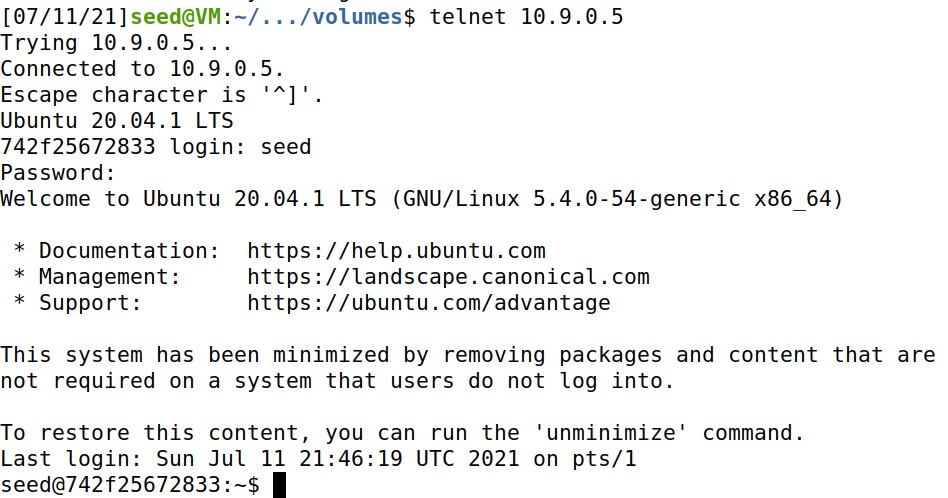
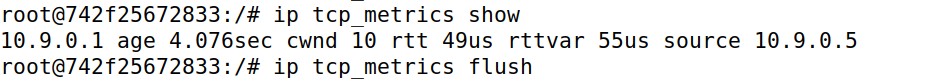
# Task1

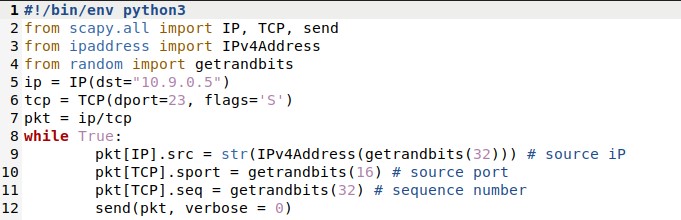
## 1.1

首先是没有运行SYN Flood攻击时，我们从10.9.0.1向10.9.0.5进行telnet，可以成功连接。



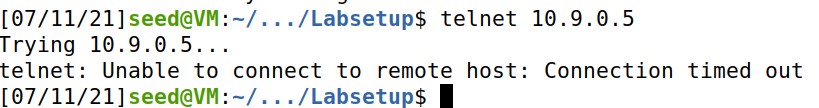
然后去10.9.0.5上面清除一下连接缓存，使用如下命令。

 根据提供的代码进行填空，如下，命名为synflood.py

 在10.9.0.1上面运行上述代码，并等待一段时间。

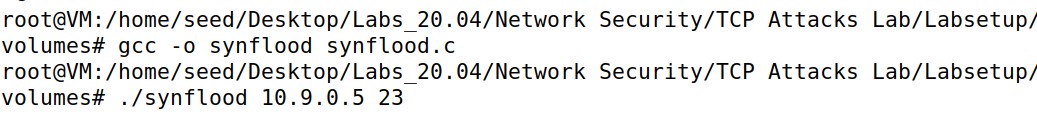


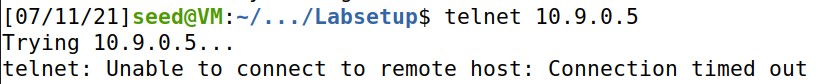
这时再次从10.9.0.1上对10.9.0.5进行telnet，发现连接超时，攻击成功。

