《漏洞利用及渗透测试基础》第四次实验报告

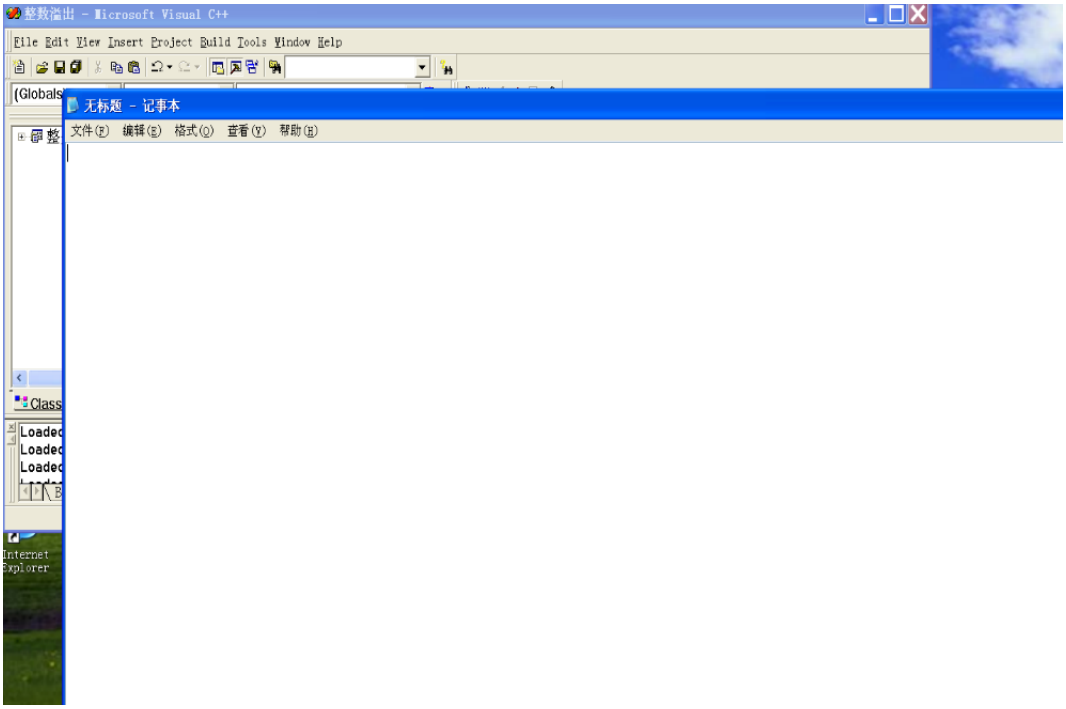
整数溢出漏洞

1811463 赵梓杰 信息安全

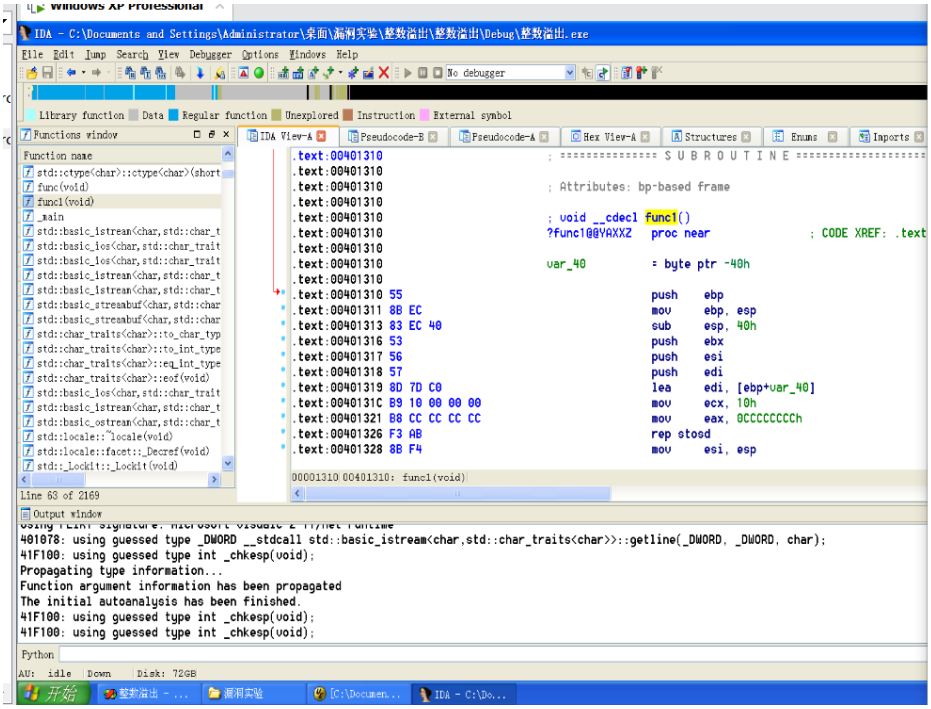
- 源代码

```
#include<iostream>
#include<windows.h>
#include<shellapi.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX_INFO 32767
using namespace std;
void func()
{
    ShellExecute(NULL,"open","notepad",NULL,NULL,SW_SHOW); //打开记事本
}
void func1()
{
    ShellExecute(NULL,"open","calc",NULL,NULL,SW_SHOW);  //打开计算器
}
int main()
{
    void (*fuc_ptr)() = func;
    char info[MAX_INFO];
    char info1[30000];
    char info2[30000];

    freopen("input.txt","r",stdin);
    cin.getline(info1,30000,' ');
    cin.getline(info2,30000,' ');

    short len1 = strlen(info1);
    short len2 = strlen(info2);
    short all_len = len1 + len2;

    if(all_len<MAX_INFO)
    {
        strcpy(info,info1);
        strcat(info,info2);
    }
    fuc_ptr();
    return 0;
}
```

