Demo Cracking & Protection Software

Josua M Sinambela, CCNP, CCNA, CEH, CompTIA Security+
RootBrain IT Training & Consulting
www.rootbrain.com

Pembahasan

- Pengertian Cracking software
- Jenis pengamanan Software
- Cara Kerja Cracker
- Cracker Tools
- Demo Cracking Software
- Demo Proteksi Software

Cracking Software

- Memodifikasi Software yang bertujuan untuk menyingkirkan proteksi seperti dari copy/duplikasi aplikasi, serial number, hardware key, pengecekan waktu, trial atau versi demo, pengecekan CD dan iklan iklan pada softawre.
- Melanggar HAKI
- Merugikan Developer dan Produsen software

Jenis Pengamanan Software

- **Serial Number**
  - Vendor/Developer menyediakan serial number yang valid
  - Serial number diberikan kepada user yang membeli software tersebut (melalui CD atau via Email)
  - Proteksi lemah, user dapat menginstall software dengan serial number yang sama di PC yang berbeda.
  - Banyak vendor software tetap menggunakannya
  - Rentan Cracking

Jenis Pengamanan Software

- **Activation Code**
  - Melengkapi proteksi dengan Serial Number
  - Software akan memeriksa spesifikasi hardware (kode HDD, Processor atau Motherboard) dan mengenerate Activation Code
  - Kode Aktivasi harus diaktifkan melalui telp atau web online ke perusahaan vendor software
  - Vendor software akan memberikan serial number khusus kode aktivasi tersebut
  - Muncul problem ketika upgrade/mengganti hardware karena dibutuhkan aktivasi kembali
  - Masih rentan Cracking

Jenis Pengamanan Software

- **Dongle**
  - Berupa hardware khusus yang dipasangkan ke PC (biasanya melalui USB Port) sebagai pengaman software
  - Dongle menyimpan informasi lisensi dalam bentuk hardware yang akan dibaca oleh software
  - Software melakukan otentikasi dan tidak akan bekerja jika dongle tidak terpasang atau tidak memiliki lisensi yang benar
  - Dongle untuk tiap PC yang terinstall aplikasi (kecuali menggunakan terminal services/ThinClient)
  - Relatif aman karena cracking membutuhkan peralatan khusus dan software khusus

Jenis Pengamanan Software

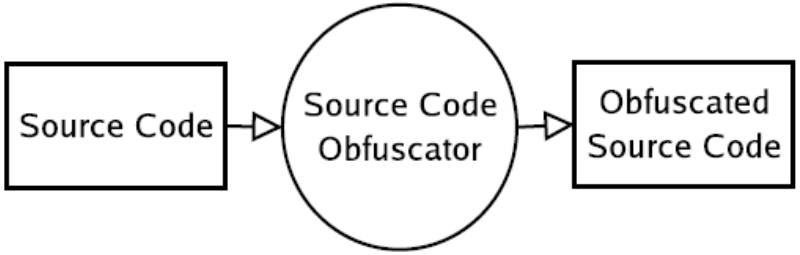
- Demo Version
  - Vendor/Developer membuat dua versi software
  - Satu versi demo yang memiliki fitur terbatas
    - Misal hanya memproses data record yang berukuran kecil
    - Menampilkan pesan-pesan iklan versi Demo.
    - Menampilkan footer dst
  - Kedua versi Full version yang memiliki seluruh fitur yang dibutuhkan
  - Versi Demo di publikasikan secara gratis di internet untuk memikat calon pembeli sehingga tertarik mencoba

Jenis Pengamanan Software

- **Hard Code**
  - **Developer menanamkan informasi dan format langsung pada source code, sehingga informasi atau format tersebut tidak dapat diganti dengan mudah**
    - **Format Laporan, Nama, Logo, Banner, Keterangan Perusahaan Client (pengguna) dst**
  - **Tampilan dan Format laporan bersifat statis dan hanya dapat diubah dari source code**
  - **Teknik ini masih mudah dibongkar.**

Jenis Pengamanan Software

- **Obfuscated Source Code**
  - **Proteksi di pada source code, sehingga tidak mudah dipahami dan di modifikasi oleh orang lain.**
  - **Sering di Implementasi pada aplikasi berbasis Web (PHP, ASP, JSP)**



Original Source Code

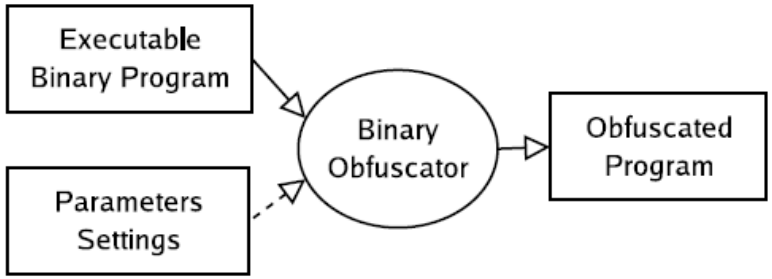
```
for (i=0; i < M.length; i++){
  // Adjust position of clock hands
  var ML=(ns)?document.layers['nsMinutes'+i]:ieMinutes[i].style;
  ML.top=y[i]+HandY+(i*HandHeight)*Math.sin(min)+scrll;
  ML.left=x[i]+HandX+(i*HandWidth)*Math.cos(min);
 }
```

Obfuscated Source Code

```
for (O79=0;O79<l6x.length;O79++ ){var O63=(l70) ? document.layers
["nsM\151\156u\164\145s"+O79]:ieMinutes[O79].style;O63.top=l61[O79]+O76+(O79*O75)*Math
.sin(O51)+l73;O63.left=l75[O79]+l77+(O79*l76)*Math.cos(O51);}
```

Jenis Pengamanan Software

- **Obfuscated Binary Code**
  - **Kode Binary melalui proses Enkripsi dan Packing (pemaketan khusus)**
  - **Contoh ASProtect, Y0da's Cryptor, NFO, and Armadillo.**



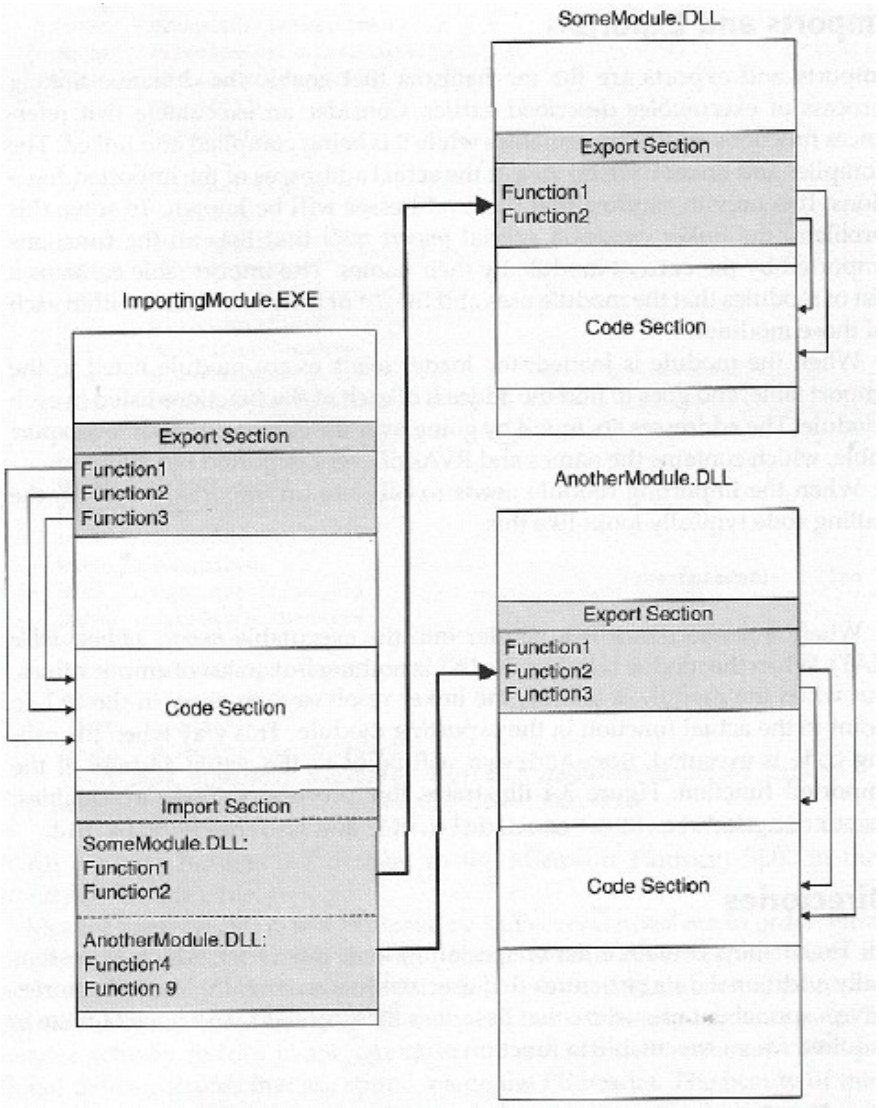
Cara Kerja Cracker

- Cracker melakukan aktifitas cracking menggunakan teknik "Reverse Engineering"
- Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction
  - About opening up a program's "box" and looking inside
  - No screwdrivers needed, but integrates several arts of
    - Code breaking
    - Puzzle solving
    - Programming
    - Logical analysis

The .exe format and DLLs

- Before loading DLLs, addresses of functions in DLL are pointing to dummy addresses in an import table

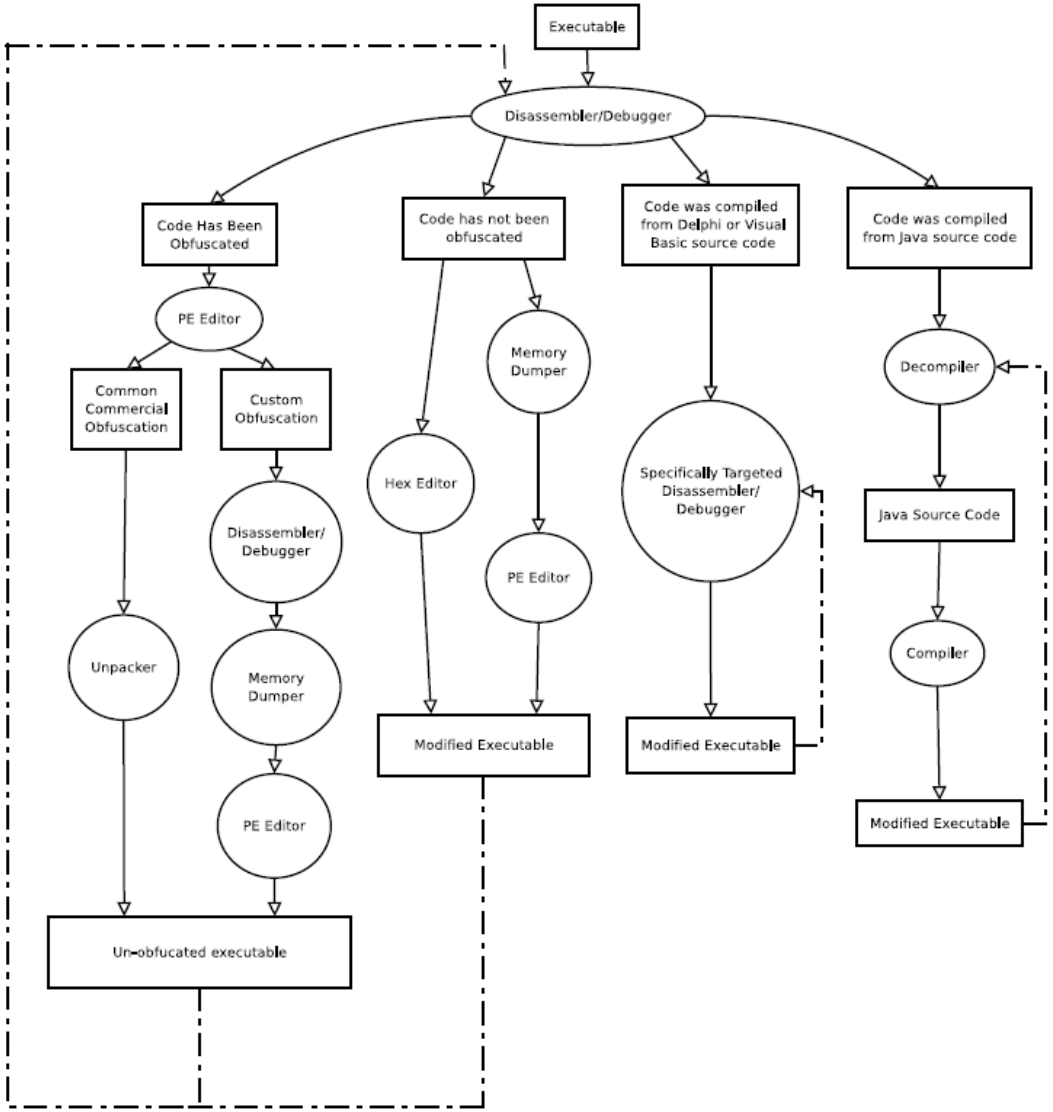
- When process is loaded, the OS loader loads every module listed in the imported table, and resolves the addresses of each of the functions listed in each modules. The addresses are found in the exported table of the module.



Reverse Engineering

- Beberapa tujuan awal reverse engineering:
  - Fun / Challenge
  - Lost Source Code (This is common)
  - Legacy Code (Original Coder Unavailable, No Source, Y2K)
  - Bug Hunting (Again, no source)
  - Virus Analysis

Proses Reverse Engineering

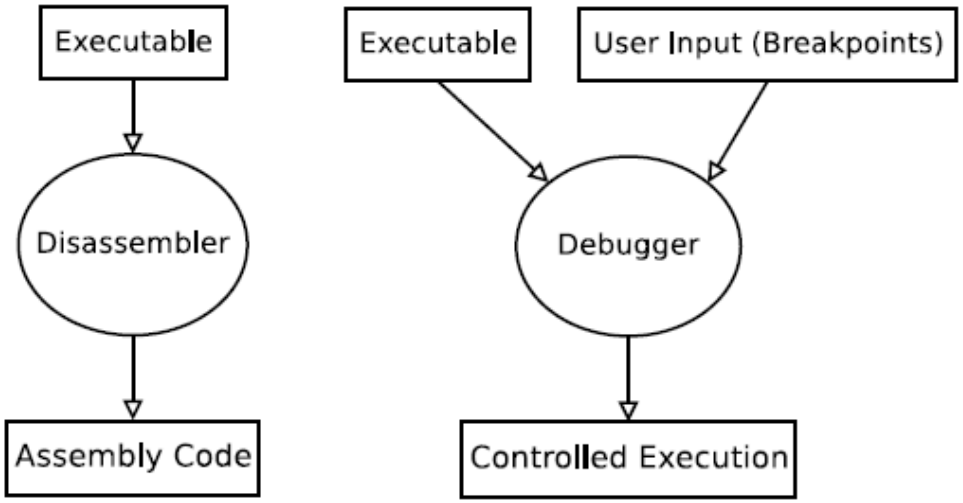


Reverse Engineering Tools

- **Disassembler/Debuggers**
  - Digunakan untuk menentukan fungsi dasar dari sebuah program executables (binary)
  - Program disassembler akan mentranslasikan binary program ke bahasa assembly. Bahasa Assembly yang dihasilkan sudah dapat dibaca oleh user tetapi tidak semudah source code aslinya.
  - Debugger digunakan untuk memantau proses eksekusi sebuah aplikasi yang dapat dihentikan pada kondisi dan status tertentu.