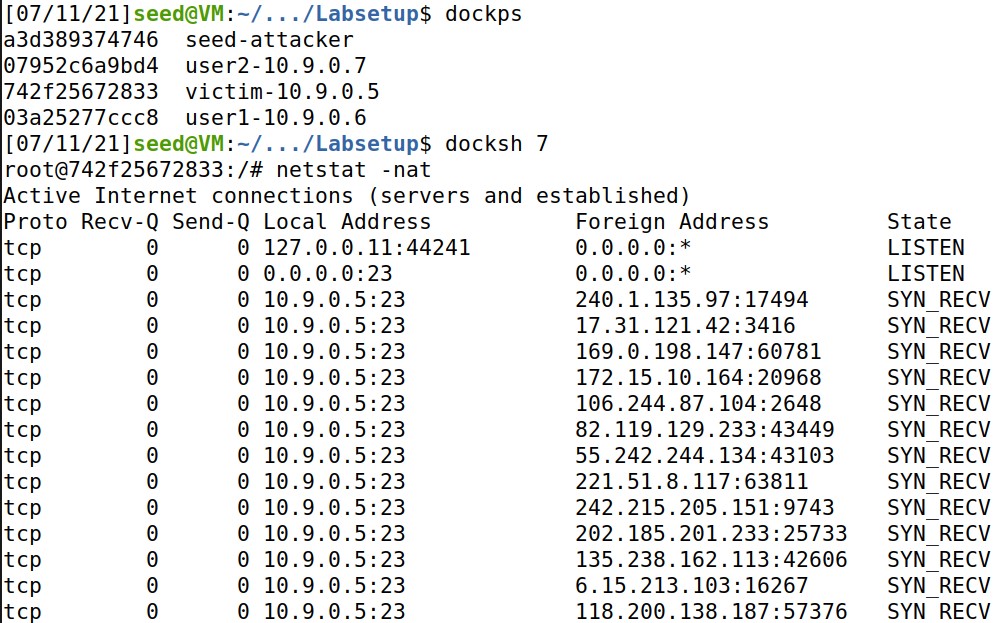


## 1.2

对所给的代码进行编译运行，如下

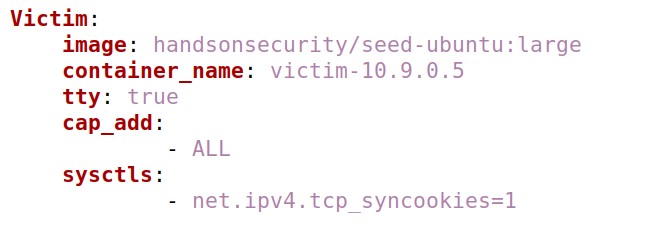
 一段时间后同样从10.9.0.1向10.9.0.5进行telnet，发现连接也超时了，攻击成功。

