

《软件安全》实验报告

姓名：王昱 学号：2212046 班级：信息安全班

实验名称：

OLLYDBG 软件破解

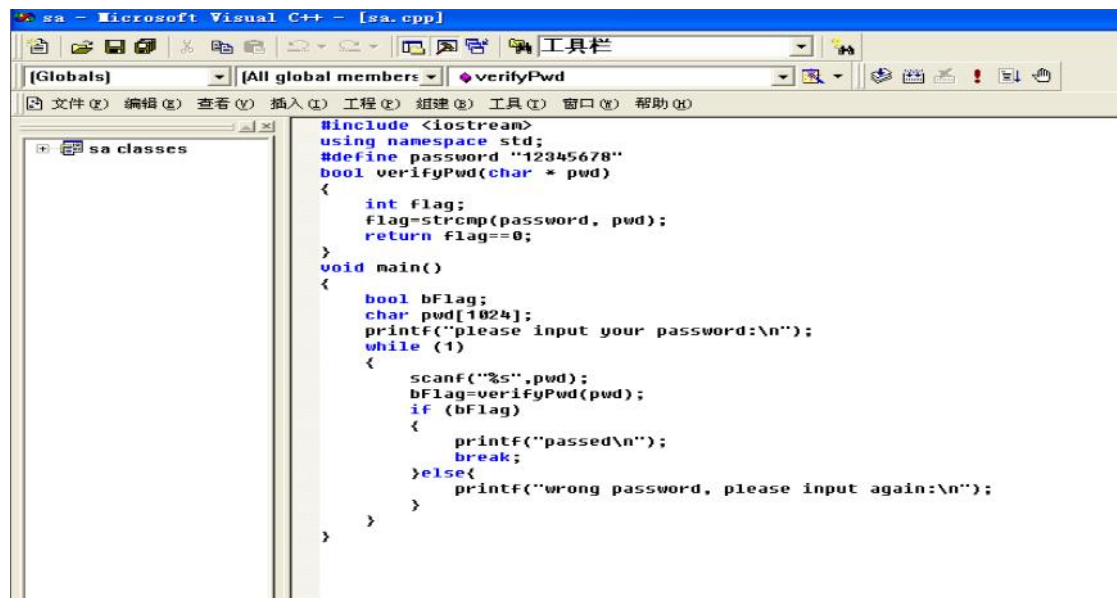
实验要求：

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：

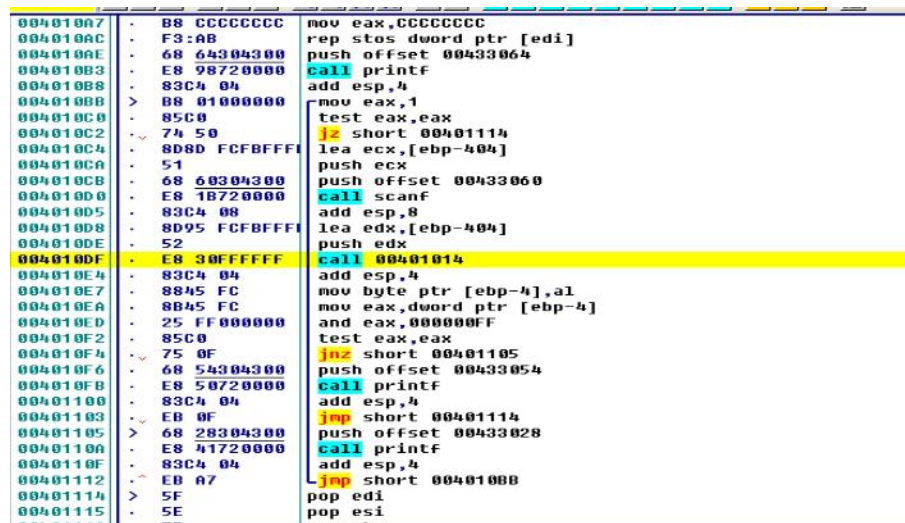
1. verifyPwd 汇编代码解释

使用 XP VC6，进入 debug 模式生成可执行文件



```
#include <iostream>
using namespace std;
#define password "12345678"
bool verifyPwd(char * pwd)
{
    int flag;
    flag=strcmp(password, pwd);
    return flag==0;
}
void main()
{
    bool bFlag;
    char pwd[1024];
    printf("please input your password:\n");
    while (1)
    {
        scanf("%s",pwd);
        bFlag=verifyPwd(pwd);
        if (bFlag)
        {
            printf("passed\n");
            break;
        }else{
            printf("wrong password, please input again:\n");
        }
    }
}
```

使用 OllyDbg 打开可执行文件



004010A7	B8 CCCCCC	mov eax,CCCCCCC
004010AC	F3:AB	rep stos dword ptr [edi]
004010AE	68 64304300	push offset 00433064
004010B3	E8 98720000	call printf
004010B8	83C4 04	add esp,4
004010BB	B8 01000000	mov eax,1
004010C0	85C0	test eax,eax
004010C2	74 50	jz short 00401114
004010C4	8D8D FCFBFF	lea ecx,[ebp-404]
004010CA	51	push ecx
004010CB	68 60304300	push offset 00433060
004010D0	E8 1B720000	call scanf
004010D5	83C4 08	add esp,8
004010D8	8D95 FCFBFF	lea edx,[ebp-404]
004010DE	52	push edx
004010DF	E8 30FFFF	call 00401014
004010E4	83C4 04	add esp,4
004010E7	8B45 FC	mov byte ptr [ebp-4],al
004010EA	8B45 FC	mov eax,dword ptr [ebp-4]
