRaCkErS | Serial : N/A |

**III – KeyGen :**

/Section 1/- GetHourOfDay và GetYear  
 /Section 3/- (( HourOfDay \* 0x5 ) + 0x3E8 ) \* Year  
 /Section 3/- Giá trị tính toán được chuyển sang dạng giá trị DEC

**IV – SourceCode ( VC++ ) :**

```
int Serial=0;
SYSTEMTIME st;
WORD wYear=0,wHour=0;
GetSystemTime (&st);
Serial = (((st.wHour + 2) * 5) + 1000) * st.wYear;
SetDlgItemInt(IDC_Serial,Serial);
```

**V – End of Tut :**

- Finished – **28/06/2004**

# Reverse Engineering Association

## CrackMe

|                    |   |                                                     |
|--------------------|---|-----------------------------------------------------|
| <b>Homepage</b>    | : | <a href="http://crackmes.de">http://crackmes.de</a> |
| <b>CrackMe</b>     | : | <b>abex 5 - Level 1</b>                             |
| <b>Coder</b>       | : | <b>abex</b>                                         |
| <b>Cracker</b>     | : | <b>Moonbaby</b>                                     |
| <b>Type</b>        | : | <b>Serial</b>                                       |
| <b>Packed</b>      | : | <b>N/A</b>                                          |
| <b>Language</b>    | : | <b>MASM32 / TASM32 [Overlay]</b>                    |
| <b>Crack Tool</b>  | : | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| <b>Unpack Tool</b> | : | <b>N/A</b>                                          |
| <b>Request</b>     | : | <b>Correct Serial / KeyGen</b>                      |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và cho ta biết CrackMe được được viết bằng **MASM32 / TASM32 [Overlay]**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "The serial you entered is not correct!" ở địa chỉ : 00401108 |. 68 3B244000 PUSH abexcm5.0040243B ;|Text = "The serial you entered is not correct!"
- Dò ngược lên trên ta nhận thấy CrackMe có sử dụng hàm **GetDlgItemTextA** nên ta đặt BreakPoint tại hàm này : 00401078 |. E8 F4000000 CALL <JMP.&USER32.GetDlgItemTextA> ;|GetDlgItemTextA

**II – Cracking :**

- CrackMe lấy tên của ổ đĩa hiện chứa file (**VolumeNameBuffer**) :

```
0040107D |. 6A 00 PUSH 0 ;|pFileSystemNameSize = NULL
0040107F |. 6A 00 PUSH 0 ;|pFileSystemNameBuffer = NULL
00401081 |. 68 C8204000 PUSH abexcm5.004020C8 ;|pFileSystemFlags = abexcm5.004020C8
```

```

00401086 |. 68 90214000 PUSH abexcm5.00402190 ; |pMaxFilenameLength =
abexcm5.00402190
0040108B |. 68 94214000 PUSH abexcm5.00402194 ; |pVolumeSerialNumber =
abexcm5.00402194
00401090 |. 6A 32 PUSH 32 ; |MaxVolumeNameSize = 32 (50.)
00401092 |. 68 5C224000 PUSH abexcm5.0040225C ; |VolumeNameBuffer = abexcm5.0040225C
00401097 |. 6A 00 PUSH 0 ; |RootPathName = NULL
00401099 |. E8 B5000000 CALL <JMP.&KERNEL32.GetVolumeInformation>;
\GetVolumeInformationA

```

- Chuỗi này được kết hợp với chuỗi mặc định :

```

0040109E |. 68 F3234000 PUSH abexcm5.004023F3 ; /StringToAdd = "4562-ABEX"
004010A3 |. 68 5C224000 PUSH abexcm5.0040225C ; |ConcatString = "Data"
004010A8 |. E8 94000000 CALL <JMP.&KERNEL32.lstrcatA> ; \lstrcatA

```

- 4 ký tự đầu của chuỗi này được cộng thêm 2 đơn vị :

```

004010AD |. B2 02 MOV DL,2 ; <== Loop 2 times
004010AF |> 8305 5C224000>|ADD DWORD PTR DS:[40225C],1 ; <== S[0]
004010B6 |. 8305 5D224000>|ADD DWORD PTR DS:[40225D],1 ; <== S[1]
004010BD |. 8305 5E224000>|ADD DWORD PTR DS:[40225E],1 ; <== S[2]
004010C4 |. 8305 5F224000>|ADD DWORD PTR DS:[40225F],1 ; <== S[3]
004010CB |. FECA |DEC DL ; <== if (i < 2)
004010CD |.^ 75 E0 \JNZ SHORT abexcm5.004010AF ; <== Continue loop

```

- Chuỗi mới mã hoá này được kết hợp thêm với chuỗi mặc định để tạo ra chuỗi serial thực :

```

004010DE |. 68 5C224000 PUSH abexcm5.0040225C ; /StringToAdd = "Fcvc4562-ABEX"
004010E3 |. 68 00204000 PUSH abexcm5.00402000 ; |ConcatString = "L2C-5781"
004010E8 |. E8 54000000 CALL <JMP.&KERNEL32.lstrcatA>; \lstrcatA

```

/Note/- Nếu VolumeNameBuffer trả về giá trị NULL, có nghĩa ô đĩa không được đặt tên thì 4 ký tự đầu tiên của chuỗi mặc định thứ nhất sẽ được cộng thêm 2 đơn vị.

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| User : REA-cRaCkErS | Serial : N/A                                           |
| User : VHT-cRaCkErS | Serial : N/A                                           |
|                     | Serial : L2C-5781Fcvc4562-ABEX , ComputerName : Data ) |

### III – KeyGen :

/Section 1/- GetVolumeNameBuffer, Kết hợp với chuỗi mặc định "4562-ABEX"

/Section 2/- Cộng thêm 2 đơn vị cho 4 ký tự đầu của chuỗi này

/Section 3/- Kết hợp thêm với chuỗi mặc định thứ hai "L2C-5781"

### IV – SourceCode ( VC++ ) :

```

char reaSerial[64] = "L2C-5781";
char reaVolumeName[255] = {0};
int i=0;
unsigned long VolumeNameSize = 255;
GetVolumeInformation(NULL, reaVolumeName, VolumeNameSize, NULL, NULL, NULL, NULL, NULL);
 lstrcat(reaVolumeName, "4562-ABEX");
 i=0;
 while (i < 4)
 {
 reaVolumeName[i] = reaVolumeName[i] + 2;
 i++;
}

```

```

 }
 lstrcat(reaSerial, reaVolumeName);
 SetDlgItemText(IDC_Serial, reaSerial);
}

```

**V – End of Tut :**

- Finished – **28/06/2004**

# Reverse Engineering Association

## CrackMe

|                    |   |                                                                                |
|--------------------|---|--------------------------------------------------------------------------------|
| <b>Homepage</b>    | : | <a href="http://crackmes.de">http://crackmes.de</a>                            |
| <b>CrackMe</b>     | : | <b>acid bytes - cff crackme #4 - Level 0</b>                                   |
| <b>Coder</b>       | : | <b>acid bytes</b>                                                              |
| <b>Cracker</b>     | : | <b>Moonbaby</b>                                                                |
| <b>Type</b>        | : | Name / Serial                                                                  |
| <b>Packed</b>      | : | <b>UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -&gt; Markus &amp; Laszlo</b> |
| <b>Language</b>    | : | Borland Delphi 4.0 - 5.0                                                       |
| <b>Crack Tool</b>  | : | <b>OllyDbg 1.09d, PeiD 0.92</b>                                                |
| <b>Unpack Tool</b> | : | <b>Manual</b>                                                                  |
| <b>Request</b>     | : | Correct Serial / KeyGen                                                        |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo**. Sau khi UnPACK và kiểm tra lại ta biết CrackMe được được viết bằng **Borland Delphi 4.0 - 5.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "The Serial you entered is in any case not valid !" ở địa chỉ : 00457E56 . BA AC7F4500 MOV EDX,unpack\_.00457FAC ; ASCII "The Serial you entered is in any case not valid !"

- Dò ngược lên trên ta đặt BreakPoint tại :  
00457BD3 . E8 08C3FCFF CALL unpack\_.00423EE0

**II – Cracking :**

- CrackMe kiểm tra chiều dài chuỗi U và S nhập :

```
00457BD8 . 837D FC 00 CMP DWORD PTR SS:[EBP-4],0 ; <== Name must be input
00457BDC . 75 18 JNZ SHORT unpack_.00457BF6
```

```
.....
```

```
00457C4E . 83F8 06 CMP EAX,6 ; <== User atleast 6 charts
```

```
00457C51 . 73 1D JNB SHORT unpack_.00457C70
```

- Quá trình mã hoá diễn ra với 6 ký tự đầu tiên của chuỗi U nhập :

```
00457C92 . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[0]
00457C97 . 6BF0 02 IMUL ESI,EAX,2 ; <== Temp = U[0] * 2
00457C9A . 71 05 JNO SHORT unpack_.00457CA1
00457C9C . E8 EBAEFAFF CALL unpack_.00402B8C
00457CA1 > 8D55 F8 LEA EDX,DWORD PTR SS:[EBP-8]
00457CA4 . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]
```

```

00457CAA . E8 31C2FCFF CALL unpack_.00423EE0
00457CAF . 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
00457CB2 . BA 02000000 MOV EDX,2
00457CB7 . 4A DEC EDX
00457CB8 . 3B50 FC CMP EDX,DWORD PTR DS:[EAX-4]
00457CBB . 72 05 JB SHORT unpack_.00457CC2
00457CBD . E8 C2AEFAFF CALL unpack_.00402B84
00457CC2 > 42 INC EDX
00457CC3 . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[1]
00457CC8 . 6BC0 02 IMUL EAX,EAX,2 ; <== Temp_01 = U[1] * 2
00457CCB . 71 05 JNO SHORT unpack_.00457CD2
00457CCD . E8 BAAEFAFF CALL unpack_.00402B8C
00457CD2 > 03F0 ADD ESI,EAX ; <== Temp = TEmp + Temp_01
00457CD4 . 71 05 JNO SHORT unpack_.00457CDB
00457CD6 . E8 B1AEFAFF CALL unpack_.00402B8C
00457CDB > 8D55 F4 LEA EDX,DWORD PTR SS:[EBP-C]
00457CDE . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]
00457CE4 . E8 F7C1FCFF CALL unpack_.00423EE0
00457CE9 . 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C]
00457CEC . BA 03000000 MOV EDX,3
00457CF1 . 4A DEC EDX
00457CF2 . 3B50 FC CMP EDX,DWORD PTR DS:[EAX-4]
00457CF5 . 72 05 JB SHORT unpack_.00457CFC
00457CF7 . E8 88AEFAFF CALL unpack_.00402B84
00457CFC > 42 INC EDX
00457CFD . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[2]
00457D02 . 6BC0 02 IMUL EAX,EAX,2 ; <== Temp_01 = U[2] * 2
00457D05 . 71 05 JNO SHORT unpack_.00457D0C
00457D07 . E8 80AEFAFF CALL unpack_.00402B8C
00457D0C > 03F0 ADD ESI,EAX ; <== Temp = TEmp + Temp_01
00457D0E . 71 05 JNO SHORT unpack_.00457D15
00457D10 . E8 77AEFAFF CALL unpack_.00402B8C
00457D15 > 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]
00457D18 . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]
00457D1E . E8 BDC1FCFF CALL unpack_.00423EE0
00457D23 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10]
00457D26 . BA 04000000 MOV EDX,4
00457D2B . 4A DEC EDX
00457D2C . 3B50 FC CMP EDX,DWORD PTR DS:[EAX-4]
00457D2F . 72 05 JB SHORT unpack_.00457D36
00457D31 . E8 4EAEFAFF CALL unpack_.00402B84
00457D36 > 42 INC EDX
00457D37 . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[3]
00457D3C . 6BC0 02 IMUL EAX,EAX,2 ; <== Temp_01 = U[3] * 2
00457D3F . 71 05 JNO SHORT unpack_.00457D46
00457D41 . E8 46AEFAFF CALL unpack_.00402B8C
00457D46 > 03F0 ADD ESI,EAX ; <== Temp = TEmp + Temp_01
00457D48 . 71 05 JNO SHORT unpack_.00457D4F
00457D4A . E8 3DAEFAFF CALL unpack_.00402B8C
00457D4F > 8D55 EC LEA EDX,DWORD PTR SS:[EBP-14]
00457D52 . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]
00457D58 . E8 83C1FCFF CALL unpack_.00423EE0

```

```

00457D5D . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14]
00457D60 . BA 05000000 MOV EDX,5
00457D65 . 4A DEC EDX
00457D66 . 3B50 FC CMP EDX,DWORD PTR DS:[EAX-4]
00457D69 . 72 05 JB SHORT unpack_.00457D70
00457D6B . E8 14AEFAFF CALL unpack_.00402B84
00457D70 > 42 INC EDX
00457D71 . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[4]
00457D76 . 6BC0 02 IMUL EAX,EAX,2 ; <== Temp_01 = U[4] * 2
00457D79 . 71 05 JNO SHORT unpack_.00457D80
00457D7B . E8 0CAEFAFF CALL unpack_.00402B8C
00457D80 > 03F0 ADD ESI,EAX ; <== Temp = Temp + Temp_01
00457D82 . 71 05 JNO SHORT unpack_.00457D89
00457D84 . E8 03AEFAFF CALL unpack_.00402B8C
00457D89 > 8D55 E8 LEA EDX,DWORD PTR SS:[EBP-18]
00457D8C . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]
00457D92 . E8 49C1FCFF CALL unpack_.00423EE0
00457D97 . 8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18]
00457D9A . BA 06000000 MOV EDX,6
00457D9F . 4A DEC EDX
00457DA0 . 3B50 FC CMP EDX,DWORD PTR DS:[EAX-4]
00457DA3 . 72 05 JB SHORT unpack_.00457DAA
00457DA5 . E8 DAADFAFF CALL unpack_.00402B84
00457DAA > 42 INC EDX
00457DAB . 0FB64410 FF MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== U[5]
00457DB0 . 6BC0 02 IMUL EAX,EAX,2 ; <== Temp_01 = U[5] * 2
00457DB3 . 71 05 JNO SHORT unpack_.00457DBA
00457DB5 . E8 D2ADFAFF CALL unpack_.00402B8C
00457DBA > 03F0 ADD ESI,EAX ; <== Temp = Temp + Temp_01
00457DBC . 71 05 JNO SHORT unpack_.00457DC3
00457DBE . E8 C9ADFAFF CALL unpack_.00402B8C
00457DC3 > 8935 40B84500 MOV DWORD PTR DS:[45B840],ESI
00457DC9 > A1 44B84500 MOV EAX,DWORD PTR DS:[45B844]
00457DCE . E8 FDFBFIAFF CALL unpack_.004079D0 ; <== Temp_01 = Len.U * 2
00457DD3 . 6BC0 02 IMUL EAX,EAX,2
00457DD6 . 73 05 JNB SHORT unpack_.00457DDD
00457DD8 . E8 AFADFAFF CALL unpack_.00402B8C
00457DDD > 33D2 XOR EDX,EDX
00457DDF . 52 PUSH EDX
00457DE0 . 50 PUSH EAX
00457DE1 . A1 40B84500 MOV EAX,DWORD PTR DS:[45B840] ; <== Temp
00457DE6 . 99 CDQ
00457DE7 . 030424 ADD EAX,DWORD PTR SS:[ESP] ; <== Temp = Temp + Temp_01

```

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |              |
|---------------------|--------------|
| User : REA-cRaCkErS | Serial : 908 |
| User : VHT-cRaCkErS | Serial : 960 |

### III – KeyGen :

- /Section 1/- Temp = Temp + U[i] \* 2      Với    ( i < 6 )
- /Section 2/- Temp = Temp + Len.U \* 2
- /Section 3/- Chuyển sang giá trị DEC

**IV – SourceCode ( VC++ ) :**

```

char reaName[64]={0};
int LenUser=0;
int i=0;
int reaSerial=0;

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 6)
{
 MessageBox("----- Your name atleast 6 charts ----- ","Hey !!
Please input your name again !! ");
}
else
{
 i=0; reaSerial = 0;
 while (i < 6)
 {
 reaSerial = reaSerial + reaName[i] * 0x2;
 i++;
 }
 reaSerial = reaSerial + LenUser * 2;
 SetDlgItemInt(IDC_Serial,reaSerial);
}

```

**V – End of Tut :**

- Finished – **28/06/2004**

# Reverse Engineering Association

## CrackMe

|                    |   |                                                     |
|--------------------|---|-----------------------------------------------------|
| <b>Homepage</b>    | : | <a href="http://crackmes.de">http://crackmes.de</a> |
| <b>CrackMe</b>     | : | <b>analyst - keygenning4newbies #1 - Level 1</b>    |
| <b>Coder</b>       | : | <b>analyst</b>                                      |
| <b>Cracker</b>     | : | <b>Moonbaby</b>                                     |
| <b>Type</b>        | : | Name / Serial                                       |
| <b>Packed</b>      | : | N / A                                               |
| <b>Language</b>    | : | Borland C++ [Overlay]                               |
| <b>Crack Tool</b>  | : | <b>OllyDbg 1.09d, PeID 0.92</b>                     |
| <b>Unpack Tool</b> | : | N / A                                               |
| <b>Request</b>     | : | Correct Serial / KeyGen                             |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Borland C++ [Overlay]**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm chuỗi thông báo và tìm được chuỗi "This serial is \*NOT\* Valid!! Try again... : UNREGISTERED" ở địa chỉ :

00401164 > \68 90B44000 PUSH k4n.0040B490 ; /Text = "This serial is \*NOT\* Valid!! Try again... : UNREGISTERED"

- Dò ngược lên trên ta đặt BreakPoint tại :

00401097 |. E8 129C0000 CALL <JMP.&USER32.GetWindowTextA> ; \GetWindowTextA

## II – Cracking :

- CrackMe kiểm tra chiều dài chuỗi U nhập :

|                                          |                                   |
|------------------------------------------|-----------------------------------|
| 004010F0  . 837D D8 03 CMP [LOCAL.10],3  | ; <== Len.U atleast 4 charts      |
| 004010F4  . 7E 7B JLE SHORT k4n.00401171 |                                   |
| 004010F6  . 90 NOP                       |                                   |
| 004010F7  . 90 NOP                       |                                   |
| 004010F8  . 90 NOP                       |                                   |
| 004010F9  . 90 NOP                       |                                   |
| 004010FA  . 33C9 XOR ECX,ECX             |                                   |
| 004010FC  . 33D2 XOR EDX,EDX             |                                   |
| 004010FE  . 33DB XOR EBX,EBX             |                                   |
| 00401100  . 33C0 XOR EAX,EAX             |                                   |
| 00401102  . 837D D8 32 CMP [LOCAL.10],32 | ; <== But not over than 50 charts |
| 00401106  . 7D 69 JGE SHORT k4n.00401171 |                                   |

- Quá trình mã hoá được diễn ra như sau :

|                                                                           |                                |
|---------------------------------------------------------------------------|--------------------------------|
| 0040110C  > /0FBE840D 48FF>/MOVsx EAX,BYTE PTR SS:[EBP+ECX-B8] ; <== U[i] |                                |
| 00401114  .  41  INC ECX                                                  | ; <== i++;                     |
| 00401115  .  33C1  XOR EAX,ECX                                            | ; <== Temp_01 = U[i-1] xor i   |
| 00401117  .  03D8  ADD EBX,EAX                                            | ; <== Temp = Temp + Temp_01    |
| 00401119  .  3B4D D8  CMP ECX,[LOCAL.10]                                  | ; <== While ( i < Len.U )      |
| 0040111C  .^\75 EE  JNZ SHORT k4n.0040110C                                | ; <== Continue Loop            |
| 0040111E  . 6BC0 06 IMUL EAX,EAX,6                                        | ; <== Temp_01 = Temp_01 * 0x6  |
| 00401121  . C1E3 07 SHL EBX,7                                             | ; <== Temp = Temp shl 0x7      |
| 00401124  . 03C3 ADD EAX,EBX                                              | ; <== Temp = Temp + Temp_01    |
| 00401126  . 8945 C8 MOV [LOCAL.14],EAX                                    |                                |
| 00401129  . FF75 C8 PUSH [LOCAL.14]                                       | ; /Arg3                        |
| 0040112C  . 68 38B44000 PUSH k4n.0040B438                                 | ;  Arg2 = 0040B438 ASCII "%lX" |
| 00401131  . 8D8D 80FEFFFF LEA ECX,[LOCAL.96]                              | ;                              |
| 00401137  . 51 PUSH ECX                                                   | ;  Arg1                        |
| 00401138  . E8 873D0000 CALL k4n.00404EC4                                 | ; \ k4n.00404EC4               |

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |              |
|---------------------|--------------|
| User : REA-cRaCkErS | Serial : 908 |
| User : VHT-cRaCkErS | Serial : 960 |

## III – KeyGen :

/Section 1/- Temp = Temp + (U[i-1] xor i)  
 /Section 2/- Temp = (U[Len.U-1] ^ Len.U)\* 0x6) + (Temp shl 0x7)  
 /Section 3/- Xuất ra theo định dạng "%lX"

## IV – SourceCode ( VC++ ) :

```
char reaName[64]={0};
char reaSerial[64]={0};
int LenUser=0;
int i = 0, Temp = 0, Temp_01 = 0;
```

```

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 4 || LenUser > 50)
{
 MessageBox("----- Your name atleast 4 charts ----- \n\n -----
===== But not over 50 charts ===== " ,"Hey !! Please input your name again !! ");
}
else
{
 i = 0; Temp = 0;
 while (i < LenUser)
 {
 i++;
 Temp = Temp + (reaName[i-1] ^ i);
 }

 Temp = ((reaName[LenUser-1] ^ i) * 0x6) + (Temp << 0x7);
 wsprintf(reaserial,"%X",Temp);
 SetDlgItemText(IDC_Serial,reaserial);
}

```

**V – End of Tut :**

- Finished – **28/06/2004**

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <b>CrackMe by v1ru5 - Level 2</b>                   |
| Coder :       | <b>v1ru5</b>                                        |
| Cracker :     | <b>Moonbaby</b>                                     |
| Type :        | Serial                                              |
| Packed :      | <b>ASPack 2.12 -&gt; Alexey Solodovnikov</b>        |
| Language :    | Microsoft Visual C++ 6.0                            |
| Crack Tool :  | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| Unpack Tool : | <b>Manual</b>                                       |
| Request :     | Correct Serial                                      |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **ASPack 2.12 -> Alexey Solodovnikov**. Sau khi UnPACK và kiểm tra lại biết được CrackMe được viết bằng **Microsoft Visual C++ 6.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm được chuỗi "Krivo nisi Cracker!!!!" ở địa chỉ : 004015B4 68 74304000 PUSH CrackMe\_.00403074 ; ASCII "Krivo nisi Cracker!!!!"
- Dò ngược lên trên ta thấy CrackMe có sử dụng hàm **kernel32.lstrlenA**. Ta đặt BreakPoint tại hàm này : 00401560 FF15 04204000 CALL DWORD PTR DS:[<&KERNEL32.lstrlen>] ; kernel32.lstrlenA

**II – Cracking :**

- Nhấn F9 để chạy CrackMe, nhập vào Fake Serial. CrackMe sẽ dừng lại tại điểm đặt BP. Quá trình kiểm tra chiều dài chuỗi S nhập :

```
00401566 8945 F0 MOV DWORD PTR SS:[EBP-10],EAX ; <== Len.S
00401569 837D F0 01 CMP DWORD PTR SS:[EBP-10],1 ; <== At least 1 charts
0040156D 73 16 JNB SHORT CrackMe_.00401585
0040156F 6A 40 PUSH 40
00401571 68 2C304000 PUSH CrackMe_.0040302C ; ASCII "CrackMe"
00401576 68 34304000 PUSH CrackMe_.00403034 ; ASCII "Krivo ti nisi Cracker!!!!"
0040157B 8B4D E0 MOV ECX,DWORD PTR SS:[EBP-20]
0040157E E8 7B050000 CALL <JMP.&MFC42.#4224>
```

- CrackMe này rất đơn giản, ta tìm thấy được ngay Correct Serial khi CrackMe load lên để đưa vào lệnh so sánh với Fake Serial ta nhập vào :

```
00401585 8D4D E4 LEA ECX,DWORD PTR SS:[EBP-1C] ; <== Correct Serial
00401588 51 PUSH ECX
00401589 8D55 F4 LEA EDX,DWORD PTR SS:[EBP-C] ; <== Fake Serial
0040158C 52 PUSH EDX
0040158D FF15 00204000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmp>] ; kernel32.lstrcmpA
- Thực ra, nếu đặt BP từ đầu Function thì ta có thể truy ra được Serial thực ngay từ câu lệnh :
0040153E 66:8B0D 28304000>MOV CX,WORD PTR DS:[403028] ; <== Correct Serial
```

/\*/\*/\* - SERIAL :

User : REA-cRaCkErS

User : VHT-cRaCkErS

Serial : 012376542

### III – KeyGen :

No Need

### IV – SourceCode ( VC++ ) :

No Need

### V – End of Tut :

- Finished – 15/06/2004

# Reverse Engineering Association

## CrackMe

|               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a>                            |
| CrackMe :     | <b>exrek - crackme #1 - Level 0</b>                                            |
| Coder :       | <b>exrek</b>                                                                   |
| Cracker :     | <b>Moonbaby</b>                                                                |
| Type :        | Name / Code1 / Code2 / Code3                                                   |
| Packed :      | <b>UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -&gt; Markus &amp; Laszlo</b> |
| Language :    | Borland Delphi 4.0 - 5.0                                                       |
| Crack Tool :  | <b>OllyDbg 1.09d, PeID 0.92</b>                                                |
| Unpack Tool : | <b>Manual</b>                                                                  |
| Request :     | Correct Code / KeyGen                                                          |

## I – Information :

- Dùng PEiD kiểm tra biết CrackMe PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo**. UnPACK và kiểm tra lại biết CrackMe viết bằng **Borland Delphi 4.0 - 5.0**
- Load CrackMe bằng Olly và tìm chuỗi, ta thấy chuỗi "You are very good cracker!! (-)" ở địa chỉ : 0045D31D . B8 48D44500 MOV EAX,eXccme#1.0045D448 ; ASCII "You are very good cracker!! (-)"
- Dò ngược lên trên ta thấy đoạn kiểm tra chiều dài chuỗi User (U) nhập :  
 0045D2D7 . 83F8 05 CMP EAX,5 ; <== Len of User  
 0045D2DA . 0F8E 2F010000 JLE eXccme#1.0045D40F ; <== Atleast 6 charts
- Tuy CrackMe không đề cập đến chiều dài tối đa của chuỗi U nhập, nhưng ô nhập U chỉ giới hạn tối đa là 10 ký tự.
- Đặt BreakPoint (BP) :  
 0045D2CA . E8 C9A6FEFF CALL eXccme#1.00447998 ; <== Set BreakPoint

## II – Cracking :

- Nhấn F9 để chạy CrackMe, nhập vào U và Fake Code. CrackMe sẽ dừng lại tại điểm đặt BP. Qua quá trình kiểm tra chiều dài U nhập, ta đến quá trình mã hoá Code thứ nhất :

0045D306 . E8 7DFDFFFF CALL eXccme#1.0045D088 ; <== CODE 1

- Dùng F7 để trace into ta đến quá trình xử lý chuỗi :

```

0045D0BB |. BF 01000000 MOV EDI,1 ; <== EDX = i = 0x1
0045D0C0 > 8B45 FC |MOV EAX,[LOCAL.1]
0045D0C3 |. 8A5C38 FF |MOV BL,BYTE PTR DS:[EAX+EDI-1] ; <== Charts of User : U[i]
0045D0C7 |. 80FB 5F |CMP BL,5F ; <== Compare with 0x5F
0045D0CA |. 76 1A |JBE SHORT eXccme#1.0045D0E6
0045D0CC |. 8D45 FC |LEA EAX,[LOCAL.1]
0045D0CF |. E8 D46CFAFF |CALL eXccme#1.00403DA8
0045D0D4 |. 8B55 FC |MOV EDX,[LOCAL.1]
0045D0D7 |. 33D2 |XOR EDX,EDX ; <== If Greater
0045D0D9 |. 8AD3 |MOV DL,BL ; <== DL = BL = U[i]
0045D0DB |. 83EA 08 |SUB EDX,8 ; <== Temp = U[i] - 0x8
0045D0DE |. 2BD7 |SUB EDX,EDI ; <== Temp = Temp - i
0045D0E0 |. 885438 FF |MOV BYTE PTR DS:[EAX+EDI-1],DL ; <== U[i] = Temp
0045D0E4 |. EB 18 |JMP SHORT eXccme#1.0045D0FE ; <== Jump to i++;
0045D0E6 > 8D45 FC |LEA EAX,[LOCAL.1]
0045D0E9 |. E8 BA6CFAFF |CALL eXccme#1.00403DA8
0045D0EE |. 8B55 FC |MOV EDX,[LOCAL.1]
0045D0F1 |. 33D2 |XOR EDX,EDX ; <== If Below or Equal
0045D0F3 |. 8AD3 |MOV DL,BL ; <== DL = BL = U[i]
0045D0F5 |. 83C2 08 |ADD EDX,8 ; <== Temp = U[i] + 0x8
0045D0F8 |. 03D7 |ADD EDX,EDI ; <== Temp = Temp + i
0045D0FA |. 885438 FF |MOV BYTE PTR DS:[EAX+EDI-1],DL ; <== U[i] = Temp
0045D0FE > 47 |INC EDI ; <== i++;
0045D0FF |. 4E |DEC ESI ; <== Len (U) --;
0045D100 |.^ 75 BE |JNZ SHORT eXccme#1.0045D0C0 ; <== Loop until Len(U) = 0x0

```

- Sau khi kết thúc vòng lặp, trace tiếp bằng F8. Nhận thấy CrackMe chuyển hoá chuỗi mã hoá thành chuỗi mặc định :

0045D32A . B8 74D44500 MOV EAX,eXccme#1.0045D474 ; ASCII "JejbWYRbSS["

0045D32F . E8 D4FCFFFF CALL eXccme#1.0045D008 ; <== "Soundgarden"

- Chuỗi này sẽ được cộng dồn giá trị của từng ký tự để có được giá trị mặc định :

0045D337 . E8 F8FDFFFF CALL eXccme#1.0045D134 ; <== CumulativeValue\_01

- Trace tiếp bằng F8 ta tìm được Code thứ hai :

```
0045D33C . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
0045D33F . 0FB652 04 MOVZX EDX,BYTE PTR DS:[EDX+4] ; <== EDX : U[4]
0045D343 . 83EA 3A SUB EDX,3A ; <== TempValue = U[4] - 0x3A
0045D346 . F7EA IMUL EDX ; <== TempValue = TempValue * CumulativeValue_01
0045D348 . 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX
0045D34B . 8D55 E4 LEA EDX,DWORD PTR SS:[EBP-1C]
0045D34E . 8B87 E8020000 MOV EAX,DWORD PTR DS:[EDI+2E8]
```

0045D354 . E8 3FA6FEFF CALL eXccme#1.00447998 ; <== CODE 2 : TempValue (DEC)

- Dùng F8 trace tiếp một đoạn, ta nhận thấy CODE 1 được cộng dồn lại để lấy giá trị :

```
0045D381 . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24] ; <== CODE 1
0045D384 . E8 ABFDFFFF CALL eXccme#1.0045D134 ; <== CumulativeValue_02
```

- Thêm một đoạn ta đến quá trình tạo phần đầu tiên của CODE thứ 3 :

```
0045D392 . E8 71FCFFFF CALL eXccme#1.0045D008 ; <== "Soundgarden"
0045D397 . 8B45 D8 MOV EAX,DWORD PTR SS:[EBP-28]
0045D39A . E8 95FDFFFF CALL eXccme#1.0045D134 ; <== CumulativeValue_01
0045D39F . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
0045D3A2 . 0FB652 04 MOVZX EDX,BYTE PTR DS:[EDX+4] ; <== EDX : U[4]
0045D3A6 . 83EA 3A SUB EDX,3A ; <== TempValue = U[4] - 0x3A
0045D3A9 . F7EA IMUL EDX ; <== TempValue = TempValue * CumulativeValue_01
0045D3AB . 5A POP EDX ; <== CumulativeValue_02
0045D3AC . 03D0 ADD EDX,EAX ; <== TempValue = TempValue + CumulativeValue_02
0045D3AE . 8955 F8 MOV DWORD PTR SS:[EBP-8],EDX
0045D3B1 . 8D55 D4 LEA EDX,DWORD PTR SS:[EBP-2C]
0045D3B4 . 8B87 F0020000 MOV EAX,DWORD PTR DS:[EDI+2F0]
```

0045D3BA . E8 D9A5FEFF CALL eXccme#1.00447998 ; <== Convert to DEC value

- Phần đầu của CODE 3 là một chuỗi mặc định được mã hoá từ chuỗi cho trước :

```
0045D3C6 . B8 88D44500 MOV EAX,eXccme#1.0045D488 ; ASCII ">hc;"
0045D3CB . E8 38FCFFFF CALL eXccme#1.0045D008 ; <== GrnG
```

- Phần đầu và phần sau của CODE 3 được kết hợp lại :

0045D3EE . E8 A568FAFF CALL eXccme#1.00403C98 ; <== GrnG-Value(DEC) : CODE 3

/\*/\*/\* - Quá trình so sánh của CrackMe này cũng rất lạ :

/+- Nếu CODE 1 đúng :

0045D316 . 83C6 06 ADD ESI,6

/+- Nếu CODE 2 đúng :

0045D373 . 83C6 0C ADD ESI,0C

/+- Nếu CODE 3 đúng :

0045D3FE . 83C6 0C ADD ESI,0C

/+- Nếu cả ba CODE đều đúng thì kết quả phải :

0045D401 >\83FE 24 CMP ESI,24

0045D404 . 75 02 JNZ SHORT eXccme#1.0045D408

/\*/\*/\* - CODE 1, CODE 2, CODE 3 tương ứng với User :

User : REAcRaCkEr

Code 1 : [OLW\_SR[V

Code 2 : 27504

Code 3 : GrnG-28370

User : VHTcRaCkEr

Code 1 : \_R\_W\_SR[V

Code 2 : 27504

Code 3 : GrnG-28396

**III – KeyGen :**

- /Code 1/- Từng ký tự của chuỗi U nhập ( U[i] ) được đếm so sánh với 0x5F :
- Nếu nhỏ hơn hay bằng : Code\_01[i-1] = U[i-1] + 0x8 + i ( i=1)
  - Nếu lớn hơn : Code\_01[i-1] = U[i-1] - 0x8 - i ( i=1)
- /Code 2/- Quá trình này gồm ba giai đoạn :
- Cộng dồn giá trị của các ký tự chuỗi mặc định : CumV\_01
  - Code\_02 = (U[4] - 0x3A) \* CumV\_01
  - Giá trị này được chuyển sang giá trị DEC tương ứng
- /Code 3/- Quá trình này gồm năm giai đoạn :
- Cộng dồn giá trị của các ký tự chuỗi mặc định : CumV\_01
  - Cộng dồn giá trị của các ký tự chuỗi Code\_01 : CumV\_02
  - Temp = { (U[4] - 0x3A) \* CumV\_01 } + CumV\_02
  - Giá trị này được chuyển sang giá trị DEC tương ứng : DecStr
  - Code\_03 : GrnG- DecStr

#### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
char reaCode_01[64]="";
char reaCode_02[64]="";
char reaCode_03[64]="";
char reaDefaultString[12]="Soundgarden";
char reaBarrie[10]="/ ";
int LenUser=0;
int i=0,Temp=0;

LenUser=GetDlgItemText(IDC_Name, reaName,64);

if (LenUser < 6 || LenUser > 10)
{
 MessageBox("Your name atleast 6 charts & not over 10 charts !!!","Hey !! Please input
your name again !! ");
}
else
{
 i=1;
 while (i<LenUser+1)
 {
 if (reaName[i-1] > 0x5F)
 {
 reaCode_01[i-1] = reaName[i-1] - 0x8 - i;
 i++;
 }
 else
 {
 reaCode_01[i-1] = reaName[i-1] + 0x8 + i;
 i++;
 }
 }
 Temp = (reaName[4] - 0x3A) * Cumulate(resetDefaultString);
 wsprintf(reetCode_02,"%i",Temp);
}

```

```

Temp = ((reaName[4] - 0x3A) * Cumulate (reaDefaultString)) + Cumulate (reaCode_01);
wsprintf(reaCode_03,"%i",Temp);

lstrcat(reaSerial, reaCode_01);
lstrcat(reaSerial, reaBarrie);
lstrcat(reaSerial, reaCode_02);
lstrcat(reaSerial, reaBarrie);
lstrcat(reaSerial,"GrnG-");
lstrcat(reaSerial, reaCode_03);

}
SetDlgItemText(IDC_Serial, reaSerial);

```

**V – End of Tut :**

- Finished – ***10/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <b>Fant0m - Crackme 5 - Level 0</b>                 |
| Coder :       | <b>Fant0m</b>                                       |
| Cracker :     | <b>Moonbaby</b>                                     |
| Type :        | Name / Serial                                       |
| Packed :      | N / A                                               |
| Language :    | MASM32 / TASM32                                     |
| Crack Tool :  | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| Unpack Tool : | N / A                                               |
| Request :     | Correct Serial / KeyGen                             |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe này không bị PACK được viết bằng **MASM32 / TASM32**
- Load CrackMe bằng Olly và tìm chuỗi, ta thấy chuỗi "You got it! Congrats! :)" ở địa chỉ : 004015EF . 68 31314000 PUSH CRACKME5.00403131 ; |Text = "You got it! Congrats! :)"
- Dò ngược lên trên ta đến đoạn kiểm tra chiều dài chuỗi User (U) nhập : 0040157D . 83F8 02 CMP EAX,2 ; <== Len User must be 3 charts
- 00401580 . 0F8E A1000000 JLE CRACKME5.00401627
- Đối với trường hợp có hàm **USER32.GetDlgItemTextA** thì ta đặt BreakPoint (BP) tại các hàm này : 00401554 . E8 F3010000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA

**II – Cracking :**

- Nhấn F9 để chạy CrackMe, nhập vào U và Fake Serial. CrackMe sẽ dừng lại tại điểm đặt BP. Dùng F8 trace tiếp ta đến đoạn code kiểm tra tính hợp lệ của Serial nhập :
- |                                                                            |
|----------------------------------------------------------------------------|
| 0040161B . E8 7C000000 CALL CRACKME5.0040169C ; <== Main Code              |
| 00401620 . 83F8 00 CMP EAX,0 ; <== Your Serial is correct ??               |
| 00401623 .^ 74 C3 JE SHORT CRACKME5.004015E8 ; <== If correct : Congrats ! |

- Nhận ra ngay, lệnh **CALL CRACKME5.0040169C** sẽ chuyển đến một Function mã hoá và đồng thời kiểm tra chuỗi Serial nhập. Giá trị kiểm tra sẽ được trả về cho EAX. Như vậy, để xem xét quá trình mã hoá ta dùng F7 để Trace Into lệnh CALL này.

004016BA |. BB 45000000 MOV EBX,45 ; <== [ BH = 0x45 ]  
004016BF |. C1E3 08 SHL EBX,8 ; <== [ ]  
004016C2 |> B3 1A /MOV BL,1A ; <== BL = 0x1A  
004016C4 |. 33C0 |XOR EAX,EAX  
004016C6 |. 8A06 |MOV AL,BYTE PTR DS:[ESI] ; <== U[i]  
004016C8 |. 3C 00 |CMP AL,0  
004016CA |. 74 40 |JE SHORT CRACKME5.0040170C  
004016CC |. 46 |INC ESI ; <== Next chart  
004016CD |. 803A 00 |CMP BYTE PTR DS:[EDX],0 ; <== End of Default String ???  
004016D0 |. 75 06 |JNZ SHORT CRACKME5.004016D8 ; <== DefStr[j] = DefStr[0]  
004016D2 |. 8D15 A2304000 |LEA EDX,DWORD PTR DS:[4030A2]  
004016D8 |> 3202 |XOR AL,BYTE PTR DS:[EDX] ; <== Temp = U[i] xor DefStr[j]  
004016DA |. F6E7 |MUL BH ; <== Temp = Temp \* 0x45  
004016DC |. 66:F7D0 |NOT AX ; <== [ START of this section ]  
004016DF |. 25 FF0F0000 |AND EAX,0FFF ; <== [ if Temp < 0xFFFF =>  
004016E4 |. F6F3 |DIV BL [ Temp = ((0x0FFF - Temp) % 0x1A) + 0x41 ]  
; <== [ if (0x0FFF < Temp < 0x1FFF) =>  
; <== [ Temp = ((0x1FFF - Temp) % 0x1A) + 0x41 ]  
004016E6 |. 80C4 41 |ADD AH,41 ; <== [ if (0x1FFF < Temp < 0x2FFF) =>  
; <== [ Temp = ((0x2FFF - Temp) % 0x1A) + 0x41 ]  
004016E9 |. C1E0 10 |SHL EAX,10 ; <== [ ]  
004016EC |. C1E8 18 |SHR EAX,18 ; <== [ END of this section ]  
004016EF |. 50 |PUSH EAX ; <== [ ]  
004016F0 |. B3 05 |MOV BL,5 ; <== [ START of this section ]  
004016F2 |. 8BC1 |MOV EAX,ECX ; <== [ ]  
004016F4 |. 25 FF0F0000 |AND EAX,0FFF ; <== [ if ((i%5) == 0) ]  
004016F9 |. F6F3 |DIV BL ; <== [ ]  
004016FB |. 80FC 00 |CMP AH,0 ; <== [ Serial[i] = 0x2D ]  
004016FE |. 75 04 |JNZ SHORT CRACKME5.00401704 ; <== [ Serial[i+1] = Temp ]  
00401700 |. C607 2D |MOV BYTE PTR DS:[EDI],2D ; <== [ ]  
00401703 |. 47 |INC EDI ; <== [ Else ]  
00401704 |> 58 |POP EAX ; <== [ ]  
00401705 |. 8807 |MOV BYTE PTR DS:[EDI],AL ; <== [ Serial[i] = Temp ]  
00401707 |. 47 |INC EDI ; <== [ END of this section ]  
00401708 |. 42 |INC EDX  
00401709 |. 41 |INC ECX  
0040170A |.^ EB B6 |JMP SHORT CRACKME5.004016C2

- Tuy nhiên, có hai đoạn CODE bên trên cần đặc biệt chú ý . Lợi dụng đặc tính của ngôn ngữ ASM, CODER đã thay đổi cấu trúc một số lệnh đơn giản thành phức tạp để đánh lừa CRACKER.

/\*/\*/\*/ - SERIAL tương ứng với USER :

### **III – KeyGen :**

Để tạo KeyGen cho CrackMe này có hai cách :

/+ Copy đoạn code ASM của CrackMe này và gắn vào đoạn code của chương trình viết – không cần quan tâm nhiều đến sự thay đổi cấu trúc lệnh một cách cố tình của CODER  
 /+ Nếu CODE mới, cần xem xét thật kỹ để chuyển thành lệnh đúng tương ứng .  
 Thực ra, KeyGen này không khó để CODE.

#### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
char reaDefaultString[64]="JD39-CK4-5QV345";
char reaTemp[64]="";
int LenUser=0;
int i=0,j=0,k=0,Temp=0;

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 1)
{
 MessageBox("Your name atleast 3 chart!!!","Hey !! Please input your name again !! ");
}
else
{
 i=1; j=0; k=0;
 while (i<LenUser+1)
 {
 Temp = 0;
 Temp = (reaName[i-1] ^ reaDefaultString[j]) * 0x45 ;
 if (Temp < 0xFFFF)
 {
 reaTemp[i-1] = ((0xFFFF - Temp) % 0x1A) + 0x41;
 }
 else
 {
 if (Temp < 0x1FFF)
 {
 reaTemp[i-1] = ((0x1FFF - Temp) % 0x1A) + 0x41;
 }
 else
 {
 reaTemp[i-1] = ((0x2FFF - Temp) % 0x1A) + 0x41;
 }
 }
 if ((i%5) == 0)
 {
 reaSerial[k] = 0x2D;
 reaSerial[k+1] = reaTemp[i-1];
 reaSerial[k+2] = NULL;
 k = k+2;
 i++;
 }
 else
 {
 reaSerial[k] = reaTemp[i-1];
 }
 }
}

```

```

 reaSerial[k+1] = NULL;
 k++;
 i++;
 }
 if (j<lstrlen(reOrDefaultString)-1)
 {
 j++;
 }
 else
 {
 j=0;
 }
}
SetDlgItemText(IDC_Serial, reaSerial);

```

**V – End of Tut :**

- Finished – ***11/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <b>Fant0m - Crackme 6 - Level 1</b>                 |
| Coder :       | <b>Fant0m</b>                                       |
| Cracker :     | <b>Moonbaby</b>                                     |
| Type :        | <b>KeyFile</b>                                      |
| Packed :      | <b>N / A</b>                                        |
| Language :    | <b>MASM32 / TASM32</b>                              |
| Crack Tool :  | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| Unpack Tool : | <b>N / A</b>                                        |
| Request :     | <b>Correct KeyFile / KeyGen</b>                     |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe này không bị PACK được viết bằng **MASM32 / TASM32**

- Đối với các CrackMe tìm KeyFile thì điều quan trọng nhất là tìm ra hàm :

Hàm tạo KeyFile - CALL <JMP.&KERNEL32.CreateFileA> ; \CreateFileA

Hàm đọc file - CALL <JMP.&KERNEL32.ReadFile> ; \ReadFile

Hàm kiểm tra kích thước file

....  
CrackMe này khi tìm ở của sổ Intermodular calls ta thấy hai hàm là tạo file và đọc file. Như vậy ta suy nghĩ đến vấn đề là :

- 1- Tạo file
- 2- Kiểm tra nội dung chứa trong file đó có phù hợp với điều kiện hay không.

- Trên cơ sở đó ta tìm về hàm tạo file :

004010E1 /\$ 6A 00 PUSH 0 ; /hTemplateFile = NULL

```

004010E3 |. 68 80000000 PUSH 80 ;|Attributes = NORMAL
004010E8 |. 6A 03 PUSH 3 ;|Mode = OPEN_EXISTING
004010EA |. 6A 00 PUSH 0 ;|pSecurity = NULL
004010EC |. 6A 01 PUSH 1 ;|ShareMode = FILE_SHARE_READ
004010EE |. 68 00000080 PUSH 80000000 ;|Access = GENERIC_READ
004010F3 |. 68 17304000 PUSH CRACKME6.00403017 ;|FileName = "keyfile.dat"
004010F8 |. E8 BD000000 CALL <JMP.&KERNEL32.CreateFileA> ;|CreateFileA

```

Ta thấy ngay tên file cần tạo là "keyfile.dat". Tạo ngay file này, nhập nội dung tùy thích vào. Set BreakPoint tại ngay hàm tạo file để kiểm tra :

## II – Cracking :

- Sau khi load CrackMe lên bằng Olly, nhấn F9, CrackMe dừng lại tại điểm đặt BP . Dùng F8 trace tiếp ta thấy CrackMe tiến hành kiểm tra tên của KeyFile :

```

004010FD |. A3 88304000 MOV DWORD PTR DS:[403088],EAX
00401102 |. 83F8 FF CMP EAX,-1 ;<== had KeyFile ?
00401105 |. 75 14 JNZ SHORT CRACKME6.0040111B
00401107 |. 6A 10 PUSH 10 ;|Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
00401109 |. 68 23304000 PUSH CRACKME6.00403023 ;|Title = "Error!"
0040110E |. 68 2A304000 PUSH CRACKME6.0040302A ;|Text = "Key file not found!"
00401113 |. 6A 00 PUSH 0 ;|hOwner = NULL
00401115 |. E8 94000000 CALL <JMP.&USER32.MessageBoxA> ;|MessageBoxA

```

- Sau đó, CrackMe tiến hành kiểm tra nội dung của KeyFile. Không chấp nhận File rỗng :

```

00401134 |. 83F8 00 CMP EAX,0 ;<== KeyFile contain anything ?
00401137 |. 75 15 JNZ SHORT CRACKME6.0040114E ;<== File must be contain something
00401139 |. 6A 10 PUSH 10 ;|Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
0040113B |. 68 23304000 PUSH CRACKME6.00403023 ;|Title = "Error!"
00401140 |. 68 3E304000 PUSH CRACKME6.0040303E ;|Text = "Error reading file!"
00401145 |. 6A 00 PUSH 0 ;|hOwner = NULL
00401147 |. E8 62000000 CALL <JMP.&USER32.MessageBoxA> ;|MessageBoxA

```

- Dùng F8 trace tiếp ta đến quá trình kiểm tra nội dung của KeyFile :

```

00401154 |. 83C2 05 ADD EDX,5 ;<== i = 5
00401157 |. 803A 31 CMP BYTE PTR DS:[EDX],31 ;<== Chart[5] == 31 (ASCII : 1)
0040115A |. 75 10 JNZ SHORT CRACKME6.0040116C
0040115C |. 83C2 03 ADD EDX,3 ;<== i = 8
0040115F |. 803A 33 CMP BYTE PTR DS:[EDX],33 ;<== Chart[8] == 33 (ASCII : 3)
00401162 |. 75 08 JNZ SHORT CRACKME6.0040116C
00401164 |. 42 INC EDX ;<== i = 9
00401165 |. 803A 30 CMP BYTE PTR DS:[EDX],30 ;<== Chart[9] != 30 (ASCII : 0)
00401168 |. 74 02 JE SHORT CRACKME6.0040116C
0040116A |. EB 15 JMP SHORT CRACKME6.00401181
0040116C |> 6A 10 PUSH 10 ;|Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
0040116E |. 68 23304000 PUSH CRACKME6.00403023 ;|Title = "Error!"
00401173 |. 68 52304000 PUSH CRACKME6.00403052 ;|Text = "Invalid key!"
00401178 |. 6A 00 PUSH 0 ;|hOwner = NULL
0040117A |. E8 2F000000 CALL <JMP.&USER32.MessageBoxA> ;|MessageBoxA
0040117F |. EB 15 JMP SHORT CRACKME6.00401196
00401181 |> 6A 40 PUSH 40 ;|Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
00401183 |. 68 76304000 PUSH CRACKME6.00403076 ;|Title = "Correct!!!"

```

00401188 |. 68 5F304000 PUSH CRACKME6.0040305F ;|Text = "Correct key! Good job!"  
 0040118D |. 6A 00 PUSH 0 ;|hOwner = NULL  
 0040118F |. E8 1A000000 CALL <JMP.&USER32.MessageBoxA> ;\MessageBoxA  
 - Tuy CrackMe không đề cập đến chiều dài của chuỗi nhập, nhưng qua quá trình kiểm tra từng ký tự ta tính được chiều dài tối thiểu của KeyFile là 9 ký tự (ký tự cuối cùng không có cũng không sao ).

/\*/\*/\* - KEYFILE :

|                              |                            |
|------------------------------|----------------------------|
| User : REA-cRaCkErS          | Serial : N / A             |
| User : VHT-cRaCkErS          | Serial : N / A             |
| <b>KeyFile : keyfile.dat</b> | <b>Serial : 2003714738</b> |

### III – KeyGen :

- /Section 1/- Chiều dài chuỗi tối thiểu là 9 ký tự.
- /Section 2/- Ký tự thứ 6 phải là số 1
- /Section 3/- Ký tự thứ 9 phải là số 3
- /Section 4/- Ký tự thứ 10 không phải là số 0.

### IV – SourceCode ( VC++ ) :

```
char reaSerial[20]="";
char szDefaultString[20]="0123456789";
int LenSerial=0,i=0;
int RandomChart;
unsigned long NumWritten;
```

```
HANDLE KeyFile =
CreateFile("keyfile.dat",GENERIC_WRITE,FILE_SHARE_READ,0,CREATE_ALWAYS,FILE_ATTRIBUTE_ARCHIVE,0);

srand((unsigned)time(NULL));
while (lstrlen(reaSerial) != 10)
{
 i=0;
 while (i<10)
 {
 if (i==5)
 {
 reaSerial[i] = 0x31;
 i++;
 }
 if (i == 8)
 {
 reaSerial[i] = 0x33;
 i++;
 }
 if (i == 9)
 {
 reaSerial[i] = szDefaultString[RandomChart=1+rand() % 10];
 i++;
 }
 }
}
```

```

 {
 reaSerial[i] = szDefaultString[RandomChart=rand() % 11];
 i++;
 }
}

LenSerial=lstrlen(reaSerial);
WriteFile(KeyFile,reaSerial,LenSerial,&NumWritten,0);
CloseHandle(KeyFile);

```

**V – End of Tut :**

- Finished – ***14/06/2004***

# Reverse Engineering Association

## CrackMe

|                    |   |                                                     |
|--------------------|---|-----------------------------------------------------|
| <b>Homepage</b>    | : | <a href="http://crackmes.de">http://crackmes.de</a> |
| <b>CrackMe</b>     | : | <b>xyzero - KeygenMe #1 Tangerine - Level 1</b>     |
| <b>Coder</b>       | : | <b>xyzero</b>                                       |
| <b>Cracker</b>     | : | <b>Moonbaby</b>                                     |
| <b>Type</b>        | : | Name / Serial                                       |
| <b>Packed</b>      | : | N / A                                               |
| <b>Language</b>    | : | Microsoft Visual C++ 6.0                            |
| <b>Crack Tool</b>  | : | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| <b>Unpack Tool</b> | : | N / A                                               |
| <b>Request</b>     | : | Correct Serial / KeyGen                             |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **Microsoft Visual C++ 6.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm được chuỗi "Good job!" ở địa chỉ : 00401273 |. 68 40314000 PUSH keygenme.00403140 ;|Text = "Good job!"
- Dò ngược lên trên ta thấy CrackMe có sử dụng hàm GetDlgItemTextA . Ta đặt BreakPoint tại hàm này : 0040116C |. FFD6 CALL ESI ;\GetDlgItemTextA

**II – Cracking :**

- Nhấn F9 để chạy CrackMe, nhập vào U và Fake Serial. CrackMe sẽ dừng lại tại điểm đặt BP. Quá trình kiểm tra chiều dài chuỗi U nhập :
 

```

00401183 |. E8 02010000 CALL <JMP.&MSVCRT.strlen> ;\strlen
00401188 |. 83F8 04 CMP EAX,4 ;<== Name atleast 4 charts
0040118B |. 59 POP ECX
0040118C |. 73 1E JNB SHORT keygenme.004011AC ;<== Continue if NOT BELOW

```

- Tuy CrackMe không đề cập gì đến chiều dài tối đa của chuỗi U nhập, nhưng ô nhập U chỉ cho phép tối đa 23 ký tự . Vậy giới hạn chiều dài của U : 3 < LenUser < 24 . Đồng thời CrackMe này cũng không kiểm tra chiều dài chuỗi Serial nhập.

- Quá trình mã hoá chuỗi U để tạo thành chuỗi Serial thực chỉ là một quá trình tính toán đơn giản .  
Nhưng CODER đã phức tạp hoá bằng cách chuyển đổi qua lại các giá trị :

```

004011BA |. 8BC8 MOV ECX,EAX ; <== Len.U
004011BC |. 6BC9 2D IMUL ECX,ECX,2D ; <== Temp_01 = Len.U * 0x2D
004011BF |. 85C0 TEST EAX,EAX
004011C1 |. 8D1409 LEA EDX,DWORD PTR DS:[ECX+ECX] ; <== Temp_02 = Temp_01 * 2
004011C4 |. 8D3452 LEA ESI,DWORD PTR DS:[EDX+EDX*2] ; <== Temp_03 = Temp_02 +
Temp_02 * 2
004011C7 |. 8975 FC MOV [LOCAL.1],ESI ; <== Store Temp_03
004011CA |. 76 29 JBE SHORT keygenme.004011F5
004011CC |. 8BF0 MOV ESI,EAX
004011CE |. 0FAF75 FC IMUL ESI,[LOCAL.1] ; <== Temp_04 = Temp_03 * Len.U
004011D2 |. 0145 FC ADD [LOCAL.1],EAX ; <== Temp_Store = Temp_03 + Len.U
004011D5 |> 8B7D F8 /MOV EDI,[LOCAL.2] ; <== i
004011D8 |. 03F0 |ADD ESI,EAX ; <== Temp_04 = Temp_04 + Len.U
004011DA |. 0FBEB7C3D D4 |MOVSX EDI,BYTE PTR SS:[EBP+EDI-2C] ; <== U[i]
004011DF |. 0FAFF9 |IMUL EDI,ECX ; <== Temp_05 = Temp_01 * U[i]
004011E2 |. 03FA |ADD EDI,EDX ; <== Temp_05 = Temp_05 + Temp_02
004011E4 |. 017D EC |ADD [LOCAL.5],EDI ; <== TempSection_03 = TempSection_03 +
Temp_05
004011E7 |. 41 |INC ECX ; <== Temp_01 ++
004011E8 |. 8D3C0E |LEA EDI,DWORD PTR DS:[ESI+ECX] ; <== Temp_06 = Temp_04 +
Temp_01
004011EB |. 03D7 |ADD EDX,EDI ; <== Temp_02 = Temp_02 + Temp_06
004011ED |. FF45 F8 |INC [LOCAL.2] ; <== i++
004011F0 |. 3945 F8 |CMP [LOCAL.2],EAX ; <== if (i < Len.U)
004011F3 |.^ 72 E0 |JB SHORT keygenme.004011D5 ; <== Continue Loop
004011F5 |> 33FF XOR EDI,EDI
004011F7 |. 85C0 TEST EAX,EAX
004011F9 |. 76 1F JBE SHORT keygenme.0040121A
004011FB |. 8BF2 MOV ESI,EDX ; <== Temp_02
004011FD |. 6BF6 2D IMUL ESI,ESI,2D ; <== Temp_03 = TEmp_02 * 0x2D
00401200 |. 03F1 ADD ESI,ECX ; <== Temp_03 = Temp_03 + Temp_01
00401202 |> 0FBEB4C3D D4 /MOVSX ECX,BYTE PTR SS:[EBP+EDI-2C] ; <== U[i]
00401207 |. 41 |INC ECX ; <== U[i] + 1
00401208 |. 83C6 2D |ADD ESI,2D ; <== Temp_03 = Temp_03 + 0x2D
0040120B |. 0FAFCA |IMUL ECX,EDX ; <== Temp_04 = (U[i] + 1) * Temp_02
0040120E |. 014D F0 |ADD [LOCAL.4],ECX ; <== Section_02 = Section_02 + Temp_04
00401211 |. 0175 FC |ADD [LOCAL.1],ESI ; <== Temp_Store= Temp_Store + Temp_03
00401214 |. 42 |INC EDX ; <== Temp_02 ++
00401215 |. 47 |INC EDI ; <== i++
00401216 |. 3BF8 |CMP EDI,EAX ; <== if (i < Len.U)
00401218 |.^ 72 E8 |JB SHORT keygenme.00401202 ; <== Continue Loop
0040121A |> 33C9 XOR ECX,ECX
0040121C |. 85C0 TEST EAX,EAX
0040121E |. 76 14 JBE SHORT keygenme.00401234
00401220 |> 0FBEB740D D4 /MOVSX ESI,BYTE PTR SS:[EBP+ECX-2C] ; <== U[i]
00401225 |. 0FAF75 FC |IMUL ESI,[LOCAL.1] ; <== Temp_03 = Temp_Store * U[i]
00401229 |. 03F2 |ADD ESI,EDX ; <== Temp_03 = Temp_03 + Temp_02
0040122B |. 0175 F4 |ADD [LOCAL.3],ESI ; <== Section_01 = Section_01 + Temp_03
0040122E |. 42 |INC EDX ; <== Temp_02 ++
0040122F |. 41 |INC ECX ; <== i++

```

```

00401230 |. 3BC8 |CMP ECX,EAX ; <== if (i < Len.U)
00401232 |.^ 72 EC |JB SHORT keygenme.00401220 ; <== Continue Loop
00401234 |> 8D48 02 |LEA ECX,DWORD PTR DS:[EAX+2] ; <== Temp = Len.U + 2
00401237 |. 8B55 F0 |MOV EDX,[LOCAL.4] ; <== Section_02
0040123A |. 0FAF4D F4 |IMUL ECX,[LOCAL.3] ; <== Section_01 = Section_01 * Temp
0040123E |. 51 |PUSH ECX ; /%u>
0040123F |. 8B4D EC |MOV ECX,[LOCAL.5] ;|<== Section_03
00401242 |. OFAFC1 |IMUL EAX,ECX ;|<== Section_03 = TempSection_03 * Len.U
00401245 |. 03D1 |ADD EDX,ECX ;|<== Section_02 = Section_2 + TempSection_03
00401247 |. 52 |PUSH EDX ; |<%u>
00401248 |. 50 |PUSH EAX ; |<%u>
00401249 |. 8D45 B0 |LEA EAX,[LOCAL.20] ;|
0040124C |. 68 50314000 PUSH keygenme.00403150 ;|Format = "%u-%u-%u"
00401251 |. 50 |PUSH EAX ;|s
00401252 |. FF15 68204000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ;\wsprintfA
- Giá trị của ba đoạn SERIAL ở dạng HEX sẽ được chuyển sang dạng DEC tương ứng.

```

/\*/\*/\* - SERIAL tương ứng với USER :

|                     |                                        |
|---------------------|----------------------------------------|
| User : REA-cRaCkErS | Serial : 37833348-474016360-3560812000 |
| User : VHT-cRaCkErS | Serial : 38002320-486383966-2468743032 |

### III – KeyGen :

- |              |                                                           |
|--------------|-----------------------------------------------------------|
| /Section 1/- | Giới hạn chiều dài User nhập.                             |
| /Section 1/- | Thực hiện quá trình tính toán như được diễn giải bên trên |
| /Section 1/- | Chuyển các giá trị ở dạng HEX thành dạng DEC.             |

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
int LenUser=0;
int i=0,TempStore=0;
int Temp_01=0,Temp_02=0,Temp_03=0,Temp_04=0,Temp_05=0,Temp_06=0;
int Section_01=0,Section_02=0,Section_03=0;

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 4 || LenUser > 23)
{
 MessageBox("Your name atleast 4 chart but not more 23 charts !!!","Hey !! Please input
your name again !! ");
}
else
{
 Temp_01 = LenUser * 0x2D;
 Temp_02 = Temp_01 * 2;
 Temp_03 = Temp_02 * 3;
 Temp_04 = Temp_03 * LenUser;
 TempStore = Temp_03 + LenUser;
 i=0;
 while (i < LenUser)

```

```

{
 Temp_04 = Temp_04 + LenUser;
 Temp_05 = Temp_01 * reaName[i];
 Temp_05 = Temp_05 + Temp_02;
 Section_03 = Section_03 + Temp_05;
 Temp_01++;
 Temp_06 = Temp_04 + Temp_01;
 Temp_02 = Temp_02 + Temp_06;
 i++;
}
Temp_06 = Section_03;

Temp_03 = Temp_02 * 0x2D;
Temp_03 = Temp_03 + Temp_01;
i=0;
while (i < LenUser)
{
 Temp_03 = Temp_03 + 0x2D;
 Temp_04 = (reaName[i] + 1) * Temp_02;
 Section_02 = Section_02 + Temp_04;
 TempStore= TempStore + Temp_03;
 Temp_02++;
 i++;
}

i=0;
while (i < LenUser)
{
 Temp_03 = TempStore * reaName[i];
 Temp_03 = Temp_03 + Temp_02;
 Section_01 = Section_01 + Temp_03; // First Section of Correct Serial
 Temp_02++;
 i++;
}

Section_01 = Section_01 * (LenUser + 2); // Third Section of Correct Serial
Section_03 = Section_03 * LenUser; // Second Section of Correct Serial
Section_02 = Section_02 + Temp_06; // First Section of Correct Serial

wsprintf(reaserial,"%u-%u-%u",Section_03,Section_02,Section_01);
}
SetDlgItemText(IDC_Serial,reaserial);

```

**V – End of Tut :**

- Finished – ***14/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a>                            |
| CrackMe :     | <b>yoda - lame crackme - Level 0</b>                                           |
| Coder :       | <b>yoda</b>                                                                    |
| Cracker :     | <b>Moonbaby</b>                                                                |
| Type :        | Name / Serial                                                                  |
| Packed :      | <b>UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -&gt; Markus &amp; Laszlo</b> |
| Language :    | Borland Delphi 2.0                                                             |
| Crack Tool :  | <b>OllyDbg 1.09d, PeID 0.92</b>                                                |
| Unpack Tool : | <b>Manual</b>                                                                  |
| Request :     | Correct Serial / KeyGen                                                        |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo**. Sau khi UnPACK và kiểm tra lại biết được CrackMe được viết bằng **Borland Delphi 2.0**

- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm được chuỗi "You made it ! Now u only have 2 make a keygen for it ;)" ở địa chỉ :

004258D6 . B8 28594200 MOV EAX,crkme1\_y.00425928 ; |ASCII "You made it ! Now u only have 2 make a keygen for it ;)"

- Đặc biệt, ở CrackMe này ta không đi theo thông báo "Wrong serial !". Vì sẽ bị ròi và đồng thời sẽ không dẫn đến đâu cả.

004259E0 . 68 805A4200 PUSH crkme1\_y.00425A80 ; |Text = "Wrong serial !"

- Dò ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :

0042580E . E8 E5C1FEFF CALL crkme1\_y.004119F8 ; <== Set BreakPoint here

### II – Cracking :

- Nhấn F9 để chạy CrackMe. Ta không nhập User mà gõ U ở một chỗ khác rồi COPY vào. Ngay lập tức CrackMe sẽ dừng lại tại điểm đặt BP. Quá trình kiểm tra chiều dài chuỗi S nhập :

00425816 . E8 49DBFDFF CALL crkme1\_y.00403364

0042581B . 83F8 04 CMP EAX,4 ; <== Len.U atleast 5 charts

0042581E . 0F8E 8C000000 JLE crkme1\_y.004258B0 ; <== JMP nag if Less or Equal

- Quá trình mã hoá của CrackMe này gồm hai giai đoạn. Giai đoạn thứ nhất là thực hiện phép toán đổi với 4 vị trí của chuỗi U nhập :

00425843 . 0FB630 MOVZX ESI,BYTE PTR DS:[EAX] ; <== U[0]

00425846 . 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]

00425849 . 8B83 AC010000 MOV EAX,DWORD PTR DS:[EBX+1AC]

0042584F . E8 A4C1FEFF CALL crkme1\_y.004119F8

00425854 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10]

00425857 . 0FB678 01 MOVZX EDI,BYTE PTR DS:[EAX+1] ; <== U[1]

0042585B . 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]

0042585E . 8B83 AC010000 MOV EAX,DWORD PTR DS:[EBX+1AC]

00425864 . E8 8FC1FEFF CALL crkme1\_y.004119F8

00425869 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10]

0042586C . 0FB640 03 MOVZX EAX,BYTE PTR DS:[EAX+3] ; <== U[3]

```

00425870 . 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX
00425873 . 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]
00425876 . 8B83 AC010000 MOV EAX,DWORD PTR DS:[EBX+1AC]
0042587C . E8 77C1FEFF CALL crkme1_y.004119F8
00425881 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10]
00425884 . 0FB640 04 MOVZX EAX,BYTE PTR DS:[EAX+4] ; <== U[4]
00425888 . 03FE ADD EDI,ESI ; <== Temp_01 = U[1] + U[0]
0042588A . 0345 F8 ADD EAX,DWORD PTR SS:[EBP-8] ; <== Temp_02 = U[4] + U[3]
0042588D . 0FAFF8 IMUL EDI,EAX ; <== Temp_01 = Temp_01 * Temp_02
00425890 . 8BF7 MOV ESI,EDI ; <== Temp_02

```

- Giai đoạn thứ hai CrackMe sẽ lấy PATH của CrackMe, tính chiều dài của nó và nhân với giá trị vừa tính được ở trên :

```

0042589A . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== Full Path Name
0042589D . E8 C2DAFDFF CALL crkme1_y.00403364 ; <== Len of Full Path Name
004258A2 . F7EE IMUL ESI ; <== Temp_02 = Temp_02 * Len.path
004258A4 . 8BF0 MOV ESI,EAX ; <== Temp_02
004258A6 . 8D55 F4 LEA EDX,DWORD PTR SS:[EBP-C]
004258A9 . 8BC6 MOV EAX,ESI ; <== Temp_02
004258AB . E8 68FBFDFF CALL crkme1_y.00405418 ; <== Convert to DEC value

```

- Giá trị cuối cùng này được chuyển dạng dạng DEC, đây chính là Serial thực của CrackMe.

/\*/\*/\* - SERIAL :

|                     |                             |
|---------------------|-----------------------------|
| User : REA-cRaCkErS | Serial : 1956960 (Len = 90) |
| User : VHT-cRaCkErS | Serial : 2047680 (Len = 90) |

### III – KeyGen :

```

/Section 1/- Temp_01 = (U[0] + U[1]) * (U[3] + U[4])
/Section 2/- Serial = Temp_01 * Len.path
/Section 3/- Chuyển đổi sang giá trị ở dạng DEC

```

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaFullPath[255]="";
int reaSeial=0;
int LenUser=0,LenPath=0;

```

```

LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 5)
{
 MessageBox("Your name atleast 5 charts ","Hey !! Please input your name again !! ");
}
else
{
 HANDLE KeyFile =
CreateFile("crkme1_y.exe",GENERIC_READ,FILE_SHARE_READ,0,OPEN_EXISTING,FILE_ATTRIBUTE_ARCHIVE,NULL);
 if (KeyFile == INVALID_HANDLE_VALUE)
 {
 MessageBox("Put KeyGen at the same place with CrackMe \n\n =*=*= CrackMe :
crkme1_y.exe =*=*=","Hey !!! Please Copy me !!");
 }
}

```

```

 else
 {
 LenPath = GetFullPathName("crkme1_y.exe", 255, reaFullPath, NULL) - 13;
 reaSeial = ((reaName[0] + reaName[1]) * (reaName[3] + reaName[4])) * LenPath;
 SetDlgItemInt(IDC_Serial, reaSeial);
 }
 CloseHandle(KeyFile);
 }
}

```

**V – End of Tut :**

- Finished – ***15/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <b>Zephyrous - crackme #1 - Level 1</b>             |
| Coder :       | <b>Zephyrous</b>                                    |
| Cracker :     | <b>Moonbaby</b>                                     |
| Type :        | Name / Serial                                       |
| Packed :      | <b>N/A</b>                                          |
| Language :    | <b>Microsoft Visual C++ 6.0</b>                     |
| Crack Tool :  | <b>OllyDbg 1.09d, PeiD 0.92</b>                     |
| Unpack Tool : | <b>N/A</b>                                          |
| Request :     | Correct Serial / KeyGen                             |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **Microsoft Visual C++ 6.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta không tìm thấy chuỗi thông báo đúng hay sai, nhưng tìm được thông báo "Name Must Be More Than 3 Chars" ở địa chỉ :  
004010C5 |. 68 6C504000 PUSH keygenme.0040506C ; ASCII "Name Must Be More Than 3 Chars"  
- Từ đây ta biết được yêu cầu về chiều dài của U :  
004010C0 |> \83FF 03 CMP EDI,3 ; <== Name atleast 4 charts  
004010C3 |. 7F 07 JG SHORT keygenme.004010CC  
004010C5 |. 68 6C504000 PUSH keygenme.0040506C ; ASCII "Name Must Be More Than 3 Chars"
- Cũng như chiều dài tối thiểu của S :  
004010E5 |> \837D 08 03 CMP [ARG.1],3 ; <== Serrial atleast 4 charts  
004010E9 |. 7F 13 JG SHORT keygenme.004010FE  
004010EB |. 68 38504000 PUSH keygenme.00405038 ; ASCII "Serial Must Be More Than 3 Chars"
- Từ đó ta tìm ra điểm đặt BreakPoint :  
004010FE |> \33D2 XOR EDX,EDX ; <== Set BP here

**II – Cracking :**

- Nhấn F9 để chạy CrackMe. Nhập U và Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. Quá trình mã hoá chuỗi U của CrackMe này bao gồm 6 giai đoạn .

- Giai đoạn thứ nhất :

```

00401104 > /8A4C15 94 /MOV CL,BYTE PTR SS:[EBP+EDX-6C] ; <== U[i]
00401108 |. |OFBEC1 |MOVSX EAX,CL ; <== Temp = Temp + U[i]
0040110B |. |0145 FC |ADD [LOCAL.1],EAX ; <== Temp
0040110E |. |80F9 4A |CMP CL,4A ; <== if (U[i] >= 0x4A)
00401111 |. |7C 06 |JL SHORT keygenme.00401119 ; <== then
00401113 |. |0FAF45 FC |IMUL EAX,[LOCAL.1] ; <== Temp = Temp * U[i]
00401117 |. |EB 06 |JMP SHORT keygenme.0040111F ; <== else { if (U[i] < 0x4A) }
00401119 > |0FAF45 FC |IMUL EAX,[LOCAL.1] ; <== Temp = Temp * U[i]
0040111D |. |D1E0 |SHL EAX,1 ; <== Temp = Temp * 2
0040111F > |42 |INC EDX ; <== i++
00401120 |. |8945 FC |MOV [LOCAL.1],EAX ; <== Temp
00401123 |. |3BD7 |CMP EDX,EDI ; <== If (i < Len.U)
00401125 |.^|7C DD |JL SHORT keygenme.00401104 ; <== Continue Loop
00401127 > |8B45 FC |MOV EAX,[LOCAL.1] ; <== Temp
0040112A |. |33D2 |XOR EDX,EDX
0040112C |. |B9 24070000 |MOV ECX,724 ; <== DefaultValue : DefV
00401131 |. |F7F1 |DIV ECX ; <== Temp_01 = Temp % DefV

```

- Quá trình mã hoá thứ hai :

```

00401139 > /8A5C05 94 /MOV BL,BYTE PTR SS:[EBP+EAX-6C] ; <== U[i]
0040113D |. |0FBECB |MOVSX ECX,BL ; <== U[i]
00401140 |. |03D1 |ADD EDX,ECX ; <== Temp = Temp + U[i]
00401142 |. |80FB 40 |CMP BL,40 ; <== if (U[i] >= 0x40)
00401145 |. |7C 05 |JL SHORT keygenme.0040114C ; <== then
00401147 |. |0FAFCA |IMUL ECX,EDX ; <== Temp = Temp * U[i]
0040114A |. |EB 06 |JMP SHORT keygenme.00401152 ; <== else { if (U[i] < 0x40) }
0040114C > |0FAFCA |IMUL ECX,EDX ; <== Temp = Temp * U[i]
0040114F |. |C1E1 02 |SHL ECX,2 ; <== Temp = Temp * 4
00401152 > |40 |INC EAX ; <== i++
00401153 |. |8BD1 |MOV EDX,ECX ; <== Temp_01
00401155 |. |3BC7 |CMP EAX,EDI ; <== If (i < Len.U)
00401157 |.^|7C E0 |JL SHORT keygenme.00401139 ; <== Continue Loop
00401159 > |8BC2 |MOV EAX,EDX ; <== Temp
0040115B |. |33D2 |XOR EDX,EDX
0040115D |. |B9 25220000 |MOV ECX,2225 ; <== DefaultValue : DefV
00401162 |. |33DB |XOR EBX,EBX
00401164 |. |F7F1 |DIV ECX ; <== Temp = Temp % DefV

```

- Quá trình mã hoá thứ ba :

```

0040116C > /8A4435 94 /MOV AL,BYTE PTR SS:[EBP+ESI-6C] ; <== U[i]
00401170 |. |0FBEC8 |MOVSX ECX,AL ; <== U[i]
00401173 |. |03D1 |ADD EDX,ECX ; <== Temp = Temp + U[i]
00401175 |. |3C 54 |CMP AL,54 ; <== if (U[i] >= 0x54)
00401177 |. |7C 07 |JL SHORT keygenme.00401180 ; <== then
00401179 |. |0FAFCA |IMUL ECX,EDX ; <== Temp = Temp * U[i]
0040117C |. |8BD1 |MOV EDX,ECX ; <== Temp
0040117E |. |EB 08 |JMP SHORT keygenme.00401188 ; <== else { if (U[i] < 0x54) }
00401180 > |0FAFCA |IMUL ECX,EDX ; <== Temp = Temp * U[i]
00401183 |. |8D1449 |LEA EDX,DWORD PTR DS:[ECX+ECX*2] ; <== Temp = Temp * 3
00401186 |. |D1E2 |SHL EDX,1 ; <== Temp = Temp * 2
00401188 > |46 |INC ESI ; <== i++
00401189 |. |3BF7 |CMP ESI,EDI ; <== If (i < Len.U)
0040118B |.^|7C DF |JL SHORT keygenme.0040116C ; <== Continue Loop

```

- Quá trình mã hoá thứ tư :

```

00401193 > /8A4435 94 /MOV AL,BYTE PTR SS:[EBP+ESI-6C] ; <== U[i]
00401197 |. |0FBEC8 |MOVSX ECX,AL ; <== U[i]
0040119A |. |03D1 |ADD EDX,ECX ; <== Temp = Temp + U[i]
0040119C |. |3C 4A |CMP AL,4A ; <== if (U[i] >= 0x4A)
0040119E |. |7C 07 |JL SHORT keygenme.004011A7 ; <== then
004011A0 |. |0FAFCA |IMUL ECX,EDX ; <== Temp = Temp * U[i]
004011A3 |. |8BD1 |MOV EDX,ECX ; <== Temp
004011A5 |. |EB 06 |JMP SHORT keygenme.004011AD ; <== else { if (U[i] < 0x4A) }
004011A7 > |0FAFCA |IMUL ECX,EDX ; <== Temp_01 = Temp * U[i]
004011AA |. |8D1489 |LEA EDX,DWORD PTR DS:[ECX+ECX*4] ; <== Temp = Temp * 5
004011AD > |46 |INC ESI ; <== i++
004011AE |. |3BF7 |CMP ESI,EDI ; <== If (i < Len.U)
004011B0 |.^|7C E1 |JL SHORT keygenme.00401193 ; <== Continue Loop
004011B2 > |8BC2 |MOV EAX,EDX ; <== Temp
004011B4 |. |33D2 |XOR EDX,EDX
004011B6 |. |B9 342E0000 |MOV ECX,2E34 ; <== DefaultValue : DefV
004011BB |. |33F6 |XOR ESI,ESI
004011BD |. |F7F1 |DIV ECX ; <== Temp = Temp % DefV

```

- Quá trình mã hoá thứ năm :

```

004011C3 > /8A4C35 94 /MOV CL,BYTE PTR SS:[EBP+ESI-6C] ; <== U[i]
004011C7 |. |0FBEC1 |MOVSX EAX,CL ; <== U[i]
004011CA |. |03D0 |ADD EDX,EAX ; <== Temp = Temp + U[i]
004011CC |. |80F9 40 |CMP CL,40 ; <== if (U[i] >= 0x40)
004011CF |. |7C 05 |JL SHORT keygenme.004011D6 ; <== then
004011D1 |. |0FAFC2 |IMUL EAX,EDX ; <== Temp = Temp * U[i]
004011D4 |. |EB 06 |JMP SHORT keygenme.004011DC ; <== else { if (U[i] < 0x40) }
004011D6 > |0FAFC2 |IMUL EAX,EDX ; <== Temp = Temp * U[i]
004011D9 |. |6BC0 07 |IMUL EAX,EAX,7 ; <== Temp = Temp * 7
004011DC > |46 |INC ESI ; <== i++
004011DD |. |8BD0 |MOV EDX,EAX ; <== Temp
004011DF |. |3BF7 |CMP ESI,EDI ; <== If (i < Len.U)
004011E1 |.^|7C E0 |JL SHORT keygenme.004011C3 ; <== Continue Loop
- Quá trình mã hoá thứ sáu cũng là quá trình hình thành chuỗi Serial thực theo định dạng "%X%lu"
004011E3 > |52 |PUSH EDX ; /<%lu>
004011E4 |. |52 |PUSH EDX ; |<%X>
004011E5 |. |8D85 CCFFFE LEA EAX,[LOCAL.77] ; |
004011EB |. |68 30504000 PUSH keygenme.00405030 ; |Format = "%X%lu"
004011F0 |. |50 |PUSH EAX ; |s
004011F1 |. |FF15 B0404000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

/\*/\*/\* - SERIAL :

User : REA-cRaCkErS Serial : 820DCC712181942385

User : VHT-cRaCkErS Serial : A49714712761364593

### III – KeyGen :

- /Section 1/- Quá trình tính toán thứ nhất cho ta thông số cuối cùng là Temp
- /Section 2/- Giá trị của Temp được tính cho quá trình thứ hai. Kết quả của quá trình này là Temp
- /Section 3/- Giá trị của Temp được sử dụng cho tất cả các quá trình tính toán liền sau.
- /Section 4/- Giá trị cuối cùng của Temp được chuyển thành chuỗi theo định dạng "%X%lu"

**IV – SourceCode ( VC++ ) :**

```

char reaName[64]="";
char reaSerial[64]="";
int LenUser=0,DefaultValue=0;
int i=0;
long int Temp=0;
 LenUser=GetDlgItemText(IDC_Name,reaName,64);
 if (LenUser < 4)
 {
 MessageBox("Your name atleast 4 charts ","Hey !! Please input your name again !! ");
 }

 else
 {
 i=0;
 while (i < LenUser)
 {
 Temp = Temp + reaName[i];
 if (reaName[i] >= 0x4A)
 {
 Temp = Temp * reaName[i];
 i++;
 }
 else
 {
 Temp = Temp * reaName[i];
 Temp = Temp * 2;
 i++;
 }
 }
 DefaultValue = 0x724;
 Temp = Divided(Temp,DefaultValue);
 i=0;
 while (i < LenUser)
 {
 Temp = Temp + reaName[i];
 if (reaName[i] >= 0x40)
 {
 Temp = Temp * reaName[i];
 i++;
 }
 else
 {
 Temp = Temp * reaName[i];
 Temp = Temp * 4;
 i++;
 }
 }
 DefaultValue = 0x2225;
 Temp = Divided(Temp,DefaultValue);
 i=0;
 while (i < LenUser)
 }

```

```

{
 Temp = Temp + reaName[i];
 if (reaName[i] >= 0x54)
 {
 Temp = Temp * reaName[i];
 i++;
 }
 else
 {
 Temp = Temp * reaName[i];
 Temp = Temp + Temp * 2;

 Temp = Temp * 2;
 i++;
 }
}

i=0;
while (i < LenUser)
{
 Temp = Temp + reaName[i];
 if (reaName[i] >= 0x4A)
 {
 Temp = Temp * reaName[i];
 i++;
 }
 else
 {
 Temp = Temp * reaName[i];
 Temp = Temp + Temp * 4;
 i++;
 }
}

DefaultValue = 0x2E34;
Temp = Divided(Temp,DefaultValue);

i=0;
while (i < LenUser)
{
 Temp = Temp + reaName[i];
 if (reaName[i] >= 0x40)
 {
 Temp = Temp * reaName[i];
 i++;
 }
 else
 {
 Temp = Temp * reaName[i];
 Temp = Temp * 7;
 i++;
 }
}

```

```

 }

 wsprintf(reaSerial,"%X%lu",Temp,Temp);

 SetDlgItemText(IDC_Serial,reaSerial);
 }

```

**V – End of Tut :**

- Finished – ***15/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <a href="#">Zephyrous - crackme #2 - Level 2</a>    |
| Coder :       | <a href="#">Zephyrous</a>                           |
| Cracker :     | <a href="#">Moonbaby</a>                            |
| Type :        | Serial                                              |
| Packed :      | N/A                                                 |
| Language :    | Microsoft Visual C++ 6.0                            |
| Crack Tool :  | <a href="#">OllyDbg 1.09d, PeiD 0.92</a>            |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Serial / KeyGen                             |

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **Microsoft Visual C++ 6.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Not exactly as i want ;p" ở địa chỉ :  
0040125D . 68 30504000 PUSH keygenme.00405030 ; ASCII "Not exactly as i want ;p"
- Truy ngược lên trên ta gặp hàm **GetDlgItemTextA** nên đặt BreakPoint tại đây :  
00401076 . FF15 A4404000 CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA>;\GetDlgItemTextA

**II – Cracking :**

- Nhấn F9 để chạy CrackMe. Nhập Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. CrackMe này không có quá trình kiểm tra chiều dài chuỗi S nhập. Tuy nhiên, qua quá trình kiểm tra từng ký tự nhập vào vòng lặp là 20 lần, nếu chuỗi Fake Serial nhập đủ 20 thì chương trình tự gán các chuỗi ngẫu nhiên.

```

00401080 >/8A0C38 MOV CL,BYTE PTR DS:[EAX+EDI] ; <== S[i]
00401083 .|33D2 XOR EDX,EDX
00401085 >|80F9 39 CMP CL,39
00401088 .|7F 08 JG SHORT keygenme.00401092
0040108A .|80F9 30 CMP CL,30
0040108D .|7C 03 JL SHORT keygenme.00401092
0040108F .|6A 01 PUSH 1
00401091 .|5B POP EBX
00401092 >|83FB 01 CMP EBX,1

```

```

00401095 . |74 06 JE SHORT keygenme.0040109D
00401097 . |42 INC EDX
00401098 . |83FA 06 CMP EDX,6
0040109B . ^|72 E8 JB SHORT keygenme.00401085
0040109D > |40 INC EAX ; <== i++
0040109E . |83F8 14 CMP EAX,14 ; <== Number of Loop : 20 times
004010A1 . ^|72 DD JB SHORT keygenme.00401080
- Quá trình mã hoá chuỗi Fake Serial nhập được chia làm nhiều đoạn nhưng lại có cách xử lý giống nhau .
Tựu chung khi hiểu được một đoạn thì các đoạn khác là như nhau . Kết quả của mỗi quá trình là NULL.
Để hiểu tại sao kết quả của mỗi quá trình phải là NULL ta đi ngược lại. Xét quá trình so sánh chuỗi :
004011FC . 837D 10 00 CMP DWORD PTR SS:[EBP+10],0 ; <== Saved Value must be NULL
00401200 . 6A 15 PUSH 15
00401202 . C745 0C 20000>MOV DWORD PTR SS:[EBP+C],20
00401209 . 5E POP ESI
0040120A . 75 0E JNZ SHORT keygenme.0040121A
0040120C . 837D 14 00 CMP DWORD PTR SS:[EBP+14],0 ; <== Saved Value must be NULL
00401210 . 75 08 JNZ SHORT keygenme.0040121A
00401212 . 837D FC 00 CMP DWORD PTR SS:[EBP-4],0 ; <== Saved Value must be NULL
00401216 . 75 02 JNZ SHORT keygenme.0040121A
00401218 . 33F6 XOR ESI,ESI
0040121A > 837D F8 00 CMP DWORD PTR SS:[EBP-8],0 ; <== Saved Value must be NULL
0040121E . 75 0C JNZ SHORT keygenme.0040122C
00401220 . 85D2 TEST EDX,EDX ; <== Saved Value must be NULL
00401222 . 75 17 JNZ SHORT keygenme.0040123B
00401224 . 3955 F4 CMP DWORD PTR SS:[EBP-C],EDX ; <== Saved Value must be NULL
00401227 . 75 07 JNZ SHORT keygenme.00401230
00401229 . 2155 10 AND DWORD PTR SS:[EBP+10],EDX
0040122C > 85D2 TEST EDX,EDX ; <== Saved Value must be NULL
0040122E . 75 0B JNZ SHORT keygenme.0040123B
00401230 > 85C9 TEST ECX,ECX ; <== Saved Value must be NULL
00401232 . 75 07 JNZ SHORT keygenme.0040123B
00401234 . 85C0 TEST EAX,EAX ; <== Saved Value must be NULL
00401236 . 75 03 JNZ SHORT keygenme.0040123B
00401238 . 2145 0C AND DWORD PTR SS:[EBP+C],EAX
0040123B > 85F6 TEST ESI,ESI ; <== Saved Value must be NULL
0040123D . 75 17 JNZ SHORT keygenme.00401256
0040123F . 3975 0C CMP DWORD PTR SS:[EBP+C],ESI ; <== Saved Value must be NULL
00401242 . 75 12 JNZ SHORT keygenme.00401256
00401244 . 3975 10 CMP DWORD PTR SS:[EBP+10],ESI ; <== Saved Value must be NULL
00401247 . 75 0D JNZ SHORT keygenme.00401256
00401249 . 56 PUSH ESI
0040124A . 68 8C504000 PUSH keygenme.0040508C ; ASCII "Good Boy!"
0040124F . 68 58504000 PUSH keygenme.00405058 ; ASCII "Huh... you're good in
reversing code Well done!"
```

- Ta nhận xét ở đoạn CODE này. Để đến được thông báo đúng thì cả ba lệnh so sánh cuối cùng phải thoả .

/+/- Lệnh so sánh thứ nhất : TEST ESI,ESI : để có được giá trị ESI = 00 thì ta phải có được câu lệnh ở bên trên nó XOR ESI,ESI . Có nghĩa là 3 câu lệnh so sánh bên trên câu lệnh này phải thoả . Từ đó truy ngược lại địa chỉ các giá trị được lưu, tất cả phải là NULL.

/+/- Lệnh so sánh thứ hai : CMP DWORD PTR SS:[EBP+C],ESI : đến đoạn này thì ESI = 00 . Vậy địa chỉ lưu giá trị cũng phải là NULL. Mà ban đầu giá trị tại địa chỉ này được gán là 0x20 ( xem địa chỉ 00401202 ), sau đó tại địa chỉ 00401238 này EAX được gán vào để thay thế cho 0x20. Như vậy ta cần

EAX = 00. Hay nói cách khác, để có được EAX = NULL thì bản thân lệnh so sánh EAX và hai câu lệnh so sánh bên trên nó cũng phải có kết quả thoả . Hay các kết quả lưu phải là NULL.

/+/- Lệnh so sánh thứ ba : CMP DWORD PTR SS:[EBP+10],ESI : như vậy giá trị được lưu tại địa chỉ này cũng phải là NULL.

/+/- Kiểm tra lại tất cả các địa chỉ thì toàn bộ giá trị tính toán đều phải cho kết quả NULL.

- Ở đây ta xét đoạn đầu tiên :

```

004010A3 . 8A07 MOV AL,BYTE PTR DS:[EDI] ; <== S[0]
004010A5 . B1 5A MOV CL,5A ; <== "Z"
004010A7 . 3AC1 CMP AL,CL
004010A9 . 7F 1C JG SHORT keygenme.004010C7
004010AB . 3C 41 CMP AL,41 ; <== "A"
004010AD . 7C 18 JL SHORT keygenme.004010C7
004010AF . 0FBF0 MOVSX ESI,AL
004010B2 . 83C6 04 ADD ESI,4 ; <== S[0] + 4
004010B5 . 83FE 5B CMP ESI,5B ; <== if (S[0] + 4 > 0x5B)
004010B8 . 7E 05 JLE SHORT keygenme.004010BF ; <== then
004010BA . 83EE 06 SUB ESI,6 ; <== (S[0] + 4) - 0x6
004010BD . EB 08 JMP SHORT keygenme.004010C7 ; <== else
004010BF > 83FE 40 CMP ESI,40 ; <== S[0] + 4
004010C2 . 7D 03 JGE SHORT keygenme.004010C7
004010C4 . 83C6 1D ADD ESI,1D ; <== No meaning - DEL
004010C7 > 0FBF47 08 MOVSX EAX,BYTE PTR DS:[EDI+8] ; <== S[8]
004010CB . 33C6 XOR EAX,ESI ; <== S[0] + 4 or (S[0] + 4 - 6) == S[8]
004010CD . B3 7A MOV BL,7A ; <== "z"
004010CF . 8945 10 MOV DWORD PTR SS:[EBP+10],EAX ; <== Value saved must be
NULL

```

- Diễn giải đoạn CODE thứ nhất :

/+/- Kết quả của quá trình này được lưu ở EAX và sau đó được lưu vào địa chỉ SS:[EBP+10] . Như vậy giá trị EAX cần phải là NULL.

/+/- Để EAX là NULL thì kết quả của phép toán XOR EAX,ESI là NULL hay nói cách khác EAX = ESI. Mà EAX đại diện cho ký tự thứ 9 của chuỗi Fake Serial (DS:[EDI+8]) .

/+/- Ta đi ngược lại quá trình, từ trên xuống . Đoạn CODE từ địa chỉ 004010A3 đến 004010AD sẽ so sánh ký tự thứ nhất của chuỗi Fake Serial (DS:[EDI]) xem nó có nằm trong khoảng từ “A”-“Z” hay không. Nếu không nằm trong khoảng này thì sẽ nhảy đến địa chỉ 004010C7 ( gán EAX == S[8] ) và sau đó là thực hiện phép toán XOR với ESI = 00. Kết quả luôn là EAX != NULL. Như vậy giá trị S[0] phải là một ký tự trong khoảng từ “A”-“Z”.

/+/- Nếu ta có được S[0] thuộc “A”-“Z” thì sẽ thực hiện phép cộng với 0x4. Ở đây ta có hai điều kiện (1) nếu giá trị này lớn hơn 0x5B thì sẽ bị trừ đi 0x6 sau đó thực hiện phép XOR với S[8], như vậy giá trị của S[8] = (S[0] + 0x4) - 0x6 (2) nếu giá trị nhỏ hơn hay bằng thì sẽ chuyển đé quá trình so sánh thứ hai với giá trị là 0x40 . Ở đây ta chú ý, vì S[0] thuộc “A”-“Z” nên sau khi được cộng thêm với 4 thì luôn luôn lớn hơn 0x40 (@) . Vì thế luôn thực hiện lệnh nhảy . Lúc này S[8] = S[0] + 0x4

- Tương tự như thế cho các đoạn còn lại . Và ta nhận thấy CrackMe sử dụng 16 ký tự. Hay nói cách khác, chiều dài tối thiểu của Serial là 16 ký tự.

/\*/\*/\* - SERIAL :

User : REA-cRaCkErS

Serial : N/A

User : VHT-cRaCkErS

Serial : N/A

Serial : LgNySpKjPnTuWtPl /- PqFlNvKdTxLqRzPf

**III – KeyGen :**

/Section 1/- Tạo lập 8 ký tự đầu tiên theo đúng dạng (UpperCase hay LowerCase ) một cách ngẫu nhiên .

/Section 2/- Tính toán 8 ký tự sau theo 8 ký tự ban đầu thoả các điều kiện CrackMe yêu cầu.

#### IV – SourceCode ( VC++ ) :

```

char reaSerial[64] ="";
char reaLowerCase[27] ="abcdefghijklmnopqrstuvwxyz";
char reaUpperCase[27] ="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char reaTempUpper[20] ="";
char reaTempLower[20] ="";
int i=0;
srand((unsigned)time(NULL));

i=0;
while (i < 8)
{
 reaTempUpper[i] = reaUpperCase[rand() % 26];
 i+=2;
}
i=1;
while (i < 8)
{
 reaTempUpper[i] = reaLowerCase[rand() % 26];
 i+=2;
}
lstrcat(reaSerial,reaTempUpper);
lstrcat(reaSerial,reaTempLower);
if ((reaSerial[0] + 4) > 0x5B)
{
 reaSerial[8] = (reaSerial[0] + 4) - 0x6;
}
else
{
 reaSerial[8] = reaSerial[0] + 4;
}
if ((reaSerial[1] + 7) > 0x7B)
{
 reaSerial[9] = (reaSerial[1] + 7) - 0xB;
}
else
{
 reaSerial[9] = reaSerial[1] + 7;
}
if ((reaSerial[2] + 6) > 0x5B)
{
 reaSerial[10] = (reaSerial[2] + 6) - 0xA;
}
else
{
 reaSerial[10] = reaSerial[2] + 6;
}
if ((reaSerial[3] + 5) > 0x7B)

```

```

{
 reaSerial[11] = (reaSerial[3] + 5) - 0x9;
}
else
{
 reaSerial[11] = reaSerial[3] + 5;
}
if ((reaSerial[4] + 4) > 0x5B)
{
 reaSerial[12] = (reaSerial[4] + 4) - 0x6;
}
else
{
 reaSerial[12] = reaSerial[4] + 4;
}
if ((reaSerial[5] + 4) > 0x7B)
{
 reaSerial[13] = (reaSerial[5] + 4) - 0x5;
}
else
{
 reaSerial[13] = reaSerial[5] + 4;
}
if ((reaSerial[6] + 5) > 0x5B)
{
 reaSerial[14] = (reaSerial[6] + 5) - 0x8;
}
else
{
 reaSerial[14] = reaSerial[6] + 5;
}
if ((reaSerial[7] + 2) > 0x7B)
{
 reaSerial[15] = (reaSerial[7] + 2) - 0x8;
}
else
{
 reaSerial[15] = reaSerial[7] + 2;
}
SetDlgItemText(IDC_Serial, reaSerial);

```

**V – End of Tut :**

- Finished – ***15/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <a href="#">Zephyrous - crackme #3 - Level 3</a>    |
| Coder :       | <a href="#">Zephyrous</a>                           |
| Cracker :     | <a href="#">Moonbaby</a>                            |
| Type :        | Serial                                              |
| Packed :      | N/A                                                 |
| Language :    | Microsoft Visual C++ 6.0                            |
| Crack Tool :  | <a href="#">OllyDbg 1.09d, PeID 0.92</a>            |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Serial / KeyGen                             |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **Microsoft Visual C++ 6.0**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Please Try Again! :(" ở địa chỉ :  
004012D5 . 68 30604000 PUSH kgme3.00406030 ; ASCII "Please Try Again! :("
- Truy ngược lên trên ta gặp hàm **GetDlgItemTextA** nên đặt BreakPoint tại đây :  
004011C4 . FFD3 CALL EBX ; \GetDlgItemTextA

### II – Cracking :

- Nhấn F9 để chạy CrackMe. Nhập Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. Ở đây ta chú ý đến hai vấn đề (1) Có 4 ô nhập, mỗi ô nhập có tối đa 4 giá trị (2) các ký tự của ô nhập đều ở dạng UpperCase. Điều này rất qua trọng vì ảnh hưởng trực tiếp đến quá trình chuyển đổi từ chuỗi sang dạng giá trị HEX tương ứng. Quá trình chuyển đổi này chúng ta gặp ở các lệnh **CALL kgme3.00401000**. Quá trình này được thực hiện như sau :

```

0040101D |. BF 04000000 MOV EDI,4 ; <== Numbers of Loop
00401022 |. 8B75 FC MOV ESI,[LOCAL.1]
00401025 |> 8A0E /MOV CL,BYTE PTR DS:[ESI] ; <== Ch[i]
00401027 |. 80F9 30 |CMP CL,30 ; <== if (0x30 < Ch[i])
0040102A |. 7C 0F |JL SHORT kgme3.0040103B ; <== and
0040102C |. 80F9 39 |CMP CL,39 ; <== if (Ch[i] < 0x39)
0040102F |. 7F 05 |JG SHORT kgme3.00401036 ; <== then
00401031 |. 80E9 30 |SUB CL,30 ; <== Temp = Ch[i] - 0x30
00401034 |. EB 05 |JMP SHORT kgme3.0040103B ; <== else if (Ch[i] > 0x39)
00401036 |> 80E9 37 |SUB CL,37 ; <== Temp = Ch[i] - 0x37
00401039 |. EB 00 |JMP SHORT kgme3.0040103B ; <== else (Ch[i] < 0x30)
0040103B |> 03C1 |ADD EAX,ECX ; <== Temp = Ch[i]
0040103D |. 83FF 01 |CMP EDI,1
00401040 |. 74 03 |JE SHORT kgme3.00401045
00401042 |. C1E0 04 |SHL EAX,4 ; <== Temp = Temp * 0x10
00401045 |> 46 |INC ESI ; <== i++
00401046 |. 4F |DEC EDI ; <== if (i < 4)
00401047 |.^ 75 DC |JNZ SHORT kgme3.00401025 ; <== Continue Loop

```

- Sau khi kết thúc thì giá trị cuối cùng Temp được thực hiện thêm một phép tính nhỏ nhằm lấy 4 giá trị cuối cùng được lưu ở EAX ( ví dụ : nếu giá trị của EAX = 12345678 thì sau khi thực hiện phép toán AND với 0xFFFF thì giá trị của EAX sẽ là 5678 ) :

```
00401222 . 8D4D CC LEA ECX,DWORD PTR SS:[EBP-34] ; <== Temp
00401225 . 25 FFFF0000 AND EAX,0FFFF ; <== Temp = Temp & 0xFFFF
```

- Quá trình này được viết lại bằng một Function như sau (VC++) :

```
int STRINGtoHEX (char reaInputStream[5] "")
```

```
{
 CharUpper(reaInputStream);
 int i=0, Temp_01=0, Temp_02=0;
 while (i < 4)
 {
 if (reaInputStream[i] >= 0x30 && reaInputStream[i] <= 0x39)
 {
 Temp_01 = reaInputStream[i] - 0x30;
 i++;
 }
 else if (reaInputStream[i] < 0x30)
 {
 Temp_01 = reaInputStream[i];
 i++;
 }
 else
 {
 Temp_01 = reaInputStream[i] - 0x37;
 i++;
 }
 if (i < 4)
 {
 Temp_02 = Temp_02 + Temp_01;
 Temp_02 = Temp_02 * 16;
 }
 else if (i == 4)
 {
 Temp_02 = (Temp_02 + Temp_01) & 0xFFFF;
 break;
 }
 else
 {
 break;
 }
 }
 return Temp_02;
}
```

- CrackMe này đầu tiên mã hoá chuỗi U nhập và theo quy tắc sau :

```
004011CF . 0FBE0C02 MOVSX ECX,BYTE PTR DS:[EDX+EAX] ; <== U[i]
004011D3 . 034D 14 ADD ECX,DWORD PTR SS:[EBP+14] ; <== Temp_01 = Temp_01 + U[i]
004011D6 . 0FAFC8 IMUL ECX,EAX ; <== Temp_01 = Temp_01 * i
004011D9 . 40 INC EAX ; <== i++
004011DA . 894D 14 MOV DWORD PTR SS:[EBP+14],ECX ; <== Temp_01
004011DD . 3BC6 CMP EAX,ESI ; <== if (i < Len.U + 1)
004011DF . ^ 72 EB JB SHORT kgme3.004011CC ; <== Continue Loop
```

- Giá trị này sau đó thực hiện thêm phép toán đơn giản để có được giá trị so sánh đầu tiên :

```
0040128A . B9 0F000000 MOV ECX,0F ; <== DefaultValue = 0xF
0040128F . F7F1 DIV ECX ; <== CheckValue_01 = ValueUser % 0xF
00401291 . 8BF2 MOV ESI,EDX ; <== ESI = CheckValue_01 > 0
00401293 . 8975 0C MOV DWORD PTR SS:[EBP+C],ESI ; <== Value of Second Check
```

- Bốn ô nhập Serial được mã hoá theo thuật toán ở trên. Như vậy ta có 4 giá trị ở dạng HEX tương ứng với 4 chuỗi Serial nhập vào :

```
0040121D . E8 DEFDFFFF CALL kgme3.00401000 ; <== Temp_01
00401222 . 8D4D CC LEA ECX,DWORD PTR SS:[EBP-34] ; <== Temp_01
00401225 . 25 FFFF0000 AND EAX,0FFFF ; <== Temp_01 = Temp_01 & 0xFFFF
0040122A . 51 PUSH ECX
0040122B . 8945 F4 MOV DWORD PTR SS:[EBP-C],EAX
0040122E . E8 CDFDFFFF CALL kgme3.00401000 ; <== Temp_02
00401233 . 8D55 DC LEA EDX,DWORD PTR SS:[EBP-24] ; <== Temp_02
00401236 . 25 FFFF0000 AND EAX,0FFFF ; <== Temp_02 = Temp_02 & 0xFFFF
0040123B . 52 PUSH EDX
0040123C . 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX
0040123F . E8 BCFDFFFF CALL kgme3.00401000 ; <== Temp_03
00401244 . 25 FFFF0000 AND EAX,0FFFF ; |<== Temp_03 = Temp_03 & 0xFFFF
00401249 . 8945 08 MOV DWORD PTR SS:[EBP+8],EAX ; |
0040124C . 8D45 D4 LEA EAX,DWORD PTR SS:[EBP-2C] ; |
0040124F . 50 PUSH EAX ; |Arg1
00401250 . E8 ABFDFFFF CALL kgme3.00401000 ; \<== Temp_04
00401255 . 25 FFFF0000 AND EAX,0FFFF ; <== Temp_04 = Temp_04 & 0xFFFF
```

- Sau đó 4 giá trị này được kết hợp với nhau theo thứ tự (1 – 4) và (3 – 2) theo nguyên tắc : chuỗi thứ I và thứ III được nhân với 0x10000 để trở thành XXXX0000 sau đó cộng tương ứng với IV và II để tạo thành giá trị mới là XXXXYYYY :

```
0040125D . 8945 FC MOV DWORD PTR SS:[EBP-4],EAX
00401260 . 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C] ; <== Temp_01
00401263 . C1E0 10 SHL EAX,10 ; <== Temp_01 = Temp_01 * 0x10000
00401266 . 8B5D F8 MOV EBX,DWORD PTR SS:[EBP-8] ; <== Temp_04
00401269 . 03C3 ADD EAX,EBX ; <== Temp_01 = Temp_01 + Temp_04
0040126B . 8945 EC MOV DWORD PTR SS:[EBP-14],EAX ; <== Value saved
0040126E . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== Temp_03
00401271 . C1E0 10 SHL EAX,10 ; <== Temp_03 = Temp_03 * 0x10000
00401274 . 8B5D 08 MOV EBX,DWORD PTR SS:[EBP+8] ; <== Temp_02
00401277 . 03C3 ADD EAX,EBX ; <== Temp_03 = Temp_03 + Temp_02
00401279 . 8945 F0 MOV DWORD PTR SS:[EBP-10],EAX ; <== Value saved
```

- Sau đó hai giá trị này thực hiện quá trình lặp để cho ra giá trị so sánh thứ hai :

```
00401298 . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== Temp_01
0040129B . 8B55 F0 MOV EDX,DWORD PTR SS:[EBP-10] ; <== Temp_03
0040129E . 33C2 XOR EAX,EDX ; <== Temp_01 = Temp_01 xor Temp_02
004012A0 > D1E8 SHR EAX,1 ; <== Temp_01 = Temp_01 / 2
004012A2 . A8 01 TEST AL,1 ; <== Check value at AL
004012A4 . 74 01 JE SHORT kgme3.004012A7 ; <== if Equal then
004012A6 . 41 INC ECX ; <== CheckValue_02 ++
004012A7 > 85C0 TEST EAX,EAX ; <== if (Temp_01 > 0)
004012A9 .^ 77 F5 JA SHORT kgme3.004012A0 ; <== Continue Loop
```

- Quá trình này đặt biệt chú ý đến câu lệnh **TEST AL,1**. Giá trị cuối cùng thuộc vào điều kiện này .

- Như vậy ta đã có hai giá trị so sánh cần thiết . Giờ ta xét quá trình kiểm tra . Quá trình kiểm tra này bao gồm hai giai đoạn :

/\*/- Giai đoạn thứ nhất : Kiểm tra hai giá trị tính toán có bằng nhau không :

```

004012AB . 8B55 0C MOV EDX,DWORD PTR SS:[EBP+C] ; <== CheckValue_01 & CheckValue_02
004012AE . 3BCA CMP ECX,EDX ; <== If Equal
004012B0 . 0F94C0 SETE AL ; <== EAX = 0x1
004012B3 . 8845 0B MOV BYTE PTR SS:[EBP+B],AL ; <== First check
004012B6 . 807D 0B 01 CMP BYTE PTR SS:[EBP+B],1 ; <== If EAX = 1
004012BA . 75 12 JNZ SHORT kgme3.004012CE ; <== Go to Second Check
/*/- Giai đoạn thứ hai : Kiểm tra giá trị CheckValue_01 : giá trị này buộc phải lớn hơn 0 . Mà
đây là phần dư của phép chia giá trị tính toán của chuỗi U nhập cho 0xF . Hay nói cách khác, phép chia
này phải là phép chia có dư . Như vậy, có thể xảy ra trường hợp một số USER có tên không thoả điều
kiện .
004012BC . 85F6 TEST ESI,ESI ; <== If CheckValue_01 > 0
004012BE . 76 0E JBE SHORT kgme3.004012CE ; <== Congratulations !
/*/*/* - SERIAL :
User : REA-cRaCkErS Serial : URk$ - _K{_- DT:{ - o}#o
User : VHT-cRaCkErS Serial : +KT0 - d=BR - {H7W - z[A

```

### III – KeyGen :

/Section 1/- Tính giá trị của chuỗi U nhập. Tính phần dư của phép chia và đảm bảo là phép chia có dư . Đây là giá trị kiểm tra thứ nhất

/Section 2/- Chọn 4 chuỗi bát ký, mỗi chuỗi 4 ký tự. Chuyển các chuỗi này sang giá trị HEX tương ứng và sau đó chuyển thành hai giá trị HEX theo quy tắc của CrackMe. Tính toán hai giá trị này để có giá trị kiểm tra thứ hai

/Section 2/- So sánh giá trị kiểm tra thứ nhất và thứ hai. Chúng phải bằng nhau.

### IV – SourceCode ( VC++ ) :

```

char reaName[64]="";
char reaSerial[64]="";
char reaTemp[64]="";
char reaRandomString[255]="";
int LenUser=0,DefaultValue=0;
int i=0;
int Temp_01=0,Temp_02=0,Temp_03=0,Temp_04=0;
int Result=0,ValueUser=0;

LenUser=GetDlgItemText(IDC_Name,reaName,64);
srand((unsigned)time(NULL));
i=0;
while (i < 96)
{
 reaRandomString[i] = (char) (32+i);
 i++;
}

if (LenUser < 4)
{
 MessageBox("Your name atleast 4 charts ","Hey !! Please input your name again !! ");
}
else
{
 i=0;
 while (i < LenUser + 1)
 {

```

```

ValueUser = ValueUser + reaName[i];
ValueUser = ValueUser * i;
i++;
}

ValueUser = ValueUser & 0xFF;
ValueUser = ValueUser % 0xF;

if (ValueUser == 0)
{
MessageBox("Name is not accepted for this CrackMe","Please input another name !!");
}
else
{
 Result = 0;

while (Result == 0)
{
 i=0;
 while (i < 16)
 {
 reaSerial[i] = NULL;
 i++;
 }

 i=0;
 while (i < 4)
 {
 reaTemp[i] = reaRandomString[rand() % 96];
 i++;
 }
 lstrcat(reaSerial,reaTemp);
 Temp_01 = STRINGtoHEX(reaTemp);

 i=0;
 while (i < 4)
 {
 reaTemp[i] = reaRandomString[rand() % 96];
 i++;
 }
 lstrcat(reaSerial,"<---/-/--->");
 lstrcat(reaSerial,reaTemp);
 Temp_02 = STRINGtoHEX(reaTemp);

 i=0;
 while (i < 4)
 {
 reaTemp[i] = reaRandomString[rand() % 96];
 i++;
 }
 lstrcat(reaSerial,"<---/-/--->");
 lstrcat(reaSerial,reaTemp);
}

```

```

Temp_03 = STRINGtoHEX(reTemp);

i=0;
while (i < 4)
{
 reTemp[i] = reaRandomString[rand() % 96];
 i++;
}
lstrcat(reSerial,"<---/-/--->");
lstrcat(reSerial,reTemp);
Temp_04 = STRINGtoHEX(reTemp);
Temp_01 = (Temp_01 * 65536) + Temp_04;
Temp_02 = (Temp_03 * 65536) + Temp_02;
Temp_01 = Temp_01 ^ Temp_02;
_asm
{
 MOV EAX,Temp_01
 XOR ECX,ECX
FIRST :
 SHR EAX,0x1
 TEST AL,0x1
 JE SHORT SECOND
 INC ECX
SECOND:
 TEST EAX,EAX
 JA SHORT FIRST
 MOV Temp_01,ECX
}

if (Temp_01 == ValueUser)
{
 Result = 1;
}
else
{
 Result = 0;
}
SetDlgItemText(IDC_Serial,reSerial);
}
}

```

**V – End of Tut :**

- Finished    -    ***16/06/2004***

# Reverse Engineering Association

## CrackMe

|               |                                                     |
|---------------|-----------------------------------------------------|
| Homepage :    | <a href="http://crackmes.de">http://crackmes.de</a> |
| CrackMe :     | <b>Zero - ZeroMakesPiPi - Level 2</b>               |
| Coder :       | <b>Zero</b>                                         |
| Cracker :     | <b>Moonbaby</b>                                     |
| Type :        | Serial                                              |
| Packed :      | N/A                                                 |
| Language :    | <b>Micorsoft Visual C++ 7.0 Method2</b>             |
| Crack Tool :  | <b>OllyDbg 1.09d, PeID 0.92</b>                     |
| Unpack Tool : | N/A                                                 |
| Request :     | Correct Serial                                      |

### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK và được viết bằng **Micorsoft Visual C++ 7.0 Method2**
- Load CrackMe bằng Olly và tìm chuỗi thông báo, ta tìm thấy chuỗi thông báo "Well Done!" ở địa chỉ : 00401418 . 68 20264000 PUSH ZeroMake.00402620 ; ASCII "Well Done!"
- Truy ngược lên trên ta đặt BreakPoint tại đây tại lệnh CALL đầu tiên của Function này : 004013CF . FF15 B0214000 CALL DWORD PTR DS:[<&MFC71.#310>] ; MFC71.ATL::CStringT<wchar\_t,StrTraitMFC\_DLL<wchar\_t,ATL::ChTraitsCRT<wchar\_t>>::CStringT<wchar\_t,StrTraitMFC\_DLL<wchar\_t,ATL::ChTraitsCRT<wchar\_t>>>

### II – Cracking :

- Nhấn F9 để chạy CrackMe. Nhập Fake Serial vào. CrackMe dừng lại tại điểm đặt BP. Ở đây ta chú ý đến các lệnh CALL của CrackMe . Dùng F8 trace và ta đến quá trình chuyển Fake Serial của ta thành số dạng FloatingPoint :

```

004013F8 . 50 PUSH EAX ; /s <== Fake Serial
004013F9 . FF15 8C224000 CALL DWORD PTR DS:[<&MSVCR71.atof>] ; \atof
- Sau đó ta đến quá trình tạo chuỗi :
00401406 . E8 25FFFFFF CALL ZeroMake.00401330 ; <== Trace Into here
- Quá trình này được phân làm hai đoạn .
0040133C |. BF 02000000 MOV EDI,2 ; <== START First Section
00401341 |. C74424 0C 010>MOV DWORD PTR SS:[ESP+C],1
00401349 |. BB 204E0000 MOV EBX,4E20
0040134E |. 8BFF MOV EDI,EDI
00401350 > 83C9 FF /OR ECX,FFFFFFFF
00401353 |. 85FF |TEST EDI,EDI
00401355 |. 8BC7 |MOV EAX,EDI
00401357 |. 7D 02 |JGE SHORT ZeroMake.0040135B
00401359 |. F7D8 |NEG EAX
0040135B > BE 01000000 |MOV ESI,1
00401360 > A8 01 |TEST AL,1
00401362 |. 74 03 ||JE SHORT ZeroMake.00401367

```

```

00401364 |. 0FAFF1 ||IMUL ESI,ECX
00401367 |> D1E8 ||SHR EAX,1
00401369 |. 85C0 ||TEST EAX,EAX
0040136B |. 74 07 ||JE SHORT ZeroMake.00401374
0040136D |. 8BD1 ||MOV EDX,ECX
0040136F |. 0FAFCA ||IMUL ECX,EDX
00401372 |.^ EB EC ||JMP SHORT ZeroMake.00401360
00401374 |> 85FF |TEST EDI,EDI
00401376 |. 7D 0E |JGE SHORT ZeroMake.00401386
00401378 |. B8 01000000 |MOV EAX,1
0040137D |. 99 |CDQ
0040137E |. F7FE |IDIV ESI
00401380 |. 894424 10 |MOV DWORD PTR SS:[ESP+10],EAX
00401384 |. EB 04 |JMP SHORT ZeroMake.0040138A
00401386 |> 897424 10 |MOV DWORD PTR SS:[ESP+10],ESI
0040138A |> DB4424 10 |FILD DWORD PTR SS:[ESP+10]
0040138E |. 8B4C24 0C |MOV ECX,DWORD PTR SS:[ESP+C]
00401392 |. 83C1 02 |ADD ECX,2
00401395 |. 47 |INC EDI
00401396 |. 4B |DEC EBX
00401397 |. DA7424 0C |FIDIV DWORD PTR SS:[ESP+C]
0040139B |. 894C24 0C |MOV DWORD PTR SS:[ESP+C],ECX
0040139F |. DEC1 |FADDP ST(1),ST
004013A1 |.^ 75 AD |JNZ SHORT ZeroMake.00401350 ; <== END First Section
004013A3 |. DC0D 10264000 FMUL QWORD PTR DS:[402610] ; <== Value = Value * 4
- Và kết quả cho ta là số : 3.1415426535898247630 . Đây chính là số PI . Như vậy giá trị của vòng lặp
cho ta một giá trị là (PI/4) .
- Nhắc lại toán học. Ta có Tan(PI/4) = 1 hay nói cách khác Arctan(1) = PI/4 . Vậy có nghĩa, vòng lặp
trên tương đương với phép toán Arctan(1)

```

/\*/\*/\* - SERIAL :

User : REA-cRaCkErS

User : VHT-cRaCkErS

Serial : **3.1415426535898247630** ( PI )

### III – KeyGen :

N/A

### IV – SourceCode ( VC++ ) :

N/A

### V – End of Tut :

- Finished – **16/06/2004**

# Reverse Engineering Association

## SoftWare

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| Homepage :     | <a href="http://www.emailarms.com">http://www.emailarms.com</a> |
| Production :   | F-Key Solutions, Inc.                                           |
| SoftWare :     | <b>1st Desktop Guard 1.5</b>                                    |
| Copyright by : | Copyright © 2004 F-Key Solutions, Inc. All Rights Reserved.     |
| Type :         | <b>Serial</b>                                                   |
| Packed :       | N / A                                                           |
| Language :     | <b>Borland Delphi 6.0 - 7.0</b>                                 |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsm 10</b>                       |
| Unpack :       | N / A                                                           |
| Request :      | <b>Correct Serial / KeyGen</b>                                  |

### 1st Desktop Guard 1.5

This program gives you the ability to save, restore, manage and lock your desktop layout that includes files and folders located on your desktop, placement of desktop icons and desired wallpaper. If you choose to lock your desktop layout, every time you reboot your PC, the program will restore your desktop icons and bring them back to their original positions as well as return your old wallpaper to the background. You can create an unlimited number of desktop layouts for different purposes such as gaming, working, surfing the Internet as well as provide different users with their own desktops. ....

#### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**". Ta tìm được thông báo này tại địa chỉ :  
004903A8 |. BA D4044900 MOV EDX,desksave.004904D4 ; ASCII "Registration code is invalid!"
- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :  
0049028F |. E8 FC4DFCFF CALL desksave.00455090 ; <== Set BreakPoint here

#### II – Cracking :

- Quá trình mã hoá của chương trình này rất đơn giản . Từ BP ta trace xuống chút :  
004902AC |. E8 DBFDFFFF CALL desksave.0049008C ; <== Trace Into  
----- Trace Into -----  
0049009A |. E8 A943F7FF CALL desksave.00404448 ; <== Get Length Serial  
0049009F |. 83F8 0E CMP EAX,0E ; <== LenS must be 14 charts  
004900A2 |. 75 67 JNZ SHORT desksave.0049010B  
004900A4 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]  
004900A6 |. 8038 33 CMP BYTE PTR DS:[EAX],33 ; <== S[0] = 0x33  
004900A9 |. 0F94C0 SETE AL  
004900AC |. 83E0 7F AND EAX,7F

```

004900AF |. 03F0 ADD ESI,EAX
004900B1 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900B3 |. 8078 02 33 CMP BYTE PTR DS:[EAX+2],33 ; <== S[2] = 0x33
004900B7 |. 0F94C0 SETE AL
004900BA |. 83E0 7F AND EAX,7F
004900BD |. 03F0 ADD ESI,EAX
004900BF |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900C1 |. 8078 03 39 CMP BYTE PTR DS:[EAX+3],39 ; <== S[3] = 0x39
004900C5 |. 0F94C0 SETE AL
004900C8 |. 83E0 7F AND EAX,7F
004900CB |. 03F0 ADD ESI,EAX
004900CD |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900CF |. 8078 04 30 CMP BYTE PTR DS:[EAX+4],30 ; <== S[4] = 0x30
004900D3 |. 0F94C0 SETE AL
004900D6 |. 83E0 7F AND EAX,7F
004900D9 |. 03F0 ADD ESI,EAX
004900DB |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900DD |. 8078 07 38 CMP BYTE PTR DS:[EAX+7],38 ; <== S[7] = 0x38
004900E1 |. 0F94C0 SETE AL
004900E4 |. 83E0 7F AND EAX,7F
004900E7 |. 03F0 ADD ESI,EAX
004900E9 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900EB |. 8078 08 38 CMP BYTE PTR DS:[EAX+8],38 ; <== S[8] = 0x38
004900EF |. 0F94C0 SETE AL
004900F2 |. 83E0 7F AND EAX,7F
004900F5 |. 03F0 ADD ESI,EAX
004900F7 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004900F9 |. 8078 0A 32 CMP BYTE PTR DS:[EAX+A],32 ; <== S[10] = 0x32
004900FD |. 0F94C0 SETE AL
00490100 |. 83E0 7F AND EAX,7F
00490103 |. 03F0 ADD ESI,EAX
00490105 |. 83FE 07 CMP ESI,7 ; <== if ALL correct
00490108 |. 0F94C3 SETE BL ; <== EBX = 0x1
0049010B |> 8BC3 MOV EAX,EBX ; <== EAX = EBX
----- Trace Into -----

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 31390CT88B27PZ

### III – KeyGen :

/Section I /- Chuỗi Serial có chiều dài là 14 ký tự .

/Section II /- Các vị trí [0][2][3][4][7][8][10] là các ký tự mặc định .

### IV – End of Tut :

- Finished    –      *August 23, 2004*

# Reverse Engineering Association

## SoftWare

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| Homepage :     | <a href="http://www.emailarms.com">http://www.emailarms.com</a> |
| Production :   | F-Key Solutions, Inc.                                           |
| SoftWare :     | 1st SMTP Server 2.5                                             |
| Copyright by : | Copyright © 2004 F-Key Solutions, Inc. All Rights Reserved.     |
| Type :         | Serial                                                          |
| Packed :       | N / A                                                           |
| Language :     | Borland Delphi 6.0 - 7.0                                        |
| Crack Tool :   | OllyDbg 1.09d, PEiD 0.92, kWdsm 10                              |
| Unpack :       | N / A                                                           |
| Request :      | Correct Serial / KeyGen                                         |

### 1st SMTP Server 2.5

1st SMTP Server is a SMTP server program for Windows that lets you send email messages directly from your computer. It can be used along with virtually any mailer or email program including Microsoft Outlook, Outlook Express, Eudora, Netscape and The Bat, and it is ideal for laptop PC users who travel a lot and have to use different Internet Service Providers (ISP) on the run. You can use it instead of your ISP's SMTP server to increase your email security and privacy due to your email messages will be delivered directly to the mailboxes of your recipients. 1st SMTP Server supports all email programs but best optimized to work with Outlook Express. The email program you already use for sending and receiving messages can be connected to the server in a very easy way - just by using the word "localhost" instead of your current SMTP host. Having done so, you can send messages in a usual manner. 1st SMTP Server is very fast and reliable, it has a flexible protection that lets you protect the server from the outside of the Internet. The user interface of the program is very easy to learn, excellent documentation is included.

### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**". Ta tìm được thông báo này tại địa chỉ :  
004E7A3C |. BA 547B4E00 MOV EDX,SMPServ.004E7B54 ; ASCII "Registration code is invalid!"
- Truy ngược lên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :  
004E7927 |. E8 A08BF5FF CALL SMPServ.004404CC ; <== Set BreakPoint here

### II – Cracking :

- Quá trình mã hoá của chương trình này rất đơn giản . Từ BP ta trace xuống chút :  
004E7944 |. E8 4FFDFFFF CALL SMPServ.004E7698 ; <== Trace Into  
----- Trace Into -----  
004E76A6 |. E8 35D2F1FF CALL SMPServ.004048E0 ; <== Get Length Serial  
004E76AB |. 83F8 0E CMP EAX,0E ; <== LenS must be 14 charts

```

004E76AE |. 75 67 JNZ SHORT SMTPServ.004E7717
004E76B0 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76B2 |. 8038 30 CMP BYTE PTR DS:[EAX],30 ; <== S[0] = 0x30
004E76B5 |. 0F94C0 SETE AL
004E76B8 |. 83E0 7F AND EAX,7F
004E76BB |. 03F0 ADD ESI,EAX
004E76BD |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76BF |. 8078 02 33 CMP BYTE PTR DS:[EAX+2],33 ; <== S[2] = 0x33
004E76C3 |. 0F94C0 SETE AL
004E76C6 |. 83E0 7F AND EAX,7F
004E76C9 |. 03F0 ADD ESI,EAX
004E76CB |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76CD |. 8078 03 33 CMP BYTE PTR DS:[EAX+3],33 ; <== S[3] = 0x33
004E76D1 |. 0F94C0 SETE AL
004E76D4 |. 83E0 7F AND EAX,7F
004E76D7 |. 03F0 ADD ESI,EAX
004E76D9 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76DB |. 8078 04 31 CMP BYTE PTR DS:[EAX+4],31 ; <== S[4] = 0x31
004E76DF |. 0F94C0 SETE AL
004E76E2 |. 83E0 7F AND EAX,7F
004E76E5 |. 03F0 ADD ESI,EAX
004E76E7 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76E9 |. 8078 07 34 CMP BYTE PTR DS:[EAX+7],34 ; <== S[7] = 0x34
004E76ED |. 0F94C0 SETE AL
004E76F0 |. 83E0 7F AND EAX,7F
004E76F3 |. 03F0 ADD ESI,EAX
004E76F5 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E76F7 |. 8078 08 39 CMP BYTE PTR DS:[EAX+8],39 ; <== S[8] = 0x39
004E76FB |. 0F94C0 SETE AL
004E76FE |. 83E0 7F AND EAX,7F
004E7701 |. 03F0 ADD ESI,EAX
004E7703 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
004E7705 |. 8078 0A 35 CMP BYTE PTR DS:[EAX+A],35 ; <== S[10] = 0x35
004E7709 |. 0F94C0 SETE AL
004E770C |. 83E0 7F AND EAX,7F
004E770F |. 03F0 ADD ESI,EAX
004E7711 |. 83FE 07 CMP ESI,7 ; <== if ALL correct
004E7714 |. 0F94C3 SETE BL ; <== EBX = 0x1
004E7717 |> 8BC3 MOV EAX,EBX ; <== EAX = EBX
----- Trace Into -----

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 0C3316049O52ZD

### III – KeyGen :

/Section I /- Chuỗi Serial có chiều dài là 14 ký tự .

/Section II /- Các vị trí [0][2][3][4][7][8][10] là các ký tự mặc định .

### IV – End of Tut :

- Finished    –      *August 23, 2004*

# Reverse Engineering Association

## SoftWare

|                     |   |                                                                           |
|---------------------|---|---------------------------------------------------------------------------|
| <b>Homepage</b>     | : | <a href="http://www.popupinspector.com">http://www.popupinspector.com</a> |
| <b>Production</b>   | : | GIANT Company Software, INC.                                              |
| <b>SoftWare</b>     | : | <b>PopUp Inspector 1.5.432</b>                                            |
| <b>Copyright by</b> | : | Copyright © 2002 GIANT Company Software, INC. All Rights Reserved.        |
| <b>Type</b>         | : | <b>Serial</b>                                                             |
| <b>Packed</b>       | : | N/A                                                                       |
| <b>Language</b>     | : | <b>Microsoft Visual Basic 5.0 / 6.0</b>                                   |
| <b>Crack Tool</b>   | : | <b>OllyDbg 1.09d, PEiD 0.92, kWdsdm 10</b>                                |
| <b>Unpack</b>       | : | N/A                                                                       |
| <b>Request</b>      | : | <b>Correct Serial / KeyGen</b>                                            |

### PopUp Inspector 1.5.432

Popup Inspector's intelligent Popup Inspection Engine can block 100% of annoying pop-up and pop-under ads without interfering with your navigation. Unlike most pop-up killers, it works without human intervention; and there's no need for adding lists of blocked windows, or allowed windows. It only blocks unsolicited pop-up windows without in any way interfering with opening new windows !

#### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual Basic 5.0 / 6.0**
- Chạy thử chương trình với Fake Serial ta nhận được thông báo "**Your Registration key is not valid. Please enter your Registration Key again, or if you have forgott**" . Ta tìm được thông báo này tại địa chỉ :  
004A94F4 . BA 20604200 MOV EDX,PopUpIns.00426020 ; UNICODE "Your Registration key is not valid. Please enter your Registration Key again, or if you have forgott"
- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :  
004A8F82 . FF50 04 CALL DWORD PTR DS:[EAX+4] ; <== Set BreakPoint here

#### II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :
 

```
004A9195 . 66:395D AC CMP WORD PTR SS:[EBP-54],BX ; <== LenS must be 25 charts
004A9199 . 0F84 B8000000 JE PopUpIns.004A9257
004A919F . 68 005D4200 PUSH PopUpIns.00425D00 ; UNICODE "registration key entered is too short"
```
- Trace tiếp ta đến :
 

```
004A932A . FF51 20 CALL DWORD PTR DS:[ECX+20] ; <== Encrypt & Compare
```
- Tracce into để xem xét quá trình mã hoá . Trước hết chương trình cắt lấy 9 ký tự đầu tiên của chuỗi Serial nhập và nối vào sau đó chuỗi mặc định :
 

```
004541FB . FF75 D8 PUSH DWORD PTR SS:[EBP-28] ; <== 9 first charts of FakeSerial
```

```

004541FE . 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]
00454201 . FF30 PUSH DWORD PTR DS:[EAX] ; <== default String
00454203 . E8 FE1CFBFF CALL <JMP.&MSVBVM60.__vbaStrCat>

```

- Chuỗi này sau đó được mã hoá thành chuỗi MD5Hash :

```

00454220 . FF51 1C CALL DWORD PTR DS:[ECX+1C] ; <== MD5Hash

```

- Sau cùng quá trình mã hoá thực sự diễn ra . Quá trình này có thể được giải thích theo các bước như sau  
(1) từng byte của chuỗi MD5Hash được chia cho giá trị mặc định là 0x20, (2) phần dư của phép chia này chính là vị trí của ký tự trong chuỗi mặc định, (3) ký tự này sẽ được nối vào sau 9 ký tự đầu tiên của chuỗi Serial nhập để tạo thành chuỗi Serial thực . Chuỗi MD5Hash là 16 bytes tương đương với 16 ký tự trong chuỗi mặc định, và chiều dài chuỗi Serial thực :  $9 + 16 = 25$  ký tự :

```

0045427C >/6A 10 PUSH 10
0045427E . |58 POP EAX ; <== Len of MD5Hash : 16
0045427F . |3945 E0 CMP DWORD PTR SS:[EBP-20],EAX ; <== while (i < 16)
00454282 . |0F8F 2D010000 JG PopUpIns.004543B5 ; <== continue loop
00454288 . |8D45 E4 LEA EAX,DWORD PTR SS:[EBP-1C]
0045428B . |C785 60FFFFFF>MOV DWORD PTR SS:[EBP-A0],PopUpIns.0041A704 ; UNICODE
"&H"
00454295 . |8945 80 MOV DWORD PTR SS:[EBP-80],EAX
00454298 . |8D45 B8 LEA EAX,DWORD PTR SS:[EBP-48]
0045429B . |50 PUSH EAX
0045429C . |8B45 E0 MOV EAX,DWORD PTR SS:[EBP-20]
0045429F . |6BC0 02 IMUL EAX,EAX,2
004542A2 . |0F80 9D010000 JO PopUpIns.00454445
004542A8 . |83E8 01 SUB EAX,1
004542AB . |899D 58FFFFFF MOV DWORD PTR SS:[EBP-A8],EBX
004542B1 . |0F80 8E010000 JO PopUpIns.00454445
004542B7 . |50 PUSH EAX
004542B8 . |8D85 78FFFFFF LEA EAX,DWORD PTR SS:[EBP-88]
004542BE . |50 PUSH EAX
004542BF . |8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
004542C2 . |50 PUSH EAX
004542C3 . |897D C0 MOV DWORD PTR SS:[EBP-40],EDI
004542C6 . |897D B8 MOV DWORD PTR SS:[EBP-48],EDI
004542C9 . |C785 78FFFFFF>MOV DWORD PTR SS:[EBP-88],4008
004542D3 . |E8 261BFBFF CALL <JMP.&MSVBVM60.#632>
004542D8 . |8D85 58FFFFFF LEA EAX,DWORD PTR SS:[EBP-A8]
004542DE . |50 PUSH EAX
004542DF . |8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
004542E2 . |50 PUSH EAX
004542E3 . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
004542E6 . |50 PUSH EAX
004542E7 . |E8 421BFBFF CALL <JMP.&MSVBVM60.__vbaVarCat>
004542EC . |50 PUSH EAX
004542ED . |E8 001EFBFF CALL <JMP.&MSVBVM60.__vbaI4ErrVar>
004542F2 . |6A 20 PUSH 20 ; <== dV
004542F4 . |99 CDQ
004542F5 . |59 POP ECX ; <== MD5Hash[i]
004542F6 . |F7F9 IDIV ECX ; <== Chart = MD5Hash[i] % dV
004542F8 . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
004542FB . |50 PUSH EAX

```

```

004542FC . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
004542FF . |50 PUSH EAX
00454300 . |8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
00454303 . |50 PUSH EAX
00454304 . |8D45 B8 LEA EAX,DWORD PTR SS:[EBP-48]
00454307 . |50 PUSH EAX
00454308 . |6A 04 PUSH 4
0045430A . |8BF2 MOV ESI,EDX
0045430C . |E8 891BFBFF CALL <JMP.&MSVBVM60.__vbaFreeVarList>
00454311 . |8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18]
00454314 . |83C4 14 ADD ESP,14
00454317 . |8D95 78FFFFFF LEA EDX,DWORD PTR SS:[EBP-88]
0045431D . |8D4D B8 LEA ECX,DWORD PTR SS:[EBP-48]
00454320 . |8985 60FFFFFF MOV DWORD PTR SS:[EBP-A0],EAX
00454326 . |899D 58FFFFFF MOV DWORD PTR SS:[EBP-A8],EBX
0045432C . |C745 B0 01000>MOV DWORD PTR SS:[EBP-50],1
00454333 . |897D A8 MOV DWORD PTR SS:[EBP-58],EDI
00454336 . |C745 80 C43F4>MOV DWORD PTR SS:[EBP-80],PopUpIns.00413FC4 ; UNICODE
"0123456789ABCDEF GHJKLMNOPQRTUVWXYZ"
0045433D . |899D 78FFFFFF MOV DWORD PTR SS:[EBP-88],EBX
00454343 . |E8 7C1BFBFF CALL <JMP.&MSVBVM60.__vbaVarDup>
00454348 . |8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
0045434B . |83C6 01 ADD ESI,1
0045434E . |50 PUSH EAX
0045434F . |8D45 B8 LEA EAX,DWORD PTR SS:[EBP-48]
00454352 . |0F80 ED000000 JO PopUpIns.00454445
00454358 . |56 PUSH ESI
00454359 . |50 PUSH EAX
0045435A . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
0045435D . |50 PUSH EAX
0045435E . |E8 9B1AFBFF CALL <JMP.&MSVBVM60.#632> ; <== Chart = dStr[Chart]
00454363 . |8D85 58FFFFFF LEA EAX,DWORD PTR SS:[EBP-A8]
00454369 . |50 PUSH EAX
0045436A . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
0045436D . |50 PUSH EAX
0045436E . |8D45 88 LEA EAX,DWORD PTR SS:[EBP-78]
00454371 . |50 PUSH EAX
00454372 . |E8 B71AFBFF CALL <JMP.&MSVBVM60.__vbaVarCat> ; <== S[9 + i] = Chart
00454377 . |50 PUSH EAX
00454378 . |E8 6B1BFBFF CALL <JMP.&MSVBVM60.__vbaStrVarMove>
0045437D . |8BD0 MOV EDX,EAX
0045437F . |8D4D E8 LEA ECX,DWORD PTR SS:[EBP-18]
00454382 . |E8 851BFBFF CALL <JMP.&MSVBVM60.__vbaStrMove>
00454387 . |8D45 88 LEA EAX,DWORD PTR SS:[EBP-78]
0045438A . |50 PUSH EAX
0045438B . |8D45 98 LEA EAX,DWORD PTR SS:[EBP-68]
0045438E . |50 PUSH EAX
0045438F . |8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
00454392 . |50 PUSH EAX
00454393 . |8D45 B8 LEA EAX,DWORD PTR SS:[EBP-48]
00454396 . |50 PUSH EAX
00454397 . |6A 04 PUSH 4

```

```

00454399 . |E8 FC1AFBFF CALL <JMP.&MSVBVM60.__vbaFreeVarList>
0045439E . |83C4 14 ADD ESP,14
004543A1 . |6A 01 PUSH 1
004543A3 . |58 POP EAX
004543A4 . |0345 E0 ADD EAX,DWORD PTR SS:[EBP-20]
004543A7 . |0F80 98000000 JO PopUpIns.00454445
004543AD . |8945 E0 MOV DWORD PTR SS:[EBP-20],EAX
004543B0 . ^\E9 C7FFFF JMP PopUpIns.0045427C
004543B5 >|FF75 E8 PUSH DWORD PTR SS:[EBP-18] ; <== RealSerial
004543B8 . 8B45 0C MOV EAX,DWORD PTR SS:[EBP+C]
004543BB . FF30 PUSH DWORD PTR DS:[EAX] ; <== Fake Serial
004543BD . E8 1A1BFBFF CALL <JMP.&MSVBVM60.__vbaStrCmp>

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : OZXPA9NSX4FMU7MEL4DFU4KR2

### III – End of Tut :

- Finished – September 20, 2004

# Reverse Engineering Association SoftWare

|                |                                                               |
|----------------|---------------------------------------------------------------|
| Homepage :     | <a href="http://www.htmledit.com">http://www.htmledit.com</a> |
| Production :   | Nicolas Payart .                                              |
| SoftWare :     | Hexacolor 3.0                                                 |
| Copyright by : | Copyright © Nicolas Payart . All Rights Reserved.             |
| Type :         | Name / Serial                                                 |
| Packed :       | N/A                                                           |
| Language :     | Borland Delphi 4.0 - 5.0                                      |
| Crack Tool :   | OllyDbg 1.09d, PEiD 0.92, kWdsdm 10                           |
| Unpack :       | N/A                                                           |
| Request :      | Correct Serial / KeyGen                                       |

#### Hexacolor 3.0

Hexacolor is a color management tool for the Internet. Three ways to use Hexacolor :

- Choose a color from the palette and get the decimal and hexadecimal value
- Enter the decimal or hexadecimal values and get the color
- Pick up a color from screen (even outside the application)

### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**

- Chạy thử chương trình với Fake Serial ta nhận được thông báo "**Your Registration key is not valid. Please enter your Registration Key again, or if you have forgott**" . Ta không thì được thông báo này trong Olly . Sử dụng phương pháp STACK ta tìm đến được FUNCTION chứa đoạn CODE chính .

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :  
 00472FA6 . E8 EDC9FBFF CALL Hexacolo.0042F998 ; <== Set BreakPoint here

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

|                                                 |                            |
|-------------------------------------------------|----------------------------|
| 00472FBF . E8 806D0100 CALL Hexacolo.00489D44   | ; <== Encrypt              |
| 00472FC4 . 8BF0 MOV ESI,EAX                     | ; <== Value                |
| 00472FC6 . 33D2 XOR EDX,EDX                     |                            |
| 00472FC8 . 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8] | ; <== Fake Serial          |
| 00472FCB . E8 BC56F9FF CALL Hexacolo.0040868C   | ; <== Convert to HEX Value |
| 00472FD0 . 3BC6 CMP EAX,ESI                     | ; <== if they are EQUAL    |
| 00472FD2 . 75 58 JNZ SHORT Hexacolo.0047302C    | ; <== Congrat !!!          |

- Trace into ta biết quá trình mã hoá diễn ra như sau :

|                                                        |                            |
|--------------------------------------------------------|----------------------------|
| 00489D5E  . C70424 2A0401>MOV DWORD PTR SS:[ESP],1042A | ; <== Value = 0x1042A      |
| 00489D65  . 8BD7 MOV EDX,EDI                           |                            |
| 00489D67  . 85D2 TEST EDX,EDX                          |                            |
| 00489D69  . 7E 1E JLE SHORT Hexacolo.00489D89          |                            |
| 00489D6B  . B8 01000000 MOV EAX,1                      | ; <== i = 1                |
| 00489D70  > 8B0E /MOV ECX,DWORD PTR DS:[ESI]           | ; <== U                    |
| 00489D72  . 8A4C01 FF  MOV CL,BYTE PTR DS:[ECX+EAX-1]  | ; <== U[i-1]               |
| 00489D76  . 81E1 FF000000  AND ECX,0FF                 | ; <== U[i-1]               |
| 00489D7C  . 0FAFC8  IMUL ECX,EAX                       | ; <== Temp = U[i-1] * i    |
| 00489D7F  . 0FAFCF  IMUL ECX,EDI                       | ; <== Temp = Temp * LenU   |
| 00489D82  . 010C24  ADD DWORD PTR SS:[ESP],ECX         | ; <== Value = Value + Temp |
| 00489D85  . 40  INC EAX                                |                            |
| 00489D86  . 4A  DEC EDX                                |                            |
| 00489D87  .^ 75 E7 JNZ SHORT Hexacolo.00489D70         |                            |

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 225152

## III – End of Tut :

- Finished – September 20, 2004

# Reverse Engineering Association SoftWare

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| Homepage :     | <a href="http://www.andyhsoftware.co.uk">http://www.andyhsoftware.co.uk</a> |
| Production :   | Andy Hassall .                                                              |
| SoftWare :     | Space 1.3.4                                                                 |
| Copyright by : | Copyright © 2000-2004 Andy Hassall . All Rights Reserved.                   |
| Type :         | Name / Serial                                                               |
| Packed :       | N/A                                                                         |
| Language :     | Microsoft Visual C++ 6.0                                                    |
| Crack Tool :   | OllyDbg 1.09d, PEiD 0.92, kWdsm 10                                          |

Unpack : N/A  
 Request : Correct Serial / KeyGen

### Space 1.3.4

**Space** lets you easily visualise how your hard disk, removable disk, network drive or FTP account is being used, and allows you to quickly identify which files are clogging up your drive.

#### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 6.0**
- Chạy thử chương trình với Fake Serial ta nhận được **NAG**. Ta không thèm được thông báo này trong Olly. Sử dụng phương pháp STACK ta tìm đến được FUNCTION chứa đoạn CODE chính.
- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này:  
 00409F57 |. 55 PUSH EBP ; <== Set BreakPoint here

#### II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Trace xuống chút:

```

00409F61 |. 68 C40D4900 PUSH Space.00490DC4 ; ASCII "space"
00409F66 |. 52 PUSH EDX
00409F67 |. C78424 A40400>MOV DWORD PTR SS:[ESP+4A4],0
00409F72 |. E8 12AD0400 CALL Space.00454C89 ; <== Concat(dStr,U)
00409F77 |. 50 PUSH EAX
00409F78 |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
00409F7C |. C68424 9C0400>MOV BYTE PTR SS:[ESP+49C],1
..... rippling
00409FC3 |. 83E1 03 AND ECX,3
00409FC6 |. F3:A4 REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00409FC8 |. 8D4C24 24 LEA ECX,DWORD PTR SS:[ESP+24]
00409FCC |. E8 3FD0FFFF CALL Space.00407010 ; <== MD5Start
00409FD1 |. 8D8C24 900000>LEA ECX,DWORD PTR SS:[ESP+90]
00409FD8 |. 55 PUSH EBP
00409FD9 |. 51 PUSH ECX
00409FDA |. 8D4C24 2C LEA ECX,DWORD PTR SS:[ESP+2C]
00409FDE |. E8 5DD0FFFF CALL Space.00407040 ; <== MD5Update
00409FE3 |. 8D4C24 24 LEA ECX,DWORD PTR SS:[ESP+24]
00409FE7 |. E8 C4D2FFFF CALL Space.004072B0 ; <== MD5Finished

```

- Như vậy chuỗi U sẽ được nối vào sau chuỗi mặc định, và chuỗi này sẽ được mã hoá theo thuật toán MD5 để tạo chuỗi MD5Hash. Chuỗi MD5Hash này sẽ tham gia vào quá trình mã hoá cuối cùng để tạo chuỗi S:

```

0040A017 |. 33F6 XOR ESI,ESI
0040A019 |> 33C0 /XOR EAX,EAX
0040A01B |. 33C9 |XOR ECX,ECX
0040A01D |. 8A443E 01 |MOV AL,BYTE PTR DS:[ESI+EDI+1] ; <== MD5Hash[i+1]
0040A021 |. 8A0C3E |MOV CL,BYTE PTR DS:[ESI+EDI] ; <== MD5Hash[i]
0040A024 |. 03C1 |ADD EAX,ECX ; <== Value = MD5Hash[i] + MD5Hash{i+1}
0040A026 25 FF000080 | AND EAX,800000FF ; <== Value = Value and 0x800000FF

```

```

0040A02B |. 79 07 |JNS SHORT Space.0040A034
0040A02D |. 48 |DEC EAX
0040A02E |. 0D 00FFFFFF |OR EAX,FFFFFFFFFF00
0040A033 |. 40 |INC EAX
0040A034 |> 50 |PUSH EAX
0040A035 |. 8D5424 20 |LEA EDX,DWORD PTR SS:[ESP+20]
0040A039 |. 68 CC0D4900 |PUSH Space.00490DCC ; ASCII "%02x"
0040A03E |. 52 |PUSH EDX
0040A03F |. E8 683D0400 |CALL Space.0044DDAC ; <== Convert to String : vStr
0040A044 |. 83C4 0C |ADD ESP,0C
0040A047 |. 8D4424 1C |LEA EAX,DWORD PTR SS:[ESP+1C]
0040A04B |. 8D4C24 18 |LEA ECX,DWORD PTR SS:[ESP+18]
0040A04F |. 50 |PUSH EAX
0040A050 |. E8 43AD0400 |CALL Space.00454D98 ; <== Concat(S,vStr)
0040A055 |. 83C6 02 |ADD ESI,2 ; <== i+=2
0040A058 |. 83FE 10 |CMP ESI,10 ; <== while (i < 16)
0040A05B |.^ 7C BC |JL SHORT Space.0040A019 ; <== continue Loop
/*/*/* - SERIAL tương ứng :

```

User : REA-cRaCkErTeAm      Serial : 78d37ab13eec8817

### III – End of Tut :

- Finished – September 20, 2004

# Reverse Engineering Association SoftWare

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| Homepage :     | <a href="http://www.andyhsoftware.co.uk">http://www.andyhsoftware.co.uk</a> |
| Production :   | LenoSoft SoftWare Inc.                                                      |
| SoftWare :     | <b>EasyMoney 1.21</b>                                                       |
| Copyright by : | Copyright © 2000-2004 LenoSoft SoftWare Inc. All Rights Reserved.           |
| Type :         | <b>Name / Serial</b>                                                        |
| Packed :       | ASPack 2.12 -> Alexey Solodovnikov                                          |
| Language :     | Borland Delphi 6.0 - 7.0                                                    |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWds 10</b>                                    |
| Unpack :       | Manual                                                                      |
| Request :      | Correct Serial / KeyGen                                                     |

### EasyMoney 1.21

Very easy-to-use all-in-one personal finance manager. Balance your checkbook and credit cards, and keep track of your recurring bills and deposits..

### I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK **ASPack 2.12 -> Alexey Solodovnikov** . UnPACK và kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với Fake Serial ta nhận được "**Register code error!**" . Ta tìm được địa chỉ này tại :

0058ABBA > |B8 30AC5800 MOV EAX,\_EasyMon.0058AC30 ; ASCII "Register code error!"

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

00409F57 |. 55 PUSH EBP ; <== Set BreakPoint here

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

```
0058AB35 . E8 7A2DECFF CALL _EasyMon.0044D8B4 ; <== Set BreakPoint here
0058AB3A . 837D FC 00 CMP DWORD PTR SS:[EBP-4],0
0058AB3E . 75 29 JNZ SHORT _EasyMon.0058AB69
0058AB40 . 6A 40 PUSH 40
0058AB42 . 68 ECAB5800 PUSH _EasyMon.0058ABEC ; ASCII "Warning"
0058AB47 . 68 F4AB5800 PUSH _EasyMon.0058ABF4 ; ASCII "Please enter username"
0058AB4C . 8BC3 MOV EAX,EBX
0058AB4E . E8 C995ECFF CALL _EasyMon.0045411C
0058AB53 . 50 PUSH EAX ; |hOwner
0058AB54 . E8 87D2E7FF CALL _EasyMon.00407DE0 ; \MessageBoxA
0058AB59 . 8B83 04030000 MOV EAX,DWORD PTR DS:[EBX+304]
0058AB5F . 8B10 MOV EDX,DWORD PTR DS:[EAX]
0058AB61 . FF92 C0000000 CALL DWORD PTR DS:[EDX+C0]
0058AB67 . EB 5B JMP SHORT _EasyMon.0058ABC4
0058AB69 > 8D55 F8 LEA EDX,DWORD PTR SS:[EBP-8]
0058AB6C . 8B83 08030000 MOV EAX,DWORD PTR DS:[EBX+308]
0058AB72 . E8 3D2DECFF CALL _EasyMon.0044D8B4
0058AB77 . 837D F8 00 CMP DWORD PTR SS:[EBP-8],0
0058AB7B . 75 29 JNZ SHORT _EasyMon.0058ABA6
0058AB7D . 6A 40 PUSH 40
0058AB7F . 68 ECAB5800 PUSH _EasyMon.0058ABEC ; ASCII "Warning"
0058AB84 . 68 0CAC5800 PUSH _EasyMon.0058AC0C ; ASCII "Please enter register code"
0058AB89 . 8BC3 MOV EAX,EBX
0058AB8B . E8 8C95ECFF CALL _EasyMon.0045411C
0058AB90 . 50 PUSH EAX ; |hOwner
0058AB91 . E8 4AD2E7FF CALL _EasyMon.00407DE0 ; \MessageBoxA
0058AB96 . 8B83 08030000 MOV EAX,DWORD PTR DS:[EBX+308]
0058AB9C . 8B10 MOV EDX,DWORD PTR DS:[EAX]
0058AB9E . FF92 C0000000 CALL DWORD PTR DS:[EDX+C0]
0058ABA4 . EB 1E JMP SHORT _EasyMon.0058ABC4
0058ABA6 > 8BC3 MOV EAX,EBX
0058ABA8 . E8 9BFDFFFF CALL _EasyMon.0058A948 ; <== Trace Into
```

- Dùng F7 trace into ta đi đến quá trình mã hoá :

0058A98C |. E8 D7FBFFFF CALL \_EasyMon.0058A568 ; <== Encrypt

### ----- Encrypt -----

- Đầu tiên chương trình chuyển đổi từng ký tự của chuỗi U nhập sang mã HEX, sau đó liên kết lại với nhau để tạo thành chuỗi đầu tiên : **tStr** ( ví dụ : **U** : "ABC" được chuyển thành chuỗi **tStr** : "414243" )

0058A5B4 |> /8D4D EC /LEA ECX,[LOCAL.5]

0058A5B7 |. |8B45 FC |MOV EAX,[LOCAL.1] ; <== U

0058A5BA |. |0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== U[i]

```

0058A5BF |. |33D2 |XOR EDX,EDX
0058A5C1 |. |E8 E6F3E7FF |CALL _EasyMon.004099AC ; <== Convert to HEXString : hStr
0058A5C6 |. |8B55 EC |MOV EDX,[LOCAL.5] ; <== hStr
0058A5C9 |. |8D45 F8 |LEA EAX,[LOCAL.2]
0058A5CC |. |E8 6FA7E7FF |CALL _EasyMon.00404D40 ; <== Concat(tStr,hStr)
0058A5D1 |. |43 |INC EBX ; <== i++
0058A5D2 |. |4E |DEC ESI ; <== while (LenU-- > 0)
0058A5D3 |.^75 DF |JNZ SHORT _EasyMon.0058A5B4 ; <== continue loop

```

- Ké đó, chương trình đảo ngược chuỗi *tStr* tạo thành chuỗi *rStr* (*tStr* : “414243” --> *rStr* : “342414”)

```

0058A5E8 |> |8B45 F8 |MOV EAX,[LOCAL.2] ; <== tStr
0058A5EB |. |E8 48A7E7FF |CALL _EasyMon.00404D38 ; <== Len.tStr
0058A5F0 |. |2BC3 |SUB EAX,EBX ; <== Temp = Len.tStr - i
0058A5F2 |. |8B55 F8 |MOV EDX,[LOCAL.2] ; <== tStr
0058A5F5 |. |8A1402 |MOV DL,BYTE PTR DS:[EDX+EAX] ; <== tStr[Temp]
0058A5F8 |. |8D45 E8 |LEA EAX,[LOCAL.6]
0058A5FB |. |E8 60A6E7FF |CALL _EasyMon.00404C60
0058A600 |. |8B55 E8 |MOV EDX,[LOCAL.6]
0058A603 |. |8D45 F4 |LEA EAX,[LOCAL.3]
0058A606 |. |E8 35A7E7FF |CALL _EasyMon.00404D40
0058A60B |. |43 |INC EBX ; <== i++
0058A60C |. |4E |DEC ESI ; <== while (Len.tStr-> 0)
0058A60D |.^75 D9 |JNZ SHORT _EasyMon.0058A5E8 ; <== continue loop

```

- Tiếp theo, chương trình sẽ tiến hành cắt hai đoạn của chuỗi *rStr*, mỗi đoạn dài 4 ký tự :

```

0058A613 |. B9 04000000 MOV ECX,4 ; <== get 4 charts
0058A618 |. BA 01000000 MOV EDX,1 ; <== from the FIRST chart
0058A61D |. 8B45 F4 MOV EAX,[LOCAL.3] ; <== of rStr
0058A620 |. E8 6BA9E7FF CALL _EasyMon.00404F90 ; <== ripping
0058A625 |. 8D45 F4 LEA EAX,[LOCAL.3]
0058A628 |. 50 PUSH EAX
0058A629 |. B9 04000000 MOV ECX,4 ; <== get 4 charts
0058A62E |. BA 05000000 MOV EDX,5 ; <== from the FIFTH chart
0058A633 |. 8B45 F4 MOV EAX,[LOCAL.3] ; <== of rStr
0058A636 |. E8 55A9E7FF CALL _EasyMon.00404F90 ; <== ripping

```

- Tương tự như thế, chương trình cũng tiến hành cắt chuỗi *DefaultString* thành hai chuỗi có chiều dài mỗi chuỗi là 4 ký tự :

```

0058A6B6 |. BA 40A75800 MOV EDX,_EasyMon.0058A740 ; ASCII "Money2bar698"
0058A6BB |. E8 58A4E7FF CALL _EasyMon.00404B18
0058A6C0 |. 8D45 DC LEA EAX,[LOCAL.9]
0058A6C3 |. 50 PUSH EAX
0058A6C4 |. B9 04000000 MOV ECX,4 ; <== get 4 charts
0058A6C9 |. BA 01000000 MOV EDX,1 ; <== from the FIRST chart
0058A6CE |. 8B45 F0 MOV EAX,[LOCAL.4] ; <== of dStr
0058A6D1 |. E8 BAA8E7FF CALL _EasyMon.00404F90 ; <== ripping
0058A6D6 |. FF75 DC PUSH [LOCAL.9]
0058A6D9 |. 68 58A75800 PUSH _EasyMon.0058A758
0058A6DE |. FF75 F8 PUSH [LOCAL.2]
0058A6E1 |. 8D45 D8 LEA EAX,[LOCAL.10]
0058A6E4 |. 50 PUSH EAX
0058A6E5 |. B9 05000000 MOV ECX,5 ; <== get 4 charts

```

```

0058A6EA |. BA 05000000 MOV EDX,5 ; <== from the FIFTH chart
0058A6EF |. 8B45 F0 MOV EAX,[LOCAL.4] ; <== of dStr
0058A6F2 |. E8 99A8E7FF CALL _EasyMon.00404F90 ; <== ripping

```

- Cuối cùng chương trình sẽ nối 4 chuỗi lại với nhau theo định dạng “**Mone-XXXXy2bar-YYYY**” để tạo chuỗi Serial

----- ***Encrypt*** -----

```

0058A991 |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== RealSerial
0058A994 |. 58 POP EAX ; <== FakeSerial
0058A995 |. E8 E2A4E7FF CALL _EasyMon.00404E7C ; <== Compare

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : Mone-D614y2bar-5645

**III – End of Tut :**

- Finished – *September 21, 2004*

# Reverse Engineering Association SoftWare

|                |                                                                    |
|----------------|--------------------------------------------------------------------|
| Homepage :     | <a href="http://www.mechcad.net">http://www.mechcad.net</a>        |
| Production :   | MechCAD Software LLC.                                              |
| SoftWare :     | Ace Money 3.5.6                                                    |
| Copyright by : | Copyright © 2002 - 2004 MechCAD Software LLC. All Rights Reserved. |
| Type :         | Name / Serial                                                      |
| Packed :       | N / A                                                              |
| Language :     | Microsoft Visual C++ 6.0                                           |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsm 10</b>                          |
| Unpack :       | N / A                                                              |
| Request :      | Correct Serial / KeyGen                                            |

## Ace Money 3.5.6

AceMoney helps people organize and manage their personal finances quickly and easily. It supports all the features required for home or even small-business accounting needs: (1) - Manage multiple accounts of different types (2) - Create and manage budget . (3) - Do your financial math in multiple currencies . (4) - Know your spending habits and see where the money goes . (5) - Enjoy online banking . (6) - Don't miss the next bills .(7) - Trust, but verify !

**I – Information :**

- Dùng PEiD kiểm tra biết chương trình không bị PACK và chương trình được viết bằng **Microsoft Visual C++ 6.0**

- Chạy thử chương trình với Fake Serial ta nhận được "**Invalid user name or serial number!**" . Ta tìm được địa chỉ này tại :

00444FC7 |. 68 78655200 PUSH AceMoney.00526578 ; |Arg1 = 00526578 ASCII "Invalid user name or serial number! Both name and serial number are case sensitive. The serial number should

match the name. To avoid typing mistakes, please copy/paste the name and the key from the order confirmation e"..."

- Dò ngược lên trên và đặt BreakPoint tại :  
 00444F38 |. E8 83DAFBFF CALL AceMoney.004029C0 ; <== Set BreakPoint here

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

```
00444F88 |. E8 A3000000 CALL AceMoney.00445030 ; <== Encrypt & Compare
00444F8D |. 85C0 TEST EAX,EAX
00444F8F |. 74 10 JE SHORT AceMoney.00444FA1
00444F91 |. 6A 00 PUSH 0 ; /Arg3 = 00000000
00444F93 |. 6A 00 PUSH 0 ; |Arg2 = 00000000
00444F95 |. 68 58655200 PUSH AceMoney.00526558 ; |Arg1 = 00526558 ASCII "Thank
you for supporting us!"
00444F9A |. E8 39BF0900 CALL AceMoney.004E0ED8 ; \AceMoney.004E0ED8
```

- Như vậy ta cần trace into lệnh CALL này để xem xét quá trình mã hoá :

```
00445054 |. 50 PUSH EAX ; |Arg1 = "User"
00445055 |. E8 3D740600 CALL AceMoney.004AC497 ; \AceMoney.004AC497
```

- Trace Into tiếp, ta sẽ đến đoạn mã hoá đầu tiên : trong đoạn đầu tiên này chúng ta sẽ nhận biết được các ký tự của chuỗi Serial phải là các ký tự trong khoảng ( 0 – 9 ) và ( A – F ), và được chia làm các Section khác nhau . Các Section này được kiêm kết với nhau bằng các ký tự khác ngoài các khoảng này sao cho tổng số các ký tự ngoài khoảng này phải là 48 . Đồng thời đoạn mã hoá đầu tiên này sẽ chuyển đổi từ Section sang dạng giá trị HEX tương ứng ( ví dụ : SecI : “1234” sẽ được chuyển sang dạng hValue = 0x1234 ) :

```
004AC4F6 |. C745 E4 00000>MOV [LOCAL.7],0 ; <== FirstCheck : fChk == 0x0
004AC4FD |. C745 EC 00000>MOV [LOCAL.5],0 ; <== i == 0x0
004AC504 |. C745 E8 00000>MOV [LOCAL.6],0 ; <== i == 0x0
004AC50B |. EB 09 JMP SHORT AceMoney.004AC516
004AC50D |> 8B45 E8 /MOV EAX,[LOCAL.6] ; <== / from here
004AC510 |. 83C0 01 |ADD EAX,1 ; <== |
004AC513 |. 8945 E8 |MOV [LOCAL.6],EAX ; <== | this section mean
004AC516 |> 8B4D FC MOV ECX,[LOCAL.1] ; <== | while (i < LenS)
004AC519 |. 83C1 01 |ADD ECX,1 ; <== | Continue Loop
004AC51C |. 394D E8 |CMP [LOCAL.6],ECX ; <== |
004AC51F |. 0F8D 80000000 |JGE AceMoney.004AC5A5 ; <== \ to here
004AC525 |. 8B55 0C |MOV EDX,[ARG.2] ; <== S
004AC528 |. 0355 E8 |ADD EDX,[LOCAL.6] ; <== i
004AC52B |. 8A02 |MOV AL,BYTE PTR DS:[EDX] ; <== S[i]
004AC52D |. 50 |PUSH EAX ; /Arg1
004AC52E |. E8 3DFEFFFF |CALL AceMoney.004AC370 ; ;
\AceMoney.004AC370
```

### ===== FirstCheck =====

```
004AC37C |. 83F8 30 CMP EAX,30 ; <== / from here
004AC37F |. 7C 0E JL SHORT AceMoney.004AC38F ; <== |
004AC381 |. 8B4D 08 MOV ECX,[ARG.1] ; <== |
004AC384 |. 81E1 FF000000 AND ECX,0FF ; <== |
004AC38A |. 83F9 39 CMP ECX,39 ; <== |
```

```

004AC38D |. 7E 24 JLE SHORT AceMoney.004AC3B3 ; <== |
004AC38F |> 8B55 08 MOV EDX,[ARG.1] ; <== | if S[i] belong to
004AC392 |. 81E2 FF000000 AND EDX,0FF ; <== | (0-9) or (A - F)
004AC398 |. 83FA 41 CMP EDX,41 ; <== | Temp == 0x0
004AC39B |. 7C 0D JL SHORT AceMoney.004AC3AA ; <== | else (Temp == 1)
004AC39D |. 8B45 08 MOV EAX,[ARG.1] ; <== |
004AC3A0 |. 25 FF000000 AND EAX,0FF ; <== |
004AC3A5 |. 83F8 46 CMP EAX,46 ; <== |
004AC3A8 |. 7E 09 JLE SHORT AceMoney.004AC3B3 ; <== |
004AC3AA |> C745 FC 00000>MOV [LOCAL.1],0 ; <== |
004AC3B1 |. EB 07 JMP SHORT AceMoney.004AC3BA ; <== |
004AC3B3 |> C745 FC 01000>MOV [LOCAL.1],1 ; <== |
004AC3BA |> 8B45 FC MOV EAX,[LOCAL.1] ; <== \ to here

```

**===== FirstCheck =====**

```

004AC533 |. 83C4 04 |ADD ESP,4 ; <== if (Temp == 0x1)
004AC536 |. 85C0 |TEST EAX,EAX ; <== end Section
004AC538 |. 74 1B |JE SHORT AceMoney.004AC555
004AC53A |. 8B4D F8 |MOV ECX,[LOCAL.2]
004AC53D |. 034D EC |ADD ECX,[LOCAL.5]
004AC540 |. 8B55 0C |MOV EDX,[ARG.2] ; <== S
004AC543 |. 0355 E8 |ADD EDX,[LOCAL.6] ; <== i
004AC546 |. 8A02 |MOV AL,BYTE PTR DS:[EDX] ; <== S[i]
004AC548 |. 8801 |MOV BYTE PTR DS:[ECX],AL ; <== saved
004AC54A |. 8B4D EC |MOV ECX,[LOCAL.5] ; <== i
004AC54D |. 83C1 01 |ADD ECX,1 ; <== i++
004AC550 |. 894D EC |MOV [LOCAL.5],ECX ; <== i
004AC553 |. EB 4B |JMP SHORT AceMoney.004AC5A0 ; <== Next Chart of Serial
004AC555 |> \837D EC 00 |CMP [LOCAL.5],0
004AC559 |. 7E 45 |JLE SHORT AceMoney.004AC5A0
004AC55B |. 8B55 F8 |MOV EDX,[LOCAL.2]
004AC55E |. 0355 EC |ADD EDX,[LOCAL.5]
004AC561 |. C602 00 |MOV BYTE PTR DS:[EDX],0
004AC564 |. 8B45 F8 |MOV EAX,[LOCAL.2] ; <== Section
004AC567 |. 50 |PUSH EAX ; /Arg1
004AC568 |. E8 54FFFFFF |CALL AceMoney.004AC3C1 ; \AceMoney.004AC3C1

```

**===== Convert to hValue =====**

```

004AC3DA |. 52 PUSH EDX ; /String
004AC3DB |. FF15 24F34F00 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ; \lstrlenA
004AC3E1 |. 83F8 04 CMP EAX,4 ; <== each Section must be 4 charts
004AC3E4 |. 74 09 JE SHORT AceMoney.004AC3EF
004AC3E6 |> 66:0D FFFF OR AX,0FFFF
004AC3EA |. E9 A4000000 JMP AceMoney.004AC493
004AC3EF |> C745 EC 00000>MOV [LOCAL.5],0
004AC3F6 |. EB 09 JMP SHORT AceMoney.004AC401
004AC3F8 |> 8B45 EC /MOV EAX,[LOCAL.5] ; <== / from here
004AC3FB |. 83C0 01 |ADD EAX,1 ; <== |
004AC3FE |. 8945 EC |MOV [LOCAL.5],EAX ; <== |
004AC401 |> 837D EC 04 CMP [LOCAL.5],4 ; <== |
004AC405 |. 7D 6A |JGE SHORT AceMoney.004AC471 ; <== |
004AC407 |. 8B4D 08 |MOV ECX,[ARG.1] ; <== |
004AC40A |. 034D EC |ADD ECX,[LOCAL.5] ; <== |
004AC40D |. 0FBE11 |MOVSX EDX,BYTE PTR DS:[ECX] ; <== |

```

|                        |                                     |                             |
|------------------------|-------------------------------------|-----------------------------|
| 004AC410  . 83FA 30    | CMP EDX,30                          | ; <==                       |
| 004AC413  . 7C 23      | JL SHORT AceMoney.004AC438          | ; <==   if Sec[i] belong to |
| 004AC415  . 8B45 08    | MOV EAX,[ARG.1]                     | ; <==   ( 0 - 9 )           |
| 004AC418  . 0345 EC    | ADD EAX,[LOCAL.5]                   | ; <==   S[i] = S[i] - 0x30  |
| 004AC41B  . 0FBE08     | MOVSX ECX,BYTE PTR DS:[EAX]         | ; <==                       |
| 004AC41E  . 83F9 39    | CMP ECX,39                          | ; <==                       |
| 004AC421  . 7F 15      | JG SHORT AceMoney.004AC438          | ; <==                       |
| 004AC423  . 8B55 08    | MOV EDX,[ARG.1]                     | ; <==                       |
| 004AC426  . 0355 EC    | ADD EDX,[LOCAL.5]                   | ; <==                       |
| 004AC429  . 0FBE02     | MOVSX EAX,BYTE PTR DS:[EDX]         | ; <==                       |
| 004AC42C  . 83E8 30    | SUB EAX,30                          | ; <==                       |
| 004AC42F  . 8B4D EC    | MOV ECX,[LOCAL.5]                   | ; <==                       |
| 004AC432  . 89448D F0  | MOV DWORD PTR SS:[EBP+ECX*4-10],EAX | ; <==                       |
| 004AC436  . EB 37      | JMP SHORT AceMoney.004AC46F         | ; <==                       |
| 004AC438  > 8B55 08    | MOV EDX,[ARG.1]                     | ; <==                       |
| 004AC43B  . 0355 EC    | ADD EDX,[LOCAL.5]                   | ; <==                       |
| 004AC43E  . 0FBE02     | MOVSX EAX,BYTE PTR DS:[EDX]         | ; <==                       |
| 004AC441  . 83F8 41    | CMP EAX,41                          | ; <==                       |
| 004AC444  . 7C 23      | JL SHORT AceMoney.004AC469          | ; <==                       |
| 004AC446  . 8B4D 08    | MOV ECX,[ARG.1]                     | ; <==   if Sec[i] belong to |
| 004AC449  . 034D EC    | ADD ECX,[LOCAL.5]                   | ; <==   ( A - F )           |
| 004AC44C  . 0FBE11     | MOVSX EDX,BYTE PTR DS:[ECX]         | ; <==   S[i] = S[i] - 0x37  |
| 004AC44F  . 83FA 46    | CMP EDX,46                          | ; <==                       |
| 004AC452  . 7F 15      | JG SHORT AceMoney.004AC469          | ; <==                       |
| 004AC454  . 8B45 08    | MOV EAX,[ARG.1]                     | ; <==                       |
| 004AC457  . 0345 EC    | ADD EAX,[LOCAL.5]                   | ; <==                       |
| 004AC45A  . 0FBE08     | MOVSX ECX,BYTE PTR DS:[EAX]         | ; <==                       |
| 004AC45D  . 83E9 37    | SUB ECX,37                          | ; <==                       |
| 004AC460  . 8B55 EC    | MOV EDX,[LOCAL.5]                   | ; <==                       |
| 004AC463  . 894C95 F0  | MOV DWORD PTR SS:[EBP+EDX*4-10],ECX | ; <==                       |
| 004AC467  . EB 06      | JMP SHORT AceMoney.004AC46F         | ; <==                       |
| 004AC469  > 66:0D FFFF | OR AX,0FFFF                         | ; <==                       |
| 004AC46D  . EB 24      | JMP SHORT AceMoney.004AC493         | ; <==                       |
| 004AC46F  >^ EB 87     | JMP SHORT AceMoney.004AC3F8         | ; <== \ to here             |
| 004AC471  > 8B45 F8    | MOV EAX,[LOCAL.2]                   | ; <== / from here           |
| 004AC474  . C1E0 04    | SHL EAX,4                           | ; <==                       |
| 004AC477  . 8B4D FC    | MOV ECX,[LOCAL.1]                   | ; <==                       |
| 004AC47A  . 03C8       | ADD ECX,EAX                         | ; <==                       |
| 004AC47C  . 8B55 F4    | MOV EDX,[LOCAL.3]                   | ; <==                       |
| 004AC47F  . C1E2 08    | SHL EDX,8                           | ; <==   convert Sec         |
| 004AC482  . 03CA       | ADD ECX,EDX                         | ; <==   to hValue           |
| 004AC484  . 8B45 F0    | MOV EAX,[LOCAL.4]                   | ; <==                       |
| 004AC487  . C1E0 0C    | SHL EAX,0C                          | ; <==                       |
| 004AC48A  . 03C8       | ADD ECX,EAX                         | ; <==                       |
| 004AC48C  . 894D E8    | MOV [LOCAL.6],ECX                   | ; <==                       |
| 004AC48F  . 66:8B45 E8 | MOV AX,WORD PTR SS:[EBP-18]         | ; <== \ to here             |

**===== Convert to hValue =====**

|                       |                                |                   |
|-----------------------|--------------------------------|-------------------|
| 004AC56D  . 83C4 04   | ADD ESP,4                      |                   |
| 004AC570  . 8B4D E4   | MOV ECX,[LOCAL.7]              |                   |
| 004AC573  . 8B55 F0   | MOV EDX,[LOCAL.4]              |                   |
| 004AC576  . 66:89044A | MOV WORD PTR DS:[EDX+ECX*2],AX | ; <== save hValue |
| 004AC57A  . 8B45 E4   | MOV EAX,[LOCAL.7]              |                   |

```

004AC57D |. 8B4D F0 |MOV ECX,[LOCAL.4]
004AC580 |. 33D2 |XOR EDX,EDX
004AC582 |. 66:8B1441 |MOV DX,WORD PTR DS:[ECX+EAX*2] ; <== hValue
004AC586 |. 83FA FF |CMP EDX,-1 ; <== must be Positive
004AC589 |. 75 05 |JNZ SHORT AceMoney.004AC590
004AC58B |. E9 B6010000 |JMP AceMoney.004AC746
004AC590 |> 8B45 E4 |MOV EAX,[LOCAL.7]
004AC593 |. 83C0 01 |ADD EAX,1 ; <== fChk ++
004AC596 |. 8945 E4 |MOV [LOCAL.7],EAX ; <== fChk
004AC599 |. C745 EC 00000>|MOV [LOCAL.5],0
004AC5A0 |>^ E9 68FFFFFF |JMP AceMoney.004AC50D
004AC5A5 |> \837D E4 30 CMP [LOCAL.7],30 ; <== if (fChk == 0x30)
004AC5A9 |. 0F85 60010000 JNZ AceMoney.004AC70F ; <== Continue Check

```

- Đoạn mã hoá thứ hai chỉ diễn ra đối với Section đầu tiên của chuỗi Serial . Quá trình mã hoá này kết hợp với quá trình chuyển đổi qua lại với Floating Point Number nên khá phức tạp . Giá trị **hValueI** của SecI sẽ được chuyển sang dạng Floating Poit Number : **fValueI** . Sau đó được tính theo công thức

“ $\sqrt{fValueI - 1}$ ” , phần nguyên của giá trị này được chuyển đổi lại sang dạng HEX Value : **hValueI'** . Giá trị **hValueI'** này phải nhỏ hơn “**0x63**” mới thoả điều kiện kế tiếp . Tuy nhiên, ý nghĩa thực sự của quá trình này chính là so sánh chiều dài của chuỗi S nhập . Quá trình này sẽ được nhận xét rõ ràng trong quá trình mã hoá ra các Section còn lại của chuỗi Serial .

