**电子科技大学信息与软件工程学院**

**标 准 实 验 报 告**

**（实验）课程名称：软件安全**

**电子科技大学教务处制表**

**学生姓名：陈奎 学 号：2017221303023 指导教师： 钱伟中**

**实验地点：信软学院**  **实验时间：19.10.30**

**一、实验室名称：信软学院实验室**

**二、实验项目名称： 逆向分析**

**三、实验学时：4学时**

**四、实验原理：**

**五、实验目的：**

1）理解软件逆向的概念；

2）掌握逆向工具：Ollydbg和IDA pro；

3）掌握逆向分析软件的方法；

4）掌握对可执行文件逆向到高级语言的方法

**六、实验内容：**

现在得到一个可执行EXE执行程序，反汇编该程序得到的信息展示如下:

要求: 写出其对应的高级语言

附件1为逆向EXE的反汇编代码。

**七、实验器材（设备、元器件）：**

PC机

**八、实验步骤：**

1） 阅读反汇编程序，分析程序模块，整理成汇编程序

2） 编译程序，输出EXE文件

3） 用Ollydbg加载EXE文件，调试EXE

4) 用IDA获得C伪代码

5） 分析调整第4步获得的C伪代码

6） 用Visual studio，将第5步获得的代码输入进行测试验证。

**九、实验数据及结果分析：**

1. **阅读反汇编程序得到汇编程序**

汇编代码

|  |
| --- |
| .386  .model flat,stdcall  include msvcrt.inc  includelib msvcrt.lib  include user32.inc  includelib user32.lib  include kernel32.inc  includelib kernel32.lib  .data  s1 db 'this is a simple ',0  s2 db 'program you are annalyzing now!',0  s3 db "hello dou have get the right result?no! it't not over\n",0  s4 db "this is true end! but you should not relax yourself!!!Be careful ",0  s5 db "trap congratulation! now is the end\n",0  s6 db "their sum is %d\n",0  s7 db "can you have ",0  s8 db "the ability of ",0  s9 db "reverse analysis\n",0  .code  start:  push ebp  mov ebp,esp  sub esp,0D8h  push ebx  push esi  push edi  lea edi,[ebp+0FFFFFF28h]  mov ecx,36h  mov eax,0CCCCCCCCh  rep stos dword ptr es:[edi]      invoke crt\_printf,offset s1  invoke crt\_printf,offset s2  mov dword ptr [ebp-8],39h  mov dword ptr [ebp-14h],3Ch  mov eax,dword ptr [ebp-8]  cmp eax,dword ptr [ebp-14h]  jle r  invoke crt\_printf,offset s3  jmp s  r:  mov eax,dword ptr [ebp-14h]  push eax  mov ecx,dword ptr [ebp-8]  push ecx  call compare  add esp,8  s:  mov dword ptr [ebp-14h],32h  mov eax,dword ptr [ebp-8]  cmp eax,dword ptr [ebp-14h]  jle t  mov eax,dword ptr [ebp-14h]  push eax  mov ecx,dword ptr [ebp-8]  push ecx  call compare  add esp,8  t:  invoke crt\_printf,offset s4  invoke crt\_printf,offset s5  xor eax,eax  pop edi  pop esi  pop ebx  add esp,0D8h  pop ebp  ret    compare:  push ebp  mov ebp,esp  sub esp,0CCh  push ebx  push esi  push edi  lea edi,[ebp+0FFFFFF34h]  mov ecx,33h  mov eax,0CCCCCCCCh  rep stos dword ptr es:[edi]  mov eax,dword ptr [ebp+8]  cmp eax,dword ptr [ebp+0Ch]  jl u  mov eax,dword ptr [ebp+8]  sub eax,dword ptr [ebp+0Ch]  mov dword ptr [ebp-8],eax  jmp v  u:  mov eax,dword ptr [ebp+0Ch]  sub eax,dword ptr [ebp+8]  mov dword ptr [ebp-8],eax  v:  mov esi,esp  mov eax,dword ptr [ebp-8]  push eax  invoke crt\_printf,offset s6,eax  invoke crt\_printf,offset s7  invoke crt\_printf,offset s8  invoke crt\_printf,offset s9  mov eax,dword ptr [ebp-8]  pop edi  pop esi  pop ebx  add esp,0CCh  mov esp,ebp  pop ebp  ret  end start  end start |

