

# 工程认证课程目标

- 课程目标3.1：使学生具有一定的自学能力和信息获取能力
- 课程目标3.2：使学生具有进行程序逆向分析，阅读复杂程序的能力
- 课程目标3.3：使学生具有对实际逆向分析问题研究分析、设计解决方案的能力

# 课程目标与**OBE(outcome based education)**

## 毕业要求

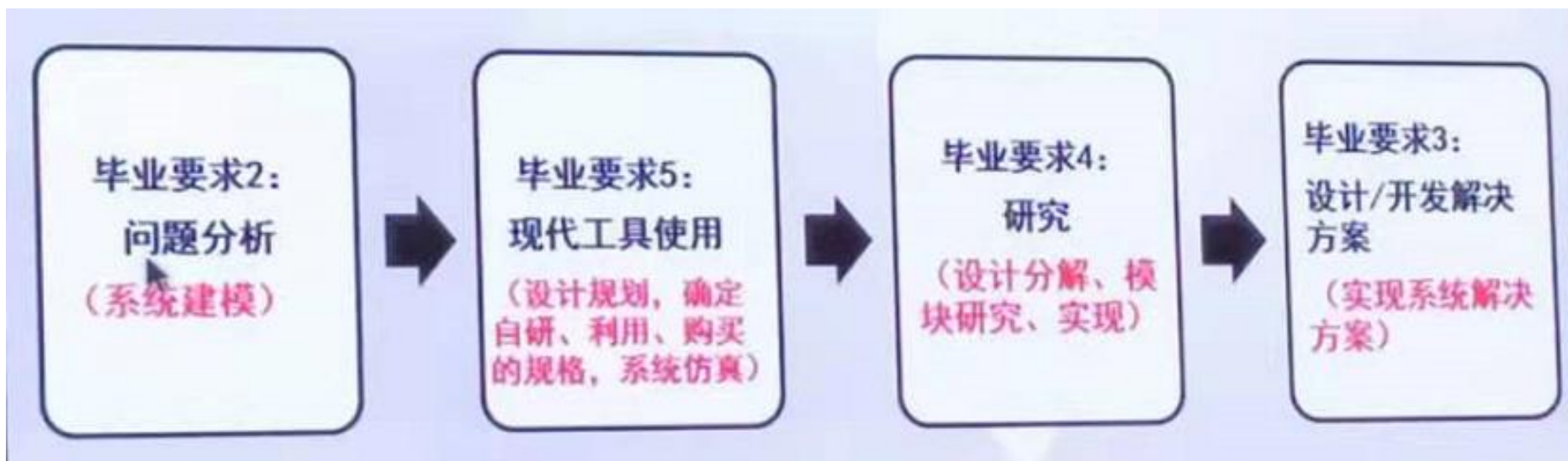
毕业要求	指标点	课程目标
4、研究	4-3-M 针对设计或开发的解决方案，能够基于信息安全领域科学原理对其进行研究，并能够通过理论证明、实验仿真或者系统实现等多种科学方案说明其有效性、合理性，并对解决方案的实施质量进行分析，通过信息综合得到合理有效的结论	3.1
5、使用现代工具	5-3-M 能够分析比较所使用的技术、资源和工具的优势和不足，并理解与表述问题解决方案的局限性。	3.2
3、设计/开发解决方案	3-3-M 充分理解信息安全领域软硬件系统的基础上，能够设计或开发满足特定需求和约束条件的信息安全系统、模块或算法流程，并能够进行系统级优化。	3.3

# 毕业要求与产品生命周期



# 毕业要求与产品生命周期

- 毕业要求2、3、4、5的能力协同



# 逆向分析技术

## 课程回顾

# 面向汇编的逐句解析 - 追踪字符串

\* KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

File View Debug Plugins Options Window Help Tools BreakPoint->

Paused

```
00401416 . 83FA 44      cmp     edx,0x44
00401419 ~ 75 2E        jnz     short Crackme3.00401449
0040141B . 0FBE4424 07  movsx   eax,byte ptr ss:[esp+0x7]
00401420 . 83E8 2E      sub     eax,0x2E
00401423 ~ 75 24        jnz     short Crackme3.00401449
00401425 . 807C24 0A 4D  cmp     byte ptr ss:[esp+0xA],0x4D
0040142A ~ 75 1D        jnz     short Crackme3.00401449
0040142C . 0FBE4C24 06  movsx   ecx,byte ptr ss:[esp+0x6]
00401431 . 83C1 0A      add     ecx,0xA
00401434 . 83F9 33      cmp     ecx,0x33
00401437 ~ 75 10        jnz     short Crackme3.00401449
00401439 . 6A 00        push    0x0
0040143B . 6A 00        push    0x0
0040143D . 68 20304000  push    Crackme3.00403020
00401442 . 8BCE        mov     ecx,esi
00401444 . E8 0D020000  call    <jmp.&MFC42.##?MessageBoxA@CWnd@@QA
00401449 > 5E          pop     esi
0040144A . 83C4 10      add     esp,0x10
0040144D . C3          retn
0040144E . 90          nop
0040144F . 90          nop
00401450 . 8B41 20      mov     eax,dword ptr ds:[ecx+0x20]
00401453 . 6A 00        push    0x0
```

ASCII "Serial is Correct!!!"

Crackme3.<ModuleEntryPoint>

kernel32.74C18494

Enable = FALSE

00403020=Crackme3.00403020 (ASCII "Serial is Correct!!")

# 面向汇编的逐句解析 - 长度判断

KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

```
004013CB  90      nop
004013CC  90      nop
004013CD  90      nop
004013CE  90      nop
004013CF  90      nop
004013D0  83EC 10  sub esp,0x10
004013D3  8D4424 00 lea eax,dword ptr ss:[esp]
004013D7  56      push esi
004013D8  6A 0F   push 0xF
004013DA  50      push eax
004013DB  8BF1    mov esi,ecx
004013DD  68 E8030000 push 0x3E8
004013E2  E8 75020000 call <jmp.&MFC42.##?GetDlgItemTextA@C
004013E7  83F8 08  cmp eax,0x8
004013EA  75 5D   jnz short Crackme3.00401449
004013EC  807C24 09 2D cmp byte ptr ss:[esp+0x9],0x2D
004013F1  75 56   jnz short Crackme3.00401449
004013F3  0FBEC24 04 movsx ecx,byte ptr ss:[esp+0x4]
004013F8  D1E1    shl ecx,1
004013FA  83F9 64  cmp ecx,0x64
004013FD  75 4A   jnz short Crackme3.00401449
004013FF  8A4424 0B mov al,byte ptr ss:[esp+0x8]
00401403  84C0    test al,al
```

eax=00000006

Project PolyPhemous-NTS-Crackme3

Serial

Serial

123123

Check

Close

Cyclops / REAL

# 面向汇编的逐句解析 - SS:[esp+0x9]

BP P VB Notepad Calc Folder CMD Exit

Paused

File View Debug Plugins Options Window Help Tools BreakPoint->

BP P VB Notepad Calc Folder CMD Exit

Registers (FPU)

EAX 00000006  
ECX 7D194322  
EDX 00000000  
EBX 00000001  
ESP 0019F4AC  
EBP 0019 Increment Plus  
ESI 0019 Decrement Minus  
EDI 0019  
EIP 0040  
C 1 ES Set to 1  
P 0 CS Modify Enter  
A 1 SS Copy selection to clipboard Ctrl+C  
Z 0 DS Copy all registers to clipboard  
S 1 FS  
T 0 GS Follow in Dump  
D 0 Follow in Stack  
O 0 Las  
EFL 0000 View MMX registers  
ST0 enpt View 3DNow! registers  
ST1 enpt View debug registers  
ST2 enpt  
ST3 enpt HW break [ESP]  
ST4 enpt  
Appearance

004013CB 90 nop  
004013CC 90 nop  
004013CD 90 nop  
004013CE 90 nop  
004013CF 90 nop  
004013D0 83EC 10 sub esp,0x10  
004013D3 8D4424 00 lea eax,dword ptr ss:[esp]  
004013D7 56 push esi  
004013D8 6A 0F push 0xF  
004013DA 50 push eax  
004013DB 8BF1 mov esi,ecx  
004013DD 68 E8030000 push 0x3E8  
004013E2 E8 75020000 call <jmp.&MFC42.??GetDlgItemTextA@C  
004013E7 83F8 08 cmp eax,0x8  
004013EA 74 5D je short Crackme3.00401449  
004013EC 807C24 09 2D cmp byte ptr ss:[esp+0x9],0x2D  
004013F1 75 56 jnz short Crackme3.00401449  
004013F3 0FB84C24 04 movsx ecx,byte ptr ss:[esp+0x4]  
004013F8 D1E1 shl ecx,1  
004013FA 83F9 64 cmp ecx,0x64  
004013FD 75 4A jnz short Crackme3.00401449  
004013FF 8A4424 00 mov al,byte ptr ss:[esp+0xB]  
00401403 84C0 test al,al

注册码长度等于8，不相等就跳出去，注册失败

Stack ss:[0019F4B5]=33 ('3')

Address Hex dump ASCII

0019F4AC D8 3F 0C 0F 31 32 33 31 32 33 00 00 D8 13 40 00 ?..123123..?@.  
0019F4BC AD 84 0F 0F D5 83 0F 0F D8 F4 19 00 6B 05 0F 0F 球...棋...何...k...  
0019F4CC D8 3F 0C 0F 00 00 00 00 11 01 00 00 10 F5 19 00 ?..?...?..?  
0019F4DC 5A 2F 0F 0F 01 00 00 00 A0 84 0F 0F 00 00 00 00 Z/...?...  
0019F4EC 0C 00 00 00 00 00 00 00 48 FE 19 00 48 FE 19 00 .....H?.H?  
0019F4FC 00 00 00 00 5C 80 1C 0F 2D 11 00 00 48 FE 19 00 ....\...-...H?  
0019F50C 32 B6 00 7D 38 F5 19 00 D7 37 12 0F 01 00 00 00 2?}8?..?..  
0019F51C 00 00 00 00 00 00 00 00 48 FE 19 00 .....H?..

