the NewbieZ-Project

30. January 2001

Basic Tutorials

IDA pro for Newbiez by medardus (version 2.0)

1. Einleitung

Normally one is the first Tools, with which a Newbie argues, W32DASM. This Tool is both easy to learn and simply serve. One hears somewhat later then of a Tool named “IDA pro”, which should be better viiieeeel than W32DASM. Fact is that W32DASM is not any longer developed further, while with IDA the pro is very much much the case. In addition fact is that IDA pro possesses some features, which make the Reverse engineering easier. But we come now finally to the actual topic…

2. What is IDA pro?

IDA pro (in the further one only “IDA” mentioned) is a these assembler and by the company DataRescue was developed. Who already knows W32DASM, knows, what a these assembler makes. For those among us, which do not know it: a these assembler is a program, with its assistance one an existing program (it is now a EXE, DLL o. A.) this-assemble, speak into the original assembler code back-change, can. This is not easy venture, which one can see on the duration of the these assembly of a normally large EXE. The result of the these assembly is anyhow the assembler code, which was prepared more or less well readable. IDA (like also W32DASM) offers now some functions, e.g. for pursuing conditioned jumps, for looking for stringer references etc., which facilitate navigation and the work in the SOURCE code. In addition IDA offers however features, which one misses with W32DASM painfully, like e.g. FLIRT (nearly LIBRARY identification and Recognition Technology), which helps, to recognize accordingly functions from different libraries and with comments provide to represent.

3. From where does one get IDA pro?

One keeps a demo of the most current IDA version (approx. 10.9 MT) direct with DataRescue. This has the advantage that her all current features, like e.g. AUTOMATIC parameter identification controls. However one cannot store (which with partial very long these assembly times the genuine deficiency is) and comes sometime a Timeout. One receives a somewhat older version (4,04, approx. 18.3 MT) under the following URLs. Here it concerns a full version, with which also storing is possible. I can only hope that soon a more current version comes in circulation, since above all the AUTOMATIC parameter identification a very nice feature is!

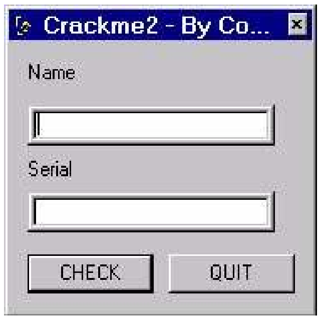
http://www.8bn.com/hambo/tools.html http://202.103.100.253/hambo/tools.html

4. Installation

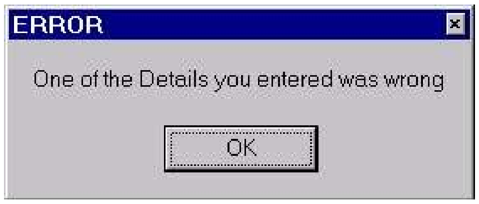
If one procured oneself the full version, one does not have to install anything. Unpack simply the ZIP files and "Idag.EXE" call. If one down-loaded oneself the demo version, one starts the Setup and answers the usual questions (where application is to be installed etc.).

5. Grundlegende Bedienung anhand eines Beispiels

In order to make the whole thing more descriptive, I settled a Crackme of a Cracker named "CoSH", which is beautifully small, so that one can reconstruct quite well the function with IDA. We start simply times the Crackme:

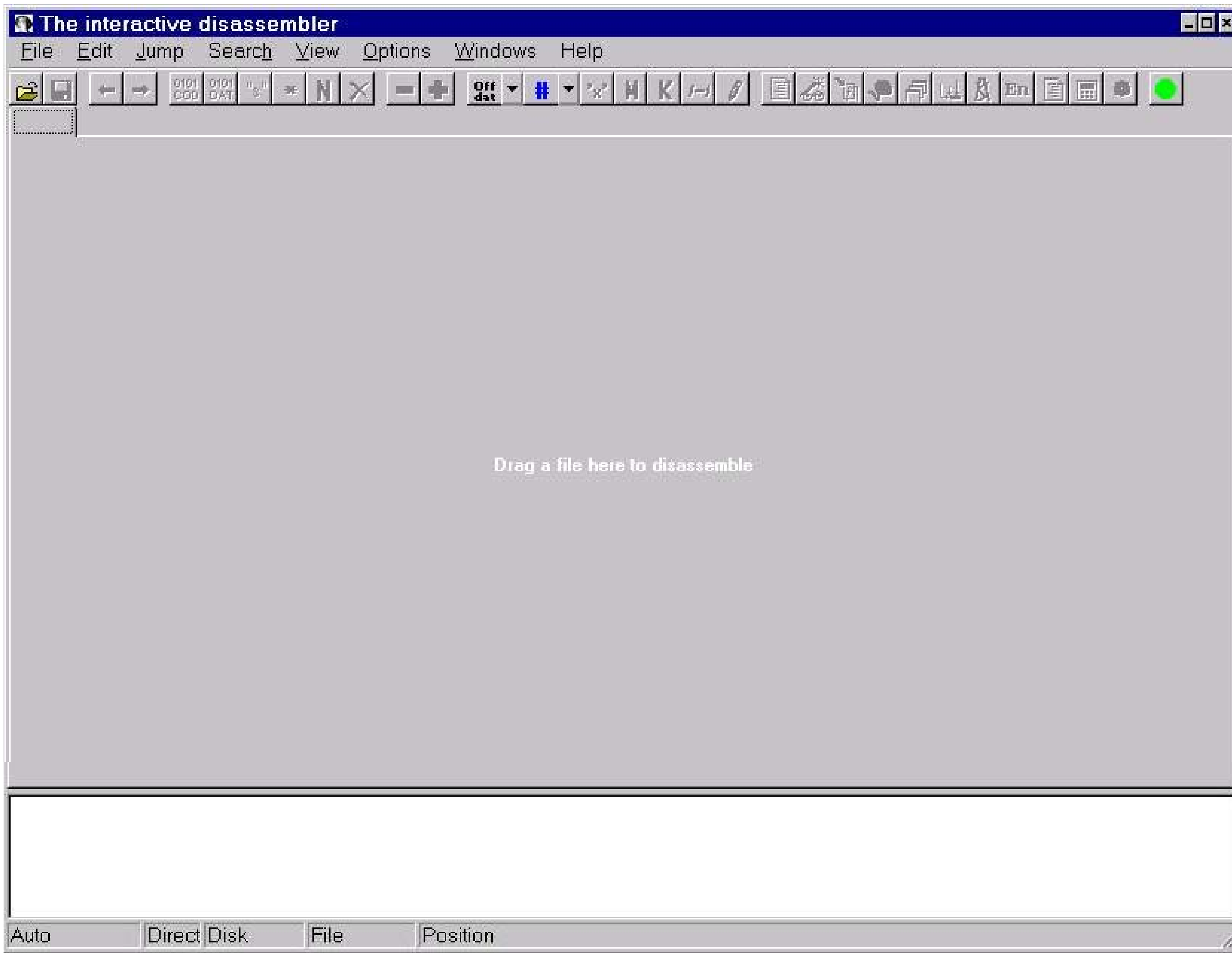


If we now our names (I take "medardus") and any Serial (I: "123456789") enters and on the "CHECKS" - Button click, appear the following message:

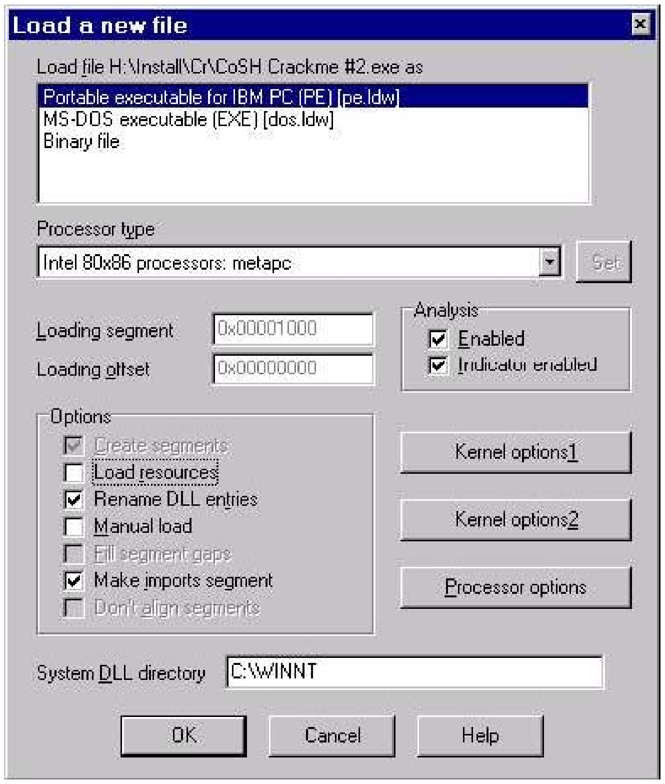

ERROR

One of the Details you entered was wrong

We have now a starting point, i.e. a stringer, which we can look for in the SOURCE code. Thus we start IDA (I refer to the full version 4,04, whereby naturally everything functions to on storing also with the demo version), by calling “IDAG.EXE”:



