《漏洞利用及渗透测试基础》实验报告

姓名：张洋　　学号：2111460　　班级：信安二班

实验名称：

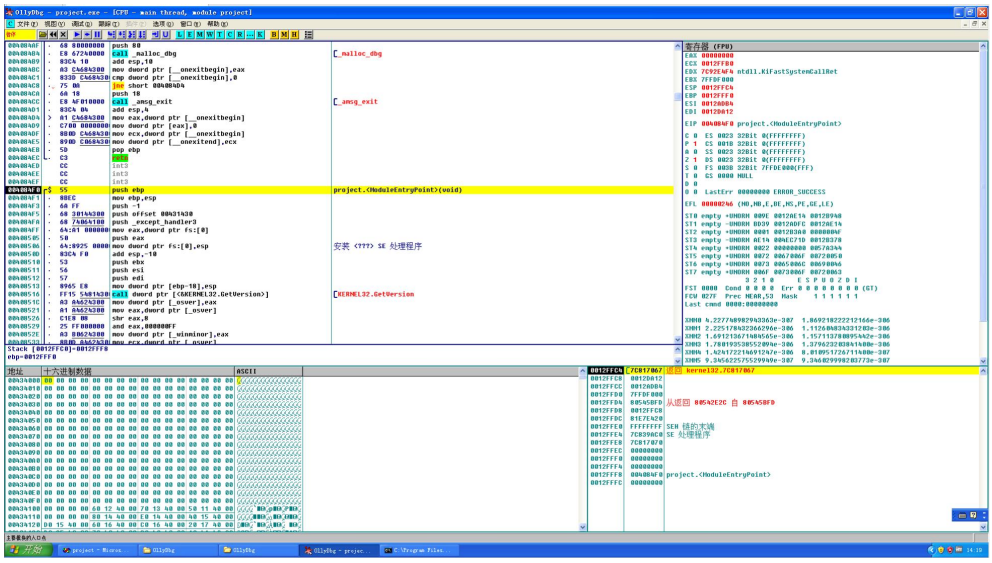
OLLYDBG 软件破解

实验要求：

1. 在 XP VC6 生成课本第三章软件破解的案例（DEBUG 模式）。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. 进入 VC 反汇编



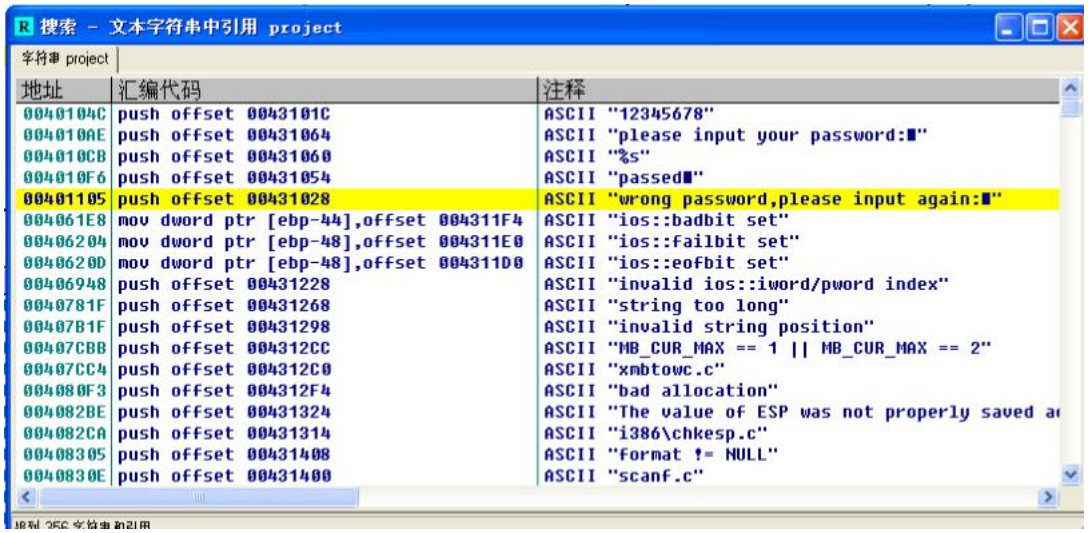
2. 获取 verifyPWD 函数对应 flag==0 的汇编代码，并对这些汇编代码进行解释。

```
00401030  >  55           push ebp                          project.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
0040106D  .  E8 3E720000  call _chkesp
00401072  .  8BE5         mov esp,ebp
00401074  .  5D           pop ebp
00401075  L. C3           retn
```

```
push ebp                        ;将主函数栈帧入栈
mov ebp,esp                     ;将主函数栈顶指针赋值给 ebp，调整栈帧的位置
sub esp,44                      ;将 esp 向低地址移动 44 字节
push ebx                        ;ebx 寄存器入栈
push esi                        ;esi 指针（源地址指针）入栈
push edi                        ;edi 指针（目的地址指针）入栈
lea edi,[ebp-44]                ;esp 地址赋值给 edi
mov ecx,11                      ;给 ecx 赋值 11，循环 11 次
mov eax,CCCCCCCC                ;用来初始化 verifyPwd 函数栈
rep stos dword ptr [edi]        ;将 eax 的值拷贝到 edi 指向的地址处
mov eax,dword ptr [ebp+8]       ;将基址寄存器 ebp+8 指向的值赋值给 eax
push eax                        ;eax 入栈
push offset 0043101C            ;字符串“12345678”入栈
call strcmp                     ;strcmp 函数
add esp,8                       ;eax 和字符串“12345678”出栈
mov dword ptr [ebp-4],eax       ;将 eax 的值赋值给基址寄存器 ebp-4 指向的单元值
xor eax,eax                     ;清零 eax 的值
cmp dword ptr [ebp-4],0         ;比较 0 和基址寄存器 ebp-4 对应的值
sete al                         ;相等则将 al 置为 1
pop edi                         ;edi 指针（目的地址指针）出栈
pop esi                         ;esi 指针（源地址指针）出栈
pop ebx                         ;ebx 寄存器出栈
add esp,44                      ;将 esp 返回栈帧的位置
cmp ebp,esp                     ;比较 ebp 和 esp 的值
call _chkesp                    ;检查 esp 是否等于函数调用前的值，不相等则报错
mov esp,ebp                     ;将 ebp 的值赋值给 esp
pop ebp                         ;ebp 出栈
retn                            ;返回主函数引用处
```

