《软件安全》实验报告

姓名：许洋 学号： 2313721 班级： 计卓

**实验名称：**

OLLYDBG软件破解

**实验要求：**

 1.请在XP VC6生成课本第三章软件破解的案例(DEBUG模式，示例3-1) 。进而，使用OllyDBG进行单步调试，获取verifyPWD函数对应flag==0的汇编代码，并对这些汇编代码进行解释。

​ 2.对生成的DEBUG程序进行破解，复现课本上提供的两种破解方法。

**实验过程：**

一、初步测试

​ 1.在VC6上建立源文件，输入实验源码后，编译生成可执行的.exe文件，并进行测试。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-12 224433.png)

​ 如图可看出当输入的密码不正确时，会输出“wrong password”，并持续重复输入，无法进入核心逻辑（即程序跳出）。那么我们如何通过修改程序的汇编语言代码来实现进入核心逻辑呢？我们需要进行以下的一些步骤。

​ 2.我们使用ollydbg，将lab2.exe文件拖入ollydbg进行动态调试。导入后如下图所示。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\Windows XP Professional-2024-03-12-22-48-13.png)

​ 下面我们就对其进行破解操作。

二、对verifyPwd函数的具体解释

​ 首先，我们知道该函数肯定是决定着我们能否进入核心过程的一个重要函数，所以我们先进行关键字查询，看看有没有出现“wrong”之类的关键词。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-12 231850.png)

​ 经过搜索，我们发现verifyPwd函数就是从此处开始的。而且，这个密码正确或者是错误的判断是从上面的一句汇编语句jz short 00401105来决定的。

​ 为了找到verifyPwd函数的位置，我们右键选择“跟随”，在跟随两次后，我们找到了该函数的位置。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-12 235310.png)

核心代码：

mov eax,dword ptr [ebp+8]

push eax

push offset 0043101C ; ASCII "12345678"

call strcmp ; [strcmp

add esp,8

mov dword ptr [ebp-4],eax

xor eax,eax

cmp dword ptr [ebp-4],0

sete al

​ 第一行将输入的password的首地址移入eax中，空出8个位置

​ 第二行将eax入栈

​ 第三行将待比对的正确密码入栈（12345678）

​ 第四行调用strcmp函数比对两个字符串是否相等，如果两个字符串相等，则eax的值为0，如果两个字符串不相等，则eax的值为00000001

​ 第五行做栈平衡

​ 第六行将strcmp的结果从eax中移送到局部变量flag中

​ 第七行将eax寄存器的值清空

​ 第八行对flag的值和0进行比较，如果相等的话就会设置ZF为1，否则ZF为0

​ 第九行根据ZF的值设置eax的低八位，如果ZF为1，就设置al为1，反之al被置0

三、两种破解方法

3.1 修改主函数中的条件跳转指令

​ 我们回到上面的verifyPwd函数部分的判定密码是否正确的部分，可以发现，基本上是由一个jz的判定语句构成的。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-12 231850.png)

​ 我们跟踪00401105语句，发现该语句是用于输出密码错误的，反之004010F6语句是用于输出密码通过测试。

​ 所以，实际上程序的逻辑应该是：如果密码符合，那么跳转到“passed”语句，否则跳转到密码错误的语句。因此，我们可以修改逻辑判断语句\*\*jz short 00401105为jnz short 00401105\*\*，这样就可以反方向跳转，输入错误的密码会跳转到passed界面。

​ 我们双击该语句，进行修改，然后右键选择“编辑”，选择“复制当前修改到所有可执行文件”，这样才是真正修改了我们的lab.exe文件。我们修改完毕后，重新打开exe文件，输入原本是错误的密码“111”，发现输出了passed，说明我们的修改起了作用。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-13 122338.png)

​ 为了进一步验证，我们输入原本正确的密码“12345678”，发现输出的是wrong，说明程序的核心判断语句已经被我们修改为反逻辑了。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-13 123019.png)

3.2 修改flag==0的返回值指令，令其永远返回1

​ 接下来我们尝试第二种方法来进行破解。基于我们在第一部分中讨论的flag==0的对应区域汇编代码，我们可以想到另一种更有效地破解方法，那就是让flag==0的返回值永远为1，这样可以避免当输入正确密码而提示密码错误的情况。这样我们就可以实现，不管输入什么密码，都会输出passed的结果。

​ 根据上面所说的，我们找到flag==0的语句，发现该语句的功能是靠cmp dword ptr [ebp-4],0来实现的。我们直接将其判断函数删除，修改为直接赋值，即直接将al寄存器的值修改为1即可。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-13 125751.png)

​ 进一步，我们再来修改下一条语句sete al。我们已经给al寄存器永久赋值为1了，所以自然不需要这一步了。我们直接将该语句使用nop指令进行填充即可。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-13 130020.png)

​ 我们同样右键选择编辑，复制当前修改到可执行文件，然后运行可执行文件。我们发现无论输入什么样的密码，窗口都一闪而过，证明我们成功破解了程序。

![](E:\学学学\本科\大二下\软件安全\软件安全\_实验\_2211044\_陆皓喆\Lab2\pic\屏幕截图 2024-03-13 130045.png)

**心得体会：**

​ 通过本次实验，我学会了ollydbg的动态调试方式，包括一些很简单的操作，比如说在整个栈内存中通过跟随操作来跟踪程序的进程；通过查找关键词来定位关键语句；通过修改程序汇编语言代码来实现程序的简单破解。

​ 通过实验，我熟悉了很多关键核心逻辑语句的寻找与判定，学会了修改核心语句来使代码错误跳转从而达到目的，对汇编语言代码的认识更加深刻了。这个实验和上学期的《汇编语言与逆向技术》中的最后一个实验十分的相似，上学期我们也是通过修改汇编语言中的核心代码来实现boss血量的修改，boss数量的修改等等。这和本次实验也是共通的，其实理解汇编语言的核心是找到准确的核心逻辑语句的位置。比如说本次实验，我们通过查找关键词的方式，在很大的代码块中找到了自己需要的东西。在CTF比赛中的某些题型，这样的分析技能是最基本的，我还需要多做该方面的练习来进一步熟练掌握该项技能。