

逆向分析技术

课程回顾

OllyDbg工具的使用 - 主函数运行过程

main函数处F2下断点，按F9运行

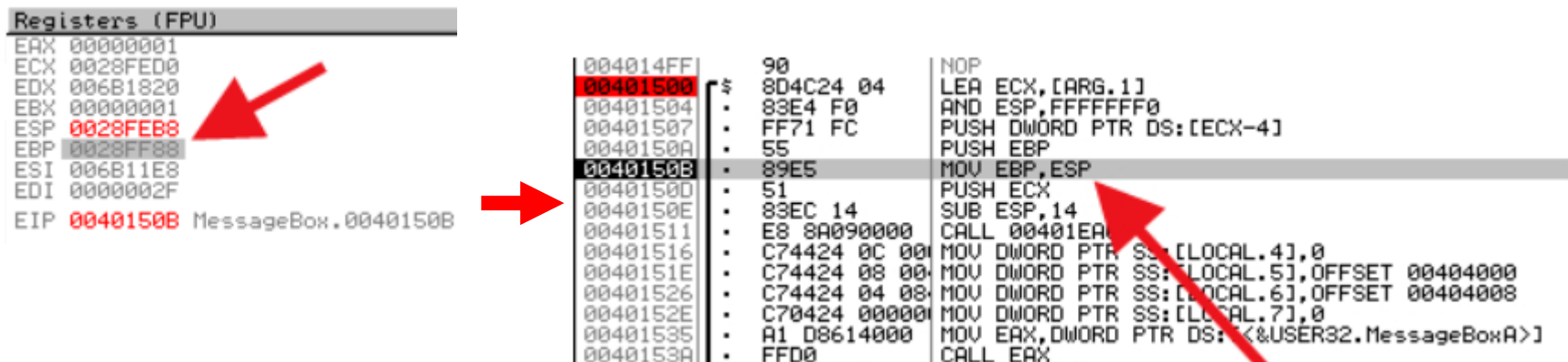
004014FD	90	NOP	
004014FE	90	NOP	
004014FF	90	NOP	
00401500	\$ 8D4C24 04	LEA ECX,[ARG.1]	
00401504	83E4 F0	AND ESP,FFFFFFF0	DQWORD (16.-byte) stack alignment
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]	
0040150A	55	PUSH EBP	
0040150D	89E5	MOV EBP,ESP	
00401510	51	PUSH ECX	
00401513	83EC 14	SUB ESP,14	
00401516	E8 8A090000	CALL 00401EA0	
00401519	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0	Type => MB_OK!MB_DEFBUTTON1!MB_APPLMODAL
0040151C	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000	Caption => "warning"
0040151F	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008	Text => "You have been hacked"
00401522	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0	hOwner => NULL
00401525	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]	USER32.MessageBoxA
00401528	FFD0	CALL EAX	
0040152B	83EC 10	SUB ESP,10	
0040152E	B8 00000000	MOV EAX,0	

F8运行到0x40150B

004014FF	90	NOP	
00401500	\$ 8D4C24 04	LEA ECX,[ARG.1]	
00401504	83E4 F0	AND ESP,FFFFFFF0	
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]	
0040150A	55	PUSH EBP	
0040150D	89E5	MOV EBP,ESP	
00401510	51	PUSH ECX	
00401513	83EC 14	SUB ESP,14	
00401516	E8 8A090000	CALL 00401EA0	
00401519	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0	
0040151C	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000	
0040151F	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008	
00401522	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0	
00401525	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]	
00401528	FFD0	CALL EAX	

OllyDbg工具的使用 - 栈

目前栈的情况，esp栈顶是0x28FEB8，ebp栈底是0x28FF88。



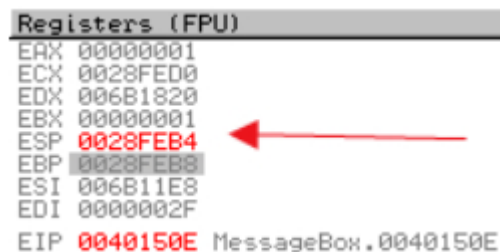
Registers (FPU)

EAX	00000001
ECX	0028FED0
EDX	006B1820
EBX	00000001
ESP	0028FEB8
EBP	0028FF88
ESI	006B11E8
EDI	0000002F
EIP	0040150B MessageBox.0040150B

Assembly Code:

004014FF	90	NOP
00401500	8D4C24 04	LEA ECX,[ARG.1]
00401504	83E4 F0	AND ESP,FFFFFFF0
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]
0040150A	55	PUSH EBP
0040150B	89E5	MOV EBP,ESP
0040150D	51	PUSH ECX
0040150E	83EC 14	SUB ESP,14
00401511	E8 8A090000	CALL 00401EA
00401516	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0
0040151E	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000
00401526	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008
0040152E	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0
00401535	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]
0040153A	FFD0	CALL EAX

继续F8两次，现在的esp是0x28FEB4，这里指令SUB ESP,14是让esp直接减14的意思



Registers (FPU)

EAX	00000001
ECX	0028FED0
EDX	006B1820
EBX	00000001
ESP	0028FEB4
EBP	0028FF88
ESI	006B11E8
EDI	0000002F
EIP	0040150E MessageBox.0040150E

OllyDbg工具的使用 - 栈

目前这5个地址的情况

0028FEA4	00405000	P@	
0028FEA8	0028FEC8	▬@	
0028FEAC	00401E7B	▬@	RETURN from MessageBox.00401720 to MessageBox.00401E7B
0028FEB0	00401E20	▬@	Entry point
0028FEB4	0028FED0	▬@	

继续F8，运行过CALL 00401EA0时发现这五个地址的值没有变化。运行过MOV DWORD PTR SS:[LOCAL.4],0时现0x28FEAC的值变成了0。