```

004AC5AF |. 8B4D F0 MOV ECX,[LOCAL.4] ; <== / from here
004AC5B2 |. 33D2 XOR EDX,EDX ; <== |
004AC5B4 |. 66:8B11 MOV DX,WORD PTR DS:[ECX] ; <== |
004AC5B7 |. 8995 5CFFFFFF MOV [LOCAL.41],EDX ; <== |
004AC5BD |. DB85 5CFFFFFF FILD [LOCAL.41] ; <== |
004AC5C3 |. DC25 28F74F00 FSUB QWORD PTR DS:[4FF728] ; <== |
004AC5C9 |. DD5D DC FSTP QWORD PTR SS:[EBP-24] ; <== |
004AC5CC |. DD45 DC FLD QWORD PTR SS:[EBP-24] ; <== |
004AC5CF |. DC1D 20F74F00 FCOMP QWORD PTR DS:[4FF720] ; <== |
004AC5D5 |. DFE0 FSTSW AX ; <== | Check length S
004AC5D7 |. F6C4 41 TEST AH,41 ; <== |
004AC5DA |. 0F85 2F010000 JNZ AceMoney.004AC70F ; <== |
004AC5E0 |. 8B45 E0 MOV EAX,[LOCAL.8] ; <== |
004AC5E3 |. 50 PUSH EAX ; <== |
004AC5E4 |. 8B4D DC MOV ECX,[LOCAL.9] ; <== |
004AC5E7 |. 51 PUSH ECX ; <== |
004AC5E8 |. E8 878F0100 CALL AceMoney.004C5574 ; <== |
004AC5ED |. 83C4 08 ADD ESP,8 ; <== |
004AC5F0 |. E8 A38E0100 CALL AceMoney.004C5498 ; <== |
004AC5F5 |. 8985 74FFFFFF MOV [LOCAL.35],EAX ; <== |
004AC5FB |. 83BD 74FFFFFF>CMP [LOCAL.35],63 ; <== |
004AC602 |. 0F8F 07010000 JG AceMoney.004AC70F ; <== \ to here

```

- Đoạn kế tiếp sẽ cho ta biết hValueII của SecII là một giá trị mặc định :

```

004AC608 |. 8B55 F0 MOV EDX,[LOCAL.4] ; <== hValueII
004AC60B |. 33C0 XOR EAX,EAX ; <== if (hValueII == 0x898)
004AC60D |. 66:8B42 02 MOV AX,WORD PTR DS:[EDX+2] ; <== hValueII
004AC611 |. 3D 98080000 CMP EAX,898 ; <== if (hValueII == 0x898)
004AC616 |. 0F85 F3000000 JNZ AceMoney.004AC70F ; <== Continue Check

```

- Quá trình mã hoá ở đoạn này được chia làm 3 đoạn :

|             |                                            |                               |
|-------------|--------------------------------------------|-------------------------------|
| 004AC61C  . | 66:C785 70FFF>MOV WORD PTR SS:[EBP-90],0   |                               |
| 004AC625  . | C745 E8 02000>MOV [LOCAL.6],2              |                               |
| 004AC62C  . | EB 09 JMP SHORT AceMoney.004AC637          |                               |
| 004AC62E  > | 8B4D E8 /MOV ECX,[LOCAL.6]                 | ; <== / form here             |
| 004AC631  . | 83C1 01  ADD ECX,1                         | ; <==                         |
| 004AC634  . | 894D E8  MOV [LOCAL.6],ECX                 | ; <==                         |
| 004AC637  > | 837D E8 2F CMP [LOCAL.6],2F                | ; <==                         |
| 004AC63B  . | 0F8D 8B000000 JGE AceMoney.004AC6CC        | ; <==                         |
| 004AC641  . | 8B55 E8  MOV EDX,[LOCAL.6]                 | ; <==   Check from Sec[2 + i] |
| 004AC644  . | 83EA 02  SUB EDX,2                         | ; <==   to Sec[2 + LenU]      |
| 004AC647  . | 3B95 74FFFFFF  CMP EDX,[LOCAL.35]          | ; <==                         |
| 004AC64D  . | 7D 60 JGE SHORT AceMoney.004AC6AF          | ; <==                         |
| 004AC64F  . | 8B45 E8  MOV EAX,[LOCAL.6]                 | ; <==                         |
| 004AC652  . | 8B4D F0  MOV ECX,[LOCAL.4]                 | ; <==                         |
| 004AC655  . | 33D2  XOR EDX,EDX                          | ; <==                         |
| 004AC657  . | 66:8B1441  MOV DX,WORD PTR DS:[ECX+EAX*2]  | ; <==                         |
| 004AC65B  . | 8B45 E8  MOV EAX,[LOCAL.6]                 | ; <==                         |
| 004AC65E  . | 83C0 3F  ADD EAX,3F                        | ; <==                         |
| 004AC661  . | 8B4D E8  MOV ECX,[LOCAL.6]                 | ; <==                         |
| 004AC664  . | 83E9 55  SUB ECX,55                        | ; <==   this equation         |
| 004AC667  . | 0FAFC1  IMUL EAX,ECX                       | ; <==   Value = Sec[2 + i] +  |
| 004AC66A  . | 03D0  ADD EDX,EAX                          | ; <==   + (((2 + i) + 0x3F) * |
| 004AC66C  . | 8995 58FFFFFF  MOV [LOCAL.42],EDX          | ; <==   * ((2 + i) - 0x55))   |
| 004AC672  . | DB85 58FFFFFF  FILD [LOCAL.42]             | ; <==   Must be               |
| 004AC678  . | DD5D DC  FSTP QWORD PTR SS:[EBP-24]        | ; <==   POSITIVE              |
| 004AC67B  . | DD45 DC  FLD QWORD PTR SS:[EBP-24]         | ; <==                         |
| 004AC67E  . | DC1D 20F74F00  FCOMP QWORD PTR DS:[4FF720] | ; <==                         |
| 004AC684  . | DFE0  FSTSW AX                             | ; <==                         |
| 004AC686  . | F6C4 41  TEST AH,41                        | ; <==                         |
| 004AC689  . | 74 05 JE SHORT AceMoney.004AC690           | ; <== \ to here               |

#### ----- == First Section == -----

Đoạn này chỉ kiểm tra các Section từ vị trí thứ II đến Section tương ứng với chiều dài U . Tuy nhiên, nếu kế hợp quá trình này và quá trình kế tiếp ta sẽ mã hoá ngược lại được chuỗi U nhập . Đây chính là cốt lõi của vấn đề.

#### ----- == First Section == -----

|             |                                                |                                 |
|-------------|------------------------------------------------|---------------------------------|
| 004AC68B  . | E9 B4000000 JMP AceMoney.004AC744              | ; <== if ( Value > 0 ) continue |
| 004AC690  > | 8B55 E0  MOV EDX,[LOCAL.8]                     | ; <== / form here               |
| 004AC693  . | 52  PUSH EDX                                   | ; <==                           |
| 004AC694  . | 8B45 DC  MOV EAX,[LOCAL.9]                     | ; <==                           |
| 004AC697  . | 50  PUSH EAX                                   | ; <==                           |
| 004AC698  . | E8 D78E0100  CALL AceMoney.004C5574            | ; <==   U[i] =                  |
|             | sqrt(Value)                                    |                                 |
| 004AC69D  . | 83C4 08  ADD ESP,8                             | ; <==                           |
| 004AC6A0  . | E8 F38D0100  CALL AceMoney.004C5498            | ; <==                           |
| 004AC6A5  . | 8B4D E8  MOV ECX,[LOCAL.6]                     | ; <==                           |
| 004AC6A8  . | 88840D 76FFFF> MOV BYTE PTR SS:[EBP+ECX-8A],AL | ; <==                           |
| 004AC6AF  > | 8B55 E8  MOV EDX,[LOCAL.6]                     | ; <== \ to here                 |

#### ----- == Second Section == -----

Như trên đã đề cập, quá trình này sẽ RE các Section tương ứng của S cho ra U . Kết hợp hai quá trình và RE chúng ta viết được công thức tính :

$$Sec = Char * Char + (0x41 + i) * (0x53 - i) \quad =====> \quad "%04X"$$

**-----== Second Section ==-----**

```
004AC6B2 |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== / form here
004AC6B5 |. 66:8B8D 70FFF>|MOV CX,WORD PTR SS:[EBP-90] ; <== |
004AC6BC |. 66:030C50 |ADD CX,WORD PTR DS:[EAX+EDX*2] ; <== | Sum = Sum + Value
004AC6C0 |. 66:898D 70FFF>|MOV WORD PTR SS:[EBP-90],CX ; <== |
004AC6C7 |.^ E9 62FFFFFF |JMP AceMoney.004AC62E ; <== \ to here
```

**-----== Third Section ==-----**

- Quá trình này cộng dồn giá trị các Section của chuỗi S đã được chuyển đổi từ trước.

- Toàn bộ vòng lặp của quá trình này bắt đầu từ đoạn thứ II đến đoạn áp chót của chuỗi S nhập.

Vì trong đoạn kiểm tra cuối cùng ta sẽ biết được Section cuối cùng sẽ là giá trị của **Sum** ("%"04X")

**-----== Third Section ==-----**

- Chuỗi U được trả về trong quá trình này sẽ được so sánh với chuỗi U ta nhập vào :

```
004AC6DA |. 8B45 08 |MOV EAX,[ARG.1]
004AC6DD |. 50 |PUSH EAX ; /String2
004AC6DE |. 8D8D 78FFFFFF LEA ECX,[LOCAL.34] ; |
004AC6E4 |. 51 |PUSH ECX ; |String1
004AC6E5 |. FF15 20F34F00 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA
004AC6EB |. 85C0 |TEST EAX,EAX ; <== if SAME
004AC6ED |. 75 20 |JNZ SHORT AceMoney.004AC70F ; <== go to Last Check
```

- Section cuối cùng sẽ là **Sum** ("%"04X")

```
004AC6EF |. 8B95 70FFFFFF MOV EDX,[LOCAL.36] ; <== Sum
004AC6F5 |. 81E2 FFFF0000 AND EDX,0FFFF ; <== Sum = Sum & 0xFFFF
004AC6FB |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== Add. of last Sec : hValue
004AC6FE |. 33C9 |XOR ECX,ECX
004AC700 |. 66:8B48 5E |MOV CX,WORD PTR DS:[EAX+5E] ; <== Sum
004AC704 |. 3BD1 |CMP EDX,ECX ; <== if (Sum == hValue)
004AC706 |. 75 07 |JNZ SHORT AceMoney.004AC70F ; <== ConGrat !!!
```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 00E2-0898-2F57-27BD-25B4-1D29-3B94-2F98-3A1C-  
26E9-421C-27FD-4827-30F0-3D34-25D5-43B4-0029-4823-18BE-6784-4AE1-3D6C-2CD6-72AE-  
6952-5F90-1649-6DF1-5AF1-41BB-26E9-01EB-0BB3-2EA6-12DB-153C-7E87-390C-0F3E-0099-  
0124-305E-440D-491C-4D06-4DB7-764C

**III – End of Tut :**

- Finished – *September 23, 2004*

# Reverse Engineering Association

## SoftWare

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| Homepage :     | <a href="http://www.acoustica.com">http://www.acoustica.com</a> |
| Production :   | Acoustica, Inc.                                                 |
| SoftWare :     | Acoustica Mixcraft 1.10.15                                      |
| Copyright by : | Copyright © 1998-2004 Acoustica, Inc. All Rights Reserved.      |
| Type :         | Name / Serial                                                   |
| Packed :       | N / A                                                           |
| Language :     | Microsoft Visual C++ 6.0                                        |

**Crack Tool :** OllyDbg 1.09d, PEiD 0.92, kWdsm 10  
**Unpack :** N / A  
**Request :** Correct Serial / KeyGen

**Acoustica Mixcraft 1.10.15**

**Mixcraft™** is a multitrack audio recording studio with effects, featuring Reverb, Delay/Echo, EQ, Compression, Flanger and Chorus, as well as resonant filters and a powerful loop editor. The high performance 32 bit sound engine supports broadcast quality WAV files and will even import compressed MP3, OGG & WMA files. Use it to record your own music, your band or even a remix for a dance recital. The amazing fact about home recording today is that you really only need a computer and a good multitrack recording program such as Mixcraft to create amazing sound! When you've finished your mix, publish it to the Internet as an MP3, OGG, WMA or RealAudio file, or render it to a WAV so that you can burn it to a CD.

**I – Information :**

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử User và Fake Serial, ta nhận được thông báo "That registration code is not valid. Double check the number you are typing in." . Tuy nhiên ta không thể tìm thấy chuỗi thông báo này trong Olly bằng các phương pháp thông thường . Sử dụng phương pháp tìm chuỗi bằng STACK :

Call stack of main thread, item 5

Address=00129CA8

Stack=00444337

Procedure / arguments=USER32.DialogBoxParamA

Called from=mixcraft.00444331

Frame=00129CA4

- Truy ngược về địa chỉ này, xác định được địa chỉ :

00444306 |. 68 38EE4600 PUSH mixcraft.0046EE38 ; ASCII "That registration code is not valid. Double check the number you are typing in."

- Trace tiếp và ta tìm được Function chính của quá trình mã hoá . trong Function này ta thấy có hàm **GetDlgItemTextA** nên đặt BreakPoint tại hàm này :

00433FD3 |. FFD6 CALL ESI ; \GetDlgItemTextA

**II – Cracking :**

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống một đoạn ta đến quá trình kiểm tra chuỗi Serial và chuỗi User xem có khoảng trắng hay không . Nếu có khoảng trắng thì sẽ được cắt bỏ

- Kiểm tra User :

```
00434016 >/8A8414 941100>/MOV AL,BYTE PTR SS:[ESP+EDX+1194]
0043401D |. |3C 20 |CMP AL,20
0043401F |. |74 03 |JE SHORT mixcraft.00434024
00434021 |. |8806 |MOV BYTE PTR DS:[ESI],AL
00434023 |. |46 |INC ESI
00434024 >|8DBC24 941100>|LEA EDI,DWORD PTR SS:[ESP+1194]
0043402B |. |83C9 FF |OR ECX,FFFFFF
0043402E |. |33C0 |XOR EAX,EAX
00434030 |. |42 |INC EDX
00434031 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
00434033 |. |F7D1 |NOT ECX
00434035 |. |49 |DEC ECX
00434036 |. |3BD1 |CMP EDX,ECX
```

```

00434038 |.^7C DC |JL SHORT mixcraft.00434016
- Kiểm tra Serial
00434091 |> /8A8414 940900>/MOV AL,BYTE PTR SS:[ESP+EDX+994]
00434098 |. |3C 20 |CMP AL,20
0043409A |. |74 03 |JE SHORT mixcraft.0043409F
0043409C |. |8806 |MOV BYTE PTR DS:[ESI],AL
0043409E |. |46 |INC ESI
0043409F |> |8DBC24 940900>|LEA EDI,DWORD PTR SS:[ESP+994]
004340A6 |. |83C9 FF |OR ECX,FFFFFF
004340A9 |. |33C0 |XOR EAX,EAX
004340AB |. |42 |INC EDX
004340AC |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
004340AE |. |F7D1 |NOT ECX
004340B0 |. |49 |DEC ECX
004340B1 |. |3BD1 |CMP EDX,ECX
004340B3 |.^7C DC |JL SHORT mixcraft.00434091

```

- Và chiều dài của chuỗi Serial được quy định :

```

004340F1 |. 83F9 17 CMP ECX,17 ; <== Length S must be 23 charts
004340F4 |. 0F85 67010000 JNZ mixcraft.00434261

```

- Ké đó chương trình kiểm tra chuỗi Serial nhập với một chuỗi mặc định, nếu chuỗi nhập trùng với chuỗi này thì thời gian dùng thử sẽ được kéo dài thêm . Chuỗi kéo dài thời gian sử dụng

#### "03XXX-GIVEM-EIMIX-CRAFT"

- Tiếp đó sẽ đến quá trình mã hoá và kiểm tra chuỗi Serial :

```

00434177 |. 51 PUSH ECX ; /Arg2
00434178 |. 52 PUSH EDX ; |Arg1
00434179 |. 8BCB MOV ECX,EBX ; |
0043417B |. E8 20040000 CALL mixcraft.004345A0 ; \mixcraft.004345A0

```

- Dùng F7 trace into ta lệnh CALL ta đến quá trình mã hoá . Đầu tiên, chương trình sẽ chuyển đổi User nhập vào thành dạng UpperCase :

```

004345DD |. E8 540C0200 CALL mixcraft.00455236 ; <== CharUpper(User)
- Tiếp đó, chương trình kiểm tra chiều dài chuỗi U, và so sánh với chiều dài chuỗi mặc định . Nếu (1) chuỗi U có chiều dài nhỏ hơn hay bằng chuỗi mặc định thì chuỗi mặc định sẽ được cắt ra một đoạn có chiều dài bằng chuỗi U, nếu (2) chuỗi U có chiều dài lớn hơn chuỗi mặc định thì chuỗi mặc định sẽ được kết hợp với chính nó để có chiều dài đúng bằng chiều dài của chuỗi U :

```

```

00434619 |> /BF FC604600 /MOV EDI,mixcraft.004660FC ; ASCII "mixcraft v1.0"
0043461E |. |83C9 FF |OR ECX,FFFFFF
00434621 |. |33C0 |XOR EAX,EAX
00434623 |. |8D9424 340400>|LEA EDX,DWORD PTR SS:[ESP+434]
0043462A |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
0043462C |. |F7D1 |NOT ECX
0043462E |. |2BF9 |SUB EDI,ECX
00434630 |. |8BF7 |MOV ESI,EDI
00434632 |. |8BD9 |MOV EBX,ECX
00434634 |. |8BFA |MOV EDI,EDX
00434636 |. |83C9 FF |OR ECX,FFFFFF
00434639 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
0043463B |. |8BCB |MOV ECX,EBX
0043463D |. |4F |DEC EDI
0043463E |. |C1E9 02 |SHR ECX,2
00434641 |. |F3:A5 |REP MOVS DWORD PTR ES:[EDI],DWORD PTR D>
00434643 |. |8BCB |MOV ECX,EBX

```

```

00434645 |. |83E1 03 |AND ECX,3
00434648 |. |F3:A4 |REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:>
0043464A |. |8DBC24 340400>|LEA EDI,DWORD PTR SS:[ESP+434]
00434651 |. |83C9 FF |OR ECX,FFFFFF
00434654 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
00434656 |. |F7D1 |NOT ECX
00434658 |. |49 |DEC ECX
00434659 |. |3BCD |CMP ECX,EBP
0043465B |.^\72 BC |JB SHORT mixcraft.00434619

```

- Sau đó, chuỗi mặc định mới được tạo thành này cũng được chuyển về dạng UpperCase :

```
0043466D |. E8 C40B0200 CALL mixcraft.00455236 ; <== CharUpper(DefaultString)
```

- Kết thúc các quá trình chuẩn bị ta đến quá trình mã hoá chuỗi đầu tiên . Quá trình mã hoá này dựa trên chuỗi U và chuỗi mặc định (dạng UpperCase ) :

```

00434688 |>/8A8C34 340400>|MOV CL,BYTE PTR SS:[ESP+ESI+434] ; <== dStr[i]
0043468F |. |8A9C34 340800>|MOV BL,BYTE PTR SS:[ESP+ESI+834] ; <== U[i]
00434696 |. |32CB |XOR CL,BL ; <== Temp = dStr[i] xor U[i]
00434698 |. |B8 4FECC44E |MOV EAX,4EC4EC4F ; <== Default Value : ValueD
0043469D |. |884C24 10 |MOV BYTE PTR SS:[ESP+10],CL ; <== Temp
004346A1 |. |8B4C24 10 |MOV ECX,DWORD PTR SS:[ESP+10] ; <== Temp
004346A5 |. |81E1 FF000000 |AND ECX,0FF ; <== Temp
004346AB |. |F7E9 |IMUL ECX ; <== Temp = Temp * ValueD
004346AD |. |C1FA 03 |SAR EDX,3 ; <== Over = Over sar 0x3
004346B0 |. |8BC2 |MOV EAX,EDX ; <== Over
004346B2 |. |C1E8 1F |SHR EAX,1F ; <== Value = Over / 0x80000000
004346B5 |. |03D0 |ADD EDX,EAX ; <== Over = Over + Value
004346B7 |. |8D0452 |LEA EAX,DWORD PTR DS:[EDX+EDX*2]; <== Value = Over + Over * 2
004346BA |. |8D0482 |LEA EAX,DWORD PTR DS:[EDX+EAX*4]; <== Value = Over + Value * 4
004346BD |. |D1E0 |SHL EAX,1 ; <== Value = Value * 2
004346BF |. |2BC8 |SUB ECX,EAX ; <== Value = Temp - Value
004346C1 |. |85D2 |TEST EDX,EDX ; <== Value
004346C3 |. |8BD9 |MOV EBX,ECX ; <== Value
004346C5 |. |7E 15 |JLE SHORT mixcraft.004346DC ; <== if (Over !=0)
004346C7 |. |8D7C24 34 |LEA EDI,DWORD PTR SS:[ESP+34] ; <== then
004346CB |. |83C9 FF |OR ECX,FFFFFF
004346CE |. |33C0 |XOR EAX,EAX ; <== Value
004346D0 |. |80C2 30 |ADD DL,30 ; <== Over = Over + 0x30
004346D3 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
004346D5 |. |F7D1 |NOT ECX
004346D7 |. |49 |DEC ECX
004346D8 |. |88540C 34 |MOV BYTE PTR SS:[ESP+ECX+34],DL ; <== Over saved at add.
004346DC |>|8D7C24 34 |LEA EDI,DWORD PTR SS:[ESP+34] ; <== and / else
004346E0 |. |83C9 FF |OR ECX,FFFFFF
004346E3 |. |33C0 |XOR EAX,EAX ; <== Value = Value + 0x41
004346E5 |. |80C3 41 |ADD BL,41 ; <== Value = Value + 0x41
004346E8 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
004346EA |. |F7D1 |NOT ECX
004346EC |. |49 |DEC ECX
004346ED |. |46 |INC ESI ; <== i++
004346EE |. |3BF5 |CMP ESI,EBP ; <== if (i < 8)
004346F0 |. |885C0C 34 |MOV BYTE PTR SS:[ESP+ECX+34],BL ; <== Value saved at add.
004346F4 |.^\7C 92 |JL SHORT mixcraft.00434688 ; <== Continue Loop

```

- Kết thúc quá trình này ta được một chuỗi mới . Do quá trình tính toán mà chiều dài của chuỗi mã hoá này có thể lớn hơn hay bằng chuỗi U nhập . Ở đây, PROGRAMMER tiến hành một bước quan trọng, ở tại vị trí “19” của chuỗi mã hoá này sẽ được gán với giá trị “NULL” :

004346FF |. C64424 46 00 MOV BYTE PTR SS:[ESP+46],0 ; <== TempSerial[19] == 0x0

- Điều này có nghĩa, (1) nếu chuỗi mã hoá có chiều dài nhỏ hơn hay bằng 18 ký tự thì sẽ không bị ảnh hưởng, nhưng (2) nếu chuỗi mã hoá có chiều dài lớn hơn 18 ký tự, do ký tự thứ “19” được chương trình gán là “NULL” nên chuỗi mã hoá sau khi qua quá trình này chỉ còn lại “18” ký tự . Các ký tự mã hoá sau đó coi như không được sử dụng .

- Hai ký tự đầu tiên của chuỗi Serial được tạo thành :

|                                                   |                                   |
|---------------------------------------------------|-----------------------------------|
| 00434708  . 49 DEC ECX                            | ; <== Lenngth TempSerial : Len.tS |
| 00434709  . BE 12000000 MOV ESI,12                | ; <== DefaultValue                |
| 0043470E  . 2BF1 SUB ESI,ECX                      | ; <== Temp = 0x12 - Len.tS        |
| 00434710  . B8 67666666 MOV EAX,66666667          | ; <== DefaultValue                |
| 00434715  . F7EE IMUL ESI                         | ; <== Value = Temp * 0x66666667   |
| 00434717  . C1FA 02 SAR EDX,2                     | ; <== Over = Over sar 0x2         |
| 0043471A  . 8BCA MOV ECX,EDX                      | ; <== Over                        |
| 0043471C  . 8BC6 MOV EAX,ESI                      | ; <== Temp                        |
| 0043471E  . C1E9 1F SHR ECX,1F                    | ; <== Value = Over / 0x80000000   |
| 00434721  . 03D1 ADD EDX,ECX                      | ; <== Over = Over + Value         |
| 00434723  . B9 0A000000 MOV ECX,0A                | ; <== DefaultValue                |
| 00434728  . 80C2 30 ADD DL,30                     | ; <== Over = Over + 0x30          |
| 0043472B  . 33DB XOR EBX,EBX                      |                                   |
| 0043472D  . 885424 14 MOV BYTE PTR SS:[ESP+14],DL | ; <== Over saved at add.          |
| 00434731  . 99 CDQ                                |                                   |
| 00434732  . F7F9 IDIV ECX                         | ; <== Temp = Temp / 0xA           |
| 00434734  . 80C2 30 ADD DL,30                     | ; <== Remain = Remain + 0x30      |
| 00434737  . 85F6 TEST ESI,ESI                     |                                   |
| 00434739  . 885424 15 MOV BYTE PTR SS:[ESP+15],DL | ; <== Remain saved at add.        |

- Quá trình mã hoá tiếp theo lại được chia làm hai trường hợp (1) nếu chuỗi mã hoá có chiều dài lớn hơn hay bằng 18 ký tự thì 18 ký tự này sẽ được nối vào sau hai ký tự được tạo đầu tiên trong chuỗi Serial .

Như vậy, chiều dài của chuỗi Serial là “20” ký tự . (2) Nếu chiều dài của chuỗi mã hoá nhỏ hơn 18 ký tự thì chương trình sẽ tiếp tục mã hoá các ký tự còn lại cho đủ 18 ký tự :

- Các ký tự được mã hoá tiếp theo này sẽ được kè liên hai ký tự đầu tiên của chuỗi Serial :

0043473F |> /E8 445F0100 /CALL mixcraft.0044A688 ; <== Value

===== Trace Into =====

|                                                   |                                |
|---------------------------------------------------|--------------------------------|
| 0044A68D  . 8B48 14 MOV ECX,DWORD PTR DS:[EAX+14] | ; <== DefaultValue == 0x1      |
| 0044A690  . 69C9 FD430300 IMUL ECX,ECX,343FD      | ; <== Temp = Temp * 0x343FD    |
| 0044A696  . 81C1 C39E2600 ADD ECX,269EC3          | ; <== Temp = Temp + 0x269EC3   |
| 0044A69C  . 8948 14 MOV DWORD PTR DS:[EAX+14],ECX | ; <== Temp                     |
| 0044A69F  . 8BC1 MOV EAX,ECX                      | ; <== Temp                     |
| 0044A6A1  . C1E8 10 SHR EAX,10                    | ; <== Value = Value / 0x10000  |
| 0044A6A4  . 25 FF7F0000 AND EAX,7FFF              | ; <== Value = Value and 0x7FFF |

===== NOTE =====

Giá trị Value sẽ được lưu lại cho các lần tính sau, như vậy ta có rất nhiều chuỗi tương ứng .

===== NOTE =====

===== Trace Into =====

|                                                        |                              |
|--------------------------------------------------------|------------------------------|
| 00434744  . 99  CDQ                                    |                              |
| 00434745  . B9 1A000000  MOV ECX,1A                    | ; <== DefaultValue           |
| 0043474A  . F7F9  IDIV ECX                             | ; <== Remain = Value % 0x1A  |
| 0043474C  . 80C2 41  ADD DL,41                         | ; <== Remain = Remain + 0x41 |
| 0043474F  . 88541C 16  MOV BYTE PTR SS:[ESP+EBX+16],DL | ; <== Remain saved at add.   |
| 00434753  . 43  INC EBX                                | ; <== i++                    |

```

00434754 |. |3BDE |CMP EBX,ESI ; <== while (i < 18 - Len.U)
00434756 |.^7C E7 |JL SHORT mixcraft.0043473F ; <== Continue Loop
- Và các ký tự của chuỗi mã hoá sẽ được nối tiếp sau chuỗi này để tạo thành chuỗi Serial có “20” ký tự . “20” ký tự này sau đó sẽ được phân chia thành “5” đoạn, được liên kết với nhau bằng ký tự “-“ :
0043478E |> /8D7C24 34 /LEA EDI,WORD PTR SS:[ESP+34]
00434792 |. |83C9 FF |OR ECX,FFFFFF
00434795 |. |33C0 |XOR EAX,EAX
00434797 |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
00434799 |. |8A4414 14 |MOV AL,BYTE PTR SS:[ESP+EDX+14]
0043479D |. |F7D1 |NOT ECX
0043479F |. |49 |DEC ECX
004347A0 |. |83FA 04 |CMP EDX,4
004347A3 |. |88440C 34 |MOV BYTE PTR SS:[ESP+ECX+34],AL
004347A7 |. |74 0A |JE SHORT mixcraft.004347B3
004347A9 |. |83FA 09 |CMP EDX,9
004347AC |. |74 05 |JE SHORT mixcraft.004347B3
004347AE |. |83FA 0E |CMP EDX,0E
004347B1 |. |75 12 |JNZ SHORT mixcraft.004347C5
004347B3 |> |8D7C24 34 /LEA EDI,WORD PTR SS:[ESP+34]
004347B7 |. |83C9 FF |OR ECX,FFFFFF
004347BA |. |33C0 |XOR EAX,EAX
004347BC |. |F2:AE |REPNE SCAS BYTE PTR ES:[EDI]
004347BE |. |F7D1 |NOT ECX
004347C0 |. |49 |DEC ECX
004347C1 |. |885C0C 34 |MOV BYTE PTR SS:[ESP+ECX+34],BL
004347C5 |> |42 |INC EDX
004347C6 |. |83FA 14 |CMP EDX,14
004347C9 |.^7C C3 |JL SHORT mixcraft.0043478E

```

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErS

Serial : 01P1F-MZ4GR-THX4D-T3V4V

### III – KeyGen :

- /Section 1/- Dựa và chiều dài chuỗi User để xác định chuỗi mặc định
- /Section 2/- Mã hoá chuỗi U và chuỗi mặc định bước thứ nhất .
- /Section 3/- Tuỳ thuộc vào chiều dài của chuỗi U mà xác định hai ký tự đầu tiên của chuỗi Serial.
- /Section 4/- Tiến hành mã hoá lần thứ hai và gắn chuỗi
- /Section 5/- Tạo thành chuỗi Serial theo định dạng **XXXXX-XXXXX-XXXXX-XXXXX**

### IV – SourceCode ( VC++ ) :

```

char reaName[64]={0};
char reaSerial[64]={0};
char reaDefault[20]="mixcraft v1.0";
char reaDefaultString[64]={0};
char reaTempSerial_01[64]={0};
char reaTempSerial_02[64]={0};
char reaTemp[64]={0};

int LenUser=0, LenTemp=0;
int i=0, j=0;

```

```

int Temp=0, Value=0, Over=0;

LenUser=GetDlgItemText(IDC_Name, reaName, 64);
if (LenUser < 1)
{
 MessageBox("-----===== Your name atleast 1 chart =====--- ", "Hey !! Please
input your name again !! ");
}
else
{
 i=0; j=0;
 while (i < LenUser)
 {
 if (reaName[i] == 0x20)
 {
 reaName[i] = NULL;
 i++;
 }
 else
 {
 reaTemp[j] = reaName[i];
 reaName[i] = NULL;
 i++;
 j++;
 }
 }

 i=0;
 while (i < lstrlen(reaTemp))
 {
 reaName[i] = reaTemp[i];
 i++;
 }
 i=0;
 while (i < lstrlen(reaTemp)+1)
 {
 reaTemp[i]= NULL;
 i++;
 }
}

LenUser = lstrlen(reaName);
if (LenUser < 13)
{
 lstrcpy(reaDefaultString, reaDefault, LenUser+1);
}
else
{
 i=0;
 while (i < (LenUser / 13))
 {
 lstrcpy(reaTemp, reaDefault, LenUser + 1);
 lstrcat(reaDefaultString, reaTemp);
}

```

```

 i++;
 }

 LenTemp = lstrlen(reaDefaultString);
 i=0;
 while (i < (LenUser - LenTemp))
 {
 reaDefaultString[LenTemp + i] = reaDefault[i];
 i++;
 }
}

CharUpper(reaNmae);
CharUpper(reDefaultString);
i=0; j=0;
while (i < LenUser)
{
 _asm
 {
 MOV ESI, i
 MOVSX ECX, reaDefaultString[ESI]
 MOVSX EBX, reaNmae[ESI]
 XOR ECX, EBX
 MOV EAX,0x4EC4EC4F
 XOR EDX, EDX
 IMUL ECX
 SAR EDX,0x3
 MOV EAX,EDX
 SHR EAX,0x1F
 ADD EDX,EAX
 LEA EAX,DWORD PTR DS:[EDX+EDX*2]
 LEA EAX,DWORD PTR DS:[EDX+EAX*4]
 SHL EAX,0x1
 SUB ECX,EAX
 MOV Over,EDX
 MOV Value,ECX
 }
 if (Over != 0)
 {
 reaTemp[j] = Over + 0x30;
 reaTemp[j+1] = Value + 0x41;
 reaTemp[j+2] = NULL;
 j+=2;
 i++;
 }
 else
 {
 reaTemp[j] = Value + 0x41;
 reaTemp[j+1] = NULL;
 j++;
 i++;
 }
}

```

```

LenTemp = lstrlen(reTemp);
if (LenUser < 13)
{
 _asm
 {
 MOV ECX, LenTemp
 MOV ESI, 0x12
 SUB ESI,ECX
 MOV EAX,0x66666667
 IMUL ESI
 SAR EDX,0x2
 MOV ECX,EDX
 MOV EAX,ESI
 SHR ECX,0x1F
 ADD EDX,ECX
 MOV ECX,0x0A
 ADD DL,0x30
 AND EDX, 0xFF
 MOV Over, EDX
 CDQ
 IDIV ECX
 ADD DL,0x30
 AND EDX, 0xFF
 MOV Value, EDX
 }
 reTempSerial_02[0] = Over;
 reTempSerial_02[1] = Value;
 i=0; Value=0x1;
 while (i < (18-lstrlen(reTemp)))
 {
 _asm
 {
 MOV ECX, Value
 IMUL ECX,ECX,0x343FD
 ADD ECX,0x269EC3
 MOV Value,ECX
 MOV EAX,ECX
 SHR EAX,0x10
 AND EAX,0x7FFF
 CDQ
 MOV ECX,0x1A
 IDIV ECX
 ADD DL,0x41
 AND EDX, 0xFF
 MOV Over, EDX
 }
 reTempSerial_02[i+2] = Over;
 i++;
 }
 lstrcat(reTempSerial_02,reTemp);
}
else
{

```

```

 reaTempSerial_02[0] = 0x30;
 reaTempSerial_02[1] = 0x30;
 lstrcpyn(reaTempSerial_01,reaTemp,19);
 lstrcat(reaTempSerial_02,reaTempSerial_01);
}

i=0; j=0;
while (i < lstrlen(reaTempSerial_02))
{
 if (i == 4 || i == 9 || i == 14)
 {
 reaSerial[j+1] = 0x2D;
 reaSerial[j] = reaTempSerial_02[i];
 j+=2;
 i++;
 }
 else
 {
 reaSerial[j] = reaTempSerial_02[i];
 j++;
 i++;
 }
}
SetDlgItemText(IDC_Serial,reaSerial);
}

```

**V – End of Tut :**

- Finished – ***13/07/2004***

# Reverse Engineering Association

## SoftWare

|                       |                                                                        |
|-----------------------|------------------------------------------------------------------------|
| <b>Homepage</b> :     | <b><a href="http://www.acoustica.com">http://www.acoustica.com</a></b> |
| <b>Production</b> :   | <b>Acoustica, Inc.</b>                                                 |
| <b>SoftWare</b> :     | <b>Acoustica MP3 CD Burner 3.01 Build 71</b>                           |
| <b>Copyright by</b> : | <b>Copyright © 1998-2004 Acoustica, Inc. All Rights Reserved.</b>      |
| <b>Type</b> :         | <b>Name / Serial</b>                                                   |
| <b>Packed</b> :       | <b>N / A</b>                                                           |
| <b>Language</b> :     | <b>Microsoft Visual C++ 6.0</b>                                        |
| <b>Crack Tool</b> :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsm 10</b>                              |
| <b>Unpack</b> :       | <b>N / A</b>                                                           |
| <b>Request</b> :      | <b>Correct Serial / KeyGen</b>                                         |

### **Acoustica MP3 CD Burner 3.01 Build 71**

**Acoustica MP3 CD Burner** lets you burn 200 songs on a CDR or 1000 songs on a DVD! Or burn normal music CDs for playback on your car or home stereo. With this software you can effortlessly rip your personal CDs, retag your songs and burn music CDs and MP3 CDs with a click! No rocket science necessary! Escape the confines of audio only burners and make your own mega

mixes with Acoustica MP3 CD Burner version 3.0!

## I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**

- Nhập thử User và Fake Serial, ta nhận được thông báo "That is not a valid registration code. Please verify the number and try again!". Tuy nhiên ta không thể tìm thấy chuỗi thông báo này trong Olly bằng các phương pháp thông thường. Sử dụng phương pháp tìm chuỗi bằng STACK :

Call stack of main thread, item 5

Address=0012D2AC

Stack=00467E4B

Procedure / arguments=USER32.DialogBoxParamA

Called from=cdburner.00467E45

Frame=0012D2A8

- Truy ngược về địa chỉ này, xác định được địa chỉ :

00467EEF . 68 F8DC4A00 PUSH cdburner.004ADCF8 ; /Text = "That is not a valid registration code. Please verify the number and try again!"

- Trace tiếp và ta tìm được Function chính của quá trình mã hoá. trong Function này ta thấy có hàm **GetDlgItemTextA** nên đặt BreakPoint tại hàm này :

00456CA7 |. FFD7 CALL EDI ; \GetDlgItemTextA

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.

|                   |                            |                          |
|-------------------|----------------------------|--------------------------|
| 00456CA7  . FFD7  | CALL EDI                   | ; \GetDlgItemTextA       |
| 00456CA9  . 85C0  | TEST EAX,EAX               | ; <== Name must be input |
| 00456CAB  . 7F 41 | JG SHORT cdburner.00456CEE | ; <== Go to next check   |

- Trace tiếp ta đến đây :

|                         |                        |                      |
|-------------------------|------------------------|----------------------|
| 00456D17  . 50          | PUSH EAX               | ; /Arg1              |
| 00456D18  . E8 630F0000 | CALL cdburner.00457C80 | ; \cdburner.00457C80 |

- Dùng F7 trace into ta đến L

|                         |                        |                      |
|-------------------------|------------------------|----------------------|
| 00457C92  . 57          | PUSH EDI               | ; /Arg1              |
| 00457C93  . BB 02000000 | MOV EBX,2              | ;                    |
| 00457C98  . E8 93FFFFFF | CALL cdburner.00457B30 | ; \cdburner.00457B30 |

- Tiếp tục dùng F7 ta trace into tiếp :

|                         |                            |                       |
|-------------------------|----------------------------|-----------------------|
| 00457B5D  . 50          | PUSH EAX                   | ; <== Fake Serial     |
| 00457B5E  . 68 70004A00 | PUSH cdburner.004A0070     |                       |
| 00457B63  . FF52 30     | CALL DWORD PTR DS:[EDX+30] | ; <== Trace Into here |

- Dùng F7 trace into thêm lần nữa ta đến quá trình mã hoá :

|                      |                                           |                                      |
|----------------------|-------------------------------------------|--------------------------------------|
| 0034C8F7 83F9 01     | CMP ECX,1                                 | ; <== Len.S atleast 1 char           |
| 0034C8FA 72 41       | JB SHORT utilitie.0034C93D                |                                      |
| 0034C8FC 56          | PUSH ESI                                  | ; <== Fake Serial                    |
| 0034C8FD E8 3E850000 | CALL utilitie.?StringIsNumeric@@YA_NPBD@Z | ; <== All charts of S must be number |
| 0034C902 83C4 04     | ADD ESP,4                                 |                                      |
| 0034C905 84C0        | TEST AL,AL                                |                                      |
| 0034C907 74 34       | JE SHORT utilitie.0034C93D                |                                      |
| 0034C909 8D4424 0C   | LEA EAX,DWORD PTR SS:[ESP+C]              |                                      |
| 0034C90D 50          | PUSH EAX                                  |                                      |
| 0034C90E 68 CC0B3700 | PUSH utilitie.00370BCC                    | ; ASCII "%u"                         |
| 0034C913 56          | PUSH ESI                                  | ; <== Fake Serial                    |

```
0034C914 E8 9EE60000 CALL utilite.0035AFB7 ;<== Convert to HEX value : ValueS
0034C919 8B4C24 18 MOV ECX,DWORD PTR SS:[ESP+18] ;<== ValueS
0034C91D 33D2 XOR EDX,EDX
0034C91F 8BC1 MOV EAX,ECX ;<== ValueS
0034C921 BE 33909700 MOV ESI,979033 ;<== Default Value : ValueD
0034C926 F7F6 DIV ESI ;<== Over = ValueS % ValueD
0034C928 83C4 0C ADD ESP,0C
0034C92B 85D2 TEST EDX,EDX ;<== Over must be ZERO
0034C92D 75 0E JNZ SHORT utilite.0034C93D
0034C92F 81F9 80969800 CMP ECX,989680 ;<== ValueS must be greater than 0x989680
```

0034C935 72 06 JB SHORT utilite.0034C93D

- Như vậy ở đây ta nhận ra ngay giá trị đúng của Serial phải lớn hơn 0x989680 và là bội số của 0x979033
- Tuy nhiên, chương trình này có hai lần kiểm tra chuỗi Serial . Nếu là kiểm ra trên chưa đúng thì chương trình tiến hành kiểm tra đợt thứ hai :

```

00457D0D |> \57 PUSH EDI ; /Arg1 = 075BCD15
00457D0E |. 8BCE MOV ECX,ESI ; |
00457D10 |. E8 2BFFFFFF CALL cdburner.00457C40 ; \cdburner.00457C40
- Trace tiếp ta đến :
00457C6E |. FF52 34 CALL DWORD PTR DS:[EDX+34] ; <== Trace Into Here
- Dùng F7 trace into ta đến quá trình thứ hai :
0034CC37 83F9 01 CMP ECX,1 ; <== Len.S atleast 1 char
0034CC3A 72 3E JB SHORT utilitie.0034CC7A
0034CC3C 56 PUSH ESI ; <== Fake Serial
0034CC3D E8 FE810000 CALL utilitie.?StringIsNumeric@@YA_NPBD@Z ; <== All charts of S
must be number.

```

must be number  
0034CC42 83C4 04 ADD ESP,4  
0034CC45 84C0 TEST AL,AL  
0034CC47 74 31 JE SHORT utilitie.0034CC7A  
0034CC49 8D4424 08 LEA EAX,DWORD PTR SS:[ESP+8]  
0034CC4D 50 PUSH EAX  
0034CC4E 68 CC0B3700 PUSH utilitie.00370BCC ; ASCII "%u"  
0034CC53 56 PUSH ESI ; <== Fake Serial  
0034CC54 E8 5EE30000 CALL utilitie.0035AFB7 ; <== Convert to HEX value : ValueS  
0034CC59 8B4424 14 MOV EAX,DWORD PTR SS:[ESP+14] ; <== ValueS  
0034CC5D 83C4 0C ADD ESP,0C  
0034CC60 3D 80969800 CMP EAX,989680 ; <== ValueS must be greater than 0x989680  
0034CC65 72 13 JB SHORT utilitie.0034CC7A  
0034CC67 33D2 XOR EDX,EDX  
0034CC69 B9 7DE83200 MOV ECX,32E87D ; <== Default Value : ValueD  
0034CC6E F7F1 DIV ECX ; <== Over = ValueS % ValueD  
0034CC70 85D2 TEST EDX,EDX ; <== Over must be ZERO  
0034CC72 75 06 JNZ SHORT utilitie.0034CC7A

- Như vậy, ở đây Serial cũng phải thoả hai điều kiện (1) lớn hơn 0x989680, và (2) bội số của 0x32E87D

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : 1062815057

### **III – KeyGen :**

/Section 1/- Điều kiện thứ nhất là phải nhỏ hơn 0x989680

/Section 2/- Điều kiện thứ hai (1) hoặc là bội số của 0x32E87D (2) hoặc bội số của 0x979033

**IV – SourceCode ( VC++ ) :**

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenUser=0;
unsigned long int Serial=0;
 LenUser=GetDlgItemText(IDC_Name,reaName,64);
 if (LenUser < 1)
 {
 MessageBox("----- Your name atleast 1 chart -----","Hey !!
Please input your name again !!");
 }
 else
 {
 Serial = 0x979033 * (2 + GetTickCount()% 430);
 wsprintf(reaSerial,"%lu",Serial);
 SetDlgItemText(IDC_Serial,reaSerial);
 }
}

```

**V – End of Tut :**

- Finished – ***14/07/2004***

# Reverse Engineering Association SoftWare

|                     |          |                                                                        |
|---------------------|----------|------------------------------------------------------------------------|
| <b>Homepage</b>     | <b>:</b> | <b><a href="http://www.emailarms.com">http://www.emailarms.com</a></b> |
| <b>Production</b>   | <b>:</b> | <b>F-Key Solutions, Inc.</b>                                           |
| <b>SoftWare</b>     | <b>:</b> | <b>Advanced Emailer 2.6</b>                                            |
| <b>Copyright by</b> | <b>:</b> | <b>Copyright © 2004 F-Key Solutions, Inc. All Rights Reserved.</b>     |
| <b>Type</b>         | <b>:</b> | <b>Serial</b>                                                          |
| <b>Packed</b>       | <b>:</b> | <b>N / A</b>                                                           |
| <b>Language</b>     | <b>:</b> | <b>Borland Delphi 6.0 - 7.0</b>                                        |
| <b>Crack Tool</b>   | <b>:</b> | <b>OllyDbg 1.09d, PEiD 0.92, kWdsdm 10</b>                             |
| <b>Unpack</b>       | <b>:</b> | <b>N / A</b>                                                           |
| <b>Request</b>      | <b>:</b> | <b>Correct Serial / KeyGen</b>                                         |

## **Advanced Emailer 2.6**

Advanced Emailer is intended for sending requested email newsletters or notifications utilizing subscription-based mailing lists. This program allows you to create and manage a customer database and generate personalized messages from predefined templates using a variety of data fields such as Name, Age, Gender, Address, Country and so on. It lets you have for example order numbers, member IDs and other data in the database along with email addresses and names and use them to generate individual messages with order or membership status and so on. You just use macro substitution patterns, to be replaced with information from the database, for each customer, right before dispatching. You can also add your own custom fields to the database if you need to have more information on each customer. Using tab delimited or CSV format you can synchronize the program with virtually any database system such as Microsoft Access, Excel, SQL, Oracle and so on. You can use all the standard message formats

like plain text, HTML or even create a rich content message in the Microsoft Outlook Express and export it into the program.

## I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**". Ta tìm được thông báo này tại địa chỉ :

005088E8 |. BA FC895000 MOV EDX,mailer.005089FC ; ASCII "Registration code is invalid!"

- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

005087D3 |. E8 B03BF7FF CALL mailer.0047C388 ; <== Set BreakPoint here

## II – Cracking :

- Quá trình mã hoá của chương trình này rất đơn giản . Từ BP ta trace xuống chút :

005087F0 |. E8 93FDFFFF CALL mailer.00508588 ; <== Trace Into

----- Trace Into -----

|                         |                            |                              |
|-------------------------|----------------------------|------------------------------|
| 00508596  . E8 A1C7EFFF | CALL mailer.00404D3C       | ; <== Get Length Serial      |
| 0050859B  . 83F8 0E     | CMP EAX,0E                 | ; <== LenS must be 14 charts |
| 0050859E  . 75 67       | JNZ SHORT mailer.00508607  |                              |
| 005085A0  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085A2  . 8038 38     | CMP BYTE PTR DS:[EAX],38   | ; <== S[0] = 0x38            |
| 005085A5  . 0F94C0      | SETE AL                    |                              |
| 005085A8  . 83E0 7F     | AND EAX,7F                 |                              |
| 005085AB  . 03F0        | ADD ESI,EAX                |                              |
| 005085AD  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085AF  . 8078 02 36  | CMP BYTE PTR DS:[EAX+2],36 | ; <== S[2] = 0x36            |
| 005085B3  . 0F94C0      | SETE AL                    |                              |
| 005085B6  . 83E0 7F     | AND EAX,7F                 |                              |
| 005085B9  . 03F0        | ADD ESI,EAX                |                              |
| 005085BB  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085BD  . 8078 03 32  | CMP BYTE PTR DS:[EAX+3],32 | ; <== S[3] = 0x32            |
| 005085C1  . 0F94C0      | SETE AL                    |                              |
| 005085C4  . 83E0 7F     | AND EAX,7F                 |                              |
| 005085C7  . 03F0        | ADD ESI,EAX                |                              |
| 005085C9  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085CB  . 8078 04 37  | CMP BYTE PTR DS:[EAX+4],37 | ; <== S[4] = 0x37            |
| 005085CF  . 0F94C0      | SETE AL                    |                              |
| 005085D2  . 83E0 7F     | AND EAX,7F                 |                              |
| 005085D5  . 03F0        | ADD ESI,EAX                |                              |
| 005085D7  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085D9  . 8078 07 39  | CMP BYTE PTR DS:[EAX+7],39 | ; <== S[7] = 0x39            |
| 005085DD  . 0F94C0      | SETE AL                    |                              |
| 005085E0  . 83E0 7F     | AND EAX,7F                 |                              |
| 005085E3  . 03F0        | ADD ESI,EAX                |                              |
| 005085E5  . 8B07        | MOV EAX,DWORD PTR DS:[EDI] |                              |
| 005085E7  . 8078 08 34  | CMP BYTE PTR DS:[EAX+8],34 | ; <== S[8] = 0x34            |
| 005085EB  . 0F94C0      | SETE AL                    |                              |

```

005085EE |. 83E0 7F AND EAX,7F
005085F1 |. 03F0 ADD ESI,EAX
005085F3 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
005085F5 |. 8078 0A 30 CMP BYTE PTR DS:[EAX+A],30 ; <== S[10] = 0x30
005085F9 |. 0F94C0 SETE AL
005085FC |. 83E0 7F AND EAX,7F
005085FF |. 03F0 ADD ESI,EAX
00508601 |. 83FE 07 CMP ESI,7 ; <== if ALL correct
00508604 |. 0F94C3 SETE BL ; <== EBX = 0x1
00508607 |> 8BC3 MOV EAX,EBX ; <== EAX = EBX
===== Trace Into =====

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 8Y6279Y94C0XYN

### III – KeyGen :

- /Section I /- Chuỗi Serial có chiều dài là 14 ký tự .
- /Section II /- Các vị trí [0][2][3][4][7][8][10] là các ký tự mặc định .

### IV – End of Tut :

- Finished – *August 23, 2004*

# Reverse Engineering Association SoftWare

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| Homepage :     | <a href="http://www.alivemedia.net">http://www.alivemedia.net</a> |
| Production :   | Alive Media, Inc.                                                 |
| SoftWare :     | Alive CD Ripper                                                   |
| Copyright by : | Copyright © 2003 Alive Media, Inc. All Rights Reserved.           |
| Type :         | Name / Serial                                                     |
| Packed :       | ASPack 2.12 -> Alexey Solodovnikov                                |
| Language :     | Borland Delphi 6.0 - 7.0                                          |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsdm 10</b>                        |
| Unpack :       | Manual                                                            |
| Request :      | Correct Serial / KeyGen                                           |

### Alive CD Ripper

Alive CD Ripper is a powerful and easy to use cd ripper tool. Convert your favorite CD tracks to MP3, WAV, WMA, OGG, VOX! Features: 1) Extract audio CDs into MP3, WAV, WMA, OGG, VOX on the fly. 2) Best Audio Codecs Support. 3) Freedb Enabled. 4) Local CD Database. 5) Full CD Playback. 6) Multiple CD drivers support. 7) Lifetime updates.

### I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **ASPack 2.12 -> Alexey Solodovnikov** . UnPACK và kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid Registration Code!**  
**Please enter an available Registration Code.**". Ta tìm được thông báo này tại địa chỉ :  
0049D4F6 |. 68 6CD54900 PUSH unpack.0049D56C ; |Text = "Invalid Registration Code!"  
Please enter an available Registration Code."

- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :  
0049D462 |. E8 D975F6FF CALL unpack.00404A40 ; <== Set BreakPoint here

## II – Cracking :

- Chạy thử chương trình với User và Fake Serial chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến :

|                                              |                  |
|----------------------------------------------|------------------|
| 0049D487  . E8 30010000 CALL unpack.0049D5BC | ; <== Trace Into |
| 0049D48C  . 8B55 F4 MOV EDX,[LOCAL.3]        | ; <== RealSerial |
| 0049D48F  . 8B45 F8 MOV EAX,[LOCAL.2]        | ; <== FakeSerial |
| 0049D492  . E8 D5B4F6FF CALL unpack.0040896C | ; <== Compare    |

- Dùng F7 để trace Into ta sẽ đến quá trình mã hoá chuỗi . Trước hết là quá trình tạo chuỗi để mã hoá . Chương trình sẽ kết hợp chuỗi U nhập với chuỗi mặc định “8ripper5793” . Chuỗi kết hợp này sẽ được chia đôi thành hai chuỗi, hai chuỗi này sẽ được đảo ngược lại để tạo thành chuỗi mới . Sau đó, dựa trên chuỗi đảo ngược này, chương trình sẽ cắt làm hai chuỗi để tiến hành mã hoá (1) chuỗi đầu tiên gồm 10 ký tự đầu tiên (2) chuỗi thứ hai gồm tất cả các ký tự còn lại, bắt đầu từ ký tự thứ 6 .

|                                                  |                                      |
|--------------------------------------------------|--------------------------------------|
| 0049D5F2  . BA 30D74900 MOV EDX,unpack.0049D730  | ; ASCII "8ripper5793"                |
| 0049D5F7  . E8 6472F6FF CALL unpack.00404860     | ; <== Concat(User,DefaultString)     |
| 0049D5FC  . 8B45 FC MOV EAX,[LOCAL.1]            | ; <== cStr                           |
| 0049D5FF  . E8 5472F6FF CALL unpack.00404858     | ; <== Len.cStr                       |
| 0049D604  . 8BF0 MOV ESI,EAX                     | ; <== Len.cStr                       |
| 0049D606  . D1FE SAR ESI,1                       | ; <== Temp = Len.cStr / 0x2          |
| 0049D608  . 79 03 JNS SHORT unpack.0049D60D      |                                      |
| 0049D60A  . 83D6 00 ADC ESI,0                    |                                      |
| 0049D60D  > 8D45 F0 LEA EAX,[LOCAL.4]            |                                      |
| 0049D610  . 50 PUSH EAX                          |                                      |
| 0049D611  . 8BCE MOV ECX,ESI                     | ; <== get Temp charts                |
| 0049D613  . BA 01000000 MOV EDX,1                | ; <== from the 1st char              |
| 0049D618  . 8B45 FC MOV EAX,[LOCAL.1]            | ; <== of cStr                        |
| 0049D61B  . E8 9074F6FF CALL unpack.00404AB0     | ; <== Ripping : tStrI                |
| 0049D620  . 8B45 F0 MOV EAX,[LOCAL.4]            |                                      |
| 0049D623  . 50 PUSH EAX                          |                                      |
| 0049D624  . 8D45 EC LEA EAX,[LOCAL.5]            |                                      |
| 0049D627  . 50 PUSH EAX                          |                                      |
| 0049D628  . 8B45 FC MOV EAX,[LOCAL.1]            |                                      |
| 0049D62B  . E8 2872F6FF CALL unpack.00404858     |                                      |
| 0049D630  . 8BC8 MOV ECX,EAX                     |                                      |
| 0049D632  . 8D56 01 LEA EDX,DWORD PTR DS:[ESI+1] | ; <== get (Len.cStr-Temp) charts     |
| 0049D635  . 8B45 FC MOV EAX,[LOCAL.1]            | ; <== of cStr                        |
| 0049D638  . E8 7374F6FF CALL unpack.00404AB0     | ; <== Ripping : tStrII               |
| 0049D63D  . 8B55 EC MOV EDX,[LOCAL.5]            | ; <== tStrII                         |
| 0049D640  . 8D45 FC LEA EAX,[LOCAL.1]            |                                      |
| 0049D643  . 59 POP ECX                           | ; <== tStrI                          |
| 0049D644  . E8 5B72F6FF CALL unpack.004048A4     | ; <== Concat(tStrII, tStrI) : reaStr |
| 0049D649  . 8D45 F8 LEA EAX,[LOCAL.2]            |                                      |
| 0049D64C  . 50 PUSH EAX                          |                                      |
| 0049D64D  . B9 0A000000 MOV ECX,0A               | ; <== Get 10 charts                  |

```

0049D652 |. BA 01000000 MOV EDX,1 ; <== from the FIRST chart
0049D657 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of reaStr
0049D65A |. E8 5174F6FF CALL unpack.00404AB0 ; <== Cutting : SecI
0049D65F |. 8D45 F4 LEA EAX,[LOCAL.3]
0049D662 |. 50 PUSH EAX
0049D663 |. 8B45 FC MOV EAX,[LOCAL.1]
0049D666 |. E8 ED71F6FF CALL unpack.00404858
0049D66B |. 8BC8 MOV ECX,EAX ; <== Get all charts
0049D66D |. BA 06000000 MOV EDX,6 ; <== from the 6th chart
0049D672 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of reaU
0049D675 |. E8 3674F6FF CALL unpack.00404AB0 ; <== Cutting : SecII

```

- Quá trình mã hoá được tiến hành như sau :

```

0049D691 |. 8B4D F4 MOV ECX,[LOCAL.3] ; <== SecII
0049D694 |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== SecI
0049D697 |. 8BC7 MOV EAX,EDI

0049D699 |. E8 12F1FFFF CALL unpack.0049C7B0 ; <== Encrypt
----- Encrypt -----
0049C805 |. BB 00010000 MOV EBX,100 ; <== Default String : dStr
0049C80A |. 8D45 F0 LEA EAX,[LOCAL.4]
0049C80D |. 50 PUSH EAX ; /Arg1
0049C80E |. C745 E4 00010>MOV [LOCAL.7],100 ; |
0049C815 |. C645 E8 00 MOV BYTE PTR SS:[EBP-18],0 ; |
0049C819 |. 8D55 E4 LEA EDX,[LOCAL.7] ; |
0049C81C |. 33C9 XOR ECX,ECX ; |
0049C81E |. B8 F8C84900 MOV EAX,unpack.0049C8F8 ; |ASCII "%1.2x"
0049C823 |. E8 D8D3F6FF CALL unpack.00409C00 ; \unpack.00409C00
0049C828 |. 8B45 FC MOV EAX,[LOCAL.1]
0049C82B |. E8 2880F6FF CALL unpack.00404858 ; <== Get length of SecI
0049C830 |. 8BF8 MOV EDI,EAX ; <== NumberLoop = LenSecI : nL
0049C832 |. 85FF TEST EDI,EDI
0049C834 |. 7E 60 JLE SHORT unpack.0049C896
0049C836 |. C745 EC 01000>MOV [LOCAL.5],1 ; <== i = 1
0049C83D |> 8B45 FC /MOV EAX,[LOCAL.1] ; <== SecI
0049C840 |. 8B55 EC |MOV EDX,[LOCAL.5] ; <== i
0049C843 |. 0FB64410 FF |MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== SecII[i-1]
0049C848 |. 03C3 |ADD EAX,EBX ; <== Temp = Temp + SecII[i-1]
0049C84A |. B9 FF000000 |MOV ECX,0FF ; <== dV
0049C84F |. 99 |CDQ
0049C850 |. F7F9 |IDIV ECX ; <== Temp = Temp % dV
0049C852 |. 8BDA |MOV EBX,EDX ; <== Temp
0049C854 |. 3B75 F4 |CMP ESI,[LOCAL.3] ; <== while (j < LenSecII)
0049C857 |. 7D 03 |JGE SHORT unpack.0049C85C ; <== then
0049C859 |. 46 |INC ESI ; <== j++
0049C85A |. EB 05 |JMP SHORT unpack.0049C861 ; <== else
0049C85C |> BE 01000000 |MOV ESI,1 ; <== j = 0x1
0049C861 |> 8B45 F8 |MOV EAX,[LOCAL.2] ; <== SecII
0049C864 |. 0FB64430 FF |MOVZX EAX,BYTE PTR DS:[EAX+ESI-1] ; <== SecII[i-1]
0049C869 |. 33D8 |XOR EBX,EAX ; <== Temp = Temp xor SecII[i-1]
0049C86B |. 8D45 E0 |LEA EAX,[LOCAL.8] ; <== Convert Value to String : tStr
0049C86E |. 50 |PUSH EAX ; /Arg1

```

```

0049C86F |. 895D E4 |MOV [LOCAL.7],EBX ;|
0049C872 |. C645 E8 00 |MOV BYTE PTR SS:[EBP-18],0 ;|
0049C876 |. 8D55 E4 |LEA EDX,[LOCAL.7] ;|
0049C879 |. 33C9 |XOR ECX,ECX ;|
0049C87B |. B8 F8C84900 |MOV EAX,unpack.0049C8F8 ;|ASCII "%1.2x"
0049C880 |. E8 7BD3F6FF |CALL unpack.00409C00 ;|\unpack.00409C00
0049C885 |. 8B55 E0 |MOV EDX,[LOCAL.8] ;<== tStr
0049C888 |. 8D45 F0 |LEA EAX,[LOCAL.4] ;<== dStr
0049C88B |. E8 D07FF6FF |CALL unpack.00404860 ;<== Concat(dStr,tStr)
0049C890 |. FF45 EC |INC [LOCAL.5] ;<== i++
0049C893 |. 4F |DEC EDI ;<== while (nL > 0)
0049C894 |.^ 75 A7 \JNZ SHORT unpack.0049C83D ;<== Continue loop
----- Encrypt -----

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 10032-EF4DC-D0245

### III – KeyGen :

/Section I /- Kết hợp với chuỗi mặc định và đảo ngược chuỗi này .

/Section II /- Chia chuỗi làm hai phần và tiến hành mã hoá .

### IV – End of Tut :

- Finished – *August 24, 2004*

# Reverse Engineering Association SoftWare

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| Homepage :     | <a href="http://www.alivemedia.net">http://www.alivemedia.net</a> |
| Production :   | Alive Media, Inc.                                                 |
| SoftWare :     | <b>Alive MP3 CD Burner</b>                                        |
| Copyright by : | Copyright © 2003 Alive Media, Inc. All Rights Reserved.           |
| Type :         | Name / Serial                                                     |
| Packed :       | ASPack 2.12 -> Alexey Solodovnikov                                |
| Language :     | Borland Delphi 6.0 - 7.0                                          |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWds 10</b>                          |
| Unpack :       | Manual                                                            |
| Request :      | Correct Serial / KeyGen                                           |

### Alive MP3 CD Burner

Alive MP3 CD Burner is a professional music cd tool, which lets you burn your favorite CDs from MP3, WAV or OGG files! 1) Burn custom music CDs from MP3, WAV or OGG Vorbis files on-the-fly. 2) ID3 tags editor for free. 3) Supports testing, burning. 4) Supports use of high write speeds. 5) Easily Create CD Labels for any CD. 6) Built-in MP3, WAV or OGG Player. 7) Multiple CD-RW drivers support. 8) Lifetime updates.

### I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng *ASPack 2.12 -> Alexey Solodovnikov* . UnPACK và kiểm tra lại biết chương trình được viết bằng *Borland Delphi 6.0 - 7.0*

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid Registration Code!**  
**Please enter an available Registration Code.**". Ta tìm được thông báo này tại địa chỉ :  
00482506 |. 68 C8254800 PUSH unpack.004825C8 ; |Text = "Invalid Registration Code!"  
Please enter an available Registration Code."

- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :  
00482456 |. E8 2925F8FF CALL unpack.00404984 ; <== Set BreakPoint here

## II – Cracking :

- Chạy thử chương trình với User và Fake Serial chương trình dừng lại tại điểm đặt BP.Trace xuống chút :  
00482497 |. E8 78010000 CALL unpack.00482614 ; <== Trace Into  
0048249C |. 8B55 F4 MOV EDX,[LOCAL.3] ; <== RealSerial

0048249F |. 8B45 F8 MOV EAX,[LOCAL.2] ; <== FakeSerial  
004824A2 |. E8 FD62F8FF CALL unpack.004087A4 ; <== Compare

- Dùng F7 để trace Into ta sẽ đến quá trình mã hoá chuỗi . Trước hết là quá trình tạo chuỗi để mã hoá . Chương trình sẽ kết hợp chuỗi U nhập với chuỗi mặc định "**B1u6rIn8e9r**" . Chuỗi kết hợp này sẽ được chia đôi thành hai chuỗi, hai chuỗi này sẽ được đảo ngược lại để tạo thành chuỗi mới . Sau đó, dựa trên chuỗi đảo ngược này, chương trình sẽ cắt làm hai chuỗi để tiến hành mã hoá (1) chuỗi đầu tiên gồm 10 ký tự đầu tiên (2) chuỗi thứ hai gồm tất cả các ký tự còn lại, bắt đầu từ ký tự thứ 6 .

```
0048264A |. BA 88274800 MOV EDX,unpack.00482788 ; ASCII "B1u6r1n8e9r"
0048264F |. E8 5021F8FF CALL unpack.004047A4 ; <== Concat(User,DefaultString)
00482654 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== cStr
00482657 |. E8 4021F8FF CALL unpack.0040479C ; <== Len.cStr
0048265C |. 8BF0 MOV ESI,EAX ; <== Len.cStr
0048265E |. D1FE SAR ESI,1 ; <== Temp = Len.cStr / 0x2
00482660 |. 79 03 JNS SHORT unpack.00482665
00482662 |. 83D6 00 ADC ESI,0
00482665 > 8D45 F0 LEA EAX,[LOCAL.4]
00482668 |. 50 PUSH EAX
00482669 |. 8BCE MOV ECX,ESI ; <== get Temp charts
0048266B |. BA 01000000 MOV EDX,1 ; <== from the 1st char
00482670 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of cStr
00482673 |. E8 7C23F8FF CALL unpack.004049F4 ; <== Ripping : tStrI
00482678 |. 8B45 F0 MOV EAX,[LOCAL.4]
0048267B |. 50 PUSH EAX
0048267C |. 8D45 EC LEA EAX,[LOCAL.5]
0048267F |. 50 PUSH EAX
00482680 |. 8B45 FC MOV EAX,[LOCAL.1]
00482683 |. E8 1421F8FF CALL unpack.0040479C
00482688 |. 8BC8 MOV ECX,EAX
0048268A |. 8D56 01 LEA EDX,DWORD PTR DS:[ESI+1] ; <== get (Len.cStr-Temp) charts
0048268D |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of cStr
00482690 |. E8 5F23F8FF CALL unpack.004049F4 ; <== Ripping : tStrII
00482695 |. 8B55 EC MOV EDX,[LOCAL.5] ; <== tStrII
00482698 |. 8D45 FC LEA EAX,[LOCAL.1]
0048269B |. 59 POP ECX ; <== tStrI
```

```

0048269C |. E8 4721F8FF CALL unpack.004047E8 ; <== Concat(tStrII, tStrI) : reaStr
004826A1 |. 8D45 F8 LEA EAX,[LOCAL.2]
004826A4 |. 50 PUSH EAX
004826A5 |. B9 0A000000 MOV ECX,0A ; <== Get 10 charts
004826AA |. BA 01000000 MOV EDX,1 ; <== from the FIRST chart
004826AF |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of reaStr
004826B2 |. E8 3D23F8FF CALL unpack.004049F4 ; <== Cutting : SecI
004826B7 |. 8D45 F4 LEA EAX,[LOCAL.3]
004826BA |. 50 PUSH EAX
004826BB |. 8B45 FC MOV EAX,[LOCAL.1]
004826BE |. E8 D920F8FF CALL unpack.0040479C
004826C3 |. 8BC8 MOV ECX,EAX ; <== Get all charts
004826C5 |. BA 06000000 MOV EDX,6 ; <== from the 6th chart
004826CA |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of reaStr
004826CD |. E8 2223F8FF CALL unpack.004049F4 ; <== Cutting : SecII

```

- Quá trình mã hoá được tiến hành như sau :

```

004826E9 |. 8B4D F4 MOV ECX,[LOCAL.3] ; <== SecII
004826EC |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== SecI
004826EF |. 8BC7 MOV EAX,EDI
004826F1 |. E8 1EF0FFFF CALL unpack.00481714 ; <== Encrypt
-----== Encrypt ==
00481769 |. BB 00010000 MOV EBX,100 ; <== Default String : dStr
0048176E |. 8D45 F0 LEA EAX,[LOCAL.4]
00481771 |. 50 PUSH EAX ; /Arg1
00481772 |. C745 E4 00010>MOV [LOCAL.7],100 ; |
00481779 |. C645 E8 00 MOV BYTE PTR SS:[EBP-18],0 ; |
0048177D |. 8D55 E4 LEA EDX,[LOCAL.7] ; |
00481780 |. 33C9 XOR ECX,ECX ; |
00481782 |. B8 5C184800 MOV EAX,unpack.0048185C ; |ASCII "%1.2x"
00481787 |. E8 C87FF8FF CALL unpack.00409754 ; \unpack.00409754
0048178C |. 8B45 FC MOV EAX,[LOCAL.1]
0048178F |. E8 0830F8FF CALL unpack.0040479C ; <== Get length of SecI
00481794 |. 8BF8 MOV EDI,EAX ; <== NumberLoop = LenSecI : nL
00481796 |. 85FF TEST EDI,EDI
00481798 |. 7E 60 JLE SHORT unpack.004817FA
0048179A |. C745 EC 01000>MOV [LOCAL.5],1 ; <== i = 1
004817A1 |> 8B45 FC /MOV EAX,[LOCAL.1] ; <== SecI
004817A4 |. 8B55 EC |MOV EDX,[LOCAL.5] ; <== i
004817A7 |. 0FB64410 FF |MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== SecI[i-1]
004817AC |. 03C3 |ADD EAX,EBX ; <== Temp = Temp + SecI[i-1]
004817AE |. B9 FF000000 |MOV ECX,0FF ; <== dV
004817B3 |. 99 |CDQ
004817B4 |. F7F9 |IDIV ECX ; <== Temp = Temp % dV
004817B6 |. 8BDA |MOV EBX,EDX ; <== Temp
004817B8 |. 3B75 F4 |CMP ESI,[LOCAL.3] ; <== while (j < LenSecII)
004817BB |. 7D 03 |JGE SHORT unpack.004817C0 ; <== then
004817BD |. 46 |INC ESI ; <== j++
004817BE |. EB 05 |JMP SHORT unpack.004817C5 ; <== else
004817C0 |> BE 01000000 |MOV ESI,1 ; <== j = 0x1
004817C5 |> 8B45 F8 |MOV EAX,[LOCAL.2] ; <== SecII

```

```

004817C8 |. 0FB64430 FF |MOVZX EAX,BYTE PTR DS:[EAX+ESI-1] ; <== SecII[i-1]
004817CD |. 33D8 |XOR EBX,EAX ; <== Temp = Temp xor SecII[i-1]
004817CF |. 8D45 E0 |LEA EAX,[LOCAL.8] ; <== Convert Value to String : tStr
004817D2 |. 50 |PUSH EAX ; /Arg1
004817D3 |. 895D E4 |MOV [LOCAL.7],EBX ; |
004817D6 |. C645 E8 00 |MOV BYTE PTR SS:[EBP-18],0 ; |
004817DA |. 8D55 E4 |LEA EDX,[LOCAL.7] ; |
004817DD |. 33C9 |XOR ECX,ECX ; |
004817DF |. B8 5C184800 |MOV EAX,unpack.0048185C ; |ASCII "%1.2x"
004817E4 |. E8 6B7FF8FF |CALL unpack.00409754 ; |\unpack.00409754
004817E9 |. 8B55 E0 |MOV EDX,[LOCAL.8] ; <== tStr
004817EC |. 8D45 F0 |LEA EAX,[LOCAL.4] ; <== dStr
004817EF |. E8 B02FF8FF |CALL unpack.004047A4 ; <== Concat(dStr,tStr)
004817F4 |. FF45 EC |INC [LOCAL.5] ; <== i++
004817F7 |. 4F |DEC EDI ; <== while (nL > 0)
004817F8 |.^ 75 A7 |JNZ SHORT unpack.004817A1 ; <== Continue loop
----- Encrypt -----

```

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm      Serial : 10074-93E47-8D569

### III – KeyGen :

- /Section I/- Kết hợp với chuỗi mặc định và đảo ngược chuỗi này .
- /Section II/- Chia chuỗi làm hai phần và tiến hành mã hoá .

### IV – End of Tut :

- Finished – *August 24, 2004*

# Reverse Engineering Association SoftWare

|                |                                                                       |
|----------------|-----------------------------------------------------------------------|
| Homepage :     | <a href="http://www.alleycode.com">http://www.alleycode.com</a>       |
| Production :   | Konae Technologies, Inc.                                              |
| SoftWare :     | <b>Alleycode HTML Editor 1.3</b>                                      |
| Copyright by : | <b>Copyright © 2003-Konae Technologies, Inc. All Rights Reserved.</b> |
| Type :         | <b>Name / Serial</b>                                                  |
| Packed :       | <b>N / A</b>                                                          |
| Language :     | <b>Borland C++ 1999</b>                                               |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsdm 10</b>                            |
| Unpack :       | <b>N / A</b>                                                          |
| Request :      | <b>Correct Serial / KeyGen</b>                                        |

### **Alleycode HTML Editor 1.3**

**Alleycode HTML 1.3** is an intuitive, robust and fully functional HTML editor. Alleycode introduces innovative features for rapid deployment such as Synchro View...Real time rendition with two way synchronized code/design view. Assignments... For quick access to projects. Turf View... An unusual real estate view of your documents with fast right click control. CSS Wizard... A comprehensive Style Sheet Wizard which creates/uploads either internal or external style sheets (CSS1 - CSS2). Optimizer... Helps you keep your meta content accurate and search engine friendly. These features and many more sit

atop a very fast and thoroughly tested HTML production engine with adjustable syntax highlighting and exhaustive Click'n'Insert HTM3.2 - 4.1, CSS and PHP function libraries. Alleycode is complete with strong Help Content including an excellent step by step Quick Start Tutorial

## I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland C++ 1999**.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Invalid unlock code. Please try again.**" Ta tìm được đoạn CODE này ở địa chỉ :  
0047E8AF |. BA 38EB4700 MOV EDX,alleycod.0047EB38 ; ASCII "Invalid unlock code. Please try again."
- Tuy nhiên khi dò ngược đến địa chỉ của thông báo này, ta thấy FUNCTION này gồm toàn bộ các thông báo xuất hiện trong quá trình đăng ký, và không có bất cứ một câu lệnh điều kiện hay kiểm tra nào . Có thể nghĩ ngay đến một phương pháp ANTICRACK . Tất cả các thông báo sẽ được nạp vào địa chỉ khi khởi động, và trong quá trình mã hoá và so sánh chuỗi, tùy theo từng giá trị trả về mà chương trình sẽ gọi đến địa chỉ lưu thông báo đúng hay sai . Như vậy, FUNCTION này không phải là FUNCTION chính của quá trình mã hoá .
- Để tìm đến được FUNCTION chính của quá trình mã hoá, ta cần sử dụng phương pháp STACK . Sau khi sử dụng phương pháp STACK này, ta xác định được FUNCTION và đặt BreakPoint tại lệnh CALL đầu tiên của Function này :  
0047EEE4 . E8 F5440900 CALL alleycod.005133E4 ; <== Set BreakPoint here

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến:

```
0047EF87 . BA B8F54700 MOV EDX,alleycod.0047F5B8 ; ASCII "GLE140463YM"
0047EF8C . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== Fake Serial
0047EF8F . E8 CC5C0000 CALL
```

[alleycod.@Tdsswfuncdevext@DecryptStr\\$qqrx17System@AnsString1](#)

===== Trace Into =====

```
00484C90 |. 8B45 F4 MOV EAX,[LOCAL.3] ; <== Fake Serial
00484C93 |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== Decrypt String "GLE140463YM"
00484C96 |.E8 21000000 CALL
```

[alleycod.@Tdsswfuncdevext@Decrypt\\$qqrr17System@AnsString17System@AnsString](#)

===== DECRYPT =====

```
00484D4D |. 8945 D8 MOV [LOCAL.10],EAX ; <== Length of DecryptString : LenD
00484D50 |. C745 E8 01000>MOV [LOCAL.6],1 ; <== i = 1
00484D57 |> 8B45 F8 /MOV EAX,[LOCAL.2] ; <== DecryptString : DeStr
00484D5A |. 8B55 E8 |MOV EDX,[LOCAL.6] ; <== i
00484D5D |. 0FB64410 FF |MOVZX EAX,BYTE PTR DS:[EAX+EDX-1] ; <== DeStr[i-1]
00484D62 |. 0145 E4 |ADD [LOCAL.7],EAX ; <== CumV = VumV + DeStr[i-1]
00484D65 |. FF45 E8 |INC [LOCAL.6] ; <== i++
00484D68 |. FF4D D8 |DEC [LOCAL.10] ; <== LenD --
00484D6B |.^ 75 EA |JNZ SHORT alleycod.00484D57 ; <== Loop Until LenD == 0x0
00484D6D |> 8D55 E0 LEA EDX,[LOCAL.8]
00484D70 |. 8B45 E4 MOV EAX,[LOCAL.7] ; <== CumV
00484D73 |. E8 1C570800 CALL alleycod.0050A494 ; <== Convert to String : CumStr
00484D78 |. EB 0B JMP SHORT alleycod.00484D85 ; <== ConCat (CheckStr, CumStr)
00484D7A |> 8D45 E0 /LEA EAX,[LOCAL.8]
```

```

00484D7D |. 8B55 E0 |MOV EDX,[LOCAL.8] ; <== CumStr
00484D80 |. E8 B3E40800 |CALL alleycod.00513238 ; <== ConCat (CheckStr, CumStr)
00484D85 |> 8B45 E0 |MOV EAX,[LOCAL.8] ; <== CheckStr
00484D88 |. E8 A3E40800 |CALL alleycod.00513230 ; <== LenCheckStr
00484D8D |. 3B45 F0 |CMP EAX,[LOCAL.4] ; <== while (LenCheckStr < 0x18)
00484D90 |.^ 7C E8 |JL SHORT alleycod.00484D7A ; <== Continue ConCat
00484D92 |. 8D45 DC |LEA EAX,[LOCAL.9]
00484D95 |. 8B55 FC |MOV EDX,[LOCAL.1]
00484D98 |. 8B12 |MOV EDX,DWORD PTR DS:[EDX]
00484D9A |. E8 A9E20800 |CALL alleycod.00513048
00484D9F |. 8B45 FC |MOV EAX,[LOCAL.1]
00484DA2 |. E8 09E20800 |CALL alleycod.00512FB0 ; <== LenS
00484DA7 |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== LenS
00484DAA |. 85C0 |TEST EAX,EAX
00484DAC |. 7E 51 |JLE SHORT alleycod.00484DFF
00484DAE |. 8945 D8 |MOV [LOCAL.10],EAX ; <== LenS
00484DB1 |. C745 E8 01000>MOV [LOCAL.6],1 ; <== i = 1
00484DB8 |> 8D45 D0 |/EA EAX,[LOCAL.12]
00484DBB |. 8B55 E0 |MOV EDX,[LOCAL.8] ; <== CheckStr
00484DBE |. 8B4D E8 |MOV ECX,[LOCAL.6] ; <== i
00484DC1 |. 8A540A FF |MOV DL,BYTE PTR DS:[EDX+ECX-1] ; <== CheckStr[i-1]
00484DC5 |. E8 8EE30800 |CALL alleycod.00513158
00484DCA |. 8B45 D0 |MOV EAX,[LOCAL.12]
00484DCD |. E8 A2570800 |CALL alleycod.0050A574 ; <== Temp = CheckStr[i-1] - 0x30
00484DD2 |. 8BD0 |MOV EDX,EAX
00484DD4 |. 8B45 DC |MOV EAX,[LOCAL.9] ; <== FakeSerial : fS
00484DD7 |. 8B4D E8 |MOV ECX,[LOCAL.6] ; <== i
00484DDA |. 025408 FF |ADD DL,BYTE PTR DS:[EAX+ECX-1] ; <== Temp = Temp + fS[i-1]
00484DDE |. 80EA 0B |SUB DL,0B ; <== Temp = Temp - 0xB
00484DE1 |. 8D45 D4 |LEA EAX,[LOCAL.11]
00484DE4 |. E8 6FE30800 |CALL alleycod.00513158 ; <== Convert to ASCII
00484DE9 |. 8B55 D4 |MOV EDX,[LOCAL.11]
00484DEC |. 8B45 FC |MOV EAX,[LOCAL.1]
00484DEF |. E8 44E40800 |CALL alleycod.00513238 ; <== Concat all charts : NewStr
00484DF4 |. 8B45 FC |MOV EAX,[LOCAL.1]
00484DF7 |. FF45 E8 |INC [LOCAL.6] ; <== i++
00484DFA |. FF4D D8 |DEC [LOCAL.10] ; <== LenS --
00484DFD |.^ 75 B9 |JNZ SHORT alleycod.00484DB8 ; <== Loop Until LenS == 0x0
-----DECRYPT-----
-----Trace Into-----

```

- Kết thúc quá trình này, chuỗi Serial được giải mã thành một chuỗi mới : NewStr.

- Trace tiếp ta đến quá trình so sánh thứ nhất :

```

0047F281 . 50 PUSH EAX ; <== At 12th
0047F282 . B9 0C000000 MOV ECX,0C ; <== Get 12 chart
0047F287 . 8B55 CC MOV EDX,DWORD PTR SS:[EBP-34]
0047F28A . 8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18] ; <== from NewStr : TempStr
0047F28D . E8 A6410900 CALL alleycod.00513438 ; <== Cutting
0047F292 . 8D85 6CFFFFFF LEA EAX,DWORD PTR SS:[EBP-94]
0047F298 . 50 PUSH EAX
0047F299 . 8D85 68FFFFFF LEA EAX,DWORD PTR SS:[EBP-98]

```

```

0047F29F . B9 CCF54700 MOV ECX,alleycod.0047F5CC ; ASCII
"ASEFTJHTIOPDEFPLFJMJKNMH"
0047F2A4 . 8B55 E4 MOV EDX,DWORD PTR SS:[EBP-1C] ; <== User
0047F2A7 . E8 D03F0900 CALL alleycod.0051327C ; <== Concat string
0047F2AC . 8B85 68FFFFFF MOV EAX,DWORD PTR SS:[EBP-98] ; <== String after Concat
0047F2B2 . B9 0C000000 MOV ECX,0C ; <== Get 12 charts
0047F2B7 . BA 01000000 MOV EDX,1 ; <== from FIRST chart
0047F2BC . E8 77410900 CALL alleycod.00513438 ; <== Cutting : uStr
0047F2C1 . 8B85 6CFFFFFF MOV EAX,DWORD PTR SS:[EBP-94] ; <== uStr
0047F2C7 . 8D95 70FFFFFF LEA EDX,DWORD PTR SS:[EBP-90]
0047F2CD . E8 FEAD0800 CALL alleycod.0050A0D0 ; <== CharUpper (uStr)
0047F2D2 . 8B95 70FFFFFF MOV EDX,DWORD PTR SS:[EBP-90] ; <== uStr
0047F2D8 . 8D45 E4 LEA EAX,DWORD PTR SS:[EBP-1C]
0047F2DB . E8 683D0900 CALL alleycod.00513048
0047F2E0 . 8B45 E0 MOV EAX,DWORD PTR SS:[EBP-20] ; <== TempStr
0047F2E3 . 8B55 E4 MOV EDX,DWORD PTR SS:[EBP-1C] ; <== uStr
0047F2E6 . E8 55400900 CALL alleycod.00513340 ; <== Muat be the SAME
0047F2EB 74 0D JE SHORT alleycod.0047F2FA ; <== if SAME go to next check

```

- Quá trình so sánh thứ hai được tiến hành với một chuỗi mặc định :

```

0047F304 . 8B90 80000000 MOV EDX,DWORD PTR DS:[EAX+80] ; <==
"77RD%6^HD75H88YR&#^88GR3G#4^*8%8N36%ERY7"
0047F30A . 8D85 5CFFFFFF LEA EAX,DWORD PTR SS:[EBP-A4]
0047F310 . 50 PUSH EAX
0047F311 . B9 CCF54700 MOV ECX,alleycod.0047F5CC ; ASCII
"ASEFTJHTIOPDEFPLFJMJKNMH"
0047F316 . 58 POP EAX
0047F317 . E8 603F0900 CALL alleycod.0051327C ; <== Concat String : cStr
0047F31C . 8B85 5CFFFFFF MOV EAX,DWORD PTR SS:[EBP-A4] ; <== cStr
0047F322 . B9 0C000000 MOV ECX,0C ; <== Get 12 charts
0047F327 . BA 01000000 MOV EDX,1 ; <== from FIRST chart
0047F32C . E8 07410900 CALL alleycod.00513438 ; <== Cutting : CutStr
0047F331 . 8B85 60FFFFFF MOV EAX,DWORD PTR SS:[EBP-A0] ; <== CutStr
0047F337 . 8D95 64FFFFFF LEA EDX,DWORD PTR SS:[EBP-9C]
0047F33D . E8 8EAD0800 CALL alleycod.0050A0D0
0047F342 . 8B95 64FFFFFF MOV EDX,DWORD PTR SS:[EBP-9C] ; <== CutStr
0047F348 . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== 12 first charts of
NewStr
0047F34B . E8 F03F0900 CALL alleycod.00513340 ; <== Must be the same

```

- Quá trình phân tích trên có vẻ rất khó hiểu, nhưng thực chất đây là một quá trình **ĐI NGUỘC**. Thông thường ta vẫn dùng cách là mã hoá chuỗi U nhập và so sánh với S. Nhưng chương trình này lại khác. Chương trình sẽ mã hoá S và so sánh với U. Như vậy ta phải đi ngược lại quá trình mã hoá S, sau đó mã hoá U theo cách thức này.

- Ở trên ta nhận thấy, chuỗi mã hoá phải gồm 24 ký tự . 12 ký tự đầu tiên là 12 ký tự mặc định "**77RD%6^HD75H**". 12 ký tự sau là do chuỗi U mã hoá . Tuy nhiên, chuỗi U ở đây có một đặc điểm . (1) Nếu (**LenU > 12**) thì ta dùng chính 12 ký tự của chuỗi U làm chuỗi mã hoá; (2) nếu (**LenU < 12**) thì sẽ thêm vào sau đó các ký tự của chuỗi mặc định "**"ASEFTJHTIOPDEFPLFJMJKNMH"**" để đủ 12 ký tự .

- Ta để ý ở đoạn CODE đầu tiên, chương trình đã tiến hành cộng dồn các giá trị của chuỗi “**GLE140463YM**”, sau đó chuyển giá trị này sang dạng chuỗi ( ta có chuỗi “**688**” ) . Kế đó là một vòng lặp, nhằm xác định chuỗi dùng để giải mã phải có chiều dài là 24 ký tự bằng cách nối các chuỗi “**688**” lại với nhau . Như vậy chuỗi dùng để giải mã sẽ là “**688688688688688688688688688**”

/\*/\*/\*/ - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : <:UI(9cKG<8KWHD2FUFFNJUW

### **III – KeyGen :**

- /Section 1/- Xác định chiều dài của chuỗi U để sử dụng cho quá trình mã hoá.
  - /Section 2/- Thêm hay bớt cho đủ 12 ký tự
  - /Section 3/- Nối vào sau chuỗi mặc định “77RD%6^HD75H”
  - /Section 4/- Tiến hành mã hoá với chuỗi mã hoá “688688688688688688688688”
  - /Section 5/- Công thức mã hoá :
 
$$reaSerial[i] = reaConCatStr[i] - (reaEncryptString[i] - 0x30) + 0xB;$$

IV – End of Tut :

- Finished = *August 11, 2004*

# Reverse Engineering Association

## SoftWare

**Homepage** : <http://www.xrilly.com>  
**Production** : xrilly software.  
**SoftWare** : Arial CD Ripper 1.3.5  
**Copyright by** : Copyright © 2004 xrilly software. All Rights Reserved.  
**Type** : Name / Email  
**Packed** : ASPack 2.12 -> Alexey Solodovnikov  
**Language** : Borland Delphi 6.0 - 7.0 ( MD5 )  
**Crack Tool** : OllyDbg 1.09d, PEiD 0.92, kWdsm 10  
**Unpack** : Manual  
**Request** : Correct Serial / KeyGen

Arial CD Ripper 1.3.5

Arial CD Ripper is a versatile and easy to use tool for CD grabbing and audio conversion. It can convert most of the popular audio formats from one to another and extract all your favourite CD tracks into MP3(MPEG Layer-3), WAV(WAVE sound files, including RAW PCM type and MS ADPCM type), OGG(OGG VORBIS), FLAC(FLAC format), APE (Monker's audio format) with ID3v1 Tag Editor supported. This program can supports conversions of (from and to) MP3, WAV, OGG, FLAC, APE. With as simple as a click, you can convert a track in a minute and the whole CD tracks within a few minutes without losing the audio quality. This program can converts each format to another between MP3, WAV, OGG, FLAC, APE. Arial CD Ripper can also works as CD Player and MP3/WAV/OGG/FLAC/APE Player.

## I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **ASPack 2.12 -> Alexey Solodovnikov** . Sau khi UnPACK kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**.

- Nhập thử User, Email và Fake Serial, ta không nhận được thông báo gì, tuy nhiên khi tìm chuỗi thông báo trong Olly ta gặp chuỗi :

00573C07 . B9 9C3C5700 MOV ECX,dump\_.00573C9C ; ASCII "Congratulations!"

00573C0C . BA B03C5700 MOV EDX,dump\_.00573CB0 ; ASCII "Register"

successfully! Thank you for your support!"

- Dò ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :

00573BCD . E8 C65EEDFF CALL dump\_.00449A98 ; <== Set BreakPoint here

## II – Cracking :

- Load và chạy chương trình, chương trình dừng lại tại điểm đặt BreakPoint . Trace tiếp ta đến :

00573BED . E8 DAEFFFFF CALL dump\_.00572BCC ; <== Trace into here

- Dùng F7 trace into ta đến quá trình mã hoá . Trước hết chương trình tạo ra chuỗi MD5 hash :

00572C04 |. E8 9707FFFF CALL dump\_.005633A0 ; <== Get MD5 hash

- Sau đó, dùng chính chuỗi này để tiến hành tạo ra chuỗi Serial thực :

00572C0F |. E8 0008FFFF CALL dump\_.00563414 ; <== Trace into here

- Dùng F7 trace into vào trong để tìm hiểu quá trình mã hoá :

0056342A |. A5 MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]; <== SectionI of MD5 hash

0056342B |. A5 MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]; <== SectionII of MD5 hash

0056342C |. A5 MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]; <== SectionIII of MD5 hash

0056342D |. A5 MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]; <== SectionIV of MD5 hash

0056342E |. 8BFA MOV EDI,EDX

00563430 |. 33C0 XOR EAX,EAX

00563432 |. 55 PUSH EBP

00563433 |. 68 AF345600 PUSH dump\_.005634AF

00563438 |. 64:FF30 PUSH DWORD PTR FS:[EAX]

0056343B |. 64:8920 MOV DWORD PTR FS:[EAX],ESP

0056343E |. 8BC7 MOV EAX,EDI

00563440 |. E8 8B19EAFF CALL dump\_.00404DD0

00563445 |. B3 10 MOV BL,10

00563447 |. 8D75 F0 LEA ESI,[LOCAL.4]

0056344A |> FF37 /PUSH DWORD PTR DS:[EDI]

0056344C |. 8D45 EC |LEA EAX,[LOCAL.5]

0056344F |. 33D2 |XOR EDX,EDX

00563451 |. 8A16 |MOV DL,BYTE PTR DS:[ESI] ; <== MD5[i]

00563453 |. C1EA 04 |SHR EDX,4 ; <== MD5[i] / 0x10

00563456 |. 83E2 0F |AND EDX,0F ; <== (MD5[i] / 0x10) and 0xF

00563459 |. 8A92 C02D5800 |MOV DL,BYTE PTR DS:[EDX+582DC0] ; <== Serial[i] = Value at (MD5[i] / 0x10) and 0xF

0056345F |. E8 4C1BEAFF |CALL dump\_.00404FB0

00563464 |. FF75 EC |PUSH [LOCAL.5]

00563467 |. 8D45 E8 |LEA EAX,[LOCAL.6]

0056346A |. 8A16 |MOV DL,BYTE PTR DS:[ESI] ; <== MD5[i]

0056346C |. 80E2 0F |AND DL,0F ; <== MD5[i] and 0xF

0056346F |. 81E2 FF000000 |AND EDX,0FF ; <== (MD5[i] and 0xF) and 0xFF

00563475 |. 8A92 C02D5800 |MOV DL,BYTE PTR DS:[EDX+582DC0] ; <== Serial[i+1] = Value at ((MD5[i] and 0xF) and 0xFF)

0056347B |. E8 301BEAFF |CALL dump\_.00404FB0

00563480 |. FF75 E8 |PUSH [LOCAL.6]

00563483 |. 8BC7 |MOV EAX,EDI

```
00563485 |. BA 03000000 |MOV EDX,3
0056348A |. E8 B91CEAFF |CALL dump_.00405148
```

```
0056348F |. 46 |INC ESI ; <== j++
00563490 |. FECB |DEC BL ; <== Len.MD5 --
00563492 |.^ 75 B6 |JNZ SHORT dump_.0056344A ; <== Loop until Len.MD5 == 0x0
- Qua cách tính trên ta dễ dàng nhận ra ngay một điều . Chương trình đã chuyển giá trị MD5 Hash thành chuỗi MD5 . Và chuỗi này chính là chuỗi Serial thực .
```

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErTeAm

Serial : 95f1e6b1b4e42701801980bf640ddd7a

### III – KeyGen :

Tạo và xuất ra chuỗi MD5 hash

### IV – End of Tut :

- Finished – 27/07/2004

# Reverse Engineering Association SoftWare

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| Homepage :     | <a href="http://www.ashampoo.com">http://www.ashampoo.com</a>                   |
| Production :   | ashampoo Technology GmbH & Co. KG                                               |
| SoftWare :     | <a href="#">Ashampoo Privacy Protector 1.04 (se)</a>                            |
| Copyright by : | Copyright © 1999 - 2004 ashampoo Technology GmbH & Co. KG. All Rights Reserved. |
| Type :         | <a href="#">Serial</a>                                                          |
| Packed :       | N / A                                                                           |
| Language :     | <a href="#">Microsoft Visual C++ 6.0</a>                                        |
| Crack Tool :   | <a href="#">OllyDbg 1.09d, PEiD 0.92, kWds 10</a>                               |
| Unpack :       | N / A                                                                           |
| Request :      | <a href="#">Correct Serial / KeyGen</a>                                         |

### [Ashampoo Privacy Protector 1.04 \(se\)](#)

This program makes powerful encryption technology accessible to everyone. Now you can protect the valuable data on your computer, burn them directly to CD and send confidential information to others without worrying about the contents being intercepted and misused. Quickly, easily and with no hassles encryption that works, without creating unnecessary extra work for you.

### I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Khi load chương trình lên để nhập Serial, ta nhận thấy BUTTON OK không được kích hoạt ( DISABLE ) . Như vậy ta hiểu ngay, chỉ có số Serial đúng mới làm nút này ENABLE . Như vậy, ta cần phải xác định được đoạn CODE kiểm tra các dữ liệu nhập vào . Và thông thường, ở các trường hợp này, nếu chọn đúng FUNCTION, chương trình sẽ tự động PAUSE khi ta nhập dữ liệu vào ô.

- Trong qua trình tìm kiếm ta thấy các dòng :

|                                                |               |
|------------------------------------------------|---------------|
| 0040590C  . 68 6CF34000 PUSH PrivProt.0040F36C | ; ASCII "SCM" |
| 00405A67  . 68 74F34000 PUSH PrivProt.0040F374 | ; ASCII "Key" |
| 00405A6C  . 68 70F34000 PUSH PrivProt.0040F370 | ; ASCII "Reg" |

- Theo như cách thức mã hoá của các chương trình khác của ASHAMPOO ta có thể dự đoán được dạng chuỗi Serial .Khi truy ngược lại function chứa các đoạn CODE này ta nhận thấy chúng nằm ở hai function khác nhau . Ở ta sẽ chọn Function chứa dòng "SCM" làm function chính .

- Dò ngược lên trên và ta đặt BP tại CODE đầu tiên của Function này :

|                                                     |                           |
|-----------------------------------------------------|---------------------------|
| 00405750 /\$ 64:A1 0000000>MOV EAX,DWORD PTR FS:[0] | ; <== Set BreakPoint here |
|-----------------------------------------------------|---------------------------|

## II – Cracking :

- Đối với chương trình dạng này, ta phải xác định Serial trước, sau đó COPY và PASTE vào các ô , ở chương trình này ta phải thực hiện ba lần mới xong. Mỗi một ô khi nhận được dữ liệu thì chương trình lập tức dừng lại tại BP. Sau khi hoàn tất quá trình nhập, chương trình dừng lại tại BP, qua quá trình kiểm tra dữ liệu đầu vào, sẽ đến đoạn kiểm tra chiều dài của các ô nhập :

|                                                     |                              |
|-----------------------------------------------------|------------------------------|
| 0040588A  > 8B4D 00 MOV ECX,DWORD PTR SS:[EBP]      | ; <== First section : fStr   |
| 0040588D  . 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8]    | ; <== Leng of first section  |
| 00405890  . 83F8 06 CMP EAX,6                       | ; <== must be 6 charts       |
| 00405893  . 0F85 A5010000 JNZ PrivProt.00405A3E     |                              |
| 00405899  . 8B5424 30 MOV EDX,DWORD PTR SS:[ESP+30] | ; <== Second section : sStr  |
| 0040589D  . 8B3A MOV EDI,DWORD PTR DS:[EDX]         | ; <== Leng of second section |
| 0040589F  . 837F F8 06 CMP DWORD PTR DS:[EDI-8],6   | ; <== must be 6 charts       |
| 004058A3  . 0F85 95010000 JNZ PrivProt.00405A3E     |                              |
| 004058A9  . 8B4424 34 MOV EAX,DWORD PTR SS:[ESP+34] |                              |
| 004058AD  . 8B10 MOV EDX,DWORD PTR DS:[EAX]         | ; <== Third section : tStr   |
| 004058AF  . 8B42 F8 MOV EAX,DWORD PTR DS:[EDX-8]    | ; <== Leng of third section  |
| 004058B2  . 83F8 04 CMP EAX,4                       | ; <== must be 4 charts       |
| 004058B5  . 0F85 83010000 JNZ PrivProt.00405A3E     |                              |

- Kế đó, đoạn thứ nhất và thứ hai của chuỗi Serial sẽ thực hiện quá trình cộng dồn giá trị các ký tự với nhau . Sau đó chuyển sang dạng chuỗi tương ứng theo định dạng “%04X” :

|                                                      |                                        |
|------------------------------------------------------|----------------------------------------|
| 004058BB  . 33F6 XOR ESI,ESI                         | ; <== CumV = 0                         |
| 004058BD  . 33C0 XOR EAX,EAX                         | ; <== i = 0                            |
| 004058BF  > 8A1C01 /MOV BL,BYTE PTR DS:[ECX+EAX]     | ; <== fStr[i]                          |
| 004058C2  . 885C24 18  MOV BYTE PTR SS:[ESP+18],BL   | ; <== fStr[i]                          |
| 004058C6  . 8B5C24 18  MOV EBX,DWORD PTR SS:[ESP+18] | ; <== fStr[i]                          |
| 004058CA  . 81E3 FF000000  AND EBX,0FF               | ; <== fStr[i] and 0xFF                 |
| 004058D0  . 03F3  ADD ESI,EBX                        | ; <== CumV = CumV + (fStr[i] and 0xFF) |
| 004058D2  . 40  INC EAX                              | ; <== i++                              |
| 004058D3  . 83F8 06  CMP EAX,6                       | ; <== while ( i < 0x6 )                |
| 004058D6  .^ 7C E7 \JL SHORT PrivProt.004058BF       | ; <== Continue Loop                    |
| 004058D8  . 33C0 XOR EAX,EAX                         | ; <== i = 0                            |
| 004058DA  > 8A0C07 /MOV CL,BYTE PTR DS:[EDI+EAX]     | ; <== sStr[i]                          |
| 004058DD  . 884C24 18  MOV BYTE PTR SS:[ESP+18],CL   | ; <== sStr[i]                          |
| 004058E1  . 8B4C24 18  MOV ECX,DWORD PTR SS:[ESP+18] | ; <== sStr[i]                          |
| 004058E5  . 81E1 FF000000  AND ECX,0FF               | ; <== sStr[i] and 0xFF                 |
| 004058EB  . 03F1  ADD ESI,ECX                        | ; <== CumV = CumV + (sStr[i] and 0xFF) |
| 004058ED  . 40  INC EAX                              | ; <== i++                              |
| 004058EE  . 83F8 06  CMP EAX,6                       | ; <== while ( i < 0x6 )                |
| 004058F1  .^ 7C E7 \JL SHORT PrivProt.004058DA       | ; <== Continue Loop                    |
| 004058F3  . 6A 10 PUSH 10                            | ; /radix = 10 (16.)                    |
| 004058F5  . 6A 00 PUSH 0                             | ;  endptr = NULL                       |

004058F7 |. 52 PUSH EDX ;|s  
 004058F8 |. 83C6 2D ADD ESI,2D ;|  
 004058FB |. FF15 44B64000 CALL DWORD PTR DS:[<&MSVCRT.strtoul>] ;\strtoul  
 - Chuỗi này được so sánh với 4 ký tự cuối cùng của chuỗi Serial :  
 00405901 |. 83C4 0C ADD ESP,0C ; <== four last charts must be  
 00405904 |. 3BC6 CMP EAX,ESI ; <== equal with CumV String  
 - Sau giao đoạn này là đến giao đoạn kiểm tra đoạn đầu tiên của chuỗi Serial với các ký tự mặc định  
 00405929 |. 8A10 MOV DL,BYTE PTR DS:[EAX] ; <== fStr[0]  
 0040592B |. 8A19 MOV BL,BYTE PTR DS:[ECX] ; <== dStr[0]  
 0040592D |. 3AD3 CMP DL,BL ; <== Must be equal  
 0040592F |. 0F85 F8000000 JNZ PrivProt.00405A2D ; <== Must be equal  
 00405935 |. 8A50 01 MOV DL,BYTE PTR DS:[EAX+1] ; <== fStr[1]  
 00405938 |. 8A59 01 MOV BL,BYTE PTR DS:[ECX+1] ; <== dStr[1]  
 0040593B |. 3AD3 CMP DL,BL ; <== Must be equal  
 0040593D |. 0F85 EA000000 JNZ PrivProt.00405A2D  
 00405943 |. 8A50 02 MOV DL,BYTE PTR DS:[EAX+2] ; <== fStr[2]  
 00405946 |. 8A59 02 MOV BL,BYTE PTR DS:[ECX+2] ; <== dStr[2]  
 00405949 |. 3AD3 CMP DL,BL ; <== Must be equal  
 0040594B |. 0F85 DC000000 JNZ PrivProt.00405A2D  
 00405951 |. 8A40 05 MOV AL,BYTE PTR DS:[EAX+5] ; <== fStr[5]  
 00405954 |. 3C 41 CMP AL,41 ; <== must from  
 00405956 |. 0F8C D1000000 JL PrivProt.00405A2D ; <== "A"  
 0040595C |. 3C 48 CMP AL,48 ; <== to  
 0040595E |. 0F8F C9000000 JG PrivProt.00405A2D ; <== "H"  
 .....  
 0040598E |. 8B6D 00 MOV EBP,DWORD PTR SS:[EBP] ; <== Serial  
 00405991 |. 8A45 03 MOV AL,BYTE PTR SS:[EBP+3] ; <== fStr[3]  
 00405994 |. 3C 30 CMP AL,30 ; <== if ( fStr[3] = 0x30 )  
 00405996 |. 75 2F JNZ SHORT PrivProt.004059C7 ; <== then  
 00405998 |. 807D 04 31 CMP BYTE PTR SS:[EBP+4],31 ; <== fStr[4] = 0x31  
 0040599C |. 75 29 JNZ SHORT PrivProt.004059C7 ; <== else  
 0040599E |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]  
 004059A2 |. C74424 24 FFF>MOV DWORD PTR SS:[ESP+24],-1  
 004059AA |. E8 513E0000 CALL <JMP.&MFC42.#800>  
 004059AF |. 5F POP EDI  
 004059B0 |. 5E POP ESI  
 004059B1 |. 5D POP EBP  
 004059B2 |. B8 08A00500 MOV EAX,5A008  
 004059B7 |. 5B POP EBX  
 004059B8 |. 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C]  
 004059BC |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX  
 004059C3 |. 83C4 18 ADD ESP,18  
 004059C6 |. C3 RETN  
 004059C7 |> 3C 35 CMP AL,35 ; <== if ( fStr[3] = 0x35 )  
 004059C9 |. 75 2F JNZ SHORT PrivProt.004059FA ; <== then  
 004059CB |. 807D 04 30 CMP BYTE PTR SS:[EBP+4],30 ; <== fStr[4] = 0x30  
 004059CF |. 75 29 JNZ SHORT PrivProt.004059FA ; <== else  
 004059D1 |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]  
 004059D5 |. C74424 24 FFF>MOV DWORD PTR SS:[ESP+24],-1  
 004059DD |. E8 1E3E0000 CALL <JMP.&MFC42.#800>  
 004059E2 |. 5F POP EDI  
 004059E3 |. 5E POP ESI

```

004059E4 |. 5D POP EBP
004059E5 |. B8 FF0A0500 MOV EAX,50AFF
004059EA |. 5B POP EBX
004059EB |. 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C]
004059EF |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX
004059F6 |. 83C4 18 ADD ESP,18
004059F9 |. C3 RETN
004059FA |> 3C 37 CMP AL,37 ; <== if (fStr[3] = 0x37)
004059FC |. 75 2F JNZ SHORT PrivProt.00405A2D ; <== then
004059FE |. 807D 04 37 CMP BYTE PTR SS:[EBP+4],37 ; <== fStr[4] = 0x37
00405A02 |. 75 29 JNZ SHORT PrivProt.00405A2D

```

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErS      Serial : SCM77H-YY7F5M-0377    or    SCM01G-JM524P-033A

### III – KeyGen :

- /Section 1/- Tạo đoạn dài tiên của chuỗi Serial theo quy tắc định trước.
- /Section 2/- Tạo đoạn thứ hai ngẫu nhiên ( 6 ký tự )
- /Section 3/- Thực hiện quá trình cộng dồn giá trị các ký tự của hai chuỗi này.
- /Section 4/- Chuyển giá trị này sang chuỗi theo định dạng “%04X” và gắn các đoạn lại với nhau .

### IV – SourceCode ( VC++ ) :

```

char reaSerial[64]="SCM";
char reaRandomString[256]="0123456789MNBVCXZASDFGHJKLMNOPUIYTRREWQ";
char reaDefaultString_01[10]="ABCDEFGH";
char reaDefaultString_02[10]="057107";
char reaCheckString[20]={0};
char reaTemp[20]={0};
int i=0;
int Value=0;
srand((unsigned)time(NULL));
reaSerial[3] = reaDefaultString_02[GetTickCount()%3];
reaSerial[4] = reaDefaultString_02[3 + GetTickCount()%3];
reaSerial[5] = reaDefaultString_01[GetTickCount()%8];
i=0; Value=0;
while (i < 0x6)
{
 Value = Value + (reaSerial[i] & 0xFF);
 i++;
}

i=0;
while (i < 6)
{
 reaTemp[i] = reaRandomString[rand() % 36];
 i++;
}

i=0;
while (i < 0x6)
{

```

```

Value = Value + (reaTemp[i] & 0xFF);
i++;
}

Value = Value + 0x2D;
wsprintf(reaCheckString,"-%04X",Value);
lstrcat(reaSerial,"-");
lstrcat(reaSerial,reaTemp);
lstrcat(reaSerial,reaCheckString);

SetDlgItemText(IDC_Serial,reaSerial);

```

**V – End of Tut :**

- Finished – **20/07/2004**

# Reverse Engineering Association

## SoftWare

|                     |   |                                                                                            |
|---------------------|---|--------------------------------------------------------------------------------------------|
| <b>Homepage</b>     | : | <b><a href="http://www.ashampoo.com">http://www.ashampoo.com</a></b>                       |
| <b>Production</b>   | : | <b>ashampoo Technology GmbH &amp; Co. KG</b>                                               |
| <b>SoftWare</b>     | : | <b>Ashampoo SeeYa! 2.2.0.5 (se)</b>                                                        |
| <b>Copyright by</b> | : | <b>Copyright © 1999 - 2004 ashampoo Technology GmbH &amp; Co. KG. All Rights Reserved.</b> |
| <b>Type</b>         | : | <b>Serial</b>                                                                              |
| <b>Packed</b>       | : | <b>N / A</b>                                                                               |
| <b>Language</b>     | : | <b>Borland Delphi 4.0 - 5.0</b>                                                            |
| <b>Crack Tool</b>   | : | <b>OllyDbg 1.09d, PEiD 0.92, kWdsm 10</b>                                                  |
| <b>Unpack</b>       | : | <b>N / A</b>                                                                               |
| <b>Request</b>      | : | <b>Correct Serial / KeyGen</b>                                                             |

### **Ashampoo SeeYa! 2.2.0.5 (se)**

It's a great ideal to send slideshows to your friends and family in a single-file, self executable format . Just image the look on their faces when they here your voice ...

**I – Information :**

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland Delphi 4.0 - 5.0**.
- Nhập thử Fake Serial, ta nhận được thông báo "**Sorry, wrong code ! Please check your input anh try again!**" Tuy nhiên ta không thể tìm được thông báo này trong Olly. Ta phải sử dụng phương pháp STACK để tìm đến đoạn Function chính .
- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :  
00446A39 |. E8 E2A7FBFF CALL [JMP.&Vcl50.@System@@LStrAddRef\\$qqry](#) ; <== Set BP here

**II – Cracking :**

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến :  
00446A8D |. E8 52ECFFFF CALL Plug\_See.004456E4 ; <== Trace Into

- Sau khi dùng F7 trace into :

0044571E |. E8 21FFFFFF CALL Plug\_See.00445644 ; <== Trace into

- Tiếp tục dùng F7 để trace into :

00445684 |. E8 67F1FFFF CALL Plug\_See.004447F0 ; <== Trace Into

- Tiếp tục dùng F7 để trace into ta đến quá trình kiểm tra chiều dài chuỗi U nhập và một số các ký tự đặc biệt

00444848 . E8 ABC9FBFF CALL [JMP.&Vcl50.@System@@LStrLen\\$qqrv](#) ; <== Len.S

0044484D . 83F8 14 CMP EAX,14 ; <== Serial must be 20 charts

00444850 . 0F85 9A020000 JNZ Plug\_See.00444AF0

00444856 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

00444859 . 8078 06 2D CMP BYTE PTR DS:[EAX+6],2D ; <== S[6] == "-"

0044485D . 0F85 8D020000 JNZ Plug\_See.00444AF0

00444863 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

00444866 . 8078 0D 2D CMP BYTE PTR DS:[EAX+D],2D ; <== S[13] == "-"

0044486A . 0F85 80020000 JNZ Plug\_See.00444AF0

- Cũng tương tự như các chương trình khác của hãng ASHAMPOO, chương trình tiến hành cắt chuỗi và nối chúng lại theo một thứ tự mặc định [5][9][10][14][15][16][19]

00444874 . B9 04000000 MOV ECX,4

00444879 . BA 01000000 MOV EDX,1

0044487E . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

00444881 . E8 B2C9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>

00444886 . 8D45 E4 LEA EAX,DWORD PTR SS:[EBP-1C]

00444889 . 50 PUSH EAX

0044488A . B9 01000000 MOV ECX,1

0044488F . BA 05000000 MOV EDX,5

00444894 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

00444897 . E8 9CC9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>

0044489C . 8D45 D8 LEA EAX,DWORD PTR SS:[EBP-28]

0044489F . 50 PUSH EAX

004448A0 . B9 02000000 MOV ECX,2

004448A5 . BA 08000000 MOV EDX,8

004448AA . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

004448AD . E8 86C9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>

004448B2 . 8D45 EC LEA EAX,DWORD PTR SS:[EBP-14]

004448B5 . 50 PUSH EAX

004448B6 . B9 01000000 MOV ECX,1

004448BB . BA 06000000 MOV EDX,6

004448C0 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

004448C3 . E8 70C9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>

004448C8 . 8D45 C4 LEA EAX,DWORD PTR SS:[EBP-3C]

004448CB . 50 PUSH EAX

004448CC . B9 02000000 MOV ECX,2

004448D1 . BA 0A000000 MOV EDX,0A

004448D6 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]

004448D9 . E8 5AC9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>

004448DE . 8B55 C4 MOV EDX,DWORD PTR SS:[EBP-3C]

004448E1 . 8D45 EC LEA EAX,DWORD PTR SS:[EBP-14]

004448E4 . E8 17C9FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>

004448E9 . 8D45 C0 LEA EAX,DWORD PTR SS:[EBP-40]

004448EC . 50 PUSH EAX

004448ED . B9 03000000 MOV ECX,3

004448F2 . BA 0F000000 MOV EDX,0F  
 004448F7 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]  
 004448FA . E8 39C9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 004448FF . 8B55 C0 MOV EDX,DWORD PTR SS:[EBP-40]  
 00444902 . 8D45 EC LEA EAX,DWORD PTR SS:[EBP-14]  
 00444905 . E8 F6C8FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 0044490A . 8D45 BC LEA EAX,DWORD PTR SS:[EBP-44]  
 0044490D . 50 PUSH EAX  
 0044490E . B9 01000000 MOV ECX,1  
 00444913 . BA 14000000 MOV EDX,14  
 00444918 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]  
 0044491B . E8 18C9FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444920 . 8B55 BC MOV EDX,DWORD PTR SS:[EBP-44]  
 00444923 . 8D45 EC LEA EAX,DWORD PTR SS:[EBP-14]  
 00444926 . E8 D5C8FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 0044492B . 8D45 E0 LEA EAX,DWORD PTR SS:[EBP-20]  
 0044492E . 50 PUSH EAX  
 0044492F . B9 02000000 MOV ECX,2  
 00444934 . BA 0C000000 MOV EDX,0C  
 00444939 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]  
 0044493C . E8 F7C8FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444941 . 8D45 B8 LEA EAX,DWORD PTR SS:[EBP-48]  
 00444944 . 50 PUSH EAX  
 00444945 . B9 02000000 MOV ECX,2  
 0044494A . BA 12000000 MOV EDX,12  
 0044494F . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]  
 00444952 . E8 E1C8FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444957 . 8B55 B8 MOV EDX,DWORD PTR SS:[EBP-48]  
 0044495A . 8D45 E0 LEA EAX,DWORD PTR SS:[EBP-20]  
 0044495D . E8 9EC8FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 - Cũng như vậy, chương trình tiến hành kiểm tra các ký tự mặc định :  
 0044499D . 8B45 B4 MOV EAX,DWORD PTR SS:[EBP-4C] ; <== "SEY"  
 004449A0 . 50 PUSH EAX  
 004449A1 . 8D55 B0 LEA EDX,DWORD PTR SS:[EBP-50]  
 004449A4 . 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C] ; <== "SEY"  
 004449A7 . E8 E4D9FBFF CALL <JMP.&Vcl50.@Sysutils@AnsiUpperCase\$qqrx>  
 004449AC . 8B55 B0 MOV EDX,DWORD PTR SS:[EBP-50] ; <== "SEY"  
 004449AF . 58 POP EAX ; <== "SEY"  
 004449B0 . E8 63C8FBFF CALL <JMP.&Vcl50.@System@@LStrCmp\$qqrv> ; <== must be the same  
 004449B5 . 0F85 35010000 JNZ Plug\_See.00444AF0  
 004449BB . 8D55 AC LEA EDX,DWORD PTR SS:[EBP-54]  
 004449BE . 8B45 E4 MOV EAX,DWORD PTR SS:[EBP-1C]  
 004449C1 . E8 CAD9FBFF CALL <JMP.&Vcl50.@Sysutils@AnsiUpperCase\$qqrx>  
 004449C6 . 8B45 AC MOV EAX,DWORD PTR SS:[EBP-54]  
 004449C9 . 50 PUSH EAX  
 004449CA . 8D55 A8 LEA EDX,DWORD PTR SS:[EBP-58]  
 004449CD . 8B45 0C MOV EAX,DWORD PTR SS:[EBP+C]  
 004449D0 . E8 BBD9FBFF CALL <JMP.&Vcl50.@Sysutils@AnsiUpperCase\$qqrx>  
 004449D5 . 8B55 A8 MOV EDX,DWORD PTR SS:[EBP-58] ; <== "B"  
 004449D8 . 58 POP EAX ; <== "B"

004449D9 . E8 3AC8FBFF CALL <JMP.&Vcl50.@System@@LStrCmp\$qqrv>; <== must be the same  
 004449DE . 0F85 0C010000 JNZ Plug\_See.00444AF0  
 - Chương trình nối các ký tự đã được cắt theo thứ tự [5][9][10][14][15][16][19][4][7][8][0][1][2][3]  
 00444A1C . B9 07000000 MOV ECX,7  
 00444A21 . BA 01000000 MOV EDX,1  
 00444A26 . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14]  
 00444A29 . E8 0AC8FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444A2E . 8D45 A4 LEA EAX,DWORD PTR SS:[EBP-5C]  
 00444A31 . 50 PUSH EAX  
 00444A32 . B9 01000000 MOV ECX,1  
 00444A37 . BA 01000000 MOV EDX,1  
 00444A3C . 8B45 E4 MOV EAX,DWORD PTR SS:[EBP-1C]  
 00444A3F . E8 F4C7FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444A44 . 8B55 A4 MOV EDX,DWORD PTR SS:[EBP-5C]  
 00444A47 . 8D45 DC LEA EAX,DWORD PTR SS:[EBP-24]  
 00444A4A . E8 B1C7FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 00444A4F . 8D45 A0 LEA EAX,DWORD PTR SS:[EBP-60]  
 00444A52 . 50 PUSH EAX  
 00444A53 . B9 02000000 MOV ECX,2  
 00444A58 . BA 01000000 MOV EDX,1  
 00444A5D . 8B45 D8 MOV EAX,DWORD PTR SS:[EBP-28]  
 00444A60 . E8 D3C7FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444A65 . 8B55 A0 MOV EDX,DWORD PTR SS:[EBP-60]  
 00444A68 . 8D45 DC LEA EAX,DWORD PTR SS:[EBP-24]  
 00444A6B . E8 90C7FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 00444A70 . 8D45 9C LEA EAX,DWORD PTR SS:[EBP-64]  
 00444A73 . 50 PUSH EAX  
 00444A74 . B9 04000000 MOV ECX,4  
 00444A79 . BA 01000000 MOV EDX,1  
 00444A7E . 8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18]  
 00444A81 . E8 B2C7FBFF CALL <JMP.&Vcl50.@System@@LStrCopy\$qqrv>  
 00444A86 . 8B55 9C MOV EDX,DWORD PTR SS:[EBP-64]  
 00444A89 . 8D45 DC LEA EAX,DWORD PTR SS:[EBP-24]  
 00444A8C . E8 6FC7FBFF CALL <JMP.&Vcl50.@System@@LStrCat\$qqrv>  
 - Quá trình mã hoá chuỗi mới này được diễn ra như sau :

00444A97 . BA 0E000000 MOV EDX,0E ; <== Number of Loop  
 00444A9C . E8 6FFCFFFF CALL Plug\_See.00444710 ; <== Trace into

===== Trace Into =====

00444758 |. BB 01000000 MOV EBX,1 ; <== j = 1  
 0044475D |. 85FF TEST EDI,EDI  
 0044475F |. 7E 62 JLE SHORT Plug\_See.004447C3  
 00444761 |. 897D F0 MOV [LOCAL.4],EDI ; <== Number of Loop : NL  
 00444764 |. BF 01000000 MOV EDI,1 ; <== i = 1  
 00444769 > 8B45 F4 /MOV EAX,[LOCAL.3] ; <== Default String : dStr  
 0044476C |. 0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== dStr[j-1]  
 00444771 |. 8BD0 |MOV EDX,EAX ; <== dStr[j-1]  
 00444773 |. C1E2 04 |SHL EDX,4 ; <== Temp = dStr[j-1] \* 0x10  
 00444776 |. 8B4D F4 |MOV ECX,[LOCAL.3] ; <== dStr  
 00444779 |. 0FB60C19 |MOVZX ECX,BYTE PTR DS:[ECX+EBX] ; <== dStr[j]  
 0044477D |. 0BD1 |OR EDX,ECX ; <== Temp = Temp or dStr[j]  
 0044477F |. 33F2 |XOR ESI,EDX ; <== Value = Value xor Temp

```

00444781 |. 8B55 F4 |MOV EDX,[LOCAL.3] ; <== dStr
00444784 |. C1E0 02 |SHL EAX,2 ; <== dStr[j-1] * 4
00444787 |. 33F0 |XOR ESI,EAX ; <== Value = Value xor (dStr[j-1] * 4)
00444789 |. 8B45 FC |MOV EAX,[LOCAL.1]
0044478C |. 8B00 |MOV EAX,DWORD PTR DS:[EAX] ; <== ConCat String : cStr
0044478E |. 0FB64438 FF |MOVZX EAX,BYTE PTR DS:[EAX+EDI-1] ; <== cStr[i]
00444793 |. 33F0 |XOR ESI,EAX ; <== Value = Value xor cStr[i]
00444795 |. 8B45 FC |MOV EAX,[LOCAL.1]
00444798 |. E8 93CAFBBFF |CALL <JMP.&Vcl50.@System@UniqueString$qqrr17System@AnsiString>
0044479D |. 8BD6 |MOV EDX,ESI ; <== Temp
0044479F |. 885438 FF |MOV BYTE PTR DS:[EAX+EDI-1],DL ; <== cStr[i] = Value
004447A3 |. 43 |INC EBX ; <== j++
004447A4 |. 8B45 F4 |MOV EAX,[LOCAL.3] ; <== dStr
004447A7 |. 803C18 00 |CMP BYTE PTR DS:[EAX+EBX],0 ; <== if end of dStr
004447AB |. 75 10 |JNZ SHORT Plug_See.004447BD ; <== then
004447AD |. 8D45 F4 |LEA EAX,[LOCAL.3]
004447B0 |. 8B55 F8 |MOV EDX,[LOCAL.2]
004447B3 |. E8 08CAFBBFF |CALL <JMP.&Vcl50.@System@LStrLASg$qqrv> ; <== |
004447B8 |. BB 01000000 |MOV EBX,1 ; <== j = 1
004447BD |> 47 |INC EDI ; <== else (i++)
004447BE |. FF4D F0 |DEC [LOCAL.4] ; <== NL --
004447C1 |.^ 75 A6 |JNZ SHORT Plug_See.00444769 ; <== Loop until NL = 0x0
===== Trace Into =====

```

- Kết thúc quá trình này ta đến quá trình mã hóa thứ hai :

```

00444AA1 . 8B4D F8 MOV ECX,DWORD PTR SS:[EBP-8] ; <== dStr
00444AA4 . BA 0E000000 MOV EDX,0E ; <== NL = 0xE
00444AA9 . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24] ; <== cStr
00444AAC . E8 7FFFxFFFF CALL Plug_See.00444630 ; <== Trace into
===== Trace Into =====

```

```

0044466E |> \8B45 F8 MOV EAX,[LOCAL.2] ; <== dStr
00444671 |. 0FB640 01 MOVZX EAX,BYTE PTR DS:[EAX+1] ; <== dStr[1]
00444675 |. C1E0 17 SHL EAX,17 ; <== Temp_01 = dStr[1] * 0x800000
00444678 |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== dStr
0044467B |. 0FB612 MOVZX EDX,BYTE PTR DS:[EDX] ; <== dStr[0]
0044467E |. C1E2 13 SHL EDX,13 ; <== Temp_02 = dStr[0] * 0x80000
00444681 |. 0BC2 OR EAX,EDX ; <== Value = Temp_01 or Temp_02
00444683 |. 8B55 F8 MOV EDX,[LOCAL.2] ; <== dStr
00444686 |. 0FB652 02 MOVZX EDX,BYTE PTR DS:[EDX+2] ; <== dStr[2]
0044468A |. C1E2 03 SHL EDX,3 ; <== Temp_01 = dStr[2] * 0x8
0044468D |. 0BC2 OR EAX,EDX ; <== Value = Temp or Temp_01
0044468F |> 8945 F4 MOV [LOCAL.3],EAX ; <== Value
00444692 |. 8D45 FC LEA EAX,[LOCAL.1]
00444695 |. E8 96CBFBFF CALL <JMP.&Vcl50.@System@UniqueString$qqrr17Syst>
0044469A |. 8BD0 MOV EDX,EAX ; <== Value
0044469C |. 8D45 F4 LEA EAX,[LOCAL.3] ; <== Value
0044469F |. 8BCB MOV ECX,EBX ; <== NL = 0xE
004446A1 |. E8 2AFFFFFF CALL Plug_See.004445D0 ; <== Trace Into
===== Trace Into =====

```

```

004445D7 |. 8B18 MOV EBX,DWORD PTR DS:[EAX] ; <== Value
004445D9 |. 81E3 FFFF0000 AND EBX,0FFFF ; <== Temp_03 = Value and 0xFFFF
004445DF |. 8B30 MOV ESI,DWORD PTR DS:[EAX] ; <== Value
===== Trace Into =====

```

```

004445E1 |. C1EE 10 SHR ESI,10 ; <== Temp_02 = Value / 0x10000
004445E4 |. 81E6 FFFF0000 AND ESI,0FFFF ; <== Temp_02 = Temp_02 and 0xFFFF
004445EA |. 85C9 TEST ECX,ECX
004445EC |. 76 33 JBE SHORT Plug_See.00444621
004445EE |. 49 DEC ECX
004445EF |. 85C9 TEST ECX,ECX
004445F1 |. 72 2E JB SHORT Plug_See.00444621
004445F3 |. 41 INC ECX
004445F4 |. 33D2 XOR EDX,EDX
004445F6 |> 8B3C24 /MOV EDI,DWORD PTR SS:[ESP] ; <== cStr
004445F9 |. 0FB63C17 |MOVZX EDI,BYTE PTR DS:[EDI+EDX] ; <== cStr[i]
004445FD |. 03DF |ADD EBX,EDI ; <== Temp_03 = Temp_03 + cStr[i]
004445FF |. 81FB F1FF0000 |CMP EBX,0FFF1 ; <== if (Temp_03 > 0xFFFF1)
00444605 |. 72 06 |JB SHORT Plug_See.0044460D ; <== then
00444607 |. 81EB F1FF0000 |SUB EBX,0FFF1 ; <== Temp_03 = Temp_03 - 0xFFFF1
0044460D |> 03F3 |ADD ESI,EBX ; <== else (Temp_02 = Temp_02 + Temp_03)
0044460F |. 81FE F1FF0000 |CMP ESI,0FFF1 ; <== if (Temp_02 > 0xFFFF1)
00444615 |. 72 06 |JB SHORT Plug_See.0044461D ; <== then
00444617 |. 81EE F1FF0000 |SUB ESI,0FFF1 ; <== Temp_02 = Temp_02 - 0xFFFF1
0044461D |> 42 |INC EDX ; <== else (i++)
0044461E |. 49 |DEC ECX ; <== NL--
0044461F |.^ 75 D5 |JNZ SHORT Plug_See.004445F6 ; <== Loop until NL = 0x0
00444621 |> C1E6 10 SHL ESI,10 ; <== Temp_02 = Temp_02 * 0x10000
00444624 |. 03DE ADD EBX,ESI ; <== Value = Temp_02 + Temp_03
00444626 |. 8918 MOV DWORD PTR DS:[EAX],EBX ; <== Value

```

===== Trace Into =====

```

004446A6 |. 8B45 F4 MOV EAX,[LOCAL.3] ; <== Value
004446A9 |. 8BD0 MOV EDX,EAX ; <== Temp_01 = Value
004446AB |. 81E2 00F80F00 AND EDX,0FF800 ; <== Temp_01 = Temp_01 and 0xFF800
004446B1 |. C1EA 0B SHR EDX,0B ; <== Temp_01 = Temp_01 / 0x800
004446B4 |. 8BC8 MOV ECX,EAX ; <== Temp_02 = Value
004446B6 |. 83E1 7F AND ECX,7F ; <== Temp_02 = Temp_02 and 0x7F
004446B9 |. C1E1 09 SHL ECX,9 ; <== Temp_02 = Temp_02 * 0x200
004446BC |. 0BD1 OR EDX,ECX ; <== Temp_01 = Temp_01 or Temp_02
004446BE |. 8BC8 MOV ECX,EAX ; <== Temp_02 = Value
004446C0 |. 81E1 80070000 AND ECX,780 ; <== Temp_02 = Temp_02 and 0x780
004446C6 |. C1E9 07 SHR ECX,7 ; <== Temp_02 = Temp_02 * 0x80
004446C9 |. 8BD8 MOV EBX,EAX ; <== Temp_03 = Value
004446CB |. 81E3 0000F0FF AND EBX,FFF00000 ; <== Temp_03 = Temp_03 and 0xFFFF00000
004446D1 |. C1EB 10 SHR EBX,10 ; <== Temp_03 = Temp_03 / 0x10000
004446D4 |. 0BCB OR ECX,EBX ; <== Temp_02 = Temp_02 or Temp_03
004446D6 |. 33D1 XOR EDX,ECX ; <== Temp_01 = Temp_01 xor Temp_02
004446D8 |. 25 FFFF0000 AND EAX,0FFFF ; <== Value = Value and 0xFFFF
004446DD |. 33D0 XOR EDX,EAX ; <== Value = Temp_01 xor Value
004446DF |. 8955 F4 MOV [LOCAL.3],EDX ; <== Value
004446E2 |. 8B5D F4 MOV EBX,[LOCAL.3] ; <== Value

```

===== Trace Into =====

- Giá trị này sau đó được chuyển thành chuỗi :

```

00444AB3 . 8BC1 MOV EAX,ECX ; <== Value
00444AB5 . 33D2 XOR EDX,EDX ; <== Value
00444AB7 . 8945 D0 MOV DWORD PTR SS:[EBP-30],EAX ; <== Value
00444ABA . 8955 D4 MOV DWORD PTR SS:[EBP-2C],EDX ; <== Value

```

```

00444ABD . 8D55 98 LEA EDX,DWORD PTR SS:[EBP-68]
00444AC0 . 8B45 E0 MOV EAX,DWORD PTR SS:[EBP-20] ; <== Value
00444AC3 . E8 E8D8FBFF CALL <JMP.&Vcl50.@Sysutils@Trim$qqrx17System@AnsiString>
- Chuỗi mới này được so sánh với các ký tự ở vị trí : [11][12][17][18]
00444AB1 . 8BC8 MOV ECX,EAX ; <== Value
00444AB3 . 8BC1 MOV EAX,ECX ; <== Value
00444AB5 . 33D2 XOR EDX,EDX
00444AB7 . 8945 D0 MOV DWORD PTR SS:[EBP-30],EAX ; <== Value
00444ABA . 8955 D4 MOV DWORD PTR SS:[EBP-2C],EDX
00444ABD . 8D55 98 LEA EDX,DWORD PTR SS:[EBP-68]
00444AC0 . 8B45 E0 MOV EAX,DWORD PTR SS:[EBP-20] ; <== Value
00444AC3 . E8 E8D8FBFF CALL <JMP.&Vcl50.@Sysutils@Trim$qqrx17System@AnsiString> ; <== vStr
00444AC8 . 8B45 98 MOV EAX,DWORD PTR SS:[EBP-68]
00444ACB . E8 B4F9FFFF CALL Plug_See.00444484 ; <== Value (HEX)
00444AD0 . 8945 C8 MOV DWORD PTR SS:[EBP-38],EAX ; <== Value (HEX)
00444AD3 . 8955 CC MOV DWORD PTR SS:[EBP-34],EDX

00444AD6 . 8B45 D0 MOV EAX,DWORD PTR SS:[EBP-30] ; <== 4 charts of Serial : cStr
00444AD9 . 8B55 D4 MOV EDX,DWORD PTR SS:[EBP-2C]
00444ADC . 3B55 CC CMP EDX,DWORD PTR SS:[EBP-34]
00444ADF . 75 0B JNZ SHORT Plug_See.00444AEC
00444AE1 . 3B45 C8 CMP EAX,DWORD PTR SS:[EBP-38] ; <== cStr must be equal vStr
00444AE4 . 75 06 JNZ SHORT Plug_See.00444AEC
- Và kiểm tra lần cuối cùng với hai ký tự ở vị trí [7][8]
004456AB |. 837D F8 4D CMP [LOCAL.2],4D ; <== [7][8] == 77
004456AF |. 75 0A JNZ SHORT Plug_See.004456BB ; <== Must be equal

```

- Như vậy các ký tự ở vị trí : [0][1][2][3][4][7][8] là các ký tự mặc định .

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : SEEYBX-77PM6A-E0EFEC or SEEYBD-77JZ00-1NCFDJ

### III – KeyGen :

N/A.

### IV – SourceCode ( VC++ ) :

```

char reaSerial[64] = "SEEYB";
char reaRandomString[256] = "0123456789MNBVCXZASDFGHJKLMNOPUIYTREWQ";
char reaDefaultString[64] = "seeY&%§3$\"D&";
char reaTempString[20] = {0};
char reaTempSerial[20] = {0};
char reaCheckString[10] = {0};
int i=0, j=0;
int Temp_01=0, Temp_02=0, Temp_03=0;
int Value=0;
 srand((unsigned)time(NULL));

 i=0;
 while (i < 7)
 {
 reaTempSerial[i] = reaRandomString[rand() % 36];

```

```

 reaTempString[i] = reaTempSerial[i];
 i++;
 }
 lstrcat(reaTempString,"B77SEYY");

 i = 0; j=0; Temp_01 = 0; Value = 0;
 while (i < 0xE)
 {
 Temp_01 = ((reaDefaultString[j] * 0x10) | reaDefaultString[j+1]) ^ (reaDefaultString[j] * 0x4);
 Value = Value ^ (Temp_01 ^ reaTempString[i]);
 reaTempString[i] = Value;
 j++;
 if (reaDefaultString[j+1] == 0x0)
 {
 j=0;
 }
 i++;
 }
 Temp_01 = ((reaDefaultString[1] * 0x800000) | (reaDefaultString[0] * 0x800000)) |
(reaDefaultString[2] * 0x8);
 Temp_02 = Temp_01 / 0x10000;
 Temp_03 = Temp_01 & 0xFFFF;

 i=0;
 while (i < 0xE)
 {
 Temp_03 = Temp_03 + (reaTempString[i] & 0xFF);
 if (Temp_03 < 0xFFFF1)
 {
 Temp_02 = Temp_02 + Temp_03;
 }
 else
 {
 Temp_03 = Temp_03 - 0xFFFF1;
 Temp_02 = Temp_02 + Temp_03;
 }
 if (Temp_02 > 0xFFFF1)

 {
 Temp_02 = Temp_02 - 0xFFFF1;
 }
 i++;
 }

 Temp_01 = (Temp_02 * 0x10000) + Temp_03;
 _asm
 {
 MOV EAX,Temp_01
 MOV EDX,EAX
 AND EDX,0x0FF800
 SHR EDX,0x0B
 MOV ECX,EAX
 AND ECX,0x7F

```

```

 SHL ECX,0x9
 OR EDX,ECX
 MOV ECX,EAX
 AND ECX,0x780
 SHR ECX,0x7
 MOV EBX,EAX
 AND EBX,0xFFFF0000
 SHR EBX,0x10
 OR ECX,EBX
 XOR EDX,ECX
 AND EAX,0xFFFF
 XOR EDX,EAX
 MOV Temp_01,EDX
 }
wsprintf(reaCheckString,"%04X",Temp_01);

reaSerial[5] = reaTempSerial[0];
reaSerial[6] = 0x2D;
lstrcat(reaSerial,"77");
reaSerial[9] = reaTempSerial[1];
reaSerial[10] = reaTempSerial[2];
reaSerial[11] = reaCheckString[0];
reaSerial[12] = reaCheckString[1];
reaSerial[13] = 0x2D;
reaSerial[14] = reaTempSerial[3];
reaSerial[15] = reaTempSerial[4];
reaSerial[16] = reaTempSerial[5];
reaSerial[17] = reaCheckString[2];
reaSerial[18] = reaCheckString[3];
reaSerial[19] = reaTempSerial[6];

SetDlgItemText(IDC_Serial,reaSerial);

```

**V – End of Tut :**

- Finished – ***19/07/2004***

# Reverse Engineering Association

## SoftWare

|                     |   |                                                                                            |
|---------------------|---|--------------------------------------------------------------------------------------------|
| <b>Homepage</b>     | : | <b><a href="http://www.ashampoo.com">http://www.ashampoo.com</a></b>                       |
| <b>Production</b>   | : | <b>ashampoo Technology GmbH &amp; Co. KG</b>                                               |
| <b>SoftWare</b>     | : | <b>Ashampoo UnInstaller Suite Plus 1.32 (se)</b>                                           |
| <b>Copyright by</b> | : | <b>Copyright © 1999 - 2004 ashampoo Technology GmbH &amp; Co. KG. All Rights Reserved.</b> |
| <b>Type</b>         | : | <b>Serial</b>                                                                              |
| <b>Packed</b>       | : | <b>N / A</b>                                                                               |
| <b>Language</b>     | : | <b>Borland Delphi 6.0 - 7.0</b>                                                            |
| <b>Crack Tool</b>   | : | <b>OllyDbg 1.09d, PEiD 0.92, kWdsdm 10</b>                                                 |
| <b>Unpack</b>       | : | <b>N / A</b>                                                                               |
| <b>Request</b>      | : | <b>Correct Serial / KeyGen</b>                                                             |

### Ashampoo UnInstaller Suite Plus 1.32 (se)

Ashampoo UnInstaller Suite works by creating two "snapshots" of your system, one before you install an application and another one directly afterwards. These snapshots record the entire status of your computer system, including all registry entries, the locations, dates, sizes and versions of all files and the contents of all configuration files. Getting this right is extremely complicated, and Ashampoo UnInstaller Suite does this difficult job very, very well.

After creating the second snapshot the program then compares it with the first and records all the differences in a special file called an "installation log file", or "log file" for short.

When you tell Ashampoo UnInstaller Suite to uninstall the application it loads this log file and uses the information it contains to restore everything on your system to the way it was before you installed the application. New files, directories and registry entries are deleted. Changed registry entries are restored to the values they had before you installed the program. If you use the File Backup feature (only available in Expert Mode) you can even restore any original files that were overwritten by the installation of the application.

#### I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**.
- Nhập thử Fake Serial, ta nhận được thông báo "**Please enter a valid trial or full version key.**" Tuy nhiên ta không thể tìm được thông báo này trong Olly. Nhưng trong quá trình tìm chuỗi ta thấy có 4 dòng CODE nằm gần nhau :

```
004674D1 |. B8 C0764600 MOV EAX,Ashampoo.004676C0 ; |ASCII "%04x"
004675E1 |. BA DC764600 MOV EDX,Ashampoo.004676DC ; |ASCII "UNI"
00467633 |. BA F4764600 MOV EDX,Ashampoo.004676F4 ; |ASCII "77"
00467651 >|BA F4764600 MOV EDX,Ashampoo.004676F4 ; |ASCII "77"
```

- Theo như cách thức mã hoá của các chương trình khác của ASHAMPOO ta có thể dự đoán được dạng chuỗi Serial . Đồng thời ta cũng chọn Function chứa 4 dòng CODE này làm Function chính .

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :

```
00467465 |. E8 4ED4F9FF CALL Ashampoo.004048B8 ; <== Set BreakPoint here
```

#### II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình sẽ kiểm tra chiều dài chuỗi Serial :

```
004674A6 |. E8 25D2F9FF CALL Ashampoo.004046D0 ; <== Get Length of Serial
004674AB |. 83F8 12 CMP EAX,12 ; <== Serial must have 18 charts
004674AE |. 0F85 B6010000 JNZ Ashampoo.0046766A
004674B4 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== Serial
004674B7 |. E8 3C020000 CALL Ashampoo.004676F8 ; <== Trace into here
```

- Dùng F7 trace into, ta đến quá trình mã hoá tạo 4 ký tự cuối của chuỗi Serial . Chuỗi tạo thành này là cộng dồn giá trị các ký tự của chuỗi S ( 14 ký tự đầu tiên ) :

```
0046771D >/8B55 FC /MOV EDX,[LOCAL.1] ; <== Serial
00467720 |. 8A5402 FF |MOV DL,BYTE PTR DS:[EDX+EAX-1] ; <== S[i-1]
00467724 |. 81E2 FF000000 |AND EDX,0FF ; <== S[i-1] and 0xFF
0046772A |. 66:03CA |ADD CX,DX ; <== CumV = CumV + S[i-1] and 0xFF
0046772D |. 40 |INC EAX ; <== i++
0046772E |. 83F8 0E |CMP EAX,0E ; <== while (i < 0xE)
00467731 |.^75 EA \JNZ SHORT Ashampoo.0046771D ; <== Continue Loop
- Giá trị này sau đó được chuyển sang chuỗi với định dạng :
004674C1 |. 50 PUSH EAX ; /Arg1 = 0012F1E0
```

```

004674C2 |. 0FB7C6 MOVZX EAX,SI ;|
004674C5 |. 8945 E4 MOV [LOCAL.7],EAX ;|
004674C8 |. C645 E8 00 MOV BYTE PTR SS:[EBP-18],0 ;|
004674CC |. 8D55 E4 LEA EDX,[LOCAL.7] ;|
004674CF |. 33C9 XOR ECX,ECX ;|
004674D1 |. B8 C0764600 MOV EAX,Ashampoo.004676C0 ;|ASCII "%04x"
004674D6 |. E8 8D20FAFF CALL Ashampoo.00409568 ;|\Ashampoo.00409568
- Chuỗi này sẽ được so sánh với 4 ký tự cuối cùng (không tham gia vào quá trình mã hoá) của chuỗi Serial :
004675B7 |> \8B45 F8 MOV EAX,[LOCAL.2] ;<== CumV[]
004675BA |. 8B55 F4 MOV EDX,[LOCAL.3] ;<== 4 last charts of Serial
004675BD |. E8 52D2F9FF CALL Ashampoo.00404814 ;<== They must be equal
- Sau đó, 3 ký tự đầu tiên của chuỗi Serial được so sánh với chuỗi mặc định :
004675CC |. B9 03000000 MOV ECX,3 ;<== Cut 3 charts
004675D1 |. BA 01000000 MOV EDX,1 ;<== from first chart
004675D6 |. 8B45 FC MOV EAX,[LOCAL.1] ;<== of Serial
004675D9 |. E8 4AD3F9FF CALL Ashampoo.00404928 ;<== Cutting : Str[]
004675DE |. 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8] ;<== Str[]
004675E1 |. BA DC764600 MOV EDX,Ashampoo.004676DC ;<== "UNI"
004675E6 |. E8 29D2F9FF CALL Ashampoo.00404814 ;<== They must be equal
- Ký tự thứ sáu của chuỗi Serial cũng được so sánh với một ký tự mặc định :
004675F5 |. B9 01000000 MOV ECX,1 ;<== Cut 1 chart
004675FA |. BA 06000000 MOV EDX,6 ;<== from 6th chart
004675FF |. 8B45 FC MOV EAX,[LOCAL.1] ;<== of Serial
00467602 |. E8 21D3F9FF CALL Ashampoo.00404928 ;<== Cutting : Str[]
00467607 |. 8B45 F8 MOV EAX,[LOCAL.2] ;<== Str[]
0046760A |. BA E8764600 MOV EDX,Ashampoo.004676E8 ;<== "C"
0046760F |. E8 00D2F9FF CALL Ashampoo.00404814 ;<== They must be equal
- Tương tự như vậy, ký tự thứ 4 và 5 cũng được so sánh với hai ký tự mặc định :
0046761E |. B9 02000000 MOV ECX,2 ;<== Cut 2 charts
00467623 |. BA 04000000 MOV EDX,4 ;<== from 4th chart
00467628 |. 8B45 FC MOV EAX,[LOCAL.1] ;<== of Serial
0046762B |. E8 F8D2F9FF CALL Ashampoo.00404928 ;<== Cutting : Str[]
00467630 |. 8B45 F8 MOV EAX,[LOCAL.2] ;<== Str[]
00467633 |. BA F4764600 MOV EDX,Ashampoo.004676F4 ;<== "77"
00467638 |. E8 D7D1F9FF CALL Ashampoo.00404814 ;<== They must be equal

```

- Như vậy ta nhận xét : 6 ký tự đầu tiên của chương trình là 6 ký tự mặc định . 4 ký tự cuối cùng là giá trị của quá trình mã hoá . còn 8 ký tự ở giữa là các ký tự ngẫu nhiên.

/\*/\*/\* - SERIAL tương ứng với USER :

User : REA-cRaCkErS      Serial : UNI77COV3B1ZY8039B    or    UNI77CA81CSYMP0383

### III – KeyGen :

/Section 1/- Tạo 8 ký tự ngẫu nhiên . Các ký tự này được gắn vào sau 6 ký tự mặc định "UNI77C"

/Section 2/- Tính cộng dồn giá trị của các ký tự này và chuyển sang dạng chuỗi : "04%X"

/Section 3/- Nối chuỗi mới này sau chuỗi ban đầu ta có chuỗi Serial thực.

### IV – SourceCode ( VC++ ) :

```

char reaSerial[64]="UNI77C";
char reaRandomString[256]="0123456789MNBVCXZASDFGHJKLMNOPIUYTREWQ";

```

```

char reaTemp[64]={0};
int i=0;
int Value=0;

 srand((unsigned)time(NULL));
 i=0;
 while (i < 8)
 {
 reaSerial[i+6] = reaRandomString[rand() % 36];
 i++;
 }

 i=0; Value=0;
 while (i < 0xD)
 {
 Value = Value + (reaSerial[i] & 0xFF);
 i++;
 }
 wsprintf(reTemp,"%04X",Value);
 lstrcat(reSerial,reTemp);
 SetDlgItemText(IDC_Serial,reaSerial);

```

**V – End of Tut :**

- Finished – ***19/07/2004***

# Reverse Engineering Association

## SoftWare

|                     |   |                                                                                        |
|---------------------|---|----------------------------------------------------------------------------------------|
| <b>Homepage</b>     | : | <b><a href="http://www.brigsoft.com/bsatomic">http://www.brigsoft.com/bsatomic</a></b> |
| <b>Production</b>   | : | <b>Brigsoft.com.</b>                                                                   |
| <b>SoftWare</b>     | : | <b>AtomicClockService 1.0</b>                                                          |
| <b>Copyright by</b> | : | <b>Copyright © 2004 Brigsoft.com. All Rights Reserved.</b>                             |
| <b>Type</b>         | : | <b>Code / Serial</b>                                                                   |
| <b>Packed</b>       | : | <b>N / A</b>                                                                           |
| <b>Language</b>     | : | <b>Microsoft Visual C++ 7.0 [Debug]</b>                                                |
| <b>Crack Tool</b>   | : | <b>OllyDbg 1.09d, PEiD 0.92, kWds 10</b>                                               |
| <b>Unpack</b>       | : | <b>N / A</b>                                                                           |
| <b>Request</b>      | : | <b>Correct Serial / KeyGen</b>                                                         |

### AtomicClockService 1.0

It is an Internet PC Clock synchronizing application with some useful options (time offset, synchronize periods, time sources choosing, etc). This application uses the list of 19 presetting synchronizing URLs all over the world. The list can be easily enhanced. The application consists of the windows service program and the console program. You can use the console for options settings, log file viewing and synchronization control. The package can be used as personal computer or office workstation synchronizing solution. The service program can synchronize PC that works under a user login with restricted rights.

## I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 7.0 [Debug]**

- Nhập thử Fake Serial, ta nhận được thông báo "*The Key is invalid.*" Tìm chuỗi thông báo này trong Olly và ta tìm được ở địa chỉ :

0040847C . 68 4C0C4600 PUSH PCAtomic.00460C4C ; ASCII "Registration Error."  
00408481 . 68 180C4600 PUSH PCAtomic.00460C18 ; ASCII "The Key is invalid."

- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :

00408416 . E8 4AEB0200 CALL PCAtomic.00436F65 ; <== Set BP here

## II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến :

00408416 . E8 4AEB0200 CALL PCAtomic.00436F65 ; <== Set BreakPoint here

0040841B . 8D6E 70 LEA EBP,DWORD PTR DS:[ESI+70]

0040841E . 8B45 00 MOV EAX,DWORD PTR SS:[EBP]

00408421 . 8378 F4 00 CMP DWORD PTR DS:[EAX-C],0

00408425 . 75 15 JNZ SHORT PCAtomic.0040843C

00408427 . 6A 10 PUSH 10

00408429 . 68 4C0C4600 PUSH PCAtomic.00460C4C ; ASCII "Registration Error."

0040842E . 68 2C0C4600 PUSH PCAtomic.00460C2C ; ASCII "The User Name must be filled."

00408433 . 8BCE MOV ECX,ESI

00408435 . E8 C3E70200 CALL PCAtomic.00436BFD

0040843A . EB 53 JMP SHORT PCAtomic.0040848F

0040843C > 53 PUSH EBX

0040843D . 57 PUSH EDI

0040843E . 8D5E 74 LEA EBX,DWORD PTR DS:[ESI+74]

00408441 . BF E8DB4700 MOV EDI,PCAtomic.0047DBE8

00408446 . 53 PUSH EBX

00408447 . 8BCF MOV ECX,EDI

00408449 . E8 1ECFFFFF CALL PCAtomic.0040536C ; <== Trace Into here

- Dùng F7 trace into ta đến quá trình mã hoá . Đầu tiên, chương trình sẽ thay thế các ký tự “-“ trong chuỗi Serial nhập bằng các ký tự “ “ ( whiteSpace ) :

00405388 |. 6A 20 PUSH 20 ; /Arg2 = 00000020

0040538A |. 33F6 XOR ESI,ESI ; |

0040538C |. 6A 2D PUSH 2D ; |Arg1 = 0000002D

0040538E |. 8D4D 08 LEA ECX,[ARG.1] ; |

00405391 |. 8975 FC MOV [LOCAL.1],ESI ; |

00405394 |. E8 05FEFFFF CALL PCAtomic.0040519E ; \PCAtomic.0040519E

- Từng phân đoạn này được xem như là các giá trị ở dạng HEX . Từng phân đoạn này được chuyển thành giá trị HEX tương ứng ( ví dụ : 1234 <=> 042Dh )

004053B8 |. 68 10034600 PUSH PCAtomic.00460310 ; ASCII "%d %d %d %d"

004053BD |. 56 PUSH ESI

004053BE |. E8 87520100 CALL PCAtomic.0041A64A

004053C3 |. 83C4 18 ADD ESP,18

004053C6 |. FF75 F0 PUSH [LOCAL.4]

004053C9 |. 8D4F 28 LEA ECX,DWORD PTR DS:[EDI+28]

004053CC |. FF75 EC PUSH [LOCAL.5]

004053CF |. FF75 E8 PUSH [LOCAL.6]

004053D2 |. FF75 E4 PUSH [LOCAL.7]

004053D5 |. E8 03C4FFFF CALL PCAtomic.004017DD ; <== Trace Into  
 - Dùng F7 trace into và trace tiếp một đoạn ta đến :  
 0040183E |. E8 ADFEFFFF CALL PCAtomic.004016F0 ; <== Trace Into  
 - Trace into ta để quá trình mã hoá như sau :  
 004016F4 |. 8B00 MOV EAX,DWORD PTR DS:[EAX] ; <== SecI  
 004016F6 |. 69C0 10270000 IMUL EAX,EAX,2710 ; <== Value = SecI \* 0x2710  
 004016FC |. 56 PUSH ESI  
 004016FD |. 8BF1 MOV ESI,ECX  
 004016FF |. 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C]  
 00401703 |. 0301 ADD EAX,DWORD PTR DS:[ECX] ; <== Value = Value + SecII  
 00401705 |. 57 PUSH EDI  
 00401706 |. 8BCE MOV ECX,ESI  
 00401708 |. 8906 MOV DWORD PTR DS:[ESI],EAX ; <== Value  
 0040170A |. E8 C1FFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 0040170F |. 8BCE MOV ECX,ESI  
 00401711 |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIII = Temp \* 5  
 00401714 |. E8 B7FFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 00401719 |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2] ; <== Temp = Temp + SecIII \* 2  
 0040171C |. 8BCE MOV ECX,ESI  
 0040171E |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIII = Temp \* 5  
 00401721 |. E8 AAFFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 00401726 |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2] ; <== Temp = Temp + SecIII \* 2  
 00401729 |. 8BCE MOV ECX,ESI  
 0040172B |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIII = Temp \* 5  
 0040172E |. E8 9DFFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 00401733 |. 8B4C24 14 MOV ECX,DWORD PTR SS:[ESP+14]  
 00401737 |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2]; <== SecIII = Temp + SecIII \* 2  
 0040173A |. 8901 MOV DWORD PTR DS:[ECX],EAX ; <== SecIII  
 0040173C |. 8BCE MOV ECX,ESI  
 0040173E |. E8 8DFFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 00401743 |. 8BCE MOV ECX,ESI  
 00401745 |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIV = Temp \* 5  
 00401748 |. E8 83FFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 0040174D |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2]; <== Temp = Temp + SecIV \* 2  
 00401750 |. 8BCE MOV ECX,ESI  
 00401752 |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIV = Temp \* 5  
 00401755 |. E8 76FFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 0040175A |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2]; <== Temp = Temp + SecIV \* 2  
 0040175D |. 8BCE MOV ECX,ESI  
 0040175F |. 8D3C80 LEA EDI,DWORD PTR DS:[EAX+EAX\*4] ; <== SecIV = Temp \* 5  
 00401762 |. E8 69FFFFFF CALL PCAtomic.004016D0 ; <== Calculation  
 00401767 |. 8B4C24 18 MOV ECX,DWORD PTR SS:[ESP+18]  
 0040176B |. 8D0478 LEA EAX,DWORD PTR DS:[EAX+EDI\*2]; <== SecIV = Temp + SecIV \* 2  
 0040176E |. 5F POP EDI  
 0040176F |. 8901 MOV DWORD PTR DS:[ECX],EAX ; <== SecIV  
 - Hai giá trị tính toán này sẽ được so sánh với hai phân đoạn thứ III và IV .

- Tuy nhiên, chương trình này còn có một chuỗi Serial mặc định :

00401853 |> \BA D2040000 MOV EDX,4D2 ; <== "1234"  
 00401858 |. 3BFA CMP EDI,EDX ; <== SecI == "1234"  
 0040185A |. 75 16 JNZ SHORT PCAtomic.00401872  
 0040185C |. B9 2E160000 MOV ECX,162E ; <== "5678"

```
00401861 |. 3BD9 CMP EBX,ECX ; <== SecII == "5678"
00401863 |. 75 0D JNZ SHORT PCAtomic.00401872
00401865 |. 3BC2 CMP EAX,EDX ; <== SecIII == "1234"
00401867 |. 75 09 JNZ SHORT PCAtomic.00401872
00401869 |. 3BF1 CMP ESI,ECX ; <== SecIV == "5678"
0040186B |. 75 05 JNZ SHORT PCAtomic.00401872
```

/\*/\*/\*/ - SERIAL tương ứng :

### **III – KeyGen :**

- /Section 1/- Tạo hai giá trị ngẫu nhiên .
  - /Section 2/- Tính toán hai giá trị còn lại thứ III và IV theo công thức được nêu ở trên .
  - /Section 3/- Xuất ra theo định dạng "%04i-%04i-%04i-%04i"

IV – End of Tut :

- Finished - **26/07/2004**

# Reverse Engineering Association

## SoftWare

**Homepage** : <http://www.e-soft.co.uk>  
**Production** : Anthemion Software Ltd.  
**SoftWare** : Audio Edit 3.5  
**Copyright by** : Copyright © 2004 E-Soft. All Rights Reserved.  
**Type** : Name / Serial  
**Packed** : N / A  
**Language** : Microsoft Visual Basic 5.0 / 6.0  
**Crack Tool** : OllyDbg 1.09d, PEiD 0.92, kWdsm 10  
**Unpack** : N / A  
**Request** : Correct Serial

Audio Edit 3.5

Audio Edit allows you to load, record and edit WAV and MP3 files. MP3 files can be saved as WAV after editing. Using the graphical interface you can select and zoom in on small areas for accurate editing, apply effects and convert to different WAV formats. Drag the mouse over the graphic to select an area to work on. Click on Zoom to show the selected area. If you don't like the result of your editing, there are three levels of undo and redo.

## I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual Basic 5.0 / 6.0**.
  - Nhập thử Name và Fake Serial, ta nhận được thông báo "**Incorrect keycode**" Ta tìm được 0052F41F C785 7CFFFFFF 7>MOV DWORD PTR SS:[EBP-84],AudioEdi.00471>; UNICODE "Incorrect keycode"

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :  
0052EB9E E8 DD4FEDFF CALL <JMP.&MSVBVM60.\_\_vbaChkstk> : <== Set BreakPoint here

## II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình này rất đơn giản . Trace xuống một chút ta đến :

```
0052EE7C 8B55 CC MOV EDX,DWORD PTR SS:[EBP-34] ;<== Fake Serial
0052EE7F 52 PUSH EDX
0052EE80 68 D8154700 PUSH AudioEdi.004715D8 ; UNICODE "Sd3R7-G76jU-3We62-vUwS3"
0052EE85 FF15 D4104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrCm>; MSVBVM60.__vbaStrCmp]
```

- Như vậy xác định ra ngay được chuỗi Serial thực .

/\*/\*/\* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : Sd3R7-G76jU-3We62-vUwS3

## III – KeyGen :

/Section 0/- N/A .

## IV – End of Tut :

- Finished – *August 10, 2004*

# Reverse Engineering Association SoftWare

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| Homepage :     | <a href="http://aremasterer.wz.cz/">http://aremasterer.wz.cz/</a> |
| Production :   | Daniel Benes                                                      |
| SoftWare :     | <b>Automatic Remasterer Disk 1.7.1.1</b>                          |
| Copyright by : | <b>Copyright © Automatic Remasterer. All Rights Reserved.</b>     |
| Type :         | <b>Name / Serial</b>                                              |
| Packed :       | N / A                                                             |
| Language :     | <b>Borland Delphi 6.0 - 7.0</b>                                   |
| Crack Tool :   | <b>OllyDbg 1.09d, PEiD 0.92, kWdsm 10</b>                         |
| Unpack :       | N / A                                                             |
| Request :      | <b>Correct Serial / KeyGen</b>                                    |

### Automatic Remasterer Disk 1.7.1.1

The main method of program isn't as expected classic equalization and doesn't change frequency characteristics of sample about some multiple (number of dB) on given band on original, but it independently on quality and frequency characteristics changes it to required state. It is not important, whether sample was few, or much distorted, result is always the same. This is proved also by fact that repeated processing doesn't lead to any other change. Program adapts sample with trimble independently on original trimble. It is very appropriate for editing sample on sound trimbles for which we've example, but we are not able to exactly revise it. Program is surely only one, which this way works with sound, since it was written mainly with my invented method nor so with other available method.

## I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**.
  - Nhập thử User và Fake Serial, ta nhận được thông báo "Changes will be applied after application next run" Như vậy ta không thể tìm chuỗi thông báo đúng hay sai trong trường hợp này.
  - Ở trong các trường hợp như vậy, thông thường dữ liệu sẽ được lưu trực tiếp vào FILE hay vào REG . Điều cần thiết là phải tìm ra được đoạn CODE gọi dữ liệu này ra để mã hoá . Và các thức thông thường nhất là tìm các chuỗi dạng "**"RegCode"**", "**"RegKey"**", "**"RegName"**", "**"Code-Registered"**", "**"Name-Registered"**"... hay đại loại những dạng như thế . Ở trong chương trình này ta tìm được các chuỗi :
- |                                                  |                      |
|--------------------------------------------------|----------------------|
| 004A0CBF . B9 60104A00 MOV ECX,ARemaste.004A1060 | ; ASCII              |
| "iPointPosition"                                 |                      |
| 004A0CC4 . BA 50104A00 MOV EDX,ARemaste.004A1050 | ; ASCII "Notepad"    |
| 004A0CD0 . 68 78104A00 PUSH ARemaste.004A1078    | ; ASCII "none"       |
| 004A0CD5 . B9 88104A00 MOV ECX,ARemaste.004A1088 | ; ASCII "Name"       |
| 004A0CDA . BA 98104A00 MOV EDX,ARemaste.004A1098 | ; ASCII "Registered" |
| 004A0CE8 . B9 AC104A00 MOV ECX,ARemaste.004A10AC | ; ASCII "Number1"    |
| 004A0CED . BA 98104A00 MOV EDX,ARemaste.004A1098 | ; ASCII "Registered" |
| 004A0CFB . B9 BC104A00 MOV ECX,ARemaste.004A10BC | ; ASCII "Number2"    |
| 004A0D00 . BA 98104A00 MOV EDX,ARemaste.004A1098 | ; ASCII "Registered" |
| 004A0D0E . B9 CC104A00 MOV ECX,ARemaste.004A10CC | ; ASCII "Number3"    |
| 004A0D13 . BA 98104A00 MOV EDX,ARemaste.004A1098 | ; ASCII "Registered" |
- Dò ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :
- ```
004A0BBC . E8 E7EDFEFF CALL ARemaste.0048F9A8
```

II – Cracking :

- Load chương trình lên, chương trình sẽ dừng lại tại BP. Trace xuống một đoạn ta đến quá trình kiểm tra DATA :

```
004A0C79 . E8 8ACEF9FF CALL ARemaste.0043DB08 ; <== We here
004A0C7E . 8983 1C030000 MOV DWORD PTR DS:[EBX+31C],EAX ; <== Input DATA or Not
004A0C84 . 83BB 1C030000>CMP DWORD PTR DS:[EBX+31C],0 ; <== If YES
004A0C8B . 0F85 8E000000 JNZ ARemaste.004A0D1F ; <== Get DATA and Check
- Sau đó là đến quá trình kiểm tra chiều dài chuỗi U nhập :
004A0DD5 . E8 6E44F6FF CALL ARemaste.00405248 ; <== Get Length of User
004A0DDA . 85C0 TEST EAX,EAX ; <== User must be input
- Quá trình mã hoá đầu tiên chỉ là một quá trình cộng dồn giá trị các ký tự của U :
```

```
004A0DDE . BA 01000000 MOV EDX,1 ; <== i = 1
004A0DE3 > 8B0D 3C9B4B00 MOV ECX,DWORD PTR DS:[4B9B3C] ; ARemaste.0052BC70
004A0DE9 . 8B09 MOV ECX,DWORD PTR DS:[ECX] ; <== User
004A0DEB . 4A DEC EDX ; <== i--
004A0DEC . 85C9 TEST ECX,ECX
004A0DEE . 74 05 JE SHORT ARemaste.004A0DF5
004A0DF0 . 3B51 FC CMP EDX,DWORD PTR DS:[ECX-4]
004A0DF3 . 72 05 JB SHORT ARemaste.004A0DFA
004A0DF5 > E8 B232F6FF CALL ARemaste.004040AC
004A0DFA > 42 INC EDX ; <== i++
004A0DFB . 0FB64C11 FF MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== U[i-1]
004A0E00 . 014D EC ADD DWORD PTR SS:[EBP-14],ECX ; <== CumV += U[i-1]
004A0E03 . 71 05 JNO SHORT ARemaste.004A0E0A
004A0E05 . E8 AA32F6FF CALL ARemaste.004040B4
004A0E0A > 42 INC EDX ; <== i++
004A0E0B . 48 DEC EAX ; <== Len.U --
```

004A0E0C .^ 75 D5 JNZ SHORT ARemaste.004A0DE3 ; <== Loop until Len.U == 0x0
 - Quá trình mã hoá thứ hai diễn ra như sau :

```

004A0E15 . E8 2E44F6FF CALL ARemaste.00405248 ; <== Get Len.U
004A0E1A . 85C0 TEST EAX,EAX
004A0E1C . 7E 3B JLE SHORT ARemaste.004A0E59
004A0E1E . BA 01000000 MOV EDX,1 ; <== i = 1
004A0E23 > 8B0D 3C9B4B00 MOV ECX,DWORD PTR DS:[4B9B3C] ; ARemaste.0052BC70
004A0E29 . 8B09 MOV ECX,DWORD PTR DS:[ECX] ; <== User
004A0E2B . 4A DEC EDX ; <== i--
004A0E2C . 85C9 TEST ECX,ECX
004A0E2E . 74 05 JE SHORT ARemaste.004A0E35
004A0E30 . 3B51 FC CMP EDX,DWORD PTR DS:[ECX-4]
004A0E33 . 72 05 JB SHORT ARemaste.004A0E3A
004A0E35 > E8 7232F6FF CALL ARemaste.004040AC
004A0E3A > 42 INC EDX ; <== i++
004A0E3B . 0FB64C11 FF MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== U[i-1]
004A0E40 . 6BF6 02 IMUL ESI,ESI,2 ; <== Value = Value * 0x2
004A0E43 . 71 05 JNO SHORT ARemaste.004A0E4A
004A0E45 . E8 6A32F6FF CALL ARemaste.004040B4
004A0E4A > 03CE ADD ECX,ESI ; <== Value = Value + U[i-1]
004A0E4C . 71 05 JNO SHORT ARemaste.004A0E53
004A0E4E . E8 6132F6FF CALL ARemaste.004040B4 ; <== Get Len.U
004A0E53 > 8BF1 MOV ESI,ECX ; <== Value
004A0E55 . 42 INC EDX ; <== i++
004A0E56 . 48 DEC EAX ; <== Len.U --
004A0E57 .^ 75 CA JNZ SHORT ARemaste.004A0E23 ; <== Loop until Len.U == 0x0
  
```

- Quá trình mã hoá thứ ba diễn ra như sau :

```

004A0E59 > 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14]
004A0E5C . 99 CDQ
004A0E5D . 52 PUSH EDX
004A0E5E . 50 PUSH EAX
004A0E5F . B8 16000000 MOV EAX,16
004A0E64 . 33D2 XOR EDX,EDX
004A0E66 . E8 954FF6FF CALL ARemaste.00405E00 ; <== CumV = CumV * 0x16
004A0E6B . 71 05 JNO SHORT ARemaste.004A0E72
004A0E6D . E8 4232F6FF CALL ARemaste.004040B4
004A0E72 > 52 PUSH EDX
004A0E73 . 50 PUSH EAX
004A0E74 . 8BC6 MOV EAX,ESI
004A0E76 . 99 CDQ
004A0E77 . 52 PUSH EDX
004A0E78 . 50 PUSH EAX
004A0E79 . B8 07000000 MOV EAX,7
004A0E7E . 33D2 XOR EDX,EDX
004A0E80 . E8 7B4FF6FF CALL ARemaste.00405E00 ; <== Value = Value * 0x7
004A0E85 . 71 05 JNO SHORT ARemaste.004A0E8C
004A0E87 . E8 2832F6FF CALL ARemaste.004040B4
004A0E8C > 290424 SUB DWORD PTR SS:[ESP],EAX ; <== Temp = Value - CumV
  
```

- Quá trình so sánh chuỗi :

```

004A0EB0 . 3B55 EC CMP EDX,DWORD PTR SS:[EBP-14] ; <== Number1 == CumV ?
004A0EB3 75 6B JNZ SHORT ARemaste.004A0F20
004A0EB5 . 8B15 789A4B00 MOV EDX,DWORD PTR DS:[4B9A78] ; ARemaste.0052BC64
  
```

```
004A0EBB . 3B32      CMP ESI,DWORD PTR DS:[EDX]          ; <== Number2 == Value ?
004A0EBD . 75 61     JNZ SHORT ARemaste.004A0F20
004A0EBF . 8B15 849E4B00 MOV EDX,DWORD PTR DS:[4B9E84]    ; ARemaste.0052BC68
004A0EC5 . 3B02      CMP EAX,DWORD PTR DS:[EDX]          ; <== Number3 == Temp?
004A0EC7 . 75 57     JNZ SHORT ARemaste.004A0F20
```

/*/*/*/ - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : 979-311027-2155651
User : VHT-cRaCkErS Serial : 1005-332019-2302023

III – KeyGen :

- | | |
|--------------|--|
| /Section 1/- | Tính giá trị cộng dồn giá trị các ký tự của chuỗi U : CumV += U[i] |
| /Section 2/- | Value = (Value * 0x2) + U[i] Với giá trị ban đầu của Value = 0x1 |
| /Section 3/- | Temp = (Value * 0x7) - (CumV * 0x16) |
| /Section 4/- | Chuỗi được xuất ra theo định dạng "%i-%i-%i" (CumV-Value-Temp) |

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[128]={0};
char reaTemp[128]={0};
int LenUser=0;
int i=0;
int CumulativeValue=0, CalculativeValue=0, TempValue = 0;
    LenUser=GetDlgItemText(IDC_Name,reaName,64);
    if (LenUser < 1)
    {
        MessageBox("===== Your name atleast 1 chart =====", "Hey !!");
        Please input your name again !! ");
    }
    else
    {
        i=0;
        while ( i < LenUser )
        {
            CumulativeValue += reaName[i];
            i++;
        }

        i=0;    CalculativeValue = 0x1;
        while ( i < LenUser )
        {
            CalculativeValue = (CalculativeValue * 0x2) + reaName[i];
            i++;
        }

        TempValue = CalculativeValue * 0x7 - CumulativeValue * 0x16;
        wsprintf(reaserial,"%i-%i-%i",CumulativeValue,CalculativeValue,TempValue);
        SetDlgItemText(IDC_Serial,reaserial);
    }
}

```

V = End of Tut :

- Finished = 05/07/2004

Reverse Engineering Association

SoftWare

Homepage	:	http://www.rimarts.co.jp/becky.htm
Production	:	RimArts, Inc.
SoftWare	:	Becky! Internet Mail 2.10.03
Copyright by	:	Copyright © 1996-2004 RimArts, Inc. All Rights Reserved.
Type	:	Name / Serial / Email
Packed	:	N / A
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

Becky! Internet Mail 2.10.03

- You can create multiple mailboxes, of course. Moreover, you can create multiple "profiles" for each mailbox. This feature is especially useful if you use laptop computer. You can switch between several different settings, like "LAN" and dialup, for the same mailbox.
- New protocols are supported -- IMAP4rev1 for e-mail and LDAP for the address book.
- Fast! You can manage thousands of e-mail at your fingertip.
- You can write HTML e-mail with Becky!. If you have Microsoft Internet Explorer ver5 or higher installed, Becky! is a complete HTML enabled e-mail client.
- Flexible template capability -- You can prepare standardized e-mail format for business and personal e-mail. You can also create HTML template, of course.etc.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử User, Email và Fake Serial, ta nhận được thông báo "Wrong Reg code". Ta không tìm thấy thông báo này trong Olly, nhưng ta tìm được đoạn CODE sau :

```

005315ED . 68 D0695B00 PUSH B2.005B69D0 ; |Arg2 = 005B69D0 ASCII "User"
005315F2 . 68 EC6A5B00 PUSH B2.005B6AEC ; |Arg1 = 005B6AEC ASCII "License"
0053160B . 68 C8695B00 PUSH B2.005B69C8 ; |Arg2 = 005B69C8 ASCII "Code"
00531610 . 68 EC6A5B00 PUSH B2.005B6AEC ; |Arg1 = 005B6AEC ASCII "License"
00531627 . 68 C0695B00 PUSH B2.005B69C0 ; |Arg2 = 005B69C0 ASCII "EMail"
0053162C . 68 EC6A5B00 PUSH B2.005B6AEC ; |Arg1 = 005B6AEC ASCII "License"

```

- Dò ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FuncTion này :

```
005314FF . E8 69640300 CALL B2.0056796D ; <== Set BreakPoint here
```

II – Cracking :

- Load và chạy chương trình, chương trình dừng lại tại điểm đặt BreakPoint . Trace tiếp ta sẽ thấy chương trình nối chuỗi Serial nhập vào sau chuỗi mặc định “RBK” tạo thành chuỗi “RBK-XXXX-YYYY-ZZZZ”. Trace tiếp và ta đến đây :

```
005315D5 . E8 A644EEFF CALL B2.00415A80 ; \B2.00415A80
```

- Dùng F7 trace into ta đến đoạn mã hoá . Đầu tiên chương trình kiểm tra chiều dài của chuỗi sau khi đã được gắn thêm chuỗi mặc định :

```
00415A8D |. 83F9 12 CMP ECX,12 ; <== New Len.S must be 18 charts
```

00415A90 |. 0F85 AC010000 JNZ B2.00415C42

- Như vậy ta tính được chiều dài của chuỗi S nhập phải là 14 ký tự . Kế đến chương trình kiểm tra các ký tự đặc biệt :

```

00415A96 |. 8A50 03    MOV DL,BYTE PTR DS:[EAX+3]      ; <== NewS[3]
00415A99 |. B1 2D      MOV CL,2D                      ; <== Default char : "-"
00415A9B |. 3AD1      CMP DL,CL                      ; <== NewS[3] == "-"
00415A9D |. 0F85 9F010000 JNZ B2.00415C42
00415AA3 |. 3848 08    CMP BYTE PTR DS:[EAX+8],CL     ; <== NewS[8] == "-"
00415AA6 |. 0F85 96010000 JNZ B2.00415C42
00415AAC |. 3848 0D    CMP BYTE PTR DS:[EAX+D],CL     ; <== NewS[13] == "-"
00415AAF |. 0F85 8D010000 JNZ B2.00415C42

```

- Từ đây ta xác định được dạng của số Serial : XXXX-YYYY-ZZZZ . Tiếp đó chương trình sẽ phân tách số Serial mới thành 4 đoạn . Trong đó đoạn đầu tiên chính là đoạn mặc định “RBK” . Từng đoạn sẽ được kiểm tra . Đoạn đầu tiên luôn đúng vì đó là đoạn mặc định .

- Kế đó chương trình chuyển hai ký tự cuối cùng của đoạn thứ hai sang giá trị HEX tương ứng (ví dụ : 13 được chuyển thành D) và tiến hành so sánh :

```

00415B76 |. 83FE 01    CMP ESI,1                    ; <== Value must be greater than 1
00415B79 |. 0F8C 96000000 JL B2.00415C15          ; <== but
00415B7F |. 83FE 0C    CMP ESI,0C                  ; <== must be less than 12
00415B82 |. 0F8F 8D000000 JG B2.00415C15

```

- Đoạn thứ hai là đoạn mặc định :

```

00415B88 |. 8B4424 14    MOV EAX,DWORD PTR SS:[ESP+14]      ; <== Second Section
00415B8C |. BE D8675B00  MOV ESI,B2.005B67D8          ; ASCII "3437"
00415B91 |> 8A10      /MOV DL,BYTE PTR DS:[EAX]
00415B93 |. 8A1E      |MOV BL,BYTE PTR DS:[ESI]
00415B95 |. 8ACA      |MOV CL,DL
00415B97 |. 3AD3      |CMP DL,BL
00415B99 |. 75 1E      |JNZ SHORT B2.00415BB9
00415B9B |. 84C9      |TEST CL,CL
00415B9D |. 74 16      |JE SHORT B2.00415BB5
00415B9F |. 8A50 01    |MOV DL,BYTE PTR DS:[EAX+1]
00415BA2 |. 8A5E 01    |MOV BL,BYTE PTR DS:[ESI+1]
00415BA5 |. 8ACA      |MOV CL,DL
00415BA7 |. 3AD3      |CMP DL,BL
00415BA9 |. 75 0E      |JNZ SHORT B2.00415BB9
00415BAB |. 83C0 02    |ADD EAX,2
00415BAE |. 83C6 02    |ADD ESI,2
00415BB1 |. 84C9      |TEST CL,CL
00415BB3 |.^ 75 DC      |JNZ SHORT B2.00415B91

```

- Quá trình so sánh đoạn thứ tư được dựa vào một bảng mặc định . Và đoạn thứ tư này được chia làm hai phần . Ba ký tự cuối được kiểm tra giống nhau, ký tự đầu tiên kiểm tra khác . Ứng với mỗi giá trị của từng ký tự sẽ có một giá trị tương ứng . Theo như bảng này thì 3 ký tự cuối phải là các ký tự trong khoảng (0-9), còn ký tự đầu tiên phải nằm trong khoảng (a-z, A-Z)

```

00415BC2 |. 8B4424 0C    MOV EAX,DWORD PTR SS:[ESP+C]      ; <== Section_III
00415BC6 |. 0FBE48 01    MOVSX ECX,BYTE PTR DS:[EAX+1]      ; <== Section_III[1]
00415BCA |. 51          PUSH ECX
00415BCB |. E8 46A41300 CALL B2.00550016                ; <== Must be (0-9)
00415BD0 |. 83C4 04    ADD ESP,4
00415BD3 |. 85C0      TEST EAX,EAX                  ; <== if correct return value = 4
00415BD5 |. 74 3E      JE SHORT B2.00415C15
00415BD7 |. 8B5424 0C    MOV EDX,DWORD PTR SS:[ESP+C]      ; <== Section_III

```

```

00415BDB |. 0FBE42 02 MOVSX EAX,BYTE PTR DS:[EDX+2] ; <== Section_III[2]
00415BDF |. 50 PUSH EAX
00415BE0 |. E8 31A41300 CALL B2.00550016 ; <== Must be (0-9)
00415BE5 |. 83C4 04 ADD ESP,4
00415BE8 |. 85C0 TEST EAX,EAX ; <== if correct return value = 4
00415BEA |. 74 29 JE SHORT B2.00415C15
00415BEC |. 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C] ; <== Section_III
00415BF0 |. 0FBE51 03 MOVSX EDX,BYTE PTR DS:[ECX+3] ; <== Section_III[3]
00415BF4 |. 52 PUSH EDX
00415BF5 |. E8 1CA41300 CALL B2.00550016 ; <== Must be (0-9)
00415BFA |. 83C4 04 ADD ESP,4
00415BFD |. 85C0 TEST EAX,EAX ; <== if correct return value = 4
00415BFF |. 74 14 JE SHORT B2.00415C15
00415C01 |. 8B4424 0C MOV EAX,DWORD PTR SS:[ESP+C] ; <== Section_III
00415C05 |. 0FBE08 MOVSX ECX,BYTE PTR DS:[EAX] ; <== Section_III[0]
00415C08 |. 51 PUSH ECX
00415C09 |. E8 B2A31300 CALL B2.0054FFC0 ; <== Must be (a-z,A-Z)
00415C0E |. 83C4 04 ADD ESP,4
00415C11 |. 85C0 TEST EAX,EAX ; <== if correct return value = 1 or 2
00415C13 75 05 JNZ SHORT B2.00415C1A
- Bảng giá trị mặc định (96 giá trị – không tính giá trị 00 – tương ứng 96 ký tự : char(0x32) - char(0x7F)
005C23D6 48 00 10 00 10 00 10 00 10 00 10 00 10 00 H. . . . .
005C23E6 10 00 10 00 10 00 10 00 10 00 10 00 10 00 . . . . .
005C23F6 84 00 84 00 84 00 84 00 84 00 84 00 84 00 „„„„„„„„„„„„
005C2406 84 00 84 00 10 00 10 00 10 00 10 00 10 00 „„„„„„„„„„„„
005C2416 10 00 81 00 81 00 81 00 81 00 81 00 81 00 01 00 .••••••••••••„
005C2426 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 . . . . .
005C2436 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 . . . . .
005C2446 01 00 01 00 01 00 10 00 10 00 10 00 10 00 10 00 . . . . .
005C2456 10 00 82 00 82 00 82 00 82 00 82 00 02 00 □„„„„„„„„„„„„
005C2466 02 00 02 00 02 00 02 00 02 00 02 00 02 00 02 00 . . . . .
005C2476 02 00 02 00 02 00 02 00 02 00 02 00 02 00 02 00 . . . . .
005C2486 02 00 02 00 02 00 10 00 10 00 10 00 10 00 20 00 . . . . .
- Ứng với mỗi ký tự trong 3 ký tự sau ta có một giá trị trong bảng này . Giá trị này sau đó thực hiện tiếp lệnh

```

0055003A |. 83E0 04 AND EAX,4

- Và theo như giá trị trong bảng, chỉ có các giá trị “84” sau khi thực hiện phép AND mới trả về giá trị là “4”, còn các giá trị khác đều trả về giá trị “0” . Tương ứng với các giá trị “84” là số (0 – 9).

- Tương tự như vậy, ký tự đầu tiên được thực hiện tiếp lệnh :

0054FFE8 |. 25 03010000 AND EAX,103

- Lập luận tương tự, chỉ có các giá trị “81”, “01”, “82”, “02” mới trả về giá trị khác “0” (“1” hoặc “2”) . Và tương ứng với các giá trị này là các ký tự (a – z, A – Z)

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : 1603-3437-L912 or 9406-3437-u419

III – KeyGen :

/Section 1/- Tạo chuỗi ngẫu nhiên gồm 4 ký tự, trong đó hai giá trị hai ký tự sau ở dạng HEX phải nằm trong khoảng $1 < \text{Value} < 12$

/Section 2/- Chuỗi mặc định : 3437

/Section 3/- Chuỗi ngẫu nhiên trong đó ký tự đầu tiên nằm trong khoảng (a – z, A – Z) ba ký tự cuối cùng nằm trong khoảng (0 – 9)

IV – SourceCode (VC++) :

```

char reaSerial[64]={0};
char reaDefaultString[256] =
"0123456789qwertyuioplkjhgfdaszxcvbnmMNBVCXZASDFGHJKLPOIUYTREWQ";
char reaThirdSection[10]={0};
int i=0;
int FirstSection=0;
srand( (unsigned)time( NULL ) );
FirstSection = (GetTickCount()%89 + 10) * 100 + (2 + GetTickCount()%10);

i=0;
while ( i < 4 )
{
    if ( i == 0 )
    {
        reaThirdSection[i] = reaDefaultString[10 + GetTickCount()%52];
        i++;
    }
    else
    {
        reaThirdSection[i] = reaDefaultString[rand() % 10];
        i++;
    }
}
wsprintf(reaSerial,"%i-3437-",FirstSection);
lstrcat(reaSerial,reaThirdSection);
SetDlgItemText(IDC_Serial,reaSerial);

```

V – End of Tut :

- Finished – ***07/07/2004***

Reverse Engineering Association

SoftWare

Homepage :	http://www.blazingtools.com
Production :	BlazingTools Software.
SoftWare :	BlazingTools Personal Antispy 1.2.0.0
Copyright by :	Copyright © 2002-2004 BlazingTools Software . All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

BlazingTools Personal Antispy 1.2.0.0

Personal Antispy can provide every computer with strong protection against most types of

unauthorized activity monitoring software, both known and unknown.

Unique intelligent algorithm of spy software detection provides unlimited privacy protection.

Personal Antispy is based on a wide knowledge of spy software behavior and made by professionals in this area.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Registration code or user name is invalid. Please check all fields and try again!**" Ta tìm được đoạn CODE này ở địa chỉ :
004041CA |. 68 00B24500 PUSH antispy.0045B200 ; |Text = "Registration code or user name is invalid. Please check all fields and try again!"
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
004040CF |. FFD7 CALL EDI ; |GetDlgItemTextA ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Đầu tiên chương trình kiểm tra xem đã nhập User chưa :

```
004040D1 |. 8A4424 6C MOV AL,BYTE PTR SS:[ESP+6C] ; <== Get Length User
004040D5 |. 84C0 TEST AL,AL ; <== User must be input
004040D7 |. 75 1E JNZ SHORT antispy.004040F7
```

- Sau đó, chương trình sẽ nối 4 đoạn Serial lại thành một chuỗi Serial (16 ký tự) :

```
004040F7 |> 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
```

```
004040FB |. 6A 0A PUSH 0A
004040FD |. 51 PUSH ECX
004040FE |. 6A 6B PUSH 6B
00404100 |. 56 PUSH ESI
00404101 |. FFD7 CALL EDI
00404103 |. 8D5424 2C LEA EDX,DWORD PTR SS:[ESP+2C]
```

```
00404107 |. 6A 0A PUSH 0A
00404109 |. 52 PUSH EDX
0040410A |. 6A 6C PUSH 6C
0040410C |. 56 PUSH ESI
0040410D |. FFD7 CALL EDI
0040410F |. 8D4424 20 LEA EAX,DWORD PTR SS:[ESP+20]
```

```
00404113 |. 6A 0A PUSH 0A
00404115 |. 50 PUSH EAX
00404116 |. 6A 6D PUSH 6D
00404118 |. 56 PUSH ESI
00404119 |. FFD7 CALL EDI
0040411B |. 8D4C24 08 LEA ECX,DWORD PTR SS:[ESP+8]
```

```
0040411F |. 6A 0A PUSH 0A
00404121 |. 51 PUSH ECX
00404122 |. 6A 6E PUSH 6E
00404124 |. 56 PUSH ESI
00404125 |. FFD7 CALL EDI
00404127 |. 8D5424 14 LEA EDX,DWORD PTR SS:[ESP+14]
```

```
0040412B |. 8D4424 38 LEA EAX,DWORD PTR SS:[ESP+38]
```

```
0040412F |. 52 PUSH EDX ; /String2
00404130 |. 50 PUSH EAX ; |String1
00404131 |. FF15 5C934400 CALL DWORD PTR DS:[<&KERNEL32.lstrcpyA>] ; \lstrcpyA
00404137 |. 8B3D 50934400 MOV EDI,DWORD PTR DS:[<&KERNEL32.lstrcat>]; kernel32.lstrcatA
```

```

0040413D |. 8D4C24 2C  LEA ECX,DWORD PTR SS:[ESP+2C]
00404141 |. 8D5424 38  LEA EDX,DWORD PTR SS:[ESP+38]
00404145 |. 51      PUSH ECX          ; /StringToAdd
00404146 |. 52      PUSH EDX          ; |ConcatString
00404147 |. FFD7    CALL EDI          ; \|strcatA
00404149 |. 8D4424 20  LEA EAX,DWORD PTR SS:[ESP+20]
0040414D |. 8D4C24 38  LEA ECX,DWORD PTR SS:[ESP+38]
00404151 |. 50      PUSH EAX          ; /StringToAdd
00404152 |. 51      PUSH ECX          ; |ConcatString
00404153 |. FFD7    CALL EDI          ; \|strcatA
00404155 |. 8D5424 08  LEA EDX,DWORD PTR SS:[ESP+8]
00404159 |. 8D4424 38  LEA EAX,DWORD PTR SS:[ESP+38]
0040415D |. 52      PUSH EDX          ; /StringToAdd
0040415E |. 50      PUSH EAX          ; |ConcatString
0040415F |. FFD7    CALL EDI          ; \|strcatA
00404161 |. 8D8C24 A00000>LEA ECX,DWORD PTR SS:[ESP+A0]
00404168 |. 8D5424 6C  LEA EDX,DWORD PTR SS:[ESP+6C]      ; <== U
0040416C |. 51      PUSH ECX
0040416D |. 68 68A04400 PUSH antispy.0044A068          ; <== Default String
00404172 |. 52      PUSH EDX
00404173 |. E8 18FFFF CALL antispy.00403F90          ; <== Trace Into
- Dùng F7 trace into lệnh CALL này ta đến quá trình mã hoá . Quá trình mã hoá này dựa trên chuỗi mặc định "_t<(6=kl/I7gIII,>" :
00403FE0 |> /8B7424 18  /MOV ESI,DWORD PTR SS:[ESP+18]      ; <== LenD
00403FE4 |> |8BC1      MOV EAX,ECX          ; <== i
00403FE6 |. |8B5C24 14  |MOV EBX,DWORD PTR SS:[ESP+14]      ; <== U
00403FEA |. |99      |CDQ
00403FEB |. |F7FE      |IDIV ESI          ; <== Char2 = i % LenD
00403FED |. |8BC1      |MOV EAX,ECX          ; <== i
00403FEE |. |8BF2      |MOV ESI,EDX          ; <== Char2
00403FF1 |. |99      |CDQ
00403FF2 |. |F7FD      |IDIV EBP          ; <== Char1 = i % LenU
00403FF4 |. |33C0      |XOR EAX,EAX
00403FF6 |. |8A041A    |MOV AL,BYTE PTR DS:[EDX+EBX]      ; <== U[Char1]
00403FF9 |. |33D2      |XOR EDX,EDX

```

===== NOTE =====

Quá trình này thực ra là dự phòng cho trường hợp chuỗi U nhập có chiều dài nhỏ hơn chiều dài của chuỗi Mặc định . Lúc đó, chuỗi U đã hết nhưng vòng lặp chưa hết . Vì vậy với phép chia này **Char1 = i % Len.U** thì khi kết thúc chuỗi U thì tự động chương trình quay ngược trở lại đầu chuỗi U.

===== NOTE =====

```

00403FFB |. 8A143E    |MOV DL,BYTE PTR DS:[ESI+EDI]      ; <== dStr[Char2]
00403FFE |. BB 190000000 |MOV EBX,19          ; <== DefaultValue : dV
00404003 |. 33C2      |XOR EAX,EDX          ; <== Value = U[Char1] xor dStr[Char2]
00404005 |. 99      |CDQ
00404006 |. F7FB      |IDIV EBX          ; <== Value = Value % dV
00404008 |. 8B4424 1C  |MOV EAX,DWORD PTR SS:[ESP+1C]      ; <== LenD
0040400C |. 80C2 41    |ADD DL,41          ; <== Value = Value + 0x41
0040400F |. 41      |INC ECX          ; <== i++
00404010 |. 3BC8      |CMP ECX,EAX          ; <== while ( i < LenD )
00404012 |. 88143E    |MOV BYTE PTR DS:[ESI+EDI],DL      ; <== S[i] = Value
00404015 |.^ 7C C9    |JL SHORT antispy.00403FE0      ; <== Continue Loop

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErTeAm

Serial : NYWR-AARP-HGRY-CDPI

III – KeyGen :

- /Section 1/- Tính *Value* = *reaDefaultString[i]* ^ *reaName[i % LenUser]*.
- /Section 2/- Tính *reaSerial[j]* = (*Value* % 0x19) + 0x41

IV – End of Tut :

- Finished – 29/07/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.crazy-soft.com
Production :	Crazy Soft Ltd.
SoftWare :	BugTracker 2.0
Copyright by :	Copyright © 2004 Crazy Soft Ltd. All Rights Reserved.
Type :	Name / Company / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Bug Tracker 2.0

Bug Tracker is an easy to use Bug Tracking System which helps you to record and track bugs and issues during developing software production. It supports Microsoft Access Database currently and SQL server, Oracle in the future version.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử User và Fake Serial, ta nhận được thông báo "**Sorry, the registration code you entered is invalid! Please check it again !**" . Tuy nhiên ta không thể tìm thấy chuỗi thông báo này trong Olly bằng các phương pháp thông thường . Sử dụng phương pháp tìm chuỗi bằng STAKCK, ta xác định được địa chỉ xuất hiện
- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :
00401158 . E8 BF250100 CALL <JMP.&MFC42.#6334>

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống một đoạn ta đến :
- | | |
|--|------------------|
| 004011D1 . E8 AAA00000 CALL Admin.0040B280 | ; <== Trace Into |
| ----- Trace Into ----- | |

0040B2C4 |. E8 CF010000 CALL Admin.0040B498 ; <== ReverseString

----- ReverseString -----

Đây là quá trình tạo chuỗi **ReverseString**. Chuỗi này có chức năng chuyển đổi chuỗi Serial nhập vào thành một chuỗi khác tương ứng (ví dụ : nếu chuỗi nhập là "V" thì ký tự được chuyển đổi sẽ là "4").

0P-1Q-2F-3R-4J-5K-6Z-7M-8N-9E-AU-BO-C1-DI-EW-F2-GG-HD-I6-J9-KB-LS-M3-N8-O0-PV-QT-RA-S5-TY-UC-V4-W7-XX-YH-ZL

----- ReverseString -----

0040B2C9 |. 8B4D F0 MOV ECX,[LOCAL.4]

0040B2CC |. E8 43040000 CALL Admin.0040B714 ; <== DefaultString

----- DefaultString -----

Quá trình này kết hợp các ký tự mặc định để tạo ra chuỗi **DefaultString**. Đây chính là chuỗi dùng để kiểm tra chuỗi Serial sau khi được chuyển đổi ở trên. Chuỗi mặc định : "**QA9WN4XF3PUB**"

----- DefaultString -----

----- Trace Into -----

- Trace tiếp ta đến :

004011E3 . 50 PUSH EAX ; /Arg2

004011E4 . 8B4D A0 MOV ECX,DWORD PTR SS:[EBP-60] ; |

004011E7 . 83C1 64 ADD ECX,64 ; |

004011EA . 51 PUSH ECX ; | Arg1 = 0012FB90

004011EB . 8D4D AC LEA ECX,DWORD PTR SS:[EBP-54] ; |

004011EE . E8 44A10000 CALL Admin.0040B337 ; \Admin.0040B337

----- Trace Into -----

0040B378 |. 50 PUSH EAX ; /Arg1 = 00327408 ASCII "XXXXXXX"

0040B379 |. 8B4D E8 MOV ECX,[LOCAL.6] ; | 0012F430, (ASCII "QA9WN4XF3PUB")

0040B37C |. E8 F8030000 CALL Admin.0040B779 ; \Admin.0040B779

----- Trace Into -----

- Trace Into tiếp, ta đến quá trình mã hoá chính. Đầu tiên, chương trình sử dụng chuỗi **ReverseString** để chuyển chuỗi Serial theo một định dạng mới :

0040B7BA |. 52 PUSH EDX ; /Arg2

0040B7BB |. 8B45 08 MOV EAX,[ARG.1] ; |

0040B7BE |. 50 PUSH EAX ; | Arg1

0040B7BF |. 8B4D A8 MOV ECX,[LOCAL.22] ; |

0040B7C2 |. E8 DFFBFFFF CALL Admin.0040B3A6 ; \Admin.0040B3A6

- Kế đến, chương trình kiểm tra chuỗi này để đảm bảo rằng chuỗi chỉ có các ký tự nằm trong khoảng (**0 - 9**) và (**A - Z**) :

0040B7E3 |> /8B55 D0 /MOV EDX,[LOCAL.12]

0040B7E6 |. |83C2 01 |ADD EDX,1

0040B7E9 |. |8955 D0 |MOV [LOCAL.12],EDX

0040B7EC |> |8D4D F0 LEA ECX,[LOCAL.4]

0040B7EF |. |E8 BC0B0000 |CALL Admin.0040C3B0 ; <== LenS

0040B7F4 |. |3945 D0 |CMP [LOCAL.12],EAX

0040B7F7 |. |7D 39 |JGE SHORT Admin.0040B832

0040B7F9 |. |8B45 D0 |MOV EAX,[LOCAL.12]

0040B7FC |. |50 |PUSH EAX ; /Arg1

0040B7FD |. |8D4D F0 |LEA ECX,[LOCAL.4] ; |

0040B800 |. |E8 CB0B0000 |CALL Admin.0040C3D0 ; \Admin.0040C3D0

0040B805 |. |50 |PUSH EAX ; /Arg1

0040B806 |. |8B4D A8 |MOV ECX,[LOCAL.22] ; |

0040B809 |. |E8 B6010000 |CALL Admin.0040B9C4 ; \Admin.0040B9C4

```

0040B80E |. |85C0      |TEST EAX,EAX
0040B810 |. |75 1E      |JNZ SHORT Admin.0040B830
0040B812 |. |C745 B8 00000>|MOV [LOCAL.18],0
0040B819 |. |C745 FC FFFFF>|MOV [LOCAL.1],-1
0040B820 |. |8D4D F0      |LEA ECX,[LOCAL.4]
0040B823 |. |E8 B27E0000  |CALL <JMP.&MFC42.#800>
0040B828 |. |8B45 B8      |MOV EAX,[LOCAL.18]
0040B82B |. |E9 84010000  |JMP Admin.0040B9B4
0040B830 |>^|EB B1      |JMP SHORT Admin.0040B7E3

```

- Kế đó, chương trình kiểm tra các ký tự ở 12 vị trí được quy định trước phải nằm trong khoảng giá trị định sẵn .

===== NOTE =====

12 vị trí được quy định :

00417B94	00 00 00 00 04 00 00 00 07 00 00 00 05 00 00 00
00417BA4	02 00 00 00 08 00 00 00 0A 00 00 00 09 00 00 00
00417BB4	01 00 00 00 03 00 00 00 0B 00 00 00 06 00 00 00

===== NOTE =====

```

0040B83B |>/8B4D D0    /MOV ECX,[LOCAL.12]
0040B83E |. |83C1 01    |ADD ECX,1
0040B841 |. |894D D0    |MOV [LOCAL.12],ECX
0040B844 |>|837D D0 0C   CMP [LOCAL.12],0C
0040B848 |. |7D 4C      |JGE SHORT Admin.0040B896
0040B84A |. |8B55 D0    |MOV EDX,[LOCAL.12]
0040B84D |. |8B0495 947B41>|MOV EAX,DWORD PTR DS:[EDX*4+417B94]
0040B854 |. |50          |PUSH EAX           ; /Arg1
0040B855 |. |8D4D F0      |LEA ECX,[LOCAL.4]     ; |
0040B858 |. |E8 730B0000  |CALL Admin.0040C3D0    ; \Admin.0040C3D0
0040B85D |. |8845 BC      |MOV BYTE PTR SS:[EBP-44],AL
0040B860 |. |6A 00          |PUSH 0             ; /Arg3 = 00000000
0040B862 |. |8A4D BC      |MOV CL,BYTE PTR SS:[EBP-44] ; |
0040B865 |. |51          |PUSH ECX          ; |Arg2
0040B866 |. |8B55 D0      |MOV EDX,[LOCAL.12]    ; |
0040B869 |. |52          |PUSH EDX          ; |Arg1
0040B86A |. |8B4D A8      |MOV ECX,[LOCAL.22]    ; |
0040B86D |. |E8 98010000  |CALL Admin.0040BA0A    ; \Admin.0040BA0A
0040B872 |. |85C0      |TEST EAX,EAX
0040B874 |. |75 1E      |JNZ SHORT Admin.0040B894
0040B876 |. |C745 B4 00000>|MOV [LOCAL.19],0
0040B87D |. |C745 FC FFFFF>|MOV [LOCAL.1],-1
0040B884 |. |8D4D F0      |LEA ECX,[LOCAL.4]
0040B887 |. |E8 4E7E0000  |CALL <JMP.&MFC42.#800>
0040B88C |. |8B45 B4      |MOV EAX,[LOCAL.19]
0040B88F |. |E9 20010000  |JMP Admin.0040B9B4
0040B894 |>^|EB A5      |JMP SHORT Admin.0040B83B

```

===== NOTE =====

- Các quy định này ta tìm thấy trong lệnh :

0040B86D |. E8 98010000 |CALL Admin.0040BA0A ; \Admin.0040BA0A

- Quy định này được viết lại như sau (các ký tự thứ [x] phải nằm trong khoảng **YYY <=> [x]** : **YYY**):

[0] : PQ	[4] : AB	[7] : 89	[5] : VWX	[2] : MNO	[8] : 345
[10] : WXY	[9] : EFG	[1] : 234	[3] : OPQ	[11] : TUV	[6] : ABC

----- NOTE -----

- Quá trình kế tiếp là quá trình kết hợp hai ký tự ở hai vị trí được xác định trước thành một ký tự . Như vậy ta sẽ có được một chuỗi gồm 12 ký tự . Chuỗi này được đem so sánh với **DefaultString** :

```

0040B896 |> 0FBE45 D4    MOVSX EAX,BYTE PTR SS:[EBP-2C]      ; <== rS[0]
0040B89A |. 0FBE4D E0    MOVSX ECX,BYTE PTR SS:[EBP-20]      ; <== rS[12]
0040B89E |. 03C1        ADD EAX,ECX
0040B8A0 |. 99          CDQ
0040B8A1 |. 2BC2        SUB EAX,EDX
0040B8A3 |. D1F8        SAR EAX,1
0040B8A5 |. 8845 C0    MOV BYTE PTR SS:[EBP-40],AL
0040B8A8 |. 0FBE45 D8    MOVSX EAX,BYTE PTR SS:[EBP-28]
0040B8AC |. 0FBE55 E7    MOVSX EDX,BYTE PTR SS:[EBP-19]
0040B8B0 |. 03C2        ADD EAX,EDX
0040B8B2 |. 99          CDQ
0040B8B3 |. 2BC2        SUB EAX,EDX
0040B8B5 |. D1F8        SAR EAX,1
0040B8B7 |. 8845 C1    MOV BYTE PTR SS:[EBP-3F],AL
0040B8BA |. 0FBE45 DB    MOVSX EAX,BYTE PTR SS:[EBP-25]
0040B8BE |. 0FBE4D E2    MOVSX ECX,BYTE PTR SS:[EBP-1E]
0040B8C2 |. 03C1        ADD EAX,ECX
0040B8C4 |. 99          CDQ
0040B8C5 |. 2BC2        SUB EAX,EDX
0040B8C7 |. D1F8        SAR EAX,1
0040B8C9 |. 8845 C2    MOV BYTE PTR SS:[EBP-3E],AL
0040B8CC |. 0FBE45 D9    MOVSX EAX,BYTE PTR SS:[EBP-27]
0040B8D0 |. 0FBE55 E6    MOVSX EDX,BYTE PTR SS:[EBP-1A]
0040B8D4 |. 03C2        ADD EAX,EDX
0040B8D6 |. 99          CDQ
0040B8D7 |. 2BC2        SUB EAX,EDX
0040B8D9 |. D1F8        SAR EAX,1
0040B8DB |. 8845 C3    MOV BYTE PTR SS:[EBP-3D],AL
0040B8DE |. 0FBE45 D6    MOVSX EAX,BYTE PTR SS:[EBP-2A]
0040B8E2 |. 0FBE4D EA    MOVSX ECX,BYTE PTR SS:[EBP-16]
0040B8E6 |. 03C1        ADD EAX,ECX
0040B8E8 |. 99          CDQ
0040B8E9 |. 2BC2        SUB EAX,EDX
0040B8EB |. D1F8        SAR EAX,1
0040B8ED |. 8845 C4    MOV BYTE PTR SS:[EBP-3C],AL
0040B8F0 |. 0FBE45 DC    MOVSX EAX,BYTE PTR SS:[EBP-24]
0040B8F4 |. 0FBE55 E9    MOVSX EDX,BYTE PTR SS:[EBP-17]
0040B8F8 |. 03C2        ADD EAX,EDX
0040B8FA |. 99          CDQ
0040B8FB |. 2BC2        SUB EAX,EDX
0040B8FD |. D1F8        SAR EAX,1
0040B8FF |. 8845 C5    MOV BYTE PTR SS:[EBP-3B],AL
0040B902 |. 0FBE45 DE    MOVSX EAX,BYTE PTR SS:[EBP-22]
0040B906 |. 0FBE4D E3    MOVSX ECX,BYTE PTR SS:[EBP-1D]
0040B90A |. 03C1        ADD EAX,ECX
0040B90C |. 99          CDQ
0040B90D |. 2BC2        SUB EAX,EDX
0040B90F |. D1F8        SAR EAX,1
0040B911 |. 8845 C6    MOV BYTE PTR SS:[EBP-3A],AL
;
```

; <== Value = Value / 2
; <== Temp[0] = Value
; <== [4]
; <== [19]
; <== Value = rS[0] + rS[12]

; <== Value = Value / 2
; <== Temp[1] = Value
; <== [7]
; <== [14]
; <== Value = rS[4] + rS[19]

; <== Value = Value / 2
; <== Temp[2] = Value
; <== [5]
; <== [18]
; <== Value = rS[7] + rS[14]

; <== Value = Value / 2
; <== Temp[3] = Value
; <== [2]
; <== [22]
; <== Value = rS[5] + rS[18]

; <== Value = Value / 2
; <== Temp[4] = Value
; <== [8]
; <== [21]
; <== Value = rS[2] + rS[22]

; <== Value = Value / 2
; <== Temp[5] = Value
; <== [10]
; <== [15]
; <== Value = rS[8] + rS[21]

; <== Value = Value / 2
; <== Temp[6] = Value

```

0040B914 |. 0FBE45 DD    MOVSX EAX,BYTE PTR SS:[EBP-23] ; <== [9]
0040B918 |. 0FBE55 E8    MOVSX EDX,BYTE PTR SS:[EBP-18] ; <== [20]
0040B91C |. 03C2        ADD EAX,EDX ; <== Value = rS[9] + rS[20]
0040B91E |. 99          CDQ
0040B91F |. 2BC2        SUB EAX,EDX
0040B921 |. D1F8        SAR EAX,1 ; <== Value = Value / 2
0040B923 |. 8845 C7    MOV BYTE PTR SS:[EBP-39],AL ; <== Temp[7] = Value
0040B926 |. 0FBE45 D5    MOVSX EAX,BYTE PTR SS:[EBP-2B] ; <== [1]
0040B92A |. 0FBE4D E1    MOVSX ECX,BYTE PTR SS:[EBP-1F] ; <== [13]
0040B92E |. 03C1        ADD EAX,ECX ; <== Value = rS[1] + rS[13]
0040B930 |. 99          CDQ
0040B931 |. 2BC2        SUB EAX,EDX
0040B933 |. D1F8        SAR EAX,1 ; <== Value = Value / 2
0040B935 |. 8845 C8    MOV BYTE PTR SS:[EBP-38],AL ; <== Temp[8] = Value
0040B938 |. 0FBE45 D7    MOVSX EAX,BYTE PTR SS:[EBP-29] ; <== [3]
0040B93C |. 0FBE55 E4    MOVSX EDX,BYTE PTR SS:[EBP-1C] ; <== [16]
0040B940 |. 03C2        ADD EAX,EDX ; <== Value = rS[3] + rS[16]
0040B942 |. 99          CDQ
0040B943 |. 2BC2        SUB EAX,EDX
0040B945 |. D1F8        SAR EAX,1 ; <== Value = Value / 2
0040B947 |. 8845 C9    MOV BYTE PTR SS:[EBP-37],AL ; <== Temp[9] = Value
0040B94A |. 0FBE45 DF    MOVSX EAX,BYTE PTR SS:[EBP-21] ; <== [11]
0040B94E |. 0FBE4D EB    MOVSX ECX,BYTE PTR SS:[EBP-15] ; <== [23]
0040B952 |. 03C1        ADD EAX,ECX ; <== Value = rS[11] + rS[23]
0040B954 |. 99          CDQ
0040B955 |. 2BC2        SUB EAX,EDX
0040B957 |. D1F8        SAR EAX,1 ; <== Value = Value / 2
0040B959 |. 8845 CA    MOV BYTE PTR SS:[EBP-36],AL ; <== Temp[10] = Value
0040B95C |. 0FBE45 DA    MOVSX EAX,BYTE PTR SS:[EBP-26] ; <== [6]
0040B960 |. 0FBE55 E5    MOVSX EDX,BYTE PTR SS:[EBP-1B] ; <== [17]
0040B964 |. 03C2        ADD EAX,EDX ; <== Value = rS[6] + rS[17]
0040B966 |. 99          CDQ
0040B967 |. 2BC2        SUB EAX,EDX
0040B969 |. D1F8        SAR EAX,1 ; <== Value = Value / 2
0040B96B |. 8845 CB    MOV BYTE PTR SS:[EBP-35],AL ; <== Temp[11] = Value
0040B96E |. 8D45 C0    LEA EAX,[LOCAL.16] ; <== Temp
0040B971 |. 50          PUSH EAX ; /String2 = Temp
0040B972 |. 8B4D A8    MOV ECX,[LOCAL.22] ; |
0040B975 |. 51          PUSH ECX ; |String1 =
DefaultString
0040B976 |. FF15 18604100 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] : \lstrcmpA

```

/*/*/*/ - SERIAL tung ứng.

