

《漏洞利用及渗透测试基础》实验报告

姓名： 许健 学号： 2013018 班级： 信安班

实验名称：

OLLYDBG 软件破解

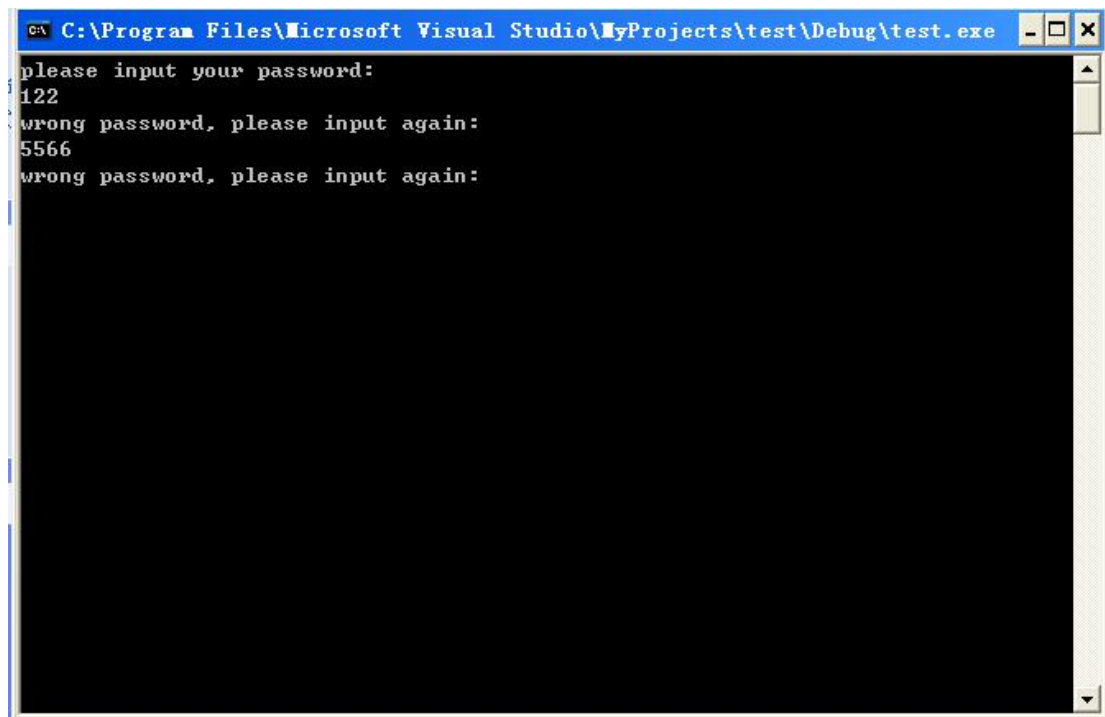
实验要求：

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：

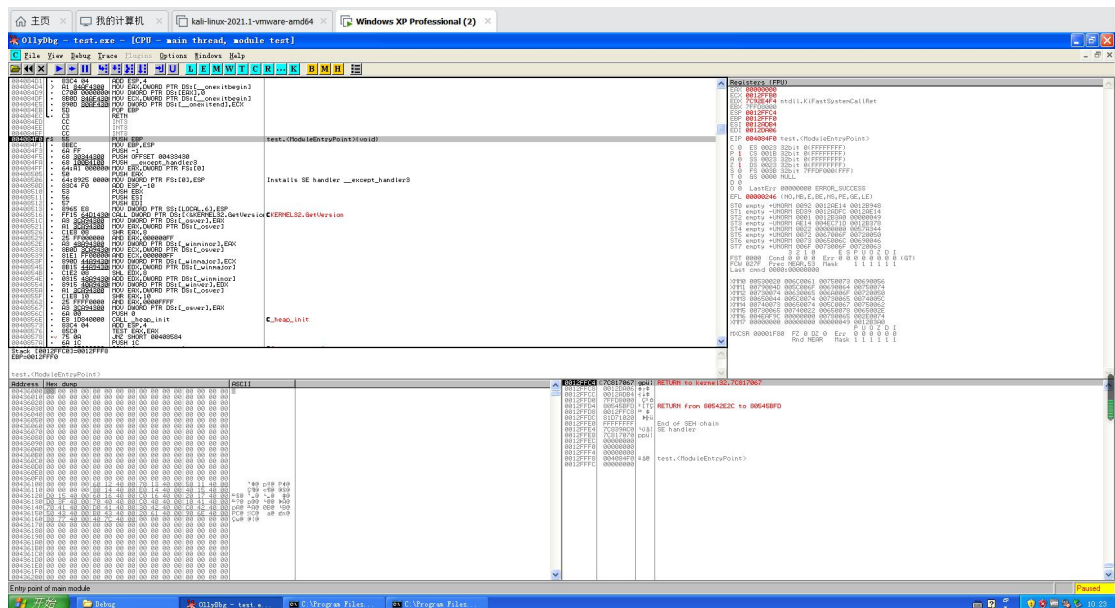
1. 使用 ollydbg 单步调试

编译源程序，获得可执行文件，运行程序结果如下：可以看到，当我们输入错误密码时，它会提示我们重新输入。



```
C:\Program Files\Microsoft Visual Studio\MyProjects\test\Debug\test.exe