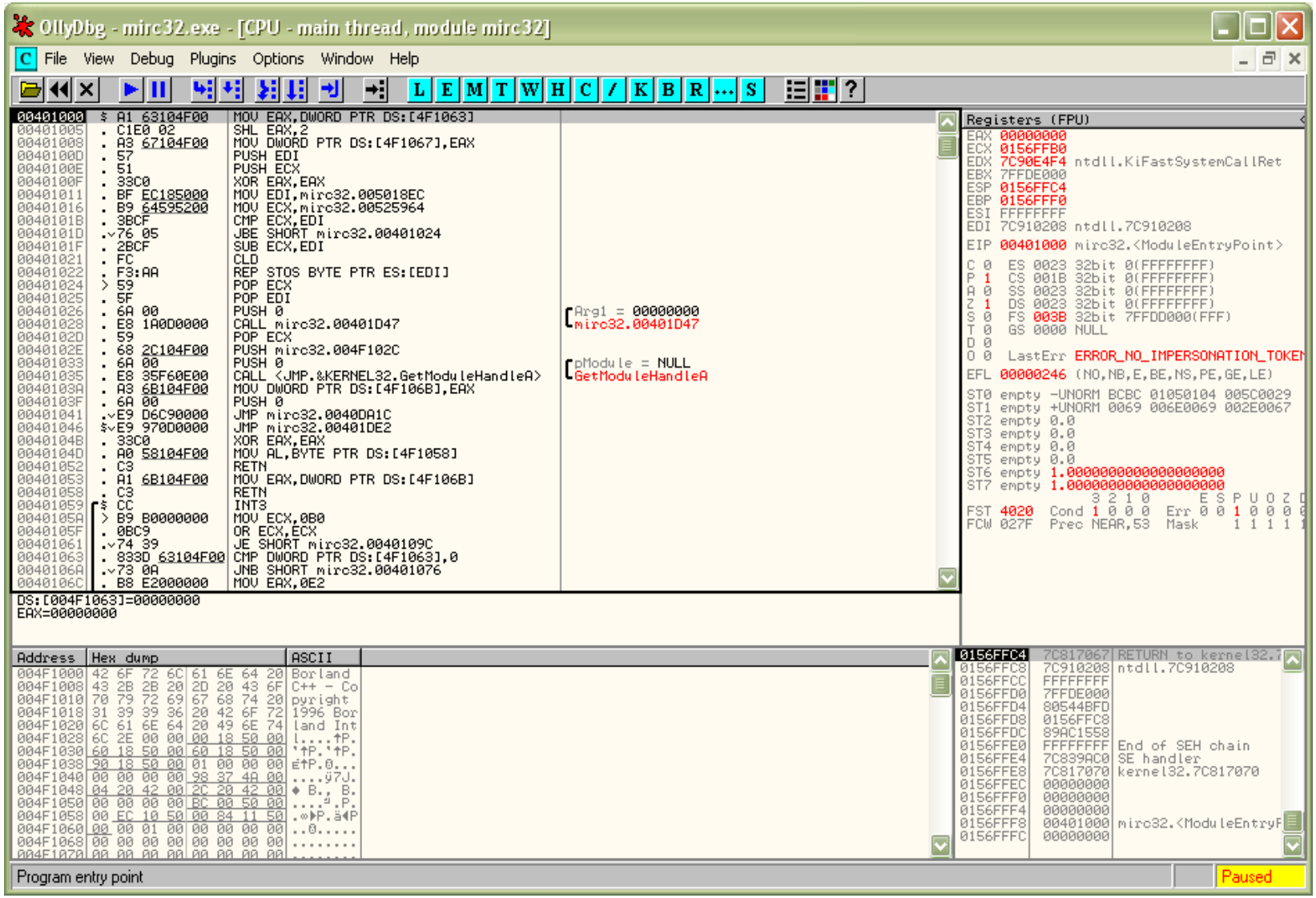
Reverse Engineering Tools

- Skema Disassembler/Debugger

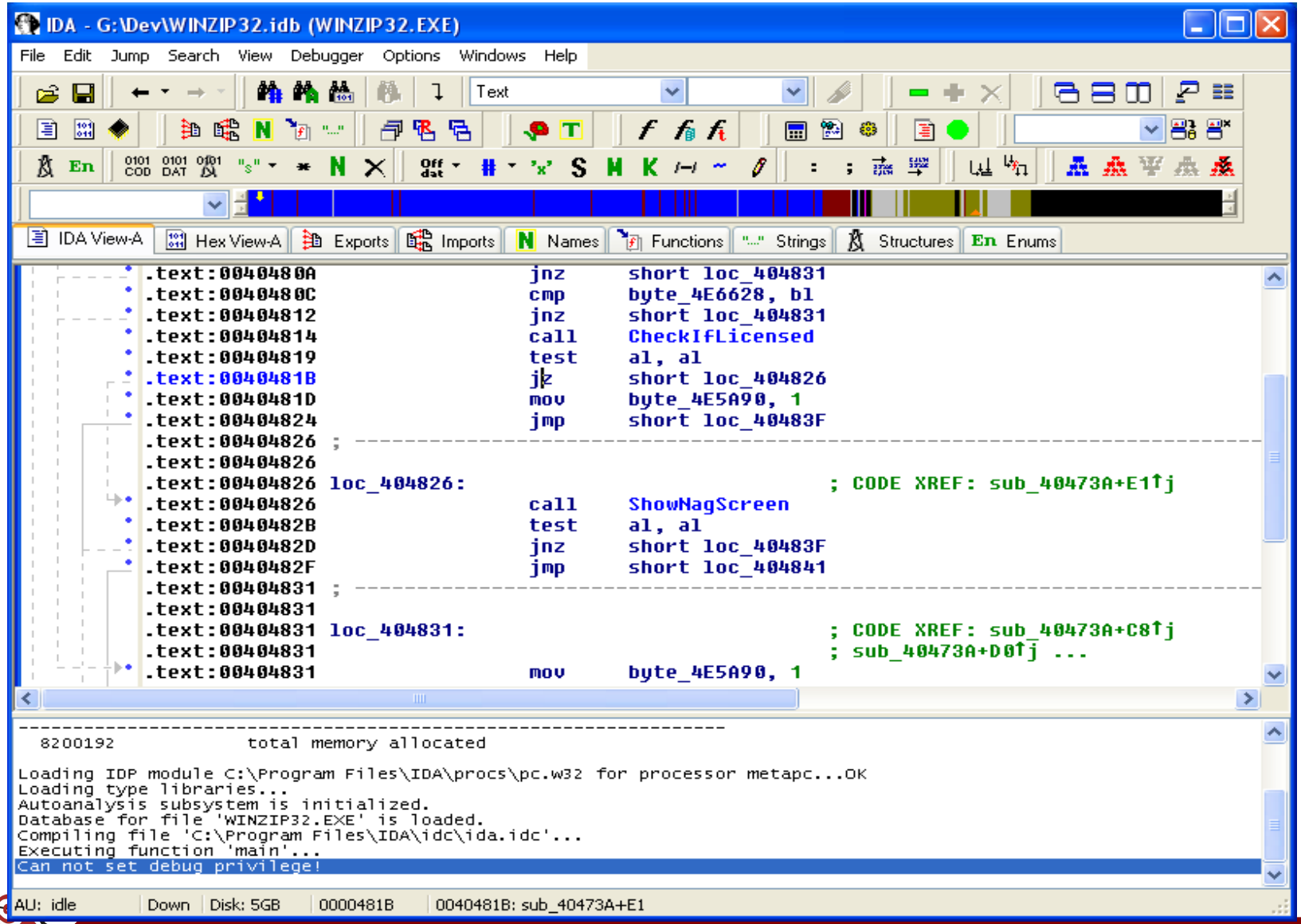


- Contoh aplikasi yang umum digunakan
  - **Debugger: SoftICE, OllyDbg, W32Dasm**
  - **Disassembler: IDA Pro, W32Dasm, Phoenix Disassembler / DSM Studio,**

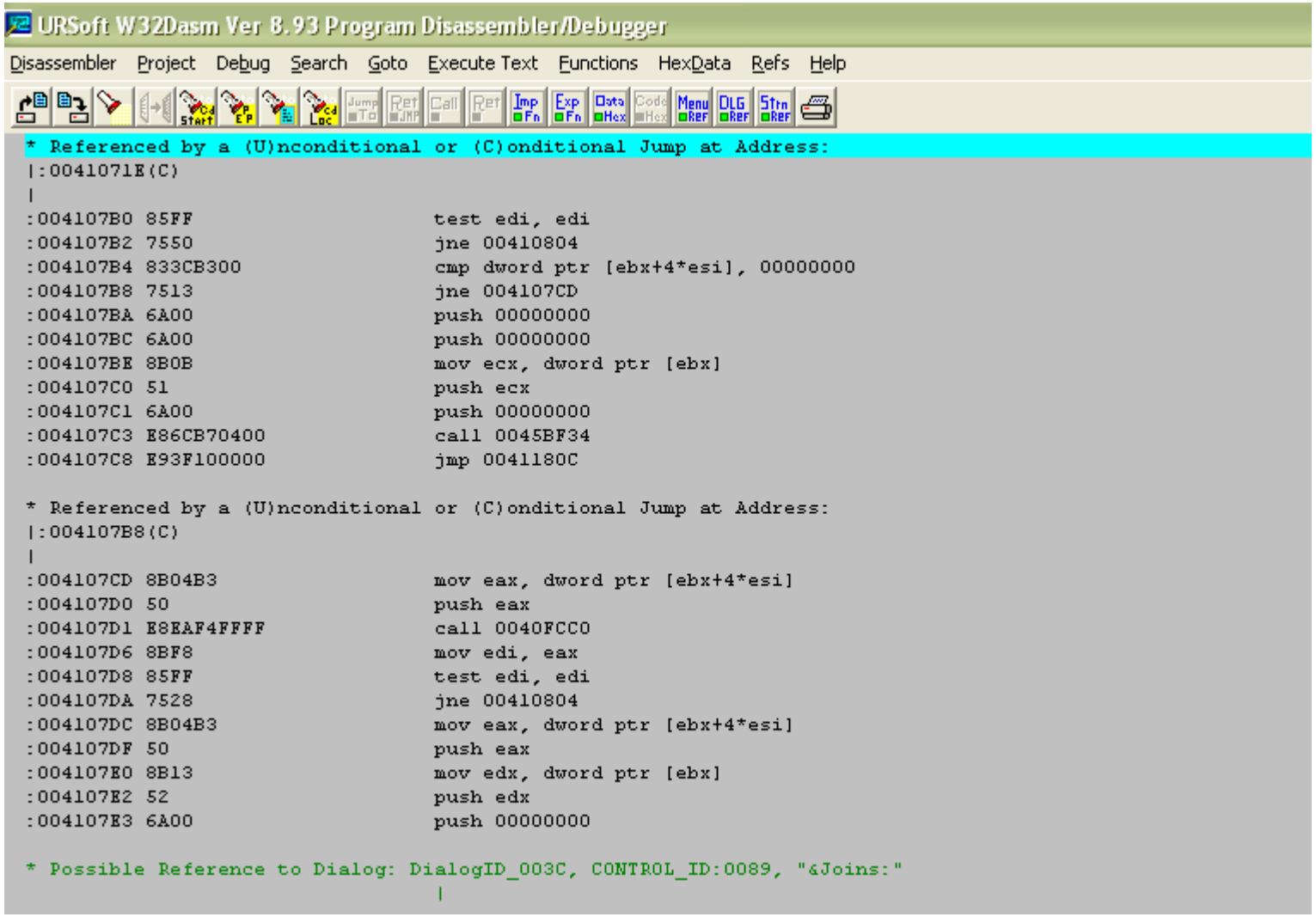
OllyDbg


OllyDbg - mirc32.exe - [CPU - main thread, module mirc32]
File View Debug Plugins Options Window Help
00401000 $ A1 63104F00 MOV EAX,DWORD PTR DS:[4F1063]
00401005 . C1E0 02 SHL EAX,2
00401008 . A3 67104F00 MOV DWORD PTR DS:[4F1067],EAX
0040100D . 57 PUSH EDI
0040100E . 51 PUSH ECX
0040100F . 33C0 XOR EAX,EAX
00401011 . BF EC185000 MOV EDI,mirc32.005018EC
00401016 . B9 64595200 MOV ECX,mirc32.00525964
0040101B . 3BCF CMP ECX,EDI
0040101D .v76 05 JBE SHORT mirc32.00401024
0040101F . 2BCF SUB ECX,EDI
00401021 . FC CLD
00401022 . F3:AA REP STOS BYTE PTR ES:[EDI]
00401024 > 59 POP ECX
00401025 . 5F POP EDI
00401026 . 6A 00 PUSH 0
00401028 . E8 1A0D0000 CALL mirc32.00401D47
0040102D . 59 POP ECX
0040102E . 68 2C104F00 PUSH mirc32.004F102C
00401033 . 6A 00 PUSH 0
00401035 . E8 35F60E00 CALL <JMP.&KERNEL32.GetModuleHandleA>
0040103A . A3 6B104F00 MOV DWORD PTR DS:[4F106B],EAX
0040103F . 6A 00 PUSH 0
00401041 .vE9 D6C90000 JMP mirc32.0040DA1C
00401046 $vE9 970D0000 JMP mirc32.00401DE2
0040104B . 33C0 XOR EAX,EAX
0040104D . A0 58104F00 MOV AL,BYTE PTR DS:[4F1058]
00401052 . C3 RETN
00401053 . A1 6B104F00 MOV EAX,DWORD PTR DS:[4F106B]
00401058 . C3 RETN
00401059 $ CC INT3
0040105A > B9 B0000000 MOV ECX,0B0
0040105F . 0BC9 OR ECX,ECX
00401061 .v74 39 JE SHORT mirc32.0040109C
00401063 . 833D 63104F00 CMP DWORD PTR DS:[4F1063],0
0040106A .v73 0A JNB SHORT mirc32.00401076
0040106C . B8 E2000000 MOV EAX,0E2
Arg1 = 00000000
mirc32.00401D47
pModule = NULL
GetModuleHandleA
DS:[004F1063]=00000000
EAX=00000000
Registers (FPU)
EAX 00000000
ECX 0156FFB0
EDX 7C90E4F4 ntdll.KiFastSystemCallRet
EBX 7FFDE000
ESP 0156FFC4
EBP 0156FFF0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 00401000 mirc32.<ModuleEntryPoint>
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDD000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_NO_IMPERSONATION_TOKEN
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty -UNORM BCBC 01050104 005C0029
ST1 empty +UNORM 0069 006E0069 002E0067
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.0000000000000000000
ST7 empty 1.0000000000000000000
3 2 1 0 E S P U O Z D
FST 4020 Cond 1 0 0 0 Err 0 0 1 0 0 0 0
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1
Address Hex dump ASCII
004F1000 42 6F 72 6C 61 6E 64 20 Borland
004F1008 43 2B 2B 20 2D 20 43 6F C++ - Co
004F1010 70 79 72 69 67 68 74 20 pyright
004F1018 31 39 39 36 20 42 6F 72 1996 Bor
004F1020 6C 61 6E 64 20 49 6E 74 land Int
004F1028 6C 2E 00 00 00 18 50 00 l....↑P.
004F1030 60 18 50 00 60 18 50 00 `↑P.`↑P.
004F1038 90 18 50 00 01 00 00 00 É↑P.☺...
004F1040 00 00 00 00 98 37 4A 00 ....ÿ7J.
004F1048 04 20 42 00 2C 20 42 00 ♦ B. , B.
004F1050 00 00 00 00 BC 00 50 00 ....¼.P.
004F1058 00 EC 10 50 00 84 11 50 .∞►P.ä◄P
004F1060 00 00 01 00 00 00 00 00 ..☺.....
004F1068 00 00 00 00 00 00 00 00 ....
004F1070 00 00 00 00 00 00 00 00 ....
0156FFC4 7C817067 RETURN to kernel32.7
0156FFC8 7C910208 ntdll.7C910208
0156FFCC FFFFFFFF
0156FFD0 7FFDE000
0156FFD4 80544BFD
0156FFD8 0156FFC8
0156FFDC 89AC1558
0156FFE0 FFFFFFFF End of SEH chain
0156FFE4 7C839AC0 SE handler
0156FFE8 7C817070 kernel32.7C817070
0156FFEC 00000000
0156FFF0 00000000
0156FFF4 00000000
0156FFF8 00401000 mirc32.<ModuleEntryF
0156FFFC 00000000
Program entry point
Paused

