

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 generate 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

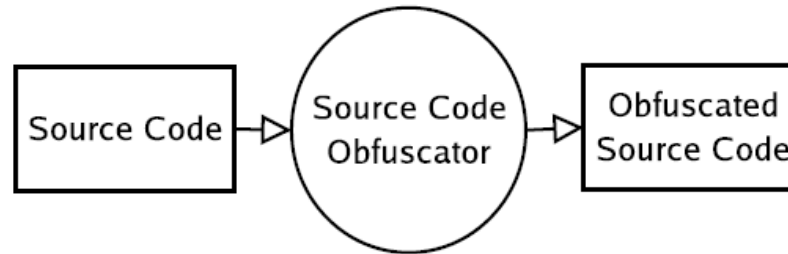
❏ Hard Code

- ◆ **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);  
}
```

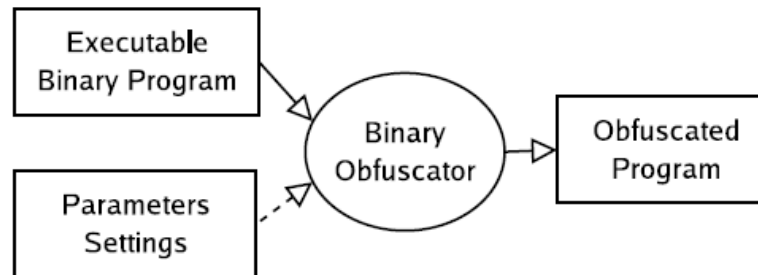
Obfuscated Source Code

```
for (O79=0;O79<l6x.length;O79++){var O63=(170)?document.layers  
["nsM\151\156u\164\145s"+O79].ieMinutes[O79].style;O63.top=161[O79]+O76+(O79*O75)*Math  
.sin(O51)+173;O63.left=175[O79]+177+(O79*176)*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.

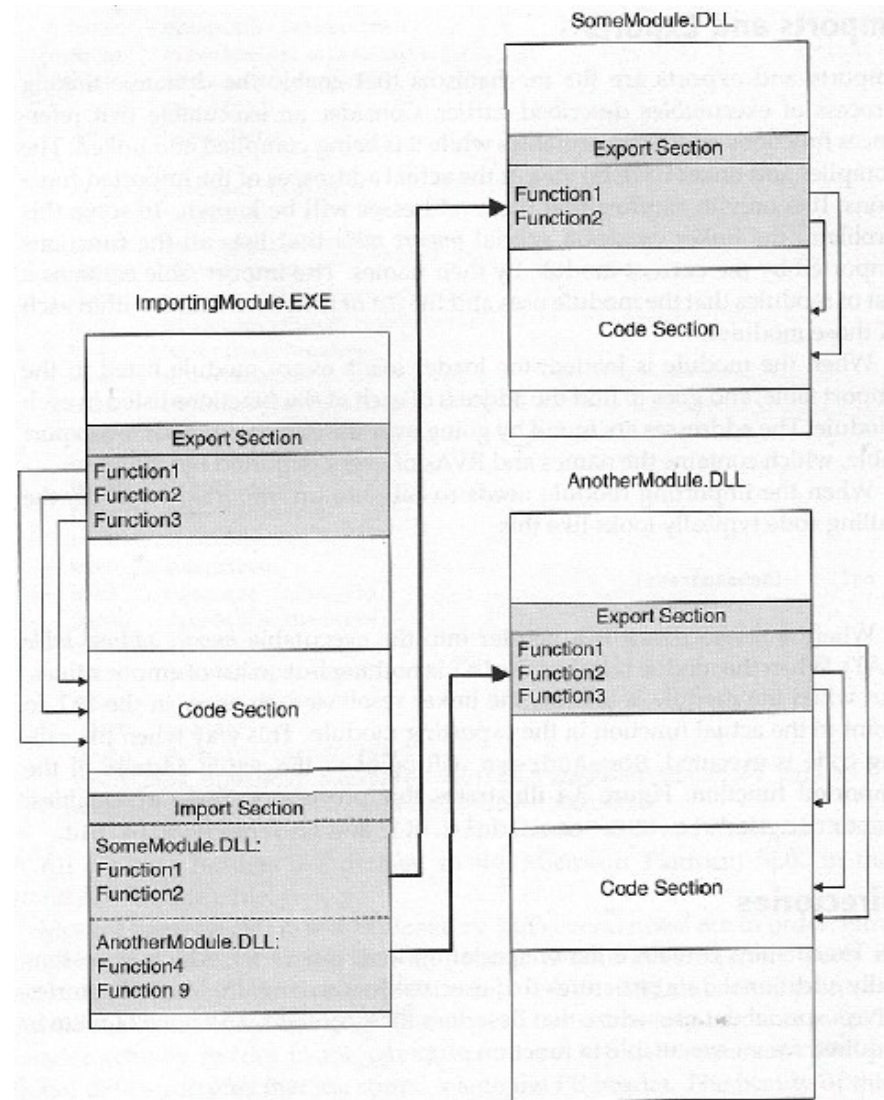


Cara Kerja Cracker

- ❑ 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.

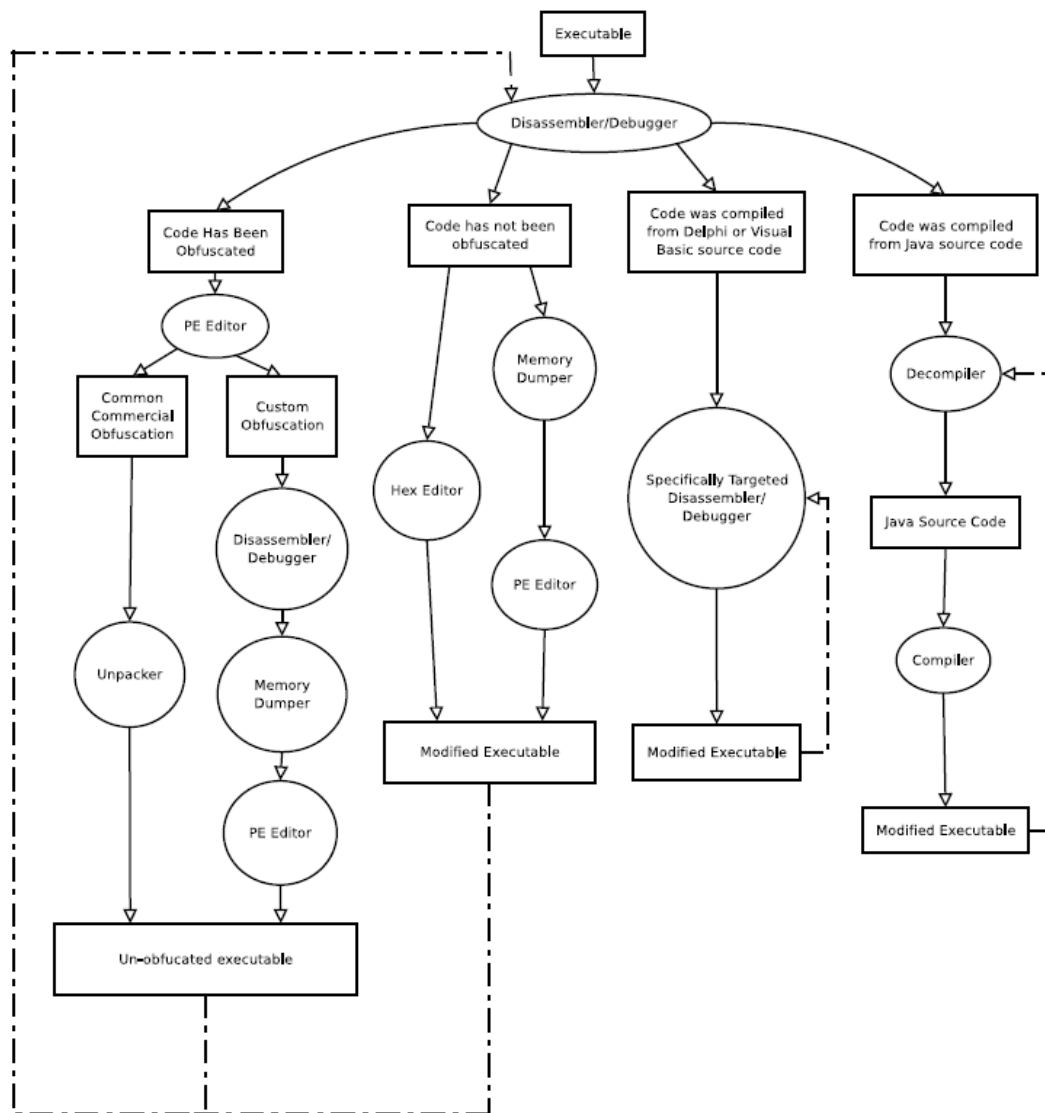


Reverse Engineering

❏ 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



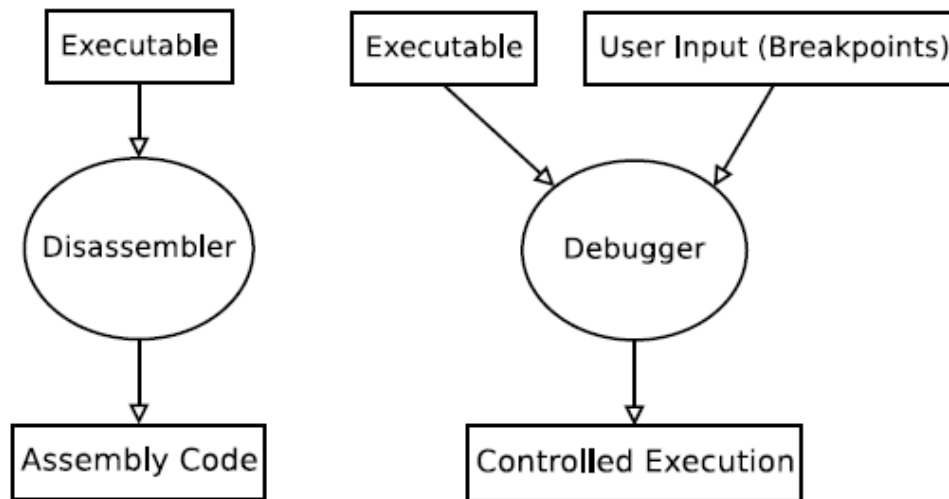
Reverse Engineering Tools

❏ 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 - mirc32.exe - [CPU - main thread, module mirc32]

File View Debug Plugins Options Window Help

LEMTWHC / KBR ... S

```

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 . 76 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 . E9 D6C90000 JMP mirc32.0040DA1C
00401046 . E9 970D0000 JMP mirc32.00401DE2
0040104B . 33C0 XOR EAX,EAX
0040104D . A0 58104F00 MOV AL,BYTE PTR DS:[4F1058]
00401052 . C3 RETH
00401053 . A1 6B104F00 MOV EAX,DWORD PTR DS:[4F106B]
00401058 . C3 RETH
00401059 . CC INT3
0040105A . B9 00000000 MOV ECX,0B0
0040105F . 0BC9 OR ECX,ECX
00401061 . 74 39 JE SHORT mirc32.0040109C
00401063 . 833D 63104F00 CMP DWORD PTR DS:[4F1063],0
0040106A . 73 0A JNB SHORT mirc32.00401076
0040106C . B8 E2000000 MOV EAX,0E2
    
```

