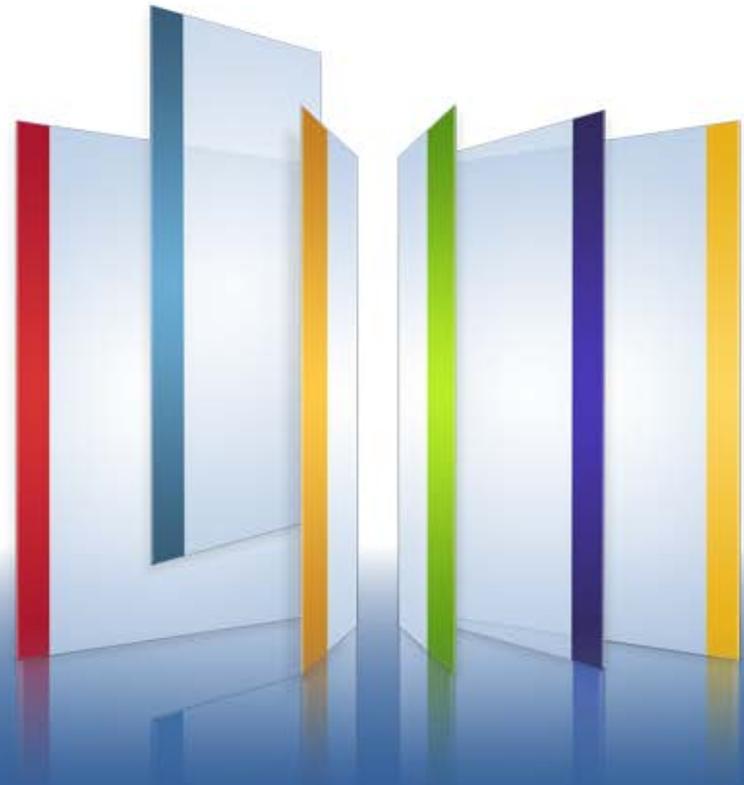




Introduction to Reverse Engineering



Inbar Raz
Malware Research Lab Manager
December 2011

What is Reverse Engineering?

Reverse engineering is the process of discovering the technological principles of a device, object, or system through analysis of its structure, function, and operation.

aka: Reversing, RE, SRE

WIKIPEDIA



Why do it?

Discover
Trade
Secrets

Find
Vulnerabilities

Academic
Research
(Yeah, right...)



Circumvent
[Copy]
Protection

Patch Binary
and
Alter Behavior

Pure
Curiosity

Analyse
Protocols

Sounds
awesome,
right?



So where's the catch?

■ Low-level is, well, low level...

```
00401000 push    ebp
00401001 mov     ebp, esp
00401003 push    ecx
00401004 push    ecx
00401005 and     dword ptr [ebp-4], 0
00401009 push    esi
0040100A mov     esi, [ebp+8]
for (Serial = 0, i = 0; i < strlen(UserName); i++) {
CurChar = (int) UserName[i];
Serial += CurChar;
0040100F call   ds:[00402008h]
Serial = (((Serial << 1) && 0xFFFFFFFF) | ((Serial >> 31) && 1));
Serial = (((Serial * CurChar) + CurChar) ^ CurChar);
}
UserSerial = ~(UserSerial ^ 0x1337C0DE) ^ 0xBADC0DE5;
0040101B jle    00401047h
0040101D movsx  ecx, byte ptr [edx+esi]
00401021 add     [ebp-4], ecx
00401024 mov     [ebp-8], ecx
00401027 rol     dword ptr [ebp-4], 1
0040102A mov     eax, ecx
0040102C imul  eax, [ebp-4]
00401030 mov     [ebp-4], eax
00401033 mov     eax, [ebp-8]
00401036 add     [ebp-4], eax
00401039 xor     [ebp-4], ecx
0040103C inc     edx
0040103D cmp     edx, edi
0040103F jl     0040101Bh
00401041 cmp     dword ptr [ebp-4], 0
```

So where's the catch?

- Low-level is, well, low level...
- Needle in a haystack
 - Average opcode size:
3 bytes
 - Average executable size:
500KB (on WinXP)
 - There are executables,
libraries, drivers....



www.jolyon.co.uk

So where's the catch?

- Low-level is, well, low level...
- Needle in a haystack
- Sometimes, the code resists
 - Packers and compressors
 - Obfuscators



So where's the catch?

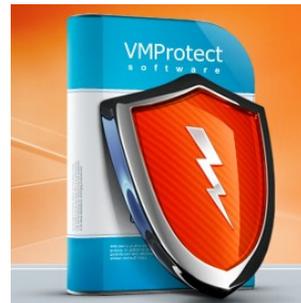
- Low-level is, well, low level...
- Needle in a haystack
- Sometimes, the code resists
- Sometimes, the code fights back
 - Detect reversing tools
 - Detect VMs and emulators



The Enigma Protector
Software Protection



ASPACK
SOFTWARE



softwareblades™

- Video clip: The Battle of Wits, “The Princess Bride”



A Battle of Wits

- Author writes code
- Reverser reverses it
- Author creates an anti-reversing technique
- Reverser bypasses it
- And so on...



So what do you need
in order to be
a good reverser?

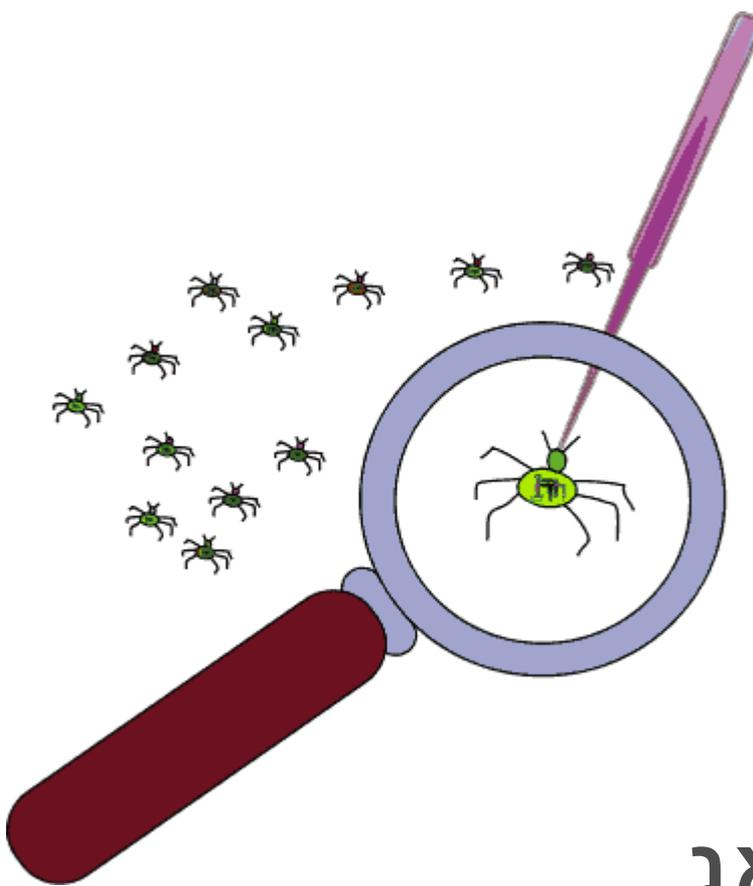


We'll come back
to this....