0028FEA4	00405000	P@	
0028FEA8	0028FEC8	▬@	
0028FEAC	00000000	▬@	
0028FEB0	00401E20	▬@	Entry point
0028FEB4	0028FED0	▬@	

windbg工具的使用 - 内核调试环境 - windbg+vmware

增加串口



输出到命名管道



管道设置

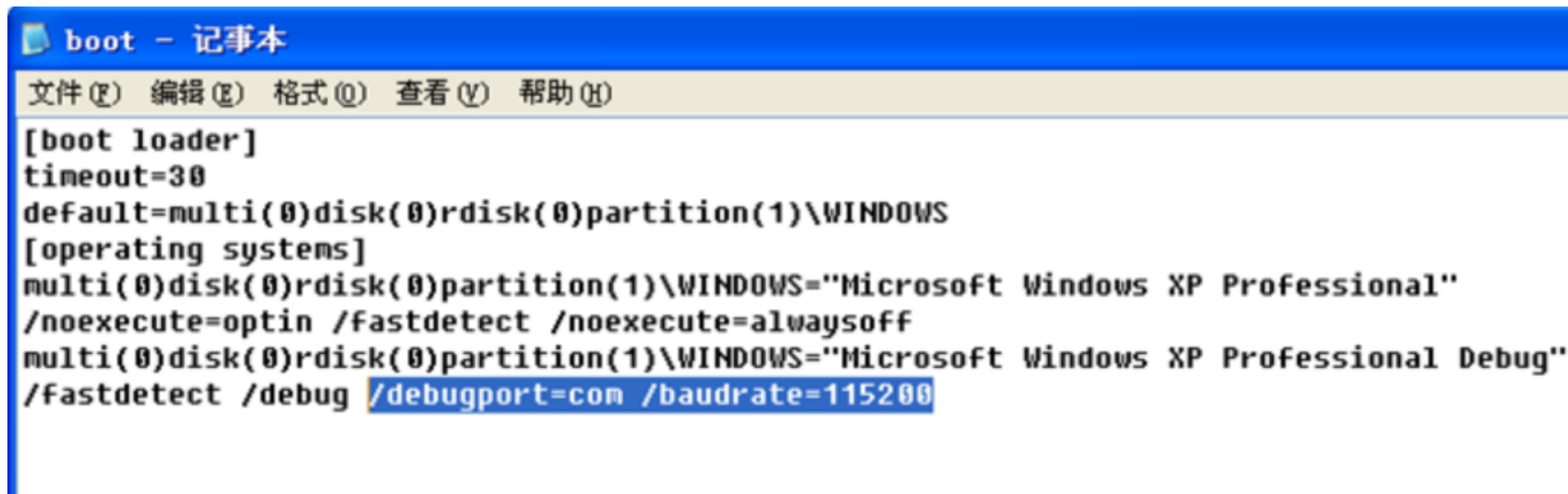


windbg工具的使用 - 内核调试 - (xp)修改Boot.ini

找到Boot.ini



进行调试修改



```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/noexecute=optin /fastdetect /noexecute=alwaysoff
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional Debug"
/fastdetect /debug /debugport=com /baudrate=115200
```

逆向分析技术

逆向工具使用 实例分析

本次课程支撑的毕业要求指标点

- 毕业要求3-3:

充分理解信息安全领域软硬件系统的基础上，能够设计或开发满足特定需求和约束条件的信息安全系统、模块或算法流程，并能够进行系统级优化。

IDA实战示例

D:\下载文件\Firefox\1.exe

请输入flag: 123456
flag错误。再试试?

; Attributes: bp-based frame

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public _main
_main proc near
```

```
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

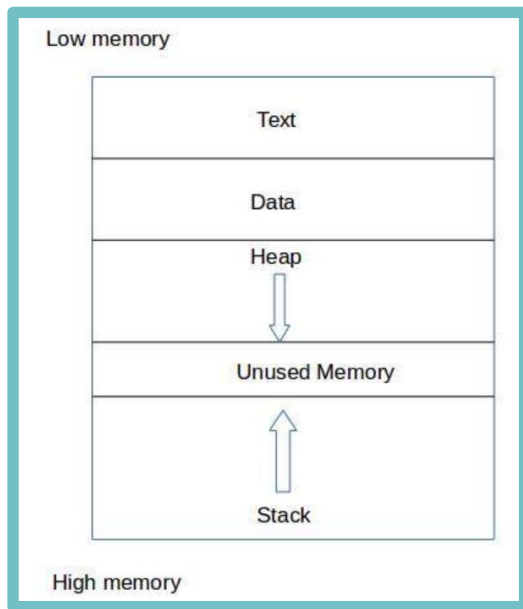
```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFF0h
sub     esp, 90h
call    __main
mov     dword ptr [esp], offset fmt ; "请输入flag: "
call    __Z6printfPKcz ; printf(char const*,...)
mov     dword ptr [esp+75h], 67616C66h
mov     dword ptr [esp+79h], 6C65577Bh
mov     dword ptr [esp+7Dh], 656D6F63h
mov     dword ptr [esp+81h], 5F6F545Fh
mov     dword ptr [esp+85h], 575F4552h
mov     dword ptr [esp+89h], 646C726Fh
mov     word ptr [esp+8Dh], 7D21h
mov     byte ptr [esp+8Fh], 0
jmp     short loc_401592
```