Registers (FPU)

```

EAX 00000000
ECX 0156FFB0
EDX 7C90E4F4 ntdll.KiFastSystemCallRet
EBX 7FFDE000
ESP 0156FFC4
EBP 0156FFFF
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 00401000 mirc32.<ModuleEntryPoint>
    
```

Arg1 = 00000000
mirc32.00401D47

pModule = NULL
GetModuleHandleA

DS:[004F1063]=00000000
EAX=00000000

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	puright
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 00 18 50 00	+P.+P.
004F1038	90 18 50 00 01 00 00 00	+P.0...
004F1040	00 00 00 00 98 37 40 00	...7J.
004F1048	04 20 42 00 2C 20 42 00	* B. B.
004F1050	00 00 00 00 BC 00 50 00	...P.
004F1058	00 EC 10 50 00 84 11 50	...P.34P
004F1060	00 00 01 00 00 00 00 00	...0....
004F1068	00 00 00 00 00 00 00 00
004F1070	00 00 00 00 00 00 00 00

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

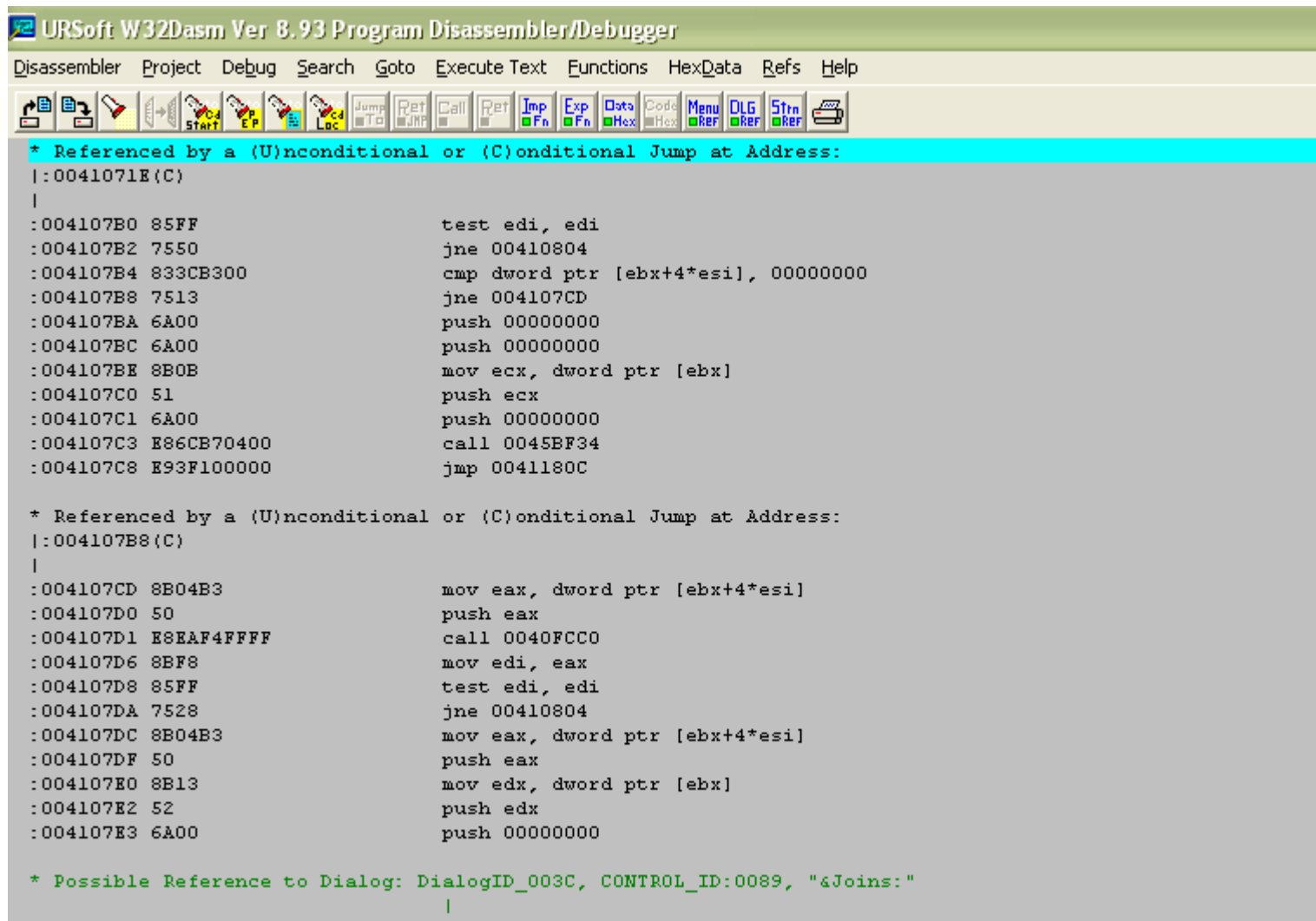
```
.text:0040480A      jnz     short loc_404831
.text:0040480C      cmp     byte_4E6628, bl
.text:00404812      jnz     short loc_404831
.text:00404814      call   CheckIfLicensed
.text:00404819      test   al, al
.text:0040481B      jk     short loc_404826
.text:0040481D      mov    byte_4E5A90, 1
.text:00404824      jmp    short loc_40483F
.text:00404826      ; -----
.text:00404826      loc_404826:
.text:00404826      call   ShowNagScreen      ; CODE XREF: sub_40473A+E1↑j
.text:0040482B      test   al, al
.text:0040482D      jnz   short loc_40483F
.text:0040482F      jmp    short loc_404841
.text:00404831      ; -----
.text:00404831      loc_404831:
.text:00404831      ; 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



The screenshot displays the W32Dasm interface with the following assembly code and references:

```
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

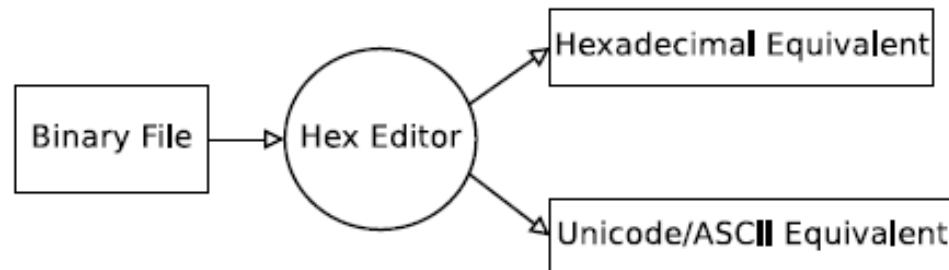
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004107B8(C)
|
:004107CD 8B04B3       mov eax, dword ptr [ebx+4*esi]
:004107D0 50           push eax
:004107D1 E8EAF4FFFF   call 0040FCCC
:004107D6 8BF8        mov edi, eax
:004107D8 85FF        test edi, edi
:004107DA 7528        jne 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 perubahan sebuah binary program melalui representasi hexadesimal suatu 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

Hex Workshop - [mirc32.exe]

File Edit Disk Options Tools Window Help

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 0123456789ABCDEF012345

00000000	4D	5A	50	00	02	00	00	00	04	00	0F	00	FF	FF	00	00	B8	00	00	00	00	00	MZP.....
00000016	00	00	40	00	1A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	..@.....
0000002C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	BA	10		
00000042	00	0E	1F	B4	09	CD	21	B8	01	4C	CD	21	90	90	54	68	69	73	20	70	72	6F!..L!..This pro
00000058	67	72	61	6D	20	6D	75	73	74	20	62	65	20	72	75	6E	20	75	6E	64	65	72	gram must be run under
0000006E	20	57	69	6E	33	32	0D	0A	24	37	00	00	00	00	00	00	00	00	00	00	00	00	Win32..\$7.....
00000084	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000009A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000DC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000F2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	09PE..L...
00000108	42	28	45	82	00	00	00	00	00	00	00	00	E0	00	8E	81	0B	01	02	19	00	00	B(E.....
0000011E	0F	00	00	5E	05	00	00	00	00	00	00	10	00	00	00	10	00	00	00	10	0F	00	...^.....
00000134	00	00	40	00	00	10	00	00	00	02	00	00	01	00	00	00	00	00	00	00	04	00	..@.....

mirc32.exe

offset: 72 [0x00000048]

- 8 BIT Signed Byte 33
- 8 BIT Unsigned Byte 33
- 16 BIT Signed Short -18399
- 16 BIT Unsigned Short 47137
- 32 BIT Signed Long 1275181089
- 32 BIT Unsigned Long 1275181089
- 64 BIT Signed Quad -8029880969974400991
- 64 BIT Unsigned Quad 10416863103735150625
- 32 BIT Float 34005124

Compare Results

Source	Count	Count	Target	Count
--------	-------	-------	--------	-------

Ready Offset: 00000048 Value: -18399 1339392 bytes OVR MOD READ

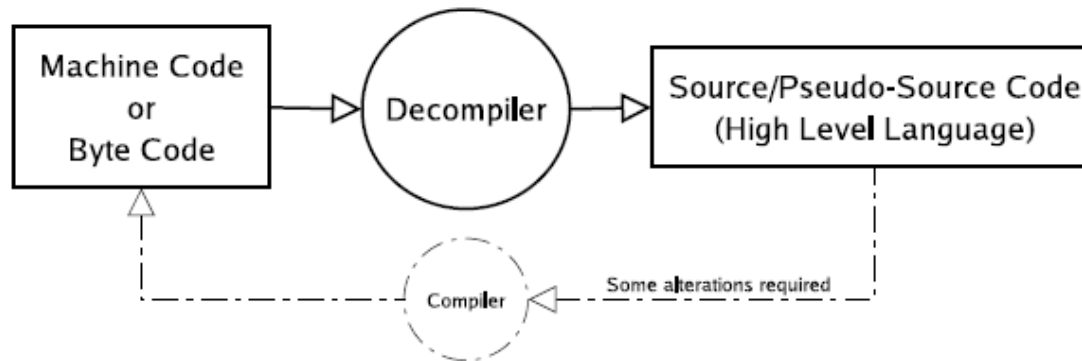
WinHEX

The screenshot shows the WinHex application window titled "WinHex - [OLLYDBG.EXE]". The main window is a hex editor displaying data from "OLLYDBG.EXE". A "Find Text" dialog box is open in the center, with the search string "password" entered. The dialog box has several options: "Match case" (unchecked), "Unicode character set" (unchecked), "Use as wildcard" (checked), "Whole words only" (unchecked), "Search: All" (selected), "Cond: offset mod 512 = 0" (unchecked), "Search in block only" (unchecked), "Search in all open windows" (checked), "Archive occurrence positions" (unchecked), and "Ignore read errors" (unchecked). The main window shows hex data in columns 0-15 and its ASCII interpretation. A "Data Interpreter" window is also visible, showing "8Bk (z) 0", "15Bk (z) 0", and "32Bk (z) 0". The status bar at the bottom shows "Page 3 of 1038", "Offset: 4F7", "=0", "Block:", "n/a", "Size:", "n/a".

