

电子科技大学信息与软件工程学院

实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 软件安全

理论教师 王 静

实验教师 杨 珊

电子科技大学

实验报告

学生姓名：袁昊男 学号：2018091618008 指导教师：王静

实验地点：信软楼 306 实验时间：2020.12.11

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：逆向分析

三、实验学时：4 学时

四、实验原理：

1、软件逆向概念

将可执行程序反汇编，通过分析反汇编代码来理解代码功能，然后用高级语言重新描述这段代码，逆向分析原始软件的思路，这个过程就称作逆向工程（Reverse Engineering），有时也简单地称作逆向（Reversing）。

2、软件逆向原理

利用逆向工程技术，从可运行的程序系统出发，运用解密、反汇编、系统分析、程序理解等多种计算机技术，对软件的结构、流程、算法、代码等进行逆向拆解和分析，推导出软件产品的源代码、设计原理、结构、算法、处理过程、运行方法及相关文档等。随着用户需求的复杂度越来越高软件开发的难度也在不断地上升快速高效的软件开发已成为项目成败的关键之一。为了提高程序员的产品率开发工具的选择尤为重要因为开发工具的自动化程度可以大大减少程序员繁琐重复的工作使其集中关注他所面临的特定领域的问题。为此当前的 IDE 不可避免地要向用户隐藏着大量的操作细节而这些细节包含了大量的有价值的技术。

3、逆向工具使用方法简介

在对软件进行逆向工程时，不可避免地需要用到多种工具，工具的合理使用，可以加快调试速度，提高逆向工程的效率。对于逆向工程的调试环节来说，没有动态调试器将使用的调试工作很难进行。可以看出，各种有效的工具在逆向工程中占据着相当重要的地位，有必要对它们的用法做一探讨。

IDA Pro 简介：IDA Pro(简称 IDA)是 DataRescue 公司(www.datarescue.com)出品的一款交互式反汇编工具，它功能强大、操作复杂，要完全掌握它，需要很

多知识。IDA 最主要的特性是交互和多处理器。操作者可以通过对 IDA 的交互来指导 IDA 更好地反汇编，IDA 并不自动解决程序中的问题，但它会按用户的指令找到可疑之处，用户的工作是通知 IDA 怎样去做。比如人工指定编译器类型，对变量名、结构定义、数组等定义等。这样的交互能力在反汇编大型软件时显得尤为重要。多处理器特点是指 IDA 支持常见处理器平台上的软件产品。IDA 支持的文件类型非常丰富，除了常见的 PE 格式，还支持 Windows, DOS, UNIX, Mac, Java, .NET 等平台的文件格式。

当我们打开之后，IDA 会提供 3 种不同的打开方式：New（新建），Go（运行），Previous（上一个）。当我们初次打开的时候选择 GO 就可以了。进入之后，选择左上角的 file 中的 open 打开文件，或者也可以直接将文件拖入窗口中。

Ollydbg 简介：OLLYDBG 是一个新的动态追踪工具，将 IDA 与 SoftICE 结合起来的理想，Ring 3 级调试器，非常容易上手，已代替 SoftICE 成为当今最为流行的调试解密工具了。同时还支持插件扩展功能，是目前最强大的调试工具之一。