User : REA-cRaCkErTeAm Serial : 1F71RERJS9TQ1VJEBUERGMBP or
0V7BRXRJVGEP3EJT1UPR9VBO

III – KeyGen :

/Section 1/- Tính giá trị các ký tự ở các vị trí theo công thức :

while (Value !=0xA2)

f

```
reqTempSerial[0] = reqFirstSection[rand()%2];
```

```

        reaTemp[0] = reaRandomString[rand()%36];
        Value = reaTemp[0] + reaTempSerial[0];
    }
    reaTempSerial[12] = reaTemp[0];
}

```

/Section 3/- Tạo ra chuỗi Serial thực dựa trên **REVERSE** quy tắc tạo chuỗi .

IV – End of Tut :

- Finished – **September 8, 2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.salfeld.com
Production :	Salfeld Computer
SoftWare :	Child Control 2004 6.889.0
Copyright by :	Copyright © 2002-2004 Salfeld Computer GmbH. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Child Control 2004 6.889.0

Child Control is an important program that enables your children responsible handling of your home PC. Please read on the following pages how Child Control enables you to protect your PC against changes to important settings, to protocol how often "the machine" runs per day and/or to define exactly how long your junior is allowed to spend on it every day. Use this tool to restrict reliably PC uptimes, internet utilization times, protect system settings and hide your personal folders you don't want to show to your kids.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**
- Nhập thử Fake Serial, ta nhận được thông báo "**Sorry, wrong serial number.**" Ta tìm được chuỗi này tại địa chỉ :

005F6532 . B8 8C685F00 MOV EAX,Kisiset.005F688C ; |ASCII "Sorry, wrong serial number."

- Dò ngược lên trên và đặt BreakPoint tại :

005F646D . E8 72E1E0FF CALL Kisiset.004045E4 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP :
- | | |
|--|---------------------------|
| 005F646D . E8 72E1E0FF CALL Kisiset.004045E4 | ; <== Set BreakPoint here |
| 005F6472 . 83F8 06 CMP EAX,6 | ; <== U atleast 6 charts |
| 005F6475 . 7F 54 JG SHORT Kisiset.005F64CB | |

- Trace tiếp ta đến :

005F64F5 . E8 F2A7FAFF CALL Kisiset.005A0CEC ; <== Encrypt & Compare

- Dùng F7 trace Into ta đến quá trình mã hoá . Đầu tiên chương trình sẽ so sánh ký tự ở vị trí [3] với ký tự mặc định :

005A0D6E . 8B45 EC MOV EAX,[LOCAL.5]	; <== S[3]
005A0D71 . BA 0C115A00 MOV EDX,Kisiset.005A110C	; <== "5"
005A0D76 . E8 B539E6FF CALL Kisiset.00404730	; <== Must be same

- Sau đó, chương trình sẽ lấy hai ký tự ở vị trí [1][2] của chuỗi Serial ghép với hai ký tự mặc định tạo thành đoạn đầu tiên của chuỗi Serial thực . Hai ký tự đó là “**0xx5**” . Chuỗi tạo thành có dạng “**0[I]/25**”

- Kế đó, chương trình lấy giá trị của các ký tự ở các vị trí **[5]/[6]/[7]/[8]** :

005A0D9F . B9 01000000 MOV ECX,1	; <== get ONE chart
005A0DA4 . BA 06000000 MOV EDX,6	; <== at 6th (S[5])
005A0DA9 . 8B45 F8 MOV EAX,[LOCAL.2]	; <== of Fake Serial
005A0DAC . E8 933AE6FF CALL Kisiset.00404844	; <== Ripping
005A0DB1 . 8B45 D0 MOV EAX,[LOCAL.12]	; <== S[5]
005A0DB4 . 33DB XOR EBX,EBX	
005A0DB6 . 8A18 MOV BL,BYTE PTR DS:[EAX]	; <== S[5]
005A0DB8 . 8D45 CC LEA EAX,[LOCAL.13]	
005A0DBB . 50 PUSH EAX	
005A0DBC . B9 01000000 MOV ECX,1	; <== get ONE chart
005A0DC1 . BA 07000000 MOV EDX,7	; <== at 7th (S[6])
005A0DC6 . 8B45 F8 MOV EAX,[LOCAL.2]	; <== of Fake Serial
005A0DC9 . E8 763AE6FF CALL Kisiset.00404844	; <== Ripping
005A0DCE . 8B45 CC MOV EAX,[LOCAL.13]	; <== S[6]
005A0DD1 . 0FB630 MOVZX ESI,BYTE PTR DS:[EAX]	; <== S[6]
005A0DD4 . 8D45 C8 LEA EAX,[LOCAL.14]	
005A0DD7 . 50 PUSH EAX	
005A0DD8 . B9 01000000 MOV ECX,1	; <== get ONE chart
005A0DDD . BA 08000000 MOV EDX,8	; <== at 8th (S[7])
005A0DE2 . 8B45 F8 MOV EAX,[LOCAL.2]	; <== of Fake Serial
005A0DE5 . E8 5A3AE6FF CALL Kisiset.00404844	; <== Ripping
005A0DEA . 8B45 C8 MOV EAX,[LOCAL.14]	; <== S[7]
005A0DED . 0FB638 MOVZX EDI,BYTE PTR DS:[EAX]	; <== S[7]
005A0DF0 . 8D45 C4 LEA EAX,[LOCAL.15]	
005A0DF3 . 50 PUSH EAX	
005A0DF4 . B9 01000000 MOV ECX,1	; <== get ONE chart
005A0DF9 . BA 09000000 MOV EDX,9	; <== at 9th (S[8])
005A0DFE . 8B45 F8 MOV EAX,[LOCAL.2]	; <== of Fake Serial
005A0E01 . E8 3E3AE6FF CALL Kisiset.00404844	; <== Ripping
005A0E06 . 8B45 C4 MOV EAX,[LOCAL.15]	; <== S[8]
005A0E09 . 0FB600 MOVZX EAX,BYTE PTR DS:[EAX]	; <== S[8]
005A0E0C . 8945 D4 MOV [LOCAL.11],EAX	

- Sau đó, hai ký tự ở vị trí thứ [1][2] của chuỗi Serial nhập sẽ được chuyển sang giá trị HEX tương ứng (**[1]/[2]** : “**23**” sẽ được chuyển thành giá trị **hValue = 0x17**) . Giá trị này sẽ cộng với giá trị của các ký tự vừa được lấy ra ở trên :

005A0E88 . 8B45 F0 MOV EAX,[LOCAL.4]	; <== S[1][2]
005A0E8B . E8 C085E6FF CALL Kisiset.00409450	; <== Convert to HEX Value : hValue

```

005A0E90 |. 03F3      ADD ESI,EBX          ; <== Value = S[5] + S[6]
005A0E92 |. 03FE      ADD EDI,ESI          ; <== Value = Value + S[7]
005A0E94 |. 037D D4    ADD EDI,[LOCAL.11]   ; <== Value = Value + S[8]
005A0E97 |. 03C7      ADD EAX,EDI          ; <== Value = Value + hValue

- Cuối cùng, chương trình sẽ dùng ngay ký tự cuối cùng của chuỗi Serial nhập để tính toán . Giá trị cuối cùng sẽ được chuyển đổi sang dạng chuỗi DEC tương ứng theo định dạng “%d”

005A0E9A |. 8B45 E8    MOV EAX,[LOCAL.6]       ; <== S[13]
005A0E9D |. E8 AE85E6FF CALL Kisiset.00409450 ; <== Temp = S[13] - 0x30
005A0EA2 |. 8D0480    LEA EAX,DWORD PTR DS:[EAX+EAX*4] ; <== Temp = Temp * 5
005A0EA5 |. 5A        POP EDX            ; <== Value
005A0EA6 |. 03D0      ADD EDX,EAX          ; <== Value = Value + Temp
005A0EA8 |. 8955 D8    MOV [LOCAL.10],EDX        ; <== Value
005A0EAB |. FF75 E0    PUSH [LOCAL.8]
005A0EAE |. 68 24115A00 PUSH Kisiset.005A1124
005A0EB3 |. 8D55 B0    LEA EDX,[LOCAL.20]
005A0EB6 |. 8B45 D8    MOV EAX,[LOCAL.10]       ; <== Value
005A0EB9 |. E8 5684E6FF CALL Kisiset.00409314 ; <== Convert VALUE to DEC String : dStr

```

- Chuỗi mới được tạo thành này cũng được gắn luôn vào sau chuỗi ở trên . Và ký tự *S[13]* của chuỗi Fake Serial được gắn vào sau cùng .

- Qua quá trình này ta nhận xét dạng chuỗi Serial như sau : *0xx5-yyyy-AAAz* . Trong đó, *xyz* là các ký tự ngẫu nhiên trong khoảng (*0-9*) ; *AAA* là chuỗi được tính ra .

- Chuỗi U không tham gia vào quá trình mã hoá và có vô số chuỗi Serial hợp lệ .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErStEaM Serial : 0275-D3DC-2861 or 0905-942C-3619

III – End of Tut :

- Finished – *September 8, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.8848soft.com
Production :	8848Soft, inc.
SoftWare :	Clever Boxman V2.5
Copyright by :	Copyright © 2001 - 8848Soft, inc. All Rights Reserved.
Type :	Name / Serial
Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Clever Boxman V2.5

Clever Box Man V2.50 is a fun logic and strategy puzzle board game, it's a brain teaser featuring the clever porter Box Man, it has beautiful sceneries, orphean background music and sound, more and more difficult levels with no solutions seemingly, but yes actually there are. This game is very easy to operate but maybe very difficult to solve, it's really a great challenge to your brain! Fun for the whole family, encourages logical and procedural thinking, enhances problem solving skills. Treasure forever, delete never! WARNING: VERY ADDICTIVE!.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK **UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo**. Sau khi unPACK và kiểm tra lại biết chương trình được viết bằng **Microsoft Visual C++ 6.0**.

- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Registration Code Error!**" Ta chỉ có thể tìm được chuỗi này bằng cách dùng kỹ thuật STACK :

0040DFDB 68 741D4400 PUSH BoxMan_e.00441D74 ; ASCII "Registration Code Error!"

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :

0040DE39 E8 74690100 CALL BoxMan_e.004247B2 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến :

0040DEE7 E8 74080000 CALL BoxMan_e.0040E760 ; <== Trace Into

----- Create NewString -----

Quá trình này được chia làm hai dạng (1) Nếu chiều dài U nhập lớn hơn 2 thì sẽ cắt hai ký tự đầu tiên của U nhập và kết hợp ngay với chuỗi mặc định; (2) Nếu chiều dài U nhỏ hơn 2 (> 1) thì U sẽ được kết hợp với ký tự “A” trước khi kết hợp với chuỗi mặc định .

0040E7AF E8 C9390100 CALL BoxMan_e.0042217D	; <== CharUpper{U}
0040E7B4 6A 42 PUSH 42	
0040E7B6 6A 2E PUSH 2E	
0040E7B8 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]	
0040E7BC E8 D1F80000 CALL BoxMan_e.0041E092	
0040E7C1 6A 42 PUSH 42	
0040E7C3 6A 20 PUSH 20	
0040E7C5 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]	
0040E7C9 E8 C4F80000 CALL BoxMan_e.0041E092	
0040E7CE 8B4C24 28 MOV ECX,DWORD PTR SS:[ESP+28]	
0040E7D2 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8]	
0040E7D5 83F8 02 CMP EAX,2	; <== If (LenU > 2)
0040E7D8 7E 4C JLE SHORT BoxMan_e.0040E826	; <== then
0040E7DA 8D5424 0C LEA EDX,DWORD PTR SS:[ESP+C]	
0040E7DE 6A 02 PUSH 2	; <== Get Two chart
0040E7E0 52 PUSH EDX	; <== from FIRST chart
0040E7E1 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]	; <== from U
0040E7E5 E8 61F90000 CALL BoxMan_e.0041E14B	; <== Cutting
0040E7EA 68 D41E4400 PUSH BoxMan_e.00441ED4	; ASCII "fasfwefawfafafae"
0040E7EF 50 PUSH EAX	
0040E7F0 8D4424 10 LEA EAX,DWORD PTR SS:[ESP+10]	
0040E7F4 C64424 24 02 MOV BYTE PTR SS:[ESP+24],2	

```

0040E7F9 50      PUSH EAX
0040E7FA E8 26370100 CALL BoxMan_e.00421F25 ; <== Concat String
0040E7FF 50      PUSH EAX
0040E800 8D4C24 2C  LEA ECX,DWORD PTR SS:[ESP+2C]
0040E804 C64424 20 03 MOV BYTE PTR SS:[ESP+20],3
0040E809 E8 FC350100 CALL BoxMan_e.00421E0A
0040E80E 8D4C24 08  LEA ECX,DWORD PTR SS:[ESP+8]
0040E812 C64424 1C 02 MOV BYTE PTR SS:[ESP+1C],2
0040E817 E8 01350100 CALL BoxMan_e.00421D1D
0040E81C 885C24 1C  MOV BYTE PTR SS:[ESP+1C],BL
0040E820 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
0040E824 EB 58      JMP SHORT BoxMan_e.0040E87E ; <== else
0040E826 68 D01E4400 PUSH BoxMan_e.00441ED0 ; ASCII "AA"
0040E82B 8D4C24 2C  LEA ECX,DWORD PTR SS:[ESP+2C]
0040E82F E8 38380100 CALL BoxMan_e.0042206C
0040E834 8D4C24 08  LEA ECX,DWORD PTR SS:[ESP+8]
0040E838 6A 02      PUSH 2
0040E83A 51      PUSH ECX
0040E83B 8D4C24 30  LEA ECX,DWORD PTR SS:[ESP+30]
0040E83F E8 07F90000 CALL BoxMan_e.0041E14B
0040E844 68 D41E4400 PUSH BoxMan_e.00441ED4 ; ASCII "fasfwefawfafafae"
0040E849 8D5424 10  LEA EDX,DWORD PTR SS:[ESP+10]
0040E84D 50      PUSH EAX
0040E84E 52      PUSH EDX
0040E84F C64424 28 04 MOV BYTE PTR SS:[ESP+28],4
0040E854 E8 CC360100 CALL BoxMan_e.00421F25
----- Create NewString -----

```

- Tiếp đó là quá trình tạo chuỗi MD5Hash dựa trên chuỗi mới này . Quá trình tạo chuỗi cũng đồng thời với việc xuất chuỗi MD5Hash :

```

0040DF16 E8 65CFFFFF CALL BoxMan_e.0040AE80 ; <== Get MD5Hash
----- MD5Hash -----
0040AEAE E8 B93B0100 CALL BoxMan_e.0041EA6C
0040AEB3 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
0040AEB7 E8 A40A0000 CALL BoxMan_e.0040B960 ; <== MD5Start
0040AEBC 56      PUSH ESI
0040AEBD 57      PUSH EDI
0040AEBE 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
0040AEC2 C74424 78 00000>MOV DWORD PTR SS:[ESP+78],0
0040AECA E8 810C0000 CALL BoxMan_e.0040BB50 ; <== MD5Update
0040AEFC 8B7424 78  MOV ESI,DWORD PTR SS:[ESP+78]
0040AED3 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
0040AED7 56      PUSH ESI
0040AED8 E8 130B0000 CALL BoxMan_e.0040B9F0 ; <== MD5Finished
----- MD5Hash -----

```

- Chuỗi MD5Hash này sa đó được cắt lấy 16 ký tự đầu tiên để tạo thành chuỗi Serial :

```

0040DF3E 6A 10      PUSH 10 ; <== Get 16 charts
0040DF40 52      PUSH EDX ; <== from the FIRST chart
0040DF41 8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18] ; <== of MD5HashString
0040DF45 E8 01020100 CALL BoxMan_e.0041E14B ; <== Cutting

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : a510-b53e-6d34-c837

III – KeyGen :

- /Section /- Kết hợp 2 ký tự của chuỗi U nhập với chuỗi mặc định.
- /Section /- Tạo chuỗi MD5Hash
- /Section /- Cắt 16 ký tự đầu tiên tạo thành chuỗi Serial.

IV – End of Tut :

- Finished – *August 10, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.colorschemer.com
Production :	Color Schemer.
SoftWare :	Color Schemer 3.1
Copyright by :	Copyright © 2002 - 2004 Color Schemer. All Rights Reserved.
Type :	Code / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Color Schemer 3.1

Color Schemer v3 is a quick, compact & easy to use color matching application that will have you cranking out creative color schemes in no time! Match colors with the new RYB color wheel mode, build color schemes with unlimited Favorite Colors, and highlight color relationships directly on Color Schemer v3.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**
 - Nhập thử Fake Serial, ta nhận được thông báo "**The registration information you have provided is invalid. Please recheck your order number and registration key.**" Tìm chuỗi thông báo này trong Olly và ta tìm được ở địa chỉ :
- | | |
|--|--|
| 004A8187 . B8 98824A00 MOV EAX,Colorsch.004A8298 | ; ASCII "The registration information you have provided is invalid. Please recheck your order number and registration key." |
| - Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này : | |
| 004A804A . E8 7DE1FAFF CALL Colorsch.004561CC | ; <== Set BP here |

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến :
- | | |
|--|-------------------|
| 004A804A . E8 7DE1FAFF CALL Colorsch.004561CC | ; <== Set BP here |
|--|-------------------|

```

004A804F |. 8B45 FC    MOV EAX,DWORD PTR SS:[EBP-4]
004A8052 |. E8 BDC7F5FF CALL Colorsch.00404814 ; <== Get Length of CODE
004A8057 |. 83F8 08    CMP EAX,8 ; <== CODE must be 8 charts
004A805A |. 0F85 1C010000 JNZ Colorsch.004A817C
004A8060 |. 8D55 F4    LEA EDX,DWORD PTR SS:[EBP-C]
004A8063 |. 8B83 04030000 MOV EAX,DWORD PTR DS:[EBX+304]
004A8069 |. E8 5EE1FAFF CALL Colorsch.004561CC
004A806E |. 8B55 F4    MOV EDX,DWORD PTR SS:[EBP-C] ; <== CODE
004A8071 |. 8D4D F8    LEA ECX,DWORD PTR SS:[EBP-8]
004A8074 |. 8BC3      MOV EAX,EBX
004A8076 |. E8 91020000 CALL Colorsch.004A830C ; <== Trace into here
- Dùng F7 trace into ta đến quá trình mã hoá đầu tiên :
004A8341 |. 8B45 FC    MOV EAX,DWORD PTR SS:[EBP-4] ; <== CODE
004A8344 |. E8 CBC4F5FF CALL Colorsch.00404814 ; <== Len.C
004A8349 |. 8BF0      MOV ESI,EAX
004A834B |. 85F6      TEST ESI,ESI
004A834D |. 7E 17      JLE SHORT Colorsch.004A8366
004A834F |. BB 01000000 MOV EBX,1 ; <== i = 1
004A8354 |> 8B45 FC    /MOV EAX,DWORD PTR SS:[EBP-4] ; <== CODE
004A8357 |. 0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== CODE[i-1]
004A835C |. C1E0 10    |SHL EAX,10 ; <== Temp = CODE[i-1] * 0x10000
004A835F |. 0145 F8    |ADD DWORD PTR SS:[EBP-8],EAX ; <== Value = Value + Temp
004A8362 |. 43        |INC EBX ; <== i++
004A8363 |. 4E        |DEC ESI ; <== Len.C --
004A8364 |.^ 75 EE    \JNZ SHORT Colorsch.004A8354 ; <== Loop until Len.C == 0x0
004A8366 |> 8D55 EC    LEA EDX,DWORD PTR SS:[EBP-14]
004A8369 |. 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8] ; <== Value
004A836C |. E8 B302F6FF CALL Colorsch.00408624 ; <== Convert to String (%i) : cStr
- Quá trình mã hoá thứ hai diễn ra như sau :
004A837F |. BB 01000000 MOV EBX,1 ; <== i = 1
004A8384 |> 8B45 FC    /MOV EAX,DWORD PTR SS:[EBP-4] ; <== CODE
004A8387 |. 0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== CODE[i-1]
004A838C |. B9 09000000 |MOV ECX,9 ; <== DefaultValue : dV
004A8391 |. 33D2      |XOR EDX,EDX
004A8393 |. F7F1      |DIV ECX ; <== Value = CODE[i-1] % dV
004A8395 |. 8955 F4    |MOV DWORD PTR SS:[EBP-C],EDX
004A8398 |. 8D45 E8    |LEA EAX,DWORD PTR SS:[EBP-18]
004A839B |. 8B55 EC    |MOV EDX,DWORD PTR SS:[EBP-14] ; <== cStr
004A839E |. 8A541A FF    |MOV DL,BYTE PTR DS:[EDX+EBX-1] ; <== cStr[i-1]
004A83A2 |. E8 95C3F5FF |CALL Colorsch.0040473C ; <== [ from here ]
004A83A7 |. 8B45 E8    |MOV EAX,DWORD PTR SS:[EBP-18] ; <== [ Temp = cStr[i-1] - 0x30 ]
004A83AA |. E8 B103F6FF |CALL Colorsch.00408760 ; <== [ to here ]
004A83AF |. 8945 F0    |MOV DWORD PTR SS:[EBP-10],EAX ; <== Temp
004A83B2 |. 8B45 F4    |MOV EAX,DWORD PTR SS:[EBP-C] ; <== Value
004A83B5 |. 0345 F0    |ADD EAX,DWORD PTR SS:[EBP-10] ; <== Value = Value + Temp
004A83B8 |. B9 09000000 |MOV ECX,9 ; <== DefaultValue : dV
004A83BD |. 99        |CDQ
004A83BE |. F7F9      |IDIV ECX ; <== Value = Value % 9
004A83C0 |. 8BC2      |MOV EAX,EDX ; <== Value
004A83C2 |. 8D55 E4    |LEA EDX,DWORD PTR SS:[EBP-1C]
004A83C5 |. E8 5A02F6FF |CALL Colorsch.00408624 ; <== %X
004A83CA |. 8B55 E4    |MOV EDX,DWORD PTR SS:[EBP-1C]

```

```

004A83CD |. 8BC7      |MOV EAX,EDI
004A83CF |. E8 48C4F5FF |CALL Colorsch.0040481C ; <== Concat String : lStr
004A83D4 |. 43         |INC EBX
004A83D5 |. 4E         |DEC ESI
004A83D6 |.^ 75 AC    |JNZ SHORT Colorsch.004A8384
- Qúa trình tạo thành chuỗi Serial thực diễn ra như sau :
004A83E1 |. B9 05000000 MOV ECX,5 ; <== Get 5 charts
004A83E6 |. BA 02000000 MOV EDX,2 ; <== from second chart
004A83EB |. 8B45 EC    MOV EAX,DWORD PTR SS:[EBP-14] ; <== of cStr
004A83EE |. E8 81C6F5FF CALL Colorsch.00404A74 ; <== Cutting
004A83F3 |. FF75 E0    PUSH DWORD PTR SS:[EBP-20] ; <== New cStr : ncStr
004A83F6 |. 8D45 DC    LEA EAX,DWORD PTR SS:[EBP-24]
004A83F9 |. 50         PUSH EAX
004A83FA |. 8B07      MOV EAX,DWORD PTR DS:[EDI] ; <== lStr
004A83FC |. B9 08000000 MOV ECX,8 ; <== Get 8 charts
004A8401 |. 33D2      XOR EDX,EDX ; <== Cutting : lStr
004A8403 |. E8 6CC6F5FF CALL Colorsch.00404A74
004A8408 |. FF75 DC    PUSH DWORD PTR SS:[EBP-24] ; <== lStr
004A840B |. 8BC7      MOV EAX,EDI ; <== Default String : CS3
004A840D |. BA 03000000 MOV EDX,3 ; <== Concat 3 String
004A8412 |. E8 BDC4F5FF CALL Colorsch.004048D4 ; <== Real Serial

```

/*/*/* - SERIAL tương ứng :

Code : 85985041

Serial : CS377872-46116548

III – KeyGen :

- /Section 1/- Tạo chuỗi ngẫu nhiên gồm 8 ký tự . (CODE)
- /Section 2/- Mã hoá lần đầu theo công thức :

$$\text{Value} = \text{Value} + ((\text{reaCode}[i] \& 0xFF) * 0x10000);$$
- /Section 3/- Mã hoá lần thứ hai theo công thức :

$$\text{Value} = (((\text{reaCode}[i] \& 0xFF) \% 9) + (\text{reaTemp_01}[i] - 0x30)) \% 9;$$
- /Section 4/- Gắn kết các chuỗi lại với nhau .

IV – End of Tut :

- Finished – 25/07/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.computer-alarm-clock-software.com
Production :	Computer Alarm Clock.
SoftWare :	Computer Alarm Clock 2.0
Copyright by :	Copyright © 2004 Computer Alarm Clock. All Rights Reserved.
Type :	Code / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10

Unpack : N / A
 Request : Correct Serial / KeyGen

Computer Alarm Clock 2.0

Computer Alarm Clock is a desktop program with full-featured alarm system, that allow an user to set multiple alarms. Each alarm can be set to play MP3 songs, CD tracks, videos and MPEG movies or just display a message. Computer Alarm Clock can be vastly customized and the whole package size is 660 Kb only!

Your alarm clock is something you use everyday, so why not have one that you can enjoy?

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**
 - Nhập thử Fake Serial, ta nhận được thông báo "Please restart Music Alarm Clock" Như vậy ta phải tìm xem chương trình sẽ lưu thông tin vào FILE hay REG . Cũng trong quá trình tìm kiếm, ta nhận thấy trong FUNCTION chứa thông báo này có dòng :
- ```
0048E42A . B9 5CE54800 MOV ECX,cac.0048E55C ; ASCII "clapeoq.lst"
```
- Và trong thư mục cài đặt cũng có FILE này . Như vậy ta có thể nghĩ là chương trình sẽ lưu thông tin vào trong FILE này . Ta tìm đến các hàm API ReadFile và WriteFile và đặt BreakPoint .
  - Load chương trình lên, chương trình sẽ dừng lại tại hàm ReadFile, truy theo quá trình ta tìm đến được FUNCTION chứa quá trình mã hoá . Ta đặt BP tại lệnh CALL đầu tiên của FUNCTION này :
- ```
00490C7B . E8 E020FCFF CALL cac.00452D60 ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống đến quá trình cộng dồn giá trị các ký tự của chuỗi U:

```
00490DA0 . BA 01000000 MOV EDX,1 ; <== i=1
00490DA5 > 8B4D F8 MOV ECX,DWORD PTR SS:[EBP-8] ; <== U
00490DA8 . 8A4C11 FF MOV CL,BYTE PTR DS:[ECX+EDX-1] ; <== U[i-1]
00490DAC . 81E1 FF000000 AND ECX,0FF
00490DB2 . 014D DC ADD DWORD PTR SS:[EBP-24],ECX ; <== CumV = CumV + U[i-1]
00490DB5 . 42 INC EDX ; <== i++
00490DB6 . 48 DEC EAX ; <== Len.U --
00490DB7 .^ 75 EC JNZ SHORT cac.00490DA5 ; <== Loop Until Len.U = 0x0
```

- Giá trị này sau đó sẽ được chuyển thành chuỗi ở dạng DEC tương ứng theo định dạng "%i" .

```
00490DB9 > \8D55 F8 LEA EDX,DWORD PTR SS:[EBP-8]
00490DBC . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24]
00490DBF . E8 647FF7FF CALL cac.00408D28
```

- Kế đến là quá trình so sánh . Quá trình so sánh này diễn ra theo ba giai đoạn . Các giai đoạn được diễn giải như sau :

```
00490E37 . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== cumStr[Len - 1]
00490E3A . 8B55 E4 MOV EDX,DWORD PTR SS:[EBP-1C] ; <== S[3]
00490E3D . E8 AE3DF7FF CALL cac.00404BF0 ; <== Must be equal
00490E42 . 75 23 JNZ SHORT cac.00490E67
00490E44 . 8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18] ; <== cumStr[0]
00490E47 . 8B55 E0 MOV EDX,DWORD PTR SS:[EBP-20] ; <== S[7]
00490E4A . E8 A13DF7FF CALL cac.00404BF0 ; <== Must be equal
00490E4F . 75 16 JNZ SHORT cac.00490E67
00490E51 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10] ; <== U[0][1]
00490E54 . BA F40E4900 MOV EDX,cac.00490EF4 ; ASCII "a1"
00490E59 . E8 923DF7FF CALL cac.00404BF0 ; <== Must be equal
```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErStEaM Serial : a158w8x1 or a1u8grp1

III – KeyGen :

- /Section 1/- Tạo chuỗi ngẫu nhiên gồm 8 ký tự, hai ký tự đầu tiên luôn là “**aI**”
- /Section 2/- Tính công dồn giá trị các ký tự của chuỗi U và chuyển theo định dạng “%i”
- /Section 3/- Thực hiện quá trình chuyển đổi sau :

```
reaSerial[3] = reaTemp[lstrlen(reaTemp)-1];
reaSerial[7] = reaTemp[0];
```

IV – End of Tut :

- Finished – **26/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.cyclonecomputing.com
Production :	Cyclone Computing, Inc.
SoftWare :	Cyclone Internet History Killer Pro 3.60
Copyright by :	Copyright © 2002-2004 Cyclone Computing, Inc. All Rights Reserved.
Type :	Name / Serial / Registration
Packed :	ASPack 2.12 -> Alexey Solodovnikov
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	Manual
Request :	Correct Registration / KeyGen

Cyclone Internet History Killer Pro 3.60

Internet History Killer Pro is a very useful program for cleaning up all the internet tracks and past computer activities. Even though, many of the tasks can be performed manually, Internet History Killer Pro can automate this process for you. The program is designed to protect your privacy by cleaning up unwanted program history data and internet tracks on your computer. It allows you to erase the cache, cookies, history, visited URLs, typed URLs, autocomplete forms data, index.dat files of your browser, and Window's swap file, temp folders, run history, search history, open/save history, recent documents and more. The cookie cleaning feature allows you to specify cookies to keep, so that you don't erase your important login cookies. In addition, you can change the paths of several system folders (Cache, Cookies, History, Favorites etc.) and customize the IE title bar text.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **ASPack 2.12 -> Alexey Solodovnikov**. Sau khi UnPACK kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**.
 - Nhập thử User, Fake Serial và Fake Registration, ta nhận được thông báo "The registration information you have entered invalid, please enter valid registration information and try again!", và tìm được chuỗi này ở địa chỉ :
- 004F6C94 . B8 146E4F00 MOV EAX,unpack.004F6E14 ; ASCII "The registration information you have entered invalid, please enter valid registration information and try again!"

- Dò ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FuncTion này :
 004F6ACE . E8 0512FAFF CALL unpack.00497CD8 ; <== Set BreakPoint here

II – Cracking :

- Load và chạy chương trình, chương trình dừng lại tại điểm đặt BreakPoint . Trace tiếp ta đến :
 004F6B85 . E8 A6D0FFFF CALL unpack.004F3C30 ; <== Trace Into Here

- Dùng F7 để trace into :

004F3C61 . E8 A60DF1FF CALL unpack.00404A0C	; <== Get Lenght of Serial
004F3C66 . 3B46 4C CMP EAX,DWORD PTR DS:[ESI+4C]	; <== Not greater than 10 charts
004F3C69 . 7F 0D JG SHORT unpack.004F3C78	
004F3C6B . 8B45 FC MOV EAX,[LOCAL.1]	
004F3C6E . E8 990DF1FF CALL unpack.00404A0C	; <== Get Lenght of Serial
004F3C73 . 3B46 50 CMP EAX,DWORD PTR DS:[ESI+50]	; <== But not less than 3 charts
004F3C76 . 7D 0C JGE SHORT unpack.004F3C84	

- Như vậy chiều dài của chuỗi Serial nằm trong khoảng (3 -10) ký tự . Trace tiếp ta đến quá trình mã hoá đầu tiên :

004F3C90 >/8B45 FC /MOV EAX,[LOCAL.1]	; <== Serial
004F3C93 . 8A4418 FF MOV AL,BYTE PTR DS:[EAX+EBX-1]	; <== Serial[Len.S-1]
004F3C97 . 25 FF000000 AND EAX,0FF	; <== Serial[Len.S-1]
004F3C9C . 33D2 XOR EDX,EDX	; <== Value = 0x0
004F3C9E . 52 PUSH EDX	; <== Value
004F3C9F . 50 PUSH EAX	; <== Serial[Len.i-i]
004F3CA0 . 8B46 68 MOV EAX,DWORD PTR DS:[ESI+68]	; <== Temp = 0x131A3C8
004F3CA3 . 8B56 6C MOV EDX,DWORD PTR DS:[ESI+6C]	
004F3CA6 . E8 BD1CF1FF CALL unpack.00405968	

===== Trace Into =====

004059DB >\F7F3 DIV EBX	; <== Temp = Temp % Code[Len.C-1]
004059DD . 92 XCHG EAX,EDX	
004059DE . 31D2 XOR EDX,EDX	

===== Trace Into =====

004F3CAB . 52 PUSH EDX	; /Arg2
004F3CAC . 50 PUSH EAX	; Arg1
004F3CAD . 8D45 E4 LEA EAX,[LOCAL.7]	;
004F3CB0 . E8 DF58F1FF CALL unpack.00409594	; \unpack.00409594

===== Trace Into =====

Chuyển giá trị Temp từ giá trị HEX sang chuỗi dạng DEC . Với định dạng tương ứng “%d”

===== Trace Into =====

004F3CB5 . 8B55 E4 MOV EDX,[LOCAL.7]	; <== NewString : nStr
004F3CB8 . 8D45 F4 LEA EAX,[LOCAL.3]	
004F3CBB . E8 540DF1FF CALL unpack.00404A14	; <== Concat all nStr
004F3CC0 . 4B DEC EBX	; <== Len.C --
004F3CC1 > 8B45 FC MOV EAX,[LOCAL.1]	; <== Serial
004F3CC4 . E8 430DF1FF CALL unpack.00404A0C	; <== Get Len.S
004F3CC9 . 83E8 06 SUB EAX,6	; <== Value = Len.S - 6
004F3CCC . 3BD8 CMP EBX,EAX	; <== Value compare with Len.S
004F3CCE . 7C 04 JL SHORT unpack.004F3CD4	; <== jmp if Len.S less than Value
004F3CD0 . 85DB TEST EBX,EBX	; <== if Value > 0
004F3CD2 .^7F BC \JG SHORT unpack.004F3C90	; <== Continue Loop

- Kết thúc vòng lặp này ta có được chuỗi mới

004F3CDA . E8 051DF1FF CALL unpack.004059E4	; <== Trace Into here
--	-----------------------

- Dùng F7 trace into ta đến quá trình mã hoá thứ hai . Đoạn đầu của quá trình nà là một loạt quá trình so sánh ký tự đầu tiên với các giá trị đặc biệt . Nhưng thực tế là không bao giờ đúng cả . Và kết quả là luôn nhảy đến một đoạn CODE mã hoá duy nhất :

```

00405B32 |> 8A442E FF |MOV AL,BYTE PTR DS:[ESI+EBP-1] ; <== nStr[i]
00405B36 |.|8BD0 |MOV EDX,EAX ; <== nStr[i]
00405B38 |.|80C2 D0 |ADD DL,0D0 ; <== Temp_01 = nStr[i] + 0xD
00405B3B |.|80EA 0A |SUB DL,0A ; <== Temp_01 = Temp_01 - 0xA
00405B3E |.|73 62 |JNB SHORT unpack.00405BA2 ; <== JMP out if End of nStr
00405B40 |.|8BF8 |MOV EDI,EAX ; <== nStr[i]
00405B42 |.|81E7 FF000000 |AND EDI,0FF ; <== nStr[i]
00405B48 |.|83EF 30 |SUB EDI,30 ; <== Temp_02 = nStr[i] - 0x30
00405B4B |.|837C24 0C 00 |CMP DWORD PTR SS:[ESP+C],0
00405B50 |.|75 09 |JNZ SHORT unpack.00405B5B
00405B52 |.|837C24 08 00 |CMP DWORD PTR SS:[ESP+8],0
00405B57 |.|72 49 |JB SHORT unpack.00405BA2
00405B59 |.|EB 02 |JMP SHORT unpack.00405B5D
00405B5B |>|7C 45 |JL SHORT unpack.00405BA2
00405B5D |>|817C24 0C CCC>|CMP DWORD PTR SS:[ESP+C],0CCCCCCC
00405B65 |.|75 0C |JNZ SHORT unpack.00405B73
00405B67 |.|817C24 08 CCC>|CMP DWORD PTR SS:[ESP+8],CCCCCCCC
00405B6F |.|76 04 |JBE SHORT unpack.00405B75
00405B71 |.|EB 2F |JMP SHORT unpack.00405BA2
00405B73 |>|7F 2D |JG SHORT unpack.00405BA2
00405B75 |>|6A 00 |PUSH 0
00405B77 |.|6A 0A |PUSH 0A ; <== Default Value : 0xA
00405B79 |.|8B4424 10 |MOV EAX,DWORD PTR SS:[ESP+10] ; <== Value
00405B7D |.|8B5424 14 |MOV EDX,DWORD PTR SS:[ESP+14]
00405B81 |.|E8 42FDFFFF |CALL unpack.004058C8 ; <== Value = Value * 0xA
----- Trace Into -----

```

```

004058CA |.|8B4424 10 |MOV EAX,DWORD PTR SS:[ESP+10]
004058CE |.|F72424 |MUL DWORD PTR SS:[ESP]
004058D1 |.|89C1 |MOV ECX,EAX
004058D3 |.|8B4424 04 |MOV EAX,DWORD PTR SS:[ESP+4]
004058D7 |.|F76424 0C |MUL DWORD PTR SS:[ESP+C]
004058DB |.|01C1 |ADD ECX,EAX
004058DD |.|8B0424 |MOV EAX,DWORD PTR SS:[ESP]
004058E0 |.|F76424 0C |MUL DWORD PTR SS:[ESP+C]
004058E4 |.|01CA |ADD EDX,ECX
----- NOTE -----

```

Điểm hay của đoạn này là PROGRAMMER đã sử dụng một thủ thuật để tính giá trị tràn . Chẳng hạn ta có số “**110**” . Ta sẽ phân thành hai phần là “**1**” và “**10**” . Khi ta nhân với một giá trị chẳng hạn là “**22**” thì ta thực hiện như sau : “**10**” * “**22**” = “**220**” . Ở đây ta tách ra làm hai phần là “**2**” và “**20**” . Và ta thực hiện tiếp phép nhân : “**1**” * “**22**” = “**22**” . Ta lấy kết quả này cộng với phần đầu tách ra của kết quả trên, ta có : “**22**” + “**2**” = “**24**” . Như vậy ta có hai số “**24**” và “**20**” . Nối hai chuỗi này lại ta có giá trị của nó bằng đúng giá trị của phép nhân “**110**” * “**22**” = “**2420**”

```

----- Trace Into -----
00405B86 |.|52 |PUSH EDX
00405B87 |.|50 |PUSH EAX ; <== Value
00405B88 |.|8BC7 |MOV EAX,EDI ; <== Temp_02
00405B8A |.|99 |CDQ
00405B8B |.|030424 |ADD EAX,DWORD PTR SS:[ESP] ; <== Value = Value + Temp_02
00405B8E |.|135424 04 |ADC EDX,DWORD PTR SS:[ESP+4]

```

```

00405B92 |. |83C4 08    |ADD ESP,8
00405B95 |. |894424 08   |MOV DWORD PTR SS:[ESP+8],EAX      ; <== Value
00405B99 |. |895424 0C   |MOV DWORD PTR SS:[ESP+C],EDX
00405B9D |. |45        |INC EBP          ; <== i++
00405B9E |. |33DB      |XOR EBX,EBX
00405BA0 |.^|EB 90     |JMP SHORT unpack.00405B32      ; <== Continue Loop
- Quá trình này thoát trông rất phức tạp nhưng có thể được viết lại như sau :

```

$$\text{Value} = (\text{Value} * 0xA) + (\text{NewString}[i] - 0x30)$$

- Kết thúc quá trình này ta có hai trường hợp (1) giá trị của Value không bị tràn (nghĩa là nhỏ hơn 0xFFFFFFFF), hoặc (2) giá trị của Value bị tràn . Lúc này, hai phần này sẽ được chuyển như sau :

```

00405C03 |> |8B4424 08   |MOV EAX,DWORD PTR SS:[ESP+8]      ; <== Low partion of Value
00405C07 |. |8B5424 0C   |MOV EDX,DWORD PTR SS:[ESP+C]      ; <== High partion of Value

```

- Sau đó, hai phần này sẽ được chuyển thành chuỗi tương ứng :

```

004F3D07 |. E8 D858F1FF  CALL unpack.004095E4      ; \unpack.004095E4
----- Trace Into I -----

```

```

004095FF |. E8 C4FFFF  CALL unpack.004094C8
----- Trace Into II -----

```

```

00409546 |> |\DF28      |FILD QWORD PTR DS:[EAX]
00409548 |> DF0424    |FILD WORD PTR SS:[ESP]
0040954B |. D9C1      |FLD ST(1)
0040954D |> 4E        |/DEC ESI
0040954E |. D9F8      |FPREM
00409550 |. DF1C24    |FISTP WORD PTR SS:[ESP]
00409553 |. DCF9      |FDIV ST(1),ST
00409555 |. 8A0424    |MOV AL,BYTE PTR SS:[ESP]
00409558 |. 04 30      |ADD AL,30
0040955A |. 3C 3A      |CMP AL,3A
0040955C |. 72 02      |JB SHORT unpack.00409560
0040955E |. 04 07      |ADD AL,7
00409560 |> 8806      |MOV BYTE PTR DS:[ESI],AL
00409562 |. D9C1      |FLD ST(1)
00409564 |. D8D3      |FCOM ST(3)
00409566 |. 9B        |WAIT
00409567 |. DFE0      |FSTSW AX
00409569 |. 9E        |SAHF
0040956A |.^ 73 E1    |JNB SHORT unpack.0040954D
0040956C |. D96C24 02  |FLDCW WORD PTR SS:[ESP+2]
00409570 |. 83C4 04    |ADD ESP,4
00409573 |. DDC3      |FFREE ST(3)
00409575 |. DDC2      |FFREE ST(2)
00409577 |. DDC1      |FFREE ST(1)
00409579 |. DDC0      |FFREE ST
0040957B |. 59        |POP ECX
0040957C |. 29F1      |SUB ECX,ESI
0040957E |. 29CA      |SUB EDX,ECX
00409580 |. 76 10      |JBE SHORT unpack.00409592
00409582 |. 29D6      |SUB ESI,EDX
00409584 |. B0 30      |MOV AL,30
00409586 |. 01D1      |ADD ECX,EDX
00409588 |. EB 03      |JMP SHORT unpack.0040958D
0040958A |> 880432    |/MOV BYTE PTR DS:[EDX+ESI],AL

```

```

0040958D |> 4A      DEC EDX
0040958E |.^ 75 FA    \JNZ SHORT unpack.0040958A
00409590 |. 8806     MOV BYTE PTR DS:[ESI],AL
----- Trace Into II -----
----- Trace Into I -----

```

- Thực chất của quá trình này như đã nói bên trên là chuyển hai phần của Value thành chuỗi, nhưng ở đây có hai điểm lưu ý : chiều dài của chuỗi tạo thành chỉ là 10 ký tự vì vậy ta có hai trường hợp (1) Nếu Value không tràn hay phần tràn của Value nhỏ hơn 2 bytes nghĩa là Value có thể được xem là có tổng số ký tự nhỏ hơn 10 thì sẽ được chèn trước đó ký tự “0” để cho đủ 10 ký tự, hoặc (2) nếu Value bị tràn mà phần tràn lớn hơn 2 bytes hay nói cách khác Value có hơn 10 ký tự thì các ký tự ở cuối sẽ được cắt bỏ sao cho phần còn lại đủ 10 ký tự .

- Cuối cùng chương trình mới kiểm tra chiều dài chính xác của chuỗi U nhập :

004F6BA6 . E8 89E4F0FF CALL unpack.00405034	; <== Get Len.S
004F6BAB . 83F8 0A CMP EAX,0A	; <== Len.S must be 10 charts
004F6BAE . 0F85 D7000000 JNZ unpack.004F6C8B	

- Chuỗi xuất ra chính là chuỗi REGISTRATION thực .

- Chuỗi U thực sự không tham gia vào quá trình mã hoá .

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : GNPBZZBSDX	Registration : 0092666062
User : VHT-cRaCkErS	Serial : 4BEDT8EQVG	Registration : 1BDB263F7A

III – KeyGen :

/Section 1/- Tính **Temp = Temp % Serial[Len.S-1]** và chuyển theo định dạng “%d”
 /Section 2/- Tính **Value = (Value * 0xA) + (NewString[i] – 0x30)** kết hợp với thủ thuật tính tràn .

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};
char reaRegistration[64]={0};
char reaRandomChar[64]="0147852369AQWERTYUIOPLKJHGFDSZXCVBNM";
char reaTemp[10] = {0};

int LenUser=0, LenSerial=0, i=0;
int Temp=0, Over=0, Value=0;

LenUser=GetDlgItemText(IDC_Name, reaName, 64);
if (LenUser < 1)
{
  MessageBox("===== Your Name atleast 1 chart ===== ", "Hey !! Please
input your Name again !! ");
}
else
{
  srand( (unsigned)time( NULL ) );

  i=0;
  while ( i < 10 )
  {

```

```

        reaSerial[i] = reaRandomChar[rand() % 36];
        i++;
    }

    LenSerial = lstrlen(reaSerial);
    while (LenSerial >= 0x4)
    {
        Temp = 0x131A3C8 % reaSerial[LenSerial - 1];
        wsprintf(reTemp,"%i",Temp);
        lstrcat(reRegistration,reTemp);
        LenSerial--;
    }
    i=0;
    while ( i < lstrlen(reRegistration) )
    {
        _asm
        {
            MOV ESI, i
            MOVSX EDI, reaRegistration[ESI]
            SUB EDI, 0x30
            MOV EAX, Over
            MOV ECX, 0xA
            CDQ
            MUL ECX
            MOV EBX, EAX
            MOV EAX, Value
            CDQ
            MUL ECX
            ADD EDX, EBX
            ADD EAX, EDI
            MOV Value, EAX
            MOV Over, EDX
        }
        i++;
    }
    if ( 0xFF >= Over )
    {
        wsprintf(reRegistration,"%02X%08X",Over, Value);
    }
    else
    {
        wsprintf(reTemp,"%X%08X",Over, Value);
        lstrcpy(reRegistration,reTemp,11);
    }
    SetDlgItemText(IDC_Serial,reaSerial);
    SetDlgItemText(IDC_Registration,reRegistration);
}

```

V – End of Tut :

- Finished – **09/07/2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.anthemion.co.uk/dialogblocks
Production :	Anthemion Software Ltd.
SoftWare :	DialogBlocks 1.52
Copyright by :	Copyright © Anthemion Software Ltd, May 2004. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

DialogBlocks 1.52

DialogBlocks is an editor for authoring cross-platform dialogs for wxWidgets applications. You can use it to produce help files for your Windows, Linux, or Mac applications, in conjunction with wxWidgets.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "*Sorry, invalid registration key...*" Ta tìm được đoạn CODE này ở địa chỉ :
 00413EDC > \B8 44BA6C00 MOV EAX,dialogbl.006CBA44 ; ASCII "Sorry, invalid registration key. The key should be of the form:EA0927CF-E8CF149B-416363EE(three groups of 8 characters).Please also check that your specified user name is the sameas the one under which you purchased DialogBlocks."...
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
 00413D96 . E8 A52F2200 CALL <JMP.&MSVCRT._EH_prolog> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Để đi đến được đoạn CODE mã hoá của chương trình ta cần trace into vài lần :

 00413E5F . E8 FC140000 CALL dialogbl.00415360 ; <== Real Serial I
 ----- **TraceInto I** -----
 004153B1]. E8 EBFEFFFF CALL dialogbl.004152A1 ; <== Trace Into
 ----- **TraceInto II** -----
 004152C7]. E8 28FEFFFF CALL dialogbl.004150F4 ; <== Trace Into here
 ----- **TraceInto III** -----

- Quá trình chuyển đổi chuỗi U nhập về dạng LowerCase :

 0041514E |>/8A07 /MOV AL,BYTE PTR DS:[EDI]
 00415150]. |3C 20 |CMP AL,20
 00415152]. |74 0E |JE SHORT dialogbl.00415162
 00415154]. |0FBEC0 |MOVSX EAX,AL
 00415157]. |50 |PUSH EAX ; /c

```

00415158 |. FF15 48746600 |CALL DWORD PTR DS:[<&MSVCRT(tolower>] ; \tolower
0041515E |. 8806      |MOV BYTE PTR DS:[ESI],AL
00415160 |. 59       |POP ECX
00415161 |. 46       |INC ESI
00415162 |> 47      |INC EDI
00415163 |. 381F      |CMP BYTE PTR DS:[EDI],BL
00415165 |.^75 E7      |JNZ SHORT dialogbl.0041514E

```

- Kết hợp với chuỗi mặc định “*Anthemion Software DialogBlocks*” :

```

00415167 |> \FF75 EC    PUSH [LOCAL.5] ; /src
0041516A |. 8D85 84FDFFFF LEA EAX,[LOCAL.159] ; |
00415170 |. 881E      MOV BYTE PTR DS:[ESI],BL ; |
00415172 |. 50       PUSH EAX ; |dest
00415173 |. E8 7A1C2200 CALL <JMP.&MSVCRT.strcat> ; \ strcat

```

- Quá trình tạo chuỗi MD5Hash :

```

00415188 |. E8 D2D0FFFF CALL dialogbl.0041225F ; <== MD5Start
0041518D |. 8D85 84FDFFFF LEA EAX,[LOCAL.159]
00415193 |. 50       PUSH EAX ; /s
00415194 |. E8 3B1C2200 CALL <JMP.&MSVCRT.strlen> ; \ strlen
00415199 |. 50       PUSH EAX
0041519A |. 8D85 84FDFFFF LEA EAX,[LOCAL.159]
004151A0 |. 50       PUSH EAX
004151A1 |. 8D45 84   LEA EAX,[LOCAL.31]
004151A4 |. 50       PUSH EAX
004151A5 |. E8 DDD0FFFF CALL dialogbl.00412287 ; <== MD5Update
004151AA |. 8D45 84   LEA EAX,[LOCAL.31]
004151AD |. 50       PUSH EAX
004151AE |. 8D45 DC   LEA EAX,[LOCAL.9]
004151B1 |. 50       PUSH EAX
004151B2 |. E8 70D1FFFF CALL dialogbl.00412327 ; <== MD5Finished

```

- Chuỗi MD5Hash được tạo thành là 16 bytes (16 ký tự). Ta chia làm 4 đoạn, mỗi đoạn 4 bytes . Quá trình tạo chuỗi Serial thực chỉ là quá trình chuyển đổi ba đoạn đầu tiên của chuỗi MD5Hash thành chuỗi Serial theo định dạng “XXXXXXX-YYYYYYY-ZZZZZZZ” :

```

004151BC |. 8D75 DC   LEA ESI,[LOCAL.9] ; <== Add of MD5Hash
004151BF |> 33C9      /XOR ECX,ECX
004151C1 |. 33C0      |XOR EAX,EAX
004151C3 |> 0FB61406  /MOVZX EDX,BYTE PTR DS:[ESI+EAX] ; <== MD5Str[i]
004151C7 |. C1E1 08    ||SHL ECX,8
004151CA |. 03CA      ||ADD ECX,EDX
004151CC |. 40       ||INC EAX
004151CD |. 83F8 04    ||CMP EAX,4
004151D0 |.^ 7C F1    |JL SHORT dialogbl.004151C3
004151D2 |. A1 F4FC6D00 |MOV EAX,DWORD PTR DS:[6DFCF4] ; <== 0x67BB54
004151D7 |. 83FF 02    |CMP EDI,2
004151DA |. 7D 32      |JGE SHORT dialogbl.0041520E
004151DC |. 8945 10    |MOV [ARG.3],EAX
004151DF |. 51       |PUSH ECX
004151E0 |. 8D45 10    |LEA EAX,[ARG.3]
004151E3 |. 68 08BE6C00 |PUSH dialogbl.006CBE08 ; ASCII "%08lX-"
004151E8 |. 50       |PUSH EAX

```

```

004151E9 |. C645 FC 03  |MOV BYTE PTR SS:[EBP-4],3
004151ED |. E8 9E9C0B00 |CALL dialogbl.004CEE90
004151F2 |. 8B45 10    |MOV EAX,[ARG.3]
004151F5 |. 83C4 0C    |ADD ESP,0C
004151F8 |. 8B48 F8    |MOV ECX,DWORD PTR DS:[EAX-8]
004151FB |. 50        |PUSH EAX           ; /Arg2
004151FC |. 51        |PUSH ECX          ; |Arg1
004151FD |. 8D4D 0C    |LEA ECX,[ARG.2]      ; |
00415200 |. E8 0B8E0B00 |CALL dialogbl.004CE010      ; \dialogbl.004CE010
00415205 |. C645 FC 02  |MOV BYTE PTR SS:[EBP-4],2
00415209 |. 8D4D 10    |LEA ECX,[ARG.3]
0041520C |. EB 30      |JMP SHORT dialogbl.0041523E
0041520E |> 8945 10    |MOV [ARG.3],EAX
00415211 |. 51        |PUSH ECX
00415212 |. 8D45 10    |LEA EAX,[ARG.3]
00415215 |. 68 00BE6C00 |PUSH dialogbl.006CBE00      ; ASCII "%08lX"
0041521A |. 50        |PUSH EAX
0041521B |. C645 FC 04  |MOV BYTE PTR SS:[EBP-4],4
0041521F |. E8 6C9C0B00 |CALL dialogbl.004CEE90
00415224 |. 8B45 10    |MOV EAX,[ARG.3]
00415227 |. 83C4 0C    |ADD ESP,0C
0041522A |. 8B48 F8    |MOV ECX,DWORD PTR DS:[EAX-8]
0041522D |. 50        |PUSH EAX           ; /Arg2
0041522E |. 51        |PUSH ECX          ; |Arg1
0041522F |. 8D4D 0C    |LEA ECX,[ARG.2]      ; |
00415232 |. E8 D98D0B00 |CALL dialogbl.004CE010      ; \dialogbl.004CE010
00415237 |. C645 FC 02  |MOV BYTE PTR SS:[EBP-4],2
0041523B |. 8D4D 10    |LEA ECX,[ARG.3]
0041523E |> E8 08BEFEFF |CALL dialogbl.0040104B
00415243 |. 47        |INC EDI
00415244 |. 83C6 04    |ADD ESI,4
00415247 |. 83FF 03    |CMP EDI,3
0041524A |.^ 0F8C 6FFFFFFF |JL dialogbl.004151BF

```

===== TraceInto III =====

```

004152CC |. 8365 FC 00  AND [LOCAL.1],0
004152D0 |. 8D45 0C    LEA EAX,[ARG.2]
004152D3 |. 50        PUSH EAX
004152D4 |. 8BCE      MOV ECX,ESI
004152D6 |. E8 559E0B00 CALL dialogbl.004CF130      ; <== Fake Serial
004152DB |. 8BF0      MOV ESI,EAX
004152DD |. 8D45 08    LEA EAX,[ARG.1]
004152E0 |. 8D4D 10    LEA ECX,[ARG.3]
004152E3 |. 50        PUSH EAX
004152E4 |. C645 FC 01  MOV BYTE PTR SS:[EBP-4],1
004152E8 |. E8 439E0B00 CALL dialogbl.004CF130      ; <== Real Serial
004152ED |. 56        PUSH ESI
004152EE |. 50        PUSH EAX
004152EF |. C645 FC 02  MOV BYTE PTR SS:[EBP-4],2
004152F3 |. E8 D782FFFF CALL dialogbl.0040D5CF      ; <== Compare

```

===== TraceInto II =====**===== TraceInto I =====**

- Tương tự như thế, quá trình tạo chuỗi Serial thực thứ hai được đưa trên một chuỗi mặc định khác : ***"Anthemion Software DialogBlocks and HelpBlocks"*** :
00413E93 . E8 C8140000 CALL dialogbl.00415360 ; <== Real Serial II
- Như vậy quá trình mã hoá của SoftWare này chỉ là quá trình tạo chuỗi MD5Hash dựa trên sự kết hợp giữa U nhập và một chuỗi mặc định . Với hai chuỗi mặc định của chương trình, ta có hai chuỗi Serial thực tương ứng .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 889B3D8D-C5F77D80-05A17F78

III – KeyGen :

- /Section 1/- Thực hiện CharLower(User), và kết hợp với một trong hai chuỗi mặc định .
- /Section 2/- Tính ***MD5hash*** .
- /Section 2/- Kết hợp 3 đoạn đầu tiên của chuỗi này theo định dạng ***"%08X-%08X-%08X"***

IV – End of Tut :

- Finished – *August 09, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://diskarcher.mastak.com
Production :	DiskArcher Software Co.
SoftWare :	DiskArcher Backup Utility 2.01
Copyright by :	Copyright © 2001-2003 DiskArcher Software Co. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0 [Debug]
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

DiskArcher Backup Utility 2.01

DiskArcher can make backup copies of files from local and network drives, stores backup copies on local and network drives. It recognizes the type of the media (removable or not), so, for example, you can spread your copies on several diskettes and several network drives. DiskArcher analyzes if some file changes, then updates its copy in the archive, makes as many copies of any selected file as you want. So, you can get the full history of file modifications and retrieve any of its versions.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng ***Microsoft Visual C++ 6.0 [Debug]***
- Trong quá trình tìm kiếm chuỗi thông báo, ta gặp được các dòng sau :

```
00417C09 |. 68 68A04200 PUSH DiskArc.0042A068 ; /s2 = "DA10-"
00417C7B |. 68 60A04200 PUSH DiskArc.0042A060 ; ASCII "-YREJ"
```

- Ta nghĩ đến dạng của Serial : "DA10-xxxxx-YREJ"

- Dò ngược lên trên, ta chọn đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

```
00417BD2 |. E8 F9100000 CALL <JMP.&MSVCRT._EH_prolog> ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Quá trình mã hoá này rất đơn giản :

```
00417BE9 |. 8378 F8 0F CMP DWORD PTR DS:[EAX-8],0F ; <== S must be 15 charts
00417BED |. 0F85 AA000000 JNZ DiskArc.00417C9D
00417BF3 |. 8D45 EC LEA EAX,[LOCAL.5]
00417BF6 |. 6A 05 PUSH 5
00417BF8 |. 50 PUSH EAX
00417BF9 |. 8D4D 0C LEA ECX,[ARG.2]
00417BFC |. E8 8B0C0000 CALL <JMP.&MFC42.#4129>
00417C01 |. 8B00 MOV EAX,DWORD PTR DS:[EAX] ; <== 5 first charts of Serial
00417C03 |. 8B35 ECD64100 MOV ESI,DWORD PTR DS:[<&MSVCRT._mbscmp>] ;
MSVCRT._mbscmp
00417C09 |. 68 68A04200 PUSH DiskArc.0042A068 ; /s2 = "DA10-"
00417C0E |. 50 PUSH EAX ; |s1
00417C0F |. FFD6 CALL ESI ; \_mbscmp
00417C11 |. 8BD8 MOV EBX,EAX
..... cutting .....
00417C2B |. 33FF XOR EDI,EDI
00417C2D |. 33C0 XOR EAX,EAX
00417C2F |. 8B51 F8 MOV EDX,DWORD PTR DS:[ECX-8]
00417C32 |. 85D2 TEST EDX,EDX
00417C34 |. 7E 0B JLE SHORT DiskArc.00417C41
00417C36 |> 0FBE1C08 /MOVSX EBX,BYTE PTR DS:[EAX+ECX] ; <== U[i]
00417C3A |. 03FB |ADD EDI,EBX ; <== CumV = CumV + U[i]
00417C3C |. 40 |INC EAX ; <== i++
00417C3D |. 3BC2 |CMP EAX,EDX ; <== while ( i < LenUser )
00417C3F |.^ 7C F5 \JL SHORT DiskArc.00417C36 ; <== Continue Loop
00417C41 |> 6A 05 PUSH 5
00417C43 |. 8D45 EC LEA EAX,[LOCAL.5]
00417C46 |. 6A 05 PUSH 5
00417C48 |. 50 PUSH EAX
00417C49 |. 8D4D 0C LEA ECX,[ARG.2]
00417C4C |. E8 290C0000 CALL <JMP.&MFC42.#4278>
00417C51 |. FF30 PUSH DWORD PTR DS:[EAX] ; /s
00417C53 |. FF15 CCD64100 CALL DWORD PTR DS:[<&MSVCRT.atol>] ; \atol
00417C59 |. 59 POP ECX
00417C5A |. 8BD8 MOV EBX,EAX
00417C5C |. 8D4D EC LEA ECX,[LOCAL.5]
00417C5F |. E8 C8080000 CALL <JMP.&MFC42.#800>
00417C64 |. 3BDF CMP EBX,EDI
00417C66 |. 5F POP EDI
00417C67 |. 75 34 JNZ SHORT DiskArc.00417C9D
```

```

00417C69 |. 6A 05      PUSH 5
00417C6B |. 8D45 EC    LEA EAX,[LOCAL.5]
00417C6E |. 6A 0A      PUSH 0A
00417C70 |. 50        PUSH EAX
00417C71 |. 8D4D 0C    LEA ECX,[ARG.2]
00417C74 |. E8 010C0000 CALL <JMP.&MFC42.#4278>
00417C79 |. 8B00        MOV EAX,DWORD PTR DS:[EAX]          ; <== 5 last charts of Serial
00417C7B |. 68 60A04200 PUSH DiskArc.0042A060              ; ASCII "-YREJ"
00417C80 |. 50        PUSH EAX
00417C81 |. FFD6        CALL ESI                           ; MSVCRT._mbscmp

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : DA10-01255-YREJ

III – KeyGen :

/Section 0 /- N/A

IV – End of Tut :

- Finished – *August 14, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.somewareonthe.net
Production :	SomeWare.
SoftWare :	Dpeg Suite 6.13
Copyright by :	Copyright © '97, '98, '99, '00, 03, 2004 SomeWare. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual Basic 5.0 / 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial

Dpeg Suite 6.13

This is not your father's dupe checker. Not your brother's, not your sister's, and certainly not your mother's! d'peg! is the most sophisticated dupe snarfer there ever was and it's back - better, stronger, faster.

Got a few hundred photos, graphic images, or mp3's? Thousands? Hundreds of thousands? Downloaded, traded, rip'd, saved, shared, it doesn't matter how you got 'em - you've got duplicates! With d'peg!, you have an intuitive & easy way to find, examine, and get rid of those extra files just wasting space & causing confusion.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng *Microsoft Visual Basic 5.0 / 6.0*.

- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Incorrect Registration Code...**" Ta tìm được 005DD04A . 68 B0B04400 PUSH dpege.0044B0B0 ; UNICODE "Incorrect Registration Code..."

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
005DCC60 . FF51 04 CALL DWORD PTR DS:[ECX+4] : <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Trace xuống một chút ta đến :

005DCD64 . E8 179CF1FF CALL dpege.004F6980 ; <== Trace Into

- Dùng F7 Trace Into và trace tiếp một đoạn, ta đến quá trình kiểm tra chuỗi U nhập. Chuỗi U nhập được chuyển thẳng dạng UpperCase, và tất cả các ký tự trong chuỗi này phải là các ký tự nằm trong khoảng từ (A-Z), Và đồng thời trong quá trình này chương trình cũng tiến hành cắt các khoảng trống nếu có giữa các ký tự. Và mặc nhiên quy định rằng, các ký tự đứng sau khoảng trắng sẽ được nối với nhau và nối với ký tự đầu tiên của chuỗi U nhập . Tạp thành đoạn đầu tiên của chuỗi Serial :

```
004F7236 . BF 01000000 MOV EDI,1
004F723B > 66:3BBD 10FDF>CMP DI,WORD PTR SS:[EBP-2F0]
004F7242 . 0F8F 66010000 JG dpege.004F73AE
004F7248 . B8 02000000 MOV EAX,2
004F724D . 8D4D DC LEA ECX,WORD PTR SS:[EBP-24]
004F7250 . 8945 B4 MOV WORD PTR SS:[EBP-4C],EAX
004F7253 . 8945 AC MOV WORD PTR SS:[EBP-54],EAX
004F7256 . 0FBFC7 MOVSX EAX,DI
004F7259 . 8D55 AC LEA EDX,WORD PTR SS:[EBP-54]
004F725C . 898D D4FDFFFF MOV WORD PTR SS:[EBP-22C],ECX
004F7262 . 52 PUSH EDX
004F7263 . 8D8D CCFDFFFF LEA ECX,WORD PTR SS:[EBP-234]
004F7269 . 50 PUSH EAX
004F726A . 8D55 9C LEA EDX,WORD PTR SS:[EBP-64]
004F726D . 51 PUSH ECX
004F726E . 52 PUSH EDX
004F726F . C785 B4FDFFFF>MOV WORD PTR SS:[EBP-24C],dpege.00446A54 ; UNICODE " A
B C D E F G H I J K L M N O P Q R S T U V W X Y Z"
004F7279 . C785 ACFDFFFF>MOV WORD PTR SS:[EBP-254],8
004F7283 . C785 CCFDFFFF>MOV WORD PTR SS:[EBP-234],4008
004F728D . FFD6 CALL ESI
004F728F . 8D85 ACFDFFFF LEA EAX,WORD PTR SS:[EBP-254]
004F7295 . 6A 01 PUSH 1
004F7297 . 8D4D 9C LEA ECX,WORD PTR SS:[EBP-64]
004F729A . 50 PUSH EAX
004F729B . 51 PUSH ECX
004F729C . 8D55 8C LEA EDX,WORD PTR SS:[EBP-74]
004F729F . 6A 00 PUSH 0
004F72A1 . 52 PUSH EDX
004F72A2 . C785 A4FDFFFF>MOV WORD PTR SS:[EBP-25C],0
004F72AC . C785 9CFDFFFF>MOV WORD PTR SS:[EBP-264],8002
004F72B6 . FF15 2C124000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaInStr>;
MSVBVM60.__vbaInStrVar
004F72BC . 50 PUSH EAX
004F72BD . 8D85 9CFDFFFF LEA EAX,WORD PTR SS:[EBP-264]
```

```

004F72C3 . 50      PUSH EAX
004F72C4 . FF15 04104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarTs>;
MSVBVM60.__vbaVarTstGt
004F72CA . 8D4D 8C    LEA ECX,DWORD PTR SS:[EBP-74]
004F72CD . 8985 18FDFFFF MOV DWORD PTR SS:[EBP-2E8],EAX
004F72D3 . 8D55 9C    LEA EDX,DWORD PTR SS:[EBP-64]
004F72D6 . 51      PUSH ECX
004F72D7 . 8D45 AC    LEA EAX,DWORD PTR SS:[EBP-54]
004F72DA . 52      PUSH EDX
004F72DB . 50      PUSH EAX
004F72DC . 6A 03    PUSH 3
004F72DE . FFD3    CALL EBX
004F72E0 . 83C4 10    ADD ESP,10
004F72E3 . 66:83BD 18FDF>CMP WORD PTR SS:[EBP-2E8],0
004F72EB . 0F84 A8000000 JE dpeg.004F7399
004F72F1 . 8B4D C4    MOV ECX,DWORD PTR SS:[EBP-3C]
004F72F4 . 8D55 DC    LEA EDX,DWORD PTR SS:[EBP-24]
004F72F7 . 898D B4FDFFFF MOV DWORD PTR SS:[EBP-24C],ECX
004F72FD . 66:8BCF    MOV CX,DI
004F7300 . 66:83C1 01    ADD CX,1
004F7304 . 8995 D4FDFFFF MOV DWORD PTR SS:[EBP-22C],EDX
004F730A . 0F80 A5080000 JO dpeg.004F7BB5
004F7310 . 8D45 AC    LEA EAX,DWORD PTR SS:[EBP-54]
004F7313 . C785 ACFDFFFF>MOV DWORD PTR SS:[EBP-254],8
004F731D . 0FBFD1    MOVSX EDX,CX
004F7320 . 50      PUSH EAX
004F7321 . 8D85 CCFDFFFF LEA EAX,DWORD PTR SS:[EBP-234]
004F7327 . 52      PUSH EDX
004F7328 . 8D4D 9C    LEA ECX,DWORD PTR SS:[EBP-64]
004F732B . 50      PUSH EAX
004F732C . 51      PUSH ECX
004F732D . C745 B4 01000>MOV DWORD PTR SS:[EBP-4C],1
004F7334 . C745 AC 02000>MOV DWORD PTR SS:[EBP-54],2
004F733B . C785 CCFDFFFF>MOV DWORD PTR SS:[EBP-234],4008
004F7345 . FFD6    CALL ESI
004F7347 . 8D55 9C    LEA EDX,DWORD PTR SS:[EBP-64]
004F734A . 8D45 8C    LEA EAX,DWORD PTR SS:[EBP-74]
004F734D . 52      PUSH EDX
004F734E . 50      PUSH EAX
004F734F . FF15 10114000 CALL DWORD PTR DS:[<&MSVBVM60.#520>] ;
MSVBVM60 rtcTrimVar
004F7355 . 8D8D ACFDFFFF LEA ECX,DWORD PTR SS:[EBP-254]
004F735B . 8D55 8C    LEA EDX,DWORD PTR SS:[EBP-74]
004F735E . 51      PUSH ECX
004F735F . 8D85 7CFFFFFF LEA EAX,DWORD PTR SS:[EBP-84]
004F7365 . 52      PUSH EDX
004F7366 . 50      PUSH EAX
004F7367 . FF15 3C124000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarCa>;
MSVBVM60.__vbaVarCat
004F736D . 50      PUSH EAX
004F736E . FF15 2C104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrVa>;
MSVBVM60.__vbaStrVarMove

```

```

004F7374 . 8BD0      MOV EDX,EAX
004F7376 . 8D4D C4    LEA ECX,DWORD PTR SS:[EBP-3C]
004F7379 . FF15 14134000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrMo>];
MSVBVM60.__vbaStrMove
004F737F . 8D8D 7CFFFFFF LEA ECX,DWORD PTR SS:[EBP-84]
004F7385 . 8D55 8C      LEA EDX,DWORD PTR SS:[EBP-74]
004F7388 . 51          PUSH ECX
004F7389 . 8D45 9C      LEA EAX,DWORD PTR SS:[EBP-64]
004F738C . 52          PUSH EDX
004F738D . 8D4D AC      LEA ECX,DWORD PTR SS:[EBP-54]
004F7390 . 50          PUSH EAX
004F7391 . 51          PUSH ECX
004F7392 . 6A 04        PUSH 4
004F7394 . FFD3        CALL EBX
004F7396 . 83C4 14      ADD ESP,14
004F7399 > B8 01000000  MOV EAX,1
004F739E . 66:03C7      ADD AX,DI
004F73A1 . 0F80 0E080000 JO dpeg.004F7BB5
004F73A7 . 8BF8        MOV EDI,EAX
004F73A9 . ^E9 8DFEFFFF JMP dpeg.004F723B

```

- Trace tiếp ta đến quá trình tạo chuỗi Serial . Quá trình này được hiểu như sau : từng ký tự của chuỗi U sau khi xử lý sẽ được so sánh với các ký tự của chuỗi mặc định, nếu ký tự nào của chuỗi U bằng với chuỗi mặc định thì tiến hành xử lý. Giá trị được xử lý sẽ được chuyển sang dạng chuỗi theo định dạng “%i”, và được nối tiếp với đoạn đầu tiên của chuỗi được tạo thành ở trên :

```

004F7782 >/66:3B85 00FDF>CMP AX,WORD PTR SS:[EBP-300]
004F7789 . |OF8F D5010000 JG dpeg.004F7964
004F778F . |0FBFF8    MOVSX EDI,AX
004F7792 . |8D4D DC      LEA ECX,DWORD PTR SS:[EBP-24]
004F7795 . |8D55 AC      LEA EDX,DWORD PTR SS:[EBP-54]
004F7798 . |898D D4FDFFFF MOV DWORD PTR SS:[EBP-22C],ECX
004F779E . |52          PUSH EDX
004F779F . |8D85 CCFDFFFF LEA EAX,DWORD PTR SS:[EBP-234]
004F77A5 . |57          PUSH EDI
004F77A6 . |8D4D 9C      LEA ECX,DWORD PTR SS:[EBP-64]
004F77A9 . |50          PUSH EAX
004F77AA . |51          PUSH ECX
004F77AB . |C785 B4FDFFFF>MOV DWORD PTR SS:[EBP-24C],dpeg.00446AE0 ; UNICODE
"AEIOUY"
004F77B5 . |C785 ACFDFFFF>MOV DWORD PTR SS:[EBP-254],8
004F77BF . |C745 B4 01000>MOV DWORD PTR SS:[EBP-4C],1
004F77C6 . |C745 AC 02000>MOV DWORD PTR SS:[EBP-54],2
004F77CD . |C785 CCFDFFFF>MOV DWORD PTR SS:[EBP-234],4008
004F77D7 . |FFD6        CALL ESI
004F77D9 . |8D95 ACFDFFFF LEA EDX,DWORD PTR SS:[EBP-254]
004F77DF . |6A 01        PUSH 1
004F77E1 . |8D45 9C      LEA EAX,DWORD PTR SS:[EBP-64]
004F77E4 . |52          PUSH EDX
004F77E5 . |50          PUSH EAX
004F77E6 . |8D4D 8C      LEA ECX,DWORD PTR SS:[EBP-74]
004F77E9 . |6A 00        PUSH 0
004F77EB . |51          PUSH ECX

```

```

004F77EC . |C785 A4FDFFFF>MOV DWORD PTR SS:[EBP-25C],0
004F77F6 . |C785 9CFDFFFF>MOV DWORD PTR SS:[EBP-264],8002
004F7800 . |FF15 2C124000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaInStr>];
MSVBVM60.__vbaInStrVar
004F7806 . |8D95 9CFDFFFF LEA EDX,DWORD PTR SS:[EBP-264]
004F780C . |50      PUSH EAX
004F780D . |52      PUSH EDX
004F780E . |FF15 04104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarTs>];
MSVBVM60.__vbaVarTstGt
004F7814 . |8985 18FDFFFF MOV DWORD PTR SS:[EBP-2E8],EAX
004F781A . |8D45 8C      LEA EAX,DWORD PTR SS:[EBP-74]
004F781D . |8D4D 9C      LEA ECX,DWORD PTR SS:[EBP-64]
004F7820 . |50      PUSH EAX
004F7821 . |8D55 AC      LEA EDX,DWORD PTR SS:[EBP-54]
004F7824 . |51      PUSH ECX
004F7825 . |52      PUSH EDX
004F7826 . |6A 03      PUSH 3
004F7828 . |FFD3      CALL EBX
004F782A . |83C4 10      ADD ESP,10
004F782D . |66:83BD 18FDF>CMP WORD PTR SS:[EBP-2E8],0          ; <== if U[i] = dStr[i]
004F7835 . |0F84 12010000 JE dpeg.004F794D                      ; <== go to Encrypt
004F783B . |8D45 DC      LEA EAX,DWORD PTR SS:[EBP-24]
004F783E . |8D4D AC      LEA ECX,DWORD PTR SS:[EBP-54]
004F7841 . |8985 D4FDFFFF MOV DWORD PTR SS:[EBP-22C],EAX
004F7847 . |51      PUSH ECX
004F7848 . |8D95 CCFDFFFF LEA EDX,DWORD PTR SS:[EBP-234]
004F784E . |57      PUSH EDI
004F784F . |8D45 9C      LEA EAX,DWORD PTR SS:[EBP-64]
004F7852 . |52      PUSH EDX
004F7853 . |50      PUSH EAX
004F7854 . |C745 B4 01000>MOV DWORD PTR SS:[EBP-4C],1
004F785B . |C745 AC 02000>MOV DWORD PTR SS:[EBP-54],2
004F7862 . |C785 CCFDFFFF>MOV DWORD PTR SS:[EBP-234],4008
004F786C . |FFD6      CALL ESI
004F786E . |8B4D C4      MOV ECX,DWORD PTR SS:[EBP-3C]
004F7871 . |8D55 9C      LEA EDX,DWORD PTR SS:[EBP-64]
004F7874 . |51      PUSH ECX
004F7875 . |8D45 C0      LEA EAX,DWORD PTR SS:[EBP-40]
004F7878 . |52      PUSH EDX
004F7879 . |50      PUSH EAX
004F787A . |FF15 30124000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrVa>];
MSVBVM60.__vbaStrVarVal
004F7880 . |50      PUSH EAX
004F7881 . |FF15 58104000 CALL DWORD PTR DS:[<&MSVBVM60.#516>]    ;
MSVBVM60.rtcAnsiValueBstr
004F7887 . |0FBFC8      MOVSX ECX,AX          ; <== U[i]
004F788A . |898D F4FCFFFF MOV DWORD PTR SS:[EBP-30C],ECX
004F7890 . |DB85 F4FCFFFF FILD DWORD PTR SS:[EBP-30C]      ; <== Convert to F.P.Number
004F7896 . |DD9D ECFCFFFF FSTP QWORD PTR SS:[EBP-314]
004F789C . |DD85 ECFCFFFF FLD QWORD PTR SS:[EBP-314]
004F78A2 . |833D 00405E00>CMP DWORD PTR DS:[5E4000],0
004F78A9 . |75 08      JNZ SHORT dpeg.004F78B3

```

```

004F78AB . |DC35 30514000 FDIV QWORD PTR DS:[405130] ; <== Value = U[i] / 4
004F78B1 . |EB 11      JMP SHORT dpeg.004F78C4
004F78B3 > |FF35 34514000 PUSH DWORD PTR DS:[405134]
004F78B9 . |FF35 30514000 PUSH DWORD PTR DS:[405130]
004F78BF . |E8 4021F1FF CALL <JMP.&MSVBVM60._adj_fdiv_m64>
004F78C4 > |DFE0      FSTSW AX
004F78C6 . |A8 0D      TEST AL,0D
004F78C8 . |OF85 E2020000 JNZ dpeg.004F7BB0
004F78CE . |FF15 48134000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFPInt>; MSVBVM60.__vbaFPInt
004F78D4 . |66:8B55 E8  MOV DX,WORD PTR SS:[EBP-18]
004F78D8 . |66:6BD2 07 IMUL DX,DX,7 ; <== Temp = (i+1)*7
004F78DC . |0F80 D3020000 JO dpeg.004F7BB5
004F78E2 . |0FBFC2    MOVSX EAX,DX
004F78E5 . |8985 E8FCFFFF MOV DWORD PTR SS:[EBP-318],EAX
004F78EB . |83EC 08    SUB ESP,8
004F78EE . |DB85 E8FCFFFF FILD DWORD PTR SS:[EBP-318]
004F78F4 . |DD9D E0FCFFFF FSTP QWORD PTR SS:[EBP-320]
004F78FA . |DC85 E0FCFFFF FADD QWORD PTR SS:[EBP-320] ; <== Value = Value + Temp
004F7900 . |DFE0      FSTSW AX
004F7902 . |A8 0D      TEST AL,0D
004F7904 . |OF85 A6020000 JNZ dpeg.004F7BB0
004F790A . |DD1C24    FSTP QWORD PTR SS:[ESP]
004F790D . |FF15 BC114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrR8>; MSVBVM60.__vbaStrR8
004F7913 . |8B3D 14134000 MOV EDI,DWORD PTR DS:[<&MSVBVM60.__vbaSt>; MSVBVM60.__vbaStrMove
004F7919 . |8BD0      MOV EDX,EAX
004F791B . |8D4D BC    LEA ECX,DWORD PTR SS:[EBP-44]
004F791E . |FFD7      CALL EDI ; <&MSVBVM60.__vbaStrMove>
004F7920 . |50        PUSH EAX
004F7921 . |FF15 74104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrCa>; MSVBVM60.__vbaStrCat
004F7927 . |8BD0      MOV EDX,EAX
004F7929 . |8D4D C4    LEA ECX,DWORD PTR SS:[EBP-3C]
004F792C . |FFD7      CALL EDI
004F792E . |8D4D BC    LEA ECX,DWORD PTR SS:[EBP-44]
004F7931 . |8D55 C0    LEA EDX,DWORD PTR SS:[EBP-40]
004F7934 . |51        PUSH ECX
004F7935 . |52        PUSH EDX
004F7936 . |6A 02      PUSH 2
004F7938 . |FF15 94124000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeS>; MSVBVM60.__vbaFreeStrList
004F793E . |8D45 9C    LEA EAX,DWORD PTR SS:[EBP-64]
004F7941 . |8D4D AC    LEA ECX,DWORD PTR SS:[EBP-54]
004F7944 . |50        PUSH EAX
004F7945 . |51        PUSH ECX
004F7946 . |6A 02      PUSH 2
004F7948 . |FFD3      CALL EBX
004F794A . |83C4 18    ADD ESP,18
004F794D > |B8 01000000 MOV EAX,1
004F7952 . |66:0345 E8  ADD AX,WORD PTR SS:[EBP-18]

```

```
004F7956 . |0F80 59020000 JO dpeg.004F7BB5
004F795C . |8945 E8      MOV DWORD PTR SS:[EBP-18],EAX
004F795F .^|E9 1EFEEEEEE JMP dpeg.004F7782
```

- User và Serial được lưu vào FILE : **dpeg.ini**

/*/*/* - SERIAL tương ứng :

User : REA cRaCkErTeAm

Serial : RC31375880101107

III – KeyGen :

- /Section 1/- CharUpper(reaName). Liên kết các ký tự sau không trắng với ký tự đầu tiên.
- /Section 2/- Cắt các khoảng trắng .
- /Section 3/- Tính giá trị theo công thức : $Value = ((reaTrimName[i] \& 0xFF)/0x4) + (i+1) * 7;$

IV – End of Tut :

- Finished – **August 10, 2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.dumeter.com
Production :	Hagel Technologies.
SoftWare :	DU Meter 3.07 Build 192
Copyright by :	Copyright © 1997-2004 Hagel Technologies. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

DU Meter 3.07 Build 192

DU Meter is an award winning utility from Hagel Technologies that provides an accurate account of the data which is flowing through your computer's network connection at any given moment. This readout is presented in both numerical and graphical format, in real time. DU Meter includes extensive logging facility, flexible events system, and more. It supports Windows 95/98/NT4/2000 and XP! DU Meter works with virtually all types of network connections: phone modems, DSL, cable modem, LAN, satellite, and more.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được "NAG". Tuy nhiên, ta không thể tìm thấy chuỗi này trong quá trình tìm kiếm. Nhưng ta nhận thấy chương trình có khoảng thời gian dừng trước khi xuất hiện NAG. Ta truy đến hàm **kernel32.Sleep**. Ta tìm được hàm này tại địa chỉ :

```
004937D3 |. E8 0CBBF7FF |CALL <JMP.&kernel32.Sleep> ; \Sleep
```

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

```
004937C7 |. E8 F44FFECC CALL DUMeter.004787C0 ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

```
00493848 |. E8 1764FCFF CALL DUMeter.00459C64 ; <== LenS
0049384D |. 837D E4 00 CMP [LOCAL.7],0 ; <== Must be input
00493851 |. 74 14 JE SHORT DUMeter.00493867
00493853 |. 8D55 E0 LEA EDX,[LOCAL.8]
00493856 |. 8B83 74030000 MOV EAX,DWORD PTR DS:[EBX+374]
0049385C |. E8 0364FCFF CALL DUMeter.00459C64 ; <== LenU
00493861 |. 837D E0 00 CMP [LOCAL.8],0 ; <== Must be input
00493865 |. 75 33 JNZ SHORT DUMeter.0049389A
```

- Trace tiếp ta đến :

```
0049399D |. E8 9AD1FFFF CALL DUMeter.00490B3C ; <== Trace Into
----- Trace Into -----
00490B78 |. FF16 CALL DWORD PTR DS:[ESI] ; <== Trace Into
----- Trace Into -----
0049DD11 . 8BF0 MOV ESI,EAX ; <== U
0049DD13 . 8BD6 MOV EDX,ESI ; <== U
0049DD15 . B8 D0DD4900 MOV EAX,DUMeter.0049DDD0 ; ASCII "D3"
0049DD1A . 59 POP ECX ; <== Fake Serial
0049DD1B . E8 16FEFEFF CALL DUMeter.0048DB36 ; <== Encrypt & Compare
----- Trace Into -----
----- Trace Into -----
```

- Tiếp tục Trace Into ta đến quá trình mã hoá chính :

```
0048DB6A |. E8 2DFDFFFF CALL DUMeter.0048D89C ; <== LenS
0048DB6F |. 83F8 18 CMP EAX,18 ; <== LenS must be 24 charts
0048DB72 |. 74 07 JE SHORT DUMeter.0048DB7B ; <== Continue check
0048DB74 |. 33C0 XOR EAX,EAX
0048DB76 |. E9 C5000000 JMP DUMeter.0048DC40
0048DB7B |> 803E 61 CMP BYTE PTR DS:[ESI],61
0048DB7E |. 7C 08 JL SHORT DUMeter.0048DB88
0048DB80 |. 803E 7A CMP BYTE PTR DS:[ESI],7A
0048DB83 |. 7F 03 JG SHORT DUMeter.0048DB88
0048DB85 |. 8006 E0 ADD BYTE PTR DS:[ESI],0E0
0048DB88 |> 803E 20 CMP BYTE PTR DS:[ESI],20
0048DB8B |. 74 15 JE SHORT DUMeter.0048DBA2
0048DB8D |. 8A16 MOV DL,BYTE PTR DS:[ESI] ; <== U[0]
0048DB8F |. 3A13 CMP DL,BYTE PTR DS:[EBX] ; <== if ( U[0] == "D" )
0048DB91 |. 75 08 JNZ SHORT DUMeter.0048DB9B ; <== Continue check
0048DB93 |. 8A4E 01 MOV CL,BYTE PTR DS:[ESI+1] ; <== U[1]
0048DB96 |. 3A4B 01 CMP CL,BYTE PTR DS:[EBX+1] ; <== if ( U[1] == "3" )
0048DB99 |. 7E 07 JLE SHORT DUMeter.0048DBA2 ; <== Continue check
```

- Trace tiếp ta đến quá trình phân cách chuỗi Serial nhập thành 4 đoạn .

Đoạn thứ nhất gồm 3 ký tự	U[3][4][5]
Đoạn thứ hai gồm 8 ký tự	U[7][8][9][10][11][12][13][14]
Đoạn thứ 3 cũng gồm 8 ký tự	U[16][17][18][19][20][21][22][23]
Đoạn cuối cùng gồm 16 ký tự	U[0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15]

```

0048DBA2 |> \8D53 03    LEA EDX,DWORD PTR DS:[EBX+3]
0048DBA5 |. 8D45 F4    LEA EAX,[LOCAL.3]
0048DBA8 |. B9 03000000 MOV ECX,3                      ; <== 3 charts
0048DBAD |. E8 14FDFFFF CALL DUMeter.0048D8C6          ; <== SecI
0048DBB2 |. C645 F7 00 MOV BYTE PTR SS:[EBP-9],0
0048DBB6 |. 8D53 07    LEA EDX,DWORD PTR DS:[EBX+7]
0048DBB9 |. 8D45 E8    LEA EAX,[LOCAL.6]
0048DBBC |. B9 08000000 MOV ECX,8                      ; <== 8 charts
0048DBC1 |. E8 00FDFFFF CALL DUMeter.0048D8C6          ; <== SecII
0048DBC6 |. C645 F0 00 MOV BYTE PTR SS:[EBP-10],0
0048DBCA |. 8D53 10    LEA EDX,DWORD PTR DS:[EBX+10]
0048DBCD |. 8D45 DC    LEA EAX,[LOCAL.9]
0048DBD0 |. B9 08000000 MOV ECX,8                      ; <== 8 charts
0048DBD5 |. E8 ECFCFFFF CALL DUMeter.0048D8C6          ; <== SecIII
0048DBDA |. C645 E4 00 MOV BYTE PTR SS:[EBP-1C],0
0048DBDE |. 8D45 C8    LEA EAX,[LOCAL.14]
0048DBE1 |. B9 10000000 MOV ECX,10                     ; <== 16 charts
0048DBE6 |. 8BD3      MOV EDX,EBX
0048DBE8 |. E8 D9FCFFFF CALL DUMeter.0048D8C6          ; <== SecIV

```

- Đầu tiên chương trình sẽ mã hoá chuỗi U nhập . Tuy nhiên, trước khi tiến hành quá trình mã hoá chương trình sẽ chuyển đổi các ký tự của chuỗi U nhập sang dạng UpperCase và loại bỏ các ký tự đặc biệt :

0048DBFF |. E8 81FDFFFF CALL DUMeter.0048D985 ; <== Encrypt U : ValueU

----- Encrypt U -----

```

0048D98C |> /C1E2 04    /SHL EDX,4                  ; <== ValueU = ValueU * 0x10
0048D98F |. |0FBE08    |MOVSX ECX,BYTE PTR DS:[EAX] ; <== U[i]
0048D992 |. |03D1      |ADD EDX,ECX                ; <== ValueU = ValueU + U[i]
0048D994 |. |8BCA      |MOV ECX,EDX                ; <== Temp = ValueU
0048D996 |. |81E1 000000F0 |AND ECX,F0000000          ; <== Temp = Temp and 0xF0000000
0048D99C |. |85C9      |TEST ECX,ECX               ; <== if ( ECX != 0x0 )
0048D99E |. |74 0B      |JE SHORT DUMeter.0048D9AB ; <== then
0048D9A0 |. |C1E9 18    |SHR ECX,18                 ; <== Temp = Temp / 0x1000000
0048D9A3 |. |33D1      |XOR EDX,ECX                ; <== Value = Value xor Temp
0048D9A5 |. |81E2 FFFFFFFF0F |AND EDX,0FFFFFFF          ; <== ValueU = ValueU & 0xFFFFFFFF
0048D9AB |> |40        |INC EAX                   ; <== i++
0048D9AC |> |8038 00    CMP BYTE PTR DS:[EAX],0          ; <== while ( i < LenU )
0048D9AF |.^|75 DB     |JNZ SHORT DUMeter.0048D98C ; <== continue Loop
----- Encrypt U -----

```

- Kế đó là mã hoá SecI . Đoạn mã hoá này cho ta biết được rằng các ký tự của SecI phải là một trong các ký tự của chuỗi mặc định "**ABCDEFGHIJKLMNPQRSTUVWXYZ987654**" :

0048DC06 |. 8D45 F4 LEA EAX,[LOCAL.3] ; <== SecI
0048DC09 |. E8 01FEFFFF CALL DUMeter.0048DA0F ; <== Encrypt SecI : ValueI

----- Encrypt SecI -----

```

0048DA19 |. E8 7EFEEEEEE CALL DUMeter.0048D89C          ; <== LenSecI
0048DA1E |. 83F8 03    CMP EAX,3                  ; <== Must be 3 charts
0048DA21 |. 73 04      JNB SHORT DUMeter.0048DA27

```

```

0048DA23 |. 33C0      XOR EAX,EAX
0048DA25 |. EB 39      JMP SHORT DUMeter.0048DA60
0048DA27 |> BA 0C174F00 MOV EDX,DUMeter.004F170C      ; ASCII
"ABCDEFGHIJKLMNOPQRSTUVWXYZ987654"
0048DA2C |. 8A03      MOV AL,BYTE PTR DS:[EBX]          ; <== SecI[0]
0048DA2E |. E8 82FFFFFF CALL DUMeter.0048D9B5          ; <== Location in DefaultString
0048DA33 |. 8BF0      MOV ESI,EAX                      ; <== LocI
0048DA35 |. 4E        DEC ESI                          ; <== LocI-
0048DA36 |. BA 0C174F00 MOV EDX,DUMeter.004F170C      ; ASCII
"ABCDEFGHIJKLMNOPQRSTUVWXYZ987654"
0048DA3B |. 8A43 01   MOV AL,BYTE PTR DS:[EBX+1]         ; <== SecI[1]
0048DA3E |. E8 72FFFFFF CALL DUMeter.0048D9B5          ; <== Location in DefaultString
0048DA43 |. 8BF8      MOV EDI,EAX                      ; <== LocII
0048DA45 |. 4F        DEC EDI                          ; <== LocII-
0048DA46 |. BA 0C174F00 MOV EDX,DUMeter.004F170C      ; ASCII
"ABCDEFGHIJKLMNOPQRSTUVWXYZ987654"
0048DA4B |. 8A43 02   MOV AL,BYTE PTR DS:[EBX+2]         ; <== SecI[2]
0048DA4E |. E8 62FFFFFF CALL DUMeter.0048D9B5          ; <== Location in DefaultString
0048DA53 |. 48        DEC EAX                          ; <== LocIII-
0048DA54 |. C1E7 05   SHL EDI,5                        ; <== ValueI = LocII * 0x20
0048DA57 |. 0BF7      OR ESI,EDI                      ; <== Value = Value or LocI
0048DA59 |. C1E0 0A   SHL EAX,0A                      ; <== LocIII = LocIII * 0x400
0048DA5C |. 0BF0      OR ESIEAX                      ; <== Value = Value or LocIII
0048DA5E |. 8BC6      MOV EAX,ESI                      ; <== Value
----- Encrypt SecI -----

```

- Tiếp đến là quá trình chuyển đổi SecII từ chuỗi sang dạng HEX Value tương ứng (Ví dụ SecII : “123AB” được chuyển sang thành giá trị ValueII = 0x123AB) . Như vậy, các ký tự của SecII phải nằm trong khoảng các ký tự sau “123456789ABCDEF” :

```

0048DC10 |. 8D45 E8      LEA EAX,[LOCAL.6]           ; <== SecII
0048DC13 |. E8 7CFEFFFF CALL DUMeter.0048DA94       ; <== Convert to HEX value : ValueII

```

- Tiếp đó là quá trình kiểm tra đầu tiên :

```

0048DC1A |. 3BF3      CMP ESI,EBX                  ; <== if ( ValueU == ValueII )
0048DC1C |. 74 04      JE SHORT DUMeter.0048DC22    ; <== Continue check

```

- Tương tự như quá trình mã hoá SecII, SecIII cũng giống như vậy :

```

0048DC22 |> \8D45 DC      LEA EAX,[LOCAL.9]           ; <== SecIII
0048DC25 |. E8 6AFEFFFF CALL DUMeter.0048DA94       ; <== Convert to HEX value :
ValueIII

```

- Quá trình mã hoá SecIV dựa trên thuật toán CRC32 :

```

0048DC2C |. 8D45 C8      LEA EAX,[LOCAL.14]           ; <== SecIV
0048DC2F |. E8 1EFDFFFF CALL DUMeter.0048D952       ; <== CRC32 Encrypt
----- CRC32 Encrypt -----

```

```

0048D95F |. 83CE FF      OR ESI,FFFFFFFF             ; <== CRC32Value
0048D962 |. 33DB      XOR EBX,EBX                  ; <== i = 0x0
0048D964 |. EB 0D      JMP SHORT DUMeter.0048D973
0048D966 |> 8A041F    /MOV AL,BYTE PTR DS:[EDI+EBX]    ; <== SecIV[i]
0048D969 |. 8BD6      |MOV EDX,ESI                  ; <== CRC32Value
----- CRC32 -----

```

```

0048D93B |. 32C2      XOR AL,DL                  ; <== Temp = CRC32Value xor SecIV[i]
0048D93D |. 25 FF000000 AND EAX,0FF                ; <== Temp = Temp and 0xFF

```

```

0048D942 |. 8B0C85 B0504F>MOV ECX,DWORD PTR DS:[EAX*4+4F50B0] ; <== Value =
CRC32Table[Temp]
0048D949 |. C1EA 08    SHR EDX,8      ; <== CRC32Value = CRC32Value / 0x100
0048D94C |. 33CA      XOR ECX,EDX    ; <== Value = Value xor CRC32Value
0048D94E |. 8BC1      MOV EAX,ECX    ; <== Value
----- CRC32 -----
0048D970 |. 8BF0      |MOV ESI,EAX   ; <== CRC32Value = Value
0048D972 |. 43       |INC EBX     ; <== i++;
0048D973 |> 8BC7      MOV EAX,EDI    ; <== SecIV
0048D975 |. E8 22FFFFFF |CALL DUMeter.0048D89C ; <== LenSecIV
0048D97A |. 3BD8      |CMP EBX,EAX   ; <== while ( i < LenSecIV )
0048D97C |.^ 72 E8    |JB SHORT DUMeter.0048D966 ; <== Continue Loop
0048D97E |. 8BC6      MOV EAX,ESI    ; <== CRC32Value
----- CRC32 Encrypt -----
0048DC34 |. 33C3      XOR EAX,EBX   ; <== CRC32Value = CRC32Value xor ValueU

```

- Quá trình kiểm tra thứ hai diễn ra như sau :

```

0048DC36 |. 3BC6      CMP EAX,ESI   ; <== if ( CRC32Value == ValueIII )
0048DC38 75 04      JNZ SHORT DUMeter.0048DC3E ; <== Congrat !!!!
```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : D3-JD5-06FFE94D-6348F8BD

III – End of Tut :

- Finished – *September 13, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.dualvdcopy.com
Production :	Dual DVD Copy.
SoftWare :	Dual DVD Copy Gold 3.0
Copyright by :	Copyright © 2004, Dual DVD Copy. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 7.0 [Debug]
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Dual DVD Copy Gold 3.0

Dual DVD Copy allows you to make copy of the Video-DVD disc on DVD-R/RW, DVD+R/RW or CD-R/RW media or save it as AVI file on your hard drive. For this purpose you need the DVD-RW, DVD+RW or CD-RW burning drive. You should n't worry if you have combo drive which is used for reading and burning. Application able to perform copying using only one CD-RW/DVD+RW/DVD-ROM device which is at first used as DVD-ROM and then as CD/DVD Burner. If the movie will not fit to one disc the application asks you to insert another one to burn the last part of the movie on second disc.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 7.0 [Debug]**
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Incorrect registration Code.**" Ta tìm được đoạn CODE này ở hai địa chỉ :
 - 00404CAD . 68 C05A4800 PUSH dripper.00485AC0 ; ASCII "Incorrect registration Code."
 - Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
 - 004049FF . E8 1B2F0700 CALL dripper.0047791F ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Chương trình chuyển hai chuỗi S và U vào lưu tại hai địa chỉ được chỉ định trước :

```

00404A04 . 8B45 70 MOV EAX,DWORD PTR SS:[EBP+70]
00404A07 . 8D5424 0C LEA EDX,DWORD PTR SS:[ESP+C]
00404A0B . 2BD0 SUB EDX,EAX
00404A0D . 8D49 00 LEA ECX,DWORD PTR DS:[ECX]
00404A10 > 8A08 MOV CL,BYTE PTR DS:[EAX]
00404A12 . 880C02 MOV BYTE PTR DS:[EDX+EAX],CL
00404A15 . 40 INC EAX
00404A16 . 84C9 TEST CL,CL
00404A18 .^ 75 F6 JNZ SHORT dripper.00404A10
00404A1A . 8B45 74 MOV EAX,DWORD PTR SS:[EBP+74]
00404A1D . 8D9424 C00000>LEA EDX,DWORD PTR SS:[ESP+C0]
00404A24 . 2BD0 SUB EDX,EAX
00404A26 > 8A08 MOV CL,BYTE PTR DS:[EAX]
00404A28 . 880C02 MOV BYTE PTR DS:[EDX+EAX],CL
00404A2B . 40 INC EAX
00404A2C . 84C9 TEST CL,CL
00404A2E .^ 75 F6 JNZ SHORT dripper.00404A26

```

- Sau đó tiến hành kiểm tra chiều dài chuỗi U nhập :

```

00404A30 . 8D7424 0C LEA ESI,DWORD PTR SS:[ESP+C]
00404A34 . 8D4E 01 LEA ECX,DWORD PTR DS:[ESI+1]
00404A37 > 8A06 MOV AL,BYTE PTR DS:[ESI]
00404A39 . 46 INC ESI
00404A3A . 84C0 TEST AL,AL
00404A3C .^ 75 F9 JNZ SHORT dripper.00404A37
00404A3E . 2BF1 SUB ESI,ECX
00404A40 . 83FE 04 CMP ESI,4 ; <== U at least 4 charts
00404A43 . 0F8C 6B020000 JL dripper.00404CB4

```

- Kế đó, chương trình sẽ kiểm thực hiện một phép toán để tính xem chiều dài chuỗi nhập và giá trị tính toán có bằng nhau hay không. Thực tế quá trình này dùng để mặc định chiều dài của chuỗi U nhập phải là 15, 30, 45, 60 ... ký tự (lý do này sẽ được giải thích rõ hơn trong quá trình phân tích). Nếu chiều dài của U nhập nhỏ hơn các giá trị này thì chương trình sẽ tự động gán thêm các giá trị trong bảng giá trị mặc định vào sau chuỗi U nhập :

```

00404A49 . B8 89888888 MOV EAX,88888889 ; <== Default Value : dV
00404A4E . F7EE IMUL ESI ; <== Loop = dV * Len.U
00404A50 . 03D6 ADD EDX,ESI ; <== Loop = Loop + Len.U
00404A52 . C1FA 03 SAR EDX,3 ; <== Loop = Loop sar 0x3
00404A55 . 8BC2 MOV EAX,EDX ; <== Loop
00404A57 . C1E8 1F SHR EAX,1F ; <== Temp = Loop / 0x80000000

```

```

00404A5A . 8D4C02 01 LEA ECX,DWORD PTR DS:[EDX+EAX+1] ; <== Loop = Loop + Temp
00404A5E . 6BC9 0F IMUL ECX,ECX,0F ; <== Loop = Loop * 0xF
00404A61 . 3BF1 CMP ESI,ECX ; <== CMP (Loop, Len.U)
00404A63 . 8BC6 MOV EAX,ESI ; <== Len.U
00404A65 . 7D 18 JGE SHORT dripper.00404A7F ; <== if ( Len.U < Loop )
00404A67 . EB 07 JMP SHORT dripper.00404A70 ; <== then
00404A69 . 8DA424 000000>LEA ESP,DWORD PTR SS:[ESP]
00404A70 > 8A90 A0584800 MOV DL,BYTE PTR DS:[EAX+4858A0]; <== Temp =
dValue[Len.U+i]
00404A76 . 885404 0C MOV BYTE PTR SS:[ESP+EAX+C],DL ; <== U[ i ] = Temp (i =
Len.U)
00404A7A . 40 INC EAX ; <== i++
00404A7B . 3BC1 CMP EAX,ECX ; <== While ( i < Loop )
00404A7D .^ 7C F1 JL SHORT dripper.00404A70 ; <== Continue Loop

```

- Chuỗi này sau đó được dùng làm chuỗi mã hoá . Quá trình này diễn ra như sau :

```

00404A90 >/33C0 XOR EAX,EAX ; <== j = 0
00404A92 .|EB 0C JMP SHORT dripper.00404AA0
00404A94 .|8DA424 000000>LEA ESP,DWORD PTR SS:[ESP]
00404A9B .|EB 03 JMP SHORT dripper.00404AA0
00404A9D |8D49 00 LEA ECX,DWORD PTR DS:[ECX]
00404AA0 >|0FB65434 0C MOVZX EDX,BYTE PTR SS:[ESP+ESI+C]; <== U[i]
00404AA5 .|8A92 A0584800 MOV DL,BYTE PTR DS:[EDX+4858A0]; <== Temp = dValue[U[i]]
00404AAB .|305404 0C XOR BYTE PTR SS:[ESP+EAX+C],DL ; <== U[j] = U[j] xor Temp
00404AAF .|40 INC EAX ; <== j++
00404AB0 .|3BC1 CMP EAX,ECX ; <== while ( j < Loop )
00404AB2 .^|7C EC JL SHORT dripper.00404AA0 ; <== Continue Loop
00404AB4 .|46 INC ESI ; <== i++
00404AB5 .|3BF1 CMP ESI,ECX ; <== while ( i < Loop )
00404AB7 .^|7C D7 JL SHORT dripper.00404A90 ; <== Continue Loop

```

- Như trên đã đề cập . Chương trình sẽ quy định chiều dài của chuỗi U là 15, 30, 45 Tuy nhiên, chỉ có 15 ký tự đầu tiên sẽ được mã hoá ra chuỗi Serial thực, còn các ký tự sau sẽ được phân ra thành từng lớp, mỗi lớp 15 ký tự . Quá trình xác định số lớp diễn ra như sau :

```

00404AB9 >\B8 89888888 MOV EAX,88888889 ; <== dV
00404ABE . F7E9 IMUL ECX ; <== Layer = dV * Loop
00404AC0 . 03D1 ADD EDX,ECX ; <== Layer = Layer + Loop
00404AC2 . C1FA 03 SAR EDX,3 ; <== Layer = Layer sar 0x3
00404AC5 . 8BC2 MOV EAX,EDX ; <== Layer
00404AC7 . C1E8 1F SHR EAX,1F ; <== Layer = Layer / 0x80000000
00404ACA . 03D0 ADD EDX,EAX ; <== Layer = Layer + Temp
00404ACC . 83FA 01 CMP EDX,1 ; <== if ( Layer > 0x1 )
00404ACF . 7E 46 JLE SHORT dripper.00404B17 ; <== then
00404AD1 . 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]
00404AD5 . 8D72 FF LEA ESI,DWORD PTR DS:[EDX-1] ; <== Layer = Layer - 1

```

- Tuỳ theo số lớp được xác định mà từng lớp này sẽ được mã hoá cho 15 ký tự đầu tiên . Ở đây phân ra hai trường hợp (1) Nếu không có lớp nào (hay nói cách khác chiều dài là 15) thì chương trình chuyển để quá trình tạo chuỗi luôn (2) nếu số lớp được xác định (chiều dài từ 30 ký tự trở lên) thì từng lớp sẽ được tiến hành mã hoá với 15 ký tự đầu sau đó mới chuyển sang quá trình tạo chuỗi Serial .

- Quá trình mã hoá lớp với 15 ký tự đầu tiên diễn ra như sau :

```

00404AE0 > 33C0 XOR EAX,EAX ; <== i = 0
00404AE2 >|8A5401 FF MOV DL,BYTE PTR DS:[ECX+EAX-1] ; <== U[0xF + i]
00404AE6 .|305404 0C XOR BYTE PTR SS:[ESP+EAX+C],DL ; <== U[ i ] = U[ i ] xor
U[0xF*j + i ]

```

```

00404AEA .|8A1401    MOV DL,BYTE PTR DS:[ECX+EAX]      ; <== U[0xF + i + 1]
00404AED .|305404 0D XOR BYTE PTR SS:[ESP+EAX+D],DL   ; <== U[i + 1] = U[ i + 1] xor
U[0xF*j + i + 1 ]
00404AF1 .|8A5401 01 MOV DL,BYTE PTR DS:[ECX+EAX+1]    ; <== U[0xF + i + 2]
00404AF5 .|305404 0E XOR BYTE PTR SS:[ESP+EAX+E],DL   ; <== U[ i + 2] = U[i + 2] xor
U[0xF*j + i + 2 ]
00404AF9 .|8A5401 02 MOV DL,BYTE PTR DS:[ECX+EAX+2]    ; <== U[0xF + i + 3]
00404AFD .|305404 0F XOR BYTE PTR SS:[ESP+EAX+F],DL   ; <== U[ i + 3] = U[i + 3] xor
U[0xF*j + i + 3 ]
00404B01 .|8A5401 03 MOV DL,BYTE PTR DS:[ECX+EAX+3]    ; <== U[0xF + i + 4]
00404B05 .|305404 10 XOR BYTE PTR SS:[ESP+EAX+10],DL  ; <== U[i + 4] = U[i + 4] xor
U[0xF*j + i + 4 ]
00404B09 .|83C0 05 ADD EAX,5                           ; <== i += 5
00404B0C .|83F8 0F CMP EAX,0F                         ; <== while ( i < 0xF )
00404B0F .^|7C D1 JL SHORT dripper.00404AE2        ; <== Continue Loop
00404B11 .|83C1 0F ADD ECX,0F                         ; <== j++
00404B14 .|4E     DEC ESI                            ; <== Layer --
00404B15 .^|75 C9 JNZ SHORT dripper.00404AE0       ; <== Loop until Layer == 0x0

```

- Quá trình tạo chuỗi Serial diễn ra như sau :

```

00404B1C . 33C0    XOR EAX,EAX                      ; <== i = 0
00404B1E . 8BFF    MOV EDI,EDI
00404B20 > 0FB64C04 0C MOVZX ECX,BYTE PTR SS:[ESP+EAX+C] ; <== U[i]
00404B25 . 8A91 A0594800 MOV DL,BYTE PTR DS:[ECX+4859A0] ; <== Temp = dStr:[U[i]]
00404B2B . 0FB64C04 0D MOVZX ECX,BYTE PTR SS:[ESP+EAX+D] ; <== U[i+1]
00404B30 . 885404 0C MOV BYTE PTR SS:[ESP+EAX+C],DL    ; <== U[i] = Temp
00404B34 . 8A91 A0594800 MOV DL,BYTE PTR DS:[ECX+4859A0] ; <== Temp = dStr:[U[i+1]]
00404B3A . 0FB64C04 0E MOVZX ECX,BYTE PTR SS:[ESP+EAX+E] ; <== U[i+2]
00404B3F . 885404 0D MOV BYTE PTR SS:[ESP+EAX+D],DL    ; <== U[i+1] = Temp
00404B43 . 8A91 A0594800 MOV DL,BYTE PTR DS:[ECX+4859A0] ; <== Temp = dStr[U[i+2]]
00404B49 . 0FB64C04 0F MOVZX ECX,BYTE PTR SS:[ESP+EAX+F] ; <== U[i+3]
00404B4E . 885404 0E MOV BYTE PTR SS:[ESP+EAX+E],DL    ; <== U[i+2] = Temp
00404B52 . 8A91 A0594800 MOV DL,BYTE PTR DS:[ECX+4859A0] ; <== Temp = dStr[U[i+3]]
00404B58 . 0FB64C04 10 MOVZX ECX,BYTE PTR SS:[ESP+EAX+10] ; <== U[i+4]
00404B5D . 885404 0F MOV BYTE PTR SS:[ESP+EAX+F],DL    ; <== U[i+3] = Temp
00404B61 . 8A91 A0594800 MOV DL,BYTE PTR DS:[ECX+4859A0] ; <== Temp = dStr[U[i+4]]
00404B67 . 885404 10 MOV BYTE PTR SS:[ESP+EAX+10],DL   ; <== U[i+4] = Temp
00404B6B . 83C0 05 ADD EAX,5                         ; <== i += 5
00404B6E . 83F8 0F CMP EAX,0F                         ; <== while ( i < 0xF )
00404B71 .^|7C AD JL SHORT dripper.00404B20        ; <== Continue Loop
00404B73 . 6A 06 PUSH 6
00404B75 . 8D4424 1A LEA EAX,DWORD PTR SS:[ESP+1A]
00404B79 . 50     PUSH EAX
00404B7A . 8D4C24 1F LEA ECX,DWORD PTR SS:[ESP+1F]
00404B7E . 51     PUSH ECX
00404B7F . E8 9C0A0600 CALL dripper.00465620
00404B84 . 6A 0C PUSH 0C
00404B86 . 8D5424 21 LEA EDX,DWORD PTR SS:[ESP+21]
00404B8A . 52     PUSH EDX
00404B8B . 8D4424 26 LEA EAX,DWORD PTR SS:[ESP+26]
00404B8F . 50     PUSH EAX
00404B90 . E8 8B0A0600 CALL dripper.00465620

```

```

00404B95 . B0 2D      MOV AL,2D          ; <== Temp = 0x2D
00404B97 . 884424 29   MOV BYTE PTR SS:[ESP+29],AL    ; <== U[5] = Temp
00404B9B . 884424 2F   MOV BYTE PTR SS:[ESP+2F],AL    ; <== U[11] = Temp

```

- Như vậy, chuỗi Serial tạo thành sẽ có dạng “XXXXX-XXXXX-XXXXX”

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErTeAm

Serial : YFV53-QC44C-XOJZ3

III – KeyGen :

- /Section 1/- Xác định chiều dài mặc định của U .
- /Section 2/- Thêm các giá trị mặc định vào sau chuỗi U nhập để U đúng theo chiều dài xác định $reaName[LenUser + i] = reaDefaultValue[LenUser + i];$
- /Section 3/- Mã hoá chuỗi U mới này theo công thức :
 $reaName[j] = reaDefaultValue[(reaName[i] & 0xFF) ^ (reaName[j] & 0xFF)];$
- /Section 4/- Xác định số lớp chuỗi U . Nếu số lớp xác định, mã hoá 15 ký tự đầu theo công thức
 $reaName[i] = (reaName[i] & 0xFF) ^ (reaName[0xF*j + i] & 0xFF);$
- :
- /Section 5/- Mã hoá tạo thành chuỗi Serial :
 $reaSerial[j] = reaDefaultString[(reaName[i] & 0xFF)];$

IV – End of Tut :

- Finished – 23/07/2004

Reverse Engineering Association

SoftWare

Homepage :	http://www.DVD-Cloner.com
Production :	DVD-Cloner Inc.
SoftWare :	DVD-Cloner 2.32
Copyright by :	Copyright © 2002-2004 DVD-Cloner Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

DVD-Cloner 2.32

* Copy DVD movies to DVD+/-R(W) discs. * Copy DVD-9 movie to DVD-5 disc. * Copy DVD-9 movie to DVD-9 disc. * Backup DVD movies to hard disk. * "Movie only" option. * Support both DVD+R/RW and DVD-R/RW. * Support 'Shut down computer when finish'. * Online Registration.

What's new in DVD-Cloner 2.32 :

* Upgraded burning engine, support more DVD burners! * Make changes on register. * Fixed lots of minor bugs. * Optimized decoding kernel. * Support copy to double layer disc (D9).

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 6.0**

- Chạy thử chương trình với Email và Fake Serial ta nhận được thông báo "**Wrong code!**". Ta tìm được thông báo này tại địa chỉ :

00407427 |. 68 FC0A4400 PUSH Dvd-clon.00440AFC ; ASCII "Wrong code!"

- Truy ngược lên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

00407403 |. E8 5CB60200 CALL <JMP.&MFC42.#535> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Email và Fake Serial, chương trình dừng lại tại điểm đặt BP .

00407403 |. E8 5CB60200 CALL <JMP.&MFC42.#535> ; <== Set BreakPoint here

00407408 |. 8B0D 306D4400 MOV ECX,DWORD PTR DS:[446D30] ; |Dvd-clon.00446488

0040740E |. 81C1 D0030000 ADD ECX,3D0 ; |

00407414 |. E8 27100200 CALL Dvd-clon.00428440 ; \Dvd-clon.00428440

- Dùng F7 trace Into ta đến quá trình mã hoá chuỗi :

00428466 |. 8378 F8 10 CMP DWORD PTR DS:[EAX-8],10 ; <== LenS must be 16 charts

0042846A |. 0F85 57010000 JNZ Dvd-clon.004285C7

- Kế đến chương trình sẽ chuyển hai ký tự đầu tiên của chuỗi Serial ([0][1]) sang giá trị HEX tương ứng (ví dụ : ([0][1] : “12” ==> HexValue = 0xC)

004284B2 |> \8A08 MOV CL,BYTE PTR DS:[EAX] ; <== S[0]

004284B4 |. 8B35 84764300 MOV ESI,DWORD PTR DS:<&MSVCRT.sscanf> ; msvert.sscanf

004284BA |. 884C24 14 MOV BYTE PTR SS:[ESP+14],CL ; <== S[0]

004284BE |. 8A40 01 MOV AL,BYTE PTR DS:[EAX+1] ; <== S[1]

004284C1 |. 8D4C24 0C LEA ECX,DWORD PTR SS:[ESP+C]

004284C5 |. 8D5424 14 LEA EDX,DWORD PTR SS:[ESP+14]

004284C9 |. 51 PUSH ECX

004284CA |. 68 540A4400 PUSH Dvd-clon.00440A54 ; |format = "%d"

004284CF |. 52 PUSH EDX ; |s

004284D0 |. 884424 21 MOV BYTE PTR SS:[ESP+21],AL ; |

004284D4 |. 885C24 22 MOV BYTE PTR SS:[ESP+22],BL ; |

004284D8 |. FFD6 CALL ESI ; \sscanf

004284DA |. 8B4424 18 MOV EAX,DWORD PTR SS:[ESP+18] ; <== HexValue of S[0][1]

004284DE |. 83C4 0C ADD ESP,0C

004284E1 |. 83F8 0C CMP EAX,0C ; <== if (HexValue < 0xC)

004284E4 |. 0F8F DD000000 JG Dvd-clon.004285C7 ; <== Continue Check

- Tiếp đó, quá trình này được thực hiện tiếp cho hai ký tự thứ [2][3] :

004284F3 |. 8B4424 2C MOV EAX,DWORD PTR SS:[ESP+2C] ; <== FakeSerial

004284F7 |. 8A48 02 MOV CL,BYTE PTR DS:[EAX+2] ; <== S[2]

004284FA |. 884C24 14 MOV BYTE PTR SS:[ESP+14],CL

004284FE |. 8A40 03 MOV AL,BYTE PTR DS:[EAX+3] ; <== S[3]

00428501 |. 884424 15 MOV BYTE PTR SS:[ESP+15],AL

00428505 |. 8D4424 0C LEA EAX,DWORD PTR SS:[ESP+C]

00428509 |. 50 PUSH EAX

0042850A |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]

0042850E |. 68 540A4400 PUSH Dvd-clon.00440A54 ; ASCII "%d"

```

00428513 |. 51      PUSH ECX
00428514 |. 885C24 22  MOV BYTE PTR SS:[ESP+22],BL
00428518 |. FFD6    CALL ESI
0042851A |. 8B4424 18  MOV EAX,DWORD PTR SS:[ESP+18]      ; <== HexValue of S[2][3]
0042851E |. 83C4 0C    ADD ESP,0C
00428521 |. 83F8 1F    CMP EAX,1F                      ; <== if (HexValue < 0x1F)
00428524 |. 0F8F 9D000000 JG Dvd-clon.004285C7          ; <== Continue Check

```

- Hai ký tự kế tiếp [4][5] cũng được thực hiện y như vậy :

```

00428533 |. 8B4424 2C  MOV EAX,DWORD PTR SS:[ESP+2C]      ; <== FakeSerial
00428537 |. 8D5424 0C  LEA EDX,DWORD PTR SS:[ESP+C]
0042853B |. 52      PUSH EDX
0042853C |. 68 540A4400 PUSH Dvd-clon.00440A54          ; ASCII "%d"
00428541 |. 8A48 04    MOV CL,BYTE PTR DS:[EAX+4]          ; <== S[4]
00428544 |. 884C24 1C  MOV BYTE PTR SS:[ESP+1C],CL
00428548 |. 8A40 05    MOV AL,BYTE PTR DS:[EAX+5]          ; <== S[5]
0042854B |. 884424 1D  MOV BYTE PTR SS:[ESP+1D],AL
0042854F |. 8D4424 1C  LEA EAX,DWORD PTR SS:[ESP+1C]
00428553 |. 50      PUSH EAX
00428554 |. 885C24 22  MOV BYTE PTR SS:[ESP+22],BL
00428558 |. FFD6    CALL ESI
0042855A |. 8B4424 18  MOV EAX,DWORD PTR SS:[ESP+18]      ; <== HexValue of S[4][5]
0042855E |. 83C4 0C    ADD ESP,0C
00428561 |. 83F8 18    CMP EAX,18                      ; <== if (HexValue < 0x18)
00428564 |. 7F 61    JG SHORT Dvd-clon.004285C7          ; <== Continue Check

```

- Cuối cùng chương trình thực hiện cùng quá trình như vậy cho ký tự thứ [6][7] :

```

0042856A |. 8B4424 2C  MOV EAX,DWORD PTR SS:[ESP+2C]      ; <== FakeSerial
0042856E |. 8D5424 14  LEA ECX,DWORD PTR SS:[ESP+C]
00428572 |. 8A48 06    MOV CL,BYTE PTR DS:[EAX+6]          ; <== S[6]
00428575 |. 884C24 14  MOV BYTE PTR SS:[ESP+14],CL
00428579 |. 8A40 07    MOV AL,BYTE PTR DS:[EAX+7]          ; <== S[7]
0042857C |. 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
00428580 |. 884424 15  MOV BYTE PTR SS:[ESP+15],AL
00428584 |. 51      PUSH ECX
00428585 |. 68 540A4400 PUSH Dvd-clon.00440A54          ; ASCII "%d"
0042858A |. 52      PUSH EDX
0042858B |. 885C24 22  MOV BYTE PTR SS:[ESP+22],BL
0042858F |. FFD6    CALL ESI
00428591 |. 8B4424 18  MOV EAX,DWORD PTR SS:[ESP+18]      ; <== HexValue of S[6][7]
00428595 |. 83C4 0C    ADD ESP,0C
00428598 |. 83F8 3C    CMP EAX,3C                      ; <== if (HexValue < 0x3C)
0042859B |. 7F 2A    JG SHORT Dvd-clon.004285C7          ; <== Continue Check

```

- 8 ký tự còn lại của chuỗi Serial cũng như chuỗi Email không tham gia vào quá trình mã hóa . Quá trình đăng ký này yêu cầu phải CONNECT với INTERNET .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 04202211PCS3OSHD

III – KeyGen :

/Section I /- 8 ký tự đầu tiên được tạo ngẫu nhiên theo quy tắc sau :

[0][1]	<= 0xC	12
[2][3]	<= 0x1F	31
[4][5]	<= 0x18	24
[6][7]	<= 0x3C	60

/Section II /- 8 ký tự sau tạo ngẫu nhiên vào được nối vào sau 8 ký tự đầu tiên .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.earmaster.com
Production :	MidiTec Denmark.
SoftWare :	EarMaster Pro 4.0
Copyright by :	Copyright © 1996-2003 MidiTec Denmark. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

EarMaster Pro 4.0

EarMaster is an advanced musical ear training program containing exercises within the areas: Chords, Intervals, Scales, Rhythms and Melodies.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**This is not a valid EarMaster serial number .**" . Tuy nhiên, ta không thể tìm thấy chuỗi này trong quá trình tìm kiếm . Nhưng ta lại tìm được chuỗi :

```
00482CA7 . B3 17      MOV BL,17          ; <== 23 Default String
00482CA9 . BE 849F5100 MOV ESI,ear40.00519F84    ; ASCII "537-1771-5612 ...
00482CAE >/8D45 DC    LEA EAX,DWORD PTR SS:[EBP-24]
00482CB1 . |8BD6      MOV EDX,ESI
00482CB3 . |E8 7812F8FF CALL ear40.00403F30
00482CB8 . |8B55 DC    MOV EDX,DWORD PTR SS:[EBP-24]    ; <== Default String
00482CBB . |8B45 FC    MOV EAX,DWORD PTR SS:[EBP-4]    ; <== Fake Serial
00482CBE . |E8 D913F8FF CALL ear40.0040409C    ; <== Compare
```

- Ta có thể nghĩ rằng đây là chuỗi mặc định của chương trình . Tuy nhiên, khi kiểm tra lại ta thấy các chuỗi này không thoả, vậy đây chính là các chuỗi Serial nằm trong BLACK LIST :

- Đối với trường hợp này, ta dò qua đoạn CODE kiểm tra chuỗi trong BLACK LIST và đặt BreakPoint ngay sau đó :

00482CD0 . 807D FB 00 CMP BYTE PTR SS:[EBP-5],0 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Dò tiếp xuống chút nữa ta để quá trình kiểm tra các ký tự “-“ trong chuỗi S nhập. Theo mặc định là phải có 3 ký tự này :

```
00482D04 . 8D45 DC LEA EAX,DWORD PTR SS:[EBP-24]
00482D07 . 50 PUSH EAX
00482D08 . 33C9 XOR ECX,ECX
00482D0A . 8ACB MOV CL,BL
00482D0C . 49 DEC ECX
00482D0D . BA 01000000 MOV EDX,1
..... cutting .....
00482D3B . 8AD3 MOV DL,BL
00482D3D . 42 INC EDX
00482D3E . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00482D41 . E8 4A14F8FF CALL ear40.00404190
00482D46 . 8B45 D8 MOV EAX,DWORD PTR SS:[EBP-28]
00482D49 . 8D55 DC LEA EDX,DWORD PTR SS:[EBP-24]
00482D4C . E8 CB64F8FF CALL ear40.0040921C
00482D51 . 8B55 DC MOV EDX,DWORD PTR SS:[EBP-24]
00482D54 . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
00482D57 . E8 4C10F8FF CALL ear40.00403DA8
00482D5C . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
00482D5F . B8 D42E4800 MOV EAX,ear40.00482ED4
00482D64 . E8 A3A6F8FF CALL ear40.0040D40C
00482D69 . 8BD8 MOV EBX,EAX
00482D6B . 84DB TEST BL,BL
00482D6D . 0F94C1 SETE CL
00482D70 . 84C9 TEST CL,CL
00482D72 . 0F85 F4000000 JNZ ear40.00482E6C
```

- Kế đó, chương trình sẽ kiểm tra chiều dài các đoạn của chuỗi S nhập :

```
00482DC5 . 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C] ; <== SecI
00482DC8 . E8 BF11F8FF CALL ear40.00403F8C
00482DCD . 83F8 03 CMP EAX,3 ; <== Must be 3 charts
00482DD0 . 75 1E JNZ SHORT ear40.00482DF0
00482DD2 . 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10] ; <== SecII
00482DD5 . E8 B211F8FF CALL ear40.00403F8C
00482DDA . 83F8 04 CMP EAX,4 ; <== Must be 4 charts
00482DDD . 75 11 JNZ SHORT ear40.00482DF0
00482DDF . 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14] ; <== SecIII
00482DE2 . E8 A511F8FF CALL ear40.00403F8C
00482DE7 . 83F8 04 CMP EAX,4 ; <== Must be 4 charts
00482DEA . 75 04 JNZ SHORT ear40.00482DF0
```

- Tiếp đến, từng đoạn này sẽ được chuyển thành giá trị ở dạng HEX tương ứng (ví dụ : **SecI “I23”** sẽ được chuyên thành **ValueI = 0x7B**)

00482DF9 . 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C] ; <== SecI

```

00482DFC . E8 A3FFF7FF CALL ear40.00402DA4 ; <== Convert String to HEX value
00482E01 . 8BD8    MOV EBX,EAX
00482E03 . 8D55 E4    LEA EDX,DWORD PTR SS:[EBP-1C]
00482E06 . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10] ; <== SecII
00482E09 . E8 96FFF7FF CALL ear40.00402DA4 ; <== Convert String to HEX value
00482E0E . 8BF0    MOV ESI,EAX
00482E10 . 8D55 E0    LEA EDX,DWORD PTR SS:[EBP-20]
00482E13 . 8B45 EC    MOV EAX,DWORD PTR SS:[EBP-14] ; <== SecIII
00482E16 . E8 89FFF7FF CALL ear40.00402DA4 ; <== Convert String to HEX value
----- Convert String to HEX value -----

```

Kỹ thuật chuyển đổi này được mô phỏng lại như sau (VC++) :

```

int StringtoHEXValue(char reaString[10])
{
    int i=0, Value=0;
    while ( i < lstrlen(reaString))
    {
        Value = (Value * 0xA) + ((reaString[i] & 0xFF)-0x30);
        i++;
    }
    return Value;
}
----- Convert String to HEX value -----

```

- Quá trình mã hoá diễn ra ngay sau đó :

```

00482E1D . 837D E8 00    CMP DWORD PTR SS:[EBP-18],0 ; <== if ( ValueI != 0 )
00482E21 . 75 31    JNZ SHORT ear40.00482E54 ; <== Next check
00482E23 . 837D E4 00    CMP DWORD PTR SS:[EBP-1C],0 ; <== if ( ValueII != 0 )
00482E27 . 75 2B    JNZ SHORT ear40.00482E54 ; <== Next check
00482E29 . 837D E0 00    CMP DWORD PTR SS:[EBP-20],0 ; <== if ( ValueIII != 0 )
00482E2D . 75 25    JNZ SHORT ear40.00482E54 ; <== Next check
00482E2F . 85DB    TEST EBX,EBX
00482E31 . 74 21    JE SHORT ear40.00482E54
00482E33 . 03F3    ADD ESI,EBX ; <== Value = ValueI + ValueII
00482E35 . 69C6 E3020000 IMUL EAX,ESI,2E3 ; <== Value = Value * 0x2E3
00482E3B . BE 10270000 MOV ESI,2710 ; <== dV
00482E40 . 99      CDQ
00482E41 . F7FE    IDIV ESI ; <== Value = Value % dV
00482E43 . 3BCA    CMP ECX,EDX ; <== if ( ValueIII == Value )
00482E45 . 75 0D    JNZ SHORT ear40.00482E54 ; <== Next check
00482E47 . 8B45 F4    MOV EAX,DWORD PTR SS:[EBP-C] ; <== SecI
00482E4A . 8078 01 30  CMP BYTE PTR DS:[EAX+1],30 ; <== if ( SecI[1] == 0x30 )
00482E4E . 75 04    JNZ SHORT ear40.00482E54 ; <== Congrat !!!!!!!!!

-----
```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 603-2169-8508

III – KeyGen :

- /Section I /- Tạo hai chuỗi ngẫu nhiên. Ký tự thứ hai của chuỗi đầu tiên phải là “0”
- /Section II /- Chuyển hai chuỗi này sang giá trị dạng HEX tương ứng .
- /Section III /- Tính toán giá trị của đoạn thứ III
- /Section IV /- Chuỗi Serial thực có định dạng "%03i-%04i-%04i"

IV – End of Tut :

- Finished – *August 26, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.softheap.com
Production	:	Ixis Research, Ltd.
SoftWare	:	Easy Desktop Keeper 1.5
Copyright by	:	Copyright © 2004 Ixis Research, Ltd. All Rights Reserved.
Type	:	Serial
Packed	:	N/A
Language	:	Borland Delphi 6.0 - 7.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N/A
Request	:	Correct Serial / KeyGen

Easy Desktop Keeper 1.5

This program gives you the ability to save, restore, manage and lock your desktop layout that includes files and folders located on your desktop, placement of desktop icons and desired wallpaper. If you choose to lock your desktop layout, every time you reboot your PC, the program will restore your desktop icons and bring them back to their original positions as well as return your old wallpaper to the background. You can create an unlimited number of desktop layouts for different purposes such as gaming, working, surfing the Internet as well as provide different users with their own desktops. Simply place your icons wherever you want them on your desktop, select desired wallpaper, double click program's tray icon and click the Save button to create a new desktop layout file. If your icons ever get moved or deleted by any user or after entering the safe mode or changing the screen resolution, just open the program and click the Load button to return your desktop to any previously saved state, or simply reboot your PC if you are in the Lock Desktop mode. Many of you, after meticulously arranging all the icons on your desktop, have been frustrated when somebody else or Windows itself randomly re-arranges them. It is getting more frustrating when you get your icons deleted from your desktop without an ability to recreate them back. This easy-to-use utility will give you new exciting abilities to manage your desktop; you'll get rid of a lot of headaches, and get a peace of mind.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**" . Ta tìm thấy chuỗi thông báo này tại địa chỉ :
004903A8 |. BA D4044900 MOV EDX,desksave.004904D4 ; ASCII "Registration code is invalid!"
- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :
0049028F |. E8 FC4DFCFF CALL desksave.00455090 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

004902AC |. E8 DBFDFFFF CALL desksave.0049008C ; <== Trace Into

===== **Trace Into** =====

0049009A . E8 A943F7FF	CALL desksave.00404448	; <== LenS
0049009F . 83F8 0E	CMP EAX,0E	; <== LenS must be 14 charts
004900A2 . 75 67	JNZ SHORT desksave.0049010B	
004900A4 . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900A6 . 8038 33	CMP BYTE PTR DS:[EAX],33	; <== S[0] == 0x33
004900A9 . 0F94C0	SETE AL	
004900AC . 83E0 7F	AND EAX,7F	
004900AF . 03F0	ADD ESI,EAX	
004900B1 . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900B3 . 8078 02 33	CMP BYTE PTR DS:[EAX+2],33	; <== S[2] == 0x33
004900B7 . 0F94C0	SETE AL	
004900BA . 83E0 7F	AND EAX,7F	
004900BD . 03F0	ADD ESI,EAX	
004900BF . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900C1 . 8078 03 39	CMP BYTE PTR DS:[EAX+3],39	; <== S[3] == 0x39
004900C5 . 0F94C0	SETE AL	
004900C8 . 83E0 7F	AND EAX,7F	
004900CB . 03F0	ADD ESI,EAX	
004900CD . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900CF . 8078 04 30	CMP BYTE PTR DS:[EAX+4],30	; <== S[4] == 0x30
004900D3 . 0F94C0	SETE AL	
004900D6 . 83E0 7F	AND EAX,7F	
004900D9 . 03F0	ADD ESI,EAX	
004900DB . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900DD . 8078 07 38	CMP BYTE PTR DS:[EAX+7],38	; <== S[7] == 0x38
004900E1 . 0F94C0	SETE AL	
004900E4 . 83E0 7F	AND EAX,7F	
004900E7 . 03F0	ADD ESI,EAX	
004900E9 . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900EB . 8078 08 38	CMP BYTE PTR DS:[EAX+8],38	; <== S[8] == 0x38
004900EF . 0F94C0	SETE AL	
004900F2 . 83E0 7F	AND EAX,7F	
004900F5 . 03F0	ADD ESI,EAX	
004900F7 . 8B07	MOV EAX,DWORD PTR DS:[EDI]	
004900F9 . 8078 0A 32	CMP BYTE PTR DS:[EAX+A],32	; <== S[10] == 0x32

===== **Trace Into** =====

- Quá trình mã hoá này cực kỳ đơn giản, chỉ là trong chuỗi Serial ở 7 vị trí xác định trước là các ký tự mặc định . Các ký tự còn lại là ngẫu nhiên .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 3Z3902N88G2M7Q

III – End of Tut :

- Finished – September 14, 2004

Reverse Engineering Association

SoftWare

Homepage :	http://www.emailarms.com
Production :	F-Key Solutions, Inc.
SoftWare :	Evidence Destructor 2.0
Copyright by :	Copyright © 2004 F-Key Solutions, Inc. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Evidence Destructor 2.0

Did you know that when you use Windows, open folders, search for files, work with various software, surf the Internet, see erotic pictures, use online banking to access your accounts, and so on, information about all your steps: all pictures you have seen, all passwords and bank accounts you use is stored in various files and in the Windows registry? Did you know that anybody with a minimal computer knowledge including your spouse and children, your boss and the police can easily extract detailed information about your activities and use it against you? Did you also know that when you delete your files using the Delete command, Windows does not actually delete the files but places them to the hidden RECYCLER folder, and even if you use the Empty Recycle Bin command, some files may stay in this folder and can not be deleted by Windows? Evidence Destructor is a complex security system that ensures your privacy by destroying all hidden information connected with your activities according to a user-defined schedule or on each boot/shutdown.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**". Ta tìm được thông báo này tại địa chỉ :
00498634 |. BA 48874900 MOV EDX,eraser.00498748 ; ASCII "Registration code is invalid!"
- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :
0049851B |. E8 583CFDFF CALL eraser.0046C178 ; <== Set BreakPoint here

II – Cracking :

- Quá trình mã hoá của chương trình này rất đơn giản . Từ BP ta trace xuống chút :
00498538 |. E8 7BFDFFFF CALL eraser.004982B8 ; <== Trace Into
----- Trace Into -----
004982C6 |. E8 CDC8F6FF CALL eraser.00404B98 ; <== Get Length Serial
004982CB |. 83F8 0E CMP EAX,0E ; <== LenS must be 14 charts
004982CE |. 75 67 JNZ SHORT eraser.00498337
004982D0 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]

```

004982D2 |. 8038 32    CMP BYTE PTR DS:[EAX],32      ; <== S[0] = 0x32
004982D5 |. 0F94C0    SETE AL
004982D8 |. 83E0 7F    AND EAX,7F
004982DB |. 03F0    ADD ESI,EAX
004982DD |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
004982DF |. 8078 02 37  CMP BYTE PTR DS:[EAX+2],37    ; <== S[2] = 0x37
004982E3 |. 0F94C0    SETE AL
004982E6 |. 83E0 7F    AND EAX,7F
004982E9 |. 03F0    ADD ESI,EAX
004982EB |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
004982ED |. 8078 03 30  CMP BYTE PTR DS:[EAX+3],30    ; <== S[3] = 0x30
004982F1 |. 0F94C0    SETE AL
004982F4 |. 83E0 7F    AND EAX,7F
004982F7 |. 03F0    ADD ESI,EAX
004982F9 |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
004982FB |. 8078 04 31  CMP BYTE PTR DS:[EAX+4],31    ; <== S[4] = 0x31
004982FF |. 0F94C0    SETE AL
00498302 |. 83E0 7F    AND EAX,7F
00498305 |. 03F0    ADD ESI,EAX
00498307 |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
00498309 |. 8078 07 33  CMP BYTE PTR DS:[EAX+7],33    ; <== S[7] = 0x33
0049830D |. 0F94C0    SETE AL
00498310 |. 83E0 7F    AND EAX,7F
00498313 |. 03F0    ADD ESI,EAX
00498315 |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
00498317 |. 8078 08 33  CMP BYTE PTR DS:[EAX+8],33    ; <== S[8] = 0x33
0049831B |. 0F94C0    SETE AL
0049831E |. 83E0 7F    AND EAX,7F
00498321 |. 03F0    ADD ESI,EAX
00498323 |. 8B07    MOV EAX,DWORD PTR DS:[EDI]
00498325 |. 8078 0A 34  CMP BYTE PTR DS:[EAX+A],34    ; <== S[10] = 0x34
00498329 |. 0F94C0    SETE AL
0049832C |. 83E0 7F    AND EAX,7F
0049832F |. 03F0    ADD ESI,EAX
00498331 |. 83FE 07    CMP ESI,7                  ; <== if ALL correct
00498334 |. 0F94C3    SETE BL                  ; <== EBX = 0x1
00498337 > 8BC3    MOV EAX,EBX                ; <== EAX = EBX
----- Trace Into -----

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 227010P3384Y88

III – KeyGen :

/Section I /- Chuỗi Serial có chiều dài là 14 ký tự .

/Section II /- Các vị trí [0][2][3][4][7][8][10] là các ký tự mặc định .

IV – End of Tut :

- Finished – *August 23, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.a7soft.com
Production	:	A7Soft
SoftWare	:	ExamXML 3.12
Copyright by	:	Copyright © 2002-2004 A7Soft. All Rights Reserved.
Type	:	Name / Email / Serial
Packed	:	N / A
Language	:	Microsoft Visual C++ 7.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

ExamXML 3.12

Fast loading and comparison.

Loading XML data from Internet resources such as XML services.

XML schema validation during loading.

Command Line for the simple starting of programs with parameters.

Possibilities to compare several XML files.....

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 7.0**
- Nhập thử Fake Serial, ta nhận được thông báo "Please try again" Ta tìm được chuỗi này tại địa chỉ : 0040D800 . 68 2C814A00 PUSH ExamXML.004A812C ; |Text = "Please try again"
- Dò ngược lên trên và đặt BreakPoint tại : 0040D7A1 . E8 6AB9FFFF CALL ExamXML.00409110 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP : 0040D7A1 . E8 6AB9FFFF CALL ExamXML.00409110 ; <== Set BreakPoint here
- 0040D7A6 . 50 PUSH EAX ; ASCII "rea@rea.net"
- 0040D7A7 . 68 C0585200 PUSH ExamXML.005258C0 ; ASCII "rea"
- 0040D7AC . 68 40595200 PUSH ExamXML.00525940 ; ASCII "REA-cRaCkErStEaM"
- 0040D7B1 . 68 C0595200 PUSH ExamXML.005259C0 ; <== Trace into here
- 0040D7B6 . E8 15F10000 CALL ExamXML.0041C8D0
- Dùng F7 trace into và trace tiếp một đoạn ta đến :
- 0041C8EE . 51 PUSH ECX ; <== Email
- 0041C8EF . E8 6CFFFFFF CALL ExamXML.0041C860 ; <== trace into here
- Lại dùng F7 trace into, ta đến quá trình mã hoá thứ nhất :
- 0041C862 . 8B7C24 0C MOV EDI,DWORD PTR SS:[ESP+C] ; <== Email
- 0041C866 . 8A07 MOV AL,BYTE PTR DS:[EDI] ; <== E[0]
- 0041C868 . 33F6 XOR ESI,ESI ; <== Value = 0x0
- 0041C86A . 84C0 TEST AL,AL
- 0041C86C . 74 27 JE SHORT ExamXML.0041C895
- 0041C86E . 8BFF MOV EDI,EDI ; <== Email

```

0041C870 |> 0FBEC0    /MOVsx EAX,AL          ; <== E[i]
0041C873 |. 50        |PUSH EAX
0041C874 |. E8 20A50700 |CALL ExamXML.00496D99 ; <== CharUpper[E[i]]
0041C879 |. 83C4 04   |ADD ESP,4
0041C87C |. 47        |INC EDI          ; <== i++
0041C87D |. 3C 20    |CMP AL,20         ; <== if ( E[i] > 0x20 )
0041C87F |. 7E 0E    |JLE SHORT ExamXML.0041C88F ; <== then
0041C881 |. 6BF6 25   |IMUL ESI,ESI,25    ; <== Value = Value * 0x25
0041C884 |. 0FBEC0    |MOVsx EAX,AL          ; <== E[i]
0041C887 |. 81E6 FFFFFFFF00 |AND ESI,0FFFFFFF ; <== Value = Value and 0xFFFFFFF
0041C88D |. 03F0    |ADD ESI,EAX         ; <== Value = Value + E[i]
0041C88F |> 8A07    |MOV AL,BYTE PTR DS:[EDI] ; <== E[i+1]
0041C891 |. 84C0    |TEST AL,AL          ; <== End of E
0041C893 |.^ 75 DB    |JNZ SHORT ExamXML.0041C870 ; <== If NOT continue Loop
0041C895 |> 8B7C24 14  MOV EDI,DWORD PTR SS:[ESP+14] ; <== DefaultValue : dV1
0041C899 |. 33C9    XOR ECX,ECX
0041C89B |. 85FF    TEST EDI,EDI
0041C89D |. 76 2E    JBE SHORT ExamXML.0041C8CD
0041C89F |. 53        PUSH EBX
0041C8A0 |. 8B5C24 1C  MOV EBX,DWORD PTR SS:[ESP+1C] ; <== dV2
0041C8A4 |. 55        PUSH EBP
0041C8A5 |> 33D2    /XOR EDX,EDX
0041C8A7 |. 8BC6    |MOV EAX,ESI          ; <== Value
0041C8A9 |. BD 28000000 |MOV EBP,28         ; <== dV3
0041C8AE |. F7F5    |DIV EBP          ; <== Char = Value % dV3
0041C8B0 |. A1 9C775000 |MOV EAX,DWORD PTR DS:[50779C] ; <== Default String : dStr
0041C8B5 |. C1EE 05   |SHR ESI,5          ; <== Value = Value / 0x20
0041C8B8 |. 0FAFF3   |IMUL ESI,EBX        ; <== Value = Value * dV2
0041C8BB |. 41        |INC ECX          ; <== i++
0041C8BC |. 3BCF    |CMP ECX,EDI         ; <== while ( i < dV1 )
0041C8BE |. 8A1402   |MOV DL,BYTE PTR DS:[EDX+EAX] ; <== Char = dStr[Char]
0041C8C1 |. 8B4424 18  |MOV EAX,DWORD PTR SS:[ESP+18]
0041C8C5 |. 885401 FF  |MOV BYTE PTR DS:[ECX+EAX-1],DL ; <== Temp[i*2] = Char
0041C8C9 |.^ 72 DA    \JB SHORT ExamXML.0041C8A5 ; <== Continue Loop

```

- Quá trình so sánh thứ nhất diễn ra như sau :

```

0041C8F4 . 8B7424 54  MOV ESI,DWORD PTR SS:[ESP+54] ; <== FakeSerial : fS
0041C8F8 . 83C4 10   ADD ESP,10
0041C8FB . 33C0    XOR EAX,EAX          ; <== i = 0
0041C8FD . 8D49 00   LEA ECX,DWORD PTR DS:[ECX]
0041C900 > 8A5404 04  MOV DL,BYTE PTR SS:[ESP+EAX+4]
0041C904 . 3A1446   CMP DL,BYTE PTR DS:[ESI+EAX*2]
0041C907 . 75 54   JNZ SHORT ExamXML.0041C95D ; <== if ( fS[i*2] == Temp[i] )
0041C909 . 40        INC EAX          ; <== then
0041C90A . 83F8 08   CMP EAX,8          ; <== i++
0041C90D .^ 7C F1    JL SHORT ExamXML.0041C900 ; <== while ( i < 8 )
                                                ; <== Continue check

```

- Quá trình này ta nhận xét được ngay, các ký tự của chuỗi Serial thực sẽ là các ký tự của chuỗi **Temp** nhưng nằm ở các vị trí "*i*2*". Các vị trí "*i*2 + 1*" là các ký tự ngẫu nhiên. Như vậy, chuỗi serial đầu tiên có 15 ký tự.

- Quá trình mã hoá lần thứ hai cũng là quá trình mã hoá lần thứ nhất, nhưng với các thông số đầu vào tay đổi như sau :

Lần thứ nhất

Lần thứ hai

Email
dV2 = 9
Tạo 8 ký tự

Temp
dV2 = 1
Tạo 5 ký tự

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErStEaM

Email : rea@rea.net

Serial : J3J2XAK8YGRUJ5VD1HSP

III – KeyGen :

- /Section 1/- Tạo chuỗi ngẫu nhiên gồm 15 ký tự.
- /Section 2/- Tính theo quá trình mã hoá thứ nhất . Xếp các ký tự mã hoá này vào chuỗi ngẫu nhiên theo thứ tự “*i*2*”
- /Section 3/- Thực hiện quá trình mã hoá thứ hai . Nối chuỗi tạo thành vào sau chuỗi đầu tiên .

IV – End of Tut :

- Finished – 27/07/2004

Reverse Engineering Association

SoftWare

Homepage :	http://www.salfeld.com
Production :	Salfeld Computer
SoftWare :	Exe Password 2004 1.1
Copyright by :	Copyright © 2002-2004 Salfeld Computer. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Exe Password 2004 1.1

Exe Password allows you to protect any EXE-file with its own password. In doing so, this password is stored directly in the EXE-file. Adding any password to a file is extremely easy: Just select the desired file on the desktop/in the start menu or in Explorer, choose the context menu (right mouse click), select password protection, enter password - finished. From now on a password must be entered when the application / EXE-file is started.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**
- Nhập thử Fake Serial, ta nhận được thông báo "**Sorry, wrong serial number.**" Ta tìm được chuỗi này tại địa chỉ :
- 00550D63 . B8 EC105500 MOV EAX,ExePW.005510EC ; |ASCII "Sorry, wrong serial number."
- Dò ngược lên trên và đặt BreakPoint tại :
- 00550BBA . E8 9539EBFF CALL ExePW.00404554 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP :


```
00550BBA . E8 9539EBFF CALL ExePW.00404554 ; <== Set BreakPoint here
00550BBF . 83F8 06    CMP EAX,6   ; <== U atleast 6 charts
00550BC2 . 7F 4F      JG SHORT ExePW.00550C13
```
- Trace tiếp ta đến :


```
00550C33 . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10] ; <== User
00550C36 . 5A         POP EDX   ; <== Fake Serial
00550C37 . E8 682B0000 CALL ExePW.005537A4 ; <== Encrypt and compare
00550C3C . 84C0      TEST AL,AL ; <== Return value
00550C3E . 0F84 ED000000 JE ExePW.00550D31 ; <== Congratulation !
```
- Dùng F7 trace into ta đến quá trình so sánh đầu tiên . Chương trình sẽ so sánh ký tự thứ 4 của chuỗi Serial với một ký tự mặc định :


```
00553826 |. 8B45 EC    MOV EAX,[LOCAL.5] ; <== S[3]
00553829 |. BA C43B5500 MOV EDX,ExePW.00553BC4 ; <== Default Charts : "5"
0055382E |. E8 6D0EEBFF CALL ExePW.004046A0 ; <== Must be the same
00553833 |. 0F85 6A010000 JNZ ExePW.005539A3 ; <== if SAME go to next check
```
- Sau đó chương trình sẽ nối hai ký tự [1][2] của chuỗi Fake Serial vào hai ký tự mặc định để tạo thành một chuỗi mới : "0[1][2]5"


```
00553839 |. 68 D03B5500 PUSH ExePW.00553BD0
0055383E |. FF75 F0    PUSH [LOCAL.4]
00553841 |. 68 C43B5500 PUSH ExePW.00553BC4
00553846 |. 8D45 E0    LEA EAX,[LOCAL.8]
00553849 |. BA 03000000 MOV EDX,3
0055384E |. E8 C10DEBFF CALL ExePW.00404614 ; <== Concat String
```
- Tiếp đến, các ký tự từ thứ [5]-[8] của chuỗi Fake Serial cũng được nối vào sau chuỗi này :


```
005538D3 |. B9 01000000 MOV ECX,1 ; <== Get 1 charts
005538D8 |. BA 06000000 MOV EDX,6 ; <== from 6th
005538DD |. 8B45 F8    MOV EAX,[LOCAL.2] ; <== of Fake Serial
005538E0 |. E8 CF0EEBFF CALL ExePW.004047B4 ; <== cutting
005538E5 |. FF75 C0    PUSH [LOCAL.16]
005538E8 |. 8D45 BC    LEA EAX,[LOCAL.17]
005538EB |. 50         PUSH EAX
005538EC |. B9 01000000 MOV ECX,1 ; <== Get 1 charts
005538F1 |. BA 07000000 MOV EDX,7 ; <== from 7th
005538F6 |. 8B45 F8    MOV EAX,[LOCAL.2] ; <== of Fake Serial
005538F9 |. E8 B60EEBFF CALL ExePW.004047B4 ; <== cutting
005538FE |. FF75 BC    PUSH [LOCAL.17]
00553901 |. 8D45 B8    LEA EAX,[LOCAL.18]
00553904 |. 50         PUSH EAX
00553905 |. B9 01000000 MOV ECX,1 ; <== Get 1 charts
0055390A |. BA 08000000 MOV EDX,8 ; <== from 8th
0055390F |. 8B45 F8    MOV EAX,[LOCAL.2] ; <== of Fake Serial
00553912 |. E8 9D0EEBFF CALL ExePW.004047B4 ; <== cutting
00553917 |. FF75 B8    PUSH [LOCAL.18]
0055391A |. 8D45 B4    LEA EAX,[LOCAL.19]
0055391D |. 50         PUSH EAX
0055391E |. B9 01000000 MOV ECX,1 ; <== Get 1 charts
00553923 |. BA 09000000 MOV EDX,9 ; <== from 9th
00553928 |. 8B45 F8    MOV EAX,[LOCAL.2] ; <== of Fake Serial
```

```

0055392B |. E8 840EEBFF CALL ExePW.004047B4 ; <== cutting
00553930 |. FF75 B4    PUSH [LOCAL.19]
00553933 |. 8D45 E0    LEA EAX,[LOCAL.8]
00553936 |. BA 06000000 MOV EDX,6
0055393B |. E8 D40CEBFF CALL ExePW.00404614 ; <== Concat String
- Qúa trình mã hoá kế tiếp được tiến hành như sau :
00553940 |. 8B45 F0    MOV EAX,[LOCAL.4] ; <== S[1][2]
00553943 |. E8 3C57EBFF CALL ExePW.00409084 ; <== Convert to HEX Value : hV
00553948 |. 03F3    ADD ESI,EBX ; <== Value = S[5] + S[6]
0055394A |. 03FE    ADD EDI,ESI ; <== Value = Value + S[7]
0055394C |. 037D D4    ADD EDI,[LOCAL.11] ; <== Value = Value + S[8]
0055394F |. 03C7    ADD EAX,EDI ; <== Value = Value + hV
00553951 |. 50    PUSH EAX ; <== Value
00553952 |. 8B45 E8    MOV EAX,[LOCAL.6]
00553955 |. E8 2A57EBFF CALL ExePW.00409084 ; <== Temp = S[13] - 0x30
0055395A |. 8D0480    LEA EAX,DWORD PTR DS:[EAX+EAX*4] ; <== Temp = Temp * 5
0055395D |. 5A    POP EDX
0055395E |. 03D0    ADD EDX,EAX ; <== Value = Value + Temp
00553960 |. 8955 D8    MOV [LOCAL.10],EDX ; <== Value
00553963 |. FF75 E0    PUSH [LOCAL.8]
00553966 |. 68 DC3B5500 PUSH ExePW.00553BDC
0055396B |. 8D55 B0    LEA EDX,[LOCAL.20]
0055396E |. 8B45 D8    MOV EAX,[LOCAL.10] ; <== Value
00553971 |. E8 D255EBFF CALL ExePW.00408F48 ; <== Trace Into
===== Trace Into =====
..... Cutting .....

```

```
00408F53 |. E8 A4FFFFFF CALL ExePW.00408EFC ; <== Trace into
```

```

===== Trace Into =====
00408F12 |$ B9 0A000000 MOV ECX,0A ; <== DefaultValue : dV
00408F17 |> 52    PUSH EDX
00408F18 |. 56    PUSH ESI
00408F19 |> 31D2    /XOR EDX,EDX
00408F1B |. F7F1    |DIV ECX ; <== Value = Value / dV
00408F1D |. 4E    |DEC ESI
00408F1E |. 80C2 30    |ADD DL,30 ; <== Chart = (Value % dV) + 0x30
00408F21 |. 80FA 3A    |CMP DL,3A
00408F24 |. 72 03    |JB SHORT ExePW.00408F29
00408F26 |. 80C2 07    |ADD DL,7
00408F29 |> 8816    |MOV BYTE PTR DS:[ESI],DL ; <== Temp[2-i] = Chart
00408F2B |. 09C0    |OR EAX,EAX ; <== i++
00408F2D |.^ 75 EA    |JNZ SHORT ExePW.00408F19 ; <== Loop until i = 0x2
===== Trace Into =====
===== Trace Into =====

```

- Chuỗi mới được tạo thành này cũng được gắn luôn vào sau chuỗi ở trên . Va ký tự S[13] của chuỗi Fake Serial được gắn vào sau cùng .

- Qua quá trình này ta nhận xét dạng chuỗi Serial như sau : **0xx5-yyyy-AAAz** . Trong đó, xyz là các ký tự ngẫu nhiên trong khoảng **(0-9)** ; AAA là chuỗi được tính ra .

- Chuỗi U không tham gia vào quá trình mã hoá và có vô số chuỗi Serial hợp lệ .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErStEaM Serial : 0645-5705-2985 or 0925-2382-3041

III – KeyGen :

/Section 1/- Tạo ba giá trị ngẫu nhiên

SecI = rand() % 100;

SecII = rand() % 10000;

SecIII = rand() % 10;

/Section 2/- Tính chuỗi AAA theo công thức :

*Value = reaTemp[0] + reaTemp[1] + reaTemp[2] + reaTemp[3] + SecI + SecIII * 5;*

reaTemp[2-i] = (Value % 0xA) + 0x30;

/Section 3/- Nối kết các chuỗi lại với nhau . Trong đó *S[0][3] == "05"*

IV – End of Tut :

- Finished – **27/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.fairstars.com
Production :	FairStars Soft.
SoftWare :	FairStars Audio Converter 1.4.0.6
Copyright by :	Copyright © 2003-2004 FairStars Soft. All Rights Reserved.
Type :	Name / Product Number / Serial
Packed :	ASPack 2.12b -> Alexey Solodovnikov
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

FairStars Audio Converter 1.4.0.6

FairStars Audio Converter is a media file conversion tool to convert WAV, RealMedia(RM, RA, RAM, RMJ), AIFF, AU, Creative VOC, Amiga IFF/SVX, APE, FLAC, OGG, VQF, MP1, MP2, MP3, WMA, WMV, ASF to WMA, MP3, VQF, OGG, FLAC, APE, WAV formats. You can convert multiple files in a single batch, regardless of their source and target formats. In addition, the program includes a built-in player. The audio format conversions are performed without any temporary files, allowing high conversion speeds. Other features include normalization (adjust volume), support for ID3 tags and more.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **ASPack 2.12b -> Alexey Solodovnikov** . UnPACK và kiểm tra lại biết chương trình được viết bằng **Microsoft Visual C++ 6.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration Number Incorrect!!!!**" . Ta tìm được thông báo này tại địa chỉ : 00411C59 |. 68 C8044E00 PUSH _AudioCo.004E04C8 ; ASCII "Registration Number Incorrect!!!!"

- Dò ngược lên trên và đặt BreakPoint tại :
 004119E3 |. E8 32EF0300 CALL <JMP.&MFC42.#6334> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến quá trình kiểm tra chiều dài các dữ liệu nhập :

```

00411A8D |. 8378 F8 01 CMP DWORD PTR DS:[EAX-8],1 ; <== Name atleast 1 charts
00411A91 |. 7D 11 JGE SHORT _AudioCo.00411AA4
00411A93 |. 55 PUSH EBP
00411A94 |. 55 PUSH EBP
00411A95 |. 68 08054E00 PUSH _AudioCo.004E0508 ; ASCII "UserName Error!"
00411A9A |. E8 2BEA0300 CALL <JMP.&MFC42.#1200>
00411A9F |. E9 06020000 JMP _AudioCo.00411CAA
00411AA4 |> 8B8E 34130000 MOV ECX,DWORD PTR DS:[ESI+1334]
00411AAA |. 8D86 34130000 LEA EAX,DWORD PTR DS:[ESI+1334]
00411AB0 |. 894424 20 MOV DWORD PTR SS:[ESP+20],EAX
00411AB4 |. 8379 F8 08 CMP DWORD PTR DS:[ECX-8],8 ; <== Produc Number must be 8 charts
00411AB8 |. 74 11 JE SHORT _AudioCo.00411ACB
00411ABA |. 55 PUSH EBP
00411ABB |. 55 PUSH EBP
00411ABC |. 68 FC044E00 PUSH _AudioCo.004E04FC ; ASCII "PN Error!"
00411AC1 |. E8 04EA0300 CALL <JMP.&MFC42.#1200>
00411AC6 |. E9 DF010000 JMP _AudioCo.00411CAA
00411ACB |> 8B17 MOV EDX,DWORD PTR DS:[EDI]
00411ACD |. 8B42 F8 MOV EAX,DWORD PTR DS:[EDX-8]
00411AD0 |. 83F8 10 CMP EAX,10 ; <== Serial must be 16 charts
00411AD3 |. 0F85 D1010000 JNZ _AudioCo.00411CAA
  
```

- Kế đó, các ký tự trong 4 ô nhập Serial sẽ được nối lại với nhau và được mã hoá theo thuật toán MD5 để tạo chuỗi MD5Hash đầu tiên . Ta gọi là **CheckI** :

```

00411ADD |. E8 FE6B0100 CALL _AudioCo.004286E0
00411AE2 |. 8B3F MOV EDI,DWORD PTR DS:[EDI]
00411AE4 |. C78424 F00000>MOV DWORD PTR SS:[ESP+F0],3
00411AEF |. 33C0 XOR EAX,EAX
00411AF1 |> 8A0C07 /MOV CL,BYTE PTR DS:[EDI+EAX]
00411AF4 |. 884C04 2C |MOV BYTE PTR SS:[ESP+EAX+2C],CL
00411AF8 |. 40 |INC EAX
00411AF9 |. 83F8 10 |CMP EAX,10
00411AFC |.^ 7C F3 |JL SHORT _AudioCo.00411AF1
00411AFE |. 8D4424 2C LEA EAX,DWORD PTR SS:[ESP+2C]
00411B02 |. 6A 10 PUSH 10
00411B04 |. 50 PUSH EAX
00411B05 |. 8D4C24 54 LEA ECX,DWORD PTR SS:[ESP+54]
00411B09 |. E8 626C0100 CALL _AudioCo.00428770
00411B0E |. 8D4C24 3C LEA ECX,DWORD PTR SS:[ESP+3C]
00411B12 |. 51 PUSH ECX ; /Arg1
00411B13 |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50] ; |
00411B17 |. E8 046D0100 CALL _AudioCo.00428820 ; \_AudioCo.00428820
00411B1C |. 8D5424 3C LEA EDX,DWORD PTR SS:[ESP+3C]
00411B20 |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
  
```

```
00411B24 |. 52      PUSH EDX
00411B25 |. E8 D6EC0300 CALL <JMP.&MFC42.#537>
```

- Chuỗi **Production Number** sẽ được mã hoá thành chuỗi dài 16 ký tự :

```
00411B30 |. 50      PUSH EAX ; /Arg1
00411B31 |. C68424 F40000>MOV BYTE PTR SS:[ESP+F4],4 ; |
00411B39 |. E8 42FBFFFF CALL _AudioCo.00411680 ; \AudioCo.00411680
```

===== Encrypt =====

```
00411849 |. 6A 07    PUSH 7 ; <== Get 7 charts
0041184B |. 51      PUSH ECX ; <== from SECOND chart
0041184C |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10] ; <== of PN
00411850 |. E8 B1EF0300 CALL <JMP.&MFC42.#5710> ; <== Ripping : tStr
00411855 |. 8B5424 1C  MOV EDX,DWORD PTR SS:[ESP+1C] ; <== tStr
00411859 |. C64424 2C 03  MOV BYTE PTR SS:[ESP+2C],3
0041185E |. 52      PUSH EDX ; /s
0041185F |. FF15 80384B00 CALL DWORD PTR DS:[<&MSVCRT.atol>] ; \atol
```

===== NOTE =====

Với hàm **API MSVCRT.atol** ta xác định được rằng 7 ký tự này phải là các chữ số (0 – 9)

===== NOTE =====

```
00411865 |. 99      CDQ
00411866 |. B9 0A000000 MOV ECX,0A ; <== dV
0041186B |. 83C4 04  ADD ESP,4
0041186E |. F7F9      IDIV ECX ; <== Temp = tStrHEXValue % dV
00411870 |. 68 60595D00 PUSH _AudioCo.005D5960
00411875 |. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
00411879 |. 8BDA      MOV EBX,EDX
0041187B |. E8 80EF0300 CALL <JMP.&MFC42.#537>
00411880 |. 68 FC034E00 PUSH _AudioCo.004E03FC ; ASCII "FSCon100"
00411885 |. 8D4C24 1C  LEA ECX,DWORD PTR SS:[ESP+1C]
00411889 |. C64424 30 04  MOV BYTE PTR SS:[ESP+30],4
0041188E |. E8 6DEF0300 CALL <JMP.&MFC42.#537>
00411893 |. 68 F0034E00 PUSH _AudioCo.004E03F0 ; ASCII "163elisa"
00411898 |. 8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18]
0041189C |. C64424 30 05  MOV BYTE PTR SS:[ESP+30],5
004118A1 |. E8 5AEF0300 CALL <JMP.&MFC42.#537>
004118A6 |. C64424 2C 06  MOV BYTE PTR SS:[ESP+2C],6
004118AB |. 33F6      XOR ESI,ESI
004118AD |> 8B5424 08  /MOV EDX,DWORD PTR SS:[ESP+8] ; <== FN
004118B1 |. 8B4424 18  |MOV EAX,DWORD PTR SS:[ESP+18] ; <== dStrI
004118B5 |. 8A0C16      |MOV CL,BYTE PTR DS:[ESI+EDX] ; <== FN[i]
004118B8 |. 8A1406      |MOV DL,BYTE PTR DS:[ESI+EAX] ; <== dStrI[i]
004118BB |. 8B4424 14  |MOV EAX,DWORD PTR SS:[ESP+14]
004118BF |. 2BC6      |SUB EAX,ESI
004118C1 |. 8A40 07    |MOV AL,BYTE PTR DS:[EAX+7] ; <== dStrII[Len-i]
004118C4 |. 32C2      |XOR AL,DL ; <== Value = dStrII[Len-i] xor dStrI[i]
004118C6 |. 8D5424 0C  |LEA EDX,DWORD PTR SS:[ESP+C]
004118CA |. 32C1      |XOR AL,CL ; <== Value = Value xor FN[i]
004118CC |. 02C3      |ADD AL,BL ; <== Value = Value + Temp
004118CE |. 0FBEC8      |MOVSX ECX,AL
004118D1 |. 51      |PUSH ECX
004118D2 |. 68 EC034E00 |PUSH _AudioCo.004E03EC
```

```

004118D7 |. 52      |PUSH EDX
004118D8 |. E8 E1EB0300 |CALL <JMP.&MFC42.#2818> ; <== "%X"
004118DD |. 83C4 0C   |ADD ESP,0C
004118E0 |. 8D4424 0C  |LEA EAX,DWORD PTR SS:[ESP+C]
004118E4 |. 8D4C24 10  |LEA ECX,DWORD PTR SS:[ESP+10]
004118E8 |. 50      |PUSH EAX
004118E9 |. E8 78EF0300 |CALL <JMP.&MFC42.#939>
004118EE |. 46      |INC ESI ; <== i++
004118EF |. 83FE 08   |CMP ESI,8 ; <== while ( i < 8 )
004118F2 |.^ 7C B9    |JL SHORT _AudioCo.004118AD ; <== Continue Loop
----- Encrypt -----

```

- Chuỗi mới (16 ký tự) này cũng được mã hoá theo thuật toán MD5 để tạo chuỗi **CheckII** :

```

00411B6C |. E8 FF6B0100 CALL _AudioCo.00428770
00411B71 |. 8D4424 3C  LEA EAX,DWORD PTR SS:[ESP+3C]
00411B75 |. 8D4C24 4C  LEA ECX,DWORD PTR SS:[ESP+4C]
00411B79 |. 50      PUSH EAX ; /Arg1
00411B7A |. E8 A16C0100 CALL _AudioCo.00428820 ; \_AudioCo.00428820
00411B7F |. 8D4C24 3C  LEA ECX,DWORD PTR SS:[ESP+3C]
00411B83 |. 51      PUSH ECX
00411B84 |. 8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18]
00411B88 |. E8 73EC0300 CALL <JMP.&MFC42.#537>

```

- Cuối cùng hai chuỗi **CheckI** và **CheckII** sẽ được so sánh với nhau :

```

00411BAE |>/8A0C02   /MOV CL,BYTE PTR DS:[EDX+EAX] ; <== CheckI[i]
00411BB1 |.|3A08     |CMP CL,BYTE PTR DS:[EAX]      ; <== if ( CheckI[i] == CheckII[i] )
00411BB3 |.|0F85 9C000000 JNZ _AudioCo.00411C55      ; <== Continue Check
00411BB9 |.|45      |INC EBP
00411BBA |.|40      |INC EAX
00411BBB |.|3BEF     |CMP EBP,EDI
00411BBD |.^7C EF    |JL SHORT _AudioCo.00411BAE

```

- Quá trình này được diễn giải lại như sau :

- 1 – Chuỗi Serial được mã hoá bằng thuật toán MD5Hash để tạo chuỗi CheckI
- 2 – Chuỗi PN được mã hoá để cho ra chuỗi trung gian dài 16 ký tự .
- 3 – Chuỗi này cũng được mã hoá MD5Hash để tạo chuỗi CheckII
- 4 – Chuỗi CheckI và chuỗi CheckII phải bằng nhau .

- Như vậy ta thấy ngay rằng, quá trình quan trọng nhất ở đây là quá trình mã hoá chuỗi PN để cho ra chuỗi trung gian dài 16 ký tự . Và nếu ta xác nhận rằng chuỗi trung gian này là chuỗi Serial thì quá trình mã hoá chính thức là quá trình chuyển đổi chuỗi PN thành chuỗi Serial .

- Chuỗi PN là chuỗi ngẫu nhiên dài 8 ký tự . Như vậy chuỗi trình có vô số cặp chuỗi PN / S hợp lệ . Chuỗi U không tham gia vào quá trình mã hoá.

- Tuy vậy, khi khởi động lại chương trình ta vẫn nhận thấy rằng chương trình vẫn xuất hiện thông báo yêu cầu nhập số đăng ký dù trong quá trình trên đã xuất hiện thông báo "**Registered OK!**" . Như vậy, ta nghĩ ngày rằng, chương trình sẽ kiểm tra thêm điều kiện khác . Vì thế ta cần phải tìm được đoạn CODE kiểm tra lại tính chính xác của PN / S khi chương trình khởi động .

- Ta biết chương trình sẽ lưu các thông số đăng ký vào file **conreg.ini**, như vậy ta tìm đến đoạn CODE

đọc file này khi khởi động, ta tìm được địa chỉ này :
 004207C7 |. 68 34054E00 PUSH _AudioCo.004E0534

; ASCII "conreg.ini"

- Đặt BP . Chạy lại chương trình, chương trình sẽ dừng lại tại đây . Tiếp tục xem xét quá trình, nhận thấy quá trình mã hoá và so sánh hoàn toàn giống như ở trên nhưng nằm trong một FUNCTION khác . Sau khi kết thúc quá trình so sánh như ở trên, chương trình tiếp đến một đoạn so sánh mới :

00420F43 > \8A02	MOV AL,BYTE PTR DS:[EDX]	; <== FN[0]
00420F45 . 3C 4E	CMP AL,4E	; <== if (FN[0] == "N")
00420F47 . 74 08	JE SHORT _AudioCo.00420F51	; <== or
00420F49 . 3C 46	CMP AL,46	; <== if (FN[0] == "F")
00420F4B . 74 04	JE SHORT _AudioCo.00420F51	; <== or
00420F4D . 3C 55	CMP AL,55	; <== if (FN[0] == "U")
00420F4F .^ 75 E0	JNZ SHORT _AudioCo.00420F31	; <== and
00420F51 > 8BCD	MOV ECX,EBP	
00420F53 > 8A0411	/MOV AL,BYTE PTR DS:[ECX+EDX]	; <== / from here
00420F56 . 3C 30	CMP AL,30	; <== all remain
00420F58 .^ 7C D7	JL SHORT _AudioCo.00420F31	; <== FN's charts
00420F5A . 3C 39	CMP AL,39	; <== must be
00420F5C .^ 7F D3	JG SHORT _AudioCo.00420F31	; <== in (0 - 9)
00420F5E . 41	INC ECX	; <== SOFTWARE
00420F5F . 83F9 08	CMP ECX,8	; <== REGISTED !!!!
00420F62 .^ 7C EF	JL SHORT _AudioCo.00420F53	; <== \ to here

- Như vậy, ngoài các điều đã được phân tích trong quá trình tạo cặp chuỗi **PN / S** thì điều kiện mới được thêm vào là ký tự đầu tiên của chuỗi **PN** phải là 1 trong các ký tự “**F**”, “**N**”, “**U**” .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 6F17-1F41-4141-443D
 PN : N1380986

III – End of Tut :

- Finished – September 16, 2004

Reverse Engineering Association

SoftWare

Homepage :	http://www.softhead.com
Production :	Ixis Research, Ltd.
SoftWare :	File and Folder Protector 1.89
Copyright by :	Copyright © 2004 Ixis Research, Ltd. All Rights Reserved.
Type :	Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

File and Folder Protector 1.89

File and Folder Protector is intended for controlling access to files and folders situated on local media of Windows 95/98/ME/NT/2000/XP at Windows kernel level. It enables you to control access to certain files and folders by using the password requirement, or to hide them securely from viewing and searching. The program does not modify your media: in protecting your files, it just uses a high-reliability VXD (SYS for Windows NT/2000/XP) driver working at Windows kernel level. This guarantees that File and Folder Protector will never lose your data, as may happen if you use other file-and folder-protecting software.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Registration code is invalid!**" . Ta tìm thấy chuỗi thông báo này tại địa chỉ :
0049AC87 |. B8 F8AC4900 MOV EAX,FFP.0049ACF8 ; |ASCII "The registration code is incorrect."
- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :
0049AC48 |. E8 C3FFFF CALL FFP.0049AB10 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :
0049AC54 |. E8 33F5FFFF CALL FFP.0049A18C ; <== Trace Into
- Đầu tiên là quá trình kiểm tra chiều dài chuỗi S nhập :
0049A1BD |. 8B45 FC MOV EAX,[LOCAL.1] ; <== S
0049A1C0 |. E8 1BA8F6FF CALL FFP.004049E0 ; <== LenS
0049A1C5 |. 83F8 10 CMP EAX,10 ; <== LenS must be 16 charts
0049A1C8 |. 0F85 A4000000 JNZ FFP.0049A272
- Kế đó, chương trình kiểm tra các ký tự của chuỗi S nhập phải nằm trong khoảng (0 – 9) :
0049A1D2 |. B9 08000000 MOV ECX,8 ; <== get 8 charts
0049A1D7 |. BA 01000000 MOV EDX,1 ; <== from the FIRST chart
0049A1DC |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of Serial
0049A1DF |. E8 5CAAF6FF CALL FFP.00404C40 ; <== Ripping : fstStr
0049A1E4 |. 8D45 F0 LEA EAX,[LOCAL.4]
0049A1E7 |. 50 PUSH EAX
0049A1E8 |. B9 08000000 MOV ECX,8 ; <== get 8 charts
0049A1ED |. BA 09000000 MOV EDX,9 ; <== from the NINETH chart
0049A1F2 |. 8B45 FC MOV EAX,[LOCAL.1] ; <== of Serial
0049A1F5 |. E8 46AAF6FF CALL FFP.00404C40 ; <== Ripping : sndStr
0049A1FA |. 8D55 F8 LEA EDX,[LOCAL.2]
0049A1FD |. 8B45 F4 MOV EAX,[LOCAL.3]
0049A200 |. E8 878FF6FF CALL FFP.0040318C ; <== All charts must be (0-9)
0049A205 |. 837D F8 00 CMP [LOCAL.2],0
0049A209 |. 75 67 JNZ SHORT FFP.0049A272
0049A20B |. 8D55 F8 LEA EDX,[LOCAL.2]

```

0049A20E |. 8B45 F0    MOV EAX,[LOCAL.4] ; <== sndStr
0049A211 |. E8 768FF6FF CALL FFP.0040318C ; <== All charts must be (0-9)
0049A216 |. 837D F8 00  CMP [LOCAL.2],0
0049A21A |. 75 56      JNZ SHORT FFP.0049A272

```

- Tiếp đến, chương trình kiểm tra các ký tự của chuỗi S nhập để đảm bảo rằng không có ký tự nào được phép lặp lại quá 5 lần :

```

0049A21C |. BE 30000000  MOV ESI,30
0049A221 |> 33C0      /XOR EAX,EAX
0049A223 |. 8945 F8    |MOV [LOCAL.2],EAX
0049A226 |. 8B45 FC    |MOV EAX,[LOCAL.1]
0049A229 |. E8 B2A7F6FF |CALL FFP.004049E0
0049A22E |. 85C0      |TEST EAX,EAX
0049A230 |. 7E 17      |JLE SHORT FFP.0049A249
0049A232 |. BA 01000000  |MOV EDX,1
0049A237 |> 8BCE      |/MOV ECX,ESI
0049A239 |. 8B7D FC    ||MOV EDI,[LOCAL.1]
0049A23C |. 3A4C17 FF   ||CMP CL,BYTE PTR DS:[EDI+EDX-1]
0049A240 |. 75 03      ||JNZ SHORT FFP.0049A245
0049A242 |. FF45 F8    ||INC [LOCAL.2]
0049A245 |> 42        ||INC EDX
0049A246 |. 48        ||DEC EAX
0049A247 |.^ 75 EE    |JNZ SHORT FFP.0049A237
0049A249 |> 837D F8 05  |CMP [LOCAL.2],5
0049A24D |. 7E 09      |JLE SHORT FFP.0049A258
0049A24F |. C745 F8 FFFFF>|MOV [LOCAL.2],-1
0049A256 |. EB 06      |JMP SHORT FFP.0049A25E
0049A258 |> 46        |INC ESI
0049A259 |. 83FE 3A    |CMP ESI,3A
0049A25C |.^ 75 C3    |JNZ SHORT FFP.0049A221
0049A25E |> 837D F8 00  CMP [LOCAL.2],0
0049A262 |. 7C 0E      JL SHORT FFP.0049A272

```

- Quá trình kiểm tra cuối cùng diễn ra :

```
0049A267 |. E8 B0FDFFFF CALL FFP.0049A01C ; <== Last Check
```

===== Last Check =====

```

0049A040 |. 8A13      MOV DL,BYTE PTR DS:[EBX] ; <== S[0]
0049A042 |. E8 C1A8F6FF CALL FFP.00404908
0049A047 |. 8B45 FC    MOV EAX,[LOCAL.1]
0049A04A |. E8 19EFF6FF CALL FFP.00408F68 ; <== TempI = S[0] - 0x30
0049A04F |. 8BF0      MOV ESI,EAX ; <== TempI
0049A051 |. 8D45 F8    LEA EAX,[LOCAL.2]
0049A054 |. 8A53 01    MOV DL,BYTE PTR DS:[EBX+1] ; <== S[1]
0049A057 |. E8 ACA8F6FF CALL FFP.00404908
0049A05C |. 8B45 F8    MOV EAX,[LOCAL.2]
0049A05F |. E8 04EFF6FF CALL FFP.00408F68 ; <== TempII = S[1] - 0x30
0049A064 |. 03F0      ADD ESI,EAX ; <== Value = TempI + TempII
0049A066 |. 8D45 F4    LEA EAX,[LOCAL.3]
0049A069 |. 8A53 02    MOV DL,BYTE PTR DS:[EBX+2] ; <== S[2]
0049A06C |. E8 97A8F6FF CALL FFP.00404908
0049A071 |. 8B45 F4    MOV EAX,[LOCAL.3]

```

```

0049A074 |. E8 EFEEF6FF CALL FFP.00408F68 ; <== TempIII = S[2] - 0x30
0049A079 |. 03F0      ADD ESI,EAX      ; <== Value = Value + TempIII
0049A07B |. 8D45 F0    LEA EAX,[LOCAL.4]
0049A07E |. 8A53 03    MOV DL,BYTE PTR DS:[EBX+3] ; <== S[3]
0049A081 |. E8 82A8F6FF CALL FFP.00404908
0049A086 |. 8B45 F0    MOV EAX,[LOCAL.4]
0049A089 |. E8 DAEEF6FF CALL FFP.00408F68 ; <== TempIV = S[3] - 0x30
0049A08E |. 03F0      ADD ESI,EAX      ; <== Value = Value + TempIV
0049A090 |. A1 74864A00 MOV EAX,DWORD PTR DS:[4A8674] ; <== Add. of DefaultValueI : dVI
0049A095 |. 3B30      CMP ESI,DWORD PTR DS:[EAX] ; <== if ( Value == dVI )
0049A097 0F85 BA000000 JNZ FFP.0049A157 ; <== Continue Check
0049A09D |. 8D45 EC    LEA EAX,[LOCAL.5]
0049A0A0 |. 8A53 04    MOV DL,BYTE PTR DS:[EBX+4] ; <== S[4]
0049A0A3 |. E8 60A8F6FF CALL FFP.00404908
0049A0A8 |. 8B45 EC    MOV EAX,[LOCAL.5]
0049A0AB |. E8 B8EEF6FF CALL FFP.00408F68 ; <== TempI = S[4] - 0x30
0049A0B0 |. 8BF0      MOV ESI,EAX      ; <== TempI
0049A0B2 |. 8D45 E8    LEA EAX,[LOCAL.6]
0049A0B5 |. 8A53 07    MOV DL,BYTE PTR DS:[EBX+7] ; <== S[7]
0049A0B8 |. E8 4BA8F6FF CALL FFP.00404908
0049A0BD |. 8B45 E8    MOV EAX,[LOCAL.6]
0049A0C0 |. E8 A3EEF6FF CALL FFP.00408F68 ; <== TempII = S[7] - 0x30
0049A0C5 |. 03F0      ADD ESI,EAX      ; <== Value = TempI + TempII
0049A0C7 |. 8D45 E4    LEA EAX,[LOCAL.7]
0049A0CA |. 8A53 0A    MOV DL,BYTE PTR DS:[EBX+A] ; <== S[10]
0049A0CD |. E8 36A8F6FF CALL FFP.00404908
0049A0D2 |. 8B45 E4    MOV EAX,[LOCAL.7]
0049A0D5 |. E8 8EEE6FF CALL FFP.00408F68 ; <== TempIII = S[10] - 0x30
0049A0DA |. 03F0      ADD ESI,EAX      ; <== Value = Value + TempIII
0049A0DC |. 8D45 E0    LEA EAX,[LOCAL.8]
0049A0DF |. 8A53 0D    MOV DL,BYTE PTR DS:[EBX+D] ; <== S[13]
0049A0E2 |. E8 21A8F6FF CALL FFP.00404908
0049A0E7 |. 8B45 E0    MOV EAX,[LOCAL.8]
0049A0EA |. E8 79EEF6FF CALL FFP.00408F68 ; <== TempIV = S[13] - 0x30
0049A0EF |. 03F0      ADD ESI,EAX      ; <== Value = Value + TempIV
0049A0F1 |. A1 AC864A00 MOV EAX,DWORD PTR DS:[4A86AC] ; <== Add. of DefaultValueII : dVII
0049A0F6 |. 3B30      CMP ESI,DWORD PTR DS:[EAX] ; <== if ( Value == dVII )
0049A0F8 75 5D      JNZ SHORT FFP.0049A157 ; <== Continue Check
0049A0FA |. 8D45 DC    LEA EAX,[LOCAL.9]
0049A0FD |. 8A53 06    MOV DL,BYTE PTR DS:[EBX+6] ; <== S[6]
0049A100 |. E8 03A8F6FF CALL FFP.00404908
0049A105 |. 8B45 DC    MOV EAX,[LOCAL.9]
0049A108 |. E8 5BEEF6FF CALL FFP.00408F68 ; <== TempI = S[6] - 0x30
0049A10D |. 8BF0      MOV ESI,EAX      ; <== TempI
0049A10F |. 8D45 D8    LEA EAX,[LOCAL.10]
0049A112 |. 8A53 09    MOV DL,BYTE PTR DS:[EBX+9] ; <== S[9]
0049A115 |. E8 EEA7F6FF CALL FFP.00404908
0049A11A |. 8B45 D8    MOV EAX,[LOCAL.10]
0049A11D |. E8 46EEF6FF CALL FFP.00408F68 ; <== TempII = S[9] - 0x30

```

```

0049A122 |. 03F0      ADD ESI,EAX           ; <== Value = TempI + TempII
0049A124 |. 8D45 D4    LEA EAX,[LOCAL.11]
0049A127 |. 8A53 0C    MOV DL,BYTE PTR DS:[EBX+C]   ; <== S[12]
0049A12A |. E8 D9A7F6FF CALL FFP.00404908
0049A12F |. 8B45 D4    MOV EAX,[LOCAL.11]
0049A132 |. E8 31EEF6FF CALL FFP.00408F68       ; <== TempIII = S[12] - 0x30
0049A137 |. 03F0      ADD ESI,EAX           ; <== Value = Value + TempIII
0049A139 |. 8D45 D0    LEA EAX,[LOCAL.12]
0049A13C |. 8A53 0F    MOV DL,BYTE PTR DS:[EBX+F]   ; <== S[15]
0049A13F |. E8 C4A7F6FF CALL FFP.00404908
0049A144 |. 8B45 D0    MOV EAX,[LOCAL.12]
0049A147 |. E8 1CEEF6FF CALL FFP.00408F68       ; <== TempIV = S[15] - 0x30
0049A14C |. 03F0      ADD ESI,EAX           ; <== Value = Value + TempIV
0049A14E|.A1 A0834A00 MOV EAX,DWORD PTR DS:[4A83A0] ; <== Add. of DefaultValueIII : dVIII
0049A153 |. 3B30      CMP ESI,DWORD PTR DS:[EAX]     ; <== if ( Value == dVIII )
0049A155 74 04      JE SHORT FFP.0049A15B       ; <== ConGrat !!!!
-----Last Check -----

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 6826835781691921

III – End of Tut :

- Finished – September 14, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.flash2x.net
Production :	Flash2X.
SoftWare :	Flash2X EXE Packager 1.0.1
Copyright by :	Copyright © 2003 Flash2X. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

Flash2X EXE Packager 1.0.1

Flash2X EXE Packager is able to package more than one flash movies into a single executable file. The executable file is a flash player itself. When users build the executable files, they can decide whether the watchers of their flash package files are allowed to get the SWF files or not. It a very useful tool for flash authors to distribute and protect their works.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "To enable the registration, please restart the program." . :

0048F8FE . BA 90F94800 MOV EDX,flash2ex.0048F990 ; ASCII "To enable the registration, please restart the program."

- Tại đây ta biết được chương trình sẽ lưu các DATA này vào REG. Ta có thể đặt BreakPoint tại các hàm API liên quan đến REG . Tuy nhiên trong quá trình tìm kiếm chuỗi ta tìm được các dòng sau nằm trong cùng một FUNCTION :

004B130F . BA DC144B00	MOV EDX,flash2ex.004B14DC	; ASCII "RegName"
004B130F . BA DC144B00	MOV EDX,flash2ex.004B14DC	; ASCII "RegName"
004B133B > \BA EC144B00	MOV EDX,flash2ex.004B14EC	; ASCII "RegCode"
004B134E . BA EC144B00	MOV EDX,flash2ex.004B14EC	; ASCII "RegCode"

- Truy ngược về FUNTION này, dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

004B12A8 . E8 3333F5FF CALL flash2ex.004045E0 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

004B13B6 . E8 4D81FFFF	CALL flash2ex.004A9508	; <== Trace Into here
004B13BB . 8B55 EC	MOV EDX,DWORD PTR SS:[EBP-14]	; <== Real Serial
004B13BE . 8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	; <== Fake Serial
004B13C1 . E8 7E35F5FF	CALL flash2ex.00404944	; <== Compare

- Trace into để xem xét quá trình mã hoá :

004A9563 > /8B45 FC	/MOV EAX,[LOCAL.1]	; <== / from here
004A9566 . 0FB64430 FF	MOVZX EAX,BYTE PTR DS:[EAX+ESI-1]	; <== this is
004A956B . 03F8	ADD EDI,EAX	; <== Cummulate
004A956D . 46	INC ESI	; <== all charts
004A956E . 4B	DEC EBX	; <== of UserInput
004A956F .^\75 F2	\JNZ SHORT flash2ex.004A9563	; <== \ to here
004A9571 > 8D45 EC	LEA EAX,[LOCAL.5]	
004A9574 . 50	PUSH EAX	
004A9575 . 8BC7	MOV EAX,EDI	; <== CumValue
004A9577 . B9 3E000000	MOV ECX,3E	; <== dV
004A957C . 99	CDQ	
004A957D . F7F9	IDIV ECX	; <== Temp = CumValue % dV
004A957F . 8BDA	MOV EBX,EDX	; <== Temp
004A9581 . 8BCB	MOV ECX,EBX	; <== Temp
004A9583 . 41	INC ECX	; <== Temp ++
004A9584 . BA 01000000	MOV EDX,1	
004A9589 . 8B45 F0	MOV EAX,[LOCAL.4]	; <== DefaultString : dStr
004A958C . E8 C7B4F5FF	CALL flash2ex.00404A58	; <== Rip "Temp" Charts : tStr
004A9591 . 8D45 E8	LEA EAX,[LOCAL.6]	

```

004A9594 |. 50      PUSH EAX
004A9595 |. 8B45 F0    MOV EAX,[LOCAL.4] ; <== dStr
004A9598 |. E8 63B2F5FF CALL flash2ex.00404800 ; <== Len.dStr
004A959D |. 8BC8      MOV ECX,EAX ; <== Len.dStr
004A959F |. 8BD3      MOV EDX,EBX ; <== Temp
004A95A1 |. 83C2 02    ADD EDX,2 ; <== Temp +=2
004A95A4 |. 8B45 F0    MOV EAX,[LOCAL.4] ; <== dStr
004A95A7 |. E8 ACB4F5FF CALL flash2ex.00404A58 ; <== Remain charts of dStr : rStr
004A95AC |. 8D45 F0    LEA EAX,[LOCAL.4] ; <== change dStr by new one
004A95AF |. 8B4D EC    MOV ECX,[LOCAL.5] ; <== tStr
004A95B2 |. 8B55 E8    MOV EDX,[LOCAL.6] ; <== rStr
004A95B5 |. E8 92B2F5FF CALL flash2ex.0040484C ; <== Reverse String : revStr
004A95BA |. 8D45 FC    LEA EAX,[LOCAL.1] ; <== revStr
004A95BD |. 8B55 F0    MOV EDX,[LOCAL.4] ; <== Concat(U,revStr) : catStr
004A95C0 |. E8 43B2F5FF CALL flash2ex.00404808
004A95C5 |. 8D45 FC    LEA EAX,[LOCAL.1]
004A95C8 |. 50      PUSH EAX
004A95C9 |. B9 14000000 MOV ECX,14 ; <== get 20 charts
004A95CE |. BA 01000000 MOV EDX,1 ; <== from the FIRST charst
004A95D3 |. 8B45 FC    MOV EAX,[LOCAL.1] ; <== of catStr
004A95D6 |. E8 7DB4F5FF CALL flash2ex.00404A58 ; <== Ripping : mainStr
004A95DB |. 8D45 F4    LEA EAX,[LOCAL.3]
004A95DE |. E8 65AFF5FF CALL flash2ex.00404548
004A95E3 |. 33FF      XOR EDI,EDI
004A95E5 |. 8B45 FC    MOV EAX,[LOCAL.1] ; <== mainStr
004A95E8 |. E8 13B2F5FF CALL flash2ex.00404800 ; <== Len.mainStr
004A95ED |. 8BD8      MOV EBX,EAX
004A95EF |. 85DB      TEST EBX,EBX
004A95F1 |. 7E 36      JLE SHORT flash2ex.004A9629
004A95F3 |. BE 01000000 MOV ESI,1
004A95F8 |> 8B45 FC    /MOV EAX,[LOCAL.1] ; <== mainStr
004A95FB |. 0FB64430 FF  |MOVZX EAX,BYTE PTR DS:[EAX+ESI-1] ; <== mainStr[i]
004A9600 |. 03F8      |ADD EDI,EAX ; <== CumValue = CumValue + mainStr[i]
004A9602 |. 8BC7      |MOV EAX,EDI ; <== Temp = CumValue
004A9604 |. B9 3E000000 |MOV ECX,3E ; <== dV
004A9609 |. 99      |CDQ
004A960A |. F7F9      |IDIV ECX ; <== Temp = Temp & 0x3E
004A960C |. 8B45 F0    |MOV EAX,[LOCAL.4] ; <== revStr
004A960F |. 8A1410    |MOV DL,BYTE PTR DS:[EAX+EDX] ; <== Temp = revStr[Temp]
004A9612 |. 8D45 D8    |LEA EAX,[LOCAL.10] ; <== / from here
004A9615 |. E8 0EB1F5FF |CALL flash2ex.00404728 ; <== |
004A961A |. 8B55 D8    |MOV EDX,[LOCAL.10] ; <== | S[i] = Temp
004A961D |. 8D45 F4    |LEA EAX,[LOCAL.3] ; <== |
004A9620 |. E8 E3B1F5FF |CALL flash2ex.00404808 ; <== \ to here
004A9625 |. 46      |INC ESI ; <== i++
004A9626 |. 4B      |DEC EBX ; <== Len.mainStr --
004A9627 |.^ 75 CF    |JNZ SHORT flash2ex.004A95F8 ; Loop until Len.mainStr == 0x0

```

- Quá trình này được diễn giải gọn như sau :

- 1 – Tính giá trị cộng dồn các ký tự của chuỗi U nhập .
- 2 – Tính phần dư của phép chia giá trị này với **0x3E** (chiều dài chuỗi mặc định)
- 3 – Cắt chuỗi thứ I có chiều dài đúng bằng giá trị phần dư của phép chia .