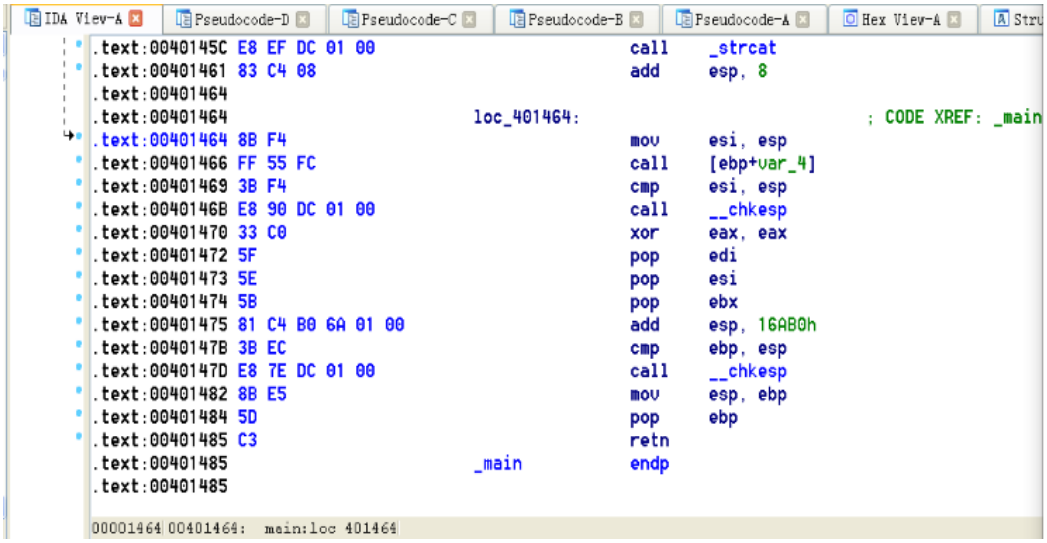
- 代码分析
    - 正常情况下，我们运行程序，会弹出记事本程序（notepad）



- 然而当字符串的长度过长，那么all_len会超限溢出，覆盖栈的其它位置；我们的目标也就是通过对len1和len2的构造使得func_ptr指向func1
- 借助IDA查看func1的指令地址位置为00401310

