

第6次实验课实验报告要求-SEH

一、实验内容

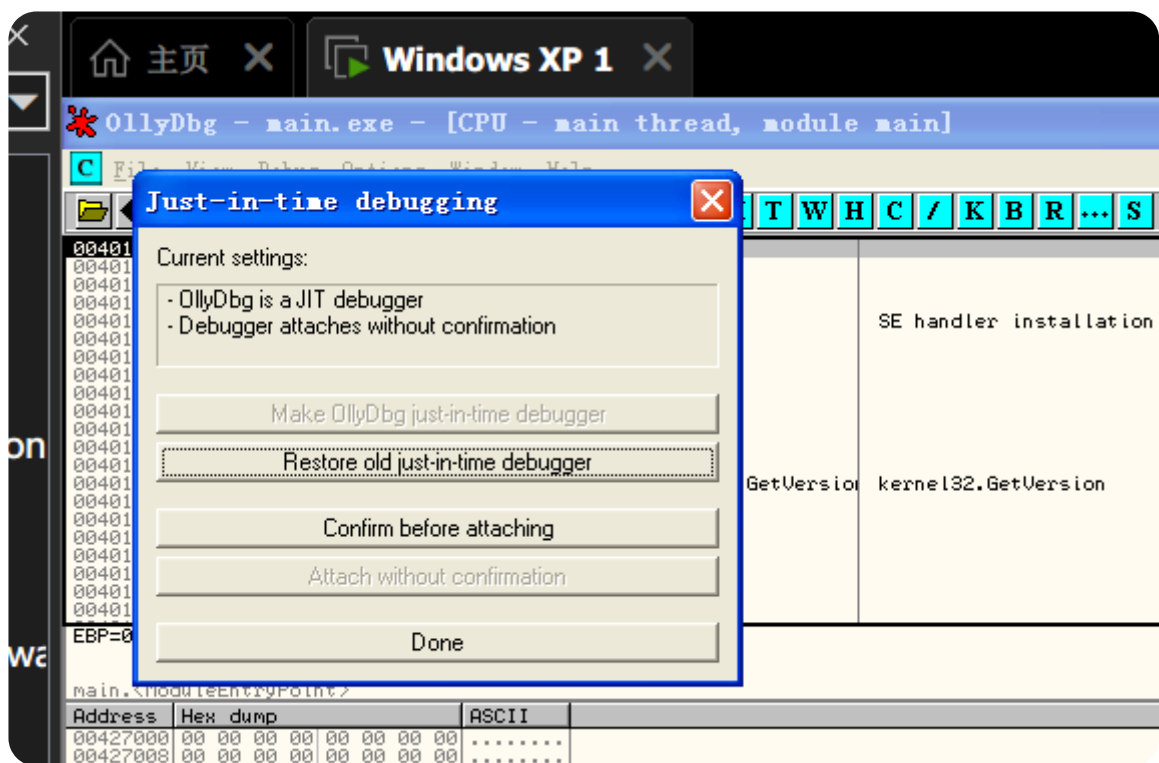
- (1) 了解SEH攻击的基本原理
- (2) 实验一：通过调试SEH攻击代码，理解Windows异常处理机制，掌握针对SEH的攻击方式，并利用OllyDbg跟踪异常状态。在报告中详述调试过程，并在分析完成后注释代码中的__asm int 3重新编译运行程序，验证攻击效果
- (3) 实验二：列举并详细解释windows2000中的2种常见异常，包括但不限于除0异常，调试异常、页异常等等，给出会触发你所列举异常的代码。

