Ethical Hacking
v5

Advanced Module

Reverse Engineering

Overview of RE

- Reverse engineering is often viewed as the craft of the cracker who uses his skills to remove copy protection from software or media.
- Digital Millennium Copyright Act (DMCA) law kicks in here to prevent that

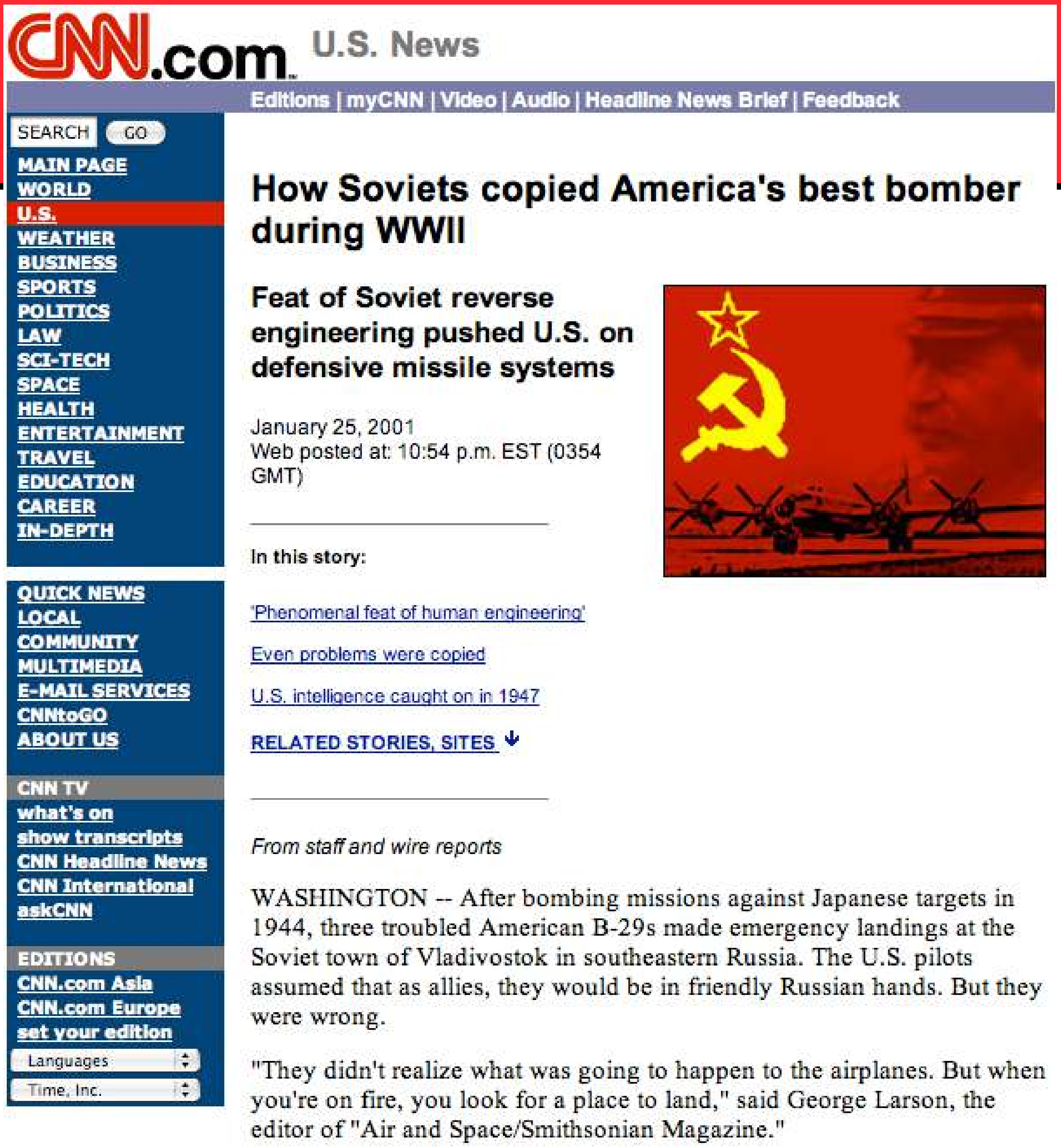
Positive Application of Reverse Engineering

- Understanding the capabilities of the product’s manufacturer
- Understanding the functions of the product in order to create compatible components
- Determining whether vulnerabilities exist in a product
- Determining whether an application contains any undocumented functionality

Ethical Reverse Engineering

- An ethical hacker may carry out reverse engineering to mitigate:
  - Failure to check for error conditions
  - Poor understanding of function behaviors
  - Poorly designed protocols
  - Improper testing for boundary conditions

Source: http://archives.cnn.com/2001/US/01/25/smithsonian.cold.war/



How Soviets copied America's best bomber during WWII

Feat of Soviet reverse engineering pushed U.S. on defensive missile systems

January 25, 2001
Web posted at: 10:54 p.m. EST (0354 GMT)

In this story:

'Phenomenal feat of human engineering'

Even problems were copied

U.S. intelligence caught on in 1947

RELATED STORIES, SITES

From staff and wire reports

WASHINGTON -- After bombing missions against Japanese targets in 1944, three troubled American B-29s made emergency landings at the Soviet town of Vladivostok in southeastern Russia. The U.S. pilots assumed that as allies, they would be in friendly Russian hands. But they were wrong.

"They didn't realize what was going to happen to the airplanes. But when you're on fire, you look for a place to land," said George Larson, the editor of "Air and Space/Smithsonian Magazine."

Case Study

- http://archives.cnn.com/2001/US/01/25/smithsonian.cold.war/

WASHINGTON -- After bombing missions against Japanese targets in 1944, three troubled American B-29s made emergency landings at the Soviet town of Vladivostok in southeastern Russia. The U.S. pilots assumed that as allies, they would be in friendly Russian hands. But they were wrong."They didn't realize what was going to happen to the airplanes. The crews dismantled one of the planes into 105,000 parts, created blueprints and then reproduced the bomber in just two years. **They took it apart component by component, panel by panel, almost rivet by rivet," Hardesty said in an interview."It was measured and copied and photographed, and then someone would get the assignment to replicate a part, like an altimeter."He said they finished the design work in one year** and produced planes in the second. The B-29 was copied almost exactly

EC-Council

DMCA Act

- The **Digital Millennium Copyright Act** (**DMCA**) is a United States copyright law which criminalizes production and dissemination of technology that can circumvent measures taken to protect copyright, not merely infringement of copyright itself, and heightens the penalties for copyright infringement on the Internet.

What is a Disassembler?

- A disassembler is the exact opposite of an assembler.
- Where an Assembler converts code written in an assembly language into binary machine code, a disassembler reverses the process and attempts to recreate the assembly code from the binary machine code

Why do you need to decompile?

- Decompilation can be used for a number of reasons
  - Recovery of lost source code (by accident or via a disgruntled employee)
  - Migration of assembly language applications to a new hardware platform
  - Translation of code written in obsolete languages no longer supported by compiler tools
  - Determination of the existence of viruses or malicious code in the program
  - Recovery of someone else's source code (to determine an algorithm for example)

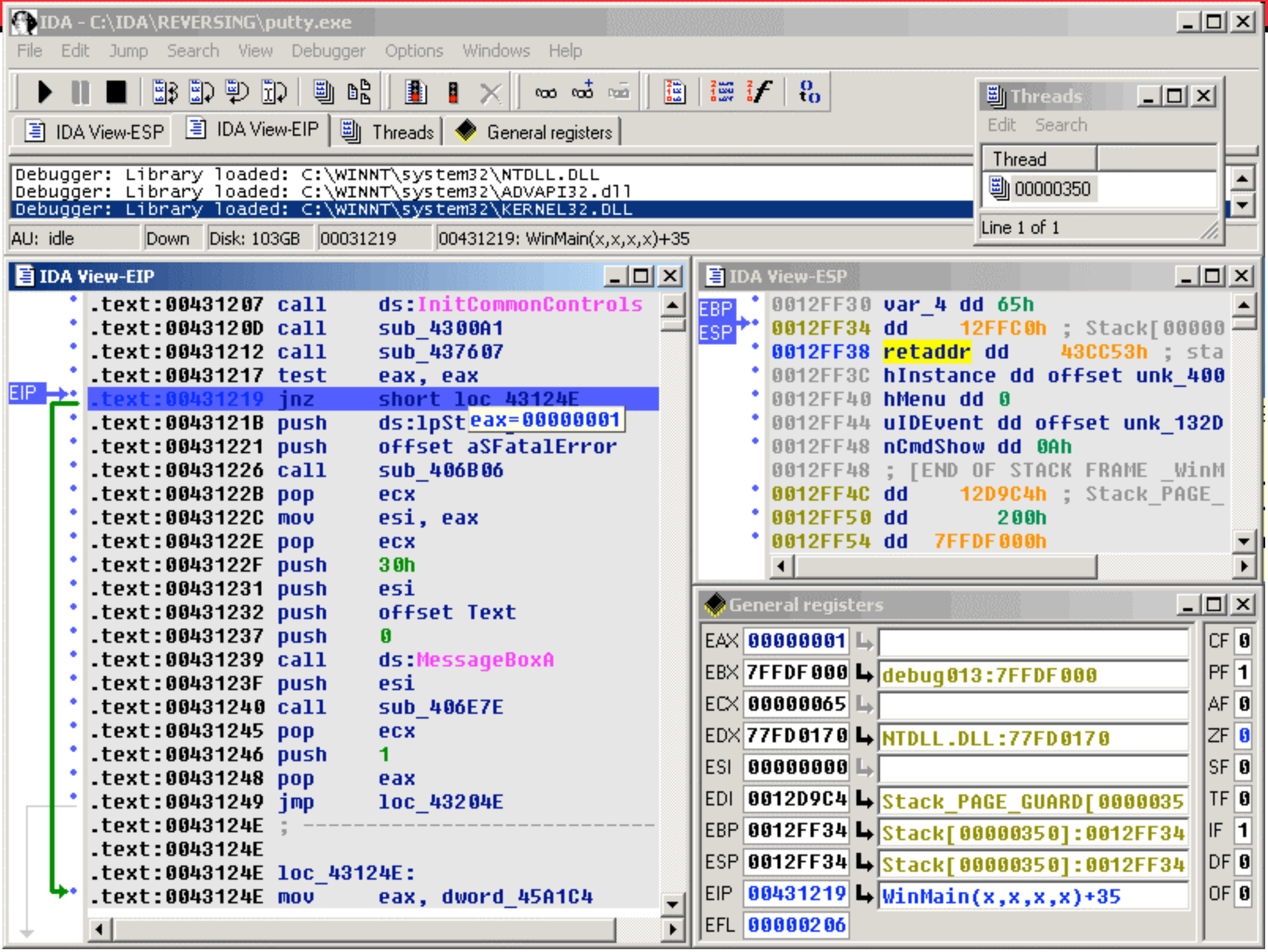
Professional Disassemblers Tools

- **IDA Pro**
  - A professional (read: expensive) disassembler that is extremely powerful, and has a whole slew of features.
- **PE Explorer** is a disassembler that "focuses on ease of use, clarity and navigation." It isn't as feature-filled as IDA Pro.
- **W32DASM**
  - W32DASM is an excellent 16/32 bit disassembler for Windows

IDAPro

- IDAPro is produced by Data Rescue
- It is used as a Disassembler in Manual Binary Code Analysis and is also a Debugger
- IDAPro is interactive and programmable
- Debugging Features:
  - Instant Debugging
  - Connects local and remote systems easily
- Disassembling Features:
  - It explores the depth of Binary data
  - Converts assembly language code into a much readable format

IDA PRO



```
Debugger: Library loaded: C:\WINNT\system32\NTDLL.DLL
Debugger: Library loaded: C:\WINNT\system32\ADVAPI32.dll
Debugger: Library loaded: C:\WINNT\system32\KERNEL32.DLL
```

```
AU: idle   Down   Disk: 103GB   00031219   00431219: WinMain(x,x,x,x)+35
```

```
Thread
00000350
Line 1 of 1
```

```
     .text:00431207 call    ds:InitCommonControls
     .text:0043120D call    sub_4300A1
     .text:00431212 call    sub_437607
     .text:00431217 test    eax, eax
EIP  .text:00431219 jnz     short loc_43124E
     .text:0043121B push    ds:lpSt eax=00000001
     .text:00431221 push    offset aSFatalError
     .text:00431226 call    sub_406B06
     .text:0043122B pop     ecx
     .text:0043122C mov     esi, eax
     .text:0043122E pop     ecx
     .text:0043122F push    30h
     .text:00431231 push    esi
     .text:00431232 push    offset Text
     .text:00431237 push    0
     .text:00431239 call    ds:MessageBoxA
     .text:0043123F push    esi
     .text:00431240 call    sub_406E7E
     .text:00431245 pop     ecx
     .text:00431246 push    1
     .text:00431248 pop     eax
     .text:00431249 jmp     loc_43204E
     .text:0043124E ; ---------------------------
     .text:0043124E
     .text:0043124E loc_43124E:
     .text:0043124E mov     eax, dword_45A1C4
```

```
EBP  0012FF30 var_4 dd 65h
ESP  0012FF34 dd     12FFC0h ; Stack[00000
     0012FF38 retaddr dd    43CC53h ; sta
     0012FF3C hInstance dd offset unk_400
     0012FF40 hMenu dd 0
     0012FF44 uIDEvent dd offset unk_132D
     0012FF48 nCmdShow dd 0Ah
     0012FF48 ; [END OF STACK FRAME _WinM
     0012FF4C dd     12D9C4h ; Stack_PAGE_
     0012FF50 dd         200h
     0012FF54 dd   7FFDF000h
```

```
EAX 00000001                                 CF 0
EBX 7FFDF000  debug013:7FFDF000              PF 1
ECX 00000065                                 AF 0
EDX 77FD0170  NTDLL.DLL:77FD0170             ZF 0
ESI 00000000                                 SF 0
EDI 0012D9C4  Stack_PAGE_GUARD[0000035       TF 0
EBP 0012FF34  Stack[00000350]:0012FF34       IF 1
ESP 0012FF34  Stack[00000350]:0012FF34       DF 0
EIP 00431219  WinMain(x,x,x,x)+35            OF 0
EFL 00000206
```

Convert Machine Code to Assembly Code

Machine Code

```
55 8B EC 83 EC 04
56 57 1E B8 94 00
50 9A八八 0E 00 3C
17 59 59 16 8D 46
FC 50 1E B8 B1 00
50八八 9A 07 00 F0
17 83 C4 08 BE 01
00 EB 3B 1E B8 B4
八八 00 50 9A 0E 00
3C 17 59 59 16 8D
46 FE 50 1E B8八八
C3 00 50 9A 07 00
F0 17 83 C4 08 FF
76 FE 9A 7C八八 00
```

Assembly Code

```
{
   register unsigned
int b = 0;
   register unsigned
int a = 0;
   do
   {
      a = a + array[b];
      b++;
   }while(b != 100);
   return b;
}
```

EC-Council

Decompilers

- A decompiler takes as input an executable file, and attempts to create a high level, compilable, possibly even maintainable source file that does the same thing.
- It is therefore the opposite of a compiler, which takes a source file and makes an executable.
- A general decompiler does not attempt to reverse every action of the decompiler, rather it transforms the input program repeatedly until the result is high level source code.
- It will not recreate the original source file

Program Obfuscation

- Obfuscation is a ProgramTransformation that makes a program harder to understand by renaming variables, inserting dead code, etc.
- Obfuscation is done to hide the business rules embedded in software by making it harder to reverse engineer the program.