0019F4AC 0F 0C3FD8 mfc42.??messageMap@CDialo  
0019F4B0 31333231  
0019F4B4 00003332  
0019F4B8 004013D0 Crackme3.004013D0  
0019F4BC 0F0F84AD RETURN to mfc42.0F0F84AD from mfc42.0F0F8350  
0019F4C0 0F0F83D5 RETURN to mfc42.0F0F83D5  
0019F4C4 0019F4D8  
0019F4C8 0F0F056B RETURN to mfc42.0F0F056B from mfc42.0F0F83C0  
0019F4CC 0F0C3FD8 mfc42.??messageMap@CDialog@@1UAFX\_MSGMAP@@\_4234

Command:

M1 M2 M3 M4 M5

ESP EBP NONE



# 面向汇编的逐句解析 - 字符判断1

KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

File View Debug Plugins Options Window Help Tools BreakPoint->

Paused

Address	Disassembly	Comment
004013CB	90	nop
004013CC	90	nop
004013CD	90	nop
004013CE	90	nop
004013CF	90	nop
004013D0	. 83EC 10	sub esp,0x10
004013D3	. 8D4424 00	lea eax,dword ptr ss:[esp]
004013D7	. 56	push esi
004013D8	. 6A 0F	push 0xF
004013DA	. 50	push eax
004013DB	. 8BF1	mov esi,ecx
004013DD	. 68 E8030000	push 0x3E8
004013E2	. E8 75020000	call <jmp.&MFC42.##?GetDlgItemTextA@C
004013E7	. 83F8 08	cmp eax,0x8
004013EA	74 5D	je short Crackme3.00401449
004013EC	. 807C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D
004013F1	75 56	jnz short Crackme3.00401449
004013F3	. 0FBE4C24 04	movsx ecx,byte ptr ss:[esp+0x4]
004013F8	. D1E1	shl ecx,1
004013FA	. 83F9 64	cmp ecx,0x64
004013FD	75 4A	jnz short Crackme3.00401449
004013FF	. 8A4424 0B	mov al,byte ptr ss:[esp+0xB]
00401403	. 84C0	test al,al

Stack ss:[0019F4B5]=33 ('3')

注册码长度等于8，不相等就跳出去，注册失败

字符“-”所以注册码第6位为-

# 面向汇编的逐句解析 - 字符判断2

KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

File View Debug Plugins Options Window Help Tools BreakPoint->

Paused [Icons] L E M T W H C / K B R ... S [Icons]

BP P VB Notepad Calc Folder CMD Exit

[Icons] 吾爱破解

```
004013D3 . 8D4424 00 lea eax,dword ptr ss:[esp]
004013D7 . 56 push esi
004013D8 . 6A 0F push 0xF
004013DA . 50 push eax
004013DB . 8BF1 mov esi,ecx
004013DD . 68 E8030000 push 0x3E8
004013E2 . E8 75020000 call <jmp.&MFC42.##?GetDlgItemTextA@C
004013E7 . 83F8 08 cmp eax,0x8
004013EA . 74 5D je short Crackme3.00401449
004013EC . 807C24 09 2D cmp byte ptr ss:[esp+0x9],0x2D
004013F1 . 74 56 je short Crackme3.00401449
004013F3 . 0FBE4C24 04 movsx ecx,byte ptr ss:[esp+0x4]
004013F8 . D1E1 shl ecx,1
004013FA . 83F9 64 cmp ecx,0x64
004013FD . 75 4A jnz short Crackme3.00401449
004013FF . 8A4424 0B mov al,byte ptr ss:[esp+0xB]
00401403 . 84C0 test al,al
00401405 . 74 42 je short Crackme3.00401449
00401407 . 807C24 08 2B cmp byte ptr ss:[esp+0x8],0x2B
0040140C . 75 3B jnz short Crackme3.00401449
0040140E . 0FBE5424 05 movsx edx,byte ptr ss:[esp+0x5]
00401413 . 83C2 0A add edx,0xA
00401416 . 83FA 44 cmp edx,0x44
```

ecx=00000062

注册码长度等于8，不相等就跳出去，注册失败

字符“-”所以注册码第6位为-

注册码第1位扩展到ECX  
左移一位

Registers (MMX)

EAX 00000006

ECX 00000062

EDX 00000000

EBX 00000001

ESP 0019F4AC

EBP 0019F4C4

ESI 0019FE48

EDI 0019FE48

EIP 004013FA Crackme3.004013FA

C 0 ES 002B 32bit 0(FFFFFFFF)

P 0 CS 0023 32bit 0(FFFFFFFF)

A 0 SS 002B 32bit 0(FFFFFFFF)

Z 0 DS 002B 32bit 0(FFFFFFFF)

S 0 FS 0053 32bit 3FD000(FFF)

T 0 GS 002B 32bit 0(FFFFFFFF)

D 0

O 0 LastErr ERROR\_SUCCESS (0000)

EFL 00000202 (NO,NB,NE,A,NS,PO,G)

MM0 0000 0000 0000 0000

MM1 0000 0000 0000 0000

MM2 0000 0000 0000 0000

# 面向汇编的逐句解析 - 字符判断3

\* KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

File View Debug Plugins Options Window Help Tools BreakPoint->

Paused

BP P VB Notepad Calc Folder CM

L E M T W H C / K B R ... S

004013DB	. 8BF1	mov esi,ecx	
004013DD	. 68 E8030000	push 0x3E8	
004013E2	. E8 75020000	call <jmp.&MFC42.#?GetDlgItemTextf	
004013E7	. 83F8 08	cmp eax,0x8	注册码长度等于8, 不相等就跳出去, 注册失败
004013EA	~ 74 5D	je short Crackme3.00401449	
004013EC	. 807C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D	字符"- "所以注册码第6位为-
004013F1	~ 74 56	je short Crackme3.00401449	
004013F3	. 0FBE4C24 04	movsx ecx,byte ptr ss:[esp+0x4]	注册码第1位扩展到ECX
004013F8	. D1E1	shl ecx,1	左移一位
004013FA	. 83F9 64	cmp ecx,0x64	字符d,把d右移一位则为32H, 说明注册码第1位是数字2
004013FD	~ 74 4A	je short Crackme3.00401449	
004013FF	. 8A4424 0B	mov al,byte ptr ss:[esp+0x8]	注册码第8位放入AL
00401403	. 84C0	test al,al	
00401405	~ 74 42	je short Crackme3.00401449	
00401407	. 807C24 08 2B	cmp byte ptr ss:[esp+0x8],0x2B	
0040140C	~ 75 3B	jnz short Crackme3.00401449	
0040140E	. 0FBE5424 05	movsx edx,byte ptr ss:[esp+0x5]	
00401413	. 83C2 0A	add edx,0xA	
00401416	. 83FA 44	cmp edx,0x44	
00401419	~ 75 2E	jnz short Crackme3.00401449	
0040141B	. 0FBE4424 07	movsx eax,byte ptr ss:[esp+0x7]	
00401420	. 83E8 2E	sub eax,0x2E	
00401423	. 75 24	jnz short Crackme3.00401449	

al=00

# 面向汇编的逐句解析 - 字符判断4

KernelMode - Crackme3.exe - [\"G.P.U\" - main thread, module Crackme3]

File View Debug Plugins Options Window Help Tools BreakPoint-> BP P VB Notepad Calc Folder CMD Exit

