# 电子科技大学

## 计算机专业类课程

# 实验报告

课程名称： 栈溢出基础实验

学　　院： 计算机科学与工程学院

专　　业： 网络空间安全

学生姓名： 黄鑫

学　　号： 2021050901013

指导教师： 牛伟纳

日　　期： 2023 年　10 月　29 日

# 电 子 科 技 大 学

# 实 验 报 告

# 实验一

## 一、实验名称：实验环境的建立

## 二、实验学时：4

## 三、实验内容和目的：

（1）实验内容

a. 寻找目标程序中存在栈溢出漏洞的函数或代码段。

b. 构造恶意输入，这个输入将导致栈溢出。

c. 利用栈溢出，覆盖函数的返回地址，以便控制程序执行流程。

d. 在返回地址中设置指向你自己编写的代码的地址，这段代码将打印学号和姓名。

e. 当程序执行到你设置的代码时，它将执行学号和姓名的打印操作。

（2）实验目的：

加深简单栈溢出的理解

掌握基础的 ida 使用

## 四、实验原理

栈溢出是一种常见的安全漏洞，通常发生在程序没有正确验证用户输入的情况下。当用户提供超出程序所分配栈空间的输入时，这种漏洞就会出现。攻击者可以将恶意代码放置在栈上，然后通过覆盖函数的返回地址，控制程序执行的下一个指令。

在这个实验中，你的目标是构造输入，以便在函数返回时，将返回地址设置为指向你自己编写的代码块。这个代码块将负责打印你的学号和姓名。通过这种方式，你可以控制程序的执行流程，以实现自己的目标，即打印学号和姓名。

这个实验是学习计算机安全的一个重要部分，帮助学生了解安全漏洞的利用和修复方法。但请注意，栈溢出攻击是非法的，只能在受控环境中进行，以合法的方式学习和实践相关技术。