- 4 – Chuỗi thứ II là phần còn lại .
 5 - Nối hai chuỗi lại với nhau nhưng đảo ngược vị trí ==> **Reverse String : revStr**
 6 – Nối chuỗi này vào sau chuỗi U nhập ==> **Concat(U,revStr) : catStr**
 7 – Cắt chuỗi dùng để mã hoá có chiều dài 20 ký tự dựa trên chuỗi **catStr** ==> **Ripping : mainStr**
 8 – Tính cộng dồn giá trị của chuỗi **mainStr** . Mỗi lần cộng dồn, giá trị được chia cho **0x3E** . Phần dư của phép chia này chính là địa chỉ của ký tự tương ứng trong chuỗi **revStr** .
 9 – Kết thúc vòng lặp có chuỗi Serial dài 20 ký tự .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : b7zK0ak5Dot07zO8Kx6a

III – End of Tut :

- Finished – September 13, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.ccountry.net
Production :	Gary Dix.
SoftWare :	FontMap 2.37
Copyright by :	Copyright © 2002-2004 Gary Dix. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Nothing found [Overlay] *
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

FontMap 2.37

FontMap may be used to view and print both installed and external fonts; it can display a table, individual characters, a keyboard view, font description, or whatever text you wish to enter. Individual characters can be displayed in a "full-screen" view and custom table parameters can be set to "zoom" in on a selected portion of a font. National character sets and unicode characters available for a font may also be displayed.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Nothing found [Overlay]** *
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Sorry, your registration key is wrong. Please check it and try again**" . Tuy nhiên khi ta tìm đến địa chỉ của thông báo này thì đây chưa phải là FUNTION chính . Tìm tiếp và ta gặp dòng thông báo đăng ký thành công tại :
00403531 . 68 09E54000 PUSH FONTMAP.0040E509 ; |Text = "Thank you for registering!"
- Dò ngược lên trên, ta thấy hàm **GetDlgItemTextA** nên đặt BreakPoint tại đây :
004033C9 . E8 FA950000 CALL <JMP.&USER32.GetDlgItemTextA> ; \GetDlgItemTextA

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.

Trace xuống chút ta đến quá trình kiểm tra chiều dài Serial nhập :

```
004033D3 . E8 00950000 CALL <JMP.&KERNEL32.lstrlenA> ; \lstrlenA
004033D8 . 83F8 08 CMP EAX,8 ; <== LenS must be 8 charts
004033DB . 0F85 5C010000 JNZ FONTMAP.0040353D
```

- Trace tiếp ta đến :

```
004033FA . E8 12FFFFFF CALL FONTMAP.00403311 ; \FONTMAP.00403311
----- Encrypt -----
0040331A |. BB 78563412 MOV EBX,12345678 ; <== DefaultValue : dV
0040331F |. C745 FC F8E44>MOV [LOCAL.1],FONTMAP.0040E4F8 ; ASCII
"ABCDEFGHPQRSWXYZ"
00403326 |. 68 A8F84000 PUSH FONTMAP.0040F8A8 ; /String = "Moonbaby"
0040332B |. E8 A8950000 CALL <JMP.&KERNEL32.lstrlenA> ; \lstrlenA
00403330 |. 8BD0 MOV EDX,EAX
00403332 |. 33C0 XOR EAX,EAX ; <== i = 0
00403334 |. 3BD0 CMP EDX,EAX
00403336 |. 76 1D JBE SHORT FONTMAP.00403355
00403338 |> 0FBEE8 A8F840>/MOVsx ECX,byte PTR DS:[EAX+40F8A8] ; <== U[i]
0040333F |. 83E1 3F |AND ECX,3F ; <== Temp = U[i] and 0x3F
00403342 |. 33D9 |XOR EBX,ECX ; <== dV = dV xor Temp
00403344 |. 8BCB |MOV ECX,EBX ; <== Temp = dV
00403346 |. C1E1 06 |SHL ECX,6 ; <== Temp = dV * 0x40
00403349 |. C1EB 1A |SHR EBX,1A ; <== dV = dV / 0x4000000
0040334C |. 0BCB |OR ECX,EBX ; <== Temp = Temp or dV
0040334E |. 8BD9 |MOV EBX,ECX ; <== dV = Temp
00403350 |. 40 |INC EAX ; <== i++
00403351 |. 3BD0 |CMP EDX,EAX ; <== while ( i < LenU )
00403353 |.^ 77 E3 |JA SHORT FONTMAP.00403338 ; <== Continue Loop
00403355 |> 33C0 XOR EAX,EAX
00403357 |> 8BD3 /MOV EDX,EBX ; <== dV
00403359 |. 83E2 0F |AND EDX,0F ; <== Value = dV and 0x0F
0040335C |. 8B4D FC |MOV ECX,[LOCAL.1] ; <== DefaultString : dStr
0040335F |. 8A1411 |MOV DL,byte PTR DS:[ECX+EDX] ; <== Value = dStr[Value]
00403362 |. 881406 |MOV byte PTR DS:[ESI+EAX],DL ; <== Serial[i] = Value
00403365 |. C1EB 04 |SHR EBX,4 ; <== dV = dV / 0x10
00403368 |. 40 |INC EAX ; <== i++
00403369 |. 83F8 08 |CMP EAX,8 ; <== while ( i < 8 )
0040336C |.^ 72 E9 |JB SHORT FONTMAP.00403357 ; <== Continue Loop
----- Encrypt -----
004033FF . 59 POP ECX ; FONTMAP.0040F59B
00403400 . 68 A0F64000 PUSH FONTMAP.0040F6A0 ; /String2 = "12345678"
00403405 . 68 9BF54000 PUSH FONTMAP.0040F59B ; |String1 = "PWSAZWEB"
0040340A . E8 05950000 CALL <JMP.&KERNEL32.lstrcmpA> ; \lstrcmpA
```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : PWSAZWEB

III – KeyGen :

- /Section I /- Tính giá trị dựa trên giá trị mặc định **DefaultValue = 0x12345678**
- /Section II /- Tính chuỗi Serial dựa trên chuỗi mặc định "**ABCDEFGHIJKLMNPQRSTUVWXYZ**"

IV – End of Tut :

- Finished – ***August 15, 2004***

Reverse Engineering Association

SoftWare

Homepage :	http://www.freshdevices.com
Production :	Freshdevices Corp.
SoftWare :	FreshDiagnose 6.70
Copyright by :	Copyright © 2001-2004 Freshdevices Corp. All Rights Reserved.
Type :	Name / Serial
Packed :	ASPack 2.12 -> Alexey Solodovnikov
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	Manual
Request :	Correct Serial

FreshDiagnose 6.70

FRESH DIAGNOSE is an utility designed to analyze and benchmark your computer system. It can analyze and benchmark many kinds of hardware, such as CPU performance, hard disk performance, video system information, mainboard information, and many more!

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **ASPack 2.12 -> Alexey Solodovnikov** . UnPACK và kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta không nhận được thông báo gì . Tuy nhiên trong quá trình tìm kiếm chuỗi ta tìm được thông báo :
00561F2F > \B8 74215600 MOV EAX,unpack.00562174 ; ASCII "FreshDiagnose has been registered successfully."
- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :
00561EB2 . E8 1176F0FF CALL unpack.004694C8 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :
00561EF2 . E8 A5FDFFFF CALL unpack.00561C9C ; <== Trace Into
----- **Trace Into** -----
- Chuỗi Serial đầu tiên là chuỗi Serial không hợp lệ . Chỉ có 1 chuỗi Serial duy nhất :

```

00561D34 |>/8D4D F0    /LEA ECX,[LOCAL.4]
00561D37 |.|0FBFD6    |MOVsx EDX,SI
00561D3A |.|8B45 FC    |MOV EAX,[LOCAL.1]
00561D3D |.|8B80 34030000 |MOV EAX,DWORD PTR DS:[EAX+334]
00561D43 |.|8B80 20020000 |MOV EAX,DWORD PTR DS:[EAX+220]
00561D49 |.|8B38      |MOV EDI,DWORD PTR DS:[EAX]
00561D4B |.|FF57 0C    |CALL DWORD PTR DS:[EDI+C] ; <== Licence of Old Version
00561D4E |.|8B55 F0    |MOV EDX,[LOCAL.4] ; <== RealSerial
00561D51 |.|A1 A4115800 |MOV EAX,DWORD PTR DS:[5811A4] ; <== Fake Serial
00561D56 |.|E8 5D34EAFF |CALL unpack.004051B8 ; <== Compare
00561D5B |.|75 0A      |JNZ SHORT unpack.00561D67
00561D5D |.|C705 98115800>|MOV DWORD PTR DS:[581198],-1
00561D67 |>|46      |INC ESI
00561D68 |.|66:FFCB    |DEC BX ; <== 1 Licence
00561D6B |.^|75 C7    |JNZ SHORT unpack.00561D34

```

- 500 chuỗi Serial của **Personal License**

```

00561D95 |>/8D4D EC    /LEA ECX,[LOCAL.5]
00561D98 |.|0FBFD6    |MOVsx EDX,SI
00561D9B |.|8B45 FC    |MOV EAX,[LOCAL.1]
00561D9E |.|8B80 2C030000 |MOV EAX,DWORD PTR DS:[EAX+32C]
00561DA4 |.|8B80 20020000 |MOV EAX,DWORD PTR DS:[EAX+220]
00561DAA |.|8B38      |MOV EDI,DWORD PTR DS:[EAX]
00561DAC |.|FF57 0C    |CALL DWORD PTR DS:[EDI+C] ; <== Personal License
00561DAF |.|8B55 EC    |MOV EDX,[LOCAL.5] ; <== RealSerial
00561DB2 |.|A1 A4115800 |MOV EAX,DWORD PTR DS:[5811A4] ; <== Fake Serial
00561DB7 |.|E8 FC33EAFF |CALL unpack.004051B8 ; <== Compare
00561DBC |.|75 0A      |JNZ SHORT unpack.00561DC8
00561DBE |.|C705 90115800>|MOV DWORD PTR DS:[581190],-1
00561DC8 |>|46      |INC ESI
00561DC9 |.|66:FFCB    |DEC BX ; <== 500 Licences
00561DCC |.^|75 C7    |JNZ SHORT unpack.00561D95

```

- 500 chuỗi Serial của **Business License**

```

00561DF6 |>/8D4D E8    /LEA ECX,[LOCAL.6]
00561DF9 |.|0FBFD6    |MOVsx EDX,SI
00561DFC |.|8B45 FC    |MOV EAX,[LOCAL.1]
00561DFF |.|8B80 30030000 |MOV EAX,DWORD PTR DS:[EAX+330]
00561E05 |.|8B80 20020000 |MOV EAX,DWORD PTR DS:[EAX+220]
00561E0B |.|8B38      |MOV EDI,DWORD PTR DS:[EAX]
00561E0D |.|FF57 0C    |CALL DWORD PTR DS:[EDI+C] ; <== Business License
00561E10 |.|8B55 E8    |MOV EDX,[LOCAL.6] ; <== RealSerial
00561E13 |.|A1 A4115800 |MOV EAX,DWORD PTR DS:[5811A4] ; <== Fake Serial
00561E18 |.|E8 9B33EAFF |CALL unpack.004051B8 ; <== Compare
00561E1D |.|75 0A      |JNZ SHORT unpack.00561E29
00561E1F |.|C705 94115800>|MOV DWORD PTR DS:[581194],-1
00561E29 |>|46      |INC ESI
00561E2A |.|66:FFCB    |DEC BX ; <== 500 Licences
00561E2D |.^|75 C7    |JNZ SHORT unpack.00561DF6
----- == Trace Into ==-----

```

- Các chuỗi Serial này là mặc định .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : Personal License : 8X6J5-XC69-BZ63-5NAN
Business License : rr2b2b-6t4dk7u6-3v3r44

III – End of Tut :

- Finished – September 15, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.anthemion.co.uk
Production :	Anthemion Software Ltd.
SoftWare :	HelpBlocks 1.12 for Windows
Copyright by :	Copyright © 2002-2004 Anthemion Software Ltd. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

HelpBlocks 1.12 for Windows

HelpBlocks is an editor for authoring cross-platform HTML help. You can use it to produce help files for your Windows, Linux, or Mac applications, in conjunction with wxWidgets or other development tools. HelpBlocks can generate wxWidgets HTML Help files, and (on Windows only) MS HTML Help files.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Sorry, invalid registration key ...**". Ta tìm được thông báo này ở địa chỉ :

0043D035 . 68 14855F00 PUSH helpbloc.005F8514 ; ASCII "Sorry, invalid registration key. The key should be of the form: EA0927CF-E8CF149B-416363EE (three groups of 8 characters). Please also check that your specified user name is the same as the one under which you purchased HelpBlocks. Tr"...

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

0043D009 . E8 520B1600 CALL <JMP.&MSVCRT._EH_prolog> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Trace xuống chút :

0043D027 . E8 5B64FCFF CALL helpbloc.00403487 ; <== Trace Into here
----- **Trace Into** -----

00403491 |. E8 396E0200 CALL helpbloc.0042A2CF ; <== Encrypt
----- *Trace Into* -----

- Trace into vào quá trình mã hoá này để tìm hiểu :

0042A310 |. 68 B84F5F00 PUSH helpbloc.005F4FB8 ; ASCII "Anthemion Software HelpBlocks"
0042A315 |. 8D4D E8 LEA ECX,[LOCAL.6]
0042A318 |. 895D E8 MOV [LOCAL.6],EBX
0042A31B |. E8 40250300 CALL helpbloc.0045C860
0042A320 |. 8B0D 08D46200 MOV ECX,DWORD PTR DS:[62D408]
0042A326 |. 8D45 E8 LEA EAX,[LOCAL.6]
0042A329 |. 50 PUSH EAX
0042A32A |. 8D45 EC LEA EAX,[LOCAL.5]
0042A32D |. 56 PUSH ESI
0042A32E |. 50 PUSH EAX
0042A32F |. 895D FC MOV [LOCAL.1],EBX
0042A332 |. E8 4CF7FFFF CALL helpbloc.00429A83 ; <== MD5Hash

----- *MD5Hash* -----

- Chương trình chuyển U nhập về dạng LowerCase :

00429ADD |>/8A07 /MOV AL,BYTE PTR DS:[EDI]
00429ADF |.|3C 20 |CMP AL,20
00429AE1 |.|74 0E |JE SHORT helpbloc.00429AF1
00429AE3 |.|0FBEC0 |MOVSX EAX,AL
00429AE6 |.|50 |PUSH EAX ; /c
00429AE7 |.|FF15 50445B00 |CALL DWORD PTR DS:[<&MSVCRT(tolower)>] ; \tolower
00429AED |.|8806 |MOV BYTE PTR DS:[ESI],AL
00429AEF |.|59 |POP ECX
00429AF0 |.|46 |INC ESI
00429AF1 |>|47 |INC EDI
00429AF2 |.|381F |CMP BYTE PTR DS:[EDI],BL
00429AF4 |.^\75 E7 |JNZ SHORT helpbloc.00429ADD

- Sau đó nối chuỗi này vào sau chuỗi mặc định “*Anthemion Software HelpBlocks*”

00429AF6 |>\FF75 EC PUSH [LOCAL.5] ; /src
00429AF9 |. 8D85 84FDFFFF LEA EAX,[LOCAL.159] ; |\
00429AFF |. 881E MOV BYTE PTR DS:[ESI],BL ; |\
00429B01 |. 50 PUSH EAX ; |dest
00429B02 |. E8 C1401700 CALL <JMP.&MSVCRT.strcat> ; \strcat

- Sau đó tiến hành quá trình tạo chuỗi MD5Hash dựa trên chuỗi này :

00429B17 |. E8 C9E40000 CALL helpbloc.00437FE5 ; <== MD5Start
00429B1C |. 8D85 84FDFFFF LEA EAX,[LOCAL.159]
00429B22 |. 50 PUSH EAX ; /s
00429B23 |. E8 88401700 CALL <JMP.&MSVCRT.strlen> ; \strlen
00429B28 |. 50 PUSH EAX
00429B29 |. 8D85 84FDFFFF LEA EAX,[LOCAL.159]
00429B2F |. 50 PUSH EAX
00429B30 |. 8D45 84 LEA EAX,[LOCAL.31]
00429B33 |. 50 PUSH EAX
00429B34 |. E8 D4E40000 CALL helpbloc.0043800D ; <== MD5Update
00429B39 |. 8D45 84 LEA EAX,[LOCAL.31]
00429B3C |. 50 PUSH EAX
00429B3D |. 8D45 DC LEA EAX,[LOCAL.9]

```
00429B40 |. 50      PUSH EAX
00429B41 |. E8 67E50000 CALL helpbloc.004380AD ; <== MD5Finished
```

- Kế đó, chương trình lấy 12 bytes đầu tiên của chuỗi MD5Hash này để tạo thành chuỗi Serial :

```
00429B4E |> /33C9    /XOR ECX,ECX
00429B50 |. |33C0    |XOR EAX,EAX
00429B52 |> |0FB61406  |/MOVZX EDX,BYTE PTR DS:[ESI+EAX] ; <== MD5Hash[i]
00429B56 |. |C1E1 08  ||SHL ECX,8
00429B59 |. |03CA    ||ADD ECX,EDX
00429B5B |. |40      ||INC EAX
00429B5C |. |83F8 04  ||CMP EAX,4
00429B5F |.^|7C F1    |JL SHORT helpbloc.00429B52
00429B61 |. |A1 ECD05F00 |MOV EAX,DWORD PTR DS:[5FD0EC]
00429B66 |. |83FF 02    |CMP EDI,2
00429B69 |. |7D 32    |JGE SHORT helpbloc.00429B9D
00429B6B |. |8945 10    |MOV [ARG.3],EAX
00429B6E |. |51      |PUSH ECX
00429B6F |. |8D45 10    |LEA EAX,[ARG.3]
00429B72 |. |68 804F5F00 |PUSH helpbloc.005F4F80 ; ASCII "%08lX-"
00429B77 |. |50      |PUSH EAX
00429B78 |. |C645 FC 03  |MOV BYTE PTR SS:[EBP-4],3
00429B7C |. |E8 CF400300 |CALL helpbloc.0045DC50
00429B81 |. |8B45 10    |MOV EAX,[ARG.3]
00429B84 |. |83C4 0C    |ADD ESP,0C
00429B87 |. |8B48 F8    |MOV ECX,DWORD PTR DS:[EAX-8]
00429B8A |. |50      |PUSH EAX ; /Arg2
00429B8B |. |51      |PUSH ECX ; |Arg1
00429B8C |. |8D4D 0C    |LEA ECX,[ARG.2] ; |
00429B8F |. |E8 DC310300 |CALL helpbloc.0045CD70 ; \helpbloc.0045CD70
00429B94 |. |C645 FC 02  |MOV BYTE PTR SS:[EBP-4],2
00429B98 |. |8D4D 10    |LEA ECX,[ARG.3]
00429B9B |. |EB 30      |JMP SHORT helpbloc.00429BCD
00429B9D |> |8945 10    |MOV [ARG.3],EAX
00429BA0 |. |51      |PUSH ECX
00429BA1 |. |8D45 10    |LEA EAX,[ARG.3]
00429BA4 |. |68 784F5F00 |PUSH helpbloc.005F4F78 ; ASCII "%08lX"
00429BA9 |. |50      |PUSH EAX
00429BAA |. |C645 FC 04  |MOV BYTE PTR SS:[EBP-4],4
00429BAE |. |E8 9D400300 |CALL helpbloc.0045DC50
00429BB3 |. |8B45 10    |MOV EAX,[ARG.3]
00429BB6 |. |83C4 0C    |ADD ESP,0C
00429BB9 |. |8B48 F8    |MOV ECX,DWORD PTR DS:[EAX-8]
00429BBC |. |50      |PUSH EAX ; /Arg2
00429BBD |. |51      |PUSH ECX ; |Arg1
00429BBE |. |8D4D 0C    |LEA ECX,[ARG.2] ; |
00429BC1 |. |E8 AA310300 |CALL helpbloc.0045CD70 ; \helpbloc.0045CD70
00429BC6 |. |C645 FC 02  |MOV BYTE PTR SS:[EBP-4],2
00429BCA |. |8D4D 10    |LEA ECX,[ARG.3]
00429BCD |> |E8 7974FDFF |CALL helpbloc.0040104B
00429BD2 |. |47      |INC EDI
00429BD3 |. |83C6 04    |ADD ESI,4
00429BD6 |. |83FF 03    |CMP EDI,3
```

00429BD9 |.^|0F8C 6FFFFFFF \JL helpbloc.00429B4E

- Chuỗi Serial này có định dạng “%08X-%08X-%08X”. Tuy nhiên, sau cùng chương trình lại chuyển dạng chuỗi này về dạng LowerCase . Nên quá trình này ta có thể coi như chương trình tạo chuỗi Serial theo định dạng “%08x-%08x-%08x”

===== MD5Hash =====

- Quá trình so sánh diễn ra :

0042A365 |. E8 B61EFEFF CALL helpbloc.0040C220 ; <== Compare

===== Compare =====

0040C234 |. 51 PUSH ECX ; /s2 = “Fake Serial”

0040C235 |. 50 PUSH EAX ; |s1 = “Real Serial”

0040C236 |. E8 6F191900 CALL <JMP.&MSVCRT.strcmp> ; \strcmp

===== Compare =====

- Chương trình này có hai chuỗi Serial thực dựa trên hai chuỗi mặc định .:

“Anthemion Software HelpBlocks”

“Anthemion Software DialogBlocks and HelpBlocks”

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : ff22731e-138df358-626ed211

III – End of Tut :

- Finished – September 13, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.htmlcodecleaner.com-http.com
Production :	HTML Code Cleaner Team, BraveWorld Networks
SoftWare :	HTML Code Cleaner 3.20.0
Copyright by :	Copyright © HTML Code Cleaner Team, BraveWorld Networks. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial

HTML Code Cleaner 3.20.0

HTML Code Cleaner is intended to optimize HTML documents, resulting in shorter downloading / uploading time. Pages will appear in client's Internet browser in the same way, but they will be about 20%-40% smaller. Whether you use visual HTML editors or notepad, your HTML code contains unnecessary stuff: unneeded spaces, returns, meta tags, default values in HTML elements, etc. They will make your web pages bigger, without serving any real purpose. Bigger web pages takes longer to load, increases data traffic of your web server, and waste precious bandwidth of its internet connection. HTML Code Cleaner's main function is to save space on your web pages by removing unnecessary characters

and tags. Though the saving percentage may seem small, load time savings is much higher as your client's Internet browser parses the pages more efficiently. HTML Code Cleaner will also help you to check/fix broken links - both page links and image links. You can be released from fussy work on checking broken links.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo. Sau khi unPACK và kiểm tra lại biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0.**
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Invalid registration serial! Program will be closed.**" Ta tìm được thông báo này tại địa chỉ :
004B52F6 |. B8 A8534B00 MOV EAX,htmlcode.004B53A8 ; |ASCII "Invalid registration serial! Program will be closed."
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
004B522C |. E8 0737FBFF CALL htmlcode.00468938 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình này rất đơn giản. Sau khi dừng lại tại BreakPoint ta trace xuống chút và đến :
004B5264 |. E8 C7FDFFFF CALL htmlcode.004B5030 ; <== Reverse Serial
- Đây là quá trình đảo ngược chuỗi Serial . Ví dụ, chuỗi S nhập là “1234” thì sau khi kết thúc quá trình nà sẽ là “4321” . Trace tiếp ta đến :
004B527D |. A1 E09B4B00 MOV EAX,DWORD PTR DS:[4B9BE0]
004B5282 |. 8B00 MOV EAX,DWORD PTR DS:[EAX]
004B5284 |. E8 27E7FFFF CALL htmlcode.004B39B0 ; <== Trace Into
----- Trace Into -----
004B39DE |. E8 6110F5FF CALL htmlcode.00404A44 ; <== get length of S
004B39E3 |. 83F8 06 CMP EAX,6 ; <== Len.S atleast 6 charts
004B39E6 |. 7E 74 JLE SHORT htmlcode.004B3A5C
004B39E8 |. 8D45 FC LEA EAX,[LOCAL.1] ; <== from Reverse Serial
004B39EB |. B9 06000000 MOV ECX,6 ; <== cutting 6 charts
004B39F0 |. BA 01000000 MOV EDX,1 ; <== form the first chart
004B39F5 |. E8 EA12F5FF CALL htmlcode.00404CE4 ; <== cutting : NewStr
004B39FA |. 8B45 FC MOV EAX,[LOCAL.1] ; <== NewStr
004B39FD |. BA AC3A4B00 MOV EDX,htmlcode.004B3AAC ; ASCII "2^7\$:-8#G@W-Y`R~]"
004B3A02 |. E8 8911F5FF CALL htmlcode.00404B90 ; <== Must be SAME
----- Trace Into -----

- Như vậy ta thấy ngay được, chuỗi Serial thực sẽ là chuỗi đảo ngược của chuỗi mặc định "**2^7\$:-8#G@W-Y`R~J'**" được thêm vào cuối 6 ký tự ngẫu nhiên .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial :]~R`Y-W@G#8-:\$7^2-REA-0

III – KeyGen :

/Section /- N/A

IV – End of Tut :

- Finished – **August 11, 2004**

Reverse Engineering Association

SoftWare

Homepage	:	http://www.minihttpserver.net
Production	:	MiniHTTP Software, Inc.
SoftWare	:	HTML Page Guardian v3.0
Copyright by	:	Copyright © 2000-2001 MiniHTTP Software, Inc. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N / A
Language	:	Borland Delphi 4.0 - 5.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

HTML Page Guardian v3.0

If you are a web designer, you know how much time, energy and money it takes to create a unique and professional Web Page, you also know how easy it is for the visitors to copy your working achievement and reuse it in their Web Page. That is called web site plagiarism. Today, web site plagiarism is something common. This applies not only to HTML or Javascript code, but also to everything else within a web page, for instance, text, links, graphics. Therefore you need a professional and effective solution - **HTML Page Guard** to help you prevent people making unauthorized copies of your Web Page.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**

- Chạy thử chương trình với User và Fake Serial ta không nhận được thông báo gì . Ta cũng không tìm thấy thông báo gì trong quá trình tìm kiếm chuỗi . Tuy nhiên, bằng PeID ta biết được chương trình có sử dụng thuật toán MD5 . Biết được địa chỉ này, ta tìm đến đây và đặt BreakPoint . Chủ yếu là để xem chương trình có sử dụng thuật toán này vào quá trình mã hóa tạo Serial hay không .

- Chạy thử với U và FK, chương trình dừng lại tại điểm đặt BP, truy ngược lần lần ta tìm được FUNCTION chính của quá trình mã hoá . Ta đặt BP tại đây :

008B2FFE |. E8 A5FDFFFF CALL htmlenc.008B2DA8 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP .

Trace xuống chút ta đến :

008B3018 . 8D45 E8 LEA EAX,[LOCAL.6]	; <== User
008B301B . BA 1C318B00 MOV EDX,htmlenc.008B311C	; ASCII
"HTMLpageguard0723"	
008B3020 . E8 EB0EF5FF CALL htmlenc.00803F10	; <== Concat two String
008B3025 . 8B45 E8 MOV EAX,[LOCAL.6]	; <== Str after Concat
008B3028 . 8D55 EC LEA EDX,[LOCAL.5]	; <== Add of Real Serial

```

008B302B |. E8 F0ECFFFF CALL htmlenc.008B1D20 ; <== Encrypt
008B3030 |. 8D55 FC    LEA EDX,[LOCAL.1]
008B3033 |. 8D45 EC    LEA EAX,[LOCAL.5]
008B3036 |. E8 71ECFFFF CALL htmlenc.008B1CAC ; <== Real Serial : MD5hash
008B303B |. 8D55 E4    LEA EDX,[LOCAL.7]
008B303E |. A1 E8BE8B00 MOV EAX,DWORD PTR DS:[8BBEE8]
008B3043 |. 8B00      MOV EAX,DWORD PTR DS:[EAX]
008B3045 |. 8B80 DC020000 MOV EAX,DWORD PTR DS:[EAX+2DC]
008B304B |. E8 A0B3F7FF CALL htmlenc.0082E3F0 ; <== Fake Serial
008B3050 |. 8B45 E4    MOV EAX,[LOCAL.7] ; <== Fake Serial
008B3053 |. 8B55 FC    MOV EDX,[LOCAL.1] ; <== Real Serial
008B3056 |. E8 BD0FF5FF CALL htmlenc.00804018 ; <== Compare

```

- Qua đây ta xác định được đoạn CODE mã hoá tạo chuỗi Serial :

```
008B302B |. E8 F0ECFFFF CALL htmlenc.008B1D20 ; <== Encrypt
```

-----**==== MD5-Encrypt ===**-----

```
008B1D38 |. E8 03000000 CALL htmlenc.008B1D40 ; <== MD5Hash
```

-----**==== MD5-Hash ===**-----

```
008B1D4E |. E8 31FEFFFF CALL htmlenc.008B1B84 ; <== MD5Start
```

```
008B1D53 |. 8BD3      MOV EDX,EBX
```

```
008B1D55 |. 8BC4      MOV EAX,ESP
```

```
008B1D57 |. 8BCE      MOV ECX,ESI
```

```
008B1D59 |. E8 56FEFFFF CALL htmlenc.008B1BB4 ; <== MD5Update
```

```
008B1D5E |. 8BD4      MOV EDX,ESP
```

```
008B1D60 |. 8BC7      MOV EAX,EDI
```

```
008B1D62 |. E8 CDFFEFF CALL htmlenc.008B1C34 ; <== MD5Finished
```

-----**==== MD5-Hash ===**-----

-----**==== MD5-Encrypt ===**-----

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 35DE1054753B04AD51F36A6346FE12DB

III – KeyGen :

/Section I /- Gắn chuỗi mặc định "**HTMLpageguard0723**" vào sau chuỗi U nhập .

/Section II /- Tạo MD5Hash cho chuỗi này . Đây chính là chuỗi Serial thực của chương trình .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.minihttpserver.net
Production :	MiniHTTP Software, Inc.
SoftWare :	HTML Password v3.1 Pro
Copyright by :	Copyright © 2000-2001 MiniHTTP Software, Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A

Language : Borland Delphi 4.0 - 5.0
 Crack Tool : OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
 Unpack : N / A
 Request : Correct Serial / KeyGen

HTML Password v3.1 Pro

If you are a web designer, you know how much time, energy and money it takes to create a unique and professional Web Page, you also know how easy it is for the visitors to copy your working achievement and reuse it in their Web Page. That is called web site plagiarism. Today, web site plagiarism is something common. This applies not only to HTML or Javascript code, but also to everything else within a web page, for instance, text, links, graphics. Therefore you need a professional and effective solution - **HTML Page Guard** to help you prevent people making unauthorized copies of your Web Page.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**
- Chạy thử chương trình với User và Fake Serial ta không nhận được thông báo gì . Ta cũng không tìm thấy thông báo gì trong quá trình tìm kiếm chuỗi . Tuy nhiên, bằng PeiD ta biết được chương trình có sử dụng thuật toán MD5 . Biết được địa chỉ này, ta tìm đến đây và đặt BreakPoint . Chủ yếu là để xem chương trình có sử dụng thuật toán này vào quá trình mã hoá tạo Serial hay không .
- Chạy thử với U và FK, chương trình dừng lại tại điểm đặt BP, truy ngược lần lần ta tìm được FUNCTION chính của quá trình mã hoá . Ta đặt BP tại đây :
004ABEE1 |. E8 76FDFFFF CALL htmlpass.004ABC5C ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP .

Trace xuống chút ta đến :

004ABEFB . 8D45 E0 LEA EAX,[LOCAL.8]	; <== User
004ABEFE . BA 2CC14A00 MOV EDX,htmlpass.004AC12C	; ASCII "HTMLPassword0828"
004ABF03 . E8 B07FF5FF CALL htmlpass.00403EB8	; <== Concat two String
004ABF08 . 8B45 E0 MOV EAX,[LOCAL.8]	; <== Str after Concat
004ABF0B . 8D55 E4 LEA EDX,[LOCAL.7]	; <== Add of Real Serial
004ABF0E . E8 61EDFFFF CALL htmlpass.004AAC74	; <== Encrypt
004ABF13 . 8D55 F8 LEA EDX,[LOCAL.2]	
004ABF16 . 8D45 E4 LEA EAX,[LOCAL.7]	
004ABF19 . E8 E2ECFFFF CALL htmlpass.004AAC00	
004ABF1E . 8D55 DC LEA EDX,[LOCAL.9]	
004ABF21 . A1 A45E4B00 MOV EAX,DWORD PTR DS:[4B5EA4]	
004ABF26 . 8B00 MOV EAX,DWORD PTR DS:[EAX]	
004ABF28 . 8B80 DC020000 MOV EAX,DWORD PTR DS:[EAX+2DC]	
004ABF2E . E8 B924F8FF CALL htmlpass.0042E3EC	
004ABF33 . 8B45 DC MOV EAX,[LOCAL.9]	; <== Fake Serial
004ABF36 . 8B55 F8 MOV EDX,[LOCAL.2]	; <== Real Serial : MD5hash
004ABF39 . E8 8280F5FF CALL htmlpass.00403FC0	; <== Compare

- Qua đây ta xác định được đoạn CODE mã hoá tạo chuỗi Serial :

```

004ABF0E |. E8 61EDFFFF CALL htmlpass.004AAC74 ; <== Encrypt
----- === MD5-Encrypt ===-
004AAC8C |. E8 03000000 CALL htmlpass.004AAC94 ; <== MD5 Hash
----- === MD5-Hash ===-
004AACAB2 |. E8 31FEFFFF CALL htmlpass.004AAAD8 ; <== MD5Start
004AACAB7 |. 8BD3      MOV EDX,EBX
004AACAB9 |. 8BC4      MOV EAX,ESP
004AACABD |. 8BCE      MOV ECX,ESI
004AACACD |. E8 56FEFFFF CALL htmlpass.004AAB08 ; <== MD5Update
004AACACB2 |. 8BD4      MOV EDX,ESP
004AACACB4 |. 8BC7      MOV EAX,EDI
004AACACB6 |. E8 CDFFEF000 CALL htmlpass.004AAB88 ; <== MD5Finish
----- === MD5-Hash ===-
----- === MD5-Encrypt ===-

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : AF2AF71D87D01920E0DD0A01C9A8C0F6

III – KeyGen :

- /Section I / - Gắn chuỗi mặc định "**HTMLPassword0828**" vào sau chuỗi U nhập .
- /Section II / - Tạo MD5Hash cho chuỗi này . Đây chính là chuỗi Serial thực của chương trình .

IV – End of Tut :

- Finished – **September 07, 2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.lighttek.com
Production :	1997-1999 Lighttek Software.
SoftWare :	IconTOY 3.1
Copyright by :	Copyright © 1997-1999 Lighttek Software. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial

IconTOY 3.1

Using various icon extractor programs, you can manually search for icons by opening individual files. But that's hard work. This program is designed specifically to extract the icons (ICO, ICL, EXE, DLL, BMP) from any file containing them. It will search entire hard drives and display all the icons. Then you can either copy a particular icon to the clipboard or save it as a single icon (with 16 Color, 256 Color, High Color, True Color) or bitmap (bmp) file. A secondary function is to manipulate the desktop wallpaper. You can use any found icon or bmp-file as wallpaper. Only one click and you can see the result on the screen. You can set the wallpaper as centered or tiled. As the wallpaper's setting is made in one touch, this function reminds mosaic game.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và được viết bằng **Borland Delphi 4.0 - 5.0**.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Registration key error!**" Ta tìm được thông báo này tại địa chỉ :
00486076 > \B8 38614800 MOV EAX,icontoy.00486138 ; ASCII "Registration key error!"
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
00485FA0 . E8 B3FCFFFF CALL icontoy.00485C58 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Chương trình này rất đơn giản. Sau khi dừng lại tại BreakPoint ta trace into luôn :

```

00485C6B |. 3D 8CA6BA00 CMP EAX,0BAA68C
00485C70 |. 0F84 B9010000 JE icontoy.00485E2F
00485C76 |. 3D 6A5A1300 CMP EAX,135A6A
00485C7B |. 0F84 AE010000 JE icontoy.00485E2F
00485C81 |. 3D E0621300 CMP EAX,1362E0
00485C86 |. 0F84 A3010000 JE icontoy.00485E2F
00485C8C |. 3D 76281300 CMP EAX,132876
00485C91 |. 0F84 98010000 JE icontoy.00485E2F
00485C97 |. 3D AEED3100 CMP EAX,31EDAE
00485C9C |. 0F84 8D010000 JE icontoy.00485E2F
00485CA2 |. 3D C6E93100 CMP EAX,31E9C6
00485CA7 |. 0F84 82010000 JE icontoy.00485E2F
00485CAD |. 3D A8D81200 CMP EAX,12D8A8
00485CB2 |. 0F84 77010000 JE icontoy.00485E2F
00485CB8 |. 3D 267F1300 CMP EAX,137F26
00485CBD |. 0F84 6C010000 JE icontoy.00485E2F
00485CC3 |. 3D B6B7B800 CMP EAX,0B8B7B6
00485CC8 |. 0F84 61010000 JE icontoy.00485E2F
00485CCE |. 3D AE7A1200 CMP EAX,127AAE
00485CD3 |. 0F84 56010000 JE icontoy.00485E2F
00485CD9 |. 3D A2D21200 CMP EAX,12D2A2
00485CDE |. 0F84 4B010000 JE icontoy.00485E2F
00485CE4 |. 3D 2AFAB900 CMP EAX,0B9FA2A
00485CE9 |. 0F84 40010000 JE icontoy.00485E2F
00485CEF |. 3D 36F8B900 CMP EAX,0B9F836
00485CF4 |. 0F84 35010000 JE icontoy.00485E2F
00485CFA |. 3D 0810C100 CMP EAX,0C11008
00485CFF |. 0F84 2A010000 JE icontoy.00485E2F
00485D05 |. 3D 8630BA00 CMP EAX,0BA3086
00485D0A |. 0F84 1F010000 JE icontoy.00485E2F
00485D10 |. 3D 78DB5B04 CMP EAX,45BDB78
00485D15 |. 0F84 14010000 JE icontoy.00485E2F
00485D1B |. 3D D8D81400 CMP EAX,14D8D8
00485D20 |. 0F84 09010000 JE icontoy.00485E2F
00485D26 |. 3D 0C541400 CMP EAX,14540C
00485D2B |. 0F84 FE000000 JE icontoy.00485E2F
00485D31 |. 3D 1A431500 CMP EAX,15431A

```

00485D36 |. 0F84 F3000000 JE icontoy.00485E2F
00485D3C |. 3D 2AF61400 CMP EAX,14F62A
00485D41 |. 0F84 E8000000 JE icontoy.00485E2F
00485D47 |. 3D 825C1400 CMP EAX,145C82
00485D4C |. 0F84 DD000000 JE icontoy.00485E2F
00485D52 |. 3D 9CF0C900 CMP EAX,0C9F09C
00485D57 |. 0F84 D2000000 JE icontoy.00485E2F
00485D5D |. 3D 5081DC07 CMP EAX,7DC8150
00485D62 |. 0F84 C7000000 JE icontoy.00485E2F
00485D68 |. 3D 246DCA00 CMP EAX,0CA6D24
00485D6D |. 0F84 BC000000 JE icontoy.00485E2F
00485D73 |. 3D 8CA6BA00 CMP EAX,0BAA68C
00485D78 |. 0F84 B1000000 JE icontoy.00485E2F
00485D7E |. 3D 4A232400 CMP EAX,24234A
00485D83 |. 0F84 A6000000 JE icontoy.00485E2F
00485D89 |. 3D C02B2400 CMP EAX,242BC0
00485D8E |. 0F84 9B000000 JE icontoy.00485E2F
00485D94 |. 3D 56F12300 CMP EAX,23F156
00485D99 |. 0F84 90000000 JE icontoy.00485E2F
00485D9F |. 3D 0E322400 CMP EAX,24320E ; UNICODE "\Messenger\MSVCR71.dll"
00485DA4 |. 0F84 85000000 JE icontoy.00485E2F
00485DAA |. 3D 262E2400 CMP EAX,242E26
00485DAF |. 74 7E JE SHORT icontoy.00485E2F
00485DB1 |. 3D 88A12300 CMP EAX,23A188
00485DB6 |. 74 77 JE SHORT icontoy.00485E2F
00485DB8 |. 3D A6CE2500 CMP EAX,25CEA6
00485DBD |. 74 70 JE SHORT icontoy.00485E2F
00485DBF |. 3D B6D26F01 CMP EAX,16FD2B6
00485DC4 |. 74 69 JE SHORT icontoy.00485E2F
00485DC6 |. 3D 2ECA2400 CMP EAX,24CA2E
00485DCB |. 74 62 JE SHORT icontoy.00485E2F
00485DCD |. 3D 22222500 CMP EAX,252222
00485DD2 |. 74 5B JE SHORT icontoy.00485E2F
00485DD4 |. 3D 2A157101 CMP EAX,171152A
00485DD9 |. 74 54 JE SHORT icontoy.00485E2F
00485DDB |. 3D 36137101 CMP EAX,1711336
00485DE0 |. 74 4D JE SHORT icontoy.00485E2F
00485DE2 |. 3D 88D38A02 CMP EAX,28AD388
00485DE7 |. 74 46 JE SHORT icontoy.00485E2F
00485DE9 |. 3D 864B7101 CMP EAX,1714B86
00485DEE |. 74 3F JE SHORT icontoy.00485E2F
00485DF0 |. 3D 382D7001 CMP EAX,1702D38
00485DF5 |. 74 38 JE SHORT icontoy.00485E2F
00485DF7 |. 3D B8A12500 CMP EAX,25A1B8
00485DFC |. 74 31 JE SHORT icontoy.00485E2F
00485DFE |. 3D DCF52400 CMP EAX,24F5DC
00485E03 |. 74 2A JE SHORT icontoy.00485E2F
00485E05 |. 3D FA0B2600 CMP EAX,260BFA
00485E0A |. 74 23 JE SHORT icontoy.00485E2F
00485E0C |. 3D AA452700 CMP EAX,2745AA
00485E11 |. 74 1C JE SHORT icontoy.00485E2F
00485E13 |. 3D 02AC2600 CMP EAX,26AC02

```

00485E18 |. 74 15      JE SHORT icontoy.00485E2F
00485E1A |. 3D 9C0B8101  CMP EAX,1810B9C
00485E1F |. 74 0E      JE SHORT icontoy.00485E2F
00485E21 |. 3D D0128B01  CMP EAX,18B12D0
00485E26 |. 74 07      JE SHORT icontoy.00485E2F
00485E28 |. 3D 24888101  CMP EAX,1818824
00485E2D |. 75 7C      JNZ SHORT icontoy.00485EAB

```

- Chuỗi Serial ta nhập sẽ được chuyển sang dạng HEX và so sánh với 1 trong 48 giá trị được lưu sẵn.

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial :]~R`Y-W@G#8-:\$7^2-REA-0

III – KeyGen :

/Section 0/- Chuyển các giá trị này sang định dạng “%lu” và xuất ra .

IV – End of Tut :

- Finished – *August 12, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.imtoo.com/audio-encoder.html
Production :	ImTOO Software Studio .
SoftWare :	ImTOO Audio Encoder 1.0.1 b 707
Copyright by :	Copyright © 2003-2004 ImTOO Software Studio . All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 7.0 [Debug]
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

ImTOO Audio Encoder 1.0.1 b 707

Powerful audio encoder which can convert between MP3, WAV, WMA etc. popular audio formats. ImTOO Audio Encoder provides users an easy way to convert audio files within a few clicks. An easy and completed way to decode/encode all popular audio files. Besides MP3, WAV, WMA, ImTOO Audio Encoder also supports MP2, OGG, APE, AAC, VQF etc. formats. All conversion processes between these formats are very easy to handle and fast. Provides different settings for different audio formats - users can easily get output files as they want. No matter what audio file you want, ImTOO Audio Encoder can do it for you! Supports ID3 tag and batch conversion. Users don't have to think so much, just need a click!

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 7.0 [Debug]**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid registration info!**" . Ta tìm được thông báo này tại địa chỉ :

0040C7C4 . 68 EC254400 PUSH audioenc.004425EC ; ASCII "Invalid registration info!"

- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

0040C7B3 . E8 38F0FFFF CALL audioenc.0040B7F0 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.

0040C7B3 . E8 38F0FFFF CALL audioenc.0040B7F0 ; <== Set BreakPoint here

0040C7B8 . E8 53F6FFFF CALL audioenc.0040BE10 ; <== Trace Into here

- Dùng F7 trace into ta đến :

0040C13B . 83F8 27 CMP EAX,27 ; <== LenS must be 39 charts

0040C13E . 0F85 D6050000 JNZ audioenc.0040C71A

- Trace tiếp ta xác định chuỗi dùng để mã hoá :

0040C3C0 . E8 EB83FFFF CALL audioenc.004047B0 ; <== String to Encrypt

-----String to Encrypt-----

Ngay tại đây, quan sát của sổ STACK ta thấy dòng :

0012D4C8 003C73B0 ASCII "12345678901234567890audioenc"

Như vậy, chương trình đã lấy 20 ký tự đầu tiên và ghép với chuỗi mặc định "**“audioenc”**". Ta có thể cho rằng chương trình sẽ dùng chuỗi này để tiến hành mã hoá, nhưng thực chất không phải vậy, chuỗi này chỉ là một phần của chuỗi thực.

Chương trình sẽ gắn chuỗi này vào sau một chuỗi mặc định thứ hai, chuỗi này gồm 19 ký tự . Có giá trị như sau :

**0x31, 0x61, 0x01, 0x64, 0x03,
0x6F, 0x05, 0x6E, 0x07, 0x75,
0x02, 0x69, 0x04, 0x65, 0x06,
0x63, 0x08, 0x30, 0x30**

Như vậy, chuỗi được dùng để mã hoá sẽ có chiều dài : 47 ký tự .

-----String to Encrypt-----

- Trace tiếp ta đến quá trình mã hoá :

0040C3F7 . E8 24800000 CALL audioenc.00414420 ; <== EnCrypt

-----EnCrypt-----

00414434 |. E8 47FFFFFF CALL audioenc.00414380 ; <== MD5Hash

-----MD5Hash-----

00414396 |. E8 B5000000 CALL audioenc.00414450 ; <== MD5Start

0041439B |. 8B4F 38 MOV ECX,DWORD PTR DS:[EDI+38]

0041439E |. 8BC1 MOV EAX,ECX

004143A0 |. 83C4 04 ADD ESP,4

004143A3 |. 8D70 01 LEA ESI,DWORD PTR DS:[EAX+1]

004143A6 |> 8A10 /MOV DL,BYTE PTR DS:[EAX]

004143A8 |. 40 |INC EAX

004143A9 |. 84D2 |TEST DL,DL

004143AB |.^ 75 F9 \JNZ SHORT audioenc.004143A6

004143AD |. 2BC6 SUB EAX,ESI

004143AF |. 50 PUSH EAX

004143B0 |. 51 PUSH ECX

```

004143B1 |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
004143B5 |. 51 PUSH ECX
004143B6 |. E8 A5090000 CALL audioenc.00414D60 ; <== MD5Update
004143BB |. 8D5424 18 LEA EDX,DWORD PTR SS:[ESP+18]
004143BF |. 52 PUSH EDX
004143C0 |. 8D5F 05 LEA EBX,DWORD PTR DS:[EDI+5]
004143C3 |. 53 PUSH EBX
004143C4 |. E8 570A0000 CALL audioenc.00414E20 ; <== MD5Finished
-----== MD5Hash ==
-----== EnCrypt ==

```

- Kế đó, chương trình tiến hành mã hoá chuỗi MD5Hash này bằng cách (1) chọn lấy các ký tự ở vị trí chẵn (2) các vị trí thứ **4, 9 14** sẽ là ký tự “-“. Như vậy chuỗi này sẽ có dạng “XXXX-XXXX-XXXX”

```

0040C431 >/85DB TEST EBX,EBX
0040C433 .|0F8C D7020000 JL audioenc.0040C710
0040C439 .|8B4424 38 MOV EAX,DWORD PTR SS:[ESP+38]
0040C43D .|3B58 F4 CMP EBX,DWORD PTR DS:[EAX-C]
0040C440 .|0F8F CA020000 JG audioenc.0040C710
0040C446 .|8A0C03 MOV CL,BYTE PTR DS:[EBX+EAX]
0040C449 .|8B46 FC MOV EAX,DWORD PTR DS:[ESI-4]
0040C44C .|8B6E F4 MOV EBP,DWORD PTR DS:[ESI-C]
0040C44F .|884C24 2B MOV BYTE PTR SS:[ESP+2B],CL
0040C453 .|B9 01000000 MOV ECX,1
0040C458 .|2BC8 SUB ECX,EAX
0040C45A .|8B46 F8 MOV EAX,DWORD PTR DS:[ESI-8]
0040C45D .|8D7D 01 LEA EDI,DWORD PTR SS:[EBP+1]
0040C460 .|2BC7 SUB EAX,EDI
0040C462 .|0BC1 OR EAX,ECX
0040C464 .|7D 0E JGE SHORT audioenc.0040C474
0040C466 .|57 PUSH EDI
..... Cutting .....
0040C48C .|8BC3 MOV EAX,EBX
0040C48E .|99 CDQ
0040C48F .|2BC2 SUB EAX,EDX
0040C491 .|D1F8 SAR EAX,1
0040C493 .|40 INC EAX
0040C494 .|25 03000080 AND EAX,80000003
0040C499 .|897E F4 MOV DWORD PTR DS:[ESI-C],EDI
0040C49C .|C60437 00 MOV BYTE PTR DS:[EDI+ESI],0
0040C4A0 .|79 05 JNS SHORT audioenc.0040C4A7
0040C4A2 .|48 DEC EAX
0040C4A3 .|83C8 FC OR EAX,FFFFFFFC
0040C4A6 .|40 INC EAX
0040C4A7 >|75 14 JNZ SHORT audioenc.0040C4BD
0040C4A9 .|6A 01 PUSH 1
0040C4AB .|68 A4214400 PUSH audioenc.004421A4
0040C4B0 .|8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
0040C4B4 .|E8 F782FFFF CALL audioenc.004047B0
0040C4B9 .|8B7424 10 MOV ESI,DWORD PTR SS:[ESP+10]
0040C4BD >|83C3 02 ADD EBX,2
0040C4C0 .|83FB 20 CMP EBX,20
0040C4C3 .^|0F8C 68FFFFFF JL audioenc.0040C431

```

- Sau đó chuỗi này sẽ được gắn vào sau 20 ký tự đầu tiên của chuỗi FakeSerial nhập . Hay nói cách khác, chương trình sẽ sử dụng 20 ký tự đầu tiên của chuỗi Serial để làm chuỗi mã hoá .

```
0040C58A . E8 91E9FFFF CALL audioenc.0040AF20 ; <== Real Serial
0040C58F . 8B7C24 20 MOV EDI,DWORD PTR SS:[ESP+20] ; <== Fake Serial
0040C593 . 8B7424 10 MOV ESI,DWORD PTR SS:[ESP+10] ; <== Real Serial
0040C597 . 57 PUSH EDI
0040C598 . 56 PUSH ESI
0040C599 . E8 2F470100 CALL audioenc.00420CCD ; <== Compare
```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 9WMHPEAZK6MBAPKVLIMU65F4-654A-CB05-4631

III – KeyGen :

- /Section I /- Tạo chuỗi ngẫu nhiên gồm 20 ký tự .
- /Section II /- Gắn vào sau chuỗi ngẫu nhiên này chuỗi mặc định “***audioenc***” .
- /Section III /- Gắn chuỗi này vào sau chuỗi mặc định thứ hai .
- /Section IV /- Tao chuỗi MD5Hash của chuỗi này.
- /Section V /- Lấy các ký tự ở vị trí chẵn chuỗi MD5Hash theo dạng “***XXXX-XXXX-XXXX***”
- /Section VI /- Gắn chuỗi này vào sau 20 ký tự ngẫu nhiên đầu tiên : Chuỗi Serial thực .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.imtoo.com/cd-ripper.html
Production :	ImTOO Software Studio .
SoftWare :	ImTOO CD Ripper 1.0.12 b 710
Copyright by :	Copyright © 2003-2004 ImTOO Software Studio . All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 7.0 [Debug]
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

ImTOO CD Ripper 1.0.12 b 710

ImTOO CD Ripper is a flexible tool used to extract CD tracks to MP3, WAV, WMA, OGG Vorbis, VQF, APE with ease. Auto-detecting your CD-Rom and auto-listing tracks makes you can rip these tracks by just one click. Support retrieving tracks' information from remote Cddb (CD database) or creating your local Cddb. You can select bitrate from 32Kbps to 320Kbps for MP3 encoder. ImTOO CD Ripper allows you to edit ID3 tag and add files to M3U/PLS playlist. Embedded CD player can playback CD tracks before ripping. It also provides volume normalization for tracks different volume levels.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 7.0 [Debug]**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid registration info!**" . Ta tìm được thông báo này tại địa chỉ :

00461274 . 68 686E5300 PUSH cdripper.00536E68 ; |Arg1 = 00536E68 ASCII "Invalid registration info!"

- Truy ngược lên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

00461263 . E8 48000000 CALL cdripper.004612B0 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.

00461263 . E8 48000000 CALL cdripper.004612B0 ; <== Set BreakPoint here
00461268 . E8 C3030000 CALL cdripper.00461630 ; <== Trace Into here

- Dùng F7 trace into ta đến :

004617FB |. 83F8 27 CMP EAX,27 ; <== LenS must be 39 charts
004617FE |. 0F85 E5030000 JNZ cdripper.00461BE9

- Trace tiếp ta xác định chuỗi dùng để mã hoá :

0040C3C0 . E8 EB83FFFF CALL audioenc.004047B0 ; <== String to Encrypt

===== String to Encrypt =====

Ngay tại đây, quan sát của sổ STACK ta thấy dòng :

0012F0A4 00D0F0D8 ASCII "12345678901234567890CD Ripper"

Như vậy, chương trình đã lấy 20 ký tự đầu tiên và ghép với chuỗi mặc định "**“CD Ripper”**". Ta có thể cho rằng chương trình sẽ dùng chuỗi này để tiến hành mã hoá, nhưng thực chất không phải vậy, chuỗi này chỉ là một phần của chuỗi thực .

Chương trình sẽ gắn chuỗi này vào sau một chuỗi mặc định thứ hai, chuỗi này gồm 21 ký tự . Có giá trị như sau :

*0x31, 0x43, 0x01, 0x20, 0x03,
0x69, 0x05, 0x70, 0x07, 0x72,
0x09, 0x44, 0x02, 0x52, 0x04,
0x70, 0x06, 0x65, 0x08, 0x30, 0x30*

Như vậy, chuỗi được dùng để mã hoá sẽ có chiều dài : 50 ký tự .

===== String to Encrypt =====

- Trace tiếp ta đến quá trình mã hoá :

00461A42 |. E8 09D4FEFF CALL cdripper.0044EE50 ; \cdripper.0044EE50

===== MD5Hash =====

0044EE6F |. E8 CCF4FFFF CALL cdripper.0044E340 ; <== MD5Start
0044EE74 |. 8B53 38 MOV EDX,DWORD PTR DS:[EBX+38]
0044EE77 |. 83C9 FF OR ECX,FFFFFF
0044EE7A |. 8BFA MOV EDI,EDX
0044EE7C |. 33C0 XOR EAX,EAX
0044EE7E |. F2:AE REPNE SCAS BYTE PTR ES:[EDI]
0044EE80 |. F7D1 NOT ECX
0044EE82 |. 49 DEC ECX
0044EE83 |. 51 PUSH ECX

```

0044EE84 |. 52      PUSH EDX
0044EE85 |. 8D5424 1C  LEA EDX,DWORD PTR SS:[ESP+1C]
0044EE89 |. 52      PUSH EDX
0044EE8A |. E8 E1F4FFFF CALL cdripper.0044E370           ; <== MD5Update
0044EE8F |. 8D4424 20  LEA EAX,DWORD PTR SS:[ESP+20]
0044EE93 |. 8D6B 05  LEA EBP,DWORD PTR DS:[EBX+5]
0044EE96 |. 50      PUSH EAX
0044EE97 |. 55      PUSH EBP
0044EE98 |. E8 93F5FFFF CALL cdripper.0044E430           ; <== MD5Finished
----- MD5Hash -----

```

- Kế đó, chương trình tiến hành mã hoá chuỗi MD5Hash này bằng cách (1) chọn lấy các ký tự ở vị trí chẵn (2) các vị trí thứ **4, 9 14** sẽ là ký tự “-“. Như vậy chuỗi này sẽ có dạng “XXXX-XXXX-XXXX”

```

00461A75 |> /8B4C24 34  /MOV ECX,DWORD PTR SS:[ESP+34]
00461A79 |. |8A140E    |MOV DL,BYTE PTR DS:[ESI+ECX]
00461A7C |. |8D4C24 08  |LEA ECX,DWORD PTR SS:[ESP+8]
00461A80 |. |885424 0C  |MOV BYTE PTR SS:[ESP+C],DL
00461A84 |. |8B4424 0C  |MOV EAX,DWORD PTR SS:[ESP+C]
00461A88 |. |50      |PUSH EAX
00461A89 |. |E8 C03E0600 |CALL cdripper.004C594E
00461A8E |. |8BC6      |MOV EAX,ESI
00461A90 |. |99      |CDQ
00461A91 |. |2BC2      |SUB EAX,EDX
00461A93 |. |D1F8      |SAR EAX,1
00461A95 |. |40      |INC EAX
00461A96 |. |25 03000080 |AND EAX,80000003
00461A9B |. |79 05      |JNS SHORT cdripper.00461AA2
00461A9D |. |48      |DEC EAX
00461A9E |. |83C8 FC   |OR EAX,FFFFFFFC
00461AA1 |. |40      |INC EAX
00461AA2 |> |75 0E      |JNZ SHORT cdripper.00461AB2
00461AA4 |. |68 F8E55200 |PUSH cdripper.0052E5F8
00461AA9 |. |8D4C24 0C  |LEA ECX,DWORD PTR SS:[ESP+C]
00461AAD |. |E8 753E0600 |CALL cdripper.004C5927
00461AB2 |> |83C6 02   |ADD ESI,2
00461AB5 |. |83FE 20   |CMP ESI,20
00461AB8 |.^\7C BB   |JL SHORT cdripper.00461A75

```

- Sau đó chuỗi này sẽ được gắn vào sau 20 ký tự đầu tiên của chuỗi FakeSerial nhập . Hay nói cách khác, chương trình sẽ sử dụng 20 ký tự đầu tiên của chuỗi Serial để làm chuỗi mã hoá .

```

00461AE2 |. E8 8DD60500 CALL cdripper.004BF174           ; <== Real Serial
00461AE7 |. 8B7424 18  MOV ESI,DWORD PTR SS:[ESP+18]       ; <== Fake Serial
00461AEB |. 8B4424 08  MOV EAX,DWORD PTR SS:[ESP+8]       ; <== Real Serial
00461AEF |> 8A10      /MOV DL,BYTE PTR DS:[EAX]
00461AF1 |. 8ACA      |MOV CL,DL
00461AF3 |. 3A16      |CMP DL,BYTE PTR DS:[ESI]
00461AF5 |. 75 1C      |JNZ SHORT cdripper.00461B13
00461AF7 |. 84C9      |TEST CL,CL
00461AF9 |. 74 14      |JE SHORT cdripper.00461B0F
00461AFB |. 8A50 01   |MOV DL,BYTE PTR DS:[EAX+1]
00461AFE |. 8ACA      |MOV CL,DL
00461B00 |. 3A56 01   |CMP DL,BYTE PTR DS:[ESI+1]

```

```

00461B03 |. 75 0E      |JNZ SHORT cdripper.00461B13
00461B05 |. 83C0 02    |ADD EAX,2
00461B08 |. 83C6 02    |ADD ESI,2
00461B0B |. 84C9      |TEST CL,CL
00461B0D |.^ 75 E0     |JNZ SHORT cdripper.00461AEF

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 5ZX8ELNWXAEFH3J70B621-AE32-FF04-DAF5

III – KeyGen :

- /Section I /- Tạo chuỗi ngẫu nhiên gồm 20 ký tự .
- /Section II /- Gắn vào sau chuỗi ngẫu nhiên này chuỗi mặc định “**CD Ripper**” .
- /Section III /- Gắn chuỗi này vào sau chuỗi mặc định thứ hai .
- /Section IV /- Tạo chuỗi MD5Hash của chuỗi này.
- /Section V /- Lấy các ký tự ở vị trí chẵn chuỗi MD5Hash theo dạng “XXXX-XXXX-XXXX”
- /Section VI /- Gắn chuỗi này vào sau 20 ký tự ngẫu nhiên đầu tiên : Chuỗi Serial thực .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.imtoo.com/dvd-ripper.html
Production :	ImTOO Software Studio .
SoftWare :	ImTOO DVD Ripper 3.0.12 b 331
Copyright by :	Copyright © 2003-2004 ImTOO Software Studio . All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

ImTOO DVD Ripper 3.0.12 b 331

ImTOO DVD Ripper is a DVD ripping tool easy to use with high ripping speed. It can backup your favorite DVD movie into almost all popular video formats such as VCD, SVCD, DivX, MPEG, AVI etc.

Compare with other DVD ripper, ImTOO DVD Ripper has more settings you can customize and it is easier than ever. Whether you are a veteran or a beginner, you will feel it was developed for you!

ImTOO DVD Ripper provides you with excellent image/sound quality and smaller file size just in a few clicks.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid registration info!**" . Ta tìm được thông báo này tại địa chỉ :

00409B84 . 68 CCF74C00 PUSH vconvert.004CF7CC ; ASCII "Invalid registration info!"

- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

00409B73 . E8 48F1FFFF CALL vconvert.00408CC0 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.

00409B73 . E8 48F1FFFF CALL vconvert.00408CC0 ; <== Set BreakPoint here

00409B78 . E8 D3F6FFFF CALL vconvert.00409250 ; <== Trace Into here

- Dùng F7 trace into ta đến :

00409514 . 83F8 27 CMP EAX,27 ; <== LenS must be 39 charts

00409517 . 0F85 CC050000 JNZ vconvert.00409AE9

- Trace tiếp ta xác định chuỗi dùng để mã hoá :

00409774 . E8 6791FFFF CALL vconvert.004028E0 ; <== ConcatString

===== String to Encrypt =====

Ngay tại đây, quan sát của sổ STACK ta thấy dòng :

00120544 010F4C78 ASCII "12345678901234567890DVD Ripper"

Như vậy, chương trình đã lấy 20 ký tự đầu tiên và ghép với chuỗi mặc định "**DVD Ripper**". Ta có thể cho rằng chương trình sẽ dùng chuỗi này để tiến hành mã hoá, nhưng thực chất không phải vậy, chuỗi này chỉ là một phần của chuỗi thực.

Chương trình sẽ gắn chuỗi này vào sau một chuỗi mặc định thứ hai, chuỗi này gồm 23 ký tự . Có giá trị như sau :

*0x31, 0x44, 0x01, 0x44, 0x03,
0x52, 0x05, 0x70, 0x07, 0x65,
0x09, 0x56, 0x02, 0x20, 0x04,
0x69, 0x06, 0x70, 0x08, 0x72,
0x0A, 0x30, 0x30*

Như vậy, chuỗi được dùng để mã hoá sẽ có chiều dài : 53 ký tự .

===== String to Encrypt =====

- Trace tiếp ta đến quá trình mã hoá :

004097C3 . E8 38B0FFFF CALL vconvert.00404800 ; <== Encrypt

===== EnCrypt =====

00404814 |. E8 47FFFFFF CALL vconvert.00404760 ; <== MD5Hash

===== MD5Hash =====

0040476D |. E8 4EF5FFFF CALL vconvert.00403CC0 ; <== MD5Start

00404772 |. 8B4F 38 MOV ECX,DWORD PTR DS:[EDI+38]

00404775 |. 8BC1 MOV EAX,ECX

00404777 |. 83C4 04 ADD ESP,4

0040477A |. 8D70 01 LEA ESI,DWORD PTR DS:[EAX+1]

0040477D |. 8D49 00 LEA ECX,DWORD PTR DS:[ECX]

00404780 |> 8A10 /MOV DL,BYTE PTR DS:[EAX]

00404782 |. 40 |INC EAX

00404783 |. 84D2 |TEST DL,DL

00404785 |.^ 75 F9 |JNZ SHORT vconvert.00404780

00404787 |. 2BC6 SUB EAX,ESI

```

00404789 |. 50      PUSH EAX
0040478A |. 51      PUSH ECX
0040478B |. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
0040478F |. 51      PUSH ECX
00404790 |. E8 3BFEEEEEE CALL vconvert.004045D0          ; <== MD5Update
00404795 |. 8D5424 18  LEA EDX,DWORD PTR SS:[ESP+18]
00404799 |. 52      PUSH EDX
0040479A |. 8D5F 05  LEA EBX,DWORD PTR DS:[EDI+5]
0040479D |. 53      PUSH EBX
0040479E |. E8 EDFEEEEEE CALL vconvert.00404690          ; <== MD5Finished
----- ==== MD5Hash ====
----- ==== EnCrypt ====

```

- Ké đó, chương trình tiến hành mã hoá chuỗi MD5Hash này bằng cách (1) chọn lấy các ký tự ở vị trí chẵn (2) các vị trí thứ **4, 9 14** sẽ là ký tự “-“. Như vậy chuỗi này sẽ có dạng “XXXX-XXXX-XXXX”

```

00409800 >/85DB    TEST EBX,EBX
00409802 .|0F8C D7020000 JL vconvert.00409ADF
00409808 .|8B4424 38  MOV EAX,DWORD PTR SS:[ESP+38]
0040980C .|3B58 F4  CMP EBX,DWORD PTR DS:[EAX-C]
0040980F .|0F8F CA020000 JG vconvert.00409ADF
00409815 .|8A0C03  MOV CL,BYTE PTR DS:[EBX+EAX]
00409818 .|8B46 FC  MOV EAX,DWORD PTR DS:[ESI-4]
0040981B .|8B6E F4  MOV EBP,DWORD PTR DS:[ESI-C]
0040981E .|884C24 2B  MOV BYTE PTR SS:[ESP+2B],CL
00409822 .|B9 01000000 MOV ECX,1
..... Cutting .....
0040985B .|8BC3      MOV EAX,EBX
0040985D .|99      CDQ
0040985E .|2BC2      SUB EAX,EDX
00409860 .|D1F8      SAR EAX,1
00409862 .|40      INC EAX
00409863 .|25 03000080 AND EAX,80000003
00409868 .|897E F4  MOV DWORD PTR DS:[ESI-C],EDI
0040986B .|C60437 00  MOV BYTE PTR DS:[EDI+ESI],0
0040986F .|79 05      JNS SHORT vconvert.00409876
00409871 .|48      DEC EAX
00409872 .|83C8 FC  OR EAX,FFFFFFFC
00409875 .|40      INC EAX
00409876 >|75 14    JNZ SHORT vconvert.0040988C
00409878 .|6A 01      PUSH 1
0040987A .|68 A4F74C00 PUSH vconvert.004CF7A4
0040987F .|8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18]
00409883 .|E8 888EFFEE CALL vconvert.00402710
00409888 .|8B7424 10  MOV ESI,DWORD PTR SS:[ESP+10]
0040988C >|83C3 02  ADD EBX,2
0040988F .|83FB 20  CMP EBX,20
00409892 .^|0F8C 68FFFFFF JL vconvert.00409800

```

- Sau đó chuỗi này sẽ được gắn vào sau 20 ký tự đầu tiên của chuỗi FakeSerial nhập. Hay nói cách khác, chương trình sẽ sử dụng 20 ký tự đầu tiên của chuỗi Serial để làm chuỗi mã hoá.

```

00409959 . E8 82EBFFFF CALL vconvert.004084E0          ; <== Real Serial
0040995E . 8B7C24 20  MOV EDI,DWORD PTR SS:[ESP+20]      ; <== Fake Serial
00409962 . 8B7424 10  MOV ESI,DWORD PTR SS:[ESP+10]      ; <== Real Serial

```

```

00409966 . 57      PUSH EDI
00409967 . 56      PUSH ESI
00409968 . E8 EC080A00 CALL vconvert.004AA259 ; <== Compare

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 5ZX8ELNWXAEMHTHF3J70D502-BD4B-05A0-F815

III – KeyGen :

- /Section I /- Tạo chuỗi ngẫu nhiên gồm 20 ký tự .
- /Section II /- Gắn vào sau chuỗi ngẫu nhiên này chuỗi mặc định “**DVD Ripper**” .
- /Section III /- Gắn chuỗi này vào sau chuỗi mặc định thứ hai .
- /Section IV /- Tạo chuỗi MD5Hash của chuỗi này.
- /Section V /- Lấy các ký tự ở vị trí chẵn chuỗi MD5Hash theo dạng “**XXXX-XXXX-XXXX**”
- /Section VI /- Gắn chuỗi này vào sau 20 ký tự ngẫu nhiên đầu tiên : Chuỗi Serial thực .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://dev4pc.com
Production :	SDS Software
SoftWare :	Installer2go 4.0.6
Copyright by :	Copyright © SDS Software 1997-2003. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Installer2go 4.0.6

Installer2Go is a powerful, easy to use, highly interactive tool for creating reliable Windows Installer packages that meet the Windows 2000 logo certification guidelines.

Installer2Go is script-free and does not require the user to have any programming background, nor does it require the user to learn any application-specific language! It allows you to use a point-and-click interface to create and manage installation projects based on your business needs. Besides that this tool will allow you to create self-extracting executable files for distribution over the Web, Internet, Intranet and e-mail your data files, graphic images or whatever else you want to distribute!

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử User và Fake Serial, ta không nhận được thông báo gì . Và cũng như ta không thể tìm thấy chuỗi thông báo trong Olly . Tuy nhiên, trong quá trình tìm kiếm ta tìm được chuỗi :

```

004047A1 |. 51      PUSH ECX          ; /BufSize
004047A2 |. 56      PUSH ESI          ; |Buffer
004047A3 |. 6A 01    PUSH 1           ; |ValueType = REG_SZ
004047A5 |. 50      PUSH EAX          ; |Reserved
004047A6 |. 68 84034F00  PUSH builder.004F0384 ; |ValueName = "username"
004047AB |. 52      PUSH EDX          ; |hKey
004047AC |. FFD3    CALL EBX          ; |\RegSetValueExA
.....
004047C0 |. 51      PUSH ECX          ; /BufSize
004047C1 |. 52      PUSH EDX          ; |Buffer
004047C2 |. 6A 01    PUSH 1           ; |ValueType = REG_SZ
004047C4 |. 50      PUSH EAX          ; |Reserved => 0
004047C5 |. 8B4424 28  MOV EAX,DWORD PTR SS:[ESP+28] ; |
004047C9 |. 68 90034F00  PUSH builder.004F0390 ; |ValueName = "regcode"
004047CE |. 50      PUSH EAX          ; |hKey
004047CF |. FFD3    CALL EBX          ; |\RegSetValueExA
- Đây chính là đoạn CODE dùng để lưu "username" và "regcode" nếu hai thông số này được chấp nhận
. Dò ngược lên trên ta thấy đoạn CODE :
00404755 |. E8 86FFFFFF  CALL builder.004046E0 ; <== Set breakPoint here

0040475A |. 84C0      TEST AL,AL        ; <== If Serial Correct AL = 0x1
0040475C |. 0F84 EA010000 JE builder.0040494C ; <== Open Registry and Saved
DATA

```

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Dùng F7 trace into, ta đến đoạn CODE kiểm tra chiều dài chuỗi Serial nhập :

```

0040470A |. 83F9 0A    CMP ECX,0A        ; <== Len.S must be 10 charts
0040470D |. 75 1D      JNZ SHORT builder.0040472C ; <== if Correct nex check

```

- Trace tiếp ta đến :

```

0040471F |. E8 9CAC0900  CALL builder.0049F3C0 ; <== Trace Into here

```

- Dùng F7 để trace into ta đến quá trình mã hoá hai ký tự đầu tiên của chuỗi Serial :

```

0049F3EB |. 8A4B 01    MOV CL,BYTE PTR DS:[EBX+1] ; <== S[1]
0049F3EE |. 55      PUSH EBP
0049F3EF |. 51      PUSH ECX
0049F3F0 |. E8 9BFFFFFF  CALL builder.0049F390 ; <== Temp_02 = S[1] - 0x30
0049F3F5 |. 8A13      MOV DL,BYTE PTR DS:[EBX]   ; <== S[0]
0049F3F7 |. 8BE8      MOV EBP,EAX
0049F3F9 |. 52      PUSH EDX
0049F3FA |. E8 91FFFFFF  CALL builder.0049F390 ; <== Temp_01 = S[0] - 0x30
0049F3FF |. C1E0 04    SHL EAX,4          ; <== Temp_01 = Temp_01 * 0x10
0049F402 |. 03E8      ADD EBP,EAX        ; <== Temp_01 = Temp_01 + Temp_02
0049F404 |. 8B4424 24  MOV EAX,DWORD PTR SS:[ESP+24]
0049F408 |. 81F5 FF000000 XOR EBP,0FF     ; <== Temp_01 = Temp_01 xor 0xFF
0049F40E |. 83C4 08    ADD ESP,8
0049F411 |. 83ED 55    SUB EBP,55         ; <== Temp_01 = Temp_01 - 0x55
0049F414 |. 8928      MOV DWORD PTR DS:[EAX],EBP ; <== Temp_01
- Nếu xem xét một cách đơn giản, nếu hai ký tự đầu tiên của Serial là NUMBER, thì hai ký tự này được giữ nguyên ( Ví dụ, nếu chuỗi vào là "25" thì chuỗi xuất cũng là "25")
- Tiếp tục trace ta đến :
0049F28F |. E8 0CFFFFFF  CALL builder.0049F1A0 ; <== Name : 10 chart

```

===== NOTE =====

- Quá trình này có thể được diễn giải như sau : Name nhập sẽ được kiểm tra chiều dài . Và được chia làm hai trường hợp (1) chiều dài lớn hơn hay bằng 10 ký tự, và (2) chiều dài nhỏ hơn 10 ký tự . (1) Nếu chiều dài nhập lớn hơn hay bằng 10 ký tự thì chuỗi Name dùng để mã hoá sẽ được cắt từ chuỗi này với chiều dài là 10. (2) Nếu chiều dài nhập nhỏ hơn 10 ký tự thì chuỗi Name dùng để mã hoá sẽ được gán thêm chuỗi Name gốc sao cho chiều dài chuỗi Name mới phải là 10 ký tự . Ví dụ : (1) Nếu chuỗi nhập là 13 ký tự “1234567890123” thì chuỗi dùng để mã hoá là “1234567890” ; (2) Nếu chuỗi nhập là 6 ký tự ””ABCDEF” thì chuỗi dùng để mã hoá sẽ là “ABCDEFABCD” .

- Đoạn CODE này được viết lại như sau :

```

if( LenUser >= 10)
{
    lstrcpy(reaTempName, reaName, 11);
}
else
{
    i=0;
    while ( i < (10 / LenUser))
    {

        lstrcpy(reaTemp, reaName, LenUser + 1);
        lstrcat(reaTempName, reaTemp);
        i++;
    }
    LenTempName = lstrlen(reaTempName);
    i=0;
    while ( i < ( 10 - LenTempName))
    {
        reaTempName[LenTempName + i] = reaName[i];
        i++;
    }
}

```

===== NOTE =====

- Tiếp tục trace ta đến :

0049F426 |. E8 35FFFFFF CALL builder.0049F260 ; <== Trace Into here

- Sau khi dùng F7 để trace into, ta sẽ đến đoạn chương trình tạo giá trị mặc định :

0049F2DD |> /8A0C28 /MOV CL,BYTE PTR DS:[EAX+EBP] ; <== "andrey&pasha" :
Chr[i]

0049F2E0 . 8BD7 MOV EDX,EDI	; <== Default Value : ValueD
0049F2E2 . 81E1 FF000000 AND ECX,0FF	; <== Chr[i] and 0xFF
0049F2E8 . 81E2 FF000000 AND EDX,0FF	; <== Temp = ValueD and 0xFF
0049F2EE . 33CA XOR ECX,EDX	; <== Value = (Chr[i] and 0xFF) xor Temp
0049F2F0 . C1EF 08 SHR EDI,8	; <== ValueD = ValueD / 0x100
0049F2F3 . 8B0C8D 986A4F> MOV ECX,DWORD PTR DS:[ECX*4+4F6A98] ;	<== Value at add.
0049F2FA . 33F9 XOR EDI,ECX	; <== ValueD = ValueD xor Value
0049F2FC . 40 INC EAX	; <== i++
0049F2FD . 3BC3 CMP EAX,EBX	; <== while (i < 0xA)
0049F2FF .^\7C DC JL SHORT builder.0049F2DD ;	<== Continue Loop

- Kết thúc quá trình này ta luôn có giá trị mặc định : **0x4D6BA4BE** . Giá trị này là giá trị đầu tiên của quá trình mã hoá tạo chuỗi Serial thực :

```

0049F306 |. 8B6C24 10 MOV EBP,DWORD PTR SS:[ESP+10] ; <== Name after Modificaiton : mU[]
0049F30A |. 8D53 FE LEA EDX,DWORD PTR DS:[EBX-2] ; <== Len.U
0049F30D |. 8D4E 02 LEA ECX,DWORD PTR DS:[ESI+2]
0049F310 |. 2BEE SUB EBP,ESI ; <== Next location of Serial
0049F312 |. 895424 2C MOV DWORD PTR SS:[ESP+2C],EDX
0049F316 |> 8A0429 /MOV AL,BYTE PTR DS:[ECX+EBP] ; <== mU[i+2]
0049F319 |. 8BD7 |MOV EDX,EDI ; <== Default Value : ValueD
0049F31B |. 25 FF000000 |AND EAX,0FF ; <== mU[i+2] and 0xFF
0049F320 |. 81E2 FF000000 |AND EDX,0FF ; <== Temp = ValueD and 0xFF
0049F326 |. 33C2 |XOR EAX,EDX ; <== Temp = (mU[i+2] and 0xFF) xor Temp
0049F328 |. 33D2 |XOR EDX,EDX
0049F32A |. C1EF 08 |SHR EDI,8 ; <== ValueD = ValueD / 0x100
0049F32D |. 8B0485 986A4F>|MOV EAX,DWORD PTR DS:[EAX*4+4F6A98] ; <== Temp:Value at add.
0049F334 |. BE 24000000 |MOV ESI,24
0049F339 |. 33F8 |XOR EDI,EAX ; <== ValueD = ValueD xor Temp
0049F33B |. 8BC7 |MOV EAX,EDI ; <== ValueD
0049F33D |. F7F6 |DIV ESI ; <== Temp = ValueD % 0x24
0049F33F |. 83FA 0A |CMP EDX,0A ; <== if ( Temp < 0xA )
0049F342 |. 73 05 |JNB SHORT builder.0049F349 ; <== then
0049F344 |. 80C2 30 |ADD DL,30 ; <== Temp = Temp + 0x30
0049F347 |. EB 03 |JMP SHORT builder.0049F34C ; <== else
0049F349 |> 80C2 37 |ADD DL,37 ; <== Temp = Temp + 0x37
0049F34C |> 8B4424 2C |MOV EAX,DWORD PTR SS:[ESP+2C] ; <== Len.U
0049F350 |. 8811 |MOV BYTE PTR DS:[ECX],DL ; <== Saved this value
0049F352 |. 41 |INC ECX ; <== Next location of Serial
0049F353 |. 48 |DEC EAX ; <== LoopNumbers --
0049F354 |. 894424 2C |MOV DWORD PTR SS:[ESP+2C],EAX ; <== LoopNumbers
0049F358 |.^ 75 BC |JNZ SHORT builder.0049F316 ; <== Loop Until NumbersLoop = 0x0
- Như ta đã biết, hai ký tự đầu tiên của Serial là hai ký tự ngẫu nhiên, chiều dài của chuỗi Serial là 10 ký tự, như vậy vòng lặp này chỉ có 8 lần, tương ứng với 8 ký tự còn lại .
- Quá trình mã hoá của chương trình này chủ yếu dựa vào bảng mặc định . Bảng này có tất cả 256 giá trị, tương ứng với 256 ký tự trong bảng mã ASCII .

```

/*/*/*/ - SERIAL tương ứng với USER :

III – KeyGen :

- /Section 1/- Tạo chuỗi Name dùng để mã hoá
 - /Section 2/- Xây dựng bảng giá trị mặc định
 - /Section 3/- Tạo hai giá trị đầu tiên của chuỗi Serial ngẫu nhiên (NUMBER)
 - /Section 4/- Tính 8 ký tự còn lại .

IV – SourceCode (VC++) :

```
char reaName[64]={0};  
char reaSerial[64]={0};  
char reaTempName[64]={0};  
char reaTemp[128]={0};  
char reaDefaultString[40]="1478523690";  
unsigned long reaDefaultTable[256]= {  
    0x00000000, 0x77073096, 0xEE0E612C, 0x990951BA, 0x076DC419, 0x706AF48F,  
    0xE963A535, 0x9E6495A3,
```

0x0EDB8832, 0x79DCB8A4, 0xE0D5E91E, 0x97D2D988, 0x09B64C2B, 0x7EB17CBD,
0xE7B82D07, 0x90BF1D91,
 0x1DB71064, 0x6AB020F2, 0xF3B97148, 0x84BE41DE, 0x1ADAD47D, 0x6DDDE4EB,
0xF4D4B551, 0x83D385C7,
 0x136C9856, 0x646BA8C0, 0xFD62F97A, 0x8A65C9EC, 0x14015C4F, 0x63066CD9,
0xFA0F3D63, 0x8D080DF5,
 0x3B6E20C8, 0x4C69105E, 0xD56041E4, 0xA2677172, 0x3C03E4D1, 0x4B04D447,
0xD20D85FD, 0xA50AB56B,
 0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940, 0x32D86CE3, 0x45DF5C75,
0xDCD60DCF, 0xABD13D59,
 0x26D930AC, 0x51DE003A, 0xC8D75180, 0xBFD06116, 0x21B4F4B5, 0x56B3C423,
0xCFBA9599, 0xB8BDA50F,
 0x2802B89E, 0x5F058808, 0xC60CD9B2, 0xB10BE924, 0x2F6F7C87, 0x58684C11,
0xC1611DAB, 0xB6662D3D,
 0x76DC4190, 0x01DB7106, 0x98D220BC, 0xEFD5102A, 0x71B18589, 0x06B6B51F,
0x9FBFE4A5, 0xE8B8D433,
 0x7807C9A2, 0x0F00F934, 0x9609A88E, 0xE10E9818, 0x7F6A0DBB, 0x086D3D2D,
0x91646C97, 0xE6635C01,
 0x6B6B51F4, 0x1C6C6162, 0x856530D8, 0xF262004E, 0x6C0695ED, 0x1B01A57B,
0x8208F4C1, 0xF50FC457,
 0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C, 0x62DD1DDF, 0x15DA2D49,
0x8CD37CF3, 0xFBD44C65,
 0x4DB26158, 0x3AB551CE, 0xA3BC0074, 0xD4BB30E2, 0x4ADFA541, 0x3DD895D7,
0xA4D1C46D, 0xD3D6F4FB,
 0x4369E96A, 0x346ED9FC, 0xAD678846, 0xDA60B8D0, 0x44042D73, 0x33031DE5,
0xAA0A4C5F, 0xDD0D7CC9,
 0x5005713C, 0x270241AA, 0xBE0B1010, 0xC90C2086, 0x5768B525, 0x206F85B3,
0xB966D409, 0xCE61E49F,
 0x5EDEF90E, 0x29D9C998, 0xB0D09822, 0xC7D7A8B4, 0x59B33D17, 0x2EB40D81,
0xB7BD5C3B, 0xC0BA6CAD,
 0xEDB88320, 0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A, 0xEAD54739, 0x9DD277AF,
0x04DB2615, 0x73DC1683,
 0xE3630B12, 0x94643B84, 0xD6D6A3E, 0x7A6A5AA8, 0xE40ECF0B, 0x9309FF9D,
0x0A00AE27, 0x7D079EB1,
 0xF00F9344, 0x8708A3D2, 0x1E01F268, 0x6906C2FE, 0xF762575D, 0x806567CB,
0x196C3671, 0xE6B06E7,
 0xFED41B76, 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC, 0xF9B9DF6F, 0x8EBEEFF9,
0x17B7BE43, 0x60B08ED5,
 0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDFF252, 0xD1BB67F1, 0xA6BC5767,
0x3FB506DD, 0x48B2364B,
 0xD80D2BDA, 0xAF0A1B4C, 0x36034AF6, 0x41047A60, 0xDF60EFC3, 0xA867DF55,
0x316E8EEF, 0x4669BE79,
 0xCB61B38C, 0xBC66831A, 0x256FD2A0, 0x5268E236, 0xCC0C7795, 0xBB0B4703,
0x220216B9, 0x5505262F,
 0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92, 0x5CB36A04, 0xC2D7FFA7, 0xB5D0CF31,
0x2CD99E8B, 0x5BDEAE1D,
 0x9B64C2B0, 0xEC63F226, 0x756AA39C, 0x026D930A, 0x9C0906A9, 0xEB0E363F,
0x72076785, 0x05005713,
 0x95BF4A82, 0xE2B87A14, 0x7BB12BAE, 0x0CB61B38, 0x92D28E9B, 0xE5D5BE0D,
0x7CDCEFB7, 0x0BDDBDF21,
 0x86D3D2D4, 0xF1D4E242, 0x68DDB3F8, 0x1FDA836E, 0x81BE16CD, 0xF6B9265B,
0x6FB077E1, 0x18B74777,

```

0x88085AE6, 0xFF0F6A70, 0x66063BCA, 0x11010B5C, 0x8F659EFF, 0xF862AE69,
0x616BFFD3, 0x166CCF45,
    0xA00AE278, 0xD70DD2EE, 0x4E048354, 0x3903B3C2, 0xA7672661, 0xD06016F7,
0x4969474D, 0x3E6E77DB,
    0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0, 0xA9BCAE53, 0xDEBB9EC5,
0x47B2CF7F, 0x30B5FFE9,
    0xBDDBDF21C, 0xCABAC28A, 0x53B39330, 0x24B4A3A6, 0xBAD03605, 0xCDD70693,
0x54DE5729, 0x23D967BF,
    0xB3667A2E, 0xC4614AB8, 0x5D681B02, 0x2A6F2B94, 0xB40BBE37, 0xC30C8EA1,
0x5A05DF1B, 0x2D02EF8D
};

int LenUser=0,LenTempName = 0;
int i=0;
int Temp=0, Value=0;
unsigned long int ValueD=0;

LenUser=GetDlgItemText(IDC_Name, reaName, 64);
srand( (unsigned)time( NULL ) );
if (LenUser < 1)
{
    MessageBox("----- Your name atleast 1 chart ----- ", "Hey !!");
Please input your name again !! ");
}
else
{
    reaSerial[0] = reaDefaultString[rand()%10];
    reaSerial[1] = reaDefaultString[rand()%10];

    if ( LenUser >= 10 )
    {
        lstrcpy(reaTempName, reaName, 11);
    }
    else
    {
        i=0;
        while ( i < (10 / LenUser) )
        {
            lstrcpy(reaTemp, reaName, LenUser + 1);
            lstrcat(reaTempName, reaTemp);
            i++;
        }
        LenTempName = strlen(reaTempName);
        i=0;
        while ( i < ( 10 - LenTempName ) )
        {
            reaTempName[LenTempName + i] = reaName[i];
            i++;
        }
    }
    i=2;    ValueD = 0x4D6BA4BE;
    while ( i < 10 )
    {
        Temp = reaTempName[i] ^ ( ValueD & 0xFF );

```

```

ValueD = (ValueD / 0x100) ^ reaDefaultTable[Temp];
Value = ValueD % 0x24;
if ( Value < 0xA )
{
    Value = Value + 0x30;
}
else
{
    Value = Value + 0x37;
}
reaSerial[i] = Value;
i++;
}

SetDlgItemText(IDC_Serial, reaSerial);
}

```

V – End of Tut :

- Finished – ***12/07/2004***

Reverse Engineering Association

SoftWare

Homepage :	http://www.softheap.com or http://www.getfreefile.com
Production :	Ixis Research, Ltd.
SoftWare :	Mail Bomber 9.1
Copyright by :	Copyright © 2004 Ixis Research, Ltd. All Rights Reserved.
Type :	Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

Mail Bomber 9.1

Mail Bomber is intended for sending electronic mail utilizing subscription-based mailing lists. It is a useful tool for anyone who needs to send requested newsletters and notifications to a large number of subscribers. This program allows you create and manage subscription-based mailing lists, and generate personalized messages from predefined templates while sending. You can create separate subscription-based mailing lists, which contain information about your subscribers, messages, and SMTP servers used for sending email messages. The interface of the program is very simple and easy to learn - nearly all functions can be performed using hotkeys on the keyboard. This software is a handy tool for keeping feedback from your clients or users, its a mailing list manager, a simple and handy address book, and a simple email program.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "*Sorry, this registration code is invalid*" . Ta không thể tìm được chuỗi thông báo này bằng phương pháp tìm kiếm thông thường . Ta có thể dùng phương pháp STACK hay ở đây ta nhận thấy chương trình có khoảng thời gian dừng trước khi hiện NAG, như vậy ta có thể tìm đến hàm *API kernel32.Sleep* :

004E1756 |. E8 4DCAF2FF CALL <JMP.&kernel32.Sleep> ; \Sleep

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

004E1672 |. E8 A937F2FF CALL mailsend.00404E20 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

004E169D |. E8 4E8DFFFF CALL mailsend.004DA3F0 ; <== Trace Into

===== *Trace Into* =====

004DA423 |. E8 10A8F2FF CALL mailsend.00404C38 ; <== Len.S

004DA428 |. 83F8 10 CMP EAX,10 ; <== Must be 16 charts

004DA42B |. 75 36 JNZ SHORT mailsend.004DA463

004DA42D |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA430 |. 8078 01 32 CMP BYTE PTR DS:[EAX+1],32 ; <== S[1] == 0x32

004DA434 |. 75 2D JNZ SHORT mailsend.004DA463

004DA436 |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA439 |. 8078 02 35 CMP BYTE PTR DS:[EAX+2],35 ; <== S[2] == 0x35

004DA43D |. 75 24 JNZ SHORT mailsend.004DA463

004DA43F |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA442 |. 8078 05 32 CMP BYTE PTR DS:[EAX+5],32 ; <== S[5] == 0x32

004DA446 |. 75 1B JNZ SHORT mailsend.004DA463

004DA448 |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA44B |. 8078 07 35 CMP BYTE PTR DS:[EAX+7],35 ; <== S[7] == 0x35

004DA44F |. 75 12 JNZ SHORT mailsend.004DA463

004DA451 |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA454 |. 8078 09 30 CMP BYTE PTR DS:[EAX+9],30 ; <== S[9] == 0x30

004DA458 |. 75 09 JNZ SHORT mailsend.004DA463

004DA45A |. 8B45 F8 MOV EAX,[LOCAL.2]

004DA45D |. 8078 0B 31 CMP BYTE PTR DS:[EAX+B],31 ; <== S[11] == 0x31

004DA461 |. 74 04 JE SHORT mailsend.004DA467

===== *Trace Into* =====

- Quá trình mã hoá này cực kỳ đơn giản, chỉ là trong chuỗi Serial ở 6 vị trí xác định trước là các ký tự mặc định . Các ký tự còn lại là ngẫu nhiên .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 125OZ2Y51091CGHY

III – End of Tut :

- Finished – *September 14, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.audiotoolsfactory.com
Production :	audiotoolsfactory.com.
SoftWare :	MP3 Cutter Joiner 1.00
Copyright by :	Copyright © 2004 audiotoolsfactory.com. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

MP3 Cutter Joiner 1.00

MP3 Cutter Joiner is a powerful and easy-to-use audio editor, which builds audio cutter and joiner into one. Supports MP3, WAV, WMA and OGG format. You can cut a new small file from a large audio file, and you can also merge multiple files to a large new one. Supports batch cutting. Can set the cutting/joining's start-time/end-time just when pre-listening to the song. Cutting/joining high precision and no quality is lost!.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid register code! Please retry!**" . Ta tìm được chuỗi thông báo này tại địa chỉ :
004BAACD . B8 2CAB4B00 MOV EAX,MP3_Cutt.004BAB2C ; |ASCII "Invalid register code! Please retry!"
- Dò ngược lên trên ta đặt BreakPoint tại :
004BA96A . 833D 0CEE4C00>CMP DWORD PTR DS:[4CEE0C],3 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình sẽ kiểm tra các ký tự của chuỗi Serial nhập . Các ký tự này phải nằm trong khoảng (0-9) :

```

004BA9FD >/8B4D F8    MOV ECX,DWORD PTR SS:[EBP-8]      ; <== / form here
004BAA00 .|0FB64C11 FF  MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== |
004BAA05 .|83F9 30    CMP ECX,30                         ; <== |
004BAA08 .|7C 08     JL SHORT MP3_Cutt.004BAA12        ; <== |
004BAA0A .|8B5D F8    MOV EBX,DWORD PTR SS:[EBP-8]      ; <== |
004BAA0D .|83F9 39    CMP ECX,39                         ; <== |
004BAA10 .|7E 1A     JLE SHORT MP3_Cutt.004BAA2C       ; <== |
004BAA12 >|6A 00     PUSH 0                           ; /Arg1 = 00000000
004BAA14 .|66:8B0D 20AB4>MOV CX,WORD PTR DS:[4BAB20]   ; |

```

```

004BAA1B .|B2 02      MOV DL,2          ; |
004BAA1D .|B8 2CAB4B00  MOV EAX,MP3_Cutt.004BAB2C ; |ASCII "Invalid register code!
Please retry!"|
004BAA22 .|E8 050AF8FF  CALL MP3_Cutt.0043B42C ; |\MP3_Cutt.0043B42C
004BAA27 .|E9 AB000000  JMP MP3_Cutt.004BAAD7 ; <== |
004BAA2C >|42       INC EDX          ; <== |
004BAA2D .|48       DEC EAX          ; <== |
004BAA2E .^|\75 CD     JNZ SHORT MP3_Cutt.004BA9FD ; <== \ to here

```

- Quá trình mã hoá của chương trình diễn ra đơn giản :

```

004BAA43 >/8B4D FC    MOV ECX,DWORD PTR SS:[EBP-4] ; <== / from here
004BAA46 .|0FB64C11 FF  MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== |
004BAA4B .|03D9       ADD EBX,ECX          ; <== | Cummulate value all charts
004BAA4D .|42       INC EDX          ; <== | of User input : CumV
004BAA4E .|48       DEC EAX          ; <== |
004BAA4F .^|\75 F2     JNZ SHORT MP3_Cutt.004BAA43 ; <== \ to here
004BAA51 > 69C3 9FF30700 IMUL EAX,EBX,7F39F ; <== CumV = CumV * 0x7F39F
004BAA57 . 83C0 20    ADD EAX,20          ; <== CumV = CumV + 0x20
004BAA5A . D1F8       SAR EAX,1          ; <== CumV = CumV / 2
004BAA5C . 79 03       JNS SHORT MP3_Cutt.004BAA61
004BAA5E . 83D0 00    ADC EAX,0
004BAA61 > 8BD8       MOV EBX,EAX
004BAA63 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8] ; <== Fake Serial
004BAA66 . E8 E5E3F4FF  CALL MP3_Cutt.00408E50 ; <== Convert to HEX Value : fV
004BAA6B . 3BD8       CMP EBX,EAX          ; <== if ( fV == CumV )
004BAA6D . 75 53       JNZ SHORT MP3_Cutt.004BAAC2 ; <== Congrat !!!!

/*/*/* - SERIAL tương ứng :
User : REA-cRaCkErTeAm           Serial : 327002188

```

III – KeyGen :

- /Section I /- Tính giá trị cộng dồn các ký tự của chuỗi U nhập .
- /Section II /- Tính giá trị $((CumV * 0x7F39F) + 0x20)/2$
- /Section III /- Chuyển giá trị này sang chuỗi theo định dạng “%lu”

IV – End of Tut :

- Finished – *Sept. 02, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.audiotoolsfactory.com
Production :	audiotoolsfactory.com.
SoftWare :	MP3 RM Converter 1.00
Copyright by :	Copyright © 2004 audiotoolsfactory.com. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 6.0 - 7.0

Crack Tool : OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
 Unpack : N/A
 Request : Correct Serial / KeyGen

MP3 RM Converter 1.00

MP3 RM Converter supports batch conversion between MP3, WAV, WMA and OGG. And more important, MP3 RM Converter supports RM format, you can convert your MP3, WMA, WAV and OGG files to RM files. MP3 RM Converter converts the audio files digitally-not through the soundcard-which enables you to make perfect copies of the originals. Convert from one to another directly and on-the-fly (without temporary files produced). Converting your files is just a button click away because this program is user-friendly-directing the users from start to finish. You'll be an experter in no time! Run stably on Windows 98/NT/Me/2K/XP/2003.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid register code! Please retry!**" . Ta tìm được chuỗi thông báo này tại địa chỉ :

004C0301 . B8 60034C00 MOV EAX,MP3_RM_C.004C0360 ; |ASCII "Invalid register code!
Please retry!"

- Dò ngược lên trên ta đặt BreakPoint tại :

004C019E . 833D 348E4C00>CMP DWORD PTR DS:[4C8E34],3 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình sẽ kiểm tra các ký tự của chuỗi Serial nhập . Các ký tự này phải nằm trong khoảng (0-9) :

```

004C0231 >/8B4D F8    MOV ECX,DWORD PTR SS:[EBP-8]
004C0234 . |0FB64C11 FF  MOVZX ECX,BYTE PTR DS:[ECX+EDX-1]
004C0239 . |83F9 30    CMP ECX,30
004C023C . |7C 08     JL SHORT MP3_RM_C.004C0246
004C023E . |8B5D F8    MOV EBX,DWORD PTR SS:[EBP-8]
004C0241 . |83F9 39    CMP ECX,39
004C0244 . |7E 1A     JLE SHORT MP3_RM_C.004C0260
004C0246 >|6A 00     PUSH 0                                ; /Arg1 = 00000000
004C0248 . |66:8B0D 54034>MOV CX,WORD PTR DS:[4C0354]      ; |
004C024F . |B2 02     MOV DL,2                                ; |
004C0251 . |B8 60034C00 MOV EAX,MP3_RM_C.004C0360          ; |ASCII "Invalid register
code! Please retry!"
004C0256 . |E8 C9AEF7FF CALL MP3_RM_C.0043B124           ; |\MP3_RM_C.0043B124
004C025B . |E9 AB000000 JMP MP3_RM_C.004C030B
004C0260 >|42       INC EDX
004C0261 . |48       DEC EAX
004C0262 .^|\75 CD   JNZ SHORT MP3_RM_C.004C0231

```

- Quá trình mã hoá của chương trình diễn ra đơn giản :

```

004C0277 >/8B4D FC    MOV ECX,DWORD PTR SS:[EBP-4]        ; <== / from here
004C027A . |0FB64C11 FF  MOVZX ECX,BYTE PTR DS:[ECX+EDX-1]  ; <== | Cummulate value
004C027F . |03D9       ADD EBX,ECX                          ; <== | all charts

```

```

004C0281 . |42      INC EDX          ; <== | of User
004C0282 . |48      DEC EAX          ; <== | input : CumV
004C0283 .^|75 F2    JNZ SHORT MP3_RM_C.004C0277 ; <== \ to here
004C0285 > 69C3 26740700 IMUL EAX,EBX,77426 ; <== CumV = CumV * 0x77426
004C028B . 83C0 52   ADD EAX,52    ; <== CumV = CumV + 0x52
004C028E . D1F8     SAR EAX,1    ; <== CumV = CumV / 2
004C0290 . 79 03     JNS SHORT MP3_RM_C.004C0295
004C0292 . 83D0 00   ADC EAX,0
004C0295 > 8BD8     MOV EBX,EAX
004C0297 . 8B45 F8   MOV EAX,DWORD PTR SS:[EBP-8] ; <== Fake Serial
004C029A . E8 A98BF4FF CALL MP3_RM_C.00408E48 ; <== Convert to HEX Value : fV
004C029F . 3BD8     CMP EBX,EAX ; <== if ( fV == CumV )
004C02A1 . 75 53     JNZ SHORT MP3_RM_C.004C02F6 ; <== Congrat !!!!

/*/*/* - SERIAL tương ứng :
User : REA-cRaCkErTeAm           Serial : 306525006

```

III – KeyGen :

- /Section I/- Tính giá trị cộng dồn các ký tự của chuỗi U nhập .
- /Section II/- Tính giá trị $((CumV * 0x77426) + 0x52)/2$
- /Section III/- Chuyển giá trị này sang chuỗi theo định dạng “%lu”

IV – End of Tut :

- Finished – *Sept. 02, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.xxxx.com
Production :	Dipl. Inform. Bernd Bock.
SoftWare :	MP3StickMan 1.1
Copyright by :	Copyright © 2003 Dipl. Inform. Bernd Bock. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

MP3StickMan 1.1

MP3StickMan manages the MP3 file upload to your USB music stick. Of course you may upload any other format your stick supports as well, e.g. WMA, OGG Vorbis, WAV, etc. Got bored of listening to the same songs over and over again? Ever dreamt of automatically filling your MP3 stick with new, always changing collections of music?

MP3StickMan is the solution: it keeps track of your music directories and remembers which files

you had uploaded already. Each time you upload new music the application will create a new, random subset of files from your music collection, so that the music on your stick is never the same! After a certain number of uploads, when all files have been "used" once on the stick, the first files will be reused. But the composition will always be different. You may create several collections, e.g. depending on your mood. Put your soul music into a "Soul" collection, your hard rock into a "hard rock" collection, and your dance music into "dance music". Whatever your current mood is, choose the appropriate collection and let MP3StickMan fill your stick!

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.

- Trong quá trình tìm kiếm chuỗi thông báo, ta gặp được các dòng sau :

00406F99 . 68 1CE24000 PUSH MP3Stick.0040E21C	; ASCII "RegName"
00406F9E . 68 0CE24000 PUSH MP3Stick.0040E20C	; ASCII "Registration"
00406FAD . 68 04E24000 PUSH MP3Stick.0040E204	; ASCII "RegKey"
00406FB2 . 68 0CE24000 PUSH MP3Stick.0040E20C	; ASCII "Registration"

- Ta nghĩ ngay đến việc chương trình sẽ gọi các DATA được lưu trữ trong REGISTRY sử dụng cho quá trình mã hóa .

- Dò ngược lên trên, ta chọn đặt BreakPoint tại lệnh CALL đầu tiên của FUNCTION này :

00406F60 . E8 290D0000 CALL <JMP.&MFC42.#6334>	; <== Set BreakPoint here
--	---------------------------

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

00406F79 . E8 C2000000 CALL MP3Stick.00407040	; <== Encrypt & Compare
00406F7E . 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]	
00406F82 . C74424 28 000>MOV DWORD PTR SS:[ESP+28],0	
00406F8A . E8 51F2FFFF CALL MP3Stick.004061E0	; <== Return compare value
00406F8F . 85C0 TEST EAX,EAX	; <== if CORRECT EAX = 0x1
00406F91 . 74 34 JE SHORT MP3Stick.00406FC7	; <== then CONGRATS !!!!

- Dùng F7 ta trace into để xem xét quá trình mã hóa và so sánh :

004070A0 . E8 9B000000 CALL MP3Stick.00407140	; <== Trace Into
---	------------------

===== Encrypt & Compare =====

00407159 . 8B46 04 MOV EAX,DWORD PTR DS:[ESI+4]	
0040715C . 8B48 F8 MOV ECX,DWORD PTR DS:[EAX-8]	
0040715F . 85C9 TEST ECX,ECX	; <== User atleast 1 chart
00407161 . 74 78 JE SHORT MP3Stick.004071DB	
00407163 . 8B4E 08 MOV ECX,DWORD PTR DS:[ESI+8]	
00407166 . 8B51 F8 MOV EDX,DWORD PTR DS:[ECX-8]	
00407169 . 85D2 TEST EDX,EDX	; <== Serial atleast 1 chart
0040716B . 74 6E JE SHORT MP3Stick.004071DB	
0040716D . 57 PUSH EDI	
0040716E . 50 PUSH EAX	
0040716F . 8BCE MOV ECX,ESI	
00407171 . E8 8A000000 CALL MP3Stick.00407200	; <== Encrypt

===== Encrypt =====

00407206 . 8A0A MOV CL,BYTE PTR DS:[EDX]	; <== U[0]
00407208 . 84C9 TEST CL,CL	
0040720A . 74 19 JE SHORT MP3Stick.00407225	

```

0040720C |. 56      PUSH ESI
0040720D |> 81E1 FF000000 /AND ECX,0FF ; <== Temp = U[i] & 0xFF
00407213 |. 8B348D 98E440>|MOV ESI,DWORD PTR DS:[ECX*4+40E498] ; <== Value =
defStr[Temp]
0040721A |. 8A4A 01  |MOV CL,BYTE PTR DS:[EDX+1] ; <== U[i]
0040721D |. 33C6  |XOR EAX,ESI ; <== Serial = Serial xor Value
0040721F |. 42    |INC EDX ; <== i++
00407220 |. 84C9  |TEST CL,CL ; <== while ( LenU > 0 )
00407222 |.^ 75 E9  |JNZ SHORT MP3Stick.0040720D ; <== Continue Loop

```

Quá trình mã hoá này cũng tương tự như dạng mã hoá CRC32, chỉ có điều bằng giá trị mặc định đã bị biến đổi.

-----*==== Encrypt =====*

```

00407176 |. 8D4C24 08  LEA ECX,DWORD PTR SS:[ESP+8]
0040717A |. 8BF8      MOV EDI,EAX
0040717C |. E8 15090000 CALL <JMP.&MFC42.#540>
00407181 |. 57      PUSH EDI
00407182 |. 8D5424 0C  LEA EDX,DWORD PTR SS:[ESP+C]
00407186 |. 68 98E84000 PUSH MP3Stick.0040E898 ; ASCII "%x"
0040718B |. 52      PUSH EDX
0040718C |. C74424 20 000>MOV DWORD PTR SS:[ESP+20],0
00407194 |. E8 3F090000 CALL <JMP.&MFC42.#2818> ; <== Convert to String
00407199 |. 8B4C24 14  MOV ECX,DWORD PTR SS:[ESP+14]
0040719D |. 8B46 08  MOV EAX,DWORD PTR DS:[ESI+8]
004071A0 |. 51      PUSH ECX ; /s2
004071A1 |. 50      PUSH EAX ; |s1
004071A2 |. FF15 DCA54000 CALL DWORD PTR DS:[<&MSVCRT._mbscmp>] ; \mbscmp
-----==== Encrypt & Compare =====

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 42c3fff9

III – KeyGen :

/Section 0 /- N/A

IV – End of Tut :

- Finished – *August 13, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.zhangduo.com
Production :	Huntersoft Corporation.
SoftWare :	My Drivers 2005 v3.11 build 2600
Copyright by :	Copyright © 2000-2006 Huntersoft Corporation. All Rights Reserved.
Type :	Name / Serial

Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

My Drivers 2005 v3.11 build 2600

My Drivers enables easy and fast detection, backup and restore of all hardware device drivers currently on your system. Also, you can even find the latest drivers for your hardware and install them onto your computer. With just one or two mouse button clicks, you will have all your hardware devices extracted and backed-up to any folder you want. When you reinstall or upgrade your system, you can restore all drivers by clicking a button. Once after a reboot, all the drivers will be re-installed and will function well. You may also choose to backup a particular driver or all the drivers with an EXE automatic installer. If you have a particular item of hardware that is troublesome, just remove the driver with this software.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid registration Code!**" . Ta tìm được thông báo này tại địa chỉ :
0049E084 68 44E44900 PUSH unpacked.0049E444 ; ASCII "Invalid Registration Code"
- Truy ngược lên trên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :
0049DA4B E8 4C5BFAFF CALL unpacked.0044359C ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP .

0049DA4B E8 4C5BFAFF CALL unpacked.0044359C	; <== Set BreakPoint here
0049DA50 8B45 E4 MOV EAX,DWORD PTR SS:[EBP-1C]	
0049DA53 E8 2074F6FF CALL unpacked.00404E78	; <== Get Length U
0049DA58 05 AE080000 ADD EAX,8AE	; <== Value = LenU + 0x8AE
0049DA5D 8D55 E8 LEA EDX,DWORD PTR SS:[EBP-18]	
0049DA60 E8 87B9F6FF CALL unpacked.004093EC	

- Kế đó, chương trình sẽ so sánh U nhập với một BLACK LIST, nếu U không nằm trong bảng này thì sẽ đến quá trình mã hoá tiếp đó . BLACK LIST :
"DiSTiNCT", "Team iNSaNE", "TNT!2000", "-=Demian/TNT!=-", "-=Demian/TNT!=-", "TSRh TEAM", "ttdown", "TMG", "Gory", "MasterPower", "SnD TeaM", "FFF", "CLUSTER", ".com", "rth77", "team", "Destroy", "Registered", "Orion", "Destroy", "Sponge Uk", "Sponge Uk", "SCF", "Nokedli [Ims]"

- Quá trình mã hoá rất đơn giản, chương trình sẽ chuyển giá trị VALUE tính ở trên sang chuỗi dạng DEC (định dạng "%i") . Chuỗi này được nối vào sau chuỗi mặc định "WDW22" . Kế đến, chương trình chuyển đổi từng của chuỗi U sang mã ASCII (dạng HEX) và nối với chuỗi ở trên :
0049DE14 FFB0 40030000 PUSH DWORD PTR DS:[EAX+340] ; <== "WDW"
0049DE1A 68 FCE24900 PUSH unpacked.0049E2FC ; ASCII "22"
0049DE1F FF75 E8 PUSH DWORD PTR SS:[EBP-18] ; <== Value at DEC value

```

0049DE22 68 08E34900 PUSH unpacked.0049E308
0049DE27 8D95 74FFFFFF LEA EDX,DWORD PTR SS:[EBP-8C]
0049DE2D 8B45 FC      MOV EAX,DWORD PTR SS:[EBP-4]
0049DE30 8B80 00030000 MOV EAX,DWORD PTR DS:[EAX+300]
0049DE36 E8 6157FAFF CALL unpacked.0044359C
0049DE3B 8B85 74FFFFFF MOV EAX,DWORD PTR SS:[EBP-8C]
0049DE41 8D95 78FFFFFF LEA EDX,DWORD PTR SS:[EBP-88]
0049DE47 E8 30FBFFFF CALL unpacked.0049D97C      ;<== Convert charts ASCII value

```

- Trace xuống chút ta đến quá trình so sánh chuỗi :

```

0049DE80 8B95 70FFFFFF MOV EDX,DWORD PTR SS:[EBP-90] ;<== Fake Serial
0049DE86 8B45 EC      MOV EAX,DWORD PTR SS:[EBP-14] ;<== Real Serial
0049DE89 E8 2673F6FF CALL unpacked.004051B4 ;<== Compare

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErGrOuP Serial : WDW222238-5245412D635261436B457247724F7550

III – KeyGen :

- /Section I /- Tính giá trị LenUser = LenUser + 0x8AE và chuyển sang dạng chuỗi DEC .
- /Section II /- Gắn vào sau chuỗi mặc định “**WDW22**” .
- /Section III /- Chuyển các giá trị của chuỗi U sang dạng mã HEX và nối với nhau .
- /Section IV /- Nối hai chuỗi lại.
- /Section V /- Chuỗi Serial thực có dạng “**WDW22XXXX-YYYYYYYYYY**”

IV – End of Tut :

- Finished – *August 26, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.verypdf.com
Production :	verypdf.com Inc.
SoftWare :	PDF Information Editor 1.1
Copyright by :	Copyright © 2001-2003 verypdf.com Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

PDF Information Editor 1.1

PDF Information Editor help users to add or change information in pdf files. These include Document summary, Initial view, Interface, Window option, Pages size, Page contents size, Page rotation, Page contents rotation, Metadata and custom info fields.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo**. Sau khi UnPACK kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**.

- Nhập thử User và Fake Serial (tuy yêu cầu là EMAIL nhưng chương trình không kiểm tra tính hợp lý của EMAIL nên coi như là USER), ta nhận được thông báo "**The serial number is wrong!**". Tuy nhiên không tìm thấy thông báo này trong Olly, vì vậy dụng phương pháp STACK để xác định địa chỉ của thông báo này

```
0052106E B8 B0115200 MOV EAX,Prjpdfin.005211B0 ; ASCII "The serial number is wrong!"
```

- Dò ngược lên trên và đặt BreakPoint tại lệnh CALL đầu tiên của Function này :

```
0052104F E8 0C96F2FF CALL Prjpdfin.0044A660 ; <== Set BreakPoint
```

II – Cracking :

- Load chương trình lên, nhập User và Fake Serial, chương trình dừng lại tại điểm đặt BreakPoint :

```
0052104F E8 0C96F2FF CALL Prjpdfin.0044A660 ; <== Set BreakPoint
```

```
00521054 8B85 F4FBFFFF MOV EAX,DWORD PTR SS:[EBP-40C]
```

```
0052105A E8 356BFFFFFF CALL Prjpdfin.00517B94 ; <== Trace Into
```

- Dùng F7 trace into ta đến quá trình kiểm tra chiều dài Serial nhập :

```
00517BBE E8 FDCFEEFF CALL Prjpdfin.00404BC0 ; <== Get Length Serial
```

```
00517BC3 83F8 10 CMP EAX,10 ; <== Len.S must be 16 charts
```

```
00517BC6 75 69 JNZ SHORT Prjpdfin.00517C31
```

- Quá trình mã hoá của chương trình này rất đơn giản :

```
00517BCC B9 01000000 MOV ECX,1 ; <== Get 1 chart
```

```
00517BD1 BA 02000000 MOV EDX,2 ; <== at [1]
```

```
00517BD6 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== of Serial
```

```
00517BD9 E8 3AD2EEFF CALL Prjpdfin.00404E18 ; <== Cutting chart
```

```
00517BDE 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
```

```
00517BE1 E8 C617EFFF CALL Prjpdfin.004093AC ; <== This chart must be NUMBER
```

```
00517BE6 8BF0 MOV ESI,EAX ; <== Ch_01 = S[1] - 0x30
```

```
00517BE8 8D45 F4 LEA EAX,DWORD PTR SS:[EBP-C]
```

```
00517BEB 50 PUSH EAX
```

```
00517BEC B9 01000000 MOV ECX,1 ; <== Get 1 chart
```

```
00517BF1 BA 09000000 MOV EDX,9 ; <== at [8]
```

```
00517BF6 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== of Serial
```

```
00517BF9 E8 1AD2EEFF CALL Prjpdfin.00404E18 ; <== Cutting chart
```

```
00517BFE 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C]
```

```
00517C01 E8 A617EFFF CALL Prjpdfin.004093AC ; <== This chart must be NUMBER
```

```
00517C06 8BF8 MOV EDI,EAX ; <== Ch_02 = S[8] - 0x30
```

```
00517C08 8D45 F0 LEA EAX,DWORD PTR SS:[EBP-10]
```

```
00517C0B 50 PUSH EAX
```

```
00517C0C B9 01000000 MOV ECX,1 ; <== Get 1 chart
```

```
00517C11 BA 0C000000 MOV EDX,0C ; <== at [11]
```

```
00517C16 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== of Serial
```

```
00517C19 E8 FAD1EEFF CALL Prjpdfin.00404E18 ; <== Cutting chart
```

```
00517C1E 8B45 F0 MOV EAX,DWORD PTR SS:[EBP-10]
```

```
00517C21 E8 8617EFFF CALL Prjpdfin.004093AC ; <== This chart must be NUMBER
```

```
00517C26 03FE ADD EDI,ESI ; <== Value = Ch_01 + Ch_02
```

```
00517C28 03C7 ADD EAX,EDI ; <== Value = Value + (Ch_03 - 0x30)
```

```
00517C2A 83F8 12 CMP EAX,12 ; <== Value == 0x12
```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : N/A
User : VHT-cRaCkErS	Serial : N/A

III – KeyGen :

N/A

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};
char reaDefaultString[64] =
"0147852369qwertyuioplkjhgfdaszxvcvbnmPLMKOIJNBHUYGVCFTRDXZSEWAQ";
int LenUser=0;
int i=0;

srand( (unsigned)time( NULL ) );
LenUser=GetDlgItemText(IDC_Name,reaName,64);
if (LenUser < 0)
{
    MessageBox("----- Your name atleast 1 chart ----- ","Hey !!
Please input your name again !! ");
}
else
{
    i=0;
    while ( i < 16 )
    {
        if ( i == 1 )
        {
            reaSerial[i] = 0x37 + rand() % 2;
        }
        else if ( i == 8 )
        {
            reaSerial[i] = 0x33 + rand() % 6;
        }
        else if ( i == 11 )
        {
            reaSerial[i] = (18 - (reaSerial[1] - 0x30) - (reaSerial[8] - 0x30)) + 0x30;
        }
        else
        {
            reaSerial[i] = reaDefaultString[rand() % 62];
        }
        i++;
    }
    SetDlgItemText(IDC_Serial,reaSerial);
}

```

V – End of Tut :

- Finished – **15/07/2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.blazingtools.com
Production :	BlazingTools Software.
SoftWare :	BlazingTools Perfect Keylogger 1.6.0.0
Copyright by :	Copyright © 2002-2004 BlazingTools Software . All Rights Reserved.
Type :	Email / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

BlazingTools Perfect Keylogger 1.6.0.0

Do you want to know what your buddy or colleague is typing? Or perhaps you just want to control your family members? With Perfect Keylogger you can do so in 2 minutes! Also it can be used for special purposes. This program runs on the installed computer, fully hidden from its users, logs in a protected file everything the user types and periodically makes screenshots.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng *Microsoft Visual C++ 6.0*.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Registration code or user name is invalid.**
Please check all fields and try again!" Ta tìm được đoạn CODE này ở hai địa chỉ :
00414BEE |. 68 88A04400 PUSH 123.0044A088 ; ASCII "Registration code or user name is invalid. Please check all fields and try again!"
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
00414A4F |. E8 3C1B0200 CALL <JMP.&MSVCRT._EH_prolog> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Đầu tiên chương trình kiểm tra xem đã nhập User chưa :
00414A6C |. E8 B7160200 CALL <JMP.&MFC42.#3098> ; <== Get Length User
00414A71 |. 80BD 4CFFFFFF>CMP BYTE PTR SS:[EBP-B4],0 ; <== User must be input
00414A78 |./5 53 JNZ SHORT 123.00414ACD
- Sau đó, chương trình sẽ nối 4 đoạn Serial lại thành một chuỗi Serial (16 ký tự) :
00414ACF |. 8D45 B4 LEA EAX,[LOCAL.19]
00414AD2 |. 5F POP EDI
00414AD3 |. 8BCE MOV ECX,ESI
00414AD5 |. 57 PUSH EDI
00414AD6 |. 50 PUSH EAX
00414AD7 |. 68 DAD60000 PUSH 0D6DA
00414ADC |. E8 47160200 CALL <JMP.&MFC42.#3098>
00414AE1 |. 8D45 D8 LEA EAX,[LOCAL.10]
00414AE4 |. 57 PUSH EDI
00414AE5 |. 50 PUSH EAX

```

00414AE6 |. 68 DBD60000 PUSH 0D6DB
00414AEB |. 8BCE      MOV ECX,ESI
00414AED |. E8 36160200 CALL <JMP.&MFC42.#3098>
00414AF2 |. 8D45 C0    LEA EAX,[LOCAL.16]
00414AF5 |. 57      PUSH EDI
00414AF6 |. 50      PUSH EAX
00414AF7 |. 68 DCD60000 PUSH 0D6DC
00414AFC |. 8BCE      MOV ECX,ESI
00414AFE |. E8 25160200 CALL <JMP.&MFC42.#3098>
00414B03 |. 8D45 CC    LEA EAX,[LOCAL.13]
00414B06 |. 57      PUSH EDI
00414B07 |. 50      PUSH EAX
00414B08 |. 68 DDD60000 PUSH 0D6DD
00414B0D |. 8BCE      MOV ECX,ESI
00414B0F |. E8 14160200 CALL <JMP.&MFC42.#3098>
00414B14 |. 8D45 B4    LEA EAX,[LOCAL.19]
00414B17 |. 50      PUSH EAX          ; /String2 =
00414B18 |. 8D45 80    LEA EAX,[LOCAL.32]   ; |
00414B1B |. 50      PUSH EAX          ; |String1
00414B1C |. FF15 38B14300 CALL DWORD PTR DS:[<&KERNEL32.lstrcpyA>] ; \lstrcpyA
00414B22 |. 8B3D 60B14300 MOV EDI,DWORD PTR DS:[<&KERNEL32.lstrcat>] ; 
kernel32.lstrcatA
00414B28 |. 8D45 D8    LEA EAX,[LOCAL.10]
00414B2B |. 50      PUSH EAX          ; /StringToAdd
00414B2C |. 8D45 80    LEA EAX,[LOCAL.32]   ; |
00414B2F |. 50      PUSH EAX          ; |ConcatString
00414B30 |. FFD7      CALL EDI          ; \lstrcatA
00414B32 |. 8D45 C0    LEA EAX,[LOCAL.16]
00414B35 |. 50      PUSH EAX          ; /StringToAdd
00414B36 |. 8D45 80    LEA EAX,[LOCAL.32]   ; |
00414B39 |. 50      PUSH EAX          ; |ConcatString
00414B3A |. FFD7      CALL EDI          ; \lstrcatA
00414B3C |. 8D45 CC    LEA EAX,[LOCAL.13]
00414B3F |. 50      PUSH EAX          ; /StringToAdd
00414B40 |. 8D45 80    LEA EAX,[LOCAL.32]   ; |
00414B43 |. 50      PUSH EAX          ; |ConcatString
00414B44 |. FFD7      CALL EDI          ; \lstrcatA
00414B46 |. 8D85 18FFFFFF LEA EAX,[LOCAL.58] ; <== Serial after ConCat
00414B4C |. 50      PUSH EAX
00414B4D |. 8D85 4CFFFFFF LEA EAX,[LOCAL.45] ; <== User
00414B53 |. 68 28EA4300 PUSH 123.0043EA28 ; ASCII "_r <&$<1-Z2[l5,^"
00414B58 |. 50      PUSH EAX          ; <== User
00414B59 |. E8 1FFEFFFF CALL 123.0041497D ; <== Trace Into
- Dùng F7 trace into lệnh CALL này ta đến quá trình mã hoá . Quá trình mã hoá này dựa trên chuỗi mặc định "_r <&$<1-Z2[l5,^" :
004149C2 |> \33F6    XOR ESI,ESI          ; <== i = 0
004149C4 |. 3975 0C    CMP [ARG.2],ESI
004149C7 |. 7E 2C    JLE SHORT 123.004149F5
004149C9 |> 8BC6    /MOV EAX,ESI          ; <== i
004149CB |. 6A 19    |PUSH 19          ; <== Default Value : dV
004149CD |. 99      |CDQ
004149CE |. F77D FC    |IDIV [LOCAL.1]     ; <== Temp = i / 0x10

```

004149D1 . 8BC6	MOV EAX,ESI	; <== i
004149D3 . 5B	POP EBX	; <== dV
004149D4 . 8D0C3A	LEA ECX,DWORD PTR DS:[EDX+EDI]	; <== Default String : dStr
004149D7 . 99	CDQ	
004149D8 . F77D F8	IDIV [LOCAL.2]	; <== Temp = i % Len.U
004149DB . 8B45 08	MOV EAX,[ARG.1]	; <== User
004149DE . 0FB60402	MOVZX EAX,BYTE PTR DS:[EDX+EAX]	; <== U[Temp]

===== NOTE =====

Quá trình này thực ra là dự phòng cho trường hợp chuỗi U nhập có chiều dài nhỏ hơn chiều dài của chuỗi Mặc định. Lúc đó, chuỗi U đã hết nhưng vòng lặp chưa hết. Vì vậy với phép chia này **Temp = i % Len.U** thì khi kết thúc chuỗi U thì tự động chương trình quay ngược trở lại đầu chuỗi U.

===== NOTE =====

004149E2 . 0FB611	MOVZX EDX,BYTE PTR DS:[ECX]	; <== dStr[i]
004149E5 . 33C2	XOR EAX,EDX	; <== Value = sStr[i] xor U[Temp]
004149E7 . 99	CDQ	
004149E8 . F7FB	IDIV EBX	; <== Value = Value % dV
004149EA . 80C2 41	ADD DL,41	; <== Value = Value + 0x41
004149ED . 46	INC ESI	; <== i++
004149EE . 3B75 0C	CMP ESI,[ARG.2]	; <== while (i < 0x10)
004149F1 . 8811	MOV BYTE PTR DS:[ECX],DL	; <== S[i] = Value
004149F3 .^ 7C D4	JL SHORT 123.004149C9	; <== Continue Loop

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErStEaM

Serial : NFWR-TSSO-UGOI-YMCT

III – KeyGen :

- /Section 1/- Tính *Value = reaDefaultString[i] ^ reaName[i % LenUser]*.
- /Section 2/- Tính *reaSerial[j] = (Value % 0x19) + 0x41*

IV – End of Tut :

- Finished – **21/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.dvdtompegx.com
Production :	SevenWolfs Software,Inc.
SoftWare :	Plato DVD Ripper 1.13
Copyright by :	Copyright © 2003-2005 SevenWolfs Software,Inc. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Plato DVD Ripper 1.13

Plato DVD Ripper is a powerful and easy to use tool for backing up your DVD movies. with Plato DVD Ripper you are able to convert DVDs to VCD(MPEG1),SVCD(MPEG2),AVI, and Divx(MPEG4) formats.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử Fake Serial, ta nhận được thông báo "**Registration Code incorrect!**" Ta tìm được đoạn CODE này ở địa chỉ :

```
00409974 . 68 84BC9300 PUSH PlatoRip.0093BC84 ; |Arg1 = 0093BC84 ASCII "Registration Code incorrect!"
```

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :

```
004098E2 . E8 F0070B00 CALL PlatoRip.004BA0D7 ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace một đoạn ta đến :

```
00409943 . E8 24FBFFFF CALL PlatoRip.0040946C ; \PlatoRip.0040946C
00409948 . 85C0 TEST EAX,EAX ; <== Serial correct or not
0040994A . 74 1A JE SHORT PlatoRip.00409966 ; <== if Correct : Congrat!!!!
0040994C . 33C0 XOR EAX,EAX
0040994E . 50 PUSH EAX ; /Arg3 => 00000000
0040994F . 50 PUSH EAX ; |Arg2 => 00000000
00409950 . 68 68BC9300 PUSH PlatoRip.0093BC68 ; |Arg1 = 0093BC68 ASCII "Thank
you for registering!"
```

- Dùng F7 trace into lệnh CALL ta đến đoạn CODE cắt chuỗi Serial thành 4 đoạn :

```
004094AA |. 6A 04 PUSH 4 ; <== Get 4 charts
004094AC |. 50 PUSH EAX ; <== from first chart of Serial
004094AD |. 52 PUSH EDX ; <== Saved at add.
004094AE |. E8 6DCB0900 CALL PlatoRip.004A6020 ; <== Cutting
004094B3 |. 83C4 0C ADD ESP,0C
..... cutting .....
004094C7 |. 6A 05 PUSH 5 ; <== Get 5 charts
004094C9 |. 83C0 05 ADD EAX,5 ; <== from 6th chart of Serial
004094CC |. 50 PUSH EAX
004094CD |. 52 PUSH EDX ; <== Saved at add.
004094CE |. E8 4DCB0900 CALL PlatoRip.004A6020 ; <== Cutting
..... cutting .....
004094E7 |. 6A 04 PUSH 4 ; <== Get 4 charts
004094E9 |. 83C0 0B ADD EAX,0B ; <== from 12th chart of Serial
004094EC |. 50 PUSH EAX
004094ED |. 52 PUSH EDX ; <== Saved at add.
004094EE |. E8 2DCB0900 CALL PlatoRip.004A6020 ; <== Cutting
..... cutting .....
00409507 |. 6A 04 PUSH 4 ; <== Get 4 charts
00409509 |. 83C0 10 ADD EAX,10 ; <== from 17th chart of Serial
0040950C |. 50 PUSH EAX
0040950D |. 52 PUSH EDX ; <== Saved at add.
0040950E |. E8 0DCB0900 CALL PlatoRip.004A6020 ; <== Cutting
```

- Qua quá trình cắt chuỗi này ta xác định được chiều dài của chuỗi Serial là 20 ký tự . Từng đoạn này sẽ được tiến hành mã hoá cùng một cách thức :

```

00409529 |. 50      PUSH EAX          ; <== Section_04
0040952A |. 52      PUSH EDX          ; <== Section_03
0040952B |. 53      PUSH EBX          ; <== Section_02
0040952C |. 56      PUSH ESI          ; <== Section_01
0040952D |. E8 2A000000 CALL PlatoRip.0040955C ; <== Trace into
----- Trace Into -----
00409563 |. 8B7C24 1C  MOV EDI,DWORD PTR SS:[ESP+1C] ; <== Section_01 : S-01
00409567 |. 33C0      XOR EAX,EAX
00409569 |. 8A37      MOV DH,BYTE PTR DS:[EDI]       ; <== S_01[0]
0040956B |. 8BCF      MOV ECX,EDI
0040956D |. 84F6      TEST DH,DH
0040956F |. 74 0C      JE SHORT PlatoRip.0040957D
00409571 |> 83C1 01   /ADD ECX,1        ; <== / from here
00409574 |. 83C0 01   |ADD EAX,1        ; <== |
00409577 |. 8A11      |MOV DL,BYTE PTR DS:[ECX]    ; <== | Get length of Section
00409579 |. 84D2      |TEST DL,DL       ; <== |
0040957B |.^ 75 F4    |JNZ SHORT PlatoRip.00409571 ; <== \ to here
0040957D |> 33DB      XOR EBX,EBX      ; <== Value = 0
0040957F |. 33D2      XOR EDX,EDX      ; <== i = 0
00409581 |. 85C0      TEST EAX,EAX
00409583 |. 7E 28      JLE SHORT PlatoRip.004095AD
00409585 |> 0FBEC3A    /MOVSX EBP,BYTE PTR DS:[EDX+EDI] ; <== S_01[i]
00409589 |. 83C2 01   |ADD EDX,1        ; <== i++
0040958C |. 83FD 39   |CMP EBP,39       ; <== if ( S_01 > 0x39 )
0040958F |. 7E 0D      |JLE SHORT PlatoRip.0040959E ; <== then
00409591 |. 8D0C1B    |LEA ECX,DWORD PTR DS:[EBX+EBX] ; <== Temp = Value * 0x2
00409594 |. 03C9      |ADD ECX,ECX      ; <== Temp = Temp * 0x2
00409596 |. 03CB      |ADD ECX,EBX      ; <== Temp = Temp + Value
00409598 |. 8D5C4D B0  |LEA EBX,DWORD PTR SS:[EBP+ECX*2-50] ; <== Value = S_01[i]
+ Temp * 0x2 - 0x50
0040959C |. EB 0B      |JMP SHORT PlatoRip.004095A9 ; <== else
0040959E |> 8D0C1B    |LEA ECX,DWORD PTR DS:[EBX+EBX] ; <== Temp = Value * 0x2
004095A1 |. 03C9      |ADD ECX,ECX      ; <== Temp = Temp * 0x2
004095A3 |. 03CB      |ADD ECX,EBX      ; <== Temp = Temp + Value
004095A5 |. 8D5C4D D0  |LEA EBX,DWORD PTR SS:[EBP+ECX*2-30] ; <== Value = S_01[i]
+ Temp * 0x2 - 0x30
004095A9 |> 3BD0      |CMP EDX,EAX      ; <== while ( i < 4 )
004095AB |.^ 7C D8    |JL SHORT PlatoRip.00409585 ; <== Continue Loop
004095AD |> 8B7C24 20  MOV EDI,DWORD PTR SS:[ESP+20] ; <== Section_02 : S-02
004095B1 |. 33C0      XOR EAX,EAX
004095B3 |. 8A37      MOV DH,BYTE PTR DS:[EDI]       ; <== S_02[0]
004095B5 |. 8BCF      MOV ECX,EDI
004095B7 |. 84F6      TEST DH,DH
004095B9 |. 74 0C      JE SHORT PlatoRip.004095C7
004095BB |> 83C1 01   /ADD ECX,1        ; <== / from here
004095BE |. 83C0 01   |ADD EAX,1        ; <== |
004095C1 |. 8A11      |MOV DL,BYTE PTR DS:[ECX]    ; <== | Get length of Section
004095C3 |. 84D2      |TEST DL,DL       ; <== |
004095C5 |.^ 75 F4    |JNZ SHORT PlatoRip.004095BB ; <== \ to here
004095C7 |> 33ED      XOR EBP,EBP      ; <== Value = 0
004095C9 |. 33D2      XOR EDX,EDX      ; <== i = 0
004095CB |. 85C0      TEST EAX,EAX

```

```

004095CD |. 7E 2E      JLE SHORT PlatoRip.004095FD
004095CF |. 8B7424 20  MOV ESI,DWORD PTR SS:[ESP+20]      ; <== Section_02 : S-02
004095D3 |> 0FBE0C32  /MOVSX ECX,BYTE PTR DS:[EDX+ESI] ; <== S_02[i]
004095D7 |. 83C2 01    |ADD EDX,1                      ; <== i++
004095DA |. 83F9 39    |CMP ECX,39                  ; <== if ( S_02 > 0x39 )
004095DD |. 7E 0E      |JLE SHORT PlatoRip.004095ED      ; <== then
004095DF |. 8D7C2D 00  |LEA EDI,DWORD PTR SS:[EBP+EBP]    ; <== Temp = Value * 0x2
004095E3 |. 03FF      |ADD EDI,EDI                  ; <== Temp = Temp * 0x2
004095E5 |. 03FD      |ADD EDI,EBP                  ; <== Temp = Temp + Value
004095E7 |. 8D6C79 B0  |LEA EBP,DWORD PTR DS:[ECX+EDI*2-50]   ; <== Value = S_02[i]
+ Temp * 0x2 - 0x50
004095EB |. EB 0C      |JMP SHORT PlatoRip.004095F9      ; <== else
004095ED |> 8D7C2D 00  |LEA EDI,DWORD PTR SS:[EBP+EBP]    ; <== Temp = Value * 0x2
004095F1 |. 03FF      |ADD EDI,EDI                  ; <== Temp = Temp * 0x2
004095F3 |. 03FD      |ADD EDI,EBP                  ; <== Temp = Temp * 0x2
004095F5 |. 8D6C79 D0  |LEA EBP,DWORD PTR DS:[ECX+EDI*2-30]   ; <== Value = S_02[i]
+ Temp * 0x2 - 0x30
004095F9 |> 3BD0      |CMP EDX,EAX                 ; <== while ( i < 5 )
004095FB |.^ 7C D6    |JL SHORT PlatoRip.004095D3      ; <== Continue Loop
004095FD |> 8B7C24 24  MOV EDI,DWORD PTR SS:[ESP+24]    ; <== Section_03 : S-03
00409601 |. 33C0      XOR EAX,EAX
00409603 |. 8A37      MOV DH,BYTE PTR DS:[EDI]        ; <== S_03[0]
00409605 |. 8BCF      MOV ECX,EDI
00409607 |. 84F6      TEST DH,DH
00409609 |. 74 0C      JE SHORT PlatoRip.00409617      ; <== / from here
0040960B |> 83C1 01  /ADD ECX,1
0040960E |. 83C0 01  |ADD EAX,1
00409611 |. 8A11      |MOV DL,BYTE PTR DS:[ECX]
00409613 |. 84D2      |TEST DL,DL
00409615 |.^ 75 F4    |JNZ SHORT PlatoRip.0040960B      ; <== \ to here
00409617 |> 33F6      XOR ESI,ESI
00409619 |. 33FF      XOR EDI,EDI
0040961B |. 85C0      TEST EAX,EAX
0040961D |. 7E 32      JLE SHORT PlatoRip.00409651
0040961F |. 8B4C24 24  MOV ECX,DWORD PTR SS:[ESP+24]    ; <== Section_03 : S-03
00409623 |. 891C24    MOV DWORD PTR SS:[ESP],EBX
00409626 |> 0FBE140F  /MOVSX EDX,BYTE PTR DS:[EDI+ECX] ; <== S_03[i]
0040962A |. 83C7 01  |ADD EDI,1
0040962D |. 83FA 39  |CMP EDX,39
00409630 |. 7E 0D      |JLE SHORT PlatoRip.0040963F      ; <== then
00409632 |. 8D1C36    |LEA EBX,DWORD PTR DS:[ESI+ESI]    ; <== Temp = Value * 0x2
00409635 |. 03DB      |ADD EBX,EBX
00409637 |. 03DE      |ADD EBX,ESI
00409639 |. 8D745A B0  |LEA ESI,DWORD PTR DS:[EDX+EBX*2-50]   ; <== Value = S_03[i]
+ Temp * 0x2 - 0x50
0040963D |. EB 0B      |JMP SHORT PlatoRip.0040964A      ; <== else
0040963F |> 8D1C36    |LEA EBX,DWORD PTR DS:[ESI+ESI]    ; <== Temp = Value * 0x2
00409642 |. 03DB      |ADD EBX,EBX
00409644 |. 03DE      |ADD EBX,ESI
00409646 |. 8D745A D0  |LEA ESI,DWORD PTR DS:[EDX+EBX*2-30]   ; <== Value = S_03[i]
+ Temp * 0x2 - 0x30
0040964A |> 3BF8      |CMP EDI,EAX

```

```

0040964C |.^ 7C D8    \JL SHORT PlatoRip.00409626      ; <== Continue Loop
0040964E |. 8B1C24    MOV EBX,DWORD PTR SS:[ESP]
00409651 |> 8B7C24 28  MOV EDI,DWORD PTR SS:[ESP+28]   ; <== Section_04 : S-04
00409655 |. 33C0      XOR EAX,EAX
00409657 |. 8A37      MOV DH,BYTE PTR DS:[EDI]        ; <== S_04[0]
00409659 |. 8BCF      MOV ECX,EDI
0040965B |. 84F6      TEST DH,DH
0040965D |. 74 0C      JE SHORT PlatoRip.0040966B      ; <== / from here
0040965F |> 83C1 01   /ADD ECX,1
00409662 |. 83C0 01   |ADD EAX,1
00409665 |. 8A11      |MOV DL,BYTE PTR DS:[ECX]
00409667 |. 84D2      |TEST DL,DL
00409669 |.^ 75 F4    |JNZ SHORT PlatoRip.0040965F      ; <== \ to here
0040966B |> 33C9      XOR ECX,ECX
0040966D |. 33D2      XOR EDX,EDX
0040966F |. 85C0      TEST EAX,EAX
00409671 |. 7E 3A      JLE SHORT PlatoRip.004096AD      ; <== Value = 0
00409673 |. 896C24 04  MOV DWORD PTR SS:[ESP+4],EBP
00409677 |. 8B6C24 28  MOV EBP,DWORD PTR SS:[ESP+28]   ; <== Section_04 : S-04
0040967B |. 891C24    MOV DWORD PTR SS:[ESP],EBX
0040967E |> 0FB E1C2A /MOVSX EBX,BYTE PTR DS:[EDX+EBP] ; <== S_04[i]
00409682 |. 83C2 01   |ADD EDX,1
00409685 |. 83FB 39   |CMP EBX,39
00409688 |. 7E 0D      |JLE SHORT PlatoRip.00409697      ; <== then
0040968A |. 8D3C09    |LEA EDI,DWORD PTR DS:[ECX+ECX]
0040968D |. 03FF      |ADD EDI,EDI
0040968F |. 03F9      |ADD EDI,ECX
00409691 |. 8D4C7B B0  |LEA ECX,DWORD PTR DS:[EBX+EDI*2-50] ; <== i = 0
+ Temp * 0x2 - 0x50
00409695 |. EB 0B      |JMP SHORT PlatoRip.004096A2      ; <== else
00409697 |> 8D3C09    |LEA EDI,DWORD PTR DS:[ECX+ECX]
0040969A |. 03FF      |ADD EDI,EDI
0040969C |. 03F9      |ADD EDI,ECX
0040969E |. 8D4C7B D0  |LEA ECX,DWORD PTR DS:[EBX+EDI*2-30] ; <== Value = S_04[i]
+ Temp * 0x2 - 0x30
004096A2 |> 3BD0      |CMP EDX,EAX
004096A4 |.^ 7C D8    \JL SHORT PlatoRip.0040967E      ; <== Continue Loop
004096A6 |. 8B6C24 04  MOV EBP,DWORD PTR SS:[ESP+4]
004096AA |. 8B1C24    MOV EBX,DWORD PTR SS:[ESP]
004096AD |> 3BF3      CMP ESI,EBX
004096AF |. 7E 27      JLE SHORT PlatoRip.004096D8      ; <== while ( i < 4 )
004096B1 |. 3BF1      CMP ESI,ECX
004096B3 |. 7E 23      JLE SHORT PlatoRip.004096D8      ; <== continue check
004096B5 |. 03DE      ADD EBX,ESI
004096B7 |. 03D9      ADD EBX,ECX
004096B9 |. 3BDD      CMP EBX,EBP
004096BB |. 75 0F      JNZ SHORT PlatoRip.004096CC      ; <== if (Value == vS_02 )
                                                               ; <== Congrat !!!
===== Trace Into =====

```

- Nếu nhìn nhận từ bên ngoài, quá trình mã hoá bên trên thật không có gì phức tạp, nhưng để tạo Serial thì không phải dễ . Tuy nhiên, nếu thật sự hiểu được ý nghĩa của đoạn CODE muốn gì thì quá trình mã hoá lại rất đơn giản .

- Ý nghĩa thực sự của quá trình này là chuyển đổi chuỗi Serial sang giá trị HEX tương ứng . Ví dụ, nếu ta có chuỗi là “1234” thì kết quả của quá trình ta có được giá trị “04D2” (04D2h = 1234).

- Với ý nghĩa thực sự của quá trình thì việc tạo KeyGen không có gì phức tạp cả .

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : 3575-11916-7150-1191 or 1773-09754-7095-0886

III – KeyGen :

- /Section 1/- Tạo giá trị ngẫu nhiên của đoạn thứ III .
- /Section 2/- Đoạn thứ I và thứ IV được tính từ đoạn III sao cho giá trị của chúng nhỏ hơn đoạn III
- /Section 3/- Giá trị của đoạn thứ II là tổng của ba đoạn kể trên.
- /Section 3/- Xuất chuỗi ra màn hình theo định dạng "%04i-%05i-%04i-%04i"

IV – End of Tut :

- Finished – **20/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.dvdtopmp3.com
Production :	SevenWolfs Software,Inc.
SoftWare :	Plato DVD to MP3 Ripper
Copyright by :	Copyright © 2003-2010 SevenWolfs Software,Inc. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Plato DVD to MP3 Ripper

@Plato DVD to MP3 Ripper is an Easy Use Software that let you Extract DVD Audio to MP3 files

Key Features:

Convert DVD Audio to MP3 . Specify the Start Position and the End Postion of DVD Audio Track and Rip it to MP3 file . Selectable Program Chain . Selectable Audio Track . Supports Dolby Surround . Could Rip any region DVD disc without changing DVD drive region setting.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.

- Nhập thử Fake Serial, ta nhận được thông báo "**serial number error!**" Ta tìm được đoạn CODE này ở địa chỉ :

00407EE9 .68 E47B7400 PUSH dvdtomp3.00747BE4 ; |Arg1 = 00747BE4 ASCII "serial number error!"

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
 00407E25 . E8 45CC0700 CALL dvdtomp3.00484A6F ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace một đoạn ta đến :

```

00407EA3 . E8 B8E6FFFF CALL dvdtomp3.00406560 ; <== Encrypt & Compare
00407EA8 . 85C0 TEST EAX,EAX ; <== Serial correct or not
00407EAA . 74 30 JE SHORT dvdtomp3.00407EDC ; <== if Correct : Congrat!!!!
00407EAC . 8B0D 083C4A00 MOV ECX,DWORD PTR DS:[4A3C08]
00407EB2 . 33C0 XOR EAX,EAX
00407EB4 . 50 PUSH EAX ; /Arg3 => 00000000
00407EB5 . 50 PUSH EAX ; |Arg2 => 00000000
00407EB6 . 68 C87B7400 PUSH dvdtomp3.00747BC8; |Arg1 = 00747BC8 ASCII "Thank you for
registering!"
00407EBB . E8 B8DDFFFF CALL dvdtomp3.00405C78 ; \dvdtomp3.00405C78
  
```

- Chiều dài của chuỗi Serial theo quy định của chương trình :

```

00406573 |./74 0C JE SHORT dvdtomp3.00406581
00406575 |>|83C1 01 /ADD ECX,1
00406578 |.|83C0 01 |ADD EAX,1
0040657B |.|8A11 |MOV DL,BYTE PTR DS:[ECX]
0040657D |.|84D2 |TEST DL,DL
0040657F |.^|75 F4 |JNZ SHORT dvdtomp3.00406575
00406581 |>|83F8 14 CMP EAX,14 ; <== LenS must be 20 charts
00406584 |.|73 0C JNB SHORT dvdtomp3.00406592
  
```

- Trace tiếp ta đến quá trình cắt chuỗi Serial thành 4 đoạn . Trong đó, ba đoạn SecI, SecII, SecIV có chiều dài là 4 ký tự, SecIII có chiều dài là 5 ký tự . Các đoạn được phân cách nhau bằng dấu “-” :

```

00406596 |.|6A 04 PUSH 4 ; <== Get 4 charts
00406598 |.|57 PUSH EDI ; <== from Fake Serial
00406599 |.|50 PUSH EAX ; <== Saved Add.
0040659A |.|E8 D1780600 CALL dvdtomp3.0046DE70 ; <== Ripping
0040659F |.|C64424 18 00 MOV BYTE PTR SS:[ESP+18],0
004065A4 |.|8D5424 1C LEA EDX,DWORD PTR SS:[ESP+1C]
004065A8 |.|8D47 05 LEA EAX,DWORD PTR DS:[EDI+5] ; <== Next Section
004065AB |.|6A 04 PUSH 4 ; <== Get 4 charts
004065AD |.|50 PUSH EAX ; <== from Fake Serial
004065AE |.|52 PUSH EDX ; <== Saved Add.
004065AF |.|E8 BC780600 CALL dvdtomp3.0046DE70 ; <== Ripping
004065B4 |.|C64424 2C 00 MOV BYTE PTR SS:[ESP+2C],0
004065B9 |.|8D5424 30 LEA EDX,DWORD PTR SS:[ESP+30]
004065BD |.|8D47 0A LEA EAX,DWORD PTR DS:[EDI+A] ; <== Next Section
004065C0 |.|6A 05 PUSH 5 ; <== Get 5 charts
004065C2 |.|50 PUSH EAX ; <== from Fake Serial
004065C3 |.|52 PUSH EDX ; <== Saved Add.
004065C4 |.|E8 A7780600 CALL dvdtomp3.0046DE70 ; <== Ripping
004065C9 |.|C64424 41 00 MOV BYTE PTR SS:[ESP+41],0
004065CE |.|8D4424 44 LEA EAX,DWORD PTR SS:[ESP+44]
004065D2 |.|8D7F 10 LEA EDI,DWORD PTR DS:[EDI+10] ; <== Next Section
004065D5 |.|6A 04 PUSH 4 ; <== Get 4 charts
  
```

```

004065D7 |. 57      PUSH EDI          ; <== from Fake Serial
004065D8 |. 50      PUSH EAX          ; <== Saved Add.
004065D9 |. E8 92780600 CALL dvdtomp3.0046DE70 ; <== Ripping

```

- Tiếp đó, chương trình sẽ chuyển đổi từng SECTION này sang dạng giá trị HEX tương ứng (*SecI : “1234”* sẽ được chuyển thành giá trị *hValueI = 0x4D2*)

```

0040669F |. E8 39720600 CALL dvdtomp3.0046D8DD ; <== Convert SecI to hValueI
.... Ripping ....
00406762 |. E8 76710600 CALL dvdtomp3.0046D8DD ; <== Convert SecII to hValueII
.... Ripping ....
0040682B |. E8 AD700600 CALL dvdtomp3.0046D8DD ; <== Convert SecIII to hValueIII
.... Ripping ....
004068F4 |. E8 E46F0600 CALL dvdtomp3.0046D8DD ; <== Convert SecIV to hValueIV

```

- Sau đó tiến hành quá trình tính toán đơn giản :

```

004068F9  8B15 E4B24900 MOV EDX,DWORD PTR DS:[49B2E4] ; <== dV ( 0x212 )
004068FF |. 83C4 04      ADD ESP,4
00406902 |. 2BDA      SUB EBX,EDX ; <== Value = hValueI - dV
00406904 |. 03DD      ADD EBX,EBP ; <== Value = Value + hValueII
00406906 |. 03D8      ADD EBX,EAX ; <== Value = Value + hValueIV
00406908 |. 3BF3      CMP ESI,EBX ; <== if (Value == hValueIII)
0040690A |.^ 0F85 76FCFFFF JNZ dvdtomp3.00406586 ; <== Congrat !!!!

/*/*/* - SERIAL tương ứng với USER :
User : REA-cRaCkErTeAm   Serial : 9741-7529-17518-0778   or   7624-0537-09179-1548
```

III – KeyGen :

- /Section 1/- Tạo giá trị ngẫu nhiên của 3 đoạn .
- /Section 2/- tính giá trị đoạn thứ III theo công thức :
 $hValueIII = hValueI + hValueII + hValueIV - 0x212$
- /Section 3/- Xuất chuỗi ra màn hình theo định dạng "%04i-%04i-%05i-%04i"

IV – End of Tut :

- Finished – *September 9, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.galcott.com
Production :	Glenn Alcott.
SoftWare :	Power Edit 2.03
Copyright by :	Copyright © Glenn Alcott. All Rights Reserved.
Type :	Serial
Packed :	N / A
Language :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial

Power Edit 2.03

Power Edit is a very powerful text editor for Windows 95, 98, ME, NT, 2000 and XP. It is a far more functional replacement for Notepad and Wordpad, and an excellent tool for creating and editing Web pages.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo. Sau khi unPACK và kiểm tra lại biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**.

- Nhập thử Name và Fake Serial, ta nhận được thông báo "**Incorrect registration code**" Ta chỉ có thể tìm được chuỗi này bằng cách dùng kỹ thuật STACK :

===== STACK =====

Nhập chuỗi Serial vào, nhấn OK, xuất hiện thông báo "Incorrect registration code". Giữ nguyên và quay trở lại Olly. Nhấn F12 để dừng chương trình, nhấn Alt-K để mở cửa sổ Call stack of main thread :

<i>Address</i>	<i>Stack</i>	<i>Procedure / arguments</i>	<i>Called from</i>	<i>Frame</i>
0012F170	77D6649A	? USER32.MessageBoxExA	USER32.77D66495	0012F16C
0012F174	000703FA	hOwner = 000703FA ('Power Edit',class='TApplication')		
0012F178	005B9C78	Text = "Incorrect registration code"		
0012F17C	0012F255	Title = "Error"		
0012F180	00000030	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL		
0012F184	00000000	LanguageID = 0 (LANG_NEUTRAL)		
0012F188	0045949F	? <JMP.&USER32.MessageBoxA>		poweredi.0045949A
0012F18C	000703FA	hOwner = 000703FA ('Power Edit',class='TApplication')		
0012F190	005B9C78	Text = "Incorrect registration code"		
0012F194	0012F255	Title = "Error"		
0012F198	00000030	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL		
0012F208	0063A92E	? poweredi.004593BC	poweredi.0063A929	

Double-Click vào dòng "**poweredi.0045949A**". Ta sẽ quay trở lại cửa sổ chính của Olly .

0045949A E8 C9F6FAFF CALL <JMP.&USER32.MessageBoxA> ; <== Here
 0045949F 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX ; <== BP here

Trace qua 4 lần RETN, nhìn lên trên chút ta thấy dòng :

005B9B49 BA 689C5B00 MOV EDX,poweredi.005B9C68 ; ASCII "Error"
 005B9B4E B8 789C5B00 MOV EAX,poweredi.005B9C78 ; ASCII "Incorrect registration code"

===== STACK =====

- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :

005B9A5D E8 5608E8FF CALL poweredi.0043A2B8 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Chương trình này rất đơn giản. Sau khi dừng lại tại BreakPoint ta trace xuống chút và đến :

005B9A7E	8B55 F4	MOV EDX,DWORD PTR SS:[EBP-C]	; <== Real Serial
005B9A81	58	POP EAX	; <== Fake Serial
005B9A82	E8 A9A8E4FF	CALL poweredi.00404330	; <== Compare

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 6508329

III – KeyGen :

/Section /- N/A

IV – End of Tut :- Finished - *August 09, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.etrusoft.com
Production :	EtruSoft Inc.
SoftWare :	Quick Screen Capture 2.2.0
Copyright by :	Copyright © 2002-2004 EtruSoft Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Quick Screen Capture 2.2.0

Quick Screenshot Maker is an all-in-one tool for screen capturing, image editing and organization. It can capture any part of the screen precisely in flexible ways, including DirectX, screen saver and movie screens. Use Quick Screenshot Maker to save time and enhance your screen shots. Add annotations, text, arrows, highlights, clipart and more. The History Bar helps you find all of your captured images and organize them quickly and easily.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Sorry, your registration key is wrong. Please check it and try again**". Ta tìm được thông báo này tạo địa chỉ :
00422675 . 68 FCA65000 PUSH Capture.0050A6FC ; ASCII "Sorry, your registration key is wrong. Please check it and try again."
- Dò ngược lên trên, ta chọn đặt BreakPoint tại :
004225C0 . E8 918B0800 CALL Capture.004AB156 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP.
- Trace xuống chút ta đến :
00422665 . E8 C60FFECC CALL Capture.00403630 ; \Capture.00403630
0042266A . 84C0 TEST AL,AL ; <== if Correct AL == 0x1

```

0042266C . 75 37    JNZ SHORT Capture.004226A5 ; <== Congrat !!!
0042266E . 6A 40    PUSH 40
00422670 . 68 28985000 PUSH Capture.00509828 ; ASCII "!Quick Screen Capture"
00422675 . 68 FCA65000 PUSH Capture.0050A6FC ; ASCII "Sorry, your registration key
is wrong. Please check it and try again."
0042267A . 8BCD      MOV ECX,EBP
0042267C . E8 597E0800 CALL Capture.004AA4DA

```

- Như vậy ta xác định được đoạn CODE chính là ở lệnh ***CALL Capture.00403630***. Ta sẽ trace into lệnh này để xác định rõ quá trình mã hoá. Đầu tiên, chương trình CẮT và GHÉP các ký tự của chuỗi Serial ở các vị trí xác định thành hai chuỗi riêng biệt :

```

0040365F |. 83F8 20    CMP EAX,20 ; <== LenS must be 32 charts
00403662 |. 0F85 20050000 JNZ Capture.00403B88
00403668 |. A1 ECBD5000 MOV EAX,DWORD PTR DS:[50BDEC]
0040366D |. 894424 14    MOV DWORD PTR SS:[ESP+14],EAX
00403671 |. 894424 10    MOV DWORD PTR SS:[ESP+10],EAX
00403675 |. 6A 01      PUSH 1
00403677 |. 8D4C24 60    LEA ECX,DWORD PTR SS:[ESP+60]
0040367B |. 6A 09      PUSH 9
0040367D |. 51      PUSH ECX
0040367E |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
00403685 |. C64424 78 03  MOV BYTE PTR SS:[ESP+78],3
0040368A |. E8 780C0A00 CALL Capture.004A4307
0040368F |. 8BF0      MOV ESI,EAX
00403691 |. 6A 01      PUSH 1
00403693 |. 8D5424 5C    LEA EDX,DWORD PTR SS:[ESP+5C]
00403697 |. 6A 1F      PUSH 1F
00403699 |. 52      PUSH EDX
0040369A |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
004036A1 |. C64424 78 04  MOV BYTE PTR SS:[ESP+78],4
004036A6 |. E8 5C0C0A00 CALL Capture.004A4307
004036AB |. 8BF8      MOV EDI,EAX
004036AD |. 6A 01      PUSH 1
004036AF |. 8D4424 58    LEA EAX,DWORD PTR SS:[ESP+58]
004036B3 |. 6A 06      PUSH 6
004036B5 |. 50      PUSH EAX
004036B6 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
004036BD |. C64424 78 05  MOV BYTE PTR SS:[ESP+78],5
004036C2 |. E8 400C0A00 CALL Capture.004A4307
004036C7 |. 8BE8      MOV EBP,EAX
004036C9 |. 6A 01      PUSH 1
004036CB |. 8D4C24 54    LEA ECX,DWORD PTR SS:[ESP+54]
004036CF |. 6A 17      PUSH 17
004036D1 |. 51      PUSH ECX
004036D2 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
004036D9 |. C64424 78 06  MOV BYTE PTR SS:[ESP+78],6
004036DE |. E8 240C0A00 CALL Capture.004A4307
004036E3 |. 8BD8      MOV EBX,EAX
004036E5 |. 6A 01      PUSH 1
004036E7 |. 8D5424 50    LEA EDX,DWORD PTR SS:[ESP+50]
004036EB |. 6A 0B      PUSH 0B
004036ED |. 52      PUSH EDX

```

004036EE |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 004036F5 |. C64424 78 07 MOV BYTE PTR SS:[ESP+78],7
 004036FA |. E8 080C0A00 CALL Capture.004A4307
 004036FF |. 894424 20 MOV DWORD PTR SS:[ESP+20],EAX
 00403703 |. 6A 01 PUSH 1
 00403705 |. 8D4424 4C LEA EAX,DWORD PTR SS:[ESP+4C]
 00403709 |. 6A 07 PUSH 7
 0040370B |. 50 PUSH EAX
 0040370C |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403713 |. C64424 78 08 MOV BYTE PTR SS:[ESP+78],8
 00403718 |. E8 EA0B0A00 CALL Capture.004A4307
 0040371D |. 894424 1C MOV DWORD PTR SS:[ESP+1C],EAX
 00403721 |. 6A 01 PUSH 1
 00403723 |. 8D4C24 48 LEA ECX,DWORD PTR SS:[ESP+48]
 00403727 |. 6A 05 PUSH 5
 00403729 |. 51 PUSH ECX
 0040372A |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403731 |. C64424 78 09 MOV BYTE PTR SS:[ESP+78],9
 00403736 |. E8 CC0B0A00 CALL Capture.004A4307
 0040373B |. 894424 18 MOV DWORD PTR SS:[ESP+18],EAX
 0040373F |. 6A 01 PUSH 1
 00403741 |. 8D5424 44 LEA EDX,DWORD PTR SS:[ESP+44]
 00403745 |. 6A 00 PUSH 0
 00403747 |. 52 PUSH EDX
 00403748 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 0040374F |. C64424 78 0A MOV BYTE PTR SS:[ESP+78],0A
 00403754 |. E8 AE0B0A00 CALL Capture.004A4307
 00403759 |. 8B4C24 18 MOV ECX,DWORD PTR SS:[ESP+18]
 0040375D |. C64424 6C 0B MOV BYTE PTR SS:[ESP+6C],0B
 00403762 |. 51 PUSH ECX
 00403763 |. 50 PUSH EAX
 00403764 |. 8D5424 44 LEA EDX,DWORD PTR SS:[ESP+44]
 00403768 |. 52 PUSH EDX
 00403769 |. E8 47920A00 CALL Capture.004AC9B5
 0040376E |. 8B4C24 1C MOV ECX,DWORD PTR SS:[ESP+1C]
 00403772 |. 8D5424 38 LEA EDX,DWORD PTR SS:[ESP+38]
 00403776 |. 51 PUSH ECX
 00403777 |. 50 PUSH EAX
 00403778 |. 52 PUSH EDX
 00403779 |. C64424 78 0C MOV BYTE PTR SS:[ESP+78],0C
 0040377E |. E8 32920A00 CALL Capture.004AC9B5
 00403783 |. 8B4C24 20 MOV ECX,DWORD PTR SS:[ESP+20]
 00403787 |. 8D5424 34 LEA EDX,DWORD PTR SS:[ESP+34]
 0040378B |. 51 PUSH ECX
 0040378C |. 50 PUSH EAX
 0040378D |. 52 PUSH EDX
 0040378E |. C64424 78 0D MOV BYTE PTR SS:[ESP+78],0D
 00403793 |. E8 1D920A00 CALL Capture.004AC9B5
 00403798 |. 53 PUSH EBX
 00403799 |. 50 PUSH EAX
 0040379A |. 8D4424 38 LEA EAX,DWORD PTR SS:[ESP+38]
 0040379E |. C64424 74 0E MOV BYTE PTR SS:[ESP+74],0E

004037A3 |. 50 PUSH EAX
 004037A4 |. E8 0C920A00 CALL Capture.004AC9B5
 004037A9 |. 55 PUSH EBP
 004037AA |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
 004037AE |. 50 PUSH EAX
 004037AF |. 51 PUSH ECX
 004037B0 |. C64424 78 0F MOV BYTE PTR SS:[ESP+78],0F
 004037B5 |. E8 FB910A00 CALL Capture.004AC9B5
 004037BA |. 57 PUSH EDI
 004037BB |. 8D5424 2C LEA EDX,DWORD PTR SS:[ESP+2C]
 004037BF |. 50 PUSH EAX
 004037C0 |. 52 PUSH EDX
 004037C1 |. C64424 78 10 MOV BYTE PTR SS:[ESP+78],10
 004037C6 |. E8 EA910A00 CALL Capture.004AC9B5
 004037CB |. 56 PUSH ESI
 004037CC |. 50 PUSH EAX
 004037CD |. 8D4424 2C LEA EAX,DWORD PTR SS:[ESP+2C]
 004037D1 |. B3 11 MOV BL,11
 004037D3 |. 50 PUSH EAX
 004037D4 |. 885C24 78 MOV BYTE PTR SS:[ESP+78],BL
 004037D8 |. E8 D8910A00 CALL Capture.004AC9B5
 004037DD |. 50 PUSH EAX
 004037DE |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
 004037E2 |. C64424 70 12 MOV BYTE PTR SS:[ESP+70],12
 004037E7 |. E8 D3900A00 CALL Capture.004AC8BF
 004037EC |. 8D4C24 24 LEA ECX,DWORD PTR SS:[ESP+24]
 004037F0 |. 885C24 6C MOV BYTE PTR SS:[ESP+6C],BL
 004037F4 |. E8 8D8F0A00 CALL Capture.004AC786
 004037F9 |. 8D4C24 28 LEA ECX,DWORD PTR SS:[ESP+28]
 004037FD |. C64424 6C 10 MOV BYTE PTR SS:[ESP+6C],10
 00403802 |. E8 7F8F0A00 CALL Capture.004AC786
 00403807 |. 8D4C24 2C LEA ECX,DWORD PTR SS:[ESP+2C]
 0040380B |. C64424 6C 0F MOV BYTE PTR SS:[ESP+6C],0F
 00403810 |. E8 718F0A00 CALL Capture.004AC786
 00403815 |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
 00403819 |. C64424 6C 0E MOV BYTE PTR SS:[ESP+6C],0E
 0040381E |. E8 638F0A00 CALL Capture.004AC786
 00403823 |. 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
 00403827 |. C64424 6C 0D MOV BYTE PTR SS:[ESP+6C],0D
 0040382C |. E8 558F0A00 CALL Capture.004AC786
 00403831 |. 8D4C24 38 LEA ECX,DWORD PTR SS:[ESP+38]
 00403835 |. C64424 6C 0C MOV BYTE PTR SS:[ESP+6C],0C
 0040383A |. E8 478F0A00 CALL Capture.004AC786
 0040383F |. 8D4C24 3C LEA ECX,DWORD PTR SS:[ESP+3C]
 00403843 |. C64424 6C 0B MOV BYTE PTR SS:[ESP+6C],0B
 00403848 |. E8 398F0A00 CALL Capture.004AC786
 0040384D |. 8D4C24 40 LEA ECX,DWORD PTR SS:[ESP+40]
 00403851 |. C64424 6C 0A MOV BYTE PTR SS:[ESP+6C],0A
 00403856 |. E8 2B8F0A00 CALL Capture.004AC786
 0040385B |. 8D4C24 44 LEA ECX,DWORD PTR SS:[ESP+44]
 0040385F |. C64424 6C 09 MOV BYTE PTR SS:[ESP+6C],9
 00403864 |. E8 1D8F0A00 CALL Capture.004AC786

00403869 |. C64424 6C 08 MOV BYTE PTR SS:[ESP+6C],8
 0040386E |. 8D4C24 48 LEA ECX,DWORD PTR SS:[ESP+48]
 00403872 |. E8 0F8F0A00 CALL Capture.004AC786
 00403877 |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
 0040387B |. C64424 6C 07 MOV BYTE PTR SS:[ESP+6C],7
 00403880 |. E8 018F0A00 CALL Capture.004AC786
 00403885 |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
 00403889 |. C64424 6C 06 MOV BYTE PTR SS:[ESP+6C],6
 0040388E |. E8 F38E0A00 CALL Capture.004AC786
 00403893 |. 8D4C24 54 LEA ECX,DWORD PTR SS:[ESP+54]
 00403897 |. C64424 6C 05 MOV BYTE PTR SS:[ESP+6C],5
 0040389C |. E8 E58E0A00 CALL Capture.004AC786
 004038A1 |. 8D4C24 58 LEA ECX,DWORD PTR SS:[ESP+58]
 004038A5 |. C64424 6C 04 MOV BYTE PTR SS:[ESP+6C],4
 004038AA |. E8 D78E0A00 CALL Capture.004AC786
 004038AF |. 8D4C24 5C LEA ECX,DWORD PTR SS:[ESP+5C]
 004038B3 |. C64424 6C 03 MOV BYTE PTR SS:[ESP+6C],3
 004038B8 |. E8 C98E0A00 CALL Capture.004AC786
 004038BD |. 6A 01 PUSH 1
 004038BF |. 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]
 004038C3 |. 6A 03 PUSH 3
 004038C5 |. 51 PUSH ECX
 004038C6 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 004038CD |. E8 350A0A00 CALL Capture.004A4307
 004038D2 |. 8BF0 MOV ESI,EAX
 004038D4 |. 6A 01 PUSH 1
 004038D6 |. 8D5424 20 LEA EDX,DWORD PTR SS:[ESP+20]
 004038DA |. 6A 19 PUSH 19
 004038DC |. 52 PUSH EDX
 004038DD |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 004038E4 |. C64424 78 13 MOV BYTE PTR SS:[ESP+78],13
 004038E9 |. E8 190A0A00 CALL Capture.004A4307
 004038EE |. 8BF8 MOV EDI,EAX
 004038F0 |. 6A 01 PUSH 1
 004038F2 |. 8D4424 24 LEA EAX,DWORD PTR SS:[ESP+24]
 004038F6 |. 6A 1C PUSH 1C
 004038F8 |. 50 PUSH EAX
 004038F9 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403900 |. C64424 78 14 MOV BYTE PTR SS:[ESP+78],14
 00403905 |. E8 FD090A00 CALL Capture.004A4307
 0040390A |. 8BE8 MOV EBP,EAX
 0040390C |. 6A 01 PUSH 1
 0040390E |. 8D4C24 28 LEA ECX,DWORD PTR SS:[ESP+28]
 00403912 |. 6A 16 PUSH 16
 00403914 |. 51 PUSH ECX
 00403915 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 0040391C |. C64424 78 15 MOV BYTE PTR SS:[ESP+78],15
 00403921 |. E8 E1090A00 CALL Capture.004A4307
 00403926 |. 8BD8 MOV EBX,EAX
 00403928 |. 6A 01 PUSH 1
 0040392A |. 8D5424 2C LEA EDX,DWORD PTR SS:[ESP+2C]
 0040392E |. 6A 1E PUSH 1E

00403930 |. 52 PUSH EDX
 00403931 |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403938 |. C64424 78 16 MOV BYTE PTR SS:[ESP+78],16
 0040393D |. E8 C5090A00 CALL Capture.004A4307
 00403942 |. 894424 54 MOV DWORD PTR SS:[ESP+54],EAX
 00403946 |. 6A 01 PUSH 1
 00403948 |. 8D4424 30 LEA EAX,DWORD PTR SS:[ESP+30]
 0040394C |. 6A 0D PUSH 0D
 0040394E |. 50 PUSH EAX
 0040394F |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403956 |. C64424 78 17 MOV BYTE PTR SS:[ESP+78],17
 0040395B |. E8 A7090A00 CALL Capture.004A4307
 00403960 |. 894424 58 MOV DWORD PTR SS:[ESP+58],EAX
 00403964 |. 6A 01 PUSH 1
 00403966 |. 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
 0040396A |. 6A 0C PUSH 0C
 0040396C |. 51 PUSH ECX
 0040396D |. 8D8C24 800000>LEA ECX,DWORD PTR SS:[ESP+80]
 00403974 |. C64424 78 18 MOV BYTE PTR SS:[ESP+78],18
 00403979 |. E8 89090A00 CALL Capture.004A4307
 0040397E |. 894424 5C MOV DWORD PTR SS:[ESP+5C],EAX
 00403982 |. 6A 01 PUSH 1
 00403984 |. C64424 70 19 MOV BYTE PTR SS:[ESP+70],19
 00403989 |. 6A 01 PUSH 1
 0040398B |. 8D5424 3C LEA EDX,DWORD PTR SS:[ESP+3C]
 0040398F |. 8D4C24 7C LEA ECX,DWORD PTR SS:[ESP+7C]
 00403993 |. 52 PUSH EDX
 00403994 |. E8 6E090A00 CALL Capture.004A4307
 00403999 |. 8B4C24 5C MOV ECX,DWORD PTR SS:[ESP+5C]
 0040399D |. 8D5424 38 LEA EDX,DWORD PTR SS:[ESP+38]
 004039A1 |. 51 PUSH ECX
 004039A2 |. 50 PUSH EAX
 004039A3 |. 52 PUSH EDX
 004039A4 |. C64424 78 1A MOV BYTE PTR SS:[ESP+78],1A
 004039A9 |. E8 07900A00 CALL Capture.004AC9B5
 004039AE |. 8B4C24 58 MOV ECX,DWORD PTR SS:[ESP+58]
 004039B2 |. 8D5424 3C LEA EDX,DWORD PTR SS:[ESP+3C]
 004039B6 |. 51 PUSH ECX
 004039B7 |. 50 PUSH EAX
 004039B8 |. 52 PUSH EDX
 004039B9 |. C64424 78 1B MOV BYTE PTR SS:[ESP+78],1B
 004039BE |. E8 F28F0A00 CALL Capture.004AC9B5
 004039C3 |. 8B4C24 54 MOV ECX,DWORD PTR SS:[ESP+54]
 004039C7 |. 8D5424 40 LEA EDX,DWORD PTR SS:[ESP+40]
 004039CB |. 51 PUSH ECX
 004039CC |. 50 PUSH EAX
 004039CD |. 52 PUSH EDX
 004039CE |. C64424 78 1C MOV BYTE PTR SS:[ESP+78],1C
 004039D3 |. E8 DD8F0A00 CALL Capture.004AC9B5
 004039D8 |. 53 PUSH EBX
 004039D9 |. 50 PUSH EAX
 004039DA |. 8D4424 4C LEA EAX,DWORD PTR SS:[ESP+4C]

004039DE |. C64424 74 1D MOV BYTE PTR SS:[ESP+74],1D
004039E3 |. 50 PUSH EAX
004039E4 |. E8 CC8F0A00 CALL Capture.004AC9B5
004039E9 |. 55 PUSH EBP
004039EA |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
004039EE |. 50 PUSH EAX
004039EF |. 51 PUSH ECX
004039F0 |. C64424 78 1E MOV BYTE PTR SS:[ESP+78],1E
004039F5 |. E8 BB8F0A00 CALL Capture.004AC9B5
004039FA |. 57 PUSH EDI
004039FB |. 8D5424 50 LEA EDX,DWORD PTR SS:[ESP+50]
004039FF |. 50 PUSH EAX
00403A00 |. 52 PUSH EDX
00403A01 |. C64424 78 1F MOV BYTE PTR SS:[ESP+78],1F
00403A06 |. E8 AA8F0A00 CALL Capture.004AC9B5
00403A0B |. 56 PUSH ESI
00403A0C |. 50 PUSH EAX
00403A0D |. 8D4424 58 LEA EAX,DWORD PTR SS:[ESP+58]
00403A11 |. C64424 74 20 MOV BYTE PTR SS:[ESP+74],20
00403A16 |. 50 PUSH EAX
00403A17 |. E8 998F0A00 CALL Capture.004AC9B5
00403A1C |. 50 PUSH EAX
00403A1D |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
00403A21 |. C64424 70 21 MOV BYTE PTR SS:[ESP+70],21
00403A26 |. E8 948E0A00 CALL Capture.004AC8BF
00403A2B |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
00403A2F |. C64424 6C 20 MOV BYTE PTR SS:[ESP+6C],20
00403A34 |. E8 4D8D0A00 CALL Capture.004AC786
00403A39 |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
00403A3D |. C64424 6C 1F MOV BYTE PTR SS:[ESP+6C],1F
00403A42 |. E8 3F8D0A00 CALL Capture.004AC786
00403A47 |. 8D4C24 48 LEA ECX,DWORD PTR SS:[ESP+48]
00403A4B |. C64424 6C 1E MOV BYTE PTR SS:[ESP+6C],1E
00403A50 |. E8 318D0A00 CALL Capture.004AC786
00403A55 |. 8D4C24 44 LEA ECX,DWORD PTR SS:[ESP+44]
00403A59 |. C64424 6C 1D MOV BYTE PTR SS:[ESP+6C],1D
00403A5E |. E8 238D0A00 CALL Capture.004AC786
00403A63 |. 8D4C24 40 LEA ECX,DWORD PTR SS:[ESP+40]
00403A67 |. C64424 6C 1C MOV BYTE PTR SS:[ESP+6C],1C
00403A6C |. E8 158D0A00 CALL Capture.004AC786
00403A71 |. 8D4C24 3C LEA ECX,DWORD PTR SS:[ESP+3C]
00403A75 |. C64424 6C 1B MOV BYTE PTR SS:[ESP+6C],1B
00403A7A |. E8 078D0A00 CALL Capture.004AC786
00403A7F |. C64424 6C 1A MOV BYTE PTR SS:[ESP+6C],1A
00403A84 |. 8D4C24 38 LEA ECX,DWORD PTR SS:[ESP+38]
00403A88 |. E8 F98C0A00 CALL Capture.004AC786
00403A8D |. 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
00403A91 |. C64424 6C 19 MOV BYTE PTR SS:[ESP+6C],19
00403A96 |. E8 EB8C0A00 CALL Capture.004AC786
00403A9B |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
00403A9F |. C64424 6C 18 MOV BYTE PTR SS:[ESP+6C],18
00403AA4 |. E8 DD8C0A00 CALL Capture.004AC786

```

00403AA9 |. 8D4C24 2C    LEA ECX,DWORD PTR SS:[ESP+2C]
00403AAD |. C64424 6C 17  MOV BYTE PTR SS:[ESP+6C],17
00403AB2 |. E8 CF8C0A00  CALL Capture.004AC786
00403AB7 |. 8D4C24 28    LEA ECX,DWORD PTR SS:[ESP+28]
00403ABB |. C64424 6C 16  MOV BYTE PTR SS:[ESP+6C],16
00403AC0 |. E8 C18C0A00  CALL Capture.004AC786
00403AC5 |. 8D4C24 24    LEA ECX,DWORD PTR SS:[ESP+24]
00403AC9 |. C64424 6C 15  MOV BYTE PTR SS:[ESP+6C],15
00403ACE |. E8 B38C0A00  CALL Capture.004AC786
00403AD3 |. 8D4C24 20    LEA ECX,DWORD PTR SS:[ESP+20]
00403AD7 |. C64424 6C 14  MOV BYTE PTR SS:[ESP+6C],14
00403ADC |. E8 A58C0A00  CALL Capture.004AC786
00403AE1 |. 8D4C24 1C    LEA ECX,DWORD PTR SS:[ESP+1C]
00403AE5 |. C64424 6C 13  MOV BYTE PTR SS:[ESP+6C],13
00403AEA |. E8 978C0A00  CALL Capture.004AC786
00403AEF |. 8D4C24 18    LEA ECX,DWORD PTR SS:[ESP+18]
00403AF3 |. C64424 6C 03  MOV BYTE PTR SS:[ESP+6C],3
00403AF8 |. E8 898C0A00  CALL Capture.004AC786
00403AF8 |. E8 898C0A00  CALL Capture.004AC786
00403AFD |. 51      PUSH ECX
00403AFE |. 8D5424 14    LEA EDX,DWORD PTR SS:[ESP+14]
00403B02 |. 8BCC      MOV ECX,ESP
00403B04 |. 896424 60    MOV DWORD PTR SS:[ESP+60],ESP
00403B08 |. 52      PUSH EDX
00403B09 |. E8 ED890A00  CALL Capture.004AC4FB           ; <== StringII
00403B0E |. 51      PUSH ECX
00403B0F |. 8D4424 1C    LEA EAX,DWORD PTR SS:[ESP+1C]
00403B13 |. 8BCC      MOV ECX,ESP
00403B15 |. 896424 60    MOV DWORD PTR SS:[ESP+60],ESP
00403B19 |. 50      PUSH EAX
00403B1A |. C64424 78 22  MOV BYTE PTR SS:[ESP+78],22
00403B1F |. E8 D7890A00  CALL Capture.004AC4FB           ; <== StringI

```

- Sau đó, chương trình sẽ tiến hành mã hóa như sau :

```

00403B24 |. 8B4C24 68    MOV ECX,DWORD PTR SS:[ESP+68]   ; |
00403B28 |. C64424 74 03  MOV BYTE PTR SS:[ESP+74],3       ; |
00403B2D |. E8 8E000000  CALL Capture.00403BC0           ; \Capture.00403BC0
----- Encrypt & Check -----

```

```

00403BE8 |. 8B40 F8    MOV EAX,DWORD PTR DS:[EAX-8]     ; <== StringI
00403BEB |. 83F8 08    CMP EAX,8                      ; <== Length must be 8 charts
00403BEE |. 0F85 E8000000 JNZ Capture.00403CDC
00403BF4 |. 8B4C24 30    MOV ECX,DWORD PTR SS:[ESP+30]   ; <== StringII
00403BF8 |. 8379 F8 08  CMP DWORD PTR DS:[ECX-8],8      ; <== Length must be 8 charts
00403BFC |. 0F85 DA000000 JNZ Capture.00403CDC
00403C02 |. 68 4C945000 PUSH Capture.0050944C          ; ASCII
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"
00403C07 |. 8D4C24 14    LEA ECX,DWORD PTR SS:[ESP+14]
00403C0B |. E8 E48B0A00  CALL Capture.004AC7F4
00403C10 |. B3 02      MOV BL,2
00403C12 |. 33F6      XOR ESI,ESI
00403C14 |. 885C24 24    MOV BYTE PTR SS:[ESP+24],BL
00403C18 |> 6A 01      /PUSH 1

```

```

00403C1A |. 8D5424 18 |LEA EDX,DWORD PTR SS:[ESP+18]
00403C1E |. 56 |PUSH ESI
00403C1F |. 52 |PUSH EDX
00403C20 |. 8D4C24 38 |LEA ECX,DWORD PTR SS:[ESP+38]
00403C24 |. E8 DE060A00 |CALL Capture.004A4307 ; <== StringI[i]
00403C29 |. 8B00 |MOV EAX,DWORD PTR DS:[EAX] ; <== /
00403C2B |. 8D4C24 10 |LEA ECX,DWORD PTR SS:[ESP+10] ; <== | if (StringI[i] == dStr[j])
00403C2F |. 50 |PUSH EAX ; <== | then
00403C30 |. C64424 28 03 |MOV BYTE PTR SS:[ESP+28],3 ; <== | ValueI = j
00403C35 |. E8 A4080A00 |CALL Capture.004A44DE ; <== \
00403C3A |. 8D4C24 14 |LEA ECX,DWORD PTR SS:[ESP+14]
00403C3E |. 8BF8 |MOV EDI,EAX
00403C40 |. 885C24 24 |MOV BYTE PTR SS:[ESP+24],BL
00403C44 |. E8 3D8B0A00 |CALL Capture.004AC786
00403C49 |. 6A 01 |PUSH 1
00403C4B |. 8D4424 1C |LEA EAX,DWORD PTR SS:[ESP+1C]
00403C4F |. 56 |PUSH ESI
00403C50 |. 50 |PUSH EAX
00403C51 |. 8D4C24 3C |LEA ECX,DWORD PTR SS:[ESP+3C]
00403C55 |. E8 AD060A00 |CALL Capture.004A4307 ; <== StringII[i]
00403C5A |. 8B00 |MOV EAX,DWORD PTR DS:[EAX] ; <== /
00403C5C |. 8D4C24 10 |LEA ECX,DWORD PTR SS:[ESP+10] ; <== | if (StringII[i] == dStr[j])
00403C60 |. 50 |PUSH EAX ; <== | then
00403C61 |. C64424 28 04 |MOV BYTE PTR SS:[ESP+28],4 ; <== | ValueII = j
00403C66 |. E8 73080A00 |CALL Capture.004A44DE ; <== \
00403C6B |. 8D4C24 18 |LEA ECX,DWORD PTR SS:[ESP+18]
00403C6F |. 8BE8 |MOV EBP,EAX
00403C71 |. 885C24 24 |MOV BYTE PTR SS:[ESP+24],BL
00403C75 |. E8 0C8B0A00 |CALL Capture.004AC786
00403C7A |. 83FF FF |CMP EDI,-1
00403C7D |. 74 4F |JE SHORT Capture.00403CCE
00403C7F |. 83FD FF |CMP EBP,-1
00403C82 |. 74 4A |JE SHORT Capture.00403CCE
00403C84 |. 8D47 0A |LEA EAX,DWORD PTR DS:[EDI+A] ; <== Check = ValueI + 0xA
00403C87 |. B9 3E000000 |MOV ECX,3E ; <== dValue
00403C8C |. 99 |CDQ
00403C8D |. F7F9 |IDIV ECX ; <== Check = Check % dValue
00403C8F |. 3BD5 |CMP EDX,EBP ; <== if (Check == ValueII)
00403C91 |. 75 3B |JNZ SHORT Capture.00403CCE ; <== Continue Check
00403C93 |. 46 |INC ESI ; <== i++
00403C94 |. 83FE 08 |CMP ESI,8 ; <== while ( j < 8 )
00403C97 |.^ 0F8C 7BFFFFFF |JL Capture.00403C18 ; <== Continue
----- Encrypt & Check -----
/*/*/* - SERIAL tương ứng :
User : REA-cRaCkErTeAm Serial : PZkUlMiG3Kb8WQP6tM2QBINDSK03s4hA

```

III – KeyGen :

- /Section I /- Tạo chuỗi ngẫu nhiên gồm 32 ký tự .
- /Section II /- Từ đây ta xác định chuỗi chính theo các vị trí : [0][5][7][11][23][6][31][9]II
- /Section III /- Xác định vị trí các ký tự của chuỗi này với chuỗi mặc định .
- /Section IV /- Từng vị trí này sẽ được tính toán theo công thức :

$$\text{Value} = (\text{ChartPosition}(\text{Value}) + 0xA) \% 0x3E;$$

- /Section V /- Ứng với mỗi vị trí này ta xác định được ký tự tương ứng trong chuỗi mặc định .
- /Section VI /- Thay thế ký tự mới tìm vào các vị trí sau trong chuỗi ngẫu nhiên ban đầu :
[I][12][13][30][22][28][25][3]
- /Section VII /- Đây chính là chuỗi Serial thực.

IV – End of Tut :

- Finished – *August 15, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.etrusoft.com
Production :	EtruSoft Inc.,
SoftWare :	Quick Screenshot Maker 2.1
Copyright by :	Copyright © 2002-2004 EtruSoft Inc,. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

Quick Screenshot Maker 2.1

Quick Screenshot Maker is an all-in-one tool for screen capturing, image editing and organization. It can capture any part of the screen precisely in flexible ways, including DirectX, screen saver and movie screens. Use Quick Screenshot Maker to save time and enhance your screen shots. Add annotations, text, arrows, highlights, clipart and more. The History Bar helps you find all of your captured images and organize them quickly and easily.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 6.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Sorry, your registration key is wrong. Please check it and try again.**" . Ta tìm được thông báo này tại địa chỉ :
00430769 . 68 906F5800 PUSH Screensh.00586F90 ; ASCII "Sorry, your registration key is wrong. Please check it and try again."

- Dò ngược lên trên và đặt BreakPoint tại :
004306E2 . E8 849D0C00 CALL Screensh.004FA46B ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :
004306E9 . 8378 F8 02 CMP DWORD PTR DS:[EAX-8],2 ; <== LenU atleast 2 charts
004306ED . 7D 26 JGE SHORT Screensh.00430715

- Trace tiếp ta đến :

```
00430750 . 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C] ; |
00430754 . C64424 2C 00 MOV BYTE PTR SS:[ESP+2C],0 ; |
00430759 . E8 7269FDFF CALL Screensh.004070D0 ; \Screensh.004070D0
```

- Trace into để xem xét quá trình mã hoá :

```
0040713B |. 83F8 02 CMP EAX,2 ; <== LenU atleast 2 charts
```

```
0040713E |. 0F8C 2C030000 JL Screensh.00407470
```

```
00407144 |. 8B5424 44 MOV EDX,DWORD PTR SS:[ESP+44]
```

```
00407148 |. 837A F8 18 CMP DWORD PTR DS:[EDX-8],18 ; <== LenS must be 24 charts
```

```
0040714C |. 0F85 1E030000 JNZ Screensh.00407470
```

- Kế đó, chương trình sẽ lấy hai ký tự đầu cuối của chuỗi U nhập, so sánh với chuỗi ký tự (a-z, A-Z, 9-0) xem hai ký tự này nằm ở vị trí thứ bao nhiêu trong chuỗi ký tự mặc định này, nếu chuỗi U nhập đầu cuối là các ký tự đặc biệt thì chương trình tự động gán giá trị là “-1” :

```
00407156 |. 6A 01 PUSH 1
00407158 |. 50 PUSH EAX
00407159 |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
0040715D |. E8 A72E0F00 CALL Screensh.004FA009
00407162 |. 8B00 MOV EAX,DWORD PTR DS:[EAX]
00407164 |. 8D4C24 20 LEA ECX,DWORD PTR SS:[ESP+20]
00407168 |. 50 PUSH EAX
00407169 |. C64424 40 04 MOV BYTE PTR SS:[ESP+40],4
0040716E |. E8 5B2F0F00 CALL Screensh.004FA0CE
00407173 |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
00407177 |. 8BF0 MOV ESI,EAX
00407179 |. C64424 3C 03 MOV BYTE PTR SS:[ESP+3C],3
0040717E |. E8 7E6C0F00 CALL Screensh.004FDE01
00407183 |. 8D46 0A LEA EAX,DWORD PTR DS:[ESI+A] ; <== U[0] --> uVI
00407186 |. B9 3E000000 MOV ECX,3E ; <== dV
0040718B |. 99 CDQ
0040718C |. F7F9 IDIV ECX ; <== uVI = uVI % dV
0040718E |. 6A 01 PUSH 1
00407190 |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
00407194 |. 8BF2 MOV ESI,EDX
00407196 |. 8D5424 34 LEA EDX,DWORD PTR SS:[ESP+34]
0040719A |. 52 PUSH EDX
0040719B |. E8 ED2D0F00 CALL Screensh.004F9F8D
004071A0 |. 8B00 MOV EAX,DWORD PTR DS:[EAX]
004071A2 |. 8D4C24 20 LEA ECX,DWORD PTR SS:[ESP+20]
004071A6 |. 50 PUSH EAX
004071A7 |. C64424 40 05 MOV BYTE PTR SS:[ESP+40],5
004071AC |. E8 1D2F0F00 CALL Screensh.004FA0CE
004071B1 |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
004071B5 |. 8BF8 MOV EDI,EAX
004071B7 |. C64424 3C 03 MOV BYTE PTR SS:[ESP+3C],3
004071BC |. E8 406C0F00 CALL Screensh.004FDE01
004071C1 |. 8D47 0A LEA EAX,DWORD PTR DS:[EDI+A] ; <== U[Len] --> uVII
004071C4 |. B9 3E000000 MOV ECX,3E ; <== dV
004071C9 |. 99 CDQ
004071CA |. F7F9 IDIV ECX ; <== uVII = uVII % dV
```

- Sau đó, dựa trên hai giá trị này chương trình tính toán đoạn đầu tiên của chuỗi Serial thực (4 ký tự) :

```

004071EB |. 8BC1      MOV EAX,ECX          ; <== V_01 = uVII
004071ED |. BF 0A000000 MOV EDI,0A        ; <== 0xA
004071F2 |. 99       CDQ
004071F3 |. F7FF      IDIV EDI          ; <== V_01 = V_01 % 0xA
004071F5 |. 8BC1      MOV EAX,ECX          ; <== V_02 = uVII
004071F7 |. B3 0B      MOV BL,0B           ; <== 0xB
004071F9 |. 0FAFC1    IMUL EAX,ECX         ; <== V_02 = V_02 * uVII
004071FC |. 8BCF      MOV ECX,EDI          ; <== 0xA
004071FE |. 885C24 3C  MOV BYTE PTR SS:[ESP+3C],BL
00407202 |. 52       PUSH EDX          ; <== V_01
00407203 |. 99       CDQ
00407204 |. F7F9      IDIV ECX          ; <== V_02 = V_02 % 0xA
00407206 |. 8D04F5 000000>LEA EAX,DWORD PTR DS:[ESI*8]
0040720D |. 2BC6      SUB EAX,ESI         ; <== V_03 = uVI * 0x8
0040720F |. 52       PUSH EDX          ; <== V_03 - uVI
00407210 |. 99       CDQ               ; <== V_02
00407211 |. F7F9      IDIV ECX          ; <== V_03 = V_03 % 0xA
00407213 |. 8BC6      MOV EAX,ESI          ; <== V_04 = uVI
00407215 |. 52       PUSH EDX          ; <== V_03
00407216 |. 99       CDQ               ; <== V_04 = V_04 % 0xA
00407217 |. F7F9      IDIV ECX          ; <== V_04
00407219 |. 52       PUSH EDX          ; <== V_04
0040721A |. 8D5424 38  LEA EDX,DWORD PTR SS:[ESP+38]
0040721E |. 68 04555800 PUSH Screensh.00585504
00407223 |. 52       PUSH EDX          ; ASCII "%d%d%d%d"
00407224 |. E8 E6310F00 CALL Screensh.004FA40F      ; <== Convert to String

```

- Tiếp theo, chương trình sẽ cắt chuỗi Serial thành 5 đoạn, mỗi đoạn 4 ký tự, và ta hình dung được dạng của Serial “**AAAA-BBBB-CCCC-DDDD-EEEE**” :

```

0040722C |. 6A 04      PUSH 4
0040722E |. 8D4424 34  LEA EAX,DWORD PTR SS:[ESP+34]
00407232 |. 6A 00      PUSH 0
00407234 |. 50       PUSH EAX
00407235 |. 8D4C24 50  LEA ECX,DWORD PTR SS:[ESP+50]
00407239 |. E8 B92C0F00 CALL Screensh.004F9EF7
0040723E |. 50       PUSH EAX
0040723F |. 8D4C24 28  LEA ECX,DWORD PTR SS:[ESP+28]
00407243 |. C64424 40 0C MOV BYTE PTR SS:[ESP+40],0C
00407248 |. E8 ED6C0F00 CALL Screensh.004FDF3A
0040724D |. 8D4C24 30  LEA ECX,DWORD PTR SS:[ESP+30]
00407251 |. 885C24 3C  MOV BYTE PTR SS:[ESP+3C],BL
00407255 |. E8 A76B0F00 CALL Screensh.004FDE01
0040725A |. 6A 04      PUSH 4
0040725C |. 8D4C24 34  LEA ECX,DWORD PTR SS:[ESP+34]
00407260 |. 6A 05      PUSH 5
00407262 |. 51       PUSH ECX
00407263 |. 8D4C24 50  LEA ECX,DWORD PTR SS:[ESP+50]
00407267 |. E8 8B2C0F00 CALL Screensh.004F9EF7
0040726C |. 50       PUSH EAX
0040726D |. 8D4C24 20  LEA ECX,DWORD PTR SS:[ESP+20]
00407271 |. C64424 40 0D MOV BYTE PTR SS:[ESP+40],0D

```

```

00407276 |. E8 BF6C0F00 CALL Screensh.004FDF3A
0040727B |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
0040727F |. 885C24 3C MOV BYTE PTR SS:[ESP+3C],BL
00407283 |. E8 796B0F00 CALL Screensh.004FDE01
00407288 |. 6A 04 PUSH 4
0040728A |. 8D5424 34 LEA EDX,DWORD PTR SS:[ESP+34]
0040728E |. 57 PUSH EDI
0040728F |. 52 PUSH EDX
00407290 |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
00407294 |. E8 5E2C0F00 CALL Screensh.004F9EF7
00407299 |. 50 PUSH EAX
0040729A |. 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]
0040729E |. C64424 40 0E MOV BYTE PTR SS:[ESP+40],0E
004072A3 |. E8 926C0F00 CALL Screensh.004FDF3A
004072A8 |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
004072AC |. 885C24 3C MOV BYTE PTR SS:[ESP+3C],BL
004072B0 |. E8 4C6B0F00 CALL Screensh.004FDE01
004072B5 |. 6A 04 PUSH 4
004072B7 |. 8D4424 34 LEA EAX,DWORD PTR SS:[ESP+34]
004072BB |. 6A 0F PUSH 0F
004072BD |. 50 PUSH EAX
004072BE |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
004072C2 |. E8 302C0F00 CALL Screensh.004F9EF7
004072C7 |. 50 PUSH EAX
004072C8 |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
004072CC |. C64424 40 0F MOV BYTE PTR SS:[ESP+40],0F
004072D1 |. E8 646C0F00 CALL Screensh.004FDF3A
004072D6 |. 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
004072DA |. 885C24 3C MOV BYTE PTR SS:[ESP+3C],BL
004072DE |. E8 1E6B0F00 CALL Screensh.004FDE01
004072E3 |. 6A 04 PUSH 4
004072E5 |. 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
004072E9 |. 6A 14 PUSH 14
004072EB |. 51 PUSH ECX
004072EC |. 8D4C24 50 LEA ECX,DWORD PTR SS:[ESP+50]
004072F0 |. E8 022C0F00 CALL Screensh.004F9EF7
004072F5 |. 50 PUSH EAX
004072F6 |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
004072FA |. C64424 40 10 MOV BYTE PTR SS:[ESP+40],10
004072FF |. E8 366C0F00 CALL Screensh.004FDF3A

```

- Quá trình so sánh đầu tiên được diễn ra giữa đoạn đầu tiên của Serial nhập với chuỗi được tính toán từ U :

00407311 . 8B5424 24 MOV EDX,DWORD PTR SS:[ESP+24]	
00407315 . 8B4424 28 MOV EAX,DWORD PTR SS:[ESP+28]	
00407319 . 52 PUSH EDX	; /Arg2
0040731A . 50 PUSH EAX	; Arg1
0040731B . E8 8AF80800 CALL Screensh.00496BAA	; \Screensh.00496BAA

- 4 đoạn còn lại của chuỗi Serial nhập sẽ được chuyển sang dạng giá trị HEX tương ứng :

00407340 . 50 PUSH EAX	
00407341 . E8 EB6E0F00 CALL Screensh.004FE231	

```

00407346 |. 50      PUSH EAX
00407347 |. E8 48F70800 CALL Screensh.00496A94 ; <== Convert to HEX Value
0040734C |. 8B5424 1C  MOV EDX,DWORD PTR SS:[ESP+1C]
00407350 |. 83C4 04  ADD ESP,4
00407353 |. 8BF0      MOV ESI,EAX
00407355 |. 8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18]
00407359 |. 8B42 F8  MOV EAX,DWORD PTR DS:[EDX-8]
0040735C |. 50      PUSH EAX
0040735D |. E8 CF6E0F00 CALL Screensh.004FE231
00407362 |. 50      PUSH EAX
00407363 |. E8 2CF70800 CALL Screensh.00496A94 ; <== Convert to HEX Value
00407368 |. 8BF8      MOV EDI,EAX
0040736A |. 8B4424 18  MOV EAX,DWORD PTR SS:[ESP+18]
0040736E |. 83C4 04  ADD ESP,4
00407371 |. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
00407375 |. 8B40 F8  MOV EAX,DWORD PTR DS:[EAX-8]
00407378 |. 50      PUSH EAX
00407379 |. E8 B36E0F00 CALL Screensh.004FE231
0040737E |. 50      PUSH EAX
0040737F |. E8 10F70800 CALL Screensh.00496A94 ; <== Convert to HEX Value
00407384 |. 8B4C24 14  MOV ECX,DWORD PTR SS:[ESP+14]
00407388 |. 83C4 04  ADD ESP,4
0040738B |. 8BD8      MOV EBX,EAX
0040738D |. 8B41 F8  MOV EAX,DWORD PTR DS:[ECX-8]
00407390 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
00407394 |. 50      PUSH EAX
00407395 |. E8 976E0F00 CALL Screensh.004FE231
0040739A |. 50      PUSH EAX
0040739B |. E8 F4F60800 CALL Screensh.00496A94 ; <== Convert to HEX Value

```

- Cuối cùng, chương trình sẽ tiến hành tính toán và so sánh như sau (1) Giá trị HEX của đoạn thứ II được tính toán và so sánh với giá trị HEX của đoạn thứ III (2) Giá trị HEX của đoạn thứ IV được tính toán và so sánh với giá trị HEX của đoạn thứ V :

```

004073D1 |. 8D94B6 04AB00>LEA EDX,DWORD PTR DS:[ESI+ESI*4+AB04] ; <== CheckI =
ValueI * 5 + 0xAB04
004073D8 |. B9 10270000  MOV ECX,2710 ; <== dV
004073DD |. 8D8456 342200>LEA EAX,DWORD PTR DS:[ESI+EDX*2+2234] ; <== CheckI =
CheckI * 2 + ValueI + 0x2234
004073E4 |. D1E0      SHL EAX,1 ; <== CheckI = CheckI * 2
004073E6 |. 99      CDQ
004073E7 |. F7F9      IDIV ECX ; <== CheckI = CheckI % dV
004073E9 |. 3BFA      CMP EDI,EDX ; <== if ( CheckI == ValueII )
004073EB |. 74 0B      JE SHORT Screensh.004073F8 ; <== Continue Check
004073ED |. C64424 3C 0A  MOV BYTE PTR SS:[ESP+3C],0A
004073F2 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
004073F6 |. EB 2D      JMP SHORT Screensh.00407425
004073F8 |> 8D83 CAEFFFF LEA EAX,DWORD PTR DS:[EBX-1536] ; <== ValueIII
004073FE |. 81C3 E2090000 ADD EBX,9E2 ; <== Temp = ValueIII + 0x9E2
00407404 |. 99      CDQ
00407405 |. 33C2      XOR EAX,EDX
00407407 |. B9 10270000  MOV ECX,2710 ; <== dV
0040740C |. 2BC2      SUB EAX,EDX

```

```

0040740E |. C64424 3C 0A MOV BYTE PTR SS:[ESP+3C],0A
00407413 |. 0FAFC3    IMUL EAX,EBX           ; <== CheckII = (0x1536 - ValueIII) * Temp
00407416 |. 99        CDQ
00407417 |. F7F9    IDIV ECX           ; <== CheckII = CheckII % dV
00407419 |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]
0040741D |. 3BEA    CMP EBP,EDX          ; <== if ( CheckII == ValueIV )
0040741F  0F84 8D000000 JE Screensh.004074B2   ; <== ConGrat!! !!!!!!!!