- Debugger (Dynamic code analysis)
 - Disassembler (Static code analysis)
 - Hex Editor
 - PE Analyzer
 - Resource Editor
- and more...





Debuggers

באג בדיזיין – זין בדיבאג

First, there was DEBUG...

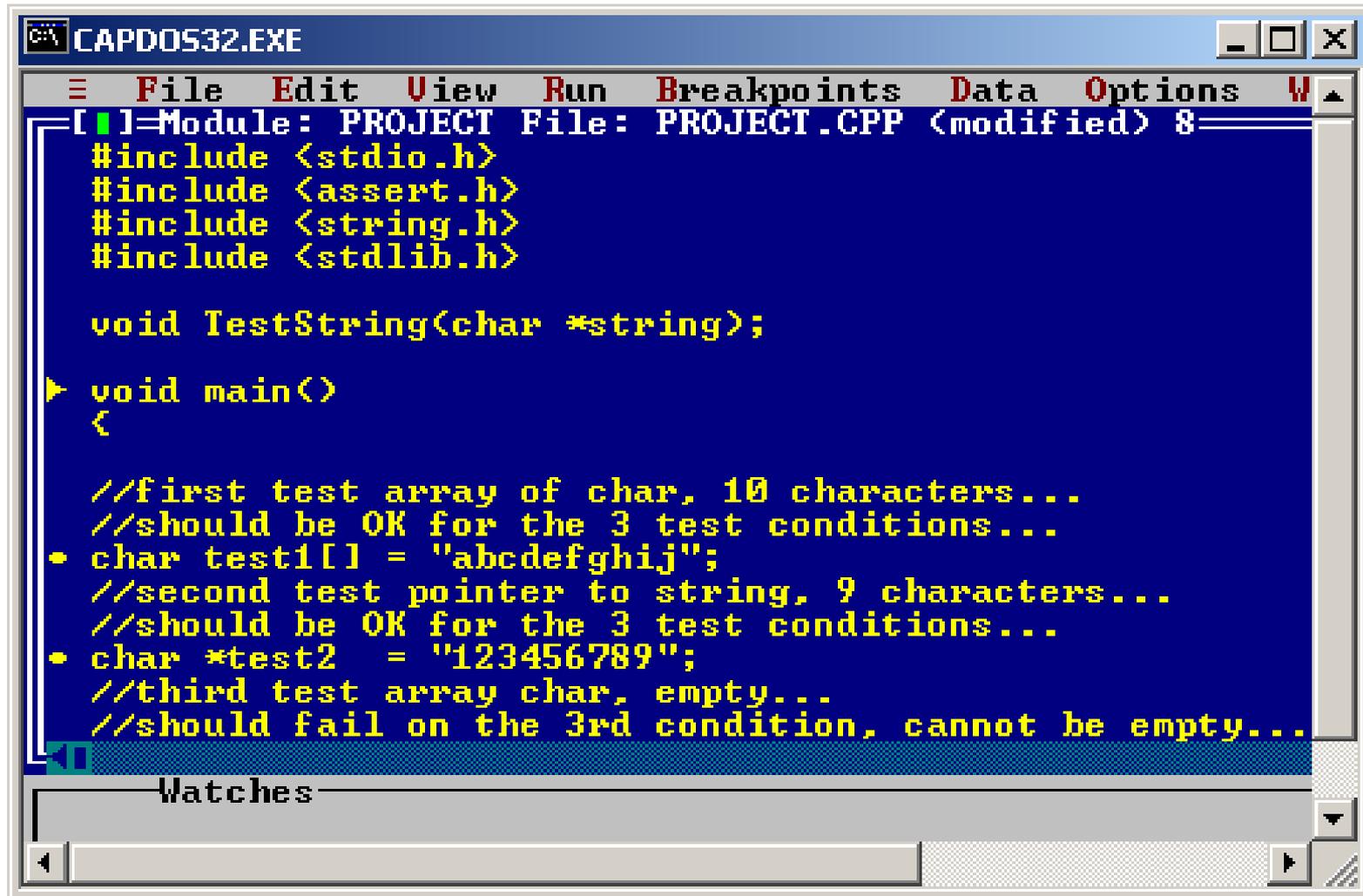
```
C:\WINDOWS\system32\cmd.exe - debug lib.exe
C:\PROGRAM~1\MICROS~2.0\UC\bin>debug lib.exe
-u 0 1 10
0B62:0000 0E          PUSH  CS
0B62:0001 1F          POP   DS
0B62:0002 BA0E00     MOV   DX,000E
0B62:0005 B409     MOV   AH,09
0B62:0007 CD21     INT   21
0B62:0009 B8014C     MOV   AX,4C01
0B62:000C CD21     INT   21
0B62:000E 54          PUSH  SP
0B62:000F 68          DB    68
-p
AX=0000 BX=0000 CX=3548 DX=0000 SP=00B6 BP=0000 SI=0000 DI=0000
DS=0B52 ES=0B52 SS=0B62 CS=0B62 IP=0001  NU UP EI PL NZ NA PO NC
0B62:0001 1F          POP   DS
-p
AX=0000 BX=0000 CX=3548 DX=0000 SP=00B8 BP=0000 SI=0000 DI=0000
DS=0B62 ES=0B52 SS=0B62 CS=0B62 IP=0002  NU UP EI PL NZ NA PO NC
0B62:0002 BA0E00     MOV   DX,000E
-p
AX=0000 BX=0000 CX=3548 DX=000E SP=00B8 BP=0000 SI=0000 DI=0000
DS=0B62 ES=0B52 SS=0B62 CS=0B62 IP=0005  NU UP EI PL NZ NA PO NC
0B62:0005 B409     MOV   AH,09
-p
AX=0900 BX=0000 CX=3548 DX=000E SP=00B8 BP=0000 SI=0000 DI=0000
DS=0B62 ES=0B52 SS=0B62 CS=0B62 IP=0007  NU UP EI PL NZ NA PO NC
0B62:0007 CD21     INT   21
-de 130
0B62:0000                                     54 68                                     Th
0B62:0010 69 73 20 70 72 6F 67 72-61 6D 20 63 61 6E 6E 6F   is program canno
0B62:0020 74 20 62 65 20 72 75 6E-20 69 6E 20 44 4F 53 20   t be run in DOS
0B62:0030 6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00   mode....$.
-p
This program cannot be run in DOS mode.
AX=0924 BX=0000 CX=3548 DX=000E SP=00B8 BP=0000 SI=0000 DI=0000
DS=0B62 ES=0B52 SS=0B62 CS=0B62 IP=0009  NU UP EI PL NZ NA PO NC
0B62:0009 B8014C     MOV   AX,4C01
-
```