 这时在10.9.0.5上查看一下网络状态，发现已经被大量的SYN请求占满，这是上面两次攻击成功的原因。



## 1.3

我们将SYN cookie机制打开



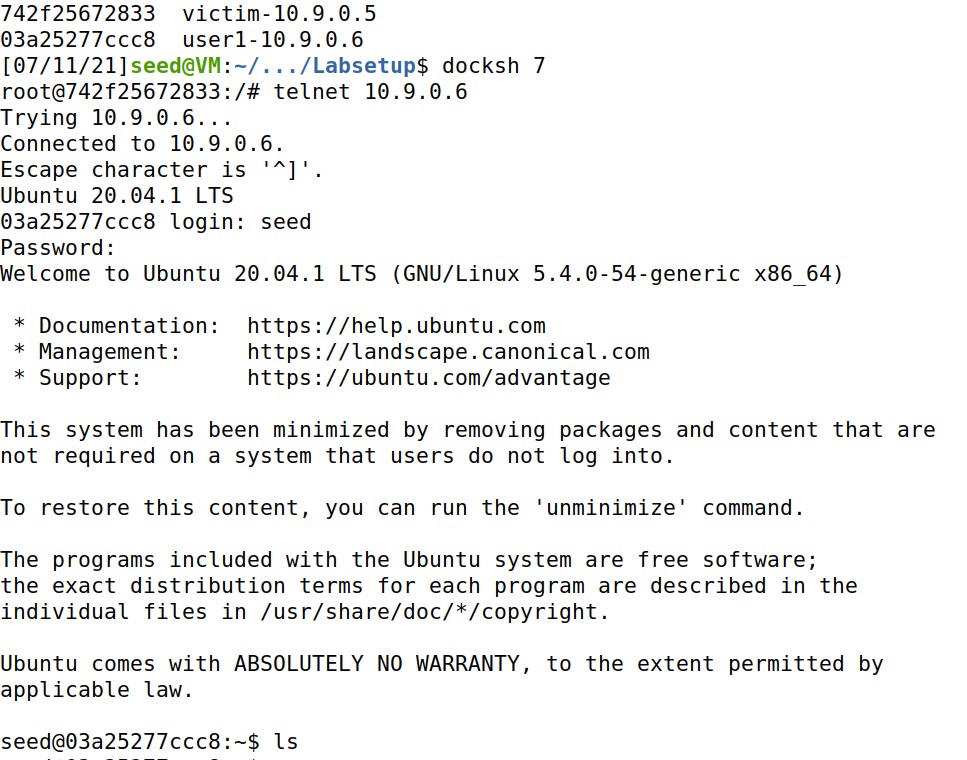
此时上述两种攻击都失败。

# Task2

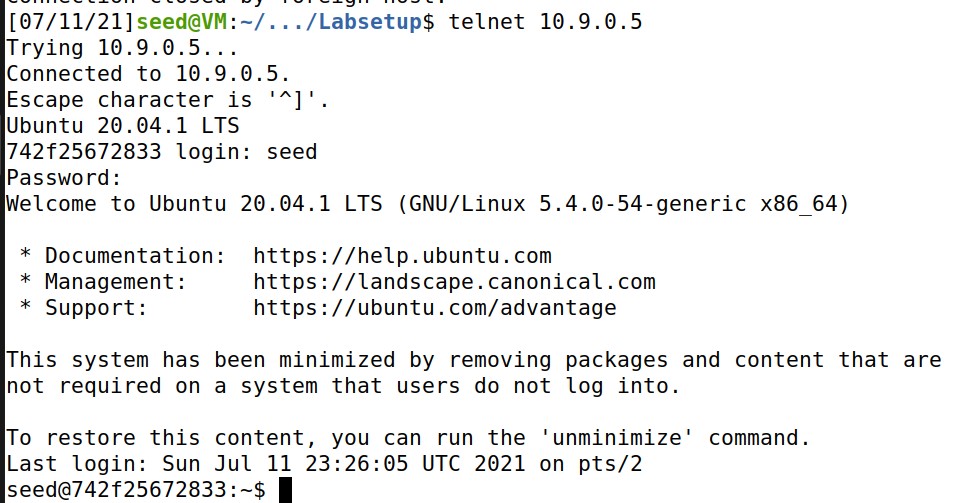
首先是RST.py的代码如下。