Reverse Engineering Tools

❏ Decompiler

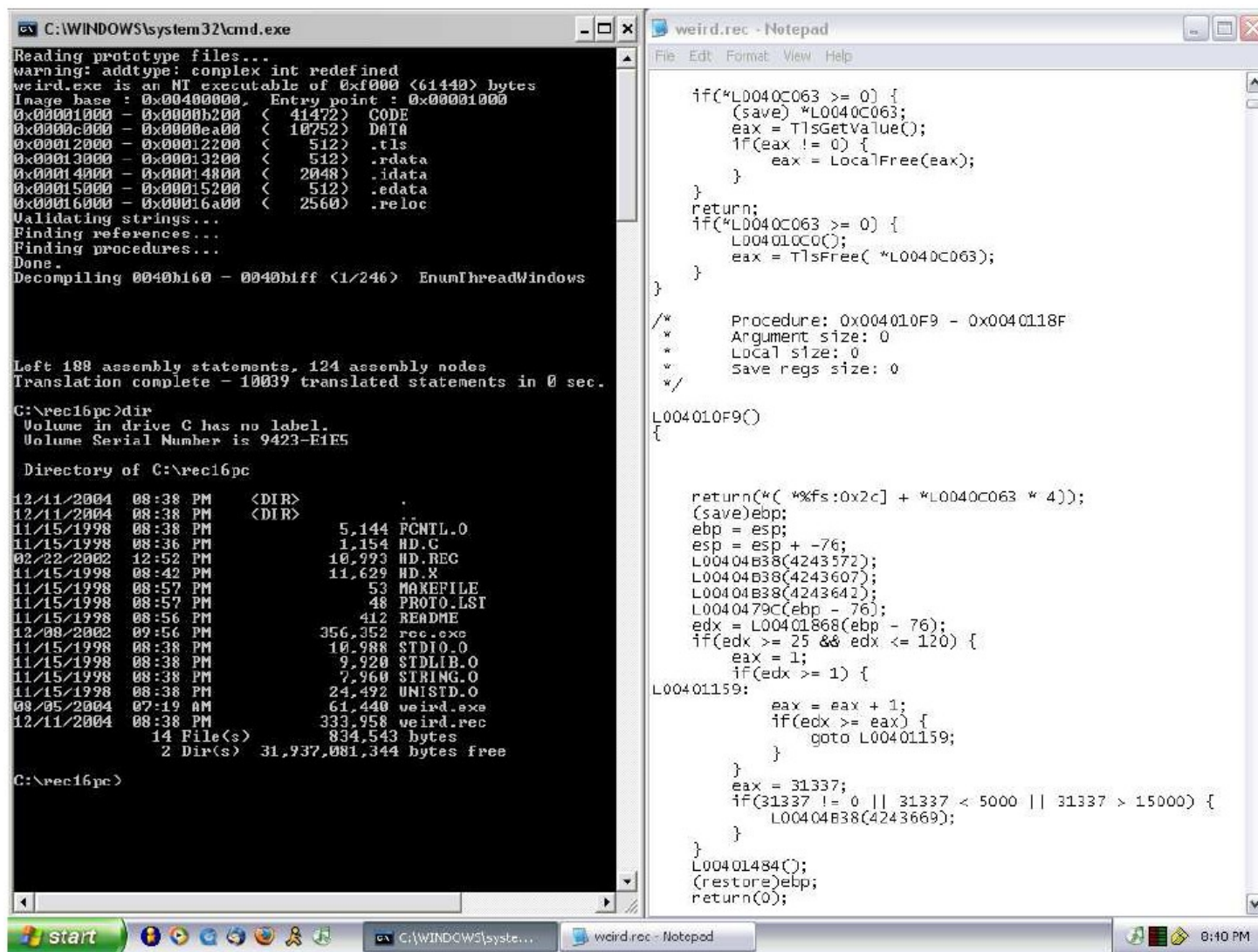
- ◆ **Mentranslasikan program Executable menjadi Source Code**



◆ Contoh :

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

C Decompiler : REC



```
C:\WINDOWS\system32\cmd.exe
Reading prototype files...
warning: addtype: complex int redefined
weird.exe is an NT executable of 0xf000 <61440> bytes
Image base : 0x00400000, Entry point : 0x00001000
0x000001000 - 0x00000b200 < 41472> CODE
0x00000c000 - 0x00000ea00 < 10752> DATA
0x000012000 - 0x000012200 < 512> .tls
0x000013000 - 0x000013200 < 512> .rdata
0x000014000 - 0x000014800 < 2048> .idata
0x000015000 - 0x000015200 < 512> .edata
0x000016000 - 0x000016a00 < 2560> .reloc
Validating strings...
Finding references...
Finding procedures...
Done.
Decompiling 0040h160 - 0040h1ff <1/246> EnumThreadWindows

Left 188 assembly statements, 124 assembly nodes
Translation complete - 10039 translated statements in 0 sec.

C:\rec16pc>dir
Volume in drive C has no label.
Volume Serial Number is 9423-E1E5

Directory of C:\rec16pc

12/11/2004 08:38 PM <DIR> .
12/11/2004 08:38 PM <DIR> ..
11/15/1998 08:38 PM 5,144 FCNTL.O
11/15/1998 08:36 PM 1,154 HD.C
02/22/2002 12:52 PM 10,993 HD.REC
11/15/1998 08:42 PM 11,629 HD.X
11/15/1998 08:57 PM 53 MAKEFILE
11/15/1998 08:57 PM 48 PROTO.LST
11/15/1998 08:56 PM 412 README
12/08/2002 09:56 PM 356,352 rec.exe
11/15/1998 08:38 PM 10,988 STDIO.O
11/15/1998 08:38 PM 9,920 STDLIB.O
11/15/1998 08:38 PM 7,960 STRING.O
11/15/1998 08:38 PM 24,492 UNISTD.O
08/05/2004 07:19 AM 61,440 weird.exe
12/11/2004 08:38 PM 333,958 weird.rec
14 File(s) 834,543 bytes
2 Dir(s) 31,937,081,344 bytes free

C:\rec16pc>
```

```
weird.rec - Notepad
File Edit Format View Help

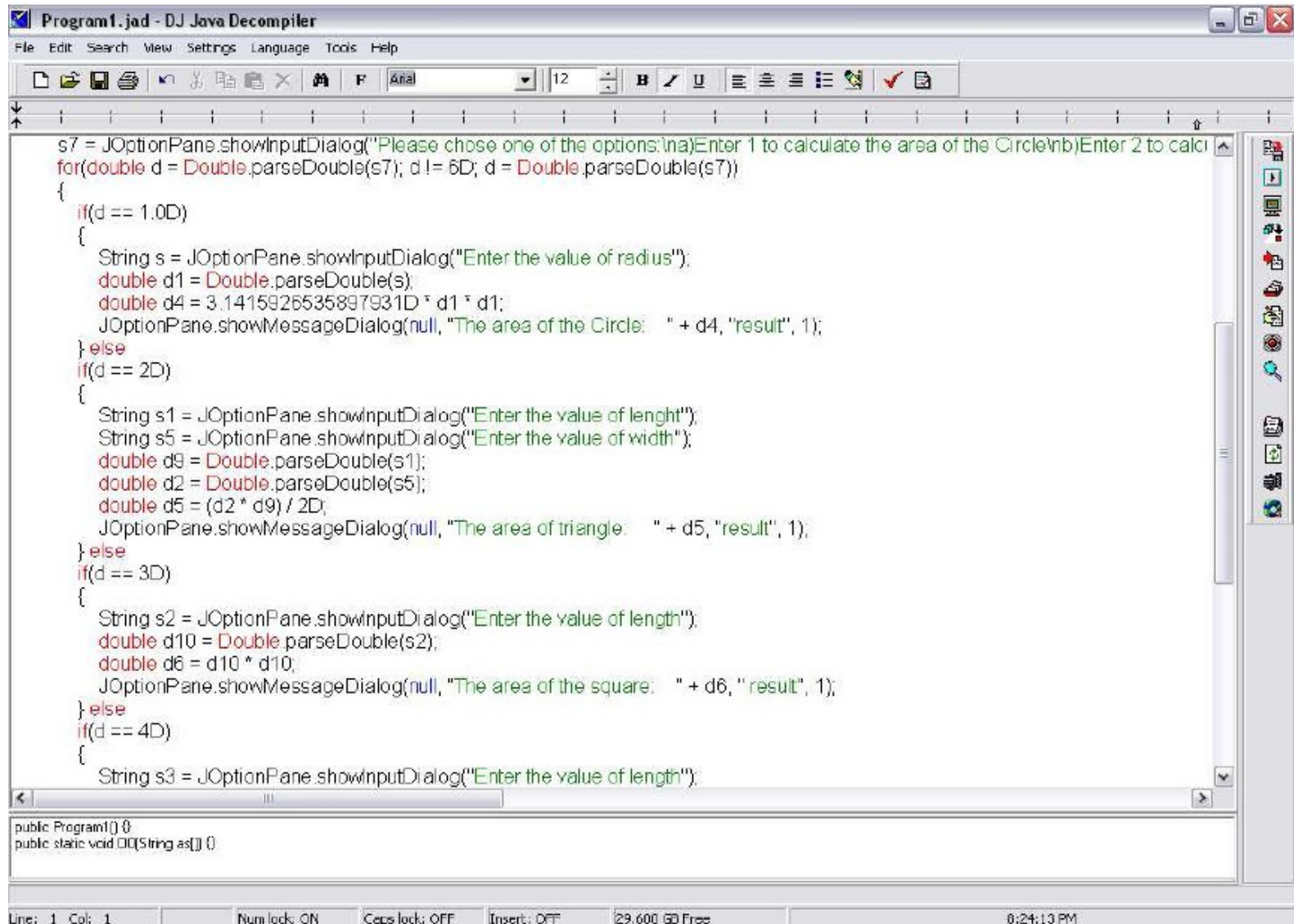
if(*L0040C063 >= 0) {
    (save) *L0040C063;
    eax = TlsGetValue();
    if(eax != 0) {
        eax = LocalFree(eax);
    }
}
return;
if(*L0040C063 >= 0) {
    L004010C0();
    eax = TlsFree( *L0040C063);
}
}

/* Procedure: 0x004010F9 - 0x0040118F
 * Argument size: 0
 * Local size: 0
 * Save regs size: 0
 */

L004010F9()
{

    return(*( *%fs:0x2c] + *L0040C063 * 4));
    (save)ebp;
    ebp = esp;
    esp = esp + -76;
    L00404B38(4243572);
    L00404B38(4243607);
    L00404B38(4243641);
    L0040479C(ebp - 76);
    edx = L00401868(ebp - 76);
    if(edx >= 25 && edx <= 120) {
        eax = 1;
        if(edx >= 1) {
L00401159:
            eax = eax + 1;
            if(edx >= eax) {
                goto L00401159;
            }
        }
        eax = 31337;
        if(31337 != 0 || 31337 < 5000 || 31337 > 15000) {
            L00404B38(4243669);
        }
    }
    L00401484();
    (restore)ebp;
    return(0);
}
```