Convert Assembly Code to C++ Code

Assembly Code

```
push ebp
mov ebp, esp
mov esi, [ebp + 8]
mov ebx, 0
mov eax, 0
mov ecx, 0
_Label_1:
mov ecx, [esi + ebx * 4]
add eax, ecx
add ebx, 4
inc ebx
cmp ebx, 100
je _Label_1
mov esp, ebp
pop ebp
ret 4
```

$\longrightarrow$

C Code

```
{
   register unsigned int b = 0;
   register unsigned int a = 0;
   do
   {
      a = a + array[b];
      b++;
   }while(b != 100);
   return b;
}
```

Machine Decompilers

- **DCC Decompiler**
  - Dcc is an excellent theoretical look at decompilation, but currently it only supports small files.http://www.itee.uq.edu.au/~cristina/dcc.html
- **Boomerang Decompiler Project**
  - Boomerang Decompiler is an attempt to make a powerful, retargetable compiler. So far, it only decompiles into C with moderate success.http://boomerang.sourceforge.net/
- **Reverse Engineering Compiler (REC)**
  - REC is a powerful "decompiler" that decompiles native assembly code into a *C-like* code representation. The code is half-way between assembly and C, but it is much more readable then the pure assembly is.http://www.backerstreet.com/rec/rec.htm
- **ExeToC**
  - ExeToC decompiler is an interactive decompiler that boasts pretty good results.http://sourceforge.net/projects/exetoc

Tool: dcc

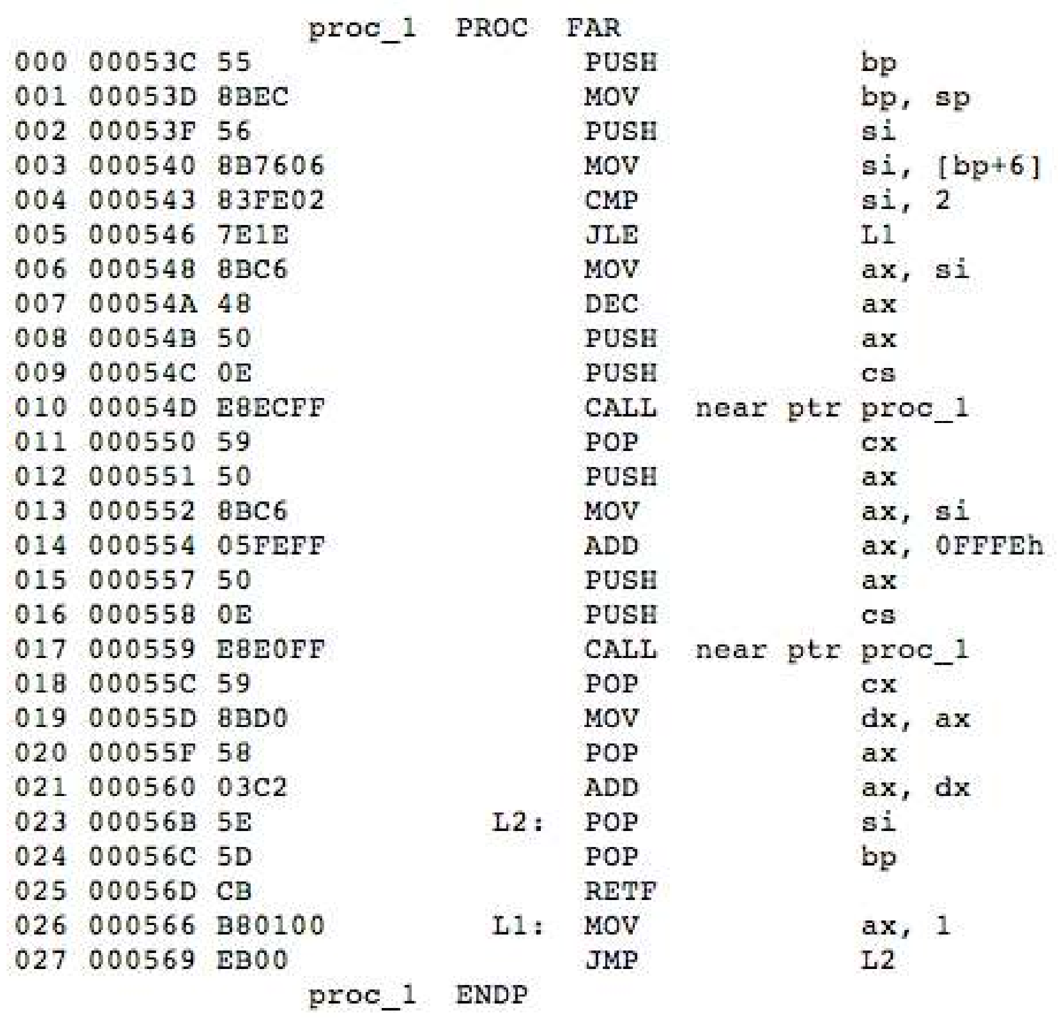
- The dcc decompiler decompiles .exe files from the (i386, DOS) platform to C programs.
- The final C program contains assembler code for any subroutines that are not possible to be decompiled at a higher level than assembler.
- It can only decompile the code to C and not C++

Machine code of compute.exe program

```
      55 8B EC 83 EC 04 56 57 1E B8 94 00 50 9A
0E 00 3C 17 59 59 16 8D 46 FC 50 1E B8 B1 00 50
9A 07 00 F0 17 83 C4 08 BE 01 00 EB 3B 1E B8 B4
00 50 9A 0E 00 3C 17 59 59 16 8D 46 FE 50 1E B8
C3 00 50 9A 07 00 F0 17 83 C4 08 FF 76 FE 9A 7C
00 3B 16 59 8B F8 57 FF 76 FE 1E B8 C6 00 50 9A
0E 00 3C 17 83 C4 08 46 3B 76 FC 7E C0 33 C0 50
9A 0A 00 49 16 59 5F 5E 8B E5 5D CB 55 8B EC 56
8B 76 06 83 FE 02 7E 1E 8B C6 48 50 0E E8 EC FF
59 50 8B C6 05 FE FF 50 0E E8 E0 FF 59 8B D0 58
03 C2 EB 07 EB 05 B8 01 00 EB 00 5E 5D CB
```

EC-Council

Assembly Code of compute.exe

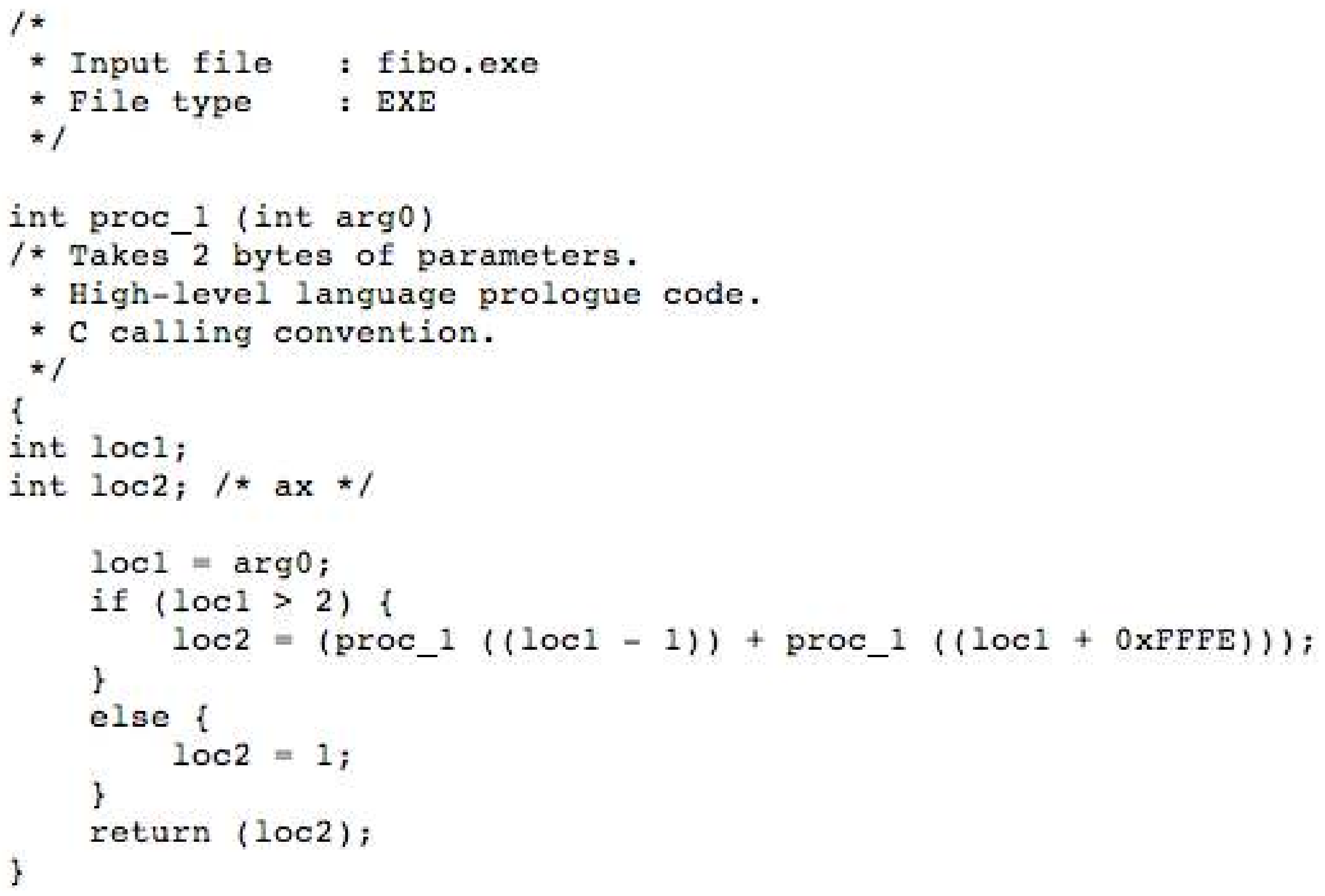


```
                proc_1  PROC   FAR
000 00053C 55                   PUSH            bp
001 00053D 8BEC                 MOV             bp, sp
002 00053F 56                   PUSH            si
003 000540 8B7606               MOV             si, [bp+6]
004 000543 83FE02               CMP             si, 2
005 000546 7E1E                 JLE             L1
006 000548 8BC6                 MOV             ax, si
007 00054A 48                   DEC             ax
008 00054B 50                   PUSH            ax
009 00054C 0E                   PUSH            cs
010 00054D E8ECFF               CALL  near ptr  proc_1
011 000550 59                   POP             cx
012 000551 50                   PUSH            ax
013 000552 8BC6                 MOV             ax, si
014 000554 05FEFF               ADD             ax, 0FFFEh
015 000557 50                   PUSH            ax
016 000558 0E                   PUSH            cs
017 000559 E8E0FF               CALL  near ptr  proc_1
018 00055C 59                   POP             cx
019 00055D 8BD0                 MOV             dx, ax
020 00055F 58                   POP             ax
021 000560 03C2                 ADD             ax, dx
023 00056B 5E              L2:  POP             si
024 00056C 5D                   POP             bp
025 00056D CB                   RETF
026 000566 B80100          L1:  MOV             ax, 1
027 000569 EB00                 JMP             L2
                proc_1  ENDP
```

EC-Council

```
                    main   PROC   FAR
000 0004C2 55                    PUSH              bp
001 0004C3 8BEC                  MOV               bp, sp
002 0004C5 83EC04                SUB               sp, 4
003 0004C8 56                    PUSH              si
004 0004C9 57                    PUSH              di
005 0004CA 1E                    PUSH              ds
006 0004CB B89400                MOV               ax, 94h
007 0004CE 50                    PUSH              ax
008 0004CF 9A0E004D01            CALL     far ptr  printf
009 0004D4 59                    POP               cx
010 0004D5 59                    POP               cx
011 0004D6 16                    PUSH              ss
012 0004D7 8D46FC                LEA               ax, [bp-4]
013 0004DA 50                    PUSH              ax
014 0004DB 1E                    PUSH              ds
015 0004DC B8B100                MOV               ax, 0B1h
016 0004DF 50                    PUSH              ax
017 0004E0 9A07000102            CALL     far ptr  scanf
018 0004E5 83C408                ADD               sp, 8
019 0004E8 BE0100                MOV               si, 1
021 000528 3B76FC           L3:  CMP               si, [bp-4]
022 00052B 7EC0                  JLE               L4
023 00052D 33C0                  XOR               ax, ax
024 00052F 50                    PUSH              ax
025 000530 9A0A005A00            CALL     far ptr  exit
026 000535 59                    POP               cx
027 000536 5F                    POP               di
028 000537 5E                    POP               si
029 000538 8BE5                  MOV               sp, bp
030 00053A 5D                    POP               bp
031 00053B CB                    RETF
032 0004ED 1E               L4:  PUSH              ds
033 0004EE B8B400                MOV               ax, 0B4h
```

Code produced by the dcc Decompiler in C



```
/*
 * Input file    : fibo.exe
 * File type     : EXE
 */

int proc_1 (int arg0)
/* Takes 2 bytes of parameters.
 * High-level language prologue code.
 * C calling convention.
 */
{
int loc1;
int loc2; /* ax */

    loc1 = arg0;
    if (loc1 > 2) {
        loc2 = (proc_1 ((loc1 - 1)) + proc_1 ((loc1 + 0xFFFE)));
    }
    else {
        loc2 = 1;
    }
    return (loc2);
}
```

EC-Council

```
void main ()
/* Takes no parameters.
 * High-level language prologue code.
 */
{
int loc1;
int loc2;
int loc3;
int loc4;

    printf ("Input number of iterations: ");
    scanf ("%d", &loc1);
    loc3 = 1;
    while ((loc3 <= loc1)) {
        printf ("Input number: ");
        scanf ("%d", &loc2);
        loc4 = proc_1 (loc2);
        printf ("fibonacci(%d) = %u\n", loc2, loc4);
        loc3 = (loc3 + 1);
    } /* end of while */
    exit (0);
}
```