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 机一台。

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、汇编程序代码

```
1.  .386
2.  ;告诉汇编器应该生成 386 处理器（或更高）的伪代码。
3.
4.  .model flat,stdcall
5.  ;使用平坦内存模式并使用 stdcall 调用习惯,函数的参数从右往左压入
6.
7.  include msvcrt.inc
8.  includelib msvcrt.lib
9.  include user32.inc
10.includelib user32.lib
11.include kernel32.inc
12.includelib kernel32.lib
13.;为了使用来自 WindowsAPI 的函数, 需要导入 dll 和包含文件
14.
15..data    ;数据段
16.s1 db 'this is a simple ',0
17.s2 db 'program you are annalyzing now!',0
18.s3 db "hello dou have get the right result?no! it't not over\n",0
19.s4 db "this is true end! but you should not relax yourself!!!Be careful ",0
20.s5 db "trap congratulation! now is the end\n",0
21.s6 db "their sum is %d\n",0
22.s7 db "can you have ",0
23.s8 db "the ability of ",0
24.s9 db "reverse analysis\n",0
25.
26..code    ;代码段
27.start:
28.    push ebp
29.    mov ebp,esp
30.    sub esp,0D8h    ;为当前栈预留本地变量空间
31.    push ebx        ;保护寄存器
32.    push esi        ;保护寄存器
33.    push edi        ;保护寄存器
34.    lea edi,[ebp+0FFFFFF28h]
35.    mov ecx,36h
36.    mov eax,0CCCCCCCCh
37.    rep stos dword ptr es:[edi]    ;初始化
```

```

38.
39.
40.     invoke crt_printf,offset s1
41.     invoke crt_printf,offset s2
42.     mov dword ptr [ebp-8],39h
43.     mov dword ptr [ebp-14h],3Ch ;初始化本地变量
44.     mov eax,dword ptr [ebp-8]
45.     cmp eax,dword ptr [ebp-14h] ;比较
46.     jle fun1
47.     invoke crt_printf,offset s3
48.     jmp fun2
49.     fun1:
50.         mov eax,dword ptr [ebp-14h]
51.         push eax      ;将参数压栈
52.         mov ecx,dword ptr [ebp-8]
53.         push ecx      ;将参数压栈
54.         call compare
55.         add esp,8      ;平衡栈
56.     fun2:
57.         mov dword ptr [ebp-14h],32h
58.         mov eax,dword ptr [ebp-8]
59.         cmp eax,dword ptr [ebp-14h]
60.         jle fun3
61.         mov eax,dword ptr [ebp-14h]
62.         push eax
63.         mov ecx,dword ptr [ebp-8]
64.         push ecx      ;将参数压栈
65.         call compare
66.         add esp,8      ;平衡栈
67.     fun3:
68.         invoke crt_printf,offset s4
69.         invoke crt_printf,offset s5
70.         xor eax,eax
71.         pop edi
72.         pop esi
73.         pop ebx
74.         add esp,0D8h
75.         pop ebp      ;恢复上一栈栈底
76.         ret
77.
78.     compare:
79.         push ebp
80.         mov ebp,esp
81.         sub esp,0CCh

```

```

82.      push ebx
83.      push esi
84.      push edi
85.      lea edi,[ebp+0FFFFFF34h]
86.      mov ecx,33h
87.      mov eax,0CCCCCCCCh
88.      rep stos dword ptr es:[edi]
89.      mov eax,dword ptr [ebp+8]    ;参数 a
90.      cmp eax,dword ptr [ebp+0Ch] ;参数 b
91.      jnl fun4
92.      mov eax,dword ptr [ebp+8]
93.      sub eax,dword ptr [ebp+0Ch]
94.      mov dword ptr [ebp-8],eax
95.      jmp fun5
96.  fun4:
97.      mov eax,dword ptr [ebp+0Ch]
98.      sub eax,dword ptr [ebp+8]
99.      mov dword ptr [ebp-8],eax
100. fun5:
101.      mov esi,esp
102.      mov eax,dword ptr [ebp-8]
103.      push eax
104.      invoke crt_printf,offset s6,eax
105.      invoke crt_printf,offset s7
106.      invoke crt_printf,offset s8
107.      invoke crt_printf,offset s9
108.      mov eax,dword ptr [ebp-8]
109.      pop edi
110.      pop esi
111.      pop ebx
112.      add esp,0CCh
113.      mov esp,ebp
114.      pop ebp
115.      ret
116.
117. end start

```

2、编译程序，输出 EXE 文件

(1) 编译、链接

```

Microsoft Visual Studio 调试控制台

Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997. All rights reserved.

Assembling: C:\expl.asm
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
驱动器 C 中的卷没有标签。
卷的序号是DA33-6185

C:      est 的目录

2020/12/11  21:06                2,551 expl.asm
2020/12/11  21:10                2,560 expl.exe
2020/12/11  21:10                1,382 expl.obj
                3 个文件          6,493 字节
                0 个文件 38,179,246,080 可用字节

```

(2) 输出 EXE 文件，执行 EXE

```

C:\Windows\system32\cmd.exe

Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation. 保留所有权利。

C:\Users\90389>C:\Users\90389\Desktop\test\Debug\expl.exe
this is a simple program you are analyzing now!
their sum is 3
can you have the ability of reverse analysis
their sum is 7
can you have the ability of reverse analysis
this is true end!but you should not relax yourself!!! be careful trap
congratulation! now is the end

