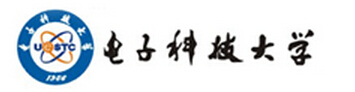
****

**逆向工程课程报告**

**课程名称：** 逆向工程实践

**报告题目：** Ollydbg破解TraceMe程序

**课程教师：** 何兴高

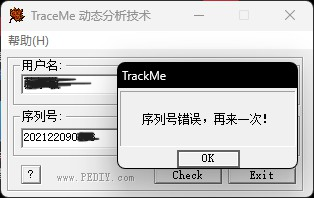
**学 院：** 信息与软件工程学院

**姓 名：** XXX

**学 号：** 202122090000

**一、题目名称：使用Ollydbg破解TraceMe程序**

**二、题目内容**

本次实验是结合课程上的实战练习，对程序TraceMe进行破解。通过Ollydbg软件进行反汇编，实现对程序进行破解。从中分析一个程序是如何从源程序到机器码的过程，并对实验结果给出一些分析性的结论，程序界面如图1所示。

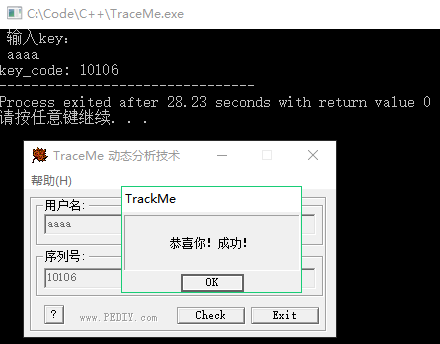


图1 TraceMe程序的界面

**三、知识点及介绍**

逆向工程又称反向工程，是一种技术过程，即对一专案标产品进行逆向分析及研究，从而演绎并得出该产品的处理流程、组织结构、功能效能规格等设计要素，以制作出功能相近，但又不完全一样的产品。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。

要对一个软件进行逆向分析，首先需要有一定的高级语言编程基础，其次是要有一定的汇编语言的基础知识。因为从源代码到可执行汇编代码，就是把高层的抽象逻辑打碎成一条条汇编代码。然后就是对一些逆向工具的使用，如：.NET工具、PE相关工具、编辑工具、编译工具、补丁工具、代码计算工具、反编译工具、反汇编工具、加壳工具、脱壳工具、监视软件、密码学工具、调试工具、其他工具等。具体有IDA Pro、DUMP、Hex Workshop、upx125w、peid和Stud\_PE以及最主要的OllyDbg。

汇编语言是任何一种用于电子计算机、微处理器、微控制器，或其他可编程器件的低级语言。在不同的设备中，汇编语言对应着不同的机器语言指令集。一种汇编语言专用于某种计算机系统结构，而不像许多高级语言，可以在不同系统平台之间移植。使用汇编语言编写的源代码，然后通过相应的汇编程序将它们转换成可执行的机器代码。这一过程被称为汇编过程。汇编语言使用助记符来代替和表示特定低级机器语言的操作。特定的汇编目标指令集可能会包括特定的操作数。许多汇编程序可以识别代表地址和常量的标签和符号，这样就可以用字符来代表操作数而无需采取写死的方式。普遍地说，每一种特定的汇编语言和其特定的机器语言指令集是一一对应的。

**四、工具及介绍：**

1.平台介绍

（1）操作系统：Windows 10

（2）破解工具：Ollydbg

（3）破解语言：汇编语言

（4）破解程序：TraceMe

2.OllyDbg介绍

OLLYDBG是一个新的动态追踪工具，将IDA与SoftICE结合起来的思想，Ring 3级调试器，非常容易上手，己代替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。

OllyDbg 可以以在任何采用奔腾处理器的 Windows 95、98、ME、NT 或是 XP（未经完全测试）操作系统中工作，但我们强烈建议您采用300-MHz以上的奔腾处理器以达到最佳效果。还有，OllyDbg 是极占内存的，因此如果您需要使用诸如追踪调试[Trace]之类的扩展功能话，建议您最好使用128MB以上的内存。

OllyDbg 的最大特点之一就是分析。它会分析函数过程、循环语句、选择语句、表[tables]、常量、代码中的字符串、欺骗性指令[tricky constructs]、API调用、函数中参数的数目，import表等等.. 这些分析增加了二进制代码的可读性，减少了出错的可能性，使得我们的调试工作更加容易。