004010ED	25 FF000000	and eax,000000FF
004010F2	85C0	test eax,eax
004010F4	75 0F	jnz short 00401105
004010F6	68 54304300	push offset 00433054
004010FB	E8 50720000	call printf
00401100	83C4 04	add esp,4
00401103	EB 0F	jmp short 00401114
00401105	68 28304300	push offset 00433028
0040110A	E8 41720000	call printf
0040110F	83C4 04	add esp,4
00401112	EB A7	jmp short 00401088
00401114	5F	pop edi
00401115	5E	pop esi
00401116	5D	pop ebx

获取 verifyPwd 函数的反汇编代码

55	push ebp
8BEC	mov ebp,esp
83EC 44	sub esp,44
53	push ebx
56	push esi
57	push edi
8D7D BC	lea edi,[ebp-44]
B9 11000000	mov ecx,11
B8 CCCCCCCC	mov eax,CCCCCCCC
F3:AB	rep stos dword ptr [edi]
8B45 08	mov eax,dword ptr [ebp+8]
50	push eax
68 1C304300	push offset 0043301C
E8 CA710000	call strcmp
83C4 08	add esp,8
8945 FC	mov dword ptr [ebp-4],eax
33C0	xor eax,eax
837D FC 00	cmp dword ptr [ebp-4],0
0F94C0	sete al
5F	pop edi
5E	pop esi
5B	pop ebx
83C4 44	add esp,44
3BEC	cmp ebp,esp
E8 3E720000	call _chkesp
8BE5	mov esp,ebp
5D	pop ebp
C3	ret

55	push ebp
8BEC	mov ebp,esp
83EC 44	sub esp,44
53	push ebx
56	push esi
57	push edi
8D7D BC	lea edi,[ebp-44]
B9 11000000	mov ecx,11
B8 CCCCCCCC	mov eax,CCCCCCCC
F3:AB	rep stos dword ptr [edi]

函数首先进行返回函数入栈，栈帧调整，开辟该函数的栈帧，将可能用到的寄存器压入栈中，随后对局部变量初始化。

```
mov eax,dword ptr [ebp+8]
push eax
push offset 0043301C
call strcmp
add esp,8
mov dword ptr [ebp-4],eax
xor eax,eax
cmp dword ptr [ebp-4],0
sete al
```