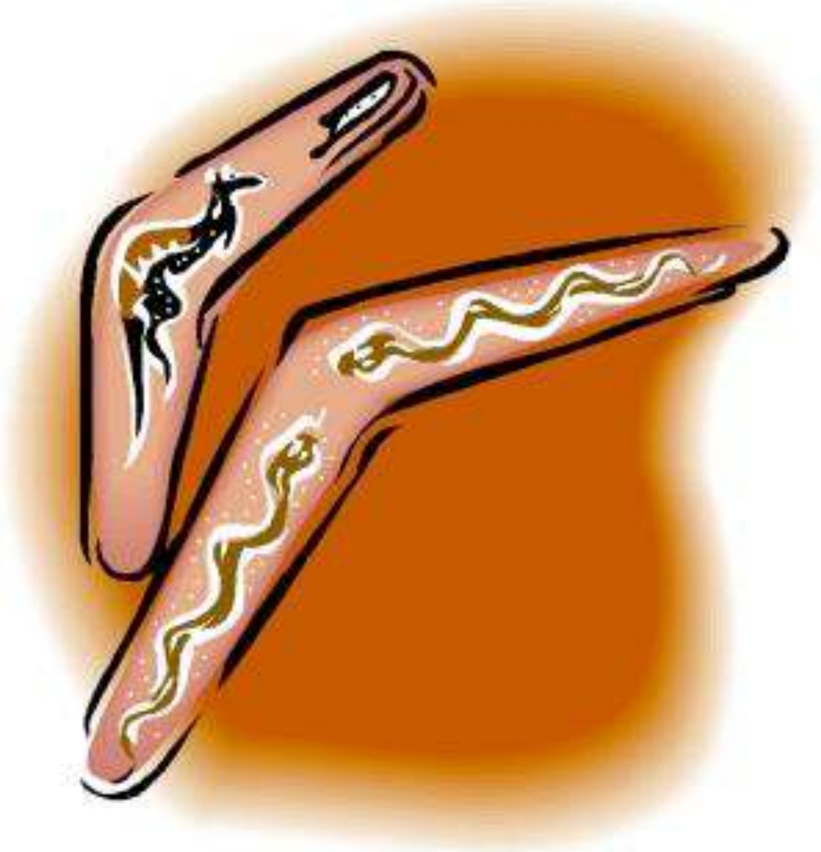
The original C code for the program compute.exe

```
#include <stdio.h>

int main()
{ int i, numtimes, number;
  unsigned value, fib();

   printf("Input number of iterations: ");
   scanf ("%d", &numtimes);
   for (i = 1; i <= numtimes; i++)
   {
      printf ("Input number: ");
      scanf ("%d", &number);
      value = fib(number);
      printf("fibonacci(%d) = %u\n", number, value);
   }
   exit(0);
}

unsigned fib(x)                 /* compute fibonacci number recursively */
int x;
{
   if (x > 2)
      return (fib(x - 1) + fib(x - 2));
   else
      return (1);
}
```

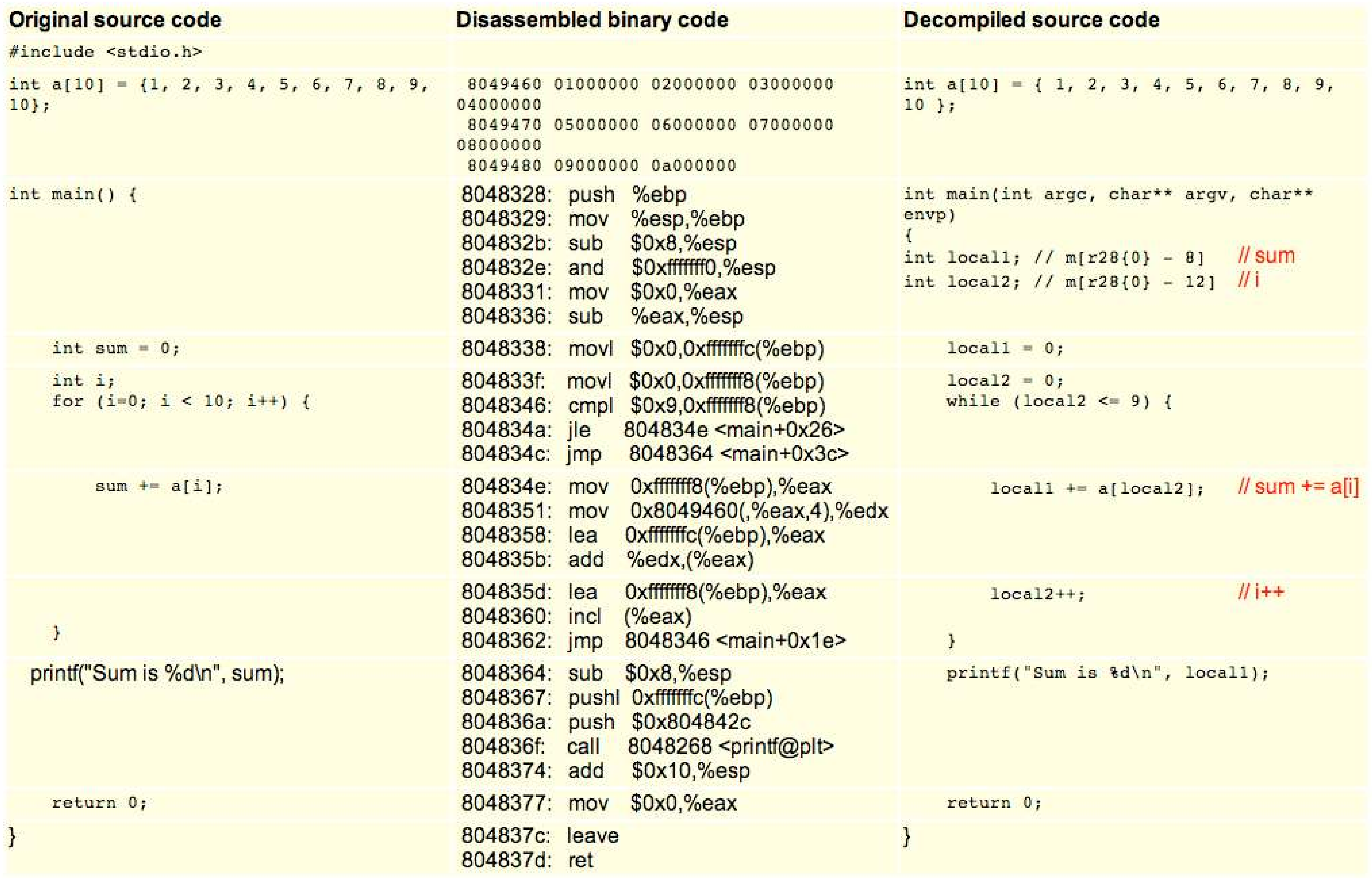
EC-Council

Tool: Boomerang

- This project is an attempt to develop a real decompiler for machine code programs through the open source community
- By transforming the semantics of individual instructions, and using powerful techniques such as Static Single Assignment dataflow analysis, Boomerang should be (largely) independent of the exact behavior of the compiler that happened to be used



What Boomerang Can Do?

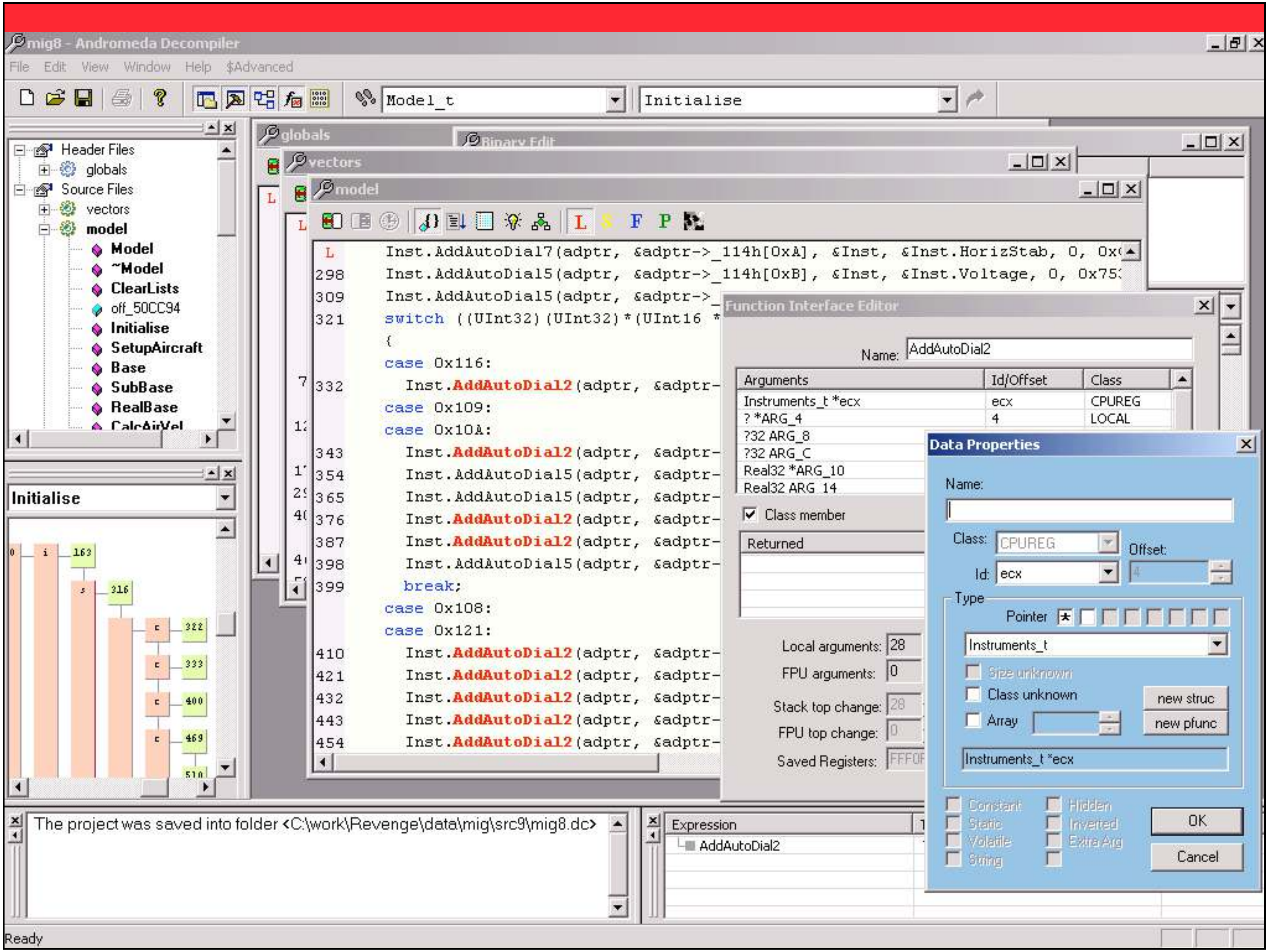


Original source code	Disassembled binary code	Decompiled source code
`#include <stdio.h>`		
`int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};`	`8049460 01000000 02000000 03000000 04000000`	`int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };`
	`8049470 05000000 06000000 07000000 08000000`	
	`8049480 09000000 0a000000`	
`int main() {`	`8048328: push %ebp`	`int main(int argc, char** argv, char** envp)`
	`8048329: mov %esp,%ebp`	`{`
	`804832b: sub $0x8,%esp`	`int local1; // m[r28{0} - 8]` // sum
	`804832e: and $0xfffffff0,%esp`	`int local2; // m[r28{0} - 12]` // i
	`8048331: mov $0x0,%eax`	
	`8048336: sub %eax,%esp`	
`int sum = 0;`	`8048338: movl $0x0,0xfffffffc(%ebp)`	`local1 = 0;`
`int i;`	`804833f: movl $0x0,0xfffffff8(%ebp)`	`local2 = 0;`
`for (i=0; i < 10; i++) {`	`8048346: cmpl $0x9,0xfffffff8(%ebp)`	`while (local2 <= 9) {`
	`804834a: jle 804834e <main+0x26>`	
	`804834c: jmp 8048364 <main+0x3c>`	
`sum += a[i];`	`804834e: mov 0xfffffff8(%ebp),%eax`	`local1 += a[local2];` // sum += a[i]
	`8048351: mov 0x8049460(,%eax,4),%edx`	
	`8048358: lea 0xfffffffc(%ebp),%eax`	
	`804835b: add %edx,(%eax)`	
	`804835d: lea 0xfffffff8(%ebp),%eax`	`local2++;` // i++
	`8048360: incl (%eax)`	
`}`	`8048362: jmp 8048346 <main+0x1e>`	`}`
`printf("Sum is %d\n", sum);`	`8048364: sub $0x8,%esp`	`printf("Sum is %d\n", local1);`
	`8048367: pushl 0xfffffffc(%ebp)`	
	`804836a: push $0x804842c`	
	`804836f: call 8048268 <printf@plt>`	
	`8048374: add $0x10,%esp`	
`return 0;`	`8048377: mov $0x0,%eax`	`return 0;`
`}`	`804837c: leave`	`}`
	`804837d: ret`	

EC-Council

Andromeda Decompiler

- AD is an attempt to create the universal interactive program environment for reversive engineering
- AD is an interactive decompiler.
- It means that the user takes active participation in the decompilation process.
- AD is not an automatic analyser of programs.



```
Inst.AddAutoDial7(adptr, &adptr->_114h[0xA], &Inst, &Inst.HorizStab, 0, 0x
Inst.AddAutoDial5(adptr, &adptr->_114h[0xB], &Inst, &Inst.Voltage, 0, 0x75
Inst.AddAutoDial5(adptr, &adptr->
switch ((UInt32)(UInt32)*(UInt16 *
{
case 0x116:
  Inst.AddAutoDial2(adptr, &adptr-
case 0x109:
case 0x10A:
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial5(adptr, &adptr-
  Inst.AddAutoDial5(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial5(adptr, &adptr-
  break;
case 0x108:
case 0x121:
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
  Inst.AddAutoDial2(adptr, &adptr-
```

Function Interface Editor

Name: AddAutoDial2

Arguments	Id/Offset	Class
Instruments_t *ecx	ecx	CPUREG
? *ARG_4	4	LOCAL
?32 ARG_8		
?32 ARG_C		
Real32 *ARG_10		
Real32 ARG_14		

[✓] Class member

Returned

Local arguments: 28
FPU arguments: 0
Stack top change: 28
FPU top change: 0
Saved Registers: FFF0F

Data Properties

Name:
Class: CPUREG
Offset: 4
Id: ecx
Type
Pointer [✓ *]
Instruments_t
Size unknown
Class unknown
Array
Instruments_t *ecx
Constant Hidden
Static Inverted
Volatile Extra Arg
String

