软件安全实验

北京邮电大学信息安全中心
张淼
zhangmiao@bupt.edu.cn

第三讲 漏洞挖掘与模糊测试

- 漏洞挖掘技术简介
- 简单的fuzzing演示
- 文件类型漏洞挖掘
- FTP漏洞挖掘

一、漏洞挖掘技术简介

- 漏洞挖掘技术简介
- 简单的fuzzing演示
- 文件类型漏洞挖掘
- FTP漏洞挖掘

一、漏洞挖掘技术简介

作为攻击者，除了精通各种漏洞利用技术之外，要想实施有效的攻击，还必须掌握一些未公布的0day漏洞；作为安全专家，他们的本质工作就是抢在攻击者之前尽可能多地挖掘出软件中的漏洞。

那么，面对着二进制级别的软件，怎样才能在错综复杂的逻辑中找到真正的漏洞呢？

一、漏洞挖掘技术简介

工业界目前普遍采用的是进行Fuzz测试，与基于功能性的测试有所不同，Fuzz的主要目的是”崩溃crash”,”中断break”,”销毁destroy”。

一、漏洞挖掘技术简介

Fuzz的测试用例往往是带有攻击性的畸形数据，用以触发各种类型的漏洞。

我们可以把Fuzz理解成为一种能自动进行”rough attack”尝试的工具。之所以说它是”rough attack”,是因为Fuzz往往可以触发一个缓冲区溢出的漏洞，但却不能实现有效的exploit。

测试人员需要实时地捕捉目标程序抛出的异常、发生的崩溃和寄存器等信息，综合判断这些错误是不是真正的可利用漏洞

一、漏洞挖掘技术简介

Fuzz技术的思想就是利用“暴力”来实现对目标程序的自动化测试，然后监视检查其最后的结果，如果符合某种情况就认为程序可能存在某种漏洞或者问题。

这里的“暴力”并不是说我们通常说得武力，而是说利用不断地向目标程序发送或者传递不同格式的数据来测试目标程序的反应。

一、漏洞挖掘技术简介

Fuzz的测试优点是很少出现误报，能够迅速地找到真正的漏洞；缺点是Fuzz永远不能保证系统里已经没漏洞。即使我们用Fuzz找到了100个严重的漏洞，系统中仍然可能存在第101个漏洞

一、漏洞挖掘技术简介

攻击者非常热衷于工具，因为软件系统的安全性并不是他们关心的事情，他们只要找到漏洞就可以了。

但是对于研究安全问题的学者则不同，他们更关心如何能够检测出所有的漏洞（尽管不可能）。因此，学术界偏向于对源代码进行静态分析，直接在程序的逻辑上寻找漏洞。

一、漏洞挖掘技术简介

静态分析这一方面的方法和理论有很多，比如数据流分析、类型验证系统、边界检验系统、状态机系统等。

静态代码分析技术的缺点是经常会产生大量的误报—如果分析工具一次产生了上千个漏洞警告，几乎没有人愿意一个一个地去排查。由于黑客们一般情况下是得不到源代码的，所以使用白盒测试方法的以QA工程师居多。

一、漏洞挖掘技术简介

动态测试技术与静态代码审计的对比

	动态测试技术	静态代码审计
主要技术	Fuzz等黑盒测试	数据流分析、类型验证系统、边界检验系统、状态机系统等
主要应用人群	攻击者	学者、QA工程师
优点	快捷，准确	检测出的漏洞数量多
缺点	不能发现系统里全部（或者大部分）漏洞	实现静态代码分析工具相当困难

二、简单的fuzzing演示

- 漏洞挖掘技术简介
- 简单的fuzzing演示
- 文件类型漏洞挖掘
- FTP漏洞挖掘

二、简单的fuzzing演示

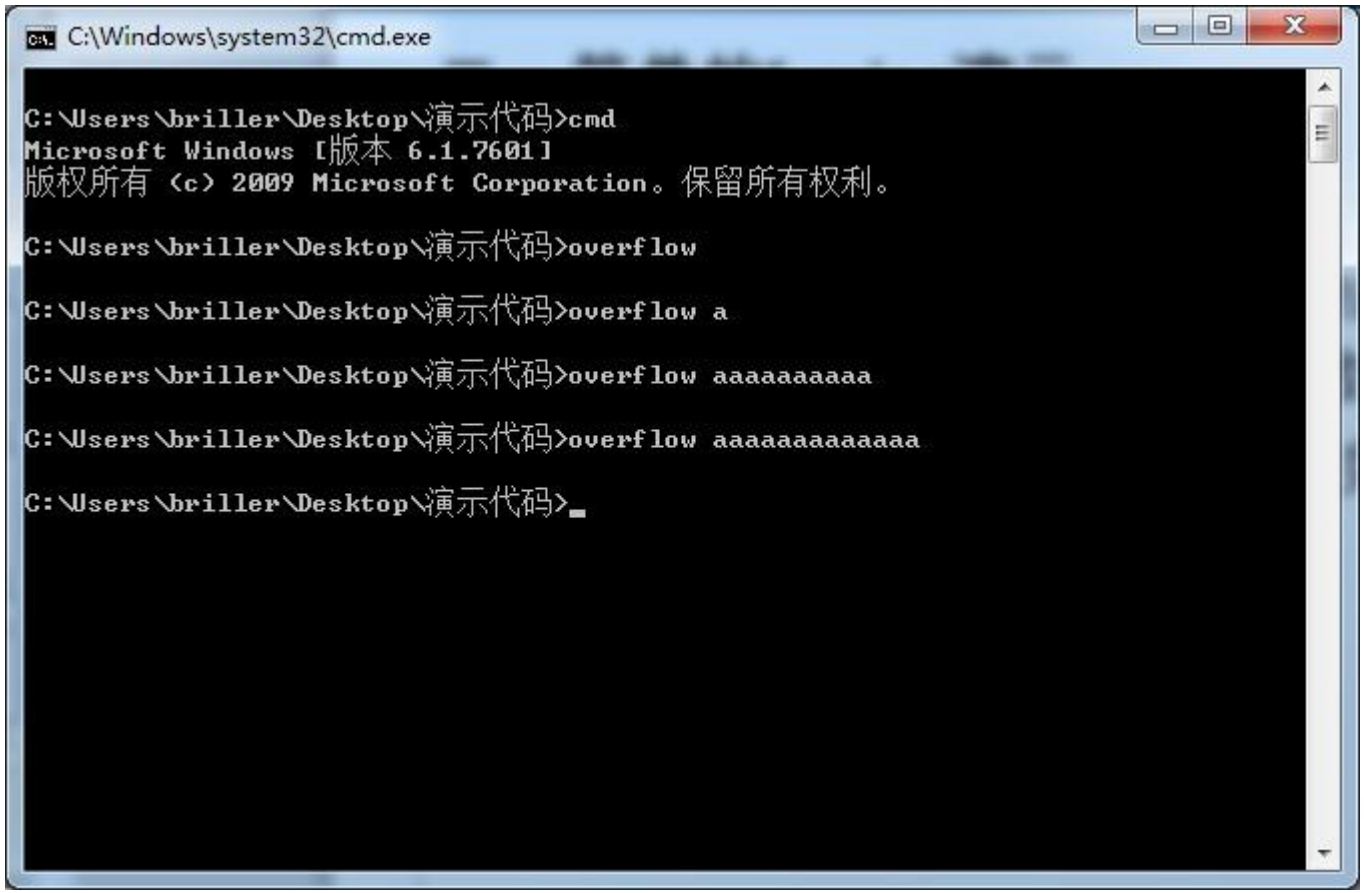
我们在发现一些溢出漏洞的时候，往往是不断地给目标程序输入不同长度的字符串变量来测试目标程序是不是存在溢出漏洞的。

二、简单的fuzzing演示

现在我们有一个名为overflow的exe文件，我们给它指定的参数为a，然后逐步加长参数的长度。

我们这样做的话，会发现如下的效果：

二、简单的fuzzing演示



二、简单的fuzzing演示

假如要是这样一点一点累加，却不知道什么时候才能有所反应，我们难免会不耐烦，这样我们就需要做一个自动化的工具，这也就是简单的fuzz雏形。我们看一下程序的关键代码。

