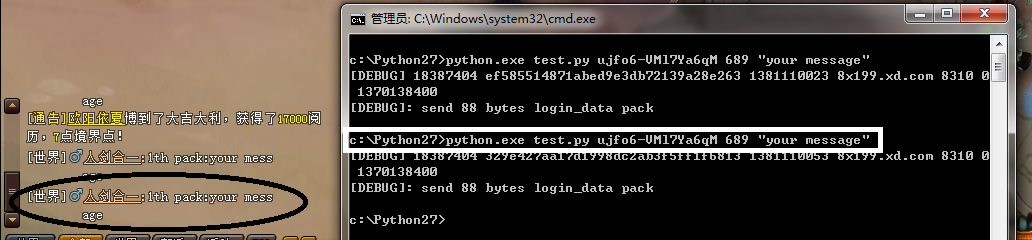
本人码农一枚，玩神仙道有小半年了，挺喜欢的。10.1期间没啥事情，研究了下神仙道游戏的通信协议，小有收获，和大家分享下，话不多说，先上DEMO， 这个DEMO提供了向世界窗口发消息的功能：  
        大家将附件中的文件解压出来， 然后在[python环境](http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi)（点击下载）中运行：  
       python.exe <你的user> <服务器编号> <想发送的信息>   
       这里提供一个小号供大家玩下(e.g. **python.exe test.py ujfo6-VMl7Ya6qM 689 "your message"**) ，

如下图所示，就可以在世界聊天窗口看到你发的消息了。





原理分析-PART0：

大家应该记得，当用户输入好密码后，下次进入的时候是不用输入密码的，所以用户的相关信息就存在cookie里面，打开之后我们发现如下6个COOKIE，不难猜测，信息应该存在sxd\_user或者user里面。



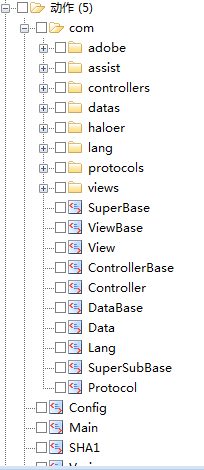
查看游戏页面的源码，仔细看会发现如下一段，没错这就是FLASH启动时的变量： 

后续的FLASH是怎么通信的呢，登陆后，我做相应的动作，发现没有相应的HTTP或者HTTPS请求，所以推断后续的FLASH应该是通过TCP通信的。而涉及到的变量就是上面网页的所有变量。

原理分析-PART1-登陆：

既然要分析FLASH的TCP通信，就需要两个工具，FLASH的反编译工具和网络抓包工具，这里我选择了swfdecompiler, wireshark。现学了一把，然后尝试着用了用。

通信协议：



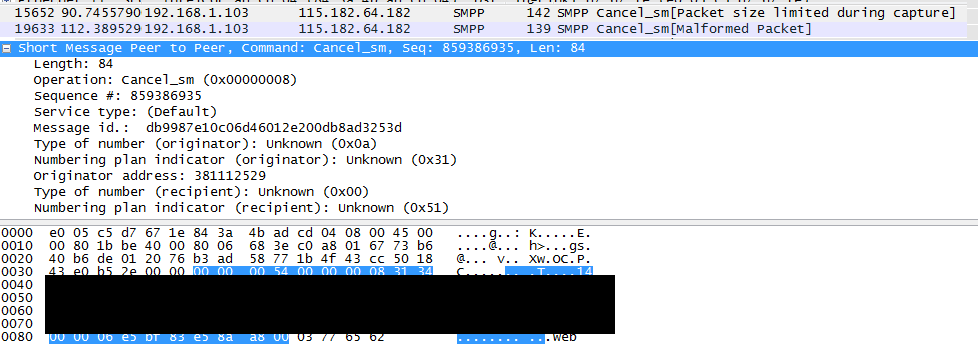
通过分析HTTP请求，发现首先是启动index.swf，然后是main.swf，index.swf基本没做什么事情，大概是读了一个协议文件。然后就是main.swf了；所幸FLASH的ACTION SCRIPT反解后是能够读的，过分析发现基本的格式如下：

**<len:int> <module:byte> <action:byte> <data>**

而所有的协议说明都定义在：com.protocols里面。

可疑的SMPP

打开登陆后的所有TCP请求包，并没有什么特别的发现，唯一一个比较可疑的SMPP，不知道是干嘛用的，网上查了一把，发现是个不知道的协议，应该没有关系，所以一度忽视的这个包。



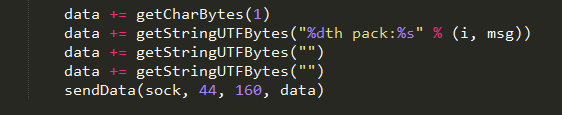
最后的最后发现，这个就是登陆请求包。立马尝试用相同的请求包发请求，发现服务器不回应我，仔细分析了下涉及如下几个字段：**userid, hashcode, time**, src\_url, regdate, id\_card, open\_time, is\_newst, stage\_name, client。每次都在变化的就只有**userid, hashcode, time**,会不会是服务器有验证机制呢？如果有验证机制那我就只能每次动态生成，对可以抓取PART-1里面的动态页面s689.sxd.xd.com，而生成那个页面是需要带COOKIE的，没错，涉及到**user**那个COOKIE，经过尝试是可行的。可以推测hashcode就是userid这个时间通信的密码。

原理分析-PART1-世界聊天：

有了前面的登陆，就可以做很多事情了，这里仅举聊天为例，有一点需要说明下，你登陆后会收到很多很多的包，开始，会觉得手足无措，后来对照协议看了，多数是更新练功经验，通知可以领取XX奖励，类似的包，所以基本可以全忽略的。

世界聊天的请求格式是

Module:44 action:160 data:[unknow:byte,msg:string, unknown:string, unknown:string]



这里还存在一个问题，前面的那一个BYTE是干嘛的，我尝试使用和其他用户相同的BYTE发送，结果不WORK，于是就直接写了个循环，发现请求发出去了。这里留个疑问吧。

写在最后：

这里只能叫做大致的分析或者是基本原理，最后分析出来原理还是蛮开心的，虽然期间有很多纠结点，比如wireshark的那个误点性的smpp。相信到这一步，后续的分析就比较容易了，有兴趣的同学可以继续，有欢迎交流。