《软件安全》实验报告

姓名： 邢清画 学号： 2211999 班级： 1023

**实验名称：**

OllyDBG软件破解实验

**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。

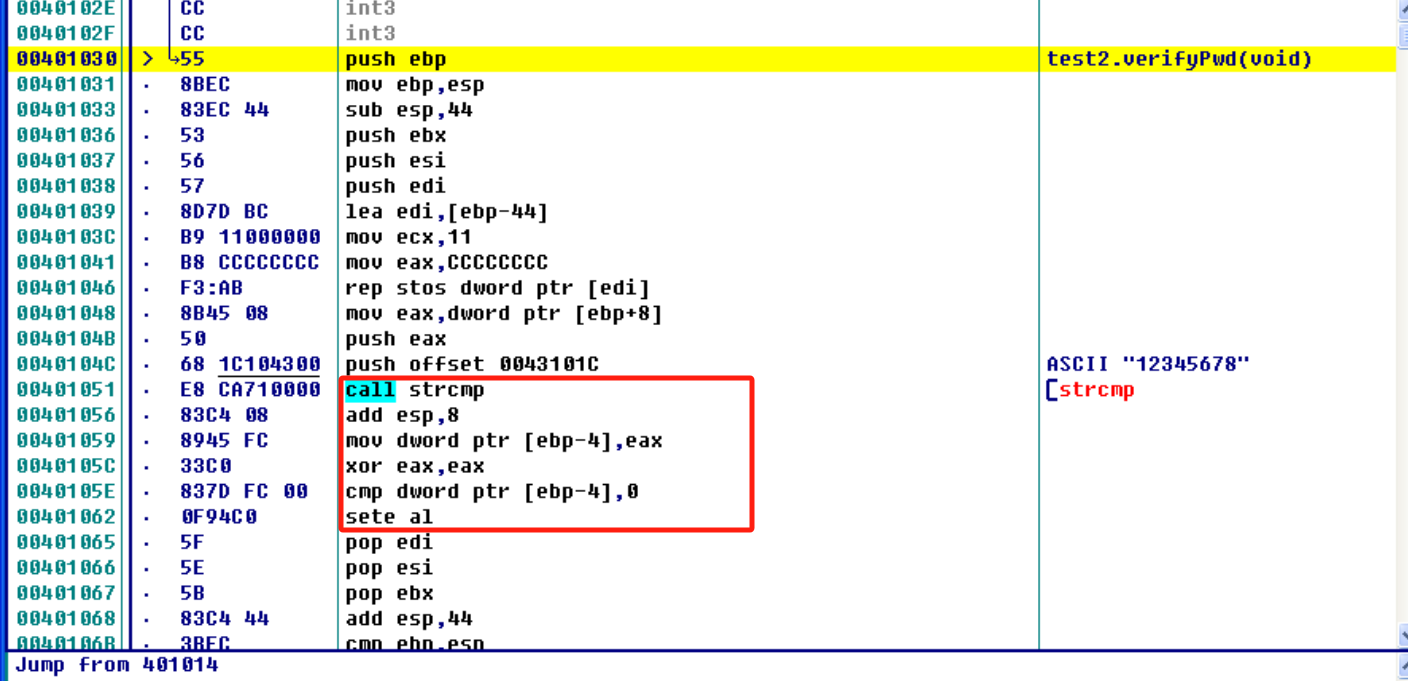
2. 对生成的DEBUG程序进行破解，复现课本上提供的两种破解方法。

**实验过程：**

1.对示例3-1的源代码生成的Debug模式的可执行文件使用OllyDBG进行破解

使用OIlyDBG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。

跟随verifyPWD函数，逐步进入函数内部，进而判断flag的值：



call strcmp//

调用strcmp函数比较两个字符串。函数的参数通常在调用前通过寄存器或栈传递，strcmp的返回值会放在eax寄存器中。strcmp函数返回0表示两个字符串相等，非0值表示不相等。

