《漏洞利用及渗透测试基础》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

反序列化漏洞

**实验要求：**

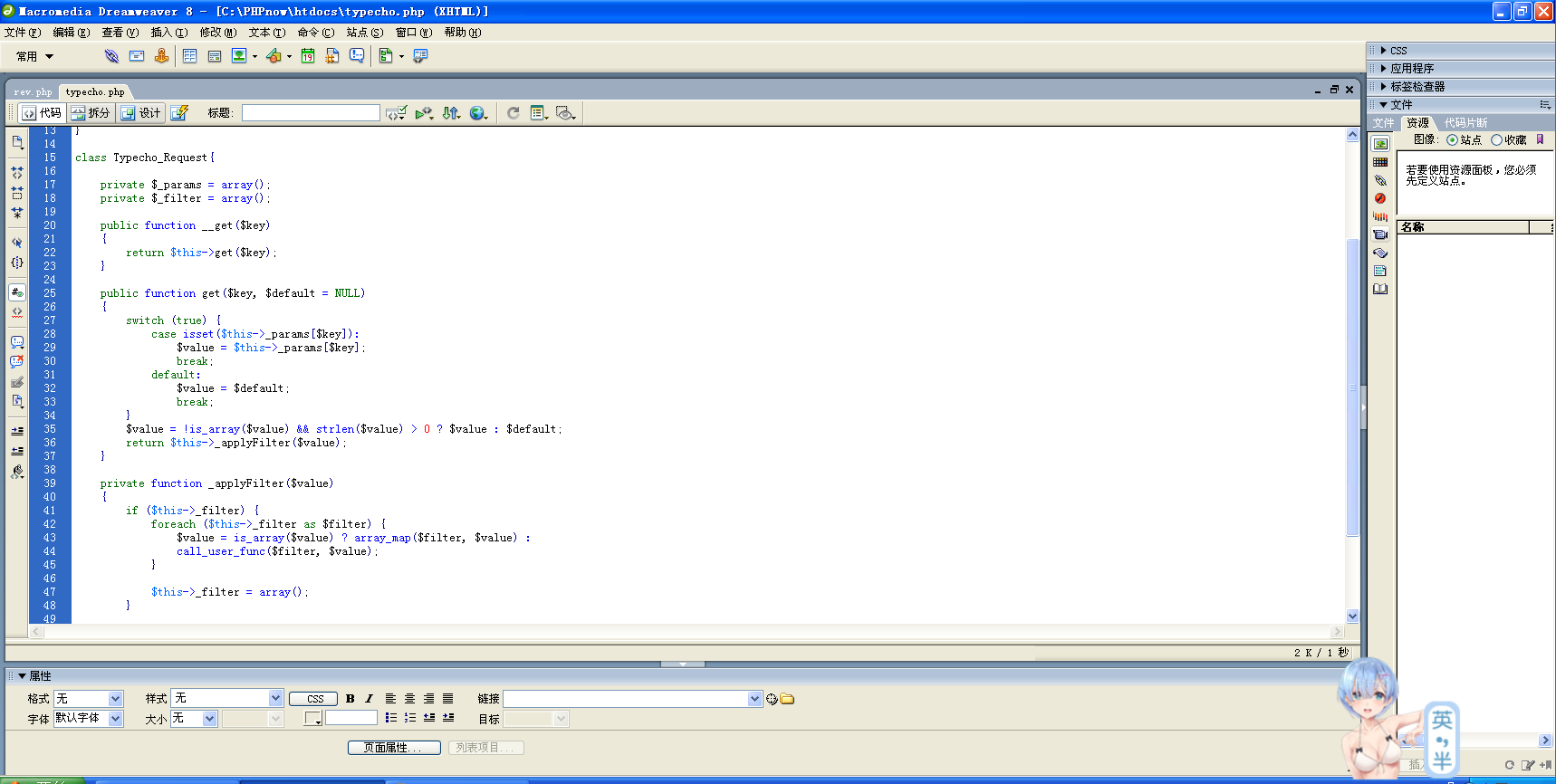
复现12.2.3中的反序列化漏洞，并执行其他的系统命令。

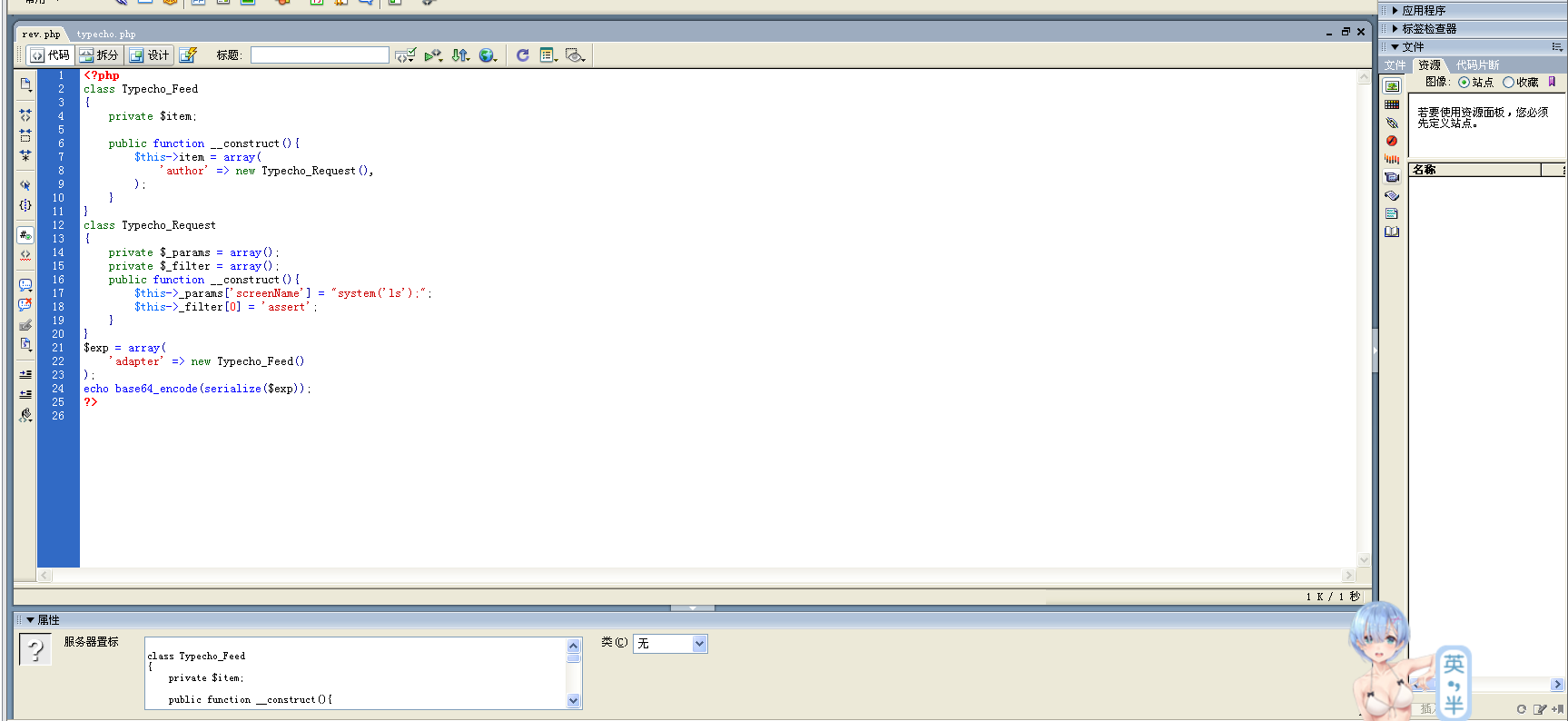
**实验过程：**

PHP反序列化漏洞又叫PHP对象注入漏洞。在一个应用中，如果传给unserialize()的参数是用户可控的，那么攻击者就可以通过传入一个精心构造的序列化字符串，利用PHP魔术方法来控制对象内部的变量甚至是函数。对这一类漏洞的利用，往往需要分析web应用的源代码。

1. 实验准备

复制书中的代码，将其放到phpnow下





1. 实验思路

反序列化漏洞是针对web中$\_GET[‘\_\_typecho\_config’]函数，这个函数能够从从用户处获取了反序列化的对象，满足反序列化漏洞的基本条件。同时unserialize()的参数是可以人为控制的，属于漏洞的入口点.