Initialise

The project was saved into folder <C:\work\Revenge\data\mig\src9\mig8.dc>

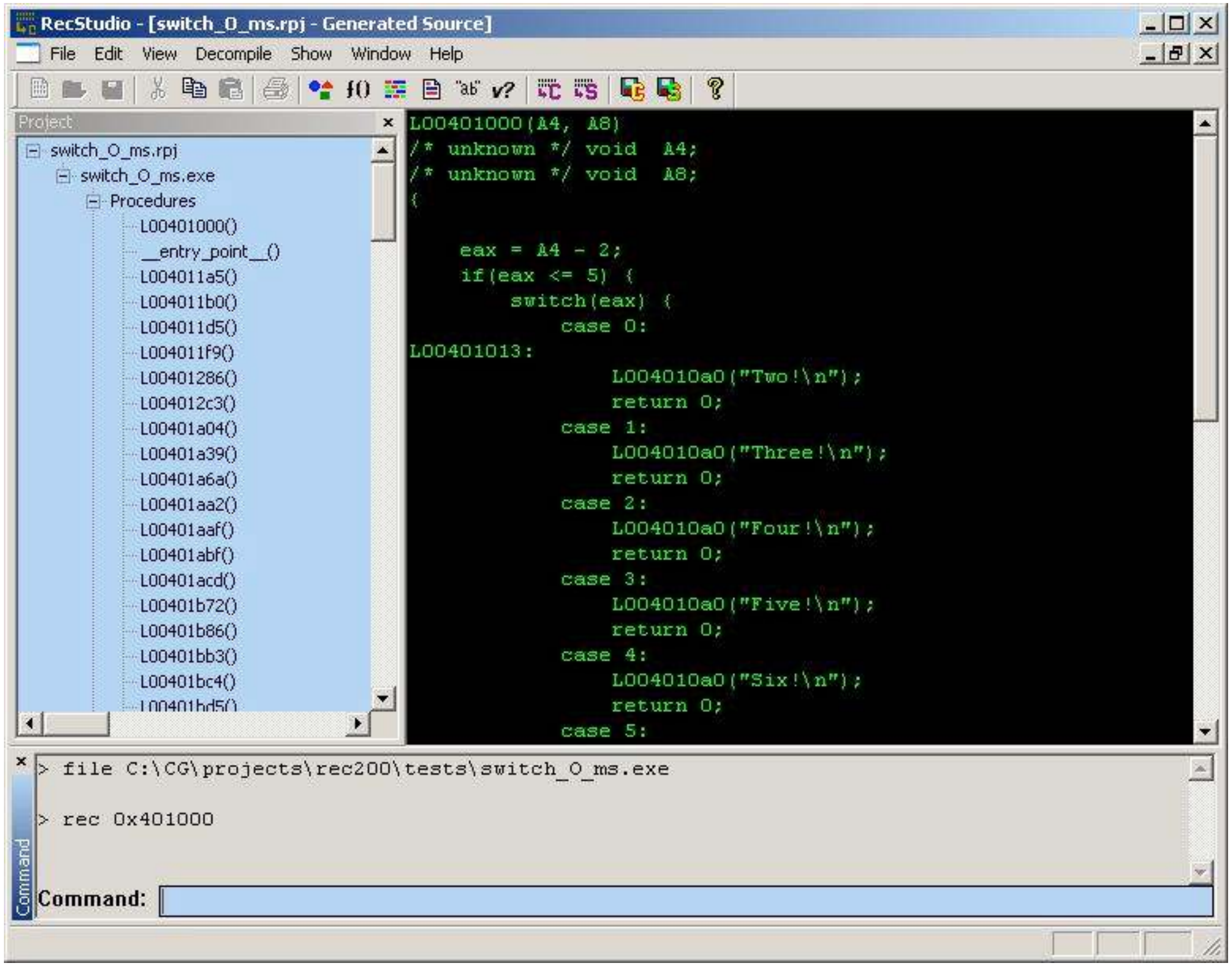
Expression
AddAutoDial2

Ready

Tool: REC Decompiler

- REC is a portable reverse engineering compiler, or **decompiler**
- It reads an executable file, and attempts to produce a C-like representation of the code and data used to build the executable file
- It is portable because it has been designed to read files produced for many different targets, and it has been compiled on several host systems

REC Tool Screenshot

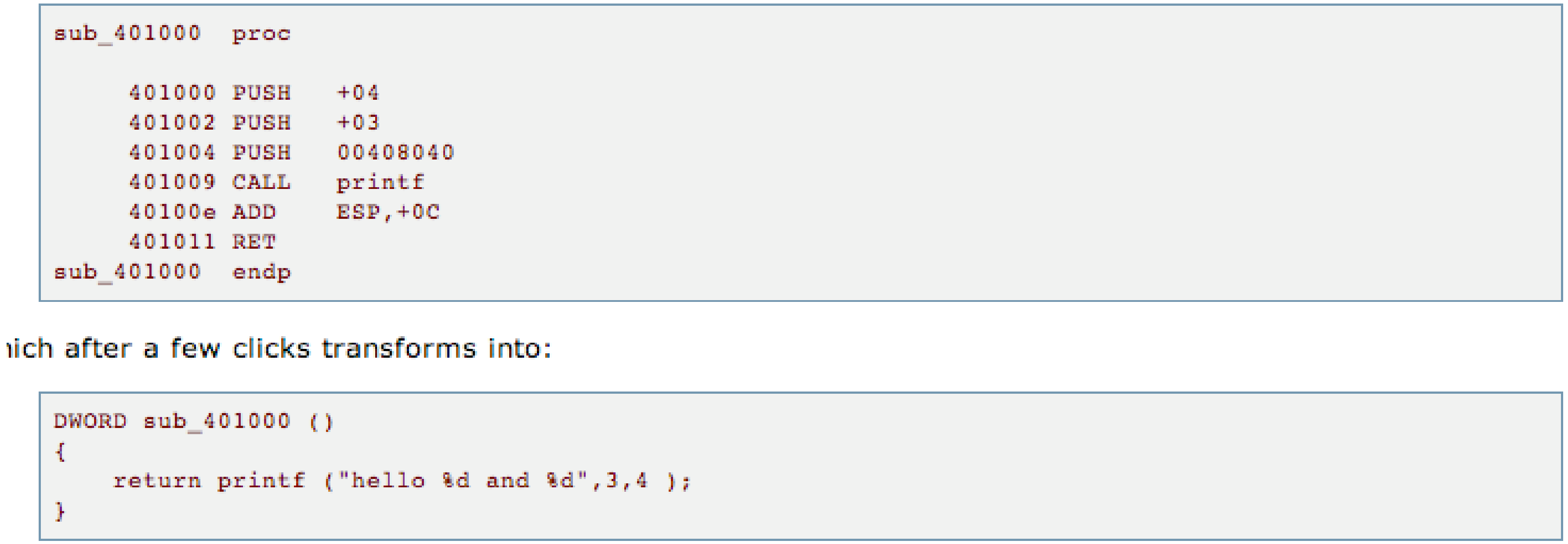


```
L00401000(A4, A8)
/* unknown */ void  A4;
/* unknown */ void  A8;
{

    eax = A4 - 2;
    if(eax <= 5) {
        switch(eax) {
            case 0:
L00401013:
                L004010a0("Two!\n");
                return 0;
            case 1:
                L004010a0("Three!\n");
                return 0;
            case 2:
                L004010a0("Four!\n");
                return 0;
            case 3:
                L004010a0("Five!\n");
                return 0;
            case 4:
                L004010a0("Six!\n");
                return 0;
            case 5:
```

```
> file C:\CG\projects\rec200\tests\switch_O_ms.exe

> rec 0x401000
```

Command:

Tool: Exe To C Decompiler



```
sub_401000  proc

     401000 PUSH   +04
     401002 PUSH   +03
     401004 PUSH   00408040
     401009 CALL   printf
     40100e ADD    ESP,+0C
     401011 RET
sub_401000  endp
```

ich after a few clicks transforms into:

```
DWORD sub_401000 ()
{
    return printf ("hello %d and %d",3,4 );
}
```

EC-Council

Delphi Decompilers

MultiRipper

- MultiRipper is a Windows and Delphi/C++ Builder ripper by Baccan Matteo and Peruch Emiliano.
- A "ripper" program extracts files inside other files. MultiRipper extracts files from Windows and Delphi/C++ Builder applications.
- Feature
  - Easy-to-use wizard interface
  - Extraction from both Delphi and C++
  - Builder exe-files
  - Extraction of all project forms and data modules with all assigned properties and events
  - Generation of Delphi projects
  - Restoration of all units from the exe-file

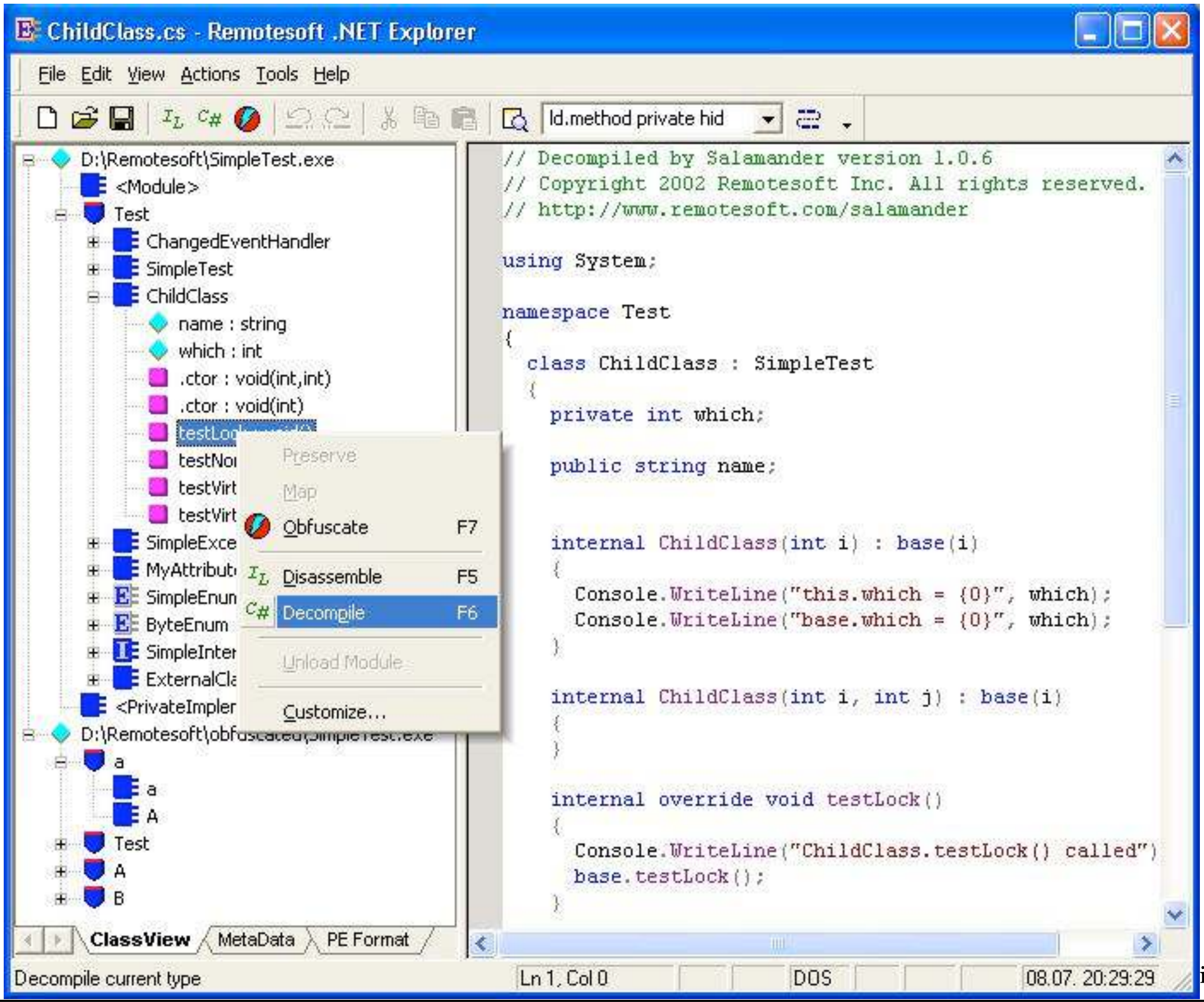
Tools for Decompiling .NET Applications

- Tools:
  - Salamander
  - Anakrino
  - LSW DotNet-Reflection-Browser
  - Lutz Roeder's Programming.NET
  - Dis#.
  - 9rays.net
  - Decompiler.net

Salamander .NET Decompiler

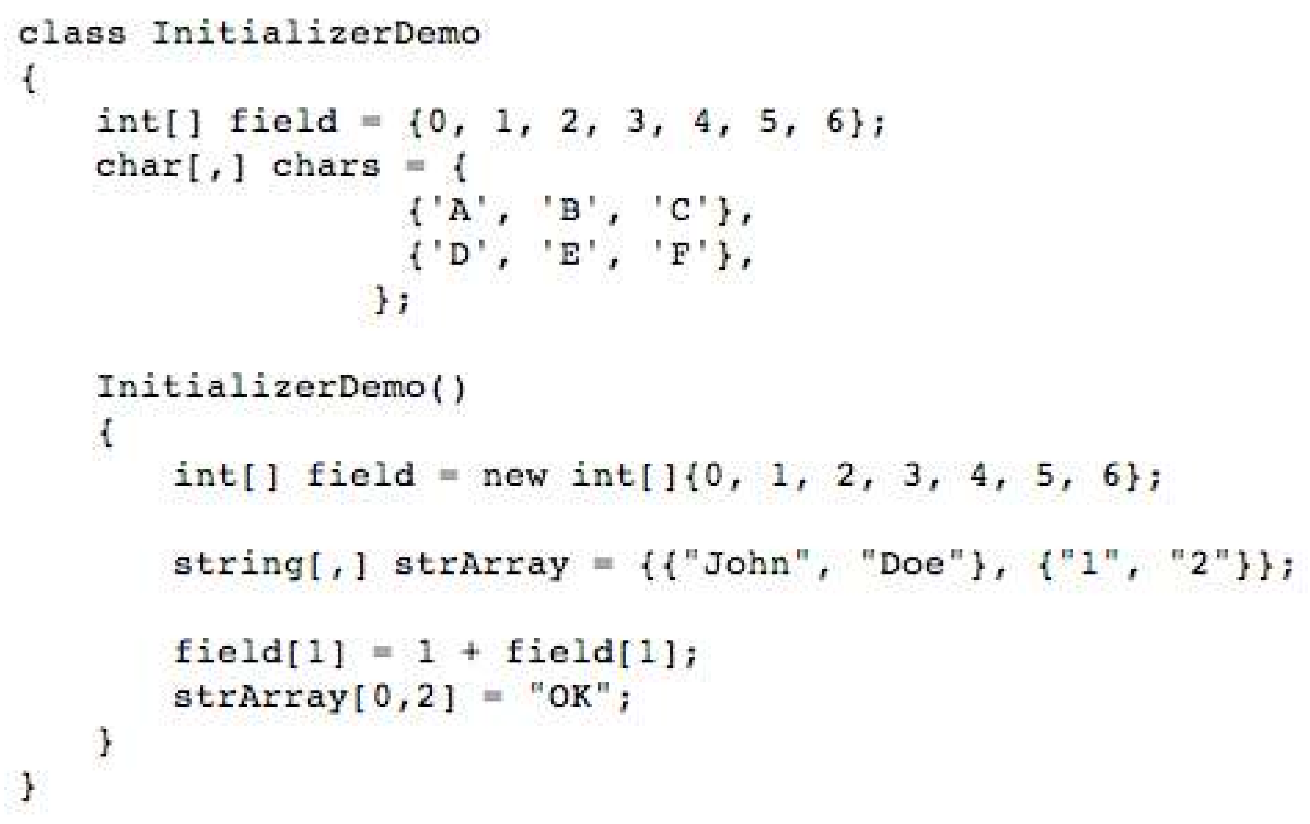
- Salamander is a .NET decompiler that converts executable files (.EXE or .DLL) from Intermediate Language (IL, MSIL, CIL) binary format to high-level source codes, such as C#, managed C++, Visual Basic.NET, etc.
- For more than 8,000 classes that have been tested, Salamander always produces equivalent and recompilable codes that are remarkably close to the original source codes.