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 3142-0041-3534-8467-7889

III – End of Tut :

- Finished – September 16, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.realrecorder.net/realspy
Production :	ShareStar Inc.
SoftWare :	Real Spy Monitor 1.99
Copyright by :	Copyright © 2002-200 ShareStar Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual Basic 5.0 / 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Real Spy Monitor 1.99

Worried about how your PC is being used? Want to keep tabs on your children, spouse, employees? Need to Prevent your children or employee from some application or web sites? Real Spy Monitor is the full solution for you. For example, you can use **Real Spy Monitor** to Monitor Keystrokes typed, Websites visited, Windows viewed, Program executed, Screen snapshots, Files/Docs accessed. Log Internet Chat conservation including AOL/ICQ/MSN/AIM Instant Messengers Spy Web Mail Content including MSN/HotMail, Yahoo! Mail Prevent your children or employee from some application or websites that include special keywords. When you left your PC, Record your PC actions and send them through Email delivery at set times.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual Basic 5.0 / 6.0**
 - Nhập thử User và Fake Serial, ta nhận được thông báo "Registration Key Wrong". Ta tìm chuỗi thông báo này trong Olly, và tìm được ở địa chỉ :
- 00461BF6 . C785 74FFFFFF>MOV DWORD PTR SS:[EBP-8C],winrsm.0041336>; UNICODE "Registration Key Wrong"
- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :
- 00461600 . FF56 04 CALL DWORD PTR DS:[ESI+4] ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống một đoạn ta đến quá trình chuyển dạng U sang UpperCase :

004616A3 . FF15 B4A34900 CALL DWORD PTR DS:[<&MSVBVM50.#528>] ;
MSVBVM50.rtcUpperCaseVar

- Chuỗi U dạngUpperCase này sẽ được dùng để mã hoá . Quá trình mã hoá diễn ra như sau :

```
004616CE . FF15 A8A44900 CALL DWORD PTR DS:[<&MSVBVM50.__vbaStrCo>];  
MSVBVM50.__vbaStrCopy  
004616D4 > 0FBF4D D8 MOVSX ECX,WORD PTR SS:[EBP-28]  
004616D8 . 8B55 E4 MOV EDX,DWORD PTR SS:[EBP-1C]  
004616DB . 898D 1CFFFFFF MOV DWORD PTR SS:[EBP-E4],ECX  
004616E1 . 52 PUSH EDX  
004616E2 . FF15 E4A24900 CALL DWORD PTR DS:[<&MSVBVM50.__vbaLenBs>];  
MSVBVM50.__vbaLenBstr  
004616E8 . 8B8D 1CFFFFFF MOV ECX,DWORD PTR SS:[EBP-E4] ; <== i = 1  
004616EE . 3BC8 CMP ECX,EAX ; <== if ( i < LenU + 1 )  
004616F0 . 0F8F 4D010000 JG winrsm.00461843 ; <== Continue Loop  
004616F6 . 8D45 E4 LEA EAX,DWORD PTR SS:[EBP-1C]  
004616F9 . 8D55 AC LEA EDX,DWORD PTR SS:[EBP-54]  
004616FC . 8985 74FFFFFF MOV DWORD PTR SS:[EBP-8C],EAX  
00461702 . 52 PUSH EDX  
00461703 . 51 PUSH ECX  
00461704 . 8D85 6CFFFFFF LEA EAX,DWORD PTR SS:[EBP-94]  
0046170A . 8D4D 9C LEA ECX,DWORD PTR SS:[EBP-64]  
0046170D . 50 PUSH EAX  
0046170E . 51 PUSH ECX  
0046170F . C745 B4 01000>MOV DWORD PTR SS:[EBP-4C],1  
00461716 . C745 AC 02000>MOV DWORD PTR SS:[EBP-54],2  
0046171D . C785 6CFFFFFF>MOV DWORD PTR SS:[EBP-94],4008  
00461727 . FF15 A0A34900 CALL DWORD PTR DS:[<&MSVBVM50.#632>] ;  
MSVBVM50.rtcMidCharVar  
0046172D . 8D55 9C LEA EDX,DWORD PTR SS:[EBP-64]  
00461730 . 52 PUSH EDX  
00461731 . FFD6 CALL ESI  
00461733 . 8BD0 MOV EDX,EAX  
00461735 . 8D4D E0 LEA ECX,DWORD PTR SS:[EBP-20]  
00461738 . FFD7 CALL EDI  
0046173A . 8D45 9C LEA EAX,DWORD PTR SS:[EBP-64]  
0046173D . 8D4D AC LEA ECX,DWORD PTR SS:[EBP-54]  
00461740 . 50 PUSH EAX  
00461741 . 51 PUSH ECX  
00461742 . 6A 02 PUSH 2  
00461744 . FFD3 CALL EBX  
00461746 . 8B55 E0 MOV EDX,DWORD PTR SS:[EBP-20]  
00461749 . 83C4 0C ADD ESP,0C  
0046174C . 52 PUSH EDX  
0046174D . FF15 00A34900 CALL DWORD PTR DS:[<&MSVBVM50.#516>] ;  
MSVBVM50.rtcAnsiValueBstr  
00461753 . 66:8B4D D8 MOV CX,WORD PTR SS:[EBP-28] ; <== i  
00461757 . 66:6BC9 06 IMUL CX,CX,6 ; <== Value = i * 6
```

```

0046175B . 0F80 C5050000 JO winrsm.00461D26
00461761 . 66:03C1 ADD AX,CX ; <== Value = Value + U[i-1]
00461764 . 0F80 BC050000 JO winrsm.00461D26
0046176A . 66:2D 0F00 SUB AX,0F ; <== Value = Value - 0xF
0046176E . 0F80 B2050000 JO winrsm.00461D26
00461774 . 66:3D 2100 CMP AX,21 ; <== if ( Value > 0x21 )
00461778 . 7C 56 JL SHORT winrsm.004617D0 ; <== else
0046177A . 66:3D 7E00 CMP AX,7E ; <== if ( Value < 0x7E )
0046177E . 7F 4A JG SHORT winrsm.004617CA ; <== Then
00461780 . 8B55 DC MOV EDX,DWORD PTR SS:[EBP-24] ; <== S[i-1] = Value
00461783 . 8D4D AC LEA ECX,DWORD PTR SS:[EBP-54]
00461786 . 0FBFC0 MOVSX EAX,AX
00461789 . 50 PUSH EAX
0046178A . 51 PUSH ECX
0046178B . 8995 74FFFFFF MOV DWORD PTR SS:[EBP-8C],EDX
00461791 . C785 6CFFFFFF>MOV DWORD PTR SS:[EBP-94],8
0046179B . FF15 50A44900 CALL DWORD PTR DS:[<&MSVBVM50.#608>] ; MSVBVM50.rtcVarBstrFromAns
004617A1 . 8D95 6CFFFFFF LEA EDX,DWORD PTR SS:[EBP-94]
004617A7 . 8D45 AC LEA EAX,DWORD PTR SS:[EBP-54]
004617AA . 52 PUSH EDX
004617AB . 8D4D 9C LEA ECX,DWORD PTR SS:[EBP-64]
004617AE . 50 PUSH EAX
004617AF . 51 PUSH ECX
004617B0 . FF15 ECA44900 CALL DWORD PTR DS:[<&MSVBVM50.__vbaVarAd>]; MSVBVM50.__vbaVarAdd
004617B6 . 50 PUSH EAX
004617B7 . FFD6 CALL ESI
004617B9 . 8BD0 MOV EDX,EAX
004617BB . 8D4D DC LEA ECX,DWORD PTR SS:[EBP-24]
004617BE . FFD7 CALL EDI
004617C0 . 8D55 9C LEA EDX,DWORD PTR SS:[EBP-64]
004617C3 . 8D45 AC LEA EAX,DWORD PTR SS:[EBP-54]
004617C6 . 52 PUSH EDX
004617C7 . 50 PUSH EAX
004617C8 . EB 5E JMP SHORT winrsm.00461828 ; <== else
004617CA > 66:3D 2100 CMP AX,21 ; <== if ( Value < 0x21 )
004617CE . 7D 05 JGE SHORT winrsm.004617D5 ; <== then
004617D0 > B8 65000000 MOV EAX,65 ; <== S[i-1] = 0x65
004617D5 > 66:3D 7E00 CMP AX,7E ; <== else if ( Value > 0x7E )
004617D9 . 7E 05 JLE SHORT winrsm.004617E0 ; <== then
004617DB . B8 2A000000 MOV EAX,2A ; <== S[i-1] = 0x2A
004617E0 > 8B4D DC MOV ECX,DWORD PTR SS:[EBP-24]
004617E3 . C785 6CFFFFFF>MOV DWORD PTR SS:[EBP-94],8
004617ED . 0FBFD0 MOVSX EDX,AX
004617F0 . 8D45 AC LEA EAX,DWORD PTR SS:[EBP-54]
004617F3 . 52 PUSH EDX
004617F4 . 50 PUSH EAX
004617F5 . 898D 74FFFFFF MOV DWORD PTR SS:[EBP-8C],ECX
004617FB . FF15 50A44900 CALL DWORD PTR DS:[<&MSVBVM50.#608>] ; MSVBVM50.rtcVarBstrFromAns
00461801 . 8D8D 6CFFFFFF LEA ECX,DWORD PTR SS:[EBP-94]

```

```
00461807 . 8D55 AC    LEA EDX,DWORD PTR SS:[EBP-54]
0046180A . 51        PUSH ECX
0046180B . 8D45 9C    LEA EAX,DWORD PTR SS:[EBP-64]
0046180E . 52        PUSH EDX
0046180F . 50        PUSH EAX
00461810 . FF15 ECA44900 CALL DWORD PTR DS:<&MSVBVM50.__vbaVarAd>;
MSVBVM50.__vbaVarAdd
00461816 . 50        PUSH EAX
00461817 . FFD6      CALL ESI
00461819 . 8BD0      MOV EDX,EAX
0046181B . 8D4D DC    LEA ECX,DWORD PTR SS:[EBP-24]
0046181E . FFD7      CALL EDI
00461820 . 8D4D 9C    LEA ECX,DWORD PTR SS:[EBP-64]
00461823 . 8D55 AC    LEA EDX,DWORD PTR SS:[EBP-54]
00461826 . 51        PUSH ECX
00461827 . 52        PUSH EDX
00461828 > 6A 02    PUSH 2
0046182A . FFD3      CALL EBX
0046182C . 66:8B45 D8  MOV AX,WORD PTR SS:[EBP-28]
00461830 . 83C4 0C    ADD ESP,0C
00461833 . 66:40      INC AX
00461835 . 0F80 EB040000 JO winrsm.00461D26
0046183B . 8945 D8    MOV DWORD PTR SS:[EBP-28],EAX
0046183E . ^E9 91FFFF  JMP winrsm.004616D4 ; <== Continue Loop
```

/*/*/*/ - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : IBD6Rg\dr**

III – KeyGen :

- /Section 1/- ***CharUpper(User)*** .
/Section 2/- Tính giá trị theo công thức : ***Value = (i * 6) + reaName[i-1] - 0xF***
/Section 3/- So sánh và xác định giá trị chính xác .

IV – SourceCode (VC++) :

```
char reaName[64]={0};  
char reaSerial[64]={0};  
int LenUser=0;  
int i=0;  
int Value=0;  
  
LenUser=GetDlgItemText(IDC_Name,reaName,64);  
if (LenUser < 1)  
{  
    MessageBox("----- Your name atleast 1 chart -----","Hey !!"  
Please input your name again !! ");  
}  
else  
{  
    CharUpper(reName);  
  
    i=1;  
    while ( i < LenUser + 1 )
```

```

    {
        Value = (i * 6) + reaName[i-1] - 0xF;

        if ( Value < 0x21 )
        {
            reaSerial[i-1] = 0x65;
        }
        else if ( Value > 0x7E )
        {
            reaSerial[i-1] = 0x2A;
        }
        else
        {
            reaSerial[i-1] = Value;
        }
        i++;
    }
    SetDlgItemText(IDC_Serial, reaSerial);
}

```

V – End of Tut :

- Finished – **14/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.rf1.net/software/player
Production :	RF1 Systems
SoftWare :	RF1 Player 1.4.0 beta
Copyright by :	Copyright © 2002 RF1 Systems. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

RF1 Player 1.4.0 beta

Audio and video player, playlist editor, mp3 tags editor. Supported formats including MP3, WAV, MID, WMA, CDA, AVI, MPG, ASF, WMV, WM, SND, AU, AIFF, M1V, MP2. The list and played position saved automatically. Also supports m3u list manipulations: load, save, modify. 5 displaying modes: micro, brief, medium, full and background.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**.
- Nhập thử User và Fake Serial, ta nhận được thông báo "Invalid code" Tìm thông báo này và ta gặp tại địa chỉ :

00505C4B > \B8 E85E5000 MOV EAX,Player.00505EE8 ; ASCII "Invalid code"

- Dò ngược lên trên và ta đặt BreakPoint tại :
 00505B42 . E8 E56DF5FF CALL Player.0045C92C ; <== Set BreakPoint here

II – Cracking :

- Load và chạy chương trình với User và Fake Serial, chương trình dừng lại tại đây. Trace tiếp đoạn ta đến :

00505B52 . E8 91F0FEFF CALL Player.004F4BE8 ; <== Trace into here

- Dùng F7 trace into ta đến quá trình mã hoá . Quá trình mã hoá này bao gồm 4 giai đoạn .

- Giai đoạn thứ nhất . Quá trình này được tính toán dựa thuật toán tính toán tinh trân . Với U[i] là tham số của phép nhân trong thuật toán này (tham khảo cách tính thuật toán này trong bài viết về soft *Cyclone Internet History Killer Pro 3.06*):

```

004F4C22 |. B8 CC92CEBB MOV EAX,BBCE92CC ; <== Default Value : dV_01
004F4C27 |. BA 2B010000 MOV EDX,12B ; <== Default Value : dV_02
004F4C2C |. E8 CF0DF1FF CALL Player.00405A00
004F4C31 |. 8945 F0 MOV [LOCAL.4],EAX
004F4C34 |. 8955 F4 MOV [LOCAL.3],EDX
004F4C37 |. 8B45 FC MOV EAX,[LOCAL.1]
004F4C3A |. E8 2100F1FF CALL Player.00404C60
004F4C3F |. 8BF0 MOV ESI,EAX
004F4C41 |. 4E DEC ESI
004F4C42 |. 85F6 TEST ES,ESI
004F4C44 |. 7C 27 JL SHORT Player.004F4C6D
004F4C46 |. 46 INC ESI
004F4C47 |. 33DB XOR EBX,EBX
004F4C49 |> 8B45 FC /MOV EAX,[LOCAL.1] ; <== U
004F4C4C |. 8A0418 |MOV AL,BYTE PTR DS:[EAX+EBX] ; <== U[i]
004F4C4F |. 25 FF000000 |AND EAX,0FF ; <== U[i]
004F4C54 |. 33D2 |XOR EDX,EDX
004F4C56 |. 52 |PUSH EDX
004F4C57 |. 50 |PUSH EAX
004F4C58 |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== Value
004F4C5B |. 8B55 F4 |MOV EDX,[LOCAL.3] ; <== Over
004F4C5E |. E8 9D0DF1FF |CALL Player.00405A00 ; <== OverCalculation
----- Trace Into -----

```

```

00405A0B |. 8B4424 04 MOV EAX,DWORD PTR SS:[ESP+4] ; <== Over
00405A0F |. F76424 0C MUL DWORD PTR SS:[ESP+C] ; <== Over = Over * Temp
00405A13 |. 01C1 ADD ECX,EAX ; <== Over
00405A15 |. 8B0424 MOV EAX,DWORD PTR SS:[ESP] ; <== Value
00405A18 |. F76424 0C MUL DWORD PTR SS:[ESP+C] ; <== Value = Value * U[i]
00405A1C |. 01CA ADD EDX,ECX ; <== Over = Over + OverValue
----- Trace Into -----

```

```

004F4C63 |. 8945 F0 |MOV [LOCAL.4],EAX ; <== Value
004F4C66 |. 8955 F4 |MOV [LOCAL.3],EDX ; <== Over
004F4C69 |. 43 |INC EBX
004F4C6A |. 4E |DEC ESI
004F4C6B |.^ 75 DC \JNZ SHORT Player.004F4C49
----- Trace Into -----

```

- Giai đoạn thứ hai chương trình cũng tiếp tục sử dụng thuật toán này thêm một lần nữa, với tham số đầu vào là kết quả của giai đoạn thứ nhất và tham số nhân là U[i] được biết đổi chút ít (**Temp_02 = U[i] + 0x7**)

004F4C7F |> /8B45 FC /MOV EAX,[LOCAL.1] ; <== U

```

004F4C82 |. |E8 D9FFF0FF |CALL Player.00404C60 ; <== Len.U
004F4C87 |. |2BC3 |SUB EAX,EBX
004F4C89 |. |8B55 FC |MOV EDX,[LOCAL.1]
004F4C8C |. |0FB64402 FF |MOVZX EAX,BYTE PTR DS:[EDX+EAX-1] ; <== U[Len.U - 1 - i]
004F4C91 |. |F7EB |IMUL EBX ; <== Temp_01 = U[Len.U-1-i] * i
004F4C93 |. |99 |CDQ
004F4C94 |. |52 |PUSH EDX
004F4C95 |. |50 |PUSH EAX
004F4C96 |. |8B45 FC |MOV EAX,[LOCAL.1] ; <== U
004F4C99 |. |0FB60418 |MOVZX EAX,BYTE PTR DS:[EAX+EBX] ; <== U[i]
004F4C9D |. |83C0 07 |ADD EAX,7 ; <== Temp_02 = U[i] + 0x7
004F4CA0 |. |33D2 |XOR EDX,EDX
004F4CA2 |. |52 |PUSH EDX
004F4CA3 |. |50 |PUSH EAX ; <== Temp_02
004F4CA4 |. |8B45 F0 |MOV EAX,[LOCAL.4] ; <== Value
004F4CA7 |. |8B55 F4 |MOV EDX,[LOCAL.3] ; <== Over
004F4CAA |. |E8 510DF1FF |CALL Player.00405A00 ; <== OverCalculation
004F4CAF |. |030424 |ADD EAX,DWORD PTR SS:[ESP] ; <== Value = Value + Temp_01
004F4CB2 |. |135424 04 |ADC EDX,DWORD PTR SS:[ESP+4] ; <== Over
004F4CB6 |. |83C4 08 |ADD ESP,8
004F4CB9 |. |8945 F0 |MOV [LOCAL.4],EAX ; <== Value
004F4CBC |. |8955 F4 |MOV [LOCAL.3],EDX ; <== Over
004F4CBF |. |43 |INC EBX ; <== i++
004F4CC0 |. |4E |DEC ESI ; <== Len = Len.U - 1
004F4CC1 |.^\75 BC |JNZ SHORT Player.004F4C7F ; <== Loop until Len = 0x0

```

- Giai đoạn thứ ba : đây là một giai đoạn đặc biệt, rất dễ bị bỏ qua . Giai đoạn này dựa vào đặt điểm lệnh TEST trong ASM . Tuỳ thuộc vào giá trị trả về của lệnh này mà kết đầu vào của giai đoạn cuối có khác biệt so với kết quả đầu ra của giai đoạn thứ hai hay không :

```

004F4CC3 |> |8B45 F0 |MOV EAX,[LOCAL.4] ; <== Value
004F4CC6 |. |8B55 F4 |MOV EDX,[LOCAL.3] ; <== Over
004F4CC9 |. |85D2 |TEST EDX,EDX
004F4CCB |. |7D 07 |JGE SHORT Player.004F4CD4
004F4CCD |. |F7D8 |NEG EAX
004F4CCF |. |83D2 00 |ADC EDX,0
004F4CD2 |. |F7DA |NEG EDX
004F4CD4 |> |8945 F0 |MOV [LOCAL.4],EAX ; <== Value
004F4CD7 |. |8955 F4 |MOV [LOCAL.3],EDX ; <== Over

```

- Giai đoạn thứ tư : đây cũng là giai đoạn tạo chuỗi Serial . Quá trình này tập trung vào đoạn sử dụng RCL :

```

004F4CEE |. |33DB |XOR EBX,EBX ; <== i = 0x0
004F4CF0 |> |85DB |/TEST EBX,EBX ; <== i
004F4CF2 |. |7E 1E |JLE SHORT Player.004F4D12
004F4CF4 |. |8BC3 |MOV EAX,EBX
004F4CF6 |. |25 03000080 |AND EAX,80000003
004F4CFB |. |79 05 |JNS SHORT Player.004F4D02
004F4CFD |. |48 |DEC EAX
004F4CFE |. |83C8 FC |OR EAX,FFFFFFFC
004F4D01 |. |40 |INC EAX
004F4D02 |> |85C0 |TEST EAX,EAX
004F4D04 |. |75 0C |JNZ SHORT Player.004F4D12
004F4D06 |. |8BC7 |MOV EAX,EDI
004F4D08 |. |BA C84D4F00 |MOV EDX,Player.004F4DC8

```

```

004F4D0D |. E8 56FFF0FF |CALL Player.00404C68 ; <== “_“
004F4D12 |> 6A 00 |PUSH 0
004F4D14 |. 6A 3E |PUSH 3E ; <== Check Value : cV
004F4D16 |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== Value
004F4D19 |. 8B55 F4 |MOV EDX,[LOCAL.3] ; <== Over
004F4D1C |. E8 7F0DF1FF |CALL Player.00405AA0 ; <== RCLCalculation
----- Trace Into -----
00405AD5 |. B9 40000000 MOV ECX,40 ; <== Numbers of Loop : nL
00405ADA |. 57 PUSH EDI
00405ADB |. 31FF XOR EDI,EDI ; <== Cal_02 = 0x0
00405ADD |. 31F6 XOR ESI,ESI ; <== Cal_02 = 0x0
00405ADF |> D1E0 /SHL EAX,1 ; <== Temp_01 = Temp_01 * 2
00405AE1 |. D1D2 |RCL EDX,1 ; <== Temp_02 = Temp_02 rcl 0x1
00405AE3 |. D1D6 |RCL ESI,1 ; <== Cal_01 = Cal_01 rcl 0x1
00405AE5 |. D1D7 |RCL EDI,1 ; <== Cal_02 = Cal_02 rcl 0x1
00405AE7 |. 39EF |CMP EDI,EBP
00405AE9 |. 72 0B JB SHORT Player.00405AF6
00405AEB |. 77 04 JA SHORT Player.00405AF1
00405AED |. 39DE |CMP ESI,EBX ; <== cmp(cV, Cal_01)
00405AEF |. 72 05 JB SHORT Player.00405AF6
00405AF1 |> 29DE |SUB ESI,EBX ; <== Cal_01 = Cal_01 - cV
00405AF3 |. 19EF |SBB EDI,EBP
00405AF5 |. 40 |INC EAX
00405AF6 |>^ E2 E7 \LOOPD SHORT Player.00405ADF
----- NOTE -----

```

Ở đây cho ta ba giá trị :

EAX	:	Value
EDX	:	Over
ESI	:	Chart (tạo chuỗi Serial)

----- NOTE -----

----- Trace Into -----

```

004F4D21 |. 8945 EC |MOV [LOCAL.5],EAX ; <== TempChar : tChr
004F4D24 |. 6A 00 |PUSH 0
004F4D26 |. 6A 3E |PUSH 3E ; <== Check Value : cV
004F4D28 |. 8B45 F0 |MOV EAX,[LOCAL.4] ; <== Value
004F4D2B |. 8B55 F4 |MOV EDX,[LOCAL.3] ; <== Over
004F4D2E |. E8 F10CF1FF |CALL Player.00405A24 ; <== RCLCalculation
004F4D33 |. 8945 F0 |MOV [LOCAL.4],EAX ; <== Value
004F4D36 |. 8955 F4 |MOV [LOCAL.3],EDX ; <== Over
004F4D39 |. 8B45 EC |MOV EAX,[LOCAL.5] ; <== tChr
004F4D3C |. 83E8 0A |SUB EAX,0A ; Switch (cases 0..3D)
004F4D3F |. 72 0C JB SHORT Player.004F4D4D
004F4D41 |. 83E8 1A |SUB EAX,1A
004F4D44 |. 72 14 JB SHORT Player.004F4D5A
004F4D46 |. 83E8 1A |SUB EAX,1A
004F4D49 |. 72 22 JB SHORT Player.004F4D6D
004F4D4B |. EB 31 JMP SHORT Player.004F4D7E
004F4D4D |> 8D55 E8 |LEA EDX,[LOCAL.6] ; Cases 0,1,2,3,4,5,6,7,8,9 of switch
004F4D3C
004F4D50 |. 8B45 EC |MOV EAX,[LOCAL.5]
004F4D53 |. E8 9045F1FF |CALL Player.004092E8 ; <== tChr = tChr + 0x30
004F4D58 |. EB 24 JMP SHORT Player.004F4D7E

```

```

004F4D5A > 8D45 E8 |LEA EAX,[LOCAL.6] ; Cases
A,B,C,D,E,F,10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23 of switch 004F4D3C
004F4D5D |. 8B55 EC |MOV EDX,[LOCAL.5]
004F4D60 |. 83C2 61 |ADD EDX,61 ; <== tChr = tChr + 0x61
004F4D63 |. 83EA 0A |SUB EDX,0A ; <== tChr = tChr - 0xA
004F4D66 |. E8 0DFEF0FF |CALL Player.00404B78
004F4D6B |. EB 11 |JMP SHORT Player.004F4D7E
004F4D6D > 8D45 E8 |LEA EAX,[LOCAL.6] ; Cases
24,25,26,27,28,29,2A,2B,2C,2D,2E,2F,30,31,32,33,34,35,36,37,38,39,3A,3B,3C,3D of switch
004F4D3C
004F4D70 |. 8B55 EC |MOV EDX,[LOCAL.5]
004F4D73 |. 83C2 41 |ADD EDX,41 ; <== tChr = tChr + 0x41
004F4D76 |. 83EA 24 |SUB EDX,24 ; <== tChr = tChr - 0x24
004F4D79 |. E8 FAFDF0FF |CALL Player.00404B78
004F4D7E > 8BC7 |MOV EAX,EDI ; Default case of switch 004F4D3C
004F4D80 |. 8B55 E8 |MOV EDX,[LOCAL.6]
004F4D83 |. E8 E0FEF0FF |CALL Player.00404C68
004F4D88 |. 43 |INC EBX ; <== i++
004F4D89 |. 4E |DEC ESI ; <== Number of Loop --
004F4D8A |.^ 0F85 60FFFFFF \JNZ Player.004F4CF0 ; <== Loop 8 times
- Giai đoạn này sử dụng hai lần quá trình tính toán RCLCalculation . Với các giá trị đầu ra khác nhau .
Nhưng thực chất, kết quả của hai quá trình này hoàn toàn y như nhau . Nên trong quá trình tạo KeyGen ta có thể bỏ qua một quá trình, với điều kiện cả ba thông số đầu ra đều được sử dụng ở quá trình này .
- Ở đây ta cũng chú ý đến đoạn đầu của vòng lặp này . Dựa vào lệnh AND EAX,80000003, với EAX chính là số lần lặp ( i ) thì giá trị trả về cho EAX luôn nằm trong khoảng (0-3) . Hay nói cách khác, giá trị S[4] luôn là “-“

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : 88Pl-WRe7

III – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};

```

```

int LenUser=0;
int i=0,j=0;
int Over=0, Value=0, Temp=0, Chart=0;

```

```

LenUser=GetDlgItemText(IDC_Name,reName,64);
if (LenUser < 1)
{
    MessageBox("----- Your name atleast 1 chart -----","Hey !! Please
input your name again !! ");
}
else
{
    i=0;    Over = 0x12B;      Value = 0xBBCE92CC;
    while ( i < LenUser )
    {
        Temp = reName[i] & 0xFF;
        OverCalculation(Over, Value, Temp);
        _asm

```

```

{
    MOV Over, EDX
    MOV Value, EAX
}
i++;
}
i=0;
while ( i < LenUser )
{
    Temp = (reaName[i] & 0xFF) + 0x7;
    OverCalculation(Over, Value, Temp);
    _asm
    {
        MOV Over, EDX
        MOV Value, EAX
    }
    Value = Value + ((reaName[LenUser -1 - i] & 0xFF) * i);
    i++;
}

_asm
{
    MOV EAX, Value
    MOV EDX, Over
    TEST EDX,EDX
    JGE Check
    NEG EAX
    ADC EDX,0
    NEG EDX
}

Check:
    MOV Over, EDX
    MOV Value, EAX
}

i=0;    j=0;
while ( i < 8 )
{
    if ( j == 4 )
    {
        reaSerial[j] = 0x2D;
        j++;
    }
    else
    {
        j=j;
    }
    RCLCalculation(Over,Value);
    _asm
    {
        MOV Chart, ECX
        MOV Over, EDX
        MOV Value, EAX
    }
}

```

```

if ((Chart - 0xA) < 0x0)
{
    reaSerial[j] = Chart + 0x30;
}

else if ((Chart - 0x24) < 0x0)
{
    reaSerial[j] = Chart + 0x61 - 0xA;
}

else if ((Chart - 0x3E) < 0x0)
{
    reaSerial[j] = Chart + 0x41 - 0x24;
}

else
{
    reaSerial[j] = Chart;
}

j++;
i++;
}

SetDlgItemText(IDC_Serial, reaSerial);
}

```

IV – End of Tut :

- Finished – ***16/07/2004***

Reverse Engineering Association

SoftWare

Homepage :	http://www.doease.com
Production :	Doease Software,Inc.
SoftWare :	Smart Video Converter 1.5.3
Copyright by :	Copyright © 2004 Doease Software,Inc. All Rights Reserved.
Type :	Email / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Smart Video Converter 1.5.3

Smart Video Converter is a powerful video converter,joiner,splitter.Lots of video formats supported,includes MPEG1(VCD),MPEG2(SVCD),MPEG2 (DVD),AVI(Divx,Xvid,MPEG4) and WMV/ASF.Furthermore,Smart Video Converter supports audio extraction and image extraction from video files.Small size,extreme fast speed,superb features and clear user interface,whatever you are a veteran or a beginner, you will feel it's developed for you!

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0.**
- Nhập thử Email và Fake Serial, ta nhận được thông báo "**Invalid Reg-Email or Reg-Code.**" Ta tìm được đoạn CODE này ở hai địa chỉ :

```
00407CD5 . 68 E0824100 PUSH VideoCon.004182E0 ; ASCII "Invalid Reg-Email or Reg-Code."
00407D16 . 68 E0824100 PUSH VideoCon.004182E0 ; ASCII "Invalid Reg-Email or Reg-Code."
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
00407AA5 . E8 FE8C0000 CALL <JMP.&MFC42.#540> ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với Email và Fake Serial, chương trình dừng lại tại điểm đặt BP. Đầu tiên chương trình kiểm tra tính hợp lệ của Email nhập :

```
00407AC2 . E8 098F0000 CALL <JMP.&MFC42.#3874> ; <== Get Length of Email
00407AC7 . 8B4C24 10 MOV ECX,DWORD PTR SS:[ESP+10] ; <== Email
00407ACB . 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8] ; <== Len.E
00407ACE . 83F8 40 CMP EAX,40 ; <== If ( Len.E <= 0x40 )
00407AD1 . 0F8F 35020000 JG VideoCon.00407D0C ; <== and
00407AD7 . 83F8 03 CMP EAX,3 ; <== If ( Len.E >= 0x3 )
00407ADA . 0F8C 2C020000 JL VideoCon.00407D0C ; <== Next check
00407AE0 . 6A 40 PUSH 40
00407AE2 . 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14] ; <== Email
00407AE6 . E8 4D900000 CALL <JMP.&MFC42.#2763> ; <== Check validity of Email
00407AEB . 83CB FF OR EBX,FFFFFFFF ; <== Email accept or not ?
00407AEE . 3BC3 CMP EAX,EBX ; <== if Accept
00407AF0 . 0F84 19020000 JE VideoCon.00407D0F ; <== Go to next check
```

- Sau đó chương trình kiểm tra chiều dài chuỗi Serial nhập :

```
00407B39 . E8 928E0000 CALL <JMP.&MFC42.#3874> ; <== Get Length of Serial
00407B3E . 8B4C24 14 MOV ECX,DWORD PTR SS:[ESP+14] ; <== Serial
00407B42 . 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8] ; <== Len.S
00407B45 . 85C0 TEST EAX,EAX ; <== Serial must be input
00407B47 . 0F84 81010000 JE VideoCon.00407CCE
00407B4D . 83F8 1D CMP EAX,1D ; <== Serial must be 29 charts
00407B50 . 0F85 78010000 JNZ VideoCon.00407CCE
00407B56 . 50 PUSH EAX ; / maxlen
00407B57 . 8D5424 20 LEA EDX,DWORD PTR SS:[ESP+20] ; |
00407B5B . 51 PUSH ECX ; |src
00407B5C . 52 PUSH EDX ; |dest
00407B5D . FF15 30344100 CALL DWORD PTR DS:<&MSVCRT.strncpy> ; \strncpy
00407B63 . 8D4424 28 LEA EAX,DWORD PTR SS:[ESP+28] ; <== Serial
00407B67 . 8D4C24 48 LEA ECX,DWORD PTR SS:[ESP+48] ; <== Email
00407B6B . 50 PUSH EAX ; <== Serial
00407B6C . 51 PUSH ECX ; <== Email
00407B6D . C64424 4D 00 MOV BYTE PTR SS:[ESP+4D],0
00407B72 . E8 F9FDFFFF CALL VideoCon.00407970 ; <== Trace into here
```

- Dùng F7 để trace into ta đến quá trình mã hoá đầu tiên :

```
0040798C >/0FBE142E /MOVSX EDX,BYTE PTR DS:[ESI+EBP] ; <== E[i]
00407990 .| 8BC2 |MOV EAX,EDX ; <== E[i]
00407992 .| BF 6F000000 |MOV EDI,6F ; <== Default Value : dV
00407997 .| C1E0 04 |SHL EAX,4 ; <== Temp = E[i] * 0x10
0040799A .| 03C2 |ADD EAX,EDX ; <== Temp = Temp + E[i]
0040799C .| 99 |CDQ
0040799D .| F7FF |IDIV EDI ; <== Temp = Temp % dV
```

```

0040799F |. |03DA    |ADD EBX,EDX          ; <== Value = Value + Temp
004079A1 |. |46      |INC ESI           ; <== i++
004079A2 |. |3BF1    |CMP ESI,ECX        ; <== while ( i < Len.E )
004079A4 |.^\7C E6   |JL SHORT VideoCon.0040798C ; <== Continue Loop
004079A6 |> 8BC3    |MOV EAX,EBX        ; <== Value
004079A8 |. 8B6C24 18 |MOV EBP,DWORD PTR SS:[ESP+18] ; <== Serial
004079AC |. 0FAFC3   |IMUL EAX,EBX       ; <== Value = Value * Value
- Quá trình mã hoá thứ hai đồng thời bao hàm luôn quá trình so sánh diễn ra ngay sau đó :
004079AF |. 33F6    |XOR ESI,ESI        ; <== i = 0x0
004079B1 |. 8BD8    |MOV EBX,EAX        ; <== Value_02 = Value
004079B3 |. 8BF8    |MOV EDI,EAX        ; <== Value_01 = Value
004079B5 |> 8BC6    |/MOV EAX,ESI       ; <== i
004079B7 |. B9 06000000 |MOV ECX,6         ; <== Default Value : dV
004079BC |. 99      |CDQ               ; <== Temp = i % 6
004079BD |. F7F9    |IDIV ECX          ; <== if ( Temp != 0x5 )
004079BF |. 83FA 05  |CMP EDX,5         ; <== then
004079C2 |. 74 25   |JE SHORT VideoCon.004079E9 ; <== Value_01
004079C4 |. 8BC7    |MOV EAX,EDI          ; <== Default Value : dV
004079C6 |. B9 05000000 |MOV ECX,5         ; <== CDQ
004079CC |. F7F9    |IDIV ECX          ; <== Temp = Value_01 %
0x5
004079CE |. 8B1495 AC8241>|MOV EDX,DWORD PTR DS:[EDX*4+4182AC] ; <== to Function
check
004079D5 |. 8915 EC894100 |MOV DWORD PTR DS:[4189EC],EDX ; <== Temp
004079DB |. 8A042E    |MOV AL,BYTE PTR DS:[ESI+EBP] ; <== S[i]
004079DE |. 50      |PUSH EAX          ; <== S[i]
004079DF |. 53      |PUSH EBX          ; <== Value_02
004079E0 |. FFD2    |CALL EDX          ; <== Get real Serial : rS[i]

```

===== Trace Into =====

===== NOTE =====

Thực tế ở đây ta chú ý đến phần dư của phép chia này. Do giá trị “**Value_01**” được chia cho “5” nên phần dư không bao giờ quá “4”. Nên giá trị của “**EDX**” chỉ có “5”, căn cứ và giá trị này mà lệnh “**CALL EDX**” sẽ chuyển đến FUNCTION kiểm tra tương ứng; hay nói cách khác, tuỳ thuộc vào giá trị của phần dư mà chương trình sẽ chuyển đến FUNCTION kiểm tra tương ứng (có “5” FUNCTION cả thảy)

===== NOTE =====

===== Function 0 =====

```

00407880 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
00407884 . B8 79787878 MOV EAX,78787879          ; <== dV
00407889 . F7E9    IMUL ECX           ; <== Temp = Value_02 * dV
0040788B . 8BC2    MOV EAX,EDX        ; <== Over
0040788D . C1F8 03 SAR EAX,3         ; <== Over = Over sar 0x3
00407890 . 8BC8    MOV ECX,EAX        ; <== Temp = Over
00407892 . C1E9 1F SHR ECX,1F        ; <== Temp = Temp / 0x80000000
00407895 . 03C1    ADD EAX,ECX        ; <== Over = Over + Temp
00407897 . B9 0A000000 MOV ECX,0A        ; <== dV
0040789C . 99      CDQ               ; <== Over = Over % dV
0040789D . F7F9    IDIV ECX          ; <== S[i]
0040789F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== rS[i]
004078A3 . 33C0    XOR EAX,EAX        ; <== rS[i] = Over + 0x30
004078A5 . 80C2 30 ADD DL,30          ; <== if ( rS[i] = S[i] )
004078A8 . 3AD1    CMP DL,CL          ; <== if ( rS[i] = S[i] )

```

```

004078AA . 0F94C0      SETE AL          ; <== AL = 0x1
----- Funtion I -----
004078B0 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
004078B4 . B8 C94216B2 MOV EAX,B21642C9          ; <== dV
004078B9 . F7E9      IMUL ECX          ; <== Temp = Value_02 * dV
004078BB . 8BC2      MOV EAX,EDX          ; <== Over
004078BD . 03C1      ADD EAX,ECX          ; <== Over = Over + Temp
004078BF . C1F8 04   SAR EAX,4          ; <== Over = Over sar 0x4
004078C2 . 8BC8      MOV ECX,EAX          ; <== Temp = Over
004078C4 . C1E9 1F   SHR ECX,1F          ; <== Temp = Temp / 0x80000000
004078C7 . 03C1      ADD EAX,ECX          ; <== Over = Over + Temp
004078C9 . B9 0A000000 MOV ECX,0A          ; <== dV
004078CE . 99        CDQ              ; <== Over = Over % dV
004078CF . F7F9      IDIV ECX          ; <== S[i]
004078D1 . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== S[i]
004078D5 . 33C0      XOR EAX,EAX          ; <== Over
004078D7 . 80C2 30   ADD DL,30          ; <== rS[i] = Over + 0x30
004078DA . 3AD1      CMP DL,CL          ; <== if ( rS[i] = S[i] )
004078DC . 0F94C0      SETE AL          ; <== AL = 0x1
----- Funtion II -----
004078E0 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
004078E4 . B8 4FECC44E MOV EAX,4EC4EC4F          ; <== dV
004078E9 . F7E9      IMUL ECX          ; <== Temp = Value_02 * dV
004078EB . 8BC2      MOV EAX,EDX          ; <== Over
004078ED . C1F8 02   SAR EAX,2          ; <== Over = Over sar 0x2
004078F0 . 8BC8      MOV ECX,EAX          ; <== Temp = Over
004078F2 . C1E9 1F   SHR ECX,1F          ; <== Temp = Temp / 0x80000000
004078F5 . 03C1      ADD EAX,ECX          ; <== Over = Over + Temp
004078F7 . B9 1A000000 MOV ECX,1A          ; <== dV
004078FC . 99        CDQ              ; <== Over = Over % dV
004078FD . F7F9      IDIV ECX          ; <== S[i]
004078FF . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== S[i]
00407903 . 33C0      XOR EAX,EAX          ; <== Over
00407905 . 80C2 41   ADD DL,41          ; <== rS[i] = Over + 0x41
00407908 . 3AD1      CMP DL,CL          ; <== if ( rS[i] = S[i] )
0040790A . 0F94C0      SETE AL          ; <== AL = 0x1
----- Funtion III -----
00407910 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
00407914 . B8 93244992 MOV EAX,92492493          ; <== dV
00407919 . F7E9      IMUL ECX          ; <== Temp = Value_02 * dV
0040791B . 8BC2      MOV EAX,EDX          ; <== Over
0040791D . 03C1      ADD EAX,ECX          ; <== Over = Over + Temp
0040791F . C1F8 02   SAR EAX,2          ; <== Over = Over sar 0x2
00407922 . 8BC8      MOV ECX,EAX          ; <== Temp = Over
00407924 . C1E9 1F   SHR ECX,1F          ; <== Temp = Temp / 0x80000000
00407927 . 03C1      ADD EAX,ECX          ; <== Over = Over + Temp
00407929 . B9 1A000000 MOV ECX,1A          ; <== dV
0040792E . 99        CDQ              ; <== Over = Over % dV
0040792F . F7F9      IDIV ECX          ; <== S[i]
00407931 . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== S[i]
00407935 . 33C0      XOR EAX,EAX          ; <== Over

```

```

00407937 . 80C2 41 ADD DL,41 ; <== rS[i] = Over + 0x41
0040793A . 3AD1 CMP DL,CL ; <== if ( rS[i] == S[i] )
0040793C . 0F94C0 SETE AL ; <== AL = 0x1

===== Funtion IV =====
00407940 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
00407944 . B8 F31ACA6B MOV EAX,6BCA1AF3 ; <== dV
00407949 . F7E9 IMUL ECX ; <== Temp = Value_02 * dV
0040794B . 8BC2 MOV EAX,EDX ; <== Over
0040794D . C1F8 03 SAR EAX,3 ; <== Over = Over sar 0x3
00407950 . 8BC8 MOV ECX,EAX ; <== Temp = Over
00407952 . C1E9 1F SHR ECX,1F ; <== Temp = Temp / 0x80000000
00407955 . 03C1 ADD EAX,ECX ; <== Over = Over + Temp
00407957 . B9 1A000000 MOV ECX,1A ; <== dV
0040795C . 99 CDQ
0040795D . F7F9 IDIV ECX ; <== Over = Over % dV
0040795F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== S[i]
00407963 . 33C0 XOR EAX,EAX
00407965 . 80C2 41 ADD DL,41 ; <== rS[i] = Over + 0x41
00407968 . 3AD1 CMP DL,CL ; <== if ( rS[i] == S[i] )
0040796A . 0F94C0 SETE AL ; <== AL = 0x1

===== Trace Into =====
004079E2 |. 83C4 08 |ADD ESP,8 ; <== if ( AL == 0x1 )
004079E5 |. 85C0 |TEST EAX,EAX ; <== then
004079E7 74 16 JE SHORT VideoCon.004079FF ; <== Continue check
004079E9 |> 46 |INC ESI ; <== i++
004079EA |. 83EF 0B |SUB EDI,0B ; <== Value_01 = Value_01 - 0xB
004079ED |. 83EB 11 |SUB EBX,11 ; <== Value_02 = Value_02 - 0x11
004079F0 |. 83FE 1D |CMP ESI,1D ; <== while ( i < 0x1D )
004079F3 |.^ 7C C0 |JL SHORT VideoCon.004079B5 ; <== Continue loop

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErStEaM@rea.net

Serial : 88TUR-3PIL1-KWE73-KY38B-R03XM

III – KeyGen :

- /Section 1/- Tính **Value** = **Value** + (((*reaEmail[i]* * 0x10) + *reaEmail[i]*) % 0x6F).
- /Section 2/- Tính **Value** = **Value** * **Value**
- /Section 3/- Tương ứng với mỗi giá trị của phép chia **Value** % 0x5. Ta có các đoạn CODE mã hoá chuỗi Serial khác nhau .

IV – End of Tut :

- Finished – 21/07/2004

Reverse Engineering Association SoftWare

Homepage : <http://www.snooper.se>
 Production : Peter Skarin
 SoftWare : Snooper 1.35.28

Copyright by : Copyright © 2004 Peter Skarin. All Rights Reserved.
 Type : Name / Serial
 Packed : UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo
 Language : Borland Delphi 6.0 - 7.0
 Crack Tool : OllyDbg 1.09d, PEiD 0.92, kWdsm 10
 Unpack : Manual
 Request : Correct Serial / KeyGen

Snooper 1.35.28

Snooper is an advanced sound activated recorder. It will start to record when sound is detected on the sound card and pauses as soon as there are no sounds automatically. Encrypted recordings can be sent by e-mail after a recording is finished. You can schedule the starting and ending times for any recording you plan to make using the recording scheduler function. Recordings are saved in compact MP3 format. Stealth mode will hide the program to record covertly.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo**. Sau khi UnPACK kiểm tra lại biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**.
- Nhập thử User và Fake Serial (tuy yêu cầu là EMAIL nhưng chương trình không kiểm tra tính hợp lý của EMAIL nên coi như là USER), ta không nhận được bất kỳ thông báo nào . Như vậy ta không thể tìm chuỗi thông báo đúng hay sai trong trường hợp này . Nhưng ở đây ta chú ý đến một điều . Sau khi nhấn OK để xác nhận DATA thì chương trình có một khoảng thời gian dừng. Điều này có nghĩa chương trình đã sử dụng hàm API **KERNEL32.Sleep** .
- Đặt BreakPoint tại tất cả các hàm **KERNEL32.Sleep** có trong chương trình bằng cách click chuột phải, chọn **Search for → All intermodular Calls** . Olly chuyển đến cửa sổ **Intermodular Calls**, tìm đến hàm này và click chuột phải chọn **Set BreakPoint all every call to Sleep** .

II – Cracking :

- Load chương trình lên, nhập User và Fake Serial, chương trình dừng lại tại :

```
004BAE75 E8 4E35F5FF CALL <JMP.&KERNEL32.Sleep> ; <== We here
```

- Quá trình mã hoá chuỗi nằm ở lệnh CALL kế tiếp :

```
004BAE82 E8 89060000 CALL snpr_exe.004BB510 ; <== Trace Into Here
```

- Sau khi dùng F7 trace into và trace thêm một đoạn ta đến :

```
004BB599 E8 F2FFFFFF CALL snpr_exe.004BA590 ; <== Trace Into Here
```

- Tiếp tục :

```
004BB599 E8 F2FFFFFF CALL snpr_exe.004BA590 ; <== Trace Into Here
```

- Sau khi vào trace into vào đây, ta đến đoạn CODE tạo chuỗi đầu tiên :

```
004BA61E 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
004BA621 8378 30 01 CMP DWORD PTR DS:[EAX+30],1
004BA625 74 06 JE SHORT snpr_exe.004BA62D
```

```
004BA627 90 NOP
004BA628 90 NOP
004BA629 90 NOP
004BA62A 90 NOP
004BA62B 90 NOP
004BA62C 90 NOP
```

```
004BA62D 8D45 DC LEA EAX,DWORD PTR SS:[EBP-24]
004BA630 50 PUSH EAX
004BA631 8D45 D8 LEA EAX,DWORD PTR SS:[EBP-28]
004BA634 50 PUSH EAX
004BA635 8B4D E8 MOV ECX,DWORD PTR SS:[EBP-18] ; <== i = 1
```

```

004BA638 8B55 F8      MOV EDX,DWORD PTR SS:[EBP-8]           ; <== User
004BA63B 8B45 FC      MOV EAX,DWORD PTR SS:[EBP-4]
004BA63E E8 89000000  CALL snpr_exe.004BA6CC                 ; <== Trace Into here
004BA643 8B45 D8      MOV EAX,DWORD PTR SS:[EBP-28]          ; <== TempString : char[]
004BA646 B9 01000000  MOV ECX,1                                ; <== j = 1
004BA64B BA 02000000  MOV EDX,2                                ; <== Len(char) = 2
004BA650 E8 3FA7F4FF  CALL snpr_exe.00404D94                ; <== Temp = char[1]
004BA655 8B55 DC      MOV EDX,DWORD PTR SS:[EBP-24]
004BA658 8B45 F4      MOV EAX,DWORD PTR SS:[EBP-C]
004BA65B E8 DCA4F4FF  CALL snpr_exe.00404B3C
004BA660 8B45 F4      MOV EAX,DWORD PTR SS:[EBP-C]
004BA663 E8 8CC7F4FF  CALL <JMP.&KERNEL32.GetTickCount>
004BA668 8945 EC      MOV DWORD PTR SS:[EBP-14],EAX
004BA66B 8B45 EC      MOV EAX,DWORD PTR SS:[EBP-14]
004BA66E 2B45 F0      SUB EAX,DWORD PTR SS:[EBP-10]
004BA671 3D F4010000  CMP EAX,1F4
004BA676 7E 08        JLE SHORT snpr_exe.004BA680
004BA678 8B45 F4      MOV EAX,DWORD PTR SS:[EBP-C]
004BA67B E8 F4A1F4FF  CALL snpr_exe.00404874
004BA680 FF45 E8      INC DWORD PTR SS:[EBP-18]           ; <== i++
004BA683 FF4D E4      DEC DWORD PTR SS:[EBP-1C]           ; <== Len.U --
004BA686 ^ 75 96      JNZ SHORT snpr_exe.004BA61E       ; <== Loop until Len.U == 0x0
- Đè tìm hiểu kỹ hơn về quá trình tạo chuỗi TempString ta dùng F7 trace into lệnh CALL
snpr_exe.004BA6CC :
004BA81E E8 45AFF4FF  CALL snpr_exe.00405768             ; <== Trace Into here
004BA823 52          PUSH EDX
004BA824 50          PUSH EAX
004BA825 8B45 F0      MOV EAX,DWORD PTR SS:[EBP-10]
004BA828 99          CDQ
004BA829 030424      ADD EAX,DWORD PTR SS:[ESP]         ; <== Value = Value + i
004BA82C 135424 04    ADC EDX,DWORD PTR SS:[ESP+4]
004BA830 83C4 08      ADD ESP,8
004BA833 52          PUSH EDX
004BA834 50          PUSH EAX
004BA835 8D55 D8      LEA EDX,DWORD PTR SS:[EBP-28]
004BA838 B8 02000000  MOV EAX,2
004BA83D E8 CEE8F4FF  CALL snpr_exe.00409110             ; <== %2X(Value)
- Tiếp tục Trace Into :
004057A7 D1E0        SHL EAX,1                         ; <== EAX == 0x6F314F0B
004057A9 D1D2        RCL EDX,1                         ; <== EDX == 0x21D
004057AB D1D6        RCL ESI,1                         ; <== ESI == 0x0 : Value
004057AD D1D7        RCL EDI,1                         ; <== EDI == 0x0
004057AF 39EF        CMP EDI,EBP
004057B1 72 0B        JB SHORT snpr_exe.004057BE
004057B3 77 04        JA SHORT snpr_exe.004057B9
004057B5 39DE        CMP ESI,EBX                      ; <== EBX == U[i-1]
004057B7 72 05        JB SHORT snpr_exe.004057BE
004057B9 29DE        SUB ESI,EBX                      ; <== Value
004057BB 19EF        SBB EDI,EBP
004057BD 40          INC EAX
004057BE ^ E2 E7      LOOPD SHORT snpr_exe.004057A7     ; <== Loop until ECX == 0x0

```

- Đối với chương trình này ta có thể diễn giải lại như sau :

/Section 1/- Sử dụng đặc tính của lệnh ROL để tìm ra giá trị tương ứng của từng ký tự .

/Section 2/- Từng giá trị này được chuyển đổi về dạng chuỗi theo định dạng “%2X” (có nghĩa : giá trị được chuyển về dạng HEX tương ứng, với hai ký tự . Ví dụ : Value == 2C thì CharString == “2C”; Nếu Value == 8 thì CharString == “08”)

/Section 3/- Chỉ có ký tự thứ hai của từng CharString được nối lại với nhau tạo chuỗi Serial thực (Ví dụ : CharString : “2C”, “08”, “3D” thì Serial : “C8D”)

- Chương trình này không phức tạp lắm, nhưng PROGRAMMER đã phức tạp hoá lên bằng các thêm nhiều đoạn CODE không có tác dụng và dấu đoạn mã hoá nằm ở khắp nơi .

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : 256F976AED67
User : VHT-cRaCkErS	Serial : ED2F976AED67

III – KeyGen :

/Section 1/- Thực hiện tính toán đối với từng ký tự

/Section 2/- Chuyển đổi định dạng . Và gắn tất cả các CharString lại với nhau

/Section 3/- Serial[j] = CharString [2*j + 1]

IV – SourceCode (VC++) :

```
char reaName[64]={0};
```

```
char reaSerial[128]={0};
```

```
char reaSerialTemp[128]={0};
```

```
char reaTemp[128]={0};
```

```
int LenUser=0;
```

```
int i=0, j=0;
```

```
int TempChar = 0;
```

```
LenUser=GetDlgItemText(IDC_Name,reaName,64);
```

```
if (LenUser < 1)
```

```
{
```

```
MessageBox("----- Your name atleast 1 chart -----","Hey !! Please  
input your name again !! ");
```

```
}
```

```
else
```

```
{
```

```
i=0;
```

```
while ( i < LenUser )
```

```
{
```

```
TempChar = reaName[i];
```

```
_asm
```

```
{
```

```
MOV EAX, 0x6F314F0B
```

```
MOV EDX, 0x21D
```

```
MOV ECX, 0x40
```

```
XOR EDI, EDI
```

```
XOR ESI, ESI
```

XOR EBX, EBX

First:

```

        SHL EAX,1
        RCL EDX,1
        RCL ESI,1
        RCL EDI,1
        CMP EDI,EBX
        JB Second
        JA Third
        CMP ESI,TempChar
        JB Second
Third:
        SUB ESI,TempChar
        SBB EDI,EBX
        INC EAX
Second:
        LOOP First
        MOV TempChar, ESI
    }
TempChar = TempChar + (i+1);
wsprintf(reaTemp,"%2X",TempChar);
lstrcat(reaSerialTemp,reaTemp);
i++;
}

i=0;
while ( i < lstrlen(reaSerialTemp))
{
    i = 2*j + 1;
    reaSerial[j] = reaSerialTemp[i];
    j++;
}
SetDlgItemText(IDC_Serial,reaSerial);
}

```

V – End of Tut :

- Finished – ***05/07/2004***

Reverse Engineering Association

SoftWare

Homepage	:	http://www.softsilver.com
Production	:	Softsilver Ltd.
SoftWare	:	Softsilver Transformer 2.5.1
Copyright by	:	Copyright © 2003 Softsilver Ltd. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N / A
Language	:	Microsoft Visual C++ 7.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

Softsilver Transformer 2.5.1

Softsilver Transformer is a Windows application written with simplicity and performance in mind. The program's function is to create and execute transformations from text files and databases to XML. These transformations can be saved as documents for re-use by the application, command line utility or COM component.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 7.0**.
- Nhập thử User và Fake Serial, ta nhận được thông báo "The registration key you entered is not valid. Please try again" Tuy nhiên ta không thể tìm thấy thông báo này bằng cách tìm chuỗi thông thường trong Olly . Tuy nhiên, ta thấy dòng "%s-%s-%s-%s" . Truy đến địa chỉ của dòng này :
004100BB |. 68 F0454500 PUSH st.004545F0 ; |Format = "%s-%s-%s-%s"
- Dò ngược lên trên ta thấy chương trình sử dụng hàm **API GetWindowTextA** để lấy các dữ liệu nhập. Nên ta đặt BreakPoint tại :
0040FFB7 |. 8B35 3C334500 MOV ESI,DWORD PTR DS:[<&USER32.GetDlgItem>;
USER32.GetItem

II – Cracking :

- Sau khi kết thúc quá trình lấy dữ liệu, chương trình tiến hành kiểm tra Serial :

```

00410010 |. 84C0      TEST AL,AL           ; <== Section I must be enter
00410012 |. 0F84 ED000000 JE st.00410105
00410018 |. 8A4424 08  MOV AL,BYTE PTR SS:[ESP+8]
0041001C |. 84C0      TEST AL,AL           ; <== Section II must be enter
0041001E |. 0F84 E1000000 JE st.00410105
00410024 |. 8A4424 10  MOV AL,BYTE PTR SS:[ESP+10]
00410028 |. 84C0      TEST AL,AL           ; <== Section III must be enter
0041002A |. 0F84 D5000000 JE st.00410105
00410030 |. 8A4424 20  MOV AL,BYTE PTR SS:[ESP+20]
00410034 |. 84C0      TEST AL,AL           ; <== Section IV must be enter
00410036 |. 0F84 C9000000 JE st.00410105
0041003C |. 8D4424 18  LEA EAX,DWORD PTR SS:[ESP+18]    ; <== Section I
00410040 |. 8D50 01   LEA EDX,DWORD PTR DS:[EAX+1]
00410043 > 8A08     /MOV CL,BYTE PTR DS:[EAX]
00410045 |. 40       |INC EAX
00410046 |. 84C9     |TEST CL,CL
00410048 |.^ 75 F9   |JNZ SHORT st.00410043
0041004A |. 2BC2     SUB EAX,EDX
0041004C |. 83F8 05   CMP EAX,5          ; <== Section I must be 5 charts
0041004F |. 0F85 B0000000 JNZ st.00410105
00410055 |. 8D4424 08  LEA EAX,DWORD PTR SS:[ESP+8]    ; <== Section II
00410059 |. 8D50 01   LEA EDX,DWORD PTR DS:[EAX+1]
0041005C |. 8D6424 00   LEA ESP,DWORD PTR SS:[ESP]
00410060 > 8A08     /MOV CL,BYTE PTR DS:[EAX]
00410062 |. 40       |INC EAX
00410063 |. 84C9     |TEST CL,CL
00410065 |.^ 75 F9   |JNZ SHORT st.00410060
00410067 |. 2BC2     SUB EAX,EDX
00410069 |. 83F8 05   CMP EAX,5          ; <== Section II must be 5 charts
0041006C |. 0F85 93000000 JNZ st.00410105

```

```

00410072 |. 8D4424 10 LEA EAX,DWORD PTR SS:[ESP+10] ; <== Section III
00410076 |. 8D50 01 LEA EDX,DWORD PTR DS:[EAX+1]
00410079 |. 8DA424 000000>LEA ESP,DWORD PTR SS:[ESP]
00410080 |> 8A08 /MOV CL,BYTE PTR DS:[EAX]
00410082 |. 40 |INC EAX
00410083 |. 84C9 |TEST CL,CL
00410085 |.^ 75 F9 |JNZ SHORT st.00410080
00410087 |. 2BC2 SUB EAX,EDX
00410089 |. 83F8 05 CMP EAX,5 ; <== Section III must be 5 charts
0041008C |. 75 77 JNZ SHORT st.00410105
0041008E |. 8D4424 20 LEA EAX,DWORD PTR SS:[ESP+20] ; <== Section IV
00410092 |. 8D50 01 LEA EDX,DWORD PTR DS:[EAX+1]
00410095 |> 8A08 /MOV CL,BYTE PTR DS:[EAX]
00410097 |. 40 |INC EAX
00410098 |. 84C9 |TEST CL,CL
0041009A |.^ 75 F9 |JNZ SHORT st.00410095
0041009C |. 2BC2 SUB EAX,EDX
0041009E |. 83F8 05 CMP EAX,5 ; <== Section IV must be 5 charts
004100A1 |. 75 62 JNZ SHORT st.00410105

```

- Kế đó là chương trình sử dụng hàm **API wsprintfA** để nối các Section này lại theo thứ tự nhập :

```

004100A3 |. 8D4C24 20 LEA ECX,DWORD PTR SS:[ESP+20]
004100A7 |. 51 PUSH ECX ; /%s>
004100A8 |. 8D5424 14 LEA EDX,DWORD PTR SS:[ESP+14] ; |
004100AC |. 52 PUSH EDX ; /%s>
004100AD |. 8D4424 10 LEA EAX,DWORD PTR SS:[ESP+10] ; |
004100B1 |. 50 PUSH EAX ; /%s>
004100B2 |. 8D4C24 24 LEA ECX,DWORD PTR SS:[ESP+24] ; |
004100B6 |. 51 PUSH ECX ; /%s>
004100B7 |. 8D5424 3C LEA EDX,DWORD PTR SS:[ESP+3C] ; |
004100BB |. 68 F0454500 PUSH st.004545F0 ; |Format = "%s-%s-%s-%s"
004100C0 |. 52 PUSH EDX ; |s
004100C1 |. FF15 10334500 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

- Trace tiếp ta đến đây :

```
004100FD |. E8 6EF50000 CALL st.0041F670 ; <== Trace Into here
```

- Dùng F7 trace into ta đến quá trình mã hoá chuỗi . Quá trình này được chia làm 3 giai đoạn .

- Giai đoạn mã hoá thứ nhất :

```

0041F747 |> 8A45 00 MOV AL,BYTE PTR SS:[EBP] ; <== Section_I[i]
0041F74A |. 84C0 TEST AL,AL
0041F74C |. 74 22 JE SHORT st.0041F770
0041F74E |. 0FBEC0 MOVSX EAX,AL
0041F751 |. 50 PUSH EAX
0041F752 |. 45 INC EBP
0041F753 |. E8 721A0200 CALL st.004411CA
0041F758 |. 03C6 ADD EAX,ESI ; <== Value_01 = Value_01 + Section_I[i]
0041F75A |. 1C1E0 03 SHL EAX,3 ; <== Value_01 = Value_01 * 8
0041F75D |. 83C4 04 ADD ESP,4
0041F760 |. 85ED TEST EBP,EBP
0041F762 |. 8BF0 MOV ESI,EAX ; <== Value_01
0041F764 |.^ 75 E1 JNZ SHORT st.0041F747 ; <== Loop until Len.SecI == 0x0
0041F766 |. EB 08 JMP SHORT st.0041F770
0041F768 |. 8DA424 000000>LEA ESP,DWORD PTR SS:[ESP]
0041F76F |. 90 NOP

```

```

0041F770 > 8B1D 10334500 MOV EBX,DWORD PTR DS:[<&USER32.wsprintfA>;
USER32.wsprintfA
0041F776 . 81E6 FFFF0F00 AND ESI,0FFFFF ; <== Value_01 = Value_01 and 0xFFFFF
0041F77C . 8BEE      MOV EBP,ESI ; <== Value_01
0041F77E . 55      PUSH EBP ; /%5x>
0041F77F . 8D4424 18 LEA EAX,DWORD PTR SS:[ESP+18] ; |
0041F783 . 68 E8584500 PUSH st.004558E8 ; |Format = "%5x"
0041F788 . 50      PUSH EAX ; |s
0041F789 . FFD3      CALL EBX ; \wsprintfA

```

- Kết thúc giai đoạn đầu ta có được chuỗi mã hoá dựa trên Section I . Chuỗi này được đem so sánh với Section II . Nếu không bằng thì thoát luôn mà không mã hoá tiếp.

- Giai đoạn mã hoá thứ hai :

```

0041F7A1 . 8B7C24 34 MOV EDI,DWORD PTR SS:[ESP+34] ; <== Default String : dStr
0041F7A5 . 33F6      XOR ESI,ESI
0041F7A7 > 8A07      MOV AL,BYTE PTR DS:[EDI] ; <== dStr[i]
0041F7A9 . 84C0      TEST AL,AL
0041F7AB . 74 18     JE SHORT st.0041F7C5
0041F7AD . 0FBEC0    MOVSX EAX,AL
0041F7B0 . 50      PUSH EAX
0041F7B1 . 47      INC EDI
0041F7B2 . E8 131A0200 CALL st.004411CA
0041F7B7 . 03C6      ADD EAX,ESI ; <== Value_02 = Value_02 + dStr[i]
0041F7B9 . C1E0 03   SHL EAX,3 ; <== Value_02 = Value_02 * 8
0041F7BC . 83C4 04   ADD ESP,4
0041F7BF . 85FF      TEST EDI,EDI
0041F7C1 . 8BF0      MOV ESI,EAX ; <== Value_02
0041F7C3 .^ 75 E2     JNZ SHORT st.0041F7A7 ; <== Loop until Len.dStr == 0x0
0041F7C5 > 0BF5      OR ESI,EBP ; <== Value_2 = Value_02 or Value_01
0041F7C7 . 81E6 FFFF0F00 AND ESI,0FFFFF ; <== Value_02 = Value_02 and 0xFFFFF
0041F7CD . 56      PUSH ESI
0041F7CE . 8D5424 18 LEA EDX,DWORD PTR SS:[ESP+18]
0041F7D2 . 68 E8584500 PUSH st.004558E8 ; ASCII "%5x"
0041F7D7 . 52      PUSH EDX
0041F7D8 . FFD3      CALL EBX

```

- Section IV không mã hoá, nhưng phải tuân theo một quy định :

```

0041F7F0 . 8B4C24 28 MOV ECX,DWORD PTR SS:[ESP+28] ; <== Section_IV
0041F7F4 . 8A01      MOV AL,BYTE PTR DS:[ECX] ; <== Section_IV[i]
0041F7F6 . 84C0      TEST AL,AL
0041F7F8 . 74 12     JE SHORT st.0041F80C
0041F7FA . 8D9B 00000000 LEA EBX,DWORD PTR DS:[EBX]
0041F800 > 3C 46     CMP AL,46 ; <== if ( Section_IV[i] < 0x46 )
0041F802 . 7F 41     JG SHORT st.0041F845 ; <== Continue check else BAD
0041F804 . 8A41 01   MOV AL,BYTE PTR DS:[ECX+1] ; <== Section_IV[i+1]
0041F807 . 41      INC ECX ; <== i++
0041F808 . 84C0      TEST AL,AL
0041F80A .^ 75 F4     JNZ SHORT st.0041F800 ; <== Loop until end of Section_IV

```

- Quá trình mã hoá chương trình này không phức tạp, nhưng chương trình đã sử dụng rất nhiều thủ thuật để đánh lừa . Rất cẩn thận khi check CODE.

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS

Serial : N/A

User : VHT-cRaCkErS

Serial : N/A

Serial : 93MYX-06300-A7310-49A31 or C0S28-53440-F7450-90E43

III – KeyGen :

- /Section 1/- Tạo chuỗi ngẫu nhiên gồm 5 ký tự
- /Section 2/- Chuỗi thứ hai và ba được tạo thành dựa trên chuỗi thứ nhất và chuỗi mặc định
- /Section 3/- Chuỗi thứ tư cũng là chuỗi ngẫu nhiên nhưng giá trị các ký tự phải nhỏ hơn 0x46
- /Section 4/- Kết hợp các chuỗi lại với nhau.

IV – SourceCode (VC++) :

```

char reaSerial[64]={0};
char reaDefaultString[50]="0123456789ABCDEFHIJKLMNOPQRSTUVWXYZ";
char reaDefault[20]="Transformer2";
char reaFourthSection[10]={0};
char reaTemp[20]={0};
int i=0;
int ValueSecondSection=0, ValueThirdSection=0, Temp=0;

srand( (unsigned)time( NULL ) );
i=0;
while ( i < 5 )
{
    reaSerial[i] = reaDefaultString[rand() % 36];
    i++;
}

i=0;    ValueSecondSection=0;
while ( i < 5 )
{
    if (reaSerial[i] < 0x61)
    {
        Temp = reaSerial[i];
    }
    else
    {
        Temp = reaSerial[i] - 0x20;
    }
    ValueSecondSection = ValueSecondSection + Temp;
    ValueSecondSection = ValueSecondSection << 0x3;
    i++;
}
ValueSecondSection = ValueSecondSection & 0xFFFF;
i=0;    ValueThirdSection=0;
while ( i < (lstrlen(reaDefault)) )
{
    if (reaDefault[i] < 0x61)
    {
        Temp = reaDefault[i];
    }
    else
    {
        Temp = reaDefault[i] - 0x20;
    }
}

```

```

ValueThirdSection = ValueThirdSection + Temp;
ValueThirdSection = ValueThirdSection << 0x3;
i++;
}
ValueThirdSection = ValueThirdSection |= ValueSecondSection;
ValueThirdSection = ValueThirdSection & 0xFFFF;

i=0;
while ( i < 5 )
{
    reaFourthSection[i] = reaDefaultString[rand() % 15];
    i++;
}
wsprintf(reTemp,"-%05.5X-%05.5X-",ValueSecondSection,ValueThirdSection);
lstrcat(reSerial,reTemp);
lstrcat(reSerial,reaFourthSection);
SetDlgItemText(IDC_Serial,reSerial);

```

V – End of Tut :

- Finished – **06/07/2004**

Reverse Engineering Association

SoftWare

Homepage	:	http://www.flashkeeper.com
Production	:	Sparkle Media System .
SoftWare	:	Sparkle Flash Keeper 3.0
Copyright by	:	Copyright © 2002-2004 Sparkle Media System . All Rights Reserved.
Type	:	Name / Serial
Packed	:	N / A
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

Sparkle Flash Keeper 3.0

Sparkle FlashKeeper is a handy flash tool kit with many easy-to-use Flash tools that help you to get many of your flash files organized and also simplify and accelerate Flash development. Preview and browse flash files, download flash from the Internet, download management, Create screensaver with batch of flash files, convert SWF and EXE format to each other individually or in batch etc. All of these features will help you enhance and extent the using possibility of existing flash files. And with friendly easy-operation user interface, SparkleMedia Flash Keeper offers you a fantastic animated flash world!

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử Fake Serial, ta nhận được thông báo "**Wrong Registration Code!**" Tìm chuỗi thông báo này trong Olly và ta tìm được ở địa chỉ :

004214AB |. 68 C0054600 PUSH FlashKee.004605C0 ; ASCII "Wrong Registration Code!"
 - Dò ngược lên trên và ta đặt BreakPoint tại :
 004213CF |. E8 BA1D0200 CALL <JMP.&MFC42.#6334> ; <== Set BP here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP .
 Chương trình kiểm tra chiều dài Serial nhập :

0042142C > \83F8 2C CMP EAX,2C	; <== LenS must be 44 charts
0042142F . 7D 20 JGE SHORT FlashKee.00421451	
00421431 . 6A 00 PUSH 0	
00421433 . 6A 00 PUSH 0	
00421435 . 68 C0054600 PUSH FlashKee.004605C0	; ASCII "Wrong Registration Code!"
0042143A . E8 991E0200 CALL <JMP.&MFC42.#1200>	

- Trace tiếp ta đến :

0042145E . E8 6D880100 CALL FlashKee.00439CD0	; <== Trace Into
--	------------------

- Dùng F7 trace into để xem xét quá trình mã hoá .

- Trace xuống một đoạn ta đến quá trình mã hoá đầu tiên . Quá trình này phân làm 4 giai đoạn . Hai giai đoạn đầu tiên dùng để tính giá trị đầu vào . Hai giai đoạn sau dùng để mã hoá chuỗi .

00439D88 . E8 33F9FFFF CALL FlashKee.004396C0	; <== Encrypt SecII based on SecI
----- Encrypt SecII -----	
00439709 . E8 D2FEFFFF CALL FlashKee.004395E0	; <== FirstCalculation : Value
----- First Calculation -----	
004395E5 . 33F6 XOR ESI,ESI	; <== Value = 0
004395E7 . 33C0 XOR EAX,EAX	; <== i = 0
004395E9 > 0FBEB1408 /MOVSD EDX,BYTE PTR DS:[EAX+ECX]	; <== SecI[i]
004395ED . 40 INC EAX	; <== i++
004395EE . 83F8 07 CMP EAX,7	; <== while (i < 7)
004395F1 . 8D7416 BF LEA ESI,DWORD PTR DS:[ESI+EDX-41]	; <== Value += SecI[i] - 0x41
004395F5 .^ 7C F2 JL SHORT FlashKee.004395E9	
004395F7 . 8D4C24 08 LEA ECX,DWORD PTR SS:[ESP+8]	
004395FB . E8 589B0000 CALL <JMP.&MFC42.#800>	
00439600 B8 93244992 MOV EAX,92492493	; <== dV
00439605 . F7EE IMUL ESI	; <== Over = Value * dV
00439607 . 8BC2 MOV EAX,EDX	; <== Over
00439609 . 03C6 ADD EAX,ESI	; <== Over = Over + Value
0043960B . 5E POP ESI	
0043960C . C1F8 02 SAR EAX,2	; <== Over = Over sar 0x2
0043960F . 8BC8 MOV ECX,EAX	; <== Over
00439611 . C1E9 1F SHR ECX,1F	; <== Temp = Over / 0x80000000
00439614 . 03C1 ADD EAX,ECX	; <== Value = Over + Temp
----- First Calculation -----	
0043970E . 8D5424 38 LEA EDX,DWORD PTR SS:[ESP+38]	
00439712 . 8BCC MOV ECX,ESP	
00439714 . 896424 14 MOV DWORD PTR SS:[ESP+14],ESP	
00439718 . 52 PUSH EDX	
00439719 . 8BF0 MOV ESI,EAX	
0043971B . E8 229B0000 CALL <JMP.&MFC42.#535>	
00439720 . E8 4BFFFFFF CALL FlashKee.00439670	; <== SecondCalculation : Cal
----- Second Calculation -----	
0043967F . BE 1A000000 MOV ESI,1A	; <== dV

```

00439684 > 0FBE3C10 /MOVsx EDI,BYTE PTR DS:[EAX+EDX] ; <== SecI[i]
00439688 |. 83EF 41 |SUB EDI,41 ; <== Temp = SecI[i] - 0x41
0043968B |. 40 |INC EAX ; <== i++
0043968C |. 8939 |MOV DWORD PTR DS:[ECX],EDI ; <== Temp[i] = Temp
0043968E |. 83C1 04 |ADD ECX,4
00439691 |. 83F8 07 |CMP EAX,7 ; <== while ( i < 7 )
00439694 |.^ 7C EE |JL SHORT FlashKee.00439684 ; <== Continue Loop
00439696 |. 8D4C24 08 LEA ECX,DWORD PTR SS:[ESP+8]
0043969A |. BA 07000000 MOV EDX,7 ; <== Numbers of Loop : nL
0043969F > 8B01 /MOV EAX,DWORD PTR DS:[ECX]
004396A1 |. 3BC6 |CMP EAX,ESI ; <== if ( Temp[i] < dV )
004396A3 |. 7D 02 |JGE SHORT FlashKee.004396A7 ; <== then
004396A5 |. 8BF0 |MOV ESI,EAX ; <== Cal = Temp[i]
004396A7 > 83C1 04 |ADD ECX,4 ; <== else
004396AA |. 4A |DEC EDX ; <== nL --
004396AB |.^ 75 F2 |JNZ SHORT FlashKee.0043969F ; <== Loop until nL = 0
----- Second Calculation -----
00439725 |. 2BF0 SUB ESI,EAX ; <== Cal = Cal - Value
00439727 |. 8D4424 18 LEA EAX,DWORD PTR SS:[ESP+18]
0043972B |. 8BFE MOV EDI,ESI ; <== Cal
0043972D |. 8B7424 38 MOV ESI,DWORD PTR SS:[ESP+38] ; <== SecI
00439731 |. 83C4 04 ADD ESP,4
00439734 |. 33C9 XOR ECX,ECX ; <== i = 0
00439736 |. 2BF0 SUB ESI,EAX
00439738 > 8D540C 14 /LEA EDX,DWORD PTR SS:[ESP+ECX+14]
0043973C |. 8A0416 |MOV AL,BYTE PTR DS:[ESI+EDX] ; <== SecI[i]
0043973F |. 2C 41 |SUB AL,41 ; <== SecI = SecI - 0x41
00439741 |. 41 |INC ECX ; <== i++
00439742 |. 83F9 08 |CMP ECX,8 ; <== while ( i < 8 )
00439745 |. 8802 |MOV BYTE PTR DS:[EDX],AL ; <== SecI[i]
00439747 |.^ 7C EF |JL SHORT FlashKee.00439738 ; <== Continue Loop
00439749 |. 33F6 XOR ESI,ESI ; <== i = 0
0043974B > 0FBE4C34 14 /MOVsx ECX,BYTE PTR SS:[ESP+ESI+14] ; <== SecI[i]
00439750 |. 03CF |ADD ECX,EDI ; <== Temp = SecI[i] + Cal
00439752 |. B8 4FECC44E |MOV EAX,4EC4EC4F ; <== dV2
00439757 |. F7E9 |IMUL ECX ; <== Over = Temp * dV2
00439759 |. 8BC2 |MOV EAX,EDX ; <== Over
0043975B |. C1F8 03 |SAR EAX,3 ; <== Over = Over sar 0x3
0043975E |. 8BD0 |MOV EDX,EAX ; <== Over
00439760 |. C1EA 1F |SHR EDX,1F ; <== Temp = Over / 0x80000000
00439763 |. 03C2 |ADD EAX,EDX ; <== Over = Over + Temp
00439765 |. B2 1A |MOV DL,1A ; <== dV3
00439767 |. F6EA |IMUL DL ; <== Over = Over * dV3
00439769 |. 2AC8 |SUB CL,AL ; <== Value = dV1 - Over
0043976B |. 80C1 41 |ADD CL,41 ; <== Value = Value + 0x41
0043976E |. 884C34 14 |MOV BYTE PTR SS:[ESP+ESI+14],CL ; <== SecI[i] = Value
00439772 |. 46 |INC ESI ; <== i++
00439773 |. 83FE 08 |CMP ESI,8 ; <== while ( i < 8 )
00439776 |.^ 7C D3 |JL SHORT FlashKee.0043974B ; <== Continue Loop
----- Encrypt SecII -----
- Quá trình mã hoá đoạn thứ ba :
00439DC2 |. E8 19FAFFFF CALL FlashKee.004397E0 ; <== Encrypt SecIII based on SecI

```

----- Encrypt SecIII -----

```

00439823 > /8D540C 0C    /LEA EDX,DWORD PTR SS:[ESP+ECX+C]
00439827 |. |8A0416    |MOV AL,BYTE PTR DS:[ESI+EDX]
0043982A |. |2C 41     |SUB AL,41
0043982C |. |41       |INC ECX
0043982D |. |83F9 08    |CMP ECX,8
00439830 |. |8802      |MOV BYTE PTR DS:[EDX],AL
00439832 |.^\7C EF    \JL SHORT FlashKee.00439823
00439834 |. 33F6      XOR ESI,ESI
00439836 > 0FBEC4C34 0C /MOVSX ECX,BYTE PTR SS:[ESP+ESI+C]      ; <== SecII[i]
0043983B |. 0FAFC9      |IMUL ECX,ECX          ; <== Temp = SecII[i] * SecII[i]
0043983E |. B8 4FECC44E  |MOV EAX,4EC4EC4F          ; <== dV1
00439843 |. F7E9      |IMUL ECX          ; <== Over = Temp * dV1
00439845 |. 8BC2      |MOV EAX,EDX          ; <== Over
00439847 |. C1F8 03    |SAR EAX,3          ; <== Over = Over sar 0x3
0043984A |. 8BD0      |MOV EDX,EAX          ; <== Over
0043984C |. C1EA 1F    |SHR EDX,1F          ; <== Temp = Over / 0x80000000
0043984F |. 03C2      |ADD EAX,EDX          ; <== Over = Over + Temp
00439851 |. B2 1A      |MOV DL,1A          ; <== dV2
00439853 |. F6EA      |IMUL DL           ; <== Value = Value * Over
00439855 |. 2AC8      |SUB CL,AL          ; <== Value = SecII[i] - Value
00439857 |. 80C1 41    |ADD CL,41          ; <== Value = Value + 0x41
0043985A |. 884C34 0C    |MOV BYTE PTR SS:[ESP+ESI+C],CL      ; <== SecII[i] = Value
0043985E |. 46       |INC ESI           ; <== i++
0043985F |. 83FE 08    |CMP ESI,8          ; <== while ( i < 8 )
00439862 |.^\7C D2    \JL SHORT FlashKee.00439836      ; <== Continue Loop

```

----- Encrypt SecIII -----

- Quá trình mã hoá đoạn thứ tư cũng phức tạp như đoạn thứ nhất . Được chia ra làm 4 giai đoạn . Giai đoạn đầu tiên giống ở quá trình mã hoá thứ nhất, giai đoạn thứ hai có một chút thay đổi . Hai giai đoạn còn lại không khác biệt gì :

```
00439DFC |. E8 CFFAFFFF CALL FlashKee.004398D0      ; <== Encrypt SecIV based on SecI
```

----- Encrypt SecIV -----

```

00439919 |. E8 C2FCFFFF CALL FlashKee.004395E0      ; <== FirstCalculation : Value
0043991E |. 8D5424 38    LEA EDX,DWORD PTR SS:[ESP+38]
00439922 |. 8BCC      MOV ECX,ESP
00439924 |. 896424 14    MOV DWORD PTR SS:[ESP+14],ESP
00439928 |. 52       PUSH EDX
00439929 |. 8BF0      MOV ESI,EAX
0043992B |. E8 12990000  CALL <JMP.&MFC42.#535>
00439930 |. E8 EBFCFFFF CALL FlashKee.00439620      ; <== SecondCalculation : Cal

```

----- Second Calculation -----

..... Cutting

```

0043964C > 8B01      /MOV EAX,DWORD PTR DS:[ECX]
0043964E |. 3BC6      |CMP EAX,ESI          ; <== if ( Temp[i] > dV )
00439650 |. 7E 02      \JLE SHORT FlashKee.00439654
00439652 |. 8BF0      |MOV ESI,EAX
00439654 > 83C1 04    |ADD ECX,4
00439657 |. 4A       |DEC EDX
00439658 |.^ 75 F2    \JNZ SHORT FlashKee.0043964C

```

----- Second Calculation -----

```

00439935 |. 2BC6      SUB EAX,ESI          ; <== Cal = Cal - Value
00439937 |. 8B7424 38    MOV ESI,DWORD PTR SS:[ESP+38]

```

```

0043993B |. 83C4 04 ADD ESP,4
0043993E |. 40 INC EAX ; <== Cal = Cal + 1
0043993F |. 8BF8 MOV EDI,EAX
00439941 |. 8D4424 14 LEA EAX,DWORD PTR SS:[ESP+14]
00439945 |. 33C9 XOR ECX,ECX
00439947 |. 2BF0 SUB ESI,EAX
00439949 > 8D540C 14 /LEA EDX,DWORD PTR SS:[ESP+ECX+14]
0043994D |. 8A0416 |MOV AL,BYTE PTR DS:[ESI+EDX]
00439950 |. 2C 41 |SUB AL,41
00439952 |. 41 |INC ECX
00439953 |. 83F9 08 |CMP ECX,8
00439956 |. 8802 |MOV BYTE PTR DS:[EDX],AL
00439958 |.^ 7C EF \JL SHORT FlashKee.00439949
0043995A |. 33F6 XOR ESI,ESI
0043995C > 0FBE4C34 14 /MOVSX ECX,BYTE PTR SS:[ESP+ESI+14]
00439961 |. 03CF |ADD ECX,EDI
00439963 |. B8 4FECC44E |MOV EAX,4EC4EC4F
00439968 |. F7E9 |IMUL ECX
0043996A |. 8BC2 |MOV EAX,EDX
0043996C |. C1F8 03 |SAR EAX,3
0043996F |. 8BD0 |MOV EDX,EAX
00439971 |. C1EA 1F |SHR EDX,1F
00439974 |. 03C2 |ADD EAX,EDX
00439976 |. B2 1A |MOV DL,1A
00439978 |. F6EA |IMUL DL
0043997A |. 2AC8 |SUB CL,AL
0043997C |. 80C1 41 |ADD CL,41
0043997F |. 884C34 14 |MOV BYTE PTR SS:[ESP+ESI+14],CL
00439983 |. 46 |INC ESI
00439984 |. 83FE 08 |CMP ESI,8
00439987 |.^ 7C D3 \JL SHORT FlashKee.0043995C
===== Encrypt SecIV =====

```

- Quá trình mã hoá đoạn thứ năm hoàn toàn tương đồng với quá trình mã hoá đoạn thứ ba :

00439E36 |. E8 B5FBFFFF CALL FlashKee.004399F0 ; <== Encrypt SecV based on SecI

```

===== Encrypt SecV =====
00439A2A |. 33C9 XOR ECX,ECX
00439A2C |. C64424 20 02 MOV BYTE PTR SS:[ESP+20],2
00439A31 |. 2BF0 SUB ESI,EAX
00439A33 > 8D540C 0C /LEA EDX,DWORD PTR SS:[ESP+ECX+C]
00439A37 |. 8A0416 |MOV AL,BYTE PTR DS:[ESI+EDX]
00439A3A |. 2C 41 |SUB AL,41
00439A3C |. 41 |INC ECX
00439A3D |. 83F9 08 |CMP ECX,8
00439A40 |. 8802 |MOV BYTE PTR DS:[EDX],AL
00439A42 |.^ 7C EF \JL SHORT FlashKee.00439A33
00439A44 |. 33F6 XOR ESI,ESI
00439A46 > 0FBE4434 0C MOVSX EAX,BYTE PTR SS:[ESP+ESI+C]
00439A4B |. 8BC8 MOV ECX,EAX
00439A4D |. 0FAFC8 IMUL ECX,EAX
00439A50 |. 0FAFC8 IMUL ECX,EAX
00439A53 |. B8 4FECC44E MOV EAX,4EC4EC4F
00439A58 |. F7E9 IMUL ECX

```

```

00439A5A |. 8BC2      MOV EAX,EDX
00439A5C |. C1F8 03   SAR EAX,3
00439A5F |. 8BD0      MOV EDX,EAX
00439A61 |. C1EA 1F   SHR EDX,1F
00439A64 |. 03C2      ADD EAX,EDX
00439A66 |. B2 1A      MOV DL,1A
00439A68 |. F6EA      IMUL DL
00439A6A |. 2AC8      SUB CL,AL
00439A6C |. 80C1 41   ADD CL,41
00439A6F |. 884C34 0C   MOV BYTE PTR SS:[ESP+ESI+C],CL
00439A73 |. 46       INC ESI
00439A74 |. 83FE 08   CMP ESI,8
00439A77 |.^ 7C CD   JL SHORT FlashKee.00439A46
----- Encrypt SecV -----

```

- Các chuỗi này cùng với chuỗi đầu tiên được nối kết lại với nhau theo định dạng :
“AAAAAAA-BBBBBBBB-CCCCCCC-DDDDDDD-EEEEEEE”

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : UNCVTFTG-GZOHFRFS-KNEZXZXK-CVKDBNBO-SNIFVVVI Or BVLKYFZA-NHXWKRLM-BZRWEZBA-OIYXLSMN-BFFMSVZA

III – KeyGen :

N/A

IV – End of Tut :

- Finished – **29/07/2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.flashkeeper.com
Production :	Sparkle Media System .
SoftWare :	Sparkle SWF Desktop 1.0
Copyright by :	Copyright © 2002-2004 Sparkle Media System . All Rights Reserved.
Type :	Name / Serial (Standard or Professional)
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Sparkle SWF Desktop 1.0

The easy to use yet great utility to enhance the power of Macromedia Flash. Creating wallpaper that gracefully integrates vector graphics, animations, dynamic content and anything you can see in browser. Sharing your cool flash work with your family, friends or your customers.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử Fake Serial, ta nhận được thông báo "**Wrong Registration Code!**" Tìm chuỗi thông báo này trong Olly và ta tìm được ở địa chỉ :
00408FAF |. 68 F4294500 PUSH SwfDeskt.004529F4 ; ASCII "Wrong Registration Code!"
- Đồng thời trong quá trình tìm chuỗi ta thấy hai dòng thông báo "**Congratulation! You have registered Sparkle SWF Desktop Professional Version successfully!**" và "**Congratulation! You have registered Sparkle SWF Desktop Standard Version successfully!**". Như vậy sẽ có hai Serial khác nhau .
- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :
00408E80 |. E8 12CA0200 CALL SwfDeskt.00435897 ; <== Set BP here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến qua trình kiểm tra chiều dài chuỗi S :

```
00408ECF |> \83F8 1B    CMP EAX,1B          ; <== Len.S == 0x1B ??  
00408ED2 |. 74 1D      JE SHORT SwfDeskt.00408EF1   ; <== or  
00408ED4 |. 83F8 29    CMP EAX,29          ; <== Len.S == 0x29 ??  
00408ED7 |. 74 18      JE SHORT SwfDeskt.00408EF1
```

- Trace xuống chút và ta đến :

```
00408EFE |. E8 2DE1FFFF CALL SwfDeskt.00407030 ; <== Trace Into here
```

- Dùng F7 trace into ta đến qua trình phân loại Serial . Ở đây, tuỳ theo chiều dài của chuỗi Serial mà chuỗi Serial thuộc dạng STD (LenSerial = 0x1B : STD) hay PRO (LenSerial = 0x29 : STD)

```
00407080 |. 83F8 1B    CMP EAX,1B          ; <== if Len.S = 0x1B  
00407083 |. 75 04      JNZ SHORT SwfDeskt.00407089 ; <== then  
00407085 |. 33F6      XOR ESI,ESI          ; <== jmp to STD  
00407087 |. EB 0E      JMP SHORT SwfDeskt.00407097 ; <== or  
00407089 |> 83F8 29    CMP EAX,29          ; <== if Len.S = 0x29  
0040708C |. 0F85 F3040000 JNZ SwfDeskt.00407585 ; <== then  
00407092 |. BE 01000000 MOV ESI,1           ; <== jmp to PRO
```

- Cho dù thuộc dạng PRO hay STD thì cách mã hoá là như nhau (Chương trình sử dụng các ký tự ở vị trí [3][4][5] của chuỗi Serial để mã hoá) . Chỉ khác biệt là các tham số mặc định đầu vào . Ở đây chỉ xét một trường hợp :

```
004070E1 |. E8 5A120000 CALL SwfDeskt.00408340 ; <== Section I
```

===== Trace into =====

```
00408361 |. 6A FB      PUSH -5            ; <== Input Value : iV_01  
00408363 |. 6A 02      PUSH 2             ; <== Input Value : iV_02  
00408365 |. 6A FD      PUSH -3            ; <== Input Value : iV_03  
00408367 |. 6A 05      PUSH 5             ; <== Input Value : iV_04  
00408369 |. 6A FE      PUSH -2            ; <== Input Value : iV_05  
0040836B |. 6A 03      PUSH 3             ; <== Input Value : iV_06  
0040836D |. 51         PUSH ECX  
0040836E |. 8D4424 3C  LEA EAX,DWORD PTR SS:[ESP+3C]  
00408372 |. 8BCC      MOV ECX,ESP  
00408374 |. 896424 24  MOV DWORD PTR SS:[ESP+24],ESP  
00408378 |. 50         PUSH EAX  
00408379 |. C74424 34 010>MOV DWORD PTR SS:[ESP+34],1  
00408381 |. E8 50E60200 CALL SwfDeskt.004369D6  
00408386 |. 8B7424 38  MOV ESI,DWORD PTR SS:[ESP+38]  
0040838A |. 56         PUSH ESI  
0040838B |. E8 10FEFFFF CALL SwfDeskt.004081A0 ; <== Trace Into
```

===== Trace into =====

```

0040822A |. 8A08      MOV CL,BYTE PTR DS:[EAX]    ; <== SectionI[3]
0040822C |. 8A5424 38  MOV DL,BYTE PTR SS:[ESP+38] ; <== iV_06
00408230 |. 02CA      ADD CL,DL                 ; <== Value = SectionI[3] + iV_06
00408232 |. 884C24 0F  MOV BYTE PTR SS:[ESP+F],CL   ; <== Temp[0] = Value
00408236 |. 8A50 01    MOV DL,BYTE PTR DS:[EAX+1]  ; <== SectionI[4]
00408239 |. 025424 3C  ADD DL,BYTE PTR SS:[ESP+3C] ; <== Value = SectionI[4] + iV_05
0040823D |. 885424 0E  MOV BYTE PTR SS:[ESP+E],DL   ; <== Temp[1] = Value
00408241 |. 8A48 02    MOV CL,BYTE PTR DS:[EAX+2]  ; <== SectionI[5]
00408244 |. 024C24 40  ADD CL,BYTE PTR SS:[ESP+40] ; <== Value = SectionI[5] + iV_04
00408248 |. 884C24 0D  MOV BYTE PTR SS:[ESP+D],CL   ; <== Temp[2] = Value
0040824C |. 8A10      MOV DL,BYTE PTR DS:[EAX]    ; <== SectionI[3]
0040824E |. 025424 44  ADD DL,BYTE PTR SS:[ESP+44] ; <== Value = SectionI[3] + iV_03
00408252 |. 885424 0C  MOV BYTE PTR SS:[ESP+C],DL   ; <== Temp[3] = Value
00408256 |. 8A48 01    MOV CL,BYTE PTR DS:[EAX+1]  ; <== SectionI[4]
00408259 |. 024C24 48  ADD CL,BYTE PTR SS:[ESP+48] ; <== Value = SectionI[4] + iV_02
0040825D |. 884C24 0B  MOV BYTE PTR SS:[ESP+B],CL   ; <== Temp[4] = Value
00408261 |. 8A50 02    MOV DL,BYTE PTR DS:[EAX+2]  ; <== SectionI[5]
00408264 |. 8A4424 4C  MOV AL,BYTE PTR SS:[ESP+4C] ; <== iV_01
00408268 |. 02D0      ADD DL,AL                 ; <== Value = SectionI[5] + iV_01
0040826A |. 8D4424 0F  LEA EAX,DWORD PTR SS:[ESP+F]
0040826E |. 50         PUSH EAX                ; <== Temp[5] = Value
0040826F |. 885424 0E  MOV BYTE PTR SS:[ESP+E],DL   ; <== Check Validity of Charts
00408273 |. E8 08FFFFFF CALL SwfDeskt.00408180

```

===== Trace into =====

```

00408180 /$ 8B4424 04  MOV EAX,DWORD PTR SS:[ESP+4] ; <== Temp[i]
00408184 |. 8038 5A    CMP BYTE PTR DS:[EAX],5A    ; <== if (Temp[i] > 0x5A)
00408187 |. 7E 03      JLE SHORT SwfDeskt.0040818C ; <== then
00408189 |. C600 5A    MOV BYTE PTR DS:[EAX],5A    ; <== Temp[i] = 0x5A
0040818C |> 8038 41    CMP BYTE PTR DS:[EAX],41    ; <== else if (Temp[i] < 0x41)
0040818F |. 7D 03      JGE SHORT SwfDeskt.00408194 ; <== then
00408191 |. C600 41    MOV BYTE PTR DS:[EAX],41    ; <== Temp[i] = 0x41

```

===== Trace into =====

```

00408278 |. 8D4C24 12  LEA ECX,DWORD PTR SS:[ESP+12]
0040827C |. 51         PUSH ECX                ; <== Check Validity of Charts
0040827D |. E8 FEFEEEEEECALL SwfDeskt.00408180
00408282 |. 8D5424 15  LEA EDX,DWORD PTR SS:[ESP+15]
00408286 |. 52         PUSH EDX                ; <== Check Validity of Charts
00408287 |. E8 F4FEFFFFCALL SwfDeskt.00408180
0040828C |. 8D4424 18  LEA EAX,DWORD PTR SS:[ESP+18]
00408290 |. 50         PUSH EAX                ; <== Check Validity of Charts
00408291 |. E8 EAFFFEFFFFCALL SwfDeskt.00408180
00408296 |. 8D4C24 1B  LEA ECX,DWORD PTR SS:[ESP+1B]
0040829A |. 51         PUSH ECX                ; <== Check Validity of Charts
0040829B |. E8 E0FEFFFFCALL SwfDeskt.00408180
004082A0 |. 8D5424 1E  LEA EDX,DWORD PTR SS:[ESP+1E]
004082A4 |. 52         PUSH EDX                ; <== Check Validity of Charts
004082A5 |. E8 D6FEFFFFCALL SwfDeskt.00408180

```

===== Trace into =====

===== Trace into =====

- Sau đó, các giá trị nà đc chuyển thành chuỗi ký tự theo định dạng :

```

004082AA |. 0FBE4424 22 MOVSX EAX,BYTE PTR SS:[ESP+22]
004082AF |. 0FBE4C24 23 MOVSX ECX,BYTE PTR SS:[ESP+23]
004082B4 |. 0FBE5424 24 MOVSX EDX,BYTE PTR SS:[ESP+24]
004082B9 |. 50      PUSH EAX
004082BA |. 51      PUSH ECX
004082BB |. 0FBE4424 2D MOVSX EAX,BYTE PTR SS:[ESP+2D]
004082C0 |. 0FBE4C24 2E MOVSX ECX,BYTE PTR SS:[ESP+2E]
004082C5 |. 52      PUSH EDX
004082C6 |. 50      PUSH EAX
004082C7 |. 0FBE5424 37 MOVSX EDX,BYTE PTR SS:[ESP+37]
004082CC |. 51      PUSH ECX
004082CD |. 52      PUSH EDX
004082CE |. 8D4424 40 LEA EAX,DWORD PTR SS:[ESP+40]
004082D2 |. 68 30294500 PUSH SwfDeskt.00452930      ; ASCII "%c%c%c%c%c%c%c"
004082D7 |. 50      PUSH EAX
004082D8 |. E8 E7970200 CALL SwfDeskt.00431AC4

```

- Tương tự như vậy, các tham số của quá trình mã hoá STD :

- 5,2,-3,5,-2,3
- 5,4,-2,2,-4,5
- 7,6,-1,7,-6,1

- Các tham số của quá trình mã hoá PRO :

- 3,7,-4,7,-4,3
- 1,5,-2,1,-2,5
- 1,-5,3,3,-5,1
- 4,-7,3,4,-7,3
- 7,-5,4,4,-5,7

/*/*/* - SERIAL tương ứng :

User : N/A

Serial : REAREN-UCSOGI-WAPPII-SAUQKG (STD)

REAHEJ-KAQDLM-MCKFJK-IAMKAK-KANKAN-OANLAQ (PRO)

III – KeyGen :

/Section 1/- Tạo chuỗi ngẫu nhiên gồm 6 ký tự . (First section à Serial)
/Section 2/- Mã hoá lần lượt theo công thức :

```

reaTemp[0] = reaProfessional[3] + 3;
reaTemp[1] = reaProfessional[4] - 4;
reaTemp[2] = reaProfessional[5] + 7;
reaTemp[3] = reaProfessional[3] - 4;
reaTemp[4] = reaProfessional[4] + 7;
reaTemp[5] = reaProfessional[5] + 3;
i=0;
while ( i < 6 )
{
    if ( reaTemp[i] > 0x5A )
    {
        reaTemp[i] = 0x5A;
    }
    else if ( reaTemp[i] < 0x41 )
    {
        reaTemp[i] = 0x41;
    }
    i++;
}

```

}
 /Section 3/- Gắn kết các chuỗi lại với nhau .

IV – End of Tut :

- Finished – **25/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.speederxp.com
Production :	Brothers Software .
SoftWare :	SpeederXP v1.60 full
Copyright by :	Copyright © 2002-2003 Brothers Software . All Rights Reserved.
Type :	Name / Serial
Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

SpeederXP v1.60 full

This program(**SpeederXP**) can be used to adjust your Windows operation system speed . All software will change their speed after you adjust the speed rating in SpeederXP.

You can use SpeederXP fast your computer and make your old slow computer fast and more efficient.

You can also use SpeederXP in your games, such as Counter-Strike and StarCraft. You will run faster in game than ever and get a new feeling in these games.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo** . UnPACK và kiểm tra lại biết chương trình được viết bằng **Microsoft Visual C++ 6.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo “**Wrong Code, please check your input**”. Tuy nhiên, ở chương trình này, nếu ta load chương trình chính bằng Olly, thì chương trình sẽ bị CRASH . Vì thế ta chuyển qua load file kiểm tra **Register.exe** .
- Tuy nhiên ta cũng không thể sử dụng phương pháp tìm kiếm chuỗi thông thường để tìm chuỗi này, ta dùng đến phương pháp STACK . Từ đây ta xác định được FUNCTION chính của chương trình này .
- Dò ngược lên trên ta đặt BreakPoint tại đây :
 00402FEE E8 FD030000 CALL register.004033F0 ; JMP to MFC42.#6334

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP .
- Trace xuống chút ta đến :

00403063 E8 D8F4FFFF CALL register.00402540 ; <== Trace Into here

- Dùng F7 trace into, và trace tiếp một đoạn ta đến :

004025ED E8 1EFAFFFF CALL register.00402010 ; <== Encrypt

- Tiếp tục dùng F7 trace into :

004020AF E8 8CF0FFFF CALL register.00401140 ; <== MD5 Encrypt

===== MD5-Encrypt =====

00401177 E8 14FFFFFF CALL register.00401090 ; <== MD5 Hash

===== MD5-Hash =====

Quá trình tạo chuỗi MD5 diễn ra bình thường, chỉ có điều MD5Update có sự thay đổi so với MD5Update chính thức . Sự thay đổi này dẫn đến kết quả MD5Hash thay đổi so với nguyên bản .

00401060 C741 04 1108791>MOV DWORD PTR DS:[ECX+4],19790811

00401067 C741 08 0512781>MOV DWORD PTR DS:[ECX+8],19781205

0040106E C741 0C FECDBA9>MOV DWORD PTR DS:[ECX+C],98BADCCE

00401075 C741 10 7654321>MOV DWORD PTR DS:[ECX+10],10325476

===== MD5-Hash =====

===== MD5-Encrypt =====

- Chuỗi MD5Hash này sẽ được chương trình cắt lấy 2 đoạn :

00402139 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]

0040213D 6A 08 PUSH 8 ; <== Get LAST 8 charts

0040213F 51 PUSH ECX

00402140 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]

00402144 E8 6D100000 CALL register.004031B6 ; JMP to MFC42.#5710

00402149 8BF0 MOV ESI,EAX

0040214B 8D5424 18 LEA EDX,DWORD PTR SS:[ESP+18]

0040214F 6A 08 PUSH 8 ; <== Get FIRST 8 charts

00402151 52 PUSH EDX

00402152 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]

00402156 C68424 08010000>MOV BYTE PTR SS:[ESP+108],5

0040215E E8 4D100000 CALL register.004031B0 ; JMP to MFC42.#4129

- Hai chuỗi này sẽ được gắn lại với nhau và đây chính là chuỗi Serial thực của chương trình .

0040260F 8B4C24 28 MOV ECX,DWORD PTR SS:[ESP+28] ; <== Fake Serial

00402613 8B5424 0C MOV EDX,DWORD PTR SS:[ESP+C] ; <== Real Serial

00402617 51 PUSH ECX

00402618 52 PUSH EDX

00402619 FF15 28424000 CALL DWORD PTR DS:[404228] ; MSVCRT._mbscmp

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : f68afac7d2a26eea

III – KeyGen :

/Section I /- Tạo chuỗi MD5Hash – MD5Update có thay đổi so với phiên bản gốc

/Section II /- 8 ký tự đầu tiên và cuối cùng được kết hợp với nhau tạo chuỗi Serial thực .

IV – End of Tut :

- Finished – August 21, 2004

Reverse Engineering Association

SoftWare

Homepage :	http://www.anthemion.co.uk/dialogblocks
Production :	Anthemion Software Ltd.
SoftWare :	StoryLines 1.27
Copyright by :	Copyright © Anthemion Software Ltd, May 2004. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

StoryLines 1.27

Anthemion StoryLines is a sophisticated but easy-to-use application that lets you quickly arrange card-sized ideas in multiple storylines, freeing you from the constraints of linear thinking. Using a highly graphical way of working, StoryLines encourages 'right-brain thinking'.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử Name và Fake Serial, ta nhận được thông báo "*Sorry, invalid registration key...*" Ta tìm được đoạn CODE này ở địa chỉ :
 00447E51 > B8 E8126100 MOV EAX,storylin.006112E8 ; ASCII "Sorry, invalid registration key. The key should be of the form: EA0927CF-E8CF149B-416363EE (three groups of 8 characters). Please also check that your specified user name is the same as the one under which you purchased StoryLines.Tr"...
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
 00447E0E . E8 FDB21400 CALL <JMP.&MSVCRT._EH_prolog> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP. Để đi đến được đoạn CODE mã hoá của chương trình ta cần trace into vài lần :

00447E2A . E8 ECE1FBFF CALL storylin.0040601B ; <== Encrypt
 ----- *TraceInto I* -----
 00406025 |. E8 02280600 CALL storylin.0046882C ; <== Trace Into
 ----- *TraceInto II* -----
 00468890 |. E8 86D00100 CALL storylin.0048591B ; <== Trace Into
 ----- *TraceInto III* -----

- Quá trình chuyển đổi chuỗi User nhập về dạng LowerCase :

00485975 |> /8A07 /MOV AL,BYTE PTR DS:[EDI]
 00485977 |.|3C 20 |CMP AL,20
 00485979 |.|74 0E |JE SHORT storylin.00485989
 0048597B |.|0FBEC0 |MOVSX EAX,AL
 0048597E |.|50 |PUSH EAX ;/c

```

0048597F |. |FF15 E8A45A00 |CALL DWORD PTR DS:[<&MSVCRT(tolower>)] ; \tolower
00485985 |. |8806      |MOV BYTE PTR DS:[ESI],AL
00485987 |. |59       |POP ECX
00485988 |. |46       |INC ESI
00485989 |> |47       |INC EDI
0048598A |. |381F      |CMP BYTE PTR DS:[EDI],BL
0048598C |.^|75 E7      |JNZ SHORT storylin.00485975

```

- Kết hợp với chuỗi mặc định “*Anthemion Software ScriptTracks*” :

```

0048598E |> |FF75 EC      PUSH [LOCAL.5] ; /src
00485991 |. |8D85 84FDFFFF LEA EAX,[LOCAL.159] ; |
00485997 |. |881E      MOV BYTE PTR DS:[ESI],BL ; |
00485999 |. |50       PUSH EAX ; |dest
0048599A |. |E8 55D91000 CALL <JMP.&MSVCRT.strcat> ; \ strcat

```

- Quá trình tạo chuỗi MD5Hash :

```

004859AF |. |E8 143D0000 CALL storylin.004896C8 ; <== MD5Start
004859B4 |. |8D85 84FDFFFF LEA EAX,[LOCAL.159]
004859BA |. |50       PUSH EAX ; /s
004859BB |. |E8 A0D81000 CALL <JMP.&MSVCRT.strlen> ; \ strlen
004859C0 |. |50       PUSH EAX
004859C1 |. |8D85 84FDFFFF LEA EAX,[LOCAL.159]
004859C7 |. |50       PUSH EAX
004859C8 |. |8D45 84      LEA EAX,[LOCAL.31]
004859CB |. |50       PUSH EAX
004859CC |. |E8 1F3D0000 CALL storylin.004896F0 ; <== MD5Update
004859D1 |. |8D45 84      LEA EAX,[LOCAL.31]
004859D4 |. |50       PUSH EAX
004859D5 |. |8D45 DC      LEA EAX,[LOCAL.9]
004859D8 |. |50       PUSH EAX
004859D9 |. |E8 B23D0000 CALL storylin.00489790 ; <== MD5Finished

```

- Chuỗi MD5Hash được tạo thành là 16 bytes (16 ký tự). Ta chia làm 4 đoạn, mỗi đoạn 4 bytes . Quá trình tạo chuỗi Serial thực chỉ là quá trình chuyển đổi ba đoạn đầu tiên của chuỗi MD5Hash thành chuỗi Serial theo định dạng “XXXXXXXX-YYYYYYYY-ZZZZZZZZ” :

```

004859E3 |. |8D75 DC      LEA ESI,[LOCAL.9]
004859E6 |> |33C9      /XOR ECX,ECX
004859E8 |. |33C0      |XOR EAX,EAX
004859EA |> |0FB61406    /|MOVZX EDX,BYTE PTR DS:[ESI+EAX]
004859EE |. |C1E1 08    ||SHL ECX,8
004859F1 |. |03CA      ||ADD ECX,EDX
004859F3 |. |40       ||INC EAX
004859F4 |. |83F8 04    ||CMP EAX,4
004859F7 |.^|7C F1      ||JL SHORT storylin.004859EA
004859F9 |. |A1 B4276200  ||MOV EAX,DWORD PTR DS:[6227B4]
004859FE |. |83FF 02    ||CMP EDI,2
00485A01 |. |7D 32      ||JGE SHORT storylin.00485A35
00485A03 |. |8945 10    ||MOV [ARG.3],EAX
00485A06 |. |51       ||PUSH ECX
00485A07 |. |8D45 10    ||LEA EAX,[ARG.3]
00485A0A |. |68 7CE46100  ||PUSH storylin.0061E47C ; ASCII "%08lX-"
00485A0F |. |50       ||PUSH EAX

```

```

00485A10 |. C645 FC 03 |MOV BYTE PTR SS:[EBP-4],3
00485A14 |. E8 C75F0200 |CALL storylin.004AB9E0
00485A19 |. 8B45 10 |MOV EAX,[ARG.3]
00485A1C |. 83C4 0C |ADD ESP,0C
00485A1F |. 8B48 F8 |MOV ECX,DWORD PTR DS:[EAX-8]
00485A22 |. 50 |PUSH EAX ; /Arg2
00485A23 |. 51 |PUSH ECX ; |Arg1
00485A24 |. 8D4D 0C |LEA ECX,[ARG.2] ; |
00485A27 |. E8 34510200 |CALL storylin.004AAB60 ; \storylin.004AAB60
00485A2C |. C645 FC 02 |MOV BYTE PTR SS:[EBP-4],2
00485A30 |. 8D4D 10 |LEA ECX,[ARG.3]
00485A33 |. EB 30 |JMP SHORT storylin.00485A65
00485A35 |> 8945 10 |MOV [ARG.3],EAX
00485A38 |. 51 |PUSH ECX
00485A39 |. 8D45 10 |LEA EAX,[ARG.3]
00485A3C |. 68 74E46100 |PUSH storylin.0061E474 ; ASCII "%08lX"
00485A41 |. 50 |PUSH EAX
00485A42 |. C645 FC 04 |MOV BYTE PTR SS:[EBP-4],4
00485A46 |. E8 955F0200 |CALL storylin.004AB9E0
00485A4B |. 8B45 10 |MOV EAX,[ARG.3]
00485A4E |. 83C4 0C |ADD ESP,0C
00485A51 |. 8B48 F8 |MOV ECX,DWORD PTR DS:[EAX-8]
00485A54 |. 50 |PUSH EAX ; /Arg2
00485A55 |. 51 |PUSH ECX ; |Arg1
00485A56 |. 8D4D 0C |LEA ECX,[ARG.2] ; |
00485A59 |. E8 02510200 |CALL storylin.004AAB60 ; \storylin.004AAB60
00485A5E |. C645 FC 02 |MOV BYTE PTR SS:[EBP-4],2
00485A62 |. 8D4D 10 |LEA ECX,[ARG.3]
00485A65 |> E8 7FB6F7FF |CALL storylin.004010E9
00485A6A |. 47 |INC EDI
00485A6B |. 83C6 04 |ADD ESI,4
00485A6E |. 83FF 03 |CMP EDI,3
00485A71 |.^ 0F8C 6FFFFFFF |JL storylin.004859E6
----- == TraceInto III ==
----- == TraceInto II ==
----- == TraceInto I ==

```

- Như vậy quá trình mã hoá của SoftWare này chỉ là quá trình tạo chuỗi MD5Hash dựa trên sự kết hợp giữa U nhập và một chuỗi mặc định .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : D9A8DDCB-EAA3ABE7-E6BB0D18

III – KeyGen :

/Section 1/- Thực hiện CharLower(User), và kết hợp với chuỗi mặc định .

/Section 2/- Tính **MD5hash** .

/Section 2/- Kết hợp 3 đoạn đầu tiên của chuỗi này theo định dạng “%08X-%08X-%08X”

IV – End of Tut :

- Finished – **August 10, 2004**

Reverse Engineering Association

SoftWare

Homepage	:	http://www.ababasoft.com
Production	:	Serge Mikhailov
SoftWare	:	TextIndexer 4.27
Copyright by	:	Copyright © 2004 Serge Mikhailov. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N / A
Language	:	Microsoft Visual C++ 7.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N / A
Request	:	Correct Serial / KeyGen

TextIndexer 4.27

TextIndexer is a Personal Information Manager, Word Processor and Internet Search Utility. TextIndexer uses an intuitive, tree interface to allow hierarchical storage of information. It allows to categorize text and picture items and to display it in a particular category.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 7.0**.
- Nhập thử User và Fake Serial, ta nhận được thông báo "Wrong Reg code". Tìm thông báo và ta gặp tại địa chỉ :
004FA568 . BA CCA64F00 MOV EDX,TextInde.004FA6CC ; ASCII "Wrong Reg code"
- Dò ngược lên trên ta đặt BreakPoint tại :
004FA463 . E8 D8BDF7FF CALL TextInde.00476240 ; <== Set BreakPoint here

II – Cracking :

- Quá trình kiểm tra chiều dài User và Serial được thực hiện :

004FA468 . 837D F4 00 CMP DWORD PTR SS:[EBP-C],0	; <== U must be input
004FA46C . 75 0C JNZ SHORT TextInde.004FA47A	
004FA46E . BA 84A64F00 MOV EDX,TextInde.004FA684	; ASCII "Please enter User Name"
004FA473 . 8BC3 MOV EAX,EBX	
004FA475 . E8 F6BDF7FF CALL TextInde.00476270	
004FA47A > 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]	
004FA47D . 8B83 1C040000 MOV EAX,DWORD PTR DS:[EBX+41C]	
004FA483 . E8 B8BDF7FF CALL TextInde.00476240	
004FA488 . 837D F0 00 CMP DWORD PTR SS:[EBP-10],0	; <== Serial must be input
004FA48C . 75 0C JNZ SHORT TextInde.004FA49A	
004FA48E . BA A4A64F00 MOV EDX,TextInde.004FA6A4	; ASCII "Please enter Registration Code"
004FA493 . 8BC3 MOV EAX,EBX	
004FA495 . E8 D6BDF7FF CALL TextInde.00476270	
- Trace tiếp ta đến quá trình mã hóa :

004FA528 . E8 73F0F0FF CALL TextInde.004095A0	; <== Convert S(DEC) to S(HEX) : Value
004FA52D . B9 11000000 MOV ECX,11	

```

004FA532 . 99      CDQ
004FA533 . F7F9    IDIV ECX
004FA535 . 8945 E0  MOV DWORD PTR SS:[EBP-20],EAX      ; <== Value = Value / 0x11
004FA538 . DB45 E0  FILD DWORD PTR SS:[EBP-20]
004FA53B . E8 8488F0FF CALL TextInde.00402DC4
004FA540 . 8BF0    MOV ESI,EAX      ; <== Value
004FA542 . C1E6 04  SHL ESI,4      ; <== Temp = Value * 0x10
004FA545 . 03F0    ADD ESI,EAX      ; <== Temp = Temp + Value
004FA547 > C705 40235000>MOV DWORD PTR DS:[502340],64
004FA551 . 8D55 DC  LEA EDX,DWORD PTR SS:[EBP-24]
004FA554 . 8BC6    MOV EAX,ESI
004FA556 . E8 D9EEF0FF CALL TextInde.00409434
004FA55B . 8B45 DC  MOV EAX,DWORD PTR SS:[EBP-24]      ; <== Temp
004FA55E . 8B55 F8  MOV EDX,DWORD PTR SS:[EBP-8]      ; <== Value
004FA561 . E8 EAA7F0FF CALL TextInde.00404D50      ; <== Two value must be equal

```

- Từ quá trình trên ta diễn tả lại như sau :

$$((Value/17) * 16) + (Value/17) = Value \Leftrightarrow 17 * Value = 17 * Value$$

- Hay nói một cách khác, Value phải là một bội số của 17 (0x11)

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : N/A
User : VHT-cRaCkErS	Serial : N/A

Serial : 2178125 or 2362201

III – KeyGen :

/Section 1/- Tạo chuỗi ngẫu nhiên gồm 5 ký tự : RanValue
 /Section 2/- Serial = RanValue * 0x11

IV – SourceCode (VC++) :

```

DWORD Serial;
Serial = GetTickCount()%1000000 * 0x11;
SetDlgItemInt(IDC_Serial,Serial);

```

V – End of Tut :

- Finished – 06/07/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.isbister.com
Production :	Chaos Software Group, Inc.
SoftWare :	Time & Chaos 6.0.2.7
Copyright by :	Copyright © 2002-2004 Chaos Software Group, Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10

Unpack : Manual
 Request : Correct Serial / KeyGen

Time & Chaos 6.0.2.7

Time & Chaos is a type of software program known as a Personal Information Manager or PIM. As a time management tool, it can help you maintain a phonebook with every phone number or email address you need, can keep a schedule of all your appointments and can give you a prioritized to do list where things will continue to display day after day until you complete them.

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo**. Sau khi UnPACK biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Nhập thử Fake Serial, ta nhận được thông báo "**Please try again, the User Name or User Code that was entered is incorrect.**" Dùng phương pháp tìm chuỗi bằng STACK ta tìm được thông báo này ở địa chỉ : 005C051F B8 90065C00 MOV EAX,dump_.005C0690 ; ASCII "Please try again, the User Name or User Code that was entered is incorrect."
- Dò ngược lên trên và đặt BreakPoint tại :
005C03CE E8 357FEBFF CALL dump_.00478308 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống một đoạn ta đến :

```
005C042C 8B45 E8      MOV EAX,DWORD PTR SS:[EBP-18]    ; <== User
005C042F 5A          POP EDX                            ; <== Fake Serial
005C0430 E8 6723F6FF  CALL dump_.0052279C            ; <== Encrypt and Compare
005C0435 3C 01        CMP AL,1                          ; <== if Correct AL = 1
005C0437 0F85 D7000000 JNZ dump_.005C0514           ; <== then CONGRAT !!!!
```

- Dùng F7 trace into ta đến quá trình kiểm tra chiều dài U và S :

```
005227D8 837D FC 00  CMP DWORD PTR SS:[EBP-4],0       ; <== LenU != 0
005227DC 0F84 AD000000 JE dump_.0052288F            ; <== U atleast 1 char
005227E2 837D F8 00  CMP DWORD PTR SS:[EBP-8],0       ; <== LenS != 0
005227E6 0F84 A3000000 JE dump_.0052288F            ; <== S atleast 1 char
```

- Kế đó chương trình kiểm tra ký tự thứ 6 của chuỗi Serial nhập với một ký tự mặc định :

```
0052280A 8B45 EC      MOV EAX,DWORD PTR SS:[EBP-14]    ; <== S[5]
0052280D BA D0285200 MOV EDX,dump_.005228D0          ; <== Default Chat : "Q"
00522812 E8 A92AEEFF CALL dump_.004052C0            ; <== They must be same
```

- Sau đó, chương trình tiến hành cắt 4 ký tự ở các vị trí [1][2][3][4], và chuyển chuỗi này sang giá trị dạng HEX tương ứng (ví dụ, chuỗi là "1234" thì chuyển thành giá trị "4D2") . Chú ý, 4 ký tự này bắt buộc phải là các số (0 – 9), vì do thuật toán chuyển hóa từ chuỗi sang giá trị HEX của chương trình chỉ đúng với các số

```
0052282B B9 04000000 MOV ECX,4                      ; <== Get 4 charts
00522830 BA 02000000 MOV EDX,2                      ; <== from second chart
00522835 8B45 F8      MOV EAX,DWORD PTR SS:[EBP-8]    ; <== of Serial
00522838 E8 972BEEFF CALL dump_.004053D4            ; <== Cutting
0052283D 8B45 E4      MOV EAX,DWORD PTR SS:[EBP-1C]
```

```
00522840 E8 BF78EEFF CALL dump_.0040A104            ; <== Convert DEC to HEX value
```

- Quá trình mã hoá và so sánh thứ hai diễn ra như sau :

```
0052286C BA 01000000 MOV EDX,1                      ; <== i = 1
```

```
00522871 8B4D FC      MOV ECX,DWORD PTR SS:[EBP-4]    ; <== U
```

```
00522874 0FB64C11 FF  MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== U[i-1]
```

```

00522879 014D F0      ADD DWORD PTR SS:[EBP-10],ECX      ; <== Value = Value + U[i]
0052287C 42          INC EDX                          ; <== i++
0052287D 48          DEC EAX                          ; <== LenU --
0052287E ^ 75 F1      JNZ SHORT dump_.00522871       ; <== Loop until LenU = 0
00522880 81EB A7080000 SUB EBX,8A7                 ; <== HEXValue = HEXValue - 0x8A7
00522886 3B5D F0      CMP EBX,DWORD PTR SS:[EBP-10]    ; <== Must be Equal qith
Value
00522889 75 04      JNZ SHORT dump_.0052288F       ; <== if Equal
0052288B C645 F7 01  MOV BYTE PTR SS:[EBP-9],1       ; <== AL = 1

```

- Chú ý :

/+/- Chương trình không đề cập đến các ký tự của chuỗi Serial từ thứ [6] trở đi, nên chương trình có thể có chuỗi Serial dài tùy thích. Trong đó ký tự thứ sáu của chuỗi Serial phải là “Q”

/+/- Từ lập luận của quá trình mã hoá thứ hai có thể suy ngược lại rằng để 4 ký tự của chuỗi Serial thoái điều kiện thì có thể tính bằng cách : Value + 0x8A7 . Tuy nhiên, nếu giá trị Value quá lớn thì dẫn đến giá trị của phép cộng này vượt quá 5 chữ số (ở dạng DEC). Vì vậy cần không chế chiều dài của chuỗi U nhập .

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 13470Q or 93470Q

III – KeyGen :

- /Section 1/- Không chế chiều dài chuỗi U từ (1 – 60) ký tự .
- /Section 2/- Tính cộng dồn giá trị các ký tự của chuỗi U và cộng thêm giá trị 0x8A7.
- /Section 3/- Chuyển giá trị này sang dạng DEC theo định dạng "%04i".
- /Section 4/- Nối các giá trị này lại với nhau, trong đó ký tự thứ [5] phải là “Q”.

IV – End of Tut :

- Finished – 28/07/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.optimussw.com/trashit
Production :	Optimus Software
SoftWare :	Trash it! 1.80
Copyright by :	Copyright © 2004 Optimus Software. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual Basic 5.0 / 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Trash it! 1.80

Enter Trash it! Trash it! will delete all files installed by a program, remove all subdirectories created by it and delete all changes made by the program in your various system files and as well as in your registry.

As if that isn't impressive enough, Trash it! also comes with an in-built scheduler that runs periodically in the background and deletes files that you don't need, such as Windows Log files, Windows temporary files, files in the \Windows\temp folder, files in your Recycle Bin, etc. If you use Trash it!, you free your hard disk of files that you don't really need, leaving more space for the ones that you really do. You can even customize the list of files or folders that you want to delete, and you can configure how often you want the Scheduler to run. The default options that come pre-built with the Scheduler works fine for most users though, and can free up quite a lot of free space by deleting files that you don't really need. The Scheduler runs in the system tray, and starts, cleans up your hard drive, and then exits, without your having to press a single button or a single mouse click. It's like magically getting more free space! So even if you are a complete novice and don't want to use the uninstallation features of Trash it!, you will find Trash it! useful. Trash it! can free up hard disk space without your having to do anything!

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual Basic 5.0 / 6.0**.
- Nhập thử User và Fake Serial, ta nhận được thông báo "Changes will be applied after application next run" Như vậy ta không thể tìm chuỗi thông báo đúng hay sai trong trường hợp này .

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial. Trace xuống một đoạn ta đến quá trình kiểm tra chiều dài chuỗi Serial nhập :

```

00479DB6 . FFD6    CALL ESI          ; <&MSVBVM60.__vbaLenBstr>
00479DB8 . 8B4D E4    MOV ECX,DWORD PTR SS:[EBP-1C]   ; <== Serial
00479DBB . 8BD8    MOV EBX,EAX        ; <== Len.S
00479DBD . BF 0A000000  MOV EDI,0A      ; <== Temp = 0xA
00479DC2 . 2BDF    SUB EBX,EDI        ; <== Temp = Len.S - Temp
00479DC4 . F7DB    NEG EBX          ; <== Negate : Temp = 0 - Temp
00479DC6 . 1BDB    SBB EBX,EBX        ; <== Substrac with borrow
00479DC8 . 51     PUSH ECX          ; <== Serial
00479DC9 . 43     INC EBX          ; <== Temp ++
00479DCA . FFD6    CALL ESI
00479DCC . 33D2    XOR EDX,EDX
00479DCE . 85C0    TEST EAX,EAX
00479DD0 . 0F9EC2   SETLE DL
00479DD3 . 0BDA    OR EBX,EDX
00479DD5 . 0F85 85000000 JNZ Trash_it.00479E60

```

- Qua đoạn CODE trên, ta tính được chiều dài của Serial phải là 10 ký tự .

- Trace tiếp một đoạn và ta đến :

```

00479E91 . E8 8A4AFDFF CALL Trash_it.0044E920      ; <== Trace Into
- Dùng F7 trace Into để đến đoạn CODE mã hoá . Quá trình mã hoá này đầu tiên kiểm tra tính hợp lý của các ký tự . Ba ký tự đầu tiên và 3 ký tự cuối cùng của chuỗi Serial phải là SỐ ( NUMBER ) :
0044EC2D . 8B1D F8104000 MOV EBX,DWORD PTR DS:<&MSVBVM60.#561> ;
MSVBVM60 rtcIsNumeric
0044EC33 . B8 08400000 MOV EAX,4008
0044EC38 . 8D55 EC    LEA EDX,DWORD PTR SS:[EBP-14]
0044EC3B . 8945 80    MOV DWORD PTR SS:[EBP-80],EAX
0044EC3E . 8985 70FFFFFF MOV DWORD PTR SS:[EBP-90],EAX
0044EC44 . 8985 60FFFFFF MOV DWORD PTR SS:[EBP-A0],EAX
0044EC4A . 83C4 0C    ADD ESP,0C
0044EC4D . 8955 88    MOV DWORD PTR SS:[EBP-78],EDX
0044EC50 . 8D85 70FFFFFF LEA EAX,DWORD PTR SS:[EBP-90]
0044EC56 . 8D4D E8    LEA ECX,DWORD PTR SS:[EBP-18]

```

```

0044EC59 . 8D55 D8    LEA EDX,DWORD PTR SS:[EBP-28]
0044EC5C . 50        PUSH EAX
0044EC5D . 898D 78FFFFFF MOV DWORD PTR SS:[EBP-88],ECX
0044EC63 . 8995 68FFFFFF MOV DWORD PTR SS:[EBP-98],EDX
0044EC69 . FFD3      CALL EBX          ; <&MSVBVM60.#561>
0044EC6B . 66:8BD0   MOV DX,AX
0044EC6E . 66:F7DA   NEG DX
0044EC71 . 8D4D 80    LEA ECX,DWORD PTR SS:[EBP-80]
0044EC74 . 1BD2      SBB EDX,EDX
0044EC76 . 42        INC EDX
0044EC77 . 51        PUSH ECX
0044EC78 . 8995 50FFFFFF MOV DWORD PTR SS:[EBP-B0],EDX
0044EC7E . FFD3      CALL EBX
0044EC80 . 8B9D 50FFFFFF MOV EBX,DWORD PTR SS:[EBP-B0]
0044EC86 . 66:F7D8   NEG AX
0044EC89 . 8D95 60FFFFFF LEA EDX,DWORD PTR SS:[EBP-A0]
0044EC8F . 52        PUSH EDX
0044EC90 . 1BC0      SBB EAX,EAX
0044EC92 . 40        INC EAX
0044EC93 . 0BD8      OR EBX,EAX
0044EC95 . F7DB      NEG EBX
0044EC97 . 1BDB      SBB EBX,EBX
0044EC99 . F7DB      NEG EBX
0044EC9B . FF15 F8104000 CALL DWORD PTR DS:<&MSVBVM60.#561> ;
SVBVM60 rtcIsNumeric

```

- Sau đó, ba ký tự đầu tiên được đếm chia cho ba ký tự cuối cùng . Sau đó giá trị của phép chia này được làm tròn và đếm so sánh với hai số 2 và 3 . Tuy theo kết quả là bằng 2 hay 3 mà ta có quá trình mã hoá tiếp tục khác nhau :

```

0044ECB9 . FFD3      CALL EBX          ; <&MSVBVM60.__vbaI4Str>
0044ECBB . 8B4D E8    MOV ECX,DWORD PTR SS:[EBP-18]       ; <== 3 last number
0044ECBE . 8985 4CFFFFFF MOV DWORD PTR SS:[EBP-B4],EAX     ; <== 3 first number
0044ECC4 . DB85 4CFFFFFF FILD DWORD PTR SS:[EBP-B4] ; <== convert to
FloatingPointNumber
0044ECCA . 51        PUSH ECX         ; <== 3 last number
0044ECCB . DD9D 44FFFFFF FSTP QWORD PTR SS:[EBP-BC] ; <== Store 3 first number to ST(7)
0044ECD1 . FFD3      CALL EBX
0044ECD3 . 8985 40FFFFFF MOV DWORD PTR SS:[EBP-C0],EAX     ; <== 3 last number
0044ECD9 . DB85 40FFFFFF FILD DWORD PTR SS:[EBP-C0] ; <== convert to
FloatingPointNumber
0044ECDF . DD9D 38FFFFFF FSTP QWORD PTR SS:[EBP-C8] ; <== Store 3 last number to ST(7)
0044ECE5 . DD85 44FFFFFF FLD QWORD PTR SS:[EBP-BC] ; <== load 3 first number to ST(0)
0044ECEB . 833D 00D04700>CMP DWORD PTR DS:[47D000],0
0044ECF2 . 75 08    JNZ SHORT Trash_it.0044ECFC
0044ECF4 . DCB5 38FFFFFF FDIV QWORD PTR SS:[EB]    ; <== Mod = Roundup(3 first / 3 last)
0044ECFA . EB 11    JMP SHORT Trash_it.0044ED0D
0044ECFC > FFB5 3CFFFFFF PUSH DWORD PTR SS:[EBP-C4]
0044ED02 . FFB5 38FFFFFF PUSH DWORD PTR SS:[EBP-C8]
0044ED08 . E8 876FFBF CALL <JMP.&MSVBVM60._adj_fdiv_m64>
0044ED0D > DFE0      FSTSW AX
0044ED0F . A8 0D      TEST AL,0D
0044ED11 . 0F85 07040000 JNZ Trash_it.0044F11E

```

0044ED17 . FF15 FC114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFpI4>];
 MSVBVM60.__vbaFpI4
 - Nếu kết quả là tròn của phép chia là bằng 2 ta có quá trình mã hoá :

0044ED1D . 83F8 02 CMP EAX,2 ; <== Value = Roundup(3 first / 3 last)
 0044ED20 . 0F85 F9000000 JNZ Trash_it.0044EE1F ; <== if Equal continue check
 0044ED26 . 8B55 D8 MOV EDX,DWORD PTR SS:[EBP-28] ; <== Two middle number of Serial
 0044ED29 . 52 PUSH EDX ; <== Convert to HEX value : Value
 0044ED2A . FF15 58114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaI2Str>];
 MSVBVM60.__vbaI2Str
 0044ED30 . 66:3D 1E00 CMP AX,1E ; <== Value
 0044ED34 . 7E 08 JLE SHORT Trash_it.0044ED3E
 0044ED36 . 8B4D 0C MOV ECX,DWORD PTR SS:[EBP+C]
 0044ED39 . 66:C701 FFFF MOV WORD PTR DS:[ECX],0FFF
 0044ED3E > 66:05 0100 ADD AX,1 ; <== Value = Value + 1
 0044ED42 . 0F80 DB030000 JO Trash_it.0044F123
 0044ED48 . 50 PUSH EAX ; <== Convert to number
 0044ED49 . FF15 04104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrI2>];
 MSVBVM60.__vbaStrI2
 0044ED4F . 8BD0 MOV EDX,EAX
 0044ED51 . 8D4D DC LEA ECX,DWORD PTR SS:[EBP-24]
 0044ED54 . FFD6 CALL ESI
 0044ED56 . 8B55 DC MOV EDX,DWORD PTR SS:[EBP-24] ; <== number
 0044ED59 . 52 PUSH EDX ; <== Get length of this number
 0044ED5A . FF15 2C104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaLenBs>];
 MSVBVM60.__vbaLenBstr
 0044ED60 . 83F8 01 CMP EAX,1 ; <== one or two chart
 0044ED63 . 75 1A JNZ SHORT Trash_it.0044ED7F ; <== if ONE
 0044ED65 . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24]
 0044ED68 . 8B1D 58104000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaSt>];
 MSVBVM60.__vbaStrCat
 0044ED6E . 68 885B4100 PUSH Trash_it.00415B88
 0044ED73 . 50 PUSH EAX
 0044ED74 . FFD3 CALL EBX ; <&MSVBVM60.__vbaStrCat>
 0044ED76 . 8BD0 MOV EDX,EAX
 0044ED78 . 8D4D DC LEA ECX,DWORD PTR SS:[EBP-24]
 0044ED7B . FFD6 CALL ESI
 0044ED7D . EB 06 JMP SHORT Trash_it.0044ED85 ; <== else if TWO
 0044ED7F > 8B1D 58104000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaSt>];
 MSVBVM60.__vbaStrCat
 0044ED85 > 8B4D EC MOV ECX,DWORD PTR SS:[EBP-14] ; <== 3 first number
 0044ED88 . 51 PUSH ECX
 0044ED89 . 68 C0844100 PUSH Trash_it.004184C0 ; <== Default chart : c
 0044ED8E . C785 58FFFFFF>MOV DWORD PTR SS:[EBP-A8],1
 0044ED98 . FFD3 CALL EBX ; MSVBVM60.__vbaStrCat
 0044ED9A . 8BD0 MOV EDX,EAX ; <== String after ConCat : Str
 0044ED9C . 8D4D C0 LEA ECX,DWORD PTR SS:[EBP-40]
 0044ED9F . FFD6 CALL ESI
 0044EDA1 . 8B55 DC MOV EDX,DWORD PTR SS:[EBP-24] ; <== Number
 0044EDA4 . 50 PUSH EAX ; <== String after ConCat : Str
 0044EDA5 . 52 PUSH EDX ; <== Number
 0044EDA6 . FFD3 CALL EBX ; MSVBVM60.__vbaStrCat
 0044EDA8 . 8BD0 MOV EDX,EAX ; <== String after ConCat : Str
 0044EDAA . 8D4D BC LEA ECX,DWORD PTR SS:[EBP-44] ; <== Default chart : f

```

0044EDAD . FFD6      CALL ESI
0044EDAF . 50        PUSH EAX ; <== String after ConCat : Str
0044EDB0 . 68 C8844100 PUSH Trash_it.004184C8 ; <== Default chart : f
0044EDB5 . FFD3      CALL EBX ; MSVBVM60.__vbaStrCat
0044EDB7 . 8BD0      MOV EDX,EAX
0044EDB9 . 8D4D B8    LEA ECX,DWORD PTR SS:[EBP-48]
0044EDBC . FFD6      CALL ESI ; <== String after ConCat : Str
0044EDBE . 50        PUSH EAX
0044EDBF . 8B45 E8    MOV EAX,DWORD PTR SS:[EBP-18] ; <== 3 last number
0044EDC2 . 50        PUSH EAX
0044EDC3 . FFD3      CALL EBX ; MSVBVM60.__vbaStrCat
0044EDC5 . 8BD0      MOV EDX,EAX ; <== String after ConCat :
Serial

```

- Ta nhận thấy ngay không thể tạo ra SERIAL đối với trường hợp bằng 2 được. Vì chuỗi Serial của chương trình tạo ra luôn lấy hai giá trị ở giữa cộng thêm với 1 để tạo ra chuỗi Serial thực.

- Xét trường hợp kết quả là tròn của phép chia bằng 3 :

```

0044EE1A ./E9 75020000 JMP Trash_it.0044F094 ; <== if NOT EQUAL go another
check

```

```

0044EE1F >|83F8 03    CMP EAX,3 ; <== Value = 3 first / 3 last
0044EE22 .|0F85 AE010000 JNZ Trash_it.0044EFD6 ; <== if NOT EQUAL jmp NAG
0044EE28 .|8B4D 08    MOV ECX,DWORD PTR SS:[EBP+8]
0044EE2B .|6A 01      PUSH 1
0044EE2D .|8D45 80    LEA EAX,DWORD PTR SS:[EBP-80]
0044EE30 .|66:C701 FFFF MOV WORD PTR DS:[ECX],0FFF
0044EE35 .|50        PUSH EAX
0044EE36 .|8D4D A0    LEA ECX,DWORD PTR SS:[EBP-60]
0044EE39 .|8D55 D8    LEA EDX,DWORD PTR SS:[EBP-28]
0044EE3C .|51        PUSH ECX
0044EE3D .|8955 88    MOV DWORD PTR SS:[EBP-78],EDX
0044EE40 .|C745 80 08400>MOV DWORD PTR SS:[EBP-80],4008
0044EE47 .|FF15 10124000 CALL DWORD PTR DS:[<&MSVBVM60.#617>] ; MSVBVM60.rtcLeftCharVar

```

```

0044EE4D .|8B1D 28104000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaSt>; MSVBVM60.__vbaStrVarMove

```

```

0044EE53 .|8D55 A0    LEA EDX,DWORD PTR SS:[EBP-60]
0044EE56 .|52        PUSH EDX

```

```

0044EE57 .|FFD3      CALL EBX ; <=&MSVBVM60.__vbaStrVarMove>

```

```

0044EE59 .|8BD0      MOV EDX,EAX

```

```

0044EE5B .|8D4D DC    LEA ECX,DWORD PTR SS:[EBP-24]

```

```

0044EE5E .|FFD6      CALL ESI

```

```

0044EE60 .|8D4D A0    LEA ECX,DWORD PTR SS:[EBP-60]

```

```

0044EE63 .|FF15 20104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeV>; MSVBVM60.__vbaFreeVar

```

```

0044EE69 .|6A 01      PUSH 1

```

```

0044EE6B .|8D4D 80    LEA ECX,DWORD PTR SS:[EBP-80]

```

```

0044EE6E .|51        PUSH ECX

```

```

0044EE6F .|8D55 A0    LEA EDX,DWORD PTR SS:[EBP-60]

```

```

0044EE72 .|8D45 D8    LEA EAX,DWORD PTR SS:[EBP-28]

```

```

0044EE75 .|52        PUSH EDX

```

```

0044EE76 .|8945 88    MOV DWORD PTR SS:[EBP-78],EAX

```

```

0044EE79 .|C745 80 08400>MOV DWORD PTR SS:[EBP-80],4008

```

```

0044EE80 . |FF15 20124000 CALL DWORD PTR DS:[<&MSVBVM60.#619>] ; 
MSVBVM60.rtcRightCharVar
0044EE86 . |8D45 A0    LEA EAX,DWORD PTR SS:[EBP-60]
0044EE89 . |50      PUSH EAX
0044EE8A . |FFD3    CALL EBX
0044EE8C . |8BD0    MOV EDX,EAX
0044EE8E . |8D4D D4    LEA ECX,DWORD PTR SS:[EBP-2C]
0044EE91 . |FFD6    CALL ESI
0044EE93 . |8D4D A0    LEA ECX,DWORD PTR SS:[EBP-60]
0044EE96 . |FF15 20104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeV>];
MSVBVM60.__vbaFreeVar
0044EE9C . |8B4D EC    MOV ECX,DWORD PTR SS:[EBP-14]           ; <== 3 first number
0044EE9F . |8B1D 58104000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaSt>];
MSVBVM60.__vbaStrCat
0044EEA5 . |51      PUSH ECX           ; <== 3 first number
0044EEA6 . |68 D0844100 PUSH Trash_it.004184D0           ; <== Default chart : d
0044EEAB . |C785 58FFFFFF>MOV DWORD PTR SS:[EBP-A8],1
0044EEB5 . |FFD3    CALL EBX           ; <&MSVBVM60.__vbaStrCat>
0044EEB7 . |8BD0    MOV EDX,EAX           ; <== String after ConCat : Str
0044EEB9 . |8D4D C0    LEA ECX,DWORD PTR SS:[EBP-40]
0044EEBC . |FFD6    CALL ESI
0044EEBE . |8B55 D8    MOV EDX,DWORD PTR SS:[EBP-28] ; <== two middle number of Serial
0044EEC1 . |50      PUSH EAX
0044EEC2 . |52      PUSH EDX
0044EEC3 . |FFD3    CALL EBX           ; MSVBVM60.__vbaStrCat
0044EEC5 . |8BD0    MOV EDX,EAX           ; <== String after ConCat : Str
0044EEC7 . |8D4D BC    LEA ECX,DWORD PTR SS:[EBP-44]
0044EECA . |FFD6    CALL ESI
0044EECC . |50      PUSH EAX           ; <== String after ConCat : Str
0044EECD . |68 D8844100 PUSH Trash_it.004184D8           ; <== Default chart : j
0044EED2 . |FFD3    CALL EBX           ; MSVBVM60.__vbaStrCat
0044EED4 . |8BD0    MOV EDX,EAX           ; <== String after ConCat : Str
0044EED6 . |8D4D B8    LEA ECX,DWORD PTR SS:[EBP-48]
0044EED9 . |FFD6    CALL ESI
0044EEDB . |50      PUSH EAX
0044EEDC . |8B45 E8    MOV EAX,DWORD PTR SS:[EBP-18]       ; <== 3 last number
0044EEDF . |50      PUSH EAX
0044EEE0 . |FFD3    CALL EBX           ; MSVBVM60.__vbaStrCat
0044EEE2 . |8BD0    MOV EDX,EAX           ; <== String after ConCat : Serial
- Như vậy chuỗi Serial thực tạo thành bằng cách thêm hai ký tự mặc định “d” và “j” vào thay thế các ký tự ở vị trí Serial[3] và Serial[6].
- Tuy nhiên, đến đây chưa phải là đã hoàn thiện. Ta còn một quá trình kiểm tra giá trị của hai ký tự ở giữa :
0044EF53 . FFD6    CALL ESI           ; <== first number
0044EF55 . 66:8B8D 36FFF>MOV CX,WORD PTR SS:[EBP-CA]       ; <== Second number
0044EF5C . 66:2BC1    SUB AX,CX           ; <== Value = First - Second
0044EF5F . 0F80 BE010000 JO Trash_it.0044F123
0044EF65 . 66:3D 0300  CMP AX,3           ; <== Value == 3 ?
0044EF69 . 0F84 25010000 JE Trash_it.0044F094       ; <== if YES --> Congrat !!
/*/*/* - SERIAL tương ứng với USER :
User : REA-cRaCkErS          Serial : N/A
User : VHT-cRaCkErS          Serial : N/A

```

III – KeyGen :

- /Section 1/- Tạo số ngẫu nhiên gồm 3 chữ số : 3 số cuối cùng : BBB
- /Section 2/- Tạo ba số đầu tiên bằng : AAA = BBB x 3
- /Section 3/- Tạo hai số ngẫu nhiên với điều kiện số đầu tiên lớn hơn số thứ hai 3 đơn vị
- /Section 4/- Kết hợp các số lại và thêm hai ký tự mặc định vào gi trị thứ [3] và [6]

IV – SourceCode (VC++) :

```

char reaSerial[20]={0};
int LastRanValue=0, FirstRanValue=0, MiddleRanValue=0;

srand( (unsigned)time( NULL ) );
LastRanValue = 100 + rand() % 233;
FirstRanValue = LastRanValue * 0x3;
SecondMiddleRanValue = rand() % 6;
FirstMiddleRanValue = SecondMiddleRanValue + 0x3;
wsprintf(reaSerial,"% id%i%ij% i",FirstRanValue,FirstMiddleRanValue,SecondMiddleRan
Value,LastRanValue);

SetDlgItemText(IDC_Serial,reaSerial);

```

V – End of Tut :

- Finished – **06/07/2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.crazy-soft.com
Production :	crazy-soft Software
SoftWare :	Ultra Calendar Reminder 2.3
Copyright by :	Copyright © 2002-2003 crazy-soft Software. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Ultra Calendar Reminder 2.3

Ultra reminder runs in the background, consumes very little system resources, and makes it impossible to forget things like your wife's birthday or important appointments. Ultra Reminder has two main function. One is an easy-to-use, event based personal event scheduler that lets you manage your important things. The other is instant message system that allows you to communicate with your classmates or workmates freely via local area net work.

Ultra Reminder also supports different event types, you can also customize event types to meet your needs. In addition to standard event that occurs only once, Ultra Reminder support recurrent event, which may occur at a regular interval.

When Ultra Reminder reminds you of an event, you can accept it, delete it and delay it.

Ultra Reminder lets you send instant messages to others who are in the same local network.

You can use message list to review all the messages you have sent and received.

Ultra Reminder provides you a customizable clock window, which you can place anywhere on the screen. And it only takes a very small place.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Microsoft Visual C++ 6.0**
- Nhập thử User và Fake Serial, ta nhận được thông báo "Your registration code is not valid. Please register it." . Tuy nhiên ta không thể tìm thấy chuỗi thông báo này trong Olly bằng các phương pháp thông thường . Sử dụng phương pháp tìm chuỗi bằng STAKCK, ta xác định được địa chỉ xuất hiện NAG : 0040260A . FF90 CC000000 CALL DWORD PTR DS:[EAX+CC] ; <== NAG
- Nhập thử User và Fake Serial, ta nhận được thông báo "Your registration code is not valid. Please register it." . Tuy nhiên ta không thể tìm thấy chuỗi thông báo này trong Olly bằng các phương pháp thông thường . Sử dụng phương pháp tìm chuỗi bằng STAKCK, ta xác định được địa chỉ xuất hiện NAG : 00402610 . EB 0E JMP SHORT UMinder.00402620
- Dò ngược lên trên và ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này : 0040258B . E8 30EDFFFF CALL UMinder.004012C0 ; \UMinder.004012C0

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống một đoạn ta đến :

```
004025A7 . 50      PUSH EAX          ; /Arg2
004025A8 . 8D4D EC    LEA ECX,DWORD PTR SS:[EBP-14] ; |
004025AB . 51      PUSH ECX         ; |Arg1
004025AC . E8 8C240100  CALL UMinder.00414A3D       ; \UMinder.00414A3D
```

- Dùng F7 để trace into lệnh CALL này :

```
00414A7B |. 50      PUSH EAX          ; /Arg1
00414A7C |. E8 89FEFFFF  CALL UMinder.0041490A       ; \UMinder.0041490A
```

- Tiếp dùng F7 để trace into và dùng F8 trace tiếp một đoạn, ta đến :

```
00414938 |. 52      PUSH EDX         ; /Arg2 = 0012F5C4
00414939 |. 8B45 08    MOV EAX,[ARG.1]   ; |
0041493C |. 50      PUSH EAX         ; |Arg1 = 00917530 ASCII "XXXXXXXX"
0041493D |. E8 87FAFFFF  CALL UMinder.004143C9       ; \UMinder.004143C9
```

- Dùng F7 trace into, ta đến đoạn CODE mã hóa chuỗi Serial lần đầu tiên :

```
004143D8 |>/8B45 FC    /MOV EAX,[LOCAL.1]
004143DB |. |83C0 01    |ADD EAX,1           ; <== i++
004143DE |. |8945 FC    |MOV [LOCAL.1],EAX
004143E1 |>|837D FC 18  |CMP [LOCAL.1],18      ; <== Number of Loop : 24 times
004143E5 |. |7D 54      |JGE SHORT UMinder.0041443B
004143E7 |. |837D FC 01  |CMP [LOCAL.1],1      ; <== if ( i == 1 )
004143EB |. |74 18      |JE SHORT UMinder.00414405 ; <== or
004143ED |. |837D FC 05  |CMP [LOCAL.1],5      ; <== if ( i == 5 )
004143F1 |. |74 12      |JE SHORT UMinder.00414405 ; <== or
004143F3 |. |837D FC 08  |CMP [LOCAL.1],8      ; <== if ( i == 8 )
004143F7 |. |74 0C      |JE SHORT UMinder.00414405 ; <== or
004143F9 |. |837D FC 0B  |CMP [LOCAL.1],0B     ; <== if ( i == 11 )
004143FD |. |74 06      |JE SHORT UMinder.00414405 ; <== or
004143FF |. |837D FC 14  |CMP [LOCAL.1],14     ; <== if ( i == 20 )
00414403 |. |75 12      |JNZ SHORT UMinder.00414417 ; <== then
00414405 |>|8B4D 0C    |MOV ECX,[ARG.2]
00414408 |. |034D FC    |ADD ECX,[LOCAL.1]
0041440B |. |8B55 08    |MOV EDX,[ARG.1]        ; <== Serial
0041440E |. |0355 FC    |ADD EDX,[LOCAL.1]
```

```

00414411 |. |8A02    |MOV AL,BYTE PTR DS:[EDX]      ; <== S[i]
00414413 |. |8801    |MOV BYTE PTR DS:[ECX],AL      ; <== Temp[i] = S[i]
00414415 |. |EB 22    |JMP SHORT UMinder.00414439
00414417 |>|8D4D F8    |LEA ECX,[LOCAL.2]      ; <== Else
0041441A |. |51      |PUSH ECX          ; /Arg2
0041441B |. |8B55 08    |MOV EDX,[ARG.1]        ; |<== Serial
0041441E |. |0355 FC    |ADD EDX,[LOCAL.1]      ; |
00414421 |. |8A02      |MOV AL,BYTE PTR DS:[EDX]      ; |<== S[i]
00414423 |. |50      |PUSH EAX          ; |Arg1
00414424 |. |B9 68474400 |MOV ECX,UMinder.00444768      ; |
00414429 |. |E8 320B0000 |CALL UMinder.00414F60      ; \UMinder.00414F60

```

===== NOTE =====

- Lệnh CALL này thực hiện quá trình chuyển đổi từng ký tự của chuỗi Serial sang một ký tự khác theo một quy tắc được định trước . Quy tắc này được thể hiện :

04-1X-2F-3C-41-5K-6S-7T-8N-9H-AU-B0-CR-DI-EY-F2-GG-HA-I6-JE-KL-LZ-M3-N8-OM-P5-QO-RB-SD-TW-U7-VP-WJ-XV-Y9-ZQ

- Ví dụ : nếu chuỗi nhập là “X” thì ký tự được chuyển đổi sẽ là “V”

- Đồng thời, đoạn CODE này được lặp 24 lần – tương ứng với 24 ký tự – vì thế nếu chuỗi Serial nhập nhỏ hơn 24 ký tự thì các ký tự ngẫu nhiên sẽ được gán cho đủ 24 ký tự .

===== NOTE =====

```

0041442E |. |8B4D 0C    |MOV ECX,[ARG.2]
00414431 |. |034D FC    |ADD ECX,[LOCAL.1]
00414434 |. |8A55 F8    |MOV DL,BYTE PTR SS:[EBP-8]      ; <== Value at this Add.
00414437 |. |8811      |MOV BYTE PTR DS:[ECX],DL      ; <== Temp[i] = Value
00414439 |>|^EB 9D    |JMP SHORT UMinder.004143D8      ; <== Continue Loop

```

- Quá trình kế tiếp là quá trình kết hợp hai ký tự ở hai vị trí đõđ các định trước thành một ký tự . Như vậy ta sẽ có được một chuỗi gồm 12 ký tự . Chuỗi này được đem so sánh với một chuỗi mặc định :

```

00414945 |. OFBE45 E8    MOVSX EAX,BYTE PTR SS:[EBP-18]      ; <== Temp[4]
00414949 |. OFBE4D F5    MOVSX ECX,BYTE PTR SS:[EBP-B]      ; <== Temp[17]
0041494D |. 03C1      ADD EAX,ECX          ; <== Value = Temp[4] + Temp[7]
0041494F |. 99      CDQ
00414950 |. 2BC2      SUB EAX,EDX
00414952 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
00414954 |. 8845 D4    MOV BYTE PTR SS:[EBP-2C],AL      ; <== Char[0] = Value
00414957 |. OFBE45 E4    MOVSX EAX,BYTE PTR SS:[EBP-1C]      ; <== Temp[0]
0041495B |. 0FBE55 F7    MOVSX EDX,BYTE PTR SS:[EBP-9]      ; <== Temp[19]
0041495F |. 03C2      ADD EAX,EDX          ; <== Value = Temp[0] + Temp[19]
00414961 |. 99      CDQ
00414962 |. 2BC2      SUB EAX,EDX
00414964 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
00414966 |. 8845 D5    MOV BYTE PTR SS:[EBP-2B],AL      ; <== Char[1] = Value
00414969 |. OFBE45 EB    MOVSX EAX,BYTE PTR SS:[EBP-15]      ; <== Temp[7]
0041496D |. 0FBE4D F2    MOVSX ECX,BYTE PTR SS:[EBP-E]      ; <== Temp[14]
00414971 |. 03C1      ADD EAX,ECX          ; <== Value = Temp[7] + Temp[14]
00414973 |. 99      CDQ
00414974 |. 2BC2      SUB EAX,EDX
00414976 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
00414978 |. 8845 D6    MOV BYTE PTR SS:[EBP-2A],AL      ; <== Char[2] = Value
0041497B |. OFBE45 E5    MOVSX EAX,BYTE PTR SS:[EBP-1B]      ; <== Temp[1]
0041497F |. 0FBE55 F8    MOVSX EDX,BYTE PTR SS:[EBP-8]      ; <== Temp[20]
00414983 |. 03C2      ADD EAX,EDX          ; <== Value = Temp[1] + Temp[20]
00414985 |. 99      CDQ

```

```

00414986 |. 2BC2      SUB EAX,EDX
00414988 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
0041498A |. 8845 D7    MOV BYTE PTR SS:[EBP-29],AL      ; <== Char[3] = Value
0041498D |. 0FBE45 EC    MOVSX EAX,BYTE PTR SS:[EBP-14]    ; <== Temp[8]
00414991 |. 0FBE4D FA    MOVSX ECX,BYTE PTR SS:[EBP-6]      ; <== Temp[22]
00414995 |. 03C1      ADD EAX,ECX          ; <== Value = Temp[8] + Temp[22]
00414997 |. 99        CDQ
00414998 |. 2BC2      SUB EAX,EDX
0041499A |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
0041499C |. 8845 D8    MOV BYTE PTR SS:[EBP-28],AL      ; <== Char[4] = Value
0041499F |. 0FBE45 ED    MOVSX EAX,BYTE PTR SS:[EBP-13]    ; <== Temp[9]
004149A3 |. 0FBE55 FB    MOVSX EDX,BYTE PTR SS:[EBP-5]      ; <== Temp[23]
004149A7 |. 03C2      ADD EAX,EDX          ; <== Value = Temp[9] + Temp[23]
004149A9 |. 99        CDQ
004149AA |. 2BC2      SUB EAX,EDX
004149AC |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
004149AE |. 8845 D9    MOV BYTE PTR SS:[EBP-27],AL      ; <== Char[5] = Value
004149B1 |. 0FBE45 E6    MOVSX EAX,BYTE PTR SS:[EBP-1A]    ; <== Temp[2]
004149B5 |. 0FBE4D F4    MOVSX ECX,BYTE PTR SS:[EBP-C]      ; <== Temp[16]
004149B9 |. 03C1      ADD EAX,ECX          ; <== Value = Temp[2] + Temp[16]
004149BB |. 99        CDQ
004149BC |. 2BC2      SUB EAX,EDX
004149BE |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
004149C0 |. 8845 DA    MOV BYTE PTR SS:[EBP-26],AL      ; <== Char[6] = Value
004149C3 |. 0FBE45 EF    MOVSX EAX,BYTE PTR SS:[EBP-11]    ; <== Temp[11]
004149C7 |. 0FBE55 F3    MOVSX EDX,BYTE PTR SS:[EBP-D]      ; <== Temp[15]
004149CB |. 03C2      ADD EAX,EDX          ; <== Value = Temp[11] + Temp[15]
004149CD |. 99        CDQ
004149CE |. 2BC2      SUB EAX,EDX
004149D0 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
004149D2 |. 8845 DB    MOV BYTE PTR SS:[EBP-25],AL      ; <== Char[7] = Value
004149D5 |. 0FBE45 E7    MOVSX EAX,BYTE PTR SS:[EBP-19]    ; <== Temp[3]
004149D9 |. 0FBE4D F6    MOVSX ECX,BYTE PTR SS:[EBP-A]      ; <== Temp[18]
004149DD |. 03C1      ADD EAX,ECX          ; <== Value = Temp[3] + Temp[18]
004149DF |. 99        CDQ
004149E0 |. 2BC2      SUB EAX,EDX
004149E2 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
004149E4 |. 8845 DC    MOV BYTE PTR SS:[EBP-24],AL      ; <== Char[8] = Value
004149E7 |. 0FBE45 EE    MOVSX EAX,BYTE PTR SS:[EBP-12]    ; <== Temp[10]
004149EB |. 0FBE55 F9    MOVSX EDX,BYTE PTR SS:[EBP-7]      ; <== Temp[21]
004149EF |. 03C2      ADD EAX,EDX          ; <== Value = Temp[10] + Temp[12]
004149F1 |. 99        CDQ
004149F2 |. 2BC2      SUB EAX,EDX
004149F4 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
004149F6 |. 8845 DD    MOV BYTE PTR SS:[EBP-23],AL      ; <== Char[9] = Value
004149F9 |. 0FBE45 E9    MOVSX EAX,BYTE PTR SS:[EBP-17]    ; <== Temp[5]
004149FD |. 0FBE4D F1    MOVSX ECX,BYTE PTR SS:[EBP-F]      ; <== Temp[13]
00414A01 |. 03C1      ADD EAX,ECX          ; <== Value = Temp[5] + Temp[13]
00414A03 |. 99        CDQ
00414A04 |. 2BC2      SUB EAX,EDX
00414A06 |. D1F8      SAR EAX,1          ; <== Value = Value sar 1
00414A08 |. 8845 DE    MOV BYTE PTR SS:[EBP-22],AL      ; <== Char[10] = Value

```

```

00414A0B |. 0FBE45 EA    MOVSX EAX,BYTE PTR SS:[EBP-16]      ; <== Temp[6]
00414A0F |. 0FBE55 F0    MOVSX EDX,BYTE PTR SS:[EBP-10]      ; <== Temp[12]
00414A13 |. 03C2        ADD EAX,EDX                      ; <== Value = Temp[6] + Temp[12]
00414A15 |. 99          CDQ
00414A16 |. 2BC2        SUB EAX,EDX
00414A18 |. D1F8        SAR EAX,1                      ; <== Value = Value sar 1
00414A1A |. 8845 DF    MOV BYTE PTR SS:[EBP-21],AL      ; <== Char[11] = Value
00414A1D |. 8D45 D4    LEA EAX,[LOCAL.11]
00414A20 |. 50          PUSH EAX                      ; /String2
00414A21 |. 68 508D4300 PUSH UMinder.00438D50      ; |String1 = "2XCV58YINTHB"
00414A26 |. FF15 F8704300 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>]; \lstrcmpA
00414A2C |. 85C0        TEST EAX,EAX                  ; <== Correct or not
00414A2E |. 75 07        JNZ SHORT UMinder.00414A37      ; <== if Correct
00414A30 |. B8 01000000 MOV EAX,1                    ; <== EAX == 0x1
00414A35 |. EB 02        JMP SHORT UMinder.00414A39      ; <== Else
00414A37 |> 33C0        XOR EAX,EAX                  ; <== EAX == 0x0

```

- Ở đây ta nhận xét : với hai ký tự ở các vị trí đã được quy định sẽ ta sẽ có được một ký tự mới .Và ký tự ở vị trí nào phải trùng với ký tự mặc định ở vị trí đó . Như vậy, để đơn giản hóa quá trình ta có thể REVERSE chuỗi mặc định này để xác định giá trị . Quá trình REVERSE này được mô tả :

```

/*----- Create Check String --- DefaultCheck : "2XCV58YINTHB" -----
i=0;
while ( i < 12 )
{
    _asm
    {
        MOV EBX, i
        MOVSX EAX, reaDefaultCheck[EBX]
        SAL EAX, 1
        MOV ValueSAL, EAX
    }
    wsprintf(reTemp, "%X", ValueSAL);
    lstrcat(reCheckString,reTemp);
    i++;
}
----- Create Check String -----*/

```

- Và giá trị tìm được tương ứng : **64 B0 86 AC 6A 70 B2 92 9C A8 90 84**

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : TU170NY02Y6K5RCGLB9EWANU or
EXLZMNUN6N6LOR82145TTA0N

III – KeyGen :

- /Section 1/- REVERSE chuỗi kiểm tra để xác định giá trị của hai ký tự ở vị trí xác định .
- /Section 2/- Xây dựng chuỗi mới dựa trên giá trị của chuỗi được REVERSE này
- /Section 3/- Tạo ra chuỗi Serial thực dựa trên REVERSE quy tắc tạo chuỗi .

IV – SourceCode (VC++) :

```

char reaName[64]={0};
char reaSerial[64]={0};

```

```

char reaDefaultString[80] =
"041X2F3C415K6S7T8N9HAUB0CRDIEYF2GGHAI6JEKLLZM3N8OMP5QORBSDTWU7VPWJXV
Y9ZQ";
char reaRandomString[40] = "0147852369MNBVCXZASDFGHJKLPOIUYTREWQ";
char reaTemp[10] = {0};
char reaTempSerial[64] = {0};

int LenUser = 0, Temp = 0, Value = 0;
int i = 0, j = 0;

LenUser = GetDlgItemText(IDC_Name, reaName, 64);
srand((unsigned)time(NULL));
if (LenUser < 1)
{
    MessageBox("----- Your name atleast 1 chart -----", "Hey !! Please
input your name again !! ");
}
else
{
    while (Value != 0x64)
    {
        i = 0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand() % 36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }
    reaTempSerial[4] = reaTemp[0];
    reaTempSerial[17] = reaTemp[1];
    while (Value != 0xB0)
    {
        i = 0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand() % 36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }
    reaTempSerial[0] = reaTemp[0];
    reaTempSerial[19] = reaTemp[1];
    while (Value != 0x86)
    {
        i = 0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand() % 36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }
}

```

```

        }
        reaTempSerial[7] = reaTemp[0];
        reaTempSerial[14] = reaTemp[1];
        while ( Value !=0xAC)
        {
            i=0;
            while (i < 2)
            {
                reaTemp[i] = reaRandomString[rand()%36];
                i++;
            }
            Value = reaTemp[0] + reaTemp[1];
        }
        reaTempSerial[1] = reaTemp[0];
        reaTempSerial[20] = reaTemp[1];
        while ( Value !=0x6A)
        {
            i=0;
            while (i < 2)
            {
                reaTemp[i] = reaRandomString[rand()%36];
                i++;
            }
            Value = reaTemp[0] + reaTemp[1];
        }
        reaTempSerial[8] = reaTemp[0];
        reaTempSerial[22] = reaTemp[1];
        while ( Value !=0x70)
        {
            i=0;
            while (i < 2)
            {
                reaTemp[i] = reaRandomString[rand()%36];
                i++;
            }
            Value = reaTemp[0] + reaTemp[1];
        }
        reaTempSerial[9] = reaTemp[0];
        reaTempSerial[23] = reaTemp[1];
        while ( Value !=0xB2)
        {
            i=0;
            while (i < 2)
            {
                reaTemp[i] = reaRandomString[rand()%36];
                i++;
            }
            Value = reaTemp[0] + reaTemp[1];
        }
        reaTempSerial[2] = reaTemp[0];
        reaTempSerial[16] = reaTemp[1];
        while ( Value !=0x92)
        {
    
```

```

        i=0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand()%36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }

    reaTempSerial[11] = reaTemp[0];
    reaTempSerial[15] = reaTemp[1];
    while (Value !=0x9C)
    {
        i=0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand()%36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }

    reaTempSerial[3] = reaTemp[0];
    reaTempSerial[18] = reaTemp[1];
    while (Value !=0xA8)
    {
        i=0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand()%36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }

    reaTempSerial[10] = reaTemp[0];
    reaTempSerial[21] = reaTemp[1];
    while (Value !=0x90)
    {
        i=0;
        while (i < 2)
        {
            reaTemp[i] = reaRandomString[rand()%36];
            i++;
        }
        Value = reaTemp[0] + reaTemp[1];
    }

    reaTempSerial[5] = reaTemp[0];
    reaTempSerial[13] = reaTemp[1];
    while (Value !=0x84)
    {
        i=0;
        while (i < 2)
        {
    
```

```

        reaTemp[i] = reaRandomString[rand()%36];
                    i++;
    }
    Value = reaTemp[0] + reaTemp[1];
}
reaTempSerial[6] = reaTemp[0];
reaTempSerial[12] = reaTemp[1];
i=0;
while ( i < 24 )
{
    if ( i == 1 || i == 5 || i == 8 || i == 11 || i == 20 )
    {
        reaSerial[i] = reaTempSerial[i];
        i++;
    }
    else
    {
        j=1;
        while ( j < lstrlen(readefaultString))
        {
            if ( reaTempSerial[i] == readefaultString[j])
            {
                reaSerial[i] = readefaultString[j-1];
                j+=2;
            }
            else
            {
                j+=2;
            }
        }
        i++;
    }
}
SetDlgItemText(IDC_Serial,reaSerial);
}

```

V – End of Tut :

- Finished – ***13/07/2004***

Reverse Engineering Association SoftWare

Homepage :	http://www.visual-mp3.com
Production :	iProgram Development.
SoftWare :	Visual MP3 CD Burner
Copyright by :	Copyright © 2002,2003 by iProgram Development. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A

Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Visual MP3 CD Burner

Visual MP3 CD Burner is a member of the award-winning Visual MP3 Family. With Visual MP3 CD Burner you can burn MP3, MP2, WMA, WAV (compressed/uncompressed) and Ogg Vorbis to audio CD On-The-Fly. Visual MP3 CD Burner uses "Track-at-Once" burning method. The audio CD can be played in all CD/DVD players - at home or in the car.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 6.0**
- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**The registration code is not valid.**". Ta tìm được thông báo này tại địa chỉ :
0040DD19 . 68 C0354500 PUSH VMP3CDBu.004535C0 ; |Arg1 = 004535C0 ASCII "The registration code is not valid."
- Truy ngược lên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :
0040DC5B . E8 A01D0200 CALL VMP3CDBu.0042FA00 ; <== Set BreakPoint here

II – Cracking :

- Quá trình mã hoá của chương trình này rất đơn giản . Từ BP ta trace xuống chút :

```

0040DCBC . 56      PUSH ESI          ; /Arg2
0040DCBD . 51      PUSH ECX         ; |Arg1
0040DCBE . 8BCB    MOV ECX,EBX       ; |
0040DCC0 . E8 5B040000 CALL VMP3CDBu.0040E120   ; \VMP3CDBu.0040E120
----- Trace Into -----
0040E13A ]. C74424 08 C43>MOV DWORD PTR SS:[ESP+8],VMP3CDBu.004536C4     ; ASCII
"60583B-"
0040E142 ]. C74424 0C BC3>MOV DWORD PTR SS:[ESP+C],VMP3CDBu.004536BC     ; ASCII
"236064-"
0040E14A ]. C74424 10 B43>MOV DWORD PTR SS:[ESP+10],VMP3CDBu.004536B4     ; ASCII
"74805F-"
0040E152 ]. C74424 14 A83>MOV DWORD PTR SS:[ESP+14],VMP3CDBu.004536A8     ; ASCII
"8A69046-"
0040E15A ]. C74424 18 9C3>MOV DWORD PTR SS:[ESP+18],VMP3CDBu.0045369C     ; ASCII
"515E406-"
0040E162 ]. C74424 1C 903>MOV DWORD PTR SS:[ESP+1C],VMP3CDBu.00453690     ; ASCII
"8032305-"
0040E16A ]. C74424 20 843>MOV DWORD PTR SS:[ESP+20],VMP3CDBu.00453684     ; ASCII
"5B584E6-"
0040E172 ]. C74424 24 783>MOV DWORD PTR SS:[ESP+24],VMP3CDBu.00453678     ; ASCII
"365D57B-"
0040E17A ]. C74424 28 6C3>MOV DWORD PTR SS:[ESP+28],VMP3CDBu.0045366C     ; ASCII
"57343A3-"
0040E182 ]. C74424 2C 643>MOV DWORD PTR SS:[ESP+2C],VMP3CDBu.00453664     ; ASCII
"1533B4-"

```

```

0040E18A ]. C74424 30 5C3>MOV DWORD PTR SS:[ESP+30],VMP3CDBu.0045365C ; ASCII
"364026-"
0040E192 ]. C74424 34 503>MOV DWORD PTR SS:[ESP+34],VMP3CDBu.00453650 ; ASCII
"5131066-"
0040E19A ]. C74424 38 443>MOV DWORD PTR SS:[ESP+38],VMP3CDBu.00453644 ; ASCII
"0553437-"

```

===== Trace Into =====

- Trace tiếp ta đến :

```

0040DCC9 . 56      PUSH ESI          ; /Arg2
0040DCCA . 52      PUSH EDX         ; |Arg1
0040DCCB . 8BCB    MOV ECX,EBX       ; |
0040DCCD . E8 5E050000 CALL VMP3CDBu.0040E230 ; \VMP3CDBu.0040E230

```

===== Trace Into =====

```

0040E24A ]. C74424 08 4C3>MOV DWORD PTR SS:[ESP+8],VMP3CDBu.0045374C ; ASCII
"3665D2F"
0040E252 ]. C74424 0C 443>MOV DWORD PTR SS:[ESP+C],VMP3CDBu.00453744 ; ASCII
"3444C33"
0040E25A ]. C74424 10 3C3>MOV DWORD PTR SS:[ESP+10],VMP3CDBu.0045373C ; ASCII
"6262435"
0040E262 ]. C74424 14 343>MOV DWORD PTR SS:[ESP+14],VMP3CDBu.00453734 ; ASCII
"488304A"
0040E26A ]. C74424 18 2C3>MOV DWORD PTR SS:[ESP+18],VMP3CDBu.0045372C ; ASCII
"315543A"
0040E272 ]. C74424 1C 243>MOV DWORD PTR SS:[ESP+1C],VMP3CDBu.00453724 ; ASCII
"4F32247"
0040E27A ]. C74424 20 1C3>MOV DWORD PTR SS:[ESP+20],VMP3CDBu.0045371C ; ASCII
"353355B"
0040E282 ]. C74424 24 143>MOV DWORD PTR SS:[ESP+24],VMP3CDBu.00453714 ; ASCII
"4E24E65"
0040E28A ]. C74424 28 0C3>MOV DWORD PTR SS:[ESP+28],VMP3CDBu.0045370C ; ASCII
"3F3F945"
0040E292 ]. C74424 2C 043>MOV DWORD PTR SS:[ESP+2C],VMP3CDBu.00453704 ; ASCII
"663145E"
0040E29A ]. C74424 30 FC3>MOV DWORD PTR SS:[ESP+30],VMP3CDBu.004536FC ; ASCII
"35B6D58"
0040E2A2 ]. C74424 34 F43>MOV DWORD PTR SS:[ESP+34],VMP3CDBu.004536F4 ; ASCII
"525F574"
0040E2AA ]. C74424 38 EC3>MOV DWORD PTR SS:[ESP+38],VMP3CDBu.004536EC ; ASCII
"13E5735"

```

===== Trace Into =====

- Sau đó, tương ứng từng giá trị ở đoạn CODE bên trên, chương trình kết hợp tương ứng với một giá trị ở đoạn CODE bên dưới . Chuỗi tạo thành sẽ được so sánh với chuỗi S ta nhập . Với số vòng lặp là 13 (0xD) ta có được 13 chuỗi Serial thực :

```

0040DCD2 . 8D4424 34 LEA EAX,DWORD PTR SS:[ESP+34]      ; <== SecII
0040DCD6 . 8D4C24 20 LEA ECX,DWORD PTR SS:[ESP+20]      ; <== SecI
0040DCDA . 50      PUSH EAX
0040DCDB . 51      PUSH ECX
0040DCDC . 8D9424 A40200>LEA EDX,DWORD PTR SS:[ESP+2A4] ; <== SecISecII
0040DCE3 . 68 0C364500 PUSH VMP3CDBu.0045360C          ; ASCII "%s%s"
0040DCE8 . 52      PUSH EDX

```

```

0040DCE9 . E8 B1290100 CALL VMP3CDBu.0042069F ; <== Keep format
0040DCEE . 8B4C24 20 MOV ECX,DWORD PTR SS:[ESP+20]
0040DCF2 . 8D8424 AC0200>LEA EAX,DWORD PTR SS:[ESP+2AC]
0040DCF9 . 50 PUSH EAX
0040DCFA . 51 PUSH ECX
0040DCFB . E8 9A2A0100 CALL VMP3CDBu.0042079A ; <== Compare with FS
0040DD00 . 83C4 18 ADD ESP,18
0040DD03 . 85C0 TEST EAX,EAX
0040DD05 . 0F84 1C010000 JE VMP3CDBu.0040DE27
0040DD0B . 46 INC ESI
0040DD0C . 83FE 0D CMP ESI,0D ; <== Loop 13 times
0040DD0F .^ 7C A7 JL SHORT VMP3CDBu.0040DCB8

```

/*/*/* - SERIAL tuong ứng :

User : REA-cRaCkErTeAm	Serial : N/A
Serial :	60583B-3665D2F 236064-3444C33 74805F-6262435
	8A69046-488304A 515E406-315543A 8032305-4F32247
	5B584E6-353355B 365D57B-4E24E65 57343A3-3F3F945
	1533B4-663145E 364026-35B6D58 5131066-525F574
	0553437-13E5735

III – KeyGen :

/Section 0 /- N/A

IV – End of Tut :

- Finished – *August 24, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.audiotoolsfactory.com
Production :	AudiotoolsFactory.com
SoftWare :	WAV MP3 Converter 1.00
Copyright by :	Copyright © 2004 AudiotoolsFactory.com. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Borland Delphi 6.0 - 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N / A
Request :	Correct Serial

WAV MP3 Converter 1.00

To small your WAV files? To convert your MP3, WMA and OGG files to CD quality WAV files for burning your own CD? Just use our WAV MP3 Converter. WAV MP3 Converter supports batch conversion between MP3, WAV, WMA and OGG. You can convert your MP3, WMA and OGG files to CD quality WAV files for burning your audio CD(Assuming you have a CD-R). You can convert your WAV, WMA and OGG files to MP3 files for listening the songs with your MP3 Player. WAV MP3 Converter converts the audio files digitally-not through the soundcard-which enables you

to make perfect copies of the originals. Convert from one to another directly and on-the-fly (without temporary files produced). WAV MP3 Converter has user-friendly-interface and converting your files is just a button click away. You'll be an experter in no time! Run stably on Windows 98/NT/Me/2K/XP/2003.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe không bị PACK được viết bằng **Borland Delphi 6.0 - 7.0**.
 - Nhập thử User và Fake Serial, ta nhận được thông báo "Invalid register code! Please retry!" . Ta tìm được ba địa chỉ của thông báo này. Tất cả cùng nằm trên một Function :
- ```
004C1BAE . B8 101C4C00 MOV EAX,WAV_MP3_.004C1C10 ; |ASCII "Invalid register code! Please retry!"
```
- Dò ngược lên trên, ta đặt BreakPoint tại lệnh CALL đầu tiên của Function này :
- ```
004C1A0E . E8 55C7F9FF CALL WAV_MP3_.0045E168 ; <== Set BreakPoint here
```

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial. Chương trình dừng lại tại điểm đặt BP . Chương trình sẽ kiểm tra User nhập và một số User mặc định :

```
004C1A42 . BB 15000000 MOV EBX,15 ; <== Number of Default User
004C1A47 . BE E0694C00 MOV ESI,WAV_MP3_.004C69E0 ; <== Add of Default User
004C1A4C > 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== User
004C1A4F . 8B16 MOV EDX,DWORD PTR DS:[ESI] ; <== Default User
004C1A51 . E8 4A30F4FF CALL WAV_MP3_.00404AA0 ; <== They must be equal
004C1A56 . 75 09 JNZ SHORT WAV_MP3_.004C1A61 ; <== if Not check another default User
004C1A58 . C605 3C8E4C00>MOV BYTE PTR DS:[4C8E3C],0 ; <== Else
004C1A5F . EB 06 JMP SHORT WAV_MP3_.004C1A67 ; <== Next check
004C1A61 > 83C6 04 ADD ESI,4 ; <== Next default User
004C1A64 . 4B DEC EBX ; <== Had 21 default Users
004C1A65 .^ 75 E5 JNZ SHORT WAV_MP3_.004C1A4C ; <== Continue loop
004C1A67 > 803D 3C8E4C00>CMP BYTE PTR DS:[4C8E3C],0 ; <== if InputUser == DefaultUser
004C1A6E . 74 1A JE SHORT WAV_MP3_.004C1A8A ; <== Next check
```

- Như vậy, đối với chương trình này, ta không thể nhập U để tạo ra Serial, mà chương trình đã quy định một số User mặc định .

- Tiếp đó, chương trình kiểm tra tính hợp lệ của Serial nhập :

```
004C1AD0 . E8 872EF4FF CALL WAV_MP3_.0040495C ; <== Get Len.S
004C1AD5 . 85C0 TEST EAX,EAX ; <== S must be input
004C1AD7 . 7E 38 JLE SHORT WAV_MP3_.004C1B11
004C1AD9 . BA 01000000 MOV EDX,1 ; <== i = 1
004C1ADE > 8B4D F8 MOV ECX,DWORD PTR SS:[EBP-8] ; <== Serial
004C1AE1 . 0FB64C11 FF MOVZX ECX,BYTE PTR DS:[ECX+EDX-1] ; <== S[i-1]
004C1AE6 . 83F9 30 CMP ECX,30 ; <== S[i-1] >= 0x30
004C1AE9 . 7C 08 JL SHORT WAV_MP3_.004C1AF3
004C1AEB . 8B5D F8 MOV EBX,DWORD PTR SS:[EBP-8]
004C1AEE . 83F9 39 CMP ECX,39 ; <== S[i-1] <= 0x39
004C1AF1 . 7E 1A JLE SHORT WAV_MP3_.004C1B0D
004C1AF3 > 6A 00 PUSH 0 ; /Arg1 = 00000000
004C1AF5 . 66:8B0D 041C4>MOV CX,WORD PTR DS:[4C1C04] ; |
004C1AFC . B2 02 MOV DL,2 ; |
004C1AFE . B8 101C4C00 MOV EAX,WAV_MP3_.004C1C10 ; |ASCII "Invalid register code!
Please retry!"
```

```
004C1B03 . E8 3C96F7FF CALL WAV_MP3_.0043B144 ; \WAV_MP3_.0043B144
004C1B08 . E9 AB000000 JMP WAV_MP3_.004C1BB8
```

```

004C1B0D > 42      INC EDX          ; <== i++
004C1B0E . 48       DEC EAX          ; <== Len.S --
004C1B0F .^ 75 CD    JNZ SHORT WAV_MP3_.004C1ADE ; <== Loop until Len.S == 0x0
- Như vậy, Serial phải là các số nằm trong khoảng ( 0 – 9 ). Ké đến là quá trình mã hoá chuỗi U mặc định
:
004C1B16 . E8 412EF4FF CALL WAV_MP3_.0040495C      ; <== get Len.DefaultU
004C1B1B . 85C0      TEST EAX,EAX
004C1B1D . 7E 13      JLE SHORT WAV_MP3_.004C1B32
004C1B1F . BB 01000000 MOV EBX,1          ; <== i = 1
004C1B24 > 8B55 FC    MOV EDX,DWORD PTR SS:[EBP-4] ; <== DefaultU
004C1B27 . 0FB6541A FF  MOVZX EDX,BYTE PTR DS:[EDX+EBX-1] ; <== DefaultU [i-1]
004C1B2C . 03F2      ADD ESI,EDX         ; <== CumV = CumV + DefaultU[i-1]
004C1B2E . 43       INC EBX          ; <== i++
004C1B2F . 48       DEC EAX          ; <== Len.DefaultU --
004C1B30 .^ 75 F2    JNZ SHORT WAV_MP3_.004C1B24 ; <== Loop until Len.DefaultU == 0x0
004C1B32 > 69C6 8E8D0900 IMUL EAX,ESI,98D8E      ; <== CumV = CumV * 0x98D8E
004C1B38 . 83C0 20    ADD EAX,20          ; <== CumV = CumV + 0x20
004C1B3B . D1F8      SAR EAX,1          ; <== CumV = CumV / 0x2
004C1B3D . 79 03      JNS SHORT WAV_MP3_.004C1B42
004C1B3F . 83D0 00    ADC EAX,0          ; <== CumV
004C1B42 > 8BF0      MOV ESI,EAX         ; <== CumV
004C1B44 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8] ; <== Serial
004C1B47 . E8 0473F4FF CALL WAV_MP3_.00408E50      ; <== Convert to HEX value : ValueS
004C1B4C . 3BF0      CMP ESI,EAX         ; <== ValueS == CumV ?
004C1B4E . 75 53      JNZ SHORT WAV_MP3_.004C1BA3 ; <== If YES --> Congrat!
004C1B50 . 6A 00      PUSH 0             ; /Arg1 = 00000000
004C1B52 . 66:8B0D 041C4>MOV CX,WORD PTR DS:[4C1C04] ; |
004C1B59 . B2 02      MOV DL,2           ; |
004C1B5B . B8 401C4C00 MOV EAX,WAV_MP3_.004C1C40      ; |ASCII "Congratuation! You
have successfully registered!"
004C1B60 . E8 DF95F7FF CALL WAV_MP3_.0043B144      ; \WAV_MP3_.0043B144
- Đây chẳng qua là một quá trình cộng dồn giá trị các ký tự của chuỗi mặc định, rồi tiến hành thêm vài
phép toán đơn giản .
- Như vậy, ứng với 25 chuỗi User mặc định ta cũng có 25 giá trị Serial mặc định .