二、实验步骤

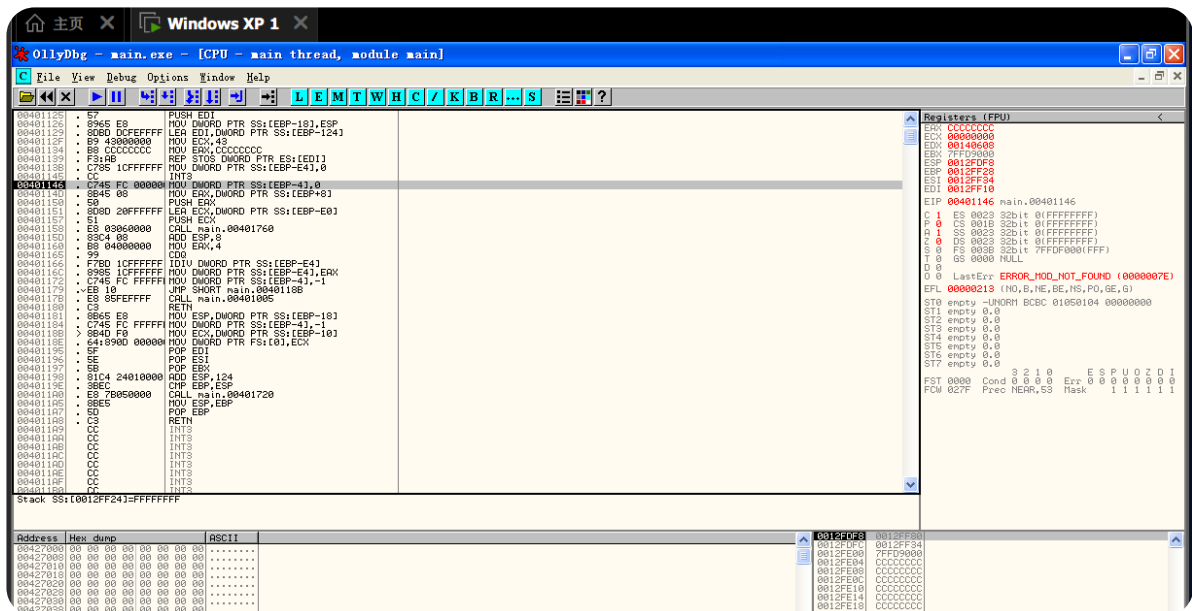
2.1实验一

调试 SEH 攻击代码

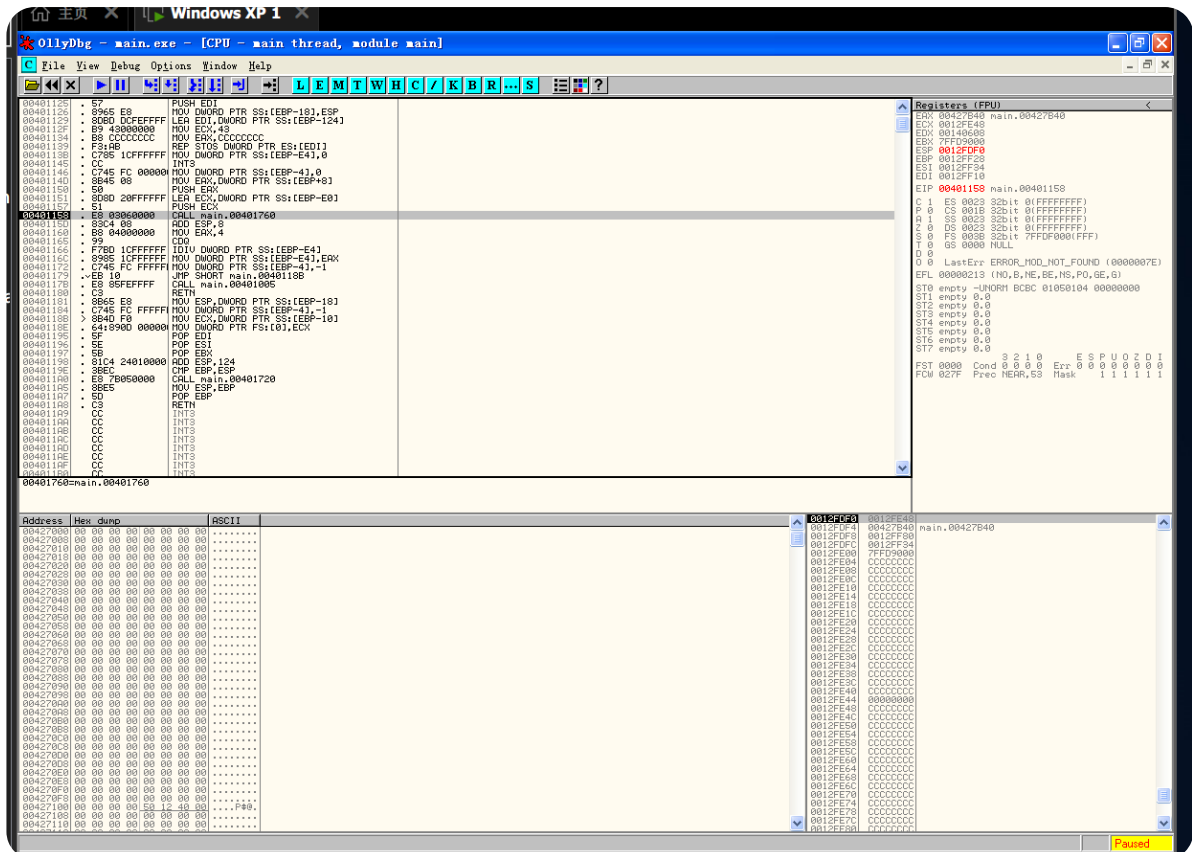
为了能出发 `int 3` 断点时启动 `ollydbg`，设置 `ollydbg` 为实时调试器