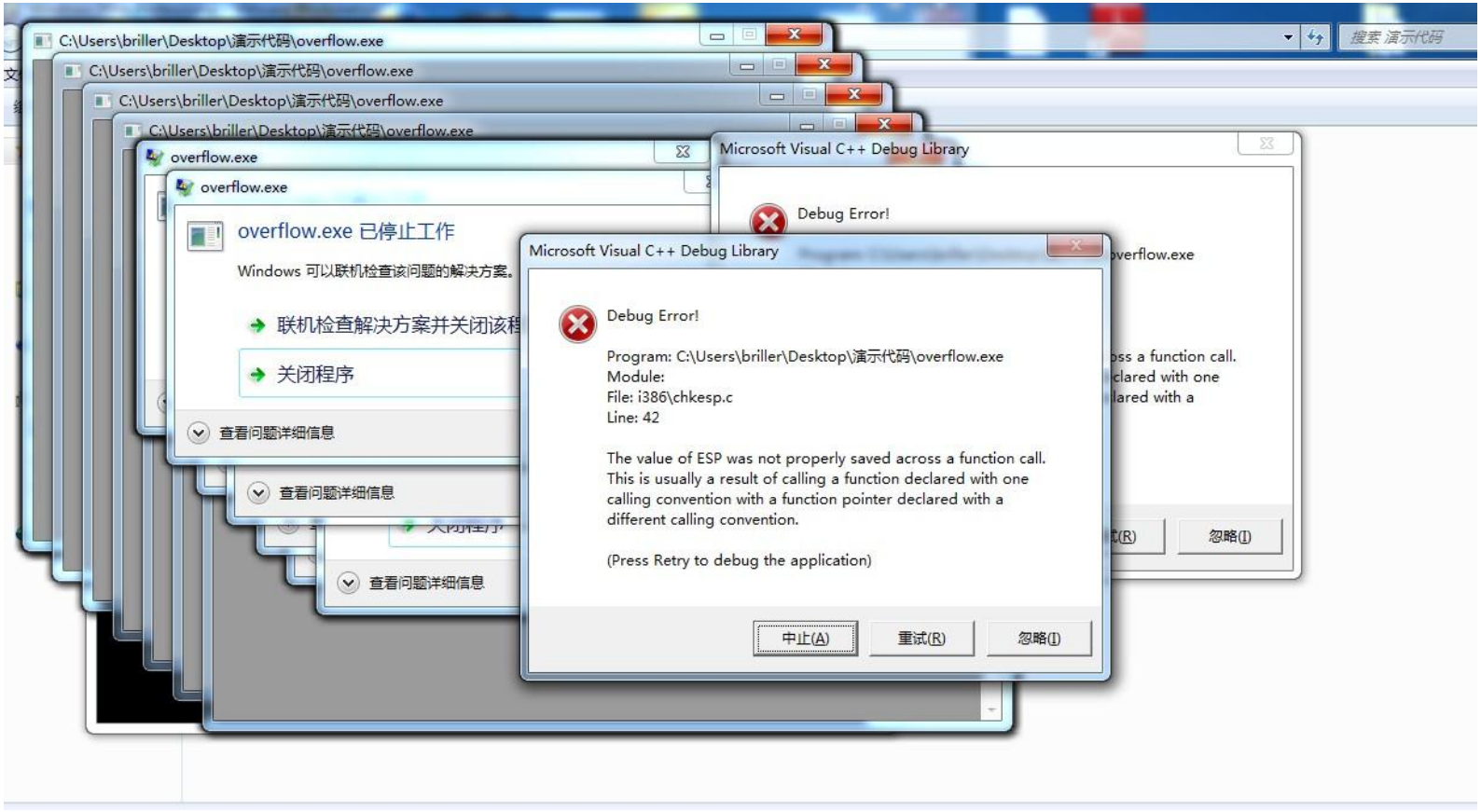
二、简单的fuzzing演示

```
for(int i=380;i<470;i=i+2)
{
testbuf=new char[i];
 memset(testbuf,0,i);
 memset(testbuf,'c',i);
 memcpy(buf,testbuf,i);
 //printf("%s\n",buf);
ShellExecute(NULL,"open",argv[1],buf,NULL,SW_NOR
MAL);
delete testbuf;
}
```

二、简单的fuzzing演示

上述的程序就是不断地调用我们指定的程序，并且将参数逐渐增加，会不会产生溢出漏洞呢？我们来看看运行程序后的结果。

二、简单的fuzzing演示



二、简单的fuzzing演示

我们从上面的演示程序可以发现，再不断地增加输入参数的字符串长度时，出现了错误的提示，说明程序很有可能出现了缓冲区溢出的漏洞，这就是简单的fuzzing的原理的演示

我们大可不必这样自己来写这么丑陋的程序进行fuzz而去使用现成的fuzz工具，也叫做fuzzer。这会使我们的效率事半功倍。

三、文件类型漏洞挖掘

- 漏洞挖掘技术简介
- 简单的fuzzing演示
- 文件类型漏洞挖掘
- FTP漏洞挖掘

三、文件类型漏洞挖掘—文件格式Fuzz的基本方法

不管IE还是OFFICE，它们都有一个共同点，那就是用文件作为程序的主要输入。从本质上来说，这些软件都是按照事先约定好的数据结构对文件中不同的数据域进行解析，以决定用什么颜色、在什么位置显示这些数据。

三、文件类型漏洞挖掘—文件格式Fuzz的基本方法

假如我们习惯了这样的思维，也就是假设他们所使用的文件是严格遵守软件规定的数据格式的。因为用Word生成的doc文件一般不会存在什么非法数据。

但是！！攻击者往往会在约定的数据格式进行稍许修改，观察软件在解析这种“畸形文件”时是否会发生错误，缓冲区是否会溢出等。

三、文件类型漏洞挖掘—文件格式Fuzz的基本方法

文件格式Fuzz就是利用这种“畸形文件”测试软件的方法。我们可以在网上找到很多FileFuzz的工具。

一个File Fuzz工具通常包括以下几个步骤：

(1)以一个正常的文件模板作为基础，按照一定规则产生一批畸形文件。

(2)将畸形文件逐一送入软件进行解析，并监视软件是否会抛出异常。

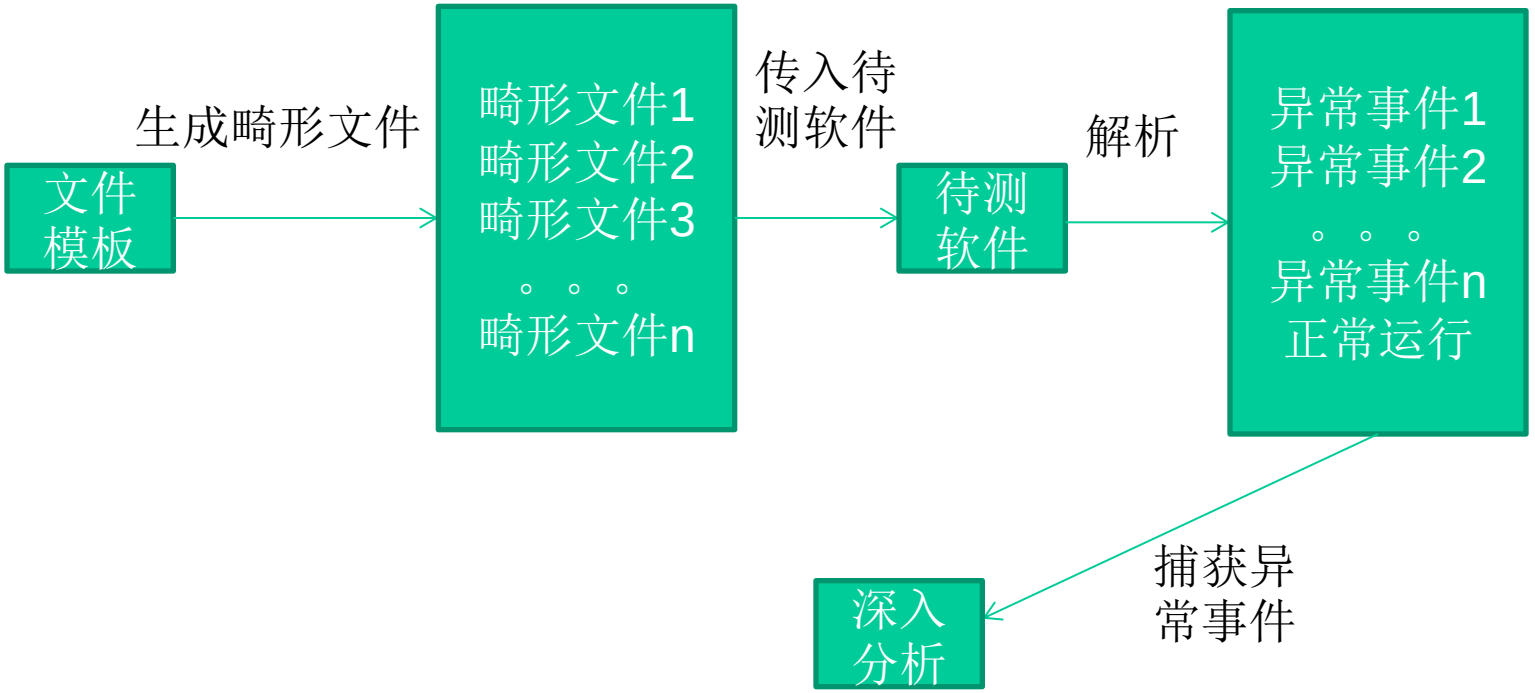
三、文件类型漏洞挖掘—文件格式Fuzz的基本方法

一个File Fuzz工具通常包括以下几个步骤：

(3)记录软件产生的错误信息，如寄存器状态、栈状态等。

(4)用日志或其他UI形式向测试人员展示异常信息，以进一步鉴定这些错误是否能被利用。

三、文件类型漏洞挖掘—文件格式Fuzz的基本方法



三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

Blind Fuzz即通常所说的“盲测”，就是在随机位置插入随机的数据以生成畸形文件。然而现代软件往往使用非常复杂的私有数据结构。

例如PPT、word、excel、mp3、RMVB、PDF、jpg、ZIP压缩包，加壳的PE文件。数据结构越复杂，解析逻辑越复杂，就越容易出现漏洞。复杂的数据结构通常具备以下特征：

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

1）拥有一批预定义的静态数据，如magic,cmd id等

2）数据结构的内容是可以动态改变的

3）数据结构之间是嵌套的

4）数据中存在多种数据关系(size of,point to, reference of,CRC)

5）有意义的数据被编码或压缩，甚至用另一种文件格式来存储，这些格式的文件被挖掘出来越来越多的漏洞……

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

对于采用复杂数据结构的复杂文件进行漏洞挖掘，传统的Blind Fuzz暴露出一些不足之处。

例如：产生测试用例的策略缺少针对性，生成大量无效测试用例，难以发现复杂解析器深层逻辑的漏洞等。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

针对Blind Fuzz的不足，Smart Fuzz被越来越多地提出和应用。通常Smart Fuzz包括三方面的特征：

面向逻辑(Logic Oriented Fuzzing)