GUI and much more: Turbo Debugger

The screenshot shows the Turbo Debugger interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Run', 'Breakpoints', 'Data', 'Options', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons. The main window is titled 'MS-DOS Prompt - TD' and contains the following assembly code and register information:

```
MS-DOS Prompt - TD
Auto
File Edit View Run Breakpoints Data Options Window Help
[ ]=CPU 80486
cs:0000 B80851 mov ax,5108
cs:0003 8ED8 mov ds,ax
cs:0005 8EC0 mov es,ax
cs:0007 B80A00 mov ax,000A
cs:000A BB1400 mov bx,0014
cs:000D B91E00 mov cx,001E
cs:0010 BA2800 mov dx,0028
cs:0013 8A260100 mov ah,[0001]
cs:0017 BE0100 mov si,0001
cs:001A B44C mov ah,4C
cs:001C A00000 mov al,[0000]
cs:001F CD21 int 21
cs:0021 0000 add [bx+si],al
cs:0023 0000 add [bx+si],al
cs:0025 0000 add [bx+si],al
ds:0000 CD 20 00 A0 00 9A F0 FE = a U=
ds:0008 1D F0 E0 01 16 1E AA 01 +=α@-▲-@
ds:0010 16 1E 89 02 71 18 E3 09 -▲é@q↑π@
ds:0018 01 01 01 00 02 FF FF FF @@@ @
ds:0020 FF FF FF FF FF FF FF FF
ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 010A d=0
ds 50F5
es 50F5
ss 5109
cs 5105
ip 0000
ss:010C 0F70
ss:010A C00F
ss:0108 0000
ss:0106 0F70
ss:0104 C00F
F1=Help F2=Bkpt F3=Mod F4=Here F5=Zoom F6=Next F7=Trace F8=Step F9=Run F10=Menu
```



```

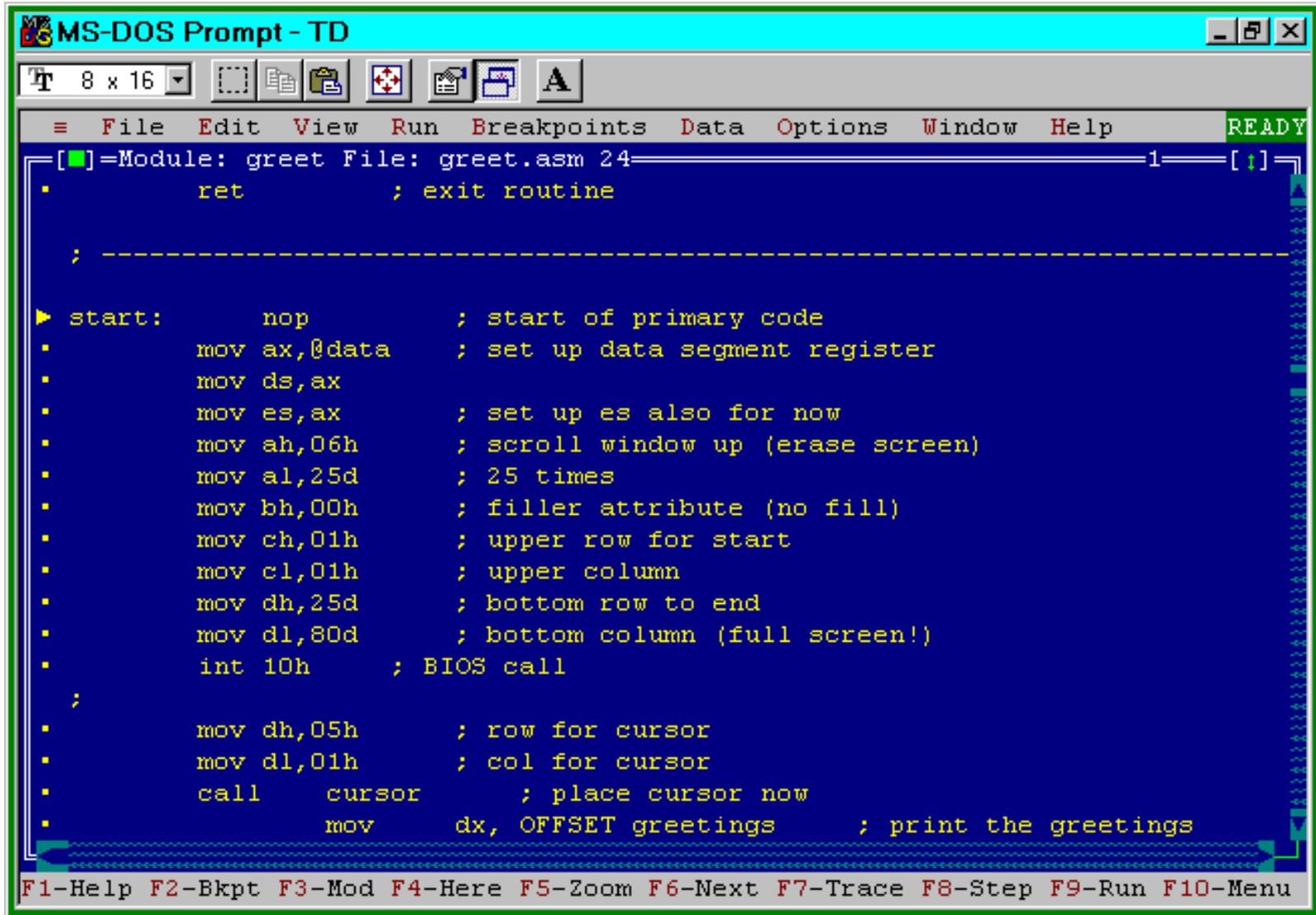
CAPDOS32.EXE
File Edit View Run Breakpoints Data Options W
[1] Module: PROJECT File: PROJECT.CPP (modified) 8
#include <stdio.h>
#include <assert.h>
#include <string.h>
#include <stdlib.h>

void TestString(char *string);
void main()
<

//first test array of char, 10 characters...
//should be OK for the 3 test conditions...
• char test1[] = "abcdefghij";
//second test pointer to string, 9 characters...
//should be OK for the 3 test conditions...
• char *test2 = "123456789";
//third test array char, empty...
//should fail on the 3rd condition, cannot be empty...

