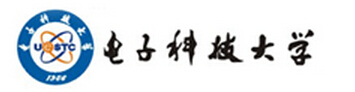
****

**逆向工程课程报告**

**课程名称：** 逆向工程

**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：** 杨庆

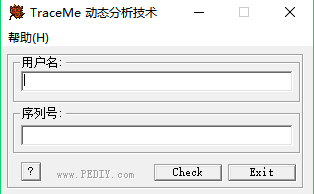
**学 号：** 201822090316

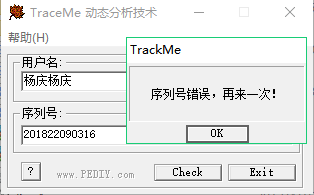
**一、题目名称**

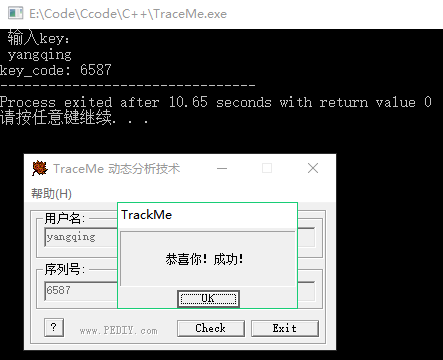
使用Ollydbg逆向破解TraceMe.exe程序

**二、题目内容**

本次实验的主要内容是通过Ollydbg软件进行反汇编，对程序TraceMe.exe进行破解。当用户输入正确的用户名和序列号之后，TraceMe提示“恭喜你，成功”，否则提示“序列号错误，再来一次”。程序界面如下图。







**三、知识点及介绍**

**1.逆向工程介绍**

逆向工程(Reverse Engineering，RE），又名反向工程。逆向工程是了解软件“所作所为”的一套最重要的技术和工具。正式地讲，逆向工程是“通过分析目标系统以识别系统的组件以及这些组件之间的相互关系并创建该系统另一种形式的表示或更高级的抽象过程”。逆向工程主要是从没有源代码的程序出发，生成对应的源程序、系统结构以及相关设计原理和算法思想的文档等，也称为二进制逆向工程。软件逆向工程融合了密码破解（code breaking）、猜字谜（puzzle solving）、程序设计和逻辑分析等技术。

要对一个软件进行逆向分析，我们需要熟悉常用的逆向工具和底层软件，包括诸如：编译器、链接器（Linker）和调试器的开发工具，诸如操作系统的基础软件，以及像汇编语言这样的低级程序设计语言可能涉及到加密和解密、反汇编、系统分析、程序理解等多种计算机技术。如果没有合适的工具，逆向是不可能实现的。主要包含.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。常用的逆向工具包括：IDA Pro、DUMP、Hex Workshop、upx125w、peid和Stud\_PE以及最重要且经常使用的OllyDbg。

**2.汇编语言介绍**

汇编语言是直接面向处理器（Processor）的程序设计语言。处理器是在指令的控制下工作的，处理器可以识别的每一条指令称为机器指令。汇编语言所操作的对象不是具体的数据，而是寄存器或者存储器，也就是说它是直接和寄存器和存储器打交道，这也是为什么汇编语言的执行速度要比其它语言快，但同时这也使编程更加复杂，因为既然数据是存放在寄存器或存储器中，那么必然就存在着寻址方式，也就是用什么方法找到所需要的数据。这也就增加了编程的复杂性，因为在高级语言中寻址这部分工作是由编译系统来完成的，而在汇编语言中是由程序员自己来完成的，这无异增加了编程的复杂程度。汇编语言位于软件链的最底层，它适合于做逆向，汇编语言就是逆向语言。对大多数情况来说，汇编语言就是逆向的“专用”语言，要成为一名真正的逆向者就必须掌握它，因为对于大多数程序而言，汇编语言是你能够“联系”到最初的源代码的唯一途径。

**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.exe

**2.OllyDbg介绍**

对于逆向工作人员而言，由 Oleh Yuschuk 编写的 OllyDbg 可能是最佳的用户模式调试器（尽管选择的余地非常小）。Olly的诱人之处在于：它一开始就是作为逆向工具而设计的，因此它具有强大的内置反汇编器。OllyDbg最大的优点：在于它提供了强大的代码分析能力的反汇编器。