## 五、实验器材（设备、元器件）

PC 机，IDA8.3freesoftwar，Hxd 文本分析软件

## 六、实验数据及结果分析：

设置实验环境：在合法、受控的环境中进行实验，例如虚拟机或容器。确保你有足够的权限来运行实验。

识别漏洞函数：使用逆向工程工具（例如 IDA Pro）来分析目标程序的代码，识别包含栈溢出漏洞的函数或代码段。

构造恶意输入：创建一个特定的输入，该输入将导致栈溢出。这可能包括输入的长度超过了栈分配的缓冲区大小。

确定返回地址偏移：通过分析目标程序的栈结构，确定恶意输入中覆盖返回地址所需的偏移量。

编写恶意代码：编写一段用于打印学号和姓名的恶意代码，可以使用汇编语言或其他低级语言编写。确保这段代码不会引起进一步的栈溢出。

覆盖返回地址：在构造的恶意输入中，将返回地址覆盖为指向你的恶意代码。这需要计算返回地址的实际地址，考虑到栈的偏移量。

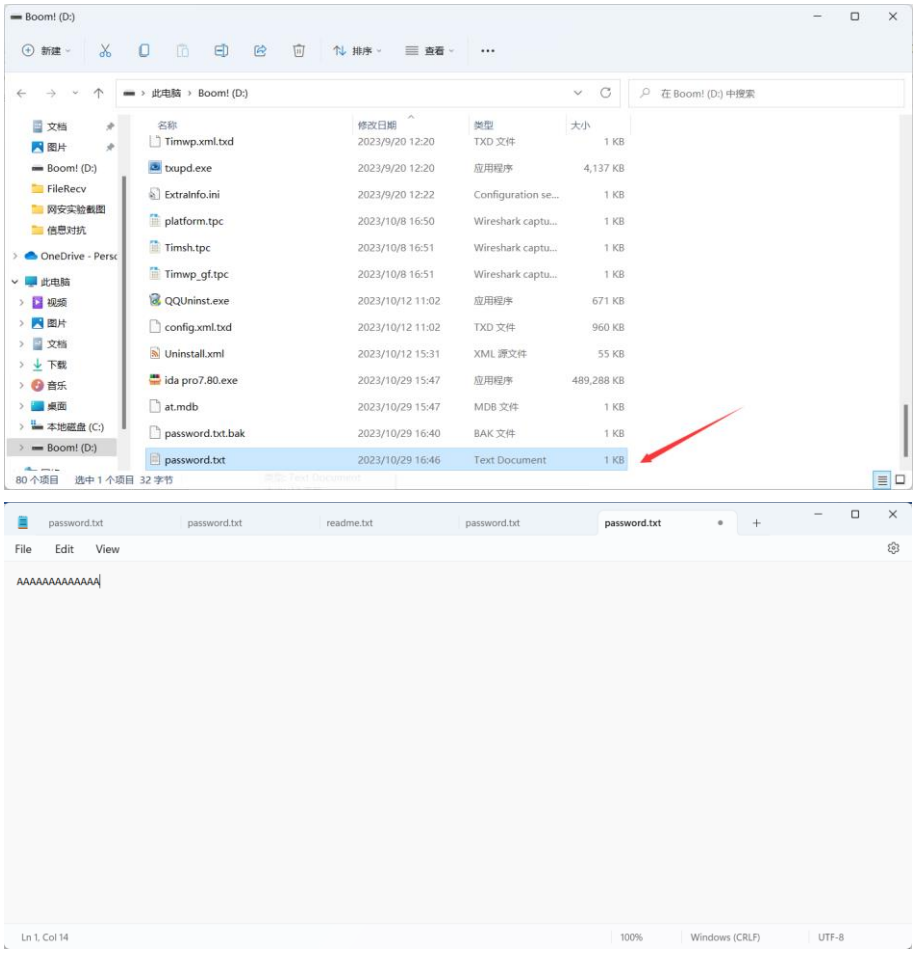
执行攻击：运行目标程序并提供构造的恶意输入。程序应该在返回时跳转到你的恶意代码，从而打印学号和姓名。