```





The screenshot shows the Turbo Debugger interface with the MS-DOS Prompt window open. The window title is "MS-DOS Prompt - TD". The menu bar includes File, Edit, View, Run, Breakpoints, Data, Options, Window, and Help. The status bar at the bottom shows keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu. The main window displays assembly code for a program named "greet.asm". The code includes a "start:" label and various instructions for setting up registers, scrolling the window, and printing greetings. A cursor is visible at the beginning of the "start:" line.

```
MS-DOS Prompt - TD
File Edit View Run Breakpoints Data Options Window Help
[ ]=Module: greet File: greet.asm 24-----1-----[ : ]
  ret          ; exit routine

; -----
start:        nop          ; start of primary code
  mov ax,@data ; set up data segment register
  mov ds,ax
  mov es,ax    ; set up es also for now
  mov ah,06h  ; scroll window up (erase screen)
  mov al,25d  ; 25 times
  mov bh,00h  ; filler attribute (no fill)
  mov ch,01h  ; upper row for start
  mov cl,01h  ; upper column
  mov dh,25d  ; bottom row to end
  mov dl,80d  ; bottom column (full screen!)
  int 10h    ; BIOS call

;
  mov dh,05h  ; row for cursor
  mov dl,01h  ; col for cursor
  call cursor ; place cursor now
  mov dx,OFFSET greetings ; print the greetings

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```



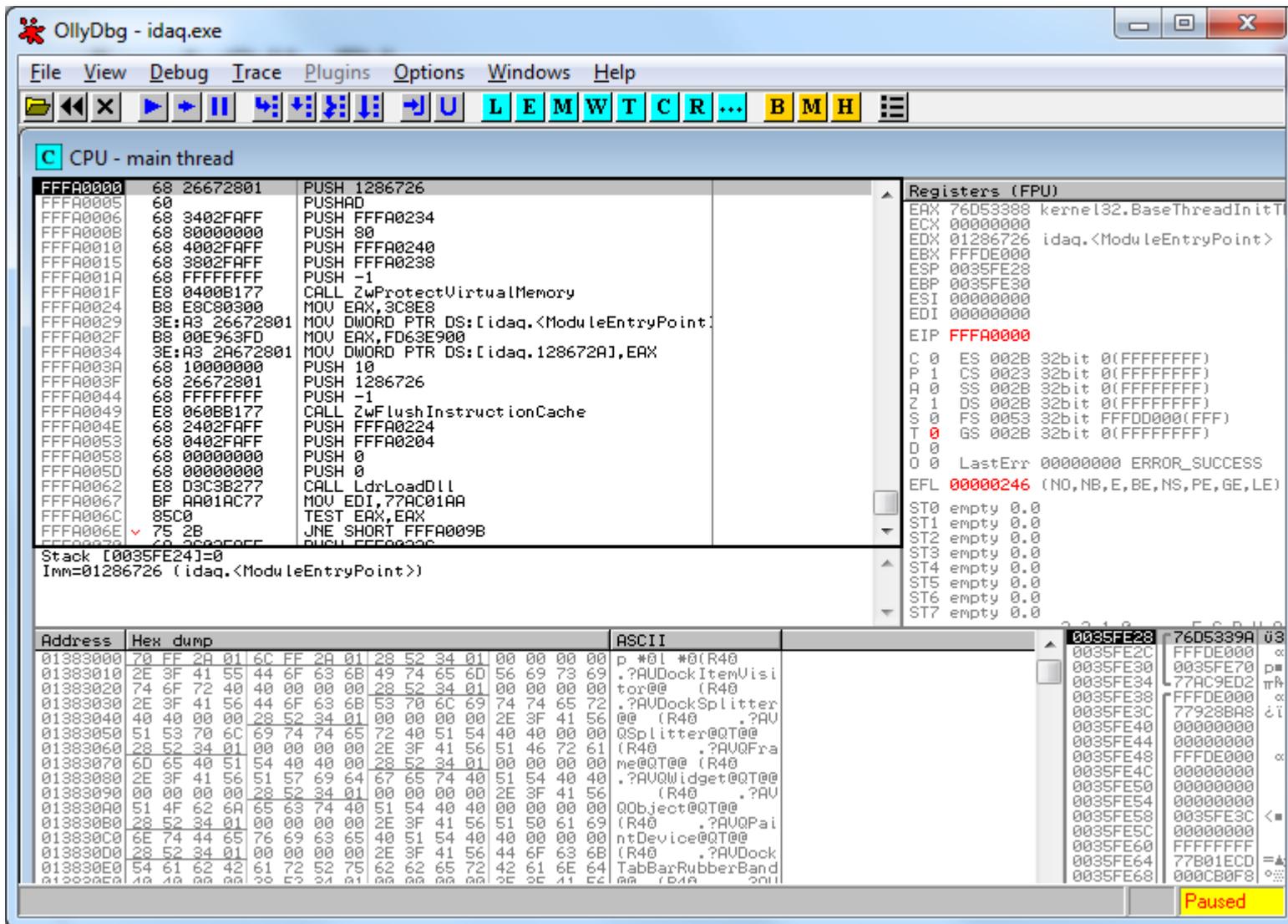
Next major step: Soft-ICE

```
EAX=03569DB6   EBX=806A1FCA   ECX=85E83890   EDX=000000E6   ESI=85E83878
EDI=8055C780   EBP=85E83800   ESP=EB427D40   EIP=806A2CB5   o d I s z a p c
CS=0008   DS=0023   SS=0010   ES=0023   FS=0030   GS=0000
```

Address	Disassembly	Comment
0008:806A2CB5	FFD3	CALL EBX
0008:806A2CB7	5B	POP EBX
0008:806A2CB8	59	POP ECX
0008:806A2CB9	894108	MOV [ECX+8], EAX
0008:806A2CBC	89510C	MOV [ECX+C], EDX
0008:806A2CBF	33C0	XOR EAX, EAX
0008:806A2CC1	C3	RET
0008:806A2CC2	8BFF	MOV EDI, EDI
0008:806A2CC4	51	PUSH ECX
0008:806A2CC5	53	PUSH EBX