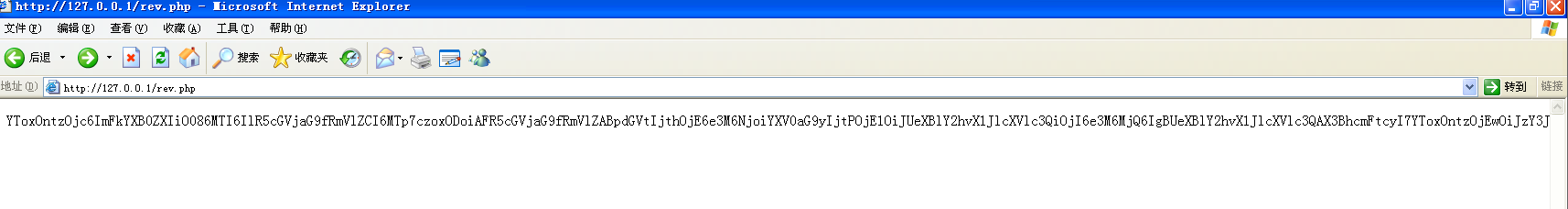
接下来，程序实例化了类Typecho\_Db，类的参数是通过反序列化得到的$config。在类Typecho\_Db的构造函数中，进行了字符串拼接的操作，会调用\_\_toString()方法。从代码中可以看到，类Typecho\_Feed中存在\_\_toString()方法。

在类Typecho\_Feed的\_\_toString()方法中，会访问类中私有变量$item[‘author’]中的screenName。如果$item[‘author’]是一个对象，并且该对象没有screenName属性，那么这个对象中的\_\_get()，方法将会被调用，在Typecho\_Request类中，正好定义了\_\_get()方法。

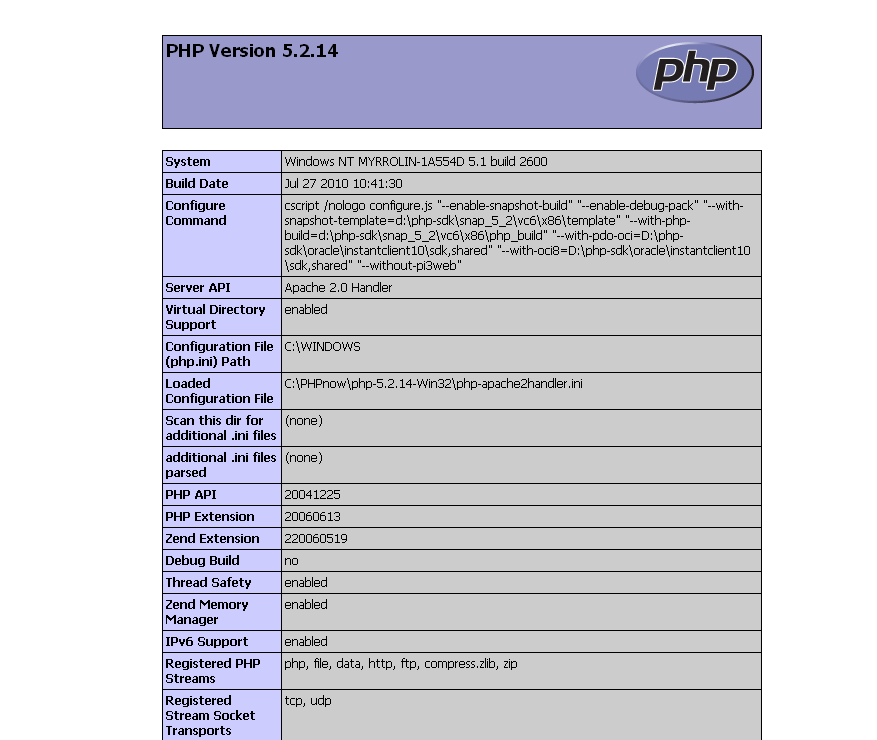
类Typecho\_Request中的\_\_get()方法会返回get()，get()中调用了\_applyFilter()方法，而在\_applyFilter()中，使用了PHP的call\_user\_function()函数，其第一个参数是被调用的函数，第二个参数是被调用的函数的参数，在这里$filter，$value都是我们可以控制的，因此可以用来执行任意系统命令。至此，一条完整的利用链构造成功。

