

《软件安全》实验报告

姓名：蒋柄言 学号：2313546 班级：信息安全班

一、实验名称

OllyDbg 软件破解实验

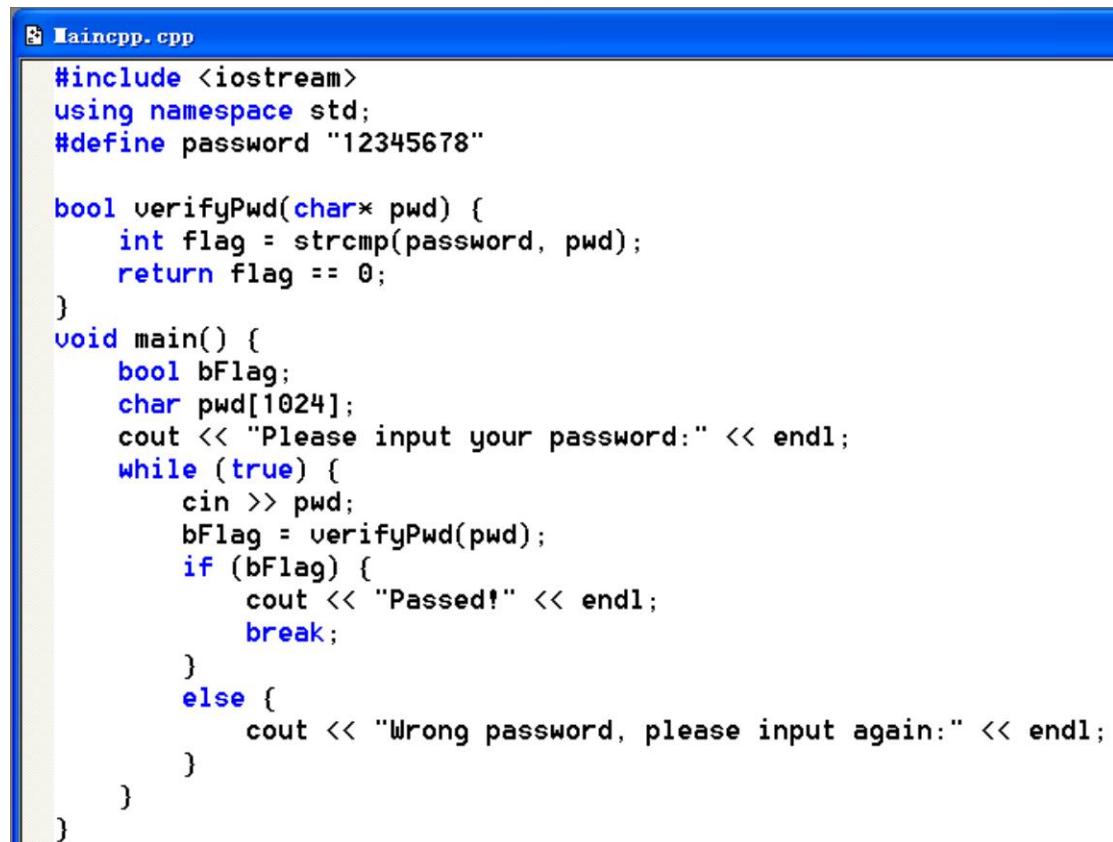
二、实验要求

1.请在 XP VC6 下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。

三、实验过程

首先打开 VC6，写一个简单的口令验证程序。

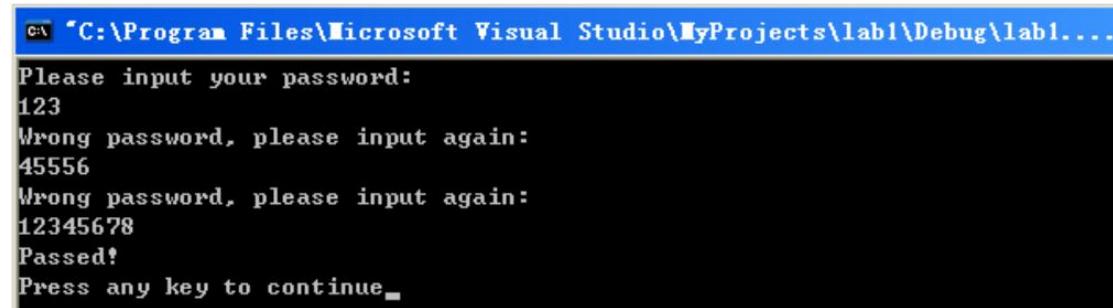


```
#include <iostream>
using namespace std;
#define password "12345678"

bool verifyPwd(char* pwd) {
    int flag = strcmp(password, pwd);
    return flag == 0;
}

void main() {
    bool bFlag;
    char pwd[1024];
    cout << "Please input your password:" << endl;
    while (true) {
        cin >> pwd;
        bFlag = verifyPwd(pwd);
        if (bFlag) {