面向数据类型(Data Type Oriented Fuzzing)

基于样本(Sample Based Fuzzing)。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

面向逻辑：测试前首先明确要测试的目标是解析文件的程序逻辑，而不是文件本身。

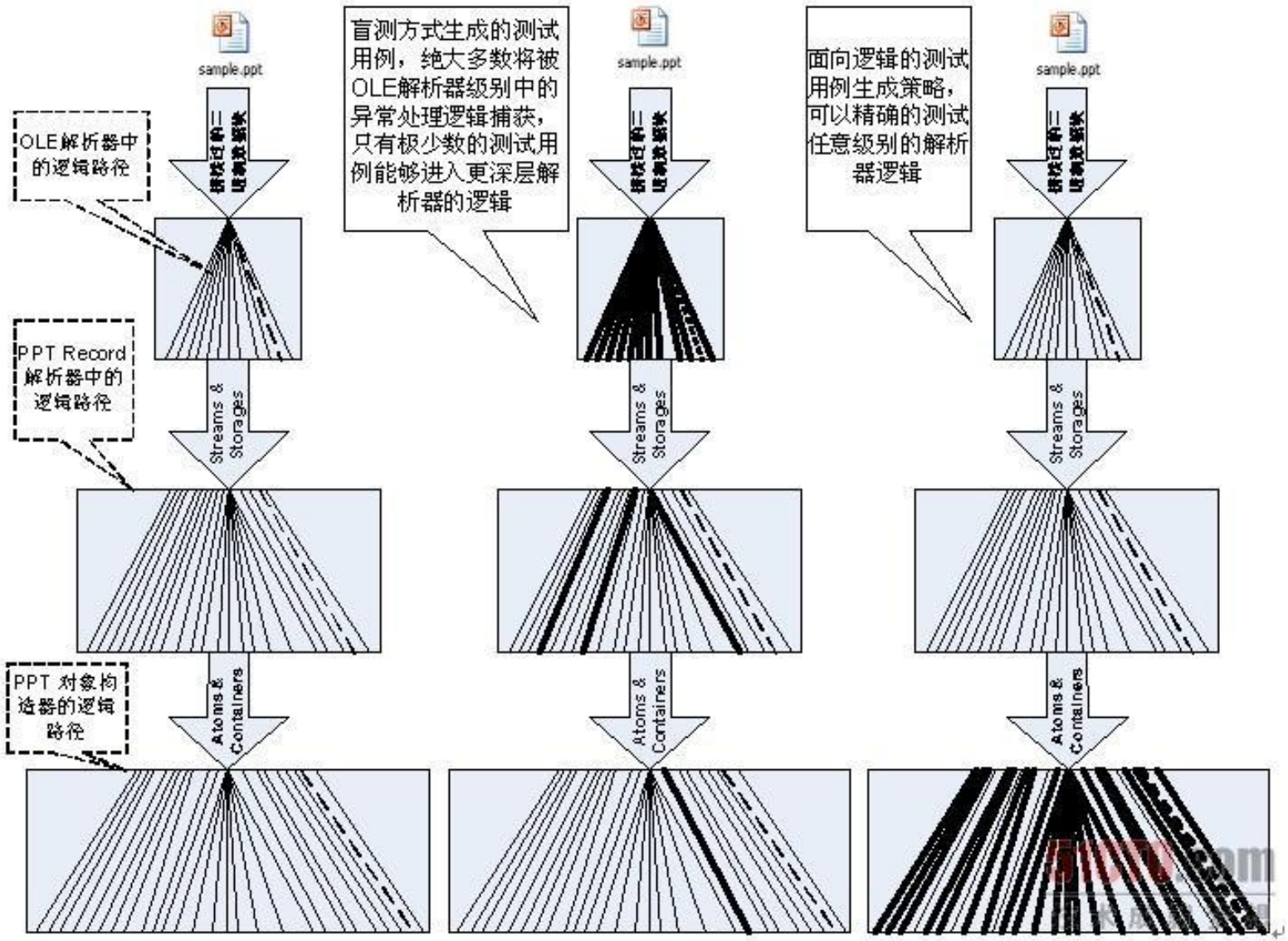
复杂的文件格式往往要经过多“层”解析，因此还需要明确测试用例正在试探的是哪一层的解析逻辑，即明确测试“深度”以及畸形数据的测试“粒度”。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

面向逻辑：明确了测试的逻辑目标后，在生成畸形数据时可以具有针对性的仅仅改动样本文件的特定位置，尽量不破坏其他数据一来关系，这样使得改动的数据能够传递到测试的解析深度，而不会在上层的解析器中被破坏。

下图是对一个PPT文件进行面向逻辑测试和盲测的比较。可以看出，盲测中生成的畸形文件都被阻断在第一层解析器OLE Parser上，而面向逻辑测试生成畸形文件可以顺利通过，达到下一层。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz



三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

面向数据类型测试：测试中可以生成的数据通常包括以下几种类型。

1）算术型：包括以HEX、ASCII、Unicode、Raw格式存在的各种数值。

2）指针型：包括Null指针、合法/非法的内存指针等。

3）字符串型：包括超长字符串、缺少终止符(0x00)的字符串等。

4）特殊字符：包括#，@，’，<,>,/,\,../等等。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

面向数据类型测试：

面向数据类型测试是指能够识别不同的数据类型，并且能够针对目标数据的类型按照不同规则来生成畸形数据。

跟Blind Fuzz相比，这种方法产生的畸形数据通常都是有效的，能够大大减少无效的畸形文件。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

基于样本：

测试前首先构造一个合法的样本文件（也叫模版文件），这时样本文件里所有数据结构和逻辑必然都是合法的。然后以这个文件为模板，每次只改动一小部分数据和逻辑来生成畸形文件，这种方法也叫做“变异”(Mutation)。

对于复杂文件来说，以现成的样本文件为基础进行畸形数据变异来生成畸形文件的方法要比上面两种的难度要小很多，也更容易实现。

三、文件类型漏洞挖掘—Blind Fuzz和Smart Fuzz

基于样本：

但是这种方法不能测试样本文件里没有包含的数据结构，比如一个文件格式包含18种数据区块(Chunk)，而给定的样本文件中只用到了其中的10种，那么基于样本测试的方法只会修改这10种区块的数据来产生畸形文件，测试不到其他8种数据对应的解析逻辑。

