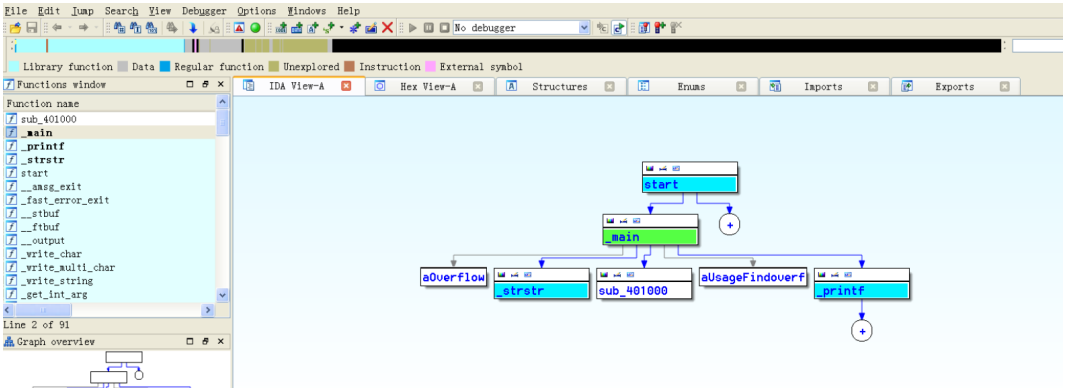
《漏洞利用及渗透测试基础》第七次实验报告

1811463 赵梓杰 信息安全

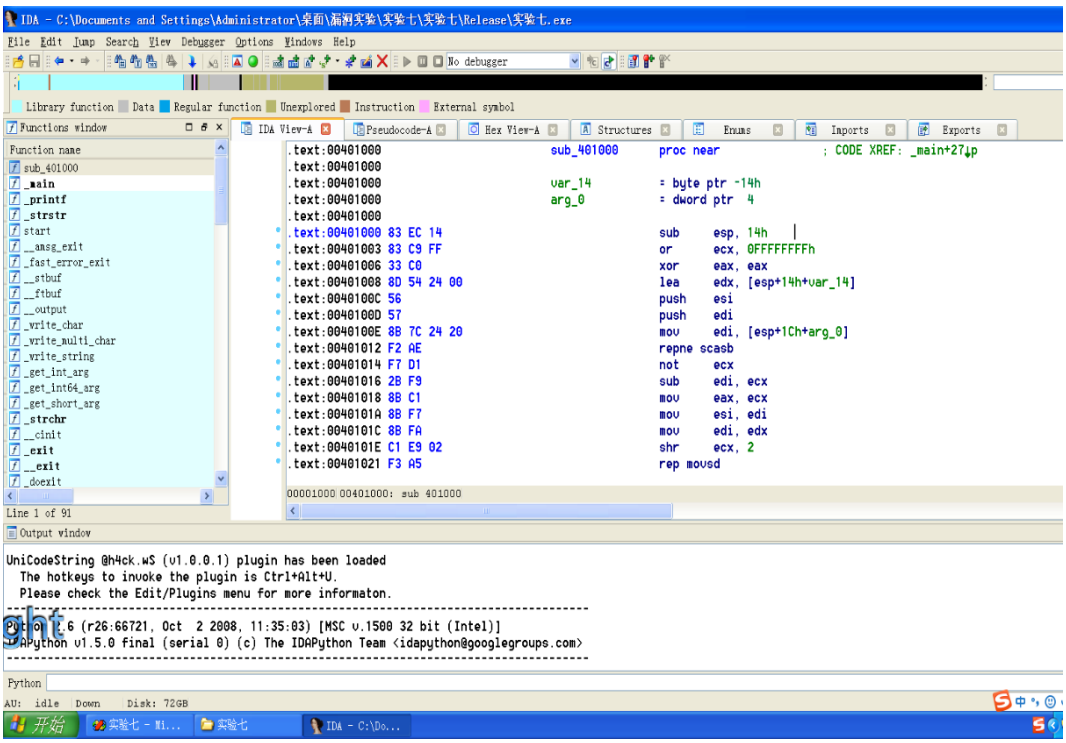
- 源代码

```
#include <stdio.h>
#include <string.h>
void makeoverflow(char *b)
{
    char des[20];
    strcpy(des,b);
}
void main(int argc,char *argv[])
{
    if(argc>1)
    {
        if(strstr(argv[1],"overflow")!=0)
        makeoverflow(argv[1]);
    }
    else
        printf("usage: findoverflow XXXXX\n");
}
```