In order to this-assemble a program, we pull it simply into the IDA window or open it manually over the file menu. In both cases the following window appears:


Load a new file

Load file H:\Install\Cr\CoSH Crackme #2.exe as

- Portable executable for IBM PC (PE) [pe.ldw]
- MS-DOS executable (EXE) [dos.ldw]
- Binary file

Processor type: Intel 80x86 processors: metapc

Loading segment: 0x00001000

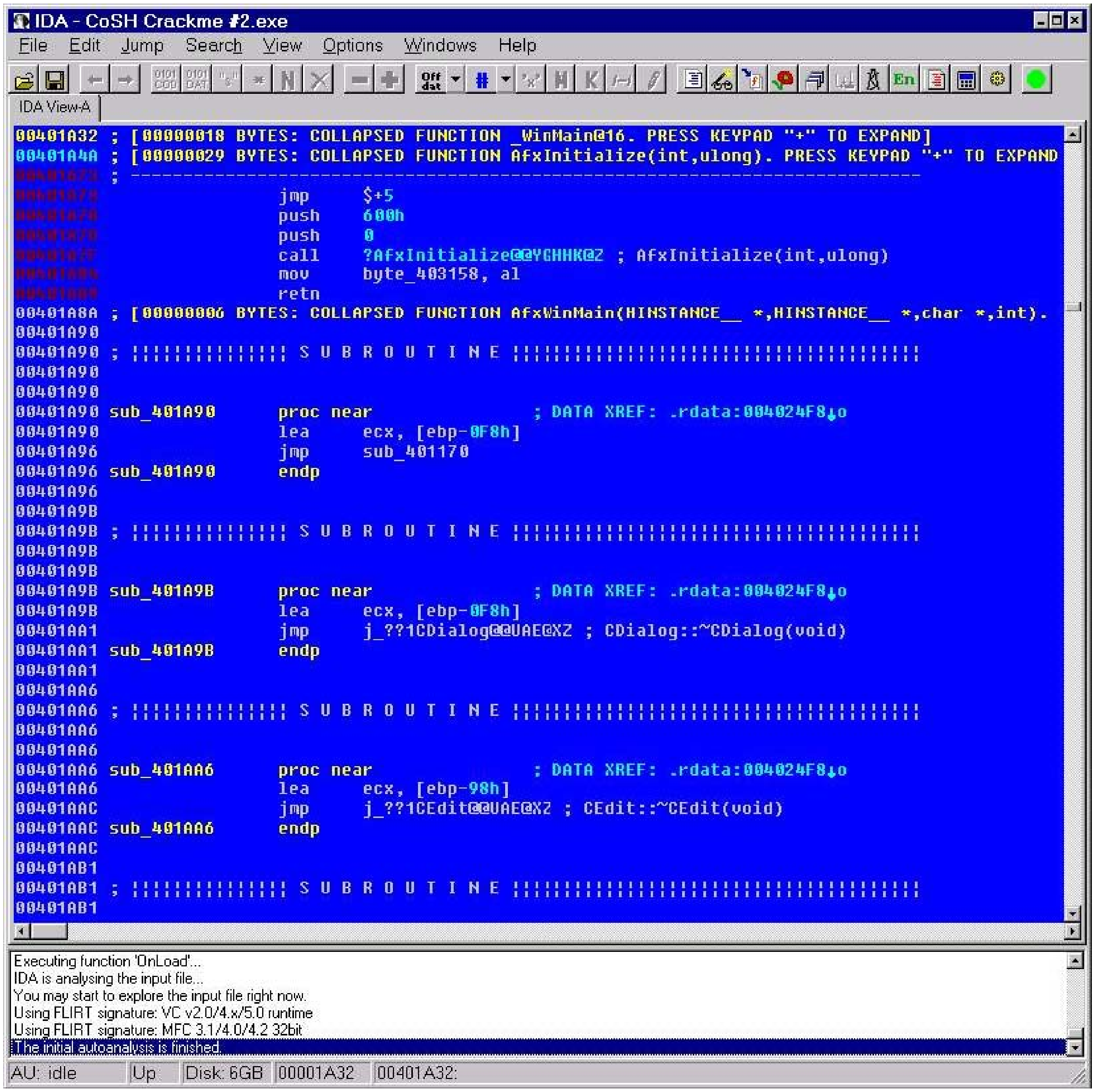
Loading offset: 0x00000000

Analysis
- ✓ Enabled
- ✓ Indicator enabled

Options
- ✓ Create segments
- ☐ Load resources
- ✓ Rename DLL entries
- ☐ Manual load
- ☐ Fill segment gaps
- ✓ Make imports segment
- ☐ Don't align segments

System DLL directory: C:\WINNT

Here we can adjust, over which acts for a kind of file it, which processor is to be put at the basis etc. we to be able however all attitudes to take over (the system DLL directory is not naturally correct, if one works under Win9x - however understands itself automatically) and clicks on “OK ONE”. IDA begins now with the analysis and the these assembling of the EXE, which - depending upon size and complexity of the program - can take one while (I already heard of people, which let IDA this-assemble over night, which reminds me much of earlier Raytracing sessions:-). Before IDA is finished, one can already look around in the SOURCE code, but I recommend to wait until IDA in the lower status window indicates the message “The initially to autoanalysis is finished”, since only at this time the code completely analyzed and all references are dissolved. We should find thus after short time a similar picture: IDA is high-grade context depending, i.e. only the features are possible, which make sense for this context. But everything in sequence! In W32DASM we would look for References now into the stringer for our stringer, but where are in IDA the stringer References? The answer is, it gives no explicit stringer References! The Newbie will say now the crisis wars and that IDA is not simply bad etc. a fear, it gives sowas something similar wIE the stringer References, i.e. the “Names”.


IDA - CoSH Crackme #2.exe
File Edit Jump Search View Options Windows Help
IDA View-A

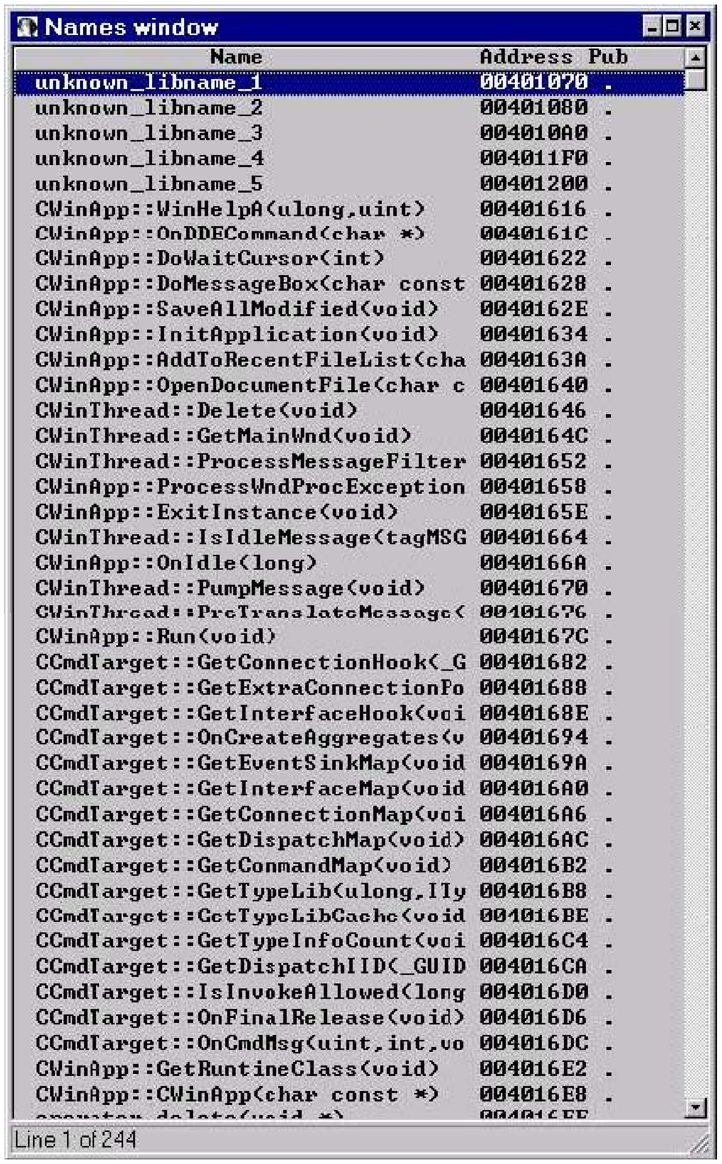
```
00401A32 ; [00000018 BYTES: COLLAPSED FUNCTION _WinMain@16. PRESS KEYPAD "+" TO EXPAND]
00401A4A ; [00000029 BYTES: COLLAPSED FUNCTION AfxInitialize(int,ulong). PRESS KEYPAD "+" TO EXPAND
00401A73 ; ---------------------------------------------------------------------------
00401A73                 jmp     $+5
00401A78                 push    600h
00401A7D                 push    0
00401A7F                 call    ?AfxInitialize@@YGHHK@Z ; AfxInitialize(int,ulong)
00401A84                 mov     byte_403158, al
00401A89                 retn
00401A8A ; [00000006 BYTES: COLLAPSED FUNCTION AfxWinMain(HINSTANCE__ *,HINSTANCE__ *,char *,int).
00401A90
00401A90 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
00401A90
00401A90
00401A90 sub_401A90      proc near               ; DATA XREF: .rdata:004024F8↓o
00401A90                 lea     ecx, [ebp-0F8h]
00401A96                 jmp     sub_401170
00401A96 sub_401A90      endp
00401A96
00401A9B
00401A9B ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
00401A9B
00401A9B
00401A9B sub_401A9B      proc near               ; DATA XREF: .rdata:004024F8↓o
00401A9B                 lea     ecx, [ebp-0F8h]
00401AA1                 jmp     j_??1CDialog@@UAE@XZ ; CDialog::~CDialog(void)
00401AA1 sub_401A9B      endp
00401AA1
00401AA6
00401AA6 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
00401AA6
00401AA6
00401AA6 sub_401AA6      proc near               ; DATA XREF: .rdata:004024F8↓o
00401AA6                 lea     ecx, [ebp-98h]
00401AAC                 jmp     j_??1CEdit@@UAE@XZ ; CEdit::~CEdit(void)
00401AAC sub_401AA6      endp
00401AAC
00401AB1
00401AB1 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
00401AB1
```

```
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: VC v2.0/4.x/5.0 runtime
Using FLIRT signature: MFC 3.1/4.0/4.2 32bit
The initial autoanalysis is finished.
```

```
AU: idle   Up   Disk: 6GB   00001A32   00401A32:
```