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _BYTE v4[3]; // [sp+11h] [bp-7Fh]@2
4     signed int v5; // [sp+75h] [bp-18h]@1
5     signed int v6; // [sp+79h] [bp-17h]@1
6     signed int v7; // [sp+7Dh] [bp-13h]@1
7     signed int v8; // [sp+81h] [bp-Fh]@1
8     signed int v9; // [sp+85h] [bp-Bh]@1
9     signed int v10; // [sp+89h] [bp-7h]@1
10    signed __int16 v11; // [sp+8Dh] [bp-3h]@1
11    char v12; // [sp+8Fh] [bp-1h]@1
12
13    __main();
14    printf("请输入flag: ");
15    v5 = 1734437990;
16    v6 = 1818580859;
17    v7 = 1701670755;
18    v8 = 1601131615;
19    v9 = 1465861458;
20    v10 = 1684828783;
21    v11 = 32033;
22    v12 = 0;
23    while ( scanf("%s", v4) != -1 && strcmp(v4, (const char *)&v5) )
24        printf("flag错误。再试试? \n");
```

loc_401592:
lea eax, [esp+11h]
mov [esp+4], eax

0401500: main (Synchronized with Hex View-1)

进程地址空间



- **Text Segment**: 代码段—指令
- **Data Segment**: 数据段—全局变量
- **Heap Segment**: 堆—`malloc/free`, 存储动态分配数据
- **Stack Segment**: 栈—函数参数; 局部变量; 返回地址

栈的基本定义

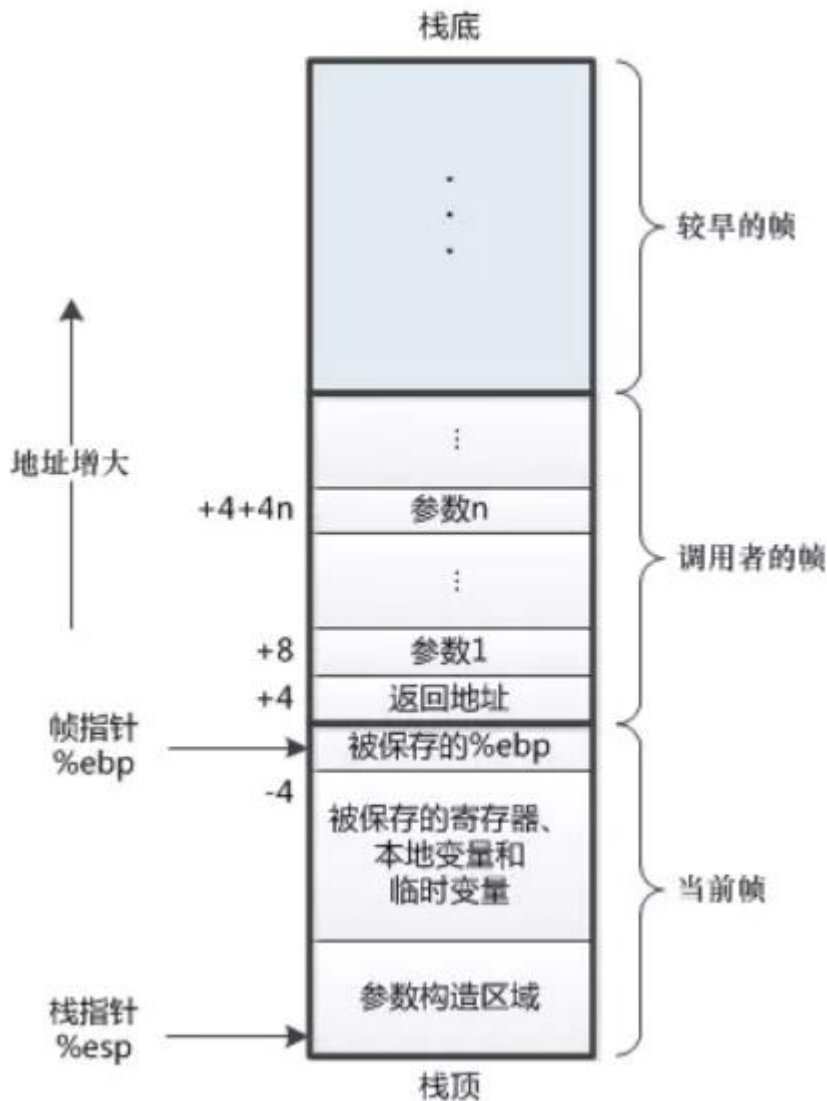
- 栈的操作方式: `push/pop`
- 栈的生长方向: 高地址 → 低地址

栈帧

● 栈帧是指系统为一个函数调用单独分配的那部分栈空间。当运行中的程序调用另一个函数时，就要进入一个新的栈帧，原来函数的栈帧称为调用者的帧，新的栈帧称为当前帧。被调用的函数运行结束后当前帧全部收缩，回到调用者的帧。

ebp指向当前帧底部

esp指向当前帧顶部



IDA实战示例

D:\下载文件\Firefox\1.exe

请输入flag: 123456
flag错误。再试试?

; Attributes: bp-based frame

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public _main
_main proc near
```

```
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFF0h
sub     esp, 90h
call    __main
mov     dword ptr [esp], offset fmt ; "请输入flag: "
call    __Z6printfPKcz ; printf(char const*,...)
mov     dword ptr [esp+75h], 67616C66h
mov     dword ptr [esp+79h], 6C65577Bh
mov     dword ptr [esp+7Dh], 656D6F63h
mov     dword ptr [esp+81h], 5F6F545Fh
mov     dword ptr [esp+85h], 575F4552h
mov     dword ptr [esp+89h], 646C726Fh
mov     word ptr [esp+8Dh], 7D21h
mov     byte ptr [esp+8Fh], 0
jmp     short loc_401592
```