```

3、用 Ollydbg 加载 EXE 文件并调试

The screenshot displays the Ollydbg interface with the following components:

- Assembly Window:** Shows assembly instructions with their addresses and hex values. Key instructions include:
 - 00401467: MOV EBP, ESP
 - 00401468: AND ESP, FFFFFFFF
 - 00401469: SUB ESP, 20
 - 00401470: CALL 00401A20
 - 00401475: MOV DWORD PTR SS:[LOCAL.8], OFFSET 00405081
 - 0040147C: CALL JUMP.Bnsuvert.printf
 - 00401481: MOV DWORD PTR SS:[LOCAL.8], OFFSET 00405092
 - 00401488: CALL JUMP.Bnsuvert.printf
 - 0040148D: MOV DWORD PTR SS:[LOCAL.1], 39
 - 00401495: MOV DWORD PTR SS:[LOCAL.2], 3C
 - 0040149D: MOV EAX, DWORD PTR SS:[LOCAL.1]
 - 004014A1: CMP EAX, DWORD PTR SS:[LOCAL.2]
 - 004014A5: JC SHORT 004014BD
 - 004014A7: MOV EAX, DWORD PTR SS:[LOCAL.2]
 - 004014AB: MOV DWORD PTR SS:[LOCAL.7], EAX
 - 004014AF: MOV EAX, DWORD PTR SS:[LOCAL.1]
- Registers Window:** Shows the state of CPU registers. EAX is 00000001, ECX is 00401990, EDI is 00000000, and EIP is 00401467.
- Stack Window:** Shows the stack memory layout. The current instruction address is 00401467.
- Command Window:** Shows the current instruction address: 00401467.

4、用 IDA 获得 C 伪代码

(1) 主函数

```
1. unsigned int start()
2. {
3.     char v1; // [esp+Ch] [ebp-D8h]
4.     int v2; // [esp+D0h] [ebp-14h]
5.     unsigned int v3; // [esp+DCh] [ebp-8h]
6.
7.     memset32(&v1, 214748364, 0x36u);
8.     sub_401278(aThisIsASimple);
9.     sub_401278(asc_40301C);
10.    sub_401278(aProgramYouAreA);
11.    sub_401278(asc_403040);
12.    v3 = 57;
13.    v2 = 60;
14.    sub_4010D5(57, 60);
15.    v2 = 50;
16.    if ( v3 > 0x32 )
17.        sub_4010D5(v3, v2);
18.    sub_401278(aThisIsTrueEndB);
19.    sub_401278(asc_403090);
20.    sub_401278(aTrap);
21.    sub_401278(asc_40309C);
22.    return v3;
23. }
```

(2) 子函数

```
1. int __cdecl sub_4010D5(unsigned int a1, unsigned int a2)
2. {
3.     char v3; // [esp+Ch] [ebp-CCh]
4.     int v4; // [esp+D0h] [ebp-8h]
5.
6.     memset32(&v3, 214748364, 0x33u);
7.     dword_403000 = a1;
8.     sub_401278(aTheValueOfAIs);
9.     sub_401210(dword_403000, &unk_4030B4);
10.    sub_401278(&unk_4030B4);
11.    sub_401278(asc_4030C8);
12.    dword_403004 = a2;
13.    sub_401278(aTheValueOfBIs);
14.    sub_401210(dword_403004, &unk_4030E0);
15.    sub_401278(&unk_4030E0);
16.    sub_401278(asc_4030F4);
17.    if ( a1 >= a2 )
```



```

18.     v4 = a1 - a2;
19.     else
20.         v4 = a2 - a1;
21.     sub_401278(aTheirSumIs);
22.     sub_401210(v4, &unk_403108);
23.     sub_401278(&unk_403108);
24.     sub_401278(asc_40311C);
25.     sub_401278(aCanYouHave);
26.     sub_401278(aTheAbilityOf);
27.     sub_401278(aReverseAnalysi);
28.     sub_401278(asc_403154);
29.     return v4;
30. }