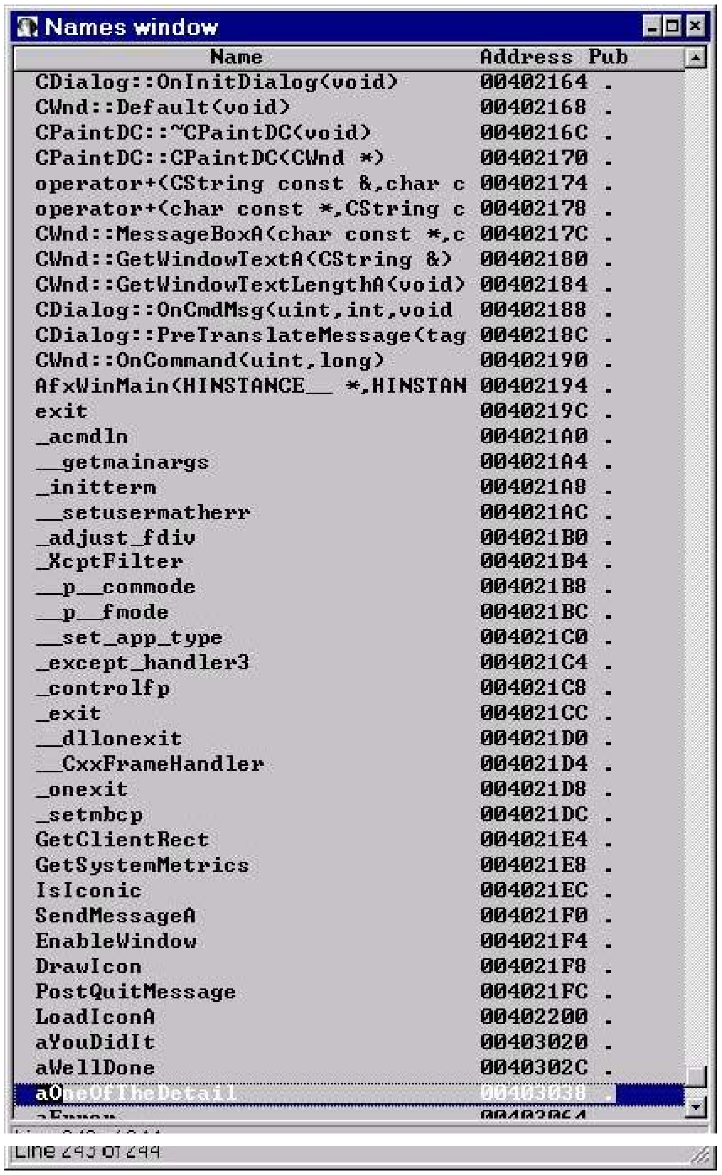
One clicks simply times in the symbol border up

and/or select in the menu “View” the point “Names”. It should open the following window:



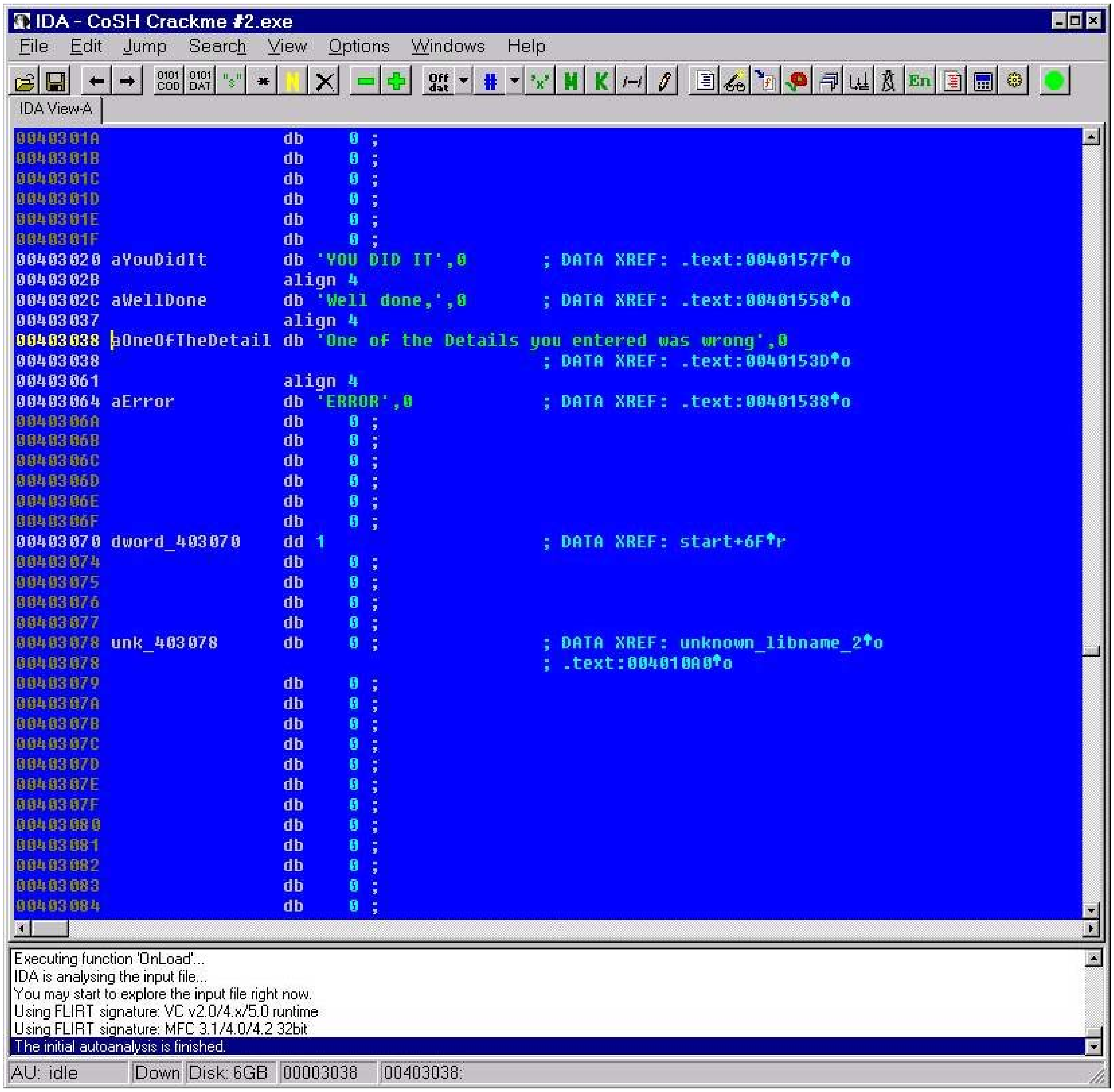
```
                 Name                   Address  Pub
unknown_libname_1                       00401070 .
unknown_libname_2                       00401080 .
unknown_libname_3                       004010A0 .
unknown_libname_4                       004011F0 .
unknown_libname_5                       00401200 .
CWinApp::WinHelpA(ulong,uint)           00401616 .
CWinApp::OnDDECommand(char *)           0040161C .
CWinApp::DoWaitCursor(int)              00401622 .
CWinApp::DoMessageBox(char const        00401628 .
CWinApp::SaveAllModified(void)          0040162E .
CWinApp::InitApplication(void)          00401634 .
CWinApp::AddToRecentFileList(cha        0040163A .
CWinApp::OpenDocumentFile(char c        00401640 .
CWinThread::Delete(void)                00401646 .
CWinThread::GetMainWnd(void)            0040164C .
CWinThread::ProcessMessageFilter        00401652 .
CWinApp::ProcessWndProcException        00401658 .
CWinApp::ExitInstance(void)             0040165E .
CWinThread::IsIdleMessage(tagMSG        00401664 .
CWinApp::OnIdle(long)                   0040166A .
CWinThread::PumpMessage(void)           00401670 .
CWinThread::PreTranslateMessage(         00401676 .
CWinApp::Run(void)                      0040167C .
CCmdTarget::GetConnectionHook(_G        00401682 .
CCmdTarget::GetExtraConnectionPo        00401688 .
CCmdTarget::GetInterfaceHook(voi        0040168E .
CCmdTarget::OnCreateAggregates(v        00401694 .
CCmdTarget::GetEventSinkMap(void        0040169A .
CCmdTarget::GetInterfaceMap(void        004016A0 .
CCmdTarget::GetConnectionMap(voi        004016A6 .
CCmdTarget::GetDispatchMap(void)        004016AC .
CCmdTarget::GetConmandMap(void)         004016B2 .
CCmdTarget::GetTypeLib(ulong,ITy        004016B8 .
CCmdTarget::GetTypeLibCache(void        004016BE .
CCmdTarget::GetTypeInfoCount(voi        004016C4 .
CCmdTarget::GetDispatchIID(_GUID        004016CA .
CCmdTarget::IsInvokeAllowed(long        004016D0 .
CCmdTarget::OnFinalRelease(void)        004016D6 .
CCmdTarget::OnCmdMsg(uint,int,vo        004016DC .
CWinApp::GetRuntineClass(void)          004016E2 .
CWinApp::CWinApp(char const *)          004016E8 .
Line 1 of 244
```