OllyDbg的代码分析器可以识别出 循环、switch控制块 以及其它主要的代码结构。它能显示所有 已知函数 和 API 的参数名，支持在代码和数据之间查找交叉引用——代码到数据或数据到代码。OllyDbg是调试器中（除了IDA Pro调试器）反汇编能力最强的一款，连内核模式都比不上。

除了具有强大的反汇编能力外，OllyDbg还提供了大量不同的视图，包括列出模块中的 导入和导出、显示被调试者拥有的 窗口 和其它 对象的列表、显示当前的 异常句柄链 以及对那些在库中正确命名的函数使用导入库（.lib文件）等等。OllyDbg还包括一个内置的编译和修补引擎，这使得它成了软件破解者的最爱。

**3.快捷键介绍**

（1）F2：下断点。

（2）Alt+b：打开断点编辑器，可编辑所有下过的断点。

（3）空格键：可快速切换断点状态。

（4）Ctrl+F9：当位于某个CALL中，这时想返回到调用这个CALL的地方时，可以按“Ctrl+F9”快捷键执行返回功能。这样OD就会停在遇到的第一个返回命令（如RET、RETF或IRET）。

（5）Alt+F9：如果跟进系统DLL提供的API函数中，此时想返回到应用程序领空里，可以按快捷键“Alt+F9”执行返回到用户代码命令。

（6）Ctrl+G：跳转到API、地址的位置。

**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++;

}

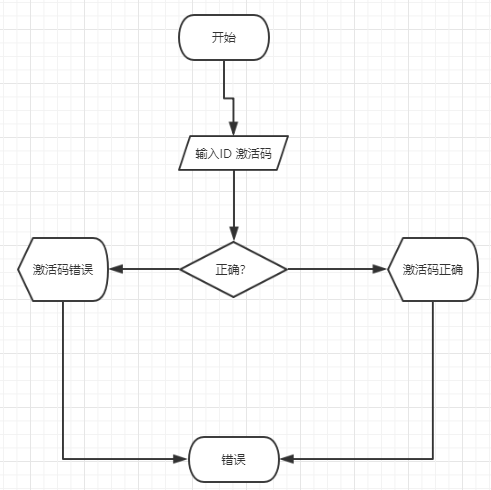
printf("key\_code: %d", key\_code);

return 0;

}

**六、过程及分析**

**1.程序运行机制**



**2.算法逆向**

（1）打开ollydbg，并将TraceMe.exe拖入到ollydbg的窗口中。

（2）在反汇编面板中搜索API：GetDlgItemTextA，（注意：这里大小写敏感），可以使用快捷键Ctrl+G键，程序会到达如下图所示的位置，在图中的第一行代码按F2键添加断点。添加断点后75D8D5C0会变红。

（3）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123456，并点击Creck按钮，这时候程序会停在刚才设置的断点处。

（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。

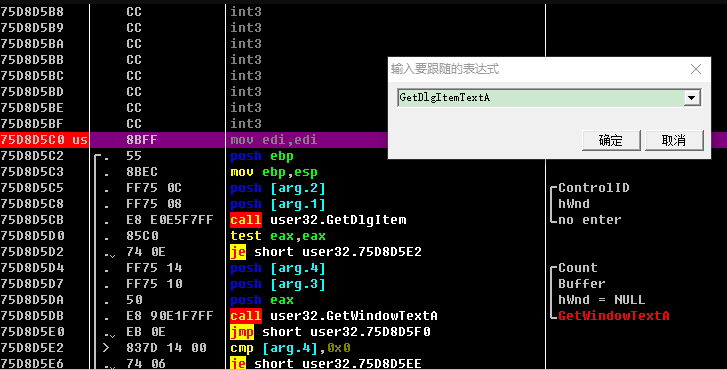
（6）选中改写好的两行，右击，复制到可执行文件-〉选择，如图所示，最后右击，保持文件。