Salamander Screenshot



```
// Decompiled by Salamander version 1.0.6
// Copyright 2002 Remotesoft Inc. All rights reserved.
// http://www.remotesoft.com/salamander

using System;

namespace Test
{
  class ChildClass : SimpleTest
  {
    private int which;

    public string name;


    internal ChildClass(int i) : base(i)
    {
      Console.WriteLine("this.which = {0}", which);
      Console.WriteLine("base.which = {0}", which);
    }

    internal ChildClass(int i, int j) : base(i)
    {
    }

    internal override void testLock()
    {
      Console.WriteLine("ChildClass.testLock() called")
      base.testLock();
    }
```

Salamander .NET Decompiler (Original Code)



```
class InitializerDemo
{
    int[] field = {0, 1, 2, 3, 4, 5, 6};
    char[,] chars = {
                    {'A', 'B', 'C'},
                    {'D', 'E', 'F'},
                  };

    InitializerDemo()
    {
        int[] field = new int[]{0, 1, 2, 3, 4, 5, 6};

        string[,] strArray = {{"John", "Doe"}, {"1", "2"}};

        field[1] = 1 + field[1];
        strArray[0,2] = "OK";
    }
}
```

EC-Council

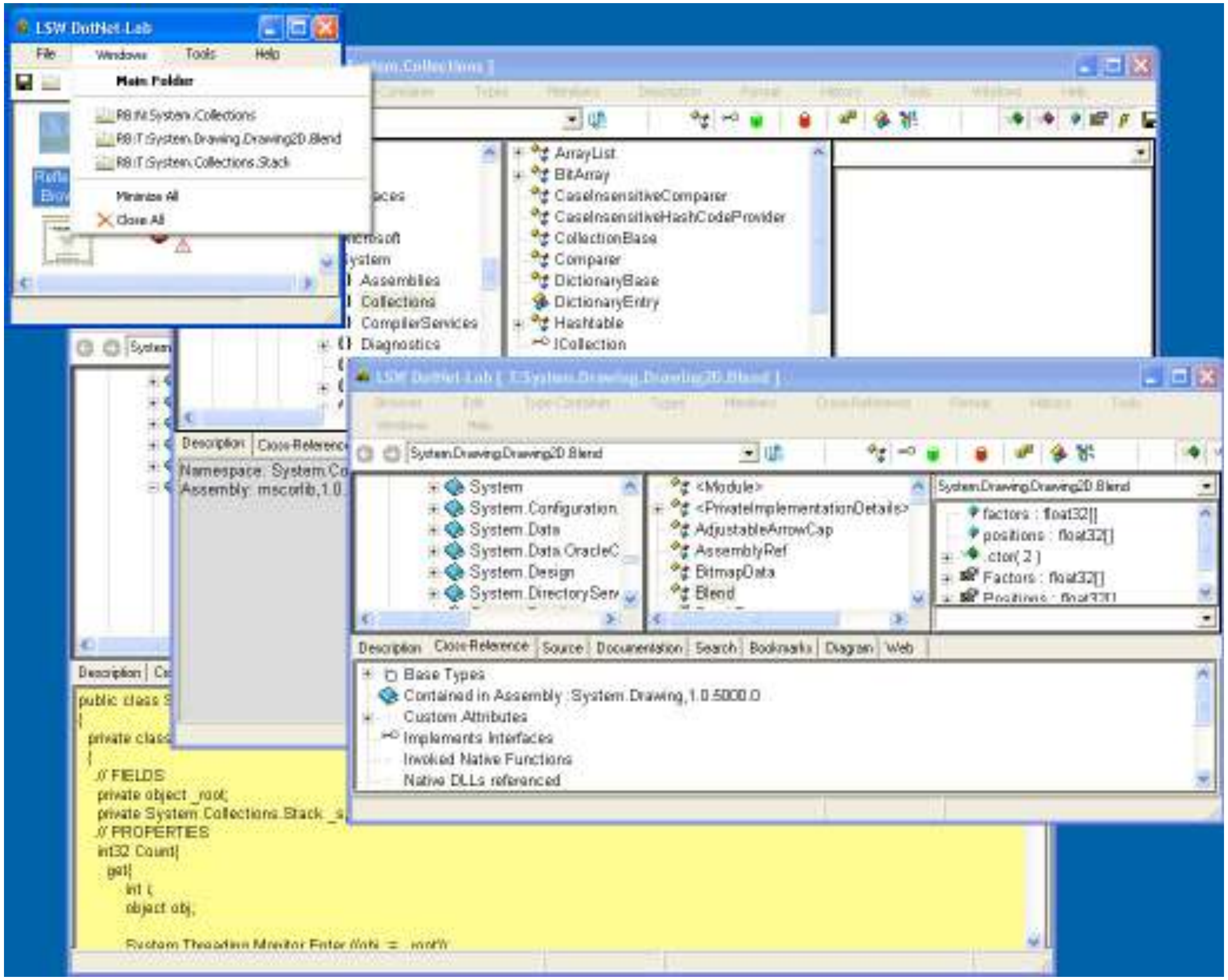
Salamander .NET Decompiler (Decompiled Code)

```
using System;
using System.Runtime.CompilerServices;

class InitializerDemo
{
  private int[] field = new int[]{0, 1, 2, 3, 4, 5, 6};

  private char[,] chars = new char[,]{
    {'A', 'B', 'C'},
    {'D', 'E', 'F'},
    };


  private InitializerDemo()
  {
    int[] nums = new int[]{0, 1, 2, 3, 4, 5, 6};
    string[,] strs2 = new string[2, 2];
    // need more work here
    strs2.Set(0, 0, "John");
    strs2.Set(0, 1, "Doe");
    strs2.Set(1, 0, "1");
    strs2.Set(1, 1, "2");
    string[,] strs1 = strs2;
    nums[1]++;
    strs1.Set(0, 2, "OK");
  }
}
```

Tool: LSW DotNet-Reflection-Browser

- LSW DotNet-Reflection-Browser is a commercial .NET object browser, disassembler, and decompiler.
- It is a native Windows application
- **LSW DotNet-Reflection-Browser** (**LSW-DNRB**) is a revolutionary Object- Browser for Microsofts .NET - Framework .
- It displays the Framework components and every component written for the Framework in same compact and convinient form as a Smalltalk System-Browser does.
- Information about .NET Components can be retrieved very fast

Tool: LSW DotNet-Reflection-Browser

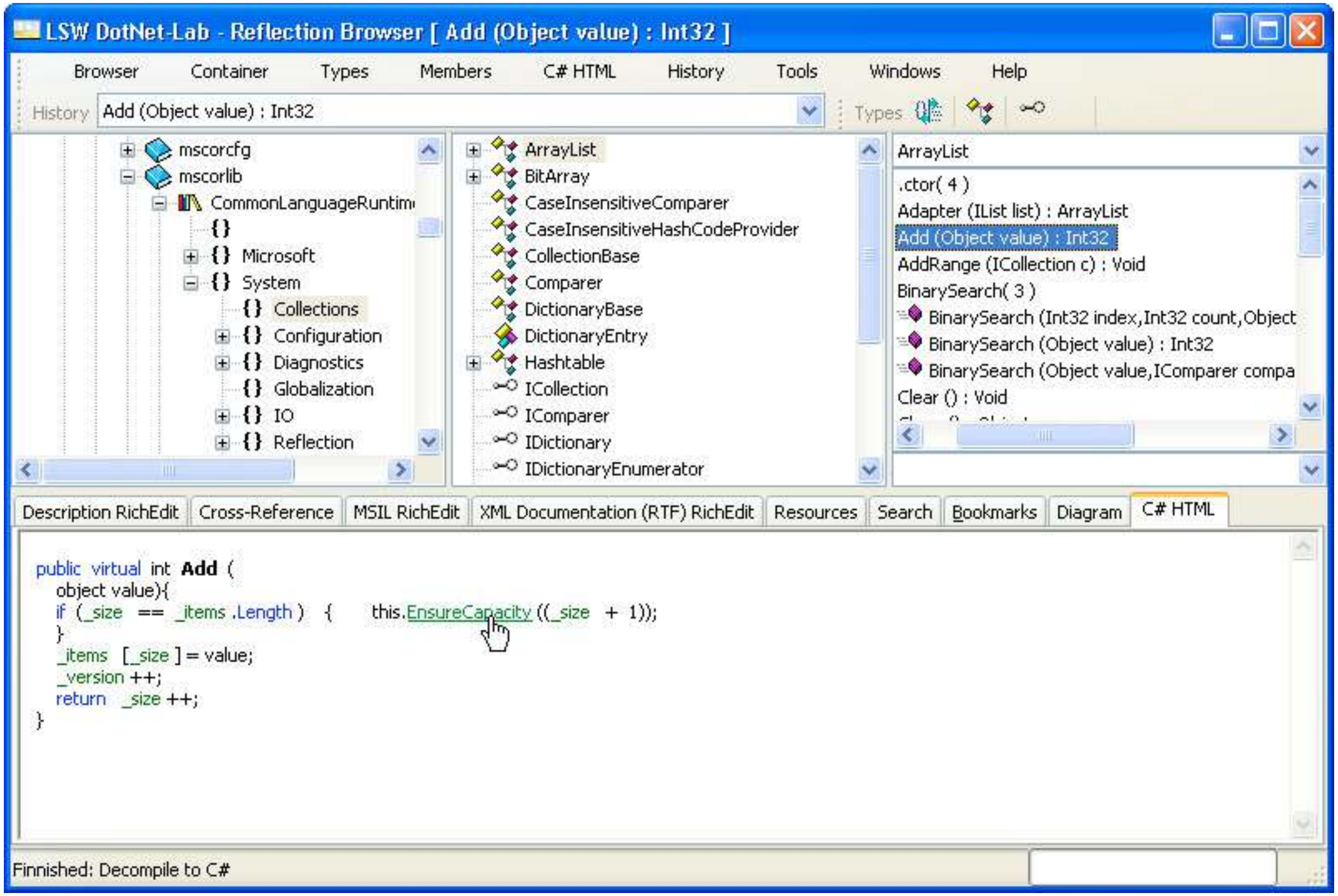


Namespace: System.Co
Assembly: mscorlib,1.0.

Base Types
Contained in Assembly :System.Drawing,1.0.5000.0
Custom Attributes
Implements Interfaces
Invoked Native Functions
Native DLLs referenced

```
public class S
{
  private class
  {
   // FIELDS
   private object _root;
   private System.Collections.Stack _s
   // PROPERTIES
   int32 Count{
     get{
       int i;
       object obj;

       System.Threading.Monitor.Enter((obj = _root));
```

Tool: LSW DotNet-Reflection-Browser


LSW DotNet-Lab - Reflection Browser [Add (Object value) : Int32]
Browser Container Types Members C# HTML History Tools Windows Help
History Add (Object value) : Int32
Types
mscorcfg
mscorlib
CommonLanguageRuntim
{}
Microsoft
System
Collections
Configuration
Diagnostics
Globalization
IO
Reflection
ArrayList
BitArray
CaseInsensitiveComparer
CaseInsensitiveHashCodeProvider
CollectionBase
Comparer
DictionaryBase
DictionaryEntry
Hashtable
ICollection
IComparer
IDictionary
IDictionaryEnumerator
ArrayList
.ctor(4)
Adapter (IList list) : ArrayList
Add (Object value) : Int32
AddRange (ICollection c) : Void
BinarySearch(3)
BinarySearch (Int32 index,Int32 count,Object
BinarySearch (Object value) : Int32
BinarySearch (Object value,IComparer compa
Clear () : Void
Description RichEdit
Cross-Reference
MSIL RichEdit
XML Documentation (RTF) RichEdit
Resources
Search
Bookmarks
Diagram
C# HTML

```
public virtual int Add (
    object value){
    if (_size == _items .Length )  {    this.EnsureCapacity ((_size + 1));
    }
    _items [_size ] = value;
    _version ++;
    return _size ++;
}
```

Finnished: Decompile to C#

EC-Council

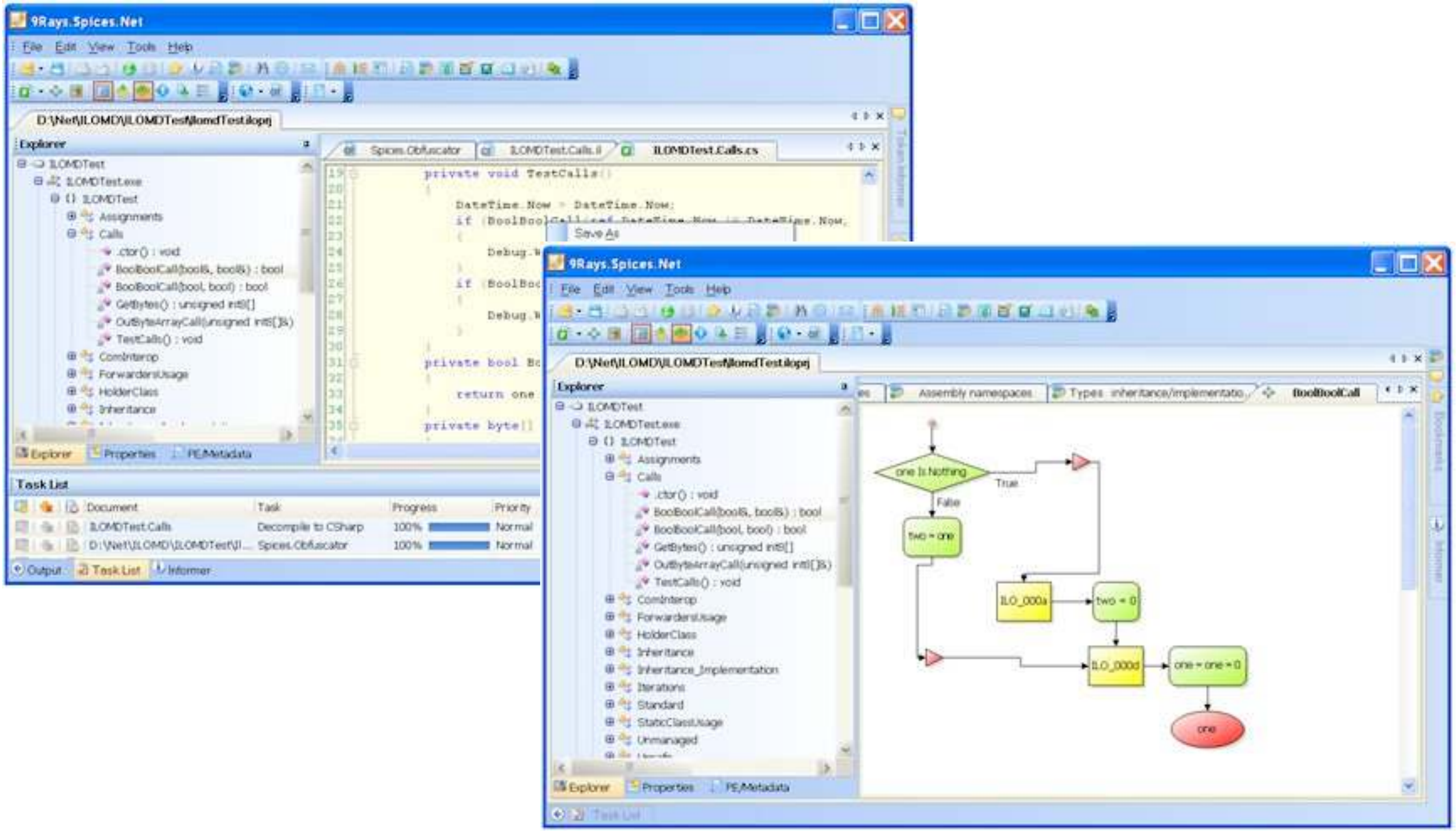
Tool: Reflector

- Reflector is a class browser for .NET components and assemblies.
- It features hierarchical assembly and namespace views, type and member dictionary index search, type reference search, custom attributes view, IL disassembler, C# decompiler, VB decompiler, viewers for C# XML docs and MSDN help.
- Assembly dependency trees, supertype/subtype hierarchies and resources can be inspected as well

Tool: Spices NET.Decompiler

- .Net Decompiler that decompiles /disassembles .Net assemblies from MSIL (MS Intermediate Language) binary format to well-formed and optimized source code (6 languages: MSIL, C#, VB.NET, Delphi.Net J# and managed C++).