In this window all references - not only on stringers but also on code etc. - are contained. Stringers default-moderately a small “A” is placed in front. In order our stringer “One OF the details you duck-talk which wrong” to search, change we into the Names window and tap “aO” (there IDA not case sensitive is here, “ao” would also function). We should see now the following:



```
Names window
                 Name                      Address  Pub
CDialog::OnInitDialog(void)                00402164 .
CWnd::Default(void)                        00402168 .
CPaintDC::~CPaintDC(void)                  0040216C .
CPaintDC::CPaintDC(CWnd *)                 00402170 .
operator+(CString const &,char c           00402174 .
operator+(char const *,CString c           00402178 .
CWnd::MessageBoxA(char const *,c           0040217C .
CWnd::GetWindowTextA(CString &)            00402180 .
CWnd::GetWindowTextLengthA(void)           00402184 .
CDialog::OnCmdMsg(uint,int,void            00402188 .
CDialog::PreTranslateMessage(tag           0040218C .
CWnd::OnCommand(uint,long)                 00402190 .
AfxWinMain(HINSTANCE__ *,HINSTAN           00402194 .
exit                                       0040219C .
_acmdln                                    004021A0 .
__getmainargs                              004021A4 .
_initterm                                  004021A8 .
__setusermatherr                           004021AC .
_adjust_fdiv                               004021B0 .
_XcptFilter                                004021B4 .
__p__commode                               004021B8 .
__p__fmode                                 004021BC .
__set_app_type                             004021C0 .
_except_handler3                           004021C4 .
_controlfp                                 004021C8 .
_exit                                      004021CC .
__dllonexit                                004021D0 .
__CxxFrameHandler                          004021D4 .
_onexit                                    004021D8 .
_setmbcp                                   004021DC .
GetClientRect                              004021E4 .
GetSystemMetrics                           004021E8 .
IsIconic                                   004021EC .
SendMessageA                               004021F0 .
EnableWindow                               004021F4 .
DrawIcon                                   004021F8 .
PostQuitMessage                            004021FC .
LoadIconA                                  00402200 .
aYouDidIt                                  00403020 .
aWellDone                                  0040302C .
aOneOfTheDetail                            00403038 .
Line 243 of 244
```

Aha! We see the text “aOneOfTheDetail”. That is our stringer reference! And which for a coincidence: over it are direct equal still two interesting stringers: and “aWellDone” “aYouDidIt”. We want to now see, where this stringer reference is used, therefore make we one let us doubleclick on ours “aOneOfTheDetail” - stringer.

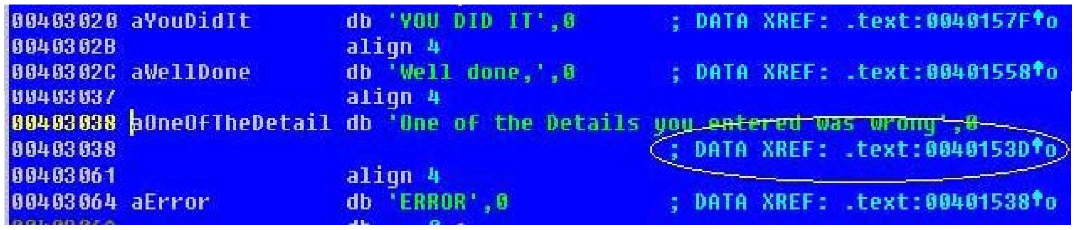

IDA - CoSH Crackme #2.exe
File Edit Jump Search View Options Windows Help
IDA View-A

```
00403 01A                     db     0 ;
00403 01B                     db     0 ;
00403 01C                     db     0 ;
00403 01D                     db     0 ;
00403 01E                     db     0 ;
00403 01F                     db     0 ;
00403 020 aYouDidIt           db 'YOU DID IT',0       ; DATA XREF: .text:0040157F↑o
00403 02B                     align 4
00403 02C aWellDone           db 'Well done,',0       ; DATA XREF: .text:00401558↑o
00403 037                     align 4
00403 038 aOneOfTheDetail db 'One of the Details you entered was wrong',0
00403 038                                             ; DATA XREF: .text:0040153D↑o
00403 061                     align 4
00403 064 aError              db 'ERROR',0            ; DATA XREF: .text:00401538↑o
00403 06A                     db     0 ;
00403 06B                     db     0 ;
00403 06C                     db     0 ;
00403 06D                     db     0 ;
00403 06E                     db     0 ;
00403 06F                     db     0 ;
00403 070 dword_403070        dd 1                    ; DATA XREF: start+6F↑r
00403 074                     db     0 ;
00403 075                     db     0 ;
00403 076                     db     0 ;
00403 077                     db     0 ;
00403 078 unk_403078          db     0 ;              ; DATA XREF: unknown_libname_2↑o
00403 078                                             ; .text:004010A0↑o
00403 079                     db     0 ;
00403 07A                     db     0 ;
00403 07B                     db     0 ;
00403 07C                     db     0 ;
00403 07D                     db     0 ;
00403 07E                     db     0 ;
00403 07F                     db     0 ;
00403 080                     db     0 ;
00403 081                     db     0 ;
00403 082                     db     0 ;
00403 083                     db     0 ;
00403 084                     db     0 ;
```

```
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: VC v2.0/4.x/5.0 runtime
Using FLIRT signature: MFC 3.1/4.0/4.2 32bit
The initial autoanalysis is finished.
```