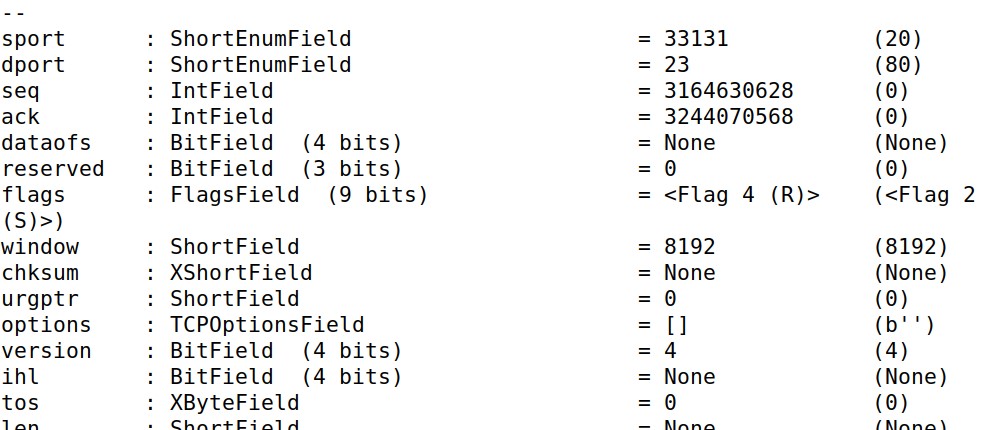


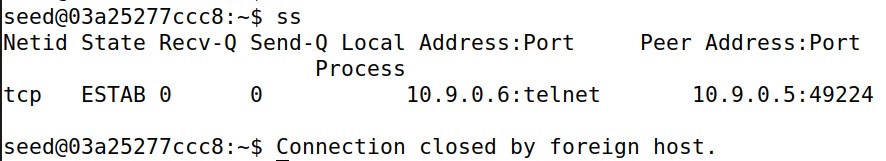
然后去10.9.0.5上面telnet 10.9.0.6，如下图。



接着返回10.9.0.1，telnet 10.9.0.5，如下。

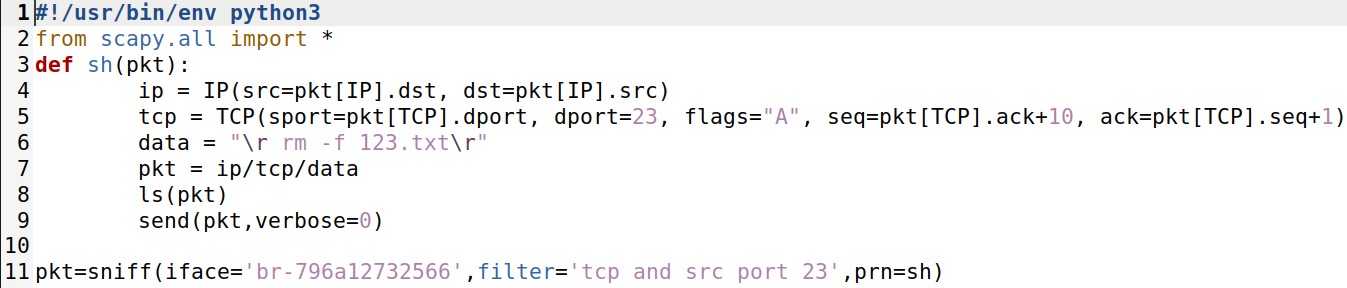
 接下来再在 10.9.0.1 上开一个终端运行 RST.py，得到的响应如下。