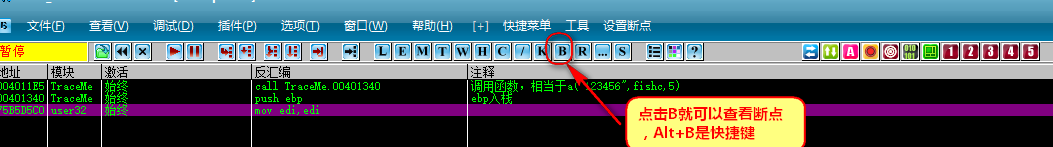
（7）双击破解后的文件，运行，输入任意用户名序列号，实现激活。

**详细步骤：**

【GetDlgItemText 函数】 用于获取对话框中指定控件的标题或文本。

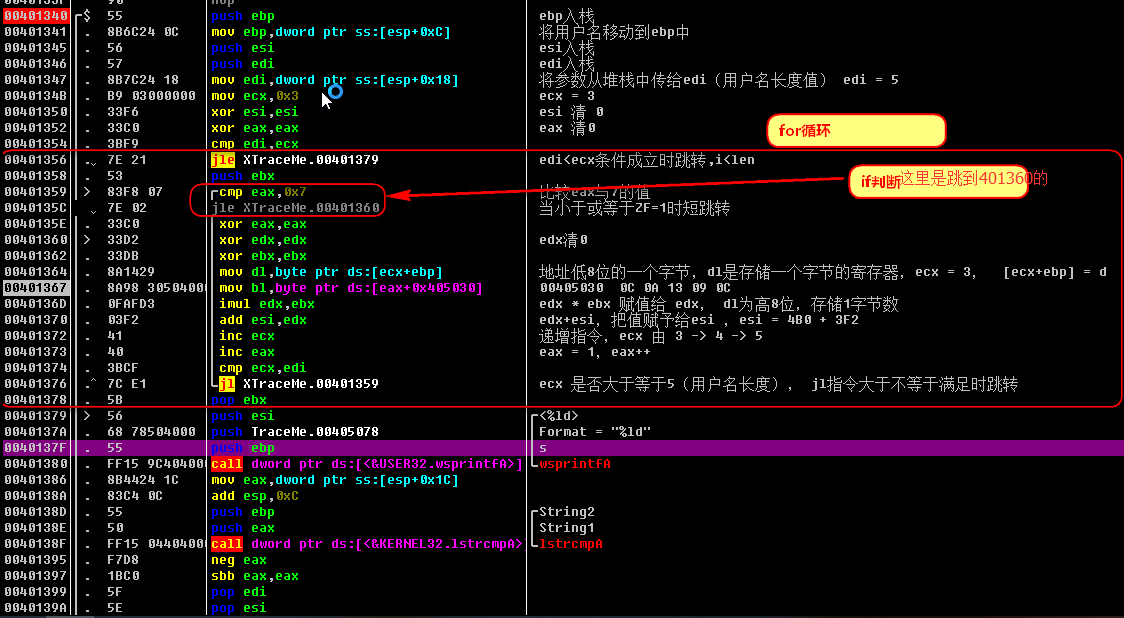
使用OD的快捷键【Ctrl+G】跳转到API的位置下断点。F2设置断点，当【GetDlgItemTextA】这个函数被调用OD就会中断。【ALT+B】快捷键可以打开断点窗口查看，在断点位置按【空格键】可以激活与禁用断点。





按快捷键【Ctrl+F9】可以回到调用函数的尾部ret处。

F7跟进004011E5地址内的函数，进入子程序call 00401340，特别值得注意的是00401359是跳转到4013680的，相关汇编代码注释如下：



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

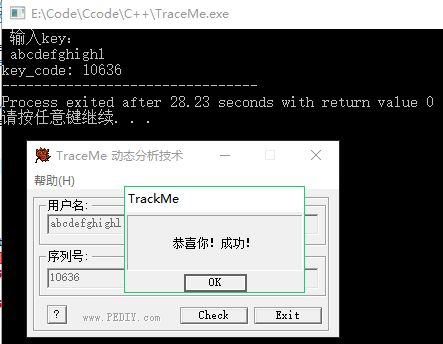
4B0 + 3F2 = 8A2

8A2对应的十进制为2210

逆向结论：

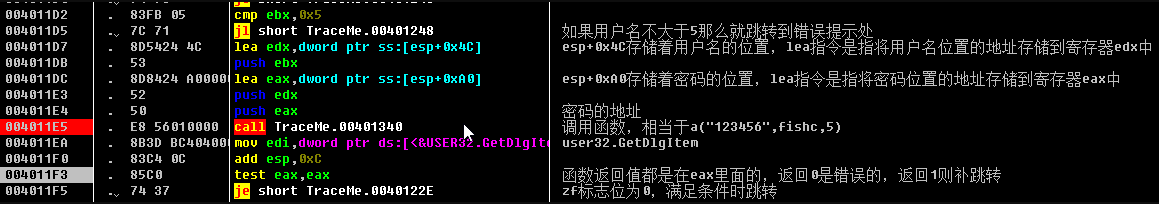
abcde转换为十六进制从索引值第3位开始逐位取值，64：d、65：e ，然后与00405030 0C 0A 13 09 0C 0B 0A 08对应的值进行相乘然后累加。得出的8A2转换成十进制2210就是密码的值。