Paused

Address	Disassembly	Comment
004013DB	8BF1	mov esi,ecx
004013DD	68 E8030000	push 0x3E8
004013E2	E8 75020000	call <jmp.&MFC42.##?GetDlgItemText@7C90321>
004013E7	83F8 08	cmp eax,0x8
004013EA	74 5D	je short Crackme3.00401449
004013EC	087C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D
004013F1	74 56	je short Crackme3.00401449
004013F3	0FBEC24 04	movsx ecx,byte ptr ss:[esp+0x4]
004013F8	D1E1	shl ecx,1
004013FA	83F9 64	cmp ecx,0x64
004013FD	74 4A	je short Crackme3.00401449
004013FF	8A4424 0B	mov al,byte ptr ss:[esp+0x8]
00401403	84C0	test al,al
00401405	74 42	je short Crackme3.00401449
00401407	087C24 08 2B	cmp byte ptr ss:[esp+0x8],0x2B
0040140C	75 3B	jnz short Crackme3.00401449
0040140E	0FBEC24 05	movsx edx,byte ptr ss:[esp+0x5]
00401413	83C2 0A	add edx,0xA
00401416	83FA 44	cmp edx,0x44
00401419	75 2E	jnz short Crackme3.00401449
0040141B	0FBEC24 07	movsx eax,byte ptr ss:[esp+0x7]
00401420	83E8 2E	sub eax,0x2E
00401423	75 24	jnz short Crackme3.00401449

al=00

注册码长度等于8，不相等就跳出去，注册失败

字符“-“所以注册码第6位为-

注册码第1位扩展到ECX  
左移一位  
字符d,把d右移一位则为32H，说明注册码第1位是数字2

注册码第8位放入AL  
第8位是否为0

Registers (MMX)

Register	Value
EAX	00000000
ECX	00000062
EDX	00000000
EBX	00000001
ESP	0019F4AC
EBP	0019F4C4
ESI	0019FE48
EDI	0019FE48
EIP	00401403 Crackme3.00401403
C 1	ES 002B 32bit 0(FFFFFFFF)
P 0	CS 0023 32bit 0(FFFFFFFF)
A 1	SS 002B 32bit 0(FFFFFFFF)
Z 0	DS 002B 32bit 0(FFFFFFFF)
S 1	FS 0053 32bit 3FD000(FFF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
O 0	LastErr ERROR_SUCCESS (0)
EFL	00000293 (NO,B,NE,BE,S,PO)
MM0	0000 0000 0000 0000
MM1	0000 0000 0000 0000
MM2	0000 0000 0000 0000

Test将两个操作数进行逻辑与运算：结果是0, ZF标志位为1；结果不是0, ZF为0

当ZF为1时，JE跳转

# 面向汇编的逐句解析 - 字符判断5

\* KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

Address	Disassembly	Comment
004013DB	8BF1	mov esi,ecx
004013DD	68 E8030000	push 0x3E8
004013E2	E8 75020000	call <jmp.&MFC42.##?GetDlgItemTextf
004013E7	83F8 08	cmp eax,0x8
004013EA	74 5D	je short Crackme3.00401449
004013EC	807C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D
004013F1	74 56	je short Crackme3.00401449
004013F3	0FBE4C24 04	movsx ecx,byte ptr ss:[esp+0x4]
004013F8	D1E1	shl ecx,1
004013FA	83F9 64	cmp ecx,0x64
004013FD	74 4A	je short Crackme3.00401449
004013FF	8A4424 0B	mov al,byte ptr ss:[esp+0xB]
00401403	84C0	test al,al
00401405	75 42	jnz short Crackme3.00401449
00401407	807C24 08 2B	cmp byte ptr ss:[esp+0x8],0x2B
0040140C	75 3B	jnz short Crackme3.00401449
0040140E	0FBE5424 05	movsx edx,byte ptr ss:[esp+0x5]
00401413	83C2 0A	add edx,0xA
00401416	83FA 44	cmp edx,0x44
00401419	75 2E	jnz short Crackme3.00401449
0040141B	0FBE4424 07	movsx eax,byte ptr ss:[esp+0x7]
00401420	83E8 2E	sub eax,0x2E
00401423	75 24	jnz short Crackme3.00401449

Stack ss:[0019F4B4]=32 ('2')

注册码长度等于8，不相等就跳出去，注册失败

字符“-”所以注册码第6位为-

注册码第1位扩展到ECX  
左移一位  
字符d，把d右移一位则为32H，说明注册码第1位是数字2

注册码第8位放入AL  
第8位是否为0  
第8位只要不为0即可  
第5位为2B即十进制43，就是+号

# 面向汇编的逐句解析 - 字符判断6

\* KernelMode - Crackme3.exe - [\*G.P.U\* - main thread, module Crackme3]

Address	Disassembly	Comment
004013D8	8BF1	mov esi,ecx
004013DD	68 E8030000	push 0x3E8
004013E2	E8 75020000	call <jmp.&MFC42.##?GetDlgItemTextf>
004013E7	83F8 08	cmp eax,0x8
004013EA	74 5D	je short Crackme3.00401449
004013EC	807C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D
004013F1	74 56	je short Crackme3.00401449
004013F3	0FB E4C24 04	movsx ecx,byte ptr ss:[esp+0x4]
004013F8	D1E1	shl ecx,1
004013FA	83F9 64	cmp ecx,0x64
004013FD	74 4A	je short Crackme3.00401449
004013FF	8A4424 0B	mov al,byte ptr ss:[esp+0x8]
00401403	84C0	test al,al
00401405	75 42	jnz short Crackme3.00401449
00401407	807C24 08 2B	cmp byte ptr ss:[esp+0x8],0x2B
0040140C	74 3B	je short Crackme3.00401449
0040140E	0FB E5424 05	movsx edx,byte ptr ss:[esp+0x5]
00401413	83C2 0A	add edx,0xA
00401416	83FA 44	cmp edx,0x44
00401419	75 2E	jnz short Crackme3.00401449
0040141B	0FB E4424 07	movsx eax,byte ptr ss:[esp+0x7]
00401420	83E8 2E	sub eax,0x2E
00401423	75 24	jnz short Crackme3.00401449