分析结果：确认实验是否成功，学号和姓名是否被打印出来。如果失败，可能需要进行进一步的分析和调试。

清理环境：完成实验后，确保关闭或重置实验环境，以防止进一步的不安全操作。

## 七、实验数据及结果分析：

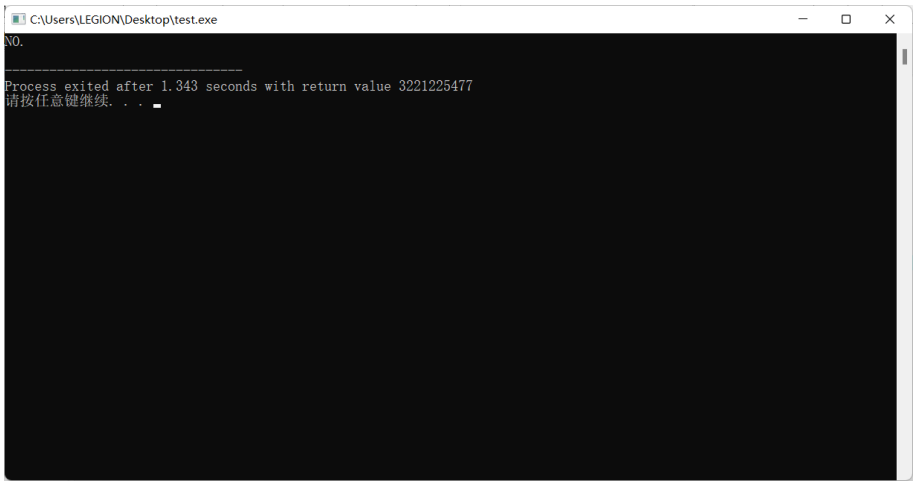
1. 在 D 盘中新建 password.txt 文件



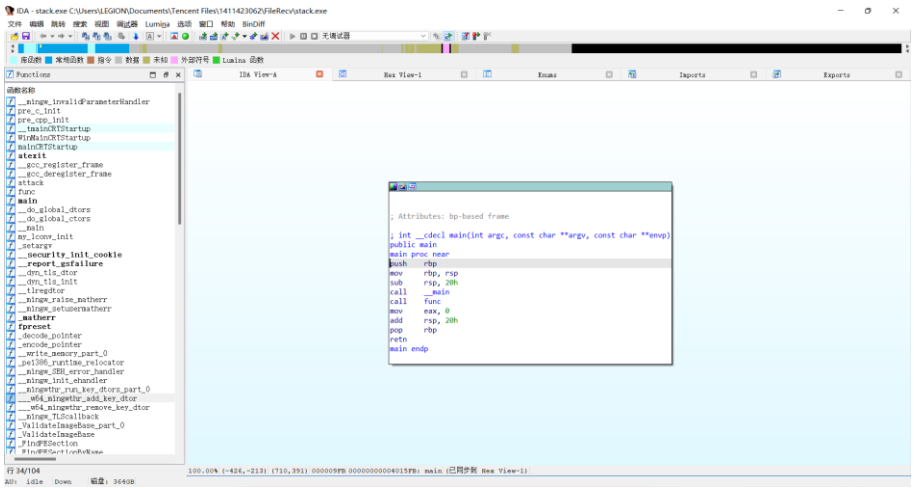

2. 编写栈溢出实验代码 stack.cpp，通过打开 password.txt 文件实现实验操作，因为与 password 数组中的密码不相同，所以会出现 NO，但不会出现 attck 函数中的内容。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void attack()
{
    printf("黄鑫 2021050901013");       //attack 函数
}
void func()                    //func 函数
{
    char password[6] = "ABCDEF";
    char str[6];
    FILE *fp;
```

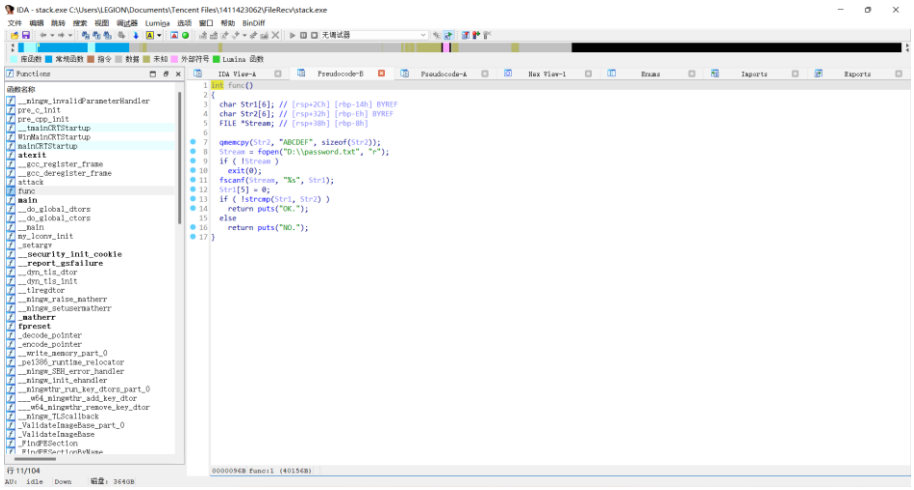
```
13.     if(!(fp=fopen("D:\\password.txt","r"))) //打开 D 盘的 password.txt 文件
14.         exit(0);
15.     fscanf(fp,"%s",str);            //将 str 的内容写入 fp
16.     str[5]='\0';
17.     if(strcmp(str,password)==0)     //判断 str 是否与 password 相同
18.         printf("OK.\n");
19.     else
20.         printf("NO.\n");