1. 实验复现

访问之前已经建立好的rev.php，可以得到一个payload

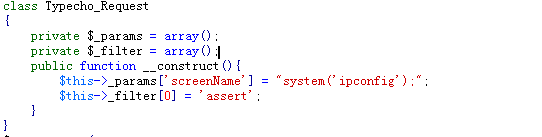


之后通过get方式请求，将其传递给typecho.php，可以看到phpinfo()正确执行

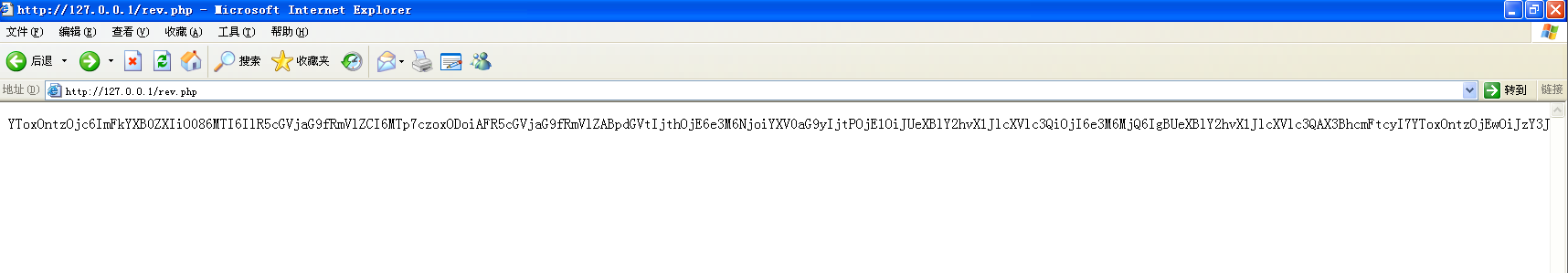


1. 执行其他系统命令

修改rev.php中的代码，将phpinfo()改成system(‘ipconfig’)