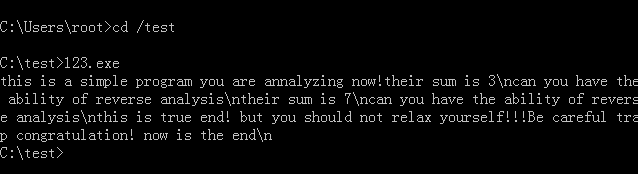
1. **编译并运行代码**



生成文件

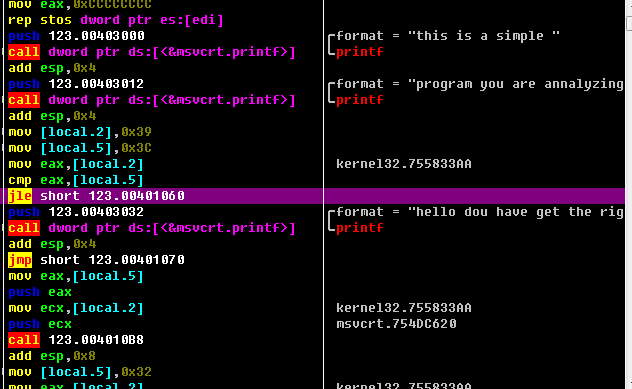


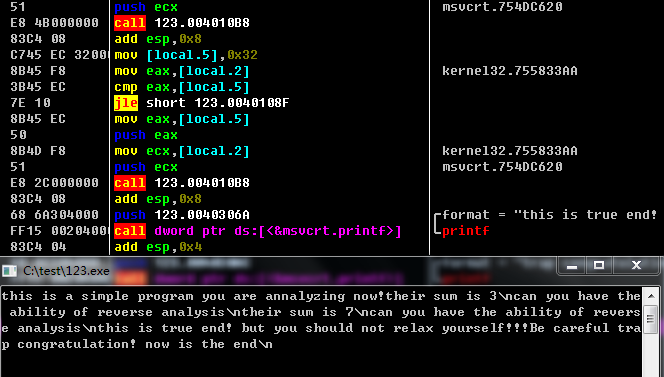
控制台执行代码



执行正确但是无法打印出换行符号

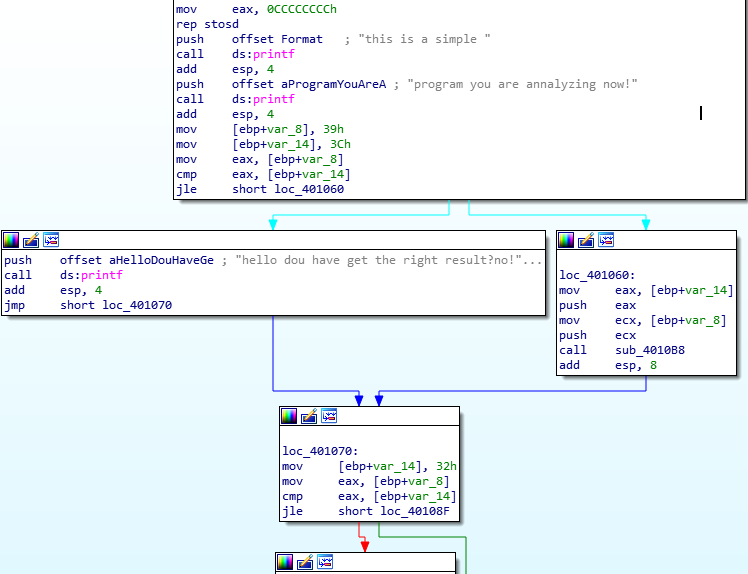
1. **Ollydbg加载EXE文件，调试EXE**





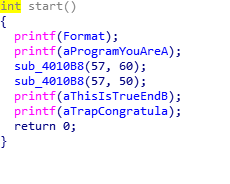
1. **用IDA获得C伪代码**

分析代码逻辑结构

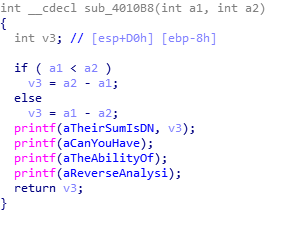


获得伪C代码

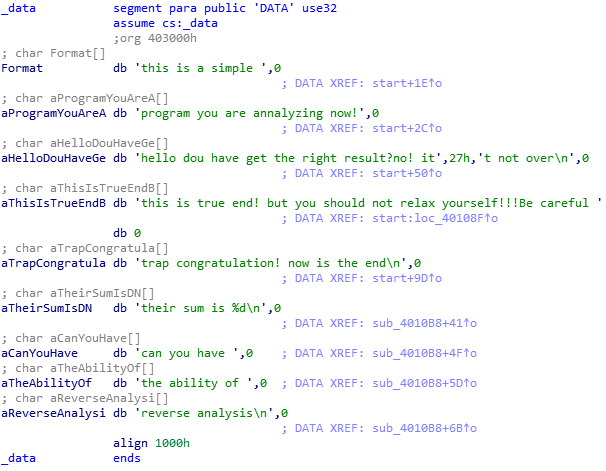
主程序：



子函数：



数据字段：

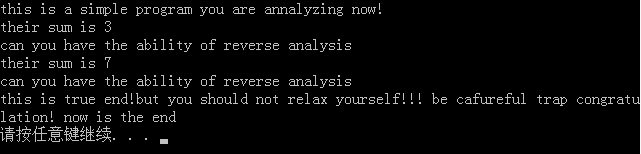


1. **调整为代码并用vs输入代码进行对比**

C源代码

|  |
| --- |
| #include "stdafx.h"  #include "stdio.h"  int compare(int a, int b) {  int c;  if (a < b) c = b - a;  else c = a - b;  printf("their sum is %d\n", c);  printf("can you have the ability of reverse analysis\n");  return c;  }  int main()  {  printf("this is a simple ");  printf("program you are annalyzing now!\n");  int a = 57;  int b = 60;  if (a <= b) {  a = 60;  b = 57;  compare(a, b);  }  else printf("hello do you have get the right result?no!it's not over\n");  a = 50;  b = 57;  if (a < b) {  compare(a, b);  }  printf("this is true end!but you should not relax yourself!!! be cafureful ");  printf("trap congratulation! now is the end\n");  return 0;  } |

执行结果



成功实现逆向

**十、实验结论：**

将反汇编代码转换成汇编代码，再将汇编代码编译连接为exe文件，通过执行exe文件得到程序的结果。然后通过Ollydbg查看具体执行步骤，再通过IDA得到伪C代码以及代码结构，最后通过visual studio成功实现逆向。

**十一、总结及心得体会：**

本次实验难度较大。一是对win32汇编的不熟悉，在编写汇编代码时需要上网查找；二是对逆向工具的不熟悉，需要熟悉Ollydbg以及IDA。另外实验的时候无法用汇编出来的exe文件打印换行。除此之外无较大bug。