为了提高测试质量，就要求在测试前构造一个能够包含几乎所有数据结构的文件（比如文字、图像、视频、声音、版权信息等数据）来作为样本

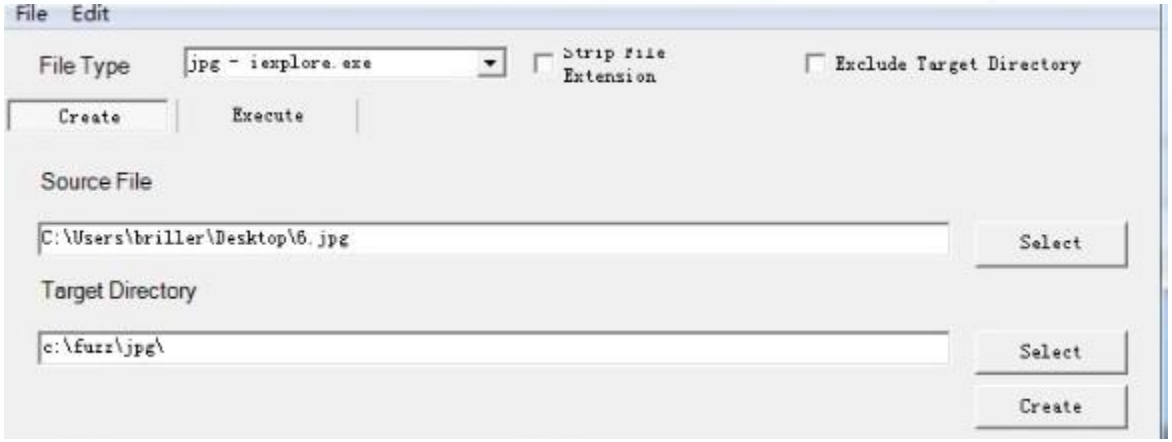
三、文件类型漏洞挖掘—FileFuzz简单演示

了解了FileFuzz的基本原理和方法后，我们来使用一款软件来看看FileFuzz的流程，我们这个软件名称就叫做FileFuzz

这个软件在互联网上就能就能够搜索到，安装也是和其他软件相似。

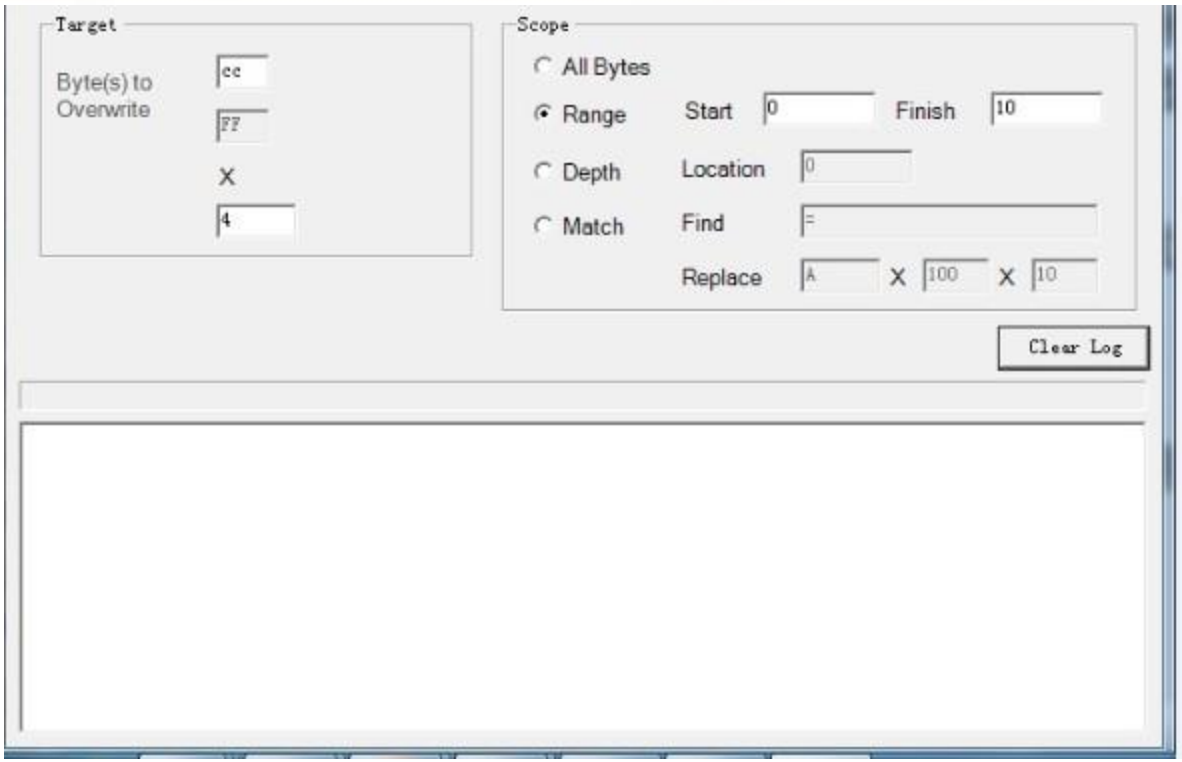
下面我们简介一下使用方法。

三、文件类型漏洞挖掘—FileFuzz简单演示



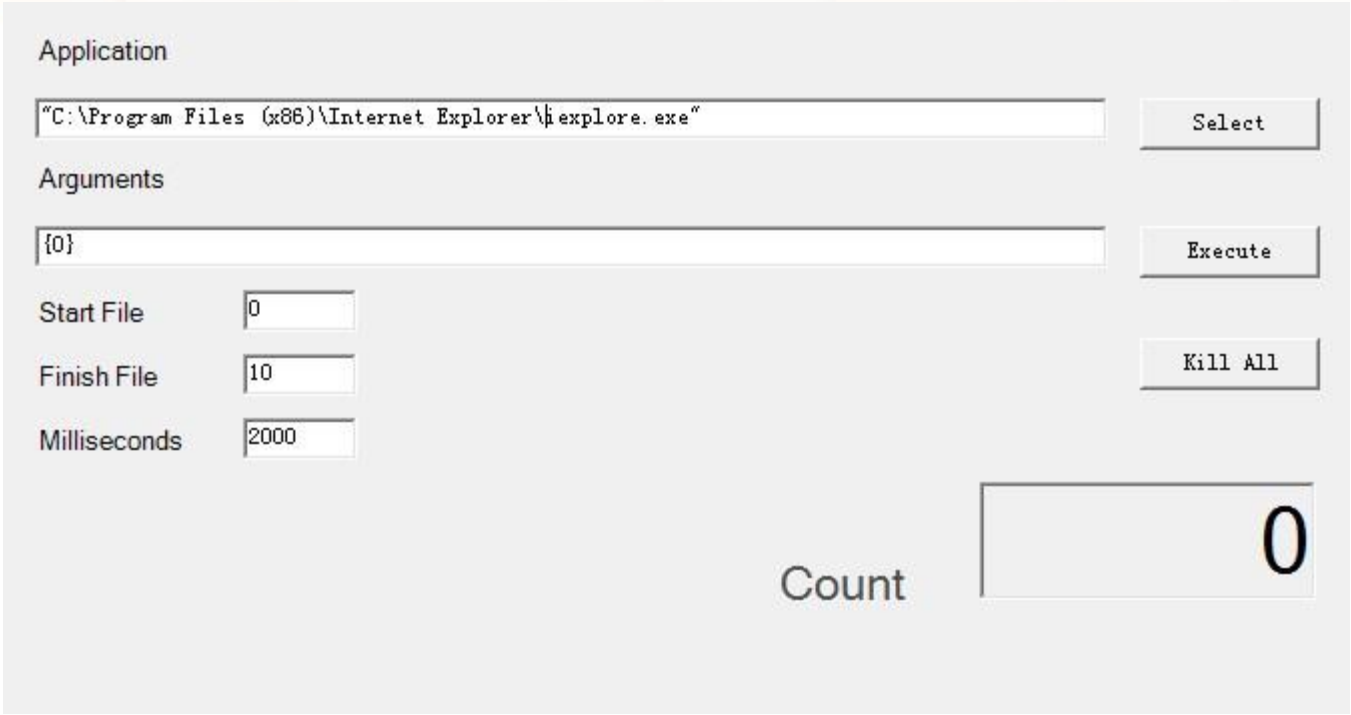
我们能看到，我们现在选择的是jpg格式的文件，打开的软件用的是iexplore，create选项卡表明我们现在是处在创建畸形文件的阶段，接下来我们选择的是文件的模板（源文件）和生成的畸形文件的存放目录

三、文件类型漏洞挖掘—FileFuzz简单演示



我们现在把字节修改为cc，然后是以4为偏移（按字节），作用域选择为0-10，然后我们点击create按钮就能够生成文件名为0.jpg,1.jpg。。。。。10.jpg这些畸形文件。

三、文件类型漏洞挖掘—FileFuzz简单演示



我们调整到execute选项卡，选择到IE的地址，然后设置下参数的形式等选项，点击execute按钮即让IE去解析那11个畸形文件了，然后我们可以查看下log，看是否有异常。

三、文件类型漏洞挖掘—FileFuzz简单演示

```
[*] "crash.exe" "C:\Program Files (x86)\Internet Explorer\iexplore.exe" 2000 c:\fuzz\jpg\8.jpg
[*] Access Violation
[*] Exception caught at 76b43932 mov eax,[esi]
[*] EAX:76b624c8 EBX:005c5340 ECX:005bdc30 EDX:0398e91c
[*] ESI:00000000 EDI:005c5338 ESP:0398e8dc EBP:0398e8e8
```

当发生异常时，这些log会显示出当时的若干寄存器的情况以便于我们进行分析(假如某个寄存器出现了cccccccc或类似情况将更有利于我们分析)。

四、文件类型漏洞挖掘

- 漏洞挖掘技术简介
- 简单的fuzzing演示
- 文件类型漏洞挖掘
- FTP漏洞挖掘

四、FTP漏洞挖掘—FTP协议简介

FTP服务全称为文件传输协议服务，英文全称为File Transfer Protocol，其工作模式采用客户端/服务器的模式，也就是C/S模式。

FTP服务在进行文件信息传输的时候采用FTP协议。FTP协议是基于TCP协议之上的一种明文协议。FTP协议默认情况下工作在21号端口，它的具体命令被规定在RFC0959当中。

四、FTP漏洞挖掘—FTP协议简介

用户从客户端连接远程FTP服务程序的过程可以分为建立连接、传送数据和释放连接3个阶段。具体过程如下。

(1)建立TCP连接。

(2)客户端向FTP服务程序发送USER命令以表示用户自己的身份，然后服务程序要求客户端输入密码。

(3)客户端发送PASS命令，同时将用户名密码发送给远程FTP服务程序。

四、FTP漏洞挖掘—FTP协议简介

用户从客户端连接远程FTP服务程序的过程可以分为建立连接、传送数据和释放连接3个阶段。具体过程如下。

(4)服务端判断并通过认证。

(5)客户端开始利用其它FTP协议命令进行文件操作。

(6)结束此次连接，用QUIT命令退出。

四、FTP漏洞挖掘—FTPFuzz

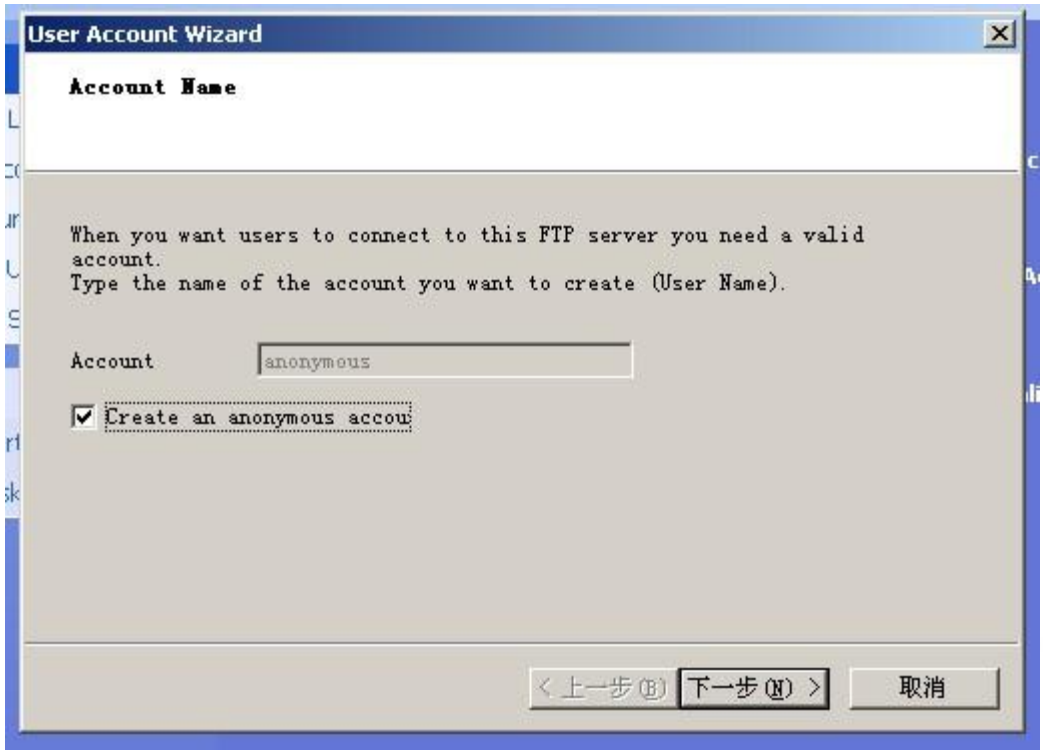
FTPFuzz是专门用来测试FTP协议安全的工具。它的基本原理就是通过对FTP协议中的命令及命令参数进行脏数据替换，构造畸形的FTP命令并发送给被测试FTP服务程序。