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _BYTE v4[3]; // [sp+11h] [bp-7Fh]@2
4     signed int v5; // [sp+75h] [bp-18h]@1
5     signed int v6; // [sp+79h] [bp-17h]@1
6     signed int v7; // [sp+7Dh] [bp-13h]@1
7     signed int v8; // [sp+81h] [bp-Fh]@1
8     signed int v9; // [sp+85h] [bp-Bh]@1
9     signed int v10; // [sp+89h] [bp-7h]@1
10    signed __int16 v11; // [sp+8Dh] [bp-3h]@1
11    char v12; // [sp+8Fh] [bp-1h]@1
12
13    __main();
14    printf("请输入flag: ");
15    v5 = 1734437990;
16    v6 = 1818580859;
17    v7 = 1701670755;
18    v8 = 1601131615;
19    v9 = 1465861458;
20    v10 = 1684828783;
21    v11 = 32033;
22    v12 = 0;
23    while ( scanf("%s", v4) != -1 && strcmp(v4, (const char *)&v5) )
24        printf("flag错误。再试试? \n");
```

loc_401592:
lea eax, [esp+11h]
mov [esp+4], eax

0401500: main (Synchronized with Hex View-1)

IDA实战示例

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4; // [esp+11h] [ebp-7Fh]
    int v5; // [esp+75h] [ebp-1Bh]    //由地址可以知道v5-v12是连着的，是一个数组
    int v6; // [esp+79h] [ebp-17h]
    int v7; // [esp+7Dh] [ebp-13h]
    int v8; // [esp+81h] [ebp-Fh]
    int v9; // [esp+85h] [ebp-Bh]
    int v10; // [esp+89h] [ebp-7h]
    __int16 v11; // [esp+8Dh] [ebp-3h]
    char v12; // [esp+8Fh] [ebp-1h]
```

IDA实战示例

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _BYTE v4[3]; // [sp+11h] [bp-7Fh]@2
4     signed int v5; // [sp+75h] [bp-18h]@1
5     signed int v6; // [sp+79h] [bp-17h]@1
6     signed int v7; // [sp+7Dh] [bp-13h]@1
7     signed int v8; // [sp+81h] [bp-Fh]@1
8     signed int v9; // [sp+85h] [bp-8h]@1
9     signed int v10; // [sp+89h] [bp-7h]@1
10    signed __int16 v11; // [sp+8Dh] [bp-3h]@1
11    char v12; // [sp+8Fh] [bp-1h]@1
12
13    __main();
14    printf("请输入flag: ");
15    v5 = 1734437990;
16    v6 = 1818580859;
17    v7 = 1701670755;
18    v8 = 1601131615;
19    v9 = 1465861458;
20    v10 = 1684828783;
21    v11 = 32033;
22    v12 = 0;
23    while ( scanf("%s", v4) != -1 && strcmp(v4, (const char *)&v5) )
24        printf("flag错误。再试试? \n");
```

IDA实战示例

```
__main();  
printf(&fmt);  
v5 = 'galf'; //这里选定需要转换的字符串，右键，选择R转换成字符串
```

v6 = 'leW{';	v5 = 1734437990;	int v5; [esp+75h]
v7 = 'emoc';	v6 = 1818580859;	'g': esp+75h
v8 = '_oT_';	v7 = 1701670755;	'a': esp+74h
v9 = 'W_ER';	v8 = 1601131615;	'l': esp+73h
v10 = 'dlro';	v9 = 1465861458;	'f': esp+72h
v11 = '}!';	v10 = 1684828783;	
v12 = 0;	v11 = 32033;	
	v12 = 0;	
flag>Welcome_To_RE_World!}		

```
while ( scanf("%s", &v4) != -1 && strcmp(&v4, (const char *)&v5) ) //这里可以知道v4是我们  
输入的字符串，v5是那个长的字符串
```

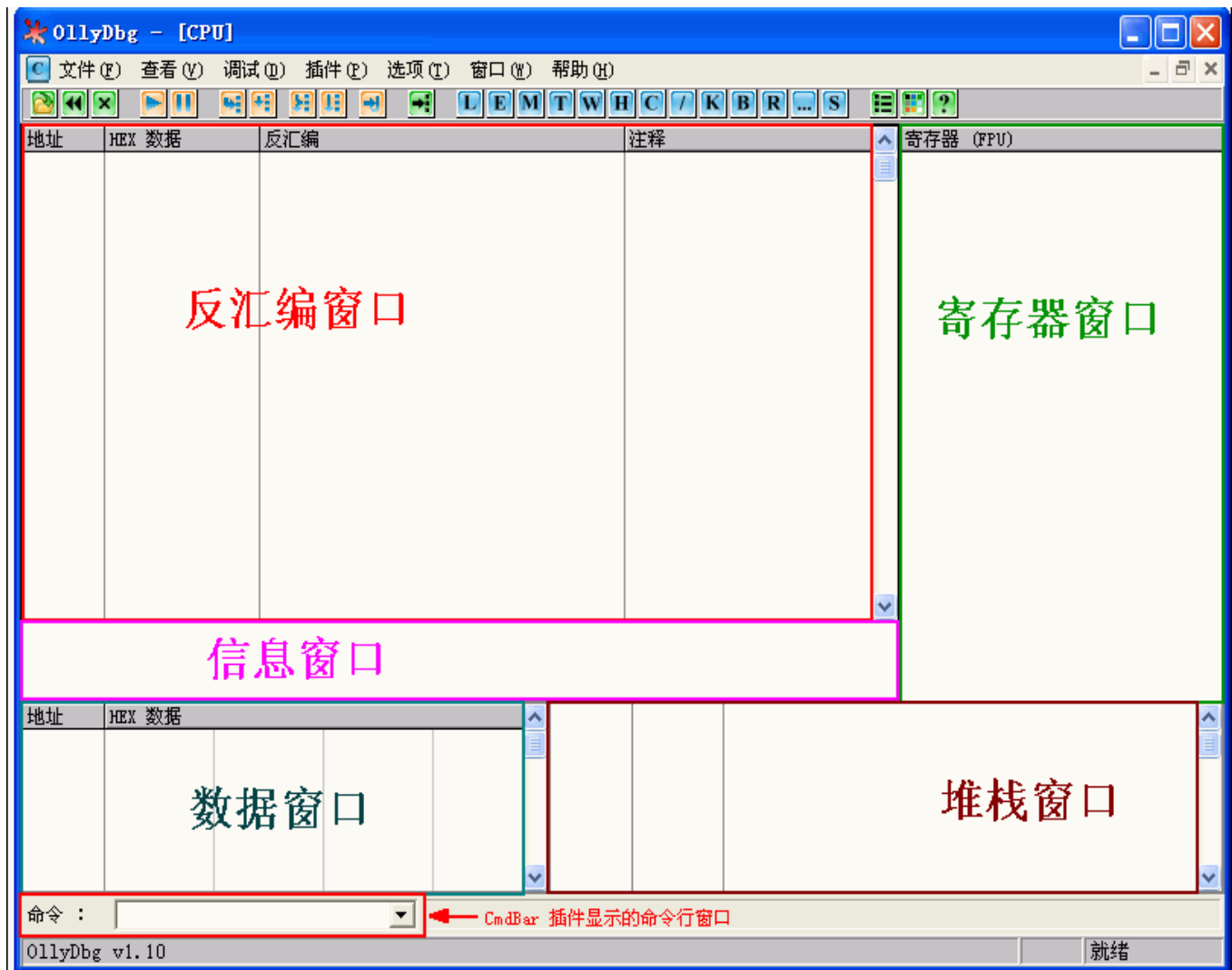
```
printf(aFlag);  
printf(aFlag_0);
```


IDA实战示例

破解成功!