这部分是函数实现的主要内容：

第一行将输入的 password 的值赋给寄存器 eax；

第二行将寄存器 eax 压入栈中；

第三行把正确的密码压入栈中；

第四行调用 strcmp 函数，比对两个字符串的值是否相等，如果相等的话，eax 的值为 0，否则为 00000001；

第五行调整栈的大小；

第六行把 eax 的值赋值给 flag 变量；

第七行通过 xor 来把 eax 清零；

第八行调用 cmp 比较 flag 是否为 0，如果为 0，ZF 标志位就会为 1，否则为 0；

第九行调用 sete，如果 ZF 为 1，就把 al 设置为 1，否则就把 al 设置为 0。

```

5F      pop edi
5E      pop esi
5B      pop ebx
83C4 44  add esp,44
3BEC    cmp ebp,esp
E8 3E720000 call _chkesp
8BE5    mov esp,ebp
5D      pop ebp
C3      ret

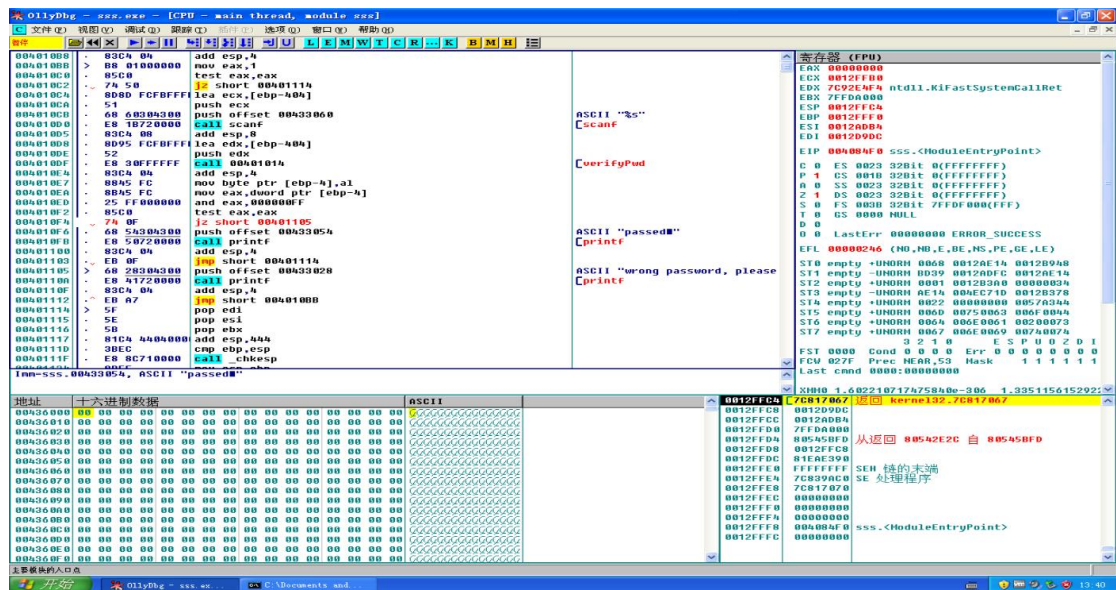
```

这部分进行了栈帧的恢复，返回到函数调用前的状态。

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。

(1) 第一种方法：

首先通过查找字符串“wrong”定位到代码部分：



分析关键代码可知，是由 jz short 00401105 跳转到的，如果 flag 为 0（不相等），就会跳转到输出错误的部分，如果 flag 为 1，那么就不会跳转。

```

004010F4 74 0F jz short 00401105
004010F6 68 54304300 push offset 00433054
004010F8 E8 50720000 call printf
00401100 83C4 04 add esp,4
00401103 EB 0F jmp short 00401114
00401105 68 28304300 push offset 00433028
0040110A E8 41720000 call printf
0040110F 83C4 04 add esp,4

