

实验1逆向破解初步尝试

实验1逆向破解初步尝试

一、实验题目

1.实验目标

2.实验环境

二、实验报告内容（包括实验题和附加题）

（1）实验题

测试

进入ida

进入x64dbg

修改汇编代码

程序运行图

（2）测试结论

（3）附加题

测试

进入x64dbg

方法1: 直接修改jne

方法2: 修改test

方法3: 修改验证密码逻辑

一、实验题目

1.实验目标

通过实验的过程掌握基本的二进制文件分析能力，并能够通过工具完成逆向破解的过程，从而在该过程中初步体会软件安全和其所遭受的威胁。

2.实验环境

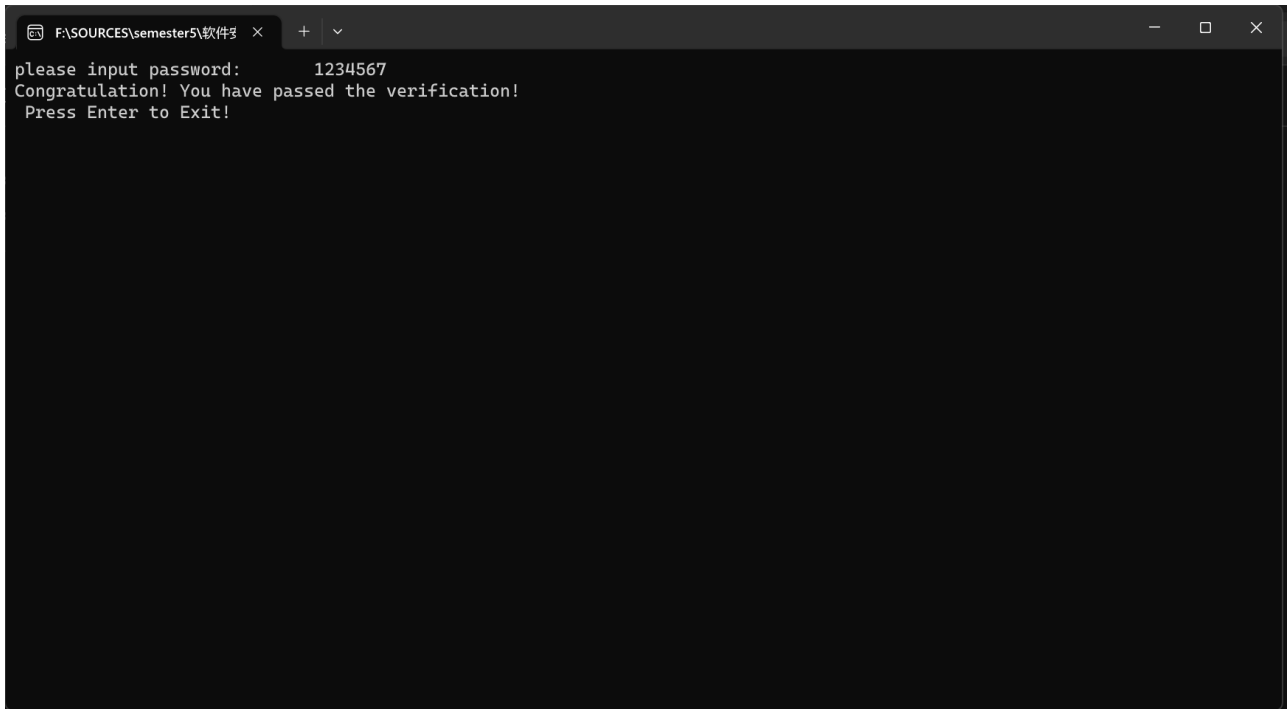
系统：Windows 2000~Windows 11皆可，也可使用wsl

工具：Ollydbg，x96dbg等