IDA Pro

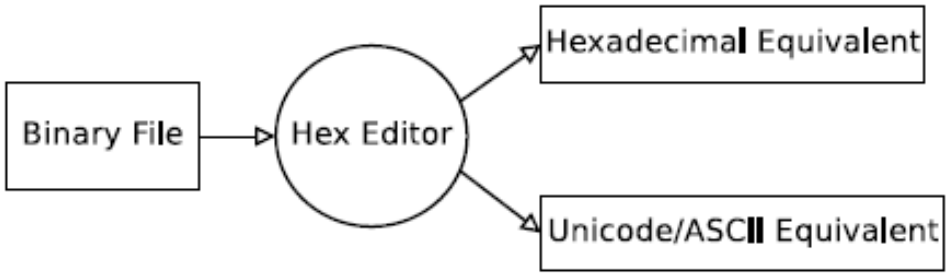

IDA - G:\Dev\WINZIP32.idb (WINZIP32.EXE)
File Edit Jump Search View Debugger Options Windows Help
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums
.text:0040480A jnz short loc_404831
.text:0040480C cmp byte_4E6628, bl
.text:00404812 jnz short loc_404831
.text:00404814 call CheckIfLicensed
.text:00404819 test al, al
.text:0040481B jz short loc_404826
.text:0040481D mov byte_4E5A90, 1
.text:00404824 jmp short loc_40483F
.text:00404826 loc_404826: ; CODE XREF: sub_40473A+E1↑j
.text:00404826 call ShowNagScreen
.text:0040482B test al, al
.text:0040482D jnz short loc_40483F
.text:0040482F jmp short loc_404841
.text:00404831 loc_404831: ; CODE XREF: sub_40473A+C8↑j
.text:00404831 ; sub_40473A+D0↑j ...
.text:00404831 mov byte_4E5A90, 1
8200192 total memory allocated
Loading IDP module C:\Program Files\IDA\procs\pc.w32 for processor metapc...OK
Loading type libraries...
Autoanalysis subsystem is initialized.
Database for file 'WINZIP32.EXE' is loaded.
Compiling file 'C:\Program Files\IDA\idc\ida.idc'...
Executing function 'main'...
Can not set debug privilege!
AU: idle
Down
Disk: 5GB
0000481B
0040481B: sub_40473A+E1

W32DASM


URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger
Disassembler Project Debug Search Goto Execute Text Functions HexData Refs Help
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0041071E(C)
|
:004107B0 85FF test edi, edi
:004107B2 7550 jne 00410804
:004107B4 833CB300 cmp dword ptr [ebx+4*esi], 00000000
:004107B8 7513 jne 004107CD
:004107BA 6A00 push 00000000
:004107BC 6A00 push 00000000
:004107BE 8B0B mov ecx, dword ptr [ebx]
:004107C0 51 push ecx
:004107C1 6A00 push 00000000
:004107C3 E86CB70400 call 0045BF34
:004107C8 E93F100000 jmp 0041180C
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004107B8(C)
|
:004107CD 8B04B3 mov eax, dword ptr [ebx+4*esi]
:004107D0 50 push eax
:004107D1 E8EAF4FFFF call 0040FCC0
:004107D6 8BF8 mov edi, eax
:004107D8 85FF test edi, edi
:004107DA 7528 jne 00410804
:004107DC 8B04B3 mov eax, dword ptr [ebx+4*esi]
:004107DF 50 push eax
:004107E0 8B13 mov edx, dword ptr [ebx]
:004107E2 52 push edx
:004107E3 6A00 push 00000000
* Possible Reference to Dialog: DialogID_003C, CONTROL_ID:0089, "&Joins:"
|

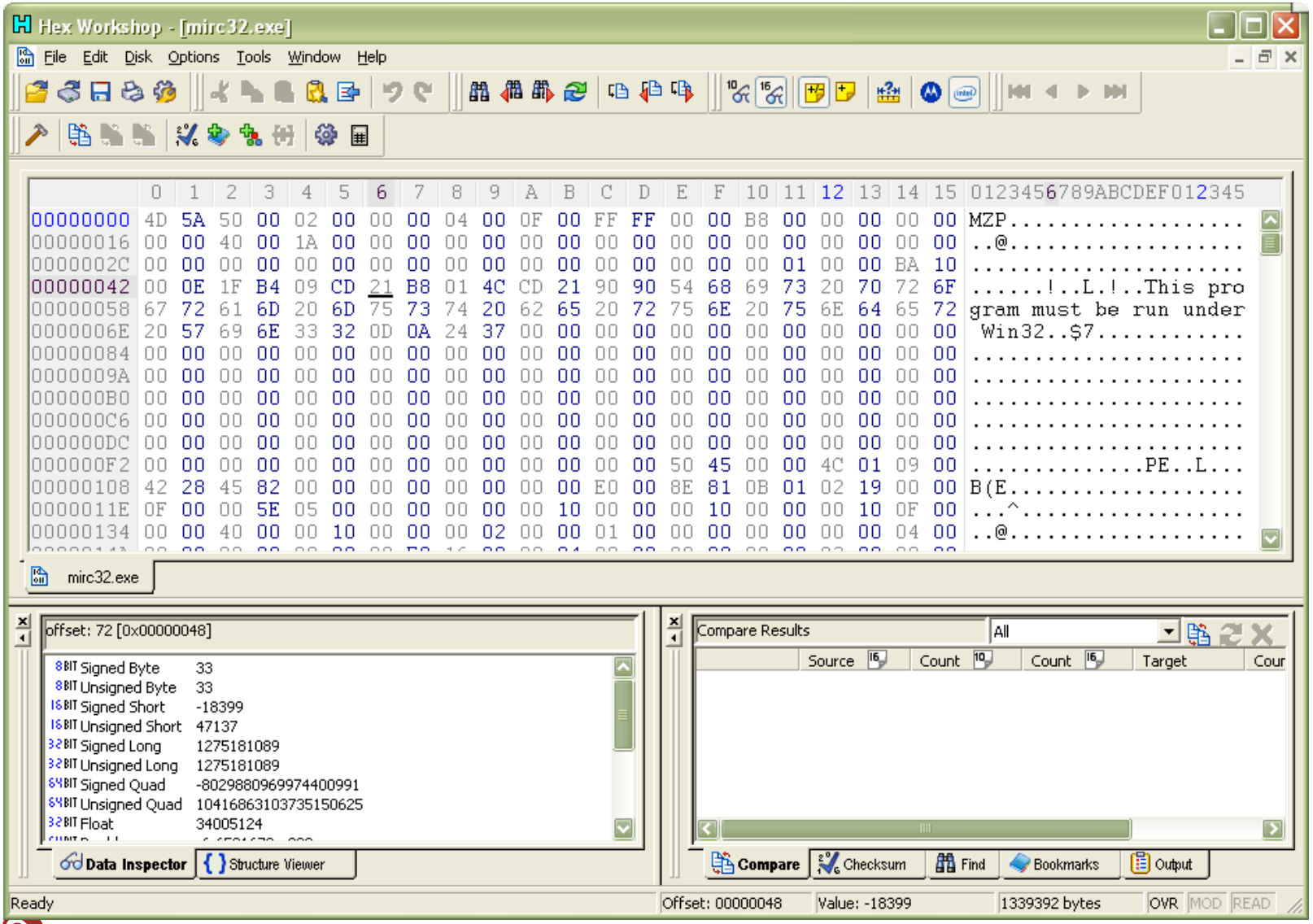
Reverse Engineering Tools

- Hex Editor
  - Program yang memfasilitasi pengubahan sebuah binary program melalui representasi hexadesimal suata data binary
  - Hex Editor juga mampu menampilkan bentuk ASCII dan Unicode suatu file binary

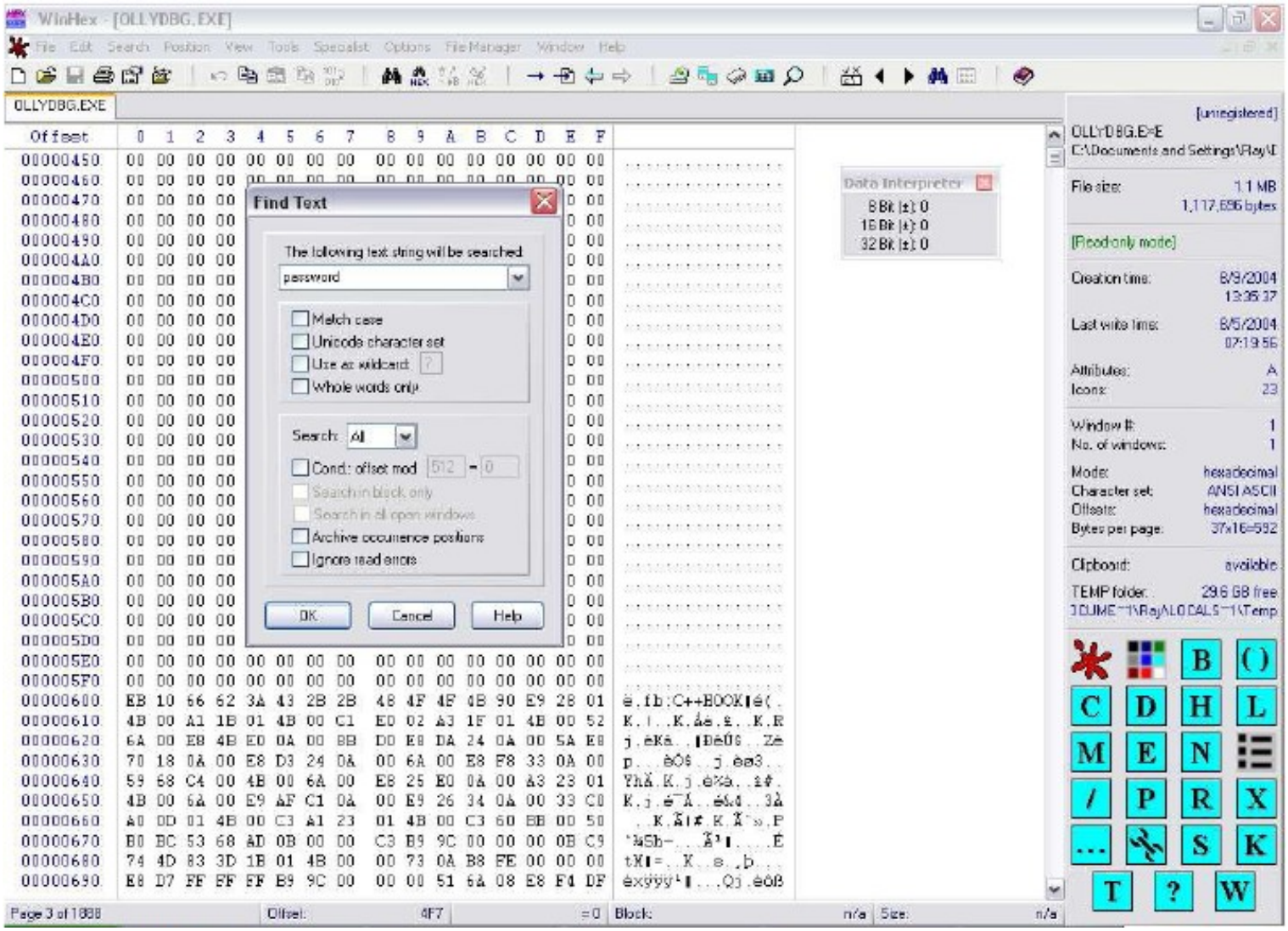


  - Contoh Hex Editor: WinHex, Hex Workshop, Hiew (Hacker View), HEdit

Hex Workshop

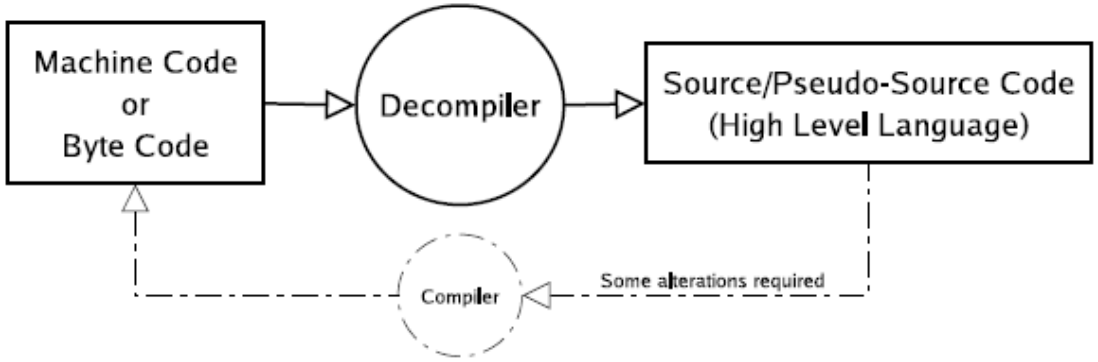


WinHEX


WinHex - [OLLYDBG.EXE]
File Edit Search Position View Tools Specialist Options File Manager Window Help
OLLYDBG.EXE
Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F
Find Text
The following text string will be searched
password
Match case
Unicode character set
Use as wildcard
Whole words only
Search: All
Cond: offset mod
Search in block only
Search in all open windows
Archive occurrence positions
Ignore read errors
OK
Cancel
Help
Data Interpreter
8 Bit (±) 0
16 Bit (±) 0
32 Bit (±) 0
[unregistered]
OLLYDBG.EXE
File size: 1.1 MB
1,117,696 bytes
(Read-only mode)
Creation time: 8/9/2004 13:35:37
Last write time: 8/5/2004 07:19:56
Attributes: A
Icons: 23
Window #: 1
No. of windows: 1
Mode: hexadecimal
Character set: ANSI ASCII
Offsets: hexadecimal
Bytes per page: 37x16=592
Clipboard: available
TEMP folder: 29.6 GB free
Page 3 of 1888
Offset: 4F7
Block: n/a
Size: n/a