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErS	Serial : N/A
User : VHT-cRaCkErS	Serial : N/A

	Serial : N / A
VS88T6-Vs86	: 242599041
S1R6P6-SV66	: 226634460
TDR6p6-EVS	: 230703863
Tsf6p6-VB1	: 240094793
B8TDf6p6-VB1	: 263572118
Osrf6p6-VB1	: 274215172
ESrg6p6-VB1	: 261380901
IUDT6-BX1	: 191574988
S1IT6-DV1	: 185001337
SNWS6-TN1	: 200652887
TDVS6-MN3	: 195957422
TV66P6-TV66	: 230077801
TDR6p6-SV66	: 242912072
TDR6p6-ST1	: 223817181
B8sf6p6-VB1	: 251989971
BS45f6p6-VB1	: 257311498
Ofrg6p6-VB1	: 270458800
IUrg6p6-VB1	: 263259087
DUIT6-tV1	: 206600476
SNMS6-DV1	: 195018329
SNWS6-MN3 :	199087732

III – KeyGen :

N/A

IV – SourceCode (VC++) :

N/A

V – End of Tut :- Finished – **07/07/2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.doease.com
Production :	Doease Software,Inc.
SoftWare :	X DVD Ripper 1.2.1
Copyright by :	Copyright © 2004 Doease Software,Inc. All Rights Reserved.
Type :	Email / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

X DVD Ripper 1.2.1

X DVD Ripper is an easy to use DVD Copying and Ripping software which can convert DVD to VCD,SVCD,AVI,Divx with excellent video and audio quality.

X DVD Ripper provides you superb features and clear interface. You can convert your favorite DVD to almost all popular video formats with several clicks.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK được viết bằng **Microsoft Visual C++ 6.0**.
- Nhập thử Email và Fake Serial, ta nhận được thông báo "**Invalid Reg-Email or Reg-Code.**" Ta tìm được đoạn CODE này ở hai địa chỉ :
 - 0040DE32 . 68 EC9F4800 PUSH X_DVD_Ri.00489FEC ; ASCII "Invalid Reg-Email or Reg-Code."
 - 00407D16 . 68 E0824100 PUSH VideoCon.004182E0 ; ASCII "Invalid Reg-Email or Reg-Code."
- Dò ngược lên trên và ta đặt BP tại lệnh CALL đầu tiên của Function này :
 - 0040DC14 . E8 D9220500 CALL <JMP.&MFC42.#540> ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với Email và Fake Serial, chương trình dừng lại tại điểm đặt BP. Đầu tiên chương trình kiểm tra tính hợp lệ của Email nhập :
 - 0040DC31 . E8 B6220500 CALL <JMP.&MFC42.#3874> ; <== Get Length of Email
 - 0040DC36 . 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C] ; <== Email

0040DC3A . 8B41 F8	MOV EAX,DWORD PTR DS:[ECX-8]	; <== Len.E
0040DC3D . 83F8 40	CMP EAX,40	; <== If (Len.E <= 0x40)
0040DC40 . 0F8F 23020000	JG X_DVD_Ri.0040DE69	; <== and
0040DC46 . 83F8 03	CMP EAX,3	; <== If (Len.E >= 0x3)
0040DC49 . 0F8C 1A020000	JL X_DVD_Ri.0040DE69	; <== Next check
0040DC4F . 6A 40	PUSH 40	
0040DC51 . 8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]	; <== Email
0040DC55 . E8 F8250500	CALL <JMP.&MFC42.#2763>	; <== check Validity of Email
0040DC5A . 83F8 FF	CMP EAX,-1	; <== if Valid
0040DC5D . 0F84 06020000	JE X_DVD_Ri.0040DE69	; <== Continue

- Sau đó chương trình kiểm tra chiều dài chuỗi Serial nhập :

0040DCA6 . E8 41220500	CALL <JMP.&MFC42.#3874>	; <== Get Length of Serial
0040DCAB . 8B4C24 10	MOV ECX,DWORD PTR SS:[ESP+10]	; <== Serial
0040DCAF . 8B41 F8	MOV EAX,DWORD PTR DS:[ECX-8]	; <== Len.S
0040DCB2 . 85C0	TEST EAX,EAX	; <== Serial must be input
0040DCB4 . 0F84 71010000	JE X_DVD_Ri.0040DE2B	
0040DCBA . 83F8 1D	CMP EAX,1D	; <== LenS must be 29 charts
0040DCBD . 0F85 68010000	JNZ X_DVD_Ri.0040DE2B	
..... Ripping		
0040DCDF . E8 6CFEFFFF	CALL X_DVD_Ri.0040DB50	; <== Encrypt & Compare

- Dùng F7 để trace into ta đến quá trình mã hoá đầu tiên :

0040DB6C >/0FBE142E	/MOVSX EDX,BYTE PTR DS:[ESI+EBP]	; <== E[i]
0040DB70 . 8BC2	MOV EAX,EDX	; <== E[i]
0040DB72 . BF 6F000000	MOV EDI,6F	; <== dV
0040DB77 . C1E0 04	SHL EAX,4	; <== Temp = E[i] * 0x10
0040DB7A . 03C2	ADD EAX,EDX	; <== Temp = Temp + E[i]
0040DB7C . 99	CDQ	
0040DB7D . F7FF	IDIV EDI	; <== Temp = Temp % dV
0040DB7F . 03DA	ADD EBX,EDX	; <== Value = Value + Temp
0040DB81 . 46	INC ESI	; <== i++
0040DB82 . 3BF1	CMP ESI,ECX	; <== while (i < LenE)
0040DB84 .^7C E6	JL SHORT X_DVD_Ri.0040DB6C	; <== Continue Loop
0040DB86 > 8BC3	MOV EAX,EBX	; <== Value
0040DB88 . 8B6C24 18	MOV EBP,DWORD PTR SS:[ESP+18]	; <== Fake Serial
0040DB8C . 0FAFC3	IMUL EAX,EBX	; <== Value = Value * Value

- Quá trình mã hoá thứ hai đồng thời bao hàm luôn quá trình so sánh diễn ra ngay sau đó :

0040DB8F . 33F6	XOR ESI,ESI	; <== i = 0
0040DB91 . 8BD8	MOV EBX,EAX	; <== Value_02 = Value
0040DB93 . 8BF8	MOV EDI,EAX	; <== Value_01 = Value
0040DB95 > 8BC6	/MOV EAX,ESI	; <== / form here
0040DB97 . B9 06000000	MOV ECX,6	; <== this section mean that
0040DB9C . 99	CDQ	; <== each section of RealSerial
0040DB9D . F7F9	IDIV ECX	; <== had only 5 charts and
0040DB9F . 83FA 05	CMP EDX,5	; <== separated by "-"
0040DBA2 . 74 25	JE SHORT X_DVD_Ri.0040DBC9	; <== \ to here
0040DBA4 . 8BC7	MOV EAX,EDI	; <== Value_01
0040DBA6 . B9 05000000	MOV ECX,5	; <== dV
0040DBAB . 99	CDQ	
0040DBAC . F7F9	IDIV ECX	; <== SecX = Value_01 % dV

```

0040DBAE . 8B1495 D89F48>|MOV EDX,DWORD PTR DS:[EDX*4+489FD8] ; <== Add of SecX
0040DBB5 . 8915 A4A54A00 |MOV DWORD PTR DS:[4AA5A4],EDX ; <== Add of SecX
0040DBBB . 8A042E |MOV AL,BYTE PTR DS:[ESI+EBP] ; <== fS[i]
0040DBBE . 50 |PUSH EAX ; <== Value = Value / dV
0040DBBF . 53 |PUSH EBX ; <== Value_02
0040DBC0 . FFD2 |CALL EDX ; <== SecX

```

===== Trace Into =====

===== NOTE =====

Thực tế ở đây ta chú ý đến phần dư của phép chia này. Do giá trị “**Value_01**” được chia cho “5” nên phần dư không bao giờ quá “4”. Nên giá trị của “**EDX**” chỉ có “5”, căn cứ và giá trị này mà lệnh “**CALL EDX**” sẽ chuyển đến FUNTION kiểm tra tương ứng; hay nói cách khác, tùy thuộc vào giá trị của phần dư mà chương trình sẽ chuyển đến FUNCTION kiểm tra tương ứng (có “5” FUNCTION cả thảy)

===== NOTE =====

===== Funtion 0 =====

```

0040DA60 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
0040DA64 . B8 4FECC44E MOV EAX,4EC4EC4F ; <== dV
0040DA69 . F7E9 IMUL ECX ; <== Over = Value_02 * dV
0040DA6B . 8BC2 MOV EAX,EDX ; <== Over
0040DA6D . C1F8 03 SAR EAX,3 ; <== Over = Over / 8
0040DA70 . 8BC8 MOV ECX,EAX ; <== Temp = Over
0040DA72 . C1E9 1F SHR ECX,1F ; <== Temp = Temp / 0x80000000
0040DA75 . 03C1 ADD EAX,ECX ; <== Over = Over + Temp
0040DA77 . B9 0A000000 MOV ECX,0A ; <== dV
0040DA7C . 99 CDQ
0040DA7D . F7F9 IDIV ECX ; <== Over = Over % dV
0040DA7F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== fS[i]
0040DA83 . 33C0 XOR EAX,EAX
0040DA85 . 80C2 30 ADD DL,30 ; <== Over = Over + 0x30
0040DA88 . 3AD1 CMP DL,CL ; <== if ( Over == fS[i] )
0040DA8A . 0F94C0 SETE AL ; <== AL = 0x1

```

===== Funtion I =====

```

0040DA90 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
0040DA94 . B8 79787878 MOV EAX,78787879 ; <== dV
0040DA99 . F7E9 IMUL ECX ; <== Over = Value_02 * dV
0040DA9B . 8BC2 MOV EAX,EDX ; <== Over
0040DA9D . C1F8 03 SAR EAX,3 ; <== Over = Over / 8
0040DAA0 . 8BC8 MOV ECX,EAX ; <== Temp = Over
0040DAA2 . C1E9 1F SHR ECX,1F ; <== Temp = Temp / 0x80000000
0040DAA5 . 03C1 ADD EAX,ECX ; <== Over = Over + Temp
0040DAA7 . B9 0A000000 MOV ECX,0A ; <== dV
0040DAAC . 99 CDQ
0040DAAD . F7F9 IDIV ECX ; <== Over = Over % dV
0040DAAF . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== fS[i]
0040DAB3 . 33C0 XOR EAX,EAX
0040DAB5 . 80C2 30 ADD DL,30 ; <== Over = Over + 0x30
0040DAB8 . 3AD1 CMP DL,CL ; <== if ( Over == fS[i] )
0040DABA . 0F94C0 SETE AL ; <== AL = 0x1

```

===== Funtion II =====

```

0040DAC0 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
0040DAC4 . B8 E9A28B2E MOV EAX,2E8BA2E9 ; <== dV
0040DAC9 . F7E9 IMUL ECX ; <== Over = Value_02 * dV
0040DACP . 8BC2 MOV EAX,EDX ; <== Over

```

```

0040DACP . D1F8      SAR EAX,1          ; <== Over = Over / 2
0040DACP . 8BC8      MOV ECX,EAX       ; <== Temp = Over
0040DAD1 . C1E9 1F   SHR ECX,1F        ; <== Temp = Temp / 0x80000000
0040DAD4 . 03C1      ADD EAX,ECX       ; <== Over = Over + Temp
0040DAD6 . B9 1A000000 MOV ECX,1A       ; <== dV
0040DADB . 99        CDQ
0040DADC . F7F9      IDIV ECX         ; <== Over = Over % dV
0040DADE . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== fS[i]
0040DAE2 . 33C0      XOR EAX,EAX
0040DAE4 . 80C2 41   ADD DL,41        ; <== Over = Over + 0x41
0040DAE7 . 3AD1      CMP DL,CL        ; <== if ( Over == fS[i] )
0040DAE9 . 0F94C0   SETE AL          ; <== AL = 0x1

```

===== Funtion III =====

```

0040DAF0 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
0040DAF4 . B8 4FECC44E MOV EAX,4EC4EC4F       ; <== dV
0040DAF9 . F7E9      IMUL ECX         ; <== Over = Value_02 * dV
0040DAFB . 8BC2      MOV EAX,EDX       ; <== Over
0040DAFD . C1F8 02   SAR EAX,2        ; <== Over = Over / 4
0040DB00 . 8BC8      MOV ECX,EAX       ; <== Temp = Over
0040DB02 . C1E9 1F   SHR ECX,1F        ; <== Temp = Temp / 0x80000000
0040DB05 . 03C1      ADD EAX,ECX       ; <== Over = Over + Temp
0040DB07 . B9 1A000000 MOV ECX,1A       ; <== dV
0040DB0C . 99        CDQ
0040DB0D . F7F9      IDIV ECX         ; <== Over = Over % dV
0040DB0F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== fS[i]
0040DB13 . 33C0      XOR EAX,EAX
0040DB15 . 80C2 41   ADD DL,41        ; <== Over = Over + 0x41
0040DB18 . 3AD1      CMP DL,CL        ; <== if ( Over == fS[i] )
0040DB1A . 0F94C0   SETE AL          ; <== AL = 0x1

```

===== Funtion IV =====

```

0040DB20 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] ; <== Value_02
0040DB24 . B8 F31ACA6B MOV EAX,6BCA1AF3       ; <== dV
0040DB29 . F7E9      IMUL ECX         ; <== Over = Value_02 * dV
0040DB2B . 8BC2      MOV EAX,EDX       ; <== Over
0040DB2D . C1F8 03   SAR EAX,3        ; <== Over = Over / 8
0040DB30 . 8BC8      MOV ECX,EAX       ; <== Temp = Over
0040DB32 . C1E9 1F   SHR ECX,1F        ; <== Temp = Temp / 0x80000000
0040DB35 . 03C1      ADD EAX,ECX       ; <== Over = Over + Temp
0040DB37 . B9 1A000000 MOV ECX,1A       ; <== dV
0040DB3C . 99        CDQ
0040DB3D . F7F9      IDIV ECX         ; <== Over = Over % dV
0040DB3F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] ; <== fS[i]
0040DB43 . 33C0      XOR EAX,EAX
0040DB45 . 80C2 41   ADD DL,41        ; <== Over = Over + 0x41
0040DB48 . 3AD1      CMP DL,CL        ; <== if ( Over == fS[i] )
0040DB4A . 0F94C0   SETE AL          ; <== AL = 0x1

```

===== Trace Into =====

```

0040DBC2 |. 83C4 08  |ADD ESP,8
0040DBC5 |. 85C0    |TEST EAX,EAX
0040DBC7  74 16   |JE SHORT X_DVD_Ri.0040DBDF
0040DBC9 |> 46    |INC ESI          ; <== i++
0040DBCA |. 83EF 0B  |SUB EDI,0B       ; <== Value_01 = Value_01 - 0xB

```

```

0040DBCD |. 83EB 11    |SUB EBX,11          ; <== Value_02 = Value_02 - 0x11
0040DBD0 |. 83FE 1D    |CMP ESI,1D          ; <== while ( i < 0x1D )
0040DBD3 |.^ 7C C0    |\JL SHORT X_DVD_Ri.0040DB95 ; <== Continue Loop

```

/*/*/* - SERIAL tương ứng với USER :

User : REA-cRaCkErStEaM@reaonline.net

Serial : TQN08-KG55K-Y01FX-58BQI-5WKB5

III – End of Tut :

- Finished – September 9, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.xilisoft.com
Production :	Xilisoft, Inc.
SoftWare :	Xilisoft DVD Audio Ripper 1.0.25 b 804
Copyright by :	Copyright © 2003-2004 Xilisoft, Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N / A
Language :	Microsoft Visual C++ 7.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N / A
Request :	Correct Serial / KeyGen

Xilisoft DVD Audio Ripper 1.0.25 b 804

Xilisoft DVD Audio Ripper is a DVD audio ripping tool easy to use with high ripping speed. It can extract your favorite DVD's audio track to mp3 and wav file. Xilisoft DVD Audio Ripper provides you with excellent sound quality and smaller file size just in a few clicks.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Microsoft Visual C++ 7.0**

- Chạy thử chương trình với User và Fake Serial ta nhận được thông báo "**Invalid registration info!**" . Ta tìm được thông báo này tại địa chỉ :

004142D4 . 68 E8114E00 PUSH soundtra.004E11E8 ; ASCII "Invalid registration info!"

- Truy ngược lên ta đặt BreakPoint tại lệnh CALL đầu tiên của FunTion này :

004142C8 . E8 73F6FFFF CALL soundtra.00413940 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP .

004142C3 . E8 58F0FFFF CALL soundtra.00413320 ; <== Set BreakPoint here

004142C8 . E8 73F6FFFF CALL soundtra.00413940 ; <== Trace Into here

- Dùng F7 trace into ta đến :

```
00413C5B . 83F8 27    CMP EAX,27          ; <== LenS must be 39 charts
00413C5E . 0F85 D6050000 JNZ soundtra.0041423A
```

- Trace tiếp ta xác định chuỗi dùng để mã hoá :

```
00413EC5 . E8 66DEF0FF CALL soundtra.00401D30 ; <== ConcatString
```

===== ***String to Encrypt*** =====

Ngay tại đây, quan sát của sổ STACK ta thấy dòng :

```
00126E1C 00F2ED60 ASCII "12345678901234567890Soundtrack"
```

Như vậy, chương trình đã lấy 20 ký tự đầu tiên và ghép với chuỗi mặc định “**“Soundtrack”**”. Ta có thể cho rằng chương trình sẽ dùng chuỗi này để tiến hành mã hoá, nhưng thực chất không phải vậy, chuỗi này chỉ là một phần của chuỗi thực.

Chương trình sẽ gắn chuỗi này vào sau một chuỗi mặc định thứ hai, chuỗi này gồm 23 ký tự. Có giá trị như sau :

```
0x31, 0x53, 0x01, 0x75, 0x03,
0x64, 0x05, 0x72, 0x07, 0x63,
0x09, 0x6F, 0x02, 0x6E, 0x04,
0x74, 0x06, 0x61, 0x08, 0x6B,
0x0A, 0x30, 0x30
```

Như vậy, chuỗi được dùng để mã hoá sẽ có chiều dài : 53 ký tự.

===== ***String to Encrypt*** =====

- Trace tiếp ta đến quá trình mã hoá :

```
00413F17 . E8 64480900 CALL soundtra.004A8780 ; <== Encrypt
```

===== ***EnCrypt*** =====

```
004A8794 |. E8 47FFFFFF CALL soundtra.004A86E0 ; <== MD5Hash
```

===== ***MD5Hash*** =====

```
004A86F6 |. E8 B5000000 CALL soundtra.004A87B0 ; <== MD5Start
```

```
004A86FB |. 8B4F 38    MOV ECX,DWORD PTR DS:[EDI+38]
```

```
004A86FE |. 8BC1      MOV EAX,ECX
```

```
004A8700 |. 83C4 04    ADD ESP,4
```

```
004A8703 |. 8D70 01    LEA ESI,DWORD PTR DS:[EAX+1]
```

```
004A8706 |> 8A10     /MOV DL,BYTE PTR DS:[EAX]
```

```
004A8708 |. 40       |INC EAX
```

```
004A8709 |. 84D2     |TEST DL,DL
```

```
004A870B |.^ 75 F9    \JNZ SHORT soundtra.004A8706
```

```
004A870D |. 2BC6     SUB EAX,ESI
```

```
004A870F |. 50       PUSH EAX
```

```
004A8710 |. 51       PUSH ECX
```

```
004A8711 |. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
```

```
004A8715 |. 51       PUSH ECX
```

```
004A8716 |. E8 A5090000 CALL soundtra.004A90C0 ; <== MD5Update
```

```
004A871B |. 8D5424 18  LEA EDX,DWORD PTR SS:[ESP+18]
```

```
004A871F |. 52       PUSH EDX
```

```
004A8720 |. 8D5F 05    LEA EBX,DWORD PTR DS:[EDI+5]
```

```
004A8723 |. 53       PUSH EBX
```

```
004A8724 |. E8 570A0000 CALL soundtra.004A9180 ; <== MD5Finished
```

===== ***MD5Hash*** =====

===== ***EnCrypt*** =====

- Kế đó, chương trình tiến hành mã hoá chuỗi MD5Hash này bằng cách (1) chọn lấy các ký tự ở vị trí chẵn (2) các vị trí thứ **4, 9 14** sẽ là ký tự “-“. Như vậy chuỗi này sẽ có dạng “XXXX-XXXX-XXXX”

```

00413F51 >/85DB TEST EBX,EBX
00413F53 . |OF8C D7020000 JL soundtra.00414230
00413F59 . |8B4424 38 MOV EAX,DWORD PTR SS:[ESP+38]
00413F5D . |3B58 F4 CMP EBX,DWORD PTR DS:[EAX-C]
00413F60 . |OF8F CA020000 JG soundtra.00414230
00413F66 . |8A0C03 MOV CL,BYTE PTR DS:[EBX+EAX]
00413F69 . |8B46 FC MOV EAX,DWORD PTR DS:[ESI-4]
00413F6C . |8B6E F4 MOV EBP,DWORD PTR DS:[ESI-C]
00413F6F . |884C24 2B MOV BYTE PTR SS:[ESP+2B],CL
00413F73 . |B9 01000000 MOV ECX,1
00413F78 . |2BC8 SUB ECX,EAX
00413F7A . |8B46 F8 MOV EAX,DWORD PTR DS:[ESI-8]
00413F7D . |8D7D 01 LEA EDI,DWORD PTR SS:[EBP+1]
00413F80 . |2BC7 SUB EAX,EDI
..... cutting .....
00413FB4 . |25 03000080 AND EAX,80000003
00413FB9 . |897E F4 MOV DWORD PTR DS:[ESI-C],EDI
00413FBC . |C60437 00 MOV BYTE PTR DS:[EDI+ESI],0
00413FC0 . |79 05 JNS SHORT soundtra.00413FC7
00413FC2 . |48 DEC EAX
00413FC3 . |83C8 FC OR EAX,FFFFFFFC
00413FC6 . |40 INC EAX
00413FC7 >|75 14 JNZ SHORT soundtra.00413FDD
00413FC9 . |6A 01 PUSH 1
00413FCB . |68 C0114E00 PUSH soundtra.004E11C0
00413FD0 . |8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
00413FD4 . |E8 5707FFFF CALL soundtra.00404730
00413FD9 . |8B7424 10 MOV ESI,DWORD PTR SS:[ESP+10]
00413FDD >|83C3 02 ADD EBX,2
00413FE0 . |83FB 20 CMP EBX,20
00413FE3 .^\0F8C 68FFFFFF JL soundtra.00413F51

```

- Sau đó chuỗi này sẽ được gắn vào sau 20 ký tự đầu tiên của chuỗi FakeSerial nhập . Hay nói cách khác, chương trình sẽ sử dụng 20 ký tự đầu tiên của chuỗi Serial để làm chuỗi mã hoá .

004140AA . E8 61E9FFFF CALL soundtra.00412A10	; <== Real Serial
004140AF . 8B7C24 20 MOV EDI,DWORD PTR SS:[ESP+20]	; <== Fake Serial
004140B3 . 8B7424 10 MOV ESI,DWORD PTR SS:[ESP+10]	; <== Real Serial
004140B7 . 57 PUSH EDI	
004140B8 . 56 PUSH ESI	
004140B9 . E8 29FE0900 CALL soundtra.004B3EE7	; <== Compare

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : QRBNWDFTGZEQWFPSGVW7663-ECEB-CBBC-A03D

III – KeyGen :

- /Section I /- Tạo chuỗi ngẫu nhiên gồm 20 ký tự .
- /Section II /- Gắn vào sau chuỗi ngẫu nhiên này chuỗi mặc định “**Soundtrack**” .
- /Section III /- Gắn chuỗi này vào sau chuỗi mặc định thứ hai .
- /Section IV /- Tạo chuỗi MD5Hash của chuỗi này.
- /Section V /- Lấy các ký tự ở vị trí chẵn chuỗi MD5Hash theo dạng “XXXX-XXXX-XXXX”

/Section VI /- Gắn chuỗi này vào sau 20 ký tự ngẫu nhiên đầu tiên : Chuỗi Serial thực .

IV – End of Tut :

- Finished – *August 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.xnview.com
Production :	Pierre-E Gougelet
SoftWare :	XnView for Windows 1.7.03
Copyright by :	Copyright © 1991-2004 Pierre-E Gougelet. All Rights Reserved.
Type :	Name / Serial
Packed :	ASPack 2.12b -> Alexey Solodovnikov
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

XnView for Windows 1.7.03

With *XnView* you can quickly and easily view, process and convert image files. The universal program is able to read more than 400 different formats and then convert them to other formats such as GIF, BMP, JPG, PNG, multi page TIFF. XnView can also display video formats, many digital camera formats and more exotic formats such as Amiga IFF and Calamus.

An additional features is that XnView is available on many operating systems e.g. Windows ?, LINUX, HP/UX. In addition, to change its display (e.g. headings, text) to over 40 different languages, including English, French and German.

I – Information :

- Dùng PEiD kiểm tra biết CrackMe bị PACK bằng *ASPack 2.12b -> Alexey Solodovnikov* . Sau khi UnPACK kiểm tra lại biết chương trình được viết bằng *Microsoft Visual C++ 6.0* .

- Nhập thử User và Fake User ta nhận được thông báo "Invalid registration" . Tuy nhiên không tìm thấy thông báo này trong Olly bằng cách thông thường . Dùng phương pháp STACK để xác định địa chỉ của chuỗi này :

00473D0E . 68 93130000 PUSH 1393 ; |RsrcID = STRING "Invalid registration"

- Dò ngược lên trên ta thấy chương trình có sử dụng hàm *GetDlgItemTextA* nên ta đặt BreakPoint tại hàm này :

00473CB5 . FFD7 CALL EDI ; \GetDlgItemTextA

II – Cracking :

- Load chương trình lên, nhập User và Fake Serial, chương trình dừng lại tại điểm đặt BP: Trace xuống chút ta đến :

00473CCA . 84C0 TEST AL,AL ; <== User atleast 1 chart

00473CCC . 0F84 3A010000 JE dump_.00473E0C

00473CD2 . 8A4424 10 MOV AL,BYTE PTR SS:[ESP+10]

```

00473CD6 . 84C0      TEST AL,AL          ; <== Serial atleast 1 chart
00473CD8 . 0F84 2E010000 JE dump_.00473E0C
00473CDE . 8D5424 08 LEA EDX,DWORD PTR SS:[ESP+8]
00473CE2 . 8D4424 70 LEA EAX,DWORD PTR SS:[ESP+70]
00473CE6 . 52        PUSH EDX
00473CE7 . 50        PUSH EAX
00473CE8 . E8 9384F9FF CALL dump_.0040C180 ; <== Trace Into here
- Dùng F7 trace into ta đến đoạn CODE mã hoá . Quá trình mã hoá này chia là năm giai đoạn . Được diễn
giải như sau :
- Giai đoạn thứ nhất :
0040C1A5 |. F3:A5      REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS>; <== Default Value
: ValueD
0040C1A7 |. 8BF0      MOV ESI,EAX
0040C1A9 |. 74 21      JE SHORT dump_.0040C1CC
0040C1AB |> 8A0C16    /MOV CL,BYTE PTR DS:[ESI+EDX] ; <== U[i]
0040C1AE |. 8AD9      |MOV BL,CL           ; <== U[i]
0040C1B0 |. 3298 ACBB5A00 |XOR BL,BYTE PTR DS:[EAX+5ABBAC] ; <== Value = U[i] xor
ValueD[j]
0040C1B6 |. 40        |INC EAX            ; <== j++
0040C1B7 |. 83F8 05    |CMP EAX,5          ; <== if ( j < 5 )
0040C1BA |. 881C16    |MOV BYTE PTR DS:[ESI+EDX],BL ; <== Value saved : : n_01U[i]
0040C1BD |. 8888 ABBC5A00 |MOV BYTE PTR DS:[EAX+5ABBAB],CL ; <== U[i] saved
0040C1C3 |. 75 02      |JNZ SHORT dump_.0040C1C7 ; <== then jmp
0040C1C5 |. 33C0      |XOR EAX,EAX         ; <== else ( j = 0x0)
0040C1C7 |> 46        |INC ESI             ; <== i++
0040C1C8 |. 3BF5      |CMP ESI,EBP          ; <== while ( i < Len.U )
0040C1CA |.^ 72 DF    |JB SHORT dump_.0040C1AB ; <== Continue Loop
- Giai đoạn thứ hai :
0040C1D4 |>/8A9F B1BB5A00 /MOV BL,BYTE PTR DS:[EDI+5ABBB1] ; <== ValueD[5+j]
0040C1DA |.|8BF5      |MOV ESI,EBP          ; <== Len.U
0040C1DC |.|2BF1      |SUB ESI,ECX         ; <== Len = Len.U - i
0040C1DE |.|4E        |DEC ESI             ; <== Len --
0040C1DF |.|8A0416    |MOV AL,BYTE PTR DS:[ESI+EDX] ; <== n_01U[Len]
0040C1E2 |.|32D8      |XOR BL,AL           ; <== Value = ValueD[5+j] xor n_01U[Len]
0040C1E4 |.|47        |INC EDI             ; <== j++
0040C1E5 |.|881C16    |MOV BYTE PTR DS:[ESI+EDX],BL ; <== Value saved : : n_02U[i]
0040C1E8 |.|8887 B0BB5A00 |MOV BYTE PTR DS:[EDI+5ABBB0],AL ; <== n_01U[Len]
0040C1EE |.|83FF 05    |CMP EDI,5           ; <== if ( j > 5 )
0040C1F1 |.|75 02      |JNZ SHORT dump_.0040C1F5 ; <== then
0040C1F3 |.|33FF      |XOR EDI,EDI         ; <== j = 0x0
0040C1F5 |>|41        |INC ECX             ; <== i++
0040C1F6 |.|3BCD      |CMP ECX,EBP          ; <== while ( i < Len.U )
0040C1F8 |.^|72 DA    |JB SHORT dump_.0040C1D4 ; <== Continue Loop
- Giai đoạn thứ ba :
0040C202 |>/8A0417    /MOV AL,BYTE PTR DS:[EDI+EDX] ; <== n_02U[i]
0040C205 |.|8A8E B6BB5A00 |MOV CL,BYTE PTR DS:[ESI+5ABBB6] ; <== ValueD[10+j]
0040C20B |.|32C8      |XOR CL,AL           ; <== Value = ValueD[10+j] xor n_02U[i]
0040C20D |.|46        |INC ESI             ; <== j++
0040C20E |.|880C17    |MOV BYTE PTR DS:[EDI+EDX],CL ; <== Value saved : : n_03U[i]
0040C211 |.|8886 B5BB5A00 |MOV BYTE PTR DS:[ESI+5ABBB5],AL ; <== n_02U[i]
0040C217 |.|83FE 05    |CMP ESI,5           ; <== if ( j > 5 )
0040C21A |.|75 02      |JNZ SHORT dump_.0040C21E ; <== then

```

- Giai đoạn thứ tư :

```

0040C21C |. |33F6    |XOR ESI,ESI          ; <== j = 0x0
0040C21E |> |47     |INC EDI           ; <== i++
0040C21F |. |3BFD    |CMP EDI,EBP        ; <== while ( i < Len.U )
0040C221 |.^|72 DF   |JB SHORT dump_.0040C202 ; <== Continue Loop

- Giai đoạn thứ năm và cũng là giai đoạn tạo chuỗi Serial :
0040C22B |>/8A9F BBBB5A00 |MOV BL,BYTE PTR DS:[EDI+5ABBBA] ; <== ValueD[15+j]
0040C231 |. |8BF5    |MOV ESI,EBP        ; <== Len.U
0040C233 |. |2BF1    |SUB ESI,ECX       ; <== Len = Len.U - i
0040C235 |. |4E      |DEC ESI           ; <== Len --
0040C236 |. |8A0416   |MOV AL,BYTE PTR DS:[ESI+EDX] ; <== n_03U[Len]
0040C239 |. |32D8    |XOR BL,AL          ; <== Value = ValueD[15+j) xor n_03U[Len]
0040C23B |. |47      |INC EDI           ; <== j++
0040C23C |. |881C16   |MOV BYTE PTR DS:[ESI+EDX],BL ; <== Value saved : : n_04U[i]
0040C23F |. |8887 BBBB5A00 |MOV BYTE PTR DS:[EDI+5ABBBA],AL ; <== n_03U[Len]
0040C245 |. |83FF 05   |CMP EDI,5          ; <== if ( j > 5 )
0040C248 |. |75 02    |JNZ SHORT dump_.0040C24C ; <== then
0040C24A |. |33FF    |XOR EDI,EDI        ; <== j = 0x0
0040C24C |> |41      |INC ECX           ; <== i++
0040C24D |. |3BCD    |CMP ECX,EBP        ; <== while ( i < Len.U )
0040C24F |.^|72 DA   |JB SHORT dump_.0040C22B ; <== Continue Loop

- Giai đoạn thứ năm và cũng là giai đoạn tạo chuỗi Serial :
0040C255 |. |33C0    XOR EAX,EAX        ; <== i = 0
0040C257 |. |85ED    TEST EBP,EBP
0040C259 |. C707 00000000 MOV DWORD PTR DS:[EDI],0
0040C25F |. 76 17    JBE SHORT dump_.0040C278
0040C261 |> |8BC8    /MOV ECX,EAX        ; <== i
0040C263 |. |83E1 03  |AND ECX,3          ; <== Temp = i and 3
0040C266 |. |8A1C39   |MOV BL,BYTE PTR DS:[ECX+EDI] ; <== S[Temp]
0040C269 |. |8D3439   |LEA ESI,DWORD PTR DS:[ECX+EDI]
0040C26C |. |8A0C10   |MOV CL,BYTE PTR DS:[EAX+EDX] ; <== n_04U[i]
0040C26F |. |02D9    |ADD BL,CL          ; <== Value = n_04U[i] + S[Temp]
0040C271 |. |40      |INC EAX           ; <== i++
0040C272 |. |3BC5    |CMP EAX,EBP        ; <== while ( i < Len.U )
0040C274 |. |881E    |MOV BYTE PTR DS:[ESI],BL ; <== S[Temp] = Value
0040C276 |.^|72 E9   |JB SHORT dump_.0040C261 ; <== Continue Loop

```

- Ghi nhớ rằng, kết thúc năm giai đoạn ta có được chuỗi ký tự . Sau đó, các ký tự này sẽ được chuyển thành giá trị HEX nhờ vào quá trình chuyển đổi :

- Đồng thời ghi nhớ thêm một điều, giá trị này được xác định ở dạng **SIGNED INTERGER**

/*/*/*/ - SERIAL tương ứng với USER :

User : REA-cRaCkErS Serial : -1667346522

III – KeyGen :

- /Section 1/- Xác định bảng giá trị mặc định
 - /Section 2/- Thực hiện 5 giai đoạn tính toán
 - /Section 3/- Chuyển đổi từ Chuỗi sang Giá Trị . Và xuất ra theo định dạng “%i”

IV – SourceCode (VC++) :

```
char reaName[64]={0};  
char reaSerial[64]={0};
```

```

int LenUser=0;
int i=0, j=0;
int Value = 0, Serial=0;
unsigned int reaDefaultTable[20]=  {
                                0xAA, 0x89, 0xC4, 0xFE, 0x46,
                                0x78, 0xF0, 0xD0, 0x03, 0xE7,
                                0xF7, 0xFD, 0xF4, 0xE7, 0xB9,
                                0xB5, 0x1B, 0xC9, 0x50, 0x73
                                };

LenUser=GetDlgItemText(IDC_Name, reaName, 64);
if (LenUser < 1)
{
    MessageBox("----- Your name atleast 1 chart ----- ", "Hey !! Please
input your name again !! ");
}
else
{
    i=0;    j=0;
    while ( i < LenUser )
    {
        Value = reaName[i] ^ reaDefaultTable[j];
        reaDefaultTable[j] = reaName[i] & 0xFF;
        reaName[i] = Value;
        if ( j < 4 )
        {
            j++;
        }
        else
        {
            j=0;
        }
        i++;
    }

    i=0;    j=0;
    while ( i < LenUser )
    {
        Value = reaDefaultTable[5 + j] ^ reaName[LenUser - 1 - i];
        reaDefaultTable[5 + j] = reaName[LenUser - 1 - i] & 0xFF;
        reaName[LenUser - 1 - i] = Value;
        if ( j < 4 )
        {
            j++;
        }
        else
        {
            j=0;
        }
        i++;
    }
}

```

```

i=0;    j=0;
while ( i < LenUser )
{
    Value = reaName[i] ^ reaDefaultTable[10 + j];
    reaDefaultTable[10 + j] = reaName[i] & 0xFF;
    reaName[i] = Value;
    if ( j < 4 )
    {
        j++;
    }
    else
    {
        j=0;
    }
    i++;
}

i=0;    j=0;
while ( i < LenUser )
{
    Value = reaDefaultTable[15 + j] ^ reaName[LenUser - 1 - i];
    reaDefaultTable[15 + j] = reaName[LenUser - 1 - i] & 0xFF;
    reaName[LenUser - 1 - i] = Value;
    if ( j < 4 )
    {
        j++;
    }
    else
    {
        j=0;
    }
    i++;
}
i=0;    j=0;

while ( i < LenUser )
{
    j = i & 3;
    reaSerial[j] = reaName[i] + reaSerial[j];
    i++;
}
asm
{
    MOV EAX, DWORD PTR reaSerial
    MOV Serial, EAX
}

SetDlgItemInt(IDC_Serial,Serial);
}

```

V – End of Tut :

- Finished - **[15/07/2004](#)**

Reverse Engineering Association

SoftWare

Homepage	:	http://www.guitar-pro.com
Production	:	Arobas Music.
SoftWare	:	Guitar Pro 4.1.0
Copyright by	:	Copyright © 1997-2004 Arobas Music. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N/A
Language	:	Borland Delphi 6.0 - 7.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N/A
Request	:	Correct Serial / KeyGen

Guitar Pro 4.1.0

Guitar Pro is a multitrack tablature editor for guitar, banjo & bass. Besides writing scores, Guitar Pro is a complete tool for young and accomplished guitarists alike to improve their skills, compose, or simply accompany themselves. Guitar Pro offers such options as listening, printing, importing and exporting MIDI and ASCII formats (thus exploiting thousands of scores on the internet), transposing, transcribing a score in standard notation, etc. Many useful functions are also available: chord diagrams, metronome, tuner, assistants, etc.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng **Borland Delphi 6.0 - 7.0**
- Chạy thử chương trình với Fake U và S ta nhận được thông báo "**Key is not valid.**" . Tuy nhiên ta không tìm được thông báo này . Dùng phương pháp STACK ta xác định được FUNCTION chính .
- Dò ngược lên trên ta đặt BreakPoint tại :
004D4C10 . E8 5BE6F9FF CALL GP4.00473270 ; <== Set breakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình , chương trình dừng lại tại điểm đặt BP . Trước tiên, chương trình kiểm tra 4 ký tự đầu tiên của U với các ký tự mặc định :
004D4C10 . E8 5BE6F9FF CALL GP4.00473270 ; <== Set breakPoint here
004D4C15 . 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
004D4C18 . 8D55 FC LEA EDX,DWORD PTR SS:[EBP-4]
004D4C1B . E8 B03E1700 CALL GP4.00648AD0
004D4C20 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
004D4C23 . B8 304F4D00 MOV EAX,GP4.004D4F30 ; ASCII "AR0B"
004D4C28 . E8 8B07F3FF CALL GP4.004053B8
004D4C2D . 85C0 TEST EAX,EAX
004D4C2F . 7E 21 JLE SHORT GP4.004D4C52
004D4C31 . 8D45 F4 LEA EAX,DWORD PTR SS:[EBP-C]
004D4C34 . 50 PUSH EAX
004D4C35 . B9 404F4D00 MOV ECX,GP4.004D4F40 ; ASCII "AROB"

```

004D4C3A . BA 304F4D00 MOV EDX,GP4.004D4F30 ; ASCII "AR0B"
004D4C3F . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
004D4C42 . E8 11351600 CALL GP4.00638158

```

- Kế đó, chương trình kết hợp 3 đoạn của Key ID thành một chuỗi duy nhất :

```

004D4C52 > \8D55 EC LEA EDX,DWORD PTR SS:[EBP-14]
004D4C55 . 8B83 18030000 MOV EAX,DWORD PTR DS:[EBX+318]
004D4C5B . E8 10E6F9FF CALL GP4.00473270 ; <== SecI
004D4C60 . FF75 EC PUSH DWORD PTR SS:[EBP-14]
004D4C63 . 8D55 E8 LEA EDX,DWORD PTR SS:[EBP-18]
004D4C66 . 8B83 20030000 MOV EAX,DWORD PTR DS:[EBX+320]
004D4C6C . E8 FFE5F9FF CALL GP4.00473270 ; <== SecII
004D4C71 . FF75 E8 PUSH DWORD PTR SS:[EBP-18]
004D4C74 . 8D55 E4 LEA EDX,DWORD PTR SS:[EBP-1C]
004D4C77 . 8B83 28030000 MOV EAX,DWORD PTR DS:[EBX+328]
004D4C7D . E8 EEE5F9FF CALL GP4.00473270 ; <== SecIII
004D4C82 . FF75 E4 PUSH DWORD PTR SS:[EBP-1C]
004D4C85 . 8D45 F0 LEA EAX,DWORD PTR SS:[EBP-10]
004D4C88 . BA 03000000 MOV EDX,3
004D4C8D . E8 A204F3FF CALL GP4.00405134 ; <== Serial ( all Sec )

```

- Quá trình mã hoá và so sánh đầu tiên diễn ra tại đây :

```

004D4C92 . 8B55 F0 MOV EDX,DWORD PTR SS:[EBP-10] ; <== S
004D4C95 . B9 90010000 MOV ECX,190
004D4C9A . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; <== U
004D4C9D . E8 B6381600 CALL GP4.00638558 ; <== Encrypt & Compare ( I )
004D4CA2 . 84C0 TEST AL,AL ; <== if Correct
004D4CA4 . 0F84 A4010000 JE GP4.004D4E4E ; <== Continue Check

```

----- == Encrypt & Compare (I) ==-----

Trước hết, chương trình tiến hành mã hoá chuỗi U nhập . Kết thúc quá trình này ta có giá trị **Value** :

```

006385C5 |. B9 01000000 MOV ECX,1
006385CA |> 8B45 FC /MOV EAX,[LOCAL.1] ; <== U
006385CD |. 49 |DEC ECX
006385CE |. 85C0 |TEST EAX,EAX
006385D0 |. 74 05 |JE SHORT GP4.006385D7
006385D2 |. 3B48 FC |CMP ECX,DWORD PTR DS:[EAX-4]
006385D5 |. 72 05 |JB SHORT GP4.006385DC
006385D7 |> E8 FCB8DCFF |CALL GP4.00403ED8
006385DC |> 41 |INC ECX
006385DD |. 0FB64408 FF |MOVZX EAX,BYTE PTR DS:[EAX+ECX-1] ; <== U[i]
006385E2 |. F7E9 |IMUL ECX ; <== Temp = U[i] * i
006385E4 |. 71 05 |JNO SHORT GP4.006385EB
006385E6 |. E8 F5B8DCFF |CALL GP4.00403EE0
006385EB |> 03D8 |ADD EBX,EAX ; <== Value = Value + Temp
006385ED |. 71 05 |JNO SHORT GP4.006385F4
006385EF |. E8 ECB8DCFF |CALL GP4.00403EE0
006385F4 |> 8BC3 |MOV EAX,EBX ; <== Value
006385F6 |. BB E8030000 |MOV EBX,3E8 ; <== dV
006385FB |. 99 |CDQ ; <== Value = Value % dV
006385FC |. F7FB |IDIV EBX ; <== Value
006385FE |. 8BDA |MOV EBX,EDX ; <== Value
00638600 |. 41 |INC ECX ; <== i++

```

```

00638601 |. 4E      |DEC ESI ; <== LenU--
00638602 |.^ 75 C6    |JNZ SHORT GP4.006385CA ; <== Loop until LenU = 0
00638604 |> 85DB     |TEST EBX,EBX ; <== if ( Value == 0x0 )
00638606 |. 75 05     |JNZ SHORT GP4.0063860D ; <== then
00638608 |. BB 01000000 MOV EBX,1 ; <== Value = 0x1
0063860D |> 8B45 F4    MOV EAX,[LOCAL.3] ; <== else dV
00638610 |. 2D DC000000 SUB EAX,0DC ; <== dV = 0xB4
00638615 |. 71 05     |JNO SHORT GP4.0063861C
00638617 |. E8 C4B8DCFF CALL GP4.00403EE0
0063861C |> 03C3     |ADD EAX,EBX ; <== Value = Value + dV
0063861E |. 71 05     |JNO SHORT GP4.00638625
00638620 |. E8 BBB8DCFF CALL GP4.00403EE0
00638625 |> 83C0 01    ADD EAX,1 ; <== Value = Value + 1
00638628 |. 71 05     |JNO SHORT GP4.0063862F
0063862A |. E8 B1B8DCFF CALL GP4.00403EE0
0063862F |> B9 E8030000 MOV ECX,3E8 ; <== dV
00638634 |. 99       |CDQ
00638635 |. F7F9     |IDIV ECX ; <== Value = Value % dV
00638637 |. 8BDA     |MOV EBX,EDX ; <== Value
00638639 |. 85DB     |TEST EBX,EBX ; <== if ( Value == 0x0 )
0063863B |. 75 05     |JNZ SHORT GP4.00638642 ; <== then
0063863D |. BB 01000000 MOV EBX,1 ; <== Value = 0x1

```

Kết quả, chương trình tiến hành mã hóa và so sánh từng ký tự. Quá trình mã hóa này thực chất là quá trình mã hóa cho đoạn thứ 3 của Key ID dựa trên giá trị Value và giá trị các ký tự trong hai Sec kia của chuỗi Key ID :

```

00638642 |> 8D45 EC    LEA EAX,[LOCAL.5]
00638645 |. BA 0B000000 MOV EDX,0B
0063864A |. 8B4D F8    MOV ECX,[LOCAL.2]
0063864D |. 4A       |DEC EDX
0063864E |. 85C9     |TEST ECX,ECX
00638650 |. 74 05     |JE SHORT GP4.00638657
00638652 |. 3B51 FC    CMP EDX,DWORD PTR DS:[ECX-4]
00638655 |. 72 05     |JB SHORT GP4.0063865C
00638657 |> E8 7CB8DCFF CALL GP4.00403ED8
0063865C |> 42       |INC EDX
0063865D |. 8A5411 FF    MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[10]
00638661 |. E8 36C9DCFF CALL GP4.00404F9C
00638666 |. 8B45 EC    MOV EAX,[LOCAL.5]
00638669 |. 50       |PUSH EAX
0063866A |. 8D45 E4    LEA EAX,[LOCAL.7]
0063866D |. BA 01000000 MOV EDX,1
00638672 |. 8B4D F8    MOV ECX,[LOCAL.2] ; <== S
00638675 |. 4A       |DEC EDX
00638676 |. 85C9     |TEST ECX,ECX
00638678 |. 74 05     |JE SHORT GP4.0063867F
0063867A |. 3B51 FC    CMP EDX,DWORD PTR DS:[ECX-4]
0063867D |. 72 05     |JB SHORT GP4.00638684
0063867F |> E8 54B8DCFF CALL GP4.00403ED8
00638684 |> 42       |INC EDX
00638685 |. 8A5411 FF    MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[0]
00638689 |. E8 0EC9DCFF CALL GP4.00404F9C

```

```

0063868E |. 8B45 E4      MOV EAX,[LOCAL.7]
00638691 |. E8 D615DDFF  CALL GP4.00409C6C          ; <== Temp01 = S[0] - 0x30
00638696 |. 8BF0      MOV ESI,EAX
00638698 |. 8D45 E0      LEA EAX,[LOCAL.8]
0063869B |. BA 02000000  MOV EDX,2
006386A0 |. 8B4D F8      MOV ECX,[LOCAL.2]
006386A3 |. 4A        DEC EDX
006386A4 |. 85C9      TEST ECX,ECX
006386A6 |. 74 05      JE SHORT GP4.006386AD
006386A8 |. 3B51 FC      CMP EDX,DWORD PTR DS:[ECX-4]
006386AB |. 72 05      JB SHORT GP4.006386B2
006386AD |> E8 26B8DCFF  CALL GP4.00403ED8
006386B2 |> 42        INC EDX
006386B3 |. 8A5411 FF      MOV DL,BYTE PTR DS:[ECX+EDX-1]    ; <== S[1]
006386B7 |. E8 E0C8DCFF  CALL GP4.00404F9C
006386BC |. 8B45 E0      MOV EAX,[LOCAL.8]
006386BF |. E8 A815DDFF  CALL GP4.00409C6C          ; <== Temp02 = S[1] - 0x30
006386C4 |. 03F0      ADD ESI,EAX
006386C6 |. 71 05      JNO SHORT GP4.006386CD          ; <== Value02 = Temp01 + Temp02
006386C8 |. E8 13B8DCFF  CALL GP4.00403EE0
006386CD |> 8BC6      MOV EAX,ESI          ; <== Value02
006386CF |. F7EB      IMUL EBX
006386D1 |. 71 05      JNO SHORT GP4.006386D8
006386D3 |. E8 08B8DCFF  CALL GP4.00403EE0
006386D8 |> B9 0A000000  MOV ECX,0A          ; <== dV
006386DD |. 99        CDQ
006386DE |. F7F9      IDIV ECX          ; <== Value02 = Value02 % dV
006386E0 |. 8BC2      MOV EAX,EDX          ; <== Value02
006386E2 |. 8D55 E8      LEA EDX,[LOCAL.6]
006386E5 |. E8 1614DDFF  CALL GP4.00409B00
006386EA |. 8B55 E8      MOV EDX,[LOCAL.6]          ; <== Value02
006386ED |. 58        POP EAX          ; <== S[10]
006386EE |. E8 CDCADCFF  CALL GP4.004051C0          ; <== if ( S[10] == Value02 )
006386F3 0F85 DC020000 JNZ GP4.006389D5          ; <== Continue Check
006386F9 |. 8D45 DC      LEA EAX,[LOCAL.9]
006386FC |. BA 0C000000  MOV EDX,0C
00638701 |. 8B4D F8      MOV ECX,[LOCAL.2]
00638704 |. 4A        DEC EDX
00638705 |. 85C9      TEST ECX,ECX
00638707 |. 74 05      JE SHORT GP4.0063870E
00638709 |. 3B51 FC      CMP EDX,DWORD PTR DS:[ECX-4]
0063870C |. 72 05      JB SHORT GP4.00638713
0063870E |> E8 C5B7DCFF  CALL GP4.00403ED8
00638713 |> 42        INC EDX
00638714 |. 8A5411 FF      MOV DL,BYTE PTR DS:[ECX+EDX-1]    ; <== S[11]
00638718 |. E8 7FC8DCFF  CALL GP4.00404F9C
0063871D |. 8B45 DC      MOV EAX,[LOCAL.9]
00638720 |. 50        PUSH EAX
00638721 |. 8D45 D4      LEA EAX,[LOCAL.11]
00638724 |. BA 03000000  MOV EDX,3
00638729 |. 8B4D F8      MOV ECX,[LOCAL.2]
0063872C |. 4A        DEC EDX

```

```

0063872D |. 85C9      TEST ECX,ECX
0063872F |. 74 05     JE SHORT GP4.00638736
00638731 |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
00638734 |. 72 05     JB SHORT GP4.0063873B
00638736 > E8 9DB7DCFF CALL GP4.00403ED8
0063873B > 42       INC EDX
0063873C |. 8A5411 FF  MOV DL,BYTE PTR DS:[ECX+EDX-1]    ; <== S[2]
00638740 |. E8 57C8DCFF CALL GP4.00404F9C
00638745 |. 8B45 D4   MOV EAX,[LOCAL.11]
00638748 |. E8 1F15DDFF CALL GP4.00409C6C           ; <== Temp01 = S[2] - 0x30
0063874D |. 8BF0       MOV ESI,EAX
0063874F |. 8D45 D0   LEA EAX,[LOCAL.12]
00638752 |. BA 04000000 MOV EDX,4
00638757 |. 8B4D F8   MOV ECX,[LOCAL.2]
0063875A |. 4A       DEC EDX
0063875B |. 85C9      TEST ECX,ECX
0063875D |. 74 05     JE SHORT GP4.00638764
0063875F |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
00638762 |. 72 05     JB SHORT GP4.00638769
00638764 > E8 6FB7DCFF CALL GP4.00403ED8
00638769 > 42       INC EDX
0063876A |. 8A5411 FF  MOV DL,BYTE PTR DS:[ECX+EDX-1]    ; <== S[3]
0063876E |. E8 29C8DCFF CALL GP4.00404F9C
00638773 |. 8B45 D0   MOV EAX,[LOCAL.12]
00638776 |. E8 F114DDFF CALL GP4.00409C6C           ; <== Temp02 = S[3] - 0x30
0063877B |. 03F0       ADD ESI,EAX
0063877D |. 71 05     JNO SHORT GP4.00638784
0063877F |. E8 5CB7DCFF CALL GP4.00403EE0
00638784 > 8BC6       MOV EAX,ESI
00638786 |. F7EB       IMUL EBX
00638788 |. 71 05     JNO SHORT GP4.0063878F
0063878A |. E8 51B7DCFF CALL GP4.00403EE0
0063878F > B9 0A000000 MOV ECX,0A
00638794 |. 99       CDQ
00638795 |. F7F9       IDIV ECX
00638797 |. 8BC2       MOV EAX,EDX
00638799 |. 8D55 D8   LEA EDX,[LOCAL.10]
0063879C |. E8 5F13DDFF CALL GP4.00409B00
006387A1 |. 8B55 D8   MOV EDX,[LOCAL.10]
006387A4 |. 58       POP EAX
006387A5 |. E8 16CADCF0 CALL GP4.004051C0
006387AA 0F85 25020000 JNZ GP4.006389D5
006387B0 |. 8D45 CC   LEA EAX,[LOCAL.13]
006387B3 |. BA 0D000000 MOV EDX,0D
006387B8 |. 8B4D F8   MOV ECX,[LOCAL.2]
006387BB |. 4A       DEC EDX
006387BC |. 85C9      TEST ECX,ECX
006387BE |. 74 05     JE SHORT GP4.006387C5
006387C0 |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
006387C3 |. 72 05     JB SHORT GP4.006387CA
006387C5 > E8 0EB7DCFF CALL GP4.00403ED8
006387CA > 42       INC EDX

```

```

006387CB |. 8A5411 FF  MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[12]
006387CF |. E8 C8C7DCFF CALL GP4.00404F9C
006387D4 |. 8B45 CC  MOV EAX,[LOCAL.13]
006387D7 |. 50      PUSH EAX
006387D8 |. 8D45 C4  LEA EAX,[LOCAL.15]
006387DB |. BA 05000000 MOV EDX,5
006387E0 |. 8B4D F8  MOV ECX,[LOCAL.2]
006387E3 |. 4A      DEC EDX
006387E4 |. 85C9      TEST ECX,ECX
006387E6 |. 74 05    JE SHORT GP4.006387ED
006387E8 |. 3B51 FC  CMP EDX,DWORD PTR DS:[ECX-4]
006387EB |. 72 05    JB SHORT GP4.006387F2
006387ED |> E8 E6B6DCFF CALL GP4.00403ED8
006387F2 |> 42      INC EDX
006387F3 |. 8A5411 FF  MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[4]
006387F7 |. E8 A0C7DCFF CALL GP4.00404F9C
006387FC |. 8B45 C4  MOV EAX,[LOCAL.15]
006387FF |. E8 6814DDFF CALL GP4.00409C6C ; <== Temp01 = S[4] - 0x30
00638804 |. 8BF0      MOV ESI,EAX ; <== Temp01
00638806 |. 8D45 C0  LEA EAX,[LOCAL.16]
00638809 |. BA 06000000 MOV EDX,6
0063880E |. 8B4D F8  MOV ECX,[LOCAL.2]
00638811 |. 4A      DEC EDX
00638812 |. 85C9      TEST ECX,ECX
00638814 |. 74 05    JE SHORT GP4.0063881B
00638816 |. 3B51 FC  CMP EDX,DWORD PTR DS:[ECX-4]
00638819 |. 72 05    JB SHORT GP4.00638820
0063881B |> E8 B8B6DCFF CALL GP4.00403ED8
00638820 |> 42      INC EDX
00638821 |. 8A5411 FF  MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[5]
00638825 |. E8 72C7DCFF CALL GP4.00404F9C
0063882A |. 8B45 C0  MOV EAX,[LOCAL.16]
0063882D |. E8 3A14DDFF CALL GP4.00409C6C ; <== Temp02 = S[5] - 0x30
00638832 |. 03F0      ADD ESI,EAX ; <== Value02 = Temp01 + Temp02
00638834 |. 71 05    JNO SHORT GP4.0063883B
00638836 |. E8 A5B6DCFF CALL GP4.00403EE0
0063883B |> 8BC6      MOV EAX,ESI ; <== Value02
0063883D |. F7EB      IMUL EBX ; <== Value02 = Value02 * Value
0063883F |. 71 05    JNO SHORT GP4.00638846
00638841 |. E8 9AB6DCFF CALL GP4.00403EE0
00638846 |> B9 0A000000 MOV ECX,0A ; <== dV
0063884B |. 99      CDQ
0063884C |. F7F9      IDIV ECX ; <== Value02 = Value02 % dV
0063884E |. 8BC2      MOV EAX,EDX ; <== Value02
00638850 |. 8D55 C8  LEA EDX,[LOCAL.14]
00638853 |. E8 A812DDFF CALL GP4.00409B00
00638858 |. 8B55 C8  MOV EDX,[LOCAL.14] ; <== Value02
0063885B |. 58      POP EAX ; <== S[12]
0063885C |. E8 5FC9DCFF CALL GP4.004051C0 ; <== if ( S[12] == Value02 )
00638861 0F85 6E010000 JNZ GP4.006389D5 ; <== Continue Check
00638867 |. 8D45 BC  LEA EAX,[LOCAL.17]
0063886A |. BA 0E000000 MOV EDX,0E

```

```

0063886F |. 8B4D F8    MOV ECX,[LOCAL.2]
00638872 |. 4A        DEC EDX
00638873 |. 85C9      TEST ECX,ECX
00638875 |. 74 05     JE SHORT GP4.0063887C
00638877 |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
0063887A |. 72 05     JB SHORT GP4.00638881
0063887C |> E8 57B6DCFF CALL GP4.00403ED8
00638881 |> 42        INC EDX
00638882 |. 8A5411 FF   MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[13]
00638886 |. E8 11C7DCFF CALL GP4.00404F9C
0063888B |. 8B45 BC   MOV EAX,[LOCAL.17]
0063888E |. 50        PUSH EAX
0063888F |. 8D45 B4   LEA EAX,[LOCAL.19]
00638892 |. BA 07000000 MOV EDX,7
00638897 |. 8B4D F8   MOV ECX,[LOCAL.2]
0063889A |. 4A        DEC EDX
0063889B |. 85C9      TEST ECX,ECX
0063889D |. 74 05     JE SHORT GP4.006388A4
0063889F |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
006388A2 |. 72 05     JB SHORT GP4.006388A9
006388A4 |> E8 2FB6DCFF CALL GP4.00403ED8
006388A9 |> 42        INC EDX
006388AA |. 8A5411 FF   MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[6]
006388AE |. E8 E9C6DCFF CALL GP4.00404F9C
006388B3 |. 8B45 B4   MOV EAX,[LOCAL.19]
006388B6 |. E8 B113DDFF CALL GP4.00409C6C ; <== Temp01 = S[6] - 0x30
006388BB |. 8BF0        MOV ESI,EAX ; <== Temp01
006388BD |. 8D45 B0   LEA EAX,[LOCAL.20]
006388C0 |. BA 08000000 MOV EDX,8
006388C5 |. 8B4D F8   MOV ECX,[LOCAL.2]
006388C8 |. 4A        DEC EDX
006388C9 |. 85C9      TEST ECX,ECX
006388CB |. 74 05     JE SHORT GP4.006388D2
006388CD |. 3B51 FC   CMP EDX,DWORD PTR DS:[ECX-4]
006388D0 |. 72 05     JB SHORT GP4.006388D7
006388D2 |> E8 01B6DCFF CALL GP4.00403ED8
006388D7 |> 42        INC EDX
006388D8 |. 8A5411 FF   MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[7]
006388DC |. E8 BBC6DCFF CALL GP4.00404F9C
006388E1 |. 8B45 B0   MOV EAX,[LOCAL.20]
006388E4 |. E8 8313DDFF CALL GP4.00409C6C ; <== Temp02 = S[7] - 0x30
006388E9 |. 03F0        ADD ESI,EAX ; <== Value02 = Temp01 + Temp02
006388EB |. 71 05     JNO SHORT GP4.006388F2
006388ED |. E8 EEB5DCFF CALL GP4.00403EE0
006388F2 |> 8BC6        MOV EAX,ESI ; <== Value02
006388F4 |. F7EB        IMUL EBX ; <== Value02 = Value02 * Value
006388F6 |. 71 05     JNO SHORT GP4.006388FD
006388F8 |. E8 E3B5DCFF CALL GP4.00403EE0
006388FD |> B9 0A000000 MOV ECX,0A ; <== dV
00638902 |. 99        CDQ ; <== Value02 = Value02 % dV
00638903 |. F7F9        IDIV ECX ; <== Value02
00638905 |. 8BC2        MOV EAX,EDX ; <== Value02

```

```

00638907 |. 8D55 B8    LEA EDX,[LOCAL.18]
0063890A |. E8 F111DDFF CALL GP4.00409B00
0063890F |. 8B55 B8    MOV EDX,[LOCAL.18]          ; <== Value02
00638912 |. 58        POP EAX                      ; <== S[13]
00638913 |. E8 A8C8DCFF CALL GP4.004051C0          ; <== if ( S[13] == Value02 )
00638918 0F85 B7000000 JNZ GP4.006389D5          ; <== Continue Check
0063891E |. 8D45 AC    LEA EAX,[LOCAL.21]
00638921 |. BA 0F000000 MOV EDX,0F
00638926 |. 8B4D F8    MOV ECX,[LOCAL.2]
00638929 |. 4A        DEC EDX
0063892A |. 85C9      TEST ECX,ECX
0063892C |. 74 05     JE SHORT GP4.00638933
0063892E |. 3B51 FC    CMP EDX,DWORD PTR DS:[ECX-4]
00638931 |. 72 05     JB SHORT GP4.00638938
00638933 |> E8 A0B5DCFF CALL GP4.00403ED8
00638938 |> 42        INC EDX
00638939 |. 8A5411 FF    MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[14]
0063893D |. E8 5AC6DCFF CALL GP4.00404F9C
00638942 |. 8B45 AC    MOV EAX,[LOCAL.21]
00638945 |. 50        PUSH EAX
00638946 |. 8D45 A4    LEA EAX,[LOCAL.23]
00638949 |. BA 09000000 MOV EDX,9
0063894E |. 8B4D F8    MOV ECX,[LOCAL.2]
00638951 |. 4A        DEC EDX
00638952 |. 85C9      TEST ECX,ECX
00638954 |. 74 05     JE SHORT GP4.0063895B
00638956 |. 3B51 FC    CMP EDX,DWORD PTR DS:[ECX-4]
00638959 |. 72 05     JB SHORT GP4.00638960
0063895B |> E8 78B5DCFF CALL GP4.00403ED8
00638960 |> 42        INC EDX
00638961 |. 8A5411 FF    MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[8]
00638965 |. E8 32C6DCFF CALL GP4.00404F9C
0063896A |. 8B45 A4    MOV EAX,[LOCAL.23]
0063896D |. E8 FA12DDFF CALL GP4.00409C6C          ; <== Temp01 = S[8] - 0x30
00638972 |. 8BF0      MOV ESI,EAX          ; <== Temp01
00638974 |. 8D45 A0    LEA EAX,[LOCAL.24]
00638977 |. BA 0A000000 MOV EDX,0A
0063897C |. 8B4D F8    MOV ECX,[LOCAL.2]
0063897F |. 4A        DEC EDX
00638980 |. 85C9      TEST ECX,ECX
00638982 |. 74 05     JE SHORT GP4.00638989
00638984 |. 3B51 FC    CMP EDX,DWORD PTR DS:[ECX-4]
00638987 |. 72 05     JB SHORT GP4.0063898E
00638989 |> E8 4AB5DCFF CALL GP4.00403ED8
0063898E |> 42        INC EDX
0063898F |. 8A5411 FF    MOV DL,BYTE PTR DS:[ECX+EDX-1] ; <== S[9]
00638993 |. E8 04C6DCFF CALL GP4.00404F9C
00638998 |. 8B45 A0    MOV EAX,[LOCAL.24]
0063899B |. E8 CC12DDFF CALL GP4.00409C6C          ; <== Temp02 = S[9] - 0x30
006389A0 |. 03F0      ADD ESI,EAX          ; <== Value02 = Temp01 + Temp02
006389A2 |. 71 05     JNO SHORT GP4.006389A9
006389A4 |. E8 37B5DCFF CALL GP4.00403EE0

```

```

006389A9 |> 8BC6      MOV EAX,ESI          ; <== Value02
006389AB |. F7EB      IMUL EBX           ; <== Value02 = Value02 * Value
006389AD |. 71 05     JNO SHORT GP4.006389B4
006389AF |. E8 2CB5DCFF CALL GP4.00403EE0
006389B4 |> B9 0A000000 MOV ECX,0A        ; <== dV
006389B9 |. 99       CDQ
006389BA |. F7F9      IDIV ECX           ; <== Value02 = Value02 % dV
006389BC |. 8BC2      MOV EAX,EDX         ; <== Value02
006389BE |. 8D55 A8    LEA EDX,[LOCAL.22]
006389C1 |. E8 3A11DDFF CALL GP4.00409B00
006389C6 |. 8B55 A8    MOV EDX,[LOCAL.22]   ; <== Value02
006389C9 |. 58       POP EAX            ; <== S[14]
006389CA |. E8 F1C7DCFF CALL GP4.004051C0   ; <== if ( S[14] == Value02 )
006389CF 75 04     JNZ SHORT GP4.006389D5   ; <== Continue Check
006389D1 |. C645 F3 01    MOV BYTE PTR SS:[EBP-D],1
006389D5 |> 33C0      XOR EAX,EAX

```

===== Encrypt & Compare (I) =====

- Kết thúc quá trình trên, ta tiếp đến quá trình so sánh thứ hai :

```

004D4CF5 . E8 5A281700 CALL GP4.00647554 ; <== Encrypt & Compare ( II )
004D4CFA . 84C0      TEST AL,AL          ; <== if Correct
004D4CFC . 0F84 4C010000 JE GP4.004D4E4E ; <== Congrat !!!!!

```

===== Encrypt & Compare (II) =====

Đầu tiên, chương trình sẽ lấy ký tự thứ **I0** (**S[9]**) của chuỗi S, và chuyển sang dạng số . Đây cũng chính là chiều dài mà chương trình sẽ tiến hành cắt chuỗi , quá trình này sẽ được đề cập ở sau :

```

006475BD |. B9 01000000 MOV ECX,1          ; <== get 1 char
006475C2 |. BA 0A000000 MOV EDX,0A         ; <== from [10]
006475C7 |. 8B45 F8    MOV EAX,[LOCAL.2]    ; <== of Serial
006475CA |. E8 05DDDBFF CALL GP4.004052D4   ; <== ripping
006475CF |. 8B45 BC    MOV EAX,[LOCAL.17]   ; <== S[9]
006475D2 |. E8 9526DCFF CALL GP4.00409C6C   ; <== Len = S[9] - 0x30
006475D7 |. 8BC8      MOV ECX,EAX          ; <== Len
006475D9 |. BA 01000000 MOV EDX,1          ; <== from the 1st chart
006475DE |. 8B45 F8    MOV EAX,[LOCAL.2]    ; <== of Serial
006475E1 |. E8 EEDCDBFF CALL GP4.004052D4   ; <== Cutting String : cStr

```

Ké đó, chương trình thực hiện lại quá trình tạo giá trị **Value** .

```

006475F2 |. 85DB      TEST EBX,EBX
006475F4 |. 7E 47      JLE SHORT GP4.0064763D
006475F6 |. C745 F0 01000>MOV [LOCAL.4],1
006475FD |> 8B45 F0    /MOV EAX,[LOCAL.4]
00647600 |. 8B55 FC    |MOV EDX,[LOCAL.1]
00647603 |. 48       |DEC EAX
00647604 |. 85D2      |TEST EDX,EDX
00647606 |. 74 05      |JE SHORT GP4.0064760D
00647608 |. 3B42 FC    |CMP EAX,DWORD PTR DS:[EDX-4]
0064760B |. 72 05      |JB SHORT GP4.00647612
0064760D |> E8 C6C8DBFF |CALL GP4.00403ED8
00647612 |> 40       |INC EAX
00647613 |. 0FB64402 FF |MOVZX EAX,BYTE PTR DS:[EDX+EAX-1]
00647618 |. F76D F0    |IMUL [LOCAL.4]
0064761B |. 71 05      |JNO SHORT GP4.00647622
..... rippling .....

```

```

00647654 > 03C7      ADD EAX,EDI
00647656 |. 71 05      JNO SHORT GP4.0064765D
00647658 |. E8 83C8DBFF CALL GP4.00403EE0
0064765D > 83C0 01    ADD EAX,1
00647660 |. 71 05      JNO SHORT GP4.00647667
00647662 |. E8 79C8DBFF CALL GP4.00403EE0
00647667 > B9 E8030000 MOV ECX,3E8
0064766C |. 99        CDQ
0064766D |. F7F9      IDIV ECX
0064766F |. 8BFA      MOV EDI,EDX
00647671 |. 85FF      TEST EDI,EDI
00647673 |. 75 05      JNZ SHORT GP4.0064767A
00647675 |. BF 01000000 MOV EDI,1

```

Chuỗi **cStr** sẽ được chuyển sang dạng giá trị HEX tương ứng (“123” chuyển thành hValue = 0x7B). Sau đó thực hiện phép chia giữa **hValue** và **Value** (phép chia được thực hiện thông qua các Floating Point Number). Kết quả của phép chia này phải thoả hai điều kiện (1) kết quả này phải lớn hơn giá trị mặc định (2) và không có số dư – nghĩa là chia hết.

```

0064767A > \8B45 F8      MOV EAX,[LOCAL.2] ; <== cStr
0064767D |. E8 EA25DCFF CALL GP4.00409C6C ; <== Convert to HEX value
00647682 |. 8945 B8      MOV [LOCAL.18],EAX ; <== hCValue
00647685 |. DB45 B8      FILD [LOCAL.18] ; <== Convert to FPN : fCValue
00647688 |. 897D B4      MOV [LOCAL.19],EDI ; <== Value
0064768B |. DB45 B4      FILD [LOCAL.19] ; <== Convert to FPN : fValue
0064768E |. DEF9        FDIVP ST(1),ST ; <== fCheck = fCValue / fValue
00647690 |. E8 5BB6DBFF CALL GP4.00402CF0 ; <== Convert to HEX value
00647695 |. 50        PUSH EAX ; <== hCheck
00647696 |. C1F8 1F      SAR EAX,1F
00647699 |. 3BC2        CMP EAX,EDX
0064769B |. 58        POP EAX
0064769C |. 74 05      JE SHORT GP4.006476A3
0064769E |. E8 35C8DBFF CALL GP4.00403ED8
006476A3 > 8BD8      MOV EBX,EAX
006476A5 |. 81FB A0860100 CMP EBX,186A0 ; <== if ( hCheck > 0x186A0 )
006476AB |. 0F8C 1B010000 JL GP4.006477CC ; <== Continue Check
006476B1 |. 8B45 F8      MOV EAX,[LOCAL.2] ; <== cStr
006476B4 |. E8 B325DCFF CALL GP4.00409C6C ; <== Convert to HEX value
006476B9 |. 8945 B8      MOV [LOCAL.18],EAX ; <== hCValue
006476BC |. DB45 B8      FILD [LOCAL.18] ; <== Convert to FPN : fCValue
006476BF |. 897D B4      MOV [LOCAL.19],EDI ; <== Value
006476C2 |. DB45 B4      FILD [LOCAL.19] ; <== Convert to FPN : fValue
006476C5 |. DEF9        FDIVP ST(1),ST ; <== fLast = fCValue / fValue
006476C7 |. 895D B0      MOV [LOCAL.20],EBX ; <== hCheck
006476CA |. DB45 B0      FILD [LOCAL.20] ; <== Convert to FPN : fCheck
006476CD |. DED9        FCOMPP ; <== Must be same
006476CF |. DFE0        FSTSW AX ; <== / it mean that
006476D1 |. 9E        SAHF ; <== | fCValue = X * fValue
006476D2 |. 0F85 F4000000 JNZ GP4.006477CC ; <== \ no remainder

```

===== Encrypt & Compare (II) =====

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

User ID : AROB638433

Key ID : 02979-50409-84666

III – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.iolo.com
Production :	iolo technologies, LLC.
SoftWare :	DriveScrubber Professional 2.0a
Copyright by :	Copyright © 1999-2002 iolo technologies, LLC. All Rights Reserved.
Type :	Name / Serial
Packed :	ASPack 2.1 -> Alexey Solodovnikov
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

DriveScrubber Professional 2.0a

Formatting a disk is one of the most common ways to erase all files from the disk. It is commonly assumed that after the format procedure, all files are erased and inaccessible, and may essentially be forgotten about..

I – Information :

- Dùng PEiD kiểm tra biết chương trình bị PACK bằng ASPack 2.1 -> Alexey Solodovnikov và biết chương trình được viết bằng **Borland Delphi 4.0 - 5.0**
- Chạy thử chương trình với Fake Serial ta nhận được thông báo "**Invalid Key !!**" . Tuy nhiên ta không tìm được thông báo này . Dùng phương pháp STACK ta xác định được FUNCTION chính .
- Dò ngược lên trên và đặt BreakPoint tại :

0047BEC2 . E8 F181FEFF CALL _Install.004640B8 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút ta đến quá trình kiểm tra đầu tiên . Chương trình sẽ kiểm tra U và S nhập có trùng với một trong những U và P nằm trong BlackList không, nếu có thì Restart :

```
0047BF22 . E8 F5D1FFFF CALL _Install.0047911C ; <== Not In Black List
0047BF27 . 84C0      TEST AL,AL ; <== if NOT
0047BF29 . 74 11      JE SHORT _Install.0047BF3C ; <== Continue check
0047BF2B . E8 5CD5FFFF CALL _Install.0047948C
```

- Kế tiếp, chương trình sẽ Capture các chương trình đang hoạt động, tìm xem tên các chương trình này có các ký tự nằm trong BlackList không, nếu có thì Restart :

```
0047BF3C > \E8 3BD9FFFF CALL _Install.0047987C ; <== Capture and must in BlackList
0047BF41 . 84C0      TEST AL,AL ; <== if NOT
```

0047BF43 74 0C JE SHORT _Install.0047BF51 ; <== Continnue check

- Kế đó là quá trình kiểm tra chuỗi U. U phải được nhập vào :

0047BF55 . E8 6EF8FFFF CALL _Install.0047B7C8 ; <== U must be Enter

0047BF5A . 84C0 TEST AL,AL

0047BF5C . 0F84 91000000 JE _Install.0047BFF3

- Chương trình này tạo hai phiên bản (1) Personal (2) Profesional . Cả hai đều sử dụng một thuật toán mã hoá như nhau . Chỉ chỉnh sửa đôi chút cho có khác biệt . Ở đây ta sét quá trình mã hoá Profesional :

0047BF76 . BA 01000000 MOV EDX,1	; <== Personnal
0047BF7B . E8 1CDBFFFF CALL _Install.00479A9C	; <== FirstSerial
0047BF80 . 8B45 E4 MOV EAX,DWORD PTR SS:[EBP-1C]	; <== RealSerial
0047BF83 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]	; <== FakeSerial
0047BF86 . E8 CD7EF8FF CALL _Install.00403E58	; <== Compare
0047BF8B . 75 05 JNZ SHORT _Install.0047BF92	
0047BF8D . BE 01000000 MOV ESI,1	
0047BF92 > 8D55 D8 LEA EDX,DWORD PTR SS:[EBP-28]	
0047BF95 . 8B83 D8020000 MOV EAX,DWORD PTR DS:[EBX+2D8]	
0047BF9B . E8 3017FBFF CALL _Install.0042D6D0	
0047BFA0 . 8B45 D8 MOV EAX,DWORD PTR SS:[EBP-28]	
0047BFA3 . 8D4D DC LEA ECX,DWORD PTR SS:[EBP-24]	
0047BFA6 . BA 02000000 MOV EDX,2	; <== Profesional
0047BFAB . E8 ECDAFFFF CALL _Install.00479A9C	; <== SecondSerial
0047BFB0 . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24]	; <== RealSerial
0047BFB3 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]	; <== FakeSerial
0047BFB6 . E8 9D7EF8FF CALL _Install.00403E58	; <== Compare

- Trace Into vào quá trình tạo chuỗi Serial cho dạng Profesional . Trước tiên chương trình sẽ kiểm tra chiều dài chuỗi U nhập . Nếu (LenU > 10) thì bỏ qua bước này nhưng nếu (LenU < 10) thì U sẽ được thêm vào một số ký tự để chiều dài đủ 10 ký tự . Và sau đó chuyển U thành dạng UpperCase :

00479ADF |> \BF 1F000000 MOV EDI,1F

00479AE4 |> BB 21000000 MOV EBX,21

00479AE9 |. EB 16 JMP SHORT _Install.00479B01

00479AEB |> 8D45 E8 /LEA EAX,[LOCAL.6]

00479AEE |. 8BD3 |MOV EDX,EBX

00479AF0 |. E8 7BA1F8FF |CALL _Install.00403C70

00479AF5 |. 8B55 E8 |MOV EDX,[LOCAL.6]

00479AF8 |. 8D45 FC |LEA EAX,[LOCAL.1]

00479AFB |. E8 50A2F8FF |CALL _Install.00403D50

00479B00 |. 43 |INC EBX

00479B01 |> 8B45 FC MOV EAX,[LOCAL.1]

00479B04 |. E8 3FA2F8FF |CALL _Install.00403D48

00479B09 |. 83F8 0A |CMP EAX,0A

00479B0C |.^ 7C DD |JL SHORT _Install.00479AEB

00479B0E |. 8D55 E4 LEA EDX,[LOCAL.7]

00479B11 |. 8B45 FC MOV EAX,[LOCAL.1]

00479B14 |. E8 D7E4F8FF CALL _Install.00407FF0 ; <==UpperCase

- Tiếp đó, chương trình tiến hành quá trình tính toán các ký tự của U và chuyển thành chuỗi DEC tương ứng

00479B32 |. BB 01000000 MOV EBX,1

00479B37 |> 8B45 FC /MOV EAX,[LOCAL.1] ; <== U

```

00479B3A |. 8A4418 FF  |MOV AL,BYTE PTR DS:[EAX+EBX-1] ; <== U[i]
00479B3E |. 3C 46    |CMP AL,46          ; <== if ( U[i] > 0x46 )
00479B40 |. 76 22    |JBE SHORT _Install.00479B64      ; <== then
00479B42 |. 8B45 FC  |MOV EAX,[LOCAL.1]        ; <== U
00479B45 |. 0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== U[i]
00479B4A |. 8D143B    |LEA EDX,DWORD PTR DS:[EBX+EDI]       ; <== Temp = dV + i
00479B4D |. 2BC2    |SUB EAX,EDX          ; <== Temp = U[i] - Temp
00479B4F |. 8D55 E0    |LEA EDX,[LOCAL.8]
00479B52 |. E8 55E8F8FF |CALL _Install.004083AC      ; <== Convert to DEC Str : tStr
00479B57 |. 8B55 E0    |MOV EDX,[LOCAL.8]
00479B5A |. 8D45 F0    |LEA EAX,[LOCAL.4]
00479B5D |. E8 EEA1F8FF |CALL _Install.00403D50      ; <== concat(dStr, tStr)
00479B62 |. EB 20    |JMP SHORT _Install.00479B84      ; <== else
00479B64 |> 8B45 FC  |MOV EAX,[LOCAL.1]        ; <== U
00479B67 |. 0FB64418 FF |MOVZX EAX,BYTE PTR DS:[EAX+EBX-1] ; <== U[i]
00479B6C |. 8D143B    |LEA EDX,DWORD PTR DS:[EBX+EDI]       ; <== Temp = dV + i
00479B6F |. 03C2    |ADD EAX,EDX          ; <== Temp = U[i] + Temp
00479B71 |. 8D55 DC    |LEA EDX,[LOCAL.9]
00479B74 |. E8 33E8F8FF |CALL _Install.004083AC      ; <== Convert to DEC Str : tStr
00479B79 |. 8B55 DC    |MOV EDX,[LOCAL.9]
00479B7C |. 8D45 F0    |LEA EAX,[LOCAL.4]
00479B7F |. E8 CCA1F8FF |CALL _Install.00403D50      ; <== concat(dStr, tStr)
00479B84 |> 47    |INC EDI          ; <== dV ++
00479B85 |. 43    |INC EBX          ; <== i++
00479B86 |. 4E    |DEC ESI          ; <== LenU --
00479B87 |.^ 75 AE   |JNZ SHORT _Install.00479B37      ; <== Loop Until LenU = 0

```

- Sau đó, chương trình sẽ xét thêm một lần nữa chiều dài của chuỗi dStr . Nếu chuỗi này có chiều dài nhỏ hơn 20 ký tự thì chương trình sẽ thêm vào sau đó lần lượt các ký mặc định để đủ 20 ký tự :

```

00479B89 |> \BB 31000000 MOV EBX,31
00479B8E |. EB 20    JMP SHORT _Install.00479BB0
00479B90 |> 8D45 D8  /LEA EAX,[LOCAL.10]
00479B93 |. 8BD3    |MOV EDX,EBX
00479B95 |. E8 D6A0F8FF |CALL _Install.00403C70
00479B9A |. 8B55 D8  |MOV EDX,[LOCAL.10]
00479B9D |. 8D45 F0  |LEA EAX,[LOCAL.4]
00479BA0 |. E8 ABA1F8FF |CALL _Install.00403D50
00479BA5 |. 43    |INC EBX
00479BA6 |. 83FB 39  |CMP EBX,39
00479BA9 |. 75 05    |JNZ SHORT _Install.00479BB0
00479BAB |. BB 31000000 |MOV EBX,31
00479BB0 |> 8B45 F0  MOV EAX,[LOCAL.4]
00479BB3 |. E8 90A1F8FF |CALL _Install.00403D48
00479BB8 |. 83F8 14  |CMP EAX,14
00479BBB |.^ 7C D3   |JL SHORT _Install.00479B90

```

- Kết thúc các quá trình này ta đến quá trình tạo đoạn Serial đầu tiên :

```

00479BBD |. BB 01000000 MOV EBX,1          ; <== i = 1
00479BC2 |> 8D45 D4  /LEA EAX,[LOCAL.11]
00479BC5 |. 8B55 F0  |MOV EDX,[LOCAL.4]        ; <== dStr
00479BC8 |. 8A541A FF  |MOV DL,BYTE PTR DS:[EDX+EBX-1] ; <== dStr[i-1]
00479BCC |. E8 9FA0F8FF |CALL _Install.00403C70      ; <== tStr : dStr[i-1]

```

```

00479BD1 |. 8B55 D4    |MOV EDX,[LOCAL.11]
00479BD4 |. 8D45 EC    |LEA EAX,[LOCAL.5]
00479BD7 |. E8 74A1F8FF |CALL _Install.00403D50          ; <== concat(S, tStr)
00479BDC |. 43          |INC EBX                      ; <== i++
00479BDD |. 83FB 06    |CMP EBX,6                     ; <== while ( i < 6 )
00479BE0 |.^ 75 E0     |JNZ SHORT _Install.00479BC2      ; <== continue Loop

```

- Tiếp theo, tuy theo loại tạo Serial là Personnal hay Profesional mà chương trình sẽ gắn thêm vào sau chuỗi Serial là các ký tự mặc định tương ứng :

```

00479BEA |. E8 61A1F8FF |CALL _Install.00403D50          ; <== "-"
00479BEF |. 837D F8 01 |CMP [LOCAL.2],1                  ; <== if Personnal
00479BF3 |. 75 0D      |JNZ SHORT _Install.00479C02
00479BF5 |. 8D45 EC    |LEA EAX,[LOCAL.5]
00479BF8 |. BA 089D4700 |MOV EDX,_Install.00479D08      ; ASCII "DP"
00479BFD |. E8 4EA1F8FF |CALL _Install.00403D50
00479C02 |> 837D F8 02 |CMP [LOCAL.2],2                  ; <== if Professional
00479C06 |. 75 0D      |JNZ SHORT _Install.00479C15
00479C08 |. 8D45 EC    |LEA EAX,[LOCAL.5]
00479C0B |. BA 149D4700 |MOV EDX,_Install.00479D14      ; ASCII "DR"
00479C10 |. E8 3BA1F8FF |CALL _Install.00403D50

```

- Đoạn kế là quá trình tạo thêm ba ký tự của đoạn Serial thứ hai :

```

00479C15 |> \BB 03000000 |MOV EBX,3                  ; <== i = 3
00479C1A |> 8B45 F0    |MOV EAX,[LOCAL.4]           ; <== U
00479C1D |. E8 26A1F8FF |CALL _Install.00403D48      ; <== LenU
00479C22 |. 8945 CC    |MOV [LOCAL.13],EAX
00479C25 |. DB45 CC    |FILD [LOCAL.13]            ; <== Len (FloatingPointNumber)
00479C28 |. D835 189D4700 |FDIV DWORD PTR DS:[479D18]   ; <== Len = Len % 2
00479C2E |. E8 598DF8FF |CALL _Install.0040298C      ; <== Len ( HEX Value )
00479C33 |. 83C0 03    |ADD EAX,3                   ; <== Temp = Len + 3
00479C36 |. 83D2 00    |ADC EDX,0
00479C39 |. 52          |PUSH EDX
00479C3A |. 50          |PUSH EAX
00479C3B |. 8BC3        |MOV EAX,EBX
00479C3D |. 99          |CDQ
00479C3E |. 290424    |SUB DWORD PTR SS:[ESP],EAX      ; <== Temp = Temp - i
00479C41 |. 195424 04  |SBB DWORD PTR SS:[ESP+4],EDX
00479C45 |. 58          |POP EAX
00479C46 |. 5A          |POP EDX                  ; <== Temp
00479C47 |. 8B55 F0    |MOV EDX,[LOCAL.4]           ; <== U
00479C4A |. 8A5402 FF    |MOV DL,BYTE PTR DS:[EDX+EAX-1]  ; <== U[Temp]
00479C4E |. 8D45 D0    |LEA EAX,[LOCAL.12]
00479C51 |. E8 1AA0F8FF |CALL _Install.00403C70      ; <== tStr : U[Temp]
00479C56 |. 8B55 D0    |MOV EDX,[LOCAL.12]
00479C59 |. 8D45 EC    |LEA EAX,[LOCAL.5]
00479C5C |. E8 EFA0F8FF |CALL _Install.00403D50      ; <== concat(S, tStr)
00479C61 |. 4B          |DEC EBX                  ; <== i--
00479C62 |. 85DB        |TEST EBX,EBX             ; <== While ( i > 0 )
00479C64 |.^ 75 B4     |JNZ SHORT _Install.00479C1A      ; <== Continue Loop

```

- Các ký tự của đoạn Serial cuối cùng chính là các ký tự của chuỗi dStr đi ngược từ cuối :

```

00479C6E |. E8 DDA0F8FF |CALL _Install.00403D50          ; <== "-"

```

```

00479C73 |. 8B45 F0    MOV EAX,[LOCAL.4]
00479C76 |. E8 CDA0F8FF CALL _Install.00403D48
00479C7B |. 8BD8    MOV EBX,EAX
00479C7D |. 8B45 F0    MOV EAX,[LOCAL.4]
00479C80 |. E8 C3A0F8FF CALL _Install.00403D48
00479C85 |. 8BF0    MOV ESI,EAX
00479C87 |. 83EE 09    SUB ESI,9
00479C8A |. 2BF3    SUB ESI,EBX
00479C8C |. 7F 1F    JG SHORT _Install.00479CAD
00479C8E |. 4E    DEC ESI
00479C8F |> 8D45 C8    /LEA EAX,[LOCAL.14]
00479C92 |. 8B55 F0    |MOV EDX,[LOCAL.4] ; <== dStr
00479C95 |. 8A541A FF    |MOV DL,BYTE PTR DS:[EDX+EBX-1] ; <== dStr[Len-1]
00479C99 |. E8 D29FF8FF  |CALL _Install.00403C70 ; <== tStr
00479C9E |. 8B55 C8    |MOV EDX,[LOCAL.14]
00479CA1 |. 8D45 EC    |LEA EAX,[LOCAL.5]
00479CA4 |. E8 A7A0F8FF  |CALL _Install.00403D50 ; <== concat(S, tStr)
00479CA9 |. 4B    |DEC EBX ; <== Len--
00479CAA |. 46    |INC ESI
00479CAB |.^ 75 E2    |JNZ SHORT _Install.00479C8F ; <== Loop until Len = 0

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm

Serial : 50103-DR113-7132152103

III – End of Tut :

- Finished – September 29, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.sofrayt.com
Production :	Sofrayt, Ltd.
SoftWare :	GetSmile 1.5 Full
Copyright by :	Copyright © 2001-2004 Sofrayt, Ltd. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

GetSmile 1.5 Full

Your e-mails look too ordinary? Your postings are just plain boring? Bring fun to your life! Delight yourself and people around you with a smile! You need just one program - GetSmile. This program lets you insert the most beautiful smilies in your messages. Just one mouse click stands between you and them.

I – Information :

- Dùng PEiD kiểm tra biết chương trình không bị PACK và biết chương trình được viết bằng ***Microsoft Visual C++ 6.0***

- Chạy thử chương trình với Fake Serial ta nhận được thông báo "***Invalid Key !!"*** . Tuy nhiên ta không tìm được thông báo này . Dùng phương pháp STACK ta xác định được FUNCTION chính .

- Dò ngược lên trên và đặt BreakPoint tại :

00421731 . E8 CAC30100 CALL GetSmile.0043DB00 ; <== Set BreakPoint here

II – Cracking :

- Load chương trình lên, chạy chương trình với User và Fake Serial, chương trình dừng lại tại điểm đặt BP . Trace xuống chút :

004217A1 . E8 BAC50100 CALL GetSmile.0043DD60 ; <== Encrypt

- Trace into để tìm hiểu quá trình mã hoá. Đầu tiên chương trình xác định chiều dài của chuỗi U nhập :

0043DDE1 |. E8 BAD80700 CALL GetSmile.004BB6A0 ; <== CharUpper

0043DDE6 |. 8B5424 08 MOV EDX,DWORD PTR SS:[ESP+8] ; <== LenU

0043DDEA |. 837A F8 03 CMP DWORD PTR DS:[EDX-8],3 ; <== LenU atleast 3 charts

0043DDEE |. 7D 2D JGE SHORT GetSmile.0043DE1D

- Kế đó chương trình tạo chuỗi mặc định “ ***this is the easy one*** ” . Tuy gọi là chuỗi mặc định nhưng ta có thể xem chương trình đã xác định 5 giá trị mặc định tương ứng với chuỗi này :

----- == Default Value == -----

```
unsigned long int reaTable_01[10]= {
    0x736968FF, 0x20736920, 0x20656874, 0x79736165, 0x656E6F20
};
```

----- == Default Value == -----

```
0043DE1D |> \B0 69      MOV AL,69
0043DE1F |. B1 73      MOV CL,73
0043DE21 |. 884424 0E    MOV BYTE PTR SS:[ESP+E],AL
0043DE25 |. 884424 11    MOV BYTE PTR SS:[ESP+11],AL
0043DE29 |. B0 65      MOV AL,65
0043DE2B |. 884C24 0F    MOV BYTE PTR SS:[ESP+F],CL
0043DE2F |. 884C24 12    MOV BYTE PTR SS:[ESP+12],CL
0043DE33 |. 884424 16    MOV BYTE PTR SS:[ESP+16],AL
0043DE37 |. 884424 18    MOV BYTE PTR SS:[ESP+18],AL
0043DE3B |. 884C24 1A    MOV BYTE PTR SS:[ESP+1A],CL
0043DE3F |. 884424 1F    MOV BYTE PTR SS:[ESP+1F],AL
0043DE43 |. B9 3B000000  MOV ECX,3B
0043DE48 |. 33C0      XOR EAX,EAX
0043DE4A |. 8D7C24 20    LEA EDI,DWORD PTR SS:[ESP+20]
0043DE4E |. C64424 0D 68  MOV BYTE PTR SS:[ESP+D],68
0043DE53 |. C64424 10 20  MOV BYTE PTR SS:[ESP+10],20
0043DE58 |. C64424 13 20  MOV BYTE PTR SS:[ESP+13],20
0043DE5D |. C64424 14 74  MOV BYTE PTR SS:[ESP+14],74
0043DE62 |. C64424 15 68  MOV BYTE PTR SS:[ESP+15],68
0043DE67 |. C64424 17 20  MOV BYTE PTR SS:[ESP+17],20
0043DE6C |. C64424 19 61  MOV BYTE PTR SS:[ESP+19],61
0043DE71 |. C64424 1B 79  MOV BYTE PTR SS:[ESP+1B],79
```

```

0043DE76 |. C64424 1C 20 MOV BYTE PTR SS:[ESP+1C],20
0043DE7B |. C64424 1D 6F MOV BYTE PTR SS:[ESP+1D],6F
0043DE80 |. C64424 1E 6E MOV BYTE PTR SS:[ESP+1E],6E
0043DE85 |. 33F6      XOR ESI,ESI
0043DE87 |. F3:AB      REP STOS DWORD PTR ES:[EDI]

```

- Quá trình mã hoá đầu tiên chương trình sẽ tạo giá trị thứ năm (5) của chuỗi mặc định bên trên :

```

0043DE90 |> /8B5424 08    /MOV EDX,DWORD PTR SS:[ESP+8]           ; <== while (i<LenU )
0043DE94 |> 3B72 F8      CMP ESI,DWORD PTR DS:[EDX-8]          ; <== Continue Loop
0043DE97 |. 7D 24        JGE SHORT GetSmile.0043DEBD          ; <== U[i]
0043DE99 |. 66:8B0472   |MOV AX,WORD PTR DS:[EDX+ESI*2]       ; <== U[i]
0043DE9D |. 8D4C24 0C    |LEA ECX,DWORD PTR SS:[ESP+C]
0043DEA1 |. 25 FF000000  |AND EAX,0FF                      ; <== U[i]
0043DEA6 |. 50          |PUSH EAX
0043DEA7 |. 6A 05        |PUSH 5
0043DEA9 |. 6A 3F        |PUSH 3F
0043DEAB |. 51          |PUSH ECX
0043DEAC |. E8 2FA5FDFF |CALL GetSmile.004183E0          ; <== Calculation

```

===== Calculation =====

```

00418406 |> /8B0E      /MOV ECX,DWORD PTR DS:[ESI]           ; <== dValue[j]
00418408 |. 33C0      |XOR EAX,EAX
0041840A |. 50          |PUSH EAX
0041840B |. 53          |PUSH EBX                      ; <== U[i]
0041840C |. 50          |PUSH EAX
0041840D |. 51          |PUSH ECX                      ; <== dValue[j]
0041840E |. E8 6D4D0800 |CALL GetSmile.0049D180          ; <== Value = dValue[j] * U[i]
00418413 |. 33C9      |XOR ECX,ECX
00418415 |. 33ED      |XOR EBP,EBP
00418417 |. 03C7      |ADD EAX,EDI                      ; <== Value = Value + TempRemain
00418419 |. 8906      |MOV DWORD PTR DS:[ESI],EAX          ; <== dValue[j] = Value
0041841B |. 8B4424 20  |MOV EAX,DWORD PTR SS:[ESP+20]
0041841F |. 13CD      |ADC ECX,EBP
00418421 |. 83C6 04  |ADD ESI,4
00418424 |. 48          |DEC EAX                      ; <== j--
00418425 |. 8D3C11  |LEA EDI,DWORD PTR DS:[ECX+EDX] ; <== TempRemain =
TempRemain + Over
00418428 |. 894424 20  |MOV DWORD PTR SS:[ESP+20],EAX
0041842C |.^75 D8    |JNZ SHORT GetSmile.00418406          ; <== Loop until j == 0x0
0041842E |. 85FF      |TEST EDI,EDI
00418430 |. 5D          |POP EBP
00418431 |. 74 1E      |JE SHORT GetSmile.00418451
00418433 |. 8B4C24 10  |MOV ECX,DWORD PTR SS:[ESP+10]
00418437 |. 8B4424 18  |MOV EAX,DWORD PTR SS:[ESP+18]
0041843B |. 0FAF1C81  IMUL EBX,DWORD PTR DS:[ECX+EAX*4] ; <== dValue[5] = dValue[5]
* U[i]
0041843F |. 03DF      |ADD EBX,EDI                      ; <== dValue[5] = dValue[5] + TempRemain

```

===== Calculation =====

```

0043DEB1 |. 83C4 10  |ADD ESP,10                      ; <== or
0043DEB4 |. 46          |INC ESI                      ; <== i++
0043DEB5 |. 81FE 00010000 |CMP ESI,100          ; <== while ( i < 256 )
0043DEBB |.^7C D3    |JL SHORT GetSmile.0043DE90          ; <== Continuer Loop

```

- Quá trình mã hoá thứ hai tạo ra một bảng giá trị 256 ký tự dựa vào bảng tính ở trên :

```

0043DEBD > \33C0      XOR EAX,EAX          ; <== i = 0
0043DEBF > 8BD0      /MOV EDX,EAX          ; <== j = i
0043DEC1 |. 81E2 07000080 |AND EDX,80000007   ; <== j = j & 0x80000007
0043DEC7 |. 79 05      |JNS SHORT GetSmile.0043DECE
0043DEC9 |. 4A        |DEC EDX
0043DECA |. 83CA F8    |OR EDX,FFFFFFF8
0043DECD |. 42        |INC EDX
0043DECE |> 8A4C14 0C  |MOV CL,BYTE PTR SS:[ESP+EDX+C] ; <== tStr[j]
0043DED2 |. 8A5404 0C  |MOV DL,BYTE PTR SS:[ESP+EAX+C] ; <== tStr[i]
0043DED6 |. 02D1      |ADD DL,CL          ; <== Value = tStr[j] + tStr[i]
0043DED8 |. 885404 0C  |MOV BYTE PTR SS:[ESP+EAX+C],DL ; <== tStr[i] = Value
0043DEDC |. 40        |INC EAX          ; <== i++
0043DEDD |. 3D 00010000 |CMP EAX,100         ; <== while ( i < 256 )
0043DEE2 |.^ 7C DB    \|JL SHORT GetSmile.0043DEBF ; <== Continue Loop

```

- Đoạn mã hoá cuối cùng là quá trình tạo chuỗi Serial thực :

```

0043DEF7 |. BF 01000000 MOV EDI,1          ; <== i = 1
0043DEFC |> 8A443C 0C  /MOV AL,BYTE PTR SS:[ESP+EDI+C] ; <== tStr[i]
0043DF00 |. 3C 80      |CMP AL,80          ; <== if ( tStr[i] < 0x80 )
0043DF02 |. 73 12      |JNB SHORT GetSmile.0043DF16 ; <== then
0043DF04 |. 25 FF000000 |AND EAX,0FF        ; <== tStr[i]
0043DF09 |. B9 0A000000 |MOV ECX,0A          ; <== dV
0043DF0E |. 99        |CDQ
0043DF0F |. F7F9      |IDIV ECX          ; <== Char = dStr[i] % 0xA
0043DF11 |. 83C2 30    |ADD EDX,30          ; <== Char = Char + 0x30
0043DF14 |. EB 10      |JMP SHORT GetSmile.0043DF26 ; <== else
0043DF16 |> 25 FF000000 |AND EAX,0FF        ; <== tStr[i]
0043DF1B |. B9 1A000000 |MOV ECX,1A          ; <== dV
0043DF20 |. 99        |CDQ
0043DF21 |. F7F9      |IDIV ECX          ; <== Char = dStr[i] % 0x1A
0043DF23 |. 83C2 41    |ADD EDX,41          ; <== Char = Char + 0x41
0043DF26 |> 66:0FBED2  |MOVSX DX,DL          ; <== Char
0043DF2A |. 52        |PUSH EDX
0043DF2B |. 8BCE      |MOV ECX,ESI
0043DF2D |. E8 49D60700 |CALL GetSmile.004BB57B
0043DF32 |. 8B0E      |MOV ECX,DWORD PTR DS:[ESI]
0043DF34 |. 8B41 F8    |MOV EAX,DWORD PTR DS:[ECX-8]
0043DF37 |. 66:837C41 FE >|CMP WORD PTR DS:[ECX+EAX*2-2],4F ; <== / from here
0043DF3D |. 74 14      |JE SHORT GetSmile.0043DF53 ; <== |
0043DF3F |. 66:837C41 FE >|CMP WORD PTR DS:[ECX+EAX*2-2],49 ; <== |
0043DF45 |. 75 04      |JNZ SHORT GetSmile.0043DF4B ; <== | if ( Chart == 0x4F
0043DF47 |. 6A 31      |PUSH 31           ; <== | or
0043DF49 |. EB 0A      |JMP SHORT GetSmile.0043DF55 ; <== | Char == 0x51 )
0043DF4B |> 66:837C41 FE >|CMP WORD PTR DS:[ECX+EAX*2-2],51 ; <== | Char = 0x30
0043DF51 |. 75 0B      |JNZ SHORT GetSmile.0043DF5E ; <== | else if (Chart == 0x49 )
0043DF53 |> 6A 30      |PUSH 30           ; <== | Char = 0x31
0043DF55 |> 48        |DEC EAX          ; <== |
0043DF56 |. 8BCE      |MOV ECX,ESI        ; <== |
0043DF58 |. 50        |PUSH EAX          ; <== |
0043DF59 |. E8 66D70700 |CALL GetSmile.004BB6C4 ; <== \ to here
0043DF5E |> B8 67666666 |MOV EAX,66666667 ; <== / from here

```

```

0043DF63 |. F7EF    |IMUL EDI          ; <== |
0043DF65 |. D1FA    |SAR EDX,1      ; <== |
0043DF67 |. 8BC2    |MOV EAX,EDX   ; <== |
0043DF69 |. C1E8 1F  |SHR EAX,1F     ; <== | if ( j == 5
0043DF6C |. 03D0    |ADD EDX,EAX   ; <== | or
0043DF6E |. 8D0C92  |LEA ECX,DWORD PTR DS:[EDX+EDX*4]; <== | j == 11
0043DF71 |. 3BCF    |CMP ECX,EDI    ; <== | or
0043DF73 |. 75 0E    |JNZ SHORT GetSmile.0043DF83 ; <== | j == 17 )
0043DF75 |. 83FF 14  |CMP EDI,14     ; <== | S[j] = 0x2D
0043DF78 |. 7D 09    |JGE SHORT GetSmile.0043DF83 ; <== | else
0043DF7A |. 6A 2D    |PUSH 2D        ; <== | S[j] = Chart
0043DF7C |. 8BCE    |MOV ECX,ESI    ; <== |
0043DF7E |. E8 F8D50700 |CALL GetSmile.004BB57B ; <== \ to here
0043DF83 |> 47      |INC EDI        ; <== i++
0043DF84 |. 83FF 14  |CMP EDI,14     ; <== while ( i < 20 )
0043DF87 |.^ 0F8E 6FFFFFFF |JLE GetSmile.0043DEFC ; <== Continue Loop

```

- Kết thúc Function Encrypt ta trở lại Function trước đó. Quá trình so sánh diễn ra ngay sau :

```

004217A1 . E8 BAC50100 CALL GetSmile.0043DD60 ; <== Encrypt
004217A6 . 83C4 08 ADD ESP,8
004217A9 . 85C0 TEST EAX,EAX
004217AB . 74 1A JE SHORT GetSmile.004217C7
004217AD . 8B5424 10 MOV EDX,DWORD PTR SS:[ESP+10] ; <== Fake Serial
004217B1 . 8B4424 1C MOV EAX,DWORD PTR SS:[ESP+1C] ; <== RealSerial
004217B5 . 52 PUSH EDX
004217B6 . 50 PUSH EAX
004217B7 . E8 79B60700 CALL GetSmile.0049CE35 ; <== Compare

```

/*/*/* - SERIAL tương ứng :

User : REA-cRaCkErTeAm Serial : 6400K-86328-2026R-4E0XX

III – End of Tut :

- Finished – September 27, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.aokx.com/
Production :	Aokx Media
SoftWare :	Ace DVD BackUp 1.2.11
Copyright by :	Copyright (C) Aokx Media 2004. All Rights Reserved.
Type :	Email/Serial
Packed :	Not
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

Ace DVD BackUp 1.2.11

Ace DVD Backup is a powerful and easy to use DVD backup tool. It enables you to backup your favorite DVDs by convert them to the most popular video formats -- VCD, SVCD, Mpeg, AVI, DivX. It provides you new experience in dvd ripping with its fast speed, wonderful output quality and clear user interface. You can backup your favorite DVDs by just several clicks.....

I – Information :

- Dùng PeiD 0.92 chúng ta biết được soft này được Code bằng **Microsoft Visual C++ 6.0** và quan trọng nữa là nó không bị Pack (ke ke quá khỏe).
- Chạy thử chương trình không nhận thấy thông báo gì cả , cứ tưởng nó free cho mình nhưng ặc ặc .. Nếu click vào menu Help , các bạn sẽ nhìn thấy có mục sau : **Enter Registration Code** . Vậy là nó không free rồi. Click vào đây chúng ta sẽ nhận được một dialog box (“Registration”) bắt chúng ta nhập **Email** and **RightCode** vào để đăng kí . Vì nếu không đăng kí chúng ta chỉ được sử dụng có 50% tính năng của chương trình này thôi. đương nhiên là chúng ta không muốn như thế rồi, nhưng trong tay chúng ta thì chẳng có cái RightCode để mà đăng kí , chỉ có mỗi OllyDbg thôi.... Vậy thì chỉ còn mỗi một công việc đó là bắt tay vào Crack nó.
- Oki , tại cái dialog box này , chúng ta nhập đại một **FM (Fake Mail)** và **FS (Fake Serial)** . Ở đây mình nhập là (**Email** : kienbigmummy@yahoo.com , **Code** : A0123-B2345-E4567-R7890-T1982 (xem trong phần Help của chương trình nó bắt phải nhập Code có dạng như thế)). Sau đó các bạn nhấn OK , ặc ặc một cái Nag bắn ra “**Invalid Email or Code**”. Oki, các bạn hãy ghi nhớ lấy.
- Bây giờ chúng ta Load chương trình này trong Olly , và tìm thấy chuỗi này tại địa chỉ sau : 0040E14F . 68 00014900 PUSH Ace_DVD_.00490100 ; ASCII "Invalid Email or Code."
- Lần ngược lên trên chúng ta sẽ đặt Break Point tại vị trí sau :
0040DF32 . E8 896A0500 CALL <JMP.&MFC42.#3874> <== Set BP

II – Cracking :

-Oki , bây giờ chúng ta bấm **F9** để Run chương trình , vào **Help --> Enter Registration Code** , chúng ta lại nhập lại FM và FS như đã nói ở trên . Sau đó nhấn OK , các bạn sẽ quay trở lại màn hình chính của Olly và **Ice** tại điểm mà chúng đã đặt Break Point trong Ollydbg :

```
0040DF32 . E8 896A0500 CALL <JMP.&MFC42.#3874>;<= We're here(Get FM & EAX:  
length(FM))  
0040DF37 . 8B4C24 10 MOV ECX,DWORD PTR SS:[ESP+10]  
0040DF3B . 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8]  
0040DF3E . 83F8 40 CMP EAX,40 ;<== Length(FM) > 64 ?  
0040DF41 . 0F8F 3F020000 JG Ace_DVD_.0040E186 ;<== Right , bắn Nag  
0040DF47 . 83F8 03 CMP EAX,3 ;<== Length(FM) < 3 ?  
0040DF4A . 0F8C 36020000 JL Ace_DVD_.0040E186 ;<== Right, bắn Nag  
0040DF50 . 6A 40 PUSH 40  
0040DF52 . 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]  
0040DF56 . E8 6B6D0500 CALL <JMP.&MFC42.#2763>  
0040DF5B . 83CB FF OR EBX,FFFFFF  
0040DF5E . 3BC3 CMP EAX,EBX  
0040DF60 . 0F84 23020000 JE Ace_DVD_.0040E189  
0040DF66 . 8A15 CC064B00 MOV DL,BYTE PTR DS:[4B06CC]
```

```

0040DF6C . B9 10000000 MOV ECX,10
0040DF71 . 33C0 XOR EAX,EAX
0040DF73 . 8D7C24 3D LEA EDI,DWORD PTR SS:[ESP+3D]
0040DF77 . 885424 3C MOV BYTE PTR SS:[ESP+3C],DL
0040DF7B . 6A 40 PUSH 40
0040DF7D . F3:AB REP STOS DWORD PTR ES:[EDI]
0040DF7F . 8D4424 40 LEA EAX,DWORD PTR SS:[ESP+40]
0040DF83 . 8BCD MOV ECX,EBP
0040DF85 . 50 PUSH EAX
0040DF86 . E8 356D0500 CALL <JMP.&MFC42.#3873>
0040DF8B . 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
0040DF8F . E8 8A690500 CALL <JMP.&MFC42.#540>
0040DF94 . 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
0040DF98 . 8DBE 60010000 LEA EDI,DWORD PTR DS:[ESI+160]
0040DF9E . 51 PUSH ECX
0040DF9F . 8BCF MOV ECX,EDI
0040DFA1 . C68424 8C0500>MOV BYTE PTR SS:[ESP+58C],1
0040DFA9 . E8 126A0500 CALL <JMP.&MFC42.#3874> ;<== Get FS & EAX : Length(FS)
0040DFAE . 8B4C24 14 MOV ECX,DWORD PTR SS:[ESP+14]
0040DFB2 . 8B41 F8 MOV EAX,DWORD PTR DS:[ECX-8]
0040DFB5 . 85C0 TEST EAX,EAX ;<== FS = "" ?
0040DFB7 . 0F84 8B010000 JE Ace_DVD_.0040E148 ;<== Băn Nag
0040DFBD . 83F8 1D CMP EAX,1D ;<== Length (FS) = 1D ?
0040DFC0 . 0F85 82010000 JNZ Ace_DVD_.0040E148 ;<== Wrong , băn Nag
0040DFC6 . 50 PUSH EAX ;/maxlen ;<== Cát Length(FS)
0040DFC7 . 8D5424 20 LEA EDX,DWORD PTR SS:[ESP+20] ; |<== EDX : FS
0040DFCB . 51 PUSH ECX ;|src
0040DFCC . 52 PUSH EDX ;|dest
0040DFCD . FF15 B0764600 CALL DWORD PTR DS:<&MSVCRT.strncpy> ;\strncpy<= EAX: FS
0040DFD3 . 8D4424 28 LEA EAX,DWORD PTR SS:[ESP+28] ;<== EAX : FS
0040DFD7 . 8D4C24 48 LEA ECX,DWORD PTR SS:[ESP+48] ;<== ECX : FM
0040DFDB . 50 PUSH EAX
0040DFDC . 51 PUSH ECX
0040DFDD . C64424 4D 00 MOV BYTE PTR SS:[ESP+4D],0
0040DFE2 . E8 69FFFF CALL Ace_DVD_.0040DE50 ;<==== Encrypt (Trace Into)
----- Trace Into -----

```

```

0040DE8F |. 33F6 XOR ESI,ESI ; <== i = 0
0040DE91 |. 8BF8 MOV EDI,EAX ; <== Value_01 = Value
0040DE93 |. 8D58 0B LEA EBX,DWORD PTR DS:[EAX+B] ;<== Value_02 = Value + 0xB
0040DE96 |> 8BC6 /MOV EAX,ESI ; <== From here
0040DE98 |. B9 06000000 |MOV ECX,6 ; <===== This section mean that
0040DE9D |. 99 |CDQ ; <== Each section of RealSerial
0040DE9E |. F7F9 |IDIV ECX; <== had only 5 charts and
0040DEA0 |. 83FA 05 |CMP EDX,5; <== separate by "-"
0040DEA3 |. 74 25 |JE SHORT Ace_DVD_.0040DECA;
0040DEA5 |. 8BC7 |MOV EAX,EDI ; <== Value = Value_01
0040DEA7 |. B9 05000000 |MOV ECX,5 ; <== dV
0040DEAC |. 99 |CDQ; <== SecX = Value % dV
0040DEAD |. F7F9 |IDIV ECX ; <== SecX = Value % dV
0040DEAF |. 8B1495 EC0049>|MOV EDX,DWORD PTR DS:[EDX*4+4900EC] ;<== Add of SecX
0040DEB6 |. 8915 E4064B00 |MOV DWORD PTR DS:[4B06E4],EDX ;<== Add of SecX

```

```

0040DEBC |. 8A042E    |MOV AL,BYTE PTR DS:[ESI+EBP];<== FS [i]
0040DEBF |. 50        |PUSH EAX           ;<== Push Value
0040DEC0 |. 53        |PUSH EBX           ;<== Push Value_02
0040DEC1 |. FFD2    |CALL EDX ;<== SecX (Trace Into)
-----=Trace Into =====

```

-----=NOTE -----

Thực tế ở đây ta chú ý đến phần dư của phép chia này . Do giá trị “Value_01” được chia cho “5” nên phần dư không bao giờ quá “4” . Nên giá trị của “EDX” chỉ có “5” , căn cứ và giá trị này mà lệnh “CALL EDX” sẽ chuyển đến FUNTION kiểm tra tương ứng; hay nói cách khác, tùy thuộc vào giá trị của phần dư mà chương trình sẽ chuyển đến FUNCTION kiểm tra tương ứng (có “5” FUNCTION tất cả trong việc kiểm tra tương ứng với từng SecX)

-----=NOTE -----

-----=Funtion 0 -----

```

0040DE20 . 8B4C24 04  MOV ECX,DWORD PTR SS:[ESP+4]; <== Value_02
0040DE24 . B8 F31ACA6B  MOV EAX,6BCA1AF3;           <== dV
0040DE29 . F7E9      IMUL ECX ;<== Over = Value_02 * dV
0040DE2B . 8BC2      MOV EAX,EDX ;<== Over
0040DE2D . C1F8 03   SAR EAX,3 ;<== Over = Over / 0x8
0040DE30 . 8BC8      MOV ECX,EAX ;<== temp = Over
0040DE32 . C1E9 1F   SHR ECX,1F; <== temp = temp / 0x80000000
0040DE35 . 03C1      ADD EAX,ECX; <== Over
0040DE37 . B9 1A000000  MOV ECX,1A ;<== dV
0040DE3C . 99        CDQ
0040DE3D . F7F9      IDIV ECX; <== Over = Over % dV
0040DE3F . 8A4C24 08  MOV CL,BYTE PTR SS:[ESP+8]; <== FS[i]
0040DE43 . 33C0      XOR EAX,EAX ;<== Clear EAX
0040DE45 . 80C2 41   ADD DL,41 ;<== Over = Over + 0x41
0040DE48 . 3AD1      CMP DL,CL ;<== If ( Over = Fs[i] ) then
0040DE4A . 0F94C0   SETE AL ; Continue Check
0040DE4D . C3        RETN ; Return Loop

```

-----=Funtion 0 -----

-----=Funtion 1 -----

```

0040DDF0 . 8B4C24 04  MOV ECX,DWORD PTR SS:[ESP+4] ; <==== Value_02
0040DDF4 . B8 93244992  MOV EAX,92492493 ;<== dV
0040DDF9 . F7E9      IMUL ECX ;<== Over = Value_02 * dV
0040DDFB . 8BC2      MOV EAX,EDX ;<== Over
0040DDFD . 03C1      ADD EAX,ECX ;<== temp = Value_02 + Over
0040DDFF . C1F8 02   SAR EAX,2 ;<== temp = temp / 0x4
0040DE02 . 8BC8      MOV ECX,EAX ;<== temp1 = temp
0040DE04 . C1E9 1F   SHR ECX,1F; <== temp1 = temp1 / 0x80000000
0040DE07 . 03C1      ADD EAX,ECX; <== temp = temp + temp01
0040DE09 . B9 1A000000  MOV ECX,1A <== dV
0040DE0E . 99        CDQ
0040DE0F . F7F9      IDIV ECX <== Over = Temp % dV
0040DE11 . 8A4C24 08  MOV CL,BYTE PTR SS:[ESP+8] <== Fs[i]
0040DE15 . 33C0      XOR EAX,EAX <== Clear EAX
0040DE17 . 80C2 41   ADD DL,41 <== Over = Over + 0x41
0040DE1A . 3AD1      CMP DL,CL <== If ( Over = Fs[i] ) then
0040DE1C . 0F94C0   SETE AL <== Continue Check
0040DE1F . C3        RETN <== Return to Loop

```

-----=Funtion 1 -----

-----=Funtion 2 -----

```

0040DDC0 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] <== Value_02
0040DDC4 . B8 4FECC44E MOV EAX,4EC4EC4F <== dV
0040DDC9 . F7E9 IMUL ECX <== Over = Value_02 * dV
0040DDCB . 8BC2 MOV EAX,EDX <== Over
0040DDCD . C1F8 02 SAR EAX,2 <== Over = Over / 0x4
0040DDD0 . 8BC8 MOV ECX,EAX
0040DDD2 . C1E9 1F SHR ECX,1F
0040DDD5 . 03C1 ADD EAX,ECX <== Over
0040DDD7 . B9 1A000000 MOV ECX,1A <== dV
0040DDDC . 99 CDQ
0040DDDD . F7F9 IDIV ECX <== Over = Over % dV
0040DDDF . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] <== Fs[i]
0040DDE3 . 33C0 XOR EAX,EAX <== Clear EAX
0040DDE5 . 80C2 41 ADD DL,41 <== Over = Over + 0x41
0040DDE8 . 3AD1 CMP DL,CL <== If (Over = Fs[i] ) then
0040DDEA . 0F94C0 SETE AL <== Continue Check
0040DDED . C3 RETN <== Return to Loop

```

Funtion 2**Funtion 3**

```

0040DD90 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] <== Value_02
0040DD94 . B8 C94216B2 MOV EAX,B21642C9 <== dV
0040DD99 . F7E9 IMUL ECX <== Over = Value_02 * dV
0040DD9B . 8BC2 MOV EAX,EDX <== Over
0040DD9D . 03C1 ADD EAX,ECX <== Temp = Value_02 + Over
0040DD9F . C1F8 04 SAR EAX,4 <== Temp = Temp / 0x10
0040DDA2 . 8BC8 MOV ECX,EAX
0040DDA4 . C1E9 1F SHR ECX,1F
0040DDA7 . 03C1 ADD EAX,ECX <== Temp
0040DDA9 . B9 0A000000 MOV ECX,0A <== dV
0040DDAE . 99 CDQ
0040DDAF . F7F9 IDIV ECX <== Over = Temp % dV
0040DDB1 . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] <== Fs[i]
0040DDB5 . 33C0 XOR EAX,EAX
0040DDB7 . 80C2 30 ADD DL,30 <== Over = Over + 0x30
0040DDBA . 3AD1 CMP DL,CL <== If (Over = Fs[i] ) then
0040DDBC . 0F94C0 SETE AL <== Continue Check
0040DDBF . C3 RETN <== Return to Loop

```

Funtion 3**Funtion 4**

```

0040DD60 . 8B4C24 04 MOV ECX,DWORD PTR SS:[ESP+4] <== Value_02
0040DD64 . B8 79787878 MOV EAX,78787879 ; <== dV
0040DD69 . F7E9 IMUL ECX <== Over = Value_02 * dV
0040DD6B . 8BC2 MOV EAX,EDX <== Over
0040DD6D . C1F8 03 SAR EAX,3 <== Over = Over / 0x8
0040DD70 . 8BC8 MOV ECX,EAX <== Over
0040DD72 . C1E9 1F SHR ECX,1F
0040DD75 . 03C1 ADD EAX,ECX <== Over
0040DD77 . B9 0A000000 MOV ECX,0A <== dV
0040DD7C . 99 CDQ
0040DD7D . F7F9 IDIV ECX <== Over = Over % dV
0040DD7F . 8A4C24 08 MOV CL,BYTE PTR SS:[ESP+8] <== Fs[i]
0040DD83 . 33C0 XOR EAX,EAX

```

```

0040DD85 . 80C2 30    ADD DL,30 <== Over = Over + 0x30
0040DD88 . 3AD1    CMP DL,CL <== If (Over = Fs[i]) then
0040DD8A . 0F94C0    SETE AL <== Continue Check
0040DD8D . C3      RETN <== Return to Loop

```

Funtion 4**Trace Into**

```

0040DEC3 |. 83C4 08    |ADD ESP,8
0040DEC6 |. 85C0    |TEST EAX,EAX
0040DEC8 |. 74 16    |JE SHORT Ace_DVD_.0040DEE0
0040DECA > 46     |INC ESI <== i++
0040DECB |. 83EF 0B    |SUB EDI,0B <== Value_01 = Value01 - 0x0B
0040DECE |. 83EB 11    |SUB EBX,11 <== Value_02 = Value_02 - 0x11
0040DED1 |. 83FE 1D    |CMP ESI,1D <== While (i < 29)
0040DED4 |.^ 7C C0    |JL SHORT Ace_DVD_.0040DE96 <== Continue Loop

```

Trace Into

```

0040DFE7 . 83C4 14    ADD ESP,14
0040DFFA . 85C0    TEST EAX,EAX <== EAX = 0 ?
0040DFEC . 0F84 56010000 JE Ace_DVD_.0040E148 <== Right ,Jump to Nag

```

/*/*/*/*Serial :

Mail : kienbigmummy@yahoo.com **Code :** ZTD24–HW89R–Q44MJ–19IXD–4DLW3

or

Mail : REA-cRaCkErTeAm@yahoo.com **Code :** KTT42–HM17B–G72XJ–37TXT–2OLM6

Thông tin đăng ký về Mail và Code sẽ được lưu vào File : AceDVDBackup.ini trong thư mục Windows

III. Keygen (VC++) :

```

char reaEmail[64]={0};
char reaSerial[64]={0};
int LenEmail=0;
int i=0,j=0;
int WhiteSpace=0, Asign=0;
int Value=0, Value_01=0, Value_02=0;

```

```

LenEmail=GetDlgItemText(IDC_NAME,reaEmail,64);
if (LenEmail < 6)
{
    MessageBox("----- Your email atleast 6 charts -----","Hey !! Please input
your email again !!");
}
else
{
    i=0;
    while ( i < LenEmail )
    {
        if ( reaEmail[i] == 0x20 )
        {
           WhiteSpace++;
        }
        if ( reaEmail[i] == 0x40 )
        {
            Asign++;
        }
    }
}

```

```

        }
        i++;
    }
    if (WhiteSpace != 0)
    {
        MessageBox("-----===== Your email had WhiteSpace =====--- ","Hey !! Please
Please input your email again !! ");
    }
    else if (Asign != 1)
    {
        MessageBox("-----===== Your email not Valid =====--- ","Hey !! Please
input your email again !! ");
    }

else
{
    i=0;    Value=0;
    while (i < LenEmail)
    {
        Value = Value + (((reaEmail[i] * 0x10) + reaEmail[i]) % 0x6F);
        i++;
    }
    Value = Value * Value;
    Value_01 = Value;
    Value_02 = Value + 0xB;
    i=0;
    while (i < 0x1D)
    {
        if (i == 5 || i == 11 || i == 17 || i == 23)
        {
            reaSerial[i] = 0x2D;
            i++;
            Value_01 = Value_01 - 0xB;
            Value_02 = Value_02 - 0x11;
        }
        if ((Value_01 % 0x5) == 0x0)
        {
            _asm
            {
                MOV ECX,Value_02
                MOV EAX,0x78787879
                IMUL ECX
                MOV EAX,EDX
                SAR EAX,0x3
                MOV ECX,EAX
                SHR ECX,0x1F
                ADD EAX,ECX
                MOV ECX,0x0A
                CDQ
                IDIV ECX
                AND EDX, 0xFF
                ADD EDX,0x30
                MOV Value, EDX
            }
        }
    }
}

```

```

        }
        reaSerial[i] = Value;
    }
    else if ( ( Value_01 % 0x5 ) == 0x1 )
    {
        _asm
        {
            MOV ECX,Value_02
            MOV EAX,0xB21642C9
            IMUL ECX
            MOV EAX,EDX
            ADD EAX,ECX
            SAR EAX,0x4
            MOV ECX,EAX
            SHR ECX,0x1F
            ADD EAX,ECX
            MOV ECX,0x0A
            CDQ
            IDIV ECX
            AND EDX, 0xFF
            ADD EDX,0x30
            MOV Value, EDX
        }
        reaSerial[i] = Value;
    }
    else if ( ( Value_01 % 0x5 ) == 0x2 )
    {
        _asm
        {
            MOV ECX,Value_02
            MOV EAX,0x4EC4EC4F
            IMUL ECX
            MOV EAX,EDX
            SAR EAX,0x2
            MOV ECX,EAX
            SHR ECX,0x1F
            ADD EAX,ECX
            MOV ECX,0x1A
            CDQ
            IDIV ECX
            AND EDX, 0xFF
            ADD EDX,0x41
            MOV Value, EDX
        }
        reaSerial[i] = Value;
    }
    else if ( ( Value_01 % 0x5 ) == 0x3 )
    {
        _asm
        {
            MOV ECX,Value_02
            MOV EAX,0x92492493
            IMUL ECX

```

```

        MOV EAX,EDX
        ADD EAX,ECX
        SAR EAX,0x2
        MOV ECX,EAX
        SHR ECX,0x1F
        ADD EAX,ECX
        MOV ECX,0x1A
        CDQ
        IDIV ECX
        AND EDX, 0xFF
        ADD EDX,0x41
        MOV Value, EDX
    }
    reaSerial[i] = Value;
}
else if ( ( Value_01 % 0x5 ) == 0x4 )
{
    _asm
    {
        MOV ECX,Value_02
        MOV EAX,0x6BCA1AF3
        IMUL ECX
        MOV EAX,EDX
        SAR EAX,0x3
        MOV ECX,EAX
        SHR ECX,0x1F
        ADD EAX,ECX
        MOV ECX,0x1A
        CDQ
        IDIV ECX
        AND EDX, 0xFF
        ADD EDX,0x41
        MOV Value, EDX
    }
    reaSerial[i] = Value;
}
i++;
Value_01 = Value_01 - 0xB;
Value_02 = Value_02 - 0x11;
}

SetDlgItemText(IDC_SERIAL,reaSerial);
}
}

```

IV – End of Tut :

- Finished – *October 8, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.srctec.com
Production	:	SourceTec Sofware Co.Ltd
SoftWare	:	Applet Cool Text Wizard Version 1.0
Copyright by	:	Copyright (C) 1998 SourceTec Software Co.Ltd . All Rights Reserved.
Type	:	Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 5.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N/A
Request	:	Correct Serial

Applet Cool Text Wizard Version 1.0

CoolText is a Java Applet program which implement scrolling graphic and message banners and each banners can contain a URL so that the banner become a hyperlink to that URL. It could be used in web homepages.

CoolText is a configurable Java Applet, users could configure varied parameters of CoolText. CoolText can contain many banners and each banner can be customized to contain a variety of effects...background style,time to show this banner,message,font size and color, text special effects, text animate speed, graphics, even URL links,etc.

CoolText Applet is a pure Java based program(JDK 1.0), it developed by SourceTec Software Co. Ltd.

I – Information :

- Dùng PEiD 0.92 detect , chúng ta biết được chương trình này không bị PACK , và nó được Code bằng **Microsoft Visual C++ 5.0** . Có thể các bạn sẽ nghĩ rằng chương trình này quá cũ nhưng theo tôi điều quan trọng không phải ở chỗ đó. Mà quan trọng là chúng ta phải Crack nó thế nào....OK

- Khi chạy chương trình chúng ta sẽ nhận được cái Nag screen thông báo cho chúng ta biết rằng nếu chúng ta không bỏ tiền ra để đăng ký hợp pháp thì sẽ chỉ được sử dụng nó trong vòng 30 ngày là hết hạn kí hợp đồng hehe. Vậy thì chúng ta phải crack thôi.

- Ngay tại cái Nag screen này , chúng ta thấy có một button là **Register** , à chắc là chỗ này bắt chúng nhập thông tin đăng ký đây. Bấm vào đây, chúng ta sẽ chuyển sang một screen mới. Cụ thể là bắt chúng ta nhập **E-mail** và **Register Code** . Nếu bác nào đăng ký hợp pháp rồi thì cứ việc nhập vô, Còn tui trong tay chẳng có gì ngoài OllyDbg hehe. Thôi thì , cứ nhập đai một FU và FS thử xem (**E-mail** : **kienbigmummy@yahoo.com** ; **Register Code** : **11111982**) , rồi nhấn Register. Bạn sẽ không phải chờ thêm một tíc tắc nào nữa, cái Nag **Sorry, That is not a valid register code** đập ngay vào mặt. Ok , chúng ta Load chương trình này trong Olly , và tìm được chuỗi này tại địa chỉ sau :

0040B0CB |. 68 582D4400 PUSH AppletCo.00442D58 ; ASCII "Sorry, that is not a valid register code."

- OK , lần ngược lên trên và chúng ta set BreakPoint tại hàm CALL có địa chỉ như sau :
0040AFEF |. E8 EA9D0100 CALL AppletCo.00424DDE

II – Cracking :

- Xong , mọi thứ đã sẵn sàng . Bây giờ chúng ta nhấn F9 để Run chương trình . Sau khi nhấn xong thì màn hình thông báo của chương trình hiện lên . Chúng ta nhập vào FU và FS (ở đây mình nhập là **E-mail : kienbigmummy@yahoo.com ; Register Code : 11111982**). Sau đó nhấn Register, chúng ta sẽ quay trở lại màn hình chính của Olly , tại đây Olly sẽ Ice tại chỗ mà chúng ta đã xé BreakPoint.:

```

0040ADEF |. E8 EA9D0100 CALL AppletCo.00424DDE      ;<== We're here
0040AFF4 |. 8B7C24 20 MOV EDI,DWORD PTR SS:[ESP+20]
0040AFF8 |. 83C6 14 ADD ESI,14
0040AFFB |. 56 PUSH ESI
0040AFFC |. 8D4424 24 LEA EAX,DWORD PTR SS:[ESP+24]
0040B000 |. 57 PUSH EDI
0040B001 |. 50 PUSH EAX
0040B002 |. C74424 24 000>MOV DWORD PTR SS:[ESP+24],0
0040B00A |. E8 E1D3FFFF CALL AppletCo.004083F0<== Hàm mã hóa tạo Register Code
0040B00F |. 83C4 0C ADD ESP,0C
0040B012 |. 50 PUSH EAX
0040B013 |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]
0040B017 |. C64424 1C 01 MOV BYTE PTR SS:[ESP+1C],1
0040B01C |. E8 01A00100 CALL AppletCo.00425022
0040B021 |. 8D4C24 20 LEA ECX,DWORD PTR SS:[ESP+20]
0040B025 |. C64424 18 00 MOV BYTE PTR SS:[ESP+18],0
0040B02A |. E8 FA9E0100 CALL AppletCo.00424F29
0040B02F |. 8B6C24 24 MOV EBP,DWORD PTR SS:[ESP+24]
0040B033 |. 8B4C24 0C MOV ECX,DWORD PTR SS:[ESP+C]<== ECX :Right Register Code
0040B037 |. 51 PUSH ECX      ;<== Cát ECX
0040B038 |. 8B55 00 MOV EDX,DWORD PTR SS:[EBP]    ;<== EDX : FS
0040B03B |. 52 PUSH EDX      ;<== Cát EDX
0040B03C |. E8 5F630000 CALL AppletCo.004113A0<== Gọi hàm so sánh FS với Right Code
0040B041 |. 83C4 08 ADD ESP,8
0040B044 |. 85C0 TEST EAX,EAX      ;<== EAX = 0?
0040B046 |. 75 7F JNZ SHORT AppletCo.0040B0C7      ;<== Không bằng thì bắn Nag
0040B048 |. E8 2E590200 CALL AppletCo.0043097B
0040B04D |. 8B70 04 MOV ESI,DWORD PTR DS:[EAX+4]
0040B050 |. 68 342D4400 PUSH AppletCo.00442D34      ; /Arg3 = 00442D34 ASCII "Yes"
0040B055 |. 68 442D4400 PUSH AppletCo.00442D44      ; |Arg2 = 00442D44 ASCII "Registered"
0040B05A |. 68 382D4400 PUSH AppletCo.00442D38      ; |Arg1 = 00442D38 ASCII "Register"
0040B05F |. 8BCE MOV ECX,ESI      ; |
0040B061 |. E8 66EF0100 CALL AppletCo.00429FCC      ; \AppletCo.00429FCC
0040B066 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
0040B068 |. 8BCE MOV ECX,ESI
0040B06A |. 50 PUSH EAX      ; /Arg3
0040B06B |. 68 242D4400 PUSH AppletCo.00442D24      ; |Arg2 = 00442D24 ASCII "Email Address"
0040B070 |. 68 382D4400 PUSH AppletCo.00442D38      ; |Arg1 = 00442D38 ASCII "Register"
0040B075 |. E8 52EF0100 CALL AppletCo.00429FCC      ; \AppletCo.00429FCC
0040B07A |. 8B4D 00 MOV ECX,DWORD PTR SS:[EBP]
0040B07D |. 51 PUSH ECX      ; /Arg3
0040B07E |. 68 142D4400 PUSH AppletCo.00442D14      ; |Arg2 = 00442D14 ASCII "Register Code"
0040B083 |. 68 382D4400 PUSH AppletCo.00442D38      ; |Arg1 = 00442D38 ASCII "Register"
0040B088 |. 8BCE MOV ECX,ESI      ; |
0040B08A |. E8 3DEF0100 CALL AppletCo.00429FCC      ; \AppletCo.00429FCC
0040B08F |. 6A 00 PUSH 0

```

```

0040B091 |. 6A 00      PUSH 0
0040B093 |. 68 842D4400 PUSH AppletCo.00442D84      ; ASCII " Thank you ! Please restart"
0040B098 |. E8 48ED0100 CALL AppletCo.00429DE5
0040B09D |. 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
0040B0A1 |. C74424 18 FFF>MOV DWORD PTR SS:[ESP+18],-1
0040B0A9 |. E8 7B9E0100 CALL AppletCo.00424F29
0040B0AE |. B8 01000000 MOV EAX,1
0040B0B3 |. 8B4C24 10  MOV ECX,DWORD PTR SS:[ESP+10]
0040B0B7 |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX
0040B0BE |. 5F      POP EDI
0040B0BF |. 5E      POP ESI
0040B0C0 |. 5D      POP EBP
0040B0C1 |. 83C4 10  ADD ESP,10
0040B0C4 |. C2 0800  RETN 8
0040B0C7 |> 6A 00      PUSH 0
0040B0C9 |. 6A 00      PUSH 0
0040B0CB |. 68 582D4400 PUSH AppletCo.00442D58 ; ASCII "Sorry, that is not a valid register
code."
0040B0D0 |. E8 10ED0100 CALL AppletCo.00429DE5

```

- Các bạn hãy để ý đến cái dòng màu đỏ ở phía trên , đó chính là hàm mã hóa tạo ra chuỗi Right Register Code đấy. Chúng ta sẽ Trace Into vào trong hàm đó để xem nó làm gì (Nếu bác nào không thích thì có thể bỏ qua bước này vì thực ra sau khi Trace qua hàm này chúng ta đã có được Register Code rồi):

0040B00A |. E8 E1D3FFFF CALL AppletCo.004083F0<== Hàm mã hóa tạo Register Code(Trace Into)

----- Trace Into -----

```

004083F0 /$ 6A FF      PUSH -1
004083F2 |. 68 C7384300 PUSH AppletCo.004338C7      ; SE handler installation
004083F7 |. 64:A1 0000000>MOV EAX,DWORD PTR FS:[0]
004083FD |. 50      PUSH EAX
004083FE |. 64:8925 00000>MOV DWORD PTR FS:[0],ESP
00408405 |. 83EC 28  SUB ESP,28
00408408 |. 53      PUSH EBX
00408409 |. 55      PUSH EBP
0040840A |. 56      PUSH ESI
0040840B |. 57      PUSH EDI
0040840C |. 6A 60      PUSH 60
0040840E |. C74424 1C 000>MOV DWORD PTR SS:[ESP+1C],0
00408416 |. E8 76C90100 CALL AppletCo.00424D91
0040841B |. 83C4 04  ADD ESP,4
0040841E |. 8BF8      MOV EDI,EAX
00408420 |. 57      PUSH EDI
00408421 |. E8 5A300000 CALL AppletCo.0040B480
00408426 |. 8B4424 50  MOV EAX,DWORD PTR SS:[ESP+50]
0040842A |. 83C4 04  ADD ESP,4
0040842D |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
00408431 |. 50      PUSH EAX
00408432 |. E8 B7C90100 CALL AppletCo.00424DEE ;<== Get Fake E-mail Address
00408437 |. 8B6C24 50  MOV EBP,DWORD PTR SS:[ESP+50]
0040843B |. C74424 40 010>MOV DWORD PTR SS:[ESP+40],1

```

00408443 |. BE E42A4400 MOV ESI,AppletCo.00442AE4 ; ASCII "SmartMails" (Magicstr1)
 00408448 |. 8B45 00 MOV EAX,DWORD PTR SS:[EBP] ;== EAX chứa chuỗi mặc định là
 "AppletCoolTextWizard100" (Magicstr2)
 0040844B |> 8A10 /MOV DL,BYTE PTR DS:[EAX] ;<== DL : Magicstr2[0]
 0040844D |. 8A1E |MOV BL,BYTE PTR DS:[ESI] ;<== BL : Magicstr1[0]
 0040844F |. 8ACA |MOV CL,DL ;<== CL : DL = Magicstr2[0]
 00408451 |. 3AD3 |CMP DL,BL ;<== DL = BL ?
 00408453 |. 75 1E |JNZ SHORT AppletCo.00408473 <== Không bằng thì nhảy tới địa chỉ
 00408473 (chắc chắn không bằng rồi, một đằng chữ A một đằng chữ S thì làm sao bằng nhau được).
 00408455 |. 84C9 |TEST CL,CL
 00408457 |. 74 16 |JE SHORT AppletCo.0040846F
 00408459 |. 8A50 01 |MOV DL,BYTE PTR DS:[EAX+1]
 0040845C |. 8A5E 01 |MOV BL,BYTE PTR DS:[ESI+1]
 0040845F |. 8ACA |MOV CL,DL
 00408461 |. 3AD3 |CMP DL,BL
 00408463 |. 75 0E |JNZ SHORT AppletCo.00408473
 00408465 |. 83C0 02 |ADD EAX,2
 00408468 |. 83C6 02 |ADD ESI,2
 0040846B |. 84C9 |TEST CL,CL
 0040846D |.^ 75 DC |JNZ SHORT AppletCo.0040844B
 0040846F |> 33C0 XOR EAX,EAX
 00408471 |. EB 05 JMP SHORT AppletCo.00408478
 00408473 |> 1BC0 SBB EAX,EAX
 00408475 |. 83D8 FF SBB EAX,-1
 00408478 |> 33C9 XOR ECX,ECX ;<== ECX = 0
 0040847A |. 85C0 TEST EAX,EAX ;<== EAX = 0?
 0040847C |. 0F95C1 SETNE CL ;<== CL = 1
 0040847F |. 84C9 TEST CL,CL ;<== CL = 0?
 00408481 |. 74 2D JE SHORT AppletCo.004084B0
 00408483 |. 8D5424 10 LEA EDX,DWORD PTR SS:[ESP+10]
 00408487 |. 8D4424 4C LEA EAX,DWORD PTR SS:[ESP+4C]
 0040848B |. 52 PUSH EDX
 0040848C |. 55 PUSH EBP
 0040848D |. 50 PUSH EAX
 0040848E |. E8 46CC0100 CALL AppletCo.004250D9 <== Sau lệnh CALL này thì chuỗi Magicstr2
 được nối với chuỗi Fake E-mail Address do User nhập vào cụ thể của tôi là :
 "AppletCoolTextWizard100kienbigmummy@yahoo.com"
 00408493 |. 50 PUSH EAX
 00408494 |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
 00408498 |. C64424 44 02 MOV BYTE PTR SS:[ESP+44],2
 0040849D |. E8 80CB0100 CALL AppletCo.00425022
 004084A2 |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
 004084A6 |. C64424 40 01 MOV BYTE PTR SS:[ESP+40],1
 004084AB |. E8 79CA0100 CALL AppletCo.00424F29
 004084B0 |> 8B4C24 10 MOV ECX,DWORD PTR SS:[ESP+10] ;<== ECX :
 "AppletCoolTextWizard100kienbigmummy@yahoo.com"
 004084B4 |. 8B51 F8 MOV EDX,DWORD PTR DS:[ECX-8] ;<== EDX chứa Length của
 chuỗi trên (chuỗi chứa trong ECX)
 004084B7 |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]
 004084BB |. 52 PUSH EDX
 004084BC |. 6A 00 PUSH 0

004084BE |. E8 17CE0100 CALL AppletCo.004252DA
 004084C3 |. 50 PUSH EAX
 004084C4 |. 57 PUSH EDI
 004084C5 |. E8 F62F0000 CALL AppletCo.0040B4C0
 004084CA |. 83C4 0C ADD ESP,0C
 004084CD |. 57 PUSH EDI
 004084CE |. E8 BD470000 CALL AppletCo.0040CC90
 004084D3 |. 8B07 MOV EAX,DWORD PTR DS:[EDI]
 004084D5 |. 83C4 04 ADD ESP,4
 004084D8 |. 894424 24 MOV DWORD PTR SS:[ESP+24],EAX
 004084DC |. 8B4F 04 MOV ECX,DWORD PTR DS:[EDI+4]
 004084DF |. 894C24 28 MOV DWORD PTR SS:[ESP+28],ECX
 004084E3 |. 8B57 08 MOV EDX,DWORD PTR DS:[EDI+8]
 004084E6 |. 895424 2C MOV DWORD PTR SS:[ESP+2C],EDX
 004084EA |. 8B47 0C MOV EAX,DWORD PTR DS:[EDI+C]
 004084ED |. 8D5424 1C LEA EDX,DWORD PTR SS:[ESP+1C]
 004084F1 |. 894424 30 MOV DWORD PTR SS:[ESP+30],EAX
 004084F5 |. 8B4F 10 MOV ECX,DWORD PTR DS:[EDI+10]
 004084F8 |. 52 PUSH EDX
 004084F9 |. 8D4424 28 LEA EAX,DWORD PTR SS:[ESP+28]
 004084FD |. 6A 14 PUSH 14
 004084FF |. 50 PUSH EAX
 00408500 |. 894C24 40 MOV DWORD PTR SS:[ESP+40],ECX
 00408504 |. E8 97FEFFFF CALL AppletCo.004083A0 ;<== mã hóa tạo ra chuỗi Right Register Code
 và lưu chuỗi này trong Stack tại địa chỉ sau : **0012F1FC 554AE9D5**
 0012F200 805AC48C
 00408509 |. 83C4 0C ADD ESP,0C
 0040850C |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14]
 00408510 |. E8 C9C80100 CALL AppletCo.00424DDE
 00408515 |. 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]
 00408519 |. 6A 08 PUSH 8
 0040851B |. 8D5424 50 LEA EDX,DWORD PTR SS:[ESP+50]
 0040851F |. 51 PUSH ECX
 00408520 |. 52 PUSH EDX
 00408521 |. C64424 4C 03 MOV BYTE PTR SS:[ESP+4C],3
00408526 |. E8 C5FDFFFF CALL AppletCo.004082F0; chuyển giá trị của Right Register Code
được lưu trong Stack thành ra chuỗi Right Register Code , cụ thể sẽ thành chuỗi :
“D5E94A558CC45A80”
 0040852B |. 83C4 0C ADD ESP,0C
 0040852E |. 50 PUSH EAX
 0040852F |. 8D4C24 18 LEA ECX,DWORD PTR SS:[ESP+18]
 00408533 |. C64424 44 04 MOV BYTE PTR SS:[ESP+44],4
 00408538 |. E8 E5CA0100 CALL AppletCo.00425022
 0040853D |. 8D4C24 4C LEA ECX,DWORD PTR SS:[ESP+4C]
 00408541 |. C64424 40 03 MOV BYTE PTR SS:[ESP+40],3
 00408546 |. E8 DEC90100 CALL AppletCo.00424F29
 0040854B |. 57 PUSH EDI
 0040854C |. E8 7CC80100 CALL AppletCo.00424DCD
 00408551 |. 8B7424 4C MOV ESI,DWORD PTR SS:[ESP+4C]
 00408555 |. 83C4 04 ADD ESP,4
 00408558 |. 8D4424 14 LEA EAX,DWORD PTR SS:[ESP+14]

```

0040855C ]. 8BCE      MOV ECX,ESI
0040855E ]. 50        PUSH EAX
0040855F ]. E8 8AC80100 CALL AppletCo.00424DEE
00408564 ]. C74424 18 010>MOV DWORD PTR SS:[ESP+18],1
0040856C ]. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]
00408570 ]. C64424 40 01  MOV BYTE PTR SS:[ESP+40],1
00408575 ]. E8 AFC90100 CALL AppletCo.00424F29
0040857A ]. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
0040857E ]. C64424 40 00  MOV BYTE PTR SS:[ESP+40],0
00408583 ]. E8 A1C90100 CALL AppletCo.00424F29
00408588 ]. 8B4C24 38  MOV ECX,DWORD PTR SS:[ESP+38]
0040858C ]. 8BC6      MOV EAX,ESI
0040858E ]. 5F        POP EDI
0040858F ]. 5E        POP ESI
00408590 ]. 5D        POP EBP
00408591 ]. 64:890D 00000>MOV DWORD PTR FS:[0],ECX
00408598 ]. 5B        POP EBX
00408599 ]. 83C4 34   ADD ESP,34
0040859C \. C3       RETN
----- Trace Into -----
*/
/* - SERIAL tương ứng :
User : kienbigmummy@yahoo.com           Serial : D5E94A558CC45A80
or
User : REA-cRaCkErTeAm@yahoo.com         Serial : 6AC8F7B5BFFC246D

```

IV – End of Tut :

- Finished – *September 22, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.srctec.com
Production :	SourceTec Sofware Co.Ltd
SoftWare :	Applet Menu Wizard Version 1.0
Copyright by :	Copyright (C) 1998 SourceTec Software Co.Ltd . All Rights Reserved.
Type :	Serial
Packed :	Not
Language :	Microsoft Visual C++ 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial

Applet Menu Wizard Version 1.0

CoolMenu is a Java Applet program which implement web navigation menu system. It could be used in web homepages. CoolMenu support both one level and two levels menu system. The one level menu looks like a group of buttons or labels, and the two levels menu looks like a mainmenu with some submenus. It will support multi-level menu system in next version....

I – Information :

- Dùng **PEiD 0.92** detect , chúng ta biết được chương trình này không bị PACK , và nó được Code bằng **Microsoft Visual C++ 5.0** . Có thể các bạn sẽ nghĩ rằng chương trình này quá cũ nhưng theo tôi điều quan trọng không phải ở chỗ đó. Mà quan trọng là chúng ta phải Crack nó thế nào....OK

- Khi chạy chương trình chúng ta sẽ nhận được cái Nag screen thông báo cho chúng ta biết rằng nếu chúng ta không bỏ tiền ra để đăng ký hợp pháp thì sẽ chỉ được sử dụng nó trong vòng 30 ngày là hết hạn kí hợp đồng hehe. Vậy thì chúng ta phải crack thôi.

- Ngay tại cái Nag screen này , chúng ta thấy có một button là **Register** , à chắc là chỗ này bắt chúng nhập thông tin đăng ký đây. Bấm vào đấy, chúng ta sẽ chuyển sang một screen mới. Cụ thể là bắt chúng ta nhập **E-mail** và **Register Code** . Nếu bác nào đăng ký hợp pháp rồi thì cứ việc nhập vô, Còn tui trong tay chẳng có gì ngoài OllyDbg hehe. Thôi thì , cứ nhập đai một FU và FS thử xem (**E-mail** : **kienbigmummy@yahoo.com ; Register Code : 11111982**), rồi nhấn Register. Bạn sẽ không phải chờ thêm một tíc tắc nào nữa, cái Nag **Sorry, That is not a valid register code** đập ngay vào mặt. Ok , chúng ta Load chương trình này trong Olly , và tìm được chuỗi này tại địa chỉ sau :

0040C7AB |. 68 D0FC4300 PUSH AppletMe.0043FCD0 ; ASCII "Sorry, that is not a valid register code."

- OK , lần ngược lên trên và chúng ta set BreakPoint tại hàm CALL có địa chỉ như sau :
0040C6CF |. E8 0F8E0100 CALL AppletMe.004254E3

II – Cracking :

- Xong , mọi thứ đã sẵn sàng . Bây giờ chúng ta nhấn F9 để Run chương trình . Sau khi nhấn xong thì màn hình thông báo của chương trình hiện lên . Chúng ta nhập vào FU và FS (ở đây mình nhập là **E-mail : kienbigmummy@yahoo.com ; Register Code : 11111982**). Sau đó nhấn Register, chúng ta sẽ quay trở lại màn hình chính của Olly , tại đây Olly sẽ Ice tại chỗ mà chúng ta đã đặt BreakPoint.:

```

0040C6CF |. E8 0F8E0100 CALL AppletMe.004254E3      ;<== We're here
0040C6D4 |. 8B7C24 20  MOV EDI,DWORD PTR SS:[ESP+20]
0040C6D8 |. 83C6 14    ADD ESI,14
0040C6DB |. 56        PUSH ESI
0040C6DC |. 8D4424 24  LEA EAX,DWORD PTR SS:[ESP+24]
0040C6E0 |. 57        PUSH EDI
0040C6E1 |. 50        PUSH EAX
0040C6E2 |. C74424 24 000>MOV DWORD PTR SS:[ESP+24],0
0040C6EA |. E8 01A0FFFF CALL AppletMe.004066F0 <== Gọi đến hàm mã hóa tạo Right Register
Code tương ứng với chuỗi mail mà bạn nhập vào.
0040C6EF |. 83C4 0C    ADD ESP,0C
0040C6F2 |. 50        PUSH EAX
0040C6F3 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
0040C6F7 |. C64424 1C 01  MOV BYTE PTR SS:[ESP+1C],1
0040C6FC |. E8 26900100 CALL AppletMe.00425727
0040C701 |. 8D4C24 20  LEA ECX,DWORD PTR SS:[ESP+20]
0040C705 |. C64424 18 00  MOV BYTE PTR SS:[ESP+18],0
0040C70A |. E8 1F8F0100 CALL AppletMe.0042562E
0040C70F |. 8B6C24 24  MOV EBP,DWORD PTR SS:[ESP+24]
0040C713 |. 8B4C24 0C    MOV ECX,DWORD PTR SS:[ESP+C] ;<== CX : Right Register Code
0040C717 |. 51        PUSH ECX      ;<== Cất vào Stack

```

```

0040C718 |. 8B55 00    MOV EDX,DWORD PTR SS:[EBP]    ;<== DX : Fake Code
0040C71B |. 52        PUSH EDX                      ;<== Cung catt vào Stack
0040C71C |. E8 CF4E0000  CALL AppletMe.004115F0      ;<== Gọi hàm so sánh 2 chuỗi này
0040C721 |. 83C4 08    ADD ESP,8
0040C724 |. 85C0        TEST EAX,EAX                ;<== EAX = 0?
0040C726 |. 75 7F        JNZ SHORT AppletMe.0040C7A7   ;<== Shot Nag
0040C728 |. E8 BF320200  CALL AppletMe.0042F9EC
0040C72D |. 8B70 04    MOV ESI,DWORD PTR DS:[EAX+4]
0040C730 |. 68 ACFC4300  PUSH AppletMe.0043FCAC    ; /Arg3 = 0043FCAC ASCII "Yes"
0040C735 |. 68 BCFC4300  PUSH AppletMe.0043FCBC    ; |Arg2 = 0043FCBC ASCII "Registered"
0040C73A |. 68 B0FC4300  PUSH AppletMe.0043FCB0    ; |Arg1 = 0043FCB0 ASCII "Register"
0040C73F |. 8BCE        MOV ECX,ESI                 ; |
0040C741 |. E8 83D40100  CALL AppletMe.00429BC9      ; \AppletMe.00429BC9
0040C746 |. 8B07        MOV EAX,DWORD PTR DS:[EDI]
0040C748 |. 8BCE        MOV ECX,ESI
0040C74A |. 50        PUSH EAX                   ; /Arg3
0040C74B |. 68 9CFC4300  PUSH AppletMe.0043FC9C    ; |Arg2 = 0043FC9C ASCII "Email Address"
0040C750 |. 68 B0FC4300  PUSH AppletMe.0043FCB0    ; |Arg1 = 0043FCB0 ASCII "Register"
0040C755 |. E8 6FD40100  CALL AppletMe.00429BC9      ; \AppletMe.00429BC9
0040C75A |. 8B4D 00    MOV ECX,DWORD PTR SS:[EBP]
0040C75D |. 51        PUSH ECX                   ; /Arg3
0040C75E |. 68 8CFC4300  PUSH AppletMe.0043FC8C    ; |Arg2 = 0043FC8C ASCII "Register Code"
0040C763 |. 68 B0FC4300  PUSH AppletMe.0043FCB0    ; |Arg1 = 0043FCB0 ASCII "Register"
0040C768 |. 8BCE        MOV ECX,ESI                 ; |
0040C76A |. E8 5AD40100  CALL AppletMe.00429BC9      ; \AppletMe.00429BC9
0040C76F |. 6A 00        PUSH 0
0040C771 |. 6A 00        PUSH 0
0040C773 |. 68 FCFC4300  PUSH AppletMe.0043FCFC    ; ASCII " Thank you ! Please restart"
0040C778 |. E8 65D20100  CALL AppletMe.004299E2
0040C77D |. 8D4C24 0C    LEA ECX,DWORD PTR SS:[ESP+C]
0040C781 |. C74424 18 FFF>MOV DWORD PTR SS:[ESP+18],-1
0040C789 |. E8 A08E0100  CALL AppletMe.0042562E
0040C78E |. B8 01000000  MOV EAX,1
0040C793 |. 8B4C24 10    MOV ECX,DWORD PTR SS:[ESP+10]
0040C797 |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX
0040C79E |. 5F        POP EDI
0040C79F |. 5E        POP ESI
0040C7A0 |. 5D        POP EBP
0040C7A1 |. 83C4 10    ADD ESP,10
0040C7A4 |. C2 0800    RETN 8
0040C7A7 |> 6A 00        PUSH 0
0040C7A9 |. 6A 00        PUSH 0
0040C7AB |. 68 D0FC4300  PUSH AppletMe.0043FCD0 ; ASCII "Sorry, that is not a valid register code."
0040C7B0 |. E8 2DD20100  CALL AppletMe.004299E2

```

- Đến đây là thành công rồi đó các bạn , hòa quang lóe sáng tại thanh ghi ECX :đó chính là Right Register Code sau khi mã hóa chuỗi Mail mà chúng ta nhập vào. Thực chất là sau khi chúng ta nhập chuỗi Mail vào thì chương trình sẽ thêm một chuỗi khác vào đầu của chuỗi Mail .Cụ thể chuỗi đó là : "AppletMenuWizard100kienbigmummy@yahoo.com". Bạn có thể thấy chuỗi này được thêm vào đầu chuỗi Mail mà tôi đã nhập. Sau đó chính chuỗi này sẽ được đem đi mã hóa tạo ra Register Code cho

chúng ta.

/*/*/* - SERIAL tương ứng :

User : kienbigmummy@yahoo.com Serial : FCEEA79A3B4A5070

or

User : REA-cRaCkErTeAm@yahoo.com Serial : 45D9C6CDF2327D37

- Note : Đối với phiên bản 2.0 của soft này cũng không có gì thay đổi trong cách thức mã hóa tạo Register Code . Cho nên mọi thông số dùng cho version 1.0 đều có thể dùng trong phiên bản 2.0

IV – End of Tut :

- Finished – September 22, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.srctec.com
Production :	SourceTec Sofware Co.Ltd
SoftWare :	Applet SlidingMenu Wizard Version 1.0
Copyright by :	Copyright (C) 1998 SourceTec Software Co.Ltd . All Rights Reserved.
Type :	Serial
Packed :	Not
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N/A
Request :	Correct Serial

Applet SlidingMenu Wizard Version 1.0

Applet SlidingMenu Wizard is a powerful tool developed by SourceTec Software Co. Ltd , which can create neat, efficient Java navigation menu applet (SlidingMenu) for your Internet or Intranet Web pages with no need of programming.

Applet SlidingMenu Wizard guides user to generate Java Menu applet. The configuration doesn't require any Java or HTML coding, so it is suitable for any web designers. With Applet SlidingMenu Wizard, you could implement applets of various effects: Sliding Tree, Radio Buttons and other special effects. It also provides pre-made templates to make work easier. In addition, you could preview the generated applet without quitting Applet SlidingMenu Wizard.

I – Information :

- Dùng PEiD 0.92 detect , chúng ta biết được chương trình này không bị PACK , và nó được Code bằng Microsoft Visual C++ 6.0 . Đến phiên bản này của hãng SourceTec thì đã được Code bằng VC++6.0 nhưng hình như tác giả quá tin tưởng vào cách mã hóa của mình nên không PACK lại (do đó keke đỡ mệt trong việc UnPack chương trình. Vậy giờ chúng ta tiến hành tìm kiếm thêm thông tin.

- Khi chạy chương trình chúng ta sẽ nhận được cái Nag screen thông báo cho chúng ta biết rằng nếu chúng ta không bỏ tiền ra để đăng ký hợp pháp thì sẽ chỉ được sử dụng nó trong vòng 30 ngày là hết hạn kí hợp đồng hehe. Vậy thì chúng ta phải crack thôi.(Giống hệt các Soft MenuWizard, CoolTextWizard của hãng này).

- Ngay tại cái Nag screen này , chúng ta thấy có một button là **Register** , à chắc là chỗ này bắt chúng nhập thông tin đăng kí đây. Bấm vào đây, chúng ta sẽ chuyển sang một screen mới. Cụ thể là bắt chúng ta nhập **E-mail và Register Code** . Nếu bác nào đăng kí hợp pháp rồi thì cứ việc nhập vô, Còn tui trong tay chẳng có gì ngoài OllyDbg hehe. Thôi thì , cứ nhập đai một FU và FS thử xem (**E-mail :**

kienbigmummy@yahoo.com ; Register Code : 11111982), rồi nhấn Register. Bạn sẽ không phải chờ thêm một tíc tắc nào nữa, cái Nag **Sorry, That is not a valid register code** đập ngay vào mặt. Ok , chúng ta Load chương trình này trong Olly , và tìm được chuỗi này tại địa chỉ sau :

00416E6D |. 68 E0514600 PUSH AppletSI.004651E0 ;|Arg1 = 004651E0 ASCII "Sorry, that is not a valid register code."

- OK , lần ngược lên trên và chúng ta set BreakPoint tại hàm CALL có địa chỉ như sau :
00416DA7 |. E8 54DCFFFF CALL AppletSI.00414A00

II – Cracking :

- Xong , mọi thứ đã sẵn sàng . Bây giờ chúng ta nhấn F9 để Run chương trình . Lần này khác với các Soft trước , sau khi nhấn xong thì màn hình thông báo của chương trình vẫn chưa thấy hiện lên gì cả. Không lẽ nó biến mất đâu rồi !!!! Các bạn đừng quá lo lắng nếu như thấy trong Olly Ice tại địa chỉ sau :

77E6D756 5E POP ESI ; AppletSI.004581E8

- Tại địa chỉ trên chúng ta nhấn Shift + F9 (1 lần) thì cái Nag bắt chúng ta phải đăng kí sẽ hiện ngay lên thôi.Chúng ta nhập vào FU và FS (ở đây mình nhập là **E-mail : kienbigmummy@yahoo.com ; Register Code : 11111982**). Sau đó nhấn Register, chúng ta sẽ quay trở lại màn hình chính của Olly , tại đây Olly sẽ Ice tại chỗ mà chúng ta đã đặt BreakPoint.:

00416DA7 |. E8 54DCFFFF CALL AppletSI.00414A00 <== We're here : đây cũng chính là nơi gọi hàm mã hóa tạo ra Register Code (Nếu muốn Keygen thì Trace Into vào trong hàm này).

00416DAC |. 83C4 0C ADD ESP,0C

00416DAF |. 50 PUSH EAX

00416DB0 |. 8D4C24 0C LEA ECX,DWORD PTR SS:[ESP+C]

00416DB4 |. C64424 18 01 MOV BYTE PTR SS:[ESP+18],1

00416DB9 |. E8 763C0100 CALL AppletSI.0042AA34

00416DBE |. 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]

00416DC2 |. C64424 14 00 MOV BYTE PTR SS:[ESP+14],0

00416DC7 |. E8 7B3B0100 CALL AppletSI.0042A947

00416DCC |. 8B6C24 20 MOV EBP,DWORD PTR SS:[ESP+20]

00416DD0 |. 8B5424 08 MOV EDX,DWORD PTR SS:[ESP+8] ;<== EDX :Right Register Code

00416DD4 |. 52 PUSH EDX ;<== Cắt vào Stack

00416DD5 |. 8B45 00 MOV EAX,DWORD PTR SS:[EBP] ;<== EAX : Fake Code

00416DD8 |. 50 PUSH EAX ;<== Đẩy vào Stack

00416DD9 |. E8 82350000 CALL AppletSI.0041A360 ;<== Để so sánh Right Code với Fake Code

00416DDE |. 83C4 08 ADD ESP,8

00416DE1 |. 85C0 TEST EAX,EAX ;<== EAX = 0?

00416DE3 |. 0F85 80000000 JNZ AppletSI.00416E69 <== Nếu không bằng thì bắn Nag vào mặt.

00416DE9 |. 56 PUSH ESI

00416DEA |. E8 93E10200 CALL AppletSI.00444F82

00416DEF |. 8B70 04 MOV ESI,DWORD PTR DS:[EAX+4]

00416DF2 |. 68 BC514600 PUSH AppletSI.004651BC ;|Arg3 = 004651BC ASCII "Yes"

00416DF7 |. 68 CC514600 PUSH AppletSI.004651CC ;|Arg2 = 004651CC ASCII "Registered"

00416DFC |. 68 C0514600 PUSH AppletSI.004651C0 ;|Arg1 = 004651C0 ASCII "Register"

```

00416E01 |. 8BCE      MOV ECX,ESI          ; |
00416E03 |. E8 BDF90100 CALL AppletSl.004367C5    ; \AppletSl.004367C5
00416E08 |. 8B07      MOV EAX,DWORD PTR DS:[EDI]
00416E0A |. 8BCE      MOV ECX,ESI
00416E0C |. 50        PUSH EAX           ; /Arg3
00416E0D |. 68 AC514600 PUSH AppletSl.004651AC ; |Arg2 = 004651AC ASCII "Email Address"
00416E12 |. 68 C0514600 PUSH AppletSl.004651C0 ; |Arg1 = 004651C0 ASCII "Register"
00416E17 |. E8 A9F90100 CALL AppletSl.004367C5 ; \AppletSl.004367C5
00416E1C |. 8B45 00    MOV EAX,DWORD PTR SS:[EBP]
00416E1F |. 8BCE      MOV ECX,ESI
00416E21 |. 50        PUSH EAX           ; /Arg3
00416E22 |. 68 9C514600 PUSH AppletSl.0046519C ; |Arg2 = 0046519C ASCII "Register Code"
00416E27 |. 68 C0514600 PUSH AppletSl.004651C0 ; |Arg1 = 004651C0 ASCII "Register"
00416E2C |. E8 94F90100 CALL AppletSl.004367C5 ; \AppletSl.004367C5
00416E31 |. 6A 00      PUSH 0            ; /Arg3 = 00000000
00416E33 |. 6A 00      PUSH 0            ; |Arg2 = 00000000
00416E35 |. 68 0C524600 PUSH AppletSl.0046520C ; |Arg1 = 0046520C ASCII " Thank you ! Please
                                                 restart"
00416E3A |. E8 3BFB0100 CALL AppletSl.0043697A    ; \AppletSl.0043697A
00416E3F |. 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
00416E43 |. C74424 18 FFF>MOV DWORD PTR SS:[ESP+18],-1
00416E4B |. E8 F73A0100 CALL AppletSl.0042A947
00416E50 |. 5E        POP ESI
00416E51 |. 5F        POP EDI
00416E52 |. B8 01000000 MOV EAX,1
00416E57 |. 5D        POP EBP
00416E58 |. 8B4C24 04  MOV ECX,DWORD PTR SS:[ESP+4]
00416E5C |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX
00416E63 |. 83C4 10    ADD ESP,10
00416E66 |. C2 0800    RETN 8
00416E69 |> 6A 00      PUSH 0            ; /Arg3 = 00000000
00416E6B |. 6A 00      PUSH 0            ; |Arg2 = 00000000
00416E6D |. 68 E0514600 PUSH AppletSl.004651E0 ; |Arg1 = 004651E0 ASCII "Sorry, that is
not a valid register code."
00416E72 |. E8 03FB0100 CALL AppletSl.0043697A ; \AppletSl.0043697A

```

- Đến đây là thành công rồi đó các bạn , hòa quang lóe sáng tại thanh ghi EDX :đó chính là Right Register Code sau khi mã hóa chuỗi Mail mà chúng ta nhập vào. Thực chất là sau khi chúng ta nhập chuỗi Mail vào thì chương trình sẽ thêm một chuỗi khác vào đầu của chuỗi Mail .Cụ thể chuỗi đó là : "AppletSlidingMenuWizard100kienbigmummy@yahoo.com". Bạn có thể thấy chuỗi này được thêm vào đầu chuỗi Mail mà tôi đã nhập. Sau đó chính chuỗi này sẽ được đếm đi mã hóa tạo ra Register Code cho chúng ta.

/*/*/* - SERIAL tương ứng :

User : kienbigmummy@yahoo.com Serial : **5B7187050FBB5BE4**

or

User : REA-cRaCkErTeAm@yahoo.com Serial : **5D1CA69EFCD771DA**

IV – End of Tut :

- Finished – **September 22, 2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.FirasE.com
Production :	Firas El - Hasan
SoftWare :	Auto Connect v1.01
Copyright by :	Copyright (C) 2000 Firas El - Hasan . All Rights Reserved.
Type :	Name/Serial
Packed :	Not
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial/Keygen

Auto Connect v1.01

AutoConnect will automatically hit the Reconnect button for you when your ISP disconnects you from your Dial Up Connection.

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng Microsoft Visual C++ 6.0 , không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là **Register...** . Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập **Name & Serial** vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ở đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (**Name : kienmanowar & Code : 11111982**). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ **Sorry , you have entered an incorrect registration code** ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Chúng ta sẽ tìm thấy tại địa chỉ sau :
00401CE1 . 68 98624000 PUSH AutoConn.00406298 ;|Text = "Sorry, you have entered an incorrect registration code."

-**** Lần ngược lên trên chúng ta sẽ đặt Break Point tại đây :
00401C5F . FF15 F4504000 CALL DWORD PTR DS:[<&USER32.GetItemTe>;\GetDlgItemTextA <== We Set BP here

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

00401C5F . FF15 F4504000 CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA>;

\GetDlgItemTextA <== We're here (Get FU)

00401C65 . 68 00010000 PUSH 100 ; /Count = 100 (256.)

00401C6A . 8D8D 00FEFFFF LEA ECX,DWORD PTR SS:[EBP-200] ; |

00401C70 . 51 PUSH ECX ; |Buffer

00401C71 . 68 FC030000 PUSH 3FC ; |ControlID = 3FC (1020.)

00401C76 . 8B55 08 MOV EDX,DWORD PTR SS:[EBP+8] ; |

00401C79 . 52 PUSH EDX ; |hWnd

00401C7A . FF15 F4504000 CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA>;

\GetDlgItemTextA ;<== Get FS

00401C80 . 8D85 00FEFFFF LEA EAX,DWORD PTR SS:[EBP-200] ; <== FS

00401C86 . 50 PUSH EAX ; /Arg2

00401C87 . 8D8D 00FFFFFF LEA ECX,DWORD PTR SS:[EBP-100] ; | <== FU

00401C8D . 51 PUSH ECX ; |Arg1

00401C8E . E8 CC030000 CALL AutoConn.0040205F ; \AutoConn.0040205F <== Encrypt (Trace Into)

----- Trace Into -----

0040205F /\$ 55 PUSH EBP

00402060 . 8BEC MOV EBP,ESP

00402062 . 81EC 04010000 SUB ESP,104

00402068 . C745 FC 00000>MOV [LOCAL.1],0

0040206F . 8D85 FCFEFFFF LEA EAX,[LOCAL.65]

00402075 . 50 PUSH EAX ; /Arg2

00402076 . 8B4D 08 MOV ECX,[ARG.1] ; |

00402079 . 51 PUSH ECX ; |Arg1

0040207A . E8 B5000000 CALL AutoConn.00402134 ; \AutoConn.00402134 <== Calculation(Trace Into)

----- Trace Into -----

00402134 /\$ 55 PUSH EBP

00402135 . 8BEC MOV EBP,ESP

00402137 . 81EC 0C010000 SUB ESP,10C

0040213D . 57 PUSH EDI

0040213E . A0 F4684000 MOV AL,BYTE PTR DS:[4068F4]

00402143 . 8885 00FFFFFF MOV BYTE PTR SS:[EBP-100],AL

00402149 . B9 3F000000 MOV ECX,3F

0040214E . 33C0 XOR EAX,EAX

00402150 . 8DBD 01FFFFFF LEA EDI,DWORD PTR SS:[EBP-FF]

00402156 . F3:AB REP STOS DWORD PTR ES:[EDI]

00402158 . 66:AB STOS WORD PTR ES:[EDI]

0040215A . AA STOS BYTE PTR ES:[EDI]

0040215B . 8B4D 08 MOV ECX,[ARG.1] ; <== FU

0040215E . 51 PUSH ECX ; /String

0040215F . FF15 24504000 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ; \lstrlenA <== Get Length(FU)

00402165 . 8985 F4FEFFFF MOV [LOCAL.67],EAX ; <== Length(FU);

0040216B . C785 F8FEFFFF>MOV [LOCAL.66],0 ; <== Temp_01 = 0

00402175 . C785 FCFEFFFF>MOV [LOCAL.65],0 ; <== i = 0;

0040217F . EB 0F JMP SHORT AutoConn.00402190

----- Calculation 0 -----

```

00402181 > 8B95 FCFFFFF /MOV EDX,[LOCAL.65] ;<== i
00402187 |. 83C2 01 |ADD EDX,1 ;<== i ++
0040218A |. 8995 FCFFFFF |MOV [LOCAL.65],EDX ;<== i
00402190 > 8B85 FCFFFFF MOV EAX,[LOCAL.65] ;<== i
00402196 |. 3B85 F4FFFFF |CMP EAX,[LOCAL.67] ;<== If (i > Length(FU)) then
0040219C |. 73 22 |JNB SHORT AutoConn.004021C0 ;<== Exit calculation
0040219E |. 8B4D 08 |MOV ECX,[ARG.1] ;<== FU
004021A1 |. 038D FCFFFFF |ADD ECX,[LOCAL.65]
004021A7 |. 0FBE11 |MOVSX EDX,BYTE PTR DS:[ECX] ;<== FU[i]
004021AA |. 0315 3C604000 |ADD EDX,DWORD PTR DS:[40603C] ;<== Temp = FU[i] + 0x1F
004021B0 |. 8B85 F8FFFFF |MOV EAX,[LOCAL.66] ;<== Temp_01
004021B6 |. 03C2 |ADD EAX,EDX ;<== Temp_01 = Temp_01 + Temp
004021B8 |. 8985 F8FFFFF |MOV [LOCAL.66],EAX ;<== Temp_01
004021BE |.^ EB C1 |JMP SHORT AutoConn.00402181 ;<== Continue Loop
004021C0 > 8B8D F8FFFFF MOV ECX,[LOCAL.66] ;<== Sec1 = Temp_01
004021C6 |. 51 PUSH ECX ;/<%u> ;<== Sec1
004021C7 |. 68 2C634000 PUSH AutoConn.0040632C ;|Format = "%u-"
004021CC |. 8B55 0C MOV EDX,[ARG.2] ;|
004021CF |. 52 PUSH EDX ;|s
004021D0 |. FF15 30514000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ;\wsprintfA

```

===== Calculation 0 =====

```

004021D6 |. 83C4 0C ADD ESP,0C
004021D9 |. C785 F8FFFFF>MOV [LOCAL.66],0 ;<== Temp_02 = 0;
004021E3 |. C785 FCFFFFF>MOV [LOCAL.65],0 ;<== i = 0;
004021ED |. EB 0F JMP SHORT AutoConn.004021FE

```

===== Calculation 1 =====

```

004021EF > 8B85 FCFFFFF /MOV EAX,[LOCAL.65] ;<== i
004021F5 |. 83C0 01 |ADD EAX,1 ;<== i ++
004021F8 |. 8985 FCFFFFF |MOV [LOCAL.65],EAX ;<== i
004021FE > 8B8D FCFFFFF MOV ECX,[LOCAL.65] ;<== i
00402204 |. 3B8D F4FFFFF |CMP ECX,[LOCAL.67] ;<== If ( i > Length(FU)) then
0040220A |. 73 23 |JNB SHORT AutoConn.0040222F ;<== Exit Calculation
0040220C |. 8B55 08 |MOV EDX,[ARG.1] ;<== FU
0040220F |. 0395 FCFFFFF |ADD EDX,[LOCAL.65]
00402215 |. 0FBE02 |MOVSX EAX,BYTE PTR DS:[EDX] ;<== FU[i]
00402218 |. 0FAF05 406040>|IMUL EAX,DWORD PTR DS:[406040] ;<== Temp = FU[i]*0xC
0040221F |. 8B8D F8FFFFF |MOV ECX,[LOCAL.66] ;<== Temp_02
00402225 |. 03C8 |ADD ECX,EAX ;<== Temp_02 = Temp_02 + Temp
00402227 |. 898D F8FFFFF |MOV [LOCAL.66],ECX ;<== Temp_02
0040222D |.^ EB C0 |JMP SHORT AutoConn.004021EF ;<== Continue Loop
0040222F > 8B95 F8FFFFF MOV EDX,[LOCAL.66] ;<== Sec2 = Temp_02
00402235 |. 52 PUSH EDX ;/<%u>
00402236 |. 68 30634000 PUSH AutoConn.00406330 ;|Format = "%u-"
0040223B |. 8D85 00FFFFFF LEA EAX,[LOCAL.64] ;|
00402241 |. 50 PUSH EAX ;|s
00402242 |. FF15 30514000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ;\wsprintfA

```

===== Calculation 1 =====

```

00402248 |. 83C4 0C ADD ESP,0C
0040224B |. 8D8D 00FFFFFF LEA ECX,[LOCAL.64]
00402251 |. 51 PUSH ECX ;/StringToAdd
00402252 |. 8B55 0C MOV EDX,[ARG.2] ;|

```

```

00402255 |. 52      PUSH EDX          ; |ConcatString
00402256 |. FF15 28504000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA
0040225C |. C785 F8FFFFFF MOV [LOCAL.66],0    ;<== Temp_03 = 0;
00402266 |. C785 FCFFFFFF MOV [LOCAL.65],0    ;<== i = 0;
00402270 |. EB 0F      JMP SHORT AutoConn.00402281

```

===== Calculation 2 =====

```

00402272 |> 8B85 FCFFFFFF /MOV EAX,[LOCAL.65] ;<== i
00402278 |. 83C0 01  |ADD EAX,1        ;<== i ++
0040227B |. 8985 FCFFFFFF |MOV [LOCAL.65],EAX ;<== i
00402281 |> 8B8D FCFFFFFF MOV ECX,[LOCAL.65] ;<== i
00402287 |. 3B8D F4FFFFFF |CMP ECX,[LOCAL.67] ;<== If ( i < Length(FU)) then
0040228D |. 73 22      |JNB SHORT AutoConn.004022B1;<== Exit Calculation
0040228F |. 8B55 08  |MOV EDX,[ARG.1]     ;<== FU
00402292 |. 0395 FCFFFFFF |ADD EDX,[LOCAL.65]
00402298 |. 0FBE02  |MOVSX EAX,BYTE PTR DS:[EDX] ;<== FU[i]
0040229B |. 0305 44604000 |ADD EAX,DWORD PTR DS:[406044] ;<== Temp = FU[i] + 0xD
004022A1 |. 8B8D F8FFFFFF |MOV ECX,[LOCAL.66]       ;<== Temp_03
004022A7 |. 03C8      |ADD ECX,EAX        ;<== Temp_03 = Temp_03 + Temp
004022A9 |. 898D F8FFFFFF |MOV [LOCAL.66],ECX     ;<== Temp_03
004022AF |.^ EB C1      |JMP SHORT AutoConn.00402272 ;<== Continue Loop
004022B1 |> 8B95 F8FFFFFF MOV EDX,[LOCAL.66]     ;<== Sec3 = Temp_03
004022B7 |. 52      PUSH EDX          ;/<%u>
004022B8 |. 68 34634000 PUSH AutoConn.00406334      ;|Format = "%u-"
004022BD |. 8D85 00FFFFFF LEA EAX,[LOCAL.64]      ;|
004022C3 |. 50      PUSH EAX          ;|s
004022C4 |. FF15 30514000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

===== Calculation 2 =====

```

004022CA |. 83C4 0C      ADD ESP,0C
004022CD |. 8D8D 00FFFFFF LEA ECX,[LOCAL.64]
004022D3 |. 51      PUSH ECX          ; /StringToAdd
004022D4 |. 8B55 0C      MOV EDX,[ARG.2]      ;|
004022D7 |. 52      PUSH EDX          ; |ConcatString
004022D8 |. FF15 28504000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA
004022DE |. C785 F8FFFFFF MOV [LOCAL.66],0    ;<== Temp_04 = 0;
004022E8 |. C785 FCFFFFFF MOV [LOCAL.65],0    ;<== i = 0;
004022F2 |. EB 0F      JMP SHORT AutoConn.00402303

```

===== Calculation 3 =====

```

004022F4 |> 8B85 FCFFFFFF /MOV EAX,[LOCAL.65]      ;<== i
004022FA |. 83C0 01  |ADD EAX,1        ;<== i ++
004022FD |. 8985 FCFFFFFF |MOV [LOCAL.65],EAX     ;<== i
00402303 |> 8B8D FCFFFFFF MOV ECX,[LOCAL.65]     ;<== i
00402309 |. 3B8D F4FFFFFF |CMP ECX,[LOCAL.67]     ;<== If ( i > Length(FU)) then
0040230F |. 73 23      |JNB SHORT AutoConn.00402334 ;<== Exit Calculation
00402311 |. 8B55 08  |MOV EDX,[ARG.1]     ;<== FU
00402314 |. 0395 FCFFFFFF |ADD EDX,[LOCAL.65]
0040231A |. 0FBE02  |MOVSX EAX,BYTE PTR DS:[EDX] ;<== FU[i]
0040231D |. 0FAF05 486040>|IMUL EAX,DWORD PTR DS:[406048] ;<== Temp = FU[i] * 0x20
00402324 |. 8B8D F8FFFFFF |MOV ECX,[LOCAL.66]       ;<== Temp_04
0040232A |. 03C8      |ADD ECX,EAX        ;<== Temp_04 = Temp_04 + Temp
0040232C |. 898D F8FFFFFF |MOV [LOCAL.66],ECX     ;<== Temp_04
00402332 |.^ EB C0      |JMP SHORT AutoConn.004022F4 ;<== Continue Loop
00402334 |> 8B95 F8FFFFFF MOV EDX,[LOCAL.66]     ;<== Sec4 = Temp_04

```

```

0040233A |. 52      PUSH EDX          ; /<%u>
0040233B |. 68 38634000 PUSH AutoConn.00406338    ; |Format = "%u"
00402340 |. 8D85 00FFFFFF LEA EAX,[LOCAL.64]    ; |
00402346 |. 50      PUSH EAX          ; |s
00402347 |. FF15 30514000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

===== Calculation 3 =====

```

0040234D |. 83C4 0C      ADD ESP,0C
00402350 |. 8D8D 00FFFFFF LEA ECX,[LOCAL.64]
00402356 |. 51      PUSH ECX          ; /StringToAdd
00402357 |. 8B55 0C      MOV EDX,[ARG.2]        ; |
0040235A |. 52      PUSH EDX          ; |ConcatString
0040235B |. FF15 28504000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA ;<===
Right Code after Calculation
00402361 |. 5F      POP EDI
00402362 |. 8BE5      MOV ESP,EBP
00402364 |. 5D      POP EBP
00402365 \. C3      RETN

```

===== Trace Into =====

```

0040207F |. 83C4 08      ADD ESP,8
00402082 |. 8D95 FCFEFFFF LEA EDX,[LOCAL.65]    ; <== Right Code
00402088 |. 52      PUSH EDX
00402089 |. 8B45 0C      MOV EAX,[ARG.2]        ; <== Fake Serial
0040208C |. 50      PUSH EAX
0040208D |. E8 7E040000 CALL AutoConn.00402510 ; <== Compare
00402092 |. 83C4 08      ADD ESP,8
00402095 |. 85C0      TEST EAX,EAX
00402097 |. 75 07      JNZ SHORT AutoConn.004020A0
00402099 |. C745 FC 01000>MOV [LOCAL.1],1
004020A0 |> 68 00010000 PUSH 100
004020A5 |. 6A 00      PUSH 0
004020A7 |. 8D8D FCFEFFFF LEA ECX,[LOCAL.65]
004020AD |. 51      PUSH ECX
004020AE |. E8 FD030000 CALL AutoConn.004024B0
004020B3 |. 83C4 0C      ADD ESP,0C
004020B6 |. 8B45 FC      MOV EAX,[LOCAL.1]
004020B9 |. 8BE5      MOV ESP,EBP
004020BB |. 5D      POP EBP
004020BC \. C3      RETN

```

===== Trace Into =====

```

00401C93 . 83C4 08      ADD ESP,8
00401C96 . 85C0      TEST EAX,EAX          ; <== Wrong Code
00401C98 . 74 40      JE SHORT AutoConn.00401CDA ; <== Jump to Nag
00401C9A . 8D95 00FEFFFF LEA EDX,DWORD PTR SS:[EBP-200]
00401CA0 . 52      PUSH EDX          ; /Arg4
00401CA1 . 8D85 00FFFFFF LEA EAX,DWORD PTR SS:[EBP-100]    ; |
00401CA7 . 50      PUSH EAX          ; |Arg3
00401CA8 . 68 44624000 PUSH AutoConn.00406244; |Arg2 = 00406244 ASCII
"Software\AutoConnect\Registration"
00401CAD . 68 01000080 PUSH 80000001      ; |Arg1 = 80000001
00401CB2 . E8 06040000 CALL AutoConn.004020BD ; \AutoConn.004020BD
00401CB7 . 83C4 10      ADD ESP,10
00401CBA . 68 68624000 PUSH AutoConn.00406268 ; /Arg2 = 00406268 ASCII
"Software\AutoConnect\Registration"

```

```
00401CBF . 68 01000080 PUSH 80000001 ; |Arg1 = 80000001
00401CC4 . E8 50020000 CALL AutoConn.00401F19 ; |\AutoConn.00401F19
00401CC9 . 83C4 08 ADD ESP,8
00401CCC . 6A 01 PUSH 1 ; /Result = 1
00401CCE . 8B4D 08 MOV ECX,DWORD PTR SS:[EBP+8] ; |
00401CD1 . 51 PUSH ECX ; |hWnd
00401CD2 . FF15 28514000 CALL DWORD PTR DS:[<&USER32.EndDialog>] ; \EndDialog
00401CD8 . EB 16 JMP SHORT AutoConn.00401CF0
00401CDA > 6A 00 PUSH 0 ; /Style = MB_OK|MB_APPLMODAL
00401CDC . 68 8C624000 PUSH AutoConn.0040628C ; |Title = "AutoConnect"
00401CE1 . 68 98624000 PUSH AutoConn.00406298 ; |Text = "Sorry, you have entered an
incorrect registration code."
00401CE6 . 8B55 08 MOV EDX,DWORD PTR SS:[EBP+8] ; |
00401CE9 . 52 PUSH EDX ; |hOwner
00401CEA . FF15 F8504000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>];
\MessageBoxA <== Shot Nag
```

/*/*/*/ - SERIAL tương ứng :

User : kienmanowar Serial : 1521-14160-1323-37760

User : REA-cRaCkErTeAm Serial : 1720-15060-1450-40160

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;
int i=0,j=0;
int Temp=0, Temp1 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0,Sec4_temp = 0;

LenName=GetDlgItemText(IDC_NAME,reaName,64);
if (LenName < 1)
{
    MessageBox("===== Your name atleast 1 chart ===== ", "Hey !! Please input
your name again !! ");
}
else
{
    // Caculation 0
    while (i < LenName)
    {
        Temp = reaName[i] + 0x1F;
        Temp1 = Temp1 + Temp ;
        i++;
    }
    Sec1_temp = Temp1;
    wsprintf(Sec1,"%u-",Sec1_temp);
    i = 0;
    Temp = 0;
    Temp1 = 0;
    // Calculation 1
    while (i < LenName)