            cout << "Passed!" << endl;
            break;
        }
        else {
            cout << "Wrong password, please input again:" << endl;
        }
    }
}
```

生成 Debug 模式的 exe 程序，运行如下：

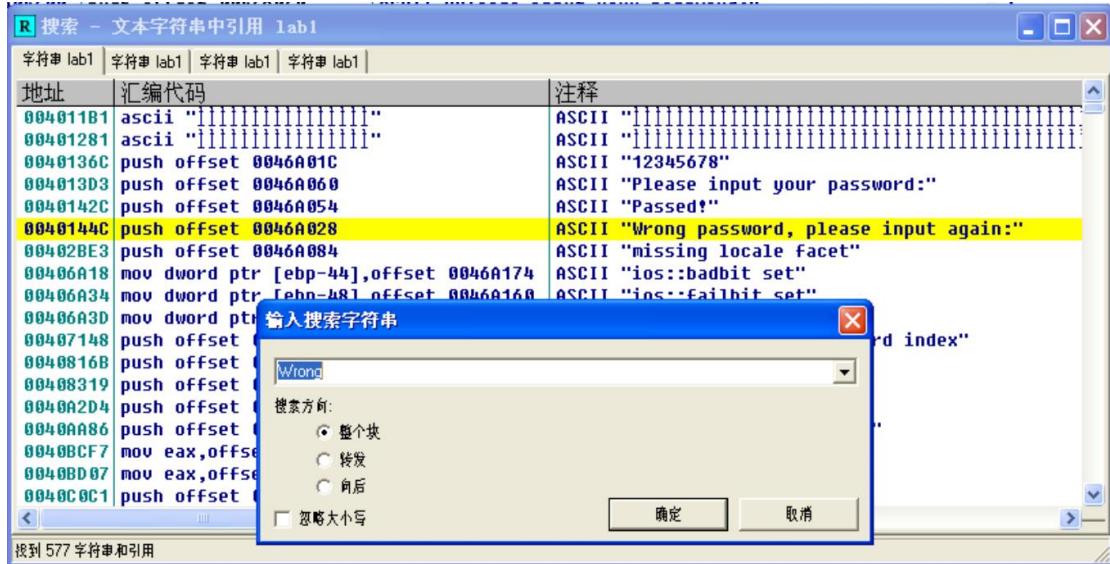


```
C:\Program Files\Microsoft Visual Studio\MyProjects\lab1\Debug\lab1...
Please input your password:
123
Wrong password, please input again:
45556
Wrong password, please input again:
12345678
Passed!
Press any key to continue...
```

接着我们使用两种方式进行破解。

(1) 修改跳转条件

1.点击右键，选择“查找->所有引用的字符串”，输入“Wrong”，即可查询到提示“Wrong password, please input again:”。点击地址即跳转到 0x0040144C 处。