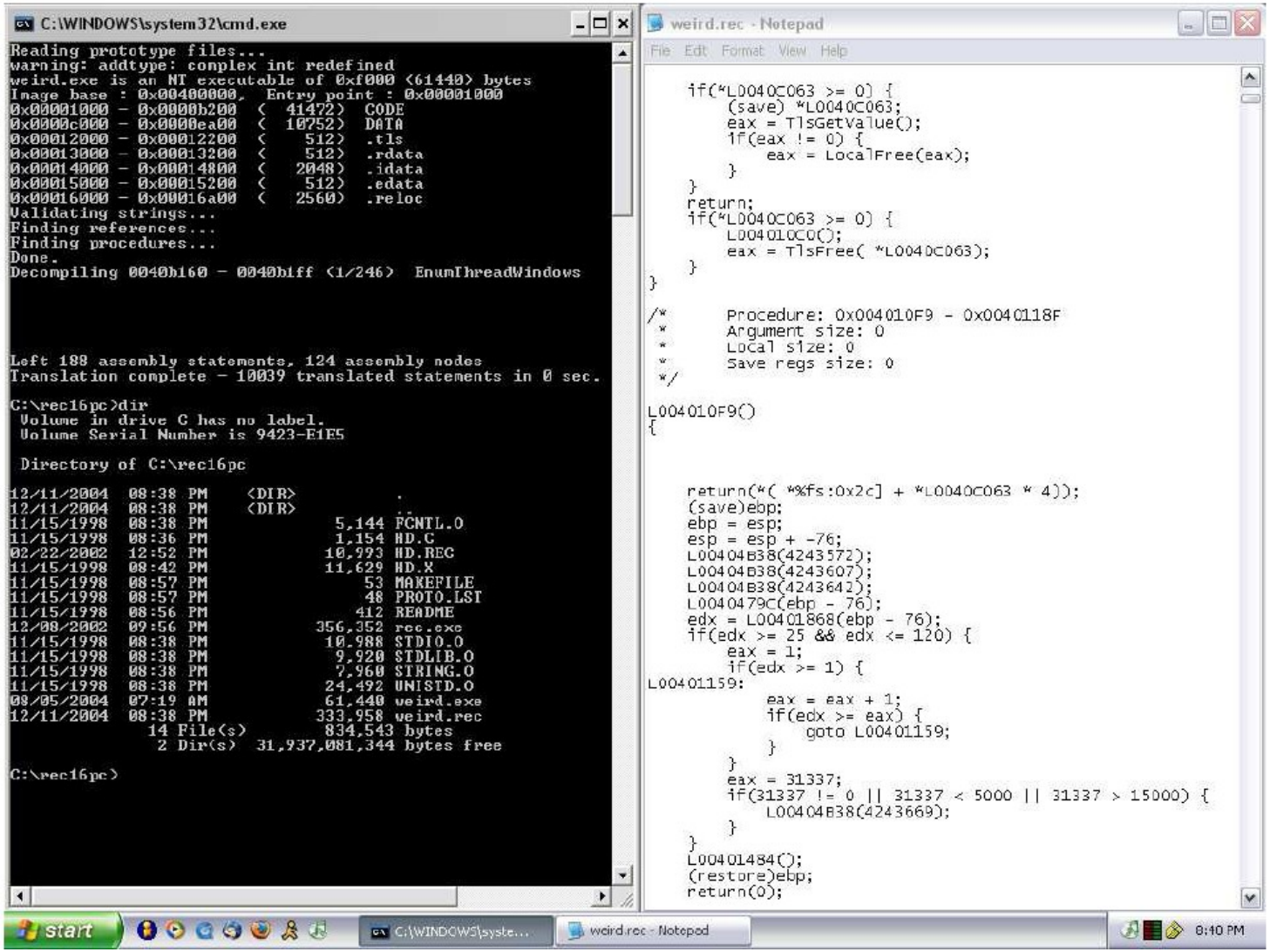
Reverse Engineering Tools

- **Decompiler**
  - **Mentranslasikan program Executable menjadi Source Code**

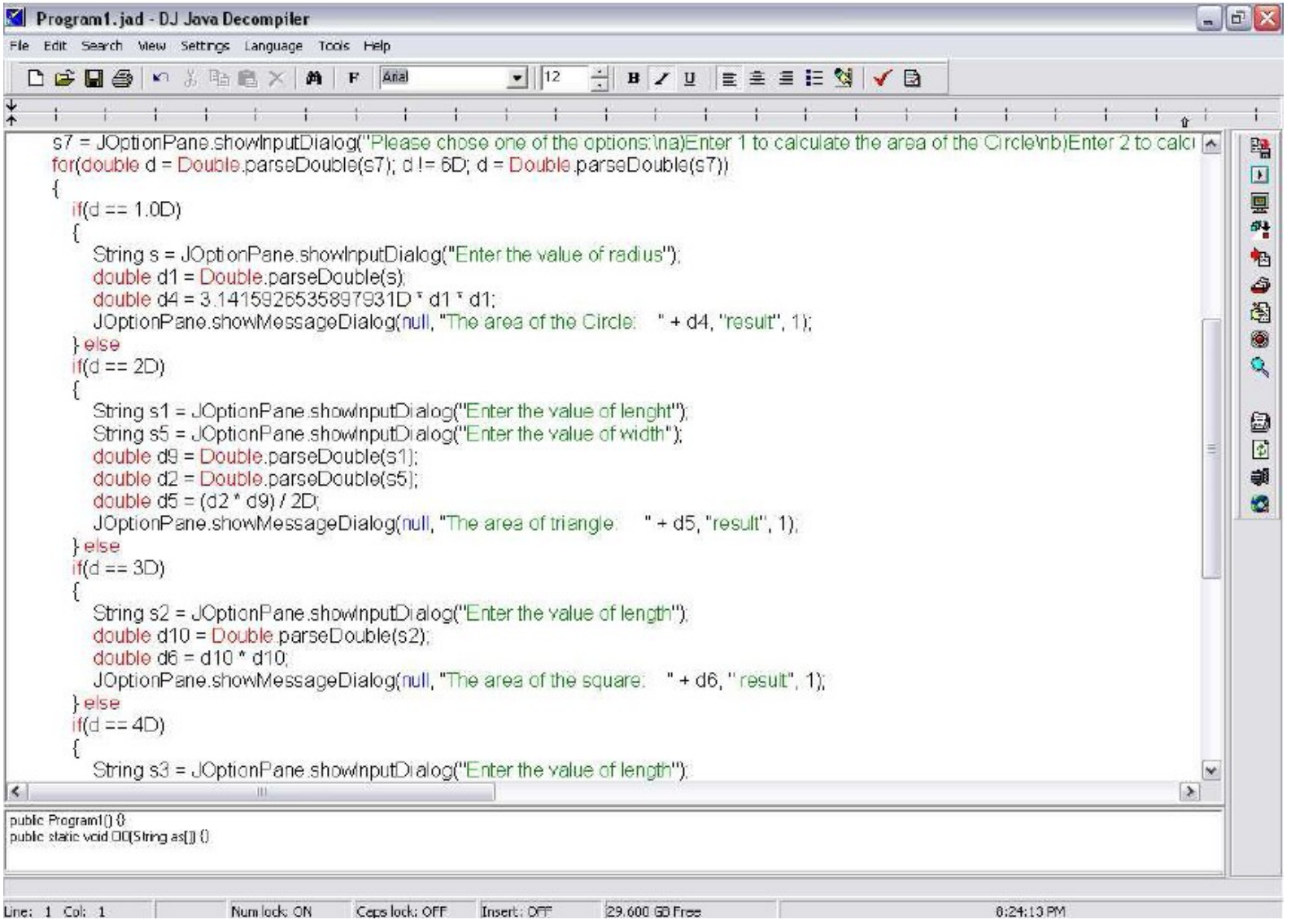


  - **Contoh :**
    - **C Decompiler : DCC, REC**
    - **Java Decompiler: DJ**
    - **C# Decompiler : Salamander**

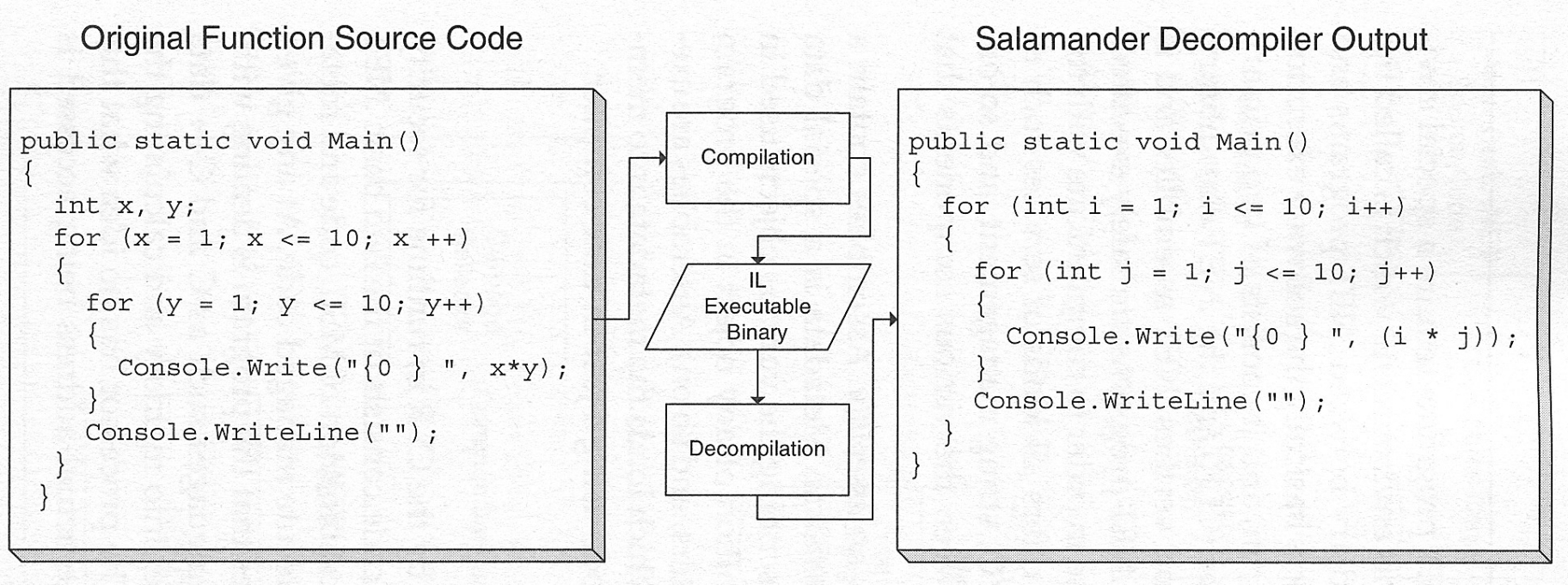
C Decompiler : REC



Java Decompiler: DJ



C# Decompiler : Salamander

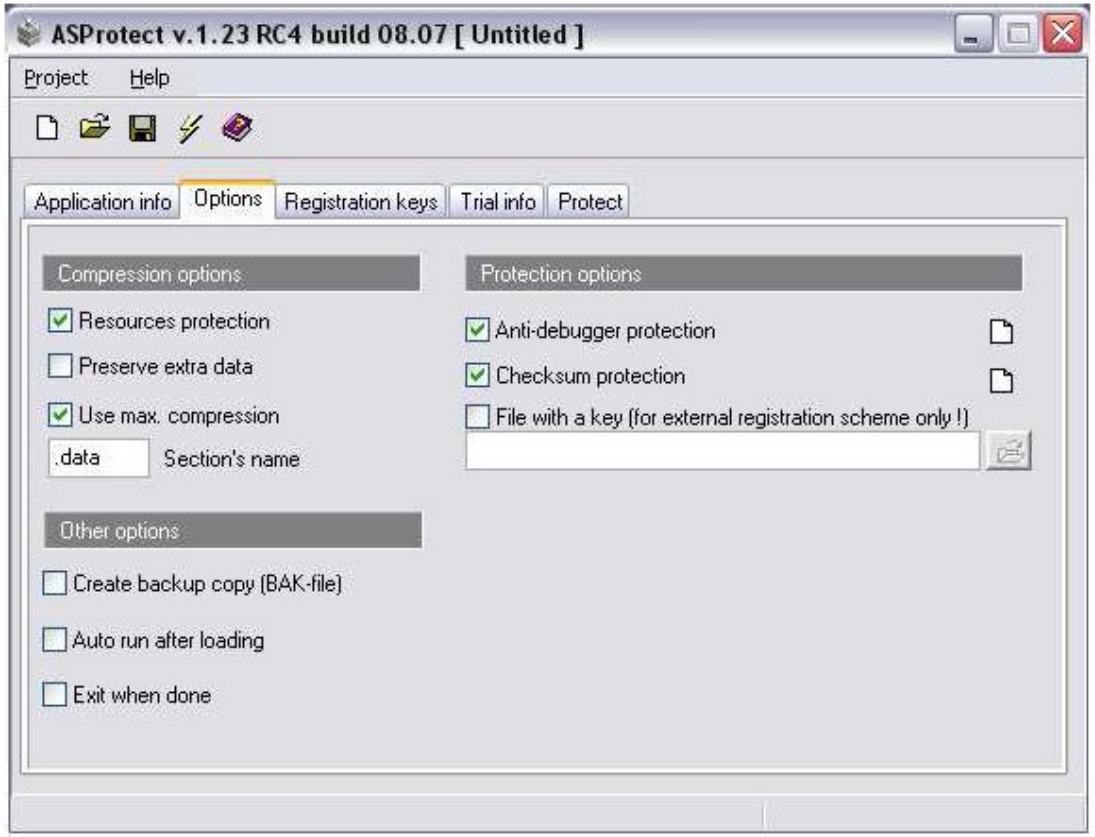


Reverse Engineering Tools

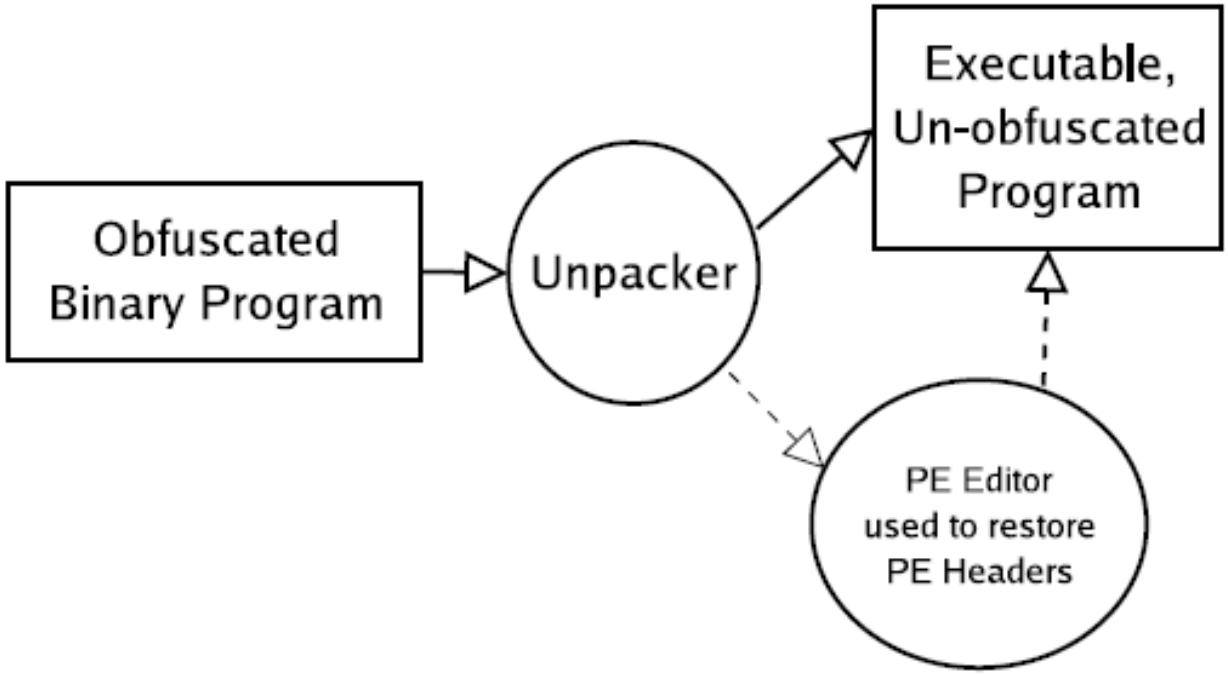
- Aplikasi yang berhubungan spt code obfuscators, PE editors, memory dumpers dan unpacker
  - Code Obfuscators : aplikasi yang mempersulit suatu source code atau binary untuk dipahami atau dibaca.
  - PE Editors : Mengextract headers dari File PE dan kemudian mempermudah PE File untuk di edit dibandingkan menggunakan Hex Editor
  - Memory dumpers berfungsi agar sebuah program yang sedang sedang berjalan (debugging) di memory dapat disimpan kedalam hardisk atau media penyimpan lainnya.
  - Unpacker: Digunakan untuk membongkar suatu program yang diproteksi (dibungkus) dengan aplikasi keamanan komersial yang telah ditemukan kelemahannya.