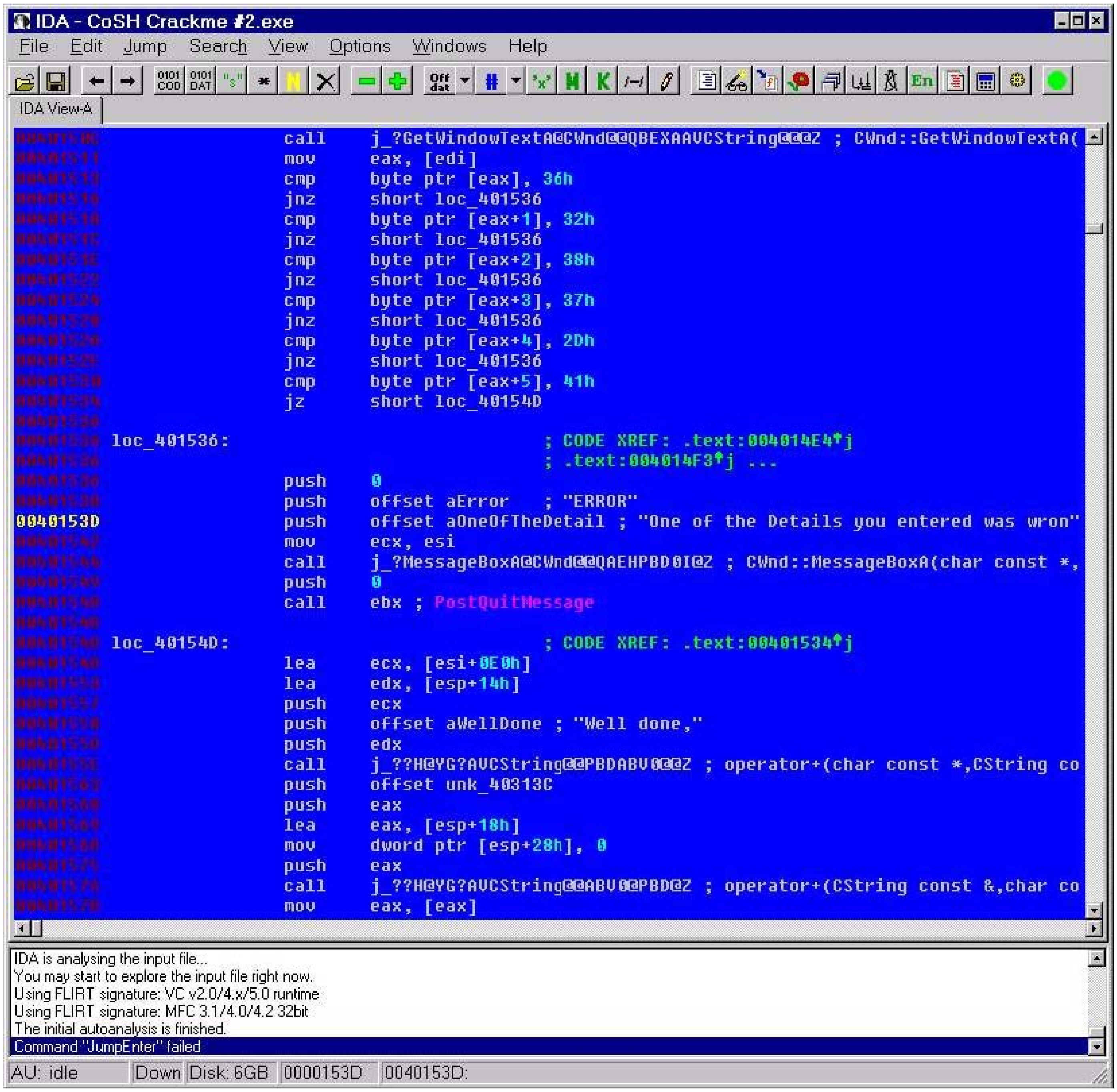
```
AU: idle    Down  Disk: 6GB  00003038   00403038:
```

But that is not what we expected. We are now correct-prove, where the stringer is defined and defined. IDA makes it however easy for us, by it us - completely in W32DASM-Manier - which indicates place (n), where the variable “aOneOfTheDetail” (their contents of the stringer “One OF the details you duck-talk which wrong” is) is used (a so-called cross-reference, shortened “XREF”):



```
00403020 aYouDidIt        db 'YOU DID IT',0       ; DATA XREF: .text:0040157F↑o
0040302B                  align 4
0040302C aWellDone        db 'Well done,',0       ; DATA XREF: .text:00401558↑o
00403037                  align 4
00403038 aOneOfTheDetail db 'One of the Details you entered was wrong',0
00403038                                          ; DATA XREF: .text:0040153D↑o
00403061                  align 4
00403064 aError           db 'ERROR',0            ; DATA XREF: .text:00401538↑o
```

As we can see, the variable is used in only one place: in the section ".text" at address 0040153D, which above the current position is, what the arrow upward suggest. The small "o" indicates the use. Here "o" stands p" for "offset", "j" for "jump" and for "for "procedure". We would like to now see, where the variable is used, therefore make we one let us doubleclick on the XREF" .text:0040153D ":


IDA - CoSH Crackme #2.exe
File Edit Jump Search View Options Windows Help
IDA View-A

```
0040150C                 call    j_?GetWindowTextA@CWnd@@QBEXAAVCString@@@Z ; CWnd::GetWindowTextA(
00401511                 mov     eax, [edi]
00401513                 cmp     byte ptr [eax], 36h
00401516                 jnz     short loc_401536
00401518                 cmp     byte ptr [eax+1], 32h
0040151C                 jnz     short loc_401536
0040151E                 cmp     byte ptr [eax+2], 38h
00401522                 jnz     short loc_401536
00401524                 cmp     byte ptr [eax+3], 37h
00401528                 jnz     short loc_401536
0040152A                 cmp     byte ptr [eax+4], 2Dh
0040152E                 jnz     short loc_401536
00401530                 cmp     byte ptr [eax+5], 41h
00401534                 jz      short loc_40154D
00401536
00401536 loc_401536:                             ; CODE XREF: .text:004014E4↑j
00401536                                         ; .text:004014F3↑j ...
00401536                 push    0
00401538                 push    offset aError   ; "ERROR"
0040153D                 push    offset aOneOfTheDetail ; "One of the Details you entered was wron"
00401542                 mov     ecx, esi
00401544                 call    j_?MessageBoxA@CWnd@@QAEHPBD0I@Z ; CWnd::MessageBoxA(char const *,
00401549                 push    0
0040154B                 call    ebx ; PostQuitMessage
0040154D
0040154D loc_40154D:                             ; CODE XREF: .text:00401534↑j
0040154D                 lea     ecx, [esi+0E0h]
00401553                 lea     edx, [esp+14h]
00401557                 push    ecx
00401558                 push    offset aWellDone ; "Well done,"
0040155D                 push    edx
0040155E                 call    j_??H@YG?AVCString@@PBDABV0@@Z ; operator+(char const *,CString co
00401563                 push    offset unk_40313C
00401568                 push    eax
00401569                 lea     eax, [esp+18h]
0040156D                 mov     dword ptr [esp+28h], 0
00401575                 push    eax
00401576                 call    j_??H@YG?AVCString@@ABV0@PBD@Z ; operator+(CString const &,char co
0040157B                 mov     eax, [eax]
```

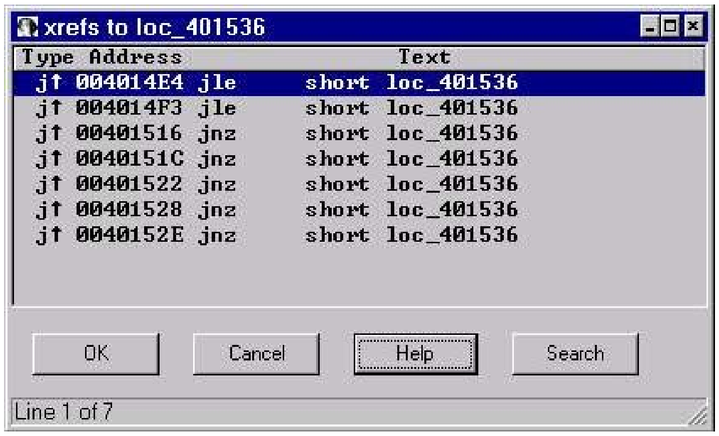
```
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: VC v2.0/4.x/5.0 runtime
Using FLIRT signature: MFC 3.1/4.0/4.2 32bit
The initial autoanalysis is finished.
Command "JumpEnter" failed
```

```
AU: idle    Down Disk: 6GB  0000153D  0040153D:
```

IDA is similarly w ← IE a Browser developed. If one liked to jump thus back for last position, one clicks simply on that

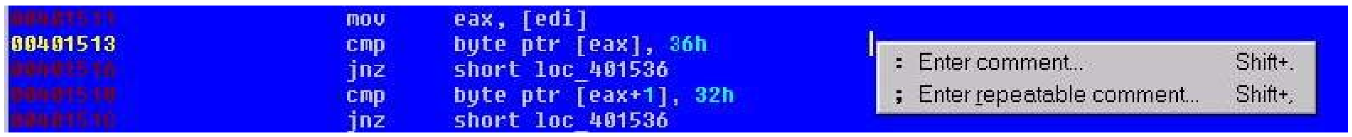
Button in the symbol border. Forward it goes accordingly with the other Button. Here we see now finally the SOURCE code. We notice that the range, in which we are straight ("loc_401536" marks by the name) in several places in the SOURCE code by a Jump are started (which is suggested by the three points at the end of the XREF list). For exercise we can doubleclick times on the first XREF, in order to arrive at the place mentioned.