add esp,8//

调用strcmp函数后，栈指针esp需要调整来清理掉传递给strcmp的参数。将esp增加8，之前有两个参数（每个参数占用4字节）被压入栈中。这是在恢复栈到调用strcmp之前的状态。

mov dword ptr [ebp-4],eax//

将eax寄存器（即strcmp的返回值）保存到基指针ebp指向的栈帧中的某个位置（偏移量为-4）。用来保存strcmp的结果以便后续处理。

xor eax, eax//

通过对eax寄存器进行异或操作自身来将eax清零。

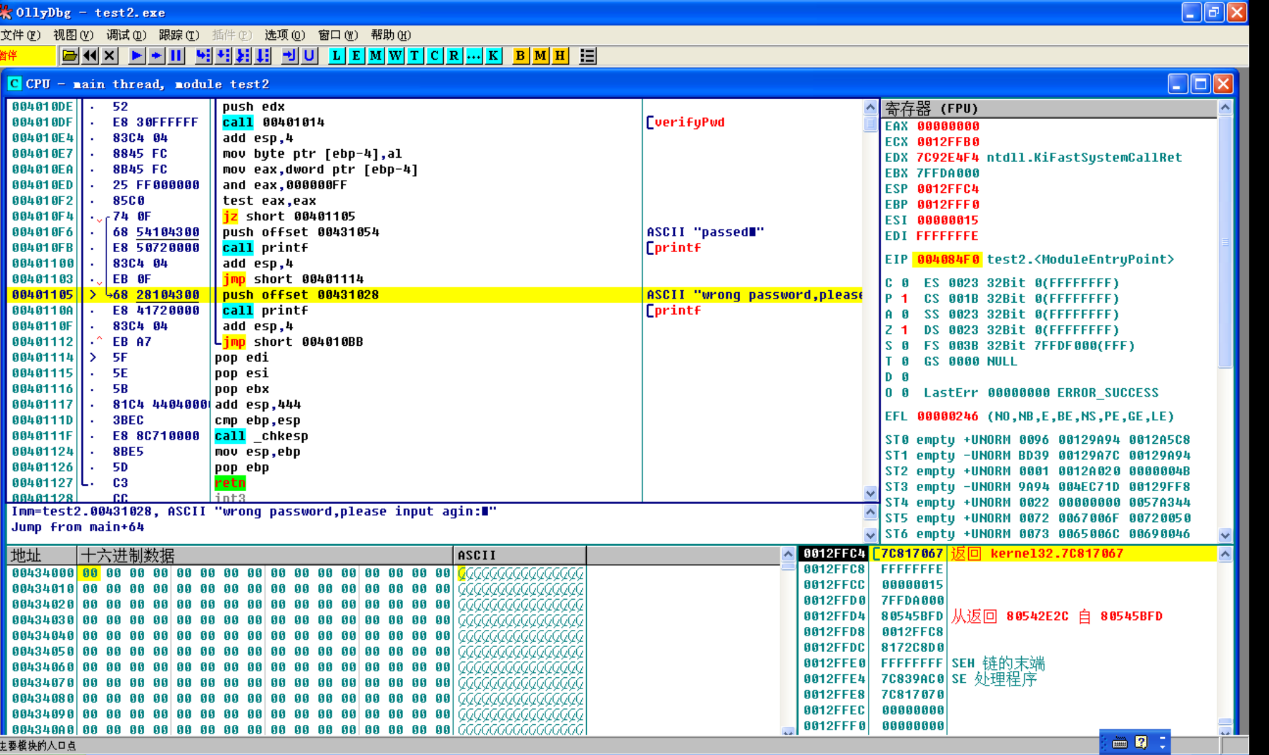
cmp dword ptr [ebp-4],0//

比较之前保存在ebp-4位置的strcmp返回值（现在被当做一个局部变量或临时存储）与0，为了检查两个字符串是否相等（strcmp返回0表示相等）。