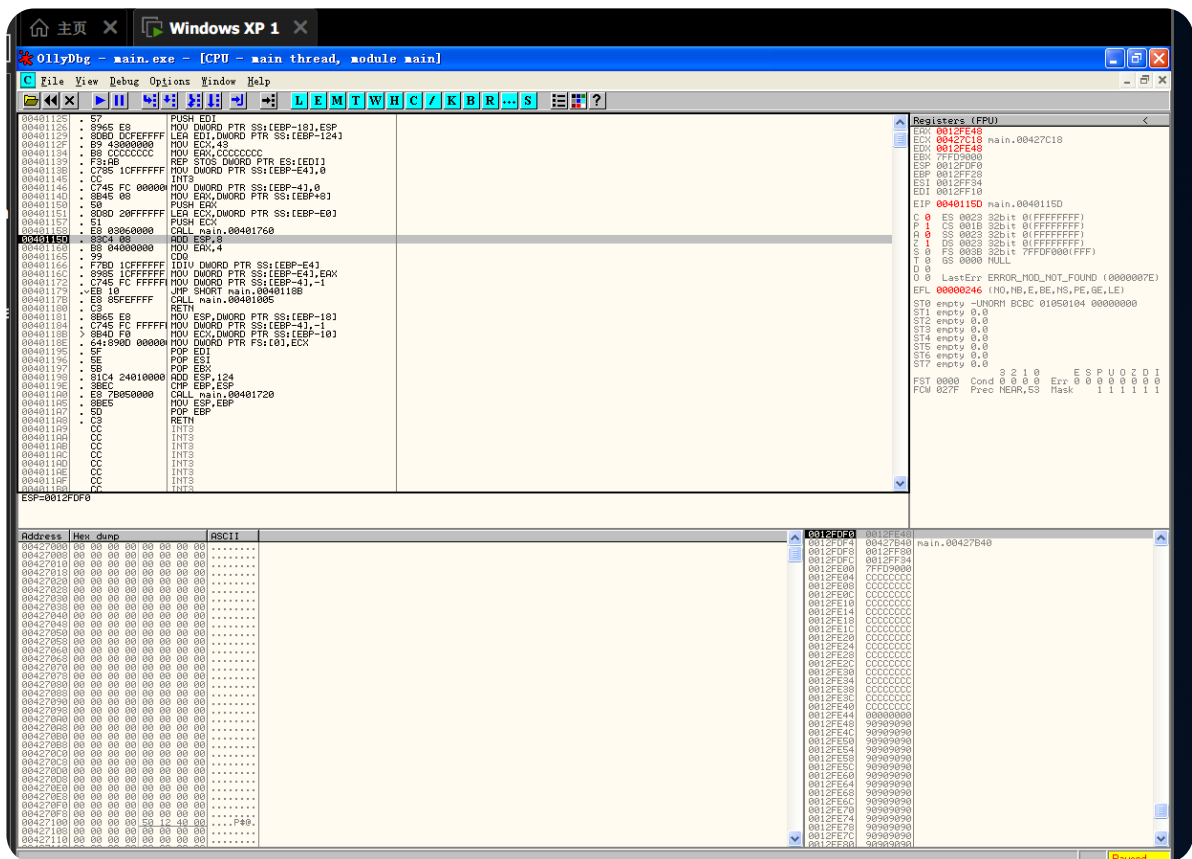
重新运行 main.exe 程序, 成功在 int 3 上启动 OllyDbg



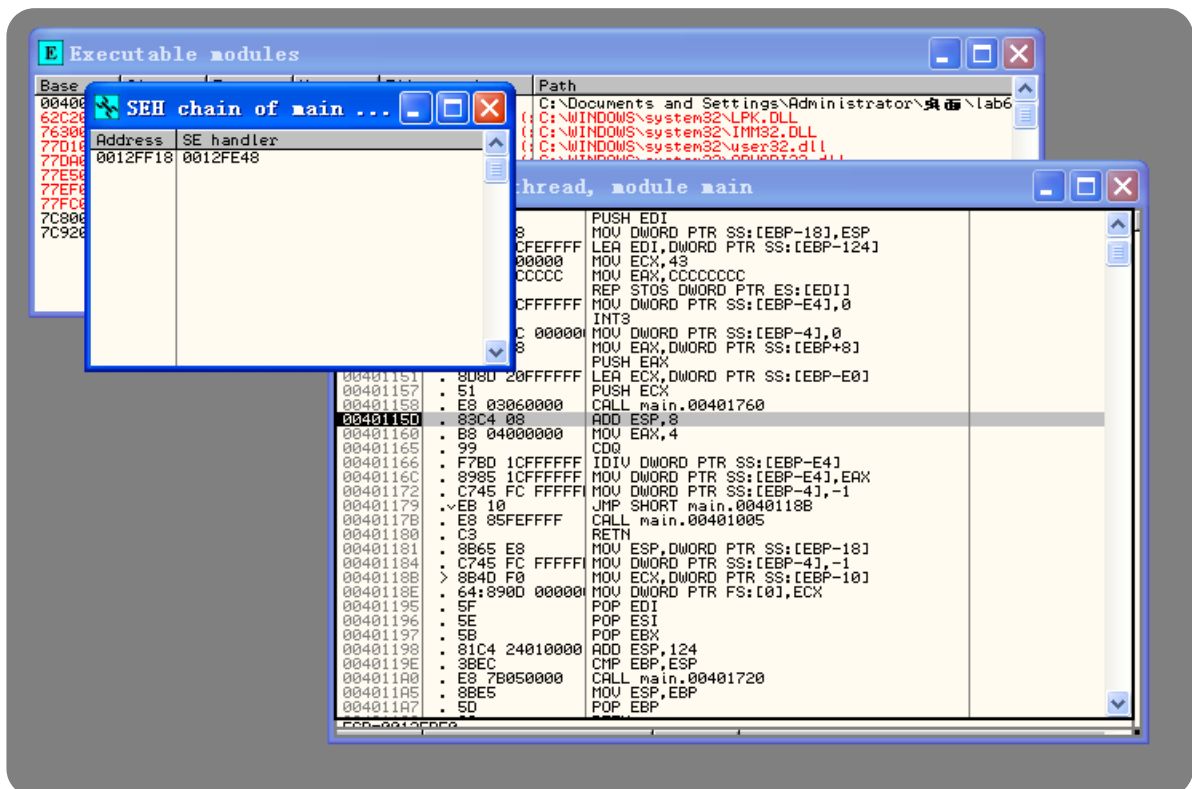
在 strcpy 函数处查看, 程序运行到此处时观察右下角缓冲区数据, 可以看到在执行 strcpy 函数之前, shellcode 的起始地址为 0x0012FE48



