《漏洞利用及渗透测试基础》实验报告

姓名：袁田 学号： 2314022 班级： 计科三班

**实验名称：**

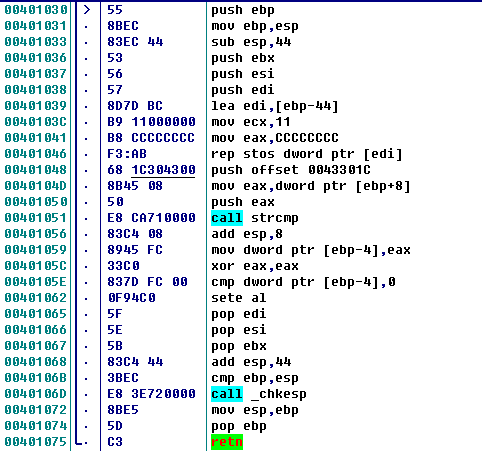
OLLYDBG软件破解

**实验要求：**

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。并对生成的DEBUG程序进行破解，复现课本上提供的两种破解方法。

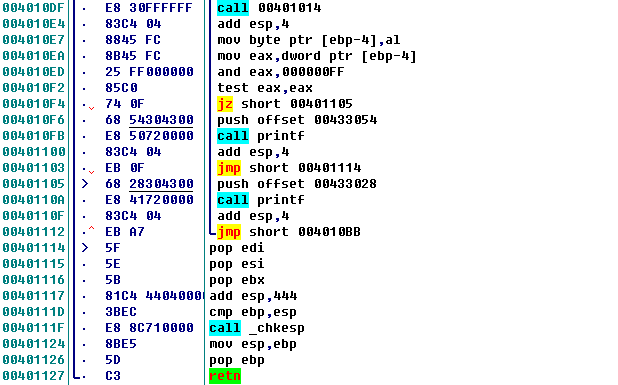
**实验过程：**

1. 使用OLLYDBG进行单步调试，获取verifyPWD函数对应flag==0的汇编代码



经过了call strcmp指令后，在寄存器eax中保存了两个字符串是否相等的情况，即相等时eax=1。随后将寄存器eax的值保存至[ebp-4]该地址中，并通过xor eax eax指令将寄存器eax中的值清空。接着通过cmp dword ptr [ebp-4],0和sete al语句，将位于[ebp-4]地址位置的数据与0进行比较(相当于代码中的flag==0的判断)，将该判断导致的ZF零标志位的变化，通过sete指令将目标操作数al设置为1或0。

2.主函数中对verifyPWD函数的调用及对应的条件判断指令

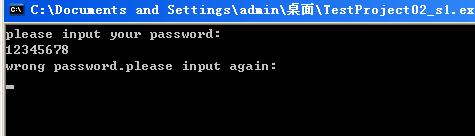


在调用verifyPWD函数(即执行call 00401014指令后)，由于对verifyPWD函数的分析可知，al中储存的为判断flag==0的结果，也就是说当al=1时，证明输入的字符串和密码相同。随后通过一系列指令语句，将al的值储存到eax寄存器中，并只保留eax的后八位。然后，经过test eax eax和jz short 00401105指令，由于test语句的作用为：进行逻辑按位与操作，但不保存结果，只根据结果设置标志位（Flags），其核心作用是快速检测某些位的状态，用于进行后续条件跳转前的判断。在此处，通过test eax eax将零标志位ZF设置为1，再通过jz short 00401105指令，若ZF=1，则跳转到地址00401105进行执行操作，相当于源代码文件中的if(BFLAG)，在BFLAG为真是进行操作。

3.两种破解方法

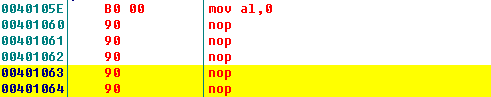
3.1 通过修改主函数中的条件判断语句，将逻辑倒转，即：输入正确密码进入到原来输入错误密码的分支中，反之输入错误密码，均进入原密码正确的条件分支中



通过将原本的指令jz改为jnz，由于jz和jnz同样依赖于零标志位，其中jz为结果为0时跳转，用于检查零值或相等条件。而jnz为结果非0时跳转，用于检查非零值或不相等条件。则恰好实现了逻辑上的相反功能，具体实现情况为：

其中12345678为正确答案，返回原本错误情况的结果。输入其他答案，程序结束

3.2通过修改verifyPWD函数，使得原本的sete al由通过比较flag是否为0来确定是否赋值为1，改变为始终赋值为1，具体改动如下：



通过将cmp dword ptr [ebp-4],0和sete al语句修改为mov al 0则可以将al始终设置为0，则verifyPWD函数返回值始终为1，即无论输入任何字符串都看作正确结果。最终修改的程序为，无论输入任何字符串都直接结束程序。

**心得体会：**

通过实验，学会了如何使用OLLYDBG对于程序进行破译操作

此外，通过本实验，掌握了多个汇编语言的用法，例如对于test指令和jz指令等的相互联系关系，以及知道了sete指令的作用是根据零标志位（ZF）的状态，将目标操作数设置为1或0，通常在比较操作后使用，将条件结果保存为布尔值。