Tool: Spices NET.Decompiler



```
private void TestCalls()
{
    DateTime Now = DateTime.Now;
    if (BoolBoolCall(ref DateTime.Now == DateTime.Now,
    {
        Debug.W
    }
    if (BoolBoo
    {
        Debug.W
    }
}
private bool Bo
    return one
}
private byte[]
```

Task List

Document	Task	Progress	Priority
ILOMDTest.Calls	Decompile to CSharp	100%	Normal
D:\Net\ILOMD\ILOMDTest\I...	Spices.Obfuscator	100%	Normal

Tool: Decompiler.net

- Decompiler.net is a combination decompiler, obfuscator, language translator, and refactoring tool for Microsoft .NET managed applications and libraries.

.NET Obfuscator and .NET Obfuscation

- Obfuscator for .NET protects intellectual property by making it extremely difficult to reverse engineer applications.
- Unprotected applications can easily be reverse engineered by decompiler tools.
- The .NET obfuscator's product counters this threat by transforming applications-- renaming identifiers to meaningless characters, obfuscating metadata, and altering control flow so that the obfuscated .NET code is much harder to understand.
- http://www.preemptive.com

Java Bytecode Decompilers

- Decompilers that read Java bytecode programs usually decompile to Java
- Tools:
  - JODE
  - Jad
  - Dava
  - SourceTec Java Decompiler
  - JReversePro
  - SourceAgain
  - ClassCracker 3
  - DCompiler
  - WingSoft
  - JReveal decompiler

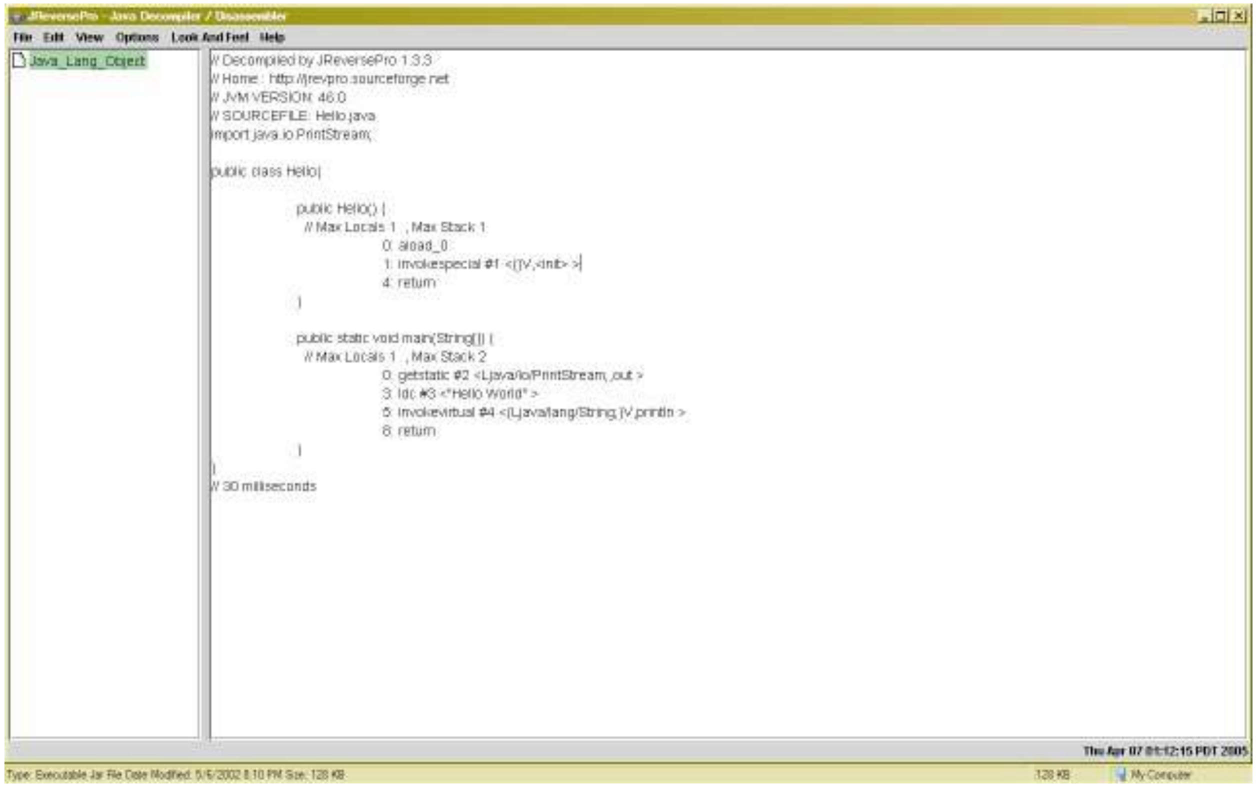
Tool: JODE Java Decompiler

- *JODE* is a java package containing a decompiler and an optimizer for java.
- This package is freely available under the GNU GPL.
- The bytecode package and the core decompiler is now under GNU Lesser General Public License, so you can integrate it in your project.
- The decompiler reads in class files and produces something similar to the original java file.

Tool: JREVERSEPRO

- JREVERSEPRO is a Java Decompiler / Disassembler written entirely in Java
- The software is written 100 % in Java which implies you can seamlessly integrate your java applications with this.
- The .class files could be disassembled to examine the JVM bytecode.

Tool: JREVERSEPRO


JReversePro - Java Decompiler / Disassembler
File Edit View Options Look And Feel Help
Java_Lang_Object

```
// Decompiled by JReversePro 1.3.3
// Home : http://jrevpro.sourceforge.net
// JVM VERSION: 46.0
// SOURCEFILE: Hello.java
import java.io.PrintStream;

public class Hello{

            public Hello() {
              // Max Locals 1  , Max Stack 1
                        0: aload_0
                        1: invokespecial #1 <()V,<init> >
                        4: return
            }

            public static void main(String[]) {
              // Max Locals 1  , Max Stack 2
                        0: getstatic #2 <Ljava/io/PrintStream;,out >
                        3: ldc #3 <"Hello World" >
                        5: invokevirtual #4 <(Ljava/lang/String;)V,println >
                        8: return
            }
}
// 30 milliseconds
```

Thu Apr 07 01:12:15 PDT 2005
Type: Executable Jar File Date Modified: 5/6/2002 8:10 PM Size: 128 KB
128 KB
My Computer

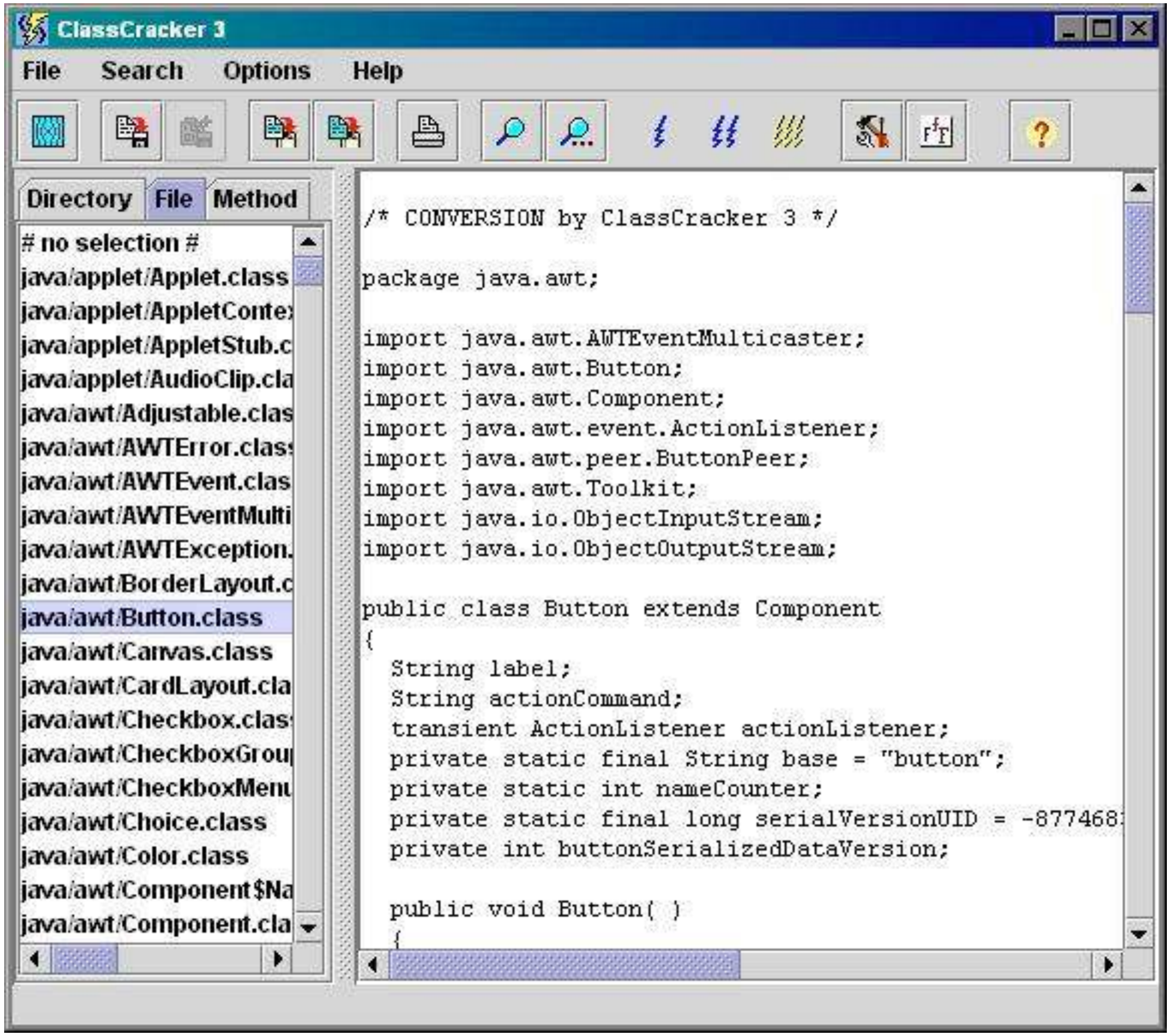
Tool: SourceAgain

- SourceAgain is a commercial Java decompiler by Ahpah Software.
- SourceAgain correctly recovers Java control structures and optimizations from the bytecode.
- It supports irreducible graphs, polymorphic type inference, recognition of packages, and more, and provides debugging support.

Tool: ClassCracker

- **ClassCracker 3** is a Java decompiler
  - A **Java decompiler** that retrieves Java source code from Java class files
  - A **Java disassembler** that produces JVM (Java Virtual Machine) bytecode; and ·
  - A **Java class file viewer** that displays Java class file structures.

Tool: ClassCracker



```
# no selection #
java/applet/Applet.class
java/applet/AppletConte
java/applet/AppletStub.c
java/applet/AudioClip.cla
java/awt/Adjustable.clas
java/awt/AWTError.clas
java/awt/AWTEvent.clas
java/awt/AWTEventMulti
java/awt/AWTException.
java/awt/BorderLayout.c
java/awt/Button.class
java/awt/Canvas.class
java/awt/CardLayout.cla
java/awt/Checkbox.clas
java/awt/CheckboxGrou
java/awt/CheckboxMenu
java/awt/Choice.class
java/awt/Color.class
java/awt/Component$Na
java/awt/Component.cla
```

```
/* CONVERSION by ClassCracker 3 */

package java.awt;

import java.awt.AWTEventMulticaster;
import java.awt.Button;
import java.awt.Component;
import java.awt.event.ActionListener;
import java.awt.peer.ButtonPeer;
import java.awt.Toolkit;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class Button extends Component
{
  String label;
  String actionCommand;
  transient ActionListener actionListener;
  private static final String base = "button";
  private static int nameCounter;
  private static final long serialVersionUID = -877468
  private int buttonSerializedDataVersion;

  public void Button( )
  {
```

EC-Council

Python Decompilers

- The decompyle service decompiles Python byte-code (in .pyc or .pyo files) into python source code.
- The 'decompyle' service converts Python byte-code back into equivalent Python source.
- It accepts byte-code from any Python version starting with 1.5 up to 2.3.3
- http://www.crazy-compilers.com

Reverse Engineering Tutorial

OllyDbg Debugger

- OllyDbg is a 32-bit assembler level analysing debugger for Microsoft Windows
- Emphasis on **binary code analysis** makes it particularly useful in cases where source is unavailable
- Features:
  - **Code analysis - traces registers**
  - **Recognizes procedures**
  - **Loops, API calls**
  - **Switches, tables, constants and strings**
  - **Directly loads and debugs DLLs**
  - **Object file scanning - locates routines from object files and libraries**

How does OllyDbg work?

- Code is executed step by step, and debugger protocols every command, together with registers and flags, in the large circular buffer
- When exception occurs, you can backtrace several (or hundreds of thousands) last commands and analyze conditions that led to error
- Run trace shows modified registers and keeps important messages and operands of known functions
- You can set conditions to pause run trace

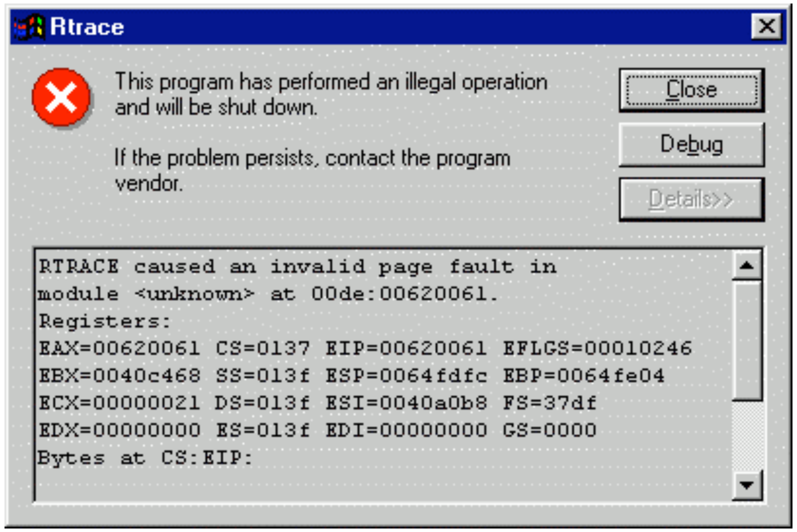
Lets debug a simple console application

```
#include <stdio.h>
void f1(void) { printf("a"); };
void f2(void) { printf("b"); };
void f3(void) { printf("c"); };
void (*f[3])() = { f1,f2,f3 };
void main(void) {
  int i,j,k;
  for (i=0; i<100; i++) {
    for (j=0; j<1000000; j++) ;  // Long code
    k=i/33;
    if (k>3) continue;
    f[k]();                      // Here error (when i==99)!
  };
  printf("\n");
};
```

EC-Council

Error

- Functions f1, f2 and f3 print letters a, b and c. Main program calls each function 33 times, then prints newline and terminates...