- 基本流程
  - 首先就在build里面的set切换成release模式，用IDA打开release模式生成的exe，然后视图查看，可以得到以下

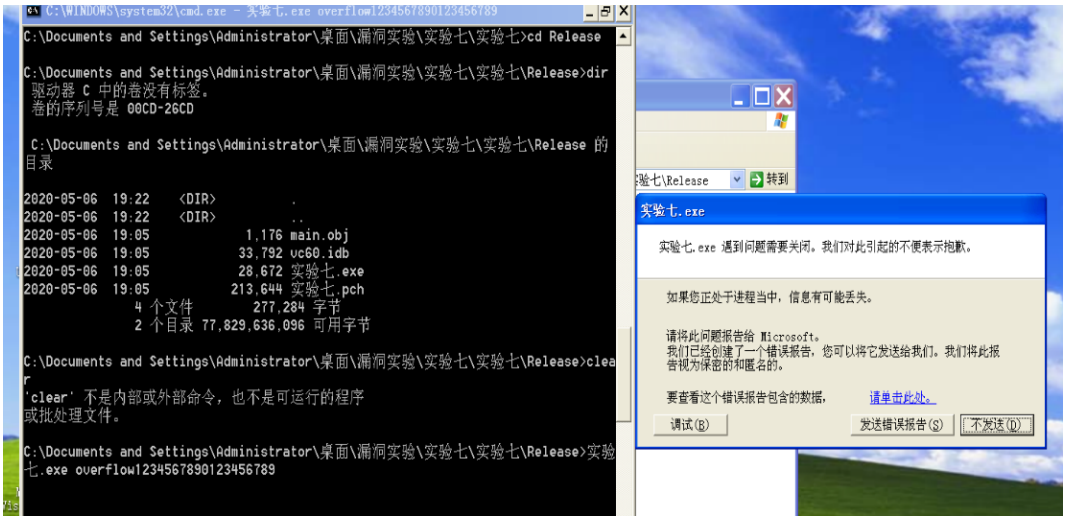


  - 因为本题中printf传递的是字符串常量，所以我们将主要的研究目标放到sub_401000函数上，我们可以简单查看sub_401000函数的汇编代码



我们简单阅读这段汇编代码，可以知道这一段并没有使用call strcpy，由此可以知道，将函数封装到了sub_401000函数内部。然后我们发现存在两个变量arg_0和var_14（我们可以通过F5查看伪代码发现一个参数一个局部变量），然后通过sub esp对esp的移动可以发现栈的大小为20（14h），然后将var_14临时变量的地址存储到了edi中，同时将输入的arg_0座位源字符串，也就是说，如果我们输入的字符串长度大于20（14h），就有可能发生栈溢出。

- 然后对我们上面得出的结论进行测试，可以得到报错，这里我们需要满足输入的字符串中含有overflow以确保可以通过strstr的函数判断，同时字符串长度大于20（14h）



- 根据课本上的案例，我们撰写fuzzer代码，代码如下

```
#include <stdio.h>
#include <windows.h>
void main(int argc,char *argv[])
{
    char buf[50] = "overflow";
    if(argc > 1)
    {
        for(int i = 0;i<20;i++)
        {
            buf[8+i] = 'a';
            printf("填入i个字符%d,字符串%s",i+1,buf);
```

```
            ShellExecute(NULL,"open",argv[1],buf,NULL,SW_NORMAL);
        }
    }
    else
    {
        printf("No exe to fuzzer");
    }
}
```

因为我两个代码不在一个工程内，所以我将下面的fuzzer代码编译后的exe复制到了上面exe的文件夹

- 运行fuzzer程序，我们一共得到了9个报错，也就是又11个不报错的，而overflow和句尾符共有九位，所以是20位溢出符合我们上面的观察