Task2

配置容器，并启动web服务。

通过nc进行栈溢出，覆盖原代码，再执行恶意代码。

报纸上的文字

描述已自动生成

图形用户界面, 文本, 应用程序, Word

描述已自动生成

文本, 信件

描述已自动生成

执行删除功能

文本

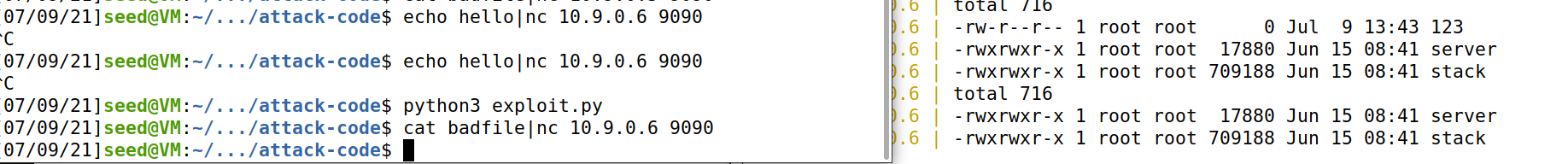
描述已自动生成

Task3

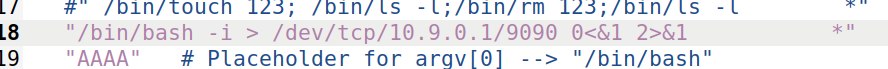
因为只知道buffer的起点，知道buffer的大小范围，只要一个个尝试，就可以最终执行badfile。

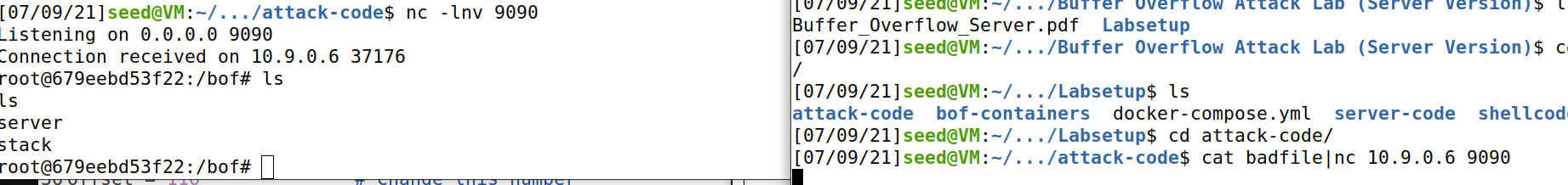
文本

描述已自动生成



然后要拿到root，所以要修改一下





Task4

因为改为了64位的Linux，首先shellcode要改为64位的，然后为了避免遇到0结束，我们将shellcode放到buffer中，并将ret放到buffer的起点。

文本, 信件

描述已自动生成

文本

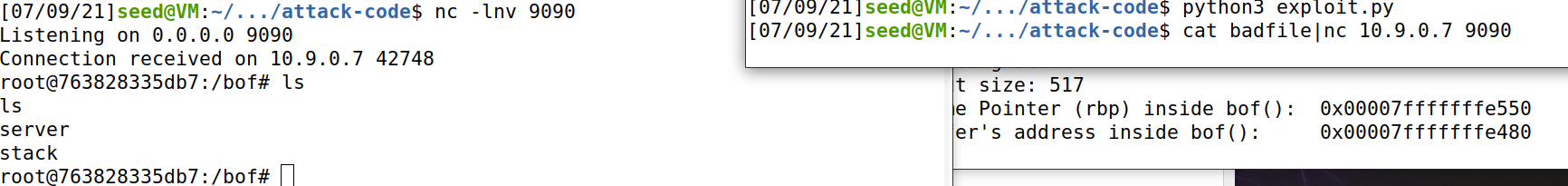
描述已自动生成

文本

描述已自动生成

然后是拿root





总结：

这次实验比较算是我的知识盲区，虽然比较片面的了解了栈溢出的大概机制，但在实际操作时，还是比较困难；尤其是有时候因为漏了如&、<、/等小小的字符，也会拿不到shell，需要对命令有一定的理解，不能粗心。