This program has performed an illegal operation and will be shut down.

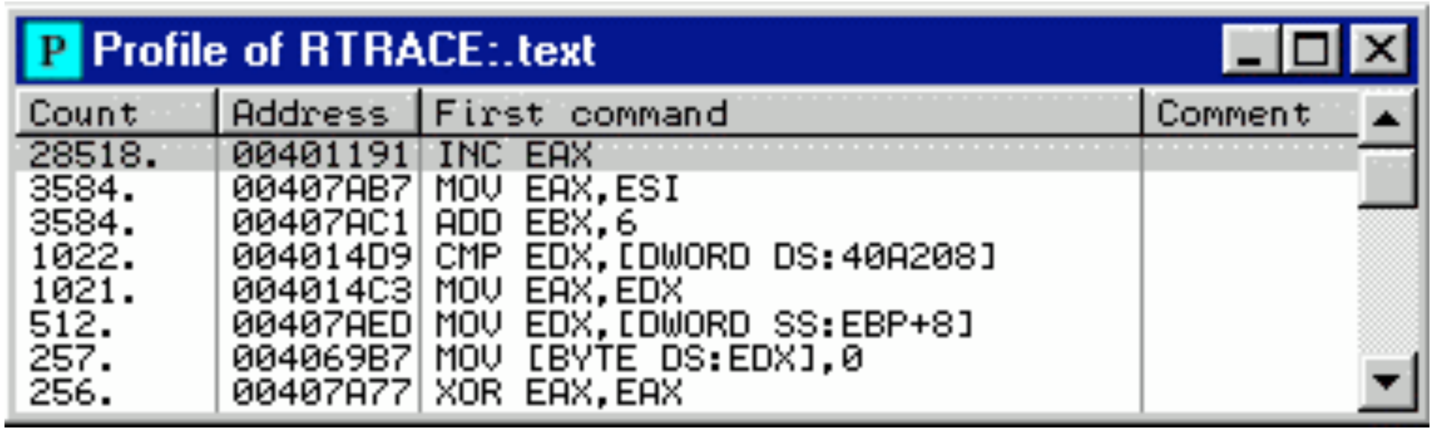
If the problem persists, contact the program vendor.

```
RTRACE caused an invalid page fault in
module <unknown> at 00de:00620061.
Registers:
EAX=00620061 CS=0137 EIP=00620061 EFLGS=00010246
EBX=0040c468 SS=013f ESP=0064fdfc EBP=0064fe04
ECX=00000021 DS=013f ESI=0040a0b8 FS=37df
EDX=00000000 ES=013f EDI=00000000 GS=0000
Bytes at CS:EIP:
```

EC-Council

Tracing

- Load the trace program in OllDbg dissembler and debug the program
- Note that points to Address 00620061 which is nowhere
- You are trying to figure which command jumped to this location
- Let's try from the very beginning. Press Ctrl+F2 (shortcut for Restart), then Ctrl+F11 (Trace into) and wait for a minute or two
- Pause run trace by pressing F12 (Pause) or Esc. In the Executable modules, click on RTRACE and select "View run trace profile":

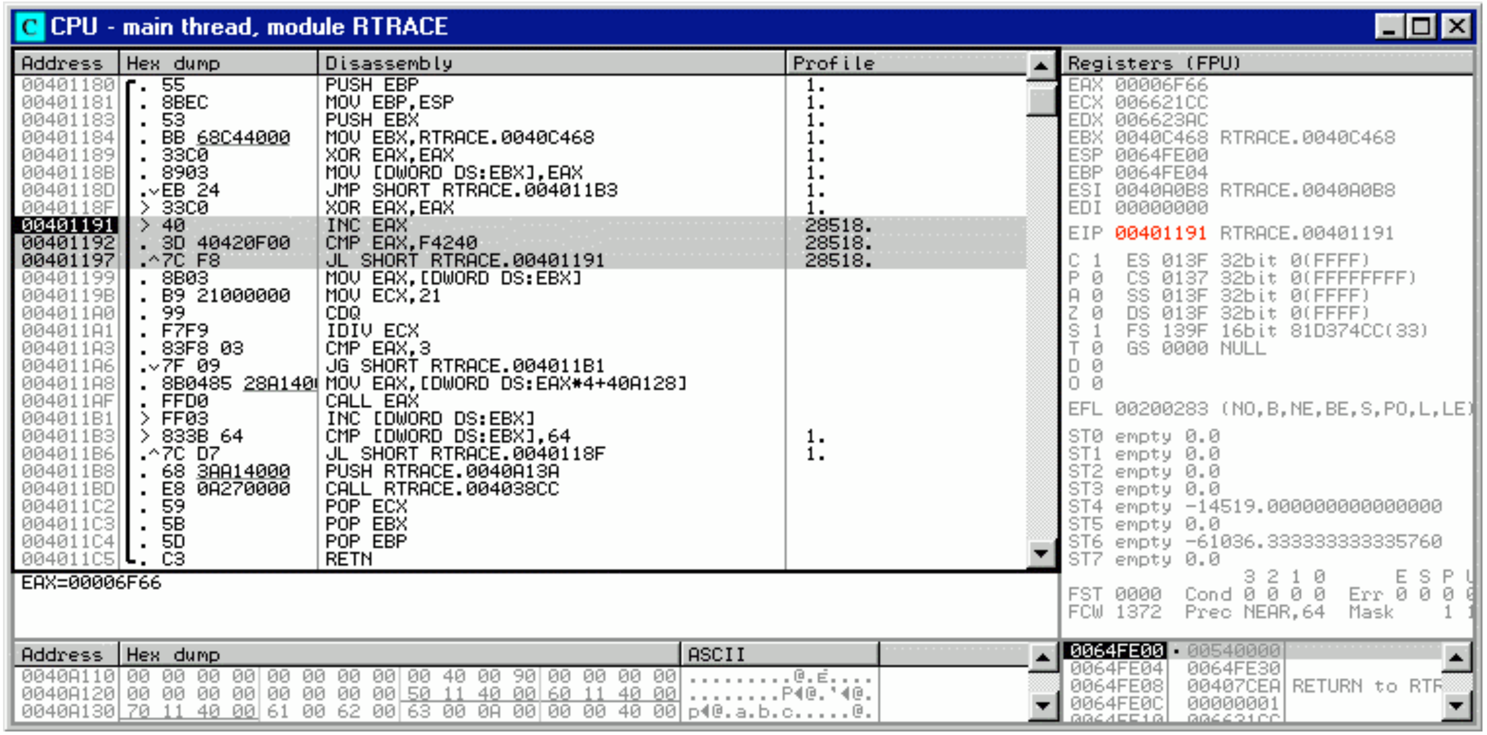


```
Count   Address   First command                      Comment
28518.  00401191  INC EAX
3584.   00407AB7  MOV EAX,ESI
3584.   00407AC1  ADD EBX,6
1022.   004014D9  CMP EDX,[DWORD DS:40A208]
1021.   004014C3  MOV EAX,EDX
512.    00407AED  MOV EDX,[DWORD SS:EBP+8]
257.    004069B7  MOV [BYTE DS:EDX],0
256.    00407A77  XOR EAX,EAX
```

EC-Council

Repetition of Commands

- A command or a sequence of commands at address 00401191 was executed more than 24000 times. Follow this line in Disassembler:



CPU - main thread, module RTRACE

```
Address   Hex dump          Disassembly                              Profile
00401180 ┌. 55              PUSH EBP                                  1.
00401181 │. 8BEC            MOV EBP,ESP                               1.
00401183 │. 53              PUSH EBX                                  1.
00401184 │. BB 68C44000     MOV EBX,RTRACE.0040C468                   1.
00401189 │. 33C0            XOR EAX,EAX                               1.
0040118B │. 8903            MOV [DWORD DS:EBX],EAX                    1.
0040118D │.vEB 24           JMP SHORT RTRACE.004011B3                 1.
0040118F │> 33C0            XOR EAX,EAX                               1.
00401191 │> 40              INC EAX                                   28518.
00401192 │. 3D 40420F00     CMP EAX,F4240                             28518.
00401197 │.^7C F8           JL SHORT RTRACE.00401191                  28518.
00401199 │. 8B03            MOV EAX,[DWORD DS:EBX]
0040119B │. B9 21000000     MOV ECX,21
004011A0 │. 99              CDQ
004011A1 │. F7F9            IDIV ECX
004011A3 │. 83F8 03         CMP EAX,3
004011A6 │.v7F 09           JG SHORT RTRACE.004011B1
004011A8 │. 8B0485 28A140   MOV EAX,[DWORD DS:EAX*4+40A128]
004011AF │. FFD0            CALL EAX
004011B1 │> FF03            INC [DWORD DS:EBX]
004011B3 │> 833B 64         CMP [DWORD DS:EBX],64                     1.
004011B6 │.^7C D7           JL SHORT RTRACE.0040118F                  1.
004011B8 │. 68 3AA14000     PUSH RTRACE.0040A13A
004011BD │. E8 0A270000     CALL RTRACE.004038CC
004011C2 │. 59              POP ECX
004011C3 │. 5B              POP EBX
004011C4 │. 5D              POP EBP
004011C5 └. C3              RETN
EAX=00006F66
```

```
Registers (FPU)
EAX 00006F66
ECX 006621CC
EDX 006623AC
EBX 0040C468 RTRACE.0040C468
ESP 0064FE00
EBP 0064FE04
ESI 0040A0B8 RTRACE.0040A0B8
EDI 00000000

EIP 00401191 RTRACE.00401191

C 1  ES 013F 32bit 0(FFFF)
P 0  CS 0137 32bit 0(FFFFFFFF)
A 0  SS 013F 32bit 0(FFFF)
Z 0  DS 013F 32bit 0(FFFF)
S 1  FS 139F 16bit 81D374CC(33)
T 0  GS 0000 NULL
D 0
O 0

EFL 00200283 (NO,B,NE,BE,S,PO,L,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty -14519.000000000000000
ST5 empty 0.0
ST6 empty -61036.333333333335760
ST7 empty 0.0
               3 2 1 0      E S P U
FST 0000  Cond 0 0 0 0  Err 0 0 0 0
FCW 1372  Prec NEAR,64  Mask    1 1
```

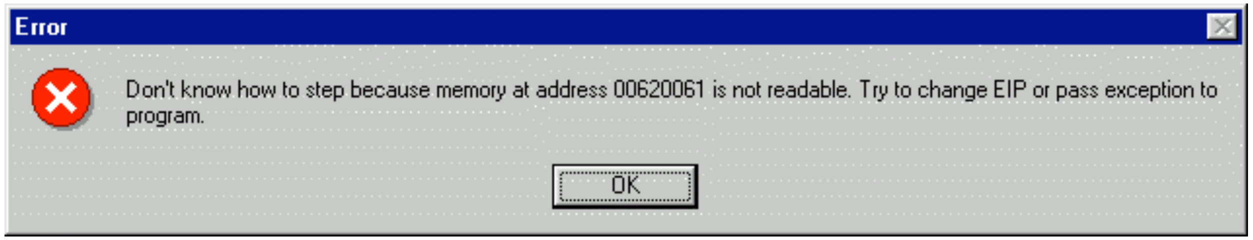
```
Address   Hex dump                                         ASCII
0040A110  00 00 00 00 00 00 00 00 00 40 00 90 00 00 00 00  .........@.É....
0040A120  00 00 00 00 00 00 00 00 50 11 40 00 60 11 40 00  ........P◄@.`◄@.
0040A130  70 11 40 00 61 00 62 00 63 00 0A 00 00 00 40 00  p◄@.a.b.c.....@.
```

```
0064FE00  00540000
0064FE04  0064FE30
0064FE08  00407CEA  RETURN to RTR
0064FE0C  00000001
```

EC-Council

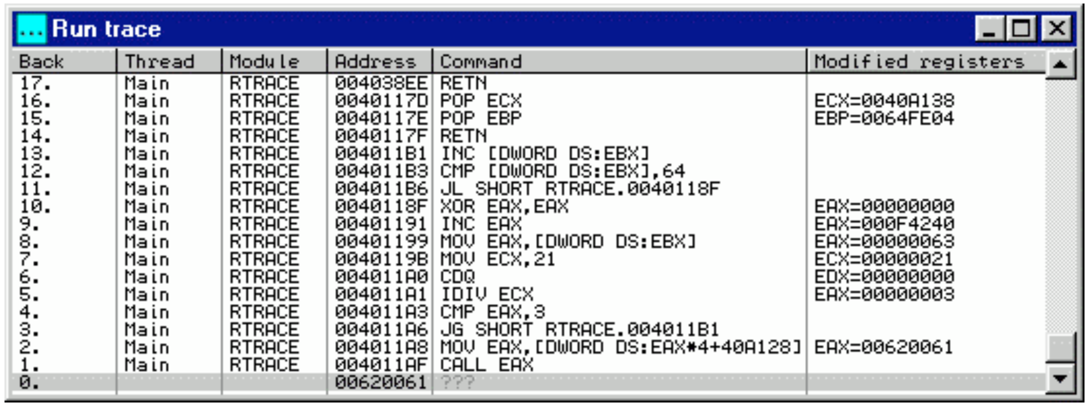
Error EIP

- A short 3-command cycle executes F4240 (decimal 1000000) times.
- At 5000 commands per second, OllyDbg will need 10 minutes to trace this cycle.
- Note that sequence is quasi-linear, i.e. has no jumps to outside.
- From the pop-up menu, choose "Run trace|Skip selection when tracing".
- Red line in the fourth column indicates that commands are excluded from run trace.
- When OllyDbg encounters excluded sequence, it sets temporary breakpoint at the command that immediately follows excluded block (in our case, 00401199) and runs it at once.


Error

Don't know how to step because memory at address 00620061 is not readable. Try to change EIP or pass exception to program.