21. }
22. int main()
23. {
24.     func();           //运行 func 函数
25.     return 0;
26. }
```



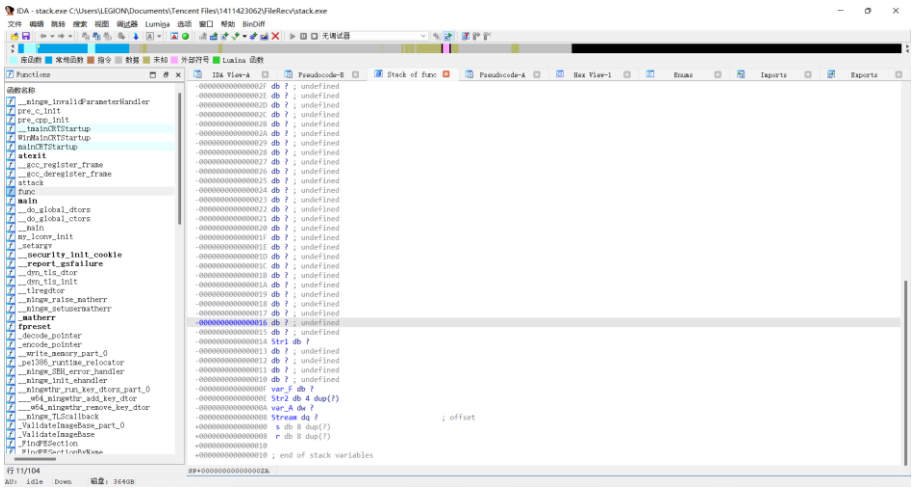

3. 将 stack.cpp 编译生成的可执行文件 stack.exe 放进 IDA 进行编译分析



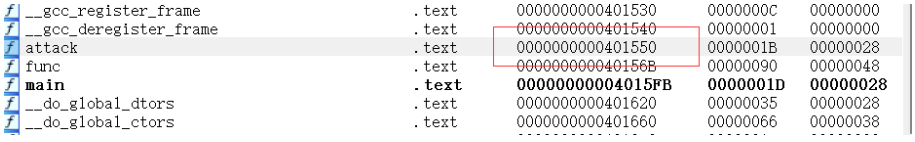
4. 点击函数中的 func 函数，然后按 F5 进入伪代码模式，



5. 查看各个变量的栈位置

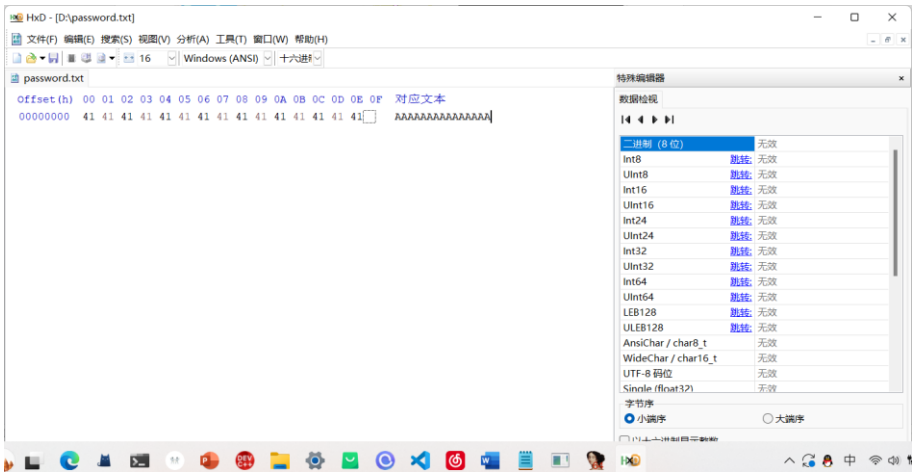


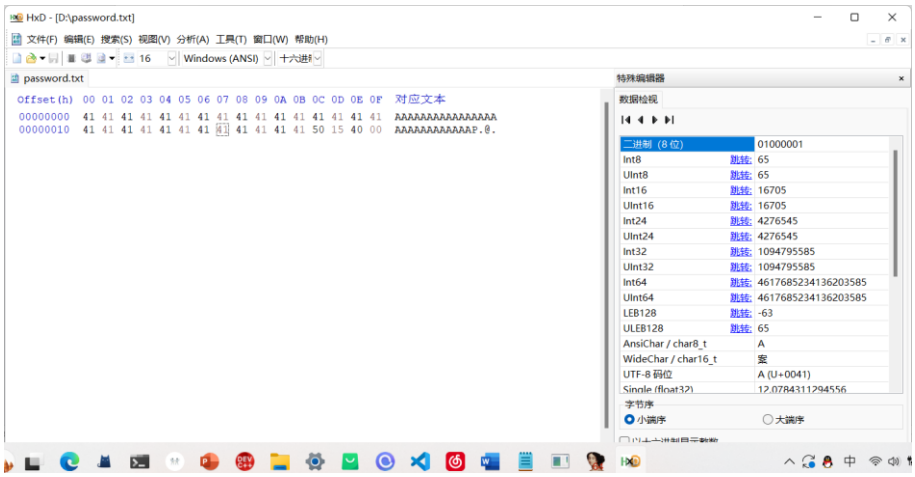
6. 通过分析得 str1 数组也就是代码中的 str 数组用于接收代码从 password.txt 文件中读取得字符串，str1 数组到 r 也就是 re 他返回地址总共有 28 个字节，所以只要让 password.txt 文件中得字符长度等于 28 字节，然后再 28 字节后添加上 attack 得地址，即可让 ret 读取 attack 地址对应得函数从而输出，所以查看 attack 函数地址



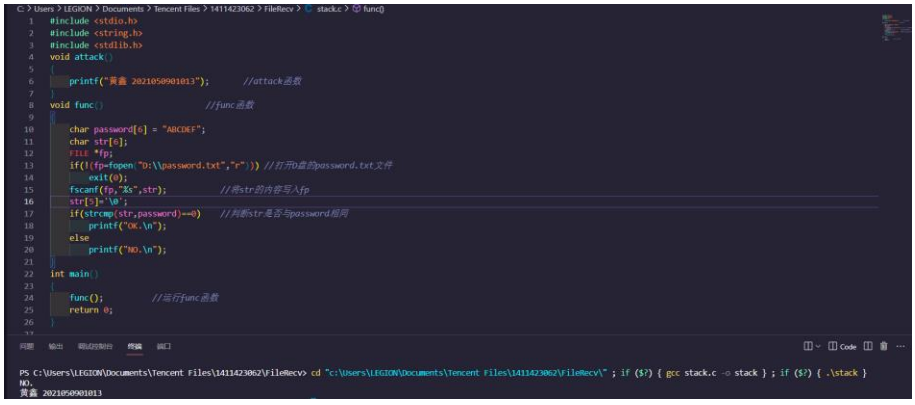
| | | | | |
|---|---|---|---|---|
| __gcc_register_frame | .text | 0000000000401530 | 0000000C | 00000000 |
| __gcc_deregister_frame | .text | 0000000000401540 | 00000001 | 00000000 |
| attack | .text | 0000000000401550 | 0000001B | 00000028 |