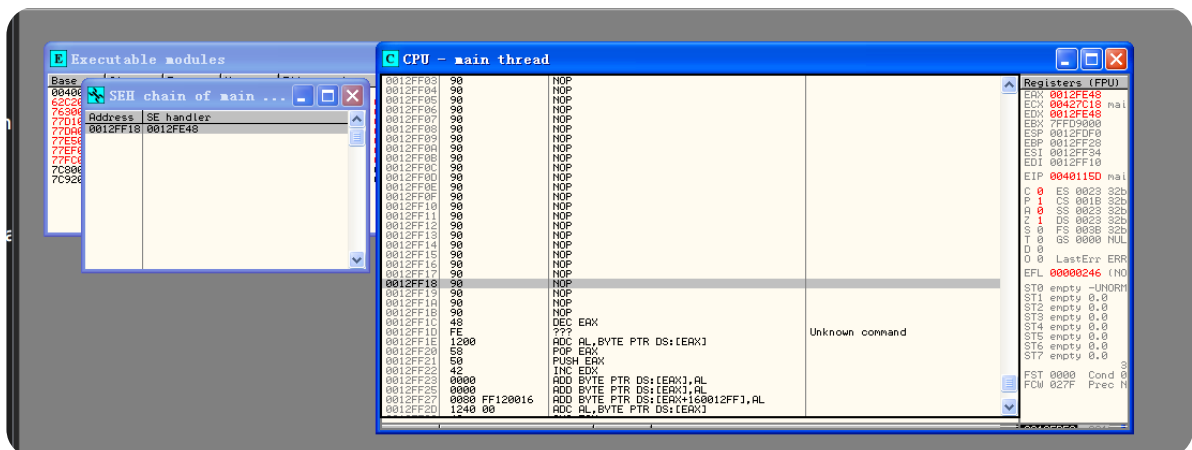
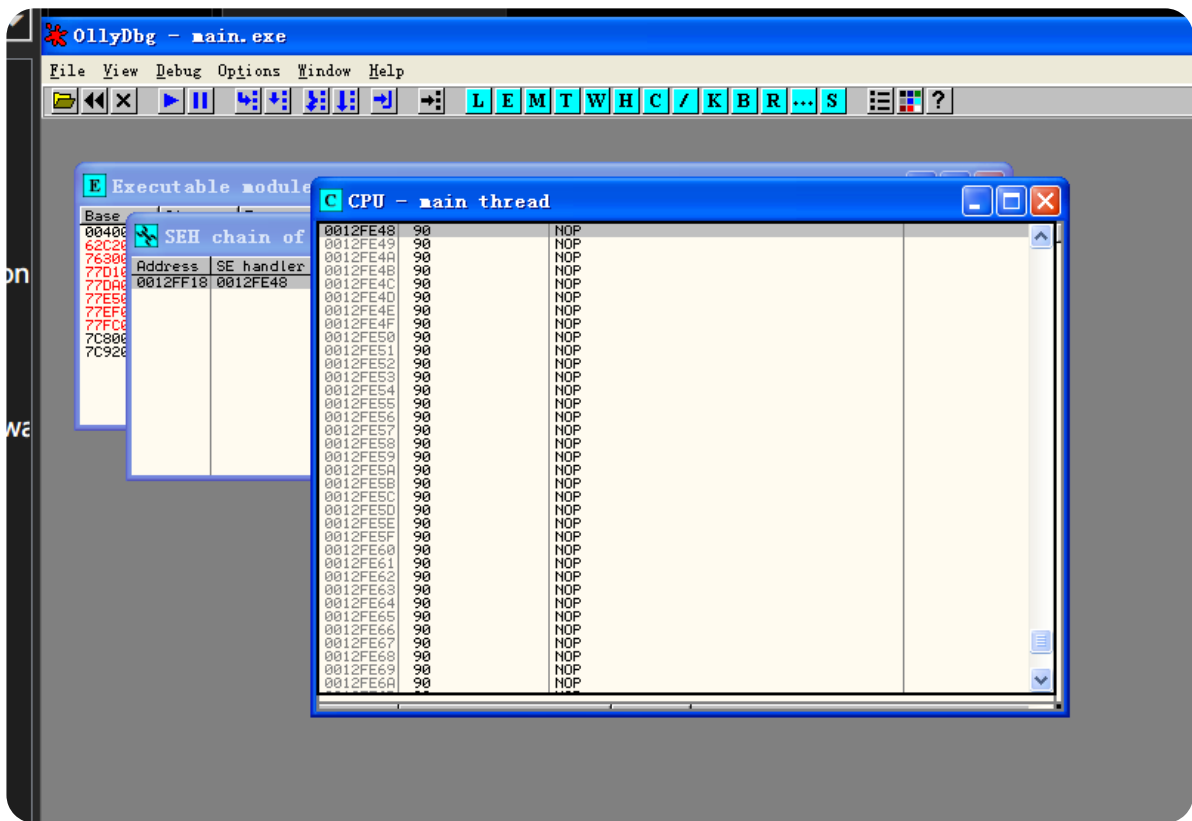
执行完 strcpy, 确认 shellcode 的起始位置是 0x0012FE48



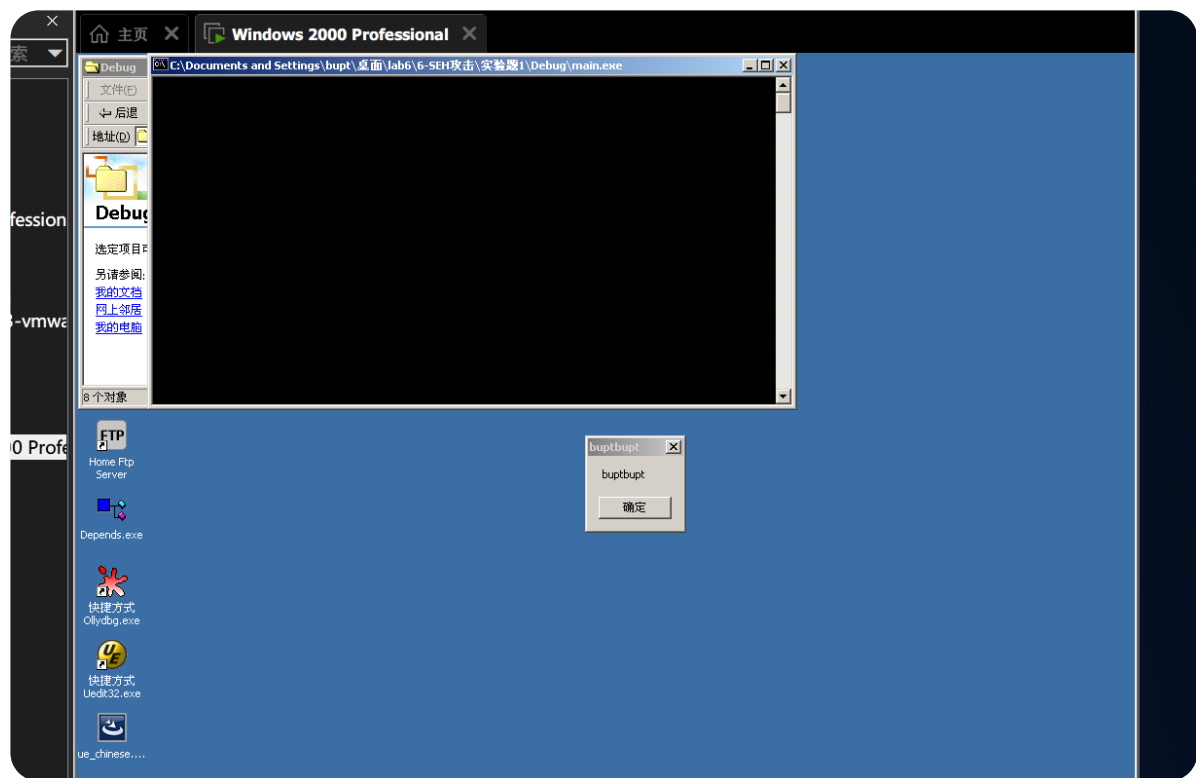
查看 S.E.H链 , 地址 0x0012FF18



查看地址 0x0012FF18 的记录, 发现其指向shellcode地址



注释掉 `_asm int 3` , 运行。



点击确定无法关闭，因为除0一直处于异常。一直在触发。

2.2实验二：列举并详细解释windows2000中的2种常见异常

1.除零异常（Divide-by-Zero Exception）：

异常描述：当程序试图将一个数除以零时，会触发除零异常。

代码示例：

```
#include <stdio.h>

int main() {
    int numerator = 10;
    int denominator = 0;
    int result;

    // Attempting to divide by zero
    result = numerator / denominator;

    printf("Result: %d\n", result);

    return 0;
}
```