Then at the best again Button above mentioned with that back jump, in order to arrive at the last position. As already said, IDA is very context depending. In order to see now all XREFs for this place, we click with the right mouse button on the XREFs and select "Jump ton of CROSS reference…" out, whereupon the following list opens, in which one sees not only the address, of where out one jumps, but also around which instruction it acts (jle, jnz,…):



```
xrefs to loc_401536
Type Address           Text
 j↑ 004014E4 jle       short loc_401536
 j↑ 004014F3 jle       short loc_401536
 j↑ 00401516 jnz       short loc_401536
 j↑ 0040151C jnz       short loc_401536
 j↑ 00401522 jnz       short loc_401536
 j↑ 00401528 jnz       short loc_401536
 j↑ 0040152E jnz       short loc_401536
```

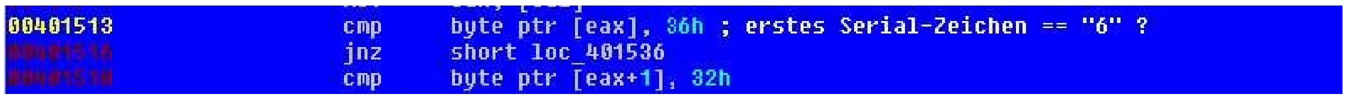
So far sogut. We learn now something over the comment functions of IDA. As we can see in the above picture, the API function "GetWindowTextA" is called, which serves for it, to store contents of a text box to an address. In this case an address is in an address on contents of the text box after call of the API function in EDI (thus a pointer on a pointer to speak over in the C-Slang). The last pointer is copied now from EAX, then hard coded comparisons follow. Since it concerns not primarily the Cracken of the Crackmes here, betrayal I (if their not anyway already noticed it) that a hard coded Serial is used here, whose comparison here we discovered evenly. The entered name and the entered Serial must be at least 6-digit, since otherwise the error message comes, which we saw above. Subsequently, each indication of the entered Serial with the corresponding indication of the hard coded Serial is compared and as soon as a difference is determined, to the error message jumped. The hard coded Serial is, as one from an ASCII table to see can, "6287-A" (36h = "6", 32h = "2" etc.). In order to illustrate it us however better, we commentate the SOURCE code a little. We click with the right mouse button somewhat right apart from the comparison with 36h and select "Enter COMMENT…":



```
00401511                mov     eax, [edi]
00401513                cmp     byte ptr [eax], 36h
00401516                jnz     short loc_401536
00401518                cmp     byte ptr [eax+1], 32h
0040151C                jnz     short loc_401536
```

: Enter comment... Shift+;
; Enter repeatable comment... Shift+;

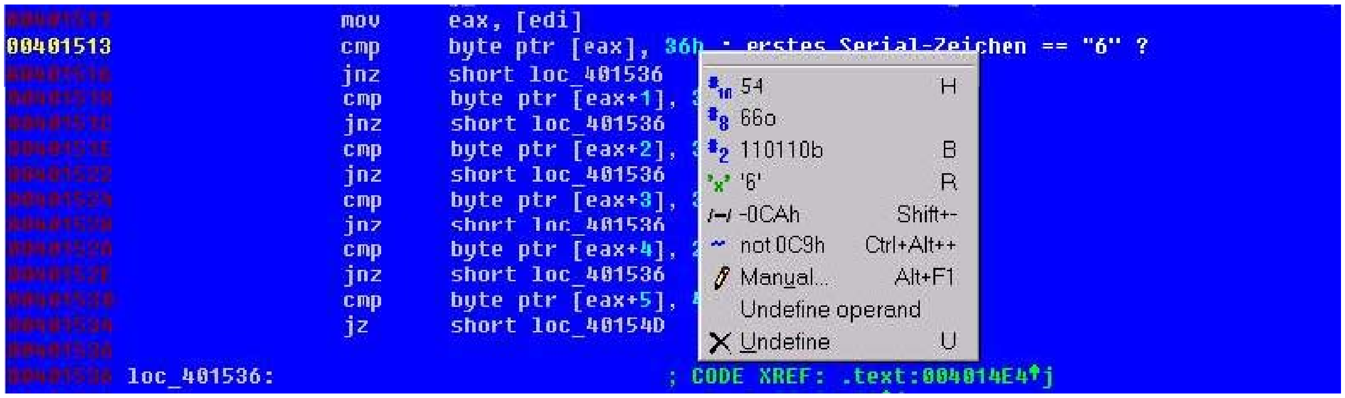
In the appearing dialog we enter as meaningful a comment as possible and click on "OK ONE". The entered comment appears now beside the line.



```
00401513                cmp     byte ptr [eax], 36h ; erstes Serial-Zeichen == "6" ?
00401516                jnz     short loc_401536
00401518                cmp     byte ptr [eax+1], 32h
```

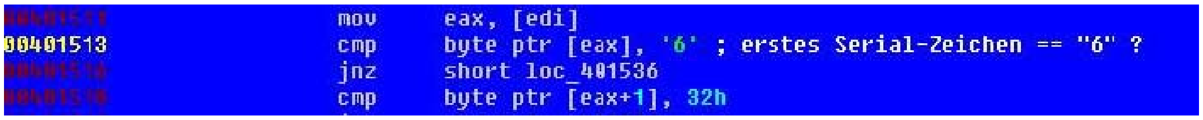
IDA offers the possibility of indicating data differently. Here according to standard the “6” as “36h” one indicates. Since we know however now the fact that it here actually concerns the ASCII value of the indication “6” would be more beautiful it, if this indication would be represented. For this we click with the right mouse button on “36h” and wählen from the appearing Popup menu

the line out, in which to “is located 6”. Alternatively we can set and - for Button from the symbol border or “r” press the cursor on “36h”. It causes that the value is interpreted and represented as ASCII character. One can select among other things also a decimal, Octal or binary notation.



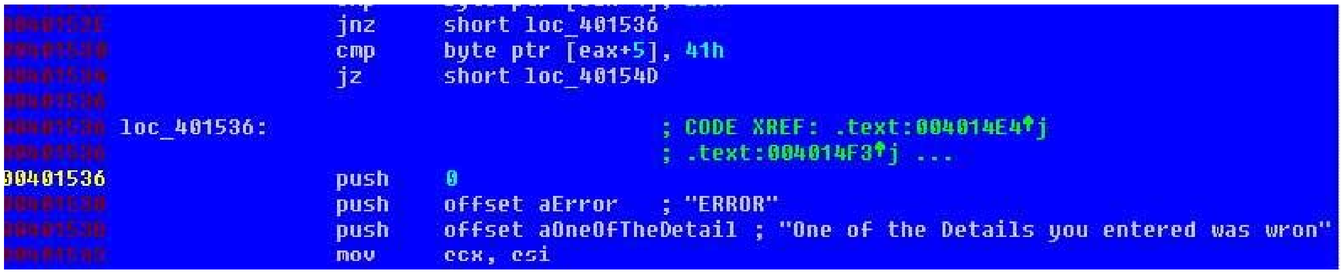
```
[illegible]                 mov     eax, [edi]
00401513                    cmp     byte ptr [eax], 36h ; erstes Serial-Zeichen == "6" ?
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+1], 3
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+2], 3
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+3], 3
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+4], 2
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+5], 4
[illegible]                 jz      short loc_40154D
[illegible]
[illegible] loc_401536:                                 ; CODE XREF: .text:004014E4↑j
```

The result of this action looks in such a way:



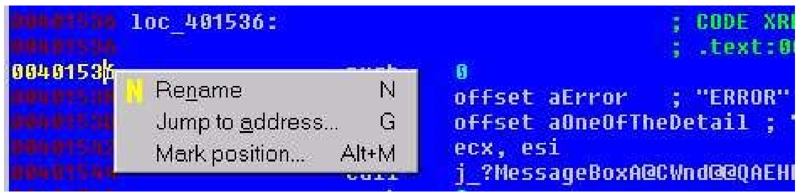
```
[illegible]                 mov     eax, [edi]
00401513                    cmp     byte ptr [eax], '6' ; erstes Serial-Zeichen == "6" ?
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+1], 32h
```

We come now to one weitern, very nice feature of IDA: renaming references. As we can see is jumped, with a missed comparison to the place “loc_401536”, which indicates the message in the following that an input was wrong (the so-called “beggar off” - or also “bath more cracker” - message). Now the designation “loc_401536” is not very meaningful straight. It would be many more beautiful, if something would stand there as “more bad_cracker”. Clearly one could insert a comment, but that would be only locally visible for this line. Which we needed, would be a possibility of renaming all references of “loc_401536”. For this we make one let us doubleclick on “loc_401536” (all the same on which reference), on which we should appear here:



```
[illegible]                 jnz     short loc_401536
[illegible]                 cmp     byte ptr [eax+5], 41h
[illegible]                 jz      short loc_40154D
[illegible]
[illegible] loc_401536:                                ; CODE XREF: .text:004014E4↑j
[illegible]                                            ; .text:004014F3↑j ...
00401536                    push    0
[illegible]                 push    offset aError   ; "ERROR"
[illegible]                 push    offset aOneOfTheDetail ; "One of the Details you entered was wron"
[illegible]                 mov     ecx, esi
```