二、实验报告内容（包括实验题和附加题）

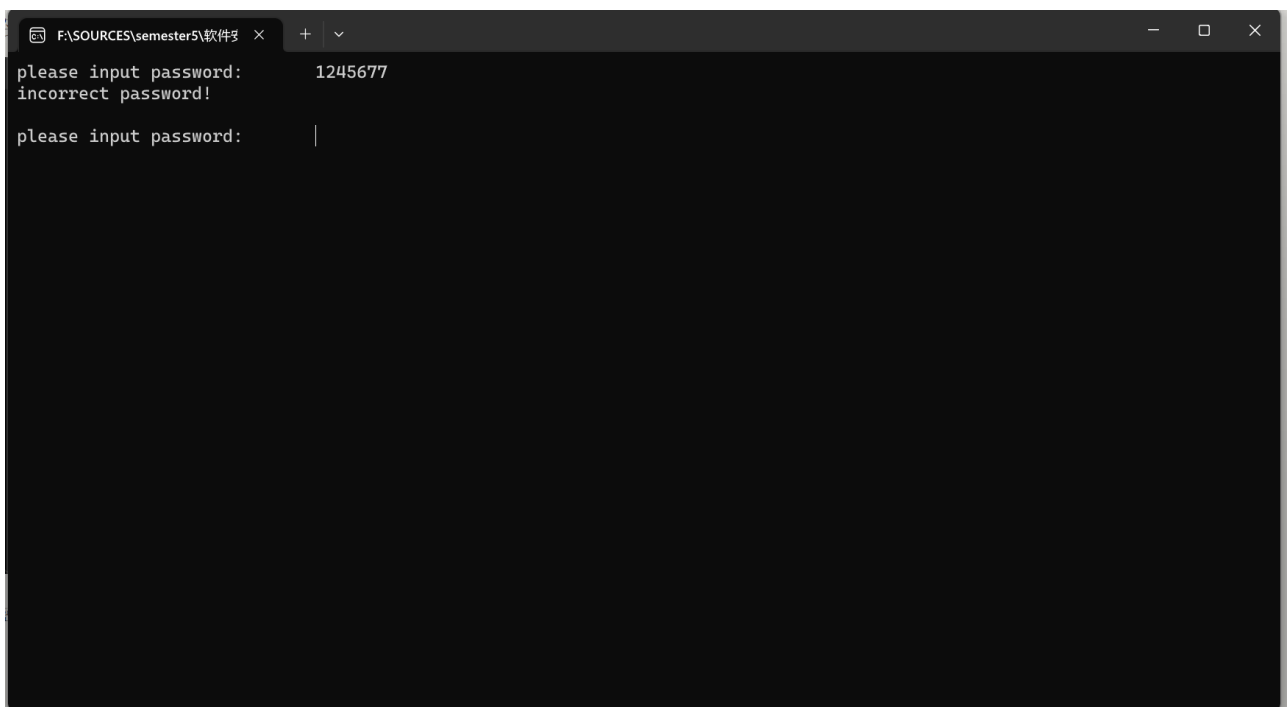
（1）实验题

测试



```
F:\SOURCES\semester5\软件3 >
please input password: 1234567
Congratulation! You have passed the verification!
Press Enter to Exit!
```

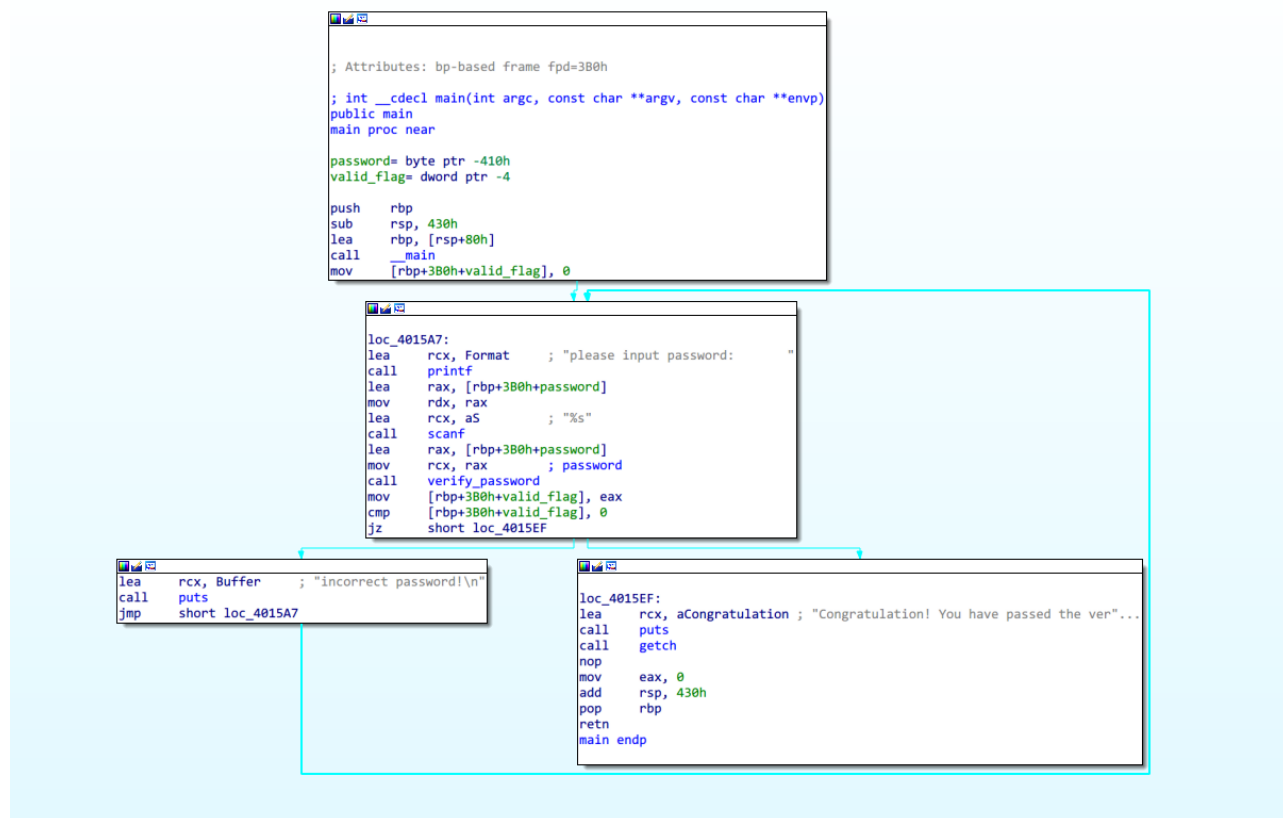
密码正确 返回对应结果



```
F:\SOURCES\semester5\软件3 >
please input password: 1245677
incorrect password!
please input password: |
```