在这个例子中，`denominator` 被设置为零，导致执行 `numerator / denominator` 时发生除零异常。

2.调试异常 (Debug Exception) :

异常描述：调试异常是由于调试器（如调试器或程序监视器）请求中断执行引发的异常。这个异常可用于单步执行程序、观察内存变化等调试操作。

代码示例：实际触发调试异常的代码通常是由调试器插入的，但是下面的示例代码通过 `int 3` 指令手动触发调试异常。

```
#include <stdio.h>

int main() {
    printf("Before Debug Exception\n");

    // Triggering a Debug Exception using int 3 instruction
    __asm {
        int 3
    }

    printf("After Debug Exception\n");

    return 0;
}
```

在这个例子中，`int 3` 指令用于触发调试异常。在正常运行时，这会中断程序执行，并交给关联的调试器处理。

3.页异常 (Page Fault Exception)

描述：Page Fault Exception 发生在程序试图访问虚拟内存中的某一页，而这一页当前未加载到物理内存中。当出现这种情况时，操作系统负责将所需的页面加载到物理内存中，然后重新启动引发异常的指令，使程序继续执行。

以下是一个简单的代码示例，演示如何触发页异常：

```

include <stdio.h>

int main() {
    int *ptr = NULL;

    // Trying to access memory through a NULL pointer
    // This will cause a Page Fault Exception
    *ptr = 10;

    printf("This line will not be reached due to the Page Fault Exception.\n");

    return 0;
}

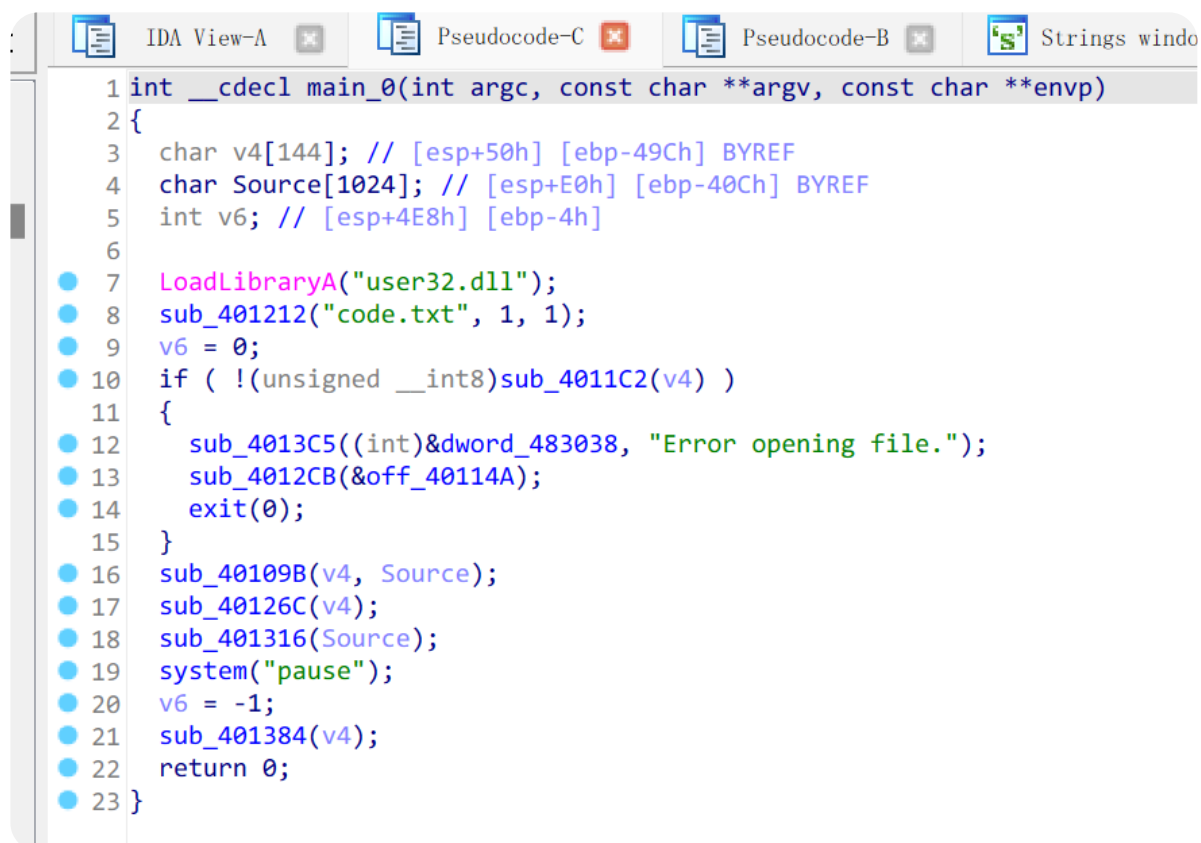
```

在这个例子中，`ptr` 是一个空指针，当程序试图通过空指针访问内存时，将引发页异常。实际上，这里的问题是程序试图写入一个地址为 `NULL` 的内存，这通常会导致页异常。当发生页异常时，操作系统会负责加载相应的页面，然后重新执行引发异常的指令。