Java Decompiler: DJ



The screenshot shows the DJ Java Decompiler interface. The main text area contains the following decompiled code:

```
s7 = JOptionPane.showInputDialog("Please chose one of the options:\na)Enter 1 to calculate the area of the Circle\nb)Enter 2 to calci\nfor(double d = Double.parseDouble(s7); d != 6D; d = Double.parseDouble(s7))\n{\n    if(d == 1.0D)\n    {\n        String s = JOptionPane.showInputDialog("Enter the value of radius");\n        double d1 = Double.parseDouble(s);\n        double d4 = 3.1415926535897931D * d1 * d1;\n        JOptionPane.showMessageDialog(null, "The area of the Circle:  " + d4, "result", 1);\n    }else\n    if(d == 2D)\n    {\n        String s1 = JOptionPane.showInputDialog("Enter the value of length");\n        String s5 = JOptionPane.showInputDialog("Enter the value of width");\n        double d9 = Double.parseDouble(s1);\n        double d2 = Double.parseDouble(s5);\n        double d5 = (d2 * d9) / 2D;\n        JOptionPane.showMessageDialog(null, "The area of triangle:  " + d5, "result", 1);\n    }else\n    if(d == 3D)\n    {\n        String s2 = JOptionPane.showInputDialog("Enter the value of length");\n        double d10 = Double.parseDouble(s2);\n        double d6 = d10 * d10;\n        JOptionPane.showMessageDialog(null, "The area of the square:  " + d6, "result", 1);\n    }else\n    if(d == 4D)\n    {\n        String s3 = JOptionPane.showInputDialog("Enter the value of length");\n
```

At the bottom of the window, the following code is visible:

```
public Program1() {\npublic static void OO(String as[]) {}
```

The status bar at the bottom of the window displays: Line: 1 Col: 1 Num lock: ON Caps lock: OFF Insert: OFF 29,600 GB Free 8:24:13 PM

C# Decompiler : Salamander

Original Function Source Code

```
public static void Main()
{
    int x, y;
    for (x = 1; x <= 10; x ++)
    {
        for (y = 1; y <= 10; y++)
        {
            Console.Write("{0 } ", x*y);
        }
        Console.WriteLine("");
    }
}
```

Compilation

IL
Executable
Binary

Decompilation

Salamander Decompiler Output

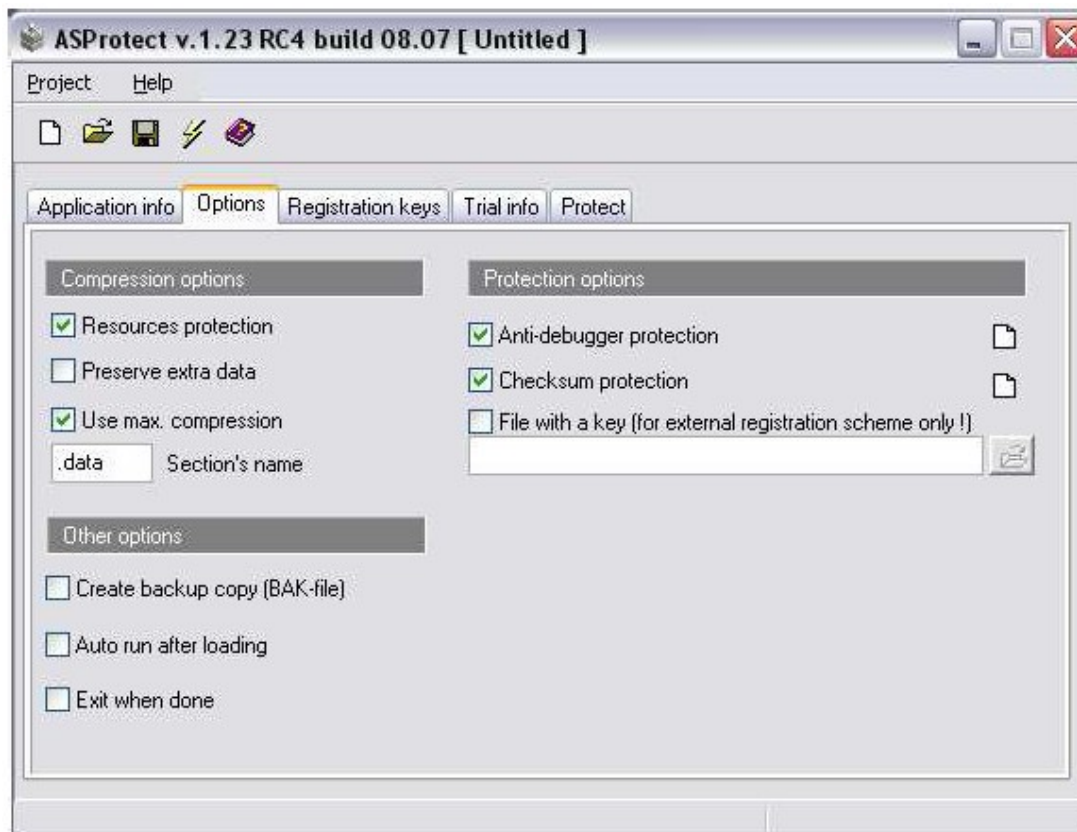
```
public static void Main()
{
    for (int i = 1; i <= 10; i++)
    {
        for (int j = 1; j <= 10; j++)
        {
            Console.Write("{0 } ", (i * j));
        }
        Console.WriteLine("");
    }
}
```

Reverse Engineering Tools

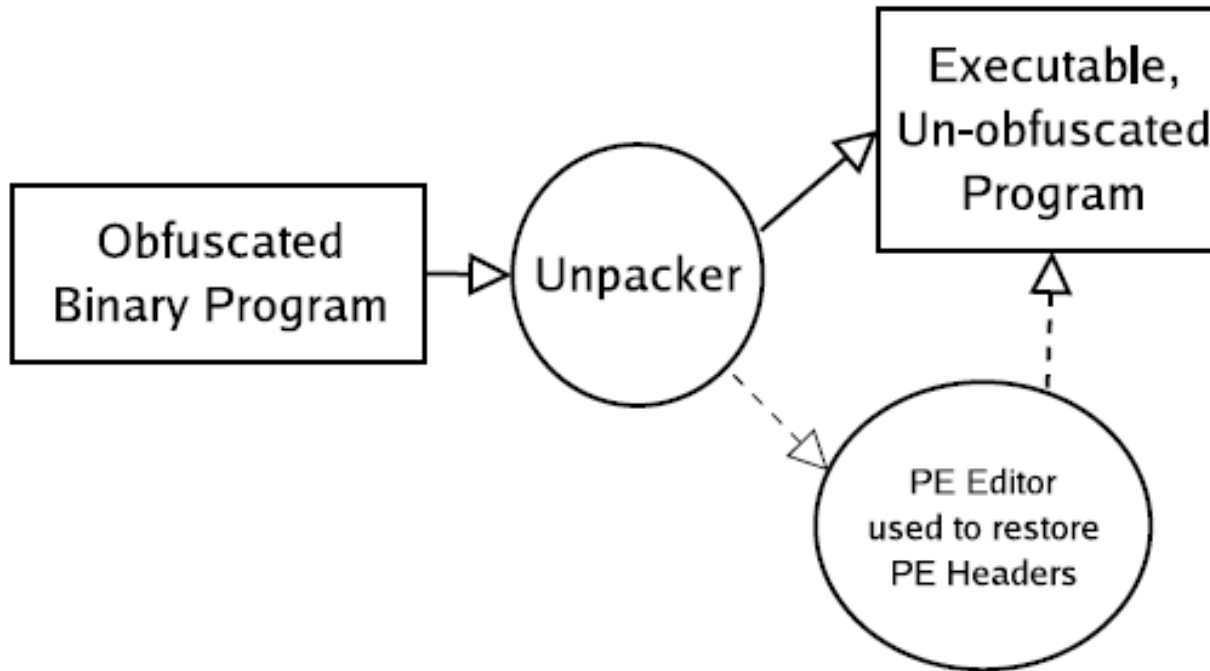
- ❑ Aplikasi yang berhubungan spt code obfuscators, PE editors, memory dumpers dan unpacker
 - ◆ Code Obfuscators : aplikasi yang mempersulit suatu source code atau binary untuk dipahami atau dibaca.
 - ◆ PE Editors : Mengextract headers dari File PE dan kemudian mempermudah PE File untuk di edit dibandingkan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

Code Obfuscators

ASProtect



Unpackers



PEditor

The screenshot displays the PEditor 1.7.7 application interface. The main window shows file information for a PE file located at 'scouting... C:\Documents and Settings\Roy\Desktop\PE\CENTargets\reverse.exe'. The File Info section includes fields for Entry Point (00001000), Image Base (00400000), Base of Code (00001000), Base of Data (00002000), Size of Image (00005000), Size of Headers (00000400), Section Alignment (00001000), File Alignment (00002000), and Subsystem (0002). The File Header section includes Machine Type (014C), Number of Sections (0004), Time Date Stamp (3C177E42), Pointer to Symbol Table (00000000), Number of Symbols (00000000), Size of Optional Header (00E0), and Characteristics (010F). The Directory Table Viewer window shows a table of directory entries with RVA and Size columns. The Section Table Viewer window shows a table of sections with Virtual Size, Virtual Offset, Raw Size, and Raw Offset columns. The Import Table Viewer window shows a table of import entries with DIName, OriginalFirstThunk, TimeDateStamp, ForwarderChain, Name, and FirstThunk columns.