edx=00000032

注册码长度等于8，不相等就跳出去，注册失败

字符“-”所以注册码第6位为-

注册码第1位扩展到ECX  
左移一位  
字符d，把d右移一位则为32H，说明注册码第1位是数字2

注册码第8位放入AL  
第8位是否为0  
第8位只要不为0即可  
第5位为2B即十进制43，就是+号

注册码第2位零扩展到EDX  
与000AH相加  
加完之后等于44H，即第2位为44-A=3A，也就是:号

逆向分析技术

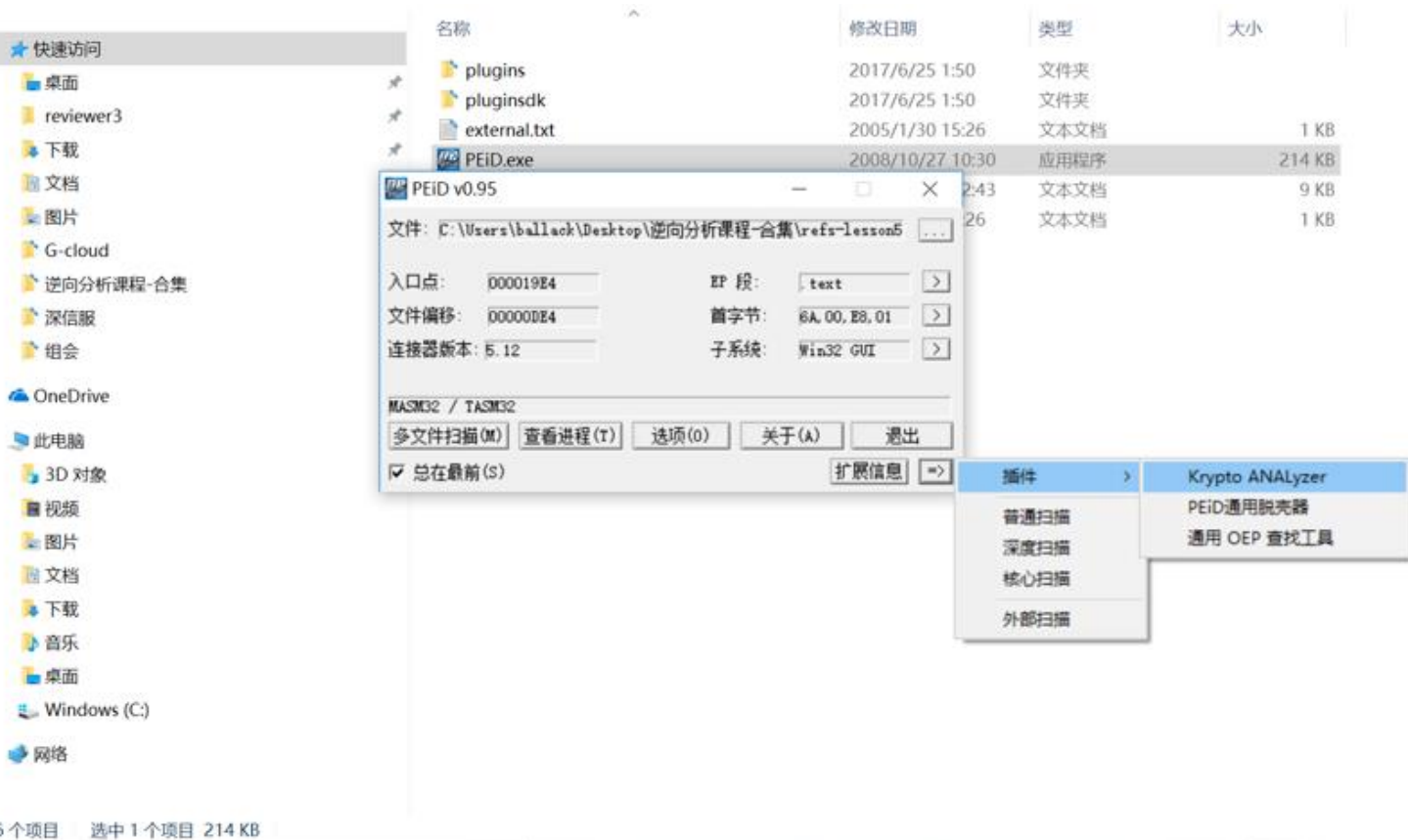
加密算法逆向

# 加密算法逆向 - 示例运行





# 加密算法逆向 - PEiD扫描



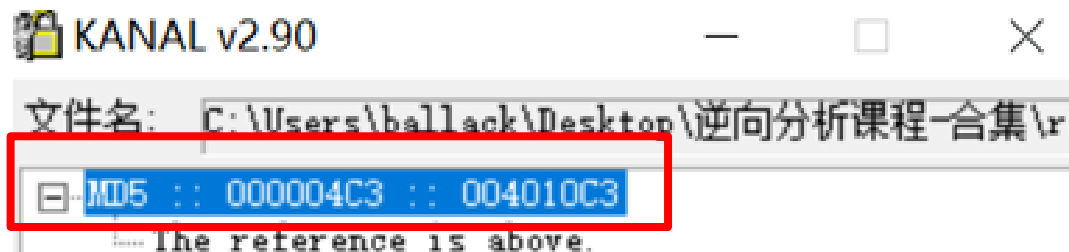
# 加密算法逆向 - PEiD扫描



## CM2: 加密算法逆向 - PEiD扫描

```
.text:004010B8      and     eax, ecx
.text:004010BA      or      eax, ebx
.text:004010BC      add     eax, [ebp+var_10]
.text:004010BF      add     eax, [edi+4]
.text:004010C2      add     eax, 0E8C7B756h
.text:004010C7      mov     ci, 0Ch
```

004010B3	D3 C0 03 45 F8 89 45 FC	8B 45 FC 8B 5D F8 8B 4D	...E...E...E...J...M
004010B3	F4 23 D0 F7 D0 23 C1 0B	C3 03 45 F0 03 47 04 05	.#...#...E..G..
004010C3	56 B7 C7 E8 B1 0C D3 C0	03 45 FC 89 45 F0 8B 45	U.....E..E..E
004010D3	F0 8B 5B FC 8B 4D F8 23	D8 F7 D0 23 C1 0B C3 03	..]..M.#...#....
004010E3	45 F4 03 47 08 05 DB 70	20 24 B1 11 D3 C0 03 45	E..G...p \$....E



MD5算法是什么？

## 加密算法逆向 - MD5

### 1、数据初始化过程

首先要对需加密消息进行数据填充，使消息的长度对512取模后等于448，消息长度  $\text{mod } 512 = 448$ 。不足位数在消息（字符串）后面进行填充，填充第一位为1，其余为0。即第一个字节为  $0x80$ ，其余字节为  $0x00$ 。

## 加密算法逆向 - MD5

### 2、添加消息长度

再填充上**原消息的长度**，例如欲加密字符为16个ASCII，即16字节 $\times$ 8位=128位，128即16进制的0x80。但它是一个QWORD值，占用8个字节(64位)。

在此步骤进行完毕后，最终**消息长度就是512的整数倍**。  
(448 + 64 = 512)

## 加密算法逆向 - MD5

### 3、数据加密处理

准备需要用到的数据:

4个种子值:  $A = 0x67452301$ ,  $B = 0xEFCDAB89$ ,  $C = 0x98BADCFE$ ,  
 $D = 0x10325476$ ;

4个函数:  $F(X,Y,Z)=(X \& Y) \mid ((\sim X) \& Z)$ ;  $G(X,Y,Z)=(X \& Z) \mid (Y \& (\sim Z))$ ;  $H(X,Y,Z)=X \wedge Y \wedge Z$ ;  $I(X,Y,Z)=Y \wedge (X \mid (\sim Z))$ ;

把消息分以512位为一分组进行处理, 每一个分组进行4轮变换, 以上面所说4个常数为起始变量进行计算, 重新输出4个变量, 以这4个变量再进行下一分组的运算, 如果已经是最后一个分组, 则这4个变量为最后的结果, 即MD5值。

# 加密算法逆向 - MD5

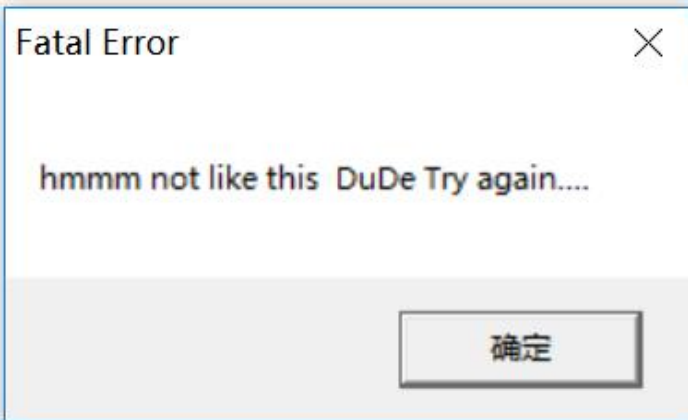


RAR 压缩文件

60 KB

文档

1 KB



# 加密算法逆向 - MD5 - 找入口

IDA - C:\Users\ballack\Desktop\逆向分析课程-合集\refs-lesson5\CM2.exe

File Edit Jump Search View Debugger Options Windows Help

BP P VB Notepad Calc Folder CMD Ext

Library function Data Regular function Unexplored Instruction External symbol

Functions window

Function name

- ☒ sub\_401000
- ☒ start
- ☒ DialogFunc
- ☒ sub\_401B79
- ☒ ExitProcess
- ☒ GetModuleHandleA
- ☒ lstrcatA
- ☒ lstrcmpA
- ☒ wsprintfA
- ☒ DialogBoxParamA
- ☒ EndDialog
- ☒ GetDlgItemTextA
- ☒ LoadIconA
- ☒ MessageBoxA
- ☒ SendMessageA

IDA View-A

Strings window

Hex View-1

Structures

Enums

Imports

Exports

```
.text:00401C0A : [00000006 BYTES: COLLAPSED FUNCTION EndDialog. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:00401C10
.text:00401C10 : ===== SUBROUTINE =====
.text:00401C10 : Attributes: thunk
.text:00401C10
.text:00401C10 : UINT __stdcall GetDlgItemTextA(HWND hDlg, int nIDDlgItem, LPSTR lpString, int cchMax)
.text:00401C10 GetDlgItemTextA proc near
.text:00401C10 : CODE XREF: DialogFunc+94fp
.text:00401C10 : DialogFunc+C9fp
.text:00401C10
.text:00401C10 hDlg          = dword ptr 4
.text:00401C10 nIDDlgItem    = dword ptr 8
.text:00401C10 lpString      = dword ptr 0Ch
.text:00401C10 cchMax        = dword ptr 10h
.text:00401C10
.text:00401C10 jmp ds:__imp_GetDlgItemTextA
.text:00401C10 GetDlgItemTextA endp
.text:00401C10
.text:00401C16 : [00000006 BYTES: COLLAPSED FUNCTION LoadIconA. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:00401C1C : [00000006 BYTES: COLLAPSED FUNCTION MessageBoxA. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:00401C22 : [00000006 BYTES: COLLAPSED FUNCTION SendMessageA. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:00401C28 align 200h
.text:00401E00 dd 80h dup(?)
.text:00401E00 _text ends
.text:00401E00
.idata:00402000 : Section 2. (virtual address 00002000)
.idata:00402000 : Virtual size : 0000015E ( 350.)
.idata:00402000 : Section size in file : 00000200 ( 512.)
.idata:00402000 : Offset to raw data for section: 00001200
.idata:00402000 : Flags 40000040: Data Readable
.idata:00402000 : Alignment : default
.idata:00402000
.idata:00402000 : Imports from KERNEL32.dll
.idata:00402000 :
```

```
loc_401A87: ; CODE XREF: DialogFunc+50fj
cmp [ebp+arg_8], 384h
jnz loc_401B60
push 12Ch ; cchMax
push offset String1 ; lpString
push 64h ; nIDDlgItem
push [ebp+hWnd] ; hDlg
call GetDlgItemTextA
or eax, eax
jnz short loc_401AC5
push 0 ; uType
push offset a00ohInputError ; "000H input Error"
push offset aYourNamePlease ; "Your name please !"
push 0 ; hWnd
call MessageBoxA
leave
```