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。



**3.暴力破解**

在定位到GetDlgItemText这个API处，F8单步步过向下跟随到有test判断的地方，注意观察数据堆栈区的位置。



汇编指令注释如下：

0040119C mov esi,dword ptr ss:[esp+0x100] ; Case 3F5 of switch 0040115E

004011A3 mov edi,dword ptr ds:[<&USER32.GetDlgIte>; user32.GetDlgItemTextA

004011A9 push ebx

004011AA lea eax,dword ptr ss:[esp+0x4C]

004011AE push 0x51 ; /Count = 51 (81.)

004011B0 push eax ; |Buffer

004011B1 push 0x6E ; |ControlID = 6E (110.)

004011B3 push esi ; |hWnd

004011B4 call edi ; \GetDlgItemTextA

004011B6 lea ecx,dword ptr ss:[esp+0x9C]

004011BD push 0x65 ; /最大字符数

004011BF push ecx ; |文本缓冲区指针

004011C0 push 0x3E8 ; |控件标识

004011C5 push esi ; |对话框句柄

004011C6 mov ebx,eax ; |将用户名的长度转到ebx中

004011C8 call edi ; \GetDlgItemTextA

004011CA mov al,byte ptr ss:[esp+0x4C] ; 将用户名的第一个字节给al

004011CE test al,al ; 检查有没有输入用户名

004011D0 je XTraceMe.00401248 ; 如果没有输入用户名跳走，告知输入的字符太少，zf=0跳转

004011D2 cmp ebx,0x5

004011D5 jl XTraceMe.00401248 ; 如果用户名不大于5那么就跳转到错误提示处

004011D7 lea edx,dword ptr ss:[esp+0x4C] ; 用户名地址放到edx中

004011DB push ebx ; 用户名长度

004011DC lea eax,dword ptr ss:[esp+0xA0] ; 密码地址放到eax

004011E3 push edx ; 用户名地址入栈

004011E4 push eax ; 密码地址入栈

004011E5 call TraceMe.00401340 ; 调用函数，相当于a("123456",abcde,5)

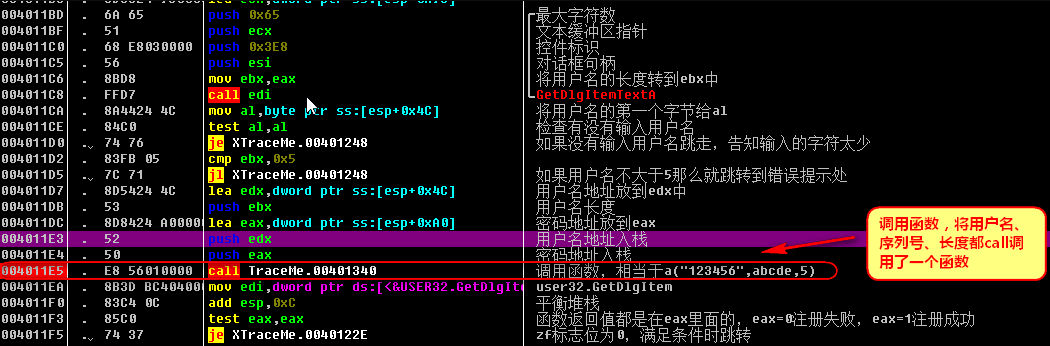
004011EA mov edi,dword ptr ds:[<&USER32.GetDlgIte>; user32.GetDlgItem

004011F0 add esp,0xC ; 平衡堆栈

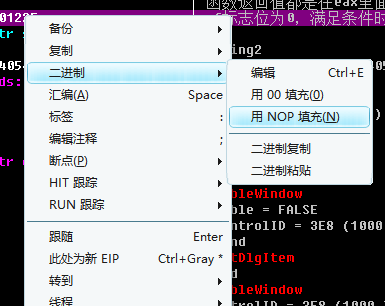
004011F3 test eax,eax ; 函数返回值都是在eax里面的，eax=0注册失败，eax=1注册成功