三、附加题

附加题：调试并分析附加题目录下的seh.exe文件，在不修改程序源代码的情况下，构造shellcode，通过利用seh植入shellcode的方式实现弹窗（同目录下的code.txt文件很重要，不要删了！）

ida打开seh.exe文件查看

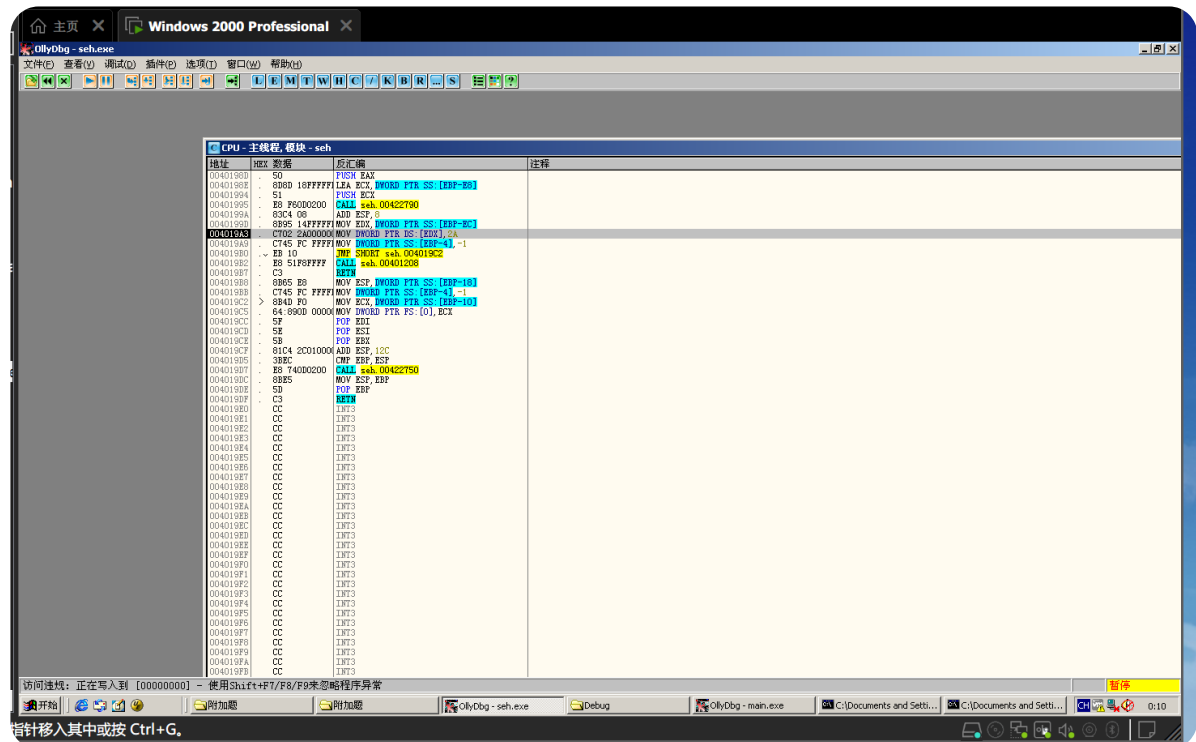


```

IDA View-A | Pseudocode-C | Pseudocode-B | Strings window
1 int __cdecl main_0(int argc, const char **argv, const char **envp)
2 {
3     char v4[144]; // [esp+50h] [ebp-49Ch] BYREF
4     char Source[1024]; // [esp+E0h] [ebp-40Ch] BYREF
5     int v6; // [esp+4E8h] [ebp-4h]
6
7     LoadLibraryA("user32.dll");
8     sub_401212("code.txt", 1, 1);
9     v6 = 0;
10    if ( !(unsigned __int8)sub_4011C2(v4) )
11    {
12        sub_4013C5((int)& dword_483038, "Error opening file.");
13        sub_4012CB(&off_40114A);
14        exit(0);
15    }
16    sub_40109B(v4, Source);
17    sub_40126C(v4);
18    sub_401316(Source);
19    system("pause");
20    v6 = -1;
21    sub_401384(v4);
22    return 0;
23 }

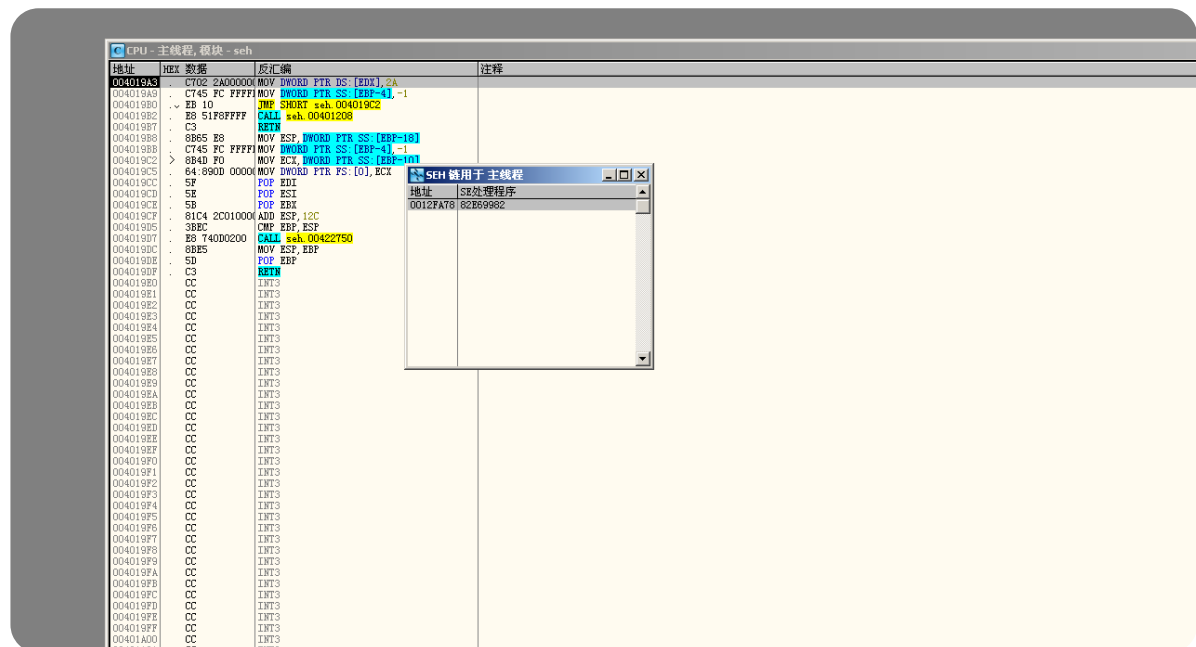
```