Line 12 of 15

Output window

Function argument information has been produced



# 加密算法逆向 - MD5 - 找入口

```
.text:00401A87 loc_401A87:                                ; CODE XREF: DialogFunc+5D↑j
.text:00401A87      cmp     [ebp+arg_8], 384h
.text:00401A8E      jnz     loc_401B60
.text:00401A94      push    12Ch                ; cchMax
.text:00401A99      push    offset String1      ; lpString
.text:00401A9E      push    64h                ; nIDDlgItem
.text:00401AA0      push    [ebp+hWnd]          ; hWnd
.text:00401AA3      call    GetDlgItemTextA
```

00401A6E	. 6A 00	push 0x0	Style = MB_OK MB_APPLMODAL
00401A70	. 68 1A324000	push CM2.0040321A	Title = "AbOut"
00401A75	. 68 00304000	push CM2.00403000	Text = " +-----"
00401A7A	. FF75 08	push dword ptr ss:[ebp+0x8]	hOwner = 002D8000
00401A7D	. E8 9A010000	call <jmp.&USER32.MessageBoxA>	MessageBoxA
00401A82	~ E9 EC000000	jmp CM2.00401B73	
00401A87	> 817D 10 8403	cmp dword ptr ss:[ebp+0x10],0x384	
00401A8E	~ 0F85 CC000000	jnz CM2.00401B60	
00401A94	. 68 2C010000	push 0x12C	Count = 12C (300.)
00401A99	. 68 80334000	push CM2.00403380	Buffer = CM2.00403380
00401A9E	. 6A 64	push 0x64	ControlID = 64 (100.)
00401AA0	. FF75 08	push dword ptr ss:[ebp+0x8]	hWnd = 002D8000
00401AA3	. E8 68010000	call <jmp.&USER32.GetDlgItemTextA>	GetDlgItemTextA
00401AA8	. 0BC0	or eax, eax	
00401AAA	~ 75 19	jnz short CM2.00401AC5	
00401AAC	. 6A 00	push 0x0	Style = MB_OK MB_APPLMODAL
00401AAE	. 68 E0324000	push CM2.004032E0	Title = "oooH input Error"
00401AB3	. 68 F1324000	push CM2.004032F1	Text = "Your name please !!!"
00401AB8	. 6A 00	push 0x0	hOwner = NULL
00401ABA	. E8 5D010000	call <jmp.&USER32.MessageBoxA>	MessageBoxA

# 加密算法逆向 - MD5 - 取用户名

吾愛破解 - CM2.exe - [LCG - 主线程, 模块 - CM2]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder C

暂停

l e m t w h c p k b r ... s

```
00401A5F ~ 0F85 0E010000 jnz CM2.00401B73
00401A65 . 817D 10 2C01 cmp dword ptr ss:[ebp+0x10],0x12C
00401A6C ~ 75 19 jnz short CM2.00401A87
00401A6E . 6A 00 push 0x0
00401A70 . 68 1A324000 push CM2.0040321A
00401A75 . 68 00304000 push CM2.00403000
00401A7A . FF75 08 push dword ptr ss:[ebp+0x8]
00401A7D . E8 9A010000 call <jmp.&USER32.MessageBoxA>
00401A82 ~ E9 EC000000 jmp CM2.00401B73
00401A87 > 817D 10 8403 cmp dword ptr ss:[ebp+0x10],0x384
00401A8E ~ 0F85 CC000000 jnz CM2.00401B60
00401A94 . 68 2C010000 push 0x12C
00401A99 . 68 80334000 push CM2.00403380
00401A9E . 6A 64 push 0x64
00401AA0 . FF75 08 push dword ptr ss:[ebp+0x8]
00401AA3 ~ E8 68010000 call <jmp.&USER32.GetDlgItemTextA>
00401AA8 . 0BC0 or eax,eax
00401AAA ~ 75 19 jnz short CM2.00401AC5
00401AAC . 6A 00 push 0x0
00401AAE . 68 E0324000 push CM2.004032E0
00401AB3 . 68 F1324000 push CM2.004032F1
00401AB8 . 6A 00 push 0x0
00401ABA . E8 5D010000 call <jmp.&USER32.MessageBoxA>
00401C10=<jmp.&USER32.GetDlgItemTextA>
```

Style = MB\_OK|MB\_APPLMODAL  
Title = "AbOut"  
Text = " +=====+  
hOwner = 00660CFE ('BytePtr's Crypto kgme #  
MessageBoxA

BytePtr's Crypto kgme ...

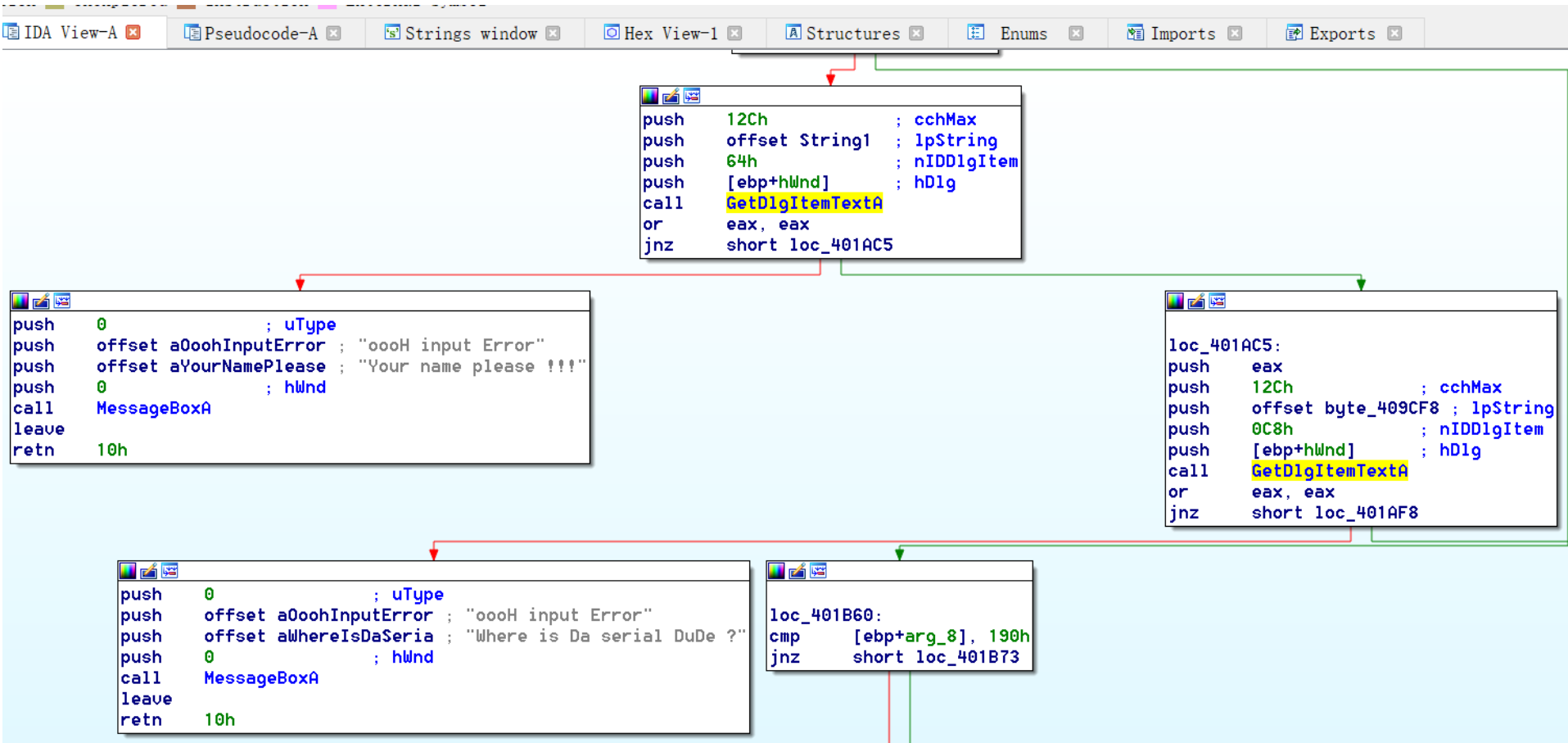
BytePtr  
LilithCPH

Name  
sdfsd

Serial  
sdfdsdf

Try Now About Exit

# 加密算法逆向 - MD5 - IDA看代码



# 加密算法逆向 - MD5 - 开启F5

```
; Attributes: bp-based frame

; INT_PTR __stdcall DialogFunc(HWND, UINT, WPARAM, LPARAM)
DialogFunc proc near

hWnd= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h

push    ebp
mov     ebp, esp
add     esp, 0FFFFFFCh
cmp     [ebp+arg_4], 110h
jnz     short loc_401A43
```

```
loc_401A58:
cmp     [ebp+arg_4], 111h
jnz     loc_401B73
```

```
cmp     [ebp+arg_8], 12Ch
```