```
flag{Welcome_To_RE_World!}  
flag正确。
```

OlllyDbg工具实战示例 - OlllyDbg基本界面



OllyDbg工具实战示例 - 基本用法

F2: 设置断点，在光标位置按F2键即可，再按则会删除断点

F8: 单步步过。每按一次执行一条指令，遇到 **CALL** 等子程序不进入

F7: 单步步入。功能同(F8)，遇到 **CALL** 等子程序时会进入其中

F4: 运行到选定位置。作用就是直接运行到光标所在位置处暂停

F9: 运行。被调试的程序将直接开始运行

CTRL+F9: 执行到返回。此命令在执行到一个 **ret** (返回指令)指令时暂停，常用于从系统地址空间返回到我们调试的程序地址空间

ALT+F9: 执行到用户代码。可用于从系统地址空间快速返回到我们调试的程序地址空间

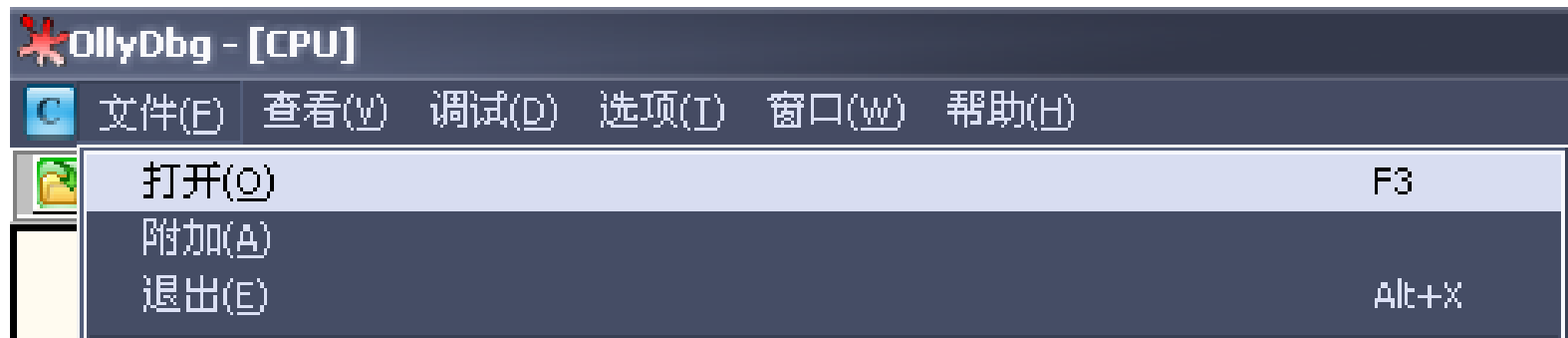
ALT+M: 打开内存窗口，查看内存信息

011yDbg工具实战示例 - 汇编基本指令

MOV 传送字或字节 如MOV A B,就是将B中的字传给A
PUSH 把字压入堆栈 如PUSH A,就是把A中的字压入栈
CALL 子程序调用指令 如CALL SUM,调用子程序SUM
RET 子程序返回指令
XOR 异或运算
AND 与运算
CMP 比较.(两操作数作减法,仅修改标志位,不保存运算结果)
JNZ/JNE 运算结果不为零转移,标志位ZF=0时跳转
JZ/JE 运算结果为零转移,标志位ZF=1时跳转
DEC 减 1 **INC** 加 1
ADD 加法 **SUB** 减法
LEA 装入有效地址 例: LEA DX,string;把偏移地址存到DX.
MOVSX 先符号扩展,再传送
REP 按计数寄存器CX/ECX中指定的次数重复执行
TEST 测试.(两操作数作与运算,仅修改标志位,不保存运算结果)

OllyDbg工具实战示例 - 打开文件

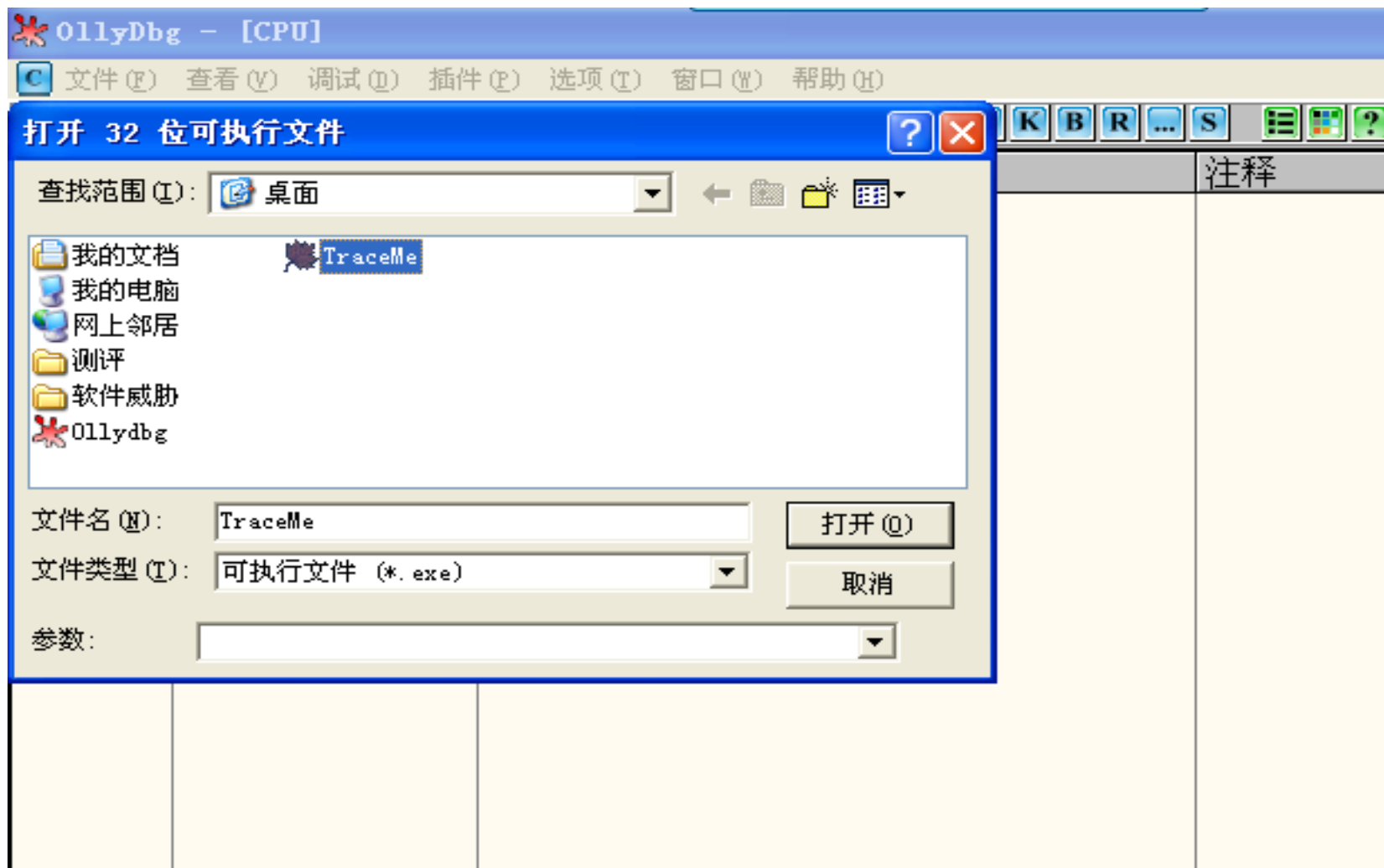
- 点击菜单 文件->打开 （快捷键是 F3） 来打开一个可执行文件进行调试
- 点击菜单 文件->附加 来附加一个已运行的进程上进行调试。注意这里要附加的程序必须已运行。



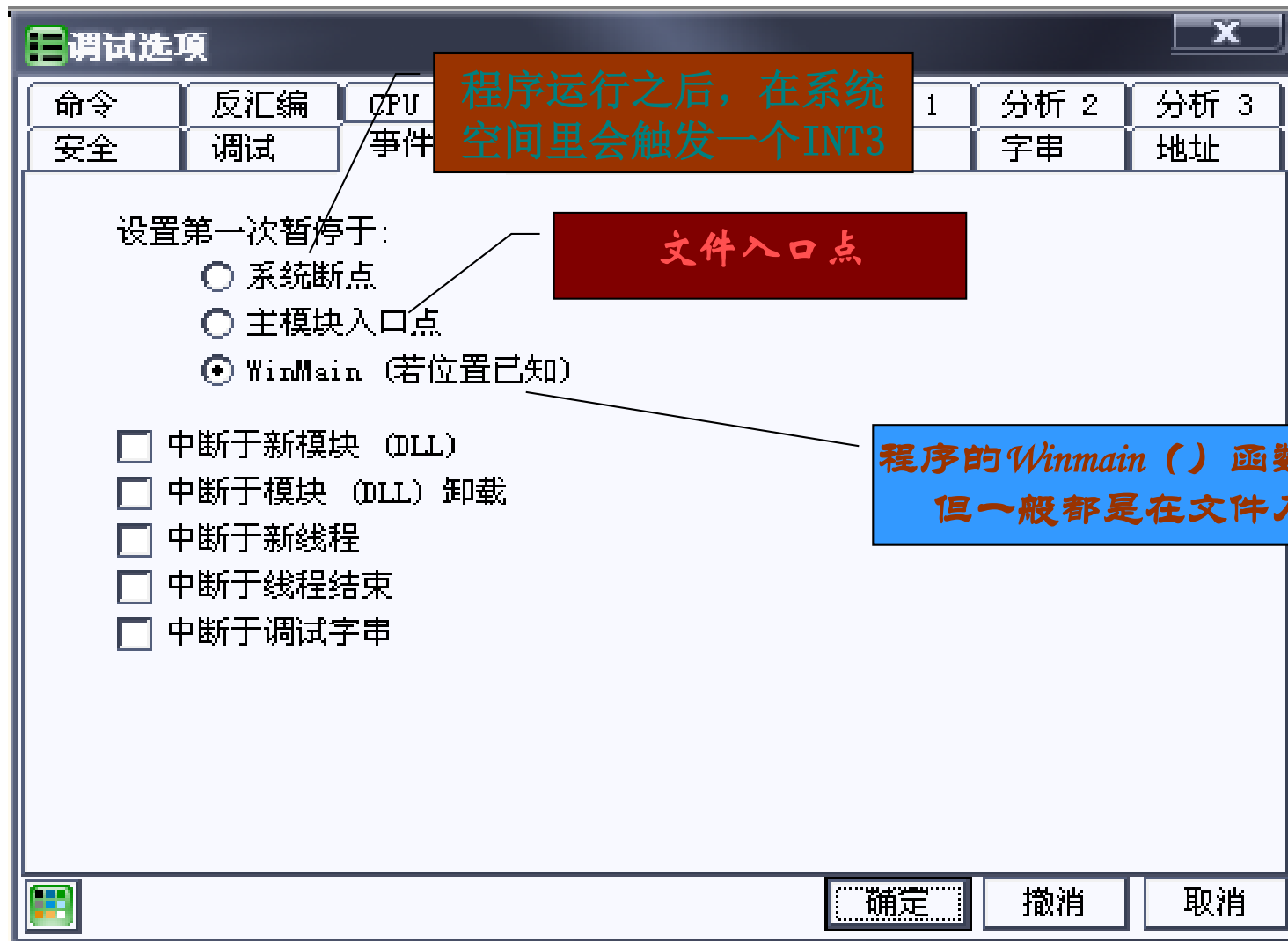
011yDbg工具实战示例



OlllyDbg工具实战示例 - 打开文件



OllyDbg工具实战示例 - 入口点



OllyDbg工具实战示例 - 载入文件

004013A0	\$ 55	PUSH EBP
004013A1	. 8BEC	MOV EBP,ESP
004013A3	. 6A FF	PUSH -1