密码错误返回错误结果

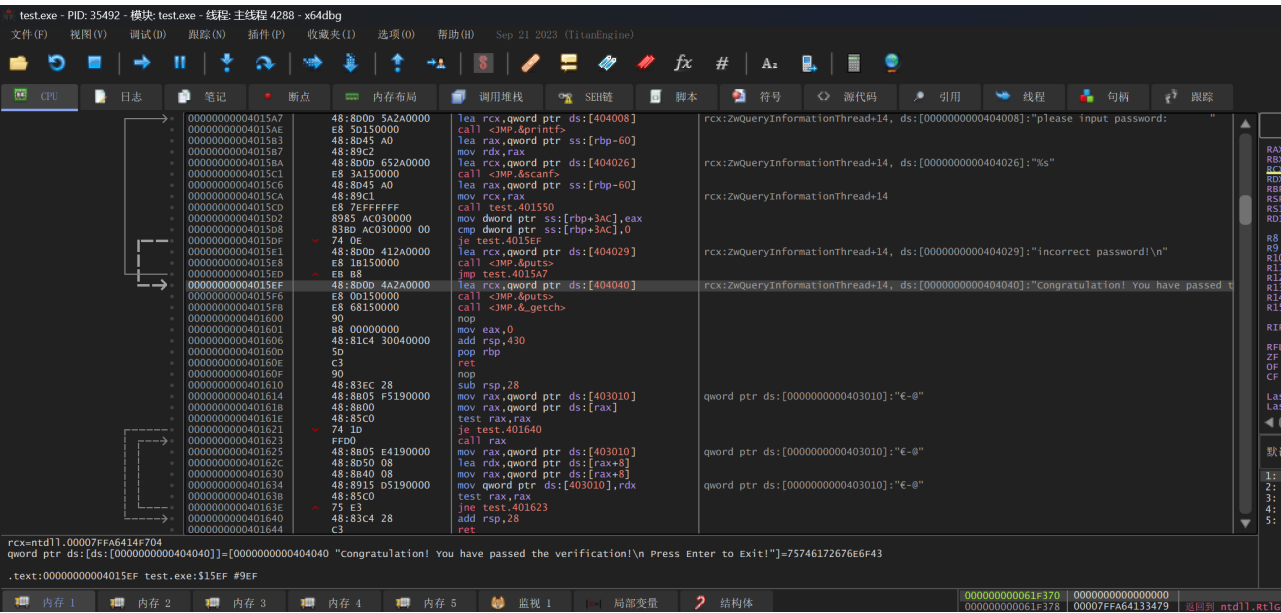
进入ida



```
IDA View-A | Pseudocode-A | Hex View-1 | Structures | Er
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char password[1024]; // [rsp+20h] [rbp-60h] BYREF
4     int valid_flag; // [rsp+42Ch] [rbp+3ACh]
5
6     _main(argc, argv, envp);
7     valid_flag = 0;
8     while ( 1 )
9     {
10        printf("please input password:      ");
11        scanf("%s", password);
12        valid_flag = verify_password(password);
13        if ( !valid_flag )
14            break;
15        puts("incorrect password!\n");
16    }
17    puts("Congratulation! You have passed the verification!\n Press Enter to Exit!");
18    getch();
19    return 0;
20 }
```

可以看到这是一个简单的c语言代码，通过输入的字符串与1234567（在verify_password中）对比，根据对比的返回结果修改flag的值，正确则返回成功，错误则重新尝试。

进入x64dbg

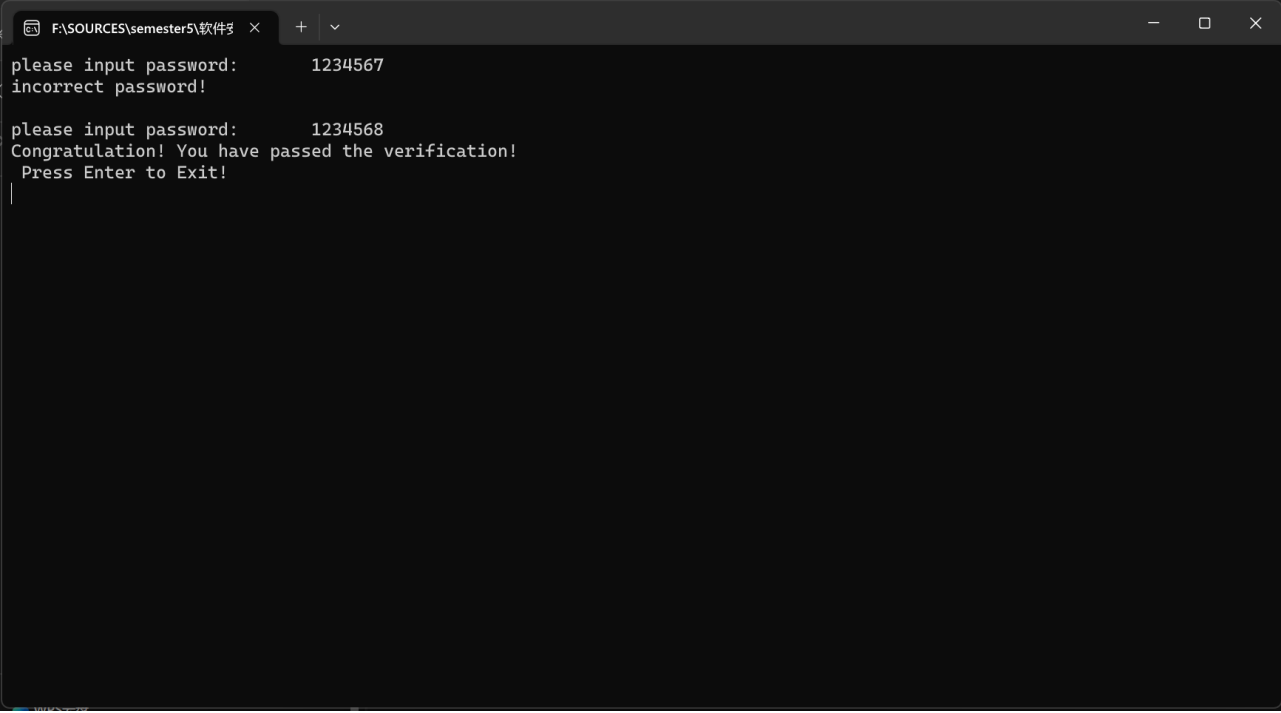


进入字符串页面搜索 Congralation! You....