00401409	. 8D95 FCFBFFFF	lea edx,[ebp-404] push edx call 00401177 add esp,4	[verifyPwd]
0040140F	. 52		
00401410	. E8 62FDFFFF	add esp,4	
00401415	. 83C4 04	mov byte ptr [ebp-4],al	
00401418	. 8845 FC	mov eax,dword ptr [ebp-4]	
0040141B	. 8B45 FC	and eax,000000FF	
0040141E	. 25 FF000000	test eax,eax	
00401423	. 85C0	jz short 00401447	
00401425	. 74 20	push 0040107D push offset 0046A054 push offset std::cout call 0040117C add esp,8	ASCII "Passed!"
00401427	. 68 7D104000		
0040142C	. 68 54A04600		
00401431	. 68 E8574700		
00401436	. E8 41FDFFFF	[std::operator<<	
0040143B	. 83C4 08	mov ecx,eax call 0040112C jmp short 00401467	[std::basic_ostream<char, std::char_traits<ch
0040143E	. 8BC8	push 0040107D push offset 0046A028	
00401440	. E8 E7FCFFFF	push offset std::cout call 0040117C add esp,8	ASCII "Wrong password, please input again!"
00401445	. EB 20		
00401447	> . 68 7D104000		
0040144C	. 68 28A04600		
00401451	. 68 E8574700		
00401456	. E8 21FDFFFF		
0040145B	. 83C4 08		

2. 观察以上汇编代码，发现核心判断分支在以下语句：

00401423 test eax, eax ; 计算 eax & eax，并设置标志位
00401425 jz short 00401447 ; 若 eax == 0 (ZF=1)，跳转到 00401447 地址处

① test 指令对两个操作数执行逻辑与 (AND) 操作，但不保存结果，仅根据结果设置标志位。当操作数为同一个寄存器（如 test eax, eax）时，等效于检查该寄存器的值是否为 0。这里的意思就是若 eax=0，结果为 0，零标志位 (ZF) 被置为 1；若 eax≠0，结果为非零值，零标志位 (ZF) 被置为 0。与 cmp 指令不同，test 指令不修改标志 CF/OF。

② jz 指令全称 Jump if Zero（零标志位为 1 时跳转）。当零标志位 (ZF) 为 1 时，执行跳转。short 表示跳转目标地址是短跳转，即目标地址距离当前指令的偏移量在 -128 到 +127 字节范围内。

这两行语句的作用是如果 bFlag 为 true (即口令输入正确)，则按顺序执行，输出 “Passed!”；如果 bFlag 为 false (即口令输入错误)，则跳转，输出 “Wrong password, please input again:”。

3. 修改指令：将 jz 修改为 jnz (不等于 0 时跳转，即零标志位为 0 时跳转)。这样一来逻辑就完全相反了，输入错误口令则会输出 “Passed!”，输入正确口令则会输出 “Wrong

password, please input again:”。



4.保存文件：此时并没有真正修改二进制文件中的有关代码，如果想要修改二进制文件中的代码，需要在反汇编窗口点击右键，选择“编辑->复制当前修改到可执行文件”。然后在弹出的代码框中再次点击右键，选择“保存文件”，保存类型为“可执行文件或 DLL”，现在保存后的可执行文件将是破解后的文件。

5.运行修改后的 exe 文件，发现只要输入任意一个错误口令，即会提示“Passed!”。

```
C:\Program Files\Microsoft Visual Studio\MyProjects\lab1\Debug>
Please input your password:
12345678
Wrong password, please input again:
879456
Passed!
Press any key to continue...
```

(2) 修改函数返回值

1.根据观察，验证口令使用的是 verifyPwd 函数，右键点击“跟随”重复两次，进入函数内部。

```
00401410 | . E8 62FDFFFF | call 00401177 [verifyPwd]
00401177 $ .E9 D4010000 | jmp verifyPwd

00401350 | $ 55           | push ebp
00401351 | . 8BEC          | mov ebp,esp
00401353 | . 83EC 44        | sub esp,44
00401356 | . 53             | push ebx
00401357 | . 56             | push esi
00401358 | . 57             | push edi
00401359 | . 8D7D BC        | lea edi,[ebp-44]
0040135C | . B9 11000000 | mov ecx,11
00401361 | . B8 CCCCCCCC | mov eax,CCCCCC
00401366 | . F3:AB          | rep stos dword ptr [edi]
00401368 | . 8B45 08          | mov eax,dword ptr [ebp+8]
0040136B | . 50             | push eax
0040136C | . 68 1CA04600 | push offset 0046A01C      ASCII "12345678"
00401371 | . E8 0AE70100 | call strcmp
00401376 | . 83C4 08          | add esp,8
00401379 | . 8945 FC          | mov dword ptr [ebp-4],eax
0040137C | . 33C0             | xor eax,eax
0040137E | . 837D FC 00        | cmp dword ptr [ebp-4],0
00401382 | . 0F94C0          | sete al
00401385 | . 5F             | pop edi
00401386 | . 5E             | pop esi
00401387 | . 5B             | pop ebx
00401388 | . 83C4 44          | add esp,44
0040138B | . 3BEC          | cmp ebp,esp
0040138D | . E8 7EE70100 | call _chkesp
00401392 | . 8BE5            | mov esp,ebp
00401394 | . 5D             | pop ebp
00401395 | L. C3             | ret
```