please input your password:
122
wrong password, please input again:
5566
wrong password, please input again:
```

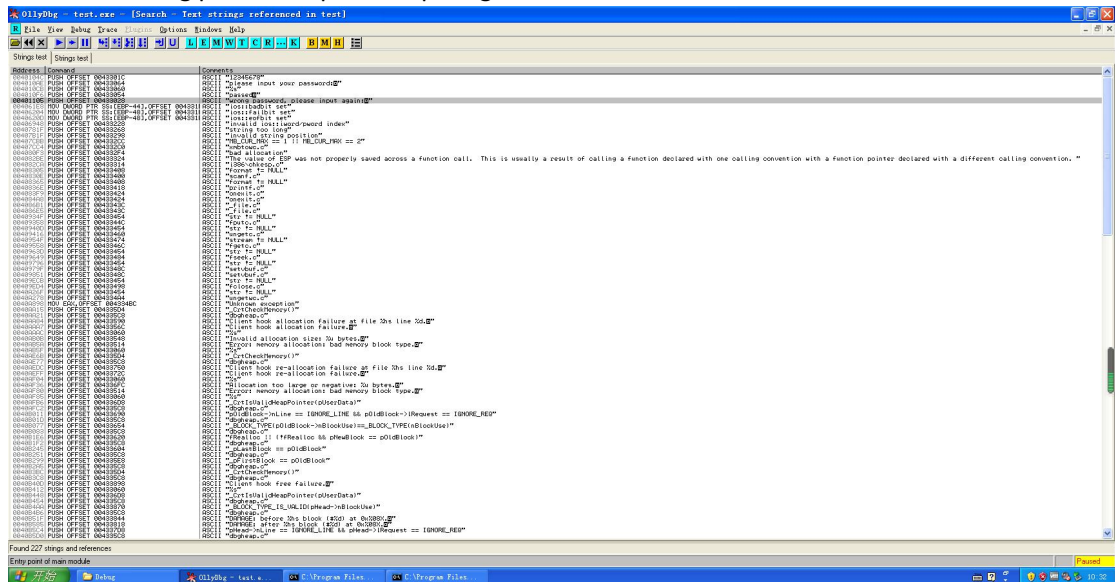
使用 `ollydbg` 运行我们的 `exe` 文件，进入调试界面，可以看到反汇编代码，以及程序的入口点。



我们需要观察关键信息，对关键信息定位得到关键分支语句

我们可以右键，查找所有引用的字符串信息

定位到“wrong password ,please input again”