## 加密算法逆向 - MD5 - 开启F5

```
7  switch ( a2 )
8  {
9      case 0x110u:
10         u4 = LoadIconA(hInstance, (LPCSTR)0x1F4);
11         SendMessageA(hWnd, 0x80u, 0, (LPARAM)u4);
12         break;
13     case 0x10u:
14         EndDialog(hWnd, 0);
15         break;
16     case 0x111u:
17         switch ( a3 )
18         {
19             case 0x12Cu:
20                 MessageBoxA(hWnd, Text, Caption, 0);
21                 break;
22             case 0x384u:
23                 u5 = GetDlgItemTextA(hWnd, 100, String1, 300);
24                 if ( !u5 )
25                     return MessageBoxA(0, aYourNamePlease, aOoohInputError, 0);
26                 u7 = u5;
27                 if ( !GetDlgItemTextA(hWnd, 200, byte_409CF8, 300) )
28                     return MessageBoxA(0, aWhereIsDaSeria, aOoohInputError, 0);
29                 lstrcatA(String1, String2);
30                 sub_401000(String1, u7, &unk_4056A8);
31                 sub_401B79();
32                 if ( lstrcmpA(byte_409CF8, byte_4079D0) )
33                 {
34                     MessageBoxA(0, aHmmmNotLikeThi, aFatalError, 0);
35                     return 0;
36                 }
37                 MessageBoxA(0, aGoodSerialNowS, aGoodWork, 0);
38                 break;
39             case 0x190u:
40                 EndDialog(hWnd, 0);
```

## 加密算法逆向 - MD5 - 核心位置

```
22     case 0x384u:
23         v5 = GetDlgItemTextA(hWnd, 100, String1, 300);
24         if ( !v5 )
25             return MessageBoxA(0, aYourNamePlease, aOoohInputError, 0);
26         v7 = v5;
27         if ( !GetDlgItemTextA(hWnd, 200, byte_409CF8, 300) )
28             return MessageBoxA(0, aWhereIsDaSeria, aOoohInputError, 0);
29         lstrcatA(String1, String2);
30         sub_401000(String1, v7, &unk_4056A8);
31         sub_401B79();
32         if ( lstrcmpA(byte_409CF8, byte_4079D0) )
33             {
```

# 加密算法逆向 - MD5 - 核心位置

吾爱破解 - CM2.exe - [LCG - 主线程, 模块 - CM2]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder CMD Exit

暂停

l e m t w h c p k b r ... s

寄存器 (FPU)

EAX 00000008  
ECX 7445A8C6 user  
EDX 00000000  
EBX 00000003  
ESP 0019F7DC  
EBP 0019F7E4  
ESI 00660CFE  
EDI 80006010  
EIP 00401ADD CM2.

C 0 ES 002B 32位  
P 1 CS 0023 32位  
A 0 SS 002B 32位  
Z 1 DS 002B 32位  
S 0 FS 0053 32位  
T 0 GS 002B 32位  
D 0  
O 0 LastErr ERROR  
EFL 00000246 (NO, I  
ST0 empty 0.0  
ST1 empty 0.0  
ST2 empty 0.0

00401AAA . 75 19 jnz short CM2.00401AC5  
00401AAC . 6A 00 push 0x0  
00401AAE . 68 E0324000 push CM2.004032E0  
00401AB3 . 68 F1324000 push CM2.004032F1  
00401AB8 . 6A 00 push 0x0  
00401ABA . E8 5D010000 call <jmp.&USER32.MessageBoxA>  
00401ABF . C9 leave  
00401AC0 . C2 1000 retn 0x10  
00401AC3 . EB 33 jmp short CM2.00401AF8  
00401AC5 > 50 push eax  
00401AC6 . 68 2C010000 push 0x12C  
00401ACB . 68 F89C4000 push CM2.00409CF8  
00401AD0 . 68 C8000000 push 0xC8  
00401AD5 . FF75 08 push dword ptr ss:[ebp+0x8]  
00401AD8 . E8 33010000 call <jmp.&USER32.GetDlgItemTextA>  
00401ADD . 0BC0 or eax,eax  
00401ADF . 75 17 jnz short CM2.00401AF8  
00401AE1 . 6A 00 push 0x0  
00401AE3 . 68 E0324000 push CM2.004032E0  
00401AE8 . 68 06334000 push CM2.00403306  
00401AED . 6A 00 push 0x0  
00401AEF . E8 28010000 call <jmp.&USER32.MessageBoxA>  
00401AF4 . C9 leave

Style = MB\_OK|MB\_APPLMODAL  
Title = "oooH input Error"  
Text = "Your name please !!!"  
hOwner = NULL  
MessageBoxA

Count = 12C (300.)  
Buffer = CM2.00409CF8  
ControlID = C8 (200.)  
hWnd = 00660CFE ('BytePtr's Crypto kgme #1'  
GetDlgItemTextA

Style = MB\_OK|MB\_APPLMODAL  
Title = "oooH input Error"  
Text = "Where is Da serial DuDe ?"  
hOwner = NULL  
MessageBoxA

eax=00000008

# 加密算法逆向 - MD5 - 开始比较

吾愛破解 - CM2.exe - [LCG - 主线程, 模块 - CM2]

```
文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder CM
暂停
00401AC6 . 68 2C010000 push 0x12C
00401ACB . 68 F89C4000 push CM2.00409CF8
00401AD0 . 68 C8000000 push 0xC8
00401AD5 . FF75 08 push dword ptr ss:[ebp+0x8]
00401AD8 . E8 33010000 call <jmp.&USER32.GetDlgItemTextA>
00401ADD . 0BC0 or eax, eax
00401ADF . 75 17 jnz short CM2.00401AF8
00401AE1 . 6A 00 push 0x0
00401AE3 . 68 E0324000 push CM2.004032E0
00401AE8 . 68 06334000 push CM2.00403306
00401AED . 6A 00 push 0x0
00401AEF . E8 28010000 call <jmp.&USER32.MessageBoxA>
00401AF4 . C9 leave
00401AF5 . C2 1000 retn 0x10
00401AF8 > 68 34334000 push CM2.00403334
00401AFD . 68 80334000 push CM2.00403380
00401B02 . E8 EB000000 call <jmp.&KERNEL32.lstrcatA>
00401B07 . 58 pop eax
00401B08 . 68 A8564000 push CM2.004056A8
00401B0D . 50 push eax
00401B0E . 68 80334000 push CM2.00403380
00401B13 . E8 E8F4FFFF call CM2.00401000
00401B18 . E8 5C000000 call CM2.00401B79
Count = 12C (300.)
Buffer = CM2.00409CF8
ControlID = C8 (200.)
hWnd = 00660CFE ('BytePtr's Crypto kgme #1')
GetDlgItemTextA

Style = MB_OK|MB_APPLMODAL
Title = "oooH input Error"
Text = "Where is Da serial DuDe ?"
hOwner = NULL
MessageBoxA

StringToAdd = "BytePtr [e!]"
ConcatString = "sdfsdf"
lstrcatA

ASCII "sdfsdf"

00403334=CM2.00403334 (ASCII "BytePtr [e!]")
跳转来自 00401AC3, 00401ADF
```

基于输入的用户名生成MD5值，然后与输入的序列号进行对比



# 加密算法逆向 - MD5 - 生成字符串X

- CM2.exe - [LCG - 主线程, 模块 - CM2]