 此时返回10.9.0.5，随意命令使代码进行到攻击位置，发现连接自动断开，如下。

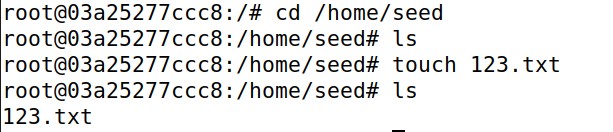
 由此可得攻击成功。

# Task3

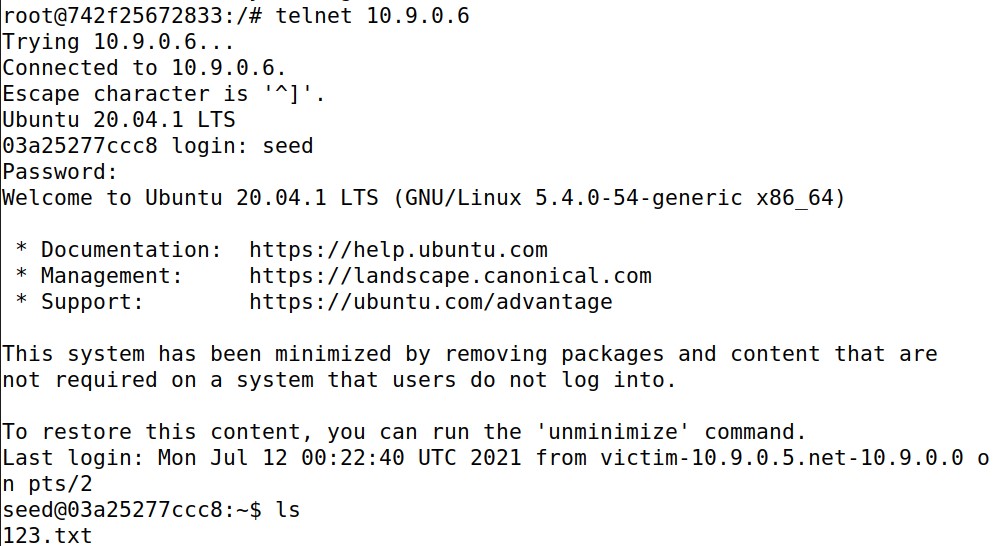
SH.py的代码如下，其中data选择删除一个文件的命令。

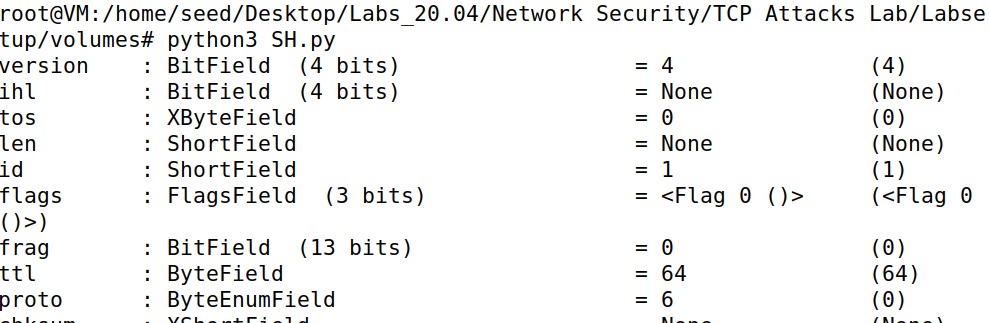


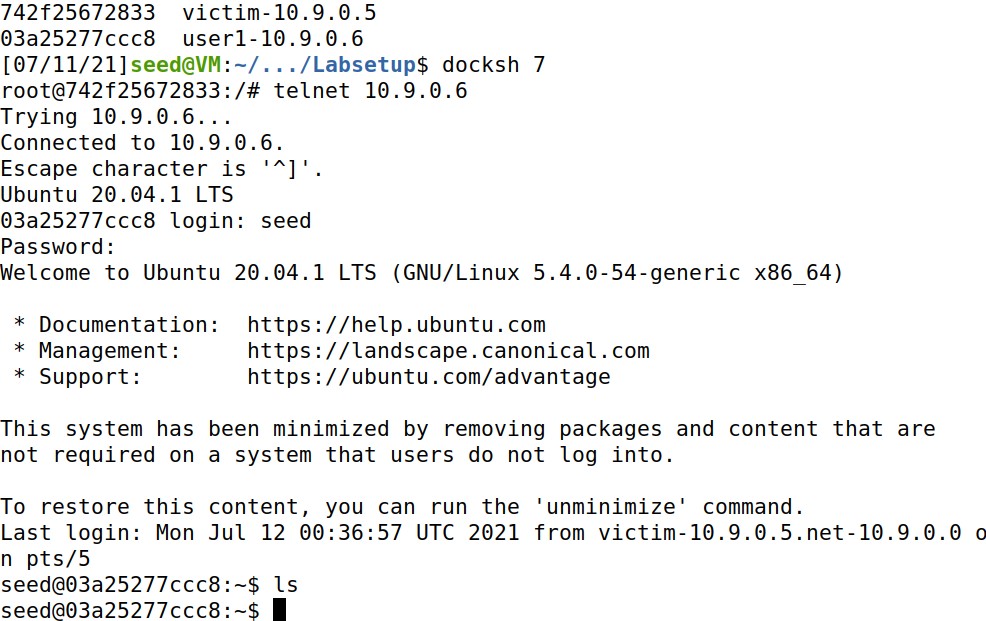
首先到10.9.0.6里的/home/seed文件夹下新建一个123.txt文件。如下。