**五、源程序**

|  |
| --- |
| #include <stdio.h>  #include <string.h>  //char name[65] = "abcdexxxx";  char name[65];  char table[8] = { 0xC ,0xA ,0x13 ,0x09 ,0x0C ,0x0B ,0x0A ,0x08 };  int main()  {  printf(" 输入key：\n ");  scanf("%s",name,65);  //会用到一个固定地址的值  //会用到姓名里的后两位  //eax = i  int user\_len = strlen(name);  int key\_code= 0; //esi    int count\_ecx = 3; //esi  int eax = 0; //eax  for (; count\_ecx<user\_len;)  {  if (eax>7)  eax = 0;  int ebx = 0;  int edx = 0;  edx = name[count\_ecx];  ebx = table[eax];  ebx = edx \* ebx;  key\_code += ebx;  count\_ecx++;  eax++;  }  printf("key\_code: %d", key\_code);  return 0;  } |

**六、过程及分析：**

1.操作流程

（1）打开ollydbg，并将TraceMe拖入到ollydbg程序的窗口中。

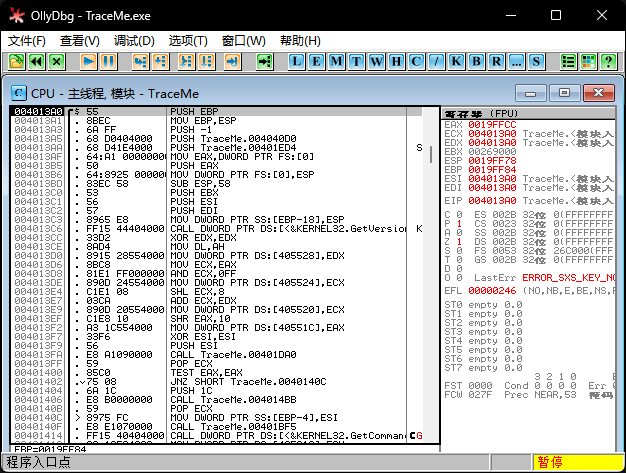


图2 TraceMe拖入到ollydbg中程序的界面

1. 在反汇编面板中搜索API：GetDlgItemTextA，，可以使用快捷键Ctrl+G键，程序会到达指定位置，在图中的第一行代码按F2键添加断点。添加断点后764A5C20变红。

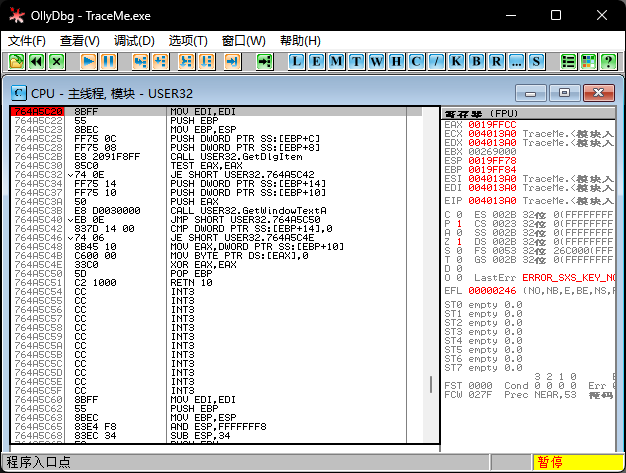


图3 程序中添加断点

1. 运行程序，程序将会停在要求输入用户名和序列号的地方，此时随意输入用户名abcde，序列号12345，并点击Check按钮，这时候程序停在刚才设置的断点处。

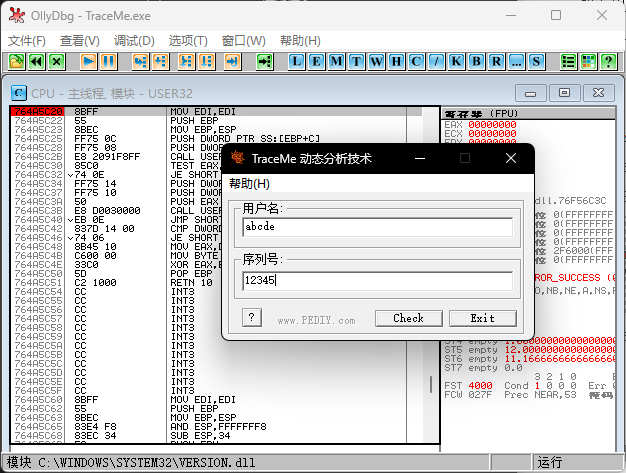
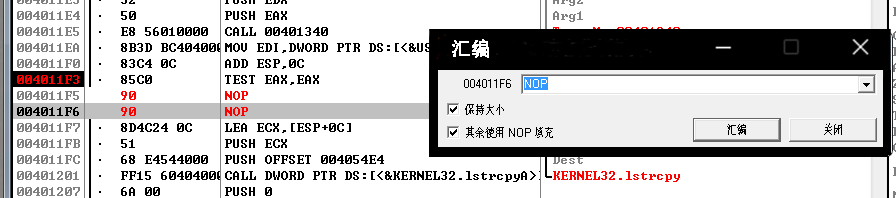