双击找到与引用有关的位置，定位到反汇编的相应代码处，如图所示：



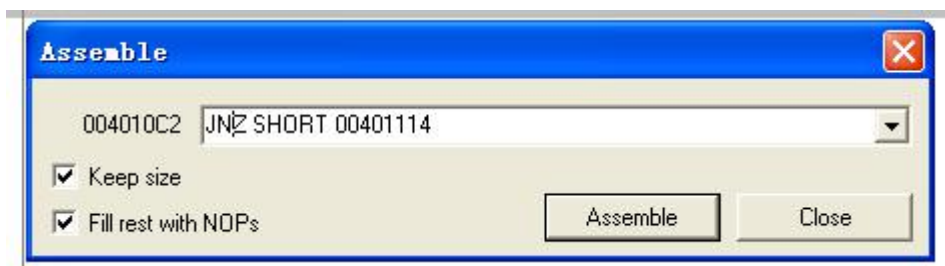
2. 破解方式一

我们可以在 ollydbg 中定位到核心代码，可以看到比较指令

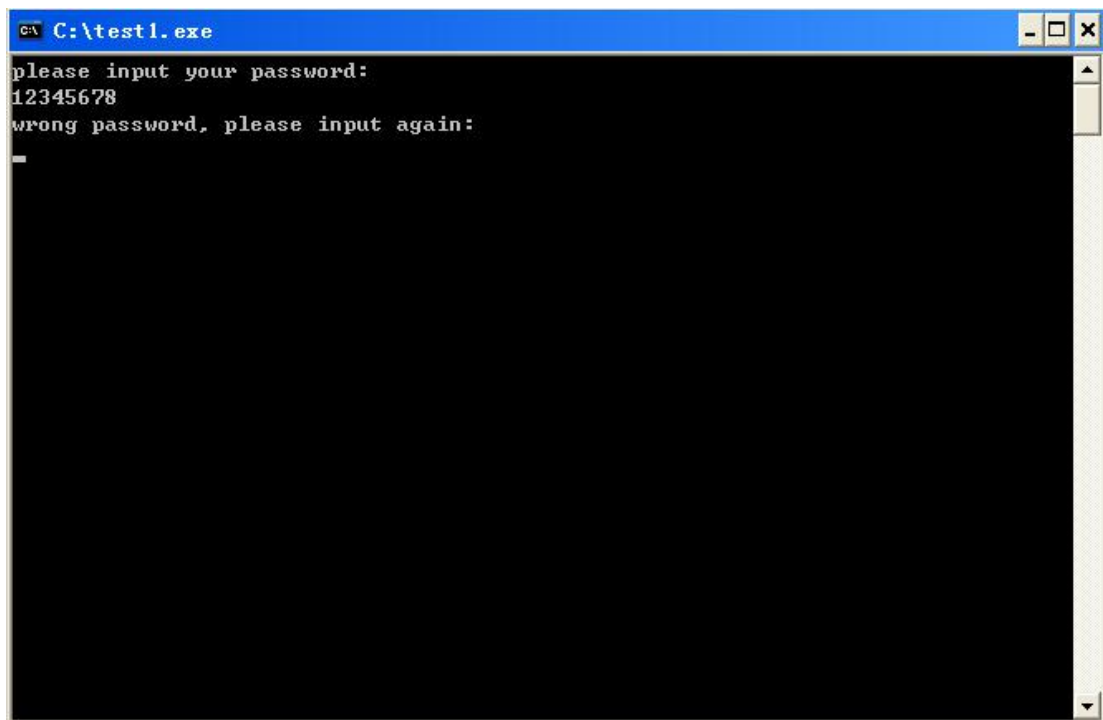
如果 jz 条件成立，则跳转到显示错误密码分支语句中。

004010C0	0503	TEST ECX,ECX	
004010C2	74 05	JZ SHORT 00401114	
004010C4	0000 FCFBFF	LEA ECX,[LOCAL.257]	
004010C6	51	PUSH ECX	
004010C8	68 00000000	PUSH OFFSET 00403060	ASCII "%s"
004010CA	CD	CALL scanf	C scanf
004010CC	8B04 05	MOV ESP,B	
004010CE	89FE FCFBFF	LEA EDI,[LOCAL.257]	
004010D0	59	PUSH EDI	
004010D2	68 00FFFFFF	CALL 00401014	C verifyPwd
004010D4	8B04 04	MOV ESP,4	
004010D6	8945 FC	MOV BYTE PTR DS:[LOCAL.13],AL	
004010D8	8945 FC	MOV EAX,WORD PTR SS:[LOCAL.13]	
004010DA	74 0F	JZ SHORT 00401105	ASCII "passed"
004010DC	68 00000000	PUSH OFFSET 00403064	C printf
004010DE	CD	CALL printf	
004010E0	8B04 04	MOV ESP,4	
004010E2	EB 0A	JMP SHORT 00401114	ASCII "wrong password, please input again:"
004010E4	68 00000000	PUSH OFFSET 00403068	