查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder CMD Exit

l e m t w h c p k b r ... s 吾爱破解

68 2C010000	push 0x12C	Count = 12C (300.)
68 F89C4000	push CM2.00409CF8	Buffer = CM2.00409CF8
68 C8000000	push 0xC8	ControlID = C8 (200.)
FF75 08	push dword ptr ss:[ebp+0x8]	hWnd = 00660CFE ('BytePtr's Crypto kgme #1')
E8 33010000	call <jmp.&USER32.GetDlgItemTextA>	GetDlgItemTextA
0BC0	or eax,eax	CM2.00403380
75 17	jnz short CM2.00401AF8	
6A 00	push 0x0	Style = MB_OK MB_APPLMODAL
68 E0324000	push CM2.004032E0	Title = "oooH input Error"
68 06334000	push CM2.00403306	Text = "Where is Da serial DuDe ?"
6A 00	push 0x0	hOwner = NULL
E8 28010000	call <jmp.&USER32.MessageBoxA>	MessageBoxA
C9	leave	
C2 1000	retn 0x10	
68 34334000	push CM2.00403334	StringToAdd = "BytePtr [e!]"
68 80334000	push CM2.00403380	ConcatString = "sdfsdFBytePtr [e!]"
E8 EB000000	call <jmp.&KERNEL32.lstrcatA>	lstrcatA
58	pop eax	CM2.00403380
68 A8564000	push CM2.004056A8	
50	push eax	CM2.00403380
68 80334000	push CM2.00403380	ASCII "sdfsdFBytePtr [e!]"
E8 E8F4FFFF	call CM2.00401000	
E8 5C000000	call CM2.00401B79	

寄存器 (FPU)  
EAX 00403380 ASCII "sdfsdFBytePtr [e!]"  
ECX 74C582EA kernel32.74C582EA  
EDX 00000000  
EBX 00000003  
ESP 0019F7DC  
EBP 0019F7E4  
ESI 00660CFE  
EDI 80006010  
EIP 00401B07 CM2.00401B07  
C 0 ES 002B 32 0(FFFFFFFF)  
P 0 CS 0023 32 0(FFFFFFFF)  
A 0 SS 002B 32 0(FFFFFFFF)  
Z 0 DS 002B 32 0(FFFFFFFF)  
S 0 FS 0053 32 2DB000(FFF)  
T 0 GS 002B 32 0(FFFFFFFF)  
D 0  
O 0 LastErr ERROR\_SUCCESS (00000000)  
EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)  
ST0 empty 0.0  
ST1 empty 0.0

# 加密算法逆向 - MD5 - 函数的步入处理

```
暂停 | [Icons] | l e m t w h c p k b r ... s | [Icons] | [Icons]
00401ADF ~ 75 17 jnz short CM2.00401AF8
00401AE1 . 6A 00 push 0x0
00401AE3 . 68 E0324000 push CM2.004032E0
00401AE8 . 68 06334000 push CM2.00403306
00401AED . 6A 00 push 0x0
00401AEF . E8 28010000 call <jmp.&USER32.MessageBoxA>
00401AF4 . C9 leave
00401AF5 . C2 1000 retn 0x10
00401AF8 > 68 34334000 push CM2.00403334
00401AFD . 68 80334000 push CM2.00403380
00401B02 . E8 EB000000 call <jmp.&KERNEL32.lstrcatA>
00401B07 . 58 pop eax
00401B08 . 68 A8564000 push CM2.004056A8
00401B0D . 50 push eax
00401B0E . 68 80334000 push CM2.00403380
00401B13 . E8 E8F4FFFF call CM2.00401000
00401B18 . E8 5C000000 call CM2.00401B79
00401B1D . 68 D0794000 push CM2.004079D0
00401B22 . 68 F89C4000 push CM2.00409CF8
00401B27 . E8 CC000000 call <jmp.&KERNEL32.lstrcmpA>
00401B2C . 85C0 test eax, eax
00401B2E ~ 75 15 jnz short CM2.00401B45
00401B30 . 6A 00 push 0x0
00401000=CM2.00401000
```

```
[Style = MB_OK|MB_APPLMODAL
Title = "oooH input Error"
Text = "Where is Da serial DuDe ?"
hOwner = NULL
MessageBoxA

StringToAdd = "BytePtr [e!]"
ConcatString = "sdfsdBytePtr [e!]"
lstrcatA
CM2.00403380

ASCII "sdfsdBytePtr [e!]"
需要跟入:F7

String2 = ""
String1 = "sdfdf sdf"
lstrcmpA

Style = MB_OK|MB_APPLMODAL
```

# 加密算法逆向 - MD5 - 定位MD5

```
.text:00401B07      pop     eax
.text:00401B08      push    offset unk_4056A8
.text:00401B0D      push    eax
.text:00401B0E      push    offset String1
.text:00401B13      call    sub_401000
.text:00401B18      call    sub_401B79
.text:00401B1D      push    offset bvar_10      = dword ptr -10h
.text:00401B22      push    offset bvar_C       = dword ptr -0Ch
.text:00401B27      call    lstrcmpA            = dword ptr -8
.text:00401B2C      test    eax, eax            = dword ptr -4
.text:00401B2E      jnz     short loc_40106F     = dword ptr 8
.text:00401B30      push    0                   = dword ptr 0Ch
.text:00401B32      push    offset arg_8         = dword ptr 10h
.text:00401B37      push    offset arg_8         = dword ptr 10h
```

Now send your Tut & KeyGen".

```
.text:00401048      mov     [edi-4], edx
.text:0040104B      mov     edx, [ebp+arg_4]
.text:0040104E      mov     edi, [ebp+arg_0]
.text:00401051      mov     esi, [ebp+arg_8]
.text:00401054      mov     dword ptr [esi], 67452301h
.text:0040105A      mov     dword ptr [esi+4], 0EFCDAB89h
.text:00401061      mov     dword ptr [esi+8], 98BADCFEh
.text:00401068      mov     dword ptr [esi+0Ch], 10325476h
.text:0040106F      loc_40106F: ; CODE XREF: sub_401000+9BF↓j
```

种子值: A = 0x67452301, B = 0xEFCDAB89,  
C = 0x98BADCFE, D = 0x10325476;

# 加密算法逆向 - MD5 - 函数的步入处理

```
暂停 | [Icons] | l e m t w h c p k b r ... s | [Icons] | [Icons]
00401ADF ~ 75 17 jnz short CM2.00401AF8
00401AE1 . 6A 00 push 0x0
00401AE3 . 68 E0324000 push CM2.004032E0
00401AE8 . 68 06334000 push CM2.00403306
00401AED . 6A 00 push 0x0
00401AEF . E8 28010000 call <jmp.&USER32.MessageBoxA>
00401AF4 . C9 leave
00401AF5 . C2 1000 retn 0x10
00401AF8 > 68 34334000 push CM2.00403334
00401AFD . 68 80334000 push CM2.00403380
00401B02 . E8 EB000000 call <jmp.&KERNEL32.lstrcatA>
00401B07 . 58 pop eax
00401B08 . 68 A8564000 push CM2.004056A8
00401B0D . 50 push eax
00401B0E . 68 80334000 push CM2.00403380
00401B13 . E8 E8F4FFFF call CM2.00401000
00401B18 . E8 5C000000 call CM2.00401B79
00401B1D . 68 D0794000 push CM2.004079D0
00401B22 . 68 F89C4000 push CM2.00409CF8
00401B27 . E8 CC000000 call <jmp.&KERNEL32.lstrcmpA>
00401B2C . 85C0 test eax, eax
00401B2E ~ 75 15 jnz short CM2.00401B45
00401B30 . 6A 00 push 0x0
00401000=CM2.00401000
```

```
[Style = MB_OK|MB_APPLMODAL
Title = "oooH input Error"
Text = "Where is Da serial DuDe ?"
hOwner = NULL
MessageBoxA

StringToAdd = "BytePtr [e!]"
ConcatString = "sdfsdBytePtr [e!]"
lstrcatA
CM2.00403380

ASCII "sdfsdBytePtr [e!]"
需要跟入:F7

String2 = ""
String1 = "sdfdf sdf"
lstrcmpA

rStule = MB_OKIMB APPLMODAL
```

# 加密算法逆向 - MD5 - 调试中比较

暂停

00401875 . C9 leave  
00401876 . C2 1000 retn 0x10  
00401879 . 60 pushad  
0040187A . BF 00794000 mov edi,CM2.004079D0  
0040187F . BE A8564000 mov esi,CM2.004056A8  
00401884 . 8B06 mov eax,dword ptr ds:[esi]  
00401886 . 8B5E 04 mov ebx,dword ptr ds:[esi+0x4]  
00401889 . 33C3 xor eax,ebx  
0040188B . 50 push eax  
0040188C . 68 5C334000 push CM2.0040335C  
00401891 . 57 push edi  
00401892 . E8 67000000 call <jmp.&USER32.wsprintfA>  
00401897 . 83C4 0C add esp,0xC  
0040189A . 83C7 08 add edi,0x8  
0040189D . 81F3 9900BD00 xor ebx,0xFBD0099  
004018A3 . 53 push ebx  
004018A4 . 68 5C334000 push CM2.0040335C  
004018A9 . 57 push edi  
004018AA . E8 4F000000 call <jmp.&USER32.wsprintfA>  
004018AF . 83C4 0C add esp,0xC  
004018B2 . 8B4E 08 mov ecx,dword ptr ds:[esi+0x8]  
004018B5 . 33CB xor ecx,ebx  
004018B7 . 83C7 08 add edi,0x8

ds:[004056A8]=912EC803  
eax=00000004

寄存器 (FPU)