我们来做个简单的实验，测试环境如下：

	推荐使用的环境
操作系统(服务端)	Win7
操作系统(客户端)	Vm win2000
Quick'n Easy FTP Server	Lite Version 3.1
FTPFuzz	1.0

四、FTP漏洞挖掘—FTPFuzz

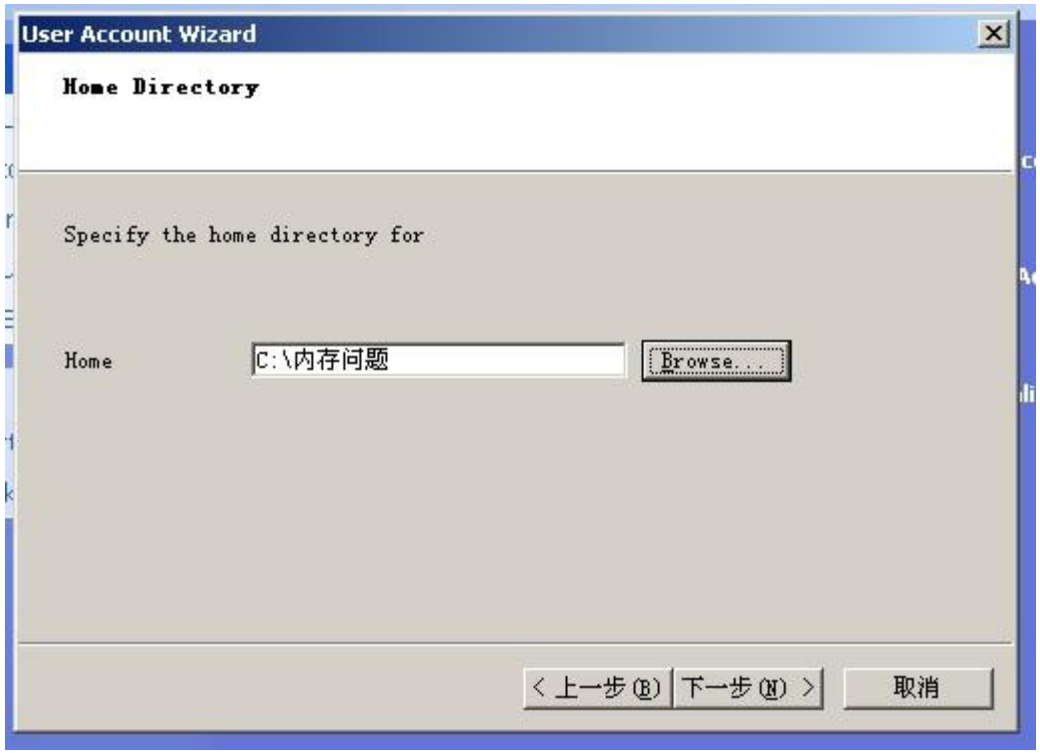
我们先用Quick'n Easy FTP Server来搭建一个简单的FTP服务器，我们看下搭建方法。



第一次运行时先创建一个匿名账号

四、FTP漏洞挖掘—FTPFuzz

我们先用Quick'n Easy FTP Server来搭建一个简单的FTP服务器，我们看下搭建方法。



设置FTP用户的文件目录

四、FTP漏洞挖掘—FTPFuzz

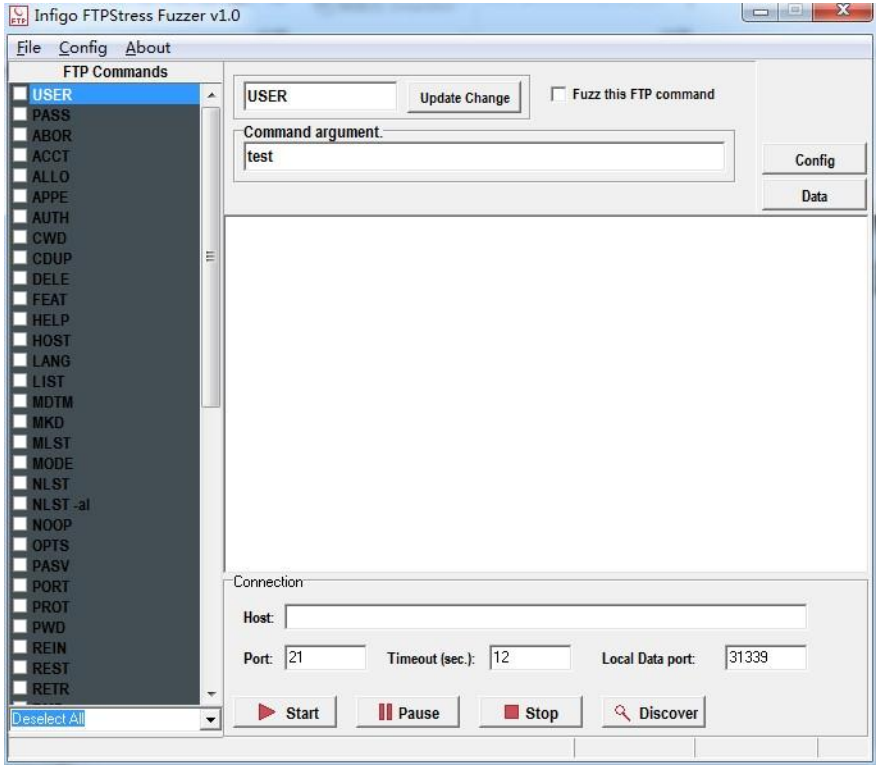
我们先用Quick'n Easy FTP Server来搭建一个简单的FTP服务器，我们看下搭建方法。

