《漏洞利用及渗透测试基础》实验报告

姓名： 许健 学号：2013018 班级：信安班

**实验名称：**

复现反序列化漏洞

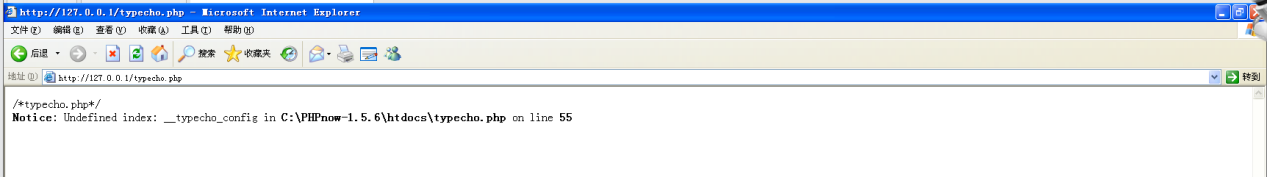
**实验要求：**

复现