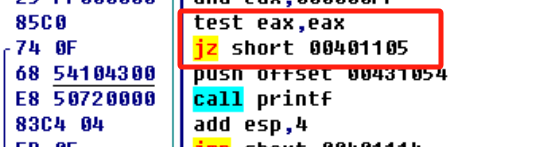
sete al//

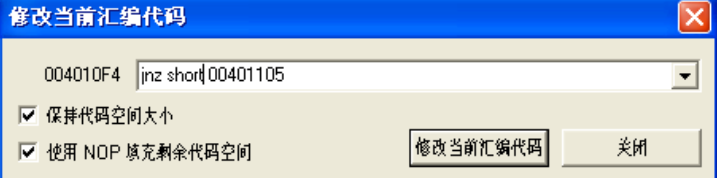
根据上一条cmp指令的结果设置al寄存器（eax的低8位）。如果比较的结果显示相等（即，strcmp返回0，满足cmp指令的条件），sete（Set if Equal）将al设置为1（真）。如果不相等，al保持为0（假）。al/eax寄存器就表示了比较的bool结果。

2. 对示例3-1的源代码生成的Debug模式的可执行文件使用OllyDBG进行破解

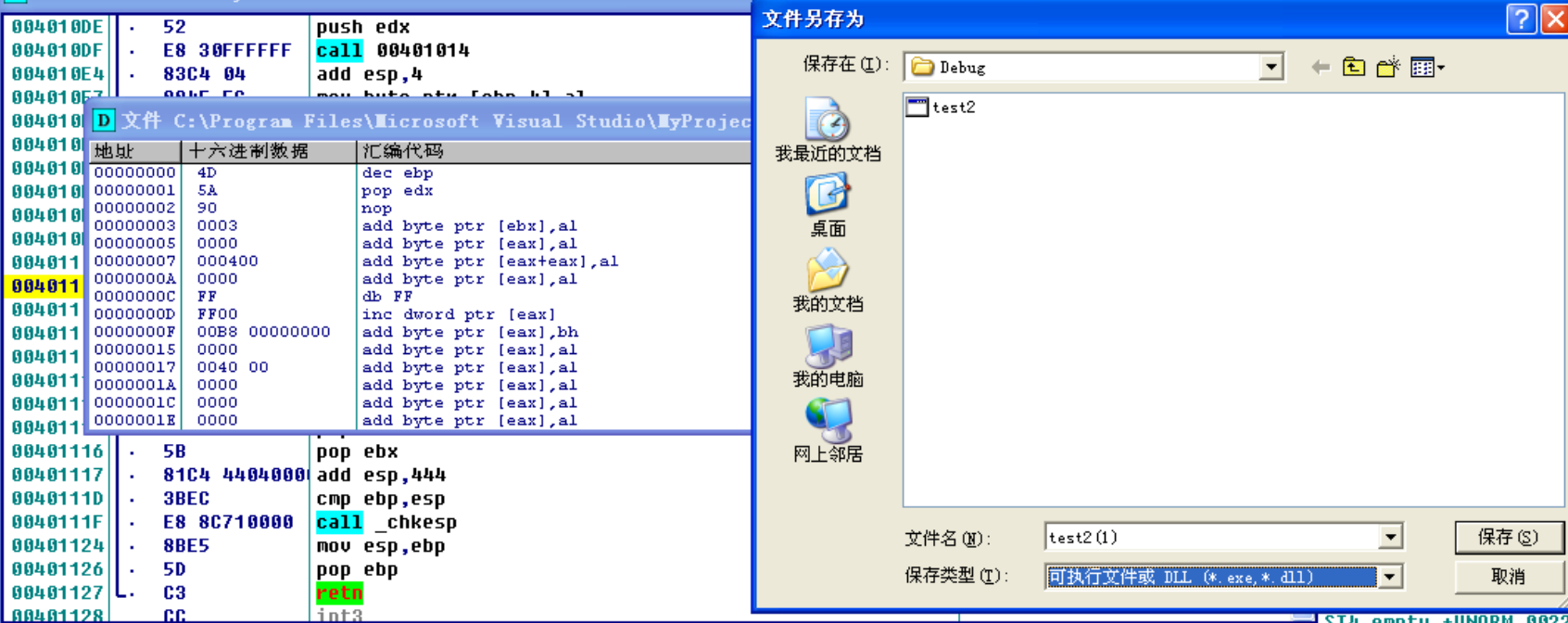
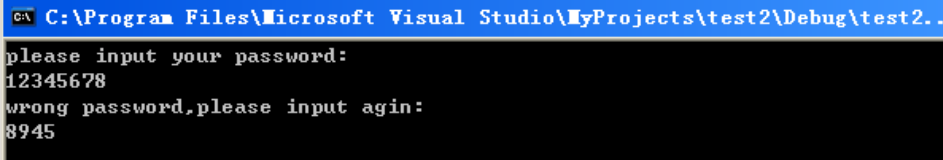
快速定位分支语句，查找“wrong”字符串，定位代码位置