```
—test—Ln(16)–Col(22)–Dim(545)–
int macro name test ( int value )
{
    return value + 1;
}
IRQL=(DISPATCH)–KTEB(85EAC088)–TID(0000)–hal!.text+00007975–CPU(#01)
:b1
00) * BPX 0x0004C813 /M NTDLL.DLL /P GameChannel (address is module start based)
:
:
Enter a command for BugChecker. Idle
```

And finally: OllyDbg



The screenshot shows the OllyDbg interface with the following components:

- Menu Bar:** File, View, Debug, Trace, Plugins, Options, Windows, Help
- Toolbar:** Navigation and execution controls (back, forward, pause, etc.) and a keyboard shortcuts bar (U, L, E, M, W, T, C, R, B, M, H).
- Disassembly Window:** Shows assembly code for the CPU - main thread. The instruction at address FFFA006E is highlighted: `JNE SHORT FFFA009B`. The instruction pointer (EIP) is FFFA0000.
- Registers (FPU) Window:** Lists registers such as EAX (76D53388), ECX (00000000), EDX (01286726), EBX (FFFDE000), ESP (0035FE28), EBP (0035FE30), ESI (00000000), EDI (00000000), and EIP (FFFA0000). It also shows segment registers (CS, SS, DS, FS, GS) and status registers (ST0-ST7).
- Stack Window:** Shows the current stack frame with `Imm=01286726 (idaq.<ModuleEntryPoint>)`.
- Hex Dump Window:** Displays memory data at address 01383000, including hex values and ASCII characters like `p *01 *0(R40`.
- Registers (GPR) Window:** Shows general purpose registers (EAX, ECX, EDI, etc.) with their current values.
- Status Bar:** Indicates the program is **Paused**.



Disassemblers



Old ages: Sourcer

```
C:\WINDOWS\system32\cmd.exe

SOURCER (R)

U COMMUNICATIONS, INC.
Copyright (c)1988-1997
07.00 s/n SW601745

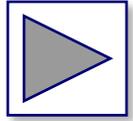
Output files: 4help.lst, 4help.sdf

Input filename 4help.exe      Label Counts      Drive f: used for output
Beginning addr 88E6:0000      Data 2774         > 100 Megs available
Ending address 9CFC:2FFF      Sub 347           37K bytes free (EMS off)
Math auto uP 8086/8088      Loc 3372         Code style .EXE      Passes 5