```

```

{
    Temp = reaName[i] * 0xC;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec2_temp = Temp1;
wsprintf(Sec2,"%u-",Sec2_temp);
//Calculation 2
i = 0;
Temp = 0 ;
Temp1 = 0;
while ( i < LenName)
{
    Temp = reaName[i] + 0xD;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec3_temp = Temp1;
wsprintf(Sec3,"%u-",Sec3_temp);
//Calculation 3
i = 0;
Temp = 0;
Temp1 = 0;
while (i< LenName)
{
    Temp = reaName[i] * 0x20;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec4_temp = Temp1;
wsprintf(Sec4,"%u",Sec4_temp);
}
// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);
SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://cmbsoftware.com
Production	:	CMB Software
SoftWare	:	CMB Audio Player v2.0.0
Copyright by	:	Copyright (C) 2001 CMB Software . All Rights Reserved.
Type	:	Serial
Packed	:	Not
Language	:	Microsoft Visual Basic 5.0/6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack	:	N/A
Request	:	Correct Serial

CMB Audio Player v2.0.0

CMB Audio Player is a system tray utility that allows you to organize your favorite music files (mp3, wav, mid, rmi) into multiple play lists and then play them back in the background while you are doing other tasks! You can create an unlimited number of play lists. Plus you can at anytime you can repeat a track, select the next or previous track, or even select a random track!

CMB Audio Player has a built in .ZIP tool that allows you to turn your favorite play lists into a .ZIP file that you can then send to a friend via your email service!

I – Information :

-**** Sau khi download chương trình này về , unzip và set up nó vào ổ cứng . Run file **CMB Audio Player.exe** , chúng ta nhận được một màn hình **Welcome** . Nhấn **Continue Button....** để tiếp tục tìm kiếm thông tin . Ấc ặc chúng ta nhận được một màn hình tiếp theo thông báo cho chúng ta biết là chúng ta chỉ có thể Run chương trình này được 7 lần mà thôi (*You can run this program 7 more times until you must register*) . Quá 7 lần là tự nghỉ hưu không cho sử dụng tiếp nữa .

-**** Tại màn hình này , chúng ta thấy có 2 button nữa là *Click here to learn how to Register the CMB Audio Player* và *Continue*. Nhấn vào button thứ nhất , chúng ta sẽ đến màn hình nhập **Registration Code** . Cũng tại màn hình này chúng ta biết được Price của chương trình là **9.95\$** (hiii một cái giá vẫn quá cao với sinh viên chúng ta).

-**** Oki , tại đây chúng ta tiến hành nhập thử **Fake Code** , mình nhập là **123456** . Sau đó nhấn Enter . Ọc chẳng thấy cái Nag nào bắn ra hết cả , khó rùi đây. Phỏng đoán nó sẽ lưu thông tin lại rùi lôi ra check ở lần run sau. Thủ close nó rùi Run lại nó vẫn thè không tìm kiếm thêm được gì.

-**** Dùng **PeiD** biết được nó không bị Pack và được tác giả của nó Code bằng **Visual Basic 5.0/6.0** . Detect crypto thì biết được thằng này không sử dụng Crypto nào :).

-**** Load file **.exe** vào trong Olly , dùng chức năng **Search String** để tìm kiếm xem có thông tin nào quí giá không. Tìm kiếm một hồi chúng ta thấy được một String rất quí giá sau : 0042DCA2 . C745 9C 08BB4>MOV DWORD PTR SS:[EBP-64],CMB_Audi.0040B>; UNICODE "Registered!"

-**** Double click lên String trên , chúng ta sẽ quay trở lại màn hình chính của Olly. Dịch lên một chút chúng ta đặt BP tại hàm sau :

```
0042DB17 . FF91 A0000000 CALL DWORD PTR DS:[ECX+A0] ;<== Set BP here
```

II – Cracking :

-**** Sau khi đặt BP tại hàm trên , chúng ta nhấn **Ctrl + F2** để Restart lại , tiếp theo nhấn **F9** để run chương trình. Chúng ta tiến hành nhập **Fake Serial** vào (chú ý các bạn nên gõ FS ra một file sau đó copy lại và paste vào, lý do là vì nếu bạn gõ thì chỉ cần gõ 1 kí tự thôi nó sẽ tự Break trong Olly ngay). Sau khi nhập xong chúng ta sẽ Break tại hàm trên :

```
0042DB17 . FF91 A0000000 CALL DWORD PTR DS:[ECX+A0] ;<== We're here
```

-**** Trace qua đoạn code này , chúng ta sẽ đến đây:

```
0042DB35 >\8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18] ;<== Fake Serial
0042DB38 . 50 PUSH EAX
0042DB39 . FF15 24104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaLenBs>; MSVBVM60.__vbaLenBstr] ;<== Get Length(FS)
0042DB3F . 33C9 XOR ECX,ECX
0042DB41 . 83F8 06 CMP EAX,6 ;<== Length (FS) must equal 6
0042DB44 . 0F95C1 SETNE CL ;<== CL = 1 if Length (FS) <> 6
0042DB47 . F7D9 NEG ECX
0042DB49 . 8BF9 MOV EDI,ECX ;<== EDI = ECX
0042DB4B . 8D4D E8 LEA ECX,DWORD PTR SS:[EBP-18]
0042DB4E . FF15 E8114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeS>; MSVBVM60.__vbaFreeStr]
0042DB54 . 8B1D E4114000 MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaFr>; MSVBVM60.__vbaFreeObj]
0042DB5A . 8D4D E4 LEA ECX,DWORD PTR SS:[EBP-1C]
0042DB5D . FFD3 CALL EBX ;<&MSVBVM60.__vbaFreeObj>
0042DB5F . 66:85FF TEST DI,DI ;<== If EDI <> 0 then
0042DB62 . 0F85 BC010000 JNZ CMB_Audi.0042DD24 ;<== Jump out of check Serial
```

-**** Qua đoạn code trên bạn đã hiểu tại sao lại phải gõ FS ra rùi paste vô textbox rùi chử. Bởi vì nếu bạn nhập bằng tay vào thì chúng ta chỉ được một kí tự thôi và Olly sẽ Break ngay lập tức , do đó chúng ta sẽ không thể vượt qua được đoạn check length . He he nhưng có một mẹo nhỏ để các bạn có thể vượt qua được đoạn check length nếu chỉ nhập một kí tự , đó là khi đến lệnh **Test DI, DI** bạn hãy **Assemble (click Space)** tại câu lệnh nhảy **JNZ** ở phía dưới và sửa lại câu lệnh này. Oki

-**** Bây giờ sau khi chúng ta đã vượt qua được đoạn check Length trên , trace tiếp chúng ta sẽ tới đoạn code sau :

```
0042DBA3 >\8B45 E8 MOV EAX,DWORD PTR SS:[EBP-18] ;<== Fake Serial
0042DBA6 . 50 PUSH EAX
0042DBA7 . FF15 C4104000 CALL DWORD PTR DS:[<&MSVBVM60.#527>; MSVBVM60.rtcUpperCaseBstr]
0042DBAD . 8BD0 MOV EDX,EAX
0042DBAF . B9 B4F04200 MOV ECX,CMB_Audi.0042F0B4
0042DBB4 . FF15 BC114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrMo>; MSVBVM60.__vbaStrMove]
```

0042DBBA . 8D4D E8 LEA ECX,DWORD PTR SS:[EBP-18]
 0042DBBD . FF15 E8114000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeS>;
 MSVBVM60.__vbaFreeStr
 0042DBC3 . 8D4D E4 LEA ECX,DWORD PTR SS:[EBP-1C]
 0042DBC6 . FFD3 CALL EBX
 0042DBC8 . 8D4D D4 LEA ECX,DWORD PTR SS:[EBP-2C]
 0042DBC9 . 51 PUSH ECX
 0042DBCC . E8 5F1DFFFF CALL CMB_Audi.0041F930 ; <== Compare FS with default serial
 0042DBD1 . 8D55 D4 LEA EDX,DWORD PTR SS:[EBP-2C]
 0042DBD4 . 8D45 94 LEA EAX,DWORD PTR SS:[EBP-6C]
 0042DBD7 . 52 PUSH EDX
 0042DBD8 . 50 PUSH EAX
 0042DBD9 . C745 9C FFFFF>MOV DWORD PTR SS:[EBP-64],-1
 0042DBE0 . C745 94 0B800>MOV DWORD PTR SS:[EBP-6C],800B
 0042DBE7 . FF15 D8104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarTs>;
 MSVBVM60.__vbaVarTstEq
 0042DBED . 8D4D D4 LEA ECX,DWORD PTR SS:[EBP-2C]
 0042DBF0 . 66:8BF8 MOV DI,AX
 0042DBF3 . FF15 20104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeV>;
 MSVBVM60.__vbaFreeVar
0042DBF9 . 66:85FF TEST DI,DI
0042DBFC . 0F84 22010000 JE CMB_Audi.0042DD24
 0042DC02 . 8B0E MOV ECX,DWORD PTR DS:[ESI]
 0042DC04 . 56 PUSH ESI
 0042DC05 . FF91 8C030000 CALL DWORD PTR DS:[ECX+38C]
 0042DC0B . 8D55 E4 LEA EDX,DWORD PTR SS:[EBP-1C]
 0042DC0E . 50 PUSH EAX
 0042DC0F . 52 PUSH EDX
 0042DC10 . FF15 7C104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaObjSe>;
 MSVBVM60.__vbaObjSet
 0042DC16 . 8BF8 MOV EDI,EAX
 0042DC18 . 6A FF PUSH -1
 0042DC1A . 57 PUSH EDI
 0042DC1B . 8B07 MOV EAX,DWORD PTR DS:[EDI]
 0042DC1D . FF90 B4010000 CALL DWORD PTR DS:[EAX+1B4]
 0042DC23 . 85C0 TEST EAX,EAX
 0042DC25 . DBE2 FCLEX
 0042DC27 . 7D 12 JGE SHORT CMB_Audi.0042DC3B
 0042DC29 . 68 B4010000 PUSH 1B4
 0042DC2E . 68 40A44000 PUSH CMB_Audi.0040A440
 0042DC33 . 57 PUSH EDI
 0042DC34 . 50 PUSH EAX
 0042DC35 . FF15 60104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaHresu>;
 MSVBVM60.__vbaHRESULTCheckObj
 0042DC3B > 8D4D E4 LEA ECX,DWORD PTR SS:[EBP-1C]
 0042DC3E . FFD3 CALL EBX
 0042DC40 . 8B0E MOV ECX,DWORD PTR DS:[ESI]
 0042DC42 . 56 PUSH ESI
 0042DC43 . FF91 8C030000 CALL DWORD PTR DS:[ECX+38C]
 0042DC49 . 8D55 E4 LEA EDX,DWORD PTR SS:[EBP-1C]
 0042DC4C . 50 PUSH EAX

```

0042DC4D . 52      PUSH EDX
0042DC4E . FF15 7C104000 CALL DWORD PTR DS:[<&MSVBVM60.__vbaObjSe>;
MSVBVM60.__vbaObjSet
0042DC54 . 8BF0      MOV ESI,EAX
0042DC56 . 68 78BA4000 PUSH CMB_Audi.0040BA78 ; UNICODE "Thank you for registering!"
.....

```

-**** Oki , trong đoạn code trên các bạn để ý thấy có một câu lệnh TEST và một câu lệnh JE tại **0042DBF9** và **0042DBFC** . Các bạn sẽ thấy là nếu như sau một hồi kiểm tra ở trước hai câu lệnh này mà giá trị của **DI** là **0** thì câu lệnh nhảy sẽ cho chúng ta nhảy qua dòng "**Thank you for resgistering**" như các bạn đã thấy ở trên. Đây là điều mà chúng ta không hề mong muốn. Vậy thì chắc chắn sẽ phải có một hàm nào đó kiểm tra Serial của chúng ta nhập vào và kết quả đúng sai sẽ trả về cho DI.... Sau khi tìm hiểu thì đoạn check đó nằm tại địa chỉ sau **0042DBCC** :

```

0042DBCC . E8 5F1DFFFF CALL CMB_Audi.0041F930 ; <== Compare FS with default serial
(Trace Into)

```

-**** Đã xác định được mục tiêu chính chúng ta hãy **Trace Into** vào trong hàm này . Oki sau khi trace vào trong hàm này thì nhìn xuống một chút các bạn sẽ thấy các **Default Serial** của chương trình này . Chương trình sẽ nạp từng Default Serial này vào sau đó đem check với Fake Serial mà chúng ta gõ vào . Các Default Serial của chương trình này như sau:

```

0041FA19 . C785 D4FEFFFF>MOV DWORD PTR SS:[EBP-12C],CMB_Audi.0040>; UNICODE
"FGA485"
.....

```

```

0041FA4D . C785 B4FEFFFF>MOV DWORD PTR SS:[EBP-14C],CMB_Audi.0040>; UNICODE
"FHA486"
.....

```

```

0041FA81 . C785 94FEFFFF>MOV DWORD PTR SS:[EBP-16C],CMB_Audi.0040>; UNICODE
"FAH487"
.....

```

```

0041FAB5 . C785 74FEFFFF>MOV DWORD PTR SS:[EBP-18C],CMB_Audi.0040>; UNICODE
"FWH488"
.....

```

```

0041FADE . C785 54FEFFFF>MOV DWORD PTR SS:[EBP-1AC],CMB_Audi.0040>; UNICODE
"FGH489"
.....

```

-**** Keke , lấy giấy bút ra và ghi lại thôi còn chần chờ gì nữa . Qua đoạn trên các bạn có thể đặt ra câu hỏi là tại sao lại có thể biết được hàm đó là check Serial thì xin thưa với các bạn như sau. Sau khi chúng ta đã thu hẹp lại khoảng cần quan tâm là nằm phía trên của lệnh **TEST DI, DI**. Các bạn để ý thấy có rất nhiều hàm nằm trong cùng hàm Call khác nhau. Vậy thì tại sao và làm thế nào để biết chính xác đâu là hàm cần tìm ? Một câu trả lời ngắn gọn như sau, trong một chương trình nó sẽ có những hàm chuẩn và những hàm mà coder tự viết. Hàm chuẩn thường là các hàm gọi API hay là các hàm bắt đầu bằng tiếp đầu ngữ là **MSVB..** đối với Visual Basic , còn đối với Visual C++ là **MFC...** Còn hàm coder tự viết thì không có các tiếp đầu ngữ như vậy, do đó việc xác định đâu là mục tiêu cần tìm sẽ được tiếp tục thu hẹp và chúng ta có được kết quả như trên.

/*/*/* Serial tương ứng :

FGA485 Or FHA486 Or FAH487 Or FWH488 Or FGH489

III – KeyGen :

N/A

IV – End of Tut :

- Finished – **December 29, 2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.southbaypc.com/
Production :	South Bay Software
SoftWare :	Folder View v2.1
Copyright by :	Copyright (C) 2000 - 2004 South Bay Software . All Rights Reserved.
Type :	Name/Serial
Packed :	Not
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial/Keygen

Folder View v2.1

FolderView lets you view the contents of any folder on your computer, and then print it out or save it to a text file on disk. This is very convenient for times when you need to send someone the contents of a directory. You can simply attach the saved file to an e-mail and send it the other person. The directory contents can be sorted by name, file size, file type, or date modified. FolderView even gives you information on your MP3s, such as the artist, song title, length of the song, and more!

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng Microsoft Visual C++ 6.0 , không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là Enter Registration... .Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập Name & Serial vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ở đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (Name : kienmanowar & Code : 11111982). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ Sorry , you have entered an incorrect registration code ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Chúng ta sẽ tìm thấy tại địa chỉ sau :

0040C1C9 . 68 90B84200 PUSH FolderVi.0042B890 ;|Text = "Sorry, you have entered an incorrect registration code."

-**** Lần ngược lên trên chúng ta sẽ đặt Break Point tại đây :
 0040C148 . FFD7 CALL EDI ; \GetDlgItemTextA

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

```
0040C148 . FFD7      CALL EDI      ; \GetDlgItemTextA <==We're here (Get FU)
0040C14A . 8D5424 08  LEA EDX,DWORD PTR SS:[ESP+8]
0040C14E . 68 00010000 PUSH 100    ; /Count = 100 (256.)
0040C153 . 52      PUSH EDX      ; |Buffer
0040C154 . 68 ED030000 PUSH 3ED    ; |ControlID = 3ED (1005.)
0040C159 . 56      PUSH ESI      ; |hWnd
0040C15A . FFD7      CALL EDI      ; \GetDlgItemTextA <== Get FS
0040C15C . E8 AFF5FFFF CALL FolderVi.0040B710
0040C161 . 85C0      TEST EAX,EAX
0040C163 . 5F      POP EDI
0040C164 . 75 5C      JNZ SHORT FolderVi.0040C1C2
0040C166 . 8D4424 04  LEA EAX,DWORD PTR SS:[ESP+4]      ; FS
0040C16A . 8D8C24 040100>LEA ECX,DWORD PTR SS:[ESP+104]    ; FU
0040C171 . 50      PUSH EAX
0040C172 . 51      PUSH ECX
0040C173 . E8 D8040000 CALL FolderVi.0040C650      ; Encrypt <== Trace Into
```

----- Trace Into -----

```
0040C650 /$ 81EC 00010000 SUB ESP,100
0040C656 |. 56      PUSH ESI
0040C657 |. 8BB424 080100>MOV ESI,DWORD PTR SS:[ESP+108]
0040C65E |. 56      PUSH ESI
0040C65F |. E8 ACFEFFFF CALL FolderVi.0040C510
0040C664 |. 83C4 04  ADD ESP,4
0040C667 |. 85C0      TEST EAX,EAX
0040C669 |. 74 0A      JE SHORT FolderVi.0040C675
0040C66B |. 33C0      XOR EAX,EAX
0040C66D |. 5E      POP ESI
0040C66E |. 81C4 00010000 ADD ESP,100
0040C674 |. C3      RETN
0040C675 > 8D4424 04  LEA EAX,DWORD PTR SS:[ESP+4]
0040C679 |. 50      PUSH EAX
0040C67A |. 56      PUSH ESI
0040C67B |. E8 B0000000 CALL FolderVi.0040C730      ; Calculation <== Trace Into
```

----- Trace Into -----

===== NOTE =====

Trong phần Caculation này có tất cả 4 đoạn Code tính toán Right Code của chương trình này , thực chất chỉ là việc cộng dồn các FU[i] lại với nhau và cộng với một default value nằm trong thanh ghi EDX hoặc là sử dụng phép nhân FU[i] với default value , sau đó đem cộng lại . Mỗi đoạn sẽ tương ứng với từng SecX. (4 function thì sẽ có 4 SecX , mỗi SecX sẽ được nối với nhau thông qua “-“.

===== NOTE =====

```
0040C730 /$ 81EC 00010000 SUB ESP,100
0040C736 |. A0 C4DA4300 MOV AL,BYTE PTR DS:[43DAC4]
0040C73B |. 53      PUSH EBX
0040C73C |. 55      PUSH EBP
0040C73D |. 56      PUSH ESI
0040C73E |. 57      PUSH EDI
```

```

0040C73F |. 884424 10  MOV BYTE PTR SS:[ESP+10],AL
0040C743 |. B9 3F000000  MOV ECX,3F
0040C748 |. 33C0      XOR EAX,EAX
0040C74A |. 8D7C24 11  LEA EDI,DWORD PTR SS:[ESP+11]
0040C74E |. F3:AB      REP STOS DWORD PTR ES:[EDI]
0040C750 |. 66:AB      STOS WORD PTR ES:[EDI]
0040C752 |. AA        STOS BYTE PTR ES:[EDI]
0040C753 |. 8BBC24 140100>MOV EDI,DWORD PTR SS:[ESP+114] ;<== FU
0040C75A |. 57        PUSH EDI          ; /String
0040C75B |. FF15 AC614200 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ; \lstrlenA
<==Length(FU)
0040C761 |. 8BF0      MOV ESI,EAX ;<== Length (FU)
0040C763 |. 33C9      XOR ECX,ECX ;<== Sec1 = 0;
0040C765 |. 33C0      XOR EAX,EAX ;<== i=0;
0040C767 |. 85F6      TEST ESI,ESI
0040C769 |. 76 13      JBE SHORT FolderVi.0040C77E
0040C76B |. 8B15 54B54200 MOV EDX,DWORD PTR DS:[42B554] <=1st Default Value (0x32)
----- Calculation 0 -----

```

```

0040C771 |> 0FBE1C38  /MOVSX EBX,BYTE PTR DS:[EAX+EDI] ;<== FU[i]
0040C775 |. 03DA      |ADD EBX,EDX ;<== Temp = FU[i] + 1st Default Value (0x32)
0040C777 |. 03CB      |ADD ECX,EBX ;<== Sec1 = Sec1 + Temp
0040C779 |. 40        |INC EAX       ;<== i++
0040C77A |. 3BC6      |CMP EAX,ESI ;<== While (i < Length(FU))
0040C77C |.^ 72 F3    \JB SHORT FolderVi.0040C771 ;<== Continue Loop
0040C77E |> 8B9C24 180100>MOV EBX,DWORD PTR SS:[ESP+118]
0040C785 |. 51        PUSH ECX          ; /%u <== Sec1 (Convert hex --> dec)
0040C786 |. 68 10B94200 PUSH FolderVi.0042B910 ;|Format = "%u-" <== Sec1 + "-"
0040C78B |. 53        PUSH EBX          ;|s
0040C78C |. FF15 70624200 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA
<==Display Sec1
----- Calculation 0 -----

```

```

0040C792 |. 83C4 0C      ADD ESP,0C
0040C795 |. 33C9      XOR ECX,ECX ;<== Sec2 = 0;
0040C797 |. 33C0      XOR EAX,EAX ;<== i = 0;
0040C799 |. 85F6      TEST ESI,ESI
0040C79B |. 76 14      JBE SHORT FolderVi.0040C7B1
0040C79D |. 8B15 58B54200 MOV EDX,DWORD PTR DS:[42B558] <=2nd Default Value (0x28)
----- Calculation 1 -----

```

```

0040C7A3 |> 0FBE2C38  /MOVSX EBP,BYTE PTR DS:[EAX+EDI] ;<== FU[i]
0040C7A7 |. 0FAFEA    |IMUL EBP,EDX ;<== Temp = FU[i] * 2nd Default Value (0x28)
0040C7AA |. 03CD      |ADD ECX,EBP ;<== Sec2 = Sec2 + Temp
0040C7AC |. 40        |INC EAX       ;i++
0040C7AD |. 3BC6      |CMP EAX,ESI ; While ( i <Length (FU))
0040C7AF |.^ 72 F2    \JB SHORT FolderVi.0040C7A3 ;<== Continue Loop
0040C7B1 |> 51        PUSH ECX          ; /%u <== Sec2 (Convert hex -->dec)
0040C7B2 |. 8D4C24 14  LEA ECX,DWORD PTR SS:[ESP+14]          ;|
0040C7B6 |. 68 10B94200 PUSH FolderVi.0042B910      ;|Format = "%u-" <== Sec2 + "-"
0040C7BB |. 51        PUSH ECX          ;|s
0040C7BC |. FF15 70624200 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA
<==Display Sec2
----- Calculation 1 -----

```

```

0040C7C2 |. 83C4 0C    ADD ESP,0C
0040C7C5 |. 8D5424 10  LEA EDX,DWORD PTR SS:[ESP+10]
0040C7C9 |. 52        PUSH EDX          ; /StringToAdd
0040C7CA |. 53        PUSH EBX          ; |ConcatString
0040C7CB |. FF15 C4614200 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA <== Sec1 Ghép với Sec2
0040C7D1 |. 33C9      XOR ECX,ECX     ; <== Sec3 = 0 ;
0040C7D3 |. 33C0      XOR EAX,EAX     ; <== i = 0 ;
0040C7D5 |. 85F6      TEST ESI,ESI
0040C7D7 |. 76 13      JBE SHORT FolderVi.0040C7EC
0040C7D9 |. 8B15 5CB54200 MOV EDX,DWORD PTR DS:[42B55C]<=3rd Default Value(0x1E)

```

===== Calculation 2 =====

```

0040C7DF |> 0FBE2C38  /MOVSX EBP,BYTE PTR DS:[EAX+EDI] ; <== FU[i]
0040C7E3 |. 03EA      |ADD EBP,EDX   ; <== Temp = FU[i] + 3rd Default Value (0x1E)
0040C7E5 |. 03CD      |ADD ECX,EBP   ; <== Sec3 = Sec3 + Temp
0040C7E7 |. 40        |INC EAX       ; <== i++
0040C7E8 |. 3BC6      |CMP EAX,ESI    ; <== While (i < Length (FU))
0040C7EA |.^ 72 F3    |JB SHORT FolderVi.0040C7DF ; <== Continue Loop
0040C7EC |> 51        PUSH ECX       ; /%u          ; <== Sec3 (Convert hex --> dec)
0040C7ED |. 8D4424 14  LEA EAX,DWORD PTR SS:[ESP+14]   ; |
0040C7F1 |. 68 10B94200 PUSH FolderVi.0042B910      ; |Format = "%u-" <== Sec3 + "-"
0040C7F6 |. 50        PUSH EAX       ; |s
0040C7F7 |. FF15 70624200 CALL DWORD PTR DS:[<&USER32.wsprintfA>]      ; \wsprintfA <== Display Sec3

```

===== Calculation 2 =====

```

0040C7FD |. 83C4 0C    ADD ESP,0C
0040C800 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
0040C804 |. 51        PUSH ECX       ; /StringToAdd
0040C805 |. 53        PUSH EBX       ; |ConcatString
0040C806 |. FF15 C4614200 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>]      ; \lstrcmpA
0040C80C |. 33C9      XOR ECX,ECX     ; <== Sec4 = 0;
0040C80E |. 33C0      XOR EAX,EAX     ; <== i = 0;
0040C810 |. 85F6      TEST ESI,ESI
0040C812 |. 76 14      JBE SHORT FolderVi.0040C828
0040C814 |. 8B15 60B54200 MOV EDX,DWORD PTR DS:[42B560] <=4th Default Value (0x0B)

```

===== Calculation 3 =====

```

0040C81A |> 0FBE2C38  /MOVSX EBP,BYTE PTR DS:[EAX+EDI] ; <== FU[i]
0040C81E |. 0FAFEA    |IMUL EBP,EDX   ; Temp = FU[i] * 4th Default Value (0x0B)
0040C821 |. 03CD      |ADD ECX,EBP   ; Sec4 = Sec4 + Temp
0040C823 |. 40        |INC EAX       ; i ++
0040C824 |. 3BC6      |CMP EAX,ESI    ; While (i < Length (FU))
0040C826 |.^ 72 F2    |JB SHORT FolderVi.0040C81A ; Continue Loop
0040C828 |> 51        PUSH ECX       ; /%u          ; Sec4 (Convert hex --> dec)
0040C829 |. 8D5424 14  LEA EDX,DWORD PTR SS:[ESP+14]   ; |
0040C82D |. 68 749E4200 PUSH FolderVi.00429E74      ; |Format = "%u"
0040C832 |. 52        PUSH EDX       ; |s
0040C833 |. FF15 70624200 CALL DWORD PTR DS:[<&USER32.wsprintfA>]      ; \wsprintfA

```

===== Calculation 3 =====

```

0040C839 |. 83C4 0C    ADD ESP,0C
0040C83C |. 8D4424 10  LEA EAX,DWORD PTR SS:[ESP+10]
0040C840 |. 50        PUSH EAX       ; /StringToAdd

```

```

0040C841 |. 53      PUSH EBX          ;|ConcatString
0040C842 |. FF15 C4614200 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ;\lstrcatA <== EAX
: Right Code
0040C848 |. 5F      POP EDI
0040C849 |. 5E      POP ESI
0040C84A |. 5D      POP EBP
0040C84B |. 5B      POP EBX
0040C84C |. 81C4 00010000 ADD ESP,100
0040C852 \. C3      RETN

```

===== Trace Into =====

```

0040C680 |. 8B9424 140100>MOV EDX,DWORD PTR SS:[ESP+114] ;<== FS
0040C687 |. 8D4C24 0C   LEA ECX,DWORD PTR SS:[ESP+C]       ;<== Right Serial
0040C68B |. 51      PUSH ECX
0040C68C |. 52      PUSH EDX
0040C68D |. E8 2EFEEEEF CALL FolderVi.0040C4C0           ;<== Compare
0040C692 |. 83C4 10   ADD ESP,10
0040C695 |. F7D8     NEG EAX
0040C697 |. 1BC0     SBB EAX,EAX
0040C699 |. 5E      POP ESI
0040C69A |. F7D8     NEG EAX
0040C69C |. 81C4 00010000 ADD ESP,100
0040C6A2 \. C3      RETN

```

===== Trace Into =====

```

0040C178 . 83C4 08   ADD ESP,8
0040C17B . 85C0     TEST EAX,EAX ;<== Wrong Code Jump to Nag
0040C17D . 74 43    JE SHORT FolderVi.0040C1C2
0040C17F . 8D5424 04   LEA EDX,DWORD PTR SS:[ESP+4]
0040C183 . 8D8424 040100>LEA EAX,DWORD PTR SS:[ESP+104]
0040C18A . 52      PUSH EDX
0040C18B . 50      PUSH EAX
0040C18C . 68 18964200 PUSH FolderVi.00429618 ; ASCII "Software\FolderView\Registration"
0040C191 . 68 01000080 PUSH 80000001
0040C196 . E8 15050000 CALL FolderVi.0040C6B0
0040C19B . 68 18964200 PUSH FolderVi.00429618 ; ASCII "Software\FolderView\Registration"
0040C1A0 . 68 01000080 PUSH 80000001
0040C1A5 . E8 16020000 CALL FolderVi.0040C3C0
0040C1AA . 83C4 18   ADD ESP,18
0040C1AD . 6A 01     PUSH 1          ; /Result = 1
0040C1AF . 56      PUSH ESI        ; |hWnd
0040C1B0 . FF15 34624200 CALL DWORD PTR DS:[<&USER32.EndDialog>] ; \EndDialog
0040C1B6 . 33C0     XOR EAX,EAX
0040C1B8 . 5E      POP ESI
0040C1B9 . 81C4 00020000 ADD ESP,200
0040C1BF . C2 1000   RETN 10
0040C1C2 > 6A 00     PUSH 0          ; /Style = MB_OK|MB_APPLMODAL
0040C1C4 . 68 74964200 PUSH FolderVi.00429674           ; |Title = "FolderView"
0040C1C9 . 68 90B84200 PUSH FolderVi.0042B890 ;|Text = "Sorry, you have entered an incorrect
                                                registration code."
0040C1CE . 56      PUSH ESI        ; |hOwner
0040C1CF . FF15 78624200 CALL DWORD PTR DS:[<&USER32.MessageBoxA>] ;
                                                \MessageBoxA

```

```

0040C1D5 > 33C0      XOR EAX,EAX      ; Default case of switch 0040C0CD
0040C1D7 . 5E        POP ESI
0040C1D8 . 81C4 00020000 ADD ESP,200
0040C1DE . C2 1000   RETN 10

```

/*/*/* - SERIAL tương ứng :

User : kienmanowar	Serial : 1730-47200-1510-12980
User : REA-cRaCkErTeAm	Serial : 2005-50200-1705-13805

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;
int i=0,j=0;
int Temp=0, Temp1 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0, Sec4_temp = 0;

LenName=GetDlgItemText(IDC_NAME,reaName,64);
if (LenName < 1)
{
    MessageBox("----- Your name atleast 1 chart -----","Hey !! Please input
your name again !! ");
}
else
{
    // Calculation 0
    while (i < LenName)
    {
        Temp = reaName[i] + 0x32;
        Temp1 = Temp1 + Temp ;
        i++;
    }
    Sec1_temp = Temp1;
    wsprintf(Sec1,"%u-",Sec1_temp);

    i = 0;
    Temp = 0;
    Temp1 = 0;
    // Calculation 1
    while (i < LenName)
    {
        Temp = reaName[i] * 0x28;
        Temp1 = Temp1 + Temp;
        i++;
    }
    Sec2_temp = Temp1;
    wsprintf(Sec2,"%u-",Sec2_temp);
    //Calculation 2
    i = 0;
    Temp = 0 ;
}

```

```

Temp1 = 0;
while ( i < LenName)
{
    Temp = reaName[i] + 0x1E;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec3_temp = Temp1;
wsprintf(Sec3,"%u-",Sec3_temp);
//Calculation 3
i = 0;
Temp = 0;
Temp1 = 0;
while (i< LenName)
{
    Temp = reaName[i] * 0xB;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec4_temp = Temp1;
wsprintf(Sec4,"%u",Sec4_temp);
}
// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);
SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.southbaypc.com/
Production	:	South Bay Software
SoftWare	:	Hot Corners v2.21
Copyright by	:	Copyright (C) 1996 - 2004 South Bay Software . All Rights Reserved.
Type	:	Name/Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack	:	N/A
Request	:	Correct Serial/Keygen

Hot Corners v 2.21

Hot Corners lets you quickly activate or disable your screen saver by moving the mouse into any

corner of the screen that you set. This is particularly useful when you need to leave your computer for a brief period, but don't want anybody to be able to access it. Using Hot Corners, you can turn on your screen saver instantly, and know that your computer is protected while you are away. For people who work in an office environment, where the information on their screen is sensitive and possibly confidential, the ability to have instant privacy is crucial. Hot Corners is very customizable, and even allows you to control the power saving features of your monitor.

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng **Microsoft Visual C++ 6.0**, không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là **Enter Registration...** . Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập **Name & Serial** vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ở đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (**Name : kienmanowar & Code : 11111982**). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ **Sorry , you have entered an incorrect registration code** ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Chúng ta sẽ tìm thấy tại địa chỉ sau :

0040499C . 68 DC054100 PUSH HotC.004105DC ;|Text = "Sorry, you have entered an incorrect registration code."

-**** Lần ngược lên trên chúng ta sẽ đặt Break Point tại đây :

00404924 . FFD7 CALL EDI ;\GetDlgItemTextA <== We set BP here

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

00404924 . FFD7 CALL EDI	;	\GetDlgItemTextA <== We're here (Get FU)
00404926 . 8D5424 08 LEA EDX,DWORD PTR SS:[ESP+8]		
0040492A . 68 00010000 PUSH 100		; /Count = 100 (256.)
0040492F . 52 PUSH EDX		; Buffer
00404930 . 68 02040000 PUSH 402		; ControlID = 402 (1026.)
00404935 . 56 PUSH ESI		; hWnd
00404936 . FFD7 CALL EDI		; \GetDlgItemTextA ;<== Get FS
00404938 . 8D4424 08 LEA EAX,DWORD PTR SS:[ESP+8]		; <== FS
0040493C . 8D8C24 080100>LEA ECX,DWORD PTR SS:[ESP+108]		; <== FU
00404943 . 50 PUSH EAX		
00404944 . 51 PUSH ECX		
00404945 . E8 86040000 CALL HotC.00404DD0		; <== Encrypt (Trace Into)

----- Trace Into -----

00404DD0 /\$ 81EC 00010000 SUB ESP,100

```

00404DD6 |. 57      PUSH EDI
00404DD7 |. 8BBC24 080100>MOV EDI,DWORD PTR SS:[ESP+108]
00404DDE |. 57      PUSH EDI
00404DDF |. E8 3CF7FFFF CALL HotC.00404520
00404DE4 |. 83C4 04   ADD ESP,4
00404DE7 |. 85C0      TEST EAX,EAX
00404DE9 |. 74 0A     JE SHORT HotC.00404DF5
00404DEB |. 33C0     XOR EAX,EAX
00404DED |. 5F      POP EDI
00404DEE |. 81C4 00010000 ADD ESP,100
00404DF4 |. C3      RETN
00404DF5 > 8D4424 04   LEA EAX,DWORD PTR SS:[ESP+4]
00404DF9 |. 56      PUSH ESI
00404DFA |. 50      PUSH EAX
00404DFB |. 57      PUSH EDI
00404DFC |. 33F6     XOR ESI,ESI

```

00404DFE |. E8 AD000000 CALL HotC.00404EB0 ;<== Calculation (Trace Into)

===== Trace Into =====

```

00404EB0 /$ 81EC 00010000 SUB ESP,100
00404EB6 |. A0 A0374100 MOV AL,BYTE PTR DS:[4137A0]
00404EBB |. 53      PUSH EBX
00404EBC |. 55      PUSH EBP
00404EBD |. 56      PUSH ESI
00404EBE |. 57      PUSH EDI
00404EBF |. 884424 10   MOV BYTE PTR SS:[ESP+10],AL
00404EC3 |. B9 3F000000 MOV ECX,3F
00404EC8 |. 33C0     XOR EAX,EAX
00404ECA |. 8D7C24 11   LEA EDI,DWORD PTR SS:[ESP+11]
00404ECE |. 8BAC24 180100>MOV EBP,DWORD PTR SS:[ESP+118]
00404ED5 |. F3:AB    REP STOS DWORD PTR ES:[EDI]
00404ED7 |. 66:AB    STOS WORD PTR ES:[EDI]
00404ED9 |. AA      STOS BYTE PTR ES:[EDI]
00404EDA |. 8B9C24 140100>MOV EBX,DWORD PTR SS:[ESP+114]
00404EE1 |. B9 40000000 MOV ECX,40
00404EE6 |. 33C0     XOR EAX,EAX
00404EE8 |. 8BF0    MOV EDI,EBP
00404EEA |. F3:AB    REP STOS DWORD PTR ES:[EDI]
00404EEC |. 8BFB    MOV EDI,EBX
00404EEE |. 83C9 FF  OR ECX,FFFFFF
00404EF1 |. F2:AE    REPNE SCAS BYTE PTR ES:[EDI]
00404EF3 |. F7D1    NOT ECX
00404EF5 |. 49      DEC ECX      ;<== Length(FU)
00404EF6 |. B8 6A000000 MOV EAX,6A ;<== Temp_01
00404EFB |. 8BF1    MOV ESI,ECX ;<== Length(FU)
00404EFD |. 33C9    XOR ECX,ECX ;<== i = 0;
00404EFF |. 85F6    TEST ESI,ESI
00404F01 |. 7E 0C    JLE SHORT HotC.00404F0F

```

===== Calculation 0 =====

```

00404F03 > 0FBE1419  /MOVSX EDX,BYTE PTR DS:[ECX+EBX] ;<== FU[i]
00404F07 |. 41      |INC ECX      ;<== i ++
00404F08 |. 3BCE    |CMP ECX,ESI ;<== While ( i < Length(FU))

```

00404F0A |. 8D0450 |LEA EAX,DWORD PTR DS:[EAX+EDX*2] ;<== Temp_01 = Temp_01 + FU[i]*0x2

00404F0D |.^ 7C F4 |JL SHORT HotC.00404F03 ;<== Continue Loop

00404F0F |> 8B3D 70E24000 MOV EDI,DWORD PTR DS:[<&USER32.wsprintfA>; USER32.wsprintfA

00404F15 |. 50 PUSH EAX ; /<%d> <== Sec1

00404F16 |. 8D4424 14 LEA EAX,DWORD PTR SS:[ESP+14] ;|

00404F1A |. 68 28064100 PUSH HotC.00410628 ; |Format = "%d-" ;<== Sec1 + "-"

00404F1F |. 50 PUSH EAX ;|s

00404F20 |. FFD7 CALL EDI ; \wsprintfA <== Display Sec1

===== Calculation 0 =====

00404F22 |. 83C4 0C ADD ESP,0C

00404F25 |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]

00404F29 |. 51 PUSH ECX ; /StringToAdd

00404F2A |. 55 PUSH EBP ; |ConcatString

00404F2B |. FF15 68E14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA

00404F31 |. 33C0 XOR EAX,EAX ;<== i = 0

00404F33 |. B9 6A000000 MOV ECX,6A ;<== Temp_02

00404F38 |. 85F6 TEST ESI,ESI

00404F3A |. 7E 0F JLE SHORT HotC.00404F4B

===== Calculation 1 =====

00404F3C |> 0FBE1418 |MOVSX EDX,BYTE PTR DS:[EAX+EBX] ;<== FU[i]

00404F40 |. 40 INC EAX ;<== i ++

00404F41 |. 8D1492 |LEA EDX,DWORD PTR DS:[EDX+EDX*4] ; Temp = FU[i] +FU[i] *0x4

00404F44 |. 3BC6 |CMP EAX,ESI ;<== while (i <Length(FU))

00404F46 |. 8D0C91 |LEA ECX,DWORD PTR DS:[ECX+EDX*4] ; Temp_02 = Temp_02 + Temp*0x4

00404F49 |.^ 7C F1 |JL SHORT HotC.00404F3C ;<== Continue Loop

00404F4B |> 51 PUSH ECX ;<== Sec2

00404F4C |. 8D4424 14 LEA EAX,DWORD PTR SS:[ESP+14]

00404F50 |. 68 28064100 PUSH HotC.00410628 ; ASCII "%d-

00404F55 |. 50 PUSH EAX

00404F56 |. FFD7 CALL EDI <== Display Sec2

===== Calculation 1 =====

00404F58 |. 83C4 0C ADD ESP,0C

00404F5B |. 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]

00404F5F |. 51 PUSH ECX ; /StringToAdd

00404F60 |. 55 PUSH EBP ; |ConcatString

00404F61 |. FF15 68E14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA

===== Calculation 2 =====

00404F67 |. 0FBE441E FF MOVSX EAX,BYTE PTR DS:[ESI+EBX-1] ;<== Temp = FU[Length(FU) - 1]

00404F6C |. 8D1480 LEA EDX,DWORD PTR DS:[EAX+EAX*4] ; Temp1 = Temp + Temp*0x4

00404F6F |. 8D0450 LEA EAX,DWORD PTR DS:[EAX+EDX*2] ; Temp = Temp + Temp1*0x2

00404F72 |. 8D5424 10 LEA EDX,DWORD PTR SS:[ESP+10]

00404F76 |. 8D4C00 01 LEA ECX,DWORD PTR DS:[EAX+EAX+1] ; Sec3 = Temp + Temp + 1

00404F7A |. 51 PUSH ECX <== Sec3

00404F7B |. 68 28064100 PUSH HotC.00410628 ; ASCII "%d-

00404F80 |. 52 PUSH EDX

00404F81 |. FFD7 CALL EDI <== Display Sec3

===== Calculation 2 =====

```

00404F83 |. 83C4 0C    ADD ESP,0C
00404F86 |. 8D4424 10  LEA EAX,DWORD PTR SS:[ESP+10]
00404F8A |. 50        PUSH EAX          ; /StringToAdd
00404F8B |. 55        PUSH EBP          ; |ConcatString
00404F8C |. FF15 68E14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA

```

===== Calculation 3 =====

```

00404F92 |. 0FBE4C1E FF  MOVSX ECX,BYTE PTR DS:[ESI+EBX-1] ;<== Temp =
FU[Length(FU) - 1]
00404F97 |. 8D4424 10  LEA EAX,DWORD PTR SS:[ESP+10]
00404F9B |. 8D148D 1D0000>LEA EDX,DWORD PTR DS:[ECX*4+1D] ;<== Sec4 = Temp*0x4 +
1D
00404FA2 |. 52        PUSH EDX <== Sec4
00404FA3 |. 68 24064100 PUSH HotC.00410624      ; ASCII "%d"
00404FA8 |. 50        PUSH EAX
00404FA9 |. FFD7      CALL EDI <== Display Sec4

```

===== Calculation 3 =====

```

00404FAB |. 83C4 0C    ADD ESP,0C
00404FAE |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
00404FB2 |. 51        PUSH ECX          ; /StringToAdd
00404FB3 |. 55        PUSH EBP          ; |ConcatString
00404FB4 |. FF15 68E14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA <== Right
Code after Calculation

```

```

00404FBA |. 5F        POP EDI
00404FBB |. 5E        POP ESI
00404FBC |. 5D        POP EBP
00404FBD |. 5B        POP EBX
00404FBE |. 81C4 00010000 ADD ESP,100
00404FC4 \. C3       RETN

```

===== Trace Into =====

```

00404E03 |. 8B9424 180100>MOV EDX,DWORD PTR SS:[ESP+118]      ;<==FS
00404E0A |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]           ;<== Right Code
00404E0E |. 51        PUSH ECX
00404E0F |. 52        PUSH EDX
00404E10 |. E8 6BFFFFFF CALL HotC.00404D80                  ;<== Compare
00404E15 |. 83C4 10    ADD ESP,10
00404E18 |. 85C0      TEST EAX,EAX
00404E1A |. 74 05     JE SHORT HotC.00404E21
00404E1C |. BE 01000000 MOV ESI,1
00404E21 |> 8BC6      MOV EAX,ESI
00404E23 |. 5E        POP ESI
00404E24 |. 5F        POP EDI
00404E25 |. 81C4 00010000 ADD ESP,100
00404E2B \. C3       RETN

```

===== Trace Into =====

```

0040494A . 83C4 08    ADD ESP,8
0040494D . 85C0      TEST EAX,EAX          ; <== Wrong Code
0040494F . 5F        POP EDI
00404950 . 74 43     JE SHORT HotC.00404995 ; <== Jump to Nag
00404952 . 8D5424 04  LEA EDX,DWORD PTR SS:[ESP+4]
00404956 . 8D8424 040100>LEA EAX,DWORD PTR SS:[ESP+104]
0040495D . 52        PUSH EDX

```

```

0040495E . 50      PUSH EAX
0040495F . 68 80004100 PUSH HotC.00410080 ; ASCII "Software\Hot Corners\Configuration"
00404964 . 68 01000080 PUSH 80000001
00404969 . E8 C2040000 CALL HotC.00404E30
0040496E . 68 80004100 PUSH HotC.00410080 ; ASCII "Software\Hot Corners\Configuration"
00404973 . 68 01000080 PUSH 80000001
00404978 . E8 F3020000 CALL HotC.00404C70
0040497D . 83C4 18 ADD ESP,18
00404980 . 6A 01      PUSH 1           ; /Result = 1
00404982 . 56      PUSH ESI           ; |hWnd
00404983 . FF15 04E24000 CALL DWORD PTR DS:[<&USER32.EndDialog>] ; \EndDialog
00404989 . 33C0      XOR EAX,EAX
0040498B . 5E      POP ESI
0040498C . 81C4 00020000 ADD ESP,200
00404992 . C2 1000    RETN 10
00404995 > 6A 00      PUSH 0           ; /Style = MB_OK|MB_APPLMODAL
00404997 . 68 AC004100 PUSH HotC.004100AC ; |Title = "Hot Corners"
0040499C . 68 DC054100 PUSH HotC.004105DC ; |Text = "Sorry, you have entered an incorrect
registration code."
004049A1 . 56      PUSH ESI           ; |hOwner
004049A2 . FF15 20E24000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>];
|MessageBoxA <== Shot Nag if Wrong Serial
004049A8 > 33C0      XOR EAX,EAX           ; Default case of switch 004048DD
004049AA . 5E      POP ESI
004049AB . 81C4 00020000 ADD ESP,200
004049B1 . C2 1000    RETN 10

```

/*/*/* - SERIAL tương ứng :

User : kienmanowar	Serial : 2466-23706-2509-485
User : REA-cRaCkErTeAm	Serial : 2616-25206-2399-465

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;
int i=0,j=0;
int Temp=0, Temp1 = 0, Temp2 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0, Sec4_temp = 0;

LenName=GetDlgItemText(IDC_NAME,reaName,64);
if (LenName < 1)
{
    MessageBox("===== Your name atleast 1 chart ===== ", "Hey !! Please input
your name again !! ");
}
else
{
    Temp = 0x6A;
    // Caculation 0
    while (i < LenName)
    {

```

```

Temp = Temp + reaName[i] * 0x2;
i++;
}
Sec1_temp = Temp;
wsprintf(Sec1,"%u-",Sec1_temp);

i = 0;
Temp1 = 0x6A;
Temp = 0;
// Calculation 1
while (i < LenName)
{
    Temp = reaName[i] + reaName[i] * 0x4;
    Temp1 = Temp1 + Temp * 0x4;
    i++;
}
Sec2_temp = Temp1;
wsprintf(Sec2,"%u-",Sec2_temp);

//Calculation 2
i = LenName - 1 ;
Temp = 0;
Temp1 = 0;
Temp = reaName[i];
Temp1 = Temp + Temp * 0x4;
Temp = Temp + Temp1*0x2;
Temp2 = Temp + Temp + 1;
Sec3_temp = Temp2;
wsprintf(Sec3,"%u-",Sec3_temp);

//Calculation 3
i = LenName - 1;
Temp = 0;
Temp = reaName[i] * 0x4 + 0x1D;
Sec4_temp = Temp;
wsprintf(Sec4,"%u",Sec4_temp);
}

// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);
SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *September 13, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://embsoftware.com
Production :	CMB Software
SoftWare :	Password Pro v2.01
Copyright by :	Copyright (C) 2001 CMB Software . All Rights Reserved.
Type :	Serial
Packed :	Not
Language :	Microsoft Visual Basic 5.0/6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial

Password Pro v2.01

Password Pro is a professional business tool for keeping track of User Id's and Passwords. Password Pro allows multiple users to maintain their own password lists. Password Pro has the following features:

- Very easy to use intuitive user interface.
- Unlimited number of password files can be created.
- Each password file can hold 500 different passwords.
- Each record can point to a "Link" which can be launched by Password Pro.
- Each password file can be optionally "Password Protected".
- Random Password Generator

I – Information :

-**** Dùng PeiD biết được chương trình được code bằng **Microsoft Visual Basic 5.0 / 6.0** và không sử dụng Crypto nào.

-**** Run thử chương trình , nhập **Fake Serial** , nhấn Enter chúng ta không nhận được cái Nag thông báo nào.

-****Load chương trình này vào trong Olly , dùng chức năng **Search String** để tìm kiếm . Chúng ta tìm được một String rất đáng giá sau :
00420AB5 . C745 84 E0B04>MOV DWORD PTR SS:[EBP-7C],Passpro.0040B0>; UNICODE "Registered!"

-**** Double Click vào dòng này, chúng ta quay trở lại chương trình chính . Việc tìm kiếm vị trí Set BP tương tự như đã phân tích ở soft **CMB Audio Player**. Chúng ta Set BP tại đây :
00420903 . E8 3810FEFF CALL <JMP.&MSVBVM50.__vbaLenBstr> ;<== Set BP here

II – Cracking :

-**** Sau khi Set BP tại hàm trên , nhấn **F9** để run chương trình . Nhập **Fake Serial** vào, chúng ta sẽ Break tại hàm trên :

```
00420903 . E8 3810FEFF CALL <JMP.&MSVBVM50.__vbaLenBstr> ;<== We're here
00420908 . 33C9      XOR ECX,ECX           ; ntdll.77F5168D
```

```

0042090A . 83F8 06    CMP EAX,6          ;<== Length (FS) = 6
0042090D . 0F95C1    SETNE CL
00420910 . F7D9      NEG ECX
00420912 . 66:898D 50FFF>MOV WORD PTR SS:[EBP-B0],CX
00420919 . 8D4D E8    LEA ECX,DWORD PTR SS:[EBP-18]
0042091C . E8 230FFEFFF CALL <JMP.&MSVBVM50.__vbaFreeStr>
00420921 . 8D4D DC    LEA ECX,DWORD PTR SS:[EBP-24]
00420924 . E8 E70FFEFFF CALL <JMP.&MSVBVM50.__vbaFreeObj>
00420929 . 0FBF85 50FFFF>MOVSX EAX,WORD PTR SS:[EBP-B0]
00420930 . 85C0      TEST EAX,EAX
00420932 . 74 05     JE SHORT Passpro.00420939 ;<== Jmp out of check serial if Length(FS) <> 6

```

-**** Oki , qua đoạn trên các bạn thấy là chiều dài của FS phải là 6. Tiếp tục trace tiếp chúng ta sẽ đến đoạn code sau :

```

004209A8 > \FF75 E8    PUSH DWORD PTR SS:[EBP-18]   ;<== FS
004209AB . E8 3A0EFEFFF CALL <JMP.&MSVBVM50.#527>
004209B0 . 8BD0      MOV EDX,EAX          ; <== FS
004209B2 . 8D4D E4    LEA ECX,DWORD PTR SS:[EBP-1C]
004209B5 . E8 320FFEFFF CALL <JMP.&MSVBVM50.__vbaStrMove>
004209BA . 50        PUSH EAX           ;<== FS
004209BB . FF35 38314200 PUSH DWORD PTR DS:[423138] ;<== Default Serial
004209C1 . E8 240EFEFFF CALL <JMP.&MSVBVM50.#527>
004209C6 . 8BD0      MOV EDX,EAX
004209C8 . 8D4D E0    LEA ECX,DWORD PTR SS:[EBP-20]
004209CB . E8 1C0FFEFFF CALL <JMP.&MSVBVM50.__vbaStrMove>
004209D0 . 50        PUSH EAX
004209D1 . E8 FE0EFEFFF CALL <JMP.&MSVBVM50.__vbaStrCmp>;<== Compare FS with
Default Serial
004209D6 . F7D8      NEG EAX
004209D8 . 1BC0      SBB EAX,EAX
004209DA . 40        INC EAX
004209DB . F7D8      NEG EAX
004209DD . 66:8985 50FFF>MOV WORD PTR SS:[EBP-B0],AX
004209E4 . 8D45 E0    LEA EAX,DWORD PTR SS:[EBP-20]
004209E7 . 50        PUSH EAX
004209E8 . 8D45 E4    LEA EAX,DWORD PTR SS:[EBP-1C]
004209EB . 50        PUSH EAX
004209EC . 8D45 E8    LEA EAX,DWORD PTR SS:[EBP-18]
004209EF . 50        PUSH EAX
004209F0 . 6A 03     PUSH 3
004209F2 . E8 E90EFEFFF CALL <JMP.&MSVBVM50.__vbaFreeStrList>
004209F7 . 83C4 10    ADD ESP,10
004209FA . 8D4D DC    LEA ECX,DWORD PTR SS:[EBP-24]
004209FD . E8 0E0FFEFFF CALL <JMP.&MSVBVM50.__vbaFreeObj>
00420A02 . 0FBF85 50FFFF>MOVSX EAX,WORD PTR SS:[EBP-B0]
00420A09 . 85C0      TEST EAX,EAX
00420A0B . 0F84 A1010000 JE Passpro.00420BB2 ;<== If FS <> Default Serial then Jmp out

```

-**** Như các bạn đã thấy ở trên chương trình không sử dụng một thủ tục nào để mã hóa FS cả mà sẽ lấy một (chương trình này chỉ có đúng 1) **Default Serial** ra để so sánh với FS mà chúng ta nhập vào . Nếu trùng thì Oki , còn không thì nó cho chúng ta đi chơi lồng vòng :)

-**** Công việc còn lại bây giờ là , lấy giấy bút ra ghi lại cái **Default Serial** mà chương trình sử dụng và nhập vào . Ke ke Done..... nó cảm ơn chúng ta kia (**"Thank You For Registering Password Pro!"**)

/*/*/*/* Serial :

PW3AXD

III – KeyGen :

N/A

IV – End of Tut :

- Finished – **December 30, 2004**

Reverse Engineering Association

SoftWare

Homepage	:	http://www.southbaypc.com/
Production	:	South Bay Software
SoftWare	:	Printer Express v1.25
Copyright by	:	Copyright (C) 2001 - 2002 South Bay Software . All Rights Reserved.
Type	:	Name/Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack	:	N/A
Request	:	Correct Serial/Keygen

Printer Express v1.25

PrinterExpress is a very useful utility for people who work with many different printers on a network. The inconvenience of having to use the Print dialog every time you want to print can quickly become a hassle. Using PrinterExpress, you only have to click once on the PrinterExpress icon located in the system tray (next to the time), and you can easily make any printer on your network your default printer. So when you need to print a document, or many documents, you can simply click on the Print icon in your favorite program instead of going through the tedious process of *File->Print...->(select printer) ->OK*. The program will easily pay for itself, just by the amount of time you will save in your work day

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng **Microsoft Visual C++ 6.0** , không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là **Enter Registration...** . Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập **Name & Serial** vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ô

đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (**Name : kienmanowar & Code : 11111982**). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ **Incorrect registration code!** ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Chúng ta sẽ tìm thấy tại địa chỉ sau :

004036A8 . 68 5CD24000 PUSH PrinterE.0040D25C ;|Text = "Incorrect registration code!"

-**** Lần ngược lên trên chúng ta sẽ đặt Break Point tại đây :

00403623 . FFD7 CALL EDI ;\GetDlgItemTextA <== We Set BP here

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

00403623 . FFD7 CALL EDI ;\GetDlgItemTextA <== We're here (Get FU)

00403625 . 8D5424 08 LEA EDX,DWORD PTR SS:[ESP+8]

00403629 . 68 00010000 PUSH 100 ;/Count = 100 (256.)

0040362E . 52 PUSH EDX ;|Buffer

0040362F . 68 EE030000 PUSH 3EE ;|ControlID = 3EE (1006.)

00403634 . 56 PUSH ESI ;|hWnd

00403635 . FFD7 CALL EDI ;\GetDlgItemTextA ;<== Get FS

00403637 . 8D4424 08 LEA EAX,DWORD PTR SS:[ESP+8] ;<== FS

0040363B . 8D8C24 080100>LEA ECX,DWORD PTR SS:[ESP+108] ;<== FU

00403642 . 50 PUSH EAX

00403643 . 51 PUSH ECX

00403644 . E8 17030000 CALL PrinterE.00403960 ;<== Encrypt(Trace Into)

----- Trace Into -----

00403960 /\$ 81EC 00010000 SUB ESP,100

00403966 |. A0 DCFF4000 MOV AL,BYTE PTR DS:[40FFDC]

0040396B |. 56 PUSH ESI

0040396C |. 57 PUSH EDI

0040396D |. 884424 08 MOV BYTE PTR SS:[ESP+8],AL

00403971 |. B9 3F000000 MOV ECX,3F

00403976 |. 33C0 XOR EAX,EAX

00403978 |. 8D7C24 09 LEA EDI,DWORD PTR SS:[ESP+9]

0040397C |. 8B9424 0C0100>MOV EDX,DWORD PTR SS:[ESP+10C]

00403983 |. F3:AB REP STOS DWORD PTR ES:[EDI]

00403985 |. 66:AB STOS WORD PTR ES:[EDI]

00403987 |. 8D4C24 08 LEA ECX,DWORD PTR SS:[ESP+8]

0040398B |. 51 PUSH ECX

0040398C |. 52 PUSH EDX

0040398D |. AA STOS BYTE PTR ES:[EDI]

0040398E |. E8 3D010000 CALL PrinterE.00403AD0 ;<== Calculation (Trace into)

----- Trace Into -----

00403AD0 /\$ 81EC 00010000 SUB ESP,100

00403AD6 |. A0 DCFF4000 MOV AL,BYTE PTR DS:[40FFDC]

00403ADB |. 53 PUSH EBX

00403ADC |. 55 PUSH EBP

00403ADD |. 56 PUSH ESI

00403ADE |. 57 PUSH EDI

00403ADF |. 884424 10 MOV BYTE PTR SS:[ESP+10],AL
 00403AE3 |. B9 3F000000 MOV ECX,3F
 00403AE8 |. 33C0 XOR EAX,EAX
 00403AEA |. 8D7C24 11 LEA EDI,DWORD PTR SS:[ESP+11]
 00403AEE |. F3:AB REP STOS DWORD PTR ES:[EDI]
 00403AF0 |. 66:AB STOS WORD PTR ES:[EDI]
 00403AF2 |. AA STOS BYTE PTR ES:[EDI]
 00403AF3 |. 8BBC24 140100>MOV EDI,DWORD PTR SS:[ESP+114] ;<== FU
 00403AFA |. 57 PUSH EDI ; /String
 00403AFB |. FF15 0CB14000 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ; \lstrlenA <== Get Length (FU)
 00403B01 |. 8BF0 MOV ESI,EAX ;<== Length(FU);
 00403B03 |. 33C9 XOR ECX,ECX ;<== Sec1 = 0;
 00403B05 |. 33C0 XOR EAX,EAX ;<== i = 0;
 00403B07 |. 85F6 TEST ESI,ESI
 00403B09 |. 7E 13 JLE SHORT PrinterE.00403B1E
00403B0B |. 8B15 4CD24000 MOV EDX,DWORD PTR DS:[40D24C] <=1st Default Value (0x18)

===== Calculation 0 =====

00403B11 |> 0FBEB1C38 /MOVSX EBX,BYTE PTR DS:[EAX+EDI] ;<== FU[i];
 00403B15 |. 03DA |ADD EBX,EDX ;<== Temp = FU[i] + **1st Default Value (0x18)**
 00403B17 |. 03CB |ADD ECX,EBX ;<== Sec1 = Sec1 + Temp
 00403B19 |. 40 |INC EAX ;<== i++
 00403B1A |. 3BC6 |CMP EAX,ESI ;<== While (i < Length(FU))
 00403B1C |.^ 7C F3 \JL SHORT PrinterE.00403B11 ;<== Continue Loop
 00403B1E |> 8B9C24 180100>MOV EBX,DWORD PTR SS:[ESP+118]
 00403B25 |. 51 PUSH ECX ; /%ld ;<== Sec1(hex-->dec)
 00403B26 |. 68 90D24000 PUSH PrinterE.0040D290; |Format = "%ld-";<== Sec1 +"-"
 00403B2B |. 53 PUSH EBX ;|s
 00403B2C |. FF15 88B14000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA <== Display Sec1

===== Calculation 0 =====

00403B32 |. 83C4 0C ADD ESP,0C
 00403B35 |. 33C9 XOR ECX,ECX ;<== Sec2 = 0;
 00403B37 |. 33C0 XOR EAX,EAX ;<== i = 0;
 00403B39 |. 85F6 TEST ESI,ESI
 00403B3B |. 7E 14 JLE SHORT PrinterE.00403B51
00403B3D |. 8B15 50D24000 MOV EDX,DWORD PTR DS:[40D250]<=2nd Default Value (0x27)

===== Calculation 1 =====

00403B43 |> 0FBEB2C38 /MOVSX EBP,BYTE PTR DS:[EAX+EDI] ;<== FU[i];
 00403B47 |. 0FAFEA |IMUL EBP,EDX ;<== Temp = FU[i] * **2nd Default Value (0x27)**
 00403B4A |. 03CD |ADD ECX,EBP ;<== Sec2 = Sec2 + Temp
 00403B4C |. 40 |INC EAX ;<== i ++
 00403B4D |. 3BC6 |CMP EAX,ESI ;<== While (i < Length(FU))
 00403B4F |.^ 7C F2 \JL SHORT PrinterE.00403B43 ;<== Continue Loop
 00403B51 |> 51 PUSH ECX ; /%ld ;<== Sec2
 00403B52 |. 8D4C24 14 LEA ECX,DWORD PTR SS:[ESP+14] ;|
 00403B56 |. 68 90D24000 PUSH PrinterE.0040D290 ; |Format = "%ld-"; Sec2 + "-"
 00403B5B |. 51 PUSH ECX ;|s
 00403B5C |. FF15 88B14000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

===== Calculation 1 =====

00403B62 |. 83C4 0C ADD ESP,0C
 00403B65 |. 8D5424 10 LEA EDX,DWORD PTR SS:[ESP+10]

```

00403B69 |. 52      PUSH EDX          ; /StringToAdd
00403B6A |. 53      PUSH EBX          ; |ConcatString
00403B6B |. FF15 FCB04000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA
00403B71 |. 33C9    XOR ECX,ECX       ;<== Sec3 = 0;
00403B73 |. 33C0    XOR EAX,EAX       ;<== i = 0;
00403B75 |. 85F6    TEST ESI,ESI
00403B77 |. 7E 13    JLE SHORT PrinterE.00403B8C
00403B79 |. 8B15 54D24000 MOV EDX,DWORD PTR DS:[40D254]<=3rd Default Value(0x1A)

```

===== Calculation 2 =====

```

00403B7F |> 0FBE2C38  /MOVsx EBP,BYTE PTR DS:[EAX+EDI] ;<== FU[i];
00403B83 |. 03EA    |ADD EBP,EDX        ;<== Temp = FU[i] + 3rd Default Value(0x1A)
00403B85 |. 03CD    |ADD ECX,EBP        ;<== Sec3 = Sec3 + Temp
00403B87 |. 40      |INC EAX           ;<== i ++
00403B88 |. 3BC6    |CMP EAX,ESI         ;<== While (i < Length(FU))
00403B8A |.^ 7C F3    \JL SHORT PrinterE.00403B7F ;<== Continue Loop
00403B8C |> 51      PUSH ECX          ; /%ld> ;<== Sec3
00403B8D |. 8D4424 14  LEA EAX,DWORD PTR SS:[ESP+14] ; |
00403B91 |. 68 90D24000 PUSH PrinterE.0040D290 ; |Format = "%ld-"
00403B96 |. 50      PUSH EAX          ; |s
00403B97 |. FF15 88B14000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

===== Calculation 2 =====

```

00403B9D |. 83C4 0C    ADD ESP,0C
00403BA0 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
00403BA4 |. 51      PUSH ECX          ; /StringToAdd
00403BA5 |. 53      PUSH EBX          ; |ConcatString
00403BA6 |. FF15 FCB04000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA
00403BAC |. 33C9    XOR ECX,ECX       ;<== Sec4 = 0
00403BAE |. 33C0    XOR EAX,EAX       ;<== i = 0
00403BB0 |. 85F6    TEST ESI,ESI
00403BB2 |. 7E 14    JLE SHORT PrinterE.00403BC8
00403BB4 |. 8B15 58D24000 MOV EDX,DWORD PTR DS:[40D258]<=4th Default Value (0x01)

```

===== Calculation 3 =====

```

00403BBA |> 0FBE2C38  /MOVsx EBP,BYTE PTR DS:[EAX+EDI] ;<== FU[i]
00403BBE |. 0FAFEA  |IMUL EBP,EDX        ;<== Temp = FU[i] *4th Default Value (0x01)
00403BC1 |. 03CD    |ADD ECX,EBP        ;<== Sec4 = Sec4 + Temp
00403BC3 |. 40      |INC EAX           ;<== i ++
00403BC4 |. 3BC6    |CMP EAX,ESI         ;<== While (i < Length(FU))
00403BC6 |.^ 7C F2    \JL SHORT PrinterE.00403BBA ;<== Continue Loop
00403BC8 |> 51      PUSH ECX          ; /%ld> ;<== Sec4
00403BC9 |. 8D5424 14  LEA EDX,DWORD PTR SS:[ESP+14] ; |
00403BCD |. 68 8CD24000 PUSH PrinterE.0040D28C ; |Format = "%ld"
00403BD2 |. 52      PUSH EDX          ; |s
00403BD3 |. FF15 88B14000 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA

```

===== Calculation 3 =====

```

00403BD9 |. 83C4 0C    ADD ESP,0C
00403BDC |. 8D4424 10  LEA EAX,DWORD PTR SS:[ESP+10]
00403BE0 |. 50      PUSH EAX          ; /StringToAdd
00403BE1 |. 53      PUSH EBX          ; |ConcatString
00403BE2 |. FF15 FCB04000 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA <==
Right Code after Calculation
00403BE8 |. 5F      POP EDI
00403BE9 |. 5E      POP ESI

```

```

00403BEA |. 5D      POP EBP
00403BEB |. 5B      POP EBX
00403BEC |. 81C4 00010000 ADD ESP,100
00403BF2 \. C3     RETN

```

===== Trace Into =====

```

00403993 |. 8BB424 180100>MOV ESI,DWORD PTR SS:[ESP+118] ;<== FS
0040399A |. 56      PUSH ESI
0040399B |. E8 90000000 CALL PrinterE.00403A30
004039A0 |. 8D4424 14  LEA EAX,DWORD PTR SS:[ESP+14]      ; <== Right Code
004039A4 |. 50      PUSH EAX
004039A5 |. E8 86000000 CALL PrinterE.00403A30
004039AA |. 8D4C24 18  LEA ECX,DWORD PTR SS:[ESP+18]
004039AE |. 51      PUSH ECX
004039AF |. 56      PUSH ESI
004039B0 |. E8 1B000000 CALL PrinterE.004039D0
004039B5 |. 83C4 18  ADD ESP,18
004039B8 |. F7D8    NEG EAX
004039BA |. 1BC0    SBB EAX,EAX
004039BC |. 5F      POP EDI
004039BD |. F7D8    NEG EAX
004039BF |. 5E      POP ESI
004039C0 |. 81C4 00010000 ADD ESP,100
004039C6 \. C3     RETN

```

===== Trace Into =====

```

00403649 . 83C4 08  ADD ESP,8
0040364C . 85C0    TEST EAX,EAX                      ;<== Wrong Code
0040364E . 5F      POP EDI
0040364F . 74 50   JE SHORT PrinterE.004036A1        ;<== Jump to Nag
00403651 . 8D5424 04  LEA EDX,DWORD PTR SS:[ESP+4]
00403655 . 52      PUSH EDX
00403656 . E8 D5030000 CALL PrinterE.00403A30
0040365B . 8D4424 08  LEA EAX,DWORD PTR SS:[ESP+8]
0040365F . 8D8C24 080100>LEA ECX,DWORD PTR SS:[ESP+108]
00403666 . 50      PUSH EAX
00403667 . 51      PUSH ECX
00403668 . 68 6CD14000 PUSH PrinterE.0040D16C ;ASCII "Software\PrinterExpress\Registration"
0040366D . 68 01000080 PUSH 80000001
00403672 . E8 D9030000 CALL PrinterE.00403A50
00403677 . 68 6CD14000 PUSH PrinterE.0040D16C; ASCII"Software\PrinterExpress\Registration"
0040367C . 68 01000080 PUSH 80000001
00403681 . E8 CA010000 CALL PrinterE.00403850
00403686 . 83C4 1C  ADD ESP,1C
00403689 . 6A 01    PUSH 1                           ; /Result = 1
0040368B . 56      PUSH ESI                         ; |hWnd
0040368C . FF15 10B24000 CALL DWORD PTR DS:<&USER32.EndDialog> ; \EndDialog
00403692 . B8 01000000 MOV EAX,1
00403697 . 5E      POP ESI
00403698 . 81C4 00020000 ADD ESP,200
0040369E . C2 1000  RETN 10
004036A1 > 6A 00    PUSH 0                           ; /Style = MB_OK|MB_APPLMODAL
004036A3 . 68 78D04000 PUSH PrinterE.0040D078 ;|Title = "PrinterExpress"
004036A8 . 68 5CD24000 PUSH PrinterE.0040D25C ;|Text = "Incorrect registration code!"

```

```

004036AD . 56      PUSH ESI           ; |hOwner
004036AE . FF15 54B14000 CALL DWORD PTR DS:[<&USER32.MessageBoxA>;
|MessageBoxA <== Shot Nag
004036B4 . B8 01000000 MOV EAX,1
004036B9 . 5E      POP ESI
004036BA . 81C4 00020000 ADD ESP,200
004036C0 . C2 1000 RETN 10

```

/*/*/* - SERIAL tương ứng :

User : kienmanowar	Serial : 1444-46020-1466-1180
User : REA-cRaCkErTeAm	Serial : 1615-48945-1645-1255

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;
int i=0,j=0;
int Temp=0, Temp1 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0, Sec4_temp = 0;

LenName=GetDlgItemText(IDC_NAME,reaName,64);
if (LenName < 1)
{
    MessageBox("-----===== Your name atleast 1 chart =====--- ","Hey !! Please
input your name again !! ");
}
else
{
    // Caculation 0
    while (i < LenName)
    {
        Temp = reaName[i] + 0x18;
        Temp1 = Temp1 + Temp ;
        i++;
    }
    Sec1_temp = Temp1;
    wsprintf(Sec1,"%u-",Sec1_temp);
    // Calculation 1
    i = 0;
    Temp = 0;
    Temp1 = 0;
    while (i < LenName)
    {
        Temp = reaName[i] * 0x27;
        Temp1 = Temp1 + Temp;
        i++;
    }
    Sec2_temp = Temp1;
    wsprintf(Sec2,"%u-",Sec2_temp);
    //Calculation 2
    i = 0;
}

```

```

Temp = 0 ;
Temp1 = 0;
while ( i < LenName)
{
    Temp = reaName[i] + 0x1A;
    Temp1 = Temp1 + Temp;
    i++;
}

Sec3_temp = Temp1;
wsprintf(Sec3,"%u-",Sec3_temp);
//Calculation 3
i = 0;
Temp = 0;
Temp1 = 0;
while (i< LenName)
{
    Temp = reaName[i] * 0x1;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec4_temp = Temp1;
wsprintf(Sec4,"%u",Sec4_temp);
}
// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);

SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.southbaypc.com/
Production	:	South Bay Software
SoftWare	:	Super Cleaner v2.75
Copyright by	:	Copyright (C) 1999 - 2004 South Bay Software . All Rights Reserved.
Type	:	Name/Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack	:	N/A
Request	:	Correct Serial/Keygen

Super Cleaner v2.75

SuperCleaner is an all in one disk cleaner for your computer. It features a variety of cleaning methods, in order to gain back your valuable hard disk space. Its garbage file scanner is able to locate up to hundreds or even thousands of megabytes of unneeded files which are simply taking up space on your computer. The Internet Privacy feature of SuperCleaner will let you erase your web browser's cache (temporary Internet files), history, and cookies.....

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng Microsoft Visual C++ 6.0 , không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là **Enter Registration...** . Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập **Name & Serial** vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ở đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (**Name : kienmanowar & Code : 11111982**). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ **Sorry , you have entered an incorrect registration code** ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Ấc ặc nhưng tìm một hồi ,chúng ta không tìm thấy cái chuỗi này trong Olly . Đành bó tay sao , Oh không à . Chúng ta lại có dịp được thực hành thêm về phương pháp Stack của Anh Moon .

-**** **Ctrl-F2** để Restart lại chương trình trong Olly . Nhấn **F9** để Run chương trình . Nhập các thông tin đăng ký vào (FU,FS). Nhấn Ok . Quay trở lại Ollydbg , nhấn **F12** để Pause chương trình lại .Nhấn **Alt+K** để tới cửa sổ **Call Stack of Main Thread**. Tại cửa sổ này chúng ta thấy như sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012E16C	77D43FBE	Includes 7FFE0304	USER32.77D43FBC	0012E1A0
0012E170	77D487A7	USER32.WaitMessage	USER32.77D487A2	0012E1A0
0012E1A4	77D4F58C	USER32.77D48607	USER32.77D4F587	0012E1A0
0012E1CC	77D6AAAE	USER32.77D4F4D8	USER32.77D6AAA9	0012E1C8
0012E484	77D6AC40	? USER32.SoftModalMessageBox	USER32.77D6AC3B	
0012E40C				
0012E558	77F54169	ntdll.77F81C55	ntdll.77F54164	0012E554

-**** Ở đây chúng ta chú ý đến dòng màu đỏ. Double Click vào cột Called From của dòng này chúng ta sẽ đến đây :

```
77D6AC3B E8 09F9FFFF CALL USER32.SoftModalMessageBox
77D6AC40 66:837E 2C 00 CMP WORD PTR DS:[ESI+2C],0 <== Set Breakpoint here
```

-**** Sau khi đặt BP tại đây, Olly sẽ dừng chương trình lại , nhấn F8 để xuất hiện lại thông báo “ **Sorry , you have entered an incorrect registration code** ”.Nhấn Ok để chấp nhận thông báo này , chương trình sẽ dừng lại tại điểm mà chúng ta Set BP ở trên. Xóa điểm đặt Bp này đi , sau đó dùng F8 để Trace tiếp.

-**** Sau khi trace qua một số lần (cụ thể ở chương trình này là 6) RETN ta sẽ đến đoạn code sau :

```
0041E33B . E8 A013FFFF CALL SuperCle.0040F6E0 <== Call Nag
0041E340 . 83C4 10 ADD ESP,10
```

-**** Lần ngược lên trên một chút chúng ta sẽ Set BP tại đây :

```
0041E2B7 . FFD7 CALL EDI ; \GetDlgItemTextA <== Set breakpoint here
```

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

```
0041E2B7 . FFD7 CALL EDI ; \GetDlgItemTextA <== We're here (Get FU)
0041E2B9 . 8D5424 08 LEA EDX,DWORD PTR SS:[ESP+8]
0041E2BD . 68 00010000 PUSH 100 ; /Count = 100 (256.)
0041E2C2 . 52 PUSH EDX ; |Buffer
0041E2C3 . 68 F3030000 PUSH 3F3 ; |ControlID = 3F3 (1011.)
0041E2C8 . 56 PUSH ESI ; |hWnd
0041E2C9 . FFD7 CALL EDI ; \GetDlgItemTextA <== Get FS
0041E2CB . E8 60E8FFFF CALL SuperCle.0041CB30 <== Check BlackList
0041E2D0 . 85C0 TEST EAX,EAX
0041E2D2 . 5F POP EDI
0041E2D3 . 75 5C JNZ SHORT SuperCle.0041E331
0041E2D5 . 8D4424 04 LEA EAX,DWORD PTR SS:[ESP+4] ;<== FS
0041E2D9 . 8D8C24 040100>LEA ECX,DWORD PTR SS:[ESP+104] ;<== FU
0041E2E0 . 50 PUSH EAX
0041E2E1 . 51 PUSH ECX
0041E2E2 . E8 89050000 CALL SuperCle.0041E870 ;<== Encrypt (Trace Into)
```

----- Trace Into -----

```
0041E870 /$ 81EC 00010000 SUB ESP,100
0041E876 |. 53 PUSH EBX
0041E877 |. 8B9C24 080100>MOV EBX,DWORD PTR SS:[ESP+108] ;<== FU
0041E87E |. 53 PUSH EBX
0041E87F |. E8 1CEDFFFF CALL SuperCle.0041D5A0
0041E884 |. 83C4 04 ADD ESP,4
0041E887 |. 85C0 TEST EAX,EAX
0041E889 |. 74 0A JE SHORT SuperCle.0041E895
0041E88B |. 33C0 XOR EAX,EAX
0041E88D |. 5B POP EBX
0041E88E |. 81C4 00010000 ADD ESP,100
0041E894 |. C3 RETN
0041E895 |> A0 74C24300 MOV AL,BYTE PTR DS:[43C274]
0041E89A |. 56 PUSH ESI
0041E89B |. 57 PUSH EDI
0041E89C |. 884424 0C MOV BYTE PTR SS:[ESP+C],AL
0041E8A0 |. B9 3F000000 MOV ECX,3F
0041E8A5 |. 33C0 XOR EAX,EAX
0041E8A7 |. 8D7C24 0D LEA EDI,DWORD PTR SS:[ESP+D]
0041E8AB |. 33F6 XOR ESI,ESI
0041E8AD |. F3:AB REP STOS DWORD PTR ES:[EDI]
0041E8AF |. 66:AB STOS WORD PTR ES:[EDI]
0041E8B1 |. 8D4C24 0C LEA ECX,DWORD PTR SS:[ESP+C]
0041E8B5 |. 51 PUSH ECX
```

0041E8B6 |. 53 PUSH EBX
 0041E8B7 |. AA STOS BYTE PTR ES:[EDI]
0041E8B8 |. E8 B3000000 CALL SuperCle.0041E970 ;<= Calculation (Trace Into)
 ===== Trace Into =====
 0041E970 /\$ 81EC 00010000 SUB ESP,100
 0041E976 |. A0 74C24300 MOV AL,BYTE PTR DS:[43C274]
 0041E97B |. 53 PUSH EBX
 0041E97C |. 55 PUSH EBP
 0041E97D |. 56 PUSH ESI
 0041E97E |. 57 PUSH EDI
 0041E97F |. 884424 10 MOV BYTE PTR SS:[ESP+10],AL
 0041E983 |. B9 3F000000 MOV ECX,3F
 0041E988 |. 33C0 XOR EAX,EAX
 0041E98A |. 8D7C24 11 LEA EDI,DWORD PTR SS:[ESP+11]
 0041E98E |. F3:AB REP STOS DWORD PTR ES:[EDI]
 0041E990 |. 66:AB STOS WORD PTR ES:[EDI]
 0041E992 |. AA STOS BYTE PTR ES:[EDI]
 0041E993 |. 8BBC24 140100>MOV EDI,DWORD PTR SS:[ESP+114] ;<= FU
 0041E99A |. 57 PUSH EDI ;/String
 0041E99B |. FF15 90024300 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ;\lstrlenA <== Get Length(FU)
 0041E9A1 |. 8BF0 MOV ESI,EAX ;<== Length (FU);
 0041E9A3 |. 33C9 XOR ECX,ECX ;<== Sec1 = 0;
 0041E9A5 |. 33C0 XOR EAX,EAX ;<== i = 0;
 0041E9A7 |. 85F6 TEST ESI,ESI
 0041E9A9 |. 7E 13 JLE SHORT SuperCle.0041E9BE
0041E9AB |. 8B15 14894300 MOV EDX,DWORD PTR DS:[438914] ;<=1st Default Value (0x26)
 ===== Calculation 0 =====
 0041E9B1 |> 0FBE1C38 /MOVSX EBX,BYTE PTR DS:[EAX+EDI] ;<= FU[i];
 0041E9B5 |. 03DA |ADD EBX,EDX ;<= Temp = FU[i] + **1st Default Value (0x26)**;
 0041E9B7 |. 03CB |ADD ECX,EBX ;<= Sec1 = Sec1 + Temp;
 0041E9B9 |. 40 |INC EAX ;<== i++
 0041E9BA |. 3BC6 |CMP EAX,ESI ;<= While (i < Length(FU))
 0041E9BC |.^ 7C F3 |JL SHORT SuperCle.0041E9B1 ;<= Continue Loop
 0041E9BE |> 8B9C24 180100>MOV EBX,DWORD PTR SS:[ESP+118]
 0041E9C5 |. 51 PUSH ECX ; /<%ld> ;<== Sec1 (convert to hex -->dec)
 0041E9C6 |. 68 308F4300 PUSH SuperCle.00438F30 ;|Format = "%ld-" ;<== Sec1 + "-"
 0041E9CB |. 53 PUSH EBX ;|s
 0041E9CC |. FF15 28044300 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ;\wsprintfA <== Display Sec1
 ===== Calculation 0 =====
 0041E9D2 |. 83C4 0C ADD ESP,0C
 0041E9D5 |. 33C9 XOR ECX,ECX ;<== Sec2 = 0;
 0041E9D7 |. 33C0 XOR EAX,EAX ;<== i = 0;
 0041E9D9 |. 85F6 TEST ESI,ESI
 0041E9DB |. 7E 14 JLE SHORT SuperCle.0041E9F1
0041E9DD |. 8B15 18894300 MOV EDX,DWORD PTR DS:[438918];<= 2nd Default Value (0x34)
 ===== Calculation 1 =====
 0041E9E3 |> 0FBE2C38 /MOVSX EBP,BYTE PTR DS:[EAX+EDI] ;<= FU[i]
 0041E9E7 |. 0FAFEA |IMUL EBP,EDX ;<= Temp = FU[i] * **2nd Default Value (0x34)**
 0041E9EA |. 03CD |ADD ECX,EBP ;<== Sec2 = Sec2 + Temp
 0041E9EC |. 40 |INC EAX ;<== i ++

```

0041E9ED |. 3BC6      |CMP EAX,ESI          ; While (i < Length (FU))
0041E9EF |.^ 7C F2    |JL SHORT SuperCle.0041E9E3; <== Continue Loop
0041E9F1 |> 51       |PUSH ECX           ; /%ld> ; <== Sec2 (Convert Hex -->Dec)
0041E9F2 |. 8D4C24 14  |LEA ECX,DWORD PTR SS:[ESP+14] ; |
0041E9F6 |. 68 308F4300 |PUSH SuperCle.00438F30 ; |Format = "%ld-" <== Sec2 + "-"
0041E9FB |. 51       |PUSH ECX           ; |s
0041E9FC |. FF15 28044300 |CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA <== Display Sec2

```

===== Calculation 1 =====

```

0041EA02 |. 83C4 0C   |ADD ESP,0C
0041EA05 |. 8D5424 10 |LEA EDX,DWORD PTR SS:[ESP+10]
0041EA09 |. 52       |PUSH EDX           ; /StringToAdd
0041EA0A |. 53       |PUSH EBX           ; |ConcatString
0041EA0B |. FF15 78024300 |CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA <== Sec1
ghép với Sec2
0041EA11 |. 33C9      |XOR ECX,ECX        ; <== Sec3 = 0;
0041EA13 |. 33C0      |XOR EAX,EAX        ; <== i = 0;
0041EA15 |. 85F6      |TEST ESI,ESI
0041EA17 |. 7E 13      |JLE SHORT SuperCle.0041EA2C
0041EA19 |. 8B15 1C894300 MOV EDX,DWORD PTR DS:[43891C];<=3rd Default Value(0x0C)

```

===== Calculation 2 =====

```

0041EA1F |> 0FBE2C38  |MOVSX EBP,BYTE PTR DS:[EAX+EDI]; <== FU[i];
0041EA23 |. 03EA      |ADD EBP,EDX         ; Temp = FU[i] + 3rd Default Value(0x0C);
0041EA25 |. 03CD      |ADD ECX,EBP         ; Sec3 = Sec3 + Temp;
0041EA27 |. 40       |INC EAX            ; i ++
0041EA28 |. 3BC6      |CMP EAX,ESI        ; While (i < Length (FU))
0041EA2A |.^ 7C F3    |JL SHORT SuperCle.0041EA1F      ; Continue Loop
0041EA2C |> 51       |PUSH ECX           ; /%ld>; <== Sec3 (Convert Hex-- > Dec)
0041EA2D |. 8D4424 14  |LEA EAX,DWORD PTR SS:[ESP+14]      ; |
0041EA31 |. 68 308F4300 |PUSH SuperCle.00438F30      ; |Format = "%ld-" <== Sec3 + "-"
0041EA36 |. 50       |PUSH EAX           ; |s
0041EA37 |. FF15 28044300 |CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA <== Display Sec3

```

===== Calculation 2 =====

```

0041EA3D |. 83C4 0C   |ADD ESP,0C
0041EA40 |. 8D4C24 10 |LEA ECX,DWORD PTR SS:[ESP+10]
0041EA44 |. 51       |PUSH ECX           ; /StringToAdd
0041EA45 |. 53       |PUSH EBX           ; |ConcatString
0041EA46 |. FF15 78024300 |CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA
0041EA4C |. 33C9      |XOR ECX,ECX        ; Sec4 = 0;
0041EA4E |. 33C0      |XOR EAX,EAX        ; i = 0;
0041EA50 |. 85F6      |TEST ESI,ESI
0041EA52 |. 7E 14      |JLE SHORT SuperCle.0041EA68
0041EA54 |. 8B15 20894300 MOV EDX,DWORD PTR DS:[438920]<= 4th Default Value (0x0E)

```

===== Calculation 3 =====

```

0041EA5A |> 0FBE2C38  |MOVSX EBP,BYTE PTR DS:[EAX+EDI]; <== FU[i];
0041EA5E |. 0FAFEA    |IMUL EBP,EDX         ; Temp = FU[i] * 4th Default Value (0x0E)
0041EA61 |. 03CD      |ADD ECX,EBP         ; Sec4 = Sec4 + Temp
0041EA63 |. 40       |INC EAX            ; i ++
0041EA64 |. 3BC6      |CMP EAX,ESI        ; While( i < Length (FU))
0041EA66 |.^ 7C F2    |JL SHORT SuperCle.0041EA5A      ; <== Continue Loop
0041EA68 |> 51       |PUSH ECX           ; /%ld>

```

```

0041EA69 |. 8D5424 14 LEA EDX,DWORD PTR SS:[ESP+14]      ; |
0041EA6D |. 68 2C8F4300 PUSH SuperCle.00438F2C           ; |Format = "%ld"
0041EA72 |. 52      PUSH EDX                ; |s
0041EA73 |. FF15 28044300 CALL DWORD PTR DS:[<&USER32.wsprintfA>] ; \wsprintfA
<==Display Sec4

```

===== Calculation 3 =====

```

0041EA79 |. 83C4 0C ADD ESP,0C
0041EA7C |. 8D4424 10 LEA EAX,DWORD PTR SS:[ESP+10]
0041EA80 |. 50      PUSH EAX               ; /StringToAdd
0041EA81 |. 53      PUSH EBX               ; |ConcatString
0041EA82 |. FF15 78024300 CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>] ; \lstrcatA <== Right
Code After Calculated
0041EA88 |. 5F      POP EDI
0041EA89 |. 5E      POP ESI
0041EA8A |. 5D      POP EBP
0041EA8B |. 5B      POP EBX
0041EA8C |. 81C4 00010000 ADD ESP,100
0041EA92 \. C3      RETN

```

===== Trace Into =====

```

0041E8BD |. 8B8424 1C0100>MOV EAX,DWORD PTR SS:[ESP+11C] ; <== FS
0041E8C4 |. 8D5424 14 LEA EDX,DWORD PTR SS:[ESP+14]       ; <== Right Code
0041E8C8 |. 52      PUSH EDX
0041E8C9 |. 50      PUSH EAX
0041E8CA |. E8 51FFFFFF CALL SuperCle.0041E820 ;<== Compare
0041E8CF |. 83C4 10 ADD ESP,10
0041E8D2 |. 85C0      TEST EAX,EAX
0041E8D4 |. 74 05      JE SHORT SuperCle.0041E8DB
0041E8D6 |. BE 01000000 MOV ESI,1
0041E8DB |> 8BC6      MOV EAX,ESI
0041E8DD |. 5F      POP EDI
0041E8DE |. 5E      POP ESI
0041E8DF |. 5B      POP EBX
0041E8E0 |. 81C4 00010000 ADD ESP,100
0041E8E6 \. C3      RETN

```

===== Trace Into =====

```

0041E2E7 . 83C4 08 ADD ESP,8
0041E2EA . 85C0      TEST EAX,EAX      ;<== Wrong Code
0041E2EC . 74 43      JE SHORT SuperCle.0041E331 ;<== Jump to Nag
0041E2EE . 8D5424 04 LEA EDX,DWORD PTR SS:[ESP+4]
0041E2F2 . 8D8424 040100>LEA EAX,DWORD PTR SS:[ESP+104]
0041E2F9 |. 52      PUSH EDX
0041E2FA |. 50      PUSH EAX
0041E2FB |. 68 54844300 PUSH SuperCle.00438454 ; ASCII "Software\SuperCleaner\Registration"
0041E300 |. 68 01000080 PUSH 80000001
0041E305 |. E8 E6050000 CALL SuperCle.0041E8F0
0041E30A |. 68 54844300 PUSH SuperCle.00438454 ; ASCII "Software\SuperCleaner\Registration"
0041E30F |. 68 01000080 PUSH 80000001
0041E314 |. E8 A7030000 CALL SuperCle.0041E6C0
0041E319 |. 83C4 18 ADD ESP,18
0041E31C |. 6A 01      PUSH 1          ; /Result = 1
0041E31E |. 56      PUSH ESI          ; |hWnd
0041E31F |. FF15 B8034300 CALL DWORD PTR DS:[<&USER32.EndDialog>] ; \EndDialog

```

```

0041E325 . 33C0      XOR EAX,EAX
0041E327 . 5E        POP ESI
0041E328 . 81C4 00020000 ADD ESP,200
0041E32E . C2 1000   RETN 10
0041E331 > 6A 00    PUSH 0
0041E333 . 68 D4314300 PUSH SuperCle.004331D4      ; ASCII "SuperCleaner"
0041E338 . 6A 0A    PUSH 0A
0041E33A . 56        PUSH ESI
0041E33B . E8 A013FFFF CALL SuperCle.0040F6E0 <== Shot Nag

```

/*/*/* - SERIAL trong ứng :

User : kienmanowar	Serial : 1598-61360-1312-16520
User : REA-cRaCkErTeAm	Serial : 1825-65260-1435-17570

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;
int i=0,j=0;
int Temp=0, Temp1 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0, Sec4_temp = 0;

LenName=GetDlgItemText(IDC_NAME,reaName,64);
if (LenName < 1)
{
    MessageBox("----- Your name atleast 1 chart -----","Hey !! Please input
your name again !! ");
}
else
{
    // Calculation 0
    while (i < LenName)
    {
        Temp = reaName[i] + 0x26;
        Temp1 = Temp1 + Temp ;
        i++;
    }
    Sec1_temp = Temp1;
    wsprintf(Sec1,"%u-",Sec1_temp);
    i = 0;
    Temp = 0;
    Temp1 = 0;
    // Calculation 1
    while (i < LenName)
    {
        Temp = reaName[i] * 0x34;
        Temp1 = Temp1 + Temp;
        i++;
    }
    Sec2_temp = Temp1;
}

```

```

wsprintf(Sec2,"%u-",Sec2_temp);
//Calculation 2
i = 0;
Temp = 0 ;

Temp1 = 0;
while ( i < LenName)
{
    Temp = reaName[i] + 0xC;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec3_temp = Temp1;
wsprintf(Sec3,"%u-",Sec3_temp);

//Calculation 3
i = 0;
Temp = 0;
Temp1 = 0;
while (i< LenName)
{
    Temp = reaName[i] * 0xE;
    Temp1 = Temp1 + Temp;
    i++;
}
Sec4_temp = Temp1;
wsprintf(Sec4,"%u",Sec4_temp);
}
// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);
SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association

SoftWare

Homepage	:	http://www.southbaypc.com/
Production	:	South Bay Software
SoftWare	:	SysDate v1.3
Copyright by	:	Copyright (C) 1998 - 2004 South Bay Software . All Rights Reserved.
Type	:	Name/Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 6.0

Crack Tool : OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
 Unpack : N/A
 Request : Correct Serial/Keygen

SysDate v1.3

SysDate displays the day of the month in the system tray (next to the time). How many times have you needed to know the current date, without fumbling through the Windows calendar? SysDate is a huge timesaver, and its affordability makes it a must have! You can customize it to show the date in different colors. It is small, and takes up very little memory, and virtually no CPU time.

I – Information :

-**** Dùng PeiD v0.92 để Detect, chúng ta biết được chương trình này được tác giả Code bằng Microsoft Visual C++ 6.0 , không hề bị Pack . Vậy là vượt qua được cửa ải Unpack mà không mất một giọt mồ hôi nào.

-**** Chạy thử chương trình , một Nag thông báo của chương trình hiện lên. Cụ thể là nó chỉ cho phép chúng ta dùng thử chương trình này trong vòng không quá 30 days , nếu quá 30 days nó tự động nghỉ hưu liền à. Ngay tại Nag này bạn có thể nhìn thấy một button là **Enter Registration...** . Oki , click vào đây , nó sẽ hiện một Dialog box yêu cầu chúng ta phải nhập **Name & Serial** vào để sử dụng chương trình một cách hợp pháp.

-**** Nhưng chúng ta làm gì có đâu để mà nhập , do đó có 2 cách một là dùng hết 30 ngày (để rồi lại reinstall lại) , hai là chúng ta phải crack nó để có được Serial hợp pháp với Name chúng ta nhập vào. Ở đây mình chọn cách thứ hai à. Nhập đại một cái Name & Code vào , ví dụ (**Name : kienmanowar & Code : 11111982**). Nhập xong nhấn Ok, ngay lập tức một Nag văng ra “ **Sorry , you have entered an incorrect registration code** ”. Nó mắng ta như thế thì cũng đành chịu chứ biết làm thế nào bây giờ, hãy ghi nhớ thông tin này lại chúng ta sẽ làm cho nó không còn xuất hiện nữa ke ke.

-**** Bây giờ chúng ta hãy Load chương trình này trong Olly , tìm chuỗi thông báo trên . Ăc ặc nhưng tìm một hồi ,chúng ta không tìm thấy cái chuỗi này trong Olly . Đành bó tay sao , Oh không à . Chúng ta lại có dịp được thực hành thêm về phương pháp Stack của Anh Moon .

-**** Ctrl-F2 để Restart lại chương trình trong Olly . Nhấn F9 để Run chương trình . Nhập các thông tin đăng kí vào (FU,FS). Nhấn Ok . Quay trở lại Ollydbg , nhấn F12 để Pause chương trình lại .Nhấn Alt+K để tới cửa sổ **Call Stack of Main Thread**. Tại cửa sổ này chúng ta thấy như sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012E6D4	77D43FBE	Includes 7FFE0304	USER32.77D43FBC	0012E708
0012E6D8	77D487A7	USER32.WaitMessage	USER32.77D487A2	0012E708
0012E70C	77D4F58C	USER32.77D48607	USER32.77D4F587	0012E708
0012E734	77D6AAAE	USER32.77D4F4D8	USER32.77D6AAA9	0012E730
0012E9EC	77D6AC40	? USER32.SoftModalMessageBox	USER32.77D6AC3B	0012E974
0012EAC0	77F54125	ntdll.RtlImageDirectoryEntryToData	ntdll.77F54120	0012EABC

-**** Ở đây chúng ta chú ý đến dòng màu đỏ. Double Click vào cột Called From của dòng này chúng ta sẽ đến đây :

77D6AC3B E8 09F9FFFF CALL USER32.SoftModalMessageBox
 77D6AC40 66:837E 2C 00 CMP WORD PTR DS:[ESI+2C],0 ;<== Set BP here

-**** Sau khi đặt BP tại đây, Olly sẽ dừng chương trình lại , nhấn F8 để xuất hiện lại thông báo “ **Sorry , you have entered an incorrect registration code** ”.Nhấn Ok để chấp nhận thông báo này , chương trình sẽ dừng lại tại điểm mà chúng ta Set BP ở trên. Xóa điểm đặt Bp này đi , sau đó dùng F8 để Trace tiếp.

-**** Sau khi trace qua một số lần (cụ thể ở chương trình này là 6) RETN ta sẽ đến đoạn code sau :
 00401F6C . E8 8FF0FFFF CALL SysDate.00401000 <==== Call Nag
 00401F71 . 83C4 10 ADD ESP,10

-**** Lần ngược lên trên một chút chúng ta sẽ Set BP tại đây :
 00401EE8 . FFD7 CALL EDI ; \GetDlgItemTextA <== We set BP here

II – Cracking :

-**** Oki , sau khi đặt BP, nhấn F9 để run chương trình , nhập lại Name & Code vào hộp thoại đăng kí , sau đó nhấn Ok. Chúng ta sẽ quay trở lại Ollydbg và Ice tại chỗ mà chúng ta đã Set BP :

00401EE8 . FFD7 CALL EDI ; \GetDlgItemTextA <== We're here (Get FU)

```
00401EEA . 8D5424 08 LEA EDX,DWORD PTR SS:[ESP+8]
00401EEE . 68 00010000 PUSH 100 ; /Count = 100 (256.)
00401EF3 . 52 PUSH EDX ; |Buffer
00401EF4 . 68 ED030000 PUSH 3ED ; |ControlID = 3ED (1005.)
00401EF9 . 56 PUSH ESI ; |hWnd
00401EFA . FFD7 CALL EDI ; \GetDlgItemTextA <== Get FS
00401EFC . E8 2FFFAFFF CALL SysDate.00401930
00401F01 . 85C0 TEST EAX,EAX
00401F03 . 5F POP EDI
00401F04 . 75 5C JNZ SHORT SysDate.00401F62
00401F06 . 8D4424 04 LEA EAX,DWORD PTR SS:[ESP+4] ;<== FS
00401F0A . 8D8C24 040100>LEA ECX,DWORD PTR SS:[ESP+104] ;<== FU
00401F11 . 50 PUSH EAX
00401F12 . 51 PUSH ECX
00401F13 . E8 48030000 CALL SysDate.00402260 <== Encrypt (Trace Into)
```

----- Trace Into -----

```
00402260 /$ 81EC 00010000 SUB ESP,100
00402266 |. 56 PUSH ESI
00402267 |. 8BB424 080100>MOV ESI,DWORD PTR SS:[ESP+108] ;<== FU
0040226E |. 56 PUSH ESI
0040226F |. E8 DCF9FFFF CALL SysDate.00401C50
00402274 |. 83C4 04 ADD ESP,4
00402277 |. 85C0 TEST EAX,EAX
00402279 |. 74 0A JE SHORT SysDate.00402285
0040227B |. 33C0 XOR EAX,EAX
0040227D |. 5E POP ESI
0040227E |. 81C4 00010000 ADD ESP,100
00402284 |. C3 RETN
00402285 |> 8D4424 04 LEA EAX,DWORD PTR SS:[ESP+4]
00402289 |. 50 PUSH EAX
0040228A |. 56 PUSH ESI
```

0040228B |. E8 B0000000 CALL SysDate.00402340 <== Calculation (Trace Into)

----- Trace Into -----

----- NOTE -----

Trong phần Caculation này có tất cả 4 đoạn Code tính toán Right Code của chương trình này , thực chất chỉ là việc cộng dồn các FU[i] lại với nhau và cộng với một default value nằm trong thanh ghi EDX hoặc là sử dụng phép nhân FU[i] với default value , sau đó đếm cộng lại . Mỗi đoạn sẽ tương ứng với từng SecX. (4 function thì sẽ có 4 SecX , mỗi SecX sẽ được nối với nhau thông qua “-“).

----- NOTE -----

00402340 /\$ 81EC 00010000 SUB ESP,100

00402346 |. A0 98EF4000 MOV AL,BYTE PTR DS:[40EF98]
 0040234B |. 53 PUSH EBX
 0040234C |. 55 PUSH EBP
 0040234D |. 56 PUSH ESI
 0040234E |. 57 PUSH EDI
 0040234F |. 884424 10 MOV BYTE PTR SS:[ESP+10],AL
 00402353 |. B9 3F000000 MOV ECX,3F
 00402358 |. 33C0 XOR EAX,EAX
 0040235A |. 8D7C24 11 LEA EDI,DWORD PTR SS:[ESP+11]
 0040235E |. F3:AB REP STOS DWORD PTR ES:[EDI]
 00402360 |. 66:AB STOS WORD PTR ES:[EDI]
 00402362 |. AA STOS BYTE PTR ES:[EDI]
 00402363 |. 8BBC24 140100>MOV EDI,DWORD PTR SS:[ESP+114] ;<== FU
 0040236A |. 57 PUSH EDI ;/String
 0040236B |. FF15 40A14000 CALL DWORD PTR DS:[<&KERNEL32.lstrlenA>] ; \lstrlenA ;<== Get Length(FU)
 00402371 |. 8BF0 MOV ESI,EAX ; Length (FU)
 00402373 |. 33C0 XOR EAX,EAX ; i = 0;
 00402375 |. 85F6 TEST ESI,ESI
 00402377 |. B9 6B000000 MOV ECX,6B ; Temp_01
 0040237C |. 76 0F JBE SHORT SysDate.0040238D

===== Calculation 0 =====

0040237E |> 0FBEBE1438 /MOVSX EDX,BYTE PTR DS:[EAX+EDI] ;<== FU[i]
 00402382 |. 40 |INC EAX ;<== i ++
 00402383 |. 8D1492 |LEA EDX,DWORD PTR DS:[EDX+EDX*4]; Temp_02 = FU[i] + FU[i] * 0x4
 00402386 |. 3BC6 |CMP EAX,ESI ; While (i < Length(FU))
 00402388 |. 8D0C91 |LEA ECX,DWORD PTR DS:[ECX+EDX*4];Temp_01 = Temp_01 + Temp_02 *0x4
 0040238B |.^ 72 F1 JB SHORT SysDate.0040237E ; Continue Loop
 0040238D |> 8B9C24 180100>MOV EBX,DWORD PTR SS:[ESP+118]
 00402394 |. 8B2D 38A24000 MOV EBP,DWORD PTR DS:[<&USER32.wsprintfA>;
 USER32.wsprintfA
 0040239A |. 51 PUSH ECX ; /%u ; <== Sec1 of Real Code (hex-->dec)
 0040239B |. 68 10C34000 PUSH SysDate.0040C310 ;|Format = "%u-" <== Sec1 + "-"
 004023A0 |. 53 PUSH EBX ;|s
 004023A1 |. FFD5 CALL EBP ; \wsprintfA <== Display Sec1

===== Calculation 0 =====

004023A3 |. 83C4 0C ADD ESP,0C
 004023A6 |. 33C0 XOR EAX,EAX ;<== i = 0;
 004023A8 |. 85F6 TEST ESI,ESI
 004023AA |. B9 6B000000 MOV ECX,6B ; Temp_01
 004023AF |. 76 12 JBE SHORT SysDate.004023C3

===== Calculation 1 =====

004023B1 |> 0FBEBE1438 /MOVSX EDX,BYTE PTR DS:[EAX+EDI] ;<== FU[i]
 004023B5 |. 40 |INC EAX ;<== i ++
 004023B6 |. 8D1492 |LEA EDX,DWORD PTR DS:[EDX+EDX*4] ;<==Temp_02= FU[i] + FU[i] * 0x4
 004023B9 |. 3BC6 |CMP EAX,ESI <== While (i < Length(FU))
 004023BB |. 8D1492 |LEA EDX,DWORD PTR DS:[EDX+EDX*4] ;<== Temp_02 = Temp_02 +Temp_02 * 0x4
 004023BE |. 8D0CD1 |LEA ECX,DWORD PTR DS:[ECX+EDX*8] ;<== Temp_01 = Temp_01 + Temp_02 * 0x8
 004023C1 |.^ 72 EE JB SHORT SysDate.004023B1

```

004023C3 |> 51      PUSH ECX ;<== Sec2 of Right Code
004023C4 |. 8D4424 14  LEA EAX,DWORD PTR SS:[ESP+14]
004023C8 |. 68 10C34000 PUSH SysDate.0040C310          ; ASCII "%u-"
004023CD |. 50      PUSH EAX
004023CE |. FFD5      CALL EBP <== Display Sec2

```

===== Calculation 1 =====

```

004023D0 |. 83C4 0C      ADD ESP,0C
004023D3 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
004023D7 |. 51      PUSH ECX          ; /StringToAdd
004023D8 |. 53      PUSH EBX          ; |ConcatString
004023D9 |. FF15 20A14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA <== Sec1
ghép với Sec2

```

===== Calculation 2 =====

```

004023DF |. 0FBE543E FF  MOVSX EDX,BYTE PTR DS:[ESI+EDI-1] ;<== Sec3 = FU
[Length(FU) - 1]
004023E4 |. 83C2 02      ADD EDX,2 ;<== Sec3 = Sec3 + 0x2
004023E7 |. 8D4424 10  LEA EAX,DWORD PTR SS:[ESP+10]
004023EB |. 52      PUSH EDX ;<== Sec3 (hex -- > dec)
004023EC |. 68 10C34000 PUSH SysDate.0040C310 ; ASCII "%u-" ;<== Sec3 + "-"
004023F1 |. 50      PUSH EAX
004023F2 |. FFD5      CALL EBP <== Display Sec3

```

===== Calculation 2 =====

```

004023F4 |. 83C4 0C      ADD ESP,0C
004023F7 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
004023FB |. 51      PUSH ECX          ; /StringToAdd
004023FC |. 53      PUSH EBX          ; |ConcatString
004023FD |. FF15 20A14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA

```

===== Calculation 3 =====

```

00402403 |. 0FBE443E FF  MOVSX EAX,BYTE PTR DS:[ESI+EDI-1] ;<==Sec4= FU[Length(FU) - 1]
00402408 |. 8D4C24 10  LEA ECX,DWORD PTR SS:[ESP+10]
0040240C |. 8D0480      LEA EAX,DWORD PTR DS:[EAX+EAX*4] ;<== Sec4 = Sec4 + Sec4 *0x4
0040240F |. 8D1480      LEA EDX,DWORD PTR DS:[EAX+EAX*4] ;<== Temp = Sec4 + Sec4 *0x4
00402412 |. 8D0495 010000>LEA EAX,DWORD PTR DS:[EDX*4+1] ;<== Sec4 = Temp *0x4 + 1
00402419 |. 50      PUSH EAX ;<== Sec4 (hex --> dec)
0040241A |. 68 0CC34000 PUSH SysDate.0040C30C ; ASCII "%u"
0040241F |. 51      PUSH ECX
00402420 |. FFD5      CALL EBP <== Display Sec4

```

===== Calculation 3 =====

```

00402422 |. 83C4 0C      ADD ESP,0C
00402425 |. 8D5424 10  LEA EDX,DWORD PTR SS:[ESP+10]
00402429 |. 52      PUSH EDX          ; /StringToAdd
0040242A |. 53      PUSH EBX          ; |ConcatString
0040242B |. FF15 20A14000 CALL DWORD PTR DS:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA <==
Right Code after Calculation

```

```

00402431 |. 5F      POP EDI
00402432 |. 5E      POP ESI
00402433 |. 5D      POP EBP
00402434 |. 5B      POP EBX
00402435 |. 81C4 00010000 ADD ESP,100
0040243B \. C3      RETN

```

===== Trace Into =====

```

00402290 |. 8B9424 140100>MOV EDX,DWORD PTR SS:[ESP+114] ;<== FS

```

```

00402297 |. 8D4C24 0C    LEA ECX,DWORD PTR SS:[ESP+C]      ;<== Right Code
0040229B |. 51        PUSH ECX
0040229C |. 52        PUSH EDX
0040229D |. E8 6FFFFFFF CALL SysDate.00402210          ;<== Compare
004022A2 |. 83C4 10    ADD ESP,10
004022A5 |. F7D8      NEG EAX
004022A7 |. 1BC0      SBB EAX,EAX
004022A9 |. 5E        POP ESI
004022AA |. F7D8      NEG EAX
004022AC |. 81C4 00010000 ADD ESP,100
004022B2 \. C3        RETN
----- Trace Into -----

```

```

00401F18 . 83C4 08    ADD ESP,8
00401F1B . 85C0      TEST EAX,EAX      ;<== Wrong Code
00401F1D . 74 43      JE SHORT SysDate.00401F62  ;<== Jump to Nag
00401F1F . 8D5424 04    LEA EDX,DWORD PTR SS:[ESP+4]
00401F23 . 8D8424 040100>LEA EAX,DWORD PTR SS:[ESP+104]
00401F2A . 52        PUSH EDX
00401F2B . 50        PUSH EAX
00401F2C . 68 B8C24000 PUSH SysDate.0040C2B8 ; ASCII "Software\SysDate\Configuration"
00401F31 . 68 01000080 PUSH 80000001
00401F36 . E8 85030000 CALL SysDate.004022C0
00401F3B . 68 B8C24000 PUSH SysDate.0040C2B8 ; ASCII "Software\SysDate\Configuration"
00401F40 . 68 01000080 PUSH 80000001
00401F45 . E8 C6010000 CALL SysDate.00402110
00401F4A . 83C4 18    ADD ESP,18
00401F4D . 6A 01      PUSH 1           ; /Result = 1
00401F4F . 56        PUSH ESI         ; |hWnd
00401F50 . FF15 28A24000 CALL DWORD PTR DS:[<&USER32.EndDialog>] ; \EndDialog
00401F56 . 33C0      XOR EAX,EAX
00401F58 . 5E        POP ESI
00401F59 . 81C4 00020000 ADD ESP,200
00401F5F . C2 1000    RETN 10
00401F62 > 6A 00      PUSH 0
00401F64 . 68 74C04000 PUSH SysDate.0040C074      ; ASCII "SysDate"
00401F69 . 6A 0A      PUSH 0A
00401F6B . 56        PUSH ESI
00401F6C . E8 8FF0FFFF CALL SysDate.00401000      ;<== Shot Nag
00401F71 . 83C4 10    ADD ESP,10
00401F74 > 33C0      XOR EAX,EAX          ; Default case of switch 00401E6D
00401F76 . 5E        POP ESI
00401F77 . 81C4 00020000 ADD ESP,200
00401F7D . C2 1000    RETN 10

```

/*/*/* - SERIAL tương ứng :

User : kienmanowar

Serial : 23707-236107-116-11401

User : REA-cRaCkErTeAm

Serial : 25207-251107-111-10901

III – Keygen :

```

char reaName[64]={0};
char reaSerial[64]={0};
int LenName=0;

```

```

int i=0,j=0;
int Temp=0, Temp1 = 0, Temp2 = 0;
char Sec1[10] = {0}, Sec2[10] = {0}, Sec3[10] = {0}, Sec4[10] = {0};
int Sec1_temp = 0, Sec2_temp = 0, Sec3_temp = 0, Sec4_temp = 0;
LenName=GetDlgItemText(IDC_NAME, reaName, 64);

if (LenName < 1)
{
    MessageBox("----- Your name atleast 1 chart -----","Hey !! Please
input your name again !! ");
}
else
{
    Temp1 = 0x6B;
    // Caculation 0
    while (i < LenName)
    {
        Temp = reaName[i] + reaName[i] * 0x4;
        Temp1 = Temp1 + Temp * 0x4;
        i++;
    }
    Sec1_temp = Temp1;
    wsprintf(Sec1,"%u-",Sec1_temp);
    i = 0;
    Temp = 0;
    Temp1 = 0x6B;
    // Calculation 1
    while (i < LenName)
    {
        Temp = reaName[i] + reaName[i] * 0x4;
        Temp2 = Temp + Temp * 0x4;

        Temp1 = Temp1 + Temp2 * 0x8;
        i++;
    }
    Sec2_temp = Temp1;
    wsprintf(Sec2,"%u-",Sec2_temp);
//Calculation 2
    i = LenName - 1 ;
    Temp = 0;
    Temp = reaName[i] + 0x2;
    Sec3_temp = Temp;
    wsprintf(Sec3,"%u-",Sec3_temp);
//Calculation 3
    i = LenName - 1;
    Temp = 0;
    Temp = reaName[i] + reaName[i] * 0x4;
    Temp = Temp + Temp * 0x4;
    Temp = Temp * 0x4 + 1;
    Sec4_temp = Temp;
    wsprintf(Sec4,"%u",Sec4_temp);
}

```

```

    }
// Link all SectionX to creat Real Serial
lstrcat (reaSerial,Sec1);
lstrcat (reaSerial,Sec2);
lstrcat (reaSerial,Sec3);
lstrcat (reaSerial,Sec4);
SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV – End of Tut :

- Finished – *October 13, 2004*

Reverse Engineering Association SoftWare

Homepage	:	http://www.sothink.com
Production	:	SourceTec Sofware Co.Ltd
SoftWare	:	Sothink SlidingMenu 2.0
Copyright by	:	Copyright (C) 1998 SourceTec Software Co.Ltd . All Rights Reserved.
Type	:	Email/Serial
Packed	:	Not
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack	:	N/A
Request	:	Correct Serial

Sothink SlidingMenu 2.0

SlidingMenu is a powerful Java Applet program used to design web navigational menu system. Its graphical interfaces guide you to complete the creation, previewing and inserting with ease but without coding. By using this tool, you can design each menu and its sub-menu(s)' text, image, linked URL, target frame, style, three status (normal, mouse over, click) of items and many other parameters. Profuse effects of color, text and background make the menu more charming.

In addition, you can use the browsers installed in the system to preview the effect momentarily....

I – Information :

- **** Dùng **PEiD 0.92** detect , chúng ta biết được chương trình này không bị PACK , và nó được Code bằng **Microsoft Visual C++ 6.0** .

- **** Khi chạy chương trình chúng ta sẽ nhận được cái Nag screen thông báo cho chúng ta biết rằng nếu chúng ta không bỏ tiền ra để đăng ký hợp pháp thì sẽ chỉ được sử dụng nó trong vòng 30 ngày là hết hạn kí hợp đồng hehe. Vậy thì chúng ta phải crack thôi.

- ***** Ngay tại cái Nag screen này , chúng ta thấy có một button là **Register** , à chắc là chỗ này bắt chúng nhập thông tin đăng ký đây. Bấm vào đây, chúng ta sẽ chuyển sang một screen mới. Cụ thể là bắt chúng ta nhập **E-mail** và **Register Code** . Nếu bác nào đăng ký hợp pháp rồi thì cứ việc nhập vô, Còn tui trong tay chẳng có gì ngoài OllyDbg hehe. Thôi thì , cứ nhập đai một FU và FS thử xem (**E-mail** : **kienbigmummy@yahoo.com ; Register Code : 11111982**) , rồi nhấn Register. Bạn sẽ không phải chờ thêm một tí tắc nào nữa, cái Nag **Sorry, That is not a valid register code** đập ngay vào mặt. Ok , chúng

ta Load chương trình này trong Olly , và tìm chuỗi này .Ặc ặc , nhưng chúng ta không thể tìm thấy trong Ollydbg cũng như trong Windasm .(Về sau các bạn sẽ thấy chuỗi này trong file .dll là **PageletSlidingMenu.dll** ----> sẽ được nói ngay ở dưới đây)

- **** Oạch thế này thì chúng ta phải bó tay hay sao . Cứ bình tĩnh mình sẽ chỉ cho các bạn cách mà bác **Moonbaby** đã chỉ cho mình nhé.Chúng ta sẽ áp dụng phương pháp **STACK** để kill chương trình này. Ok ,Now Here we go !!!...

II – Cracking :

- **** Load và chạy chương trình này trong OllyDbg , nhấn F9 để Run chương trình . Chúng ta nhập **Fake Email** và **Fake Register Code** . Sau đó nhấn Register , chương trình sẽ bắn Nag : “**Sorry, That is not a valid register code**” .

- **** Ok , giữ nguyên chương trình , chúng ta quay trở lại Ollydbg nhấn F12 để **Pause** chương trình lại , sau đó bạn nhấn tiếp **ALT + K** để mở cửa sổ “**Call Stack of Main Thread**”. Tại cửa sổ này chúng ta sẽ nhìn thấy những thông tin như sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012DB9C	77D43FBE	Includes 7FFE0304	USER32.77D43FBC	0012DBD0
0012DBA0	77D487A7	USER32.WaitMessage	USER32.77D487A2	0012DBD0
0012DBD4	77D4F58C	USER32.77D48607	USER32.77D4F587	0012DBD0
0012DBFC	77D6AAAE	USER32.77D4F4D8	USER32.77D6AAA9	0012DBF8
0012DEB4	77D6AC40	? USER32.SoftModalMessageBox		
USER32.77D6AC3B	0012DE3C			
0012DF88	77F51898	ntdll.77F5189D	ntdll.77F51893	0012DF84
0012DFA8	77F54125	? ntdll.RtlImageDirectoryEntryToData	ntdll.77F54120	0012DFA4

- **** Ở đây chúng ta chú ý đến dòng sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012DEB4	77D6AC40	? USER32.SoftModalMessageBox	USER32.77D6AC3B	0012DE3C

- **** Ok , tại dòng trên chúng ta Double Click tại cột **Called from** , chúng ta sẽ quay trở lại chương trình chính , chúng ta sẽ ở đây :

```
77D6AC3B E8 09F9FFFF CALL USER32.SoftModalMessageBox <== We're here
77D6AC40 66:837E 2C 00 CMP WORD PTR DS:[ESI+2C],0 <== Set break point here
77D6AC45 8BF8      MOV EDI,EAX
77D6AC47 0F85 F0840100 JNZ USER32.77D8313D
77D6AC4D 8BC7      MOV EAX,EDI
77D6AC4F 5F        POP EDI
77D6AC50 5E        POP ESI
77D6AC51 5B        POP EBX
77D6AC52 83C5 74   ADD EBP,74
77D6AC55 C9        LEAVE
77D6AC56 C2 0400   RETN 4
```

- **** Sau khi đặt Break Point , nhấn F8 để chương trình hiện Nag :” **Sorry, That is not a valid register code**” . Nhấn Ok chúng ta sẽ dừng lại tại điểm mà chúng ta đã đặt BP. Xóa điểm đặt BP này đi , chúng ta sẽ Trace qua một số lần (ở chương trình này là 8 lần) **RETN** chúng ta sẽ đến đây :

```
10015627 E8 44B1FFFF CALL PAGELE~1.10010770
1001562C 83C4 0C   ADD ESP,0C
```

```

1001562F 50      PUSH EAX
10015630 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
10015634 C64424 18 01 MOV BYTE PTR SS:[ESP+18],1
10015639 E8 DB410100 CALL PAGELE~1.10029819
1001563E 8D4C24 1C  LEA ECX,DWORD PTR SS:[ESP+1C]
10015642 C64424 14 00 MOV BYTE PTR SS:[ESP+14],0
10015647 E8 94400100 CALL PAGELE~1.100296E0
1001564C 8B6C24 20  MOV EBP,DWORD PTR SS:[ESP+20]
10015650 8B5424 08  MOV EDX,DWORD PTR SS:[ESP+8]
10015654 52      PUSH EDX
10015655 8B45 00   MOV EAX,DWORD PTR SS:[EBP]
10015658 50      PUSH EAX
10015659 E8 40750000 CALL PAGELE~1.1001CB9E
1001565E 83C4 08   ADD ESP,8
10015661 85C0      TEST EAX,EAX<== EAX = 0?
10015663 0F85 80000000 JNZ PAGELE~1.100156E9 <== Không bắn Nag vào mặt
10015669 56      PUSH ESI
1001566A E8 A5E30100 CALL PAGELE~1.10033A14
1001566F 8B70 04   MOV ESI,DWORD PTR DS:[EAX+4]
10015672 68 68DE0410 PUSH PAGELE~1.1004DE68 ; ASCII "Yes"
10015677 68 78DE0410 PUSH PAGELE~1.1004DE78 ; ASCII "Registered"
1001567C 68 6CDE0410 PUSH PAGELE~1.1004DE6C ; ASCII "Register"
10015681 8BCE      MOV ECX,ESI
10015683 E8 33B70100 CALL PAGELE~1.10030DBB
10015688 8B07      MOV EAX,DWORD PTR DS:[EDI]
1001568A 8BCE      MOV ECX,ESI
1001568C 50      PUSH EAX
1001568D 68 58DE0410 PUSH PAGELE~1.1004DE58 ; ASCII "Email Address"
10015692 68 6CDE0410 PUSH PAGELE~1.1004DE6C ; ASCII "Register"
10015697 E8 1FB70100 CALL PAGELE~1.10030DBB
1001569C 8B45 00   MOV EAX,DWORD PTR SS:[EBP]
1001569F 8BCE      MOV ECX,ESI
100156A1 50      PUSH EAX
100156A2 68 48DE0410 PUSH PAGELE~1.1004DE48 ; ASCII "Register Code"
100156A7 68 6CDE0410 PUSH PAGELE~1.1004DE6C ; ASCII "Register"
100156AC E8 0AB70100 CALL PAGELE~1.10030DBB
100156B1 6A FF      PUSH -1
100156B3 6A 00      PUSH 0
100156B5 68 D7000000 PUSH 0D7
100156BA E8 E9B80100 CALL PAGELE~1.10030FA8
100156BF 8D4C24 0C  LEA ECX,DWORD PTR SS:[ESP+C]
100156C3 C74424 18 FFFF>MOV DWORD PTR SS:[ESP+18],-1
100156CB E8 10400100 CALL PAGELE~1.100296E0
100156D0 5E      POP ESI
100156D1 5F      POP EDI
100156D2 B8 01000000 MOV EAX,1
100156D7 5D      POP EBP
100156D8 8B4C24 04  MOV ECX,DWORD PTR SS:[ESP+4]
100156DC 64:890D 00000000>MOV DWORD PTR FS:[0],ECX
100156E3 83C4 10   ADD ESP,10
100156E6 C2 0800   RETN 8
100156E9 6A FF      PUSH -1

```

```

100156EB 6A 00      PUSH 0
100156ED 68 0A010000 PUSH 10A
100156F2 E8 B1B80100 CALL PAGELE~1.10030FA8
100156F7 8D4C24 08  LEA ECX,DWORD PTR SS:[ESP+8] <== After RETN 8 times

```

Note : Tại đây các bạn nhìn lên thanh Title Bar của Olly , các bạn sẽ thấy những thông tin như sau :

OllyDbg – SothingSlidingMenu.exe – [CPU – main thread , module PAGELE~1]

ke ke như các bạn thấy chương trình sẽ gọi module trong file **PageletSlidingMenu.dll** để kiểm tra các thông tin mà chúng ta nhập vào.

- **** Đến đây là thành công rồi đó các bạn , Nhìn xuống cửa sổ **Stack window** các bạn sẽ thấy được Right Register code .Ví dụ tại cửa sổ Stack Window của mình như sau :

stack window:

0012E248 00AA4388 ASCII "5B7187050FBB5BE4" <== Right Code

0012E24C 0012E388 Pointer to next SEH record

0012E250 1003BB00 SE handler

/*/*/* - SERIAL tương ứng :

User : kienbigmummy@yahoo.com Serial : **5B7187050FBB5BE4**

or

User : REA-cRaCkErTeAm@yahoo.com Serial : **5D1CA69EFCD771DA**

IV – End of Tut :

- Finished – **October 3, 2004**

Reverse Engineering Association SoftWare

Homepage :	http://www.sothink.com
Production :	SourceTec Sofware Co.Ltd
SoftWare :	Sothink CoolMenu v3.0
Copyright by :	Copyright (C) 1998 SourceTec Software Co.Ltd . All Rights Reserved.
Type :	Email/Serial
Packed :	Not
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial

Sothink CoolMenu v3.0

Sothink CoolMenu is a flexible menu tool to create pop-up Java menu. Using CoolMenu, users can create unlimited menu items at their pleasure without knowing HTML or Java codes.

It provides two ways to create a new effect so that users can select a well-designed template to begin the configuration as before or exert their imagination to create a new one from blank. The newly added preview window is convenient for users to observe each tiny change of the menu during the work. A tabbed interface is available for users to set every conceivable parameter for the menu. It also enables

users to insert image and sound to their menus for the most ideal effect. In addition, the configured cool menu can be added into pages as easily as mouse clicking.

It supports working with FrontPage as well as working with Sothink HTML Editor (previous name is CutePage) perfectly

I – Information :

- **** Dùng **PEiD 0.92** detect , chúng ta biết được chương trình này không bị PACK , và nó được Code bằng **Microsoft Visual C++ 6.0** .

- **** Khi chạy chương trình chúng ta sẽ nhận được cái Nag screen thông báo cho chúng ta biết rằng nếu chúng ta không bỏ tiền ra để đăng ký hợp pháp thì sẽ chỉ được sử dụng nó trong vòng 30 ngày là hết hạn kí hợp đồng hehe. Vậy thì chúng ta phải crack thôi.(Giống hệt các Soft MenuWizard, CoolTextWizard của hãng này).

- ***** Ngay tại cái Nag screen này , chúng ta thấy có một button là **Register** , à chắc là chỗ này bắt chúng nhập thông tin đăng ký đây. Bấm vào đây, chúng ta sẽ chuyển sang một screen mới. Cụ thể là bắt chúng ta nhập **E-mail** và **Register Code** . Nếu bác nào đăng ký hợp pháp rồi thì cứ việc nhập vô, Còn tui trong tay chẳng có gì ngoài OllyDbg hehe. Thôi thì , cứ nhập đai một FU và FS thử xem (**E-mail** : **kienbigmummy@yahoo.com ; Register Code : 11111982**) , rồi nhấn Register. Bạn sẽ không phải chờ thêm một tí tắc nào nữa, cái Nag **Sorry, That is not a valid register code** đập ngay vào mặt. Ok , chúng ta Load chương trình này trong Olly , và tìm chuỗi này .Ặc ặc , nhưng chúng ta không thể tìm thấy trong Ollydbg cũng như trong Windasm . Vậy chúng ta phải làm sao đây , ở đây nếu các bạn tìm hiểu kĩ sẽ thấy cái chuỗi trên sẽ nằm trong một file **dll (PageletCoolMenu.dll)**. Khi Load file này lên trong Olly1.10 bạn sẽ tìm thấy nó ở địa chỉ sau :

1001ABFD |. 68 84B30510 PUSH PageletC.1005B384 ;|Arg1 = 1005B384 ASCII "Sorry, that is not a valid register code."

- **** Ouch thế này thì chúng ta phải bó tay hay sao (vì chúng ta chưa crack *.dll bao giờ) . Cứ bình tĩnh mình sẽ chỉ cho các bạn cách mà anh **Moonbaby** đã chỉ cho mình nhé.Chúng ta sẽ áp dụng phương pháp **STACK** để kill chương trình này. Ok , Let's begin...

II – Cracking :

- **** Load và chạy chương trình này trong OllyDbg , nhấn F9 để Run chương trình . Chúng ta nhập **Fake Email** và **Fake Register Code** . Sau đó nhấn Register , chương trình sẽ bắn Nag : "**Sorry, That is not a valid register code**" .

- **** Ok , giữ nguyên chương trình , chúng ta quay trở lại OllyDbg nhấn F12 để **Pause** chương trình lại , sau đó bạn nhấn tiếp **ALT + K** để mở cửa sổ "**Call Stack of Main Thread**". Tại cửa sổ này chúng ta sẽ nhìn thấy những thông tin như sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012DC14	77D43FBE	Includes 7FFE0304	USER32.77D43FBC	0012DC48
0012DC18	77D487A7	USER32.WaitMessage	USER32.77D487A2	0012DC48
0012DC4C	77D4F58C	USER32.77D48607	USER32.77D4F587	0012DC48
0012DC74	77D6AAAE	USER32.77D4F4D8	USER32.77D6AAA9	0012DC70
0012DF2C	77D6AC40	? USER32.SoftModalMessageBox	USER32.77D6AC3B	0012DEB4
0012E074	77D6ADCC	? USER32.77D6AB06	USER32.77D6ADC7	0012DFFC
0012E0C8	77D6AE8A	USER32.MessageBoxTimeoutW	USER32.77D6AE85	0012E0C4
0012E0FC	77D6AE17	? USER32.MessageBoxTimeoutA	USER32.77D6AE12	0012E0F8
0012E11C	77D6ADFB	? USER32.MessageBoxExA	USER32.77D6ADF6	0012E118
0012E120	00180230	hOwner = 00180230 ('Register Sothink Cool		

```

0012E124 1005B384 Text = "Sorry, that is not a valid regist
0012E128 00A73D10 Title = "Sothink CoolMenu"
0012E12C 00000030 Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLM
0012E130 00000000 LanguageID = 0 (LANG_NEUTRAL)
0012E134 1003A785 ? USER32.MessageBoxA PAGELE~1.1003A77F
0012E138 00180230 hOwner = 00180230 ('Register Sothink Cool
0012E13C 1005B384 Text = "Sorry, that is not a valid regist
0012E140 00A73D10 Title = "Sothink CoolMenu"
0012E144 00000030 Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLM

```

- **** Ở đây chúng ta chú ý đến dòng sau :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from
0012E134	1003A785	? USER32.MessageBoxA	PAGELE~1.1003A77F

Note : Hì hì chõ này chính là chõ mà chương trình gọi tới cái file .dll để bắn ra Nag Screen khi các bạn nhập sai đầy..keke.

- **** Ok , tại dòng trên chúng ta Double Click tại cột **Called from** , chúng ta sẽ quay trở lại chương trình chính , chúng ta sẽ ở đây :

```

1003A77F FF15 F0830410 CALL DWORD PTR DS:[<&USER32.MessageBoxA>];
USER32.MessageBoxA <== We're here
1003A785 85F6 TEST ESI,ESI <== Set break point here
1003A787 8BF8 MOV EDI,EAX
1003A789 74 05 JE SHORT PAGELE~1.1003A790
1003A78B 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
1003A78E 8906 MOV DWORD PTR DS:[ESI],EAX
1003A790 837D FC 00 CMP DWORD PTR SS:[EBP-4],0
1003A794 74 0B JE SHORT PAGELE~1.1003A7A1

```

- **** Sau khi đặt Break Point , nhấn F8 để chương trình hiện Nag :" **Sorry, That is not a valid register code**" . Nhấn Ok chúng ta sẽ dừng lại tại điểm mà chúng ta đã đặt BP. Xóa điểm đặt BP này đi , chúng ta sẽ Trace qua một số lần (ở chương trình này là 2 lần) **RETN** chúng ta sẽ đến đây :

1001AB37 E8 B49BFFFF CALL PAGELE~1.100146F0 <== Encrypt

```

1001AB3C 83C4 0C ADD ESP,0C
1001AB3F 50 PUSH EAX
1001AB40 8D4C24 0C LEA ECX,DWORD PTR SS:[ESP+C]
1001AB44 C64424 18 01 MOV BYTE PTR SS:[ESP+18],1
1001AB49 E8 527F0100 CALL PAGELE~1.10032AA0
1001AB4E 8D4C24 1C LEA ECX,DWORD PTR SS:[ESP+1C]
1001AB52 C64424 14 00 MOV BYTE PTR SS:[ESP+14],0
1001AB57 E8 0B7E0100 CALL PAGELE~1.10032967
1001AB5C 8B6C24 20 MOV EBP,DWORD PTR SS:[ESP+20]
1001AB60 8B5424 08 MOV EDX,DWORD PTR SS:[ESP+8]
1001AB64 52 PUSH EDX
1001AB65 8B45 00 MOV EAX,DWORD PTR SS:[EBP]
1001AB68 50 PUSH EAX
1001AB69 E8 B1AA0000 CALL PAGELE~1.1002561F
1001AB6E 83C4 08 ADD ESP,8
1001AB71 85C0 TEST EAX,EAX
1001AB73 OF85 80000000 JNZ PAGELE~1.1001ABF9

```

```

1001AB79 56      PUSH ESI
1001AB7A E8 19290200 CALL PAGELE~1.1003D498
1001AB7F 8B70 04    MOV ESI,DWORD PTR DS:[EAX+4]
1001AB82 68 60B30510 PUSH PAGELE~1.1005B360 ; ASCII "Yes"
1001AB87 68 70B30510 PUSH PAGELE~1.1005B370 ; ASCII "Registered"
1001AB8C 68 64B30510 PUSH PAGELE~1.1005B364 ; ASCII "Register"
1001AB91 8BCE      MOV ECX,ESI
1001AB93 E8 67FA0100 CALL PAGELE~1.1003A5FF
1001AB98 8B07      MOV EAX,DWORD PTR DS:[EDI]
1001AB9A 8BCE      MOV ECX,ESI
1001AB9C 50      PUSH EAX
1001AB9D 68 50B30510 PUSH PAGELE~1.1005B350 ; ASCII "Email Address"
1001ABA2 68 64B30510 PUSH PAGELE~1.1005B364 ; ASCII "Register"
1001ABA7 E8 53FA0100 CALL PAGELE~1.1003A5FF
1001ABAC 8B45 00    MOV EAX,DWORD PTR SS:[EBP]
1001ABAF 8BCE      MOV ECX,ESI
1001ABB1 50      PUSH EAX
1001ABB2 68 40B30510 PUSH PAGELE~1.1005B340 ; ASCII "Register Code"
1001ABB7 68 64B30510 PUSH PAGELE~1.1005B364 ; ASCII "Register"
1001ABBC E8 3EFA0100 CALL PAGELE~1.1003A5FF
1001ABC1 6A 00      PUSH 0
1001ABC3 6A 00      PUSH 0
1001ABC5 68 B0B30510 PUSH PAGELE~1.1005B3B0 ; ASCII " Thank you ! Please restart"
1001ABCA E8 E5FB0100 CALL PAGELE~1.1003A7B4
1001ABCF 8D4C24 0C    LEA ECX,DWORD PTR SS:[ESP+C]
1001ABD3 C74424 18 FFFF>MOV DWORD PTR SS:[ESP+18],-1
1001ABDB E8 877D0100 CALL PAGELE~1.10032967
1001ABE0 5E      POP ESI
1001ABE1 5F      POP EDI
1001ABE2 B8 01000000 MOV EAX,1
1001ABE7 5D      POP EBP
1001ABE8 8B4C24 04    MOV ECX,DWORD PTR SS:[ESP+4]
1001ABEC 64:890D 0000000>MOV DWORD PTR FS:[0],ECX
1001ABF3 83C4 10    ADD ESP,10
1001ABF6 C2 0800    RETN 8
1001ABF9 6A 00      PUSH 0
1001ABFB 6A 00      PUSH 0
1001ABFD 68 84B30510 PUSH PAGELE~1.1005B384 ; ASCII "Sorry, that is not a valid register
code."
1001AC02 E8 ADFB0100 CALL PAGELE~1.1003A7B4
1001AC07 8D4C24 08    LEA ECX,DWORD PTR SS:[ESP+8] <==Sau 2 lần RETN

```

- **** Đến đây là thành công rồi đó các bạn , Nhìn xuống cửa sổ **Stack window** các bạn sẽ thấy được Right Register code .Ví dụ tại cửa sổ Stack Window của mình như sau :

stack window:

0012E298 00A742C0 ASCII "FCEEA79A3B4A5070" <== This is Right Code

0012E29C 0012E3D8 Pointer to next SEH record

0012E2A0 10046C00 SE handler

/*/*/* - SERIAL tương ứng :

User : kienbigmummy@yahoo.com

Serial : FCEEA79A3B4A5070

or

IV – End of Tut :- Finished – *October 3, 2004*

Reverse Engineering Association SoftWare

Homepage :	http://www.coffeecup.com
Production :	CoffeeCup Software ,Inc
SoftWare :	CoffeeCup Firestarter v6.5
Copyright by :	Copyright (C) 2001 - 2003 CoffeeCup Software ,Inc. All Rights Reserved.
Type :	Register Online / Trial Time
Packed :	Not
Language :	Borland Delphi 4.0 – 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Patch to use over 30 days

I – Information :

-**** Dùng **PEiD v0.92** để detect , chúng ta biết được chương trình này không bị Pack và nó được Code bằng **Borland Delphi 4.0 - 5.0**

- **** Sau khi cài đặt chương trình , các bạn hãy Run chương trình này lên . Sau khi run chương trình chúng ta sẽ nhận được một Nag Screen thông báo về version của chương trình và hai button chính sau : **Oder Now** và **Open FireStarter** . Cái **Oder Now** bắt chúng ta vào Website của hãng để mua chính thức , còn button **Open FireStarter** cho phép chúng ta dùng thử các tính năng của chương trình sau đó nếu thấy hay thì các bạn có thể đăng ký để mua nó. Vì chưa biết nó như thế nào , nên chúng ta quyết định click vào **Open FireStarter** để xem thử chương trình hay tới mức nào mà bắt chúng ta mua.

- **** Khi sau khi vào , chúng ta thấy giao diện của chương trình cũng bắt mắt đầy chừ. Quan sát một lúc mình để ý thấy có một button như sau trên thanh Toolbar của chương trình : **"\$" (Click here to Register)** . Hả hả đây rồi , chắc đây là chỗ bắt chúng ta đăng kí (có thể là nhập Serial đây!!!) . Ăc ặc nhưng khi click vào thì hóa ra là nó cũng bắt chúng ta đăng kí để mua nó thôi à (chương trình này bắt chẹt End-User như chúng ta quá).

- **** Ăc ặc , không có gì để nhập vào thì chúng ta đành thoát ra vậy , Close chương trình lại . Bùm bùm , lại thêm cái Nag nữa bắn ra . Lần này nó cho chúng ta biết trị giá của chương trình là **\$49.00** lận.Thật là bức mình thằng này không từ thủ đoạn nào để quảng cáo cho chương trình và bắt chúng ta phải mua nó .(Nhưng tiền đãt thẻ thì mình chịu)

-**** Đến đây , có thể các bạn sẽ hỏi thế thì làm sao để crack chương trình này (ăc ặc để crack được Right Serial thì mình chịu a`). Các bạn có để ý thấy phần giới thiệu chương trình này không (**Request : Patch to use over 30 days trial**) , các bạn cũng có thể hỏi tại sao mình lại biết là chương trình này nó chỉ cho phép sử dụng 30 ngày , xin thưa đây chỉ là phán đoán thôi. Mình đã chỉnh ngày hệ thống để cho nó quá 30 ngày , sau đó Run lại chương trình chúng ta sẽ nhận được thông báo sau : **"Your 30 days trial period has expired. Please order the full version now!"**

Click “Yes” thì lại nhảy vào trang Web của nó , mà Click “No” thì hối ôi chương trình nó chạy đâu mất tiêu à.

-**** Oki , bây giờ chúng ta đã có cơ sở để tiến hành crack chương trình . Load chương trình lên trong Ollydbg , chúng ta tìm thấy chuỗi trên tại địa chỉ sau :

0054C8E2 . B8 78C95400 MOV EAX,FireStar.0054C978 ;|ASCII "Your 30 day trial period has expired. Please order the full version now."

-**** Lần ngược lên trên một chút , ta Set Break Point tại đây :

0054C8CE . E8 DD84FEFF CALL FireStar.00534DB0 <== We set BP here

II – Cracking :

- **** Nhấn F9 để Run chương trình , cái Nag hiện lên chúng ta nhấn vào nút **Open Firestarter** . Chúng ta sẽ quay trở lại Olly và Ice tại chỗ chúng ta set BP :

```
0054C8CE . E8 DD84FEFF CALL FireStar.00534DB0 ;<== We're here
0054C8D3 . 84C0      TEST AL,AL          ;<== AL = 0?
0054C8D5 . 74 46     JE SHORT FireStar.0054C91D ;<== Jump out of Nag
0054C8D7 . 6A 00     PUSH 0             ;/Arg1 = 00000000
0054C8D9 . 66:8B0D 6CC95>MOV CX,WORD PTR DS:[54C96C]   ;|
0054C8E0 . B2 03     MOV DL,3           ;|
0054C8E2 . B8 78C95400 MOV EAX,FireStar.0054C978 ;|ASCII "Your 30 day trial period has
expired. Please order the full version now."
```

0054C8E7 . E8 28F6F0FF CALL FireStar.0045BF14 ;\FireStar.0045BF14<== Bắn Nag nếu AL <> 0

-**** Chúng ta sẽ xem xem AL sẽ nhận giá trị từ đâu để thực hiện lệnh TEST ở trên. Vậy chúng ta sẽ Trace Into vào trong hàm Call mà chúng ta đã đặt BP ở trên

0054C8CE . E8 DD84FEFF CALL FireStar.00534DB0<== Trace into

----- Trace Into -----

```
00534DB0 /$ 55      PUSH EBP
00534DB1 |. 8BEC      MOV EBP,ESP
00534DB3 |. 51       PUSH ECX
00534DB4 |. E8 F7FCFFFF CALL FireStar.00534AB0
00534DB9 |. A0 B8DE5800 MOV AL,BYTE PTR DS:[58DEB8] <== AL nhận giá trị của
DS:[58DEB8]
00534DBE |. 8845 FF    MOV BYTE PTR SS:[EBP-1],AL
00534DC1 |. 8A45 FF    MOV AL,BYTE PTR SS:[EBP-1]
00534DC4 |. 59       POP ECX
00534DC5 |. 5D       POP EBP
00534DC6 \. C3       RETN
```

----- Trace Into -----

-**** Oki , vậy là chúng ta đã biết là chương trình này sẽ nhận giá trị của vùng nhớ [58DEB8] ----> đây có lẽ là giá trị xác định xem chương trình đã được sử dụng quá 30 ngày hay chưa . Nếu chưa quá 30 ngày thì nó sẽ có giá trị là 0 , còn nếu quá 30 ngày thì nó sẽ có giá trị là 1.

-**** Bây giờ trong Olly chúng ta nhấn **Ctrl + F2** để Restart lại chương trình (Note : Các bạn nhớ bỏ điểm đặt BP ở trên đi nhé) , nhấn **Ctrl + G** , sau đó gõ vào địa chỉ là **58DEB8**. Ngay tại vị trí bộ nhớ đó chúng ta set **BP Memory on Write**. Tiếp theo nhấn **F9** để Run chương trình và quan sát xem đoạn code nào sẽ ghi giá trị vào vùng nhớ này . Chúng ta có những vị trí sau :

00534C1E . C605 B8DE5800>MOV BYTE PTR DS:[58DEB8],0 <== Vị trí thứ nhất.

```

00534C25 > 803D B8DE5800>CMP BYTE PTR DS:[58DEB8],0
00534C2C . 74 0C      JE SHORT FireStar.00534C3A
00534C2E . C705 B4DE5800>MOV DWORD PTR DS:[58DEB4],2E
00534C38 . EB 34      JMP SHORT FireStar.00534C6E
00534C3A > 6A 00      PUSH 0
00534C3C . 6A 18      PUSH 18
00534C3E . 6A 00      PUSH 0
00534C40 . 6A 3C      PUSH 3C
00534C42 . 6A 00      PUSH 0
00534C44 . 6A 3C      PUSH 3C
00534C46 . D905 C44C5300 FLD DWORD PTR DS:[534CC4]
00534C4C . DC4D F0      FMUL QWORD PTR SS:[EBP-10]
00534C4F . D835 C84C5300 FDIV DWORD PTR DS:[534CC8]
00534C55 . E8 BADEECFF CALL FireStar.00402B14
00534C5A . E8 871FEDFF CALL FireStar.00406BE6
00534C5F . E8 821FEDFF CALL FireStar.00406BE6
00534C64 . E8 7D1FEDFF CALL FireStar.00406BE6
00534C69 . A3 B4DE5800 MOV DWORD PTR DS:[58DEB4],EAX
00534C6E > 833D B4DE5800>CMP DWORD PTR DS:[58DEB4],2E
00534C75 . 7C 09      JL SHORT FireStar.00534C80
00534C77 . C605 B8DE5800>MOV BYTE PTR DS:[58DEB8],1      ;<== Vị trí thứ 2
00534C7E . EB 07      JMP SHORT FireStar.00534C87
00534C80 > C605 B8DE5800>MOV BYTE PTR DS:[58DEB8],0      ;<== Vị trí thứ 3
00534C87 > 803D B8DE5800>CMP BYTE PTR DS:[58DEB8],1
00534C8E . 75 0A      JNZ SHORT FireStar.00534C9A
00534C90 . C705 C4DE5800>MOV DWORD PTR DS:[58DEC4],42
00534C9A > E8 2D000000 CALL FireStar.00534CCC

```

-**** Ví trí thứ 1 và vị trí thứ 3 chúng ta không cần quan tâm vì tại những vị trí này thì [58DEB8] sẽ mang giá trị 0 sau lệnh MOV. Vị trí mà chúng ta phải lưu ý ở đây là vị trí thứ 2 vì tại đây [58DEB8] sẽ có giá trị là 1 , sau đó giá trị này sẽ được gán cho thanh ghi AL để dùng trong lệnh Test như các bạn đã thấy ở trên , mà điều này là điều mà chúng ta không muốn vì nếu AL có giá trị là 1 thì nó sẽ bắn Nag vào mặt chúng ta đây. Do đó chỗ cần phải Patch là vị trí thứ 2. (Note : thực ra nếu các bác Trace tiếp thì câu lệnh tại vị trí thứ 2 sẽ không thực hiện đâu , nên nếu cứ Trace trong Olly cho đến đoạn **Test AL, AL** thì các bác sẽ thấy thanh ghi AL sẽ là 0 và chúng ta sẽ nhảy qua Nag ngon lành. Chương trình lại Run bình thường như chưa có gì xảy ra ke ke và chương trình lại trở về **“cái thủa ban đầu ngo’ ngác ấy”**. Nhưng hết 30 ngày thì nó lại bắn Nag à . Do đó tốt nhất chúng ta Patch luôn cho đỡ mệt.)

-**** Do đó tại vị trí thứ 2 , chúng ta Right Click vào đó , chọn Assemble . Một diaglog box hiện lên , chúng ta hãy sửa 1 thành 0 , rồi nhấn **Assemble**. Sau đó tại màn hình chính của Olly , chúng ta lại Right Click một lần nữa sau đó chọn **Copy to Executable ----> All modifications** . Tiếp theo chọn **Copy All** , một cửa sổ mới hiện ra (**Dump of file**) , tại cửa sổ này chúng ta Right Click chọn **Save File** để lưu đè lên file cũ hoặc lưu dưới một file mới.

-**** Phù , thế là xong bây giờ thoát Olly Run chương trình , chỉnh ngày quá 30 ngày , chương trình vẫn Run đều đẽu.

IV – End of Tut :

- Finished – **October 5, 2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.conceptworld.com
Production :	Conceptworld Corporation
SoftWare :	Quick Notes Plus v5.0 (Build 40)
Copyright by :	Copyright (C) 2001 - 2003 Conceptworld Corporation. All Rights Reserved.
Type :	Serial
Packed :	Not
Language :	Microsoft Visual C++ 7.0 Method2
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack :	N/A
Request :	Correct Serial / Keygen

Quick Notes Plus v5.0 (Build 40)

QNP allows you to create virtual sticky notes (Virtual Post-It Notes) on your windows desktop and memoboards. Virtual sticky notes are just post-it like notes for your computer. Jot down messages, ideas, appointments, phone numbers, birthdays or just place a popular quote in to a sticky note and stick it on the desktop. Sticky notes are right in front of your eyes on the desktop. Accessing them is very easy. The sticky notes are even resizable. You can set alarm to each sticky note. Let QNP do all the hard work of reminding you about your appointments, anniversaries or things to do.....

I – Information :

-**** Hiiiii muốn có được thông tin chi tiết về chương trình này thì trước tiên các bạn phải download nó về cái đĩa . Sau khi chúng ta đã có được nó rồi thì tiến hành Setup chương trình này vào trong ổ cứng.

-**** Dùng PeiD v0.92 để detect chúng ta biết được chương trình được tác giả code bằng **Microsoft Visual C++ 7.0 Method2** , vậy là chương trình không bị Pack :). Tiếp tục sử dụng Plugin là **Krypto Analyzer** , chúng ta biết được chương trình này có sử dụng Crypto là **BASE64 table** .

-**** Sau khi Setup chương trình này , nó sẽ tạo một Icon nhỏ nằm ở System Tray . Chúng ta right click lên biểu tượng này, chọn **Help --> Enter Registration Code....** Chương trình sẽ xuất hiện một dialog box yêu cầu chúng ta nhập thông tin đăng ký . Cụ thể là **Name , Company, Code**. Chúng ta nhập Fake Info vào xem thế nào, sau khi nhập xong nhấn **Ok** thì ngay lập tức một Nag Screen bắn ra : **“Invalid Registration Code . Make sure the Registration code is of Format XXXXX- XXXXX- XXXXX- XXXXX- XXXXX. Please copy and paste the registration code without any extra trailing spaces.”**

-**** Load chương trình này vào trong Olly , sử dụng chức năng **Search String** để tìm kiếm chuỗi trên , nhưng trong mớ bòng bong String đó chúng ta không tìm thấy chuỗi nào giống như chuỗi mà chúng ta cần tìm .

-**** Áp dụng phương pháp **Stack** chúng ta tìm được chuỗi trên tại địa chỉ sau :

```

0040670A . 8B7D EC      MOV EDI,DWORD PTR SS:[EBP-14] ;<== Nag String
0040670D . 6A 30      PUSH 30
0040670F . FF30      PUSH DWORD PTR DS:[EAX]
00406711 . 8BCE      MOV ECX,ESI
00406713 . 57      PUSH EDI

```

```
00406714 . C645 FC 02 MOV BYTE PTR SS:[EBP-4],2
00406718 . E8 43060400 CALL <JMP.&MFC70.#3890> ;<== Shot Nag to your face
```

-**** Dịch lên trên một chút chúng ta set BP tại đây:

```
004066B2 . E8 77050400 CALL <JMP.&MFC70.#3565> ; <== Set BP
```

II – Cracking :

-**** Oki giờ chúng ta đã tìm được mục tiêu , nhấn **Ctrl + F2** để Restart , nhấn **F9** để Run chương trình . Chúng ta nhập vào thông tin đăng kí với định dạng mà chương trình đã yêu cầu . Sau đó nhấn OK. Chúng ta sẽ Break tại điểm mà chúng ta đã Set BP:

```
004066B2 . E8 77050400 CALL <JMP.&MFC70.#3565> ; <== We're here(Get FS)
004066B7 . 51 PUSH ECX ; <== Push FS into Stack
004066B8 . 8D45 E8 LEA EAX,DWORD PTR SS:[EBP-18]
004066BB . 8965 E4 MOV DWORD PTR SS:[EBP-1C],ESP
004066BE . 8BCC MOV ECX,ESP
004066C0 . 50 PUSH EAX
004066C1 . E8 C8AEFFFF CALL QNPlus.0040158E
004066C6 . E8 E3020400 CALL <JMP.&MFC70.#977>
004066CB . 8B48 04 MOV ECX,DWORD PTR DS:[EAX+4]
004066CE . E8 E0B30200 CALL QNPlus.00431AB3 ; <== Calculation (Trace Into)
```

----- == Calculation == -----

```
00431AE9 |. 8B7D 08 MOV EDI,[ARG.1] ; <== FS
00431AEC |. 8B47 F4 MOV EAX,DWORD PTR DS:[EDI-C] ; <== Get Length(FS)
00431AEF |. 83F8 1D CMP EAX,1D ; <== If Length (FS) < 29 then
00431AF2 |. C645 FC 02 MOV BYTE PTR SS:[EBP-4],2
00431AF6 0F85 1A010000 JNZ QNPlus.00431C16 ; <== Jump out of Calculation
00431AFC |. 53 PUSH EBX
00431AFD |. BE A0644600 MOV ESI,QNPlus.004664A0 ; ASCII "Q50"
00431B02 |. 56 PUSH ESI ; /s => "Q50"
00431B03 |. E8 7E5C0100 CALL <JMP.&MSVCR70.strlen> ; \strlen
00431B08 |. 59 POP ECX
00431B09 |. 50 PUSH EAX
00431B0A |. 8D45 F0 LEA EAX,[LOCAL.4]
00431B0D |. 50 PUSH EAX
00431B0E |. 8D4D 08 LEA ECX,[ARG.1]
00431B11 |. E8 8A24FDFF CALL QNPlus.00403FA0 ; <== Temp = Get 3 charts of FS
00431B16 |. 50 PUSH EAX
00431B17 |. 8D4D E8 LEA ECX,[LOCAL.6]
00431B1A |. C645 FC 03 MOV BYTE PTR SS:[EBP-4],3
00431B1E |. E8 6AFCFCFF CALL QNPlus.0040178D
00431B23 |. 8B4D F0 MOV ECX,[LOCAL.4] ; <== Temp
00431B26 |. 83C1 F0 ADD ECX,-10
00431B29 |. C645 FC 02 MOV BYTE PTR SS:[EBP-4],2
00431B2D |. E8 63F6FCFF CALL QNPlus.00401195
00431B32 |. 8B1D 2CEC4400 MOV EBX,DWORD PTR DS:[<&MSVCR70._mbscmp>] ; MSVCR70._mbscmp
00431B38 |. 56 PUSH ESI ; /s2 => "Q50"
00431B39 |. FF75 E8 PUSH [LOCAL.6] ; |s1 => Temp
00431B3C |. FFD3 CALL EBX ; \_mbscmp
```

```

00431B3E |. 85C0      TEST EAX,EAX          ; <== If Temp = "Q50" then
00431B40 |. 59        POP ECX
00431B41 |. 59        POP ECX
00431B42 |. 74 07      JE SHORT QNPlus.00431B4B ; <== Continue
00431B44 |> 33F6      XOR ESI,ESI
00431B46 |. E9 CA000000 JMP QNPlus.00431C15 ; <== Else Jump out of Calculation
00431B4B |> 68 98644600 PUSH QNPlus.00466498 ; /s = "Q50UL"
00431B50 |. E8 315C0100 CALL <JMP.&MSVCR70.strlen> ; \strlen
00431B55 |. 59        POP ECX
00431B56 |. 6A 1D      PUSH 1D
00431B58 |. 59        POP ECX
00431B59 |. 2BC8      SUB ECX,EAX
00431B5B |. 51        PUSH ECX
00431B5C |. 8D45 F0    LEA EAX,[LOCAL.4]
00431B5F |. 50        PUSH EAX
00431B60 |. 8D4D 08    LEA ECX,[ARG.1]
00431B63 |. E8 1AC6FDFF CALL QNPlus.0040E182 ;

```

/ Note : Hàm Call này thực hiện quá trình loại bỏ 5 kí tự đầu tiên của chuỗi Registration Code. Cụ thể là như sau nếu Registration của chúng ta nhập vào là : XXXXX-XXXXX-XXXXX-XXXXX-XXXXX , thì sau hàm Call chuỗi này sẽ như sau : -XXXXX-XXXXX-XXXXX-XXXXX. Ta gọi chuỗi này là CutString*/*

```

00431B68 |. 50        PUSH EAX
00431B69 |. 8D4D 08    LEA ECX,[ARG.1]
00431B6C |. C645 FC 04  MOV BYTE PTR SS:[EBP-4],4
00431B70 |. E8 18FCFCFF CALL QNPlus.0040178D
00431B75 |. 8B4D F0    MOV ECX,[LOCAL.4]
00431B78 |. 83C1 F0    ADD ECX,-10
00431B7B |. C645 FC 02  MOV BYTE PTR SS:[EBP-4],2
00431B7F |. E8 11F6FCFF CALL QNPlus.00401195
00431B84 |. 6A 2D      PUSH 2D          ; /Arg1 = 0000002D
00431B86 |. 8D4D 08    LEA ECX,[ARG.1]   ; |
00431B89 |. E8 9B23FDFF CALL QNPlus.00403F29 ; \QNPlus.00403F29

```

/ Note : Hàm Call này thực hiện việc loại bỏ các kí tự “-“ nằm trong chuỗi CutString , nó chỉ giữ lại các kí tự X mà thôi. Cụ thể là với chuỗi CutString trên : -XXXXX-XXXXX-XXXXX-XXXXX. Sau hàm Call này sẽ là như sau : XXXXXXXXXXXXXXXXXXXX. Ta gọi chuỗi này là TString */*

```

00431B8E |. 8B7D 08    MOV EDI,[ARG.1]      ; <== TString
00431B91 |. 837F F4 14  CMP DWORD PTR DS:[EDI-C],14 ; <== If Length(TmpString) <> 20
00431B95 |.^ 75 AD     JNZ SHORT QNPlus.00431B44 ; <== Jump out of Calculation
00431B97 |. 8BF7      MOV ESI,EDI          ; <== TString
00431B99 |. C745 F0 A8644>MOV [LOCAL.4],QNPlus.004664A8
00431BA0 |> 8B45 F0    /MOV EAX,[LOCAL.4]
00431BA3 |. 8338 00    |CMP DWORD PTR DS:[EAX],0
00431BA6 |. 0FBE06    |MOVSX EAX,BYTE PTR DS:[ESI] ; <== TString[i]
00431BA9 |. 50        |PUSH EAX           ; /c   ; <== TString[i]
00431BAA |. 75 17      |JNZ SHORT QNPlus.00431BC3 ; |
00431BAC |. FF15 ACEB4400 |CALL DWORD PTR DS:<&MSVCR70._ismbclower>; \_ismbclower
00431BB2 |. 85C0      |TEST EAX,EAX          ; <== If TString[i] is a lower chart then
00431BB4 |. 59        |POP ECX
00431BB5 |.^ 75 8D     |JNZ SHORT QNPlus.00431B44 ; <== Jump out of Calculation
00431BB7 |. 0FBE06    |MOVSX EAX,BYTE PTR DS:[ESI]
00431BBA |. 50        |PUSH EAX           ; /c

```

```

00431BBB |. FF15 B0EB4400 |CALL DWORD PTR DS:[<&MSVCR70._ismbcalpha>]; \_ismbcalpha
;<== Else TString[i] is Alpha
00431BC1 |. EB 06      |JMP SHORT QNPlus.00431BC9      ;<== then Continue
00431BC3 > FF15 B4EB4400 |CALL DWORD PTR DS:[<&MSVCR70._ismbcdigit>] ; \_ismbcdigit
00431BC9 > 85C0      |TEST EAX,EAX
00431BCB |. 59      |POP ECX
00431BCC |.^ 0F84 72FFFFFF |JE QNPlus.00431B44
00431BD2 |. 8345 F0 04 |ADD [LOCAL.4],4
00431BD6 |. 46      |INC ESI           ;<== i ++
00431BD7 |. 817D F0 F8644>|CMP [LOCAL.4],QNPlus.004664F8
00431BDE |.^ 7C C0      |JL SHORT QNPlus.00431BA0

```

/* Note : Tóm lại quá trình kiểm tra trên như sau : Các kí tự trong chuỗi TString phải là kí tự và chữ số . Nếu là kí tự thì bắt buộc phải là kí tự đó phải là chữ hoa mà không được là chữ thường. Và chuỗi TString này phải tuân theo qui luật sau : **XIXIXX11XXX11XXXIXIX** (trong đó X đại diện cho một kí tự là chữ hoa , còn một đại diện cho một kí tự là chữ số). Từ đó ta suy ra chuỗi CutString sẽ phải có dạng sau đây : **-XIXIX-X11XX-X11XX-XIXIX**. Nếu không tuân theo quy luật này thì sẽ thoát khỏi đoạn Calculation ngay */

```

00431BE0 |. 68 386F4500 PUSH QNPlus.00456F38 ; ASCII "M7P8C-QR13D-ED35F-J7R3X-
LU05Q"          ;<== DefaultString
00431BE5 |. 8D4D EC    LEA ECX,[LOCAL.5]
00431BE8 |. E8 A503FDFF CALL QNPlus.00401F92
00431BED |. 8D4D EC    LEA ECX,[LOCAL.5]
00431BF0 |. C645 FC 05 MOV BYTE PTR SS:[EBP-4],5
00431BF4 |. E8 22FAFFFF CALL QNPlus.0043161B ;<== ReverseString = Invert DefaultString
00431BF9 |. FF75 EC    PUSH [LOCAL.5]       ;<== ReverseString
00431BFC |. FF75 E4    PUSH [LOCAL.7]       ;<== FakeSerial
00431BFF |. FFD3      CALL EBX            ;<== Compare
00431C01 |. 59      POP ECX
00431C02 |. 33F6      XOR ESI,ESI        ;<== ESI = 0
00431C04 |. 85C0      TEST EAX,EAX       ;<== If FakeSerial = ReverseString then
00431C06 |. 59      POP ECX
00431C07 |. 74 01      JE SHORT QNPlus.00431C0A ;<== ESI = 0
00431C09 |. 46      INC ESI           ;<== Else ESI = 1 (very important)
00431C0A > 8B4D EC    MOV ECX,[LOCAL.5]
00431C0D |. 83C1 F0    ADD ECX,-10
00431C10 |. E8 80F5FCFF CALL QNPlus.00401195
00431C15 > 5B      POP EBX
00431C16 > 8B4D E4    MOV ECX,[LOCAL.7]
00431C19 |. 83C1 F0    ADD ECX,-10
00431C1C |. E8 74F5FCFF CALL QNPlus.00401195
00431C21 |. 8B4D E8    MOV ECX,[LOCAL.6]
00431C24 |. 83C1 F0    ADD ECX,-10
00431C27 |. E8 69F5FCFF CALL QNPlus.00401195
00431C2C |. 8D4F F0    LEA ECX,DWORD PTR DS:[EDI-10]
00431C2F |. E8 61F5FCFF CALL QNPlus.00401195
00431C34 |. 8B4D F4    MOV ECX,[LOCAL.3]
00431C37 |. 5F      POP EDI
00431C38 |. 8BC6      MOV EAX,ESI        ;<== EAX = ESI
00431C3A |. 5E      POP ESI
00431C3B |. 64:890D 00000>MOV DWORD PTR FS:[0],ECX

```

```
00431C42 ]. C9      LEAVE
00431C43 \. C2 0400  RETN 4
```

===== Calculation =====

```
004066D3 . 85C0    TEST EAX,EAX          ; <== If EAX <> 0 then
004066D5 . 75 59   JNZ SHORT QNPlus.00406730 ; <== Jump out of Shot Nag
004066D7 . E8 C2040400 CALL <JMP.&MFC70.#982>
004066DC . 8B10    MOV EDX,DWORD PTR DS:[EAX]
004066DE . 8BC8    MOV ECX,EAX
004066E0 . FF52 0C  CALL DWORD PTR DS:[EDX+C]
004066E3 . 83C0 10  ADD EAX,10
004066E6 . 8945 EC  MOV DWORD PTR SS:[EBP-14],EAX
004066E9 . 6A 3D    PUSH 3D
004066EB . 8D4D EC  LEA ECX,DWORD PTR SS:[EBP-14]
004066EE . C645 FC 01  MOV BYTE PTR SS:[EBP-4],1
004066F2 . E8 76B0FFFF CALL QNPlus.0040176D
004066F7 . E8 B2020400 CALL <JMP.&MFC70.#977>
004066FC . 8B40 04  MOV EAX,DWORD PTR DS:[EAX+4]
004066FF . 8D4D E4  LEA ECX,DWORD PTR SS:[EBP-1C]
00406702 . 51      PUSH ECX
00406703 . 8BC8    MOV ECX,EAX
00406705 . E8 F3AE0200 CALL QNPlus.004315FD
0040670A . 8B7D EC  MOV EDI,DWORD PTR SS:[EBP-14] ; <== Nag String
0040670D . 6A 30    PUSH 30
0040670F . FF30    PUSH DWORD PTR DS:[EAX]
00406711 . 8BCE    MOV ECX,ESI
00406713 . 57      PUSH EDI
00406714 . C645 FC 02  MOV BYTE PTR SS:[EBP-4],2
00406718 . E8 43060400 CALL <JMP.&MFC70.#3890>     ; <== Shot Nag to your face
```

-**** Vậy thông qua quá trình phân tích trên chúng ta có thể tóm tắt lại quá trình đăng nhập thông tin vào để đăng ký chương trình này như sau.

1. Chuỗi **Name** và **Company** không tham gia vào quá trình mã hóa.
2. Chuỗi Registration Code phải có chiều dài là 29 , và tuân theo định dạng sau **XXXX-XXXX-XXXX-XXXX-XXXX**.
3. Tiếp theo trong quá trình kiểm tra thì 3 kí tự đầu tiên của chuỗi Registration Code phải là **“Q50”**, Các kí tự còn lại bắt đầu từ vị trí thứ 6 cho tới vị trí 28 phải tuân theo quy luật sau : - **XIXIX-X11XX-X11XX-X11XX** (Trong đó X đại diện cho 1 kí tự và phải là chữ hoa , còn 1 đại diện cho một chữ số).
4. Sau khi qua khỏi đoạn kiểm tra , chuỗi Registration Code của chúng ta sẽ được đem so sánh với chuỗi DefaultString là : **“M7P8C-QR13D-ED35F-J7R3X-LU05Q”** . Chuỗi DefaultString này sẽ được đảo ngược lại thành **“Q50UL-X3R7J-F53DE-D31RQ-C8P7M”** rồi mới đem đi so sánh.
5. Nếu như chuỗi Registration Code mà chúng ta nhập vào trùng với chuỗi DefaultString đã đảo ngược thì kết quả trình so sánh sẽ cho ra kết quả sai và sẽ bắn ra Nag “Invalid...” . Do đó 2 chuỗi này phải khác nhau.

III – KeyGen :

```
char reaSerial[64]={0};
char reaTempString[17]={0};
char reaRandom1[11] = "0123456789";
char reaRandom2[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int i=0,j=0, randomChart=0;
```

```

//3 vi tri dau tien phai la "Q50"
reaSerial[0] = 0x51;
reaSerial[1] = 0x35;
reaSerial[2] = 0x30;
    // Cac vi tri 5, 11, 17, 23 la '-'
reaSerial[5] = reaSerial[11] = reaSerial[17]= reaSerial[23] = 0x2D;
i = 7;
while (j < 4)
{
    reaSerial[i] = reaRandom1[randomChart = rand() % 10];
    i = i + 6;
    j++;
}

j = 0;
i = 4;
while (j < 5)
{
    reaSerial[i] = reaRandom2[randomChart = rand() % 26];
    i = i + 6;
    j++;
}

j = 0;
i = 6;
while (j < 4)
{
    reaSerial[i] = reaRandom2[randomChart = rand() % 26];
    i = i + 6;
    j++;
}

reaSerial[3] = reaRandom2[randomChart = rand() % 26];
reaSerial[15] = reaRandom2[randomChart = rand() % 26];
reaSerial[21] = reaRandom2[randomChart = rand() % 26];
reaSerial[8] = reaRandom2[randomChart = rand() % 26];
reaSerial[26] = reaRandom2[randomChart = rand() % 26];

reaSerial[9] = reaRandom1[randomChart = rand() % 10];
reaSerial[14] = reaRandom1[randomChart = rand() % 10];
reaSerial[20] = reaRandom1[randomChart = rand() % 10];
reaSerial[27] = reaRandom1[randomChart = rand() % 10];
SetDlgItemText(IDC_SERIAL, reaSerial);

```

/*/*/* Serial tương ứng :

Name : kienmanowar
Company : REA-cRaCkErTeAm
Serial : Q50FZ-B1N9D-X32FB-U18VT-O3C4U

IV – End of Tut :

- Finished – **January 13, 2005**

Reverse Engineering Association

SoftWare

Homepage :	http://www.magictweak.com
Production :	Efreesky Software
SoftWare :	MagicTweak v2.70
Copyright by :	Copyright (C) 2000 - 2003 Efreesky Software . All Rights Reserved.
Type :	Name/Serial
Packed :	PECompact 1.68 - 1.84 -> Jeremy Collake
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10, Import REC v1.6, LordPE v1.4
Unpack :	Manual
Request :	Correct Serial

Magic Tweak v2.70

MagicTweak is a special program designed to optimize and personalize Microsoft Windows. It provides one-stop, instant access to a variety of Windows settings that can be altered for a friendlier Windows environment. This unique software makes it easy to tweak hundreds of hidden settings in Windows XP/2000/Me/98, so there is no longer any need to dig through the registry looking for that specific setting (from Start Menu, Desktop, IE skin, System Icon to System Security) that just doesn't seem to be there. With the ability to customize almost any aspect of Windows, you can become a Windows expert almost instantly!

I – Information :

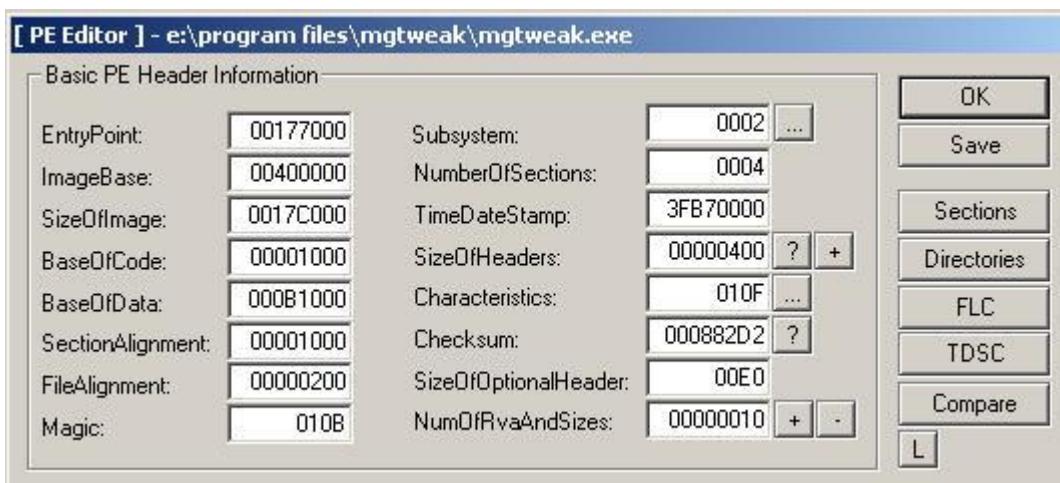
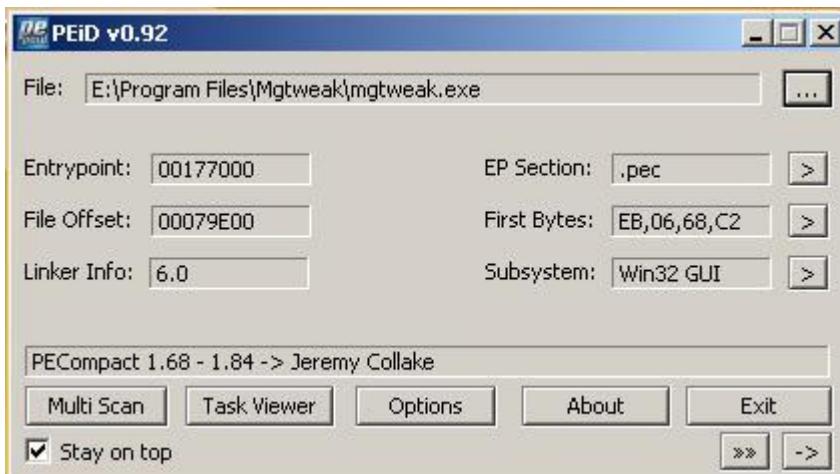
-**** Dùng **PeiD v0.92** để Detect , chúng ta biết được chương trình này đã được tác giả Pack lại bằng **PECompact 1.68 - 1.84 -> Jeremy Collake**. Vậy là nó được Pack khác với thằng **Magic Utilities**, nhưng không sao mọi vấn đề đều có cách giải quyết. Chúng ta sẽ thử **UnPack** thằng này xem sao .

-**** Run thử chương trình xem có gì đặc biệt , ngay lập tức một Nag Screen bắn ra : “**Enter the registration here**” , tại Nag Screen này chúng ta sẽ thấy có hai textbox bắt nhập **User Name** và **Registration Code** . Vậy là cách thức bảo vệ của chương trình là N/S. Nếu nhập đúng thì Oki. Tại thời điểm này chúng ta chưa có được nên đành click tạm vào Button **Continue Evaluation** để xem tiếp thông tin.

-**** Sau khi click chúng ta sẽ đến màn hình chính của chương trình , trên title bar chúng ta thấy một dòng như sau : **Magic Tweak Unregistered – Day 1 of 15**. Vậy có nghĩa là nếu chúng ta không đăng ký hợp pháp thì chúng ta sẽ chỉ được dùng thử các tính năng của chương trình trong vòng 15 days mà thôi. Quá date là “anh ơi ở lại em đi nhé hiiiiiii”. Một điều nữa muốn nói thêm là chương trình này về mặt User Interface trông rất bắt mắt good looking nên giá của nó cũng hơi bị đắt **29.95\$**.

II – UnPacking :

-**** Dùng **PeiD** và **LordPE** để Detect , chúng ta có được một số thông tin như sau :



1. Tìm OEP :

-**** Như chúng ta đã biết trong PeiD có một Plugin “**Generic OEP Finder**”. Đối với những chương trình bị Pack bằng **PECompact 1.68 - 1.84 -> Jeremy Collake** thì đây là một good plugin để tìm ra **OEP**, sử dụng Plugin này chúng ta sẽ có được OEP như sau :

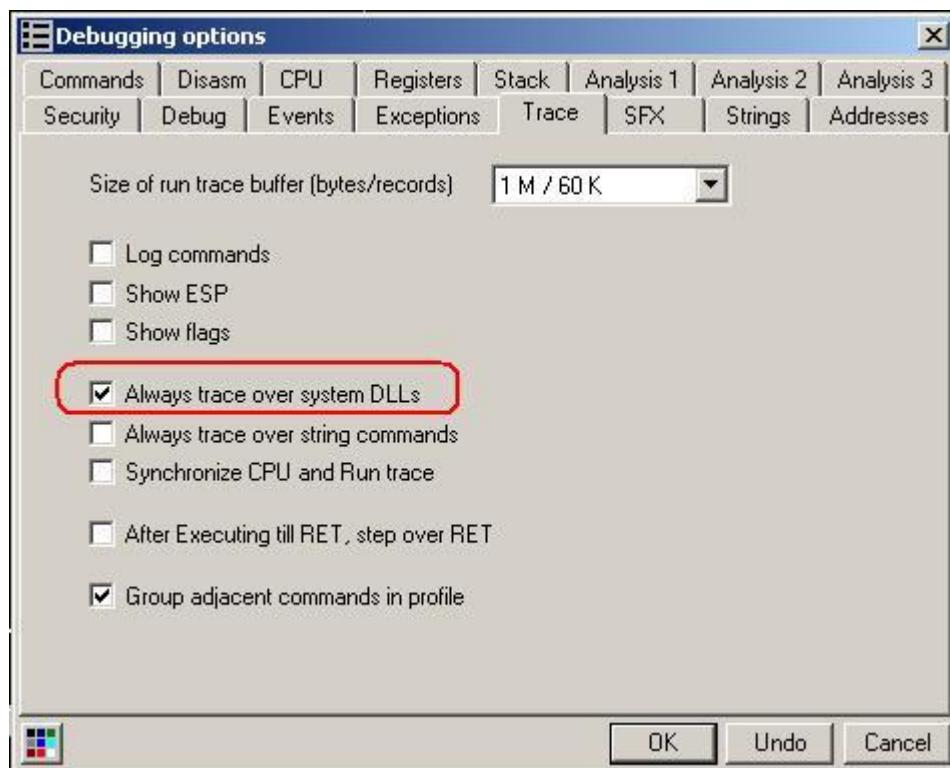


-**** Ke ke ke , OEP mà PeiD tìm được là **0047A6C2** . Oki , vậy theo công thức cũ chúng ta có thể tìm thấy **real OEP** như sau :

Real OEP:=OEP find in PEiD- Image Base = 47A6C2-400000= **7A6C2**.

2.Dump file :

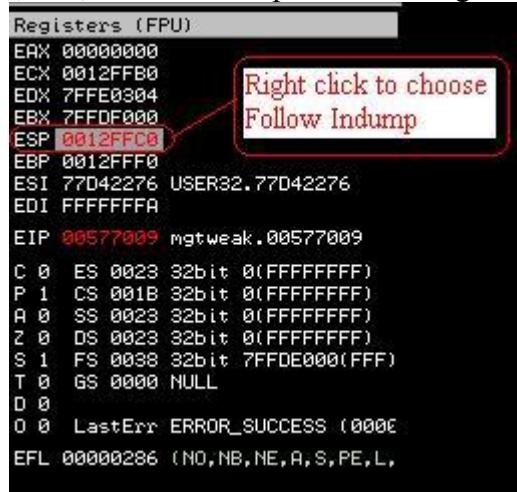
-**** Để có thể Dumping , chúng ta phải cấu hình Olly lại như sau . Tại màn hình chính của Olly , click vào menu **Option --- > Debugging Options (Alt + O)**. Chuyển tới Tab **Trace**. Chính lại giống như hình dưới đây :



-**** Sau đó Load file cần Unpack vào Olly (**mgtweak.exe**) , chọn **No (không Analysis)**. Chúng ta sẽ đến đây :

```
00577000 > /EB 06      JMP SHORT mgtweak.00577008 ;== We're here
00577002 |68 C2A60700  PUSH 7A6C2
00577007 |C3          RETN
00577008 |9C          PUSHFD
00577009 60          PUSHAD           ;<== Push F8 trace to here
0057700A E8 02000000  CALL mgtweak.00577011
0057700F 33C0        XOR EAX,EAX
```

-**** Nhấn F8 để Trace tới lệnh **PUSHAD** ở trên, sau khi trace đến đó các bạn để ý tới cửa sổ **Register (FPU)** . Click chuột phải tại thanh ghi **ESP** chọn **Follow in dump** như sau :



-**** Cửa sổ **Hex Dump** sẽ chuyển sang cửa sổ mới, chọn **4 bytes** đầu tiên của cửa sổ này và click chuột phải chọn **Break ==> Hardware, on acces ==> Word** :



-**** Sau đó nhấn **F9**, chúng ta sẽ đến đây :

```
00578550 50      PUSH EAX ; <== We're here
00578551 68 C2A64700  PUSH mgtweak.0047A6C2
00578556 C2 0400    RETN 4
```

-**** Nhấn F8 để **Trace** qua lệnh **RETN4**, chúng ta sẽ đến **OEP** :

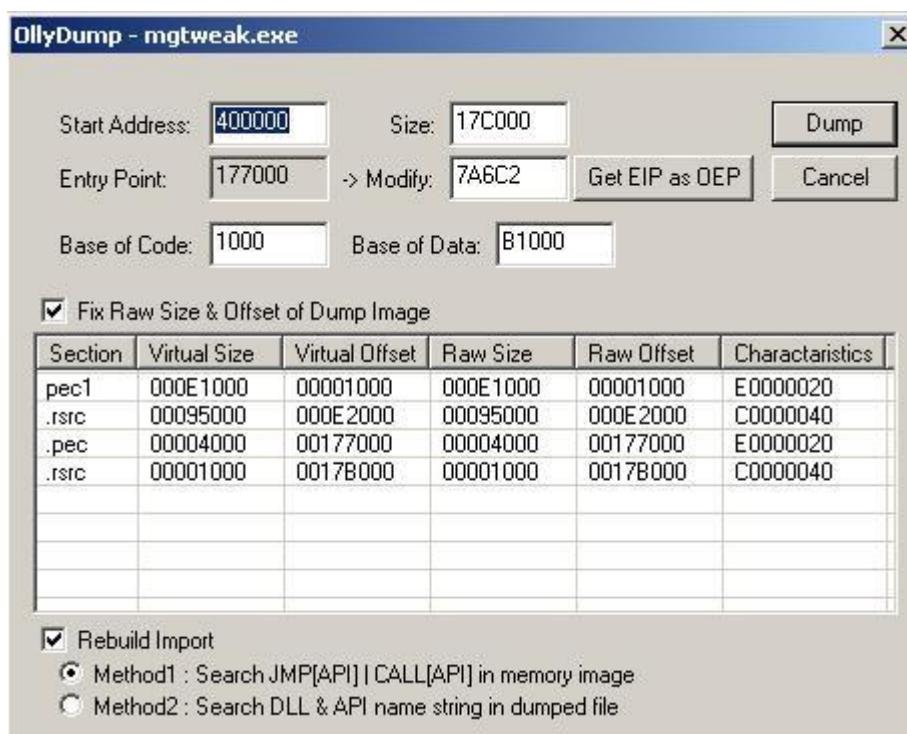
```
0047A6C2 55      PUSH EBP ; <== We're here : OEP
0047A6C3 8BEC    MOV EBP,ESP
0047A6C5 6A FF    PUSH -1
```

Vậy OEP mà chúng ta tìm được là : **47A6C2**. Bây giờ chúng ta sẽ tính toán tìm ra real OEP của chương trình theo công thức như sau :

Real OEP := OEP find in Olly- Image Base = 47A6C2-400000 = **7A6C2**. (trùng với **OEP** tìm theo PEid)

2.Dump file :

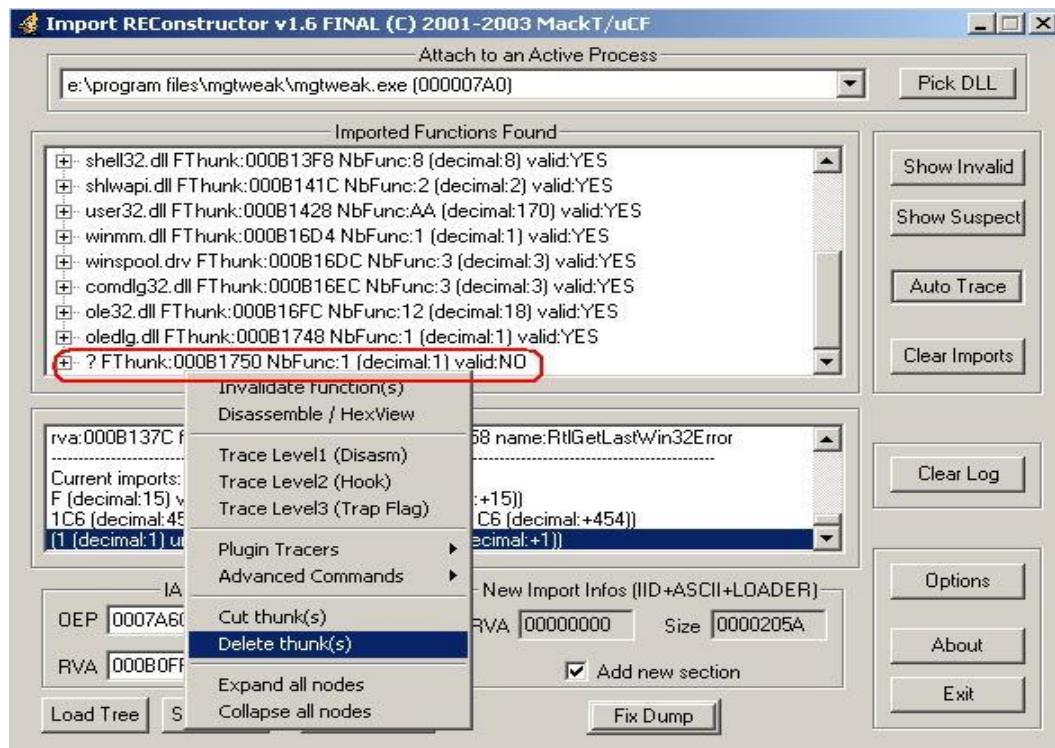
-**** Tại địa chỉ **0047A6C2** , chúng ta vào menu **Plugin-->OllyDump-->Dump debugged process** hoặc click chuột phải chọn **Dump debugged process**. Chúng ta sẽ đến màn hình **OllyDump** :



-**** Check **Rebuild Import** , tại ô Modify chính là **Real OEP** mà chúng ta đã tính toán được theo công thức ở trên. Click **Dump**, chúng ta Save lại với một tên bất kì . Ở đây mình save là **dumped.exe**.

3. Tìm và Sửa IAT (Import Address Table) :

-**** Giữ nguyên chương trình , mở **Import REConstructor v1.6F** load file **mgtweak.exe**. Thay giá trị trong ô OEP bằng giá trị mà chúng ta đã tìm được và tính toán được ở trên (**7A6C2**) sau đó chọn **IAT AutoSearch** rồi click **Get Imports** . Chúng ta sẽ được như sau :

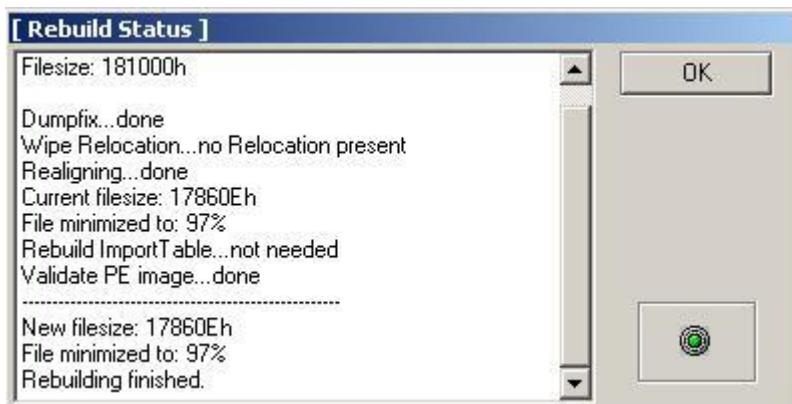


-**** Có một **Import Function InValid** , click chuột phải tại **Function** đó và chọn **Delete Thunk(s)**. Bây giờ chúng ta nhấn **Fix Dump** để **Fix IAT** cho file **dumped.exe** mà chúng ta đã save ở trên.

4. Làm sạch và giảm kích thước của File sau khi đã UnPack :

-**** Để hoàn thiện hơn chúng ta tiến hành quá trình làm sạch file và giảm kích thước của file sau khi đã **Fix Dump** (mục đích làm cho file càng nhỏ càng tốt). Nếu không thích các bạn có thể step over .

-**** Dùng **LordPE v1.4** . Load chương trình này lên , chọn **Rebuilt PE** . Sau đó chọn file mà chúng ta đã **Fix Dump** (chính là file **dumped_.exe**). Thê là chúng ta đã có một file mới hoàn chỉnh rồi đó các bạn .Ác ặc mệt quá.....nhưng nhờ đó mà công việc tiếp theo của chúng ta sẽ trở nên dễ dàng hơn.



-**** Dùng **PeiD v0.92** detect lại thì chúng ta biết được chương trình này được tác giả code bằng **Microsoft Visual C++ 6.0** . Nhìn thấy mà phê con mắt bên phải , nổ con mắt bên trái quá hiiiiiiiiii

III – Cracking :

-**** Bây giờ đến giai đoạn tiếp theo . Chúng ta sẽ tiến hành cracking trên file **dumped_.exe**. Run file này lên , Nag Screen bắn ra yêu cầu chúng ta nhập **Name** và **Registration Code** . Uh thì nó bảo nhập thì ta nhập , nhập đại một FU và FS vào xem sao. Ở đây mình nhập là : (**User Name : kienmnnowar Registration Code : 11111982**). Sau đó nhấn Register , một Nag thông báo hiện lên làm cho người ta mừng hụt tưởng là nó đồng ý rùi : **"Thanks for your registering. Please restart the program to validate the registration code "** . Nhấn Oki , sau đó Run lại chương trình , ặc ặc Màn hình bắt đăng ký lại hiện ra kia .

-**** Các soft của cùng một hãng sẽ có cùng một kiểu Protect . Cho nên đối với soft này mọi thông tin mà chúng ta nhập vào sẽ được chương trình ghi nhận lại vào file "**mgwin.ini**". Vậy chúng ta sẽ tìm đến đoạn code đọc thông tin từ file "**mgwin.ini**". Chúng ta tìm thấy tại đây trong Olly :
0044ACCF . 68 38114D00 PUSH dump_.004D1138 ; ASCII "mgwin.ini"

-**** Set BP tại địa chỉ trên , sau đó chúng ta Run lại chương trình trong Olly (F9). Chương trình sẽ dừng tại đây

```

0044ACCF . 68 38114D00 PUSH dump_.004D1138 ; ASCII "mgwin.ini" <==We're here
0044ACD4 . 8D8424 300100>LEA EAX,DWORD PTR SS:[ESP+130]
0044ACDB . 6A 28 PUSH 28
0044ACDD . 50 PUSH EAX
0044ACDE . 68 B8CD4D00 PUSH dump_.004DCDB8
0044ACE3 . 68 2C114D00 PUSH dump_.004D112C ; ASCII "UserName"
0044ACE8 . 68 28114D00 PUSH dump_.004D1128 ; ASCII "REG"
0044ACED . C68424 581B00>MOV BYTE PTR SS:[ESP+1B58],0C
0044ACF5 . FFD7 CALL EDI ;<== Get FU from file
0044ACF7 . 68 38114D00 PUSH dump_.004D1138 ; ASCII "mgwin.ini"
0044ACFC . 8D8C24 900000>LEA ECX,DWORD PTR SS:[ESP+90]
0044AD03 . 6A 28 PUSH 28
0044AD05 . 51 PUSH ECX
0044AD06 . 68 B8CD4D00 PUSH dump_.004DCDB8
0044AD0B . 68 C4604D00 PUSH dump_.004D60C4 ; ASCII "RegCode"
0044AD10 . 68 28114D00 PUSH dump_.004D1128 ; ASCII "REG"
0044AD15 . 8BD8 MOV EBX,EAX
0044AD17 . FFD7 CALL EDI ;<== Get FS from file
0044AD19 . 8D9424 2C0100>LEA EDX,DWORD PTR SS:[ESP+12C]
```