File Info:

- Entry Point: 00001000
- Image Base: 00400000
- Base of Code: 00001000
- Base of Data: 00002000
- Size of Image: 00005000
- Size of Headers: 00000400
- Section Alignment: 00001000
- File Alignment: 00002000
- Subsystem: 0002

File Header:

- Machine Type: 014C
- Number of Sections: 0004
- Time Date Stamp: 3C177E42
- Pointer to Symbol Table: 00000000
- Number of Symbols: 00000000
- Size of Optional Header: 00E0
- Characteristics: 010F

Directory Table Viewer:

	RVA	Size
Export Table:	00000000	00000000
Import Table:	00002058	0000003C
Resources:	00004000	00000340
Exceptions:	00000000	00000000
Security:	00000000	00000000
BaseReloc:	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

Section Table Viewer:

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset
.text	00000344	00001000	00000400	00000400
.pdata	00000250	00002000	00000400	00000800
.data	000002CC	00003000	00000200	00000C00
.rsrc	00000360	00004000	00000400	00000E00

Import Table Viewer:

DIName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
USER32.dll	000020A4	00000000	00000000	00002202	00002010
KERNEL32.dll	00002094	00000000	00000000	00002242	00002000

Import Table Viewer (Detailed):

RVA	Offset	Hint	Name
00002170	00000910	006D	DestroyWindow
00002142	00000942	0158	GetWindowTextA
00002154	00000954	0197	LoadCursorA
00002162	00000962	0198	LoadIconA
0000216E	0000096E	01B5	MessageBoxA
0000217C	0000097C	010D	PostQuitMessage
00002120	00000920	0094	DispatchMessageA
00002134	00000934	0128	GetMessageA
00002182	00000982	022B	SetFocus
0000218E	0000098E	0259	SetWindowTextA

Demo Cracking Software

❏ **Target : Patching mIRC**

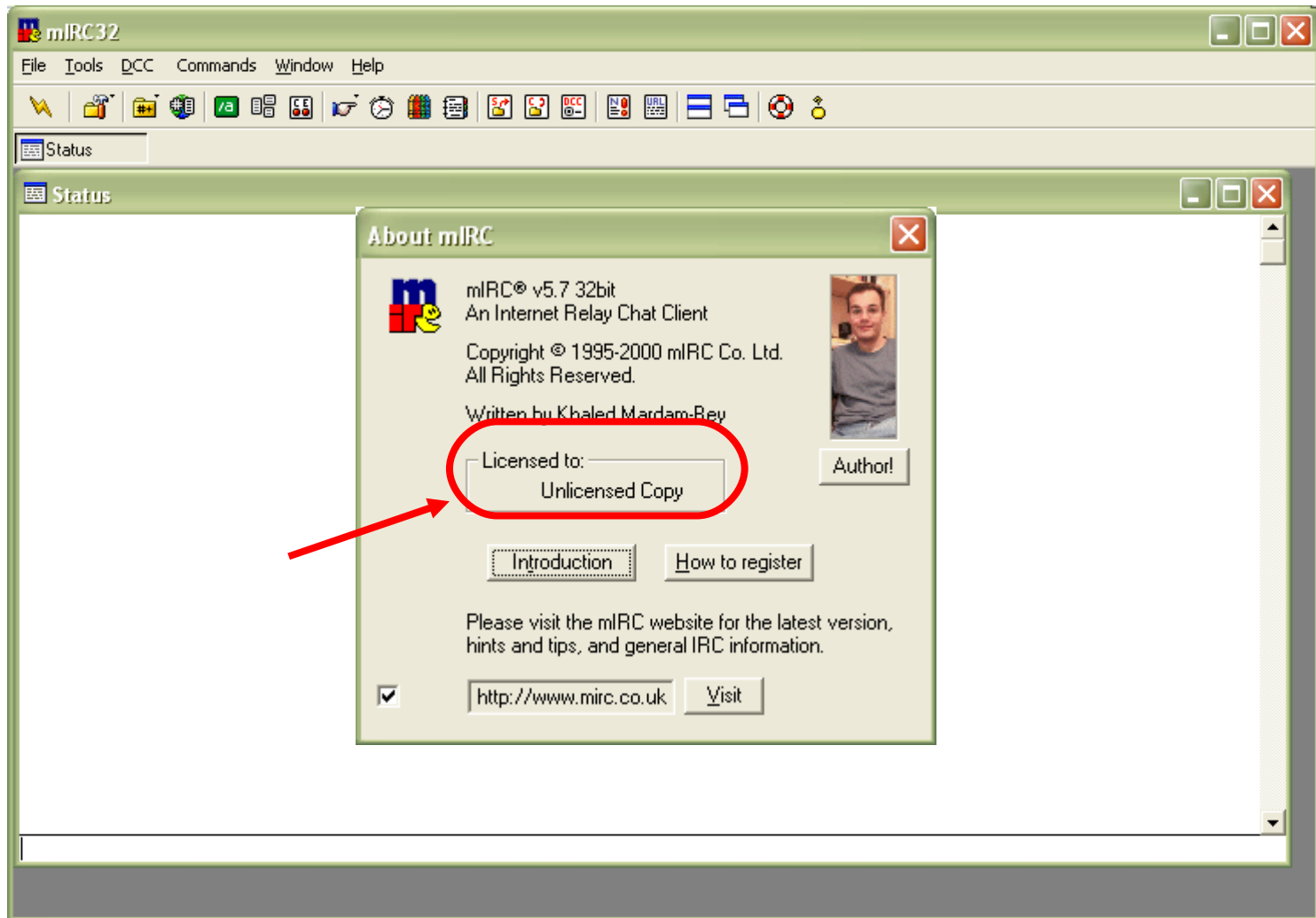
❏ **Download :**

http://www.oldversion.com/download_mIRC_

❏ **Tools : W32DASM & Hex Workshop**

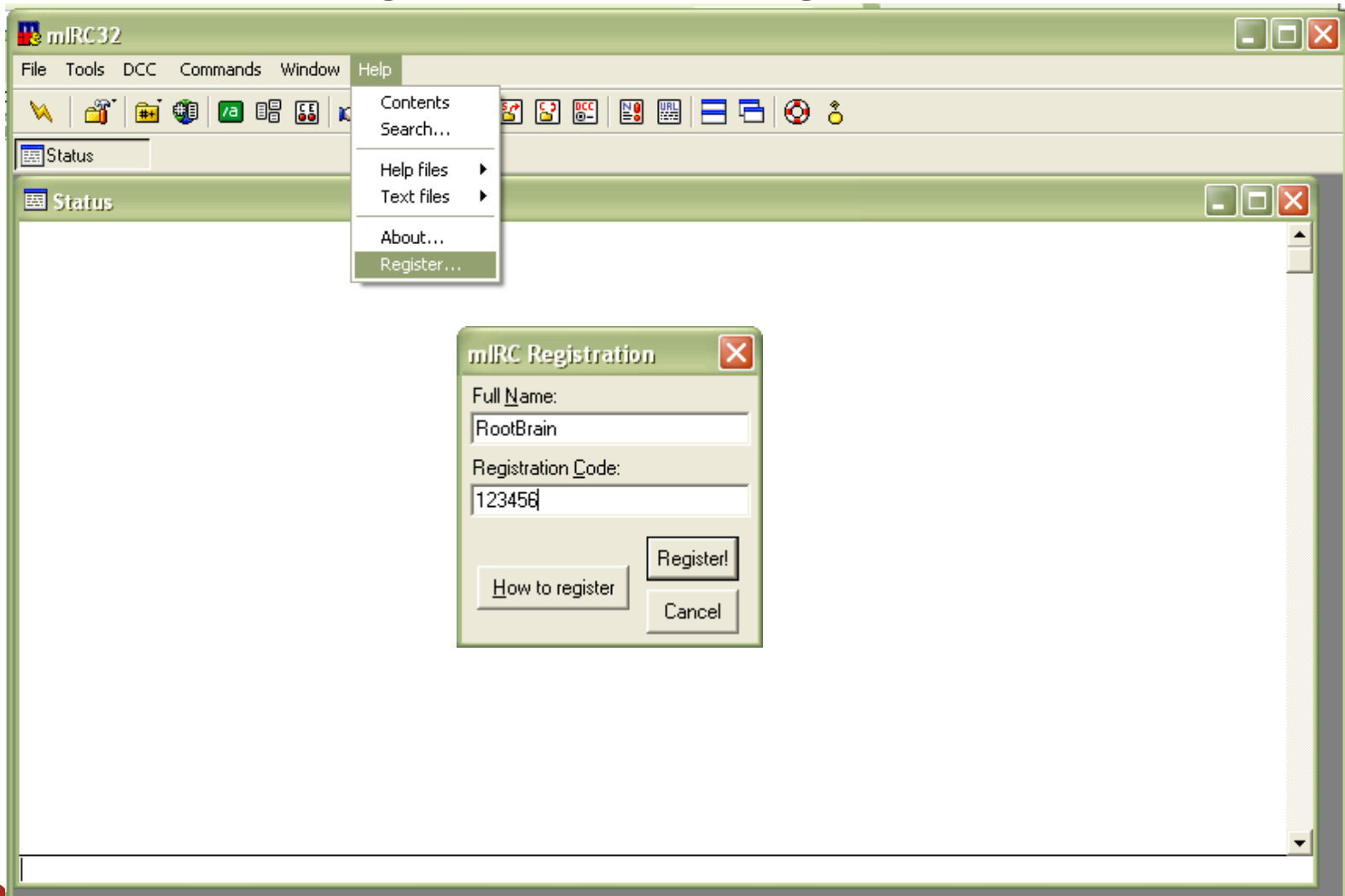
Langkah pertama

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



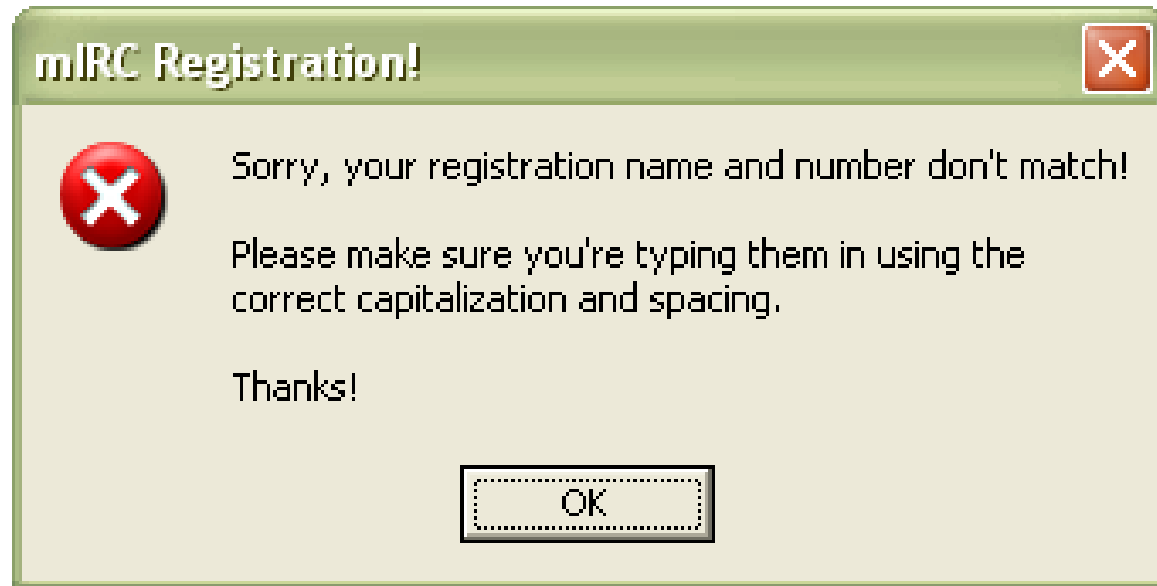
Langkah kedua

❏ 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

```
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
:004107C2 5A00 push 00000000
:004107C3 E86CE70400 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 E8EAF0FFFF 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 5A00 push 00000000

CONTROL_ID:0089, "&Joins:"
```