设置FTP用户的使用权限

四、FTP漏洞挖掘—FTPFuzz

完成搭建后点击START就可以开始运行FTP服务端了。

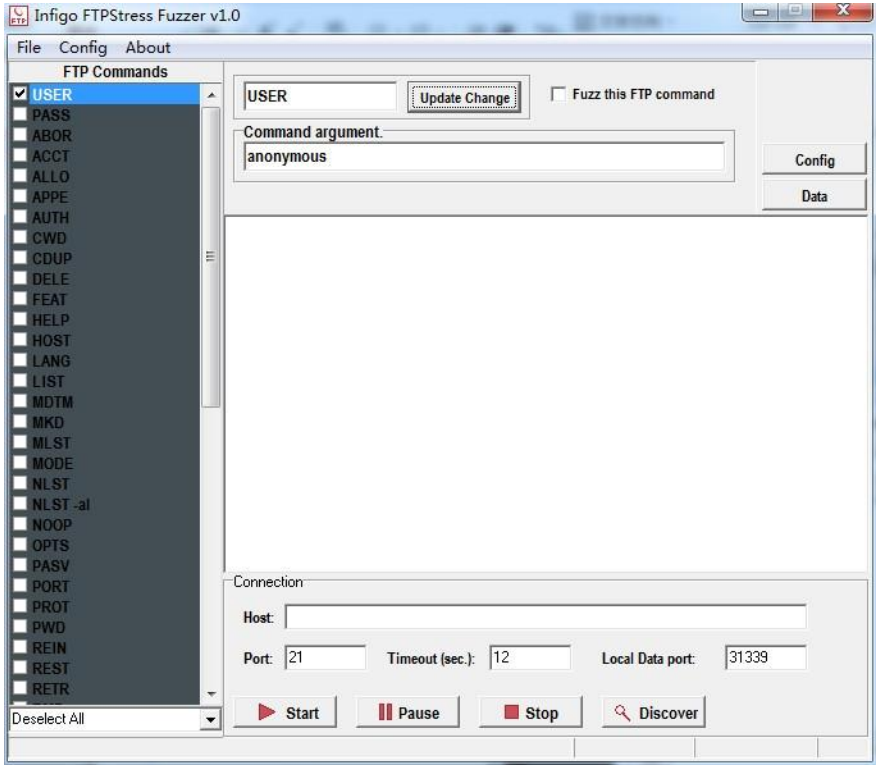
接下来我们看看Infigo FTPStress Fuzzer的使用方法。



我们去掉左侧的全部对勾，或者选择Deselect All

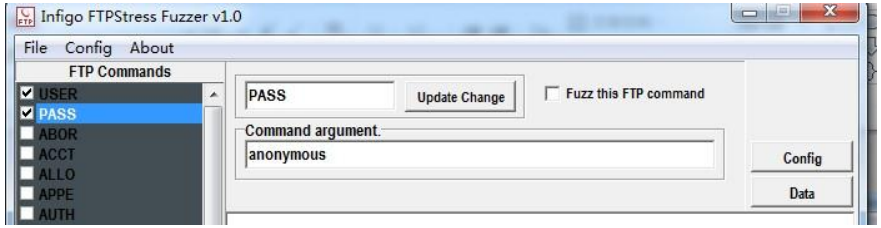
四、FTP漏洞挖掘—FTPFuzz

点选USER对勾，并且把USER的参数设定为anonymous，然后点击update change 按钮保存更改

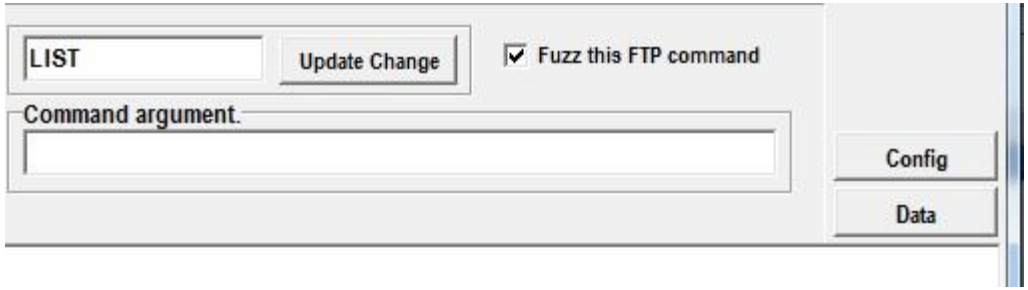


四、FTP漏洞挖掘—FTPFuzz

对PASS参数进行同样的修改

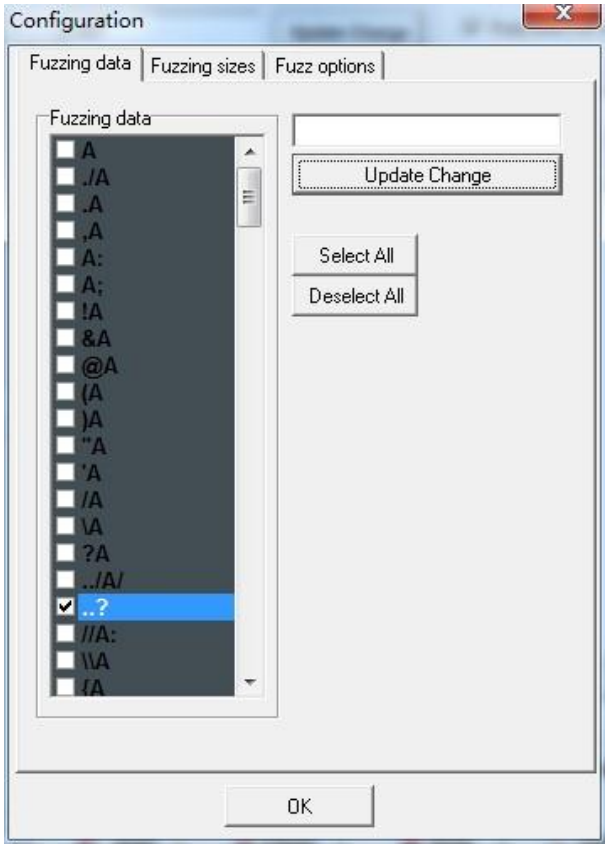


点选LIST选项，选择它为待测命令，Fuzz this FTP Command打勾。



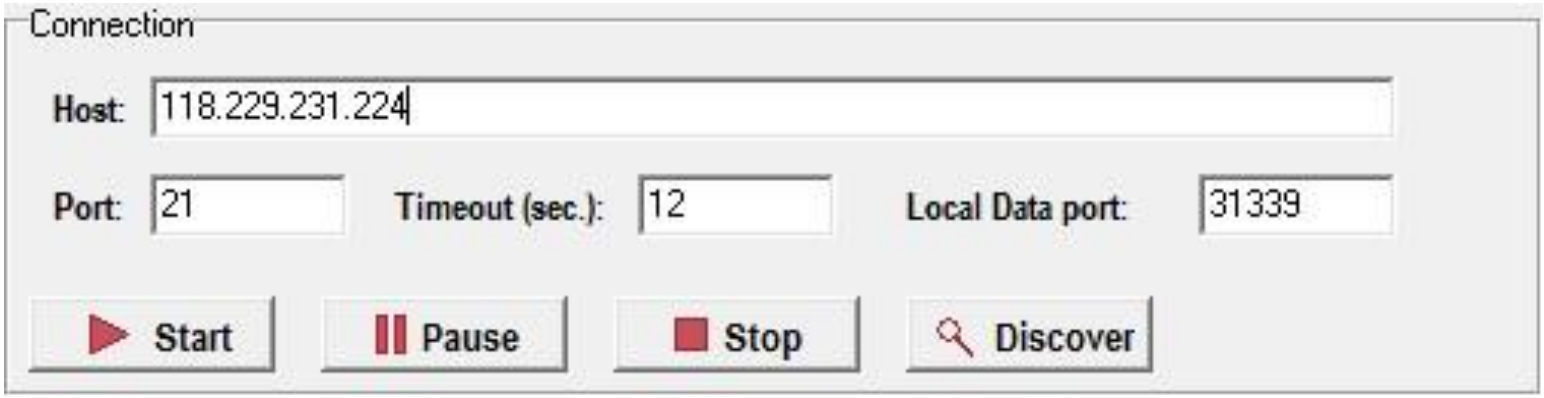
四、FTP漏洞挖掘—FTPFuzz

点击config菜单，然后选中Fuzzing data选项卡，这里面我们就可以设定我们要设置的脏数据，我们先只选择”..?”这一项即可



四、FTP漏洞挖掘—FTPFuzz

都设置好后我们设置FTP主机的IP地址，点击start就可以开始fuzz了。



四、FTP漏洞挖掘—FTPFuzz

启动后可以看到相关的信息

```
[ Connecting to 118.229.231.224:21... ]
[ Connected, starting fuzz process... ]
220 Welcome to Quick 'n Easy FTP Server

[ USER: [anonymous] ]
331 Password required for anonymous

[ PASS: [anonymous] ]
230 User successfully logged in.

[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 30 ]
RECV: 150 Opening ASCII mode data connection for directory list.

[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 70 ]
RECV: 425 Can't open data connection.

[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 150 ]
```

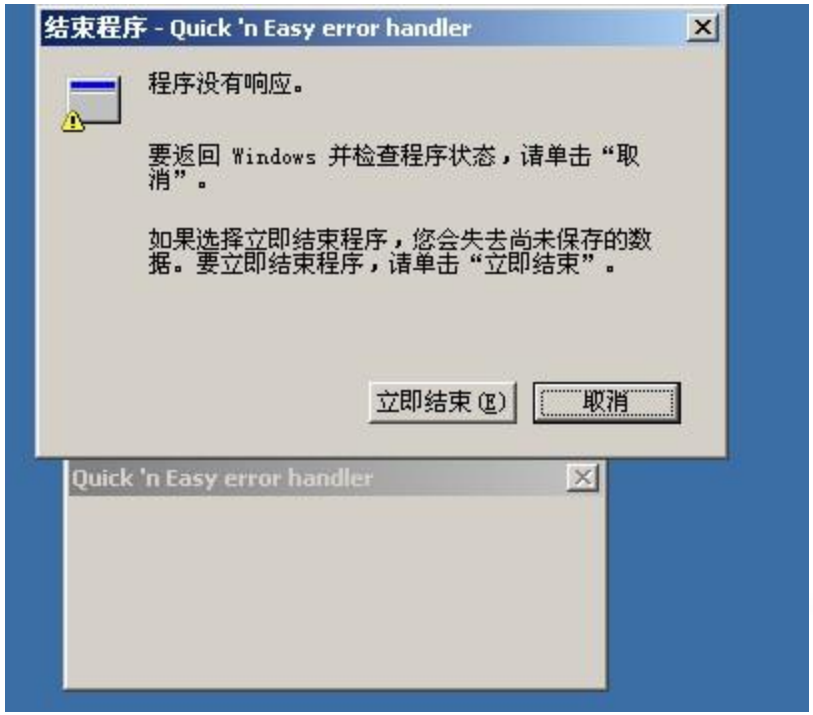
四、FTP漏洞挖掘—FTPFuzz

红字的部分说明了已经fuzz成功了，FTP服务器已经因为脏数据而崩溃。

```
[ USER: [anonymous] ]
[ PASS: [anonymous] ]
[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 2300 ]
[ Connecting to 118.229.231.224:21... ]
[ Connected, starting fuzz process... ]
[ USER: [anonymous] ]
[ PASS: [anonymous] ]
[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 3000 ]
[ Connecting to 118.229.231.224:21... ]
[ Connected, starting fuzz process... ]
[ USER: [anonymous] ]
[ PASS: [anonymous] ]
[ CMD: [LIST] FUZZ: [..?..?..?..?..?..?..]      SIZE: 4700 ]
[ Connecting to 118.229.231.224:21... ]
[ ERROR: Cannot connect to target!!!          ]
[ SERVER IS MAYBE DEAD BECAUSE OF FUZZING!!! ]
```

四、FTP漏洞挖掘—FTPFuzz

我们回头看一下虚拟机里面FTP主机的情况，确实是崩溃了，说明了我们的Fuzz的确生效了。



四、FTP漏洞挖掘—FTPFuzz

正是FTPFuzz造成了Quick’n Easy FTP Server程序发生了崩溃。这个程序帮助我们快速挖掘出了被测试目标FTP服务程序的安全隐患，大大提高了工作效率。