Code Obfuscators

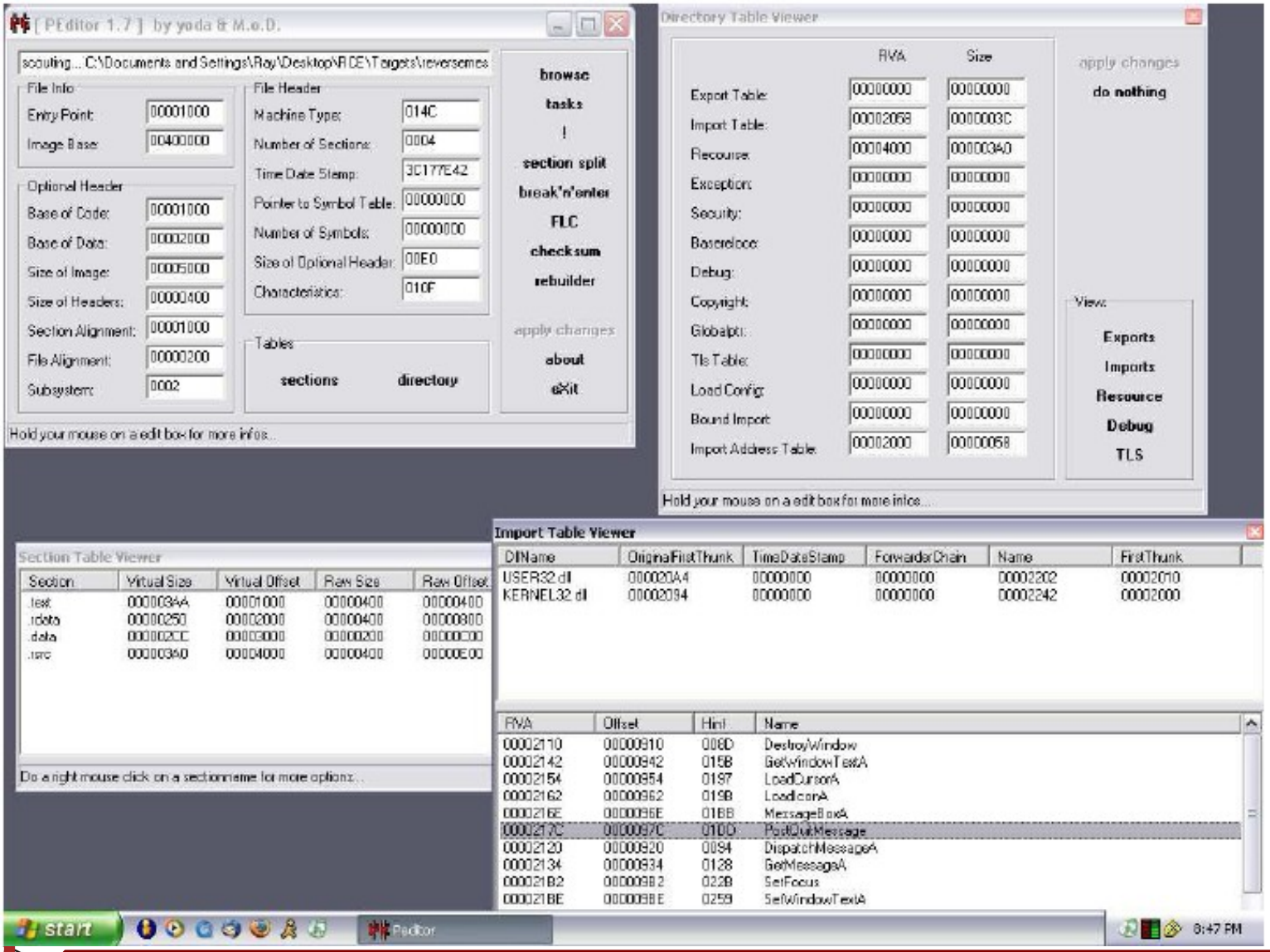
- **ASProtect**



Unpackers


Executable,
Un-obfuscated
Program
Obfuscated
Binary Program
Unpacker
PE Editor
used to restore
PE Headers

PEditor


[PEditor 1.7] by yoda & M.e.D.
scouting... C:\Documents and Settings\Ray\Desktop\RCE\Targets\reversemes
File Info
Entry Point: 00001000
Image Base: 00400000
File Header
Machine Type: 014C
Number of Sections: 0004
Time Date Stamp: 3D177E42
Pointer to Symbol Table: 00000000
Number of Symbols: 00000000
Size of Optional Header: 00E0
Characteristics: 010F
Optional Header
Base of Code: 00001000
Base of Data: 00002000
Size of Image: 00005000
Size of Headers: 00000400
Section Alignment: 00001000
File Alignment: 00000200
Subsystem: 0002
Tables
sections
directory
browse
tasks
!
section split
break'n'enter
FLC
checksum
rebuilder
apply changes
about
eXit
Hold your mouse on a edit box for more infos...
Directory Table Viewer
RVA
Size
Export Table: 00000000 00000000
Import Table: 00002068 0000003C
Recourse: 00004000 00000340
Exception: 00000000 00000000
Security: 00000000 00000000
Basereloce: 00000000 00000000
Debug: 00000000 00000000
Copyright: 00000000 00000000
Globalptr: 00000000 00000000
Tls Table: 00000000 00000000
Load Config: 00000000 00000000
Bound Import: 00000000 00000000
Import Address Table: 00002000 00000058
apply changes
do nothing
View:
Exports
Imports
Resource
Debug
TLS
Hold your mouse on a edit box for more infos...
Section Table Viewer
Section Virtual Size Virtual Offset Raw Size Raw Offset
.text 000003AA 00001000 00000400 00000400
.rdata 00000250 00002000 00000400 00000800
.data 000002CC 00003000 00000200 00000C00
.rsrc 00000340 00004000 00000400 00000E00
Do a right mouse click on a sectionname for more options...
Import Table Viewer
DllName OriginalFirstThunk TimeDateStamp ForwarderChain Name FirstThunk
USER32.dll 000020A4 00000000 00000000 00002202 00002010
KERNEL32.dll 00002094 00000000 00000000 00002242 00002000
RVA Offset Hint Name
00002110 00000910 008D DestroyWindow
00002142 00000942 015B GetWindowTextA
00002154 00000954 0197 LoadCursorA
00002162 00000962 019B LoadIconA
0000216E 0000096E 01BB MessageBoxA
0000217C 0000097C 01DD PostQuitMessage
00002120 00000920 0094 DispatchMessageA
00002134 00000934 0128 GetMessageA
000021B2 000009B2 022B SetFocus
000021BE 000009BE 0259 SetWindowTextA
start
PEditor
8:47 PM