Alamat Memory

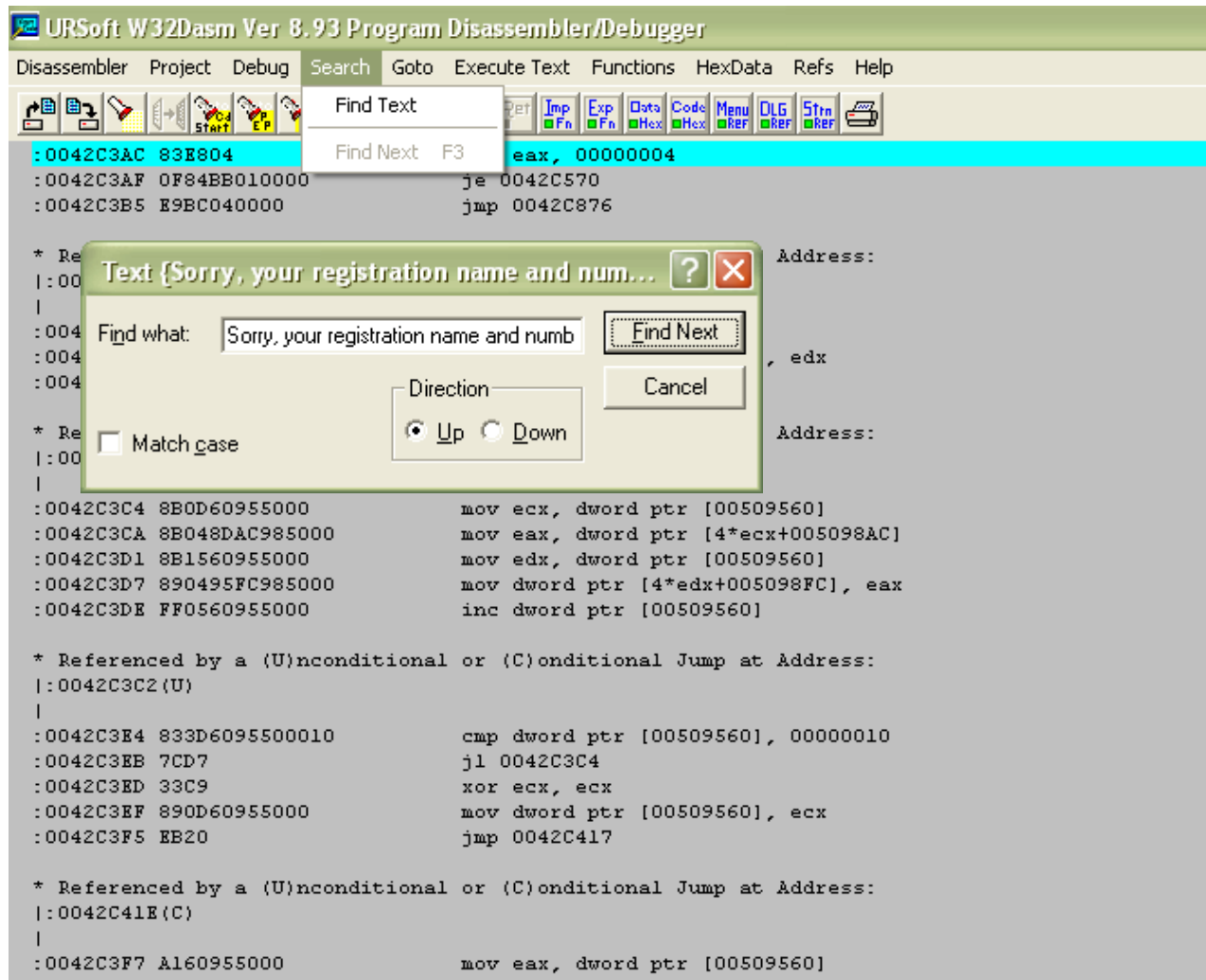
Bahasa Assembly

Kode mesin dalam format Hexadesimal



Langkah kelima

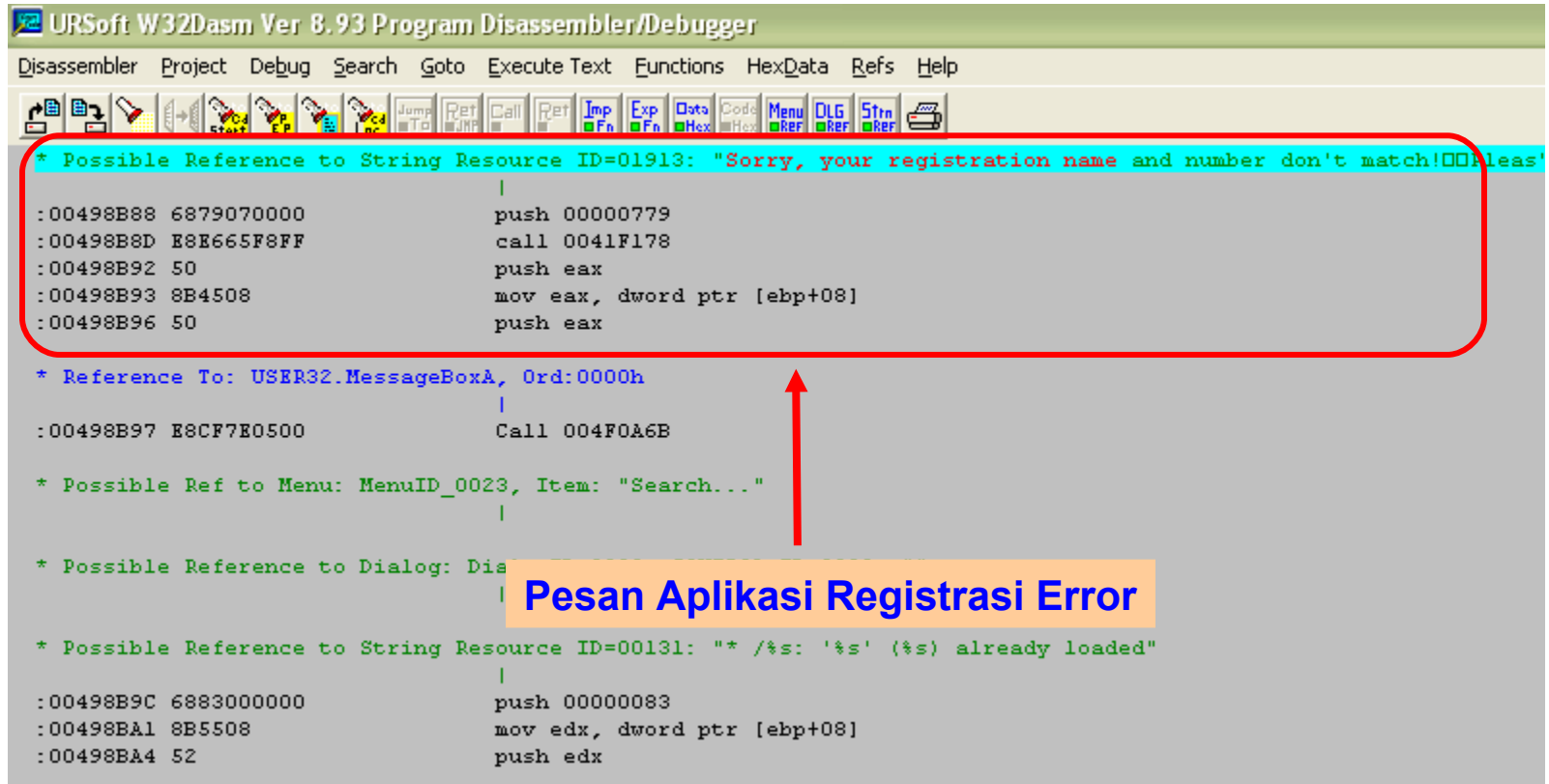
❏ Cari kata kunci pesan tadi



The screenshot shows the URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger interface. The main window displays assembly code with the instruction `mov eax, dword ptr [eax], 00000004` highlighted in cyan. A search dialog box is open, titled "Text {Sorry, your registration name and num...". The "Find what:" field contains the text "Sorry, your registration name and numb". The "Direction:" section has the "Up" radio button selected. The "Match case" checkbox is unchecked. The "Find Next" button is highlighted. The background assembly code includes instructions like `je 0042C570`, `jmp 0042C876`, `mov ecx, dword ptr [00509560]`, `mov eax, dword ptr [4*ecx+005098AC]`, `mov edx, dword ptr [00509560]`, `mov dword ptr [4*edx+005098FC], eax`, `inc dword ptr [00509560]`, `cmp dword ptr [00509560], 00000010`, `j1 0042C3C4`, `xor ecx, ecx`, `mov dword ptr [00509560], ecx`, `jmp 0042C417`, and `mov eax, dword ptr [00509560]`.

Langkah kelima

Hasil pencarian



```
URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger
Disassembler Project Debug Search Goto Execute Text Functions HexData Refs Help

* Possible Reference to String Resource ID=01913: "Sorry, your registration name and number don't match!
:00498B88 6879070000      push 00000779
:00498B8D E8E665F8FF      call 0041F178
:00498B92 50              push eax
:00498B93 8B4508         mov eax, dword ptr [ebp+08]
:00498B96 50              push eax

* Reference To: USER32.MessageBoxA, Ord:0000h
:00498B97 E8CF7E0500      Call 004F0A6B

* Possible Ref to Menu: MenuID_0023, Item: "Search..."
|
* Possible Reference to Dialog: Dis
|
* Possible Reference to String Resource ID=00131: "* /%s: '%s' (%s) already loaded"
:00498B9C 6883000000      push 00000083
:00498BA1 8B5508         mov edx, dword ptr [ebp+08]
:00498BA4 52              push edx
```

Pesan Aplikasi Registrasi Error

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

The screenshot shows the URSoft W32Dasm interface. The assembly list contains the following instructions:

```
:00498B33 E8337F0500 Call 004F0A6B
:00498B38 B801000000 mov eax, 00000001
:00498B3D E9C0000000 jmp 00498C02
```