在程序崩溃时我们也可以用Ollydbg等程序进行调试，来发现更详细的错误信息帮助我们分析程序的漏洞。

四、FTP漏洞挖掘—FTPFuzz

相关知识：

上面案例演示的漏洞就是”Quick’n Easy FTP Server Lite Version 3.1远程拒绝服务漏洞”,FTP协议中的”LIST”命令后的用户数据长度如果超过232，同时最后字符以”?”结尾，就会造成这个程序的崩溃。

四、FTP漏洞挖掘—Fuzz DIY

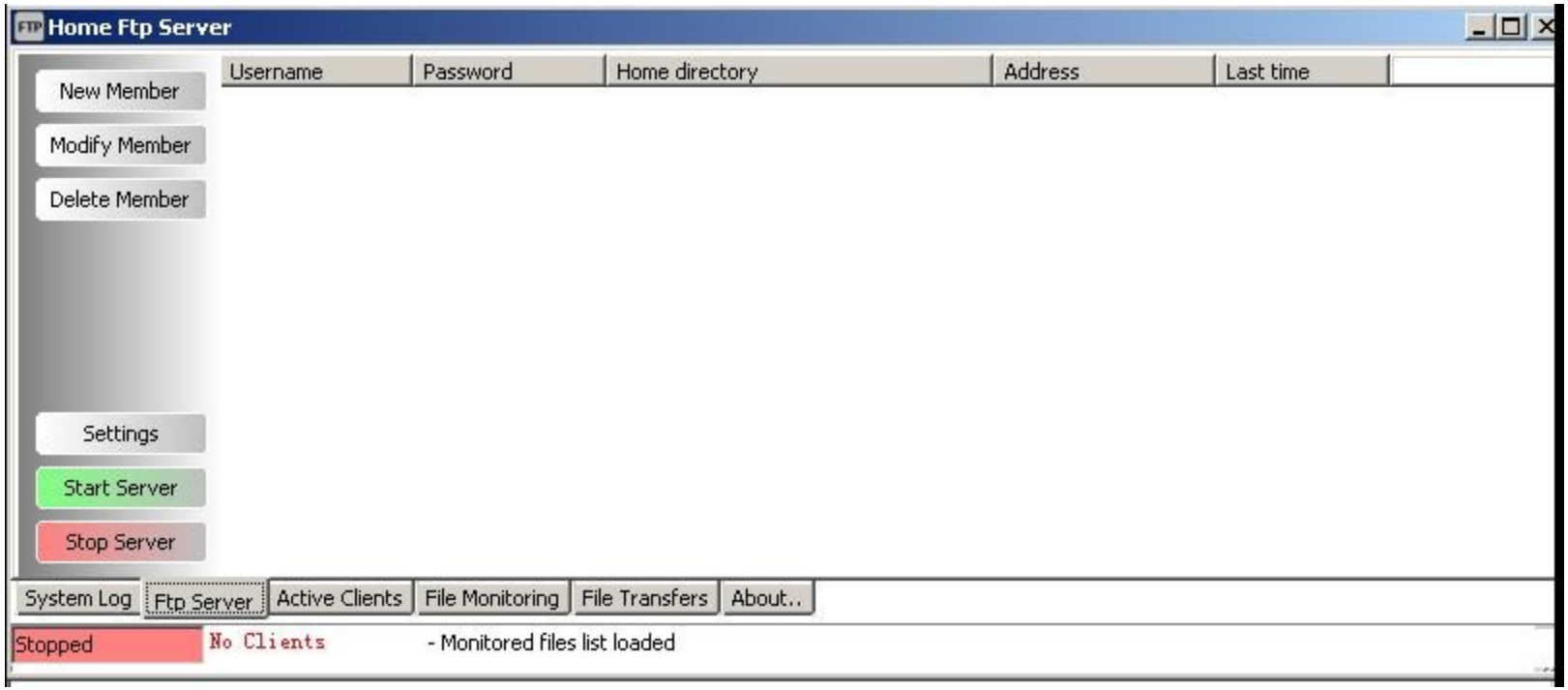
我们知道了FTP的基本工作原理，又看了一些软件的Fuzz效果，我们现在来自己编写一个脚本来实现FTP 的简单的Fuzz。

下面是我们的实验环境。

	推荐使用的环境
操作系统(服务端)	Vm win2000
操作系统(客户端)	Vm win2000
Home Ftp Server	1.10.1
Fuzz.py	我们自己写的Python脚本

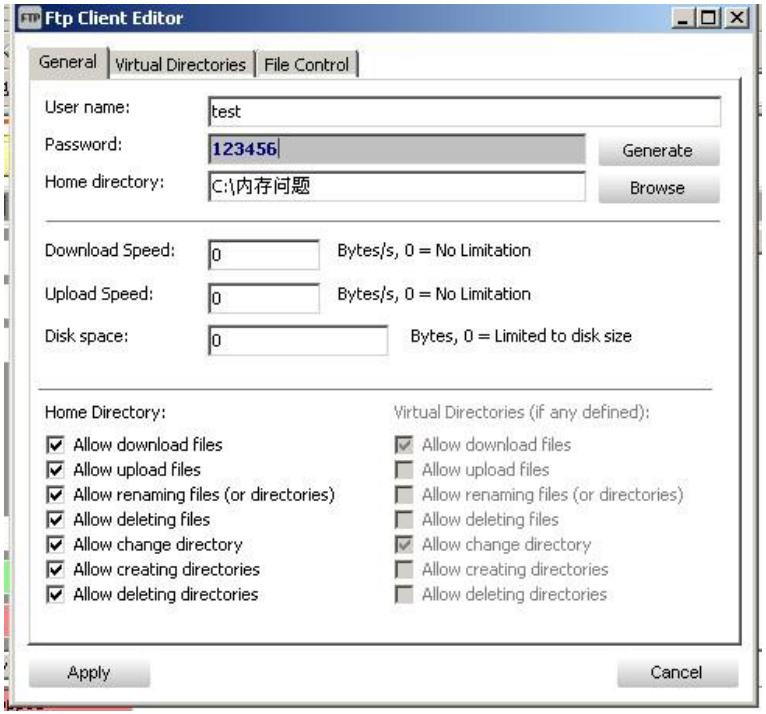
四、FTP漏洞挖掘—Fuzz DIY

我们这次要测试的对象是名为”Home FTP Server”的FTP程序。在服务端的操作系统中安装好该程序后，运行并点击FTP Server选项卡后得到如下画面。



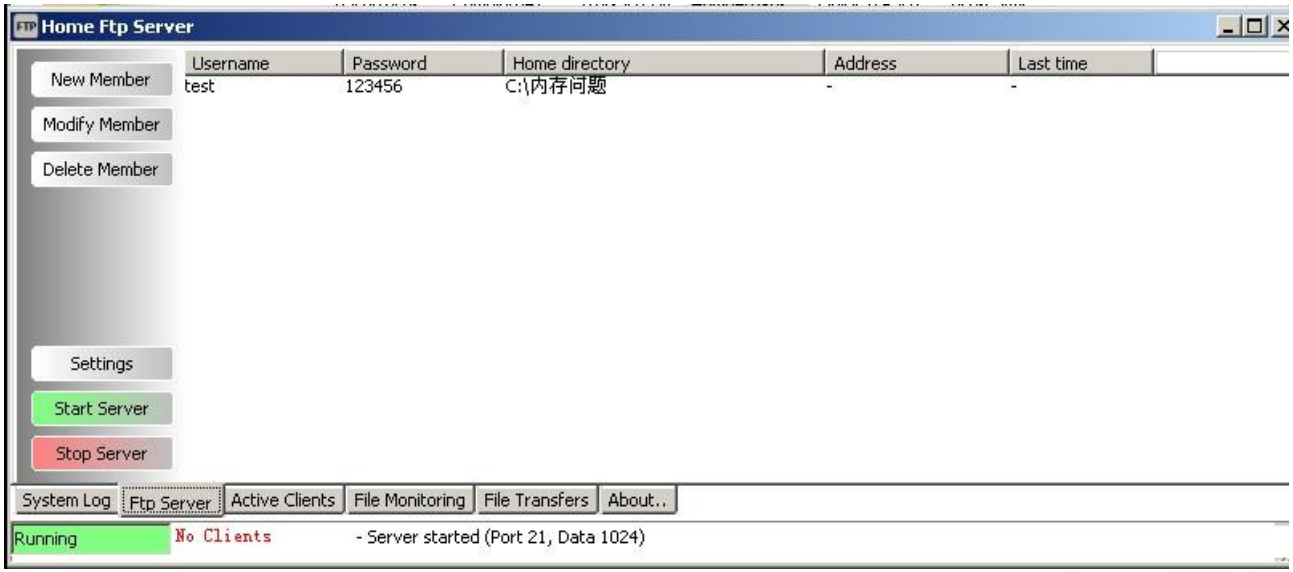
四、FTP漏洞挖掘—Fuzz DIY

我们这点击New Member后添加用户名为Test,密码为123456的用户，并且设置好访问文件目录，新建用户就成功了。



四、FTP漏洞挖掘—Fuzz DIY

然后我们点击Start Server就可以运行FTP了，看到左下角的Running表示正常工作。



四、FTP漏洞挖掘—Fuzz DIY

接下来我们要使用Python来编写一个Fuzz脚本,开始我们先引入模块，定义函数，设定一些变量:

ip和port是FTP的地址和端口(21)，fuzzcmd是FTP的命令集合

```
import socket,sys
def ftp_test(ip,port1):
    target = ip
    port = port1
    buf = 'a'*272
    j=1
    fuzzcmd = ['mdelete ','cd ','mkdir ','delete ','cwd ','mdir ','mput ','mls ','rename ','site index ']
```