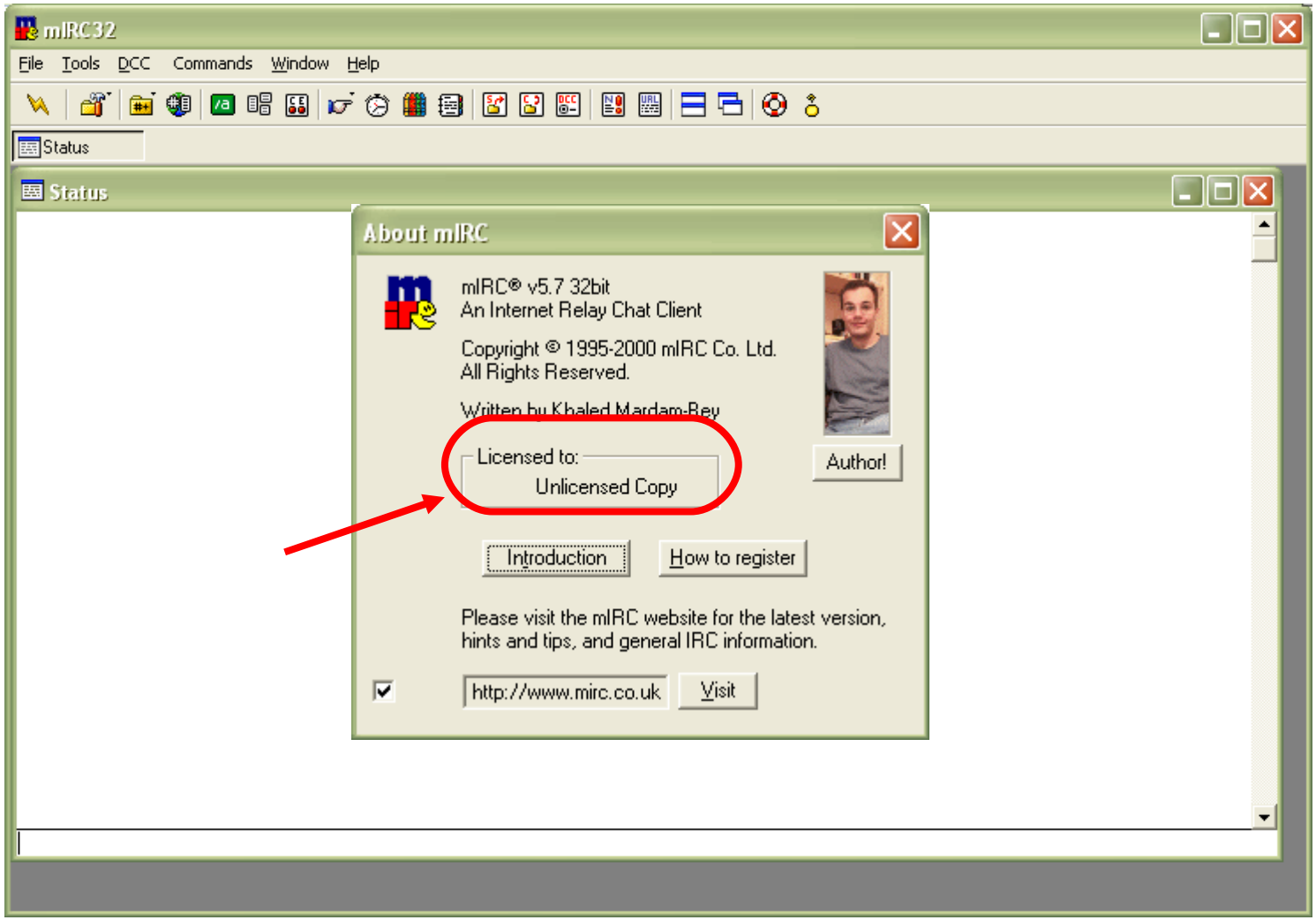
Demo Cracking Software

- **Target : Patching mIRC**
- **Download :**

  http://www.oldversion.com/download_mIRC_
- **Tools :** W32DASM & Hex Workshop

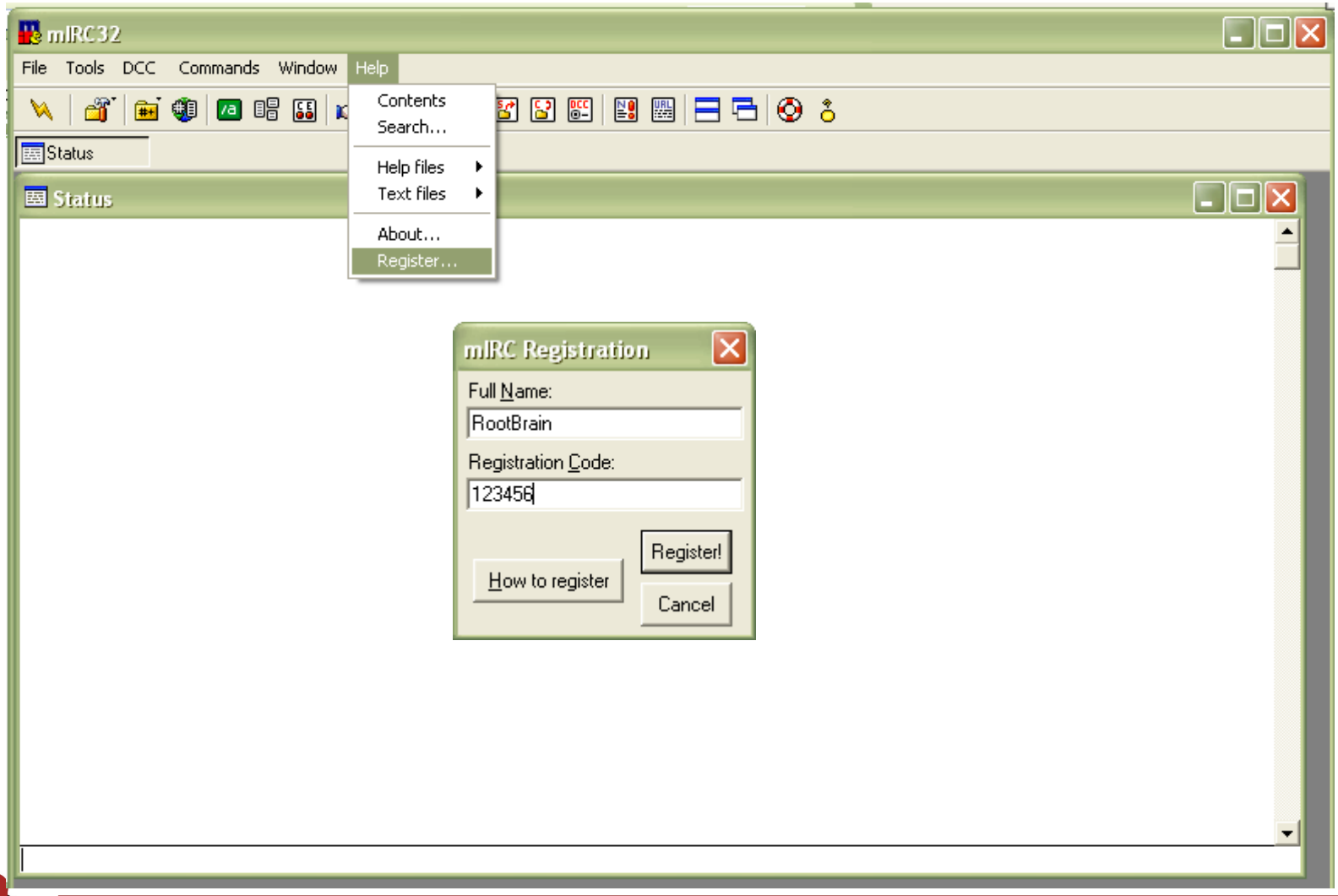
Langkah pertama

- Download dan Install mIRC 5.7 , check Help >> About



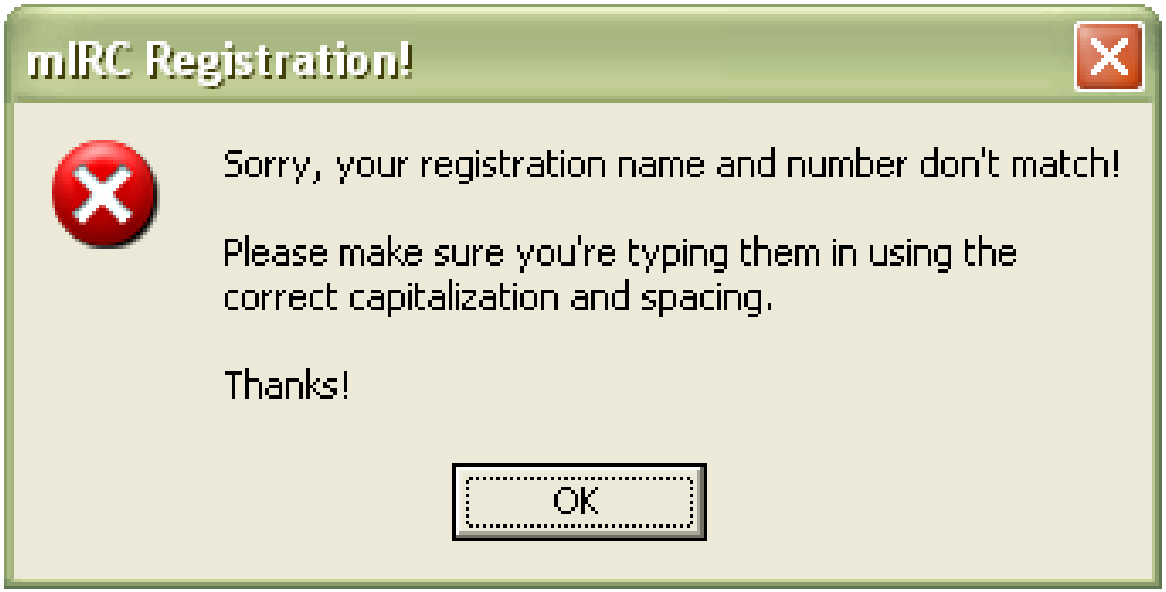
Langkah kedua

- Coba daftar dengan user sembarang



Langkah ketiga

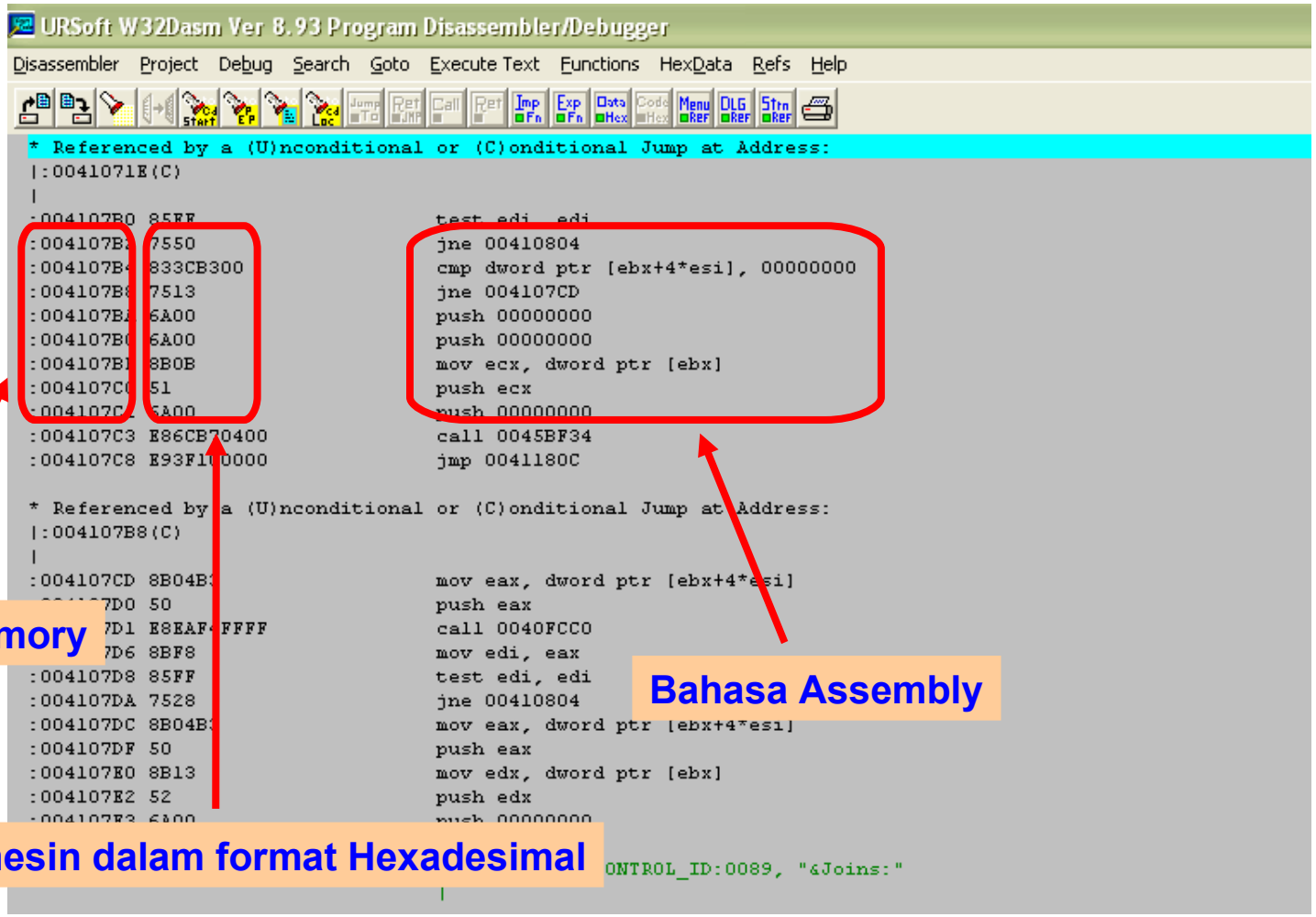
- **Hasilnya menampilkan :**



- **Catat pesan diatas untuk digunakan mencari lokasi fungsi yang menampilkan pesan tersebut**