通过本次实验加深了逆向的理解以及对汇编的运用。

**十二、对本实验过程及方法、手段的改进建议：**

**无**

# 附件1 实验反汇编源代码：

00411490 push ebp //

00411491 mov ebp,esp // 形成自己的栈坐标

00411493 sub esp,0D8h //

00411499 push ebx //---------------------------

0041149A push esi // 寄存器保护

0041149B push edi //

0041149C lea edi,[ebp+FFFFFF28h] //-------------------------

004114A2 mov ecx,36h //

004114A7 mov eax,0CCCCCCCCh // 初始化临时变量区

004114AC rep stos dword ptr es:[edi] //-------------------------

004114AE mov esi,esp

004114B0 push 415AB0h

004114B5 call dword ptr ds:[004182BCh] // printf函数

004114BB add esp,4 // 堆栈平衡调整

004114BE cmp esi,esp

004114C0 call 00411145 //安全机制函数，逆向忽略

004114C5 mov esi,esp

004114C7 push 4157ACh

004114CC call dword ptr ds:[004182BCh] // printf函数

004114D2 add esp,4

004114D5 cmp esi,esp //

004114D7 call 00411145 //安全机制函数，逆向忽略

004114DC mov dword ptr [ebp-8],39h

004114E3 mov dword ptr [ebp-14h],3Ch

004114EA mov eax,dword ptr [ebp-8]

004114ED cmp eax,dword ptr [ebp-14h]

004114F0 jle 0041150B

004114F2 mov esi,esp

004114F4 push 4162E4h

004114F9 call dword ptr ds:[004182BCh] // printf函数

004114FF add esp,4

00411502 cmp esi,esp

00411504 call 00411145 //安全机制函数，逆向忽略

00411509 jmp 0041151B

0041150B mov eax,dword ptr [ebp-14h]

0041150E push eax

0041150F mov ecx,dword ptr [ebp-8]

00411512 push ecx

00411513 call 00411168

00411518 add esp,8

0041151B mov dword ptr [ebp-14h],32h

00411522 mov eax,dword ptr [ebp-8]

00411525 cmp eax,dword ptr [ebp-14h]

00411528 jle 0041153A

0041152A mov eax,dword ptr [ebp-14h]

0041152D push eax

0041152E mov ecx,dword ptr [ebp-8]

00411531 push ecx

00411532 call 00411168

00411537 add esp,8

0041153A mov esi,esp

0041153C push 4162A0h

00411541 call dword ptr ds:[004182BCh] // printf函数

00411547 add esp,4

0041154A cmp esi,esp

0041154C call 00411145 //安全机制函数，逆向忽略

00411551 mov esi,esp

00411553 push 415BC8h

00411558 call dword ptr ds:[004182BCh] // printf函数

0041155E add esp,4

00411561 cmp esi,esp

00411563 call 00411145 //安全机制函数，逆向忽略

00411568 xor eax,eax // 程序状态返回置0

0041156A pop edi // return 0

0041156B pop esi // 程序收尾返回

0041156C pop ebx //

0041156D add esp,0D8h //

00411573 cmp ebp,esp //

00411575 call 00411145 //安全机制函数，逆向忽略

0041157A mov esp,ebp //

0041157C pop ebp

0041157D ret

00411168 jmp 004113B0

004113B0 push ebp //

004113B1 mov ebp,esp // 形成自己的栈坐标

004113B3 sub esp,0CCh //-----------------------------

004113B9 push ebx //

004113BA push esi // 寄存器保护

004113BB push edi //----------------------------

004113BC lea edi,[ebp+FFFFFF34h] //

004113C2 mov ecx,33h //

004113C7 mov eax,0CCCCCCCCh // 初始化临时变量区

004113CC rep stos dword ptr es:[edi] //---------------------------

004113CE mov eax,dword ptr [ebp+8]

004113D1 cmp eax,dword ptr [ebp+0Ch]

004113D4 jl 004113E1

004113D6 mov eax,dword ptr [ebp+8]

004113D9 sub eax,dword ptr [ebp+0Ch]

004113DC mov dword ptr [ebp-8],eax

004113DF jmp 004113EA

004113E1 mov eax,dword ptr [ebp+0Ch]