找到跳转关键节点，可以看到是从0004015DF（jz 0x000000000004015EF）跳转到这里，所以我们知道0004015DF是关键跳转判断。

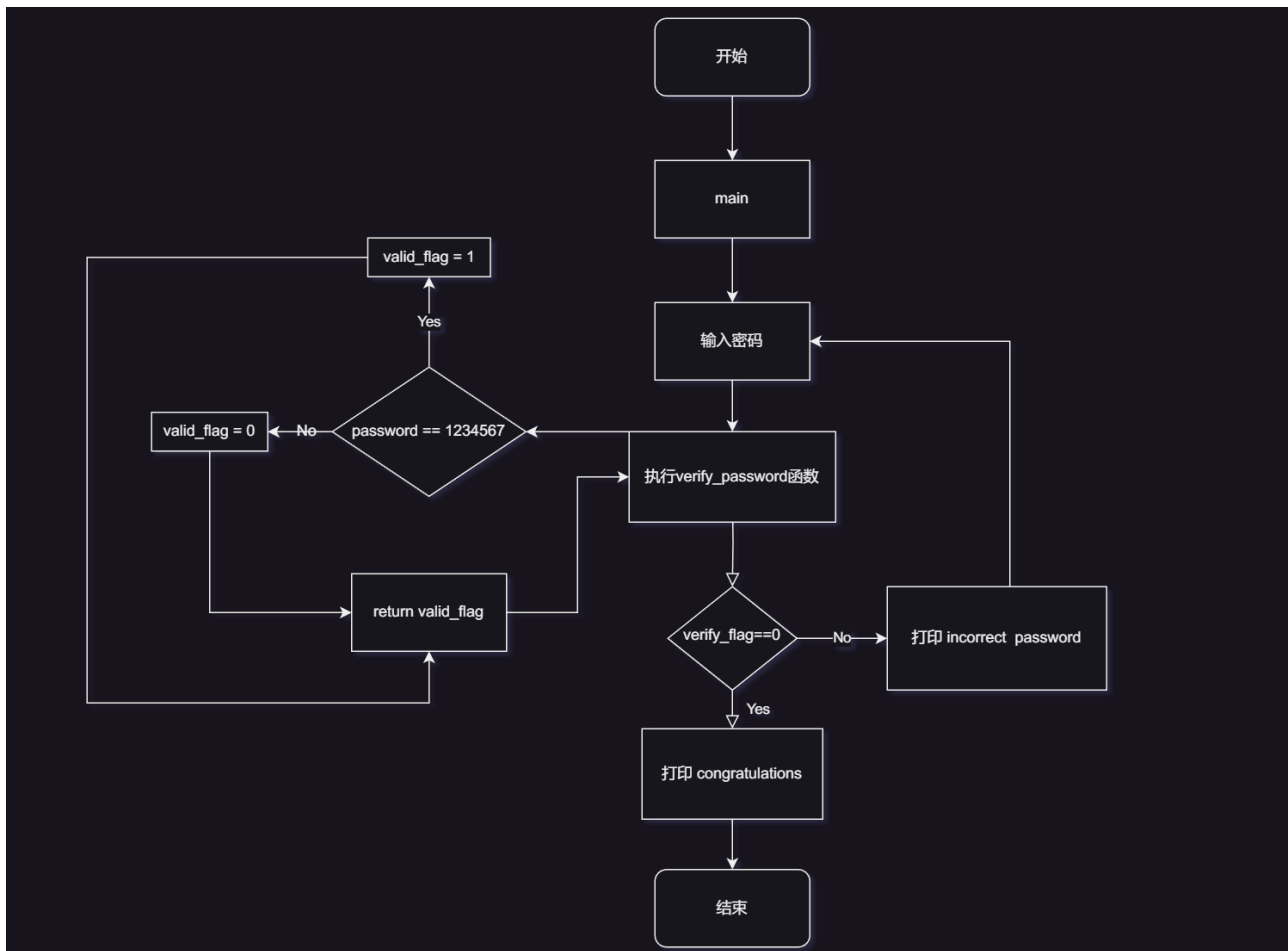
修改汇编代码

修改jz 0x000000000004015EF 为jnz 0x000000000004015EF ,并ctrl+p保存为test1.exe。然后打开



这时候我们发现原来输入正确的密码返回错误，随便输入错误的代码反而返回正确。根本原因在于判断逻辑被修改。

程序运行图

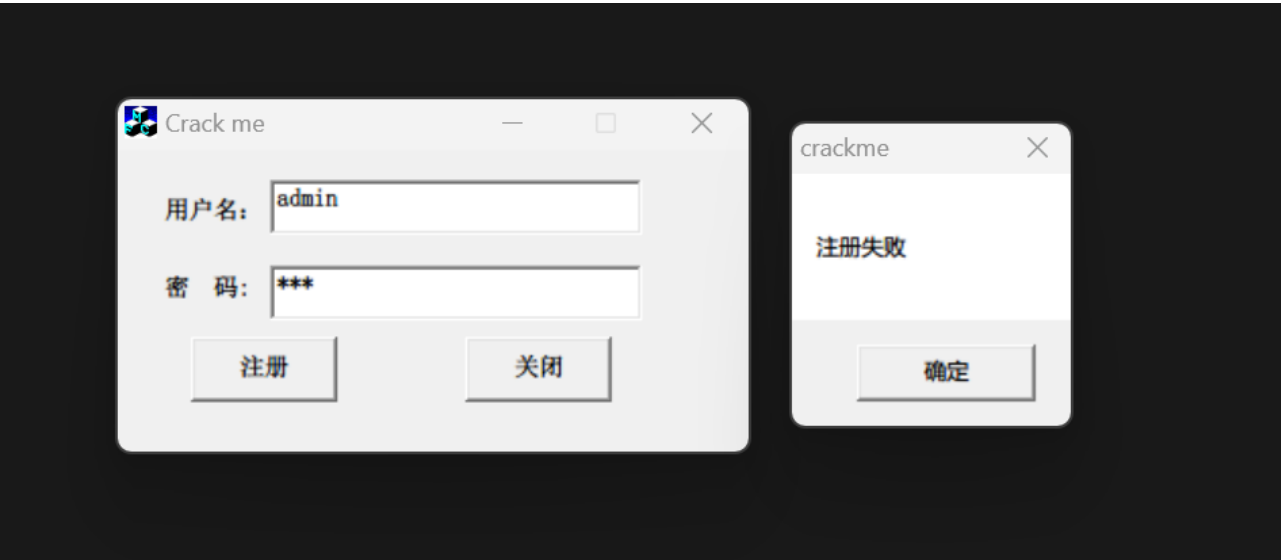


(2) 测试结论

- 我们成功地使用逆向工程技术来修改程序的行为，从而绕过密码验证。
- 在实际应用中，这种漏洞可能会导致安全问题，因为攻击者可以绕过密码验证来访问受保护的资源。
- 虽然这是一个简简单单的程序，但却是我们关于软件安全学习的一大步
- 程序没有对密码的长度进行判断，虽然密码是1234567，但是输入12345678仍然正确。