3. 用两种方法对生成的 DEBUG 程序进行破解

（1）找到字符串“wrong password,please input again:”对应的反汇编代码位置



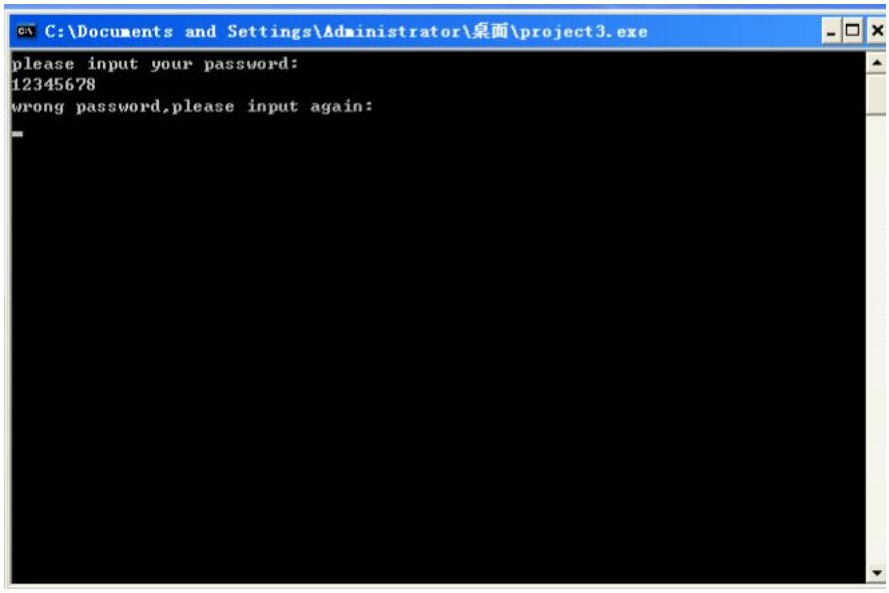
找到对应的代码后，分析反汇编代码。当输入的字符串与给定的字符串“12345678”相等时，输出“passed”，不相等则输出“wrong password,please input again:”

```
004010F4  .  74 0F          jz short 00401105
004010F6  .  68 54104300    push offset 00431054          ASCII "passed"
004010FB  .  E8 50720000    call printf                   printf
00401100  .  83C4 04        add esp,4
00401103  .  EB 0F          jmp short 00401114
00401105  >  68 28104300    push offset 00431028          ASCII "wrong password,please
0040110A  .  E8 41720000    call printf                   printf
```

此时，我们修改 jz short 00401105 为 jnz short 00401105，如下图。

```
jz short 00401105  ⟶  jnz short 00401105
```

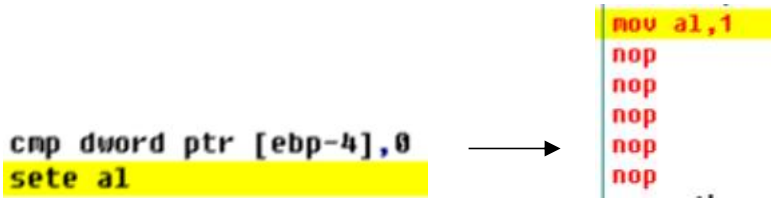
修改后运行程序，输入“12345678”显示“wrong password,please input again:”输入除“12345678”外的其他字符串都会显示通过。



（2）进入 verifyPWD 函数，通过上述对此函数的分析，当输入字符串与给定字符串相等时，al 的值置为 1。

```
00401030  >  55            push ebp                      project.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
0040106D  .  E8 3E720000   call _chkesp
00401072  .  8BE5          mov esp,ebp
00401074  .  5D            pop ebp
00401075  L. C3            retn
00401076     CC            int3
```

此时我们对 cmp 和 sete 两个语句进行修改，修改操作如下图所示。



不用进行第一步的比较操作，直接把 al 的值置为 1。修改后运行程序输入任何字符串都会 pass。

心得体会：

通过实验，对 verifyPWD 函数的汇编语言有了更深入的理解，通过解析每一条语句了解 ebp,esp 的变化情况。调用函数后的入栈顺序为：参数入栈，返回地址入栈，ebp 入栈，局部变量入栈。

通过对生成的 DEBUG 程序进行破解，修改反汇编代码，最终实现密码的破解。这个过程非常有趣，完成之后很有成就感，我也从中学习到了很多。