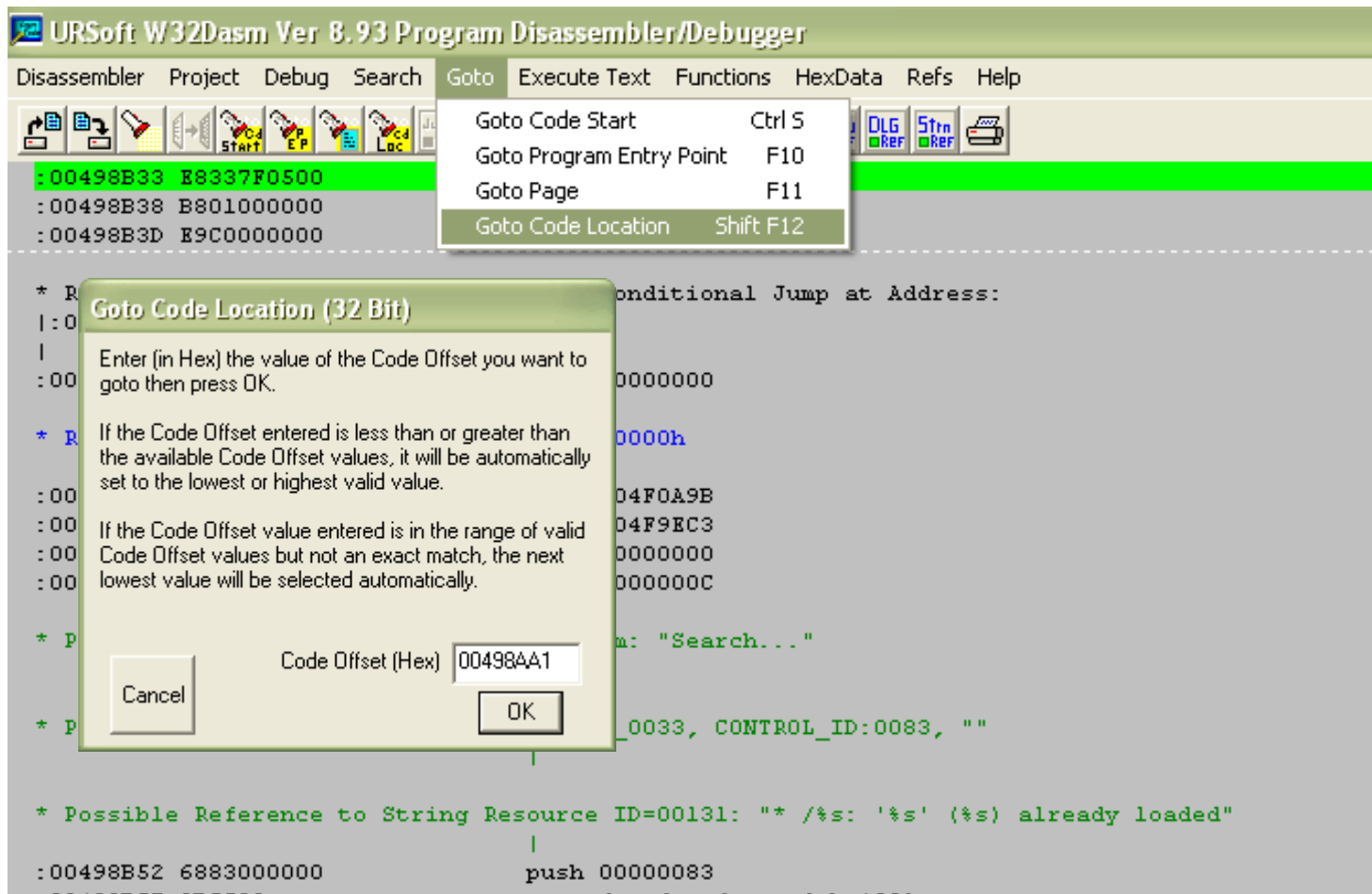
A red box highlights the message: `* Referenced by a (U)nconditional or (C)onditional Jump at Address: |:00498AA1(C)`. A red arrow points from this message to the instruction `:00498B42 6A00 push 00000000`. Below this, another message states: `* Reference To: USER32.MessageBeep, Ord:0000h`. The instruction `:00498B44 E8527F0500 Call 004F0A9B` is also visible.

Catat Alamat Instruksi Jump “00498AA1”

Alamat Jump Instruction

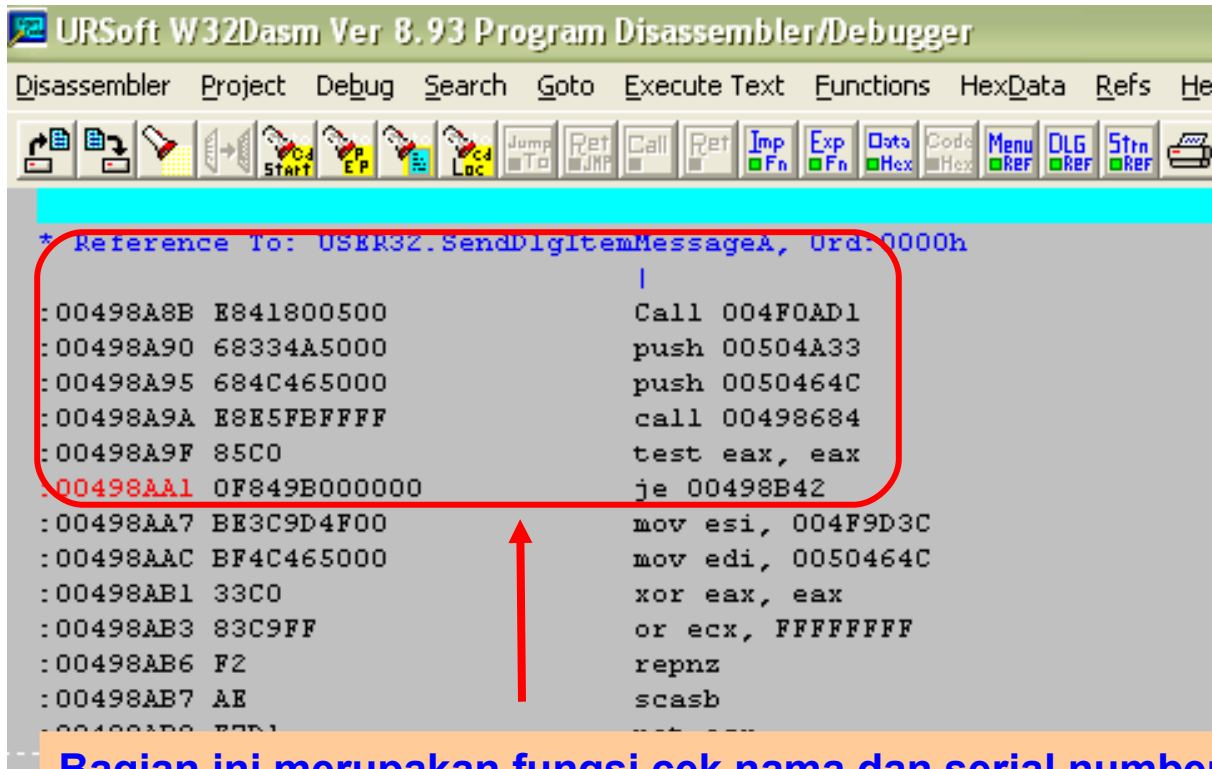
Goto Code Location

❏ Cari Alamat Instruksi Jump “00498AA1”



The screenshot shows the URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger interface. The menu bar includes Disassembler, Project, Debug, Search, Goto, Execute Text, Functions, HexData, Refs, and Help. The Goto menu is open, showing options: Goto Code Start (Ctrl S), Goto Program Entry Point (F10), Goto Page (F11), and Goto Code Location (Shift F12). The main window displays assembly code with address 00498B33 highlighted. A dialog box titled "Goto Code Location (32 Bit)" is open, prompting the user to enter a Code Offset (Hex) value. The value "00498AA1" is entered in the text box. The dialog box also contains instructions: "Enter (in Hex) the value of the Code Offset you want to goto then press OK.", "If the Code Offset entered is less than or greater than the available Code Offset values, it will be automatically set to the lowest or highest valid value.", and "If the Code Offset value entered is in the range of valid Code Offset values but not an exact match, the next lowest value will be selected automatically." The dialog box has "Cancel" and "OK" buttons.

Akan menemukan:



```
URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger
Disassembler Project Debug Search Goto Execute Text Functions HexData Refs He
* Reference To: USER32.SendDlgItemMessageA, Urd:0000h
:00498A8B E841800500          Call 004F0AD1
:00498A90 68334A5000          push 00504A33
:00498A95 684C465000          push 0050464C
:00498A9A E8E5FBFFFF          call 00498684
:00498A9F 85C0                test eax, eax
:00498AA1 0F849B000000        je 00498B42
:00498AA7 BE3C9D4F00          mov esi, 004F9D3C
:00498AAC BF4C465000          mov edi, 0050464C
:00498AB1 33C0                xor eax, eax
:00498AB3 83C9FF             or ecx, FFFFFFFF
:00498AB6 F2                 repnz
:00498AB7 AE                 scasb
```

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

```
URSoft W32Dasm Ver 8.93 Program Disassembler/Debugger
Disassembler Project Debug Search Goto Execute Text Functions HexData Refs Help

* Reference To: USER32.SendDlgItemMessageA, Ord:0000h
|
:00498A8E E841800500 Call 004F0AD1
:00498A90 68334A5000 push 00504A33
:00498A95 684C465000 push 0050464C
:00498A9A E8E5FBFFFF call 00498684
:00498A9F 85C0 test eax, eax
:00498AA1 0F849B000000 je 00498B42
:00498AA7 BE3C9D4F00 mov esi, 004F9D3C
:00498AAC BF4C465000 mov edi, 0050464C
:00498AB1 33C0 xor eax, eax
:00498AB3 83C9FF or ecx, FFFFFFFF
:00498AB6 F2 repnz
:00498AB7 AE scasb
:00498AB8 F7D1 not ecx
:00498ABA 2BF9 sub edi, ecx
:00498ABC 87F7 xchg edi, esi
:00498ABE 8BC7 mov eax, edi
:00498AC0 8BD1 mov edx, ecx
:00498AC2 C1E902 shr ecx, 02
:00498AC5 F3 repz
:00498AC6 A5 movsd
:00498AC7 8BCA mov ecx, edx
:00498ACE 68334A5000 push 00504A33
:00498AD3 684C465000 push 0050464C
:00498AD8 E8CFFBFFFF call 004989AC
:00498ADD 6A00 push 00000000

Line:319132 Pg 3940 and 3941 of 6238 Code Data @:00498AA1 @ Offset 000980A1h in File:D:\mirc'
```

Catat Offset "00090A1h"

Run Hex Workshop dan load mirc32.exe

The screenshot shows the Hex Workshop interface with the file mirc32.exe loaded. The main window displays a hex dump with columns for hexadecimal values and their corresponding ASCII characters. Three red boxes highlight specific areas: one on the left side of the hex dump, one in the middle, and one on the right side. Red arrows point from these boxes to labels at the bottom of the image. The Data Inspector at the bottom left shows the current data type as 8-bit Signed Byte with a value of 117. The Compare Results window at the bottom right is empty.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	0123456789ABCDEF01	
00000000	4D	5A	50	00	02	00	00	00	04	00	0E	00	FF	FF	00	00	B8	00	MZP
00000012	00	00	00	00	00	00	40	00	1A	00	00	00	00	00	00	00	00	00@.....
00000024	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000036	00	00	00	00	00	00	00	01	00	00	BA	10	00	0E	1F	B4	09	CD
00000048	21	B8	01	4C	CD	21	90	90	54	68	69	73	20	70	72	6F	67	72	!..L!..This progr
0000005A	61	6D	20	6D	75	73	74	20	62	65	20	72	75	6E	20	75	6E	64	am must be run und
0000006C	65	72	20	57	69	6E	33	32	0D	0A	24	37	00	00	00	00	00	00	er Win32..\$7.....
0000007E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000A2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

List alamat Kode

Kode Mesin dalam Hexa

Kode Mesin dalam ASCII

Goto Offset (hex) 00980A1

The screenshot shows the Hex Workshop application window with the 'Goto' dialog box open. The dialog box has a text field for 'Offset' containing '00980A1', which is circled in red. Below the text field are radio buttons for 'Dec' and 'Hex', with 'Hex' selected. To the right of the dialog box is the 'From Where' section with radio buttons for 'Beginning of File', 'Current Position', and 'End of File (back from)'. The background shows the Hex Workshop interface with a menu open, a hex editor, and a data inspector.