004113E4 sub eax,dword ptr [ebp+8]

004113E7 mov dword ptr [ebp-8],eax

004113EA mov esi,esp

004113EC mov eax,dword ptr [ebp-8]

004113EF push eax

004113F0 push 41576Ch

004113F5 call dword ptr ds:[004182BCh] // printf函数

004113FB add esp,8

004113FE cmp esi,esp

00411400 call 00411145 //安全机制函数，逆向忽略

00411405 mov esi,esp

00411407 push 41575Ch

0041140C call dword ptr ds:[004182BCh] // printf函数

00411412 add esp,4

00411415 cmp esi,esp

00411417 call 00411145 //安全机制函数，逆向忽略

0041141C mov esi,esp

0041141E push 415748h

00411423 call dword ptr ds:[004182BCh] // printf函数

00411429 add esp,4

0041142C cmp esi,esp

0041142E call 00411145 // 安全机制函数，逆向忽略

00411433 mov esi,esp

00411435 push 415858h

0041143A call dword ptr ds:[004182BCh] // printf函数

00411440 add esp,4

00411443 cmp esi,esp

00411445 call 00411145 // 安全机制函数，逆向忽略

0041144A mov eax,dword ptr [ebp-8] // 返回值保存传递返回

0041144D pop edi //

0041144E pop esi //

0041144F pop ebx // 程序收尾返回

00411450 add esp,0CCh //

00411456 cmp ebp,esp //

00411458 call 00411145 // 安全机制函数，逆向忽略

0041145D mov esp,ebp //

0041145F pop ebp //

00411460 ret //

**内存数据区**：

0x00415740 00 00 00 00 00 00 00 00 74 68 65 20 61 62 69 6c ........the abil

0x00415750 69 74 79 20 6f 66 20 00 00 00 00 00 63 61 6e 20 ity of .....can

0x00415760 79 6f 75 20 68 61 76 65 20 00 00 00 74 68 65 69 you have ...thei

0x00415770 72 20 73 75 6d 20 69 73 20 a3 ba 25 64 0a 00 00 r sum is ..%d...

0x00415780 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0x00415790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0x004157A0 00 00 00 00 00 00 00 00 00 00 00 00 70 72 6f 67 ............prog

0x004157B0 72 61 6d 20 79 6f 75 20 61 72 65 20 61 6e 61 6c ram you are anal

0x004157C0 79 7a 69 6e 67 20 6e 6f 77 21 00 00 00 00 00 00 yzing now!......

0x004157D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0x00415840 72 00 63 00 5c 00 63 00 72 00 74 00 65 00 78 00 r.c.\.c.r.t.e.x.

0x00415850 65 00 2e 00 63 00 00 00 72 65 76 65 72 73 65 20 e...c...reverse

0x00415860 61 6e 61 6c 79 73 69 73 00 00 00 00 00 00 00 00 analysis........

0x00415870 5f 00 5f 00 6e 00 61 00 74 00 69 00 76 00 65 00 \_.\_.n.a.t.i.v.e.

0x00415B70 67 20 63 6f 6e 76 65 6e 74 69 6f 6e 20 77 69 74 g convention wit

0x00415B80 68 20 61 20 66 75 6e 63 74 69 6f 6e 20 70 6f 69 h a function poi

0x00415B90 6e 74 65 72 20 64 65 63 6c 61 72 65 64 20 77 69 nter declared wi

0x00415BA0 74 68 20 61 20 64 69 66 66 65 72 65 6e 74 20 63 th a different c

0x00415BB0 61 6c 6c 69 6e 67 20 63 6f 6e 76 65 6e 74 69 6f alling conventio

0x00415BC0 6e 2e 0a 0d 00 00 00 00 74 72 61 70 a3 ac 63 6f n.......trap..co

0x00415BD0 6e 67 72 61 74 75 6c 61 74 69 6f 6e 21 20 6e 6f ngratulation! no

0x00415BE0 77 20 69 73 20 74 68 65 20 65 6e 64 0a 00 00 00 w is the end....