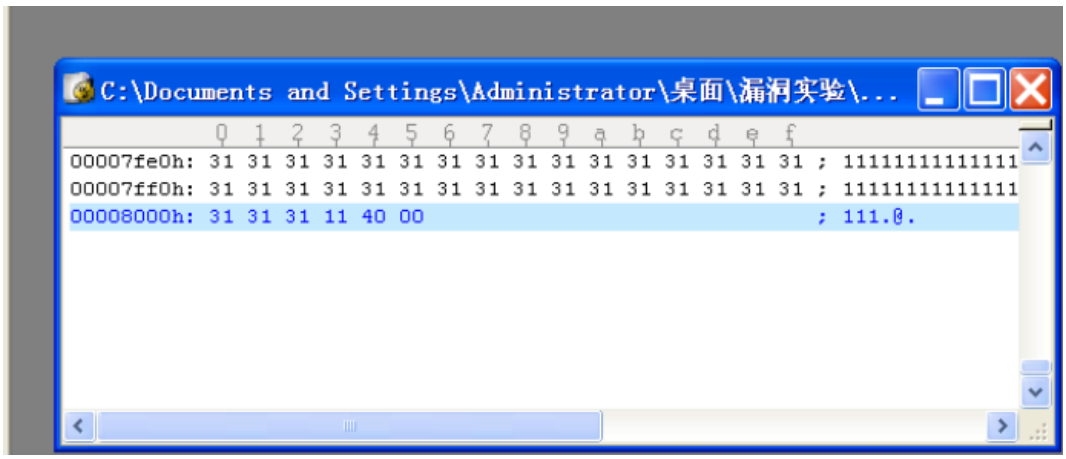


- 然后我们就需要找到跳入到func的func_ptr（），发现其步入func的指令

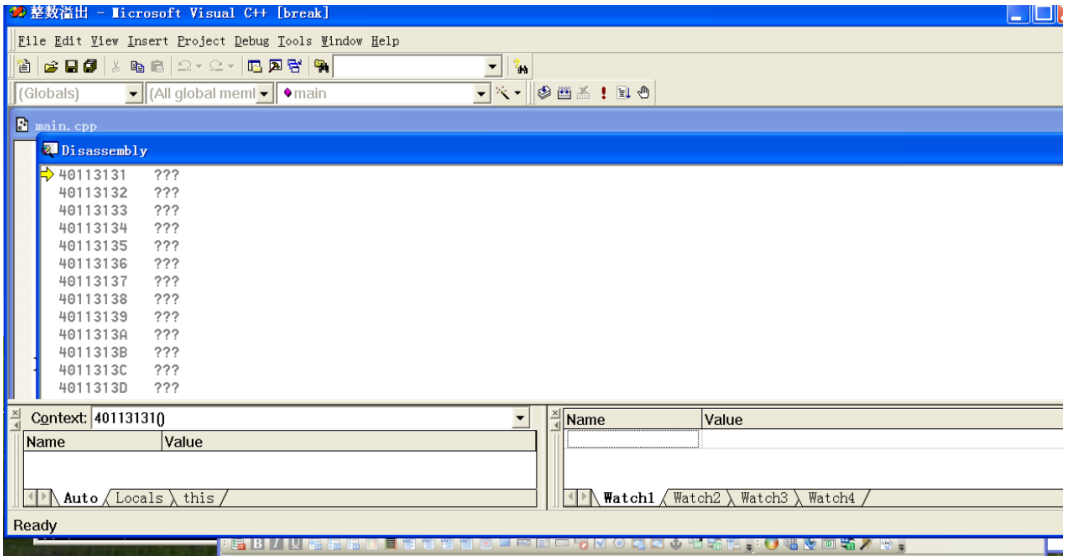


- 然后我们步入执行就可以找到

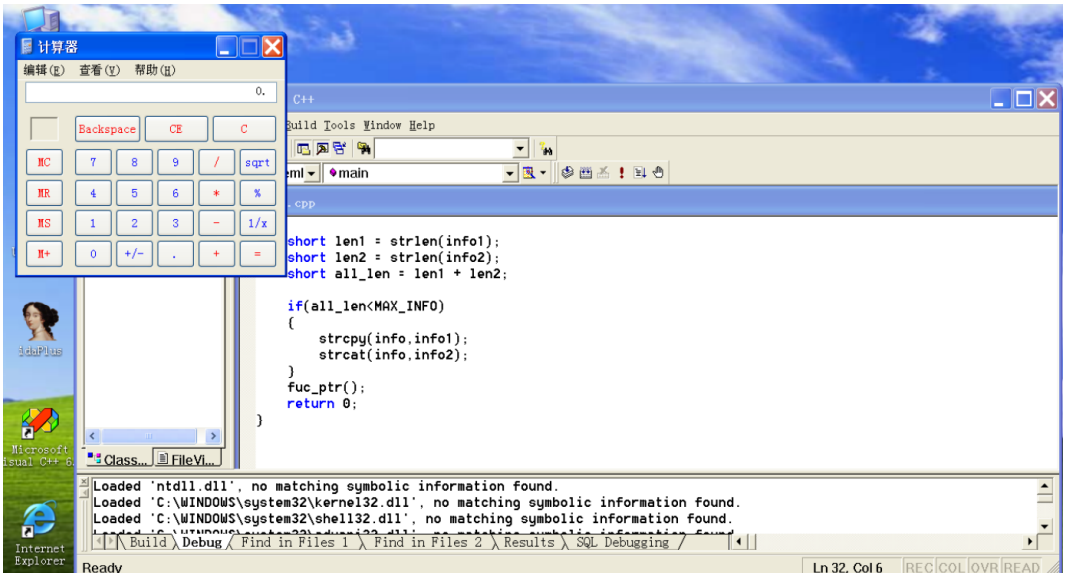
- 所以我们只需要将00401073改成00401131即可完成相关的跳转，即我们先找到func_ptc的位置，让字符串溢出覆盖func_ptc



然后发现报错了，感觉可能是计算位数的时候出现了差错，这里我发现我的VC报错为以下



- 根据这个提示的40113131我发现这个不是我们想要的目的地，它时返回的比我们00，所以我们删除一个'31'，然后可以直接调用出计算器



- 心得体会

  我在构造payload溢出的时候出现了没数清楚'20'的问题，也就是' '，但是可以很迅速的通过VC的bug报错找到问题所在，简单了解了整数溢出的相关知识和操作