SOURCER COMPLETED  Total number of lines = 36843  Lines per min = 31832

F:\Sourcer7>_
```


Welcome to Windows: W32DASM



- Started as an Interactive Dis-Assembler, enabling user interaction with the disassembler's decisions.
- Slowly evolved into an automatic RE tool:
 - Built-in full-control script language
 - Library recognition (including user-generated)
 - Function prototype information
 - Display
 - Propagate throughout the code
 - Support for plug-ins
 - Support for Python scripting
 - Multi-architecture, cross-platform support
 - Full incorporation with built-in and external debuggers

Hex-Editor

010 Editor - C:\Users\inbarr\Documents\TAU\Calc\calc.exe

File Edit Search View Scripts Templates Tools Window Help

Edit As: Hex

Workspace

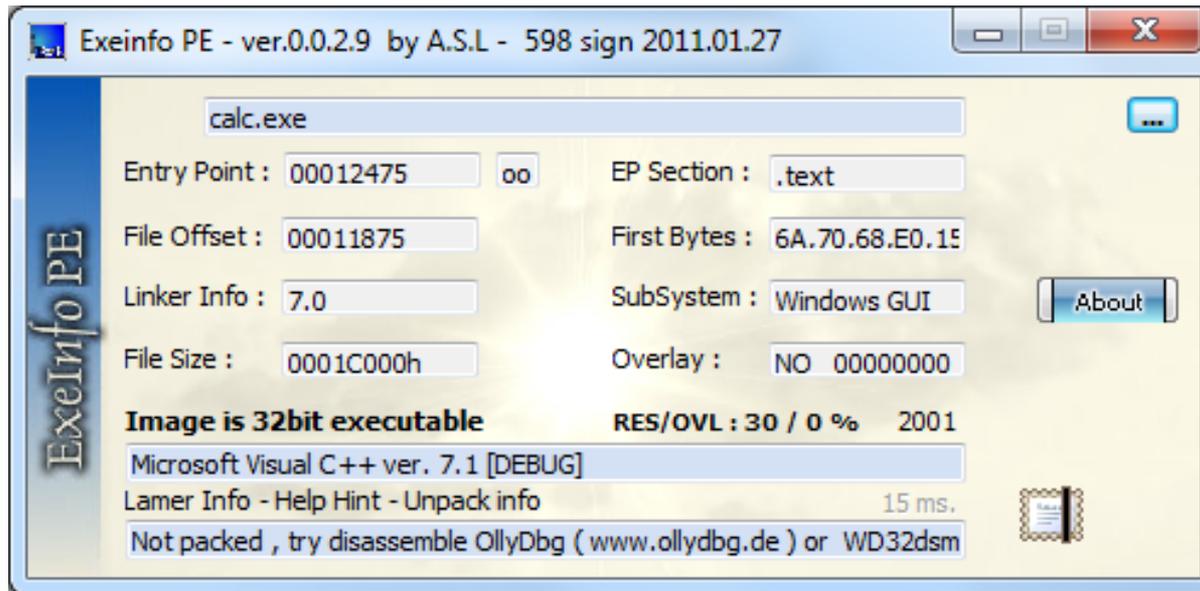
- Open Files
 - C:\Users\inbarr\... \TAU\Calc\calc.exe
- Favorite Files
- Recent Files
 - C:\Users\inbarr\... \EasyCrack.exe
- Bookmarked Files

Inspector

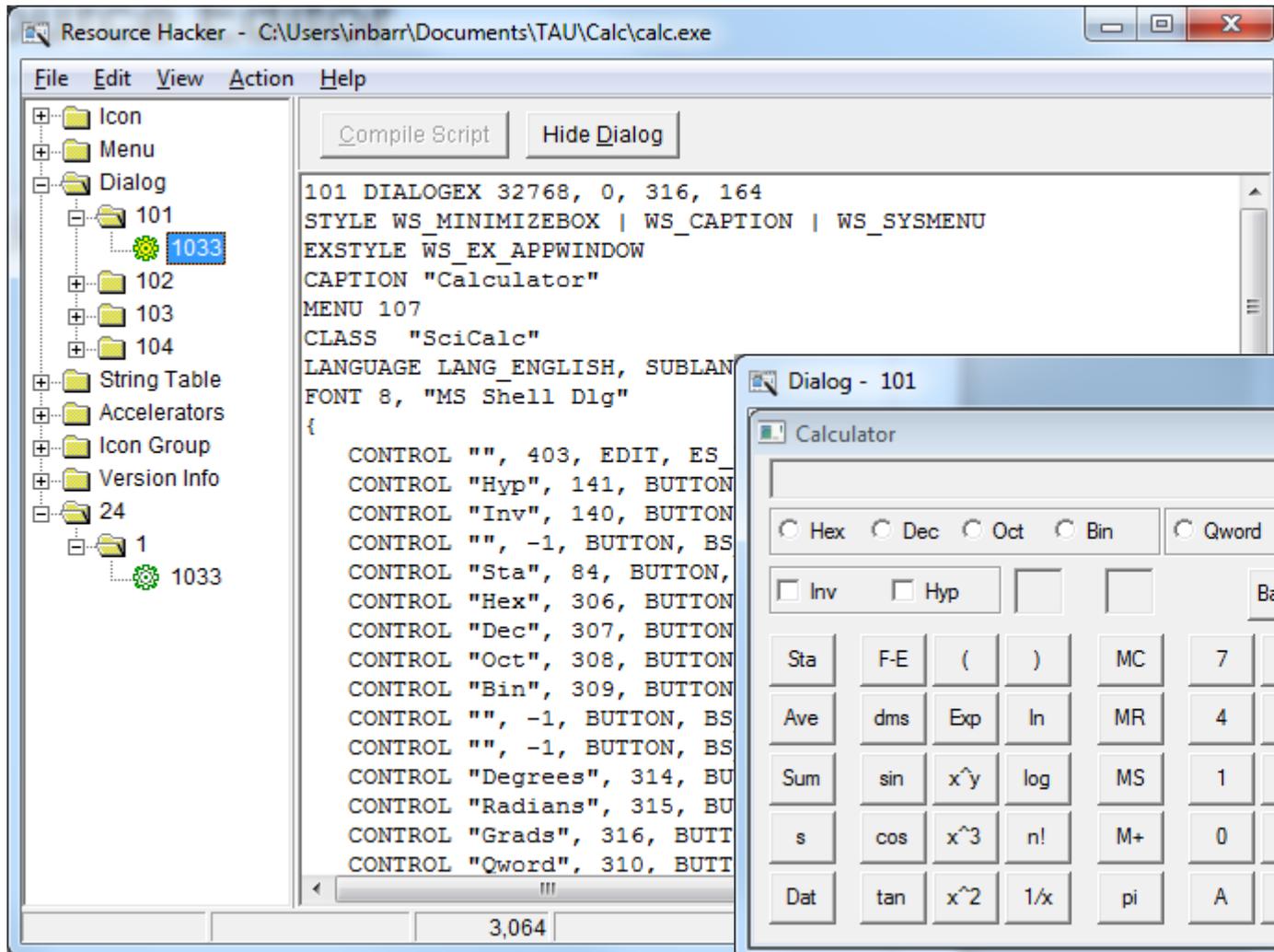
Type	Value
Signed Byte	77
Unsigned Byte	77
Signed Short	23117
Unsigned Short	23117
Signed Int	9460301
Unsigned Int	9460301
Signed Int64	12894362189
Unsigned Int64	12894362189
Float	1.325671e-38

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
0010h:	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030h:	00	00	00	00	00	00	00	00	00	00	00	00	F0	00	00	00ð...
0040h:	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..°..'Í!.,LÍ!Th
0050h:	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
0060h:	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
0070h:	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....
0080h:	87	45	16	64	C3	24	78	37	C3	24	78	37	C3	24	78	37	±E.dÃ\$×7Ã\$×7Ã\$×7
0090h:	39	07	38	37	C6	24	78	37	19	07	64	37	C8	24	78	37	9.87Æ\$×7..d7È\$×7
00A0h:	C3	24	78	37	C2	24	78	37	C3	24	79	37	44	24	78	37	Ã\$×7Ã\$×7Ã\$×7D\$×7
00B0h:	39	07	61	37	CE	24	78	37	54	07	3D	37	C2	24	78	37	9.a7Î\$×7T.=7Ã\$×7
00C0h:	19	07	65	37	DF	24	78	37	39	07	45	37	C2	24	78	37	..e7ß\$×79.E7Ã\$×7
00D0h:	52	69	63	68	C3	24	78	37	00	00	00	00	00	00	00	00	RichÃ\$×7.....
00E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0h:	50	45	00	00	4C	01	03	00	10	84	7D	3B	00	00	00	00	PE..L.....};...
0100h:	00	00	00	00	E0	00	0F	01	0B	01	07	00	00	28	01	00à.....(..
0110h:	00	9C	00	00	00	00	00	00	75	24	01	00	00	10	00	00	.œ.....u\$.....
0120h:	00	40	01	00	00	00	00	01	00	10	00	00	00	02	00	00	.@.....
0130h:	05	00	01	00	05	00	01	00	04	00	00	00	00	00	00	00
0140h:	00	F0	01	00	00	04	00	00	FC	D7	01	00	02	00	00	80	.ð.....ü×.....€
0150h:	00	00	04	00	00	10	00	00	00	00	10	00	00	10	00	00
0160h:	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00

Opened file 'C:\Users\inbarr\Documents\TAU\Calc\calc.exe'. Pos: 0 [0h] Val: 77 4Dh 01001101b Size: 114688 ANSI LIT W OVR



Resource Editor



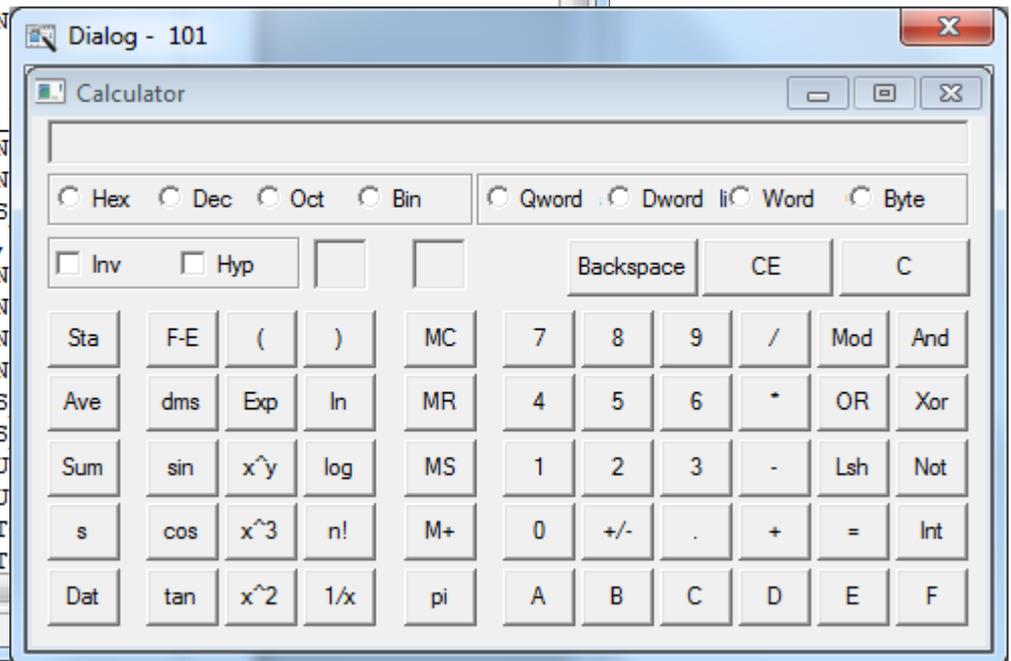
Resource Hacker - C:\Users\inbarr\Documents\TAU\Calc\calc.exe

File Edit View Action Help

Icon
Menu
Dialog
 101
 102
 103
 104
String Table
Accelerators
Icon Group
Version Info
24
 1
 1033

Compile Script Hide Dialog