```

5、调整后的 C 语言代码

```

1. #include "stdafx.h"
2. #include "stdio.h"
3.
4. int compare(int a, int b) {
5.     int c;
6.     if (a < b) c = b - a;
7.     else c = a - b;
8.     printf("their sum is %d\n", c);
9.     printf("can you have the ability of reverse analysis\n");
10.    return c;
11. }
12.
13. int main()
14. {
15.     printf("this is a simple ");
16.     printf("program you are annalyzing now!\n");
17.     int a = 57;
18.     int b = 60;
19.     if (a <= b) {
20.         a = 60;
21.         b = 57;
22.         compare(a, b);
23.     }
24.     else printf("hello do you have get the right result?no!it'
s not over\n");
25.
26.     a = 50;
27.     b = 57;

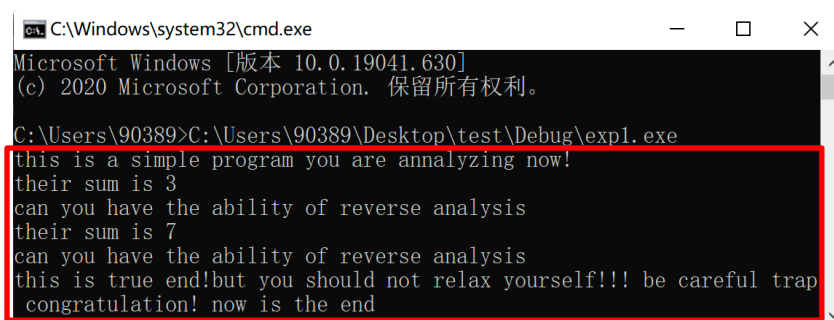
```

```

28.     if (a < b) {
29.         compare(a, b);
30.     }
31.     printf("this is true end!but you should not relax yourself
        !!! be careful ");
32.     printf("trap congratulation! now is the end\n");
33.     return 0;
34. }

```

6、用 Visual studio 将第 5 步获得的代码输入进行测试验证



```

C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation. 保留所有权利。

C:\Users\90389>C:\Users\90389\Desktop\test\Debug\expl.exe
this is a simple program you are analyzing now!
their sum is 3
can you have the ability of reverse analysis
their sum is 7
can you have the ability of reverse analysis
this is true end!but you should not relax yourself!!! be careful trap
congratulation! now is the end

```

结论：与汇编得到的程序结果一致。

十、实验结论

从给出的反汇编源代码逆向分析出程序执行过程，并编写汇编代码进行编译、链接、执行；使用 IDA 分析汇编代码生成的 EXE 文件，生成 C 语言伪代码；整理伪代码，得到清晰的 C 语言高级程序语言代码，并使用 Visual Studio 对其进行编译，运行结果与之前汇编语言生成 EXE 程序运行结果一致，说明逆向分析与代码编写正确无误。

十一、总结及心得体会

- 1、在本质上，汇编代码和高级语言代码的逻辑是相同的，都是由二进制指令与数据构成的，且高级语言在编译成汇编语言时会优化代码，如采用更快的参数调用方式等。
- 2、汇编语言对程序的调用是通过切换函数栈帧进行的，一般有将参数压栈、设置新栈底、预留函数中本地变量空间、保护寄存器、恢复栈帧、平衡栈等操作，实现函数与函数之间的递归调用及参数传递。逆向工程最重要的部分就是弄清楚函数之间的相互调用及参数传递。
- 3、掌握汇编语言非常重要，不仅可以进行逆行分析程序，而且提升对高级语言底层内核的理解。经过实际分析反汇编代码与编写汇编代码，了解了寄存器的锁机制：将内容写入寄存器后，一旦寄存器使用一次就会被

释放，所以可能出现前后的 `eax` 值完全不同的现象，这也是汇编语言和高级语言不同的地方。

十二、对本实验过程及方法、手段的改进建议

本实验设计与教材结合紧密、具有较大难度。通过分析反汇编代码、逆向分析出源程序代码的汇编代码与高级程序语言代码，并重新编译、运行后得到了验证。本实验强化了学生对软件逆向原理的理解，通过对 IDA 及 Ollydbg 的使用，掌握了软件逆向的工具，此外，还使学生熟悉汇编语言编译环境，对软件安全的深入学习打下了坚实的基础。

报告评分：

指导教师签字：