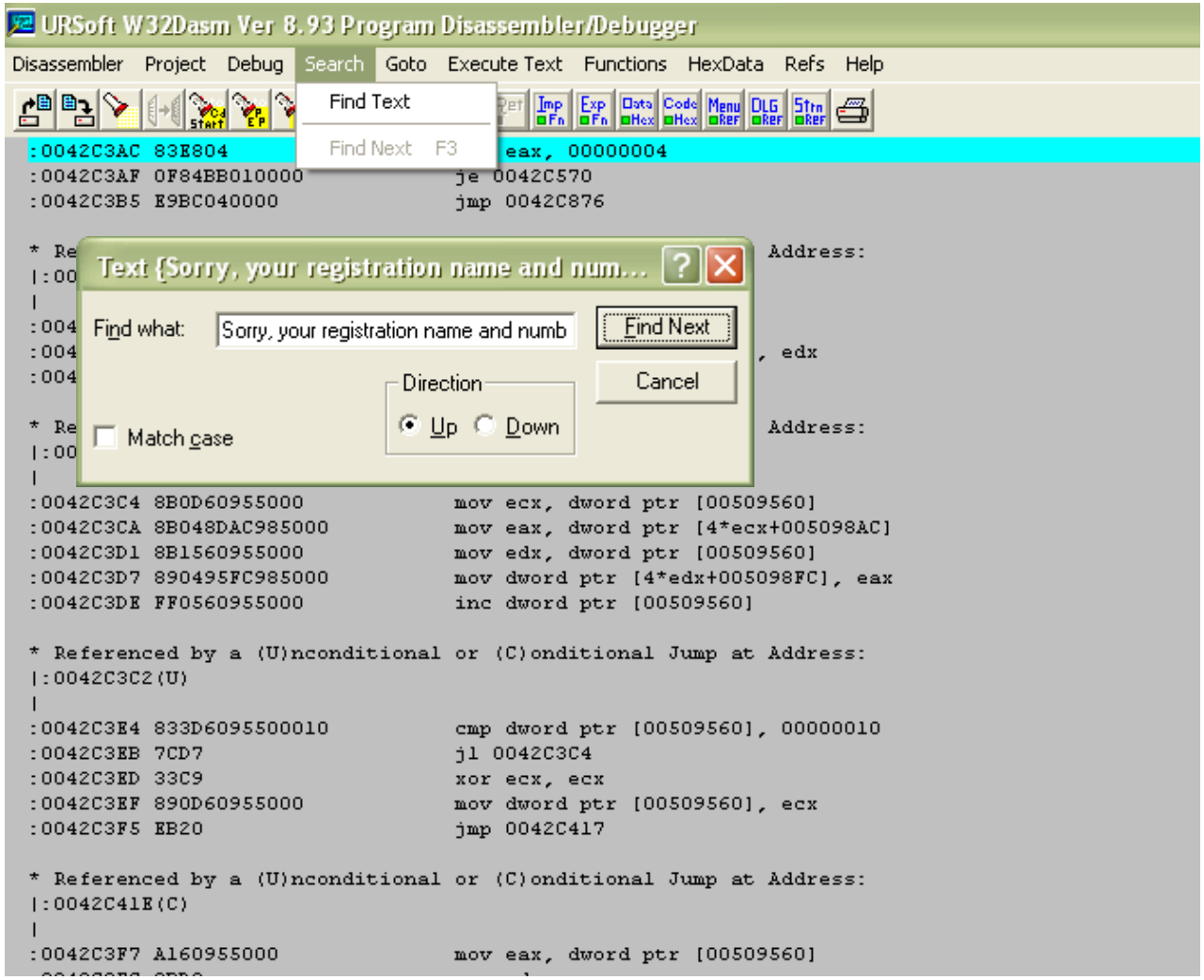
Langkah ke-empat

- Gunakan W32DASM membuka file mirc32.exe



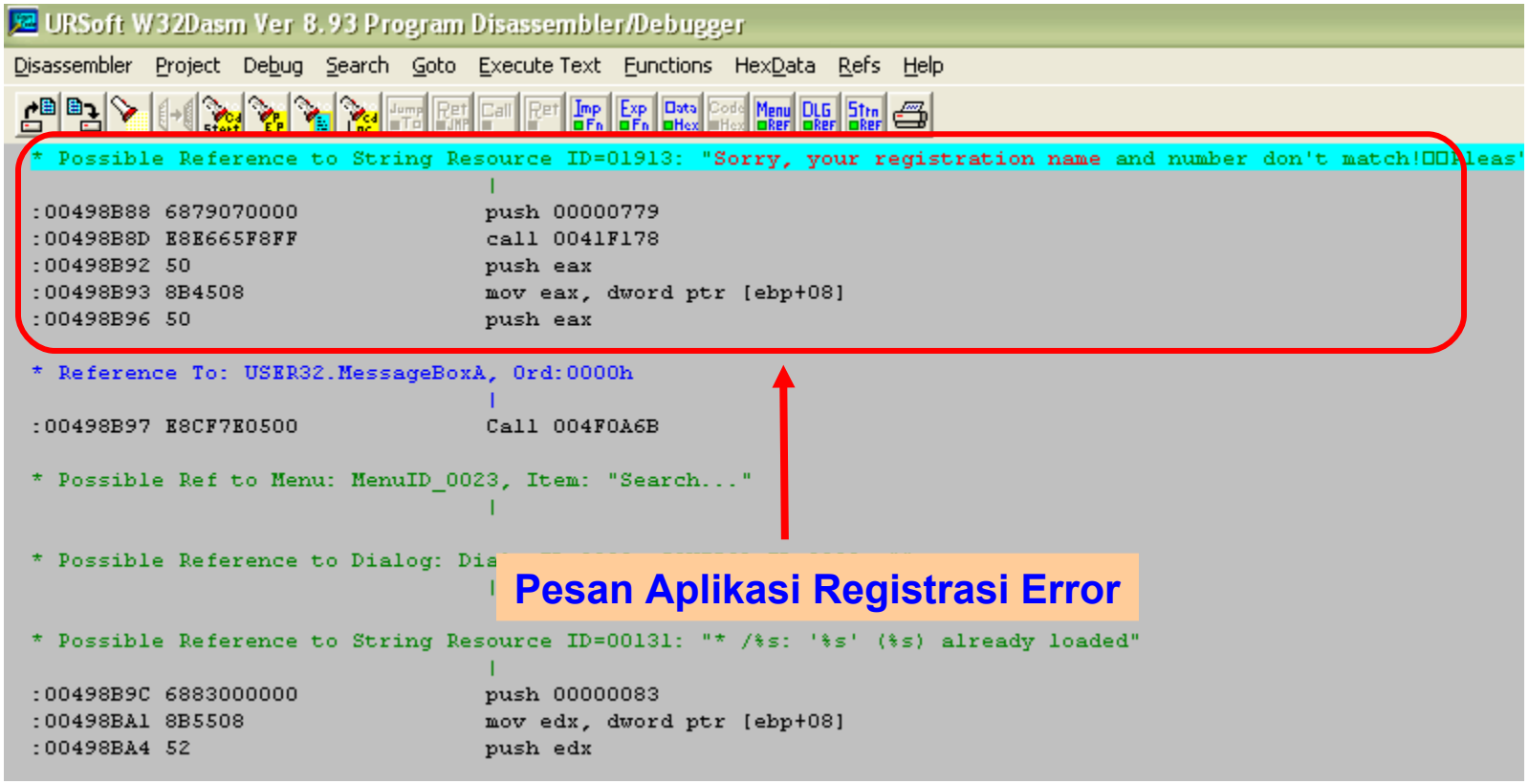
Langkah kelima

- Cari kata kunci pesan tadi



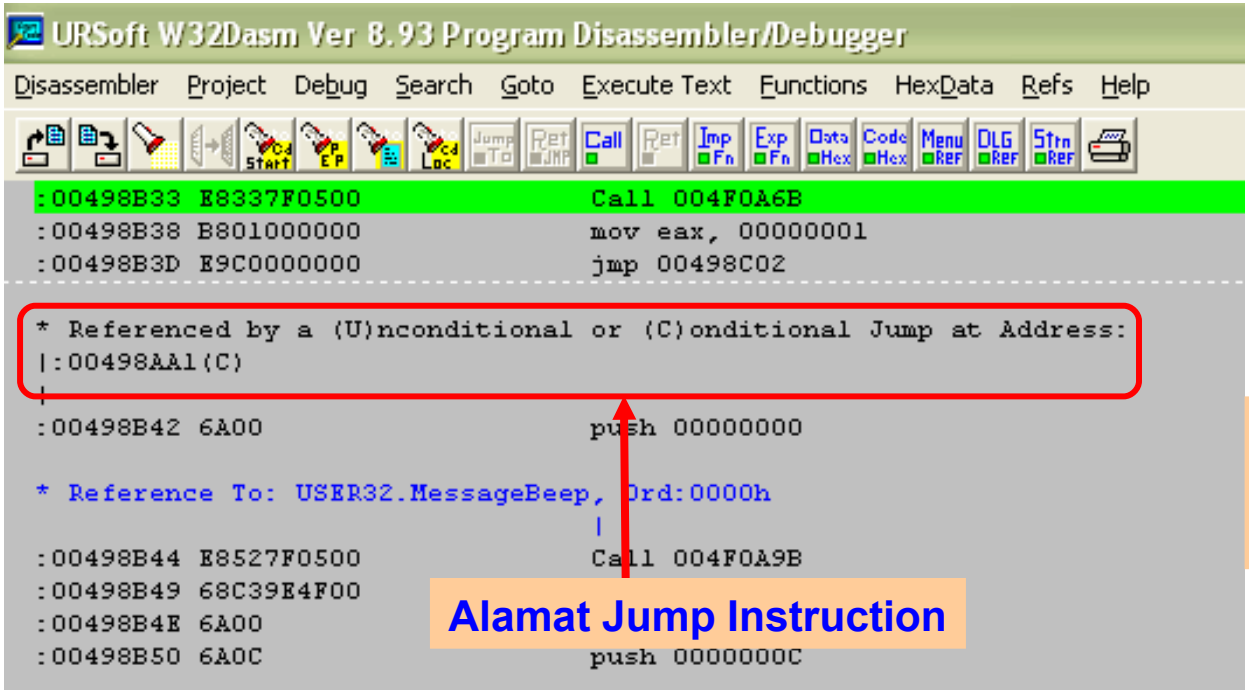
Langkah kelima

- Hasil pencarian



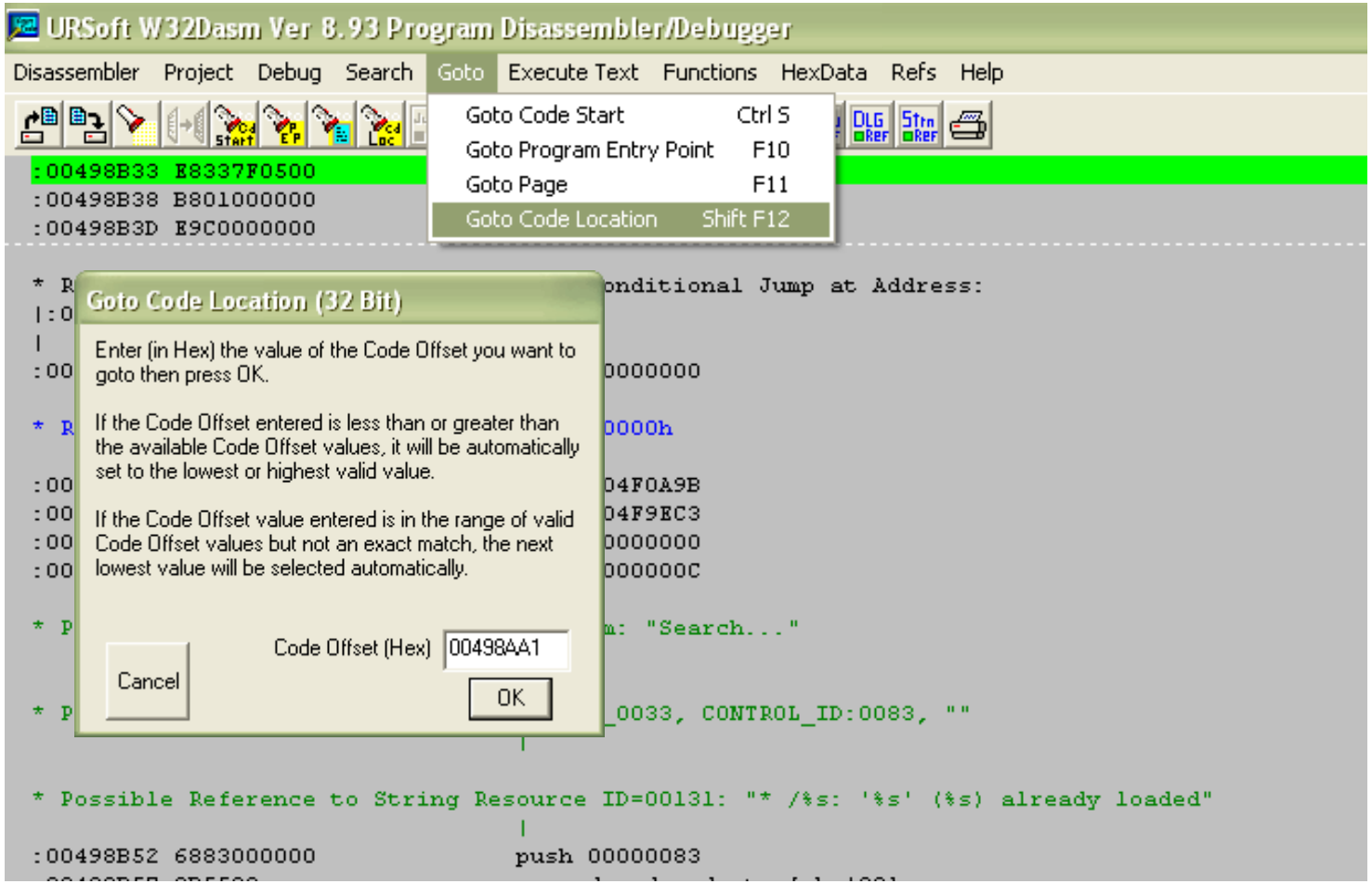
Analisis pesan error

- Kenapa muncul pesan error
- Identifikasi fungsi yang melakukan pengecekan nama dan kode registrasi. Jika nama dan kode registrasi tidak sesuai maka Jump ke alamat tertentu dan memberikan pesan kesalahan. Kita harus menemukan instruksi Jump tersebut

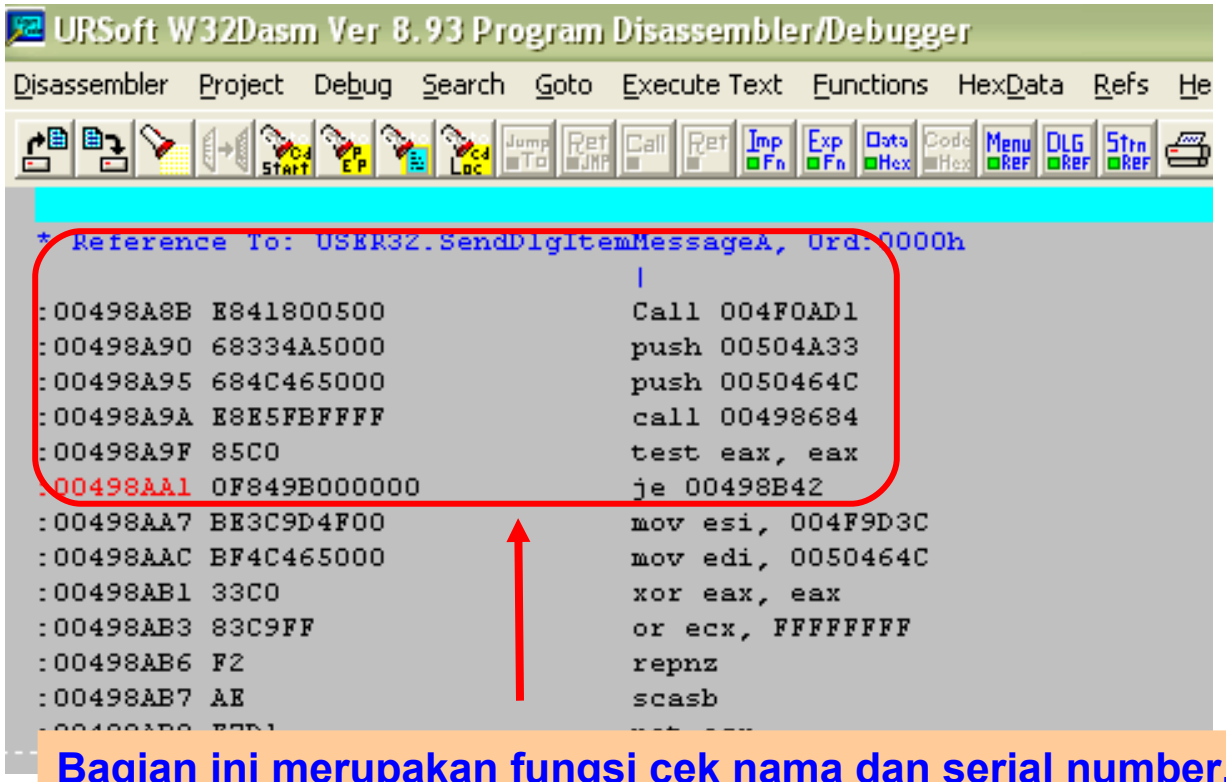


Goto Code Location

- Cari Alamat Instruksi Jump “00498AA1”



Akan menemukan:



Bagian ini merupakan fungsi cek nama dan serial number, Jika nama dan serial tidak sesuai maka, jump ke alamat "00498B42" (menampilkan pesan Sorry, your registration..)

NOP : No Operation

Kita hanya butuh memberikan instruksi nop agar dapat membypas pengecekan nama dan serial number pada baris perintah ini

```
:00498AA1    0F849B000000  je   00498B42
```

Memberi NOP

Setiap kode mesin terdiri dari nomor Hexdecimal, seperti 0F849B000000**.**

Dan untuk merepresentasikan sebuah nomor hexadesimal dibutuhkan 4 bits.

Jadi untuk instruksi ini "0F849B000000"**, dibutuhkan 6 (=12*4/8) bytes patch kode mesin.**

Assembly & Machine Code

Assembly Code	Machine Code	Description
inc eax	40	increase 1 to eax register
dec eax	48	decrease 1 to eax register
inc ebx	43	increase 1 to ebx register
dec ebx	4B	decrease 1 to ebx register
inc ecx	41	increase 1 to ecx register
dec ecx	49	decrease 1 to ecx register
inc edx	42	increase 1 to edx register
dec edx	4A	decrease 1 to ecx register
nop	90	nop means do nothing
je	74 xx	je means jump if equal
jne	75 xx	jne means jump if not equal
je	0F84 xxxx xxxx	jump if equal
jne	0F85 xxxx xxxx	jemp if not equal

Memberikan Instruksi NOP

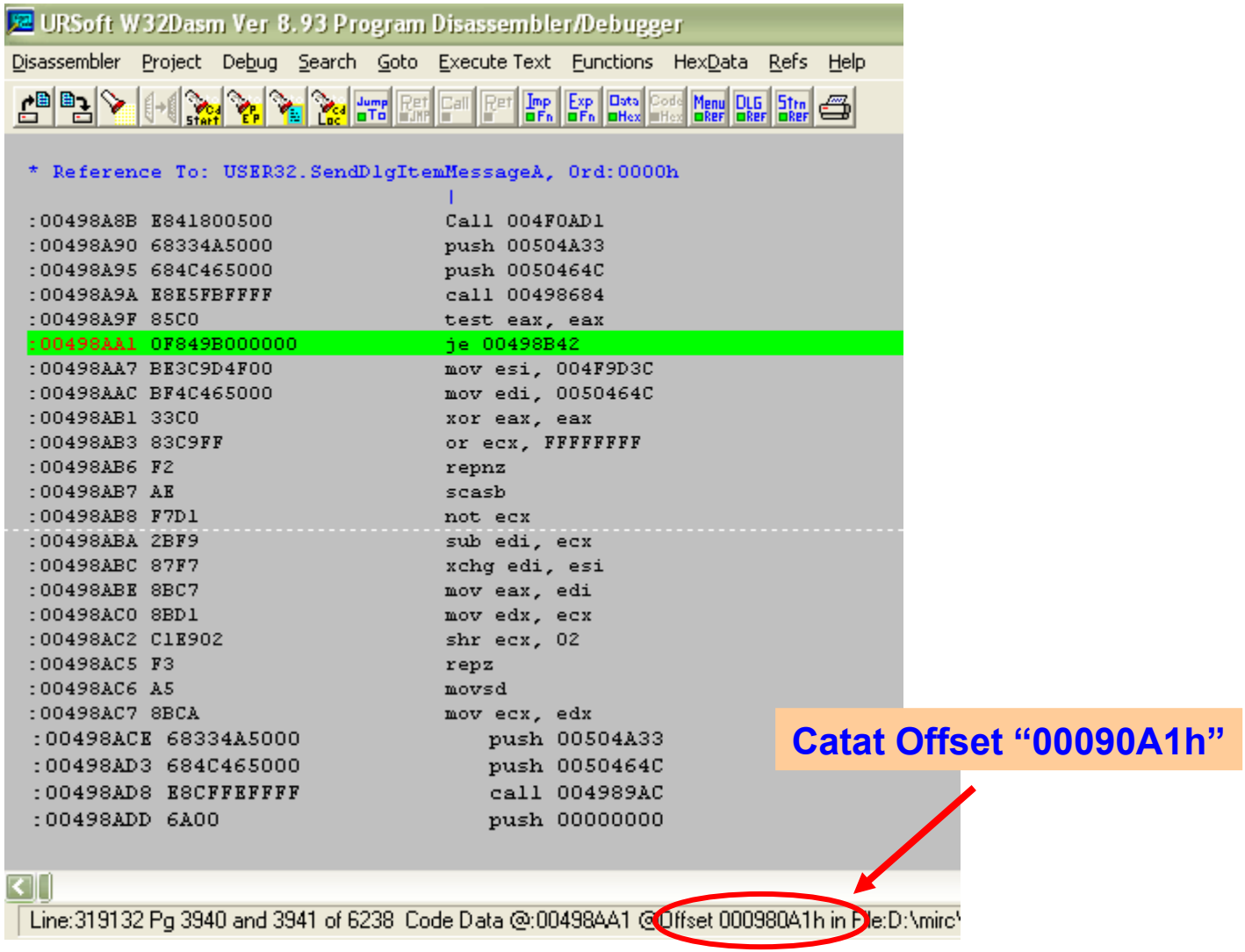
Kita dapat menggunakan

"41494048424A"