```
101 DIALOGEX 32768, 0, 316, 164
STYLE WS_MINIMIZEBOX | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "Calculator"
MENU 107
CLASS "SciCalc"
LANGUAGE LANG_ENGLISH, SUBLANG_DEFAULT
FONT 8, "MS Shell Dlg"
{
CONTROL "", 403, EDIT, ES_
CONTROL "Hyp", 141, BUTTON, BS_
CONTROL "Inv", 140, BUTTON, BS_
CONTROL "", -1, BUTTON, BS_
CONTROL "Sta", 84, BUTTON, BS_
CONTROL "Hex", 306, BUTTON, BS_
CONTROL "Dec", 307, BUTTON, BS_
CONTROL "Oct", 308, BUTTON, BS_
CONTROL "Bin", 309, BUTTON, BS_
CONTROL "", -1, BUTTON, BS_
CONTROL "", -1, BUTTON, BS_
CONTROL "Degrees", 314, BU
CONTROL "Radians", 315, BU
CONTROL "Grads", 316, BUTT
CONTROL "Qword", 310, BUTT
```



Dialog - 101

Calculator

Hex Dec Oct Bin Qword Dword li Word Byte

Inv Hyp Backspace CE C

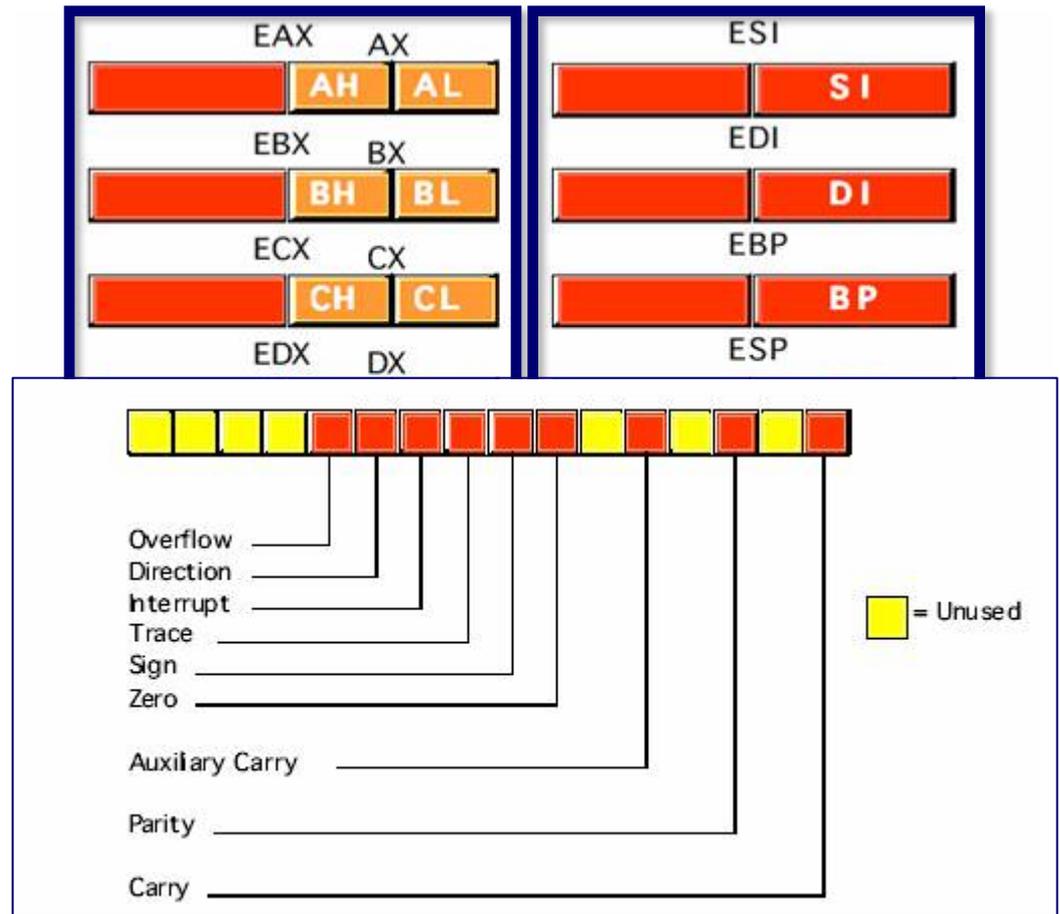
Sta	F-E	()	MC	7	8	9	/	Mod	And
Ave	dms	Exp	ln	MR	4	5	6	*	OR	Xor
Sum	sin	x^y	log	MS	1	2	3	-	Lsh	Not
s	cos	x^3	n!	M+	0	+/-	.	+	=	Int
Dat	tan	x^2	1/x	pi	A	B	C	D	E	F

Let's play
with them tools...



60 seconds on x86 registers

- General purpose registers:
32bit/16bit/8bit
- Index registers:
32bit/16bit
- Segment registers:
16bit
- Flags:
32bit/16bit

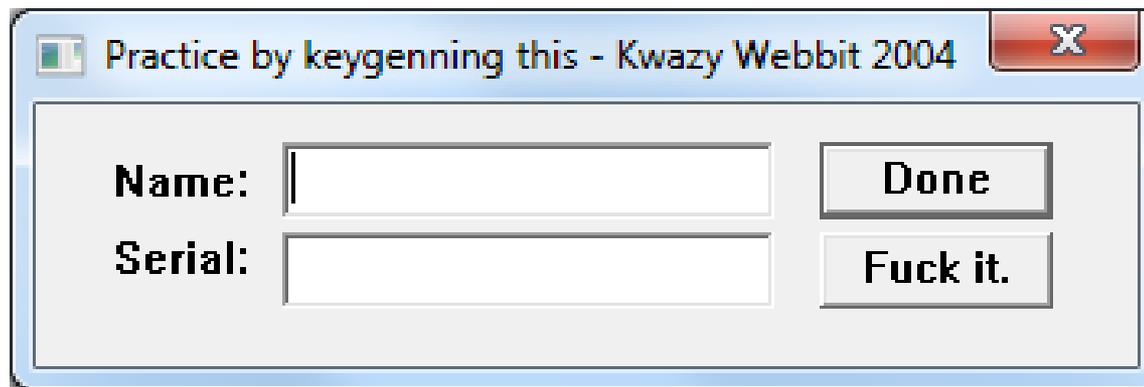


Exercise 1: Static Reversing



Exercise 1: Static Reversing

- Target: a 2004 “Crack-Me”



- Tools: IDA-Pro

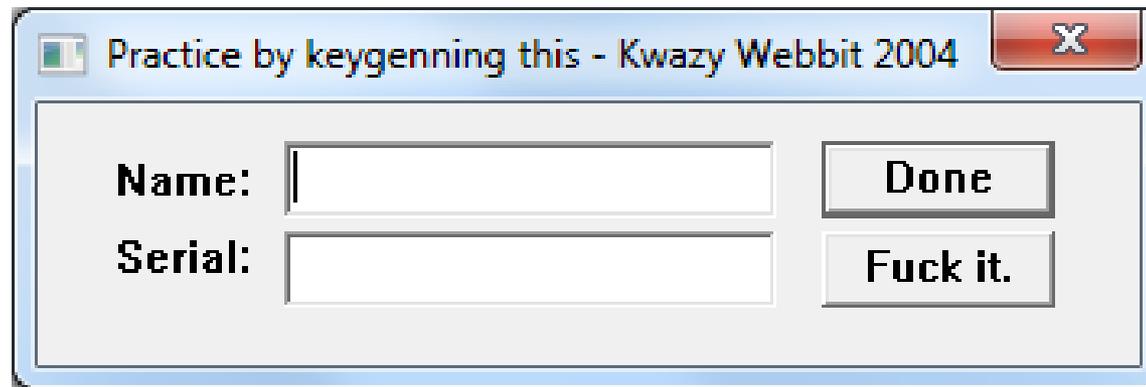


Exercise 2: Dynamic Reversing



Exercise 2: Dynamic Reversing

- Target: a 2004 “Crack-Me”



- Tools: OllyDbg, IDA-Pro



Exercise 3:

Simple Anti-Debugging



