# 《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 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。

跟随 verifyPWD 函数，逐步进入函数内部，进而判断 flag 的值：

```
0040102E    CC          int3
0040102F    CC          int3
00401030  > 55          push ebp                        test2.verifyPwd(void)
00401031  . 8BEC        mov ebp,esp
00401033  . 83EC 44     sub esp,44
00401036  . 53          push ebx
00401037  . 56          push esi
00401038  . 57          push edi
00401039  . 8D7D BC     lea edi,[ebp-44]
0040103C  . B9 11000000 mov ecx,11
00401041  . B8 CCCCCCCC mov eax,CCCCCCCC
00401046  . F3:AB       rep stos dword ptr [edi]
00401048  . 8B45 08     mov eax,dword ptr [ebp+8]
0040104B  . 50          push eax
0040104C  . 68 1C104300 push offset 0043101C            ASCII "12345678"
00401051  . E8 CA710000 call strcmp                     [strcmp
00401056  . 83C4 08     add esp,8
00401059  . 8945 FC     mov dword ptr [ebp-4],eax
0040105C  . 33C0        xor eax,eax
0040105E  . 837D FC 00  cmp dword ptr [ebp-4],0
00401062  . 0F94C0      sete al
00401065  . 5F          pop edi
00401066  . 5E          pop esi
00401067  . 5B          pop ebx
00401068  . 83C4 44     add esp,44
0040106B  . 3BEC        cmp ebp,esp
Jump from 401014
```

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，

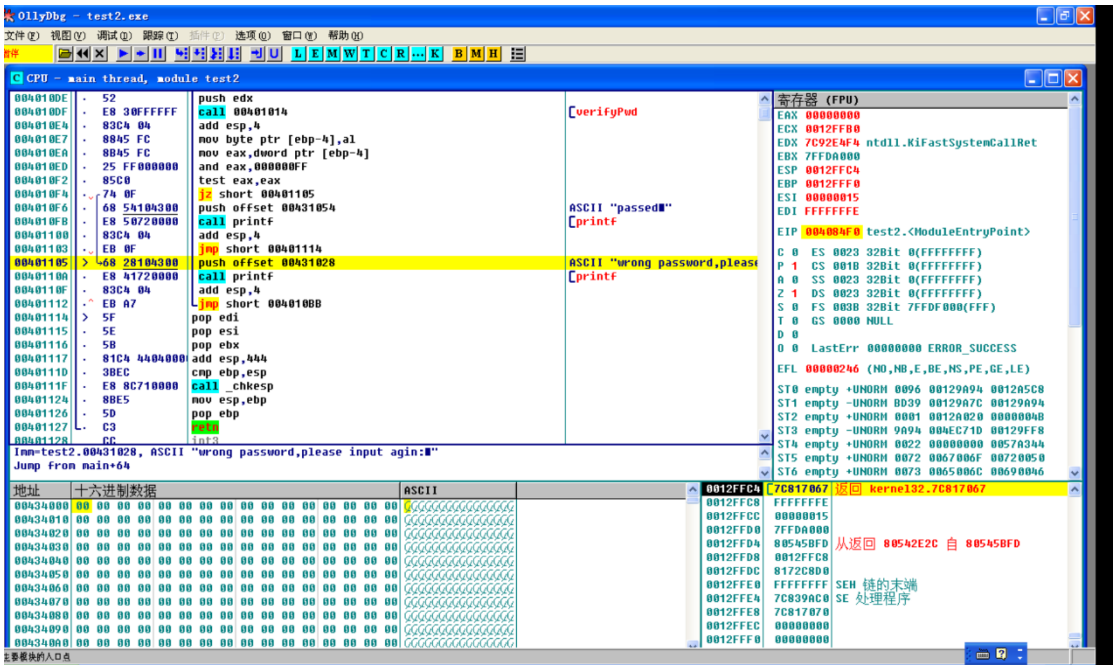
为了检查两个字符串是否相等（strcmp 返回 0 表示相等）。

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。

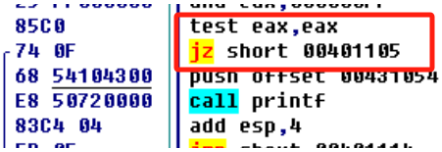
2. 对示例 3-1 的源代码生成的 Debug 模式的可执行文件使用 OllyDBG 进行破解