```

0044AD20 . 8D4C24 2C LEA ECX,DWORD PTR SS:[ESP+2C]
0044AD24 . 52 PUSH EDX
0044AD25 . E8 C0590400 CALL dump_.004906EA
0044AD2A . 8D8424 8C0000>LEA EAX,DWORD PTR SS:[ESP+8C]
0044AD31 . 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
0044AD35 . 50 PUSH EAX
0044AD36 . E8 AF590400 CALL dump_.004906EA
0044AD3B . 8B6C24 24 MOV EBP,DWORD PTR SS:[ESP+24]
0044AD3F . 8D8C24 8C0000>LEA ECX,DWORD PTR SS:[ESP+8C]
0044AD46 . 8D9424 2C0100>LEA EDX,DWORD PTR SS:[ESP+12C]
0044AD4D . 51 PUSH ECX ; /Arg2 ;<== ECX : FS
0044AD4E . 52 PUSH EDX ; | Arg1 ;<== EDX : FU
0044AD4F . 8BCD MOV ECX,EBP ; |
0044AD51 . E8 EA090000 CALL dump_.0044B740 ; \dump_.0044B740 (Rút kinh nghiệm không
thì lại bị ăn quả lừa tại đây đây hiiiiiiiiii)
0044AD56 . 85C0 TEST EAX,EAX
0044AD58 . 74 0F JE SHORT dump_.0044AD69
0044AD5A . 83FB 04 CMP EBX,4
0044AD5D . C705 34DB4D00>MOV DWORD PTR DS:[4DDB34],1
0044AD67 . 73 0A JNB SHORT dump_.0044AD73
0044AD69 > C705 34DB4D00>MOV DWORD PTR DS:[4DDB34],0
0044AD73 > 8D8424 8C0000>LEA EAX,DWORD PTR SS:[ESP+8C] ;<== EAX : FS
0044AD7A . 8BCD MOV ECX,EBP
0044AD7C . 50 PUSH EAX

```

0044AD7D . E8 1E0D0000 CALL dump_.0044BAA0 <== Trace Into (Hiiii All Serials)

===== Trace Into =====

.....
004D7D08=dump_.004D7D08 (ASCII
"8GDDEEG0,90FGEHG0,0H0F1M0E,D0GMG7HE,E7EEDEGE,B79EME90,EEG8FW7E,J7G0B975,1
GMF0GEJ,EEM880DE,8C9MHEEF,GG0MEDWD,01900GEE,GGMDE00G,EWD0DE81,WH7FFDFE,
D0E8ME9M,FE91B91G,GDF5EFEE,20BBEFFG,CEDFGH9B,EHEWHDGM,DG0E7W18,FMHDGG
GF,BGGM8)

.....
004D7528=dump_.004D7528 (ASCII
"B3F1CDB2,DCAFCCDD6,BLD3DDAC,8CDLL3C,9FCDBCAA,7ADCDDD2,C83ADDAD,5LDAQ0
56,DACC3DLA,DCA3L33C,8DDF2ACD,8C8C8DD3,CBBC8FDF,6CD8DDD3,DL2AEDL3,LDDD873
D,DF3WCWQC,CD7ALLLA,LDDA8CBD,D83CB83A,CC3LD39D,D373AL6D,LCLBDFCL,DLC
DCC,AXP91)

.....
004D6D50=dump_.004D6D50 (ASCII
"CCI6CGCI,C3ICMHGC,3IIM3BI6,0I6CC6M6,6IGD0DC6,IQHM6H0C,DIG5ICI6,G0CQC0M0,CI6FE
0IF,H6C15IIC,CCCC5MHI,1FI80DCH,IBC0CC6I,6DCF6H5I,MMC6CI8H,HCIC6I60,HC808QG6,1E6
ECIIC,1H72B72M,I660BG8C,0GH0BCFC,CMII6G5C,E06C1F00,MCC60C65,CFM5H)

.....
004D6A1C=dump_.004D6A1C (ASCII
"88EE8BKE,7EHKD8KD,V4K5EFEB,78H4KM7K,DYGXB4GK,4WUHKST,TW7FG2VK,SVTUXK
78,4X8GRKGU,55XGER7K,RY9N9FT8,YBWKV288,VHTS7H57,BX8YF878,74TMK4X4,K74SRKX
G,KVCXZR84,V8K7FS2F,C7W7ODK7,HHTXKKF7,7CKYXS4H,4CTVFVG8,VK74R74F,8SY8HEB
K,2VK4X)

.....
004D6244=dump_.004D6244 (ASCII

"HC3B3FHB,G2H4TD3E,HGG3HETE,CT2ESEA2,ETEACAH3,DCF3EF3S,ATE4GBGE,ECECECAC,BANED1EF,F2HT4TGH,T2UXPCHA,TETI4AHF,TEC3BB2T,2ANE3F4T,33SESTIF,FHTEETEC,FCI4ICEE,TD3DCGTN,2LA33GHA,TE2CEEIS,CEFCEEEE,H3GT3E4C,D33ETEC4,3EH23S24,EE34F)

.....
Ac ặc ặc thằng này nhiều chảng kém gì thằng **Magic Utilities** các bác à.....tha hồ mà dùng

===== **Trace Into** =====
0044AD82 . 85C0 TEST EAX,EAX
0044AD84 . 0F85 3D020000 JNZ dump_.0044AFC7

*/**/* Serial tương ứng : Có thể dùng bất kì một Serial nào trong các Serial trên ứng với UserName mà bạn gõ vào :

User Name : **kienmanowar**

Serial : **BANED1EF**

hoặc

User Name : **REA-cRaCkErTeAm**

Serial : **88EE8BKE**

IV – KeyGen :

N/A

V – End of Tut :

- Finished – **November 17, 2004**

Reverse Engineering Association

SoftWare

Homepage :	http://www.magictweak.com
Production :	Efreesky Software
SoftWare :	Magic Utilities 2004 v3.10
Copyright by :	Copyright (C) 2000 - 2004 Efreesky Software . All Rights Reserved.
Type :	Name/Serial
Packed :	PECompact 2.x -> Jeremy Collake
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10, Import REC v1.6, LordPE v1.4
Unpack :	Manual
Request :	Correct Serial

Magic Utilities v3.10

Magic Utilities is a cute program designed to make your computer clean and more stable.These utilities include Uninstiller Plus,StartUp Organizer,Process Killer.Magic Utilities enables you to easily and safely uninstall programs;inspect and manage the programs that automatically start when you turn on or logon to your computer;list and control all currently running processes(system and hidden processes are also shown).With a cool and user-friendly interface makes it easy for anyone to use Magic Utilities.

I – Information :

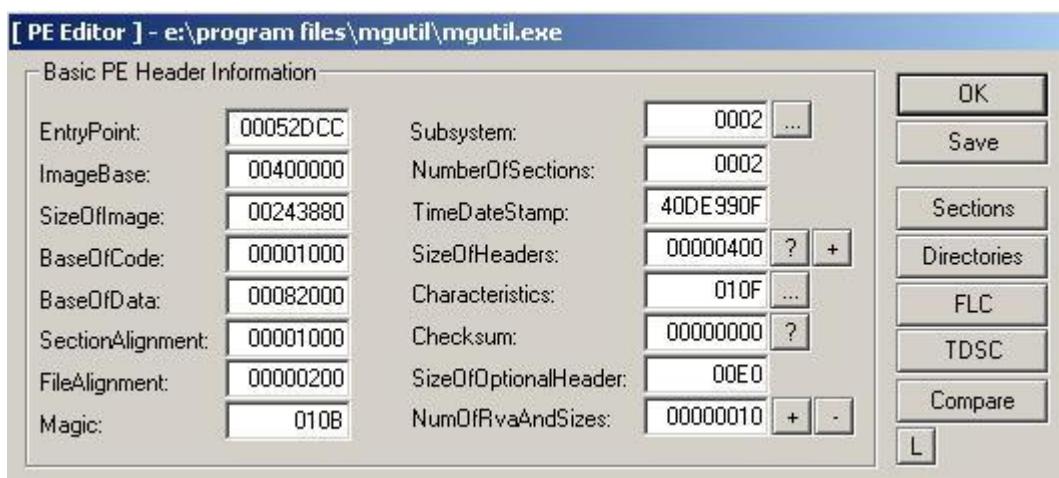
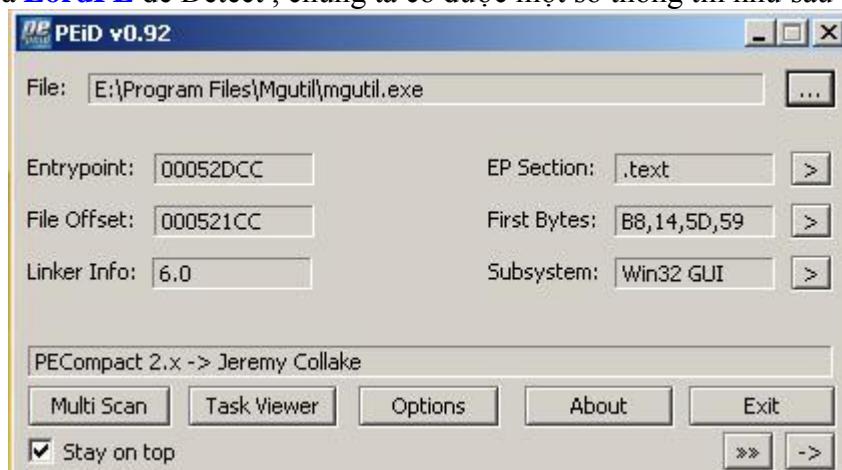
**** Dùng **PeiD v0.92** để Detect , chúng ta biết được chương trình này đã được tác giả **Pack** lại bằng **PECompact 2.x -> Jeremy Collake**. Vậy là khó cho chúng ta rồi, nhưng không sao có khó mới lò cái khôn. Chúng ta sẽ thử UnPack thằng này xem sao .

-**** Run thử chương trình xem có gì đặc biệt , ngay lập tức một Nag Screen bắn ra : “**Enter the registration here**” , tại Nag Screen này chúng ta sẽ thấy có hai textbox bắt nhập **User Name** và **Registration Code** . Vậy là cách thức bảo vệ của chương trình là N/S. Nếu nhập đúng thì Oki. Tại thời điểm này chúng ta chưa có được nên đành click tạm vào Button **Continue Evaluation** để xem tiếp thông tin.

-**** Sau khi click chúng ta sẽ đến màn hình chính của chương trình , trên title bar chúng ta thấy một dòng như sau : **Magic Utilities 2004 Unregistered – Day 1 of 15**. Vậy có nghĩa là nếu chúng ta không đăng ký hợp pháp thì chúng ta chỉ được dùng thử các tính năng của chương trình trong vòng 15 days mà thôi. Quá date là “anh oi ở lại em đi nhé hiiiiiii”. Một điều nữa muốn nói thêm là chương trình này về mặt **User Interface** trông rất bắt mắt good looking nên giá của nó cũng hơi bị đep **29.95\$**.

II – UnPacking :

-**** Dùng **PeID** và **LordPE** để Detect , chúng ta có được một số thông tin như sau :



1. Tìm OEP :

-**** Load chương trình này trong **Olly**, Chọn **No (không Analysis)**. Chúng ta sẽ đến đây :

```
00452DCC > B8 145D5900 MOV EAX,mgutil.00595D14 ;<== We're here
00452DD1 50 PUSH EAX
00452DD2 64:FF35 0000000>PUSH DWORD PTR FS:[0]
00452DD9 64:8925 0000000>MOV DWORD PTR FS:[0],ESP
```

```

00452DE0 33C0    XOR EAX,EAX
00452DE2 8908    MOV DWORD PTR DS:[EAX],ECX
00452DE4 50      PUSH EAX
00452DE5 45      INC EBP

```

-**** Nhấn F9 hai lần chúng ta sẽ đến dòng lệnh sau :

```

00452DE2 8908    MOV DWORD PTR DS:[EAX],ECX ;<== We're here
00452DE4 50      PUSH EAX
00452DE5 45      INC EBP
00452DE6 43      INC EBX
00452DE7 6F      OUTS DX,DWORD PTR ES:[EDI]    ; I/O command

```

-**** Tại đây chúng ta nhấn tổ hợp phím **Alt + M** để mở cửa sổ **Memory Map** trong Olly:

Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00360000	00001000			Priv	RW	RW		
003E0000	00004000			Priv	RW	RW		
003F0000	00002000			Map	R	R		
00400000	00001000	mgutil		PE header	Imag	R		RWE
00401000	0016B000	mgutil	.text	code	Imag	R	RWE	
0056C000	000D8000	mgutil	.rsrc	imports,reso	Imag	R		RWE
00650000	00003000			Map	R E	R E		
00710000	00002000			Map	R E	R E		

-**** Tại dòng màu đỏ ở trên , chúng ta click chuột phải sau đó chọn **Set memory breakpoint on access.**

-**** Sau đó nhấn **Shift + F9** chúng ta sẽ lại đến đoạn code sau :

```

00595D29 C602 E9    MOV BYTE PTR DS:[EDX],0E9 ;<== We're here
00595D2C 83C2 05    ADD EDX,5
00595D2F 2BCA        SUB ECX,EDX
00595D31 894A FC    MOV DWORD PTR DS:[EDX-4],ECX
00595D34 33C0        XOR EAX,EAX
00595D36 C3          RETN

```

-**** Tiếp theo chúng nhấn **Shift + F9** thêm 3 lần nữa chúng ta sẽ đến một đoạn code khác như sau :

```

00595D37 B8 BD4CCDFF  MOV EAX,FFCD4CBD ;<== We're here
00595D3C 64:8F05 0000000>POP DWORD PTR FS:[0]
00595D43 83C4 04    ADD ESP,4
00595D46 55          PUSH EBP
00595D47 53          PUSH EBX

```

-**** Tại đây chúng ta nhấn **Ctrl + F12** , chúng ta sẽ nhìn thấy đoạn code sau :

```

00B50289 F3:A4      REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[> ;<== We're here , rất gần
với OEP rùi các bạn ơi.
00B5028B 8BF3        MOV ESI,EBX

```

```

00B5028D 8D8D D7128C00 LEA ECX,DWORD PTR SS:[EBP+8C12D7]
00B50293 51          PUSH ECX
00B50294 E8 4D020000 CALL 00B504E6
00B50299 8B4E 2C     MOV ECX,DWORD PTR DS:[ESI+2C]
00B5029C 8B56 24     MOV EDX,DWORD PTR DS:[ESI+24]
00B5029F 0356 08     ADD EDX,DWORD PTR DS:[ESI+8]

```

-**** Cuối cùng chúng ta nhấn **Ctrl + F12** một lần nữa , OEP sẽ hiện ra ngay trước mắt chúng ta :

```

00452DCC > 55          PUSH EBP <== Our OEP is 00452DCC
00452DCD 8BEC          MOV EBP,ESP
00452DCF 6A FF          PUSH -1
00452DD1 68 908D4800    PUSH mgutil.00488D90
00452DD6 68 488D4500    PUSH mgutil.00458D48
00452DDB 64:A1 00000000  MOV EAX,DWORD PTR FS:[0]

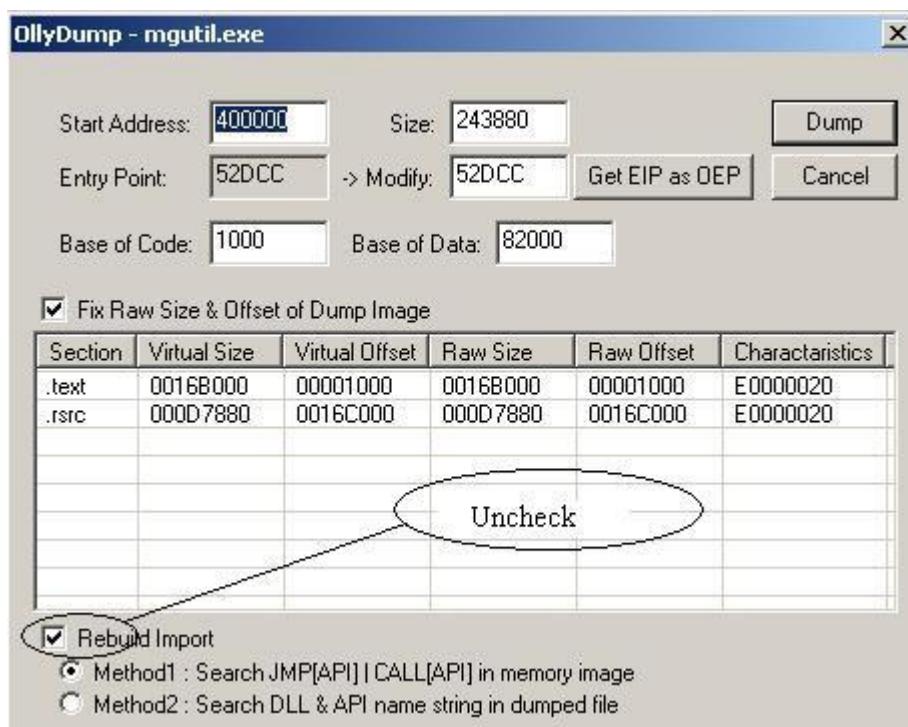
```

Vậy OEP mà chúng ta tìm được là : **452DCC**. Bây giờ chúng ta sẽ tính toán tìm ra **real OEP** của chương trình theo công thức như sau :

Real OEP :=OEP find in Olly- Image Base = 452DCC-400000 = **52DCC**.

2.Dump file :

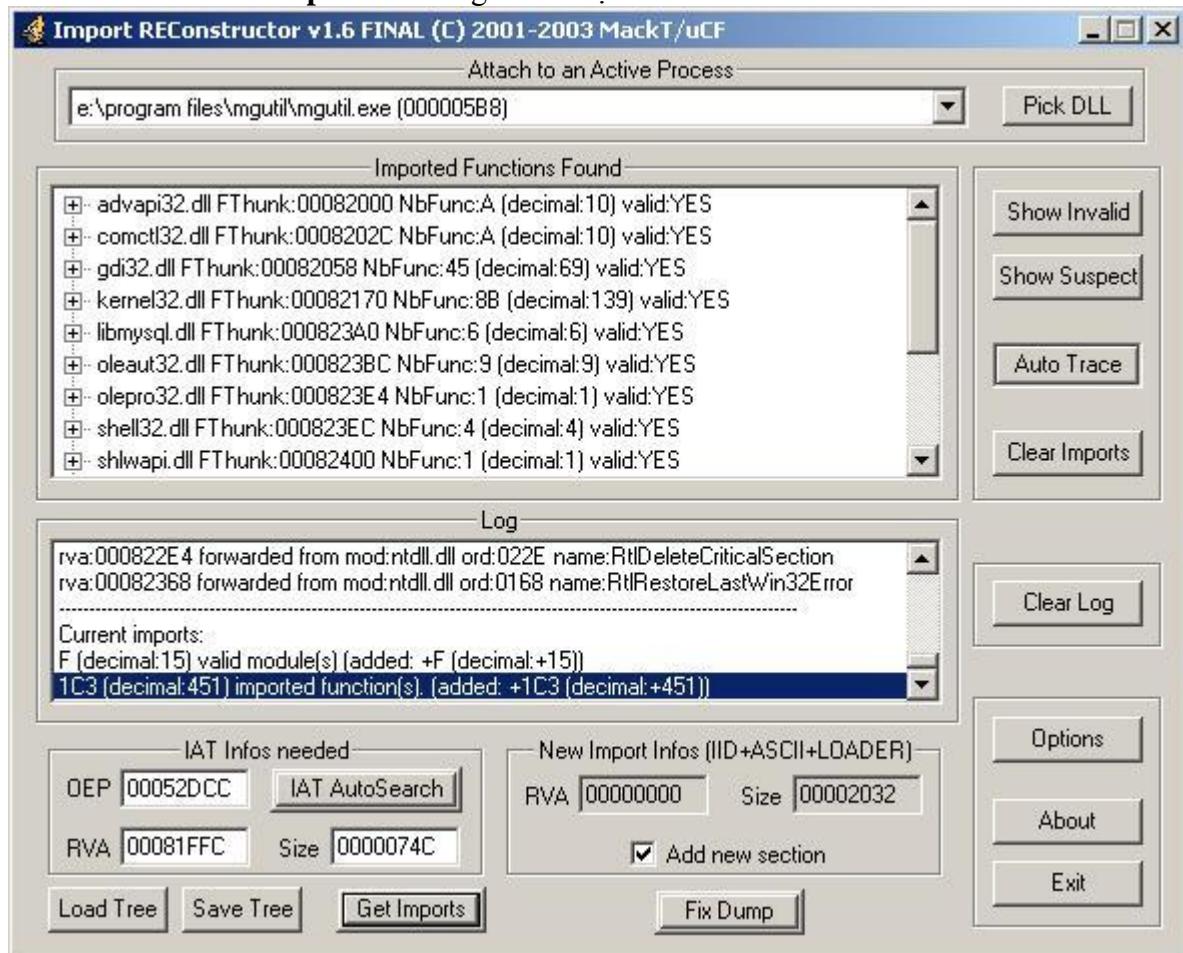
-**** Tại địa chỉ **00452DCC** , chúng ta vào menu **Plugin-->OllyDump-->Dump debugged process** hoặc click chuột phải chọn **Dump debugged process**. Chúng ta sẽ đến màn hình **OllyDump** :



-**** Uncheck **Rebuild Import** , tại ô Modify chính là Real OEP mà chúng ta đã tính toán được theo công thức ở trên. Click Dump, chúng ta Save lại với một tên bất kì . Ở đây mình save là **dumped.exe**.

3. Tìm và Sửa IAT (Import Adress Table) :

-**** Giữ nguyên chương trình , mở **Import REConstructor v1.6F** load file **mgutil.exe**. Thay giá trị trong ô OEP bằng giá trị mà chúng ta đã tìm được và tính toán được ở trên (**52DCC**) sau đó chọn **IAT AutoSearch** rồi click **Get Imports** . Chúng ta sẽ được như sau :

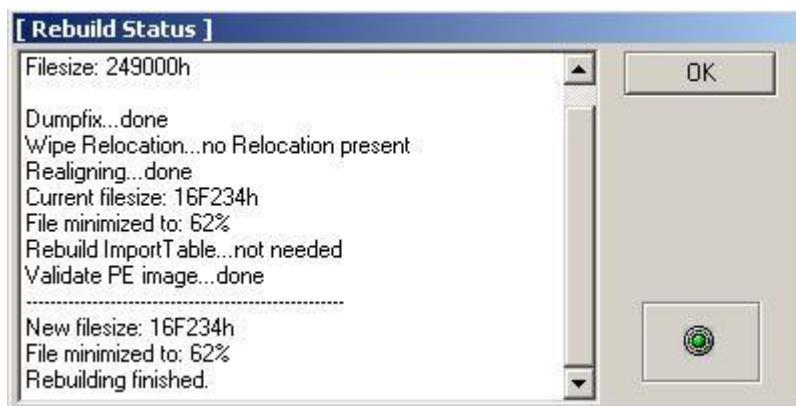


-**** Tất cả các **Import Function** đều **Valid** quá đã rùi. Bây giờ chúng ta nhấn **Fix Dump** để **Fix IAT** cho file **dumped.exe** mà chúng ta đã save ở trên.

4. Làm sạch và giảm kích thước của File sau khi đã UnPack :

-**** Để hoàn thiện hơn chúng ta tiến hành quá trình làm sạch file và giảm kích thước của file sau khi đã **Fix Dump** (mục đích làm cho file càng nhỏ càng tốt). Nếu không thích các bạn có thể step over .

-**** Dùng **LordPE v1.4** . Load chương trình này lên , chọn **Rebuilt PE** . Sau đó chọn file mà chúng ta đã **Fix Dump** (chính là file **dumped_.exe**). Thế là chúng ta đã có một file mới hoàn chỉnh rồi đó các bạn .Ác ặc mệt quá.....



-**** Dùng **PeID v0.92** detect lại thì chúng ta biết được chương trình này được tác giả code bằng **Microsoft Visual C++ 6.0**. Nhìn thấy mà phê con mắt bên phải, nổ con mắt bên trái quá hiiiiiiiiii

III – Cracking :

-**** Bây giờ đến giai đoạn tiếp theo. Chúng ta sẽ tiến hành cracking trên file **dumped_.exe**. Run file này lên, Nag Screen bắn ra yêu cầu chúng ta nhập **Name** và **Registration Code**. Uh thì nó bảo nhập thì ta nhập, nhập đại một FU và FS vào xem sao. Ở đây mình nhập là : (**User Name : kienmnnowar Registration Code : 11111982**). Sau đó nhấn Register, một Nag thông báo hiện lên làm cho người ta mừng hụt tưởng là nó đồng ý rùi : **"Thanks for your registering. Please restart the program to validate the registration code"**. Nhấn Oki, sau đó Run lại chương trình, xác Màn hình bắt đăng ký lại hiện ra kìa

-**** Chúng ta ghi nhớ đoạn Nag trên, load chương trình này trong Olly (nhớ là file **dumped_.exe** nhé). Tìm chuỗi trên, nhưng lần mò một hồi không thấy. Vậy chúng ta phải dùng đến phương pháp Stack của anh Moon rùi.

-**** Nhấn F9 để run chương trình, nhập User Name và Registration Code vào. Nhấn Register, Nag thông báo sẽ xuất hiện. Giữ nguyên chương trình, dừng nhấn OK vội, chúng ta quay trở lại Olly. Nhấn F12 để Pause chương trình lại, sau đó nhấn Alt-K để mở cửa sổ **Call stack of main thread**. Ta sẽ thấy được như sau :

Address	Stack	Procedure / arguments	Called from	Frame
00127240	77D43FB	Includes 7FFE0304	USER32.77D043FB	00127274
00127244	77D487A7	USER32.WaitMessage	USER32.77D0487A2	00127274
00127278	77D4F58C	USER32.77D048607	USER32.77D04F587	00127274
001272A0	77D6AAAE	USER32.77D04F4D8	USER32.77D06AA9	0012729C
00127558	77D6AC40	? USER32.SoftModalMessageBox	USER32.77D06AC3B	001274E0
001276A0	77D6ADCC	? USER32.77D06AB06	USER32.77D06DC7	00127628
001276F4	77D6AE8A	USER32.MessageBoxTimeoutW	USER32.77D06AE85	001276F0
00127728	77D6AE17	? USER32.MessageBoxTimeoutA	USER32.77D06AE12	00127724
00127748	77D6ADFB	? USER32.MessageBoxExA	USER32.77D06ADF6	00127744
0012774C	00270282	hOwner = 00270282 ('Register',class='#32770') Text = "Thanks for your registering.Please restart the program to validate"		
00127750	00B482C0	Title = "Register"		
00127754	00B472A8	Style = MB_OK MB_ICONASTERISK MB_APPLMODAL		
0012775C	00000000	LanguageID = (LANG_NEUTRAL)		
00127760	00468C70	? USER32.MessageBoxA	dumped_.00468C6A	
00127764	00270282	hOwner = 00270282 ('Register',class='#32770') Text = "Thanks for your registering.Please restart the program to validate"		
00127768	00B482C0	Title = "Register"		
0012776C	00B472A8	Style = MB_OK MB_ICONASTERISK MB_APPLMODAL		
00127770	00000000			
00127778	0043A25F	? dumped_.00468C42	dumped_.0043A25A	
001277C	00B482C0	Hrg1 = 00B482C0 ASCII "Thanks for your registering.Please restart the progr"		
00127780	00B472A8	Arg2 = 00B472A8 ASCII "Register"		
00127784	00000040	Arg3 = 00000040		

Dbl Click here

-**** Double tại địa chỉ đánh dấu ở trên chúng ta sẽ đến đây :

0043A25A . E8 E3E90200 CALL dumped_.00468C42 ;===== Shot Nag to your face
0043A25F . 8B86 7C100000 MOV EAX,DWORD PTR DS:[ESI+107C]

-**** Bây giờ chúng ta đã biết được thông tin về cái Nag đó , bây giờ nhấn **Ctrl+F2** to restart , nhấn **Ctrl-G** , sau đó gõ vào địa chỉ **0043A25A** chúng ta sẽ nhảy tới chỗ hàm gọi Nag. Lần ngược lên trên một chút chúng ta sẽ Set Bp tại đây :

```

0043A0E3 . E8 55F40200 CALL dumped_.0046953D <== We set BP here
0043A0E8 . A1 F8CF4900 MOV EAX,DWORD PTR DS:[49CFF8]
0043A0ED . 894424 10    MOV DWORD PTR SS:[ESP+10],EAX
0043A0F1 . C74424 74 000>MOV DWORD PTR SS:[ESP+74],0
0043A0F9 . 894424 0C    MOV DWORD PTR SS:[ESP+C],EAX
0043A0FD . 8D4424 10    LEA EAX,DWORD PTR SS:[ESP+10]
0043A101 . 885C24 74    MOV BYTE PTR SS:[ESP+74],BL
0043A105 . 50          PUSH EAX
0043A106 . 51          PUSH ECX
0043A107 . 8BCC        MOV ECX,ESP
0043A109 . 896424 1C    MOV DWORD PTR SS:[ESP+1C],ESP
0043A10D . 68 F4CA4900 PUSH dumped_.0049CAF4 ; ASCII "IDS_REGISTER"
0043A112 . E8 13C80200 CALL dumped_.0046692A
0043A117 . 51          PUSH ECX
0043A118 . C68424 800000>MOV BYTE PTR SS:[ESP+80],2
0043A120 . 8BCC        MOV ECX,ESP
0043A122 . 896424 24    MOV DWORD PTR SS:[ESP+24],ESP
0043A126 . 68 68CB4900 PUSH dumped_.0049CB68 ; ASCII "Register"
0043A12B . E8 FAC70200 CALL dumped_.0046692A
0043A130 . 889C24 800000>MOV BYTE PTR SS:[ESP+80],BL
0043A137 . E8 9456FEFF CALL dumped_.0041F7D0
0043A13C . 8B86 7C100000 MOV EAX,DWORD PTR DS:[ESI+107C] ; |
0043A142 . 68 B4F54900 PUSH dumped_.0049F5B4 ; |Arg2 = 0049F5B4
0043A147 . 50          PUSH EAX ; |Arg1
0043A148 . E8 58830100 CALL dumped_.004524A5 ; \dumped_.004524A5
0043A14D . 83C4 14    ADD ESP,14
0043A150 . 85C0        TEST EAX,EAX
0043A152 . 75 56    JNZ SHORT dumped_.0043A1AA
0043A154 . 8D4C24 0C    LEA ECX,DWORD PTR SS:[ESP+C]
0043A158 . 51          PUSH ECX
0043A159 . 51          PUSH ECX
0043A15A . 8BCC        MOV ECX,ESP
0043A15C . 896424 20    MOV DWORD PTR SS:[ESP+20],ESP
0043A160 . 68 9CCB4900 PUSH dumped_.0049CB9C ; ASCII
"IDM_MSG_REGISTER1"
0043A165 . E8 C0C70200 CALL dumped_.0046692A
0043A16A . 51          PUSH ECX
0043A16B . C68424 800000>MOV BYTE PTR SS:[ESP+80],3
0043A173 . 8BCC        MOV ECX,ESP
0043A175 . 896424 20    MOV DWORD PTR SS:[ESP+20],ESP
0043A179 > 68 68CB4900 PUSH dumped_.0049CB68 ; ASCII "Register"
0043A17E . E8 A7C70200 CALL dumped_.0046692A
0043A183 . 889C24 800000>MOV BYTE PTR SS:[ESP+80],BL
0043A18A . E8 4156FEFF CALL dumped_.0041F7D0
0043A18F . 8B5424 1C    MOV EDX,DWORD PTR SS:[ESP+1C]
0043A193 . 8B4424 18    MOV EAX,DWORD PTR SS:[ESP+18]
0043A197 . 83C4 0C    ADD ESP,0C
0043A19A . 8BCE        MOV ECX,ESI

```

```

0043A19C . 6A 10      PUSH 10
0043A19E . 52         PUSH EDX
0043A19F . 50         PUSH EAX
0043A1A0 . E8 9DEA0200 CALL dumped_.00468C42
0043A1A5 . E9 F9000000 JMP dumped_.0043A2A3
0043A1AA > 8B86 78100000 MOV EAX,DWORD PTR DS:[ESI+1078]
0043A1B0 . 68 B4F54900 PUSH dumped_.0049F5B4           ; /Arg2 = 0049F5B4
0043A1B5 . 50         PUSH EAX             ; |Arg1
0043A1B6 . E8 EA820100 CALL dumped_.004524A5           ; \dumped_.004524A5
0043A1BB . 83C4 08     ADD ESP,8
0043A1BE . 85C0         TEST EAX,EAX
0043A1C0 . 75 27       JNZ SHORT dumped_.0043A1E9
0043A1C2 . 8D4C24 0C   LEA ECX,DWORD PTR SS:[ESP+C]
0043A1C6 . 51         PUSH ECX
0043A1C7 . 51         PUSH ECX
0043A1C8 . 8BCC         MOV ECX,ESP
0043A1CA . 896424 20   MOV DWORD PTR SS:[ESP+20],ESP
0043A1CE . 68 88CB4900 PUSH dumped_.0049CB88           ; ASCII
"IDM_MSG_REGISTER2"
0043A1D3 . E8 52C70200 CALL dumped_.0046692A
0043A1D8 . 51         PUSH ECX
0043A1D9 . C68424 800000>MOV BYTE PTR SS:[ESP+80],4
0043A1E1 . 8BCC         MOV ECX,ESP
0043A1E3 . 896424 20   MOV DWORD PTR SS:[ESP+20],ESP
0043A1E7 .^ EB 90       JMP SHORT dumped_.0043A179
0043A1E9 > 8B86 7C100000 MOV EAX,DWORD PTR DS:[ESI+107C]
0043A1EF . 8B3D 08254800 MOV EDI,DWORD PTR DS:[<&user32.wsprintfA>; 
USER32.wsprintfA
0043A1F5 . 8D4C24 1C   LEA ECX,DWORD PTR SS:[ESP+1C]
0043A1F9 . 50         PUSH EAX             ; |Format
0043A1FA . 51         PUSH ECX             ; |s
0043A1FB . FFD7         CALL EDI             ; \wsprintfA
0043A1FD . 8B86 78100000 MOV EAX,DWORD PTR DS:[ESI+1078]
0043A203 . 8D5424 4C   LEA EDX,DWORD PTR SS:[ESP+4C]
0043A207 . 50         PUSH EAX
0043A208 . 52         PUSH EDX
0043A209 . FFD7         CALL EDI
0043A20B . 83C4 10     ADD ESP,10
0043A20E . 8D4424 0C   LEA EAX,DWORD PTR SS:[ESP+C]
0043A212 . 50         PUSH EAX
0043A213 . 51         PUSH ECX
0043A214 . 8BCC         MOV ECX,ESP
0043A216 . 896424 20   MOV DWORD PTR SS:[ESP+20],ESP
0043A21A . 68 74CB4900 PUSH dumped_.0049CB74           ; ASCII
"IDM_MSG_REGISTER3"
0043A21F . E8 06C70200 CALL dumped_.0046692A
0043A224 . 51         PUSH ECX
0043A225 . C68424 800000>MOV BYTE PTR SS:[ESP+80],5
0043A22D . 8BCC         MOV ECX,ESP
0043A22F . 896424 20   MOV DWORD PTR SS:[ESP+20],ESP
0043A233 . 68 68CB4900 PUSH dumped_.0049CB68           ; ASCII "Register"
0043A238 . E8 EDC60200 CALL dumped_.0046692A

```

```

0043A23D . 889C24 800000>MOV BYTE PTR SS:[ESP+80],BL
0043A244 . E8 8755FEFF CALL dumped_.0041F7D0
0043A249 . 8B4C24 1C MOV ECX,DWORD PTR SS:[ESP+1C]
0043A24D . 8B5424 18 MOV EDX,DWORD PTR SS:[ESP+18]
0043A251 . 83C4 0C ADD ESP,0C
0043A254 . 6A 40 PUSH 40
0043A256 . 51 PUSH ECX
0043A257 . 52 PUSH EDX
0043A258 . 8BCE MOV ECX,ESI
0043A25A . E8 E3E90200 CALL dumped_.00468C42 ;<== Shot Nag "Thank you....."
0043A25F . 8B86 7C100000 MOV EAX,DWORD PTR DS:[ESI+107C]
0043A265 . 8B3D 50224800 MOV EDI,DWORD PTR DS:[<&kernel32.WritePr>; kernel32.WritePrivateProfileStringA
0043A26B . 68 60994900 PUSH dumped_.00499960 ; /FileName = "mgutil_win.ini"
0043A270 . 50 PUSH EAX ; |String
0043A271 . 68 54994900 PUSH dumped_.00499954 ; |Key = "UserName"
0043A276 . 68 50994900 PUSH dumped_.00499950 ; |Section = "REG"
0043A27B . FFD7 CALL EDI ; \WritePrivateProfileStringA
0043A27D . 8B86 78100000 MOV EAX,DWORD PTR DS:[ESI+1078]
0043A283 . 68 60994900 PUSH dumped_.00499960 ; /FileName = "mgutil_win.ini"
0043A288 . 50 PUSH EAX ; |String
0043A289 . 68 48994900 PUSH dumped_.00499948 ; |Key = "RegCode"
0043A28E . 68 50994900 PUSH dumped_.00499950 ; |Section = "REG"
0043A293 . FFD7 CALL EDI ; \WritePrivateProfileStringA
0043A295 . 6A 00 PUSH 0
0043A297 . E8 1AF80300 CALL dumped_.00479AB6
0043A29C . 8BCE MOV ECX,ESI
0043A29E . E8 FA1A0300 CALL dumped_.0046BD9D
0043A2A3 > 8D4C24 0C LEA ECX,DWORD PTR SS:[ESP+C]
0043A2A7 . C64424 74 00 MOV BYTE PTR SS:[ESP+74],0
0043A2AC . E8 0BC60200 CALL dumped_.004668BC
0043A2B1 . 8D4C24 10 LEA ECX,DWORD PTR SS:[ESP+10]
0043A2B5 . C74424 74 FFF>MOV DWORD PTR SS:[ESP+74],-1
0043A2BD . E8 FAC50200 CALL dumped_.004668BC
0043A2C2 . 8B4C24 6C MOV ECX,DWORD PTR SS:[ESP+6C]
0043A2C6 . 5F POP EDI
0043A2C7 . 5E POP ESI
0043A2C8 . 64:890D 00000>MOV DWORD PTR FS:[0],ECX
0043A2CF . 5B POP EBX
0043A2D0 . 83C4 6C ADD ESP,6C
0043A2D3 . C3 RETN

```

/** Note : Thực chất cả đoạn mã rất dài trên đây , nếu chúng ta tìm kiếm một hồi sẽ không thể nào tìm ra được hàm mã hóa tạo **Right Serial** cả. Tất cả phần trên chỉ là kiểm tra xem chúng ta có nhập **User Name** hay **Registration Code** hay không mà thôi. Nếu chúng ta không nhập **User Name** thì ăn một Nag là "**No User Name**" ngược lại không nhập **Registration Code** thì sẽ bị bắn bằng "**No Registration Code**".

Ngược lại khi chúng ta đã nhập đủ cả hai thông số trên mà chương trình yêu cầu , thì không bắn 1 trong 2 Nag nữa mà nó sẽ Popup ra cái Nag "**Thank you.....**"(nhìn thấy mà sướng!!). Nhưng thực chất chương trình quái ở chỗ sẽ lưu lại tất cả những thông tin mà chúng ta nhập vào vào trong file "**mgutil_win.ini**" nằm trong thư mục Windows. Đến khi chúng ta Run lại chương trình thì nó vẫn yêu cầu chúng ta nhập lại , vậy khả năng nghi ngờ là ở chỗ chương trình sẽ đọc lại nội dung trong file "**mgutil_win.ini**" , sau đó tạo ra một **Right Serial** so sánh với cái Serial trong file **.ini** kia . Note**/

-**** Nghi ngờ thế nào thì chúng ta làm như thế , chúng ta sẽ tìm đến đoạn CODE đọc thông tin từ file "mgutil_win.ini" khi chúng ta run lại chương trình . Chúng ta tìm thấy đoạn CODE này tại địa chỉ :
00420306 . 68 60994900 PUSH dumped_.00499960 ; ASCII "mgutil_win.ini"

-**** Set BP tại địa chỉ trên , sau đó chúng ta Run lại chương trình trong Olly (F9). Chương trình sẽ dừng tại đây.

00420306 . 68 60994900 PUSH dumped_.00499960 ; ASCII "mgutil_win.ini" <== We're here
0042030B . 8D9424 540100>LEA EDX,DWORD PTR SS:[ESP+154]
00420312 . 6A 28 PUSH 28
00420314 . 52 PUSH EDX
00420315 . 68 B4F54900 PUSH dumped_.0049F5B4
0042031A . 68 54994900 PUSH dumped_.00499954 ; ASCII "UserName"
0042031F . 68 50994900 PUSH dumped_.00499950 ; ASCII "REG"
00420324 . C68424 008000>MOV BYTE PTR SS:[ESP+8000],0D
0042032C . FFD7 CALL EDI ;|<== Get FU from file "mgutil_win.ini"
0042032E . 8BD8 MOV EBX,EAX
00420330 . 68 60994900 PUSH dumped_.00499960 ; ASCII "mgutil_win.ini"
00420335 . 8D8424 B40000>LEA EAX,DWORD PTR SS:[ESP+B4]
0042033C . 6A 28 PUSH 28
0042033E . 50 PUSH EAX
0042033F . 68 B4F54900 PUSH dumped_.0049F5B4
00420344 . 68 48994900 PUSH dumped_.00499948 ; ASCII "RegCode"
00420349 . 68 50994900 PUSH dumped_.00499950 ; ASCII "REG"
0042034E . FFD7 CALL EDI ;|<== Get FS from file "mgutil_win.ini"
00420350 . 8D8C24 500100>LEA ECX,DWORD PTR SS:[ESP+150] ;|<== ECX : FU
00420357 . 51 PUSH ECX
00420358 . 8D4C24 38 LEA ECX,DWORD PTR SS:[ESP+38]
0042035C . E8 E4660400 CALL dumped_.00466A45 ;|<== Get Length (FU)
00420361 . 8D9424 B00000>LEA EDX,DWORD PTR SS:[ESP+B0] ;|<== EDX : FS
00420368 . 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
0042036C . 52 PUSH EDX
0042036D . E8 D3660400 CALL dumped_.00466A45 ;|<== Get Length(FS)
00420372 . 8B6C24 18 MOV EBP,DWORD PTR SS:[ESP+18]
00420376 . 8D8424 B00000>LEA EAX,DWORD PTR SS:[ESP+B0] ;|<== EAX : FS
0042037D . 8D8C24 500100>LEA ECX,DWORD PTR SS:[ESP+150] ;|<== ECX : FU
00420384 . 50 PUSH EAX ; /Arg2
00420385 . 51 PUSH ECX ; |Arg1
00420386 . 8BCD MOV ECX,EBP ; |
00420388 . E8 53120000 CALL dumped_.004215E0 ; \dumped_.004215E0 ;|<== Encrypt (Trace Into)

===== Trace Into =====

004215E0 /\$ 81EC 80000000 SUB ESP,80
004215E6]. B0 77 MOV AL,77
004215E8]. 53 PUSH EBX ;|<== Length (FU)
004215E9]. 884424 06 MOV BYTE PTR SS:[ESP+6],AL
004215ED]. 884424 1D MOV BYTE PTR SS:[ESP+1D],AL
004215F1]. B3 6C MOV BL,6C
004215F3]. B2 C5 MOV DL,0C5
004215F5]. B1 85 MOV CL,85
004215F7]. B0 E3 MOV AL,0E3
004215F9]. 56 PUSH ESI
004215FA]. 57 PUSH EDI
004215FB]. C64424 0C 49 MOV BYTE PTR SS:[ESP+C],49

00421600 |. C64424 0D EF MOV BYTE PTR SS:[ESP+D],0EF
00421605 |. C64424 0F 1D MOV BYTE PTR SS:[ESP+F],1D
0042160A |. C64424 10 67 MOV BYTE PTR SS:[ESP+10],67
0042160F |. C64424 11 69 MOV BYTE PTR SS:[ESP+11],69
00421614 |. C64424 12 A1 MOV BYTE PTR SS:[ESP+12],0A1
00421619 |. C64424 13 1B MOV BYTE PTR SS:[ESP+13],1B
0042161E |. C64424 14 7A MOV BYTE PTR SS:[ESP+14],7A
00421623 |. C64424 15 8C MOV BYTE PTR SS:[ESP+15],8C
00421628 |. C64424 16 47 MOV BYTE PTR SS:[ESP+16],47
0042162D |. C64424 17 F8 MOV BYTE PTR SS:[ESP+17],0F8
00421632 |. C64424 18 54 MOV BYTE PTR SS:[ESP+18],54
00421637 |. C64424 19 95 MOV BYTE PTR SS:[ESP+19],95
0042163C |. C64424 1A 97 MOV BYTE PTR SS:[ESP+1A],97
00421641 |. C64424 1B 5F MOV BYTE PTR SS:[ESP+1B],5F
00421646 |. C64424 1C 78 MOV BYTE PTR SS:[ESP+1C],78
0042164B |. C64424 1D D9 MOV BYTE PTR SS:[ESP+1D],0D9
00421650 |. C64424 1E DA MOV BYTE PTR SS:[ESP+1E],0DA
00421655 |. 885C24 1F MOV BYTE PTR SS:[ESP+1F],BL
00421659 |. C64424 20 59 MOV BYTE PTR SS:[ESP+20],59
0042165E |. C64424 21 D7 MOV BYTE PTR SS:[ESP+21],0D7
00421663 |. C64424 22 6B MOV BYTE PTR SS:[ESP+22],6B
00421668 |. C64424 23 35 MOV BYTE PTR SS:[ESP+23],35
0042166D |. 885424 24 MOV BYTE PTR SS:[ESP+24],DL
00421671 |. 884C24 26 MOV BYTE PTR SS:[ESP+26],CL
00421675 |. C64424 27 18 MOV BYTE PTR SS:[ESP+27],18
0042167A |. C64424 28 2A MOV BYTE PTR SS:[ESP+28],2A
0042167F |. C64424 29 0E MOV BYTE PTR SS:[ESP+29],0E
00421684 |. C64424 2A 52 MOV BYTE PTR SS:[ESP+2A],52
00421689 |. C64424 2B FF MOV BYTE PTR SS:[ESP+2B],0FF
0042168E |. C64424 2C 00 MOV BYTE PTR SS:[ESP+2C],0
00421693 |. 884424 2D MOV BYTE PTR SS:[ESP+2D],AL
00421697 |. C64424 2E 1B MOV BYTE PTR SS:[ESP+2E],1B
0042169C |. C64424 2F 71 MOV BYTE PTR SS:[ESP+2F],71
004216A1 |. C64424 30 8D MOV BYTE PTR SS:[ESP+30],8D
004216A6 |. C64424 31 34 MOV BYTE PTR SS:[ESP+31],34
004216AB |. C64424 32 63 MOV BYTE PTR SS:[ESP+32],63
004216B0 |. C64424 33 EB MOV BYTE PTR SS:[ESP+33],0EB
004216B5 |. C64424 34 91 MOV BYTE PTR SS:[ESP+34],91
004216BA |. C64424 35 C3 MOV BYTE PTR SS:[ESP+35],0C3
004216BF |. C64424 36 24 MOV BYTE PTR SS:[ESP+36],24
004216C4 |. C64424 37 0F MOV BYTE PTR SS:[ESP+37],0F
004216C9 |. C64424 38 B7 MOV BYTE PTR SS:[ESP+38],0B7
004216CE |. C64424 39 C2 MOV BYTE PTR SS:[ESP+39],0C2
004216D3 |. C64424 3A F8 MOV BYTE PTR SS:[ESP+3A],0F8
004216D8 |. 884424 3B MOV BYTE PTR SS:[ESP+3B],AL
004216DC |. C64424 3C B6 MOV BYTE PTR SS:[ESP+3C],0B6
004216E1 |. C64424 3D 54 MOV BYTE PTR SS:[ESP+3D],54
004216E6 |. C64424 3E 4C MOV BYTE PTR SS:[ESP+3E],4C
004216EB |. C64424 3F 35 MOV BYTE PTR SS:[ESP+3F],35
004216F0 |. C64424 40 54 MOV BYTE PTR SS:[ESP+40],54
004216F5 |. C64424 41 D7 MOV BYTE PTR SS:[ESP+41],0D7
004216FA |. C64424 42 C9 MOV BYTE PTR SS:[ESP+42],0C9

004216FF |. C64424 43 49 MOV BYTE PTR SS:[ESP+43],49
 00421704 |. C64424 44 28 MOV BYTE PTR SS:[ESP+44],28
 00421709 |. C64424 45 A3 MOV BYTE PTR SS:[ESP+45],0A3
 0042170E |. 884C24 46 MOV BYTE PTR SS:[ESP+46],CL
 00421712 |. C64424 47 11 MOV BYTE PTR SS:[ESP+47],11
 00421717 |. C64424 48 0B MOV BYTE PTR SS:[ESP+48],0B
 0042171C |. C64424 49 2C MOV BYTE PTR SS:[ESP+49],2C
 00421721 |. C64424 4A 68 MOV BYTE PTR SS:[ESP+4A],68
 00421726 |. C64424 4B FB MOV BYTE PTR SS:[ESP+4B],0FB
 0042172B |. C64424 4C EE MOV BYTE PTR SS:[ESP+4C],0EE
 00421730 |. C64424 4D 7D MOV BYTE PTR SS:[ESP+4D],7D
 00421735 |. C64424 4E F6 MOV BYTE PTR SS:[ESP+4E],0F6
 0042173A |. 885C24 4F MOV BYTE PTR SS:[ESP+4F],BL
 0042173E |. 884424 50 MOV BYTE PTR SS:[ESP+50],AL
 00421742 |. C64424 51 9C MOV BYTE PTR SS:[ESP+51],9C
 00421747 |. C64424 52 2D MOV BYTE PTR SS:[ESP+52],2D
 0042174C |. C64424 53 E4 MOV BYTE PTR SS:[ESP+53],0E4
 00421751 |. C64424 54 72 MOV BYTE PTR SS:[ESP+54],72
 00421756 |. 884C24 57 MOV BYTE PTR SS:[ESP+57],CL
 0042175A |. 884424 5C MOV BYTE PTR SS:[ESP+5C],AL
 0042175E |. 884C24 76 MOV BYTE PTR SS:[ESP+76],CL
 00421762 |. B0 2B MOV AL,2B
 00421764 |. B1 41 MOV CL,41
 00421766 |. 889C24 890000>MOV BYTE PTR SS:[ESP+89],BL
 0042176D |. 8B9C24 900000>MOV EBX,DWORD PTR SS:[ESP+90]
 00421774 |. 884424 6E MOV BYTE PTR SS:[ESP+6E],AL
 00421778 |. 884424 77 MOV BYTE PTR SS:[ESP+77],AL
 0042177C |. 884C24 78 MOV BYTE PTR SS:[ESP+78],CL
 00421780 |. 884C24 7E MOV BYTE PTR SS:[ESP+7E],CL
 00421784 |. 8BFB MOV EDI,EBX
 00421786 |. 83C9 FF OR ECX,FFFFFF
 00421789 |. 33C0 XOR EAX,EAX
 0042178B |. F2:AE REPNE SCAS BYTE PTR ES:[EDI]
 0042178D |. F7D1 NOT ECX
 0042178F |. 49 DEC ECX
 00421790 |. C64424 55 C3 MOV BYTE PTR SS:[ESP+55],0C3
 00421795 |. 85C9 TEST ECX,ECX
 00421797 |. C64424 56 BB MOV BYTE PTR SS:[ESP+56],0BB
 0042179C |. C64424 58 1A MOV BYTE PTR SS:[ESP+58],1A
 004217A1 |. C64424 59 12 MOV BYTE PTR SS:[ESP+59],12
 004217A6 |. C64424 5A 3C MOV BYTE PTR SS:[ESP+5A],3C
 004217AB |. C64424 5B 32 MOV BYTE PTR SS:[ESP+5B],32
 004217B0 |. C64424 5D 6B MOV BYTE PTR SS:[ESP+5D],6B
 004217B5 |. C64424 5E 4F MOV BYTE PTR SS:[ESP+5E],4F
 004217BA |. C64424 5F 4D MOV BYTE PTR SS:[ESP+5F],4D
 004217BF |. C64424 60 F4 MOV BYTE PTR SS:[ESP+60],0F4
 004217C4 |. C64424 61 A9 MOV BYTE PTR SS:[ESP+61],0A9
 004217C9 |. C64424 62 24 MOV BYTE PTR SS:[ESP+62],24
 004217CE |. C64424 63 C8 MOV BYTE PTR SS:[ESP+63],0C8
 004217D3 |. C64424 64 FA MOV BYTE PTR SS:[ESP+64],0FA
 004217D8 |. C64424 65 78 MOV BYTE PTR SS:[ESP+65],78
 004217DD |. C64424 66 AD MOV BYTE PTR SS:[ESP+66],0AD

004217E2 |. C64424 67 23 MOV BYTE PTR SS:[ESP+67],23
 004217E7 |. C64424 68 A1 MOV BYTE PTR SS:[ESP+68],0A1
 004217EC |. C64424 69 E4 MOV BYTE PTR SS:[ESP+69],0E4
 004217F1 |. C64424 6A 6D MOV BYTE PTR SS:[ESP+6A],6D
 004217F6 |. C64424 6B 9A MOV BYTE PTR SS:[ESP+6B],9A
 004217FB |. C64424 6C 04 MOV BYTE PTR SS:[ESP+6C],4
 00421800 |. C64424 6D CE MOV BYTE PTR SS:[ESP+6D],0CE
 00421805 |. 885424 6F MOV BYTE PTR SS:[ESP+6F],DL
 00421809 |. C64424 70 B6 MOV BYTE PTR SS:[ESP+70],0B6
 0042180E |. 885424 71 MOV BYTE PTR SS:[ESP+71],DL
 00421812 |. C64424 72 EF MOV BYTE PTR SS:[ESP+72],0EF
 00421817 |. C64424 73 93 MOV BYTE PTR SS:[ESP+73],93
 0042181C |. C64424 74 5C MOV BYTE PTR SS:[ESP+74],5C
 00421821 |. C64424 75 A8 MOV BYTE PTR SS:[ESP+75],0A8
 00421826 |. C64424 79 37 MOV BYTE PTR SS:[ESP+79],37
 0042182B |. C64424 7A 72 MOV BYTE PTR SS:[ESP+7A],72
 00421830 |. C64424 7B FA MOV BYTE PTR SS:[ESP+7B],0FA
 00421835 |. C64424 7C 57 MOV BYTE PTR SS:[ESP+7C],57
 0042183A |. C64424 7D 45 MOV BYTE PTR SS:[ESP+7D],45
 0042183F |. C64424 7F A1 MOV BYTE PTR SS:[ESP+7F],0A1
 00421844 |. C68424 800000>MOV BYTE PTR SS:[ESP+80],20
 0042184C |. C68424 810000>MOV BYTE PTR SS:[ESP+81],4F
 00421854 |. C68424 820000>MOV BYTE PTR SS:[ESP+82],80
 0042185C |. C68424 830000>MOV BYTE PTR SS:[ESP+83],0B3
 00421864 |. C68424 840000>MOV BYTE PTR SS:[ESP+84],0D5
 0042186C |. C68424 850000>MOV BYTE PTR SS:[ESP+85],23
 00421874 |. C68424 860000>MOV BYTE PTR SS:[ESP+86],2
 0042187C |. C68424 870000>MOV BYTE PTR SS:[ESP+87],64
 00421884 |. C68424 880000>MOV BYTE PTR SS:[ESP+88],3F
 0042188C |. C68424 8A0000>MOV BYTE PTR SS:[ESP+8A],0F1
 00421894 |. C68424 8B0000>MOV BYTE PTR SS:[ESP+8B],0F
 0042189C |. 7E 4D JLE SHORT dumped_.004218EB
 0042189E |. 8D7424 0C LEA ESI,DWORD PTR SS:[ESP+C]
 004218A2 |. 8BC3 MOV EAX,EBX ;|== EAX : FU
 004218A4 |. 2BF3 SUB ESI,EBX

===== Calculation =====

004218A6 |> 8A10 /MOV DL,BYTE PTR DS:[EAX] ;|== FU[i];
 004218A8 |. 84D2 |TEST DL,DL
 004218AA |. 7D 04 |JGE SHORT dumped_.004218B0
 004218AC |. F6DA |NEG DL
 004218AE |. 8810 |MOV BYTE PTR DS:[EAX],DL
 004218B0 |> 8A1406 |MOV DL,BYTE PTR DS:[ESI+EAX] ;|== reaTable[i];
 004218B3 |. 3210 |XOR DL,BYTE PTR DS:[EAX] ;|== Temp = reaTable[i] ^ FU[i];
 004218B5 |. 80FA 30 |CMP DL,30 ;|== If (Temp < 0x30)
 004218B8 |. 8810 |MOV BYTE PTR DS:[EAX],DL ;|== PTR DS:[EAX] = Temp;
 004218BA |. 7D 12 |JGE SHORT dumped_.004218CE ;|== then
 004218BC |. 0FBED2 |MOVSX EDX,DL ;|== Temp_01 = Temp;
 004218BF |. 83FA 30 |CMP EDX,30 ;|== If (Temp_01 < 0x30)
 004218C2 |. 7D 0A |JGE SHORT dumped_.004218CE ;|== then
 004218C4 |> 83C2 03 |/ADD EDX,3 ;|== Temp_01 = Temp_01 + 0x3
 004218C7 |. 83FA 30 ||CMP EDX,30 ;|== If (Temp_01 < 0x30)
 004218CA |. 8810 ||MOV BYTE PTR DS:[EAX],DL ;|== Then

```

004218CC |.^ 7C F6    \|JL SHORT dumped_.004218C4 ;| <==== Continue Until Temp_01 > 0x30
004218CE |> 8A10    |MOV DL,BYTE PTR DS:[EAX] ;| <==== else Temp
004218D0 |. 80FA 39   |CMP DL,39          ;| <==== If (Temp > 0x39)
004218D3 |. 7E 12     |JLE SHORT dumped_.004218E7 ;| <==== Then
004218D5 |. 0FBED2   |MOVSX EDX,DL      ;| <==== Temp_01 = Temp;
004218D8 |. 83FA 39   |CMP EDX,39        ;| <==== If (Temp_01 >0x39)
004218DB |. 7E 0A     |JLE SHORT dumped_.004218E7 ;| <==== then
004218DD |> 83EA 03   |/SUB EDX,3         ;| <==== Temp_01 = Temp_01 - 0x3
004218E0 |. 83FA 39   ||CMP EDX,39       ;| <==== If (Temp_01 >0x39)
004218E3 |. 8810     ||MOV BYTE PTR DS:[EAX],DL ;| <==== Then
004218E5 |.^ 7F F6   ||JG SHORT dumped_.004218DD ;| <==== Continue Until Temp_01 < 0x39
004218E7 |> 40       |INC EAX           ;| <==== else i++
004218E8 |. 49       |DEC ECX            ;| <==== Length(FU) = Length(FU) - 1
004218E9 |.^ 75 BB   |JNZ SHORT dumped_.004218A6

/* Đoạn tính toán này thực hiện việc lấy từng kí tự trong chuỗi FU đem Xor với các giá trị mặc định ở trên (tạm gọi là reaTable[i]). Quá trình thực hiện nhằm tạo ra các số nằm trong khoảng từ 0 – 9 tương ứng với từng kí tự nhập vào. Chuỗi số này có chiều dài đúng bằng chiều dài của FU . Sau quá trình tính toán trên chúng ta sẽ có được một dãy số (ví dụ : kienmanowar thì tmpSerial : 11071228125). Tại sao ở đây mình chỉ coi chuỗi số sau khi đã mã hóa được là tmpSerial , mọi thứ đều có nguyên do của nó . Lúc đầu em cứ tưởng cái chuỗi tmpSerial trên là Right Serial nên em sướng quá. Tắt ngay Olly , rung đùi ngồi nhập .... nó cũng hiện ra cái Nag “Thank you.....” trên hihi. Nhưng đến khi em run lại thì nó lại bắt nhập lại cú không chịu được . Mở Olly Trace tiếp thì hối ôi.....một cú lừa ngoan mục như anh Moon đã nói*/

```

===== Calculation =====

```

004218EB |> 8BB424 940000>MOV ESI,DWORD PTR SS:[ESP+94] ;| <== ESI : FS
004218F2 |. 8BC3     MOV EAX,EBX          ;| <== EAX : tmpSerial
004218F4 |> 8A10    /MOV DL,BYTE PTR DS:[EAX] ;| <== tmpSerial[i]
004218F6 |. 8A1E    |MOV BL,BYTE PTR DS:[ESI] ;| <== FS [i]
004218F8 |. 8ACA    |MOV CL,DL
004218FA |. 3AD3    |CMP DL,BL          ;| <== tmpSerial [i] = FS[i]
004218FC |. 75 1E    |JNZ SHORT dumped_.0042191C
004218FE |. 84C9    |TEST CL,CL
00421900 |. 74 16    |JE SHORT dumped_.00421918
00421902 |. 8A50 01  |MOV DL,BYTE PTR DS:[EAX+1] ;| <== tmpSerial [i+1]
00421905 |. 8A5E 01  |MOV BL,BYTE PTR DS:[ESI+1] ;| <== FS[i];
00421908 |. 8ACA    |MOV CL,DL
0042190A |. 3AD3    |CMP DL,BL          ;| <== Compare
0042190C |. 75 0E    |JNZ SHORT dumped_.0042191C
0042190E |. 83C0 02  |ADD EAX,2          ;| <== i = i + 2
00421911 |. 83C6 02  |ADD ESI,2
00421914 |. 84C9    |TEST CL,CL
00421916 |.^ 75 DC   |JNZ SHORT dumped_.004218F4 ;|
00421918 |> 33C0    XOR EAX,EAX        ;| <== EAX =0x0
0042191A |. EB 05    JMP SHORT dumped_.00421921
0042191C |> 1BC0    SBB EAX,EAX
0042191E |. 83D8 FF  SBB EAX,-1
00421921 |> 33C9    XOR ECX,ECX        ;| <== ECX =0x0
00421923 |. 5F      POP EDI
00421924 |. 85C0    TEST EAX,EAX
00421926 |. 0F94C1  SETE CL          ;| <== CL = 0x1
00421929 |. 5E      POP ESI

```

```

0042192A |. 8BC1      MOV EAX,ECX          ;| <== EAX = ECX = 0x1
0042192C |. 5B        POP EBX
0042192D |. 81C4 80000000 ADD ESP,80
00421933 \. C2 0800  RETN 8

```

==== Trace Into =====

```

0042038D . 85C0      TEST EAX,EAX         ; <== EAX = 0x0 ?
0042038F ./4 0F       JE SHORT dumped_.004203A0
00420391 . |83FB 04    CMP EBX,4           ; <== Length (FU) > 0x4 (FU at least 4 charts)
00420394 . |C705 DCF64900>MOV DWORD PTR DS:[49F6DC],1
0042039E . |73 0A       JNB SHORT dumped_.004203AA
004203A0 > \C705 DCF64900>MOV DWORD PTR DS:[49F6DC],0
004203AA > 8D9424 B00000>LEA EDX,DWORD PTR SS:[ESP+B0]
004203B1 . 8BCD      MOV ECX,EBP
004203B3 . 52        PUSH EDX
004203B4 . E8 87150000 CALL dumped_.00421940 ;| <== Ác ặc điểm quan trọng của chương trình
nó nằm ở đây đây các bác oi (Trace Into)

```

==== Trace Into =====

.....

```

004219A0 |. BF 9CB54900 MOV EDI,dumped_.0049B59C      ; ASCII
"JGDDCCGJ,DJDGCFGJ,JFJDGCJC,DJGCGEFE,EEECDCGE,GEDCCEDJ,CCGJDEEE,XEGJGDEW
,GGCDJGEX,EECJJDC,JADCFeed,GGJCEDED,JGDJJGCC,GGCDEJJG,CEDJDEJG,EFEDDDDE,
DJEJCEDC,DEDGGDGG,GDDWCDEC,CJGGEDDG,ACDDGFDG,EFCEFCGC,DGJEEEGJ,DCFVG
GGD,GGGCJECD,GDGGEJGC,GDD"...

```

.....

```

004219CA |. BF BCAD4900 MOV EDI,dumped_.0049ADBC      ; ASCII
"E0CICGEC,GFACCGGB,EEG0GGAC,DFGEDE0F,ECCGECAA,CADCAGGC,FD0DGDDG,BEGAD
FBB,GAFC0GEA,GFA0E00C,FD2CG4D4,DFDFDDG0,CEEFDCDC,BCGDGGG0,GECABGE0,EDG
GDC0G,GC05C5DF,CDCAEEEA,M4U43688,GD0FED0A,CF0EG0EG,G0C0DEBG,ECEEGCCE,GEC
FGGCF,BF2CCG1C,GCEFCCDC,GCC"...

```

.....

```

004219F3 |. BF E4A54900 MOV EDI,dumped_.0049A5E4      ; ASCII
"DC43D0D4,DE4CD30D,G5CLEXLA,443DD3D3,340A4AD3,4E1D334D,A4014D43,04DED4D4,D43
0F440,13DA144D,DDDD1D34,A04B4AD3,4BC4DD34,3AC03314,DDC3C4B1,3D4D3434,1CB4BE03
,AF3FC44C,A14BB4BD,4334B0BC,4034BD0D,DD44301C,F43DA044,DDD34D31,D0D11ADD,B00D
3F00,444"...

```

.....

```

00421A1C |. BF B0A24900 MOV EDI,dumped_.0049A2B0      ; ASCII
"33F33B3E,KCDAFGGA,FF3GEFEB,230F3A23,DGG8BFG3,F3E03CCD,D22FGDF3,CFBE8323,F83
GF3GE,GG8GB223,3A22F2B2,GBD3FD33,FCCC20G2,B833F323,2FDA3F8F,32FC338G,3FC8FF3F,
F3320CDF,C232BD32,00F833F2,2C3D8CF0,FC0FFF3,32F32FF,3CG305B3,AC3GCFDC,30DF3EC
2,GCF"...

```

.....

```

00421A3C |. BF D89A4900 MOV EDI,dumped_.00499AD8      ; ASCII
"LXDLGXKD,HB517EAF,5HHA5F7E,C7BE6FAB,F7FACA5A,EDGAEGA6,A7F1HCHE,FDEDECA
C,CAIFEIAG,GB517H5,35A4F1FF,7F731A5G,7FCACCB7,BAIFAG17,AA6EC73G,G57EE7FD,GC3
63DFF,7EAECH7I,B4AAAHH5A,7BBBDF36,CFGCFEFE,5AH7AF1C,EAAE7FA1,C300C0H0,EFA1G7
A6,547176DA,7HH"...

```

.....

/ Note : Trên đây chính là các **Serial** mặc định của chương trình này. Nó sẽ lấy các **default Serial** để so sánh với FS mà chúng ta đã nhập vào , thông tin của FS được lưu trong file mgutil_win.ini . Nếu trùng thì Okihic hic nhiều vật vã */*

==== Trace Into =====

004203B9 . 85C0 TEST EAX,EAX ;| <== If (tmpSerial == defaultSerial)
 004203BB . 0F85 EE010000 JNZ dumped_.004205AF ;| <== Then Jump to Accept and Register
 Code
 */**/* Serial tương ứng : Có thể dùng bất kì một Serial nào trong các Serial trên ứng với UserName mà bạn gõ vào :

User Name : kienmanowar

Serial : LXDLGXKD

hoặc

User Name : REA-cRaCkErTeAm

Serial : 33F33B3E

IV – KeyGen :

N/A

V – End of Tut :

- Finished – November 13, 2004

Reverse Engineering Association SoftWare

Homepage :	http://www.magictweak.com
Production :	Efreesky Software
SoftWare :	Privacy Inspector v1.51
Copyright by :	Copyright (C) 2000 - 2004 Efreesky Software . All Rights Reserved.
Type :	Name/Serial
Packed :	PECompact 1.68 - 1.84 -> Jeremy Collake
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsm 10, Import REC v1.6, LordPE v1.4
Unpack :	Manual
Request :	Correct Serial

Privacy Inspector v1.51

Do you have a spouse, children, or a boss who you do not want knowing what you were doing on the computer? Why worry about other people using the computer after you. You need **Privacy Inspector** before someone else gets your personal information from your computer. Privacy Inspector cleans up your tracks left by Windows, your browser and many other programs. Save storage space and improve performance of your computer.

I – Information :

**** Dùng PeiD v0.92 để Detect , chúng ta biết được chương trình này đã được tác giả Pack lại bằng **PECompact 1.68 - 1.84 -> Jeremy Collake**. Vậy là khó cho chúng ta rồi, nhưng không sao có khó mới ló cái khôn. Chúng ta sẽ thử UnPack thắng này xem sao .

**** Run thử chương trình xem có gì đặc biệt , ngay lập tức một Nag Screen bắn ra : “**Enter the registration here**” , tại Nag Screen này chúng ta sẽ thấy có hai textbox bắt nhập **User Name** và **Registration Code** . Vậy là cách thức bảo vệ của chương trình là N/S. Nếu nhập đúng thì Oki. Tại thời điểm này chúng ta chưa có được nêu đành click tạm vào Button **Continue Evaluation** để xem tiếp thông tin.

-**** Sau khi click chúng ta sẽ đến màn hình chính của chương trình , trên title bar chúng ta thấy một dòng như sau : **Privacy Inspector Unregistered – Day 1 of 15**. Vậy có nghĩa là nếu chúng ta không đăng kí hợp pháp thì chúng ta sẽ chỉ được dùng thử các tính năng của chương trình trong vòng 15 days mà thôi. Quá date là “anh ơi ở lại em đi nhé hiiiiiii”. Một điều nữa muốn nói thêm là chương trình này về mặt User Interface trông rất bắt mắt good looking nên giá của nó cũng hơi bị đắt **29.95\$**.

II – UnPacking :

-**** Hoàn toàn tương tự như chúng ta đã làm với soft **Magic Tweak** của hãng này

III – Cracking :

-**** Bây giờ đến giai đoạn tiếp theo . Chúng ta sẽ tiến hành cracking trên file **dumped_.exe**. Run file này lên , Nag Screen bắn ra yêu cầu chúng ta nhập **Name** và **Registration Code** . Uh thì nó bảo nhập thì ta nhập , nhập đại một FU và FS vào xem sao. Ở đây mình nhập là : (**User Name : kienmnnowar Registration Code : 11111982**). Sau đó nhấn Register , một Nag thông báo hiện lên làm cho người ta mừng hụt tưởng là nó đồng ý rồi : **“Thanks for your registering. Please restart the program to validate the registration code”** . Nhấn Oki , sau đó Run lại chương trình , ặc ặc Màn hình bắt đăng kí lại hiện ra kia .

-**** Các soft của cùng một hãng sẽ có cùng một kiểu Protect . Cho nên đối với soft này mọi thông tin mà chúng ta nhập vào sẽ được chương trình ghi nhận lại vào file **“privacy_win.ini”**. Vậy chúng ta sẽ tìm đến đoạn code đọc thông tin từ file **“privacy_win.ini”**. Chúng ta tìm thấy tại đây trong Olly :

004372FF . 68 5C614900 PUSH dumped_.0049615C ; ASCII "privacy_win.ini"

-**** Set BP tại địa chỉ trên , sau đó chúng ta Run lại chương trình trong Olly (F9). Chương trình sẽ dừng tại đây

```

004372FF . 68 5C614900 PUSH dumped_.0049615C ; ASCII
"privacy_win.ini"<==We're here
00437304 . 8D8424 300100>LEA EAX,DWORD PTR SS:[ESP+130]
0043730B . 6A 28 PUSH 28
0043730D . 50 PUSH EAX
0043730E . 68 B8CB4900 PUSH dumped_.0049CBB8
00437313 . 68 50614900 PUSH dumped_.00496150 ; ASCII "UserName"
00437318 . 68 4C614900 PUSH dumped_.0049614C ; ASCII "REG"
0043731D . C68424 481B00>MOV BYTE PTR SS:[ESP+1B48],0C
00437325 . FFD7 CALL EDI
00437327 . 68 5C614900 PUSH dumped_.0049615C ; ASCII "privacy_win.ini"
0043732C . 8D8C24 E00000>LEA ECX,DWORD PTR SS:[ESP+E0]
00437333 . 6A 28 PUSH 28
00437335 . 51 PUSH ECX
00437336 . 68 B8CB4900 PUSH dumped_.0049CBB8
0043733B . 68 F4864900 PUSH dumped_.004986F4 ; ASCII "RegCode"
00437340 . 68 4C614900 PUSH dumped_.0049614C ; ASCII "REG"
00437345 . 8BD8 MOV EBX,EAX
00437347 . FFD7 CALL EDI
00437349 . 8D9424 2C0100>LEA EDX,DWORD PTR SS:[ESP+12C]
00437350 . 8D4C24 30 LEA ECX,DWORD PTR SS:[ESP+30]
00437354 . 52 PUSH EDX
00437355 . E8 C7D30200 CALL dumped_.00464721
0043735A . 8D8424 DC0000>LEA EAX,DWORD PTR SS:[ESP+DC]
```

```

00437361 . 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
00437365 . 50 PUSH EAX
00437366 . E8 B6D30200 CALL dumped_.00464721
0043736B . 8B6C24 28 MOV EBP,DWORD PTR SS:[ESP+28]
0043736F . 8D8C24 DC0000>LEA ECX,DWORD PTR SS:[ESP+DC]
00437376 . 8D9424 2C0100>LEA EDX,DWORD PTR SS:[ESP+12C]
0043737D . 51 PUSH ECX ; /Arg2
0043737E . 52 PUSH EDX ; | Arg1
0043737F . 8BCD MOV ECX,EBP ; |
00437381 . E8 0A0A0000 CALL dumped_.00437D90 ; \dumped_.00437D90
00437386 . 85C0 TEST EAX,EAX
00437388 . 74 0F JE SHORT dumped_.00437399
0043738A . 83FB 04 CMP EBX,4
0043738D . C705 2CD94900>MOV DWORD PTR DS:[49D92C],1
00437397 . 73 0A JNB SHORT dumped_.004373A3
00437399 > C705 2CD94900>MOV DWORD PTR DS:[49D92C],0
004373A3 > 8D8424 DC0000>LEA EAX,DWORD PTR SS:[ESP+DC]
004373AA . 8BCD MOV ECX,EBP
004373AC . 50 PUSH EAX

```

004373AD . E8 3E0D0000 CALL dumped_.004380F0 <== (Let's trace Into , All Serial are here)

===== Trace Into =====

.....
00499B78=dumped_.00499B78 (ASCII)

"0CJI77CF,0F1G75AF,GG0A43F7,9FG3D85H,88H7J7IC,F8073C0D,D7D9188C,88ADB086,FC3A0C
CE,CC5G9DI7,9B05DCHA,GAF3DJ8I,D40D3AF7,AA5FCF3A,78IDIF9H,8G8EAJAF,I0D93H05,1H0
4BF4A,CJJG71H7,20BB8EAG,B7JGIG0V,FG7853G5,JAFHD849,A55IADDAA,VGG)

.....
00499398=dumped_.00499398 (ASCII)

"3BGDAEE0,EJAH5E05,CAGH73AF,IIEAD7HI,JHVECC3A,6ADVGE70,DI9B3DB3,B5E777H3,03IC
9G73,3IA9B1H5,I7GH03DE, ID8D82G1,VCCJ8HDH,5FG8G3EF,G70AG3A9,B2E0861E,BIAJE8J4,V
D6A77A3,7DDAIJCG,GI1JC8FA,FIH771J3,G969BA5G,BVBCGH57,GA5J0ECD,E8F)

.....
00498BC0=dumped_.00498BC0 (ASCII)

"B0JGB7EJ,55481G73,5HHF5BI9,2JJE3G1J,GF7A2A5A,3IHF9G03,AI78HB49,7CEIE212,3J9ED0IE,
HJ568FH5,3EB381G3,6EJVDA5G,FBC0BBJF,JA0EAG8F,FF898JVB,G5FE9FGC,HCVDVI7G,6DAD
C4J0,A7CJAFIH,F9JCB7V8,27G2BEEE,5FHFA78C,D0AE6ECD,FE5J03J8,EEF)

.....
00498894=dumped_.00498894 (ASCII)

"33V73B1E,FG0JBBC6,7J0EE6EB,I3HJ12A1,DAA5BIA0,J10HD8C9,94I6ABV1,8VB051I3,35AJDA
0,4AH30IEG,D2AIVGBG,8BBDVB33,VHC84HEI,B5J76JG7,G3D20353,0GJ8DF5A,F7C53I3I,7FFIF8
B6,EJF5FG12,HH751F6A,ICD958IH,JCHV67A3,VDG3J436,H88JH0BD,BVF)

===== Trace Into =====

004373B2 . 85C0 TEST EAX,EAX

004373B4 . 0F85 3D020000 JNZ dumped_.004375F7

/*/*/*/ Serial tương ứng : Có thể dùng bất kì một Serial nào trong các Serial trên ứng với UserName mà bạn gõ vào :

User Name : kienmanowar

Serial : D7D9188C

hoặc

User Name : REA-cRaCkErTeAm

Serial : GF7A2A5A

IV – KeyGen :

N/A

V – End of Tut :- Finished – *November 17, 2004*

Reverse Engineering Association

SoftWare

Homepage :	http://www.cmfperception.com
Production :	CmfPerception, Inc
SoftWare :	AutoZIP Backup v1.1
Copyright by :	Copyright (C) 2004 CmfPerception, Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

AutoZIP Backup v1.1

AutoZIP Backup is professional software that lets you automate the backup of your precious files to a ZIP archive. Although the interface is very user-friendly we recommend you read the help file so that you can take advantage of all the features this program has to offer

I – Information :

- Dùng PEid v0.92 để Detect , chúng ta biết chương trình ko bị pack & được viết bằng **Borland Delphi 4.0 - 5.0**. Dùng tiếp Plugin Kanal của PEid để Detect Crypto thì ta thấy hàng loạt 14 Crypto Signatures như sau: **BASE64 table , BlowFish [sbox] , CRC 32, DES [key schedule] , Rijndeal , Ripemd , Square , TowFish , Zip 2.....** Ấc ặc choáng , lúc sau mới hoàn hồn :((. Hihi..... Sau khi lấy lại được bình tĩnh thì mới phát hiện ra đó toàn là đồ trang trí , giả hết . Đúng là quân lừa đảo, may mà mình vẫn còn tỉnh táo .

- Chạy thử chương trình , chúng ta thấy hiện ra một 1 Dialog Box bắt chúng ta phải đăng kí . Nếu ko nó chỉ cho phép chúng ta dùng thử trong 30 days . Và nếu chúng ta muốn tiếp tục sử dụng thì chúng ta phải mua nó với giá **\$34.95 USD** . ặc ặc sao đắt thế . Ở phía trên , thanh Title của soft còn có chữ Unregistered , nhìn thấy mà ghét . Vào menu Help/ Register/Order... ta nhập đại vào 1 cái Email xem nó bảo sao . Ở đây mình nhập là : **Email:hoa_dong_noi_06@yahoo.com & Code : 0361985** . Nhấn Button Register . Oh ! 1 cái nag sẽ văng ra , đập ngay vào mặt chúng ta : **" Name and code do not match "** . OK ! hãy nhớ lấy nó , chúng ta sẽ dùng chính nó để thịt nó . Hihiiii....!

- Bây giờ chúng ta hãy load chương trình vào trong OllyDbg . Sau đó click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . Ấc ặc..... mất 1 phút mới tìm thấy nó . Double click vào dòng đó , chúng ta sẽ quay trở lại màn hình chính của Olly :
004CF601 > \B8 C0F64C00 MOV EAX, autozipb.004CF6C0 ; ASCII "Name and Code do not match.\n" ==> We're here

004CF606 . E8 D59FF8FF CALL autozipb.004595E0

- Lần ngược lên trên 1 chút , chúng ta sẽ đặt BreakPoint tại đây :

004CF515 . 8BC3 MOV EAX, EBX

004CF517 . E8 8CF8FFFF CALL autozipb.004CEDA8 ==> Set BreakPoint here

004CF51C . 8B45 F8 MOV EAX, DWORD PTR SS:[EBP-8]

II – Cracking :

- OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình .Chúng ta sẽ nhập lại Email và code như trên . Nhấn Register, Olly sẽ Ice tiến trình tại điểm mà ta vừa đặt BP .

004CF517 . E8 8CF8FFFF CALL autozipb.004CEDA8 ;<== We're here

-----Trace Into-----

```

004CEDA8 /$ 55      PUSH  EBP
004CEDA9 ]. 8BEC    MOV    EBP, ESP
004CEDAB ]. 6A 00   PUSH   0
004CEDAD ]. 6A 00   PUSH   0
004CEDAF ]. 6A 00   PUSH   0
004CEDB1 ]. 6A 00   PUSH   0
004CEDB3 ]. 6A 00   PUSH   0
004CEDB5 ]. 53     PUSH   EBX
004CEDB6 ]. 56     PUSH   ESI
004CEDB7 ]. 8BF1    MOV    ESI, ECX
004CEDB9 ]. 8955 FC  MOV    [LOCAL.1], EDX ;<== Input
004CEDBC ]. 8BD8    MOV    EBX, EAX
004CEDBE ]. 8B45 FC  MOV    EAX, [LOCAL.1] ;<== đưa Input vào EAX
004CEDC1 ]. E8 F252F3FF CALL  autozipb.004040B8
004CEDC6 ]. 33C0    XOR    EAX, EAX ;<== EAX =0
004CEDC8 ]. 55     PUSH   EBP
004CEDC9 ]. 68 A9EE4C00 PUSH  autozipb.004CEE9
004CEDCE ]. 64:FF30  PUSH  DWORD PTR FS:[EAX]
004CEDD1 ]. 64:8920  MOV   DWORD PTR FS:[EAX], ESP
004CEDD4 ]. 837D FC 00  CMP   [LOCAL.1], 0 ==> Len(Input) >0
004CEDD8 ]. 75 0C    JNZ   SHORT autozipb.004CEDE6
004CEDDA ]. 8BC6    MOV   EAX, ESI
004CEDDC ]. E8 A34EF3FF CALL  autozipb.00403C84
004CEDE1 ]. E9 A8000000 JMP   autozipb.004CEE8E
004CEDE6 > 8D4D EC  LEA   ECX, [LOCAL.5]
004CEDE9 ]. BA 07000000 MOV   EDX, 7
004CEDEE ]. 8B83 14030000 MOV   EAX, DWORD PTR DS:[EBX+314] ;<== đưa String ASCII
"AutoZIP Backup" vào EAX
004CEDF4 ]. E8 B3B0F8FF CALL  autozipb.00459EAC ;<== gọi hàm mã hoá .
004CEDF9 ]. 8B45 EC  MOV   EAX, [LOCAL.5] ;<== đưa String (ASII"AutoZip") sau
khi khi đã mã hoá vào EAX
004CEDFC ]. 8D55 F0  LEA   EDX, [LOCAL.4]
004CEDFF ]. E8 489DF3FF CALL  autozipb.00408B4C ;<== tiếp tục gọi hàm mã hoá String
004CEE04 ]. 8B45 F0  MOV   EAX, [LOCAL.4]
004CEE07 ]. BA C0EE4C00 MOV   EDX, autozipb.004CEEC0 ; ASCII "autozip"
004CEE0C ]. E8 0352F3FF CALL  autozipb.00404014 ;<== gọi hàm mã hoá String ASCII "autozip"
004CEE11 ]. 75 0F    JNZ   SHORT autozipb.004CEE22
004CEE13 ]. 8D45 F8  LEA   EAX, [LOCAL.2]
004CEE16 ]. BA D0EE4C00 MOV   EDX, autozipb.004CEED0 ; ASCII "ZBackup:Try BUYING!!"

```

004CEE1B |. E8 FC4EF3FF CALL autozipb.00403D1C ==> gọi hàm mã hoá String ASCII "ZBackup:Try BUYING!!"
 004CEE20 |. EB 0D JMP SHORT autozipb.004CEE2F
 004CEE22 |> 8D45 F8 LEA EAX, [LOCAL.2]
 004CEE25 |. BA F0EE4C00 MOV EDX, autozipb.004CEEFO ; ASCII "Backup: Try BUYING!!"
 004CEE2A |. E8 ED4EF3FF CALL autozipb.00403D1C
 004CEE2F |> BB 01000000 MOV EBX, 1

Start Loop

004CEE34 |> 8B45 FC /MOV EAX, [LOCAL.1] ==> đưa Input vào EAX
 004CEE37 |.E8 C850F3FF |CALL autozipb.00403F04
 004CEE3C |.50 |PUSH EAX
 004CEE3D |.8BC3 |MOV EAX, EBX
 004CEE3F |.48 |DEC EAX
 004CEE40 |.5A |POP EDX
 004CEE41 |.8BCA |MOV ECX, EDX
 004CEE43 |.99 |CDQ
 004CEE44 |.F7F9 |IDIV ECX
 004CEE46 |.8B45 FC |MOV EAX, [LOCAL.1] ==> Input
 004CEE49 |.8A0410 |MOV AL, BYTE PTR DS:[EAX+EDX] ==> chuyển S[0] của Input vào AL
 004CEE4C |.8B55 F8 |MOV EDX, [LOCAL.2] ==> đưa String ASCII "Backup: Try BUYING!!" vào EDX
 004CEE4F |.8A541A FF |MOV DL, BYTE PTR DS:[EDX+EBX-1] ;=> chuyển S[0] của String vào DL
 004CEE53 |.32C2 |XOR AL, DL ==> Xor 2 char trên .
 004CEE55 |.25 FF000000 |AND EAX, 0FF
 004CEE5A |.8D55 F4 |LEA EDX, [LOCAL.3]
 004CEE5D |.E8 7AA0F3FF |CALL autozipb.00408EDC
 004CEE62 |.8B45 F4 |MOV EAX, [LOCAL.3]
 004CEE65 |.E8 9A50F3FF |CALL autozipb.00403F04
 004CEE6A |.8B55 F4 |MOV EDX, [LOCAL.3]
 004CEE6D |.FF7402 FF |PUSH DWORD PTR DS:[EDX+EAX-1]
 004CEE71 |.8D45 F8 |LEA EAX, [LOCAL.2]
 004CEE74 |.E8 5B52F3FF |CALL autozipb.004040D4
 004CEE79 |.5A |POP EDX
 004CEE7A |.885418 FF |MOV BYTE PTR DS:[EAX+EBX-1], DL ==> chuyển S[1] vào DL
 004CEE7E |.43 |INC EBX ==> i++
 004CEE7F |.83FB 15 |CMP EBX, 15 ==> If i<15
 004CEE82 |.^ 75 B0 |JNZ SHORT autozipb.004CEE34 ==> then Loop

End Loop

004CEE84 |.8BC6 MOV EAX, ESI
 004CEE86 |.8B55 F8 MOV EDX, [LOCAL.2] ==> đưa Real Serial sau khi đã mã hoá vào EDX
 004CEE89 |.E8 4A4EF3FF CALL autozipb.00403CD8 ==> gọi hàm kiểm tra 1 lần nữa .
 004CEE8E |> 33C0 XOR EAX, EAX
 004CEE90 |.5A POP EDX
 004CEE91 |.59 POP ECX
 004CEE92 |.59 POP ECX
 004CEE93 |.64:8910 MOV DWORD PTR FS:[EAX], EDX
 004CEE96 |.68 B0EE4C00 PUSH autozipb.004CEEBO ; ASCII "^[å]Ã"
 004CEE9B |> 8D45 EC LEA EAX, [LOCAL.5]
 004CEE9E |.BA 05000000 MOV EDX, 5
 004CEEA3 |.E8 004EF3FF CALL autozipb.00403CA8
 004CEEA8 \. C3 RETN

```
004CF51C . 8B45 F8      MOV    EAX, DWORD PTR SS:[EBP-8] ==> đưa Real Serial vào EAX .
004CF51F . 50          PUSH   EAX
```

/*/*/* - SERIAL tương ứng :

Email : hoa_dong_noi_06@yahoo.com Serial : 05005603182391195838

Hoặc :

Email : REA-cRaCkErTeAm@yahoo.com Serial : 87288971351690295838

III/ End Tut :

- Finished : 19/12/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.jklsoft.com
Production :	JKLNSoft , Inc
SoftWare :	Batch Image Resizer v2.01
Copyright by :	Copyright (C) 2004 JKLNSoft.com. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N/A
Request :	Correct Serial / KeyGen

Batch Image Resizer v2.01

I – Information :

- Dùng PEid v0.92 để Detect , chúng ta biết chương trình ko bị pack & được viết bằng Microsoft Visual C++ 6.0. Phù thê là ko phải unpack --> sướng . hihiii..... :)

- Chạy thử chương trình , chúng ta thấy hiện lên một 1 Dialog Box :"Wellcome to Batch Image Resizer ". Nhìn xuống dưới 1 chút ta thấy dòng : "Try to register...." , nhìn thấy mà tức cả mắt :((. Nó đồng nghĩa với việc họ chỉ cho chúng ta dùng thử nếu chúng ta ko Register (tức là ta phải đưa money cho họ đó bạn . Đưa money để buy soft , chúng ta đâu có thói quen này . Và lần này cũng vậy thôi "no Money, no Register , and Free all " ==> đó là mục tiêu của chúng ta . Hihiiii.....! :D

- OK ! típ nhé . Ở đây ta còn thấy 1 button : Continue , click thử vào xem sao . Oh ! Cái đầu tiên chúng ta cần tìm là chỗ họ bắt ta Register , nhìn xuống dưới , Oh !đây rùi , button :Enter Code . Click vào đó , chúng ta sẽ điền các thông tin cần thiết xem nó bảo sao . Ở đây mình nhập là :Email :hoa_dong_noi_06@yahoo.com & Code : 0361985 . Nhấn OK , một cái Nag văng ra , nó làm ta mừng hụt : "Thanks for Registratinon. Please restart Batch Image Resizer ". Nhấn tiếp OK , mở lại chương trình ta vẫn thấy hiện Dialog box bắt đăng ký . Như vậy ta suy đoán chương trình sẽ lưu thông tin ta Registered vào Reg rồi check nó mỗi khi ta Run chương trình . . :)

- Ok ! nghĩ sao làm vậy .Bây giờ chúng ta hãy load chương trình vào trong OllyDbg . Nếu nó hiện ra 1 bảng thông báo thì cứ nhấn OK . Sau đó click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . ah ! nó đây rùi , double click vào String đó ta được đưa trở lại Olly . Ấc ..(((nhìn lướt qua mà thấy toàn là hàm Call MFC , ít manh mối quá . Để coi trong mớ bòng bong String có gì đáng giá ko ? ở đây chúng ta tìm thấy 1 loạt String như sau :

Text strings referenced in BatchIma:.text

Address	Disassembly	Text string
00405AAB	PUSH BatchIma.0041D35C	ASCII "Email"
00405AB4	PUSH BatchIma.0041D348	ASCII "Registration Info"
00405AC6	PUSH BatchIma.0041D334	ASCII "Registration Code"
00405ACF	PUSH BatchIma.0041D348	ASCII "Registration Info"
00405AEC	PUSH BatchIma.0041D494	ASCII "bono@mail.com"
00405B12	PUSH BatchIma.0041D480	ASCII "54QEKKCHERPL6X8Q"
00405C33	PUSH BatchIma.0041D4B8	ASCII "aaaaaaaa"
00405D98	PUSH BatchIma.0041D4A4	ASCII "0000000000000000"

- Chúng ta thấy 2 Template String :

00405AEC	PUSH BatchIma.0041D494	ASCII "bono@mail.com"
00405B12	PUSH BatchIma.0041D480	ASCII "54QEKKCHERPL6X8Q"

Một ý nghĩ chạy qua trong đầu : biết đâu 2 Template String này lại là RU & RS thì sao ? nghĩ sao làm vậy , ta chạy chương trình Batch Image , nhập vào 2 Template String trên , hiện nag , restart lại chương trình , chắc chắn .. nó vẫn bắt ta Register , như vậy 2 String trên là ko đúng rồi . Trở lại Olly , ta cứ thử Double click vào Temp String trên xem có khai thác được gì từ đây ko ? Ta được đưa đến đoạn Code sau :

```
00405B12 |. 68 80D44100 PUSH BatchIma.0041D480 ; ASCII "54QEKKCHERPL6X8Q" ==>
We're here . [/B] [/COLOR]
00405B17 |. 50      PUSH EAX
00405B18 |. FFD7    CALL NEAR EDI
00405B1A |. 83C4 08 ADD  ESP, 8
00405B1D |. 85C0    TEST EAX, EAX
00405B1F |. 75 0E    JNZ  SHORT BatchIma.00405B2F
00405B21 |. 68 04DB4100 PUSH BatchIma.0041DB04
```

- Nhìn xuống phía dưới 1 chút chúng ta sẽ Set BreakPoint tại đây :

```
00405B58 |. 50      PUSH EAX           ; |Arg1
00405B59 |. E8 82000000 CALL  BatchIma.00405BE0       ; \BatchIma.00405BE0 ==> Set
BreakPoint here .
```

II – Cracking :

OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình , Olly sẽ Ice tiến trình tại điểm mà ta vừa đặt BP . Chúng ta sẽ Trace Into vào trong hàm Call này nếu bạn muốn xem cơ chế tạo Ser của nó . có thể trace over cũng lấy dc RS :

```
00405B59 |. E8 82000000 CALL  BatchIma.00405BE0 ; \BatchIma.00405BE0 ==> We're here .
```

-----Trace Into-----

```
00405BE0 /$ 6A FF      PUSH -1
00405BE2 |. 68 D74A4100 PUSH BatchIma.00414AD7      ; SE handler installation
00405BE7 |. 64:A1 0000000>MOV  EAX, DWORD PTR FS:[0]
00405BED |. 50      PUSH EAX
00405BEE |. 64:8925 00000>MOV  DWORD PTR FS:[0], ESP
00405BF5 |. 83EC 2C    SUB  ESP, 2C
```

```

00405BF8 |. 53      PUSH    EBX
00405BF9 |. 56      PUSH    ESI
00405BFA |. C74424 0C 000>MOV    DWORD PTR SS:[ESP+C], 0
00405C02 |. 8D4424 48  LEA     EAX, DWORD PTR SS:[ESP+48]
00405C06 |. BB 01000000 MOV    EBX, 1
00405C0B |. 50      PUSH    EAX
00405C0C |. 8D4C24 0C  LEA     ECX, DWORD PTR SS:[ESP+C]
00405C10 |. 895C24 40  MOV    DWORD PTR SS:[ESP+40], EBX
00405C14 |. E8 25DF0000 CALL   <JMP.&MFC42.#535>
00405C19 |. 8B4C24 08  MOV    ECX, DWORD PTR SS:[ESP+8] ==> đưa Input vào ECX .
00405C1D |. C64424 3C 02  MOV    BYTE PTR SS:[ESP+3C], 2
00405C22 |. 8B41 F8  MOV    EAX, DWORD PTR DS:[ECX-8]
00405C25 |. 83F8 0A  CMP    EAX, 0A
00405C28 |. 0F8E 98010000 JLE    BatchIma.00405DC6
00405C2E |. 83F8 10  CMP    EAX, 10
00405C31 |. 7D 0E  JGE    SHORT BatchIma.00405C41
00405C33 |. 68 B8D44100 PUSH   BatchIma.0041D4B8 ; ASCII "aaaaaaaa"
00405C38 |. 8D4C24 0C  LEA     ECX, DWORD PTR SS:[ESP+C]
00405C3C |. E8 91DE0000 CALL   <JMP.&MFC42.#941>
00405C41 |> 8B7424 08  MOV    ESI, DWORD PTR SS:[ESP+8] ==> đưa Input vào ESI
00405C45 |. 8D4424 20  LEA     EAX, DWORD PTR SS:[ESP+20]
00405C49 |. 33D2  XOR    EDX, EDX ;==> bắt đầu tính toán với EDX=0
00405C4B |. 2BF0  SUB    ESI, EAX
=====chương trình sẽ thực hiện vòng lặp này để lấy và check từng char FU mà ta nhập vào=====
00405C4D |> 8D4C14 20  /LEA   ECX, DWORD PTR SS:[ESP+EDX+20]
00405C51 |. 8A040E  |MOV    AL, BYTE PTR DS:[ESI+ECX] ==> đưa S[0] vào AL
00405C54 |. 3C 61  |CMP    AL, 61 ;==> so sánh S[0] với 61
00405C56 |. 8801  |MOV    BYTE PTR DS:[ECX], AL
00405C58 |. 7C 08  |JL    SHORT BatchIma.00405C62 ;==> nhảy nếu nhỏ hơn
00405C5A |. 3C 66  |CMP    AL, 66 ;==> so sánh S[0] với 66
00405C5C |. 7F 04  |JG    SHORT BatchIma.00405C62 ;==> nhảy nếu lớn hơn
00405C5E |. 24 CE  |AND    AL, 0CE ;==> nếu < thì And tiếp AL với 0CE
00405C60 |. 8801  |MOV    BYTE PTR DS:[ECX], AL
00405C62 |> 8A01  |MOV    AL, BYTE PTR DS:[ECX] ;==> đưa S[0] vào AL
00405C64 |. 3C 41  |CMP    AL, 41 ;==> so sánh S[0] với 41
00405C66 |. 7C 08  |JL    SHORT BatchIma.00405C70 ;==> nhảy nếu nhỏ hơn
00405C68 |. 3C 46  |CMP    AL, 46 ;==> so sánh S[0] với 46
00405C6A |. 7F 04  |JG    SHORT BatchIma.00405C70 ;==> nhảy nếu lớn hơn
00405C6C |. 2C 42  |SUB    AL, 42 ;==> nếu < thì tiếp tục trừ AL cho 42
00405C6E |. EB 02  |JMP    SHORT BatchIma.00405C72
00405C70 |> 2C 32  |SUB    AL, 32 ;==> trừ AL cho 32
00405C72 |> 42    |INC    EDX ;==> tăng EDX lên 1
00405C73 |. 8801  |MOV    BYTE PTR DS:[ECX], AL
00405C75 |. 83FA 10 |CMP    EDX, 10 ;==> so sánh EDX với 10
00405C78 |.^ 7C D3  |JL    SHORT BatchIma.00405C4D ;==> tiếp tục vòng lặp nếu nhỏ hơn .
=====lặp 16 lần thì check xong & hết vòng lặp thứ nhất=====
00405C7A |. 55      PUSH    EBP
00405C7B |. 8D6C24 14  LEA    EBP, DWORD PTR SS:[ESP+14]
00405C7F |. 57      PUSH    EDI
00405C80 |. B8 06000000 MOV    EAX, 6
00405C85 |. 83ED 06  SUB    EBP, 6
=====bắt đầu vòng lặp thứ 2 để tạo Rel Ser=====

```

00405C88 |> 8D70 FA /LEA ESI, DWORD PTR DS:[EAX-6]
 00405C8B |. 8BD8 |MOV EBX, EAX
 00405C8D |. 8BCE |MOV ECX, ESI
 00405C8F |. 83E3 0F |AND EBX, 0F
 00405C92 |. 83E1 0F |AND ECX, 0F
 00405C95 |. 8D78 01 |LEA EDI, DWORD PTR DS:[EAX+1]
 00405C98 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00405C9D |. 0FBE540C 28 |MOVSX EDX, BYTE PTR SS:[ESP+ECX+28]
 00405CA2 |. 8D48 FC |LEA ECX, DWORD PTR DS:[EAX-4]
 00405CA5 |. 03D6 |ADD EDX, ESI
 00405CA7 |. 83E1 0F |AND ECX, 0F
 00405CAA |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00405CAF |. 0FAFD1 |IMUL EDX, ECX
 00405CB2 |. 8D48 FD |LEA ECX, DWORD PTR DS:[EAX-3]
 00405CB5 |. 83E1 0F |AND ECX, 0F
 00405CB8 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00405CBD |. 03CE |ADD ECX, ESI
 00405CBF |. 0FAFCB |IMUL ECX, EBX
 00405CC2 |. 03D1 |ADD EDX, ECX
 00405CC4 |. 8D48 FB |LEA ECX, DWORD PTR DS:[EAX-5]
 00405CC7 |. 8D58 FF |LEA EBX, DWORD PTR DS:[EAX-1]
 00405CCA |. 83E1 0F |AND ECX, 0F
 00405CCD |. 83E3 0F |AND EBX, 0F
 00405CD0 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00405CD5 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00405CDA |. 0FAFCB |IMUL ECX, EBX
 00405CDD |. 03D1 |ADD EDX, ECX
 00405CDF |. 8D48 FE |LEA ECX, DWORD PTR DS:[EAX-2]
 00405CE2 |. 8BDF |MOV EBX, EDI
 00405CE4 |. 83E1 0F |AND ECX, 0F
 00405CE7 |. 83E3 0F |AND EBX, 0F
 00405CEA |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00405CEF |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00405CF4 |. 0FAFCB |IMUL ECX, EBX
 00405CF7 |. 03D1 |ADD EDX, ECX
 00405CF9 |. 8D48 03 |LEA ECX, DWORD PTR DS:[EAX+3]
 00405CFC |. 83E1 0F |AND ECX, 0F
 00405CFF |. 8D58 07 |LEA EBX, DWORD PTR DS:[EAX+7]
 00405D02 |. 83E3 0F |AND EBX, 0F
 00405D05 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00405D0A |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00405D0F |. 03CE |ADD ECX, ESI
 00405D11 |. 0FAFCB |IMUL ECX, EBX
 00405D14 |. 8D58 02 |LEA EBX, DWORD PTR DS:[EAX+2]
 00405D17 |. 83E3 0F |AND EBX, 0F
 00405D1A |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00405D1F |. 03DE |ADD EBX, ESI
 00405D21 |. 8D70 04 |LEA ESI, DWORD PTR DS:[EAX+4]
 00405D24 |. 83E6 0F |AND ESI, 0F
 00405D27 |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
 00405D2C |. 0FAFDE |IMUL EBX, ESI
 00405D2F |. 03CB |ADD ECX, EBX

```

00405D31 |. 8D70 F8    |LEA   ESI, DWORD PTR DS:[EAX-8]
00405D34 |. 8D58 06    |LEA   EBX, DWORD PTR DS:[EAX+6]
00405D37 |. 83E6 0F    |AND   ESI, OF
00405D3A |. 83E3 0F    |AND   EBX, OF
00405D3D |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
00405D42 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
00405D47 |. 0FAFF3      |IMUL  ESI, EBX
00405D4A |. 03CE       |ADD   ECX, ESI
00405D4C |. 8D70 F9    |LEA   ESI, DWORD PTR DS:[EAX-7]
00405D4F |. 8D58 05    |LEA   EBX, DWORD PTR DS:[EAX+5]
00405D52 |. 83E6 0F    |AND   ESI, OF
00405D55 |. 83E3 0F    |AND   EBX, OF
00405D58 |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
00405D5D |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
00405D62 |. 0FAFF3      |IMUL  ESI, EBX
00405D65 |. 03CE       |ADD   ECX, ESI
00405D67 |. 0FAFCA      |IMUL  ECX, EDX
00405D6A |. C1F9 03    |SAR   ECX, 3
00405D6D |. 83E1 1F    |AND   ECX, 1F
00405D70 |. 83F9 0A    |CMP   ECX, 0A
00405D73 |. 7D 05      |JGE   SHORT BatchIma.00405D7A
00405D75 |. 83C1 50    |ADD   ECX, 50
00405D78 |. EB 0D      |JMP   SHORT BatchIma.00405D87
00405D7A |> 83F9 12    |CMP   ECX, 12
00405D7D |. 7D 05      |JGE   SHORT BatchIma.00405D84
00405D7F |. 83C1 28    |ADD   ECX, 28
00405D82 |. EB 03      |JMP   SHORT BatchIma.00405D87
00405D84 |> 83C1 2F    |ADD   ECX, 2F
00405D87 |> 880C28      |MOV   BYTE PTR DS:[EAX+EBP], CL
00405D8A |. 8BC7      |MOV   EAX, EDI
00405D8C |. 8D50 FA    |LEA   EDX, DWORD PTR DS:[EAX-6]
00405D8F |. 83FA 10    |CMP   EDX, 10
00405D92 |.^ 0F8C F0FEFFFF \JL   BatchIma.00405C88
=====Loop=16 times , End loop 2=====
00405D98 |. 68 A4D44100 PUSH  BatchIma.0041D4A4      ; ASCII "0000000000000000"
00405D9D |. 8D4C24 14 LEA   ECX, DWORD PTR SS:[ESP+14]
00405DA1 |. E8 8ADC0000 CALL  <JMP.&MFC42.#860>
00405DA6 |. 5F      POP   EDI
00405DA7 |. 33F6      XOR   ESI, ESI ==> bắt đầu với ESI = 0
00405DA9 |. 5D      POP   EBP
=====Start Loop 3 , loop=16=====
00405DAA |> 8A4434 10 /MOV   AL, BYTE PTR SS:[ESP+ESI+10] ==> lần lượt đưa từng char
của Rel Ser vào EAX
00405DAE |. 8D4C24 08 |LEA   ECX, DWORD PTR SS:[ESP+8]
00405DB2 |. 50      |PUSH  EAX
00405DB3 |. 56      |PUSH  ESI
00405DB4 |. E8 6FDE0000 |CALL  <JMP.&MFC42.#5856>
00405DB9 |. 46      |INC   ESI           ;==> ESI ++
00405DBA |. 83FE 10    |CMP   ESI, 10          ;==> so sánh ESI với 10
00405DBD |.^ 7C EB    \JL   SHORT BatchIma.00405DAA ==> nếu nhỏ hơn thì tiếp tục vòng lặp
=====End loop 3=====
=====đến đây nhìn sang cửa sổ bên cạnh , ta sẽ thấy Real Ser của c/t sau khi tính toán được đặt trong

```

thanh ghi EAX=====

```

00405DBF |. BB 01000000 MOV EBX, 1
00405DC4 |. EB 09 JMP SHORT BatchIma.00405DCF
00405DC6 > 8D4C24 08 LEA ECX, DWORD PTR SS:[ESP+8]
00405DCA |. E8 53DE0000 CALL <JMP.&MFC42.#2614>
00405DCF > 8B7424 44 MOV ESI, DWORD PTR SS:[ESP+44]
00405DD3 |. 8D4C24 08 LEA ECX, DWORD PTR SS:[ESP+8]
00405DD7 |. 51 PUSH ECX
00405DD8 |. 8BCE MOV ECX, ESI
00405DDA |. E8 5FDD0000 CALL <JMP.&MFC42.#535>
00405DDF |. 895C24 0C MOV DWORD PTR SS:[ESP+C], EBX
00405DE3 |. 8D4C24 08 LEA ECX, DWORD PTR SS:[ESP+8]
00405DE7 |. 885C24 3C MOV BYTE PTR SS:[ESP+3C], BL
00405DEB |. E8 02DB0000 CALL <JMP.&MFC42.#800>
00405DF0 |. 8D4C24 48 LEA ECX, DWORD PTR SS:[ESP+48]
00405DF4 |. C64424 3C 00 MOV BYTE PTR SS:[ESP+3C], 0
00405DF9 |. E8 F4DA0000 CALL <JMP.&MFC42.#800>
00405DFE |. 8B4C24 34 MOV ECX, DWORD PTR SS:[ESP+34]
00405E02 |. 8BC6 MOV EAX, ESI
00405E04 |. 5E POP ESI
00405E05 |. 5B POP EBX
00405E06 |. 64:890D 00000>MOV DWORD PTR FS:[0], ECX
00405E0D |. 83C4 38 ADD ESP, 38
00405E10 \. C2 0800 RETN 8

```

```

00405B5E |. 8B7424 14 MOV ESI, DWORD PTR SS:[ESP+14]
00405B62 |. 8B4424 18 MOV EAX, DWORD PTR SS:[ESP+18]

```

- Nhìn sang của sổ bên cạnh, chúng ta thấy 1 vàng hào quang sáng chói hiện . Vâng đó chính là RS- cái mà chúng ta cần tìm : ASCII "MWDCTXIYD4HUPHVA"

/*/*/* - SERIAL tương ứng :

Email :hoa_dong_noi_06@yahoo.com Serial : MWDCTXIYD4HUPHVA
Hoặc :

Email :REA-cRaCkErTeAm@yahoo.com Serial : CYSSRW4U4LXJ2FMF

III/ Keygen :

```

char reaEmail[64]={0};
char reaSerial[64]={0};
char reaTempString[17]={0};
int LenEmail=0;
intWhiteSpace=0, Asign=0;
int i=0, j=0, k=0, n=0, t=0;
int Value = 0;
int Value1=0, Temp1=0, Temp2=0, Temp3=0, Temp4=0, Temp5=0, Temp6=0, Temp7=0;
int Temp8=0, Temp9=0, Temp10=0;

```

```

LenEmail = GetDlgItemText(IDC_NAME, reaEmail, 64);
if (LenEmail < 16)
{
    MessageBox("===== Your email atleast 16 charts ===== ", "Hey !! Please

```

```

input your email again !! ");
}
else
{
    i = 0;
    while ( i < LenEmail )
    {
        if ( reaEmail[i] == 0x20 )
        {
           WhiteSpace++;
        }
        if ( reaEmail[i] == 0x40 )
        {
            Asign++;
        }
        i++;
    }
    if (WhiteSpace !=0 )
    {
        MessageBox("----- Your email had WhiteSpace ----- ","Hey
!! Please input your email again !! ");
    }
    else if ( Asign != 1 )
    {
        MessageBox("----- Your email not Valid ----- ","Hey !!
Please input your email again !! ");
    }
    else
    {
        i = 0;
        j = 0;
        while (i < 16)
        {
            if (reaEmail[i] < 0x61 || reaEmail[i] > 0x66)
            {
                Value = reaEmail[i];
            }
            else
            {
                Value = reaEmail[i] & 0x0CE;
            }

            if (Value < 0x41 || Value > 0x46)
            {
                Value = Value - 0x32;

            }
            else
            {
                Value = Value - 0x42;

            }
            reaTempString[j] = Value & 0xFF;
        }
    }
}

```

```

        i++;
        j++;
    }
    i = 0;
    j = 6;
while (i < 16)
{
    Value1 = j-6;
    k = j & 0x0F;
    n = Value1 & 0x0F;
    t = (j + 1);

    Temp1 = reaTempString[k];
    Temp2 = reaTempString[n];
    n = (j - 4) & 0x0F;

    Temp2 = Temp2 + Value1;
    Temp3 = reaTempString[n];
    Temp2 = Temp2 * Temp3;
    n = (j - 3) & 0x0F;

    Temp4 = reaTempString[n];
    Temp4 = Temp4 + Value1;
    Temp4 = Temp4 * Temp1;
    Temp2 = Temp2 + Temp4;

    n = (j - 5) & 0x0F;
    k = (j - 1) & 0x0F;
    Temp5 = reaTempString[n];
    Temp1 = reaTempString[k];
    Temp5 = Temp5 * Temp1;
    Temp2 = Temp2 + Temp5;

    n = (j - 2) & 0x0F;
    k = t & 0x0F;
    Temp6 = reaTempString[n];
    Temp1 = reaTempString[k];
    Temp6 = Temp6 * Temp1;
    Temp2 = Temp2 + Temp6;
    n = (j + 3) & 0x0F;
    k = (j + 7) & 0x0F;

    Temp7 = reaTempString[n];
    Temp1 = reaTempString[k];
    Temp7 = Temp7 + Value1;
    Temp7 = Temp7 * Temp1;
    k = (j + 2) & 0x0F;

    Temp1 = reaTempString[k];
    Temp1 = Temp1 + Value1;
    Value1 = (j + 4) & 0x0F;

    Temp8 = reaTempString[Value1];

```

```

Temp1 = Temp1 * Temp8;
Temp7 = Temp7 + Temp1;
Value1 = (j - 8) & 0x0F;
k = (j + 6) & 0x0F;

Temp9 = reaTempString[Value1];
Temp1 = reaTempString[k];
Temp9 = Temp9 * Temp1;
Temp7 = Temp7 + Temp9;
Value1 = (j - 7) & 0x0F;
k = (j + 5) & 0x0F;

Temp10 = reaTempString[Value1];
Temp1 = reaTempString[k];
Temp10 = Temp10 * Temp1;
Temp7 = Temp7 + Temp10;
Temp7 = Temp7 * Temp2;
_asm
{
    mov ECX,Temp7;
    sar ECX, 0x3;
    and ECX, 0x1F;
    mov Temp7,ECX;
}
if (Temp7 >= 0x0A && Temp7 >=0x12)
{
    Temp7 = Temp7 + 0x2F;
    reaSerial[i] = Temp7;
}
else if (Temp7 < 0x0A)
{
    Temp7 = Temp7 + 0x50;
    reaSerial[i] = Temp7;
}
else if (Temp7 < 0x12)
{
    Temp7 = Temp7 + 0x28;
    reaSerial[i] = Temp7;
}
j = t;
i++;
}
}
}

SetDlgItemText(IDC_SERIAL,reaSerial);

```

IV/ End Tut :

- Finished : 01/12/2004

Reverse Engineering Association

SoftWare

Homepage	:	http://www.jklsoft.com
Production	:	JKLNSoft , Inc
SoftWare	:	Easy Photo Editor v1.6
Copyright by	:	Copyright (C) 2004 JKLNSoft.com. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N/A
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack	:	N/A
Request	:	Correct Serial / KeyGen

Easy Photo Editor v1.6

I – Information :

- Dùng PEid v0.92 để Detect , chúng ta biết chương trình ko bị pack & được viết bằng Microsoft Visual C++ 6.0. Phù thê là ko phải unpack --> sướng . hihiiii..... :)

- Chạy thử chương trình , chúng ta thấy hiện lên một 1 Dialog Box : “Wellcome to Easy Photo Editor ”. Nhìn xuống dưới 1 chút ta thấy dòng : "Try to register...." , nhìn thấy mà tức cả mắt :pinch: . Nó đồng nghĩa với việc họ chỉ cho chúng ta dùng thử nếu chúng ta ko Register (tức là ta phải đưa 19,95\$ cho họ đó bạn) . Đưa money để buy soft , chúng ta đâu có thói quen này . "no Money, no Register , and Free all ==> đó là mục tiêu của chúng ta mà . Hihiiiii.....! :)

- OK ! típ nhẹ . Ở đây ta còn thấy 1 button : Continue , click thử vào xem sao . Oh ! Cái đầu tiên chúng ta cần tìm là chỗ họ bắt ta Register , vào menu Help / Register . Sau đó ta nhập các thông tin cần thiết vào xem nó bảo sao . Ở đây mình nhập là : Email :hoa_dong_noi_06@yahoo.com & Code : 0361985. Nhấn OK , một cái Nag văng ra , nó làm ta mừng hụt :"Thanks for Registratinon. Please restart Easy Photo Editor ".Nhấn tiếp OK , mở lại chương trình ta vẫn thấy hiện Dialog box bắt đăng ký. Như vậy ta suy đoán chương trình sẽ lưu thông tin ta Registered vào Reg rồi check nó mỗi khi ta Run chương trình .

- Ok ! nghĩ sao làm vậy .Bây giờ chúng ta hãy load chương trình vào trong OllyDbg . Nếu nó hiện ra 1 bảng thông báo thì cứ nhấn OK . Sau đó click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . ah ! nó đây rùi , double click vào String đó ta được đưa trở lại Olly . Ấc ặc..... nhìn lướt qua mà thấy toàn là hàm Call MFC , ít mạnh mẽ quá . Để coi trong mớ bòng bong String có gì đáng giá ko ? ở đây chúng ta tìm thấy 1 loạt String như sau :

Text strings referenced in ppp:.text

Address	Disassembly	Text string
004023A7	PUSH ppp.004181B4	ASCII "aaaaaaaaaaaaaaaaaaaaaaaa"
0040250E	PUSH ppp.004181A0	ASCII "0000000000000000"
0040259F	PUSH ppp.0041817C	ASCII "UserName"
004025A8	PUSH ppp.00418168	ASCII "Registration Info"
004025BA	PUSH ppp.004181D0	ASCII "Registration Code"
004025C3	PUSH ppp.00418168	ASCII "Registration Info"

Theo như kinh nghiệm cracked soft trên thì ta thấy chuỗi toàn char 0 này : “0000000000000000” dường như là 1 Template String để chương trình đặt lần lượt từng char RS sau khi đã mã hoá vào đó .

OK ! bây giờ chúng ta Double click vào String đó xem sao , chúng ta sẽ đến 1 Procedure Code . Đặt vết sáng vào đầu Procedure này , nhấn Ctrl + R để mở cửa sổ “References in pp ”, ta sẽ thấy Procedure này được gọi bởi 1 hàm Call :

```
004025F5 CALL ppp.00402360
```

- Double click vào hàm Call này chúng ta sẽ đến được đoạn Code sau :

```
004025F4 |. 50 PUSH EAX ; |Arg1
004025F5 |. E8 66FDFFFF CALL ppp.00402360 ; \ppp.00402360 ==> Set BreakPoint here.
004025FA |. 8B7424 0C MOV ESI, DWORD PTR SS:[ESP+C]
004025FE |. 8B4424 08 MOV EAX, DWORD PTR SS:[ESP+8]
```

II – Cracking :

OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình , Olly sẽ Ice tiến trình tại điểm mà ta vừa đặt BP . Chúng ta sẽ Trace Into vào trong hàm Call này nếu bạn muốn xem cơ chế tạo Ser của nó . có thể trace over cũng lấy dc RS :

```
004025F5 |. E8 66FDFFFF CALL ppp.00402360 ; \ppp.00402360 ==> We're here
```

-----Trace into-----

```
00402360 /$ 6A FF PUSH -1
00402362 |. 68 87F34000 PUSH ppp.0040F387 ; SE handler installation
00402367 |. 64:A1 00000000>MOV EAX, DWORD PTR FS:[0]
0040236D |. 50 PUSH EAX
0040236E |. 64:8925 00000>MOV DWORD PTR FS:[0], ESP
00402375 |. 83EC 2C SUB ESP, 2C
00402378 |. C74424 04 000>MOV DWORD PTR SS:[ESP+4], 0
00402380 |. 8D4424 40 LEA EAX, DWORD PTR SS:[ESP+40]
00402384 |. 8D4C24 00 LEA ECX, DWORD PTR SS:[ESP]
00402388 |. 50 PUSH EAX
00402389 |. C74424 38 010>MOV DWORD PTR SS:[ESP+38], 1
00402391 |. E8 F8C20000 CALL <JMP.&MFC42.#535>
00402396 |. 8B4C24 00 MOV ECX, DWORD PTR SS:[ESP] ==> đưa Input vào ECX .
0040239A |. C64424 34 02 MOV BYTE PTR SS:[ESP+34], 2
0040239F |. 8B41 F8 MOV EAX, DWORD PTR DS:[ECX-8]
004023A2 |. 83F8 10 CMP EAX, 10
004023A5 |. 7D 0E JGE SHORT ppp.004023B5
004023A7 |. 68 B4814100 PUSH ppp.004181B4 ; ASCII "aaaaaaaaaaaaaaaaaaaaaaaaaaa"
004023AC |. 8D4C24 04 LEA ECX, DWORD PTR SS:[ESP+4]
004023B0 |. E8 35C10000 CALL <JMP.&MFC42.#941>
004023B5 |> 56 PUSH ESI
004023B6 |. 8B7424 04 MOV ESI, DWORD PTR SS:[ESP+4] ==> đưa Input vào ESI .
004023BA |. 8D4424 1C LEA EAX, DWORD PTR SS:[ESP+1C]
004023BE |. 33D2 XOR EDX, EDX ==> bắt đầu tính toán với EDX=0
004023C0 |. 2BF0 SUB ESI, EAX
-----chương trình sẽ thực hiện vòng lặp này để lấy và check từng char FU mà ta nhập vào-----
```

```
004023C2 |> 8D4C14 1C /LEA ECX, DWORD PTR SS:[ESP+EDX+1C]
004023C6 |. 8A040E |MOV AL, BYTE PTR DS:[ESI+ECX] ==> đưa S[0] vào AL
004023C9 |. 3C 61 |CMP AL, 61 ==> so sánh S[0] với 61
004023CB |. 8801 |MOV BYTE PTR DS:[ECX], AL
004023CD |. 7C 08 |JL SHORT ppp.004023D7 ==> nhảy nếu nhỏ hơn
004023CF |. 3C 66 |CMP AL, 66 ==> so sánh S[0] với 66
```

004023D1 |. 7F 04 |JG SHORT ppp.004023D7 ==> nhảy nếu lớn hơn
 004023D3 |. 24 CE |AND AL, 0CE ==> nếu nhỏ hơn thì And tiếp AL với 0CE
 004023D5 |. 8801 |MOV BYTE PTR DS:[ECX], AL
 004023D7 > 8A01 |MOV AL, BYTE PTR DS:[ECX] ==> đưa S[0] vào AL
 004023D9 |. 3C 41 |CMP AL, 41 ==> so sánh S[0] với 41
 004023DB |. 7C 08 |JL SHORT ppp.004023E5 ==> nhảy nếu nhỏ hơn
 004023DD |. 3C 46 |CMP AL, 46 ==> so sánh S[0] với 46
 004023DF |. 7F 04 |JG SHORT ppp.004023E5 ==> nhảy nếu lớn hơn
 004023E1 |. 2C 42 |SUB AL, 42 ==> nếu nhỏ hơn thì tiếp tục trừ AL cho 42
 004023E3 |. EB 02 |JMP SHORT ppp.004023E7
 004023E5 > 2C 32 |SUB AL, 32 ==> trừ tiếp AL cho 32
 004023E7 > 42 |INC EDX ==> tăng EDX lên 1
 004023E8 |. 8801 |MOV BYTE PTR DS:[ECX], AL
 004023EA |. 83FA 10 |CMP EDX, 10 ==> so sánh EDX với 10
 004023ED |.^ 7C D3 |JL SHORT ppp.004023C2 ==> tiếp tục vòng lặp nếu nhỏ hơn .
 -----lặp 16 lần thì check xong & hết vòng lặp thứ nhất-----

004023EF |. 53 PUSH EBX
 004023F0 |. 55 PUSH EBP
 004023F1 |. 8D6C24 14 LEA EBP, DWORD PTR SS:[ESP+14]
 004023F5 |. 57 PUSH EDI
 004023F6 |. B8 06000000 MOV EAX, 6
 004023FB |. 83ED 06 SUB EBP, 6

-----bắt đầu vòng lặp thứ 2 để tạo Rel Ser-----

004023FE > 8D70 FA /LEA ESI, DWORD PTR DS:[EAX-6]
 00402401 |. 8BD8 |MOV EBX, EAX
 00402403 |. 8BCE |MOV ECX, ESI
 00402405 |. 83E3 0F |AND EBX, OF
 00402408 |. 83E1 0F |AND ECX, OF
 0040240B |. 8D78 01 |LEA EDI, DWORD PTR DS:[EAX+1]
 0040240E |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00402413 |. 0FBE540C 28 |MOVSX EDX, BYTE PTR SS:[ESP+ECX+28]
 00402418 |. 8D48 FC |LEA ECX, DWORD PTR DS:[EAX-4]
 0040241B |. 03D6 |ADD EDX, ESI
 0040241D |. 83E1 0F |AND ECX, OF
 00402420 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00402425 |. 0FAFD1 |IMUL EDX, ECX
 00402428 |. 8D48 FD |LEA ECX, DWORD PTR DS:[EAX-3]
 0040242B |. 83E1 0F |AND ECX, OF
 0040242E |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00402433 |. 03CE |ADD ECX, ESI
 00402435 |. 0FAFCB |IMUL ECX, EBX
 00402438 |. 03D1 |ADD EDX, ECX
 0040243A |. 8D48 FB |LEA ECX, DWORD PTR DS:[EAX-5]
 0040243D |. 8D58 FF |LEA EBX, DWORD PTR DS:[EAX-1]
 00402440 |. 83E1 0F |AND ECX, OF
 00402443 |. 83E3 0F |AND EBX, OF
 00402446 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 0040244B |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00402450 |. 0FAFCB |IMUL ECX, EBX
 00402453 |. 03D1 |ADD EDX, ECX
 00402455 |. 8D48 FE |LEA ECX, DWORD PTR DS:[EAX-2]
 00402458 |. 8BDF |MOV EBX, EDI

0040245A |. 83E1 0F |AND ECX, 0F
 0040245D |. 83E3 0F |AND EBX, 0F
 00402460 |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00402465 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 0040246A |. 0FAFCB |IMUL ECX, EBX
 0040246D |. 03D1 |ADD EDX, ECX
 0040246F |. 8D48 03 |LEA ECX, DWORD PTR DS:[EAX+3]
 00402472 |. 83E1 0F |AND ECX, 0F
 00402475 |. 8D58 07 |LEA EBX, DWORD PTR DS:[EAX+7]
 00402478 |. 83E3 0F |AND EBX, 0F
 0040247B |. 0FBE4C0C 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00402480 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00402485 |. 03CE |ADD ECX, ESI
 00402487 |. 0FAFCB |IMUL ECX, EBX
 0040248A |. 8D58 02 |LEA EBX, DWORD PTR DS:[EAX+2]
 0040248D |. 83E3 0F |AND EBX, 0F
 00402490 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00402495 |. 03DE |ADD EBX, ESI
 00402497 |. 8D70 04 |LEA ESI, DWORD PTR DS:[EAX+4]
 0040249A |. 83E6 0F |AND ESI, 0F
 0040249D |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
 004024A2 |. 0FAFDE |IMUL EBX, ESI
 004024A5 |. 03CB |ADD ECX, EBX
 004024A7 |. 8D70 F8 |LEA ESI, DWORD PTR DS:[EAX-8]
 004024AA |. 8D58 06 |LEA EBX, DWORD PTR DS:[EAX+6]
 004024AD |. 83E6 0F |AND ESI, 0F
 004024B0 |. 83E3 0F |AND EBX, 0F
 004024B3 |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
 004024B8 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 004024BD |. 0FAFF3 |IMUL ESI, EBX
 004024C0 |. 03CE |ADD ECX, ESI
 004024C2 |. 8D70 F9 |LEA ESI, DWORD PTR DS:[EAX-7]
 004024C5 |. 8D58 05 |LEA EBX, DWORD PTR DS:[EAX+5]
 004024C8 |. 83E6 0F |AND ESI, 0F
 004024CB |. 83E3 0F |AND EBX, 0F
 004024CE |. 0FBE7434 28 |MOVSX ESI, BYTE PTR SS:[ESP+ESI+28]
 004024D3 |. 0FBE5C1C 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 004024D8 |. 0FAFF3 |IMUL ESI, EBX
 004024DB |. 03CE |ADD ECX, ESI
 004024DD |. 0FAFCA |IMUL ECX, EDX
 004024E0 |. C1F9 0A |SAR ECX, 0A
 004024E3 |. 83E1 1F |AND ECX, 1F
 004024E6 |. 83F9 0A |CMP ECX, 0A
 004024E9 |. 7D 05 |JGE SHORT ppp.004024F0
 004024EB |. 83C1 50 |ADD ECX, 50
 004024EE |. EB 0D |JMP SHORT ppp.004024FD
 004024F0 |> 83F9 12 |CMP ECX, 12
 004024F3 |. 7D 05 |JGE SHORT ppp.004024FA
 004024F5 |. 83C1 28 |ADD ECX, 28
 004024F8 |. EB 03 |JMP SHORT ppp.004024FD
 004024FA |> 83C1 2F |ADD ECX, 2F
 004024FD |> 880C28 |MOV BYTE PTR DS:[EAX+EBP], CL

```

00402500 |. 8BC7      MOV    EAX, EDI
00402502 |. 8D50 FA    LEA    DWORD PTR DS:[EAX-6]
00402505 |. 83FA 10    CMP    EDX, 10
00402508 |.^ 0F8C F0FEFFFF JL    ppp.004023FE
-----Loop=16 times , End loop 2-----
0040250E |. 68 A0814100 PUSH   ppp.004181A0          ; ASCII "0000000000000000"
00402513 |. 8D4C24 14  LEA    ECX, DWORD PTR SS:[ESP+14]
00402517 |. E8 7EBE0000 CALL   <JMP.&MFC42.#860>
0040251C |. 5F      POP    EDI
0040251D |. 5D      POP    EBP
0040251E |. 33F6      XOR    ESI, ESI
00402520 |. 5B      POP    EBX
-----Start Loop 3 , loop=16-----
00402521 |> 8A4434 0C  MOV    AL, BYTE PTR SS:[ESP+ESI+C] ==> lần lượt đưa từng char của
Rel Ser vào EAX
00402525 |. 8D4C24 04  LEA    ECX, DWORD PTR SS:[ESP+4]
00402529 |. 50      PUSH   EAX
0040252A |. 56      PUSH   ESI
0040252B |. E8 58C10000 CALL   <JMP.&MFC42.#5856>
00402530 |. 46      INC    ESI ==> ESI ++
00402531 |. 83FE 10    CMP    ESI, 10 ==> so sánh ESI với 10
00402534 |.^ 7C EB    JL    SHORT ppp.00402521 ==> nếu nhỏ hơn thì tiếp tục vòng lặp
-----End loop 3-----
-----đến đây nhìn sang cửa sổ bên cạnh , ta sẽ thấy Real Ser của c/t sau khi tính toán được đặt
trong thanh ghi EAX-----
00402536 |. 8B7424 40  MOV    ESI, DWORD PTR SS:[ESP+40]
0040253A |. 8D4C24 04  LEA    ECX, DWORD PTR SS:[ESP+4]
0040253E |. 51      PUSH   ECX
0040253F |. 8BCE      MOV    ECX, ESI
00402541 |. E8 48C10000 CALL   <JMP.&MFC42.#535>
00402546 |. B8 01000000 MOV    EAX, 1
0040254B |. 894424 08  MOV    DWORD PTR SS:[ESP+8], EAX
0040254F |. 8D4C24 04  LEA    ECX, DWORD PTR SS:[ESP+4]
00402553 |. 884424 38  MOV    BYTE PTR SS:[ESP+38], AL
00402557 |. E8 32BE0000 CALL   <JMP.&MFC42.#800>
0040255C |. 8D4C24 44  LEA    ECX, DWORD PTR SS:[ESP+44]
00402560 |. C64424 38 00 MOV    BYTE PTR SS:[ESP+38], 0
00402565 |. E8 24BE0000 CALL   <JMP.&MFC42.#800>
0040256A |. 8B4C24 30  MOV    ECX, DWORD PTR SS:[ESP+30]
0040256E |. 8BC6      MOV    EAX, ESI
00402570 |. 5E      POP    ESI
00402571 |. 64:890D 00000>MOV    DWORD PTR FS:[0], ECX
00402578 |. 83C4 38    ADD    ESP, 38
0040257B \. C2 0800    RETN   8
-----
004025FA |. 8B7424 0C  MOV    ESI, DWORD PTR SS:[ESP+C]
004025FE |. 8B4424 08  MOV    EAX, DWORD PTR SS:[ESP+8]

```

- Nhìn sang cửa sổ bên cạnh, chúng ta thấy 1 vầng hào quang sáng chói hiện ra . Vâng đó chính là RS- cái mà chúng ta cần tìm : ASCII "A5I8A4LQ4X5IK3DN"

/*/*/* - SERIAL tương ứng :

Email :hoa_dong_noi_06@yahoo.com Serial : A5I8A4LQ4X5IK3DN

Hoặc :

Email :REA-cRaCkErTeAm@yahoo.com Serial : WGMUVJ6N5QKX5MSI

III/ Keygen :

```

char reaEmail[64]={0};
char reaSerial[64]={0};
char reaTempString[17]={0};
int LenEmail=0;
intWhiteSpace=0, Asign=0;
int i=0, j=0, k=0, n=0, t=0;
int Value = 0;
int Value1=0, Temp1=0, Temp2=0, Temp3=0, Temp4=0, Temp5=0, Temp6=0, Temp7=0;
int Temp8=0, Temp9=0, Temp10=0;

LenEmail = GetDlgItemText(IDC_NAME, reaEmail,64);
if (LenEmail < 16)
{
    MessageBox("----- Your email atleast 16 charts ----- ","Hey !! Please
input your email again !! ");
}
else
{
    i = 0;
    while ( i < LenEmail )
    {
        if ( reaEmail[i] == 0x20 )
        {
           WhiteSpace++;
        }
        if ( reaEmail[i] == 0x40 )
        {
            Asign++;
        }
        i++;
    }
    if (WhiteSpace !=0 )
    {
        MessageBox("----- Your email hadWhiteSpace ----- ","Hey
!! Please input your email again !! ");
    }
    else if ( Asign != 1 )
    {
        MessageBox("----- Your email not Valid ----- ","Hey !!
Please input your email again !! ");
    }
}
else
{
    i = 0;
    j = 0;
    while (i < 16)

```

```

{
    if (reaEmail[i] < 0x61 || reaEmail[i] > 0x66)
    {
        Value = reaEmail[i];
    }
    else
    {
        Value = reaEmail[i] & 0x0CE;
    }

    if (Value < 0x41 || Value > 0x46)
    {
        Value = Value - 0x32;

    }
    else
    {
        Value = Value - 0x42;

    }
    reaTempString[j] = Value & 0xFF;
    i++;
    j++;
}
i = 0;
j = 6;
while (i < 16)
{
    Value1 = j-6;
    k = j & 0x0F;
    n = Value1 & 0x0F;
    t = (j + 1);

    Temp1 = reaTempString[k];
    Temp2 = reaTempString[n];
    n = (j - 4) & 0x0F;

    Temp2 = Temp2 + Value1;
    Temp3 = reaTempString[n];
    Temp2 = Temp2 * Temp3;
    n = (j - 3) & 0x0F;

    Temp4 = reaTempString[n];
    Temp4 = Temp4 + Value1;
    Temp4 = Temp4 * Temp1;
    Temp2 = Temp2 + Temp4;

    n = (j - 5) & 0x0F;
    k = (j - 1) & 0x0F;
    Temp5 = reaTempString[n];
    Temp1 = reaTempString[k];
    Temp5 = Temp5 * Temp1;
    Temp2 = Temp2 + Temp5;
}

```

```

n = (j - 2) & 0x0F;
k = t & 0x0F;
Temp6 = reaTempString[n];
Temp1 = reaTempString[k];
Temp6 = Temp6 * Temp1;
Temp2 = Temp2 + Temp6;
n = (j + 3) & 0x0F;
k = (j + 7) & 0x0F;

Temp7 = reaTempString[n];
Temp1 = reaTempString[k];
Temp7 = Temp7 + Value1;
Temp7 = Temp7 * Temp1;
k = (j + 2) & 0x0F;

Temp1 = reaTempString[k];
Temp1 = Temp1 + Value1;
Value1 = (j + 4) & 0x0F;

Temp8 = reaTempString[Value1];
Temp1 = Temp1 * Temp8;
Temp7 = Temp7 + Temp1;
Value1 = (j - 8) & 0x0F;
k = (j + 6) & 0x0F;

Temp9 = reaTempString[Value1];
Temp1 = reaTempString[k];
Temp9 = Temp9 * Temp1;
Temp7 = Temp7 + Temp9;
Value1 = (j - 7) & 0x0F;
k = (j + 5) & 0x0F;

Temp10 = reaTempString[Value1];
Temp1 = reaTempString[k];
Temp10 = Temp10 * Temp1;
Temp7 = Temp7 + Temp10;
Temp7 = Temp7 * Temp2;
_asm
{
    mov ECX,Temp7;
    sar ECX, 0x0A;
    and ECX, 0x1F;
    mov Temp7,ECX;
}
if (Temp7 >= 0x0A && Temp7 >=0x12)
{
    Temp7 = Temp7 + 0x2F;
    reaSerial[i] = Temp7;
}
else if (Temp7 < 0x0A)
{
    Temp7 = Temp7 + 0x50;
}

```

```

        reaSerial[i] = Temp7;
    }
    else if (Temp7 < 0x12)
    {
        Temp7 = Temp7 + 0x28;
        reaSerial[i] = Temp7;
    }
    j = t;
    i++;
}
}

SetDlgItemText(IDC_SERIAL, reaSerial);

```

III/ End Tut :

- Finished : 01/12/2004

Reverse Engineering Association SoftWare

Homepage	:	http://www.jklsoft.com
Production	:	JKLNSoft , Inc
SoftWare	:	Picture Finder Pro v2.6
Copyright by	:	Copyright (C) 2004 JKLNSoft.com. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N/A
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsm 10
Unpack	:	N/A
Request	:	Correct Serial / KeyGen

Picture Finder Pro v2.6

I – Information :

- Dùng PEid v0.92 để Detect , chúng ta biết chương trình ko bị pack & được viết bằng Microsoft Visual C++ 6.0. Phù thê là ko phải unpack --> sướng . hihiiii..... :)

- Chạy thử chương trình , chúng ta thấy hiện lên một 1 Dialog Box : "Wellcome to Picture Finder Pro". Nhìn xuống dưới 1 chút ta thấy dòng : "Try to register...." , nhìn thấy mà tức cả mắt. Nó đồng nghĩa với việc họ chỉ cho chúng ta dùng thử 14 days & limit 50 files nếu chúng ta ko Register (tức là ta phải đưa money cho họ đó bạn). Đưa money để buy soft , chúng ta đâu có thói quen này."no Money, no Register , and Free all "=> đó là mục tiêu của chúng ta . Hihiiii.....! :P

- OK ! típ nhé. Ở đây ta còn thấy 1 button : Enter Code , click vào đó , chúng ta sẽ điền các thông tin cần thiết xem nó bảo sao. Ở đây mình nhập là : Email :hoa_dong_noi_06@yahoo.com & Code : 0361985 . Nhấn OK , một cái Nag văng ra , nó làm ta mừng hụt : "Thanks for Registratinon. Please restart Picture Finder Pro ". Nhấn tiếp OK , mở lại chương trình ta vẫn thấy hiện Dialog box bắt đăng ký. Như vậy ta

suy đoán chương trình sẽ lưu thông tin ta Registed vào Reg rồi check nó mỗi khi ta Run chương trình .

- Ok ! nghĩ sao làm vậy. Bây giờ chúng ta hãy load chương trình vào trong OllyDbg. Nếu nó hiện ra 1 bảng thông báo thì cứ nhấn OK. Sau đó click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . ah ! nó đây rùi , double click vào String đó ta được đưa trở lại Olly . Ác ặc..... nhìn lướt qua mà thấy toàn là hàm Call MFC , ít manh mối quá. Để coi trong mớ bòng bong String có gì đáng giá ko ? Ở đây chúng ta tìm thấy 1 loạt String như sau :

Text strings referenced in PictureF:.text

Address	Disassembly	Text string
00408ACA	PUSH PictureF.0047E9B4	ASCII "Email"
00408AD3	PUSH PictureF.0047E9A0	ASCII "Registration Info"
00408AE5	PUSH PictureF.0047E98C	ASCII "Registration Code"
00408AEE	PUSH PictureF.0047E9A0	ASCII "Registration Info"
00408B9B	PUSH PictureF.0047E398	ASCII "Analyse.dll"
00408D47	PUSH PictureF.0047E9E0	ASCII "aaaaaaaa"
00408EAE	PUSH PictureF.0047E9CC	ASCII "0000000000000000"

- Theo kinh nghiệm cracked soft trên thì ta thấy chuỗi toàn char 0 này : “0000000000000000” dường như là 1 Template String để chương trình đặt lần lượt từng char RS sau khi đã mã hoá vào đó .

OK ! bây giờ chúng ta Double click vào String đó xem sao , chúng ta sẽ đến 1 Procedure Code. Đặt vệt sáng vào đầu Procedure này , nhấn Ctrl + R để mở cửa sổ “References ” , ta sẽ thấy Procedure này được gọi bởi 2 hàm Call :

```
00408B35 CALL PictureF.00408D00
00408C51 CALL PictureF.00408D00
```

Click chuột phải vào cửa sổ này chọn : Set breakpoint on very command . Quay trở lại Olly

II – Cracking :

- OK, sau khi đặt BP tại 2 hàm Call đó , chúng ta nhấn F9 để Run chương trình , Olly sẽ Ice tiến trình tại điểm mà ta vừa đặt BP . Chúng ta sẽ Trace Into vào trong hàm Call này nếu bạn muốn xem cơ chế tạo Ser của nó . có thể trace over cũng lấy dc RS :

```
00408B35 |.E8 C6010000 CALL PictureF.00408D00 ;\PictureF.00408D00 ==> Set BP. We're here
-----Trace Into-----
```

```
00408D00 /$ 6A FF      PUSH -1
00408D02 |. 68 071E4600 PUSH PictureF.00461E07      ; SE handler installation
00408D07 |. 64:A1 0000000>MOV EAX, DWORD PTR FS:[0]
00408D0D |. 50      PUSH EAX
00408D0E |. 64:8925 00000>MOV DWORD PTR FS:[0], ESP
00408D15 |. 83EC 2C      SUB ESP, 2C
00408D18 |. C74424 04 000>MOV DWORD PTR SS:[ESP+4], 0
00408D20 |. 8D4424 40      LEA EAX, DWORD PTR SS:[ESP+40]
00408D24 |. 8D4C24 00      LEA ECX, DWORD PTR SS:[ESP]
00408D28 |. 50      PUSH EAX
00408D29 |. C74424 38 010>MOV DWORD PTR SS:[ESP+38], 1
00408D31 |. E8 D1720300 CALL PictureF.00440007
00408D36 |. 8B4C24 00      MOV ECX, DWORD PTR SS:[ESP] ==> đưa Input vào ECX .
00408D3A |. C64424 34 02      MOV BYTE PTR SS:[ESP+34], 2
00408D3F |. 8B41 F8      MOV EAX, DWORD PTR DS:[ECX-8]
00408D42 |. 83F8 10      CMP EAX, 10
00408D45 |. 7D 0E      JGE SHORT PictureF.00408D55
```

00408D47 |. 68 E0E94700 PUSH PictureF.0047E9E0 ; ASCII "aaaaaaaa"
 00408D4C |. 8D4C24 04 LEA ECX, DWORD PTR SS:[ESP+4]
 00408D50 |. E8 19790300 CALL PictureF.0044066E
 00408D55 > 56 PUSH ESI
 00408D56 |. 8B7424 04 MOV ESI, DWORD PTR SS:[ESP+4] ==> đưa Input vào ESI
 00408D5A |. 8D4424 1C LEA EAX, DWORD PTR SS:[ESP+1C]
 00408D5E |. 33D2 XOR EDX, EDX ==> bắt đầu tính toán với EDX=0
 00408D60 |. 2BF0 SUB ESI, EAX
 =====chương trình sẽ thực hiện vòng lặp này để lấy và check từng char FU mà ta nhập vào=====
 00408D62 > 8D4C14 1C /LEA ECX, DWORD PTR SS:[ESP+EDX+1C]
 00408D66 |. 8A040E |MOV AL, BYTE PTR DS:[ESI+ECX] ==> đưa S[0] vào AL
 00408D69 |. 3C 61 |CMP AL, 61 ==> so sánh S[0] với 61
 00408D6B |. 8801 |MOV BYTE PTR DS:[ECX], AL
 00408D6D |. 7C 08 |JL SHORT PictureF.00408D77 ==> nhảy nếu nhỏ hơn
 00408D6F |. 3C 66 |CMP AL, 66 ==> so sánh S[0] với 66
 00408D71 |. 7F 04 |JG SHORT PictureF.00408D77 ==> nhảy nếu lớn hơn
 00408D73 |. 24 CE |AND AL, 0CE ==> nếu nhỏ hơn thì And tiếp AL với 0CE
 00408D75 |. 8801 |MOV BYTE PTR DS:[ECX], AL
 00408D77 > 8A01 |MOV AL, BYTE PTR DS:[ECX] ==> đưa S[0] vào AL
 00408D79 |. 3C 41 |CMP AL, 41 ==> so sánh S[0] với 41
 00408D7B |. 7C 08 |JL SHORT PictureF.00408D85 ==> nhảy nếu nhỏ hơn
 00408D7D |. 3C 46 |CMP AL, 46 ==> so sánh S[0] với 46
 00408D7F |. 7F 04 |JG SHORT PictureF.00408D85 ==> nhảy nếu lớn hơn
 00408D81 |. 2C 42 |SUB AL, 42 ==> nếu nhỏ hơn thì tiếp tục trừ AL cho 42
 00408D83 |. EB 02 |JMP SHORT PictureF.00408D87
 00408D85 > 2C 32 |SUB AL, 32 ==> trừ tiếp AL cho 32
 00408D87 > 42 |INC EDX ==> tăng EDX lên 1
 00408D88 |. 8801 |MOV BYTE PTR DS:[ECX], AL
 00408D8A |. 83FA 10 |CMP EDX, 10 ==> so sánh EDX với 10
 00408D8D |.^ 7C D3 |JL SHORT PictureF.00408D62 ==> tiếp tục vòng lặp nếu nhỏ hơn .
 =====lặp 16 lần thì check xong & End Loop1=====
 00408D8F |. 53 PUSH EBX
 00408D90 |. 55 PUSH EBP
 00408D91 |. 8D6C24 14 LEA EBP, DWORD PTR SS:[ESP+14]
 00408D95 |. 57 PUSH EDI
 00408D96 |. B8 06000000 MOV EAX, 6
 00408D9B |. 83ED 06 SUB EBP, 6
 =====bắt đầu vòng lặp thứ 2 để tạo Rel Ser=====
 00408D9E > 8D70 FA /LEA ESI, DWORD PTR DS:[EAX-6]
 00408DA1 |. 8BD8 |MOV EBX, EAX
 00408DA3 |. 8BCE |MOV ECX, ESI
 00408DA5 |. 83E3 0F |AND EBX, OF
 00408DA8 |. 83E1 0F |AND ECX, OF
 00408DAB |. 8D78 01 |LEA EDI, DWORD PTR DS:[EAX+1]
 00408DAE |. 0fbe5c1c 28 |MOVSX EBX, BYTE PTR SS:[ESP+EBX+28]
 00408DB3 |. 0fbe540c 28 |MOVSX EDX, BYTE PTR SS:[ESP+ECX+28]
 00408DB8 |. 8D48 FC |LEA ECX, DWORD PTR DS:[EAX-4]
 00408DBB |. 03D6 |ADD EDX, ESI
 00408DBD |. 83E1 0F |AND ECX, OF
 00408DC0 |. 0fbe4c0c 28 |MOVSX ECX, BYTE PTR SS:[ESP+ECX+28]
 00408DC5 |. 0fafd1 |IMUL EDX, ECX
 00408DC8 |. 8D48 FD |LEA ECX, DWORD PTR DS:[EAX-3]

```

00408DCB |. 83E1 0F    |AND   ECX, 0F
00408DCE |. 0FBE4C0C 28 |MOVSX  ECX, BYTE PTR SS:[ESP+ECX+28]
00408DD3 |. 03CE      |ADD    ECX, ESI
00408DD5 |. 0FAFCB    |IMUL   ECX, EBX
00408DD8 |. 03D1      |ADD    EDX, ECX
00408DDA |. 8D48 FB    |LEA    ECX, DWORD PTR DS:[EAX-5]
00408DDD |. 8D58 FF    |LEA    EBX, DWORD PTR DS:[EAX-1]
00408DE0 |. 83E1 0F    |AND   ECX, 0F
00408DE3 |. 83E3 0F    |AND   EBX, 0F
00408DE6 |. 0FBE4C0C 28 |MOVSX  ECX, BYTE PTR SS:[ESP+ECX+28]
00408DEB |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]
00408DF0 |. 0FAFCB    |IMUL   ECX, EBX
00408DF3 |. 03D1      |ADD    EDX, ECX
00408DF5 |. 8D48 FE    |LEA    ECX, DWORD PTR DS:[EAX-2]
00408DF8 |. 8BDF      |MOV    EBX, EDI
00408DFA |. 83E1 0F    |AND   ECX, 0F
00408DFD |. 83E3 0F    |AND   EBX, 0F
00408E00 |. 0FBE4C0C 28 |MOVSX  ECX, BYTE PTR SS:[ESP+ECX+28]
00408E05 |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]
00408E0A |. 0FAFCB    |IMUL   ECX, EBX
00408E0D |. 03D1      |ADD    EDX, ECX
00408E0F |. 8D48 03    |LEA    ECX, DWORD PTR DS:[EAX+3]
00408E12 |. 83E1 0F    |AND   ECX, 0F
00408E15 |. 8D58 07    |LEA    EBX, DWORD PTR DS:[EAX+7]
00408E18 |. 83E3 0F    |AND   EBX, 0F
00408E1B |. 0FBE4C0C 28 |MOVSX  ECX, BYTE PTR SS:[ESP+ECX+28]
00408E20 |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]
00408E25 |. 03CE      |ADD    ECX, ESI
00408E27 |. 0FAFCB    |IMUL   ECX, EBX
00408E2A |. 8D58 02    |LEA    EBX, DWORD PTR DS:[EAX+2]
00408E2D |. 83E3 0F    |AND   EBX, 0F
00408E30 |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]
00408E35 |. 03DE      |ADD    EBX, ESI
00408E37 |. 8D70 04    |LEA    ESI, DWORD PTR DS:[EAX+4]
00408E3A |. 83E6 0F    |AND   ESI, 0F
00408E3D |. 0FBE7434 28 |MOVSX  ESI, BYTE PTR SS:[ESP+ESI+28]
00408E42 |. 0FAFDE    |IMUL   EBX, ESI
00408E45 |. 03CB      |ADD    ECX, EBX
00408E47 |. 8D70 F8    |LEA    ESI, DWORD PTR DS:[EAX-8]
00408E4A |. 8D58 06    |LEA    EBX, DWORD PTR DS:[EAX+6]
00408E4D |. 83E6 0F    |AND   ESI, 0F
00408E50 |. 83E3 0F    |AND   EBX, 0F
00408E53 |. 0FBE7434 28 |MOVSX  ESI, BYTE PTR SS:[ESP+ESI+28]
00408E58 |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]
00408E5D |. 0FAFF3    |IMUL   ESI, EBX
00408E60 |. 03CE      |ADD    ECX, ESI
00408E62 |. 8D70 F9    |LEA    ESI, DWORD PTR DS:[EAX-7]
00408E65 |. 8D58 05    |LEA    EBX, DWORD PTR DS:[EAX+5]
00408E68 |. 83E6 0F    |AND   ESI, 0F
00408E6B |. 83E3 0F    |AND   EBX, 0F
00408E6E |. 0FBE7434 28 |MOVSX  ESI, BYTE PTR SS:[ESP+ESI+28]
00408E73 |. 0FBE5C1C 28 |MOVSX  EBX, BYTE PTR SS:[ESP+EBX+28]

```

```

00408E78 |. 0FAFF3    |IMUL ESI, EBX
00408E7B |. 03CE      |ADD ECX, ESI
00408E7D |. 0FAFCA    |IMUL ECX, EDX
00408E80 |. C1F9 09   |SAR ECX, 9
00408E83 |. 83E1 1F   |AND ECX, 1F
00408E86 |. 83F9 0A   |CMP ECX, 0A
00408E89 |. 7D 05     |JGE SHORT PictureF.00408E90
00408E8B |. 83C1 50   |ADD ECX, 50
00408E8E |. EB 0D     |JMP SHORT PictureF.00408E9D
00408E90 |> 83F9 12   |CMP ECX, 12
00408E93 |. 7D 05     |JGE SHORT PictureF.00408E9A
00408E95 |. 83C1 28   |ADD ECX, 28
00408E98 |. EB 03     |JMP SHORT PictureF.00408E9D
00408E9A |> 83C1 2F   |ADD ECX, 2F
00408E9D |> 880C28    |MOV BYTE PTR DS:[EAX+EBP], CL
00408EA0 |. 8BC7      |MOV EAX, EDI
00408EA2 |. 8D50 FA   |LEA EDX, DWORD PTR DS:[EAX-6]
00408EA5 |. 83FA 10   |CMP EDX, 10
00408EA8 |.^ 0F8C F0FEFFFF \JL PictureF.00408D9E
-----Loop=16 times , End loop 2-----

```

```

00408EAE |. 68 CCE94700 PUSH PictureF.0047E9CC ; ASCII "0000000000000000"
00408EB3 |. 8D4C24 14 LEA ECX, DWORD PTR SS:[ESP+14]
00408EB7 |. E8 5F750300 CALL PictureF.0044041B
00408EBC |. 5F         POP EDI
00408EBD |. 5D         POP EBP
00408EBE |. 33F6      XOR ESI, ESI ==> bắt đầu với ESI = 0
00408EC0 |. 5B         POP EBX
-----Start Loop 3 , loop=16-----

```

```

00408EC1 |> 8A4434 0C MOV AL, BYTE PTR SS:[ESP+ESI+C] ==> lần lượt đưa từng char của
Rel Ser vào EAX
00408EC5 |. 8D4C24 04 LEA ECX, DWORD PTR SS:[ESP+4]
00408EC9 |. 50         PUSH EAX
00408ECA |. 56         PUSH ESI
00408ECB |. E8 F6780300 CALL PictureF.004407C6
00408ED0 |. 46         INC ESI ==> ESI ++
00408ED1 |. 83FE 10   CMP ESI, 10 ==> so sánh ESI với 10
00408ED4 |.^ 7C EB     JL SHORT PictureF.00408EC1 ==> nếu nhỏ hơn thì tiếp tục vòng lặp
-----End loop 3-----

```

-----đến đây nhìn sang cửa sổ bên cạnh , ta sẽ thấy Real Ser của c/t sau khi tính toán được đặt trong thanh ghi EAX-----

```

00408ED6 |. 8B7424 40 MOV ESI, DWORD PTR SS:[ESP+40]
00408EDA |. 8D4C24 04 LEA ECX, DWORD PTR SS:[ESP+4]
00408EDE |. 51         PUSH ECX
00408EDF |. 8BCE       MOV ECX, ESI
00408EE1 |. E8 21710300 CALL PictureF.00440007
00408EE6 |. B8 01000000 MOV EAX, 1
00408EEB |. 894424 08 MOV DWORD PTR SS:[ESP+8], EAX
00408EEF |. 8D4C24 04 LEA ECX, DWORD PTR SS:[ESP+4]
00408EF3 |. 884424 38 MOV BYTE PTR SS:[ESP+38], AL
00408EF7 |. E8 96730300 CALL PictureF.00440292
00408EFC |. 8D4C24 44 LEA ECX, DWORD PTR SS:[ESP+44]
00408F00 |. C64424 38 00 MOV BYTE PTR SS:[ESP+38], 0

```

```

00408F05 |. E8 88730300 CALL PictureF.00440292
00408F0A |. 8B4C24 30 MOV ECX, DWORD PTR SS:[ESP+30]
00408F0E |. 8BC6 MOV EAX, ESI
00408F10 |. 5E POP ESI
00408F11 |. 64:890D 00000>MOV DWORD PTR FS:[0], ECX
00408F18 |. 83C4 38 ADD ESP, 38
00408F1B \. C2 0800 RETN 8

-----
00408B3A |. 8B7424 10 MOV ESI, DWORD PTR SS:[ESP+10]
00408B3E |. 8B4424 18 MOV EAX, DWORD PTR SS:[ESP+18]

```

- Nhìn sang của sổ bên cạnh, chúng ta thấy 1 vầng hào quang sáng chói hiện ra .Vầng đó chính là RS- cái mà chúng ta cần tìm : ASCII "TICQTGIRG8JDHE2N"

/*/*/* - SERIAL tương ứng :

Email :hoa_dong_noi_06@yahoo.com Serial : TICQTGIRG8JDHE2N

Hoặc :

Email :REA-cRaCkErTeAm@yahoo.com Serial : 69K24FLMJRG9ILVC

III/ End Tut :

- Finished : 01/12/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.verypdf.com
Production :	Verypdf.com , Inc
SoftWare :	EnCrypt PDF v2.1
Copyright by :	Copyright (C) 2001 - 2003 Verypdf.com , Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo
Language :	Microsoft Visual C++ 6.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	Manual
Request :	Correct Serial / KeyGen

EnCrypt PDF v2.1

Encrypt PDF software is a very flexible and powerful program, Encrypt PDF software allows you to encrypt (using standard 40-bit or 128-bit supported by Acrobat Reader 5.0 and up) existing PDFs, set permissions, add user and owner password. For example you can encrypt a PDF without to allow to print it. The button to print the file will be disabled in Acrobat Reader application, you also can encrypt a PDF allowing the user to read it only if he know the correct password.

Note, The source PDF must be an unencrypted PDF. Acrobat Reader will show a key in the bottom status bar if the PDF is encrypted.....

I – Information :

- Now , bây giờ chúng ta lại tiếp tục lôi đồ nghề ra nào .Dùng PEid v0.92 để Detect , chúng ta biết chương trình được pack bằng **UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo** . Ta dùng tool : UPX Unpack để unpack chương trình này , hoặc chúng ta cũng có thể dùng Plugin của PEiD (có ReBuildPE) để unpack . Trước khi Unpack Size là : 24KB , sau khi Unpack Size là : 260 KB . Sau khi unpack xong , lại dùng PEiD v0.92 để Detect , chúng ta biết chương trình được viết bằng Microsoft Visual C++ 6.0.

- Vẫn là động tác quen thuộc: Chạy thử chương trình , chúng ta thấy hiện lên một 1 Dialog Box : “**Please register EnCrypt PDF v2.1**” , và chương trình chỉ cho phép chúng ta dùng thử 100 lần (chắc chắc.... sao ít thế nếu chúng ta ko đưa money cho họ và đổi lại họ sẽ đưa số Serial cho chúng ta . Đưa money để buy soft á . Nghe hơi lạ tai , chúng ta đâu có thói quen này , và lần này cũng vậy thôi hăng VeryPdf ạ (trừ khi Free All cho chúng ta)! Hihiiii..... Muốn vậy thì chúng ta chỉ có 1 cách duy nhất là Kill & Kill nó thôi .

- OK ! bây giờ nhìn vào Dialog ta thấy mã ID của Software : Product ID : [ZLTA987654325JNX] (cái này ta ko cần quan tâm làm gì cho mệt óc) . Nhìn xuống phía dưới là chỗ mà ta phải nhập Series , đây mới là chỗ ta cần . OK ! Ta nhập đại vào 1 cái Series xem nó bảo sao . Ở đây em nhập là : Series : 0361985 . Nhấn OK . Oh ! Tất lê dĩ ngẫu là 1 cái nag sẽ văng ra , đập ngay vào mặt chúng ta (ko văng nag ra mới là chuyện lạ) : "**Series number error , please check it and try again**" , đã vậy khi chúng ta exit cái Dialog Box đó nó còn pop-up ra 1 cái nag nhở ta : “Please register” , đòi tiền như đòi nợ ấy . Ok , chúng ta hãy ghi nhớ lấy tất cả những thứ đó .

- Bây giờ chúng ta hãy load chương trình vào trong OllyDbg . Nếu nó hiện ra 1 bảng thông báo thì cứ nhấn OK , sau đó nó lại cho ta chọn Yes (Analysis) & No (Analysis) . Ta chọn No (Analysis) . Sau đó click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . Ấc.....mất 1 phút mà vẫn không thấy cái chuỗi thông báo đó đâu cả . Chẳng nhẽ chúng ta lại chịu bó tay trước thời cuộc sao . Không đâu các bạn cứ bình tĩnh , chúng ta lại sử dụng phương pháp Stack của anh Moonbaby .

- Nhấn Ctrl + F2 để load lại chương trình vào trong OllyDbg . Nhấn F9 để Run chương trình . Chúng ta lại nhập FS như ở trên vào . Xong nhấn OK , xuất hiện thông báo : "Series number error , please check it and try again ". Giữ nguyên chương trình , quay trở lại Ollydbg , nhấn F12 , Olly sẽ dừng chương trình lại . Nhấn tiếp Alt + K để hiện cửa sổ : “Call stack of main thread “. Ta thấy ngay thông báo :

- Ở đây chúng ta chú ý đến dòng màu đỏ :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012EE08	003218A9	? USER32.MessageBoxA		verypdf.003218A3
0012EE0C	001803CA	hOwner = 001803CA ('Please registe		
0012EE10	00324254	Text = "Series number error, pleas		
0012EE14	00000000	Title = NULL		
0012EE18	00000010	Style = MB_OK MB_ICONHAND MB_APPLM		

- Double-click vào cột Called from của dòng này , chúng ta đến địa chỉ :

0032189D 68 54423200 PUSH verypdf.00324254 ; ASCII "Series number error, please check it and try again."

003218A2 56 PUSH ESI

003218A3 FF15 88313200 CALL NEAR DWORD PTR DS:[323188] ; USER32.MessageBoxA
==> We're here .

003218A9 68 FB030000 PUSH 3FB ==> Set BreakPoint here .

003218AE 56 PUSH ESI

- Sau khi đặt BP tại đây , Ollydbg sẽ dừng chương trình lại , nhấn F8 để xuất hiện lại thông báo : “Series number error, please check it and try again ”. Nhấn OK để chấp nhận thông báo này , chương trình sẽ dừng lại tại điểm BP mà chúng ta vừa đặt . Xoá điểm BP này đi . RETN , nhìn lên trên 1 chút chúng ta sẽ

thấy đoạn code sau . Đây là tử huyệt , chúng ta sẽ đặt BP tại đây :

```
00321849 FF15 80313200 CALL NEAR DWORD PTR DS:[323180] ; USER32.GetDlgItemTextA
==> Set BreakPoint Here .
0032184F 68 C8433200 PUSH verypdf.003243C8 ; ASCII "0361985"
```

II – Cracking :

- OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình , ta sẽ thấy hộp thoại đăng ký xuất hiện lại , sau khi đã nhập FS đầy đủ như ở trên ta đã làm , nhấn OK . Chúng ta sẽ quay trở lại Olly và chương trình sẽ Ice tại điểm mà chúng ta set BP. Nhìn xuống phía dưới 1 chút , chúng ta sẽ thấy hàm Call . Đây chính là lệnh gọi hàm kiểm tra Series :

```
00321854 E8 A7F7FFFF CALL verypdf.00321000
```

- Ở đây chúng ta cũng thấy lệnh text thanh ghi EAX . Vì vậy chúng ta sẽ Trace Into vào trong hàm Call trên xem nó làm gì với FS của chúng ta :

```
00321849 FF15 80313200 CALL NEAR DWORD PTR DS:[323180]; USER32.GetDlgItemTextA
==> We're here .
```

```
0032184F 68 C8433200 PUSH verypdf.003243C8 ; ASCII "0361985"
00321854 E8 A7F7FFFF CALL verypdf.00321000 ==> gọi hàm check Serial .
```

-----Trace Into-----

```
00321000 83EC 18 SUB ESP, 18
00321003 53 PUSH EBX
00321004 55 PUSH EBP
00321005 56 PUSH ESI
00321006 8B7424 28 MOV ESI, DWORD PTR SS:[ESP+28] ==> Đưa Input vào ESI
0032100A 8D5424 0C LEA EDX, DWORD PTR SS:[ESP+C]
0032100E 57 PUSH EDI
0032100F 8A46 0E MOV AL, BYTE PTR DS:[ESI+E] ==> Đưa ký tự thứ 15 trong chuỗi FS vào AL
00321012 8A4E 0F MOV CL, BYTE PTR DS:[ESI+F] ==> Đưa ký tự thứ 16 trong chuỗi FS vào CL
00321015 8B3D 60313200 MOV EDI, DWORD PTR DS:[323160] ; msvcrt atoi
0032101B 32DB XOR BL, BL
0032101D 52 PUSH EDX
0032101E 884424 20 MOV BYTE PTR SS:[ESP+20], AL ==> đoạn SS:[ESP+20] chứa ký tự thứ 15
00321022 885C24 21 MOV BYTE PTR SS:[ESP+21], BL
00321026 884C24 14 MOV BYTE PTR SS:[ESP+14], CL ==> đoạn SS:[ESP+14] chứa ký tự thứ 16
0032102A 885C24 15 MOV BYTE PTR SS:[ESP+15], BL
0032102E FFD7 CALL NEAR EDI
00321030 8BE8 MOV EBP, EAX ==> Đưa ký tự thứ 16 trong chuỗi FS vào EBP
00321032 8D4424 20 LEA EAX, DWORD PTR SS:[ESP+20] ==> EAX chứa ký tự thứ 15
00321036 50 PUSH EAX
00321037 FFD7 CALL NEAR EDI
00321039 03E8 ADD EBP, EAX ==> lấy tổng của ký tự 15 + ký tự 16
0032103B 83C4 08 ADD ESP, 8
0032103E 83FD 0B CMP EBP, 0B ==> so sánh với 0B
00321041 74 0A JE SHORT verypdf.0032104D ==> Nếu bằng thì tiếp tục quá trình tính toán Serial
00321043 5F POP EDI
00321044 5E POP ESI
00321045 5D POP EBP
00321046 33C0 XOR EAX, EAX
00321048 5B POP EBX
00321049 83C4 18 ADD ESP, 18
```

```

0032104C C3          RETN
0032104D 8A0E MOV CL, BYTE PTR DS:[ESI] ==> Đưa ký tự thứ nhất trong chuỗi FS vào CL
0032104F 8A56 01 MOV DL, BYTE PTR DS:[ESI+1] ==> Đưa ký tự thứ 2 trong chuỗi FS vào DL
00321052 8D4424 10    LEA   EAX, DWORD PTR SS:[ESP+10]
00321056 884C24 1C    MOV   BYTE PTR SS:[ESP+1C], CL
0032105A 50          PUSH  EAX
0032105B 885C24 21    MOV   BYTE PTR SS:[ESP+21], BL
0032105F 885424 14    MOV   BYTE PTR SS:[ESP+14], DL
00321063 885C24 15    MOV   BYTE PTR SS:[ESP+15], BL
00321067 FFD7        CALL  NEAR EDI
00321069 8D4C24 20    LEA   ECX, DWORD PTR SS:[ESP+20]
0032106D 8BD8        MOV   EBX, EAX
0032106F 51          PUSH  ECX
00321070 FFD7        CALL  NEAR EDI
00321072 03D8        ADD   EBX, EAX ==> lấy tổng của ký tự thứ nhất + ký tự thứ 2
00321074 83C4 08      ADD   ESP, 8
00321077 83FB 09      CMP   EBX, 9 ==> so sánh với 9
0032107A 74 0A JE SHORT verypdf.00321086 ==> Nếu bằng thì tiếp tục quá trình tính toán Serial
0032107C 5F          POP   EDI
0032107D 5E          POP   ESI
0032107E 5D          POP   EBP
0032107F 33C0        XOR   EAX, EAX
00321081 5B          POP   EBX
00321082 83C4 18      ADD   ESP, 18
00321085 C3          RETN
00321086 8A4E 05    MOV  CL, BYTE PTR DS:[ESI+5] ==> Đưa ký tự thứ 6 trong chuỗi FS vào CL
00321089 33C0        XOR   EAX, EAX
0032108B 5F          POP   EDI
0032108C 80F9 2A    CMP  CL, 2A [COLOR=Yellow][B]==> so sánh ký tự thứ 6 với ký tự '*' 
0032108F 5E          POP   ESI
00321090 5D          POP   EBP
00321091 0F94C0      SETE AL
00321094 5B          POP   EBX
00321095 83C4 18      ADD   ESP, 18
00321098 C3          RETN
-----
00321859 83C4 04      ADD   ESP, 4
0032185C 85C0        TEST  EAX, EAX ==> kiểm tra EAX= 0 ?
0032185E 74 39 JE     SHORT verypdf.00321899 ==> nếu EAX=0 thì nhảy đến bắn Bad boy .
00321860 6A 40 PUSH  40
00321862 68 B4423200 PUSH  verypdf.003242B4 ; ASCII "Thank you purchased the"
00321867 68 88423200 PUSH  verypdf.00324288 ; ASCII "Thank you purchased the encryptpdf
v2.1."
0032186C 56          PUSH  ESI

```

- Như vậy theo phân tích ở trên ta thấy Real Serial của chương trình sẽ phải thoả mãn các yêu cầu sau :
- 1. Chiều dài tối thiểu phải là 16 ký tự
- 2. Ở vị trí thứ 6 trong chuỗi Ser phải là ký tự mặc định ‘ * ’
- 3. Các ký tự ở vị trí S[0] , S[1] , S[14] , S[15] phải là các ký tự thoả mãn điều kiện sau :
 $S[0] + S[1] = 0x9$
 $S[14] + S[15] = 0xB$
- 4. Các ký tự còn lại là mặc định .

- Từ đó ta có thể dễ dàng suy ra Real Serial của chương trình là : 54321*0123456783

- Chúng ta lại làm tiếp những động tác quen thuộc cuối cùng , tắt Ollydbg , chạy thử chương trình , nhập vào : Series : 54321*0123456783

OK ! Done ! Chương trình cảm ơn chúng rồi rít ta kia “thank you”,

/*/*/* - SERIAL tương ứng :

Serial : 54321*0123456783

III/ Keygen :

```
char reaSerial[20]={0};
char
reaRandom[63]="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
char reaRandom2[9]={"23456789"};
int i=0, randomChart=0;
    // random chart
while (i <16)
{
    reaSerial[i] = reaRandom[randomChart = rand() % 62];
    i++;
}

//Default cho vi tri thu 5 trong reaSerial
reaSerial[5]= 0x2A;

//tong cua vi tri thu nhat va thu 2 phai bang 0x9
reaSerial[0] = reaRandom[randomChart = rand() % 10];
reaSerial[1] = (0x9 - ((reaSerial[0] & 0xFF) - 0x30)) + 0x30;

// tong cua vi tri thu 14 va 15 phai bang 0xB
reaSerial[14] = reaRandom2[randomChart = rand() % 8];
reaSerial[15] = (0xB - ((reaSerial[14] & 0xFF) - 0x30)) + 0x30;

SetDlgItemText(IDC_SERIAL,reaSerial);
```

III/ End Tut :

- Finished : 20/11/2004

Reverse Engineering Association

SoftWare

Homepage	:	http://www.verypdf.com
Production	:	Verypdf.com , Inc
SoftWare	:	PDF Extract TIFF v1.5
Copyright by	:	Copyright (C) 2001 - 2003 Verypdf.com , Inc. All Rights Reserved.
Type	:	Name / Serial
Packed	:	N/A
Language	:	Microsoft Visual C++ 6.0
Crack Tool	:	OllyDbg 1.09d, PEiD 0.92, kWdsdm 10
Unpack	:	Manual
Request	:	Correct Serial / KeyGen

PDF Extract TIFF v1.5

PDF Extract TIFF software is a simple and affordable extraction tool that facilitates the reuse of PDF files by enabling you to extract images from PDF files and save them as TIFF images. You can then use or edit the images in other Windows applications such as MS Word, Adobe Photoshop, or any other image enhancement/manipulation program. PDF Extract TIFF software can be used to extract pictures from your PDF file on the fly, but it can't convert text, graphics and other information into the TIFF or JPG files.

I – Information :

- OK! Bây giờ chúng ta lại bắt đầu làm những động tác “nhà nghề” quen thuộc . Now, Let’s go !. Dùng PEid v0.92 để Detect , chúng ta biết chương trình không bị Pack , và được viết bằng **Microsoft Visual C++ 6.0**. Phù , thế là không phải Unpack . Hihiii.....!
- Chạy thử chương trình , chúng ta thấy hiện lên một 1 Dialog Box : “Please register PDF Extract TIFF v1.5 ”, và chương trình chỉ cho phép chúng ta dùng thử 100 lần nếu chúng ta ko đưa money cho họ và đổi lại họ sẽ đưa số Serial cho chúng ta . Muốn vậy thì chúng ta chỉ có 1 cách duy nhất là Kill & Kill nó thôi .
- OK ! bây giờ nhìn vào Dialog ta thấy mã ID của Software : Product ID : [ZLTA987654325JNX] (cái này ta ko cần quan tâm làm gì cho mệt óc) . Nhìn xuống phía dưới là chỗ mà ta phải nhập Series , đây mới là chỗ ta cần . OK ! ta nhập đại vào 1 cái Series xem nó bảo sao . Ở đây em nhập là : Series : 0361985 . Nhấn OK . Oh ! 1 cái nag văng ra đập ngay vào mặt chúng ta : "Series number error , please check it and try again " . Thế có ức ko cơ chứ , mà cũng đúng thôi , nếu chúng ta nhập đúng thì chúng ta đã chẳng phải ngồi đây . Khi chúng ta exit cái Dialog Box đó nó còn pop-up ra 1 cái nag nhắc nhở ta : "Please register" , Ok , chúng ta hãy ghi nhớ lấy tất cả những thứ đó .

- Bây giờ chúng ta hãy load chương trình vào trong OllyDbg , click chuột phải chọn Search for / All referenced text strings để tìm chuỗi thông báo trên . Ấc ặc..... mất mấy vài phút mà vẫn không thấy cái thông báo đó đâu cả . Chẳng nhẽ chúng ta lại chịu bó tay trước thời cuộc sao . Không đâu các bạn cứ bình tĩnh , chúng ta lại sử dụng phương pháp Stack của anh **Moonbaby** .

- Nhấn Ctrl + F2 để load lại chương trình vào trong OllyDbg . Nhấn F9 để Run chương trình . Chúng ta lại nhập FS như ở trên vào , ở đây ta nhập là Series : 0361985 . Xong nhấn Ok , xuất hiện thông báo : "Series number error, please check it and try again ". Giữ nguyên chương trình , quay trở lại Ollydbg , nhấn F12 , Olly sẽ dừng chương trình lại.Nhấn tiếp Alt + K để hiện cửa sổ : "Call stack of main thread " .

Ta thấy ngay thông báo :

*** Ở đây chúng ta chú ý đến dòng màu đỏ :

Call stack of main thread

Address	Stack	Procedure / arguments	Called from	Frame
0012EF60	10001867	? USER32.MessageBoxA	verypdf.10001861	
0012EF64	00150356	hOwner = 00150356 ('Please registe		
0012EF68	100172D0	Text = "Series number error, pleas		
0012EF6C	00000000	Title = NULL		
0012EF70	00000010	Style = MB_OK MB_ICONHAND MB_APPLM		

- Double-click vào cột “Called from” của dòng màu đỏ này , chúng ta đến địa chỉ sau :

1000185B 68 D0720110 PUSH verypdf.100172D0 ; ASCII "Series number error, please check it and try again."

10001860 56 PUSH ESI

10001861 FF15 34330110 CALL NEAR DWORD PTR DS:[<&USER32.Mess>;

USER32.MessageBoxA ==> We're here .

10001867 68 FB030000 PUSH 3FB ==> Set BreakPoint here .

1000186C 56 PUSH ESI

- Sau khi đặt BP tại đây , Ollydbg sẽ dừng chương trình lại , nhấn F8 để xuất hiện lại thông báo : "Series number error, please check it and try again ". Nhấn OK để chấp nhận thông báo này , chương trình sẽ dừng lại tại điểm BP mà chúng ta vừa đặt . Xoá điểm BP này đi . RETN , nhìn lên trên 1 chút chúng ta sẽ thấy đoạn code sau . Đây là từ huyệt , ta sẽ đặt BP tại đây :

10001807 FF15 2C330110 CALL NEAR DWORD PTR DS:[<&USER32.GetD>;

USER32.GetDlgItemTextA ==> Set BreakPoint Here .

1000180D 68 20A80110 PUSH verypdf.1001A820 ; ASCII "0361985"

II – Cracking :

- OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình , ta sẽ thấy hộp thoại đăng ký xuất hiện lại , sau khi đã nhập FS đầy đủ như ở trên ta đã làm , nhấn OK. Chúng ta sẽ quay trở lại Olly và chương trình sẽ Ice tại điểm mà chúng ta set BP.Nhìn xuống phía dưới 1 chút , chúng ta sẽ thấy hàm Call . Đây chính là lệnh gọi hàm kiểm tra Series :

10001812 E8 E9F7FFFF CALL verypdf.10001000

Ở đây chúng ta cũng thấy lệnh text thanh ghi EAX . Vì vậy chúng ta sẽ Trace Into vào trong hàm Call trên xem nó làm gì với FS của chúng ta :

10001807 FF15 2C330110 CALL NEAR DWORD PTR DS:[<&USER32.GetD>;
USER32.GetDlgItemTextA ==> We're here .

1000180D 68 20A80110 PUSH verypdf.1001A820

10001812 E8 E9F7FFFF CALL verypdf.10001000 ==> gọi hàm check Serial .

-----Trace Into-----

10001000 83EC 18 SUB ESP, 18

10001003 56 PUSH ESI

10001004 8B7424 20 MOV ESI, DWORD PTR SS:[ESP+20] >>> Đưa Input vào ESI

10001008 8D5424 04 LEA EDX, DWORD PTR SS:[ESP+4]

1000100C 57 PUSH EDI

1000100D 8A06 MOV AL, BYTE PTR DS:[ESI] ==> Đưa ký tự thứ nhất trong chuỗi FS vào AL

1000100F 8A4E 0F MOV CL, BYTE PTR DS:[ESI+F] ==> Đưa ký tự thứ 16 trong chuỗi FS vào CL

10001012 884424 14 MOV BYTE PTR SS:[ESP+14], AL => đoạn SS:[ESP+14] chứa ký tự thứ nhất

```

10001016 32C0          XOR   AL, AL
10001018 52             PUSH  EDX
10001019 884424 19     MOV    BYTE PTR SS:[ESP+19], AL
1000101D 884C24 0C     MOV    BYTE PTR SS:[ESP+C], CL => đoạn SS:[ESP+C] chứa ký tự thứ 16
10001021 884424 0D     MOV    BYTE PTR SS:[ESP+D], AL => đoạn SS:[ESP+D] chứa ký tự thứ nhất .
10001025 E8 FB140000   CALL   verypdf.10002525
1000102A 8BF8           MOV    EDI, EAX ==> Đưa ký tự thứ 16 trong chuỗi FS vào EDI
1000102C 8D4424 18     LEA    EAX, DWORD PTR SS:[ESP+18] ==> Đưa ký tự thứ nhất trong chuỗi
FS vào EAX
10001030 50             PUSH  EAX
10001031 E8 EF140000   CALL   verypdf.10002525
10001036 03F8           ADD    EDI, EAX ==> lấy tổng của ký tự thứ nhất + ký tự thứ 16
10001038 83C4 08         ADD    ESP, 8
1000103B 83FF 0A         CMP    EDI, 0A ==> so sánh với 0A
1000103E 74 08 JE       SHORT verypdf.10001048 ==> Nếu bằng thì tiếp tục quá trình tính toán Serial
10001040 5F             POP   EDI
10001041 33C0           XOR   EAX, EAX
10001043 5E             POP   ESI
10001044 83C4 18         ADD   ESP, 18
10001047 C3             RETN
10001048 807E 01 38     CMP   BYTE PTR DS:[ESI+1], 38 ==> So sánh ký tự thứ 2 với 8
1000104C 74 08 JE       SHORT verypdf.10001056 ==> Nếu bằng thì tiếp tục quá trình tính toán Serial .
1000104E 5F             POP   EDI
1000104F 33C0           XOR   EAX, EAX
10001051 5E             POP   ESI
10001052 83C4 18         ADD   ESP, 18
10001055 C3             RETN
10001056 8A4E 05 MOV CL, BYTE PTR DS:[ESI+5] ==> Đưa ký tự thứ 6 trong chuỗi FS vào CL
10001059 33C0           XOR   EAX, EAX
1000105B 80F9 2A CMP CL, 2A ==> so sánh ký tự thứ 6 với ký tự ' * '
1000105E 5F             POP   EDI
1000105F 0F94C0         SETE  AL
10001062 5E             POP   ESI
10001063 83C4 18         ADD   ESP, 18
10001066 C3             RETN
-----  

10001817 83C4 04         ADD   ESP, 4
1000181A 85C0           TEST  EAX, EAX ==> kiểm tra EAX= 0 ?
1000181C 74 39 JE       SHORT verypdf.10001857 ==> nếu EAX=0 thì nhảy đến bắn Bad boy .
1000181E 6A 40 PUSH 40
10001820 68 30730110   PUSH   verypdf.10017330 ; ASCII "Thank you registered"
10001825 68 04730110   PUSH   verypdf.10017304 ; ASCII "Thank you registered
PDFExtractTIFF v1.5."

```

- Như vậy theo phân tích ở trên ta thấy Real Serial của chương trình sẽ phải thoả mãn các yêu cầu sau :
- 1. Chiều dài tối thiểu phải là 16 ký tự
- 2. Ở các vị trí thứ 2, 6 trong chuỗi Ser phải là các ký tự mặc định :
 - vị trí thứ 2 là 8
 - vị trí thứ 6 là *
- 3. Các ký tự ở vị trí S[0] , S[15] phải là các ký tự từ 1 -> 9 và phải thoả mãn điều kiện sau :
 $S[0] + S[15] = 0xA$
- 4. Các ký tự còn lại là mặc định .

- Từ đó ta có thể dễ dàng suy ra Real Serial của chương trình là : 18901*7890123459
- Chúng ta lại làm tiếp những động tác quen thuộc cuối cùng :) , tắt Ollydbg , chạy thử chương trình , nhập vào :

Series : 18901*7890123459 Hoặc :

Series : 58234*thanhvan85

OK ! Done ! Chương trình cảm ơn chúng rồi rít ta kia “thanks”, ngại quá.

/*/*/* - SERIAL tương ứng :

Serial : 58234*thanhvan85

III/ End Tut :

- Finished : 20/11/2004

Reverse Engineering Association SoftWare

Homepage :	http://www.cmfperception.com
Production :	CmfPerception, Inc
SoftWare :	Lite FTP v2.5
Copyright by :	Copyright (C) 2004 CmfPerception, Inc. All Rights Reserved.
Type :	Name / Serial
Packed :	N/A
Language :	Borland Delphi 4.0 - 5.0
Crack Tool :	OllyDbg 1.09d, PEiD 0.92, kWds 10
Unpack :	N/A
Request :	Correct Serial

Lite FTP v2.5

LiteFTP is a feature-packed FTP server with transfer statistics and integrated DynDNS and No-IP dynamic DNS clients .

I – Information :

- Dùng PEid v0.92 để Detect , chúng ta biết chương trình ko bị pack & được viết bằng **Borland Delphi 4.0 - 5.0**. Dùng tiếp Plugin Kanal của PEiD để Detect thì ta thấy nó dùng Crypto BASE64 table .

- Chạy thử chương trình , chúng ta thấy hiện ra một 1 Dialog Box bắt chúng ta phải đăng kí . Nếu ko nó chỉ cho phép chúng ta dùng thử trong 12 hours . Và nếu chúng ta muốn tiếp tục sử dụng thì chúng ta phải mua nó với giá **\$29.95** . Ở phía trên , thanh Title của soft còn có chữ Unregistered , nhìn thấy mà ghét . Vào menu Help/ Register/Order... ta nhập đại vào 1 cái Email xem nó bảo sao . Ở đây mình nhập là :Email:hoa_dong_noi_06@yahoo.com & Key : 0361985. Nhấn Button Register . Oh ! 1 cái nag sẽ văng ra , đập ngay vào mặt chúng ta : " Name and code do not match . Try again " . OK ! hãy nhớ lấy nó.

- Bây giờ chúng ta hãy load chương trình vào trong OllyDbg . Sau đó click chuột phải chọn **Search for / All referenced text strings** để tìm chuỗi thông báo trên . Ấc ặc..... mất 1 phút mới tìm thấy nó . Double click vào dòng đó , chúng ta sẽ quay trở lại màn hình chính của Olly :

```
0048D5EC . B8 A8D64800 MOV EAX, liteftp.0048D6A8 ; |ASCII "Name and key don't
match!\n\nTry again ;)" ==> We're here .
0048D5F1 . E8 7A9DFCFF CALL liteftp.00457370 ; \liteftp.00457370
```

- Lần ngược lên trên 1 chút , chúng ta sẽ đặt BreakPoint tại đây :

```
0048D500 . E8 73B50000 CALL liteftp.00498A78 ==> Set BreakPoint here.
0048D505 . 8B45 F4 MOV EAX, DWORD PTR SS:[EBP-C]
```

II – Cracking :

- OK, sau khi đặt BP tại đó , chúng ta nhấn F9 để Run chương trình .Chúng ta sẽ nhập lại Email và Key như trên . Nhấn Register, Olly sẽ Ice tiến trình tại điểm mà ta vừa đặt BP :

```
0048D500 . E8 73B50000 CALL liteftp.00498A78 ==> We're here .
```

-----Trace Into-----

```
00498A78 /$ 55 PUSH EBP
00498A79 |. 8BEC MOV EBP, ESP
00498A7B |. 6A 00 PUSH 0
00498A7D |. 6A 00 PUSH 0
00498A7F |. 6A 00 PUSH 0
00498A81 |. 53 PUSH EBX
00498A82 |. 56 PUSH ESI
00498A83 |. 8BF1 MOV ESI, ECX
00498A85 |. 8955 FC MOV [LOCAL.1], EDX ==> Input
00498A88 |. 8B45 FC MOV EAX, [LOCAL.1] ==> đưa Input vào EAX
00498A8B |. E8 C0B6F6FF CALL liteftp.00404150
00498A90 |. 33C0 XOR EAX, EAX ==> EAX =0
00498A92 |. 55 PUSH EBP
00498A93 |. 68 348B4900 PUSH liteftp.00498B34
00498A98 |. 64:FF30 PUSH DWORD PTR FS:[EAX]
00498A9B |. 64:8920 MOV DWORD PTR FS:[EAX], ESP
00498A9E |. 837D FC 00 CMP [LOCAL.1], 0 ==> Len(Input) >0
00498AA2 |. 75 09 JNZ SHORT liteftp.00498AAD
00498AA4 |. 8BC6 MOV EAX, ESI
00498AA6 |. E8 71B2F6FF CALL liteftp.00403D1C
00498AAB |. EB 6C JMP SHORT liteftp.00498B19
00498AAD > 8D45 F8 LEA EAX, [LOCAL.2]
00498AB0 |. BA 4C8B4900 MOV EDX, liteftp.00498B4C ; ASCII "Try BUYinG LiteFTP!!" ==>
đưa String ASCII "Try BUYinG LiteFTP!!" vào EDX
00498AB5 |. E8 FAB2F6FF CALL liteftp.00403DB4 ==> gọi hàm mã hoá .
00498ABA |. BB 01000000 MOV EBX, 1
```

-----Start Loop-----

```
00498ABF > 8B45 FC /MOV EAX, [LOCAL.1] ==> đưa Input vào EAX
00498AC2 |. E8 D5B4F6FF |CALL liteftp.00403F9C
00498AC7 |. 50 |PUSH EAX
00498AC8 |. 8BC3 |MOV EAX, EBX
00498ACA |. 48 |DEC EAX
00498ACB |. 5A |POP EDX
00498ACC |. 8BCA |MOV ECX, EDX
00498ACE |. 99 |CDQ
00498ACF |. F7F9 |IDIV ECX
00498AD1 |. 8B45 FC |MOV EAX, [LOCAL.1] ==> Input
```

00498AD4 |. 8A0410 |MOV AL, BYTE PTR DS:[EAX+EDX] ==> chuyển S[0] của Input vào AL
 00498AD7 |. 8B55 F8 |MOV EDX, [LOCAL.2] ==> đưa String ASCII "Try BUYinG LiteFTP!!"
 vào EDX
 00498ADA|. 8A541A FF |MOV DL, BYTE PTR DS:[EDX+EBX-1]==>chuyển S[0] của String vào DL
 00498ADE |. 32C2 |XOR AL, DL ==> Xor 2 char trên .
 00498AE0 |. 25 FF000000 |AND EAX, 0FF
 00498AE5 |. 8D55 F4 |LEA EDX, [LOCAL.3]
 00498AE8 |. E8 E307F7FF |CALL liteftp.004092D0
 00498AED |. 8B45 F4 |MOV EAX, [LOCAL.3]
 00498AF0 |. E8 A7B4F6FF |CALL liteftp.00403F9C
 00498AF5 |. 8B55 F4 |MOV EDX, [LOCAL.3]
 00498AF8 |. FF7402 FF |PUSH DWORD PTR DS:[EDX+EAX-1]
 00498AFC |. 8D45 F8 |LEA EAX, [LOCAL.2]
 00498AFF |. E8 68B6F6FF |CALL liteftp.0040416C
 00498B04 |. 5A |POP EDX
 00498B05 |.885418 FF |MOV BYTE PTR DS:[EAX+EBX-1], DL ==> chuyển S[1] của String vào DL
 00498B09 |. 43 |INC EBX ==> i++
 00498B0A |. 83FB 15 |CMP EBX, 15 ==> If i<15
 00498B0D |.^ 75 B0 |JNZ SHORT liteftp.00498ABF ==> then Loop
 -----End Loop-----
 00498B0F |. 8BC6 MOV EAX, ESI
 00498B11 |. 8B55 F8 MOV EDX, [LOCAL.2] ==> đưa Real Serial sau khi đã mã hoá vào EDX
 00498B14 |. E8 57B2F6FF CALL liteftp.00403D70 ==> gọi hàm kiểm tra 1 lần nữa .
 00498B19 |> 33C0 XOR EAX, EAX
 00498B1B |. 5A POP EDX
 00498B1C |. 59 POP ECX
 00498B1D |. 59 POP ECX
 00498B1E |. 64:8910 MOV DWORD PTR FS:[EAX], EDX
 00498B21 |. 68 3B8B4900 PUSH liteftp.00498B3B ; ASCII "^[å]"
 00498B26 |> 8D45 F4 LEA EAX, [LOCAL.3]
 00498B29 |. BA 03000000 MOV EDX, 3
 00498B2E |. E8 0DB2F6FF CALL liteftp.00403D40
 00498B33 \. C3 RETN

0048D505 . 8B45 F4 MOV EAX, DWORD PTR SS:[EBP-C] ==> đưa Real Serial vào EAX .
 0048D508 . 50 PUSH EAX

/*/*/* - SERIAL tương ứng :

Email : hoa_dong_noi_06@yahoo.com Serial : [09478854919748365938](#)

Hoặc :

Email : REA-cRaCkErTeAm@yahoo.com Serial : [65633762522423865938](#)

III/ End Tut :

- Finished : 19/12/2004

-----==RETEAM INFORMATION==-----

```
--=[Computer_Angel.....]=-....-[ Admin/Cracker/Coder ]=-
--=[Moonbaby.....]=-....-[ Admin/Cracker/Coder ]=-
--=[Zombie_Deathman..]=-....-[ Admin/Cracker/Coder ]=-
--=[Nini.....]=-....-[ Super Mod/Designer ]=-
--=[RCA.....]=-....-[ Sub Mod/Cracker/Coder ]=-
--=[Benina.....]=-....-[Sub Mod/Cracker/Translator]=-.
--=[Deux.....]=-....-[ Sub Mod/Cracker/Coder ]=-
--=[Exctlong.....]=-....-[ Sub Mod/Coder ]=-
--=[LittleBoy.....]=-....-[ Sub Mod/Cracker ]=-
--=[QhQCrk...]=-....-[ Sub Mod/Coder ]=-
--=[kienmanowar.....]=-....-[ Sub Mod/Cracker ]=-
--=[hoadongnoi.....]=-....-[ Sub Mod/Cracker ]=-
--=[the_Lighthouse.....]=-....-[ Sub Mod/Cracker ]=-
```

REA send GREETS to:

[HACNHO - DQLTN - HVA - ARTEAM - All REA's members & friends]

28/02/2005

==== kienmanowar ===



- Thank to my family, Computer_Angel, Moonbaby , Zombie_Deathman, Littleboy, Benina, QHQCrk... the_Lighthouse, Hoadongnoi, Nini ... all REA's members, HacNho, RongChauA, Deux.... all my friend, and YOU.