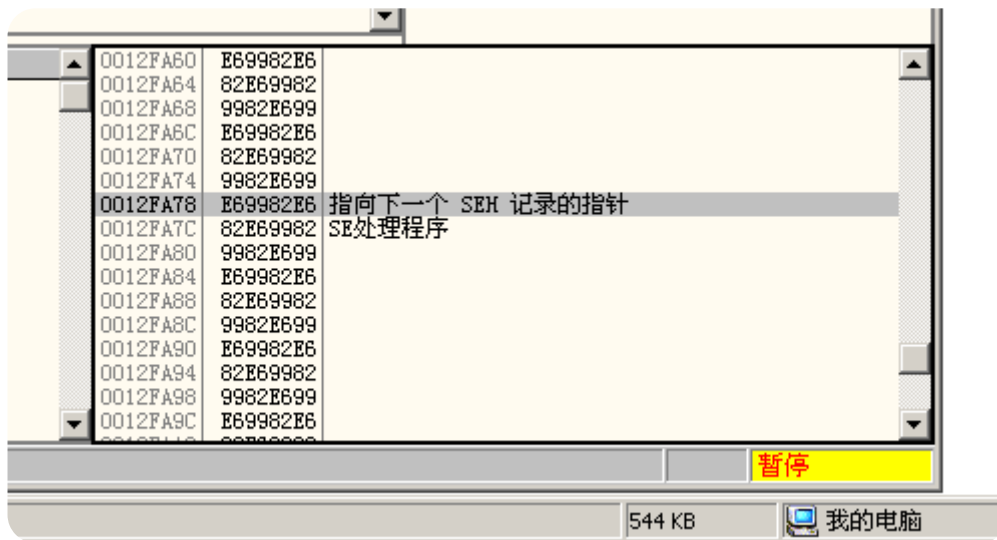
用ollydbg调试一下。



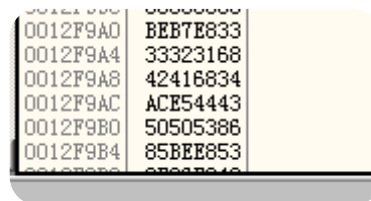
一直运行程序，发现运行到这里，出现了异常。发现是MOV DWORD PTR DS:[EDX],2A（其中EDX为0），对访问不到的数据进行了赋值引起了异常。