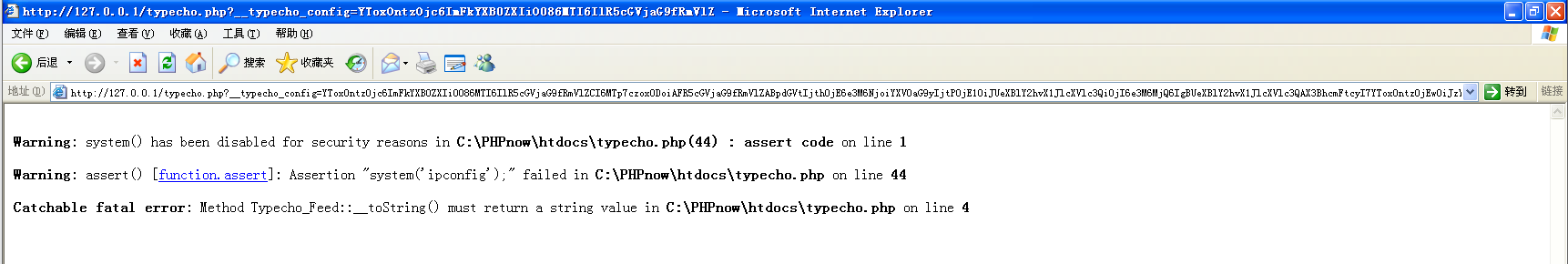


再重新访问rev.php，获取新的payload



再利用get将请求传给typecho.php

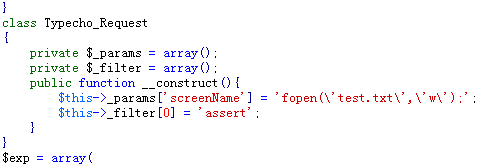
得到：



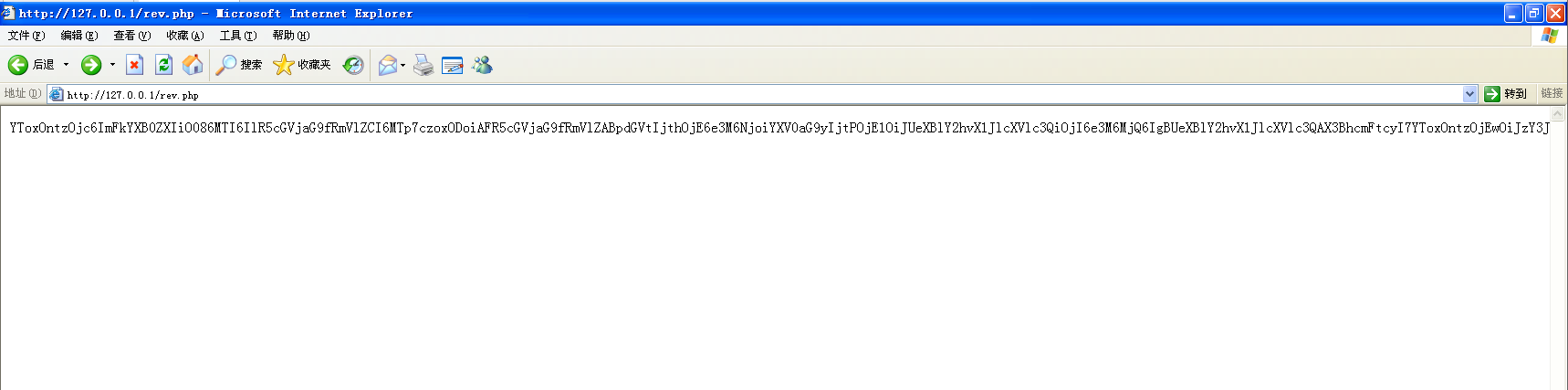
尝试失败

猜测失败原因：可能是因为安全性的问题导致不能够运行这种Terminal的指令。

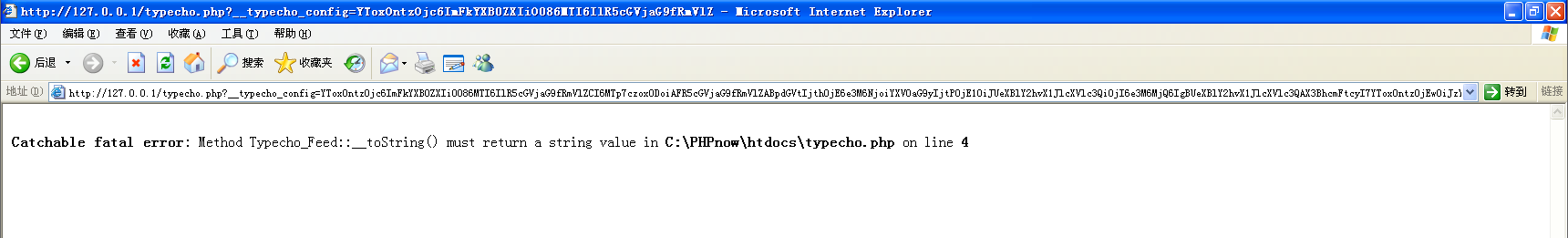
第二次尝试使用fopen(\‘test.txt\’,\’w\’);



得到payload

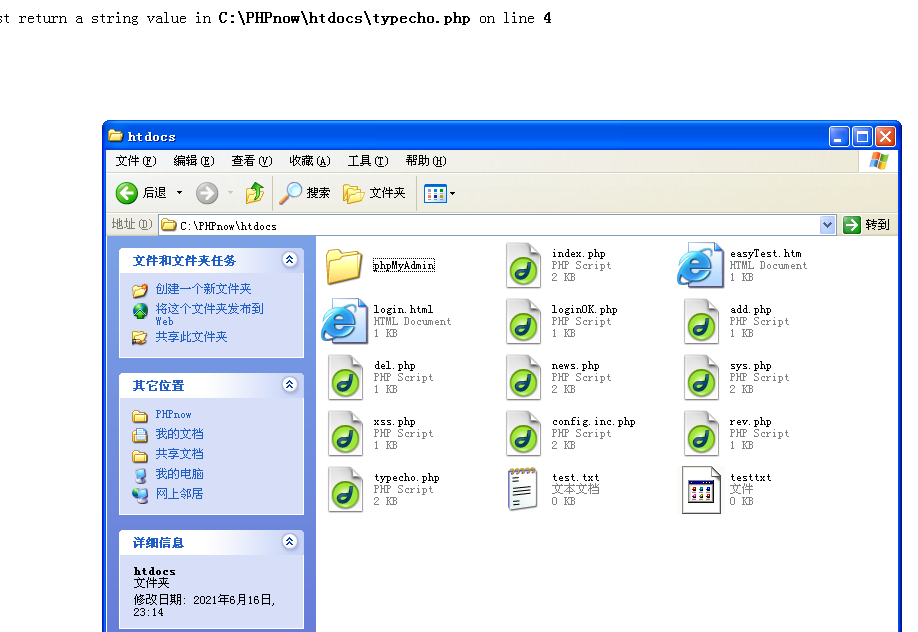


将其用get方式传递给typecho.php，得到如下反馈



应该是说\_\_toString()这个函数需要一个返回值，在这里我们要实现的功能是让他在当前目录下打开一个txt文件，如果没有就会进行创建，不需要返回值，这里没有什么影响

打开php所在的目录，可以看到：



成功创建了test.txt

实验成功

**心得体会：**

反序列化漏洞可以说是目前学的这些里面最难的一个，里面的逻辑关系比较复杂，需要将其理清楚，同时需要精心构造一个能够实现相应功能的字符串，并且可以利用get请求的特点将我们已经构造好了的payload作为参数传递过去，从而达到我们的目标。通过本次的实验，我对反序列化漏洞有了一定的了解，知道了他的一些基本运行过程，了解了基本的利用方式，但是目前还是只会根据书上的实验讲解进行复现，还不会自己去编写构造payload的代码，不会去在没有任何提示的情况下利用这个漏洞。