```

所以，可以将逻辑取反，即 flag 为 0 的时候不跳转，而 flag 为 1 的时候跳转。将 jz 改为 jnz。这样密码正确的时候跳转，错误的时候不跳转。



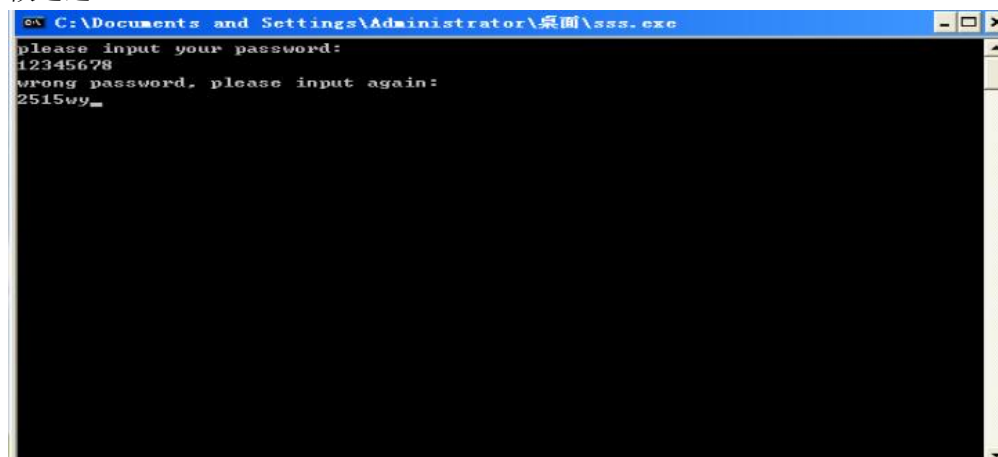
修改成功！

```

75 0F jnz short 00401105

```

保存修改之后，执行修改后的可执行，发现密码正确的时候输出“wrong!”，密码错误的时候通过。



(2) 第二种方法

通过跟随进入函数内部，可以发现，函数对应 sete 语句，当密码正确，al 设置为 1，否则就设置为 0，所以第二种思路是取修改 verify 函数，想办法使其永远返回的都是真值，即 eax 中的值不为零。我们定位到以下代码。

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	. C3	ret

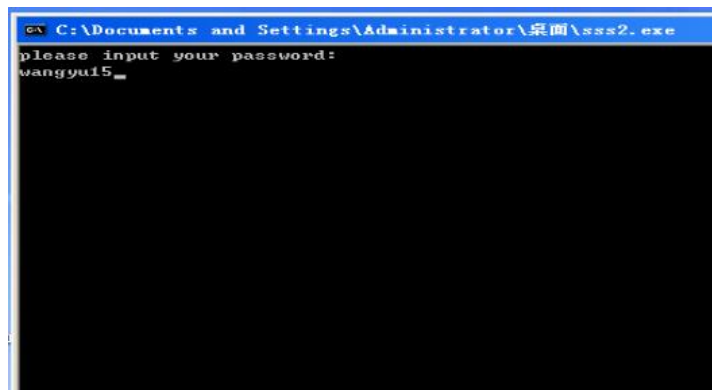
通过把 cmp 指令修改为 mov 指令，将 eax 中的值直接设置为 1，同时考虑到 sete 指令会根据 ZF 的状态设置 eax 的低 8 位，为了消除 sete 的影响，将其他指令 nop 掉。

```

add esp,8
mov dword ptr [ebp-4],eax
xor eax,eax
mov al,1
nop
nop
nop
nop
nop
pop edi
pop esi
pop ebx

```

修改后无论输入任何值都会通过。



心得体会：

1. 通过本次实验，首先对 OllyDbg 有了初步的了解和认识，可以通过软件进行简单的反汇编和密码破解。
2. 对汇编语言有了更深刻的认识。对其内部实现逻辑以及汇编的各种指令有了更加深刻的了解与认识。
3. 实验中的破解方法大大拓展了我的思维，也给了我很多启迪。