| func | .text | 000000000040156B | 00000090 | 00000048 |
| **main** | **.text** | **00000000004015FB** | **0000001D** | **00000028** |
| __do_global_dtors | .text | 0000000000401620 | 00000035 | 00000028 |
| __do_global_ctors | .text | 0000000000401660 | 00000066 | 00000038 |

7. 通过HxD文本分析工具打开password.txt，在第28字节位置处添加上attack函数的地址，因为本主机所对应的cpu是小端序，所以加入：50 15 40 00。




8. 再次运行代码，即可看到结果已经成功的将attck函数成功运行，打印出我们的学号以及姓名，说明通过栈上溢出的数据实现了成功的覆盖，从而达到攻击的目的。



## 八、实验结论、心得体会和改进建议：

1.实验结论：

通过本次栈溢出基础实验，我成功地构造了恶意输入，覆盖了函数的返回地址，以控制程序执行流程，并在执行中打印出了我的学号和姓名。这实验证明了栈溢出漏洞的危险性，以及攻击者如何利用这种漏洞来执行恶意代码。同时，我也学到了如何使用逆向工程工具（IDA Pro）来分析目标程序的代码，识别漏洞函数和栈结构，以确定返回地址的偏移量。

2.心得体会：

这次实验是计算机安全领域的一次深入学习，让我更加了解了栈溢出漏洞的工作原理和攻击方法。通过实际操作，我深刻体会到了安全编程的重要性，以及编写安全代码的必要性。同时，我也明白了合法环境下进行学习和实践相关技术的必要性，以确保不会滥用这些知识。

3.改进建议：

在今后的实验中，可以考虑增加更多的安全实践内容，包括如何预防和修复栈溢出漏洞，以及如何编写更安全的代码。

总的来说，这次实验为我提供了宝贵的安全知识，帮助我更好地理解计算机安全领域的重要性和挑战，同时也增强了我的技能和意识，以更好地保护计算机系统的安全。