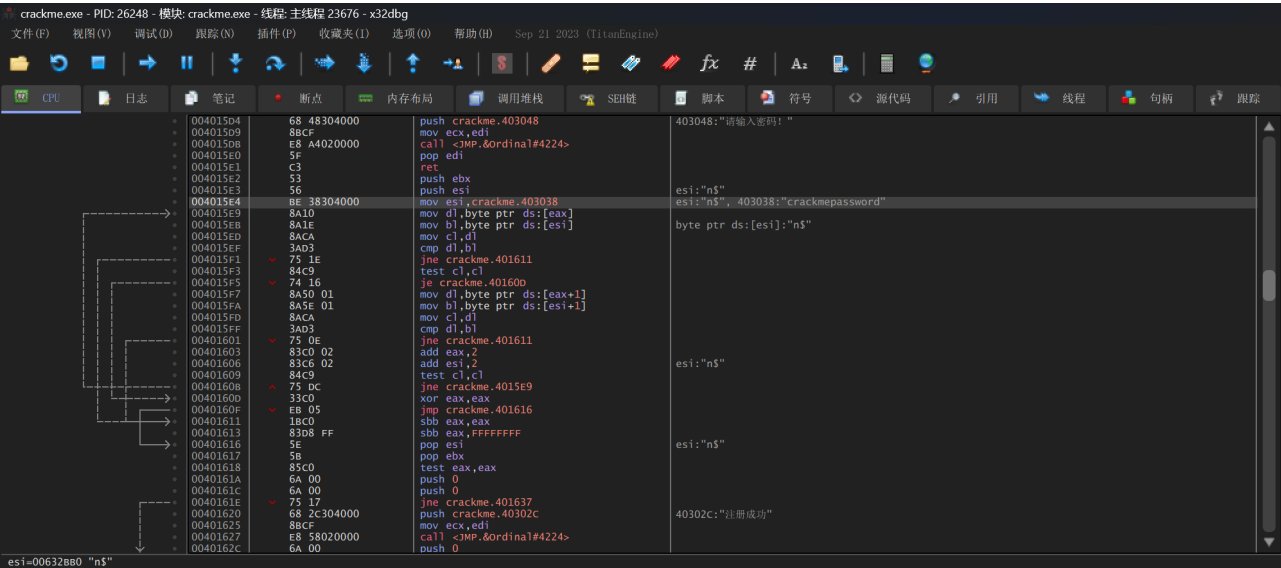
(3) 附加题

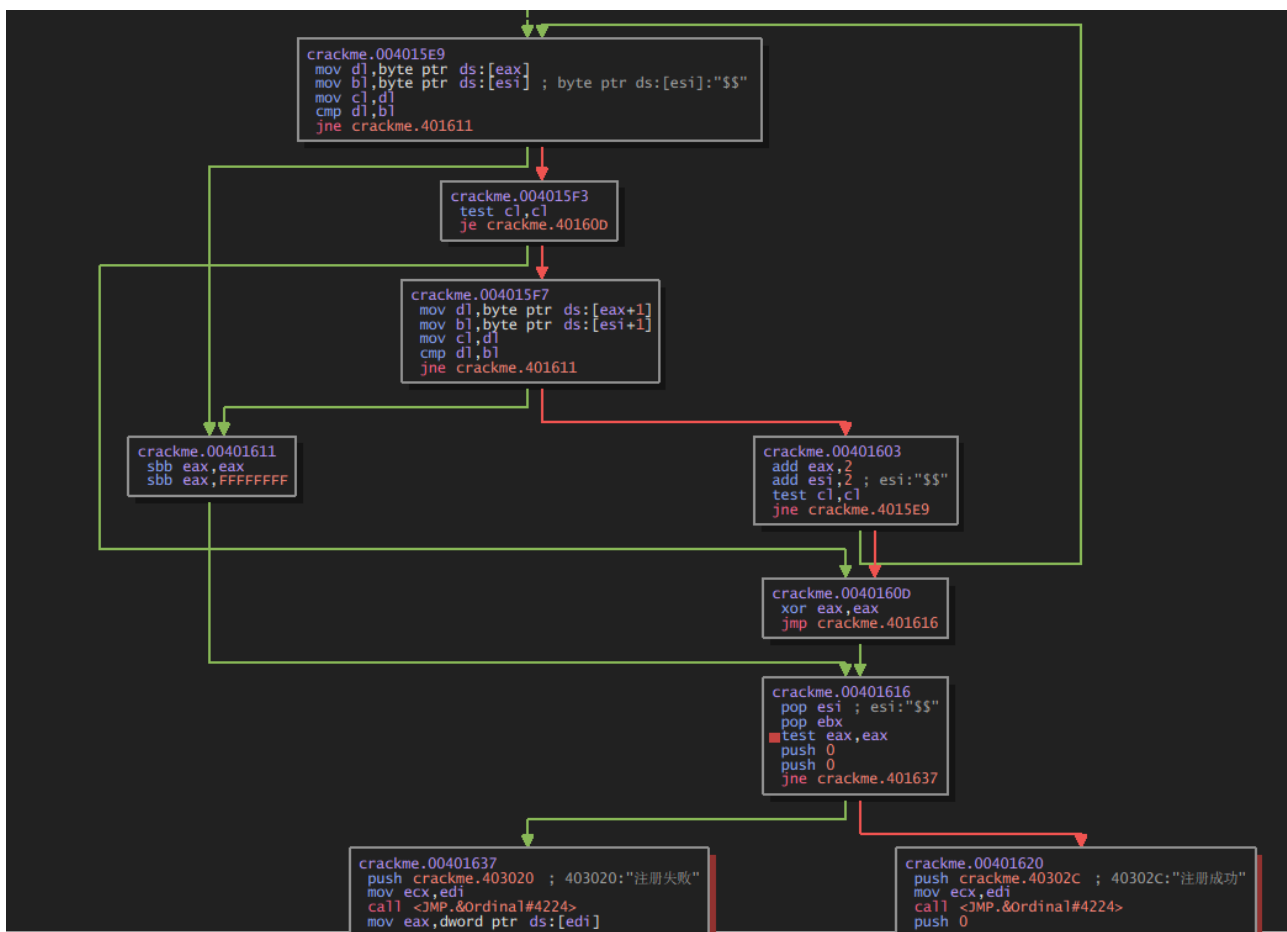
测试



进入是一个注册界面，需要输入用户名和密码才可注册成功。随便输入一个发现注册失败。

进入x64dbg





打开字符串查找“注册成功”，在这里我们可以看到注册成功的跳转代码，并且还有真正的密码传入，此时我们在ida中寻找对应的密码存放位置，发现密码实际为"crackmepassword"。但是我们要求使用密码绕过的方式，所以换种方法。并且我们发现注册界面时不对用户名进行验证的，只需要输入正确的密码即可

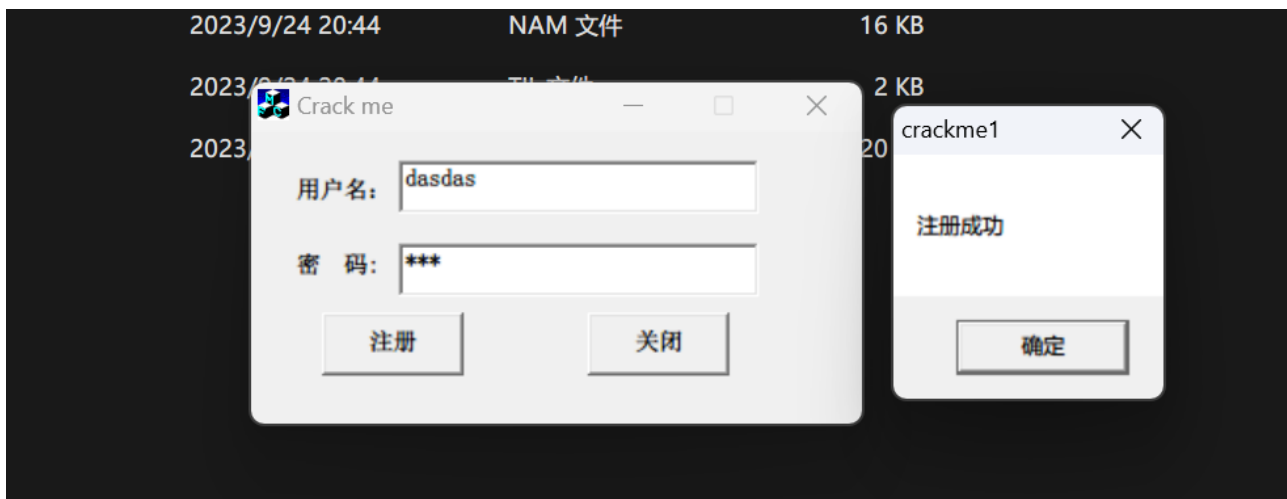
方法1: 直接修改jne

0040161A	6A 00	push 0	
0040161C	6A 00	push 0	
0040161E	75 17	jne crackme.401637	
00401620	68 2C304000	push crackme.40302C	40302C:"注册成功"
00401625	8BCF	mov ecx,edi	
00401627	E8 58020000	call <JMP.&Ordinal#4224>	
0040162C	6A 00	push 0	
0040162E	8BCF	mov ecx,edi	
00401630	E8 55020000	call <JMP.&Ordinal#6334>	
00401635	5F	pop edi	
00401636	C3	ret	
00401637	68 20304000	push crackme.403020	403020:"注册失败"
0040163C	8BCF	mov ecx,edi	
0040163E	E8 41020000	call <JMP.&Ordinal#4224>	
00401643	8B07	mov eax,dword ptr ds:[edi]	
00401645	8BCF	mov ecx,edi	
00401647	FF90 D0000000	call dword ptr ds:[eax+D0]	
0040164D	6A 00	push 0	
0040164F	8BCF	mov ecx,edi	
00401651	E8 34020000	call <JMP.&Ordinal#6334>	
00401656	5F	pop edi	
00401657	C3	ret	
00401658	90	nop	

我们可以看到当40161E不跳转时，就可以进入到我们注册成功的这个操作。我们可以直接修改40161E的值，改变其为跳转。只要不跳转到401637这样我们就可以实现我们注册成功的操作，完成密码绕过。