Hex Workshop - [mirc32.exe]

File Edit Disk Options Tools Window Help

Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Copy As
Paste Ctrl+V
Paste Special...
Select All Ctrl+A
Select Block
Insert... Ctrl+Ins
Delete
Find... Ctrl+F
Find Next F3
Find Previous Shift+F3
Replace... Ctrl+H
Goto... Ctrl+G
Goto Again F5
Properties... Alt+Enter

Offset: 00980A1
From Where
 Beginning of File
 Current Position
 End of File (back from)

Go Cancel Help

Right click in your document for more goto options

6 7 8 9 A B C D E F 10 11 0123456789ABCDEF01
00 00 04 00 0F 00 FF FF 00 00 B8 00 MZP.....
40 00 1A 00 00 00 00 00 00 00 00 00@.....
.....L!..This progr
must be run und
Win32..\$7.....

Source	Count	Count	Target
	16	10	

Compare Results

Compare Checksum Find Bookm... Output

Offset: 0000005F Value: 29557 1339392 bytes OVR MOD READ

Menghasilkan :

The screenshot shows the Hex Workshop interface for the file mirc32.exe. The main window displays a hex dump with the following data:

Offset	Hex	ASCII
0009809A	E8 E5 FB FF FF 85 C0 0F 84 9B 00 00 00 BE 3C 9D 4F 00<.O.
000980AC	BF 4C 46 50 00 33 C0 83 C9 FF F2 AE F7 D1 2B F9 87 F7	.LFP.3.....+...
000980BE	8B C7 8B D1 C1 E9 02 F3 A5 8B CA 83 E1 03 F3 A4 68 33h3
000980D0	4A 50 00 68 4C 46 50 00 E8 CF FE FF FF 6A 00 68 86 00	JP.hLFP.....j.h..
000980E2	00 00 A1 C0 55 51 00 50 E8 AE 80 05 00 68 EC 7D 4F 00	...UQ.P.....h.}O.
000980F4	68 BA 9E 4F 00 68 B4 9E 4F 00 68 AD 9E 4F 00 E8 64 A7	h..O.h..O.h..O..d.
00098106	FE FF 6A 01 8B 55 08 52 E8 DE 80 05 00 6A 40 6A 00 68	..j..U.R.....j@j.h
00098118	78 07 00 00 E8 57 66 F8 FF 50 6A 00 68 77 07 00 00 E8	x....Wf..Pj.hw....
0009812A	4A 66 F8 FF 50 8B 4D 08 51 E8 33 7F 05 00 B8 01 00 00	Jf..P.M.Q.3.....
0009813C	00 E9 C0 00 00 00 6A 00 E8 52 7F 05 00 68 C3 9E 4F 00j..R..h..O.

An orange text box overlaid on the hex dump reads: "Ganti kode '0F 84 9B 00 00 00' menjadi '90 90 42 4A 90 90'".

The Data Inspector window shows the following data types:

- 8BIT Signed Byte: 15
- 8BIT Unsigned Byte: 15
- 16BIT Signed Short: -31729
- 16BIT Unsigned Short: 33807
- 32BIT Signed Long: 10191887
- 32BIT Unsigned Long: 10191887
- 64BIT Signed Quad
- 64BIT Unsigned Quad
- 32BIT Float: 1.4281876e-038

The Compare Results window is empty.

At the bottom, the status bar indicates: "Jumped to position 0x000980A1 (622753)", "Offset: 000980A1", "Sel: 0x6 bytes", "1339392 bytes", and "OVR MOD READ".

Hasil :

The screenshot shows the Hex Workshop interface with the following data visible in the main window:

Address	Hex	ASCII
0009809A	E8 E5 FB FF FF 85 C0 90 90 42 4A 90 90 BE 3C 9D 4F 00BJ...<.O.
000980AC	BF 4C 46 50 00 33 C0 83 C9 FF F2 AE F7 D1 2B F9 87 F7	.LFP.3.....+...
000980BE	8B C7 8B D1 C1 E9 02 F3 A5 8B CA 83 E1 03 F3 A4 68 33h3
000980D0	4A 50 00 68 4C 46 50 00 E8 CF FE FF FF 6A 00 68 86 00	JP.hLFP.....j.h..
000980E2	00 00 A1 C0 55 51 00 50 E8 AE 80 05 00 68 EC 7D 4F 00	...UQ.P.....h.}O.
000980F4	68 BA 9E 4F 00 68 B4 9E 4F 00 68 AD 9E 4F 00 E8 64 A7	h..O.h..O.h..O..d.

An orange text box is overlaid on the hex data with the following text:

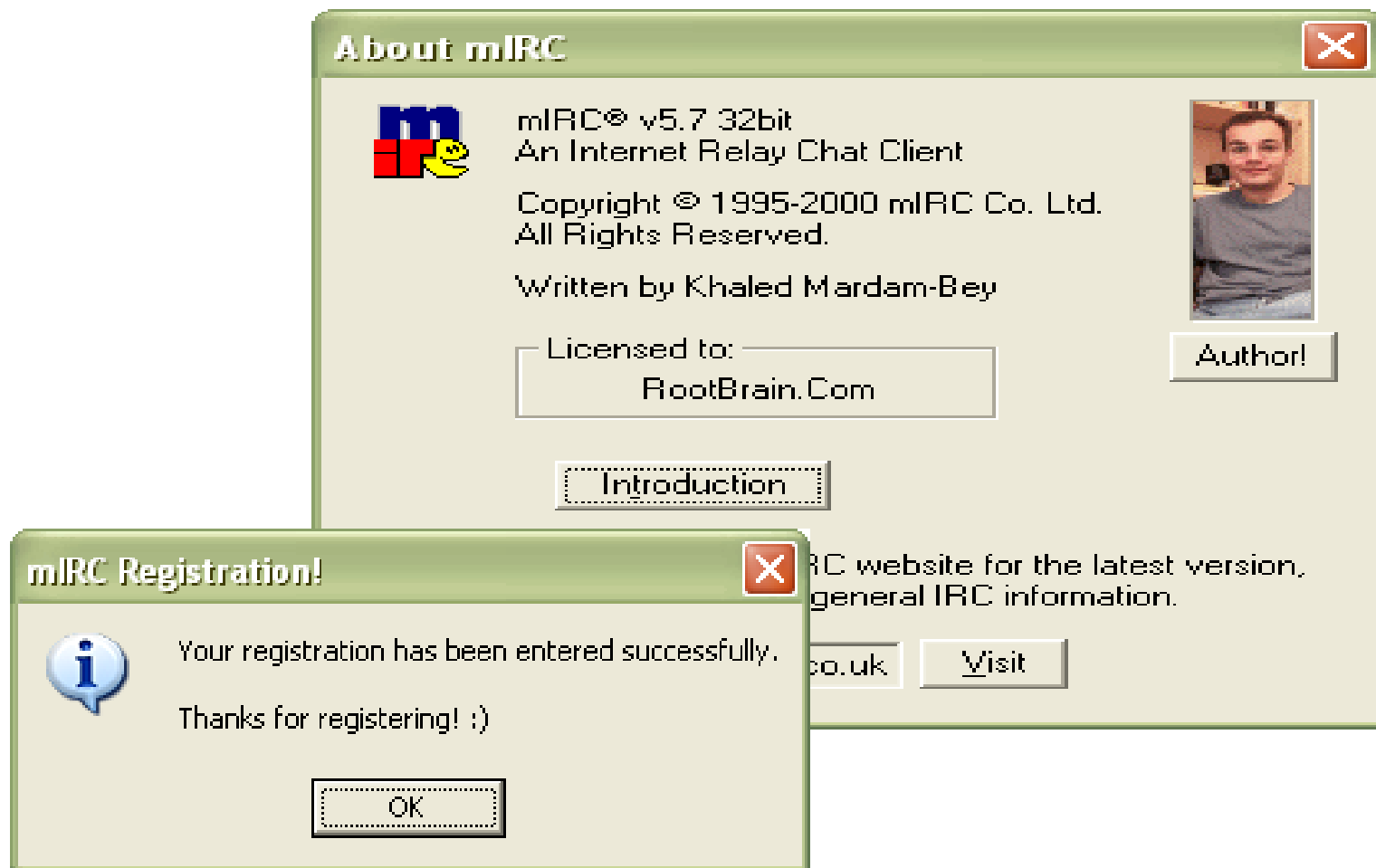
**Simpan dan Jalankan kembali Program mirc32.exe
Silakan register dengan nama & serial sembarang**

The interface also shows a Data Inspector window on the left with the following data:

Offset	Value
8 BIT Signed Byte	-112
8 BIT Unsigned Byte	144
16 BIT Signed Short	-28528
16 BIT Unsigned Short	37008
32 BIT Signed Long	1245876368
32 BIT Unsigned Long	1245876368
64 BIT Signed Quad	
64 BIT Unsigned Quad	
32 BIT Float	3187748.

The Compare Results window on the right is empty, and the status bar at the bottom shows: Ready, Offset: 000980A1, Sel: 0x6 bytes, 1339392 bytes, OVR MOD READ.

Registration Success, Lihat About

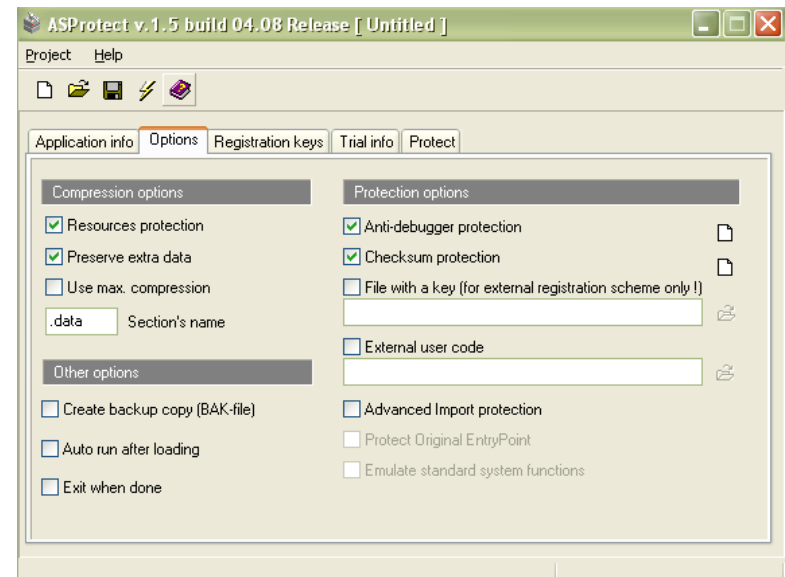


Demo Proteksi Software

❏ 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/> Software Protection & Cracking Tutorial
- ◆ <http://www.woodmann.com/crackz/> CrackZ's Reverse Engineering