查看此时的SEH链，地址SEH为0012FA78

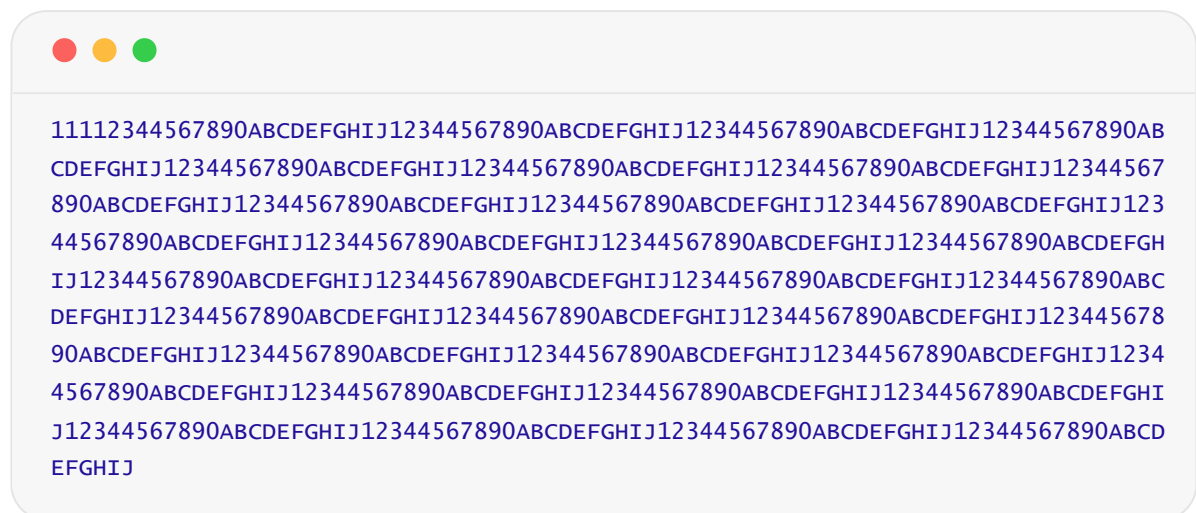


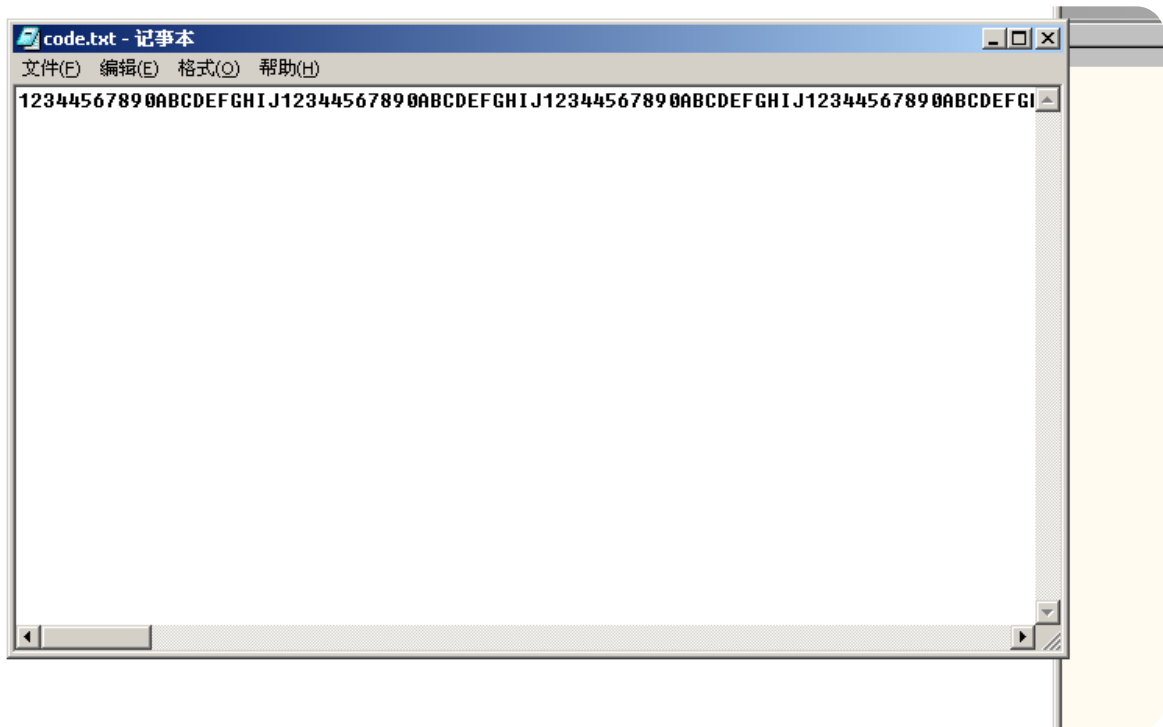


并且我们在栈的上方发现了我们code.txt的内容

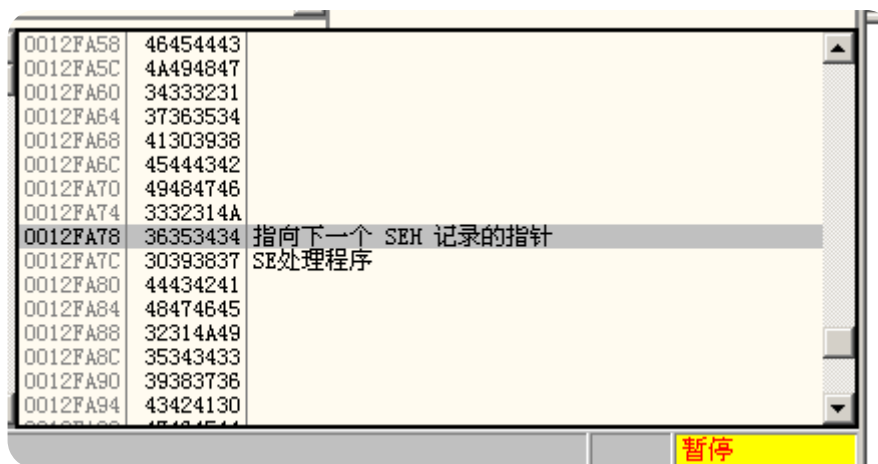


修改code.txt。

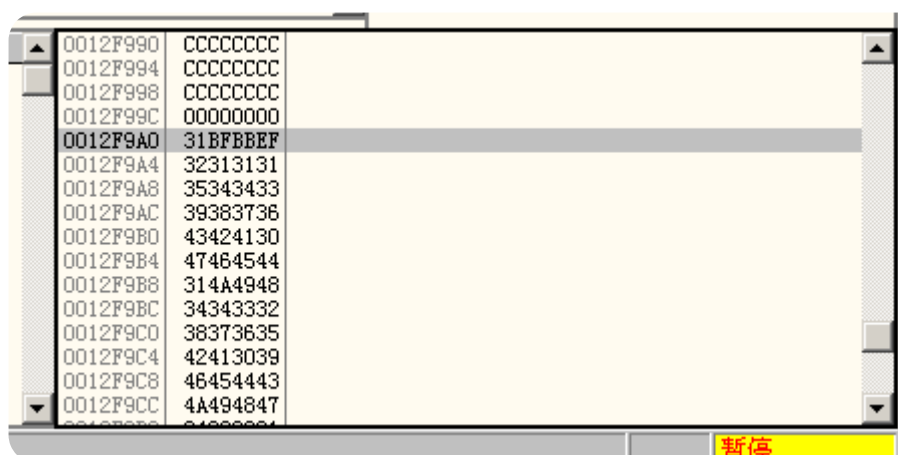




调试看看。



发现我们是可以通过栈溢出的发现，在触发异常时，修改SEH的处理程序。所以我们要修改0012FA78为shellcode的起始地址。



我们可以看到是从0012F9A0开始复制的。或者说我们的shellcode入口地址为0012F9A0。

所以构造一下code.txt。

[illegible]

通过本次实验，我深入了解了Windows中的异常处理机制以及SEH（Structured Exception Handling）攻击的基本原理。在实验过程中，我掌握了以下关键内容：

1. **SEH攻击的基本原理：** SEH是Windows操作系统中的异常处理机制，用于处理程序运行时发生的异常情况。SEH攻击利用了异常处理机制的漏洞，通过覆盖SEH链表中的异常处理程序指针，来执行恶意代码。这可以导致程序的非法行为，例如执行Shellcode。
2. **实验一：调试SEH攻击代码：** 通过调试SEH攻击代码，我深入理解了Windows异常处理机制的运作过程。通过OllyDbg等工具，我跟踪了异常状态，观察了SEH链表的变化，并成功构造了一个简单的SEH攻击，注入Shellcode并实现攻击效果。
3. **实验二：常见异常及代码示例：** 我列举并详细解释了Windows2000中的两种常见异常，包括除零异常和调试异常。通过示例代码，我展示了如何触发这些异常，深入理解了异常的产生和处理过程。
4. **附加题：SEH植入Shellcode：** 我成功完成了附加题，通过分析和调试程序，构造了一个Shellcode，并通过SEH链的修改，实现了在程序执行过程中弹出一个窗口的效果。这进一步加深了我对SEH攻击的理解。

总的来说，本次实验帮助我深入学习了Windows异常处理机制及SEH攻击，提高了我的调试和分析能力。通过实际操作，我更好地理解了操作系统中的异常处理流程，并学会了如何应对一些异常情况。这对于理解系统安全性和编写更安全的代码都具有重要的意义。