0x00415BF0 00 00 00 00 e8 5a 41 00 90 59 41 00 68 59 41 00 .....ZA..YA.hYA.

0x00415C00 28 59 41 00 f4 58 41 00 d0 58 41 00 01 00 00 00 (YA..XA..XA.....

0x00415C10 00 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 ................

0x00415C20 01 00 00 00 53 74 61 63 6b 20 61 72 6f 75 6e 64 ....Stack around

0x00415C30 20 74 68 65 20 76 61 72 69 61 62 6c 65 20 27 00 the variable '.

0x00416280 41 00 00 00 41 44 56 41 50 49 33 32 2e 44 4c 4c A...ADVAPI32.DLL

0x00416290 00 00 00 00 98 71 41 00 f0 71 41 00 00 00 00 00 .....qA..qA.....

0x004162A0 74 68 69 73 20 69 73 20 74 72 75 65 20 65 6e 64 this is true end

0x004162B0 21 20 62 75 74 20 79 6f 75 20 73 68 6f 75 6c 64 ! but you should

0x004162C0 20 6e 6f 74 20 72 65 6c 61 78 20 79 6f 75 72 73 not relax yours

0x004162D0 65 6c 66 21 21 21 20 42 65 20 63 61 72 65 66 75 elf!!! Be carefu

0x004162E0 6c 20 00 00 68 65 6c 6c 6f a3 ac 64 6f 20 79 6f l ..hello..do yo

0x004162F0 75 20 68 61 76 65 20 67 65 74 20 74 68 65 20 72 u have get the r

0x00416300 69 67 68 74 20 72 65 73 75 6c 74 3f 20 6e 6f 21 ight result? no!

0x00416310 20 69 74 27 73 20 6e 6f 74 20 6f 76 65 72 0a 00 it's not over..

0x00415A50 43 68 61 6e 67 69 6e 67 20 74 68 65 20 63 6f 64 Changing the cod

0x00415A60 65 20 69 6e 20 74 68 69 73 20 77 61 79 20 77 69 e in this way wi

0x00415A70 6c 6c 20 6e 6f 74 20 61 66 66 65 63 74 20 74 68 ll not affect th

0x00415A80 65 20 71 75 61 6c 69 74 79 20 6f 66 20 74 68 65 e quality of the

0x00415A90 20 72 65 73 75 6c 74 69 6e 67 20 6f 70 74 69 6d resulting optim

0x00415AA0 69 7a 65 64 20 63 6f 64 65 2e 0a 0d 00 00 00 00 ized code.......

0x00415AB0 74 68 69 73 20 69 73 20 20 61 20 73 69 6d 70 6c this is a simpl

0x00415AC0 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e...............

0x00415AD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0x00415BA0 74 68 20 61 20 64 69 66 66 65 72 65 6e 74 20 63 th a different c

0x00415BB0 61 6c 6c 69 6e 67 20 63 6f 6e 76 65 6e 74 69 6f alling conventio

0x00415BC0 6e 2e 0a 0d 00 00 00 00 74 72 61 70 a3 ac 63 6f n.......trap..co

0x00415BD0 6e 67 72 61 74 75 6c 61 74 69 6f 6e 21 20 6e 6f ngratulation! no

0x00415BE0 77 20 69 73 20 74 68 65 20 65 6e 64 0a 00 00 00 w is the end....

0x00415BF0 00 00 00 00 e8 5a 41 00 90 59 41 00 68 59 41 00 .....ZA..YA.hYA.

0x00415C00 28 59 41 00 f4 58 41 00 d0 58 41 00 01 00 00 00 (YA..XA..XA.....

**报告评分：**

**指导教师签字：**