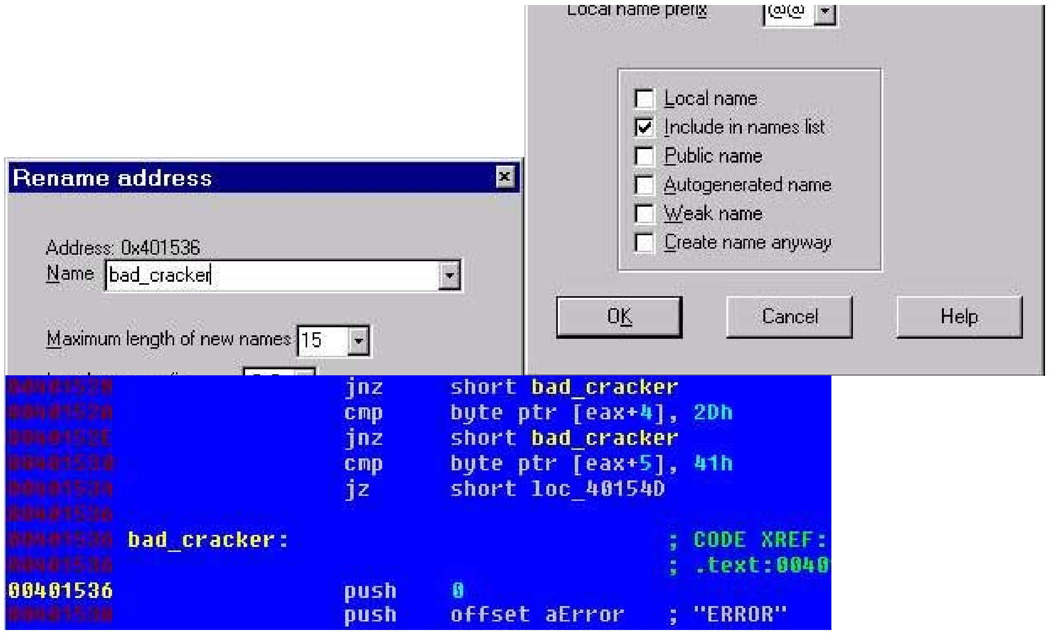
We click now with the right mouse button in the address “00401536” so that the following Popup menu appear:



```
00401536 loc_401536:                            ; CODE XR
00401536                                        ; .text:0
00401536            push    0
00401538            push    offset aError       ; "ERROR"
0040153D            push    offset aOneOfTheDetail ; '
00401542            mov     ecx, esi
00401544            call    j_?MessageBoxA@CWnd@@QAEH
```

Here we select “Rename” from which us to the following D ialog lead (alternatively we would have that also simply with the cursor in the address “00401536” to go and “n” press or

- Button from the symbol border to click can): We give a meaningful name like e.g. “more bad_cracker” and we press on “OK ONE” immediately see that all references on “loc_401536” were changed in our new names:



Rename address

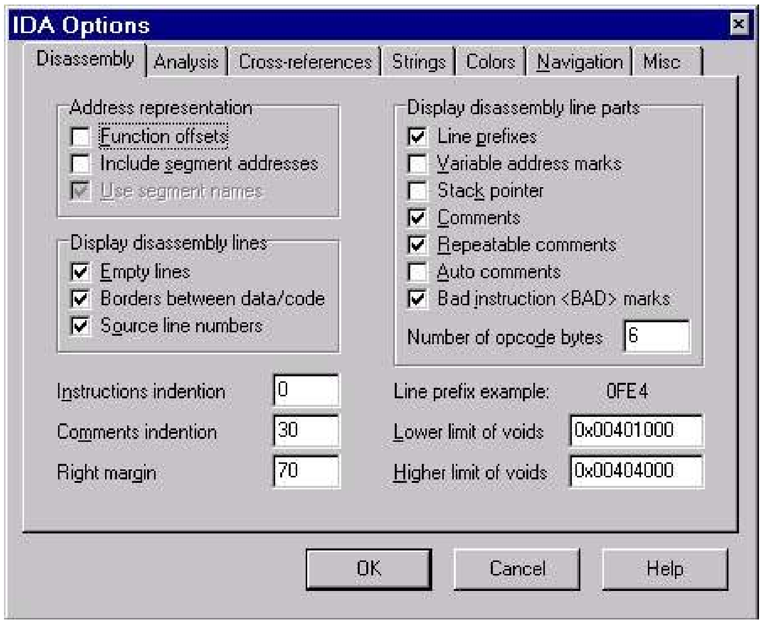
Address: 0x401536
Name: bad_cracker
Maximum length of new names: 15
Local name prefix: @@

- [ ] Local name
- [x] Include in names list
- [ ] Public name
- [ ] Autogenerated name
- [ ] Weak name
- [ ] Create name anyway

```
00401528            jnz     short bad_cracker
0040152A            cmp     byte ptr [eax+4], 2Dh
0040152E            jnz     short bad_cracker
00401530            cmp     byte ptr [eax+5], 41h
00401534            jz      short loc_40154D
00401536
00401536 bad_cracker:                           ; CODE XREF:
00401536                                        ; .text:0040
00401536            push    0
00401538            push    offset aError       ; "ERROR"
```

For exercise the same with according to other name should (e.g.) with the reference “loc_40154D” to be made “more good_cracker”.

As short module we treat now, as one configures to IDA. We call” the point “from the menu “option general…” up, on which the following dialog appears:


IDA Options
Disassembly | Analysis | Cross-references | Strings | Colors | Navigation | Misc
OK
Cancel
Help

Address representation
- [ ] Function offsets
- [ ] Include segment addresses
- [✓] Use segment names

Display disassembly lines
- [✓] Empty lines
- [✓] Borders between data/code
- [✓] Source line numbers

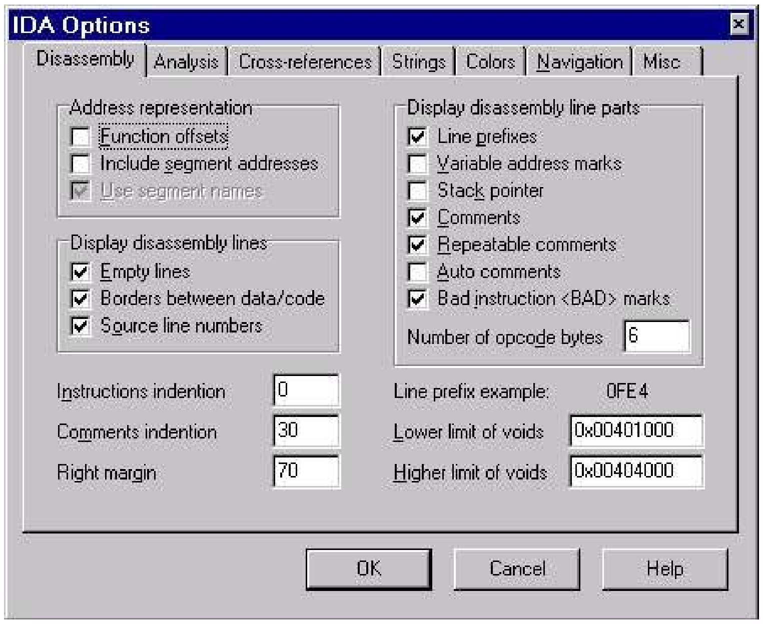
Instructions indention: 0
Comments indention: 30
Right margin: 70

Display disassembly line parts
- [✓] Line prefixes
- [ ] Variable address marks
- [ ] Stack pointer
- [✓] Comments
- [✓] Repeatable comments
- [ ] Auto comments
- [✓] Bad instruction <BAD> marks

Number of opcode bytes: 6

Line prefix example: 0FE4
Lower limit of voids: 0x00401000
Higher limit of voids: 0x00404000

Who would like to see e.g. gladly the Hexopcodes, OF registers opcode bytes” the number of bytes with “NUMBER, which are to be indicated (6 is here a good value). That is particularly meaningful to the Patchen, since one can see those equal to patchenden bytes. With “Instructions indention” one should reduce then however the original value of 16 to 0, so that the announcement does not look so shifted. For assembler beginners the attitude “car COMMENT” is recommendable, since then IDA automatically attaches a short description behind some assembler instructions as comment, which describes, what the respective instruction effectuation.


IDA Options
Disassembly | Analysis | Cross-references | Strings | Colors | Navigation | Misc
OK
Cancel
Help

Address representation
- [ ] Function offsets
- [ ] Include segment addresses
- [✓] Use segment names

Display disassembly lines
- [✓] Empty lines
- [✓] Borders between data/code
- [✓] Source line numbers

Instructions indention: 0
Comments indention: 30
Right margin: 70

Display disassembly line parts
- [✓] Line prefixes
- [ ] Variable address marks
- [ ] Stack pointer
- [✓] Comments
- [✓] Repeatable comments
- [ ] Auto comments
- [✓] Bad instruction <BAD> marks

Number of opcode bytes: 6

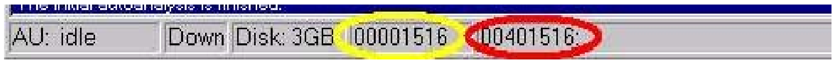
Line prefix example: 0FE4
Lower limit of voids: 0x00401000
Higher limit of voids: 0x00404000

Who would like to have indicated gladly more than only two Crossreferences, this can stop NUMBERs under “CROSS references” at the point “OF displayed xrefs”.

If one liked the fact that IDA generated longer variable names must edit directly the file "ida.cfg". The entry for this reads:

MAX_NAMES_LENGTH = 30//maximum length OF of new names (you May specify VALUEs UP ton of 511)

Who would like to know the file offset of the current instruction (to the Patchen for example), looks into the info. line completely down at the IDA window:

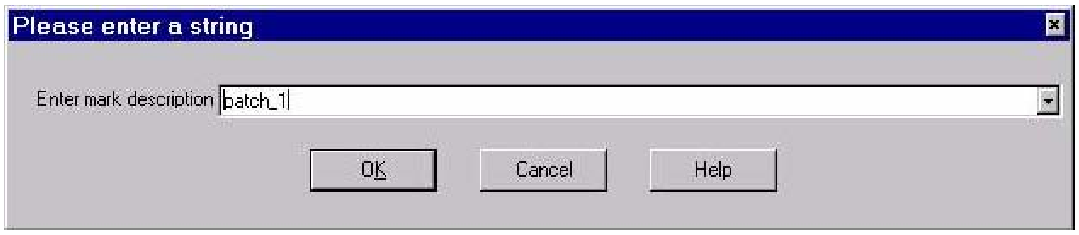


AU: idle | Down | Disk: 3GB | 00001516 | 00401516:

The yellow Kringel shows the file offset and the red Kringel the RVA (relative virtual ADDRESS).