**2.1破解方法一**：

逻辑上，若输入密码错误则跳转到报错区域，正确则跳转到正确区域，核心分支判断为：  
如果jz条件成立，则会跳转到00401105处，即显示错误密码分支语句，对jz取反jnz，当输入错误密码会跳转到正确的分支中，修改jz部分代码：

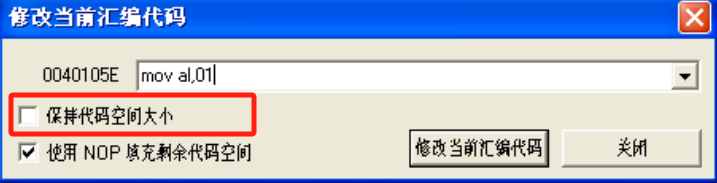


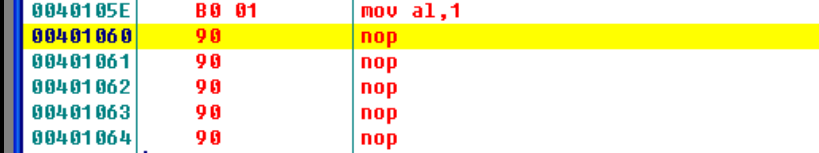
在反汇编窗口，“编辑 复制当前修改到可执行文件”，修改完成，保存文件为可执行文件。运行修改过的文件，输入正确显示错误，错误密码显示成功。



**2.1破解方法二：**

修改函数语句，使用户不论输入正确还是错误都能成功，即强制更改al为01。

通过在jz short之前出现的call函数，跳转到verifyPwd函数，继续跟随，逐步进入该函数，观察到最后返回的是bool类型的值（0,1），发现cmp和sete实现了只有相等才设置al的值为01，因此我们应该实现不管是否相等，都强制设置al值为01。



（ps:不保持代码空间大小不变，如果新代码超长会无法完成更改）

最后编辑保存所有修改，验证成功。

**心得体会：**

1. 学习和实践如何使用调试器（如OllyDBG）进行单步调试，更深入地理解程序是如何在计算机上执行的，包括函数调用、条件判断、循环等基本控制流程，以及更高级的概念，如堆栈管理和寄存器使用。
2. 仔细分析软件的执行流程，找出验证机制，并思考如何绕过它们，使用不同的破解方法破解。学习到如何使用逆向工程工具和技术。
3. xor eax, eax是汇编语言中常见的将寄存器值设置为0的方法。Sete al相等则置1。
4. 内存访问通过指针加偏移量的方式进行，如[ebp-4]，eax（用于存储函数返回值和临时结果）。