快速定位分支语句，查找“wrong”字符串，定位代码位置

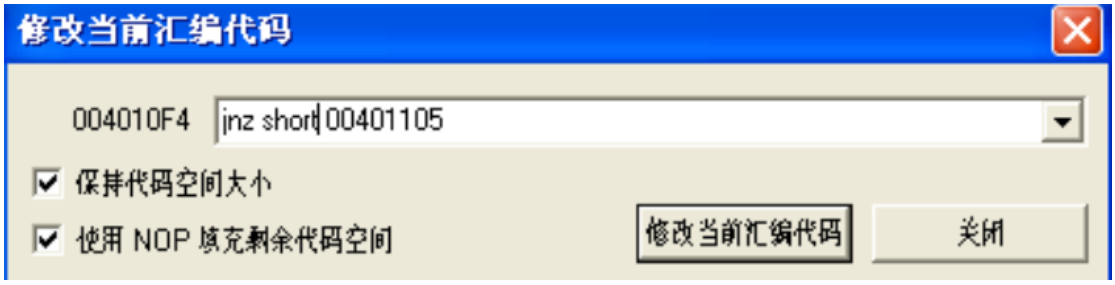



2.1 破解方法一：

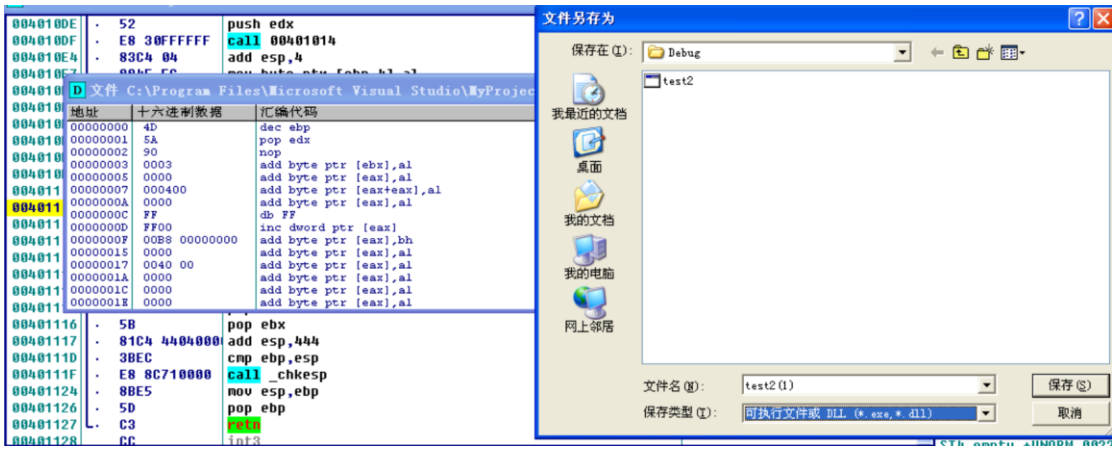
逻辑上，若输入密码错误则跳转到报错区域，正确则跳转到正确区域，核心分支判断为：

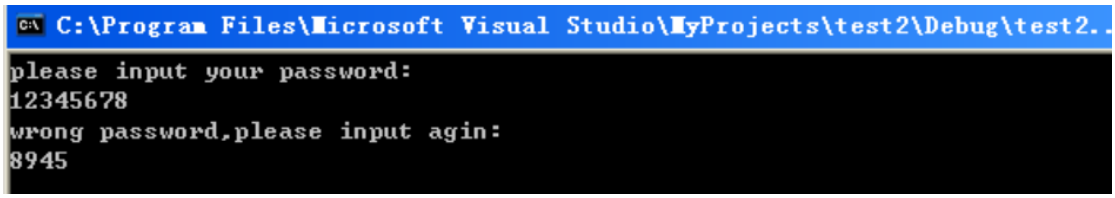



如果 jz 条件成立，则会跳转到 00401105 处，即显示错误密码分支语句，对 jz 取反 jnz，当输入错误密码会跳转到正确的分支中，修改 jz 部分代码：




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。

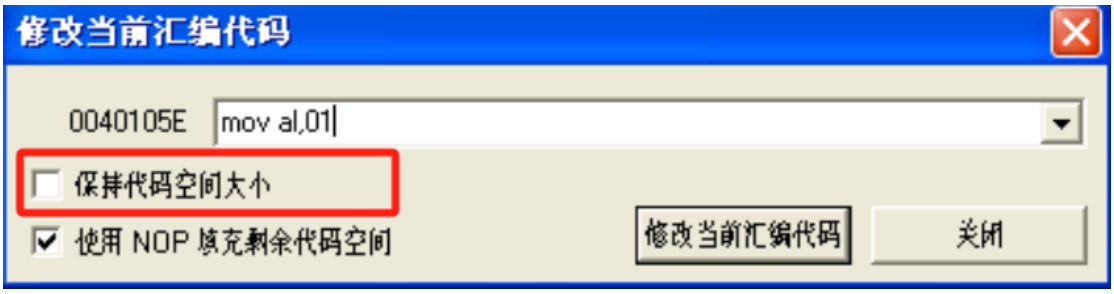


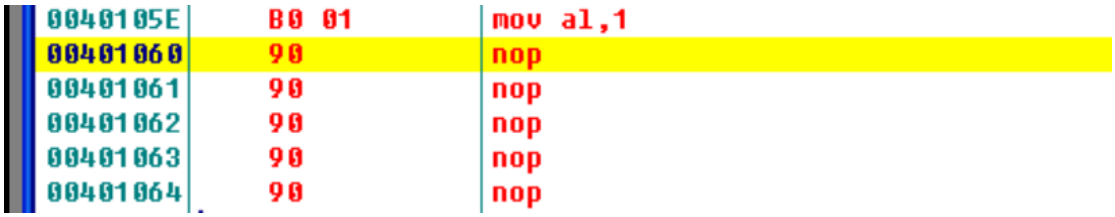



**2.1 破解方法二：**

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）。