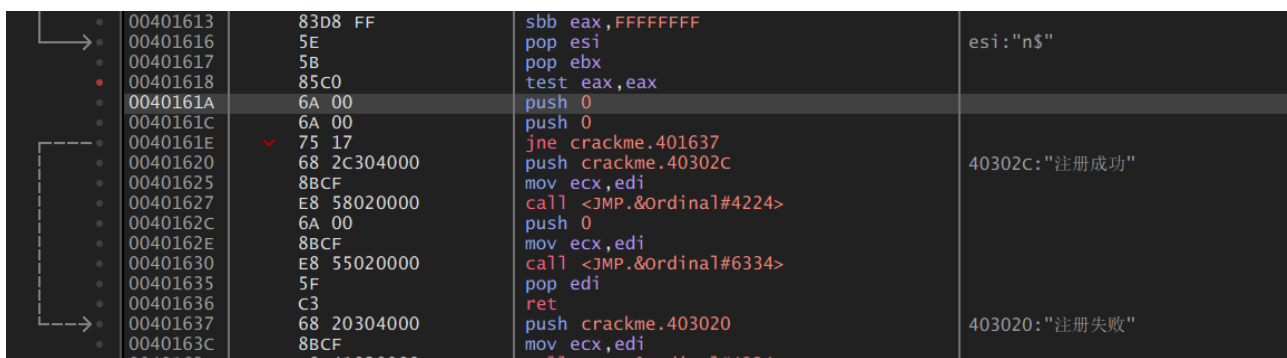


更改jne（jnz）401367为jz 401637，并保存为crackme1.exe并运行。

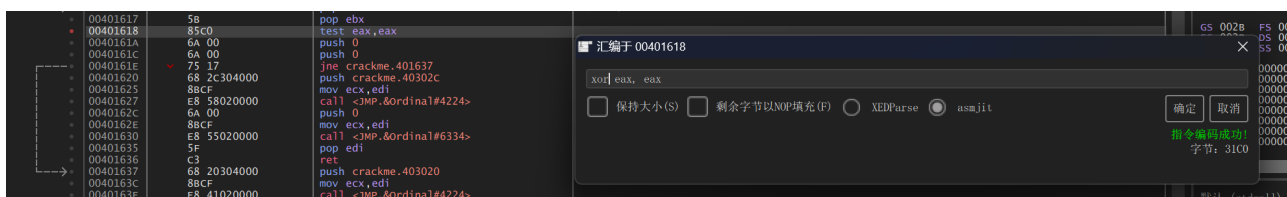


此时我们随便输入用户名和密码 **dasdas**和**123**。发现此时注册就成功了。（只要密码不是正确的密码就行，我们改变了代码逻辑为输入错误的密码即可注册成功）

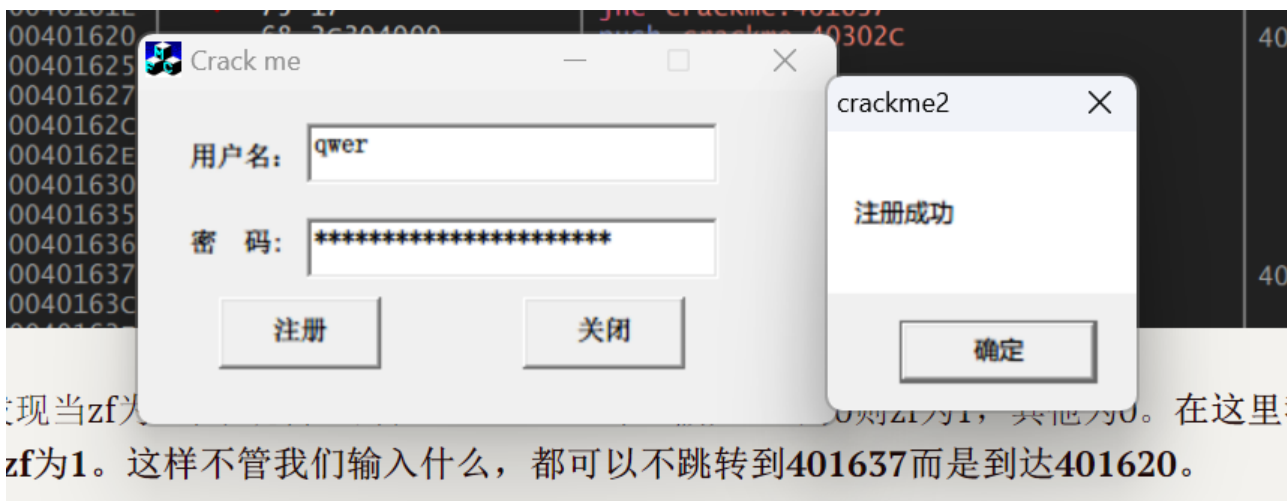
方法2: 修改test



观察jne，发现当zf为1时不跳转，而在401618test时，假如eax为0则zf为1，其他为0。在这里我们可以修改**test**为**XOR**或者是**SUB**是的**zf**为**1**。这样不管我们输入什么，都可以不跳转到**401637**而是到达**401620**。