004011F5 nop ; zf标志位为0，满足条件时跳转

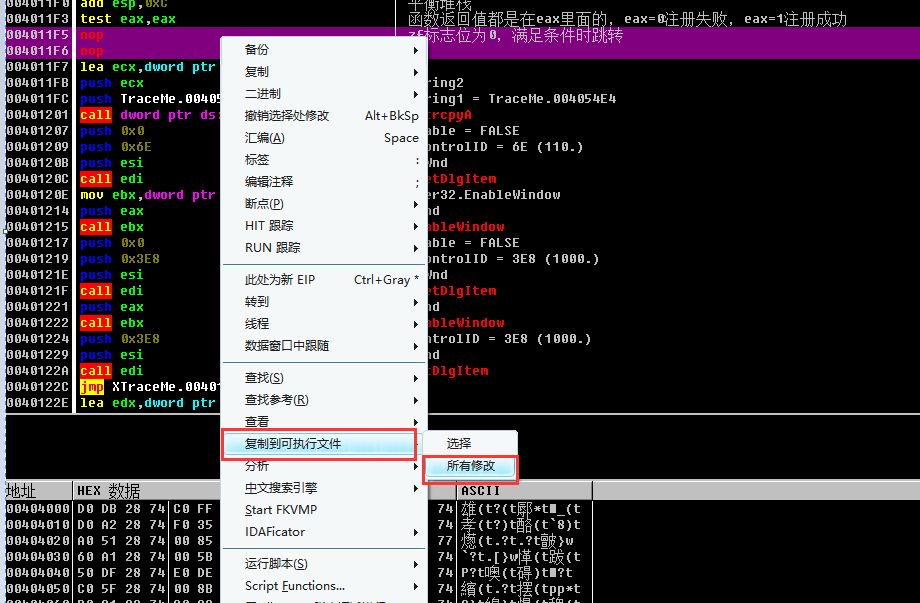
第一个test指令首先对比用户名是否大于5，不大于5就跳转到弹出错误提示的地方。否则继续执行。F8单步步过进入【GetDlgItemTextA】后面的调用查看相关的代码。004011E5地址处call调用一个函数，并且在之前push了三个参数。



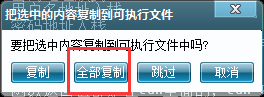
第二处test指令下面那条je跳转指令用nop填充掉。



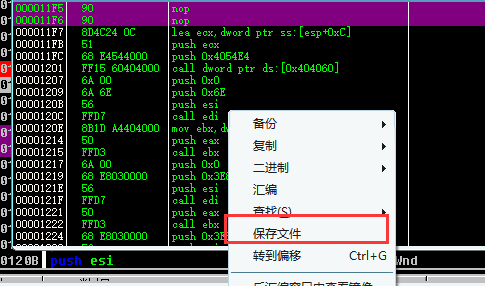
然后右键【复制到可执行文件】-【所有修改】。



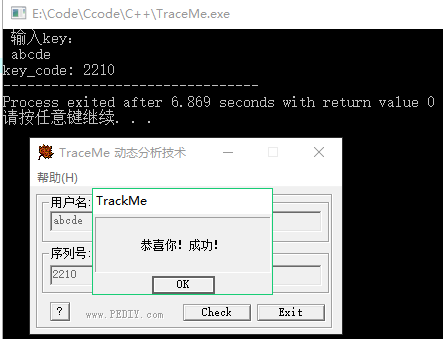
选择全部复制。



选择【保存文件】，暴力破解就完成了。



成功后输入任何密码都提示成功弹框，截图如下：



**七、心得体会**

通过本次逆向工程这门课程的学习，我熟悉了各种逆向工具，进一步巩固了汇编语言的语法知识及与高阶代码的相互关系。

在学习这门课的过程中，学到了许多概念和逆向技术，让我对计算机底层有了更进一步的认识。我明白了逆向与开发的知识是成正比关系，只有对开发特别熟悉，逆向一个程序才能猜测到该用哪个关键的API才能快速定位到程序的数据处理。通过PEID查看程序特征，根据程序语言用IDE生成特征或者熟悉开发的API函数。就可以更方便地让我们定位到获取edit值的函数。

并且我学会了从高阶和低阶结合的角度来看待软件，这样才能体会到软件的漏洞与安全性，有那些潜在的威胁。对我以后的工作和发展有着很大的帮助。