EAX 00000004  
ECX 74C582EA kernel32.74C582EA  
EDX 00000000  
EBX 00000003  
ESP 0019F7BC  
EBP 0019F7E4  
ESI 004056A8 CM2.004056A8  
EDI 004079D0 CM2.004079D0  
EIP 00401884 CM2.00401884

C 0 ES 002B 32 0(FFFFFFFF)  
P 1 CS 0023 32 0(FFFFFFFF)  
A 0 SS 002B 32 0(FFFFFFFF)  
Z 0 DS 002B 32 0(FFFFFFFF)  
S 0 FS 0053 32 3FD000(FFF)  
T 0 GS 002B 32 0(FFFFFFFF)  
D 0  
0 0 LastErr ERROR\_SUCCESS (00000000)  
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)  
ST0 empty 0.0  
ST1 empty 0.0  
ST2 empty 0.0  
ST3 empty 0.0  
ST4 empty 1.00000000000000000000

地址 HEX 数据 ASCII

004056A8 03 C8 2E 91 E4 49 CE B2 8D 06 41 A5 70 B5 5A 49 ?或I尾?A 礪I  
004056B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0019F7BC 80006010  
0019F7C0 00350D88  
0019F7C4 0019F7E4

# 加密算法逆向 - MD5 - 注册码第一部分

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder CMD

暂停

```
00401B75 . C9 leave
00401B76 . C2 1000 retn 0x10
00401B79 $ 60 pushad
00401B7A . BF D0794000 mov edi,CM2.004079D0
00401B7F . BE A8564000 mov esi,CM2.004056A8
00401B84 . 8B06 mov eax,dword ptr ds:[esi]
00401B86 . 8B5E 04 mov ebx,dword ptr ds:[esi+0x4]
00401B89 . 33C3 xor eax,ebx
00401B8B . 50 push eax
00401B8C . 68 5C334000 push CM2.0040335C
00401B91 . 57 push edi
00401B92 . E8 67000000 call <jmp.&USER32.wsprintfA>
00401B97 . 83C4 0C add esp,0xC
00401B9A . 83C7 08 add edi,0x8
00401B9D . 81F3 9900BD00 xor ebx,0xFBD0099
00401BA3 . 53 push ebx
00401BA4 . 68 5C334000 push CM2.0040335C
00401BA9 . 57 push edi
00401BAA . E8 4F000000 call <jmp.&USER32.wsprintfA>
00401BAF . 83C4 0C add esp,0xC
00401BB2 . 8B4E 08 mov ecx,dword ptr ds:[esi+0x8]
00401BB5 . 33CB xor ecx,ebx
00401BB7 . 83C7 08 add edi,0x8
```

存放最后的结果  
取md5的值  
取第一组数据放eax  
取第二组数据放ebx  
进行xor运算,结果存eax

<%.8X> = 0x23E081E7  
Format = "%.8X"  
s = CM2.004079D0  
wsprintfA  
结果A作为注册码的第一部分

<%.8X> = 0xB2CE49E4  
Format = "%.8X"  
s = CM2.004079D0  
wsprintfA

eax=23E081E7

# 加密算法逆向 - MD5 - 注册码第二部分

00401B75	. C9	leave	<p>存放最后的结果 取md5的值 取第一组数据放eax 取第二组数据放ebx 进行xor运算, 结果存eax</p> <p>&lt;%.8X&gt; = 0x8 Format = "%.8X" s = CM2.004079D8 wsprintfA 结果A作为注册码的第一部分</p> <p>获得结果B &lt;%.8X&gt; = 0xBD73497D Format = "%.8X" s = CM2.004079D8 wsprintfA 注册码第二部分</p>
00401B76	. C2 1000	retn 0x10	
00401B79	\$ 60	pushad	
00401B7A	. BF D0794000	mov edi,CM2.004079D0	
00401B7F	. BE A8564000	mov esi,CM2.004056A8	
00401B84	. 8B06	mov eax,dword ptr ds:[esi]	
00401B86	. 8B5E 04	mov ebx,dword ptr ds:[esi+0x4]	
00401B89	. 33C3	xor eax,ebx	
00401B8B	. 50	push eax	
00401B8C	. 68 5C334000	push CM2.0040335C	
00401B91	. 57	push edi	<p>结果A作为注册码的第一部分</p> <p>获得结果B &lt;%.8X&gt; = 0xBD73497D Format = "%.8X" s = CM2.004079D8 wsprintfA 注册码第二部分</p>
00401B92	. E8 67000000	call <jmp.&USER32.wsprintfA>	
00401B97	. 83C4 0C	add esp,0xC	
00401B9A	. 83C7 08	add edi,0x8	
00401B9D	. 81F3 9900BD00	xor ebx,0xFBD0099	
00401BA3	. 53	push ebx	
00401BA4	. 68 5C334000	push CM2.0040335C	
00401BA9	. 57	push edi	
00401BAA	. E8 4F000000	call <jmp.&USER32.wsprintfA>	
00401BAF	. 83C4 0C	add esp,0xC	
00401BB2	. 8B4E 08	mov ecx,dword ptr ds:[esi+0x8]	<p>注册码第二部分</p>
00401BB5	. 33CB	xor ecx,ebx	
00401BB7	. 83C7 08	add edi,0x8	

ebx=BD73497D



# 加密算法逆向 - MD5 - 注册码第三、四部分

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> BP P VB Notepad Calc Folder C

暂停

```
00401BA4 . 68 5C334000 push CM2.0040335C
00401BA9 . 57          push edi
00401BAA . E8 4F000000 call <jmp.&USER32.wsprintfA>
00401BAF . 83C4 0C     add esp,0xC
00401BB2 . 8B4E 08     mov ecx,dword ptr ds:[esi+0x8]
00401BB5 . 33CB       xor ecx,ebx
00401BB7 . 83C7 08     add edi,0x8
00401BBA . 51         push ecx
00401BBB . 68 5C334000 push CM2.0040335C
00401BC0 . 57          push edi
00401BC1 . E8 38000000 call <jmp.&USER32.wsprintfA>
00401BC6 . 83C4 0C     add esp,0xC
00401BC9 . 81F3 090A0C09 xor ebx,0xB0C0A09
00401BCF . 8B56 0C     mov edx,dword ptr ds:[esi+0xC]
00401BD2 . 83C7 08     add edi,0x8
00401BD5 . 52         push edx
00401BD6 . 68 5C334000 push CM2.0040335C
00401BDB . 57          push edi
00401BDC . E8 1D000000 call <jmp.&USER32.wsprintfA>
00401BE1 . 83C4 0C     add esp,0xC
00401BE4 . 61         popad
00401BE5 . C3         retn
00401BE6 .- FF25 0C204000 jmp dword ptr ds:[<&KERNEL32.ExitProcess> kernel32.ExitProcess]

Format = "%.8X"
s = CM2.004079E8
wsprintfA
注册码第二部分B

<%.8X> = 0xFCFE5ADF
Format = "%.8X"
s = CM2.004079E8
wsprintfA
第三部分C

<%.8X> = 0x4079EB
Format = "%.8X"
s = CM2.004079E8
wsprintfA
第四部分D

esp=0019F7B0, (ASCII E8,"y@")
```



# 加密算法逆向 - 最后进行比较

00401B7A . BF D0794000 mov edi,CM2.004079D0

存放最后的结果

```
00401B07 . 58          pop eax
00401B08 . 68 A8564000 push CM2.004056A8
00401B0D . 50          push eax
00401B0E . 68 80334000 push CM2.00403380
00401B13 . E8 E8F4FFFF call CM2.00401000
00401B18 . E8 5C000000 call CM2.00401B79
00401B1D . 68 D0794000 push CM2.004079D0
00401B22 . 68 F89C4000 push CM2.00409CF8
00401B27 . E8 CC000000 call <jmp.&KERNEL32.lstrcmpA>
00401B2C . 85C0        test eax,eax
00401B2E . 75 15       jnz short CM2.00401B45
00401B30 . 6A 00       push 0x0
00401B32 . 68 20324000 push CM2.00403220
00401B37 . 68 2A324000 push CM2.0040322A
00401B3C . 6A 00       push 0x0
00401B3E . E8 D9000000 call <jmp.&USER32.MessageBoxA>
00401B43 . EB 2E       jmp short CM2.00401B73
00401B45 > 6A 00       push 0x0
00401B47 . 68 AD324000 push CM2.004032AD
00401B4C . 68 B9324000 push CM2.004032B9
00401B51 . 6A 00       push 0x0
00401B53 . E8 C4000000 call <jmp.&USER32.MessageBoxA>
00401B58 . 33C0        xor eax,eax
```

计算结果  
输入的序列号

# 感谢大家！



南京邮电大学  
Nanjing University of Posts and Telecommunications