然后从10.9.0.5下telnet 10.9.0.6，发现我们新建的文件此时还存在。如下。



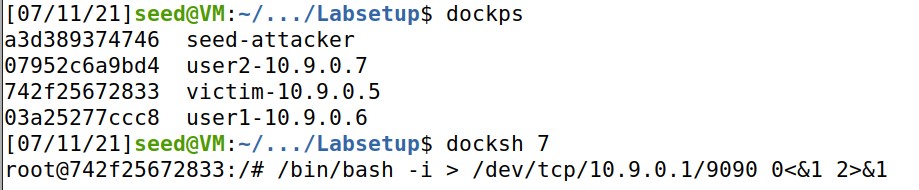
然后回到10.9.0.1下，运行SH.py，响应如下。  然后回到10.9.0.5，随意输入几个命令至攻击代码处，发现已经不能继续输入了，这时关掉这个终端，重新打开再telnet 10.9.0.6，发现123.txt已经消失了，如下。

 由此可得攻击成功。

# Task4

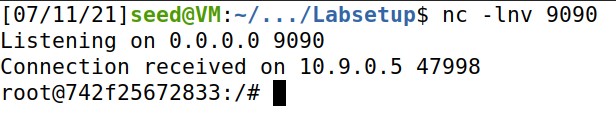
首先在10.9.0.1里面监听9090端口，如下。

 然后到10.9.0.5下面输入所给命令，如下。

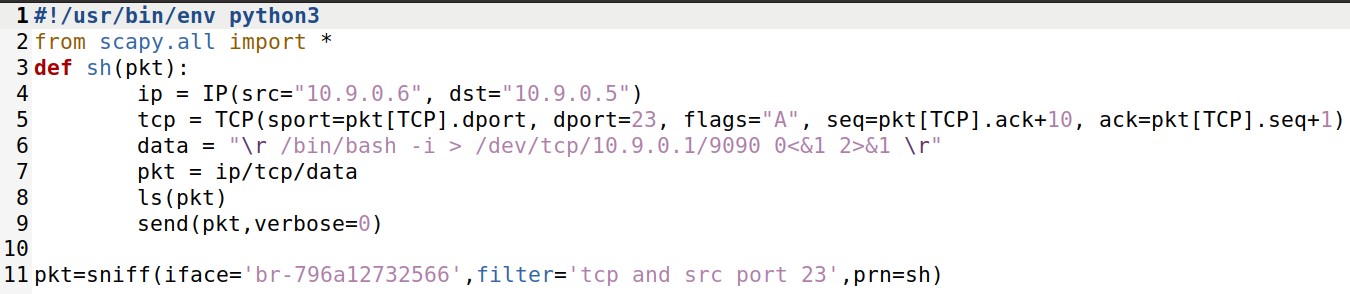