I mentioned further above that one with that

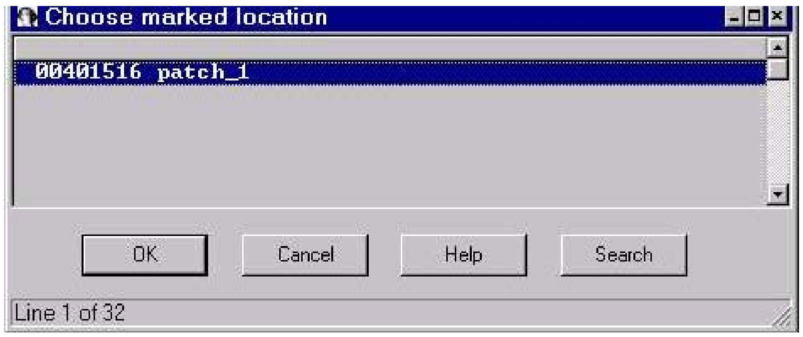
Button, similarly as with a Webbrowser, for position visited last to jump can. One can fritter away however with an extensive program easily. Which we needed, a feature would be similar to the Bookmarks. Naturally IDA offers something comparable. In order to set a so-called mark, one selects the entry "Mark of position from the menu "Jump"…", on which the following dialog appears:



Please enter a string

Enter mark description: patch_1

Here one must indicate a name for those mark which can be set. I entered patch_1" as example ", in order to mark that is to be patchen here. If one would like to jump now to such a mark, one selects the entry "Jump ton from the menu "Jump" marked position…". In the appearing dialog all marks defined so far stand. From these one selects the desired mark and makes one doubleclicks on it.



Choose marked location

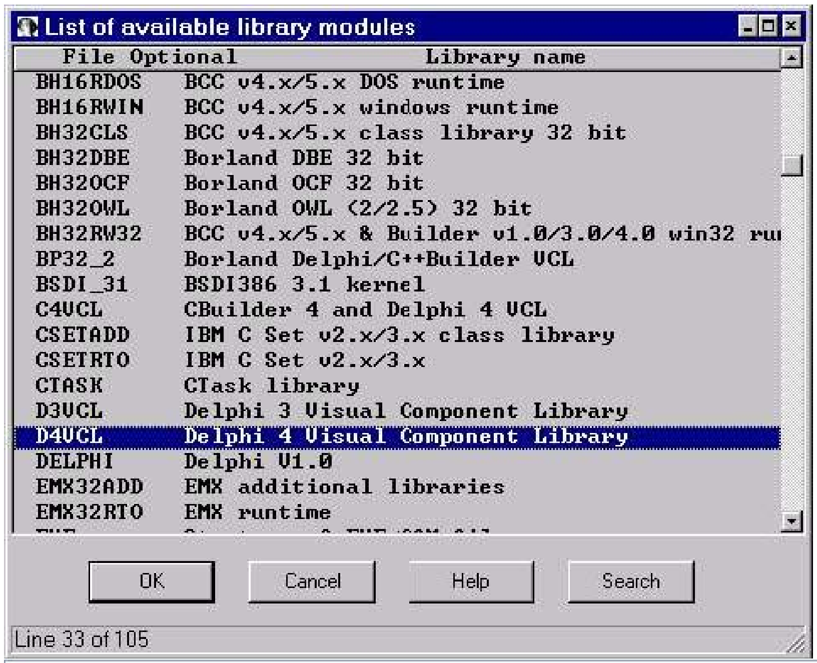
```
00401516 patch_1
```

Line 1 of 32

It can occur that IDA does not recognize the language, in which the program which can be analyzed was written, and thus not uses the available flirtation signatures. If that is the case, one can merge the signatures manually. Accepted, that program which can be analyzed was written in Delphi 4. We call, after IDA isassembliert finished D, from the menu “View”

the point “Signatures” on (alternatively with that

Button). A window appears, in which all used flirtation signatures are normally located. We can use a new signature, by clicking with the right mouse button into the window and selecting “Apply new signature”. IDA shows thereupon in a new window all available signatures on. We select the signature “Delphi 4 Visual Component LIBRARY” in this example then.


List of available library modules

```
  File Optional            Library name
BH16RDOS   BCC v4.x/5.x DOS runtime
BH16RWIN   BCC v4.x/5.x windows runtime
BH32CLS    BCC v4.x/5.x class library 32 bit
BH32DBE    Borland DBE 32 bit
BH32OCF    Borland OCF 32 bit
BH32OWL    Borland OWL (2/2.5) 32 bit
BH32RW32   BCC v4.x/5.x & Builder v1.0/3.0/4.0 win32 ru
BP32_2     Borland Delphi/C++Builder VCL
BSDI_31    BSDI386 3.1 kernel
C4VCL      CBuilder 4 and Delphi 4 VCL
CSETADD    IBM C Set v2.x/3.x class library
CSETRTO    IBM C Set v2.x/3.x
CTASK      CTask library
D3VCL      Delphi 3 Visual Component Library
D4VCL      Delphi 4 Visual Component Library
DELPHI     Delphi V1.0
EMX32ADD   EMX additional libraries
EMX32RTO   EMX runtime
```

OK Cancel Help Search

Line 33 of 105

6. Plug-Ins

Who knows W32DASM and whose stringer References (or also import/export) misses with IDA painfully, which can are helped. The Cracker “Amante4” of the group of Immortal Descendants has Plug into for IDA in IDC, which Script language of IDA, written, which offers exactly the features mentioned. The Plug in can on the above mentioned homepage under the column “The DATA cousin/release” be down-loaded. The installation is relative easy and is well described in README.NFO in the ZIP file, likewise the operation. I would like to deal here not further with these Plug in, since her A) am well described and b) necessarily am not necessary, in order with IDA meaningfully to work to be able (which did not mean that I want to diminish the achievement of Amante4 in any way).

Current IDA versions offer a beautiful feature named “AUTOMATIC parameter Recognition”, which causes the fact that with API functions the parameters are designated (instead of “push eax” stands as comment evenly still behind it, which parameter it concerns, e.g. “hWnd”). Unfortunately the version 4.04 of IDA, available in the net, does not have this feature. The well-known Cracker +Spath wrote therefore a IDC script named W32param, with whose assistance a similar function is realized. The script under http://frogsprint.cjb.net is available. In the associated file API32.H, in which the definitions of the API functions stand, as for instance a GetDlgItemTextA or a GetDlgItemInt is missing to few important API functions. One would have to insert these independently. Gotten one does the appropriate definitions from the hopefully available API description or from header files of a development package. In the menu “file” one finally calls the point “IDC file…” up and the script selects W32param.idc. The file API32.H must be thereby in the IDA listing, not in the IDC sublist!

7. Conclusion

I think that the fundamental operation is hereby discussed by IDA pro. Naturally there is still enormous amount at features and functionalities, which remained unmentioned here, but finally concerns it “also only” a Newbie Tutorial! I would like to say some more to the speed of IDA opposite pro of W32DASM. It is undisputed that W32DASM, particularly with large EXE files, is faster finished around lengths than IDA. However this speed projection/lead by missing features, like e.g. the possibility of renaming references etc., bought. IDA pro develops a complete data base, in which all references etc. are with the analysis, whereby the features of IDA become only possible at all. One may not disparage of the two tools Gods or, because everyone has its authorization. If one would like to this-assemble times evenly fast a small EXE file, in order to cracken and/or examine the program, W32DASM is perfectly sufficient. If one would like to analyze a larger EXE file or to operate “correct” Reverse engineering (under what I understand, to reconstruct the program completely and not only those few lines code, which constitute the protection), IDA pro with its features can shine. One partly does not come around around IDA pro anyway, since it in the meantime some cheat gives, which prevent the these assembling of programs with W32DASM, where IDA pro has however nevertheless success.

Finally I hope that this Tutorial could näherbringen the operation of IDA pro to the Newbie somewhat. Take nevertheless simply a small Crackme or a small EXE file, let it from IDA this-assemble, play then with the different buttons and attitudes and look, what happens. Still if questions should arise or if someone should want constructional criticism or praise, then it may write this gladly me:-).

Me only, a few Greetings (in indefinite order) remains loose-will on: Cosmo Connor, Instructor, Folko, figugegl, Ebeneezer, ArturDent, Peacemaker, kUbUs, exa, fakeraol, subZZero, radeOn and to all Newbiez of this world!

Tutorial of: medardus PAGE arranges: 29. May 2000 of ArturDent