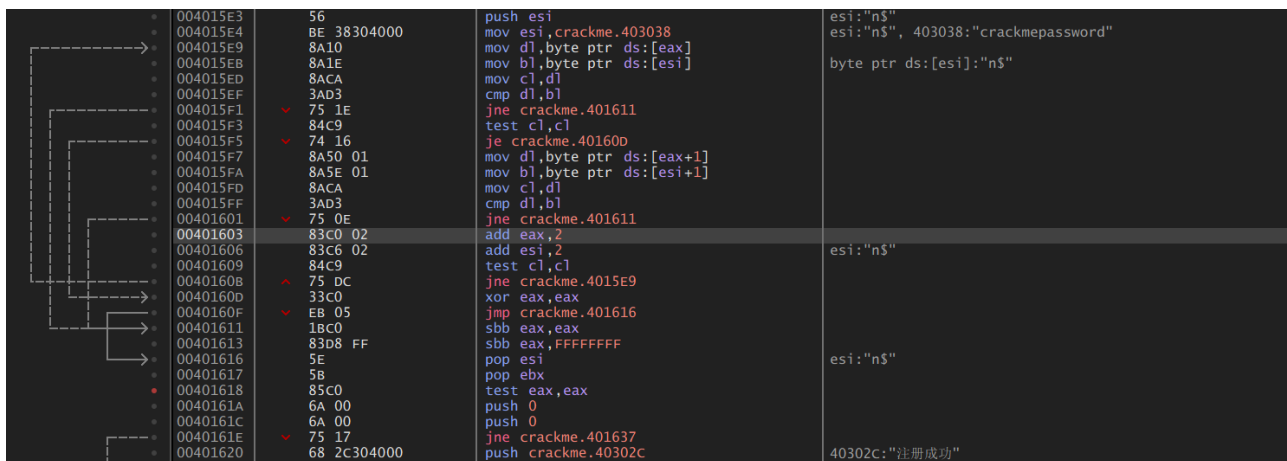


修改为xor eax, eax。保存为crackme2.exe并运行。

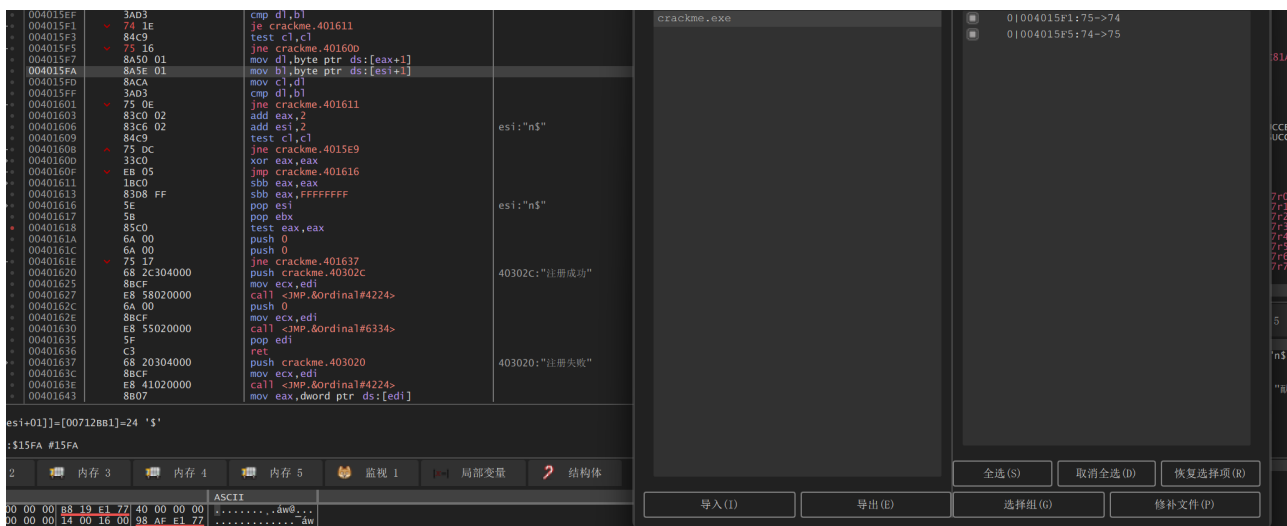


此时不管我们输入什么密码都能注册成功。

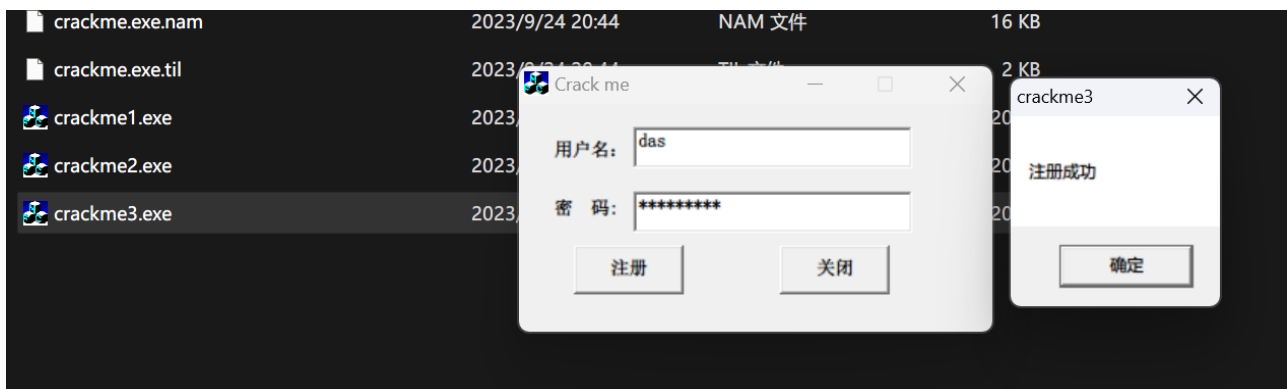
方法3: 修改验证密码逻辑



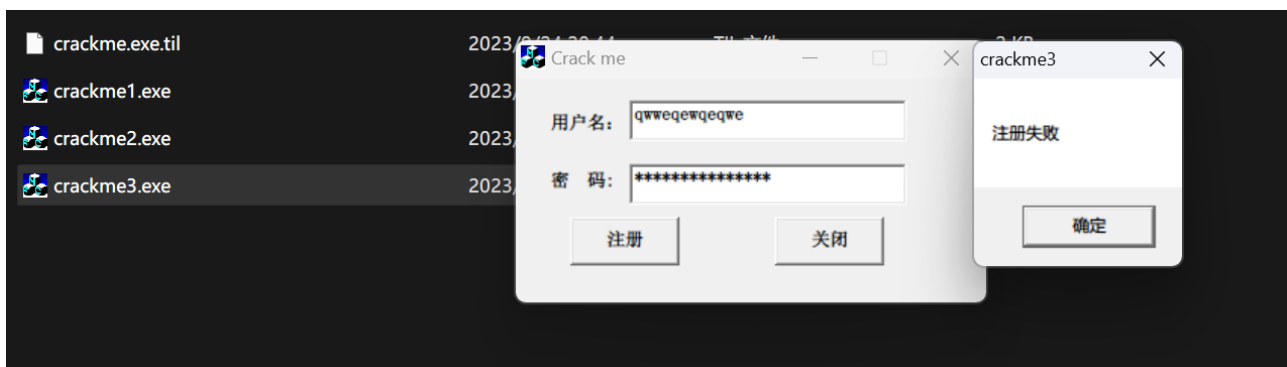
观察4015E3-401616可以发现这里存在着一个循环，应该是对密码的验证与对比。观察这几个跳转地址，004015F1、004015F5、004015F5、0040160B、还有sbb eax, eax、sbb eax FFFFFFFF。我们可以知道假如运行到sbb eax, eax、sbb eax FFFFFFFF那么ZF会为0，这样40161E就会跳转到401367。所以我们要使ZF为1。即一定不能跳转/到达至401611。所以必须在在 004015F1处不跳转、在 004015F5 跳转时，才能到达401620处执行注册成功指令。



所以我们可以将 004015F1 处 JNZ（JZ）指令改为 JE 指令，将 004015F5 处 JE（JZ）指令改为 JNZ 指令。保存为crackme3.exe并运行。



这时候我们随便输入任意密码（只要不是正确密码），都可以注册成功。



而假如我们输入正确的密码这个时候就会注册失败，因为我们已经修改了代码逻辑，密码错误才可以注册成功！