图4 程序停在断点处

（4）这时按F8单步调试，跳出GetDlgItemTextA函数，回到主程序。

（5）单步调试，找到序列号比较的汇编代码，双击跳转语句，输入nop，点击汇编按钮。



（6）选中改写好的两行，右击，复制到可执行文件的选择，最后右击保持文件。

（7）双击破解后的文件，运行，输入任意用户名序列号，实现激活。

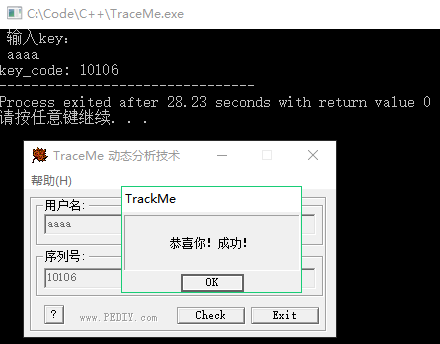


图5 程序激活

加密函数汇编注释如下：

00401340 push ebp ; ebp入栈

00401341 mov ebp,dword ptr ss:[esp+0xC] ; 将用户名移动到ebp中

00401345 push esi ; esi入栈

00401346 push edi ; edi入栈

00401347 mov edi,dword ptr ss:[esp+0x18] ; 将参数从堆栈中传给edi（用户名长度值） edi = 5

0040134B mov ecx,0x3 ; ecx = 3

00401350 xor esi,esi ; esi 清 0

00401352 xor eax,eax ; eax 清0

00401354 cmp edi,ecx

00401356 jle XTraceMe.00401379 ; edi<ecx条件成立时跳转,i<len

00401358 push ebx

00401359 /cmp eax,0x7 ; 比较eax与7的值

0040135C jle XTraceMe.00401360 ; 当等于7，ZF=1短跳转

0040135E |xor eax,eax

00401360 |xor edx,edx ; edx清0

00401362 |xor ebx,ebx

00401364 |mov dl,byte ptr ds:[ecx+ebp] ; 地址低8位的一个字节，dl是存储一个字节的寄存器，ecx = 3， [ecx+ebp] = d

00401367 |mov bl,byte ptr ds:[eax+0x405030] ; 00405030 0C 0A 13 09 0C 0B 0A 08

0040136D |imul edx,ebx ; edx \* ebx 赋值给 edx， dl为高8位，存储1字节数

00401370 |add esi,edx ; edx+esi，把值赋予给esi ，esi = 4B0 + 3F2

00401372 |inc ecx ; 递增指令，ecx 由 3 -> 4 -> 5

00401373 |inc eax ; eax = 1，eax++

00401374 |cmp ecx,edi

00401376 \jl XTraceMe.00401359 ; ecx 是否大于等于5（用户名长度）， jl指令大于不等于满足时跳转

00401378 pop ebx

00401379 push esi ; /<%ld>

0040137A push TraceMe.00405078 ; |Format = "%ld"

0040137F push ebp ; |s

00401380 call dword ptr ds:[<&USER32.wsprintfA>] ; \wsprintfA

00401386 mov eax,dword ptr ss:[esp+0x1C]

0040138A add esp,0xC

0040138D push ebp ; /String2

0040138E push eax ; |String1

0040138F call dword ptr ds:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA

程序原先输入的用户名：abcde、序列号：123456。在堆栈窗口看到的是d、e，也就是

name[3]=‘d’,

name[4]=‘e’

根据00401367 地址处判断，密码数组索引第3位之后的值逐位取出与固定地址的值比对。然后在0040137A 会输出密钥的值2201。

仔细逆推一遍：

edx = 64 \* 0C = 4B0

edx = 65 \* 0A = 3F2

4B0 + 3F2 = 8A2

8A2对应的十进制为2210

逆向结论：

abcde转换为十六进制从索引值第3位开始逐位取值，64：d、65：e ，然后与00405030 0C 0A 13 09 0C 0B 0A 08对应的值进行相乘然后累加。得出的8A2转换成十进制2210就是密码的值。

把加密函数取出来就成了算号注册机，上面第五小节源程序就是是加密函数反汇编转换来的C代码。

**七、总结及心得体会：**

通过学习《逆向工程实践》这门课程，能够从更高的层面上认识软件的安全性，熟练掌握各种逆向工具，对软件进行逆向分析。通过这门课，认识到了汇编语言的重要性，汇编语言就是逆向工程语言。逆向工程是一个实践性很强的课程，通过上机实验使我在本次课程的学习中收获很多，通过对程序的逆向分析，本人对计算机技术有了更深的认识。熟练掌握了汇编语言，才能更好地对软件进行逆向工程。在学习这门课的过程中，学到了许多以前没有接触的的概念和技术，拓宽了我的知识面。在今后的工作中，逆向工程会对我有很大的帮助。