如果将 jz 该指令改为 jnz，则程序截然相反。输入了错误密码，将进入验证成功的分支中。



保存修改后的 exe 文件，运行结果如下，可以看到原来正确的密码变成错误的在，而输入错误的密码则跳转到正确的分支



3. 破解方式二

我们还可以通过修改函数的方式修改代码，验证命令使用的是 `verifyPwd` 函数，我们选择跟随跳转，可以得到关键代码

```
00401000  E9 86010000 JMP std::ctype<unsigned short>::id
00401005  E9 21020000 JMP std::ctype<unsigned short>::id
0040100A  E9 7C000000 JMP main
00401014  E8 17000000 JMP verifyPwd
00401015  CC          INT3
00401016  CC          INT3
00401017  CC          INT3
00401018  CC          INT3

CPU - main thread, module CrackPWD
00411B02  B9 33000000 mov ecx,33
00411B07  B8 CCCCCCCC mov eax,CCCCCCCC
00411B0C  F3:AB       rep stos dword ptr [edi]
00411B0E  8B45 08     mov eax,dword ptr [ebp+8]
00411B11  50         push eax
00411B12  68 90574100 push offset 00415790
00411B17  E8 B5F6FFFF call 004111D1
00411B1C  83C4 08     add esp,8
00411B1F  8945 F8     mov dword ptr [ebp-8],eax
00411B22  33C0       xor eax,eax
00411B24  837D F8 00  cmp dword ptr [ebp-8],0
00411B28  0F94C0     sete al
00411B2B  5F         pop edi
00411B2C  5E         pop esi
00411B2D  5B         pop ebx
00411B2E  81C4 CC0000 add esp,0CC
00411B34  3BEC       cmp ebp,esp
00411B36  E8 FB5FFFFF call 00411136
00411B3B  8BE5       mov esp,ebp
00411B3D  5D         pop ebp
00411B3E  C3         retm
00411B3F  CC          int3

ASCII "12345678"
Jump to MSVC80D.strcmp
_CRT_CompEsp
```

函数 `strcmp` 的返回值保存在 `eax` 寄存器中

下面对 `verifyPWD` 函数对应 `flag==0` 的汇编代码进行解释：

对于函数中的代码：

```
flag=strcmp(password, pwd);
return flag==0;
```

被解释为汇编语言：

```
Mov dword ptr [ebp-8], eax //将 strcmp 函数调用后的返回值赋值给变量 flag
Xor eax, eax //将 eax 的值清空
Cmp dword ptr [ebp-8], 0 //将 flag 的值与 0 进行比较，即 flag==0;
//注意 cmp 运算的结果只会影响一些状态寄存器的值
Sete al //sete 是根据状态寄存器的值，如果相等，则设置，如果不等，则不设置
```

注意，这里设置的状态寄存器的值则用于后面的比较，如果为 1 则正确执行，不为 1 则错误执行，所以我们只需要更改状态寄存器的值，使它恒为 1，那么程序输入密码无论对错，都会正确执行。

将 `cmp dword ptr [ebp-8]` 更改为：`mov al,01`

将 `sete al` 改为 `NOP`

更改后代码如图所示：

```
00401000  E9 86010000 JMP std::ctype<unsigned short>::id
00401005  E9 21020000 JMP std::ctype<unsigned short>::id
0040100A  E9 7C000000 JMP main
00401014  E8 17000000 JMP verifyPwd
00401015  CC          INT3
00401016  CC          INT3
00401017  CC          INT3
00401018  CC          INT3

CPU - main thread, module CrackPWD
00411B02  B9 33000000 mov ecx,33
00411B07  B8 CCCCCCCC mov eax,CCCCCCCC
00411B0C  F3:AB       rep stos dword ptr [edi]
00411B0E  8B45 08     mov eax,dword ptr [ebp+8]
00411B11  50         push eax
00411B12  68 90574100 push offset 00415790
00411B17  E8 B5F6FFFF call 004111D1
00411B1C  83C4 08     add esp,8
00411B1F  8945 F8     mov dword ptr [ebp-8],eax
00411B22  33C0       xor eax,eax
00411B24  837D F8 00  cmp dword ptr [ebp-8],0
00411B28  0F94C0     sete al
00411B2B  5F         pop edi
00411B2C  5E         pop esi
00411B2D  5B         pop ebx
00411B2E  81C4 CC0000 add esp,0CC
00411B34  3BEC       cmp ebp,esp
00411B36  E8 FB5FFFFF call 00411136
00411B3B  8BE5       mov esp,ebp
00411B3D  5D         pop ebp
00411B3E  C3         retm
00411B3F  CC          int3

ASCII "12345678"
_CRT_CompEsp
```

最后将更改后的 `exe` 文件保存运行，此时无论输入正确与否的密码，均将通过测试！

心得体会：

通过一个简单的密码实验，掌握使用 `ollydbg` 动态调试的一些基本技巧，以及修改程序关键代码的技术：我们可以更改比较指令的判断条件，或者更改某些状态寄存器的值。还有像查找关键字符串、跟随函数分析反汇编代码、分析 `JMP` 指令的能力得以提升。