回到10.9.0.1，发现已经得到10.9.0.5的Reverse Shell，可以对

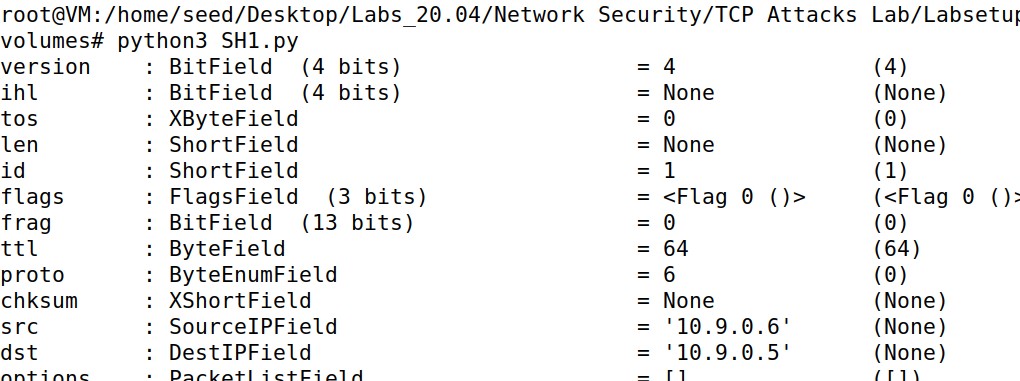
其执行操作，如下。



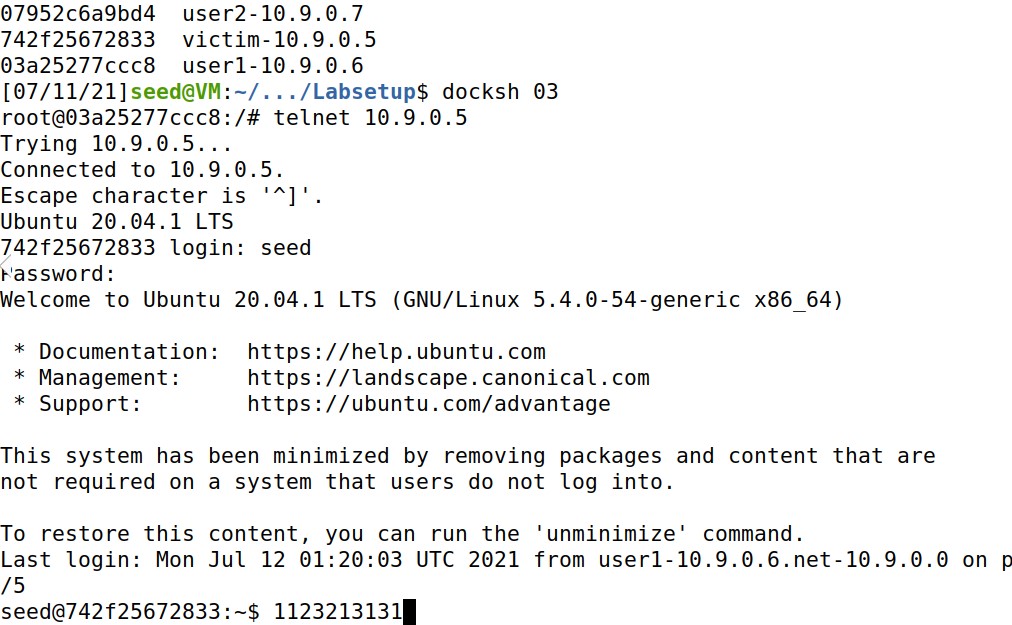
将上述命令放入SH1.py中，代码如下。

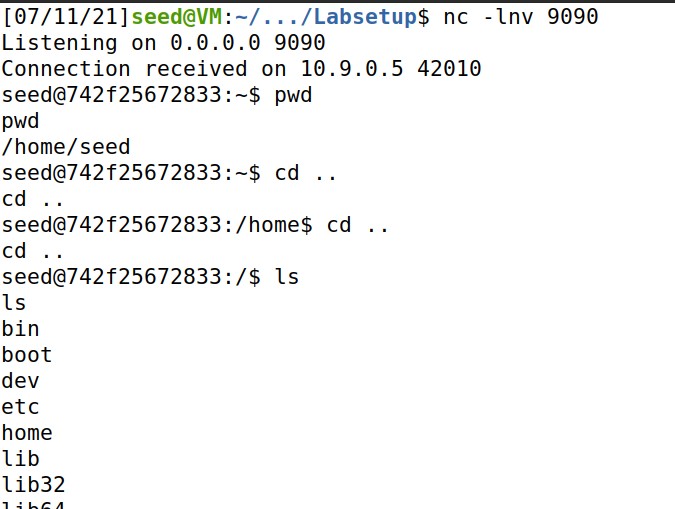


然后去10.9.0.6里面telnet 10.9.0.5。继续监听9090端口，同时在10.9.0.1运行SH1.py，如下。



再在telnet里面随意输入几个命令至攻击代码处，发现输入不了了。如下。

 这时，返回10.9.0.1，发现已经得到了10.9.0.5的Reverse Shell，可以进行各种操作。

 由此可得攻击成功。