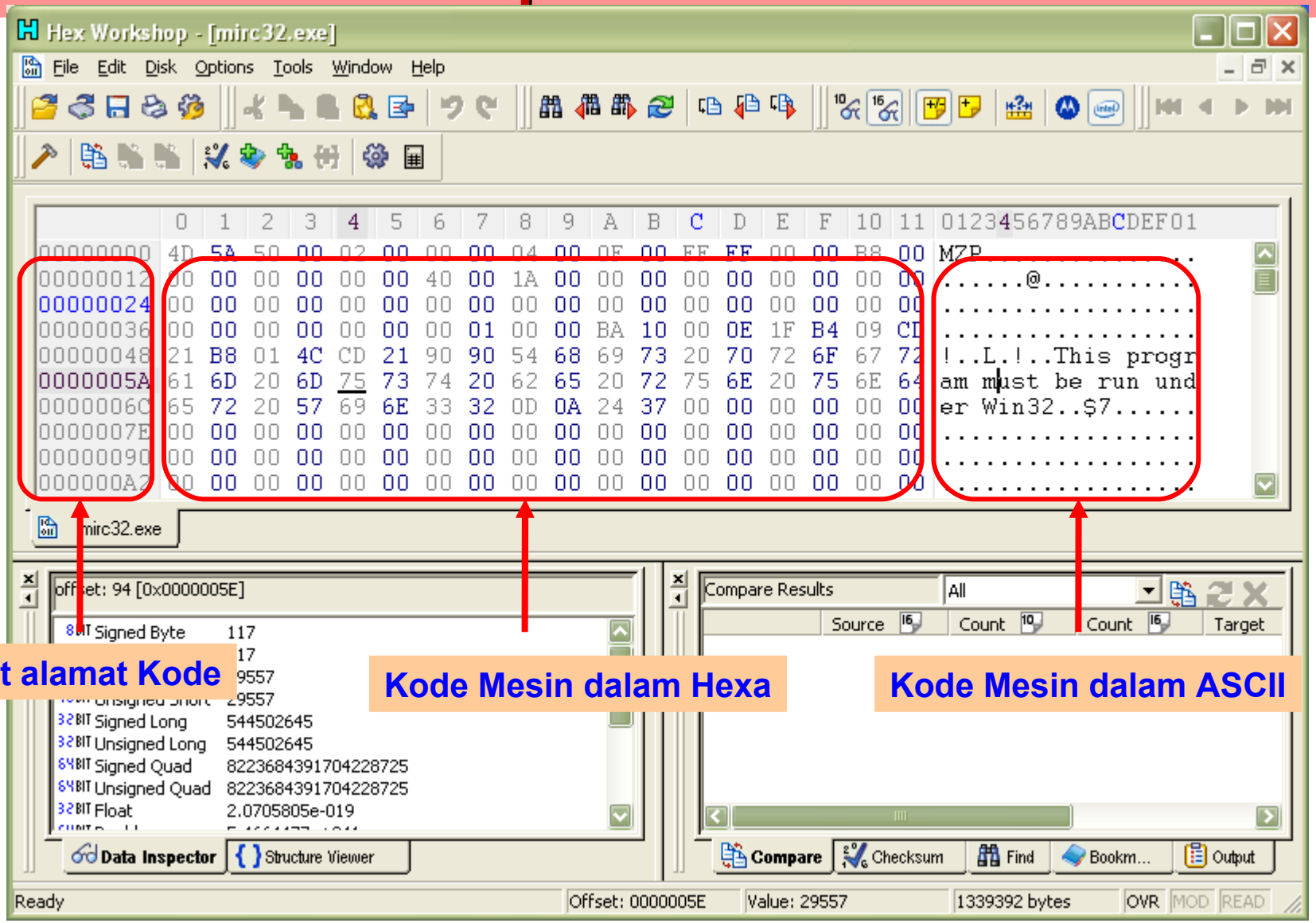
"434B40414849“

atau "9090424A9090” << saya coba ini

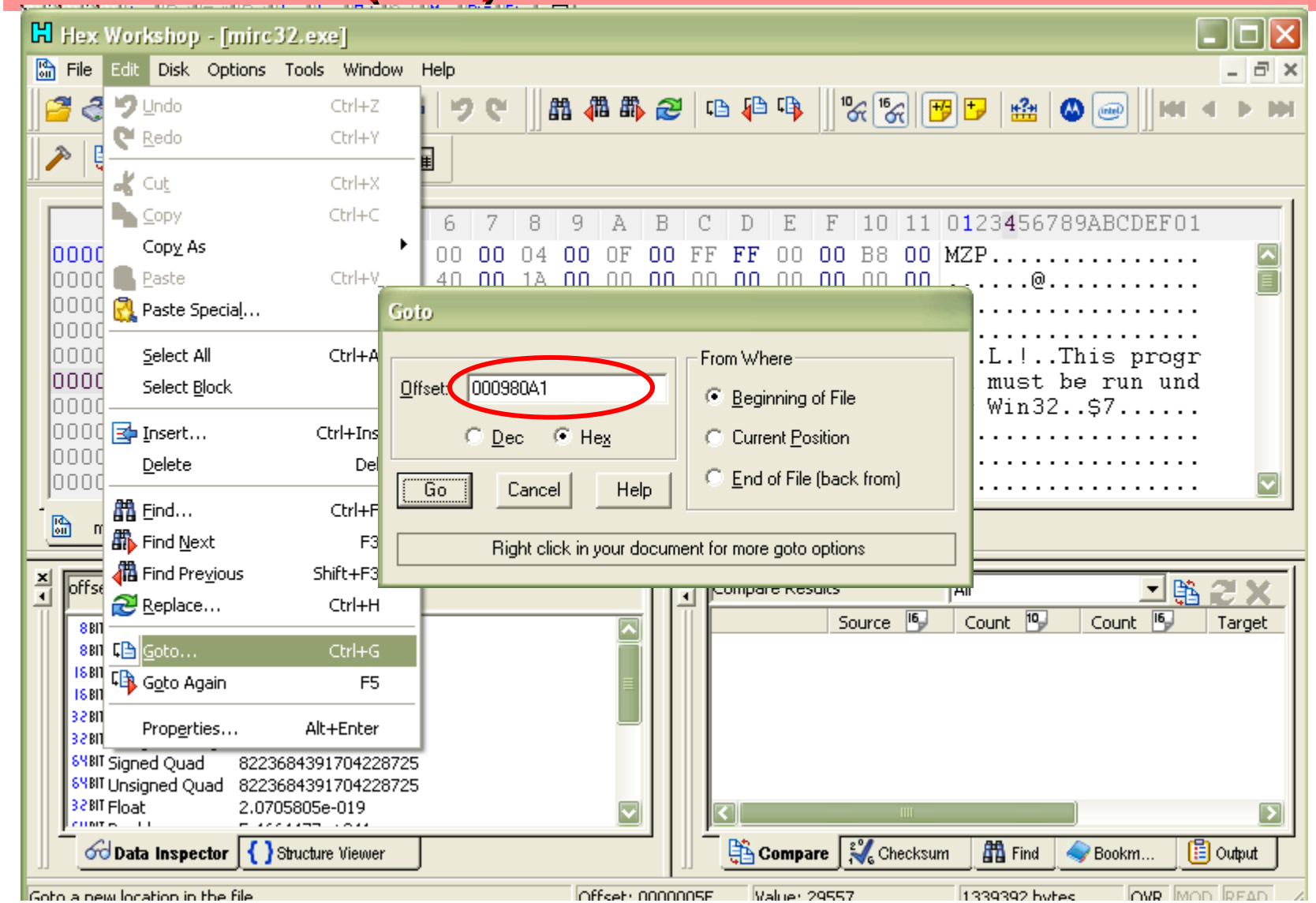
Kembali ke W32DASM, double klik alamat 00498AA1



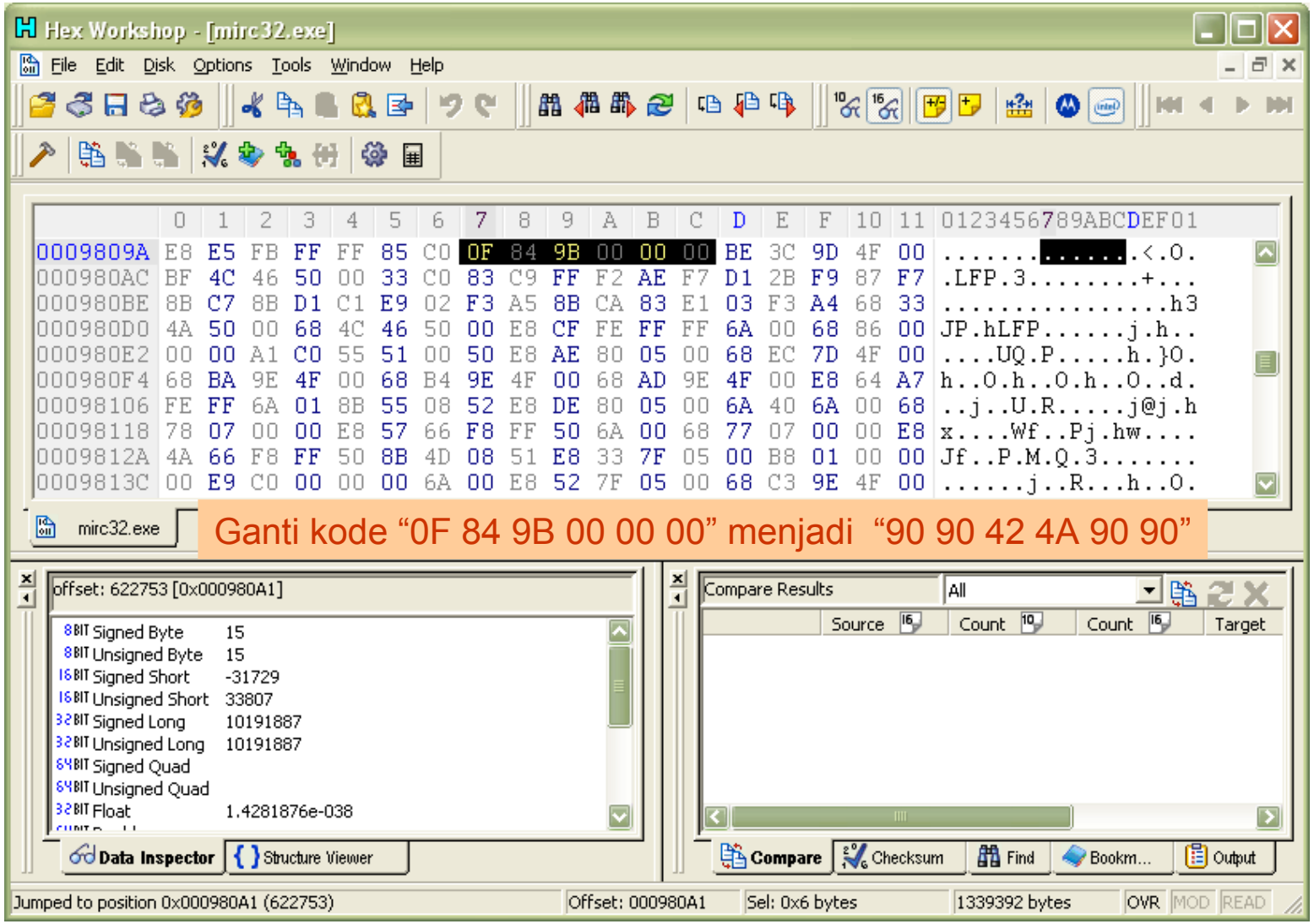
Run Hex Workshop dan load mirc32.exe



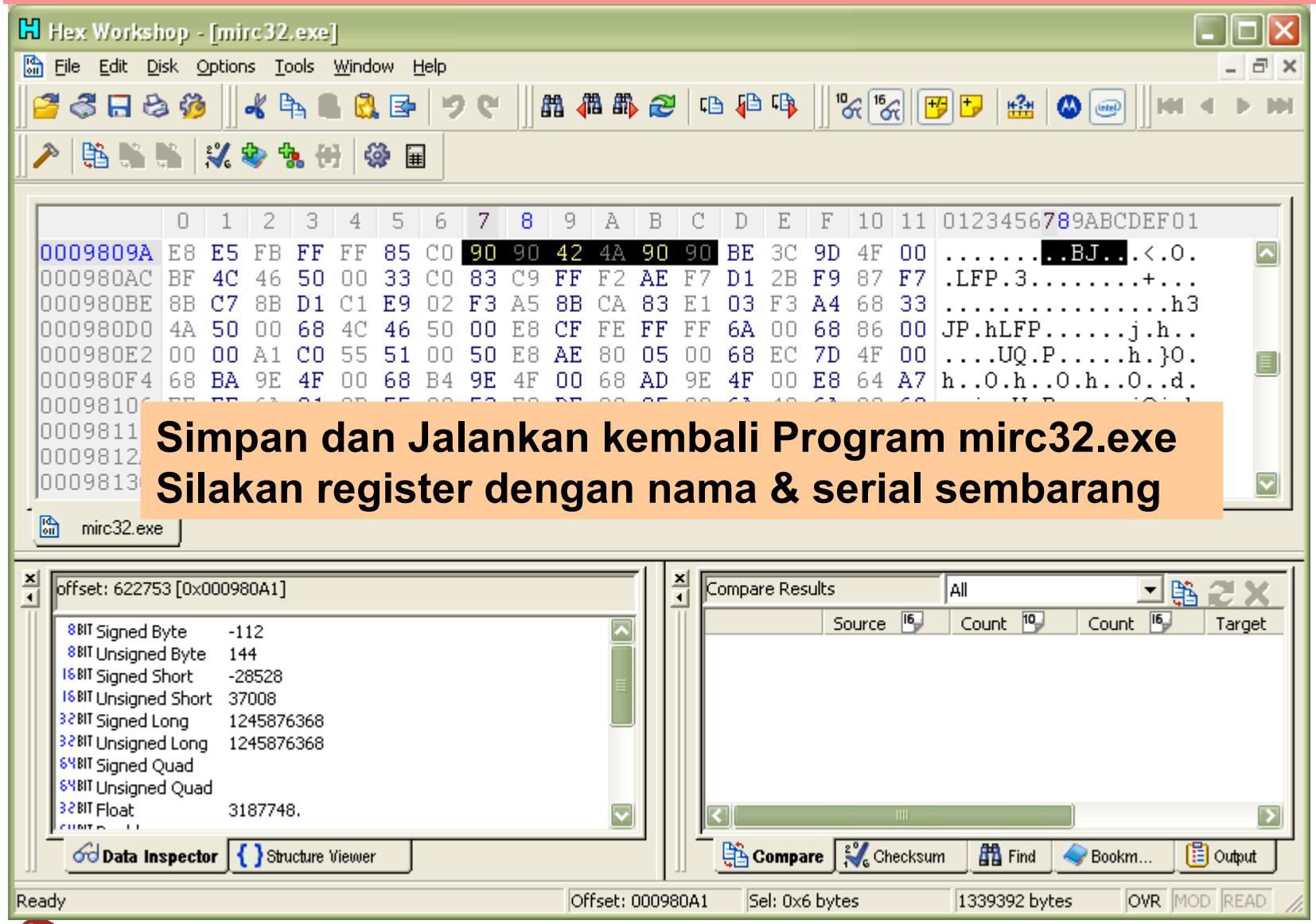
Goto Offset (hex) 00980A1



Menghasilkan :



Hasil :



Simpan dan Jalankan kembali Program mirc32.exe
Silakan register dengan nama & serial sembarang

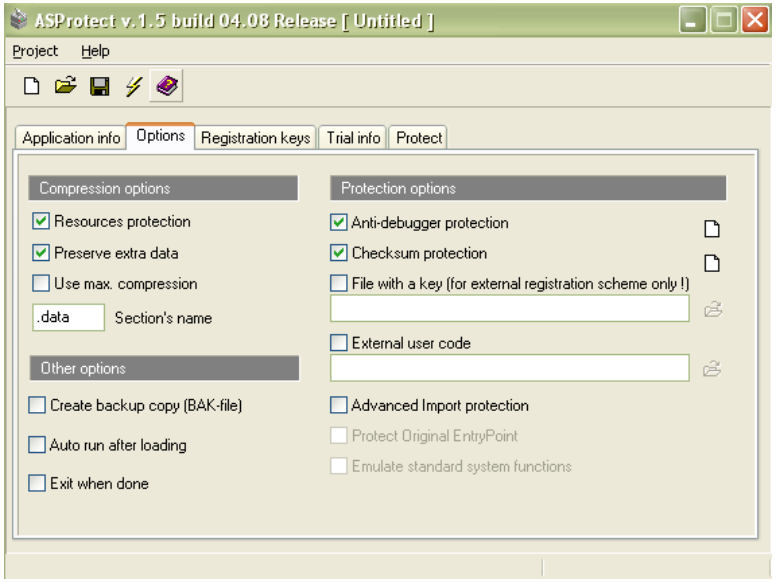
Registration Success, Lihat About

Demo Proteksi Sofware

- Tools ASProtect (Demo Version)

  (Download from http://www.aspack.com)

  - Trial/Shareware
  - Registration Key
  - Checksum Protection
  - Resource Protection
  - Antidebug Protection



Thanks - Questions

- Referece
  - http://ref.x86asm.net/ Assembly Opcode & Instruction
  - http://www.woodmann.com Forum for Reverse Engineering
  - http://www.woodmann.com/krobar/ Sofware Protection & Cracking Tutorial
  - http://www.woodmann.com/crackz/ CrackZ's Reverse Engineering