004013A5	. 68 D0404000	PUSH TraceMe.004040D0
004013AA	. 68 D41E4000	PUSH TraceMe.00401ED4
004013AF	. 64:A1 00000000	MOV EAX,DWORD PTR FS:[0]

虚拟地址

机器码：
CPU执行的机器
代码

汇编指令：
和机器码对应的
程序代码

OllyDbg工具实战示例 - 载入文件

OllyDbg - [CPU - 主线程, 模块 - TraceMe]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H)

文件入口点

地址	HEX 数据	反汇编
004013A0	55	PUSH EBP
004013A1	8BEC	MOV EBP,ESP
004013A3	6A FF	PUSH -1
004013A5	68 D0404000	PUSH TraceMe.004040D0
004013AA	68 D41E4000	PUSH TraceMe.00401ED4
004013AF	64:A1 000000	MOV EAX,DWORD PTR FS:[0]
004013B5	50	PUSH EAX
004013B6	64:8925 0000	MOV DWORD PTR FS:[0],ESP

SE 处理程序安装

然后我们要找到GetDlgItemTextA函数，因为程序从文本框中将内容读取出来，需要用到这个函数。我们就用Ctrl+G打开跟随表达式的窗口，在里面输入函数名就可以跟踪到函数名存在的地方

输入要跟随的表达式

GetDlgItemTextA

确定 取消

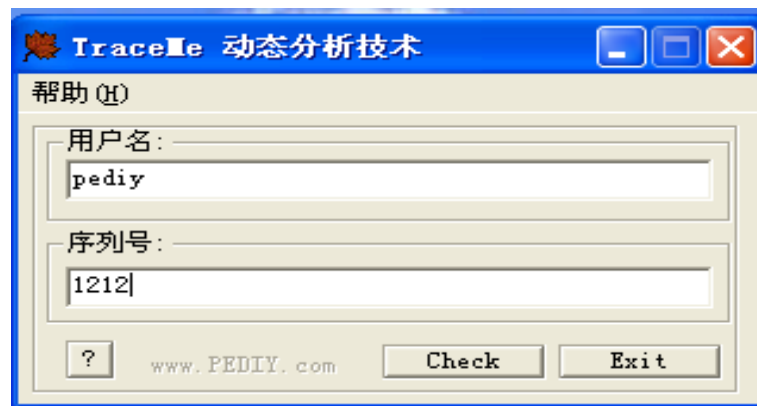
地址	HEX 数据	反汇编
77D6B05E	8BFF	MOV EDI,EDI
77D6B060	55	PUSH EBP
77D6B061	8BEC	MOV EBP,ESP

OllyDbg工具实战示例 - 你在哪里

注意：此时的地址空间是模块USER32，地址空间就是在某一时刻，CPU的CS:EIP所指向代码的所有者

CPU - 主线程, 模块 - TraceMe			地址空间	CPU - 主线程, 模块 - USER32		
地址	HEX 数据	反汇编		地址	HEX 数据	反汇编
004013A0	\$ 55	PUSH EBP		77D6B05E	8BFF	MOV EDI, EDI
004013A1	8BEC	MOV EBP, ESP		77D6B060	55	PUSH EBP

接着，在USER32的地址空间中，在77D6B05E代码行按下F2，下断点，然后按F9运行，然后键入下图所示：