- Confirm this error, open Run trace window (button with period '...' in the toolbar) and scroll it to the bottom:

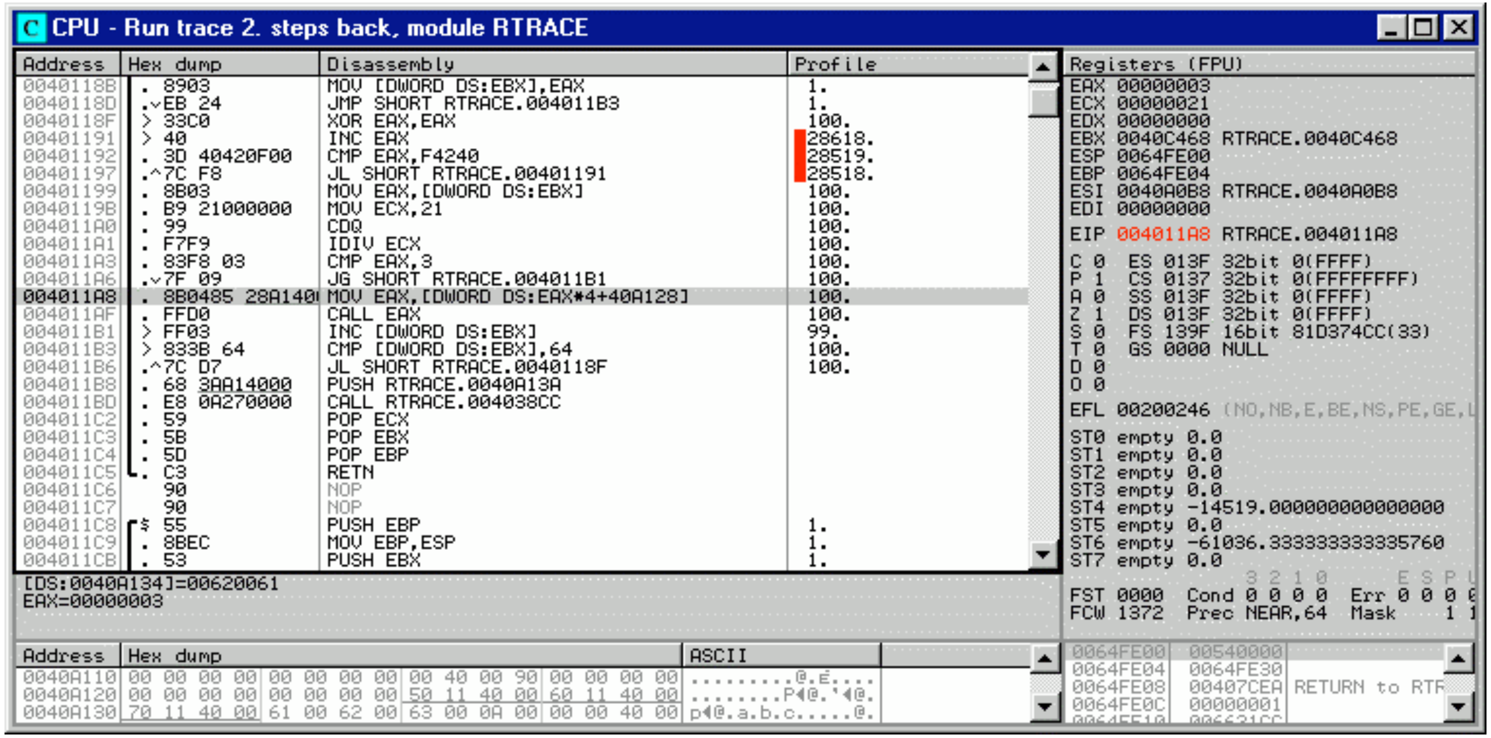


Run trace

```
Back    Thread  Module  Address   Command                            Modified registers
17.     Main    RTRACE  004038EE  RETN
16.     Main    RTRACE  0040117D  POP ECX                            ECX=0040A138
15.     Main    RTRACE  0040117E  POP EBP                            EBP=0064FE04
14.     Main    RTRACE  0040117F  RETN
13.     Main    RTRACE  004011B1  INC [DWORD DS:EBX]
12.     Main    RTRACE  004011B3  CMP [DWORD DS:EBX],64
11.     Main    RTRACE  004011B6  JL SHORT RTRACE.0040118F
10.     Main    RTRACE  0040118F  XOR EAX,EAX                        EAX=00000000
9.      Main    RTRACE  00401191  INC EAX                            EAX=000F4240
8.      Main    RTRACE  00401199  MOV EAX,[DWORD DS:EBX]             EAX=00000063
7.      Main    RTRACE  0040119B  MOV ECX,21                         ECX=00000021
6.      Main    RTRACE  004011A0  CDQ                                EDX=00000000
5.      Main    RTRACE  004011A1  IDIV ECX                           EAX=00000003
4.      Main    RTRACE  004011A3  CMP EAX,3
3.      Main    RTRACE  004011A6  JG SHORT RTRACE.004011B1
2.      Main    RTRACE  004011A8  MOV EAX,[DWORD DS:EAX*4+40A128]    EAX=00620061
1.      Main    RTRACE  004011AF  CALL EAX
0.                      00620061  ???
```

Call Eax

- Now we can see that command that jumped to 00620061 was CALL EAX at 004011AF, and invalid address was calculated one command before the call.
- Doubleclick this line to see it in Disassembler. Registers and information are grayed to emphasize that they are not actual, but taken from the trace:



CPU - Run trace 2. steps back, module RTRACE

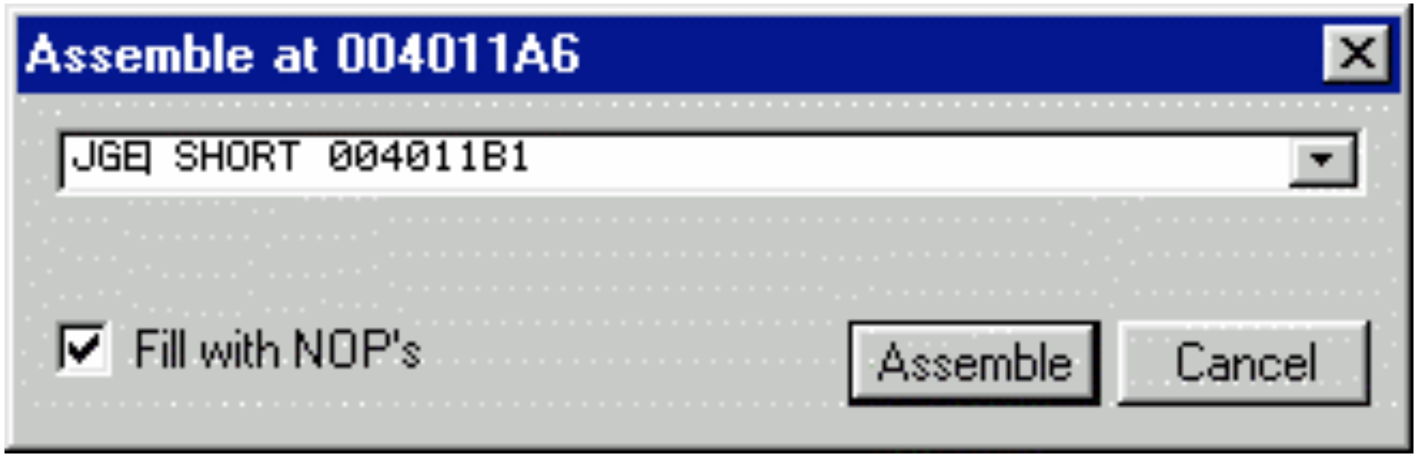
```
Address   Hex dump           Disassembly                          Profile
0040118B  . 8903             MOV [DWORD DS:EBX],EAX               1.
0040118D  .vEB 24            JMP SHORT RTRACE.004011B3            1.
0040118F  > 33C0             XOR EAX,EAX                          100.
00401191  > 40               INC EAX                              28618.
00401192  . 3D 40420F00      CMP EAX,F4240                        28519.
00401197  .^7C F8            JL SHORT RTRACE.00401191             28518.
00401199  . 8B03             MOV EAX,[DWORD DS:EBX]               100.
0040119B  . B9 21000000      MOV ECX,21                           100.
004011A0  . 99               CDQ                                  100.
004011A1  . F7F9             IDIV ECX                             100.
004011A3  . 83F8 03          CMP EAX,3                            100.
004011A6  .v7F 09            JG SHORT RTRACE.004011B1             100.
004011A8  . 8B0485 28A140    MOV EAX,[DWORD DS:EAX*4+40A128]      100.
004011AF  . FFD0             CALL EAX                             100.
004011B1  > FF03             INC [DWORD DS:EBX]                   99.
004011B3  > 833B 64          CMP [DWORD DS:EBX],64                100.
004011B6  .^7C D7            JL SHORT RTRACE.0040118F             100.
004011B8  . 68 3AA14000      PUSH RTRACE.0040A13A
004011BD  . E8 0A270000      CALL RTRACE.004038CC
004011C2  . 59               POP ECX
004011C3  . 5B               POP EBX
004011C4  . 5D               POP EBP
004011C5  . C3               RETN
004011C6    90               NOP
004011C7    90               NOP
004011C8  $ 55               PUSH EBP                             1.
004011C9  . 8BEC             MOV EBP,ESP                          1.
004011CB  . 53               PUSH EBX                             1.
[DS:0040A134]=00620061
EAX=00000003

Address   Hex dump                                          ASCII
0040A110  00 00 00 00 00 00 00 00 00 40 00 90 00 00 00 00  .........@.É....
0040A120  00 00 00 00 00 00 00 00 50 11 40 00 60 11 40 00  ........P◄@.`◄@.
0040A130  70 11 40 00 61 00 62 00 63 00 0A 00 00 00 40 00  p◄@.a.b.c.....@.
```

```
Registers (FPU)
EAX 00000003
ECX 00000021
EDX 00000000
EBX 0040C468 RTRACE.0040C468
ESP 0064FE00
EBP 0064FE04
ESI 0040A0B8 RTRACE.0040A0B8
EDI 00000000

EIP 004011A8 RTRACE.004011A8

C 0  ES 013F 32bit 0(FFFF)
P 1  CS 0137 32bit 0(FFFFFFFF)
A 0  SS 013F 32bit 0(FFFF)
Z 1  DS 013F 32bit 0(FFFF)
S 0  FS 139F 16bit 81D374CC(33)
T 0  GS 0000 NULL
D 0
O 0

EFL 00200246 (NO,NB,E,BE,NS,PE,GE,L

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty -14519.0000000000000000
ST5 empty 0.0
ST6 empty -61036.333333333335760
ST7 empty 0.0
                3 2 1 0      E S P U
FST 0000  Cond 0 0 0 0  Err 0 0 0 0
FCW 1372  Prec NEAR,64  Mask      1 1

0064FE00  00540000
0064FE04  0064FE30
0064FE08  00407CEA  RETURN to RTF
0064FE0C  00000001
```

EC-Council

Invalid Jump

- Address constant 0040A128 points to array of 3 fixups containing addresses of functions f1, f2 and f3.
- When this command was executed for the last time, EAX contained index 3, outside the array's bounds.
- Two previous commands should perform bounds checking, but condition is invalid: jump is taken when EAX is greater than 3. Correct condition would be "greater or equal".
- Doubleclick invalid line and correct condition:



Assemble at 004011A6

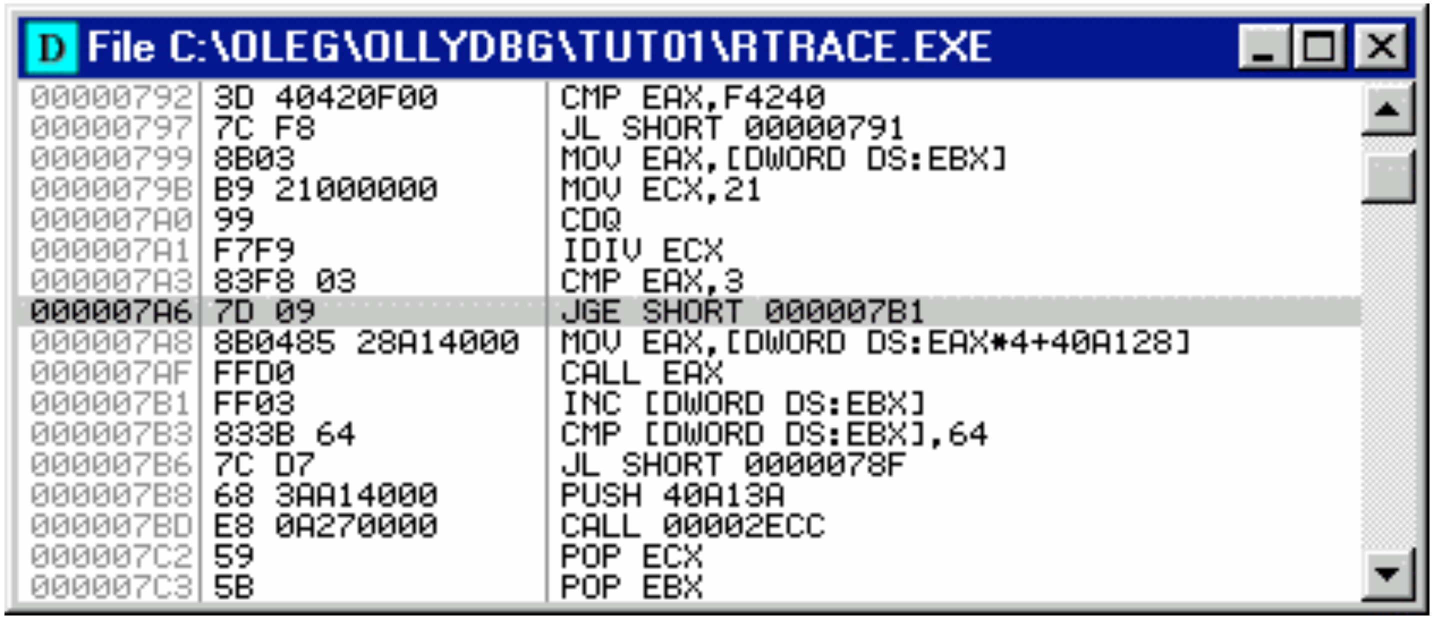
```
JGE SHORT 004011B1
```

[✓] Fill with NOP's

EC-Council

Copy to Executable File

- After you assemble new command, line in Disassembler gets red, indicating that command is modified.
- Select it again and in the pop-up menu choose the powerful item "Copy to executable file".
- This applies your modification directly to the executable file
- Save the modified file under a different name



```
00000792  3D 40420F00     CMP EAX,F4240
00000797  7C F8           JL SHORT 00000791
00000799  8B03            MOV EAX,[DWORD DS:EBX]
0000079B  B9 21000000     MOV ECX,21
000007A0  99              CDQ
000007A1  F7F9            IDIV ECX
000007A3  83F8 03         CMP EAX,3
000007A6  7D 09           JGE SHORT 000007B1
000007A8  8B0485 28A14000 MOV EAX,[DWORD DS:EAX*4+40A128]
000007AF  FFD0            CALL EAX
000007B1  FF03            INC [DWORD DS:EBX]
000007B3  833B 64         CMP [DWORD DS:EBX],64
000007B6  7C D7           JL SHORT 0000078F
000007B8  68 3AA14000     PUSH 40A13A
000007BD  E8 0A270000     CALL 00002ECC
000007C2  59              POP ECX
000007C3  5B              POP EBX
```

EC-Council

End of Slides