四、FTP漏洞挖掘—Fuzz DIY

```
try:
    connct = s.connect((target,port))//连接主机
    print "[+] Connected!"
except:
    print "[!] Connection Failed!"
    sys.exit(0)
s.recv(1024)
s.send('USER test\r\n')//登录用户名为test,密码为123456的用户
s.recv(1024)
s.send('PASS 123456\r\n')
s.recv(1024)
print "[+] Sending payload..."
```

四、FTP漏洞挖掘—Fuzz DIY

```
for i in fuzzcmd://对于每一个fuzzcmd中的元素，
        //提取出来的元素即i
    s.send(i + buf*j + '\r\n')
    s.send(i + buf*j*4 + '\r\n')
    s.send(i + buf*j*8 + '\r\n')
    s.send(i + buf*j*40 + '\r\n')
    s.send(i + buf + ‘ ’ + buf +‘\r\n’)
        //发送cmd + 若干个’a’这种脏数据
    try:
      s.recv(1024)//如果接受无异常说明fuzz失败
      print "[!] Fuzz failed!"
    except:
      print “[+] Maybe we find a bug!“
        //出现异常说明可能出现了漏洞
```

四、FTP漏洞挖掘—Fuzz DIY

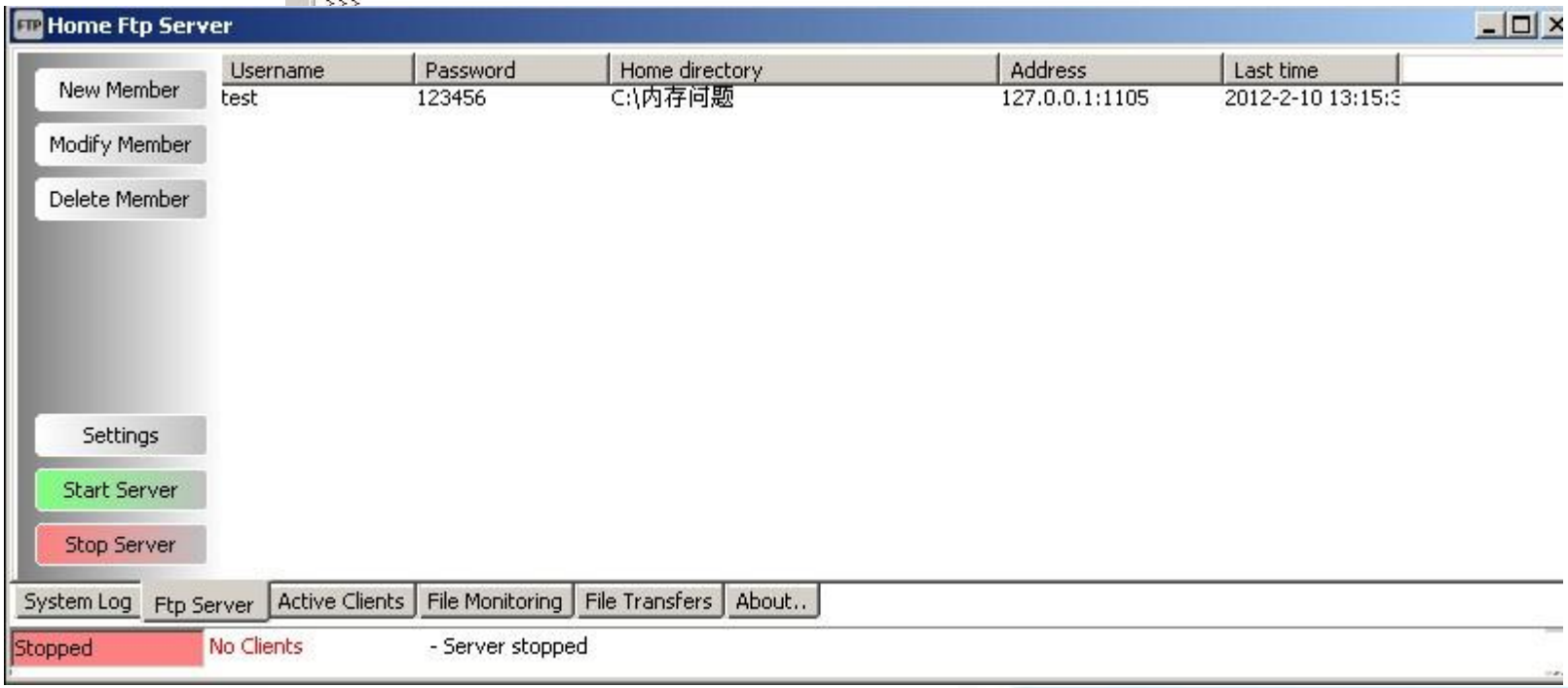
```
if __name__ == '__main__':
//if这句话是Python语言调用的main函数的形式
 ftp_test("127.0.0.1",21)
//调用我们定义的函数并传入IP,PORT
```

程序很简单，下面我们来运行一下，看会得到什么结果。

四、FTP漏洞挖掘—Fuzz DIY

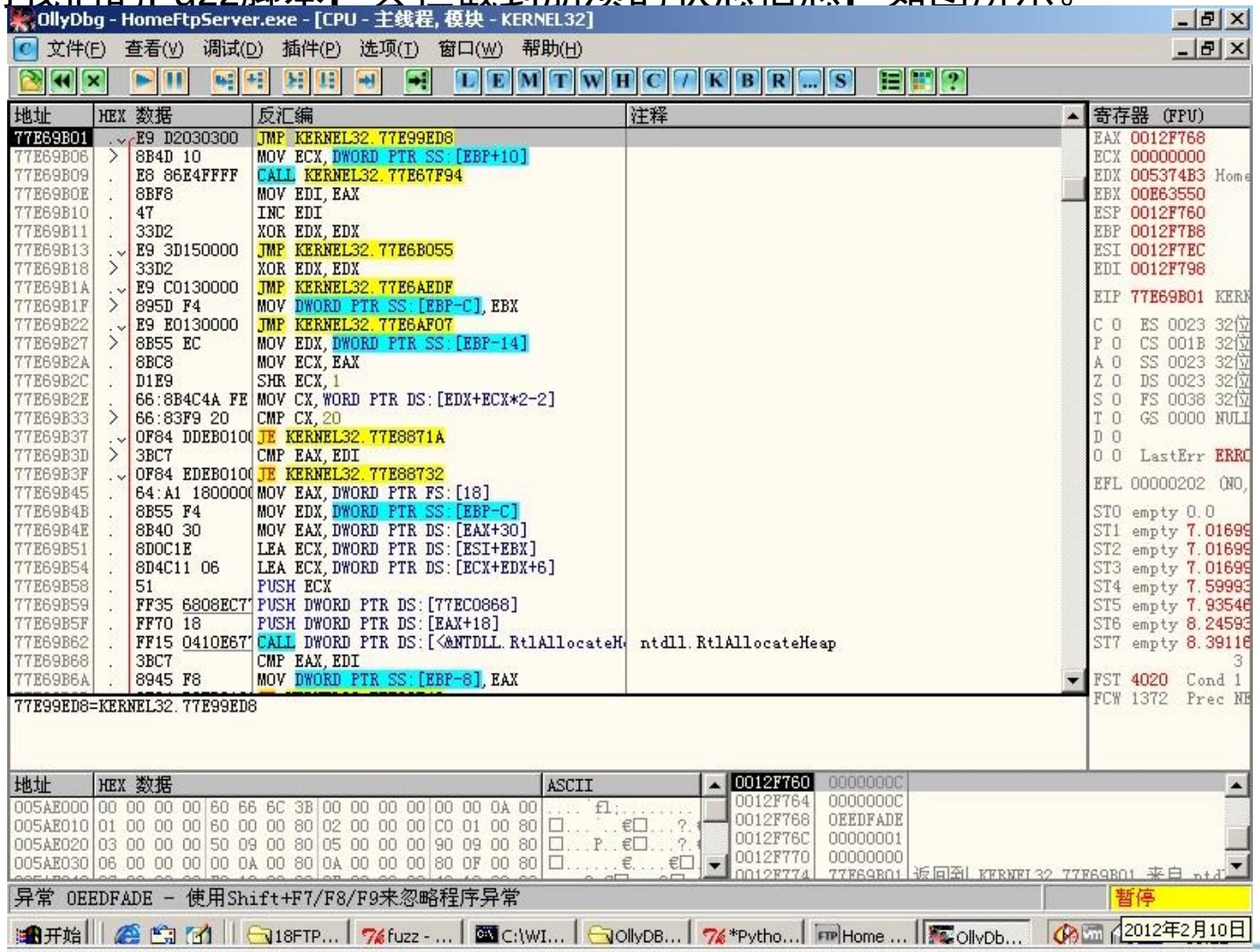
我们发现脚本没显示出来什么信息，但是我们回头观察一下FTP程序，已经停止了运行。

```
>>>
[+] Connected!
[+] Sending payload...
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
[!] Fuzz failed!
>>>
```



四、FTP漏洞挖掘—Fuzz DIY

为了发现更多信息，我们使用OllyDbg附加在Home Ftp Server进程上，再使用我们的Fuzz脚本，会拦截到崩溃的状态信息，如图所示。



四、FTP漏洞挖掘—Fuzz DIY

构造出了异常的场景只是个刚刚的开始，具体的漏洞的挖掘和利用还需要我们一步一步慢慢的分析，仔细的调试，是需要深厚的功力的

而这节课所讲的Fuzz只是漏洞挖掘中的一个简单的部分，希望大家可以了解这个技术的思想和使用环境，让它来帮助你实现漏洞的挖掘

谢谢观赏！
Thanks！