OllyDbg工具实战示例 - 跳转

点check，可以看到程序被OD截停在下断点的地方，如图所示：

地址	HEX 数据	反汇编	注释
77D6B05E	8BFF	MOV EDI,EDI	USER32.GetDlgItemTextA
77D6B060	55	PUSH EBP	
77D6B061	8BFC	MOV EBP,ESP	

接着，按Alt+F9，返回到用户代码，可以回到：

地址	HEX 数据	反汇编	注释
004011B6	. 8D8C24 9C000	LEA ECX,DWORD PTR SS:[ESP+9C]	
004011BD	. 6A 65	PUSH 65	Count = 65 (101.) Buffer ControlID = 3E8 (1000.) hWnd GetDlgItemTextA
004011BF	. 51	PUSH ECX	
004011C0	. 68 E8030000	PUSH 3E8	
004011C5	. 56	PUSH ESI	
004011C6	. 8BD8	MOV EBX,EAX	
004011C8	. FFD7	CALL EDI	

注意：这里又回到TraceMe的地址空间了

可以按Alt+B，将GetDlgItemTextA的断点改为已禁止

B 断点			
地址	模块	激活	反汇编
77D6B05E	USER32	已禁止	MOV EDI,EDI

OllyDbg工具实战示例 - 爆破

在进行真实的爆破攻击时，我们需要要明白这些指令的含义，此处只是爆破的示例，所以接下来就是修改004011F5处的指令

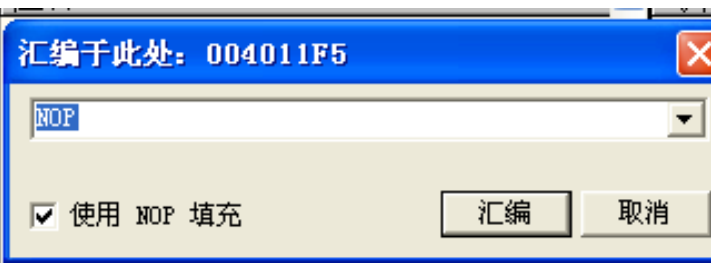
004011E4	. 50	PUSH EAX	
004011E5	. E8 56010000	CALL TraceMe.00401340	序号号计算的CALL
004011EA	. 8B3D BC40400	MOV EDI,DWORD PTR DS:[&USER32.GetDlgIt	USER32.GetDlgItem
004011F0	. 83C4 0C	ADD ESP,0C	
004011F3	. 85C0	TEST EAX,EAX	EXA=0,注册失败; EXA=1, 注册成功
004011F5	. 74 37	JE SHORT TraceMe.0040122E	不跳转则成功

OllyDbg工具实战示例 - 爆破

004011F5 . 74 37 JE SHORT TraceMe.0040122E

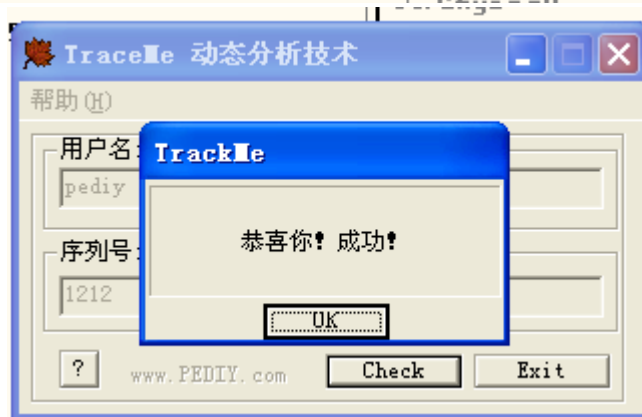
修改反汇编代码段，双击反汇编列后按空格键，键入NOP，汇编