Exercise 3: Simple Anti Debugging

- Target: a 2006 “Crack-Me”



- Tools: OllyDbg



- Malware is comprised of the following building blocks:
 - Infection Vector
 - Concealment
 - Operation
 - Communications
- Check Point's Anti-Malware Software Blade sits at the gateway
- Therefore, communications interest us the most



- A CrimeWare ToolKit, originating in Russia.
- Used mostly for stealing financial information, but will settle for any other identity information and key logging...
- Like any serious trojan, Spy Eye compresses its traffic and encrypts it
 - Compression is performed using a public library (LZO)
 - Encryption algorithm is proprietary



Act 1: Encryption



Act 2:

Configuration Download



Act 3: Another Encryption



So what do you need
in order to be
a good reverser?



What makes a good reverser?

Qualities

- Patient
 - Curious
 - Persistent
 - Outside-the-Box Thinking
-
- Optional: Good lookin'

Knowledge

- Assembly Language
- Some High-Level programming
 - Best: origin of binary
- Operating System Internals
 - API
 - Data Structures
 - File Structures
- Good scripting skills
- Anti-Debugging Tricks

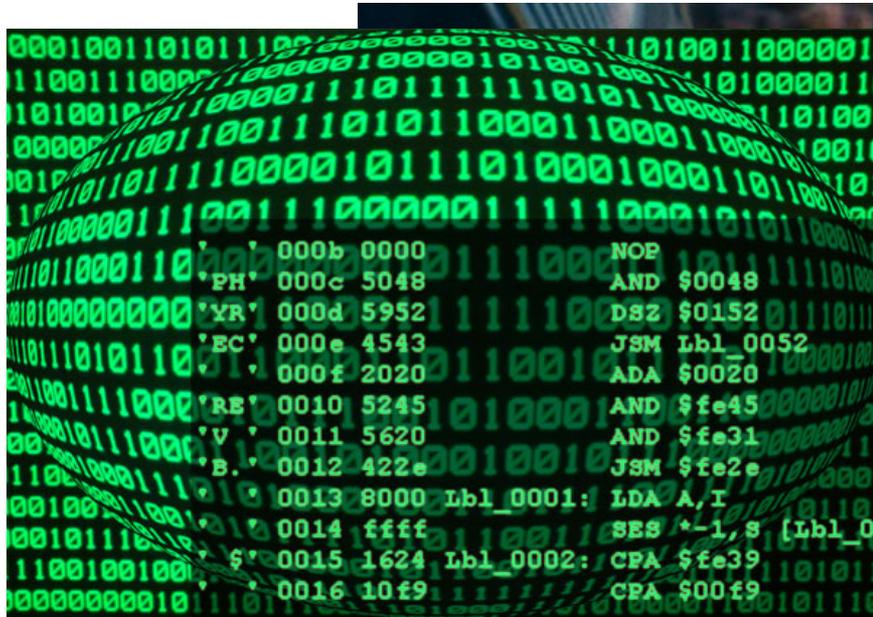


Outside-the-Box Thinking

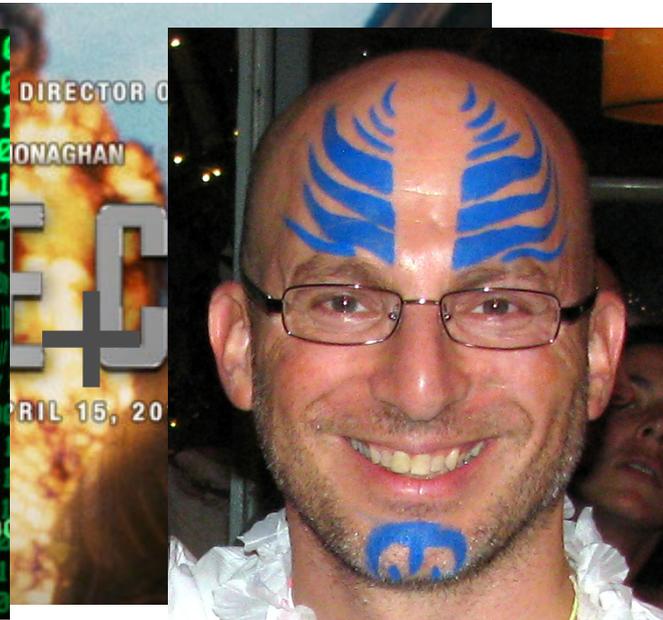


And remember, kids:

Binary



Reverse Engineer



=?

Which means...



Questions?



Thank you!

inbarr@checkpoint.com



All images and videos have their origin URL in the “Alt Text” property.
All rights belong to their respective owner.