2. 函数的返回值会保存在 eax 寄存器中，这里返回 bool 值，保存在低 8 位的 al 中。在返回指令 ret 之前有如下代码：

```
00401379 mov dword ptr [ebp-4], eax ; 将 eax 的值赋给 ebp-4 位置  
0040137C xor eax, eax ; 清空 eax 的值（置 0）  
0040137E cmp dword ptr [ebp-4], 0 ; 比较 ebp-4 位置的值和 0，即 flag == 0  
00401382 sete al ; 若相等则将 al 的值赋成 01
```

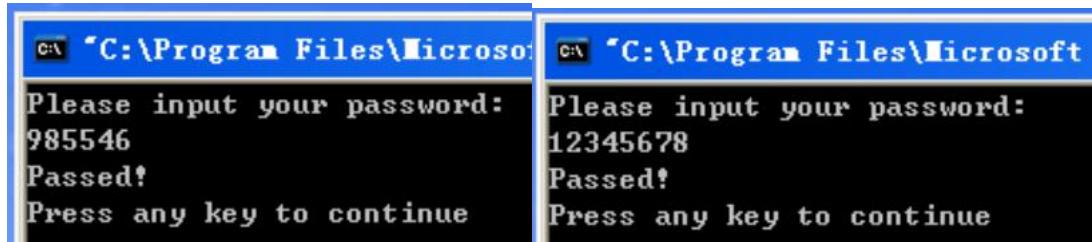
sete 全称“Set if Equal”（相等时设置），是一条条件设置指令，当零标志位 ZF = 1 时将目标操作数（此处为 al 寄存器）设置为 1，否则设为 0。

想要破解口令，只需要使函数的返回值永远为 true（即将 al 人为设置成 01）。则将指令修改为如下（取消勾选“保持代码空间大小”）：

```
00401379 mov dword ptr [ebp-4], eax ; 将 eax 的值赋给 ebp-4 位置  
0040137C xor eax, eax ; 清空 eax 的值（置 0）  
0040137E mov al, 1 ; 强制将 al 的值设为 1  
00401380 nop  
00401381 nop  
00401382 nop ; 只需修改后两行代码  
00401383 nop ; 多余空间会自动用 nop 填充  
00401384 nop
```

00401379	. 8945 FC	mov dword ptr [ebp-4],eax
0040137C	. 33C0	xor eax, eax
0040137E	B8 01	mov al, 1
00401380	90	nop
00401381	90	nop
00401382	90	nop
00401383	90	nop
00401384	90	nop

3. 按同样的方法保存文件，运行修改后的 exe 文件，发现无论输入什么口令，都会提示“Passed!”。破解成功。



四、心得体会

通过本次实验，我理解了简单的逆向工程的基本方法。使用 OllyDbg 调试时，直观观察到条件跳转与寄存器操作对程序逻辑的影响，强化了对底层执行机制的认识。此次实践不仅提升了调试技能，更让我认识到软件安全防护的必要性，以及合法、合规使用逆向技术的责任感。

温馨提示： 初始化字符串 pwd 的时候不能写成 `char pwd[1024]{};`，因为 VC6 是 1998 年的编译器，不支持 C++11 的列表初始化语法（如使用 {} 初始化数组）。