004011DC	. 8D8424 A0000	LEA EAX,DWORD PTR SS:[ESP+A0]
004011E3	. 52	PUSH EDX
004011E4	. 50	PUSH EAX
004011E5	. E8 56010000	CALL TraceMe.00401340
004011EA	. 8B3D BC40400	MOV EDI,DWORD PTR DS:[<&USER32.GetDlgIt
004011F0	. 83C4 0C	ADD ESP,0C
004011F3	. 85C0	TEST EAX,EAX
004011F5	90	NOP
004011F6	90	NOP
004011F7	. 8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]
004011F8	. 51	PUSH ECX



不跳转则成功

最后 F9运行，你会看到：



OllyDbg工具实战示例 - 保存修改后的代码

Address	Disassembly	Comment
004011F0	83C4 0C	ADD ESP,0C
004011F3	85C0	TEST EAX,EAX
004011F5	90	NOP
004011F6	90	NOP
004011F7	8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C
004011F8	51	PUSH ECX
004011FC	68 E4544000	PUSH TraceMe.004054E4
00401201	FF15 60404000	CALL DWORD PTR DS:[<&KERNEL
00401207	6A 00	PUSH 0
00401209	6A 6E	PUSH 6E
0040120B	56	PUSH ESI
0040120C	FFD7	CALL EDI
0040120E	8B1D A4404000	MOV EBX,DWORD PTR DS:[<&USE
00401214	50	PUSH EAX
00401215	FFD3	CALL EBX
00401217	6A 00	PUSH 0

HIT 跟踪

RUN 跟踪

此处为新 ZIP Ctrl+Gray *

转到

线程

数据窗口中跟随

查找 (S)

查找参考 (R)

查看

复制到可执行文件

分析

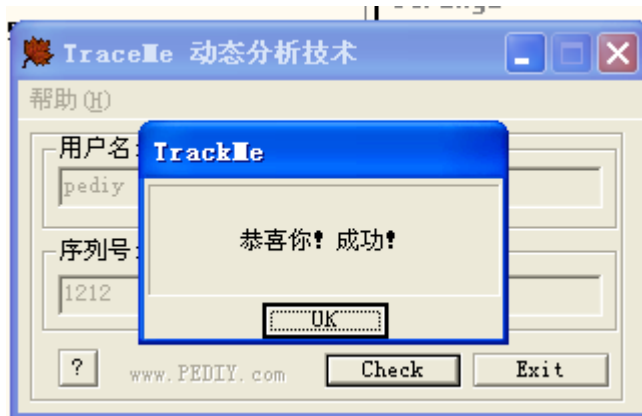
AJunk

004054E4

110.)

00

011yDbg工具实战示例 - 再次运行



感谢大家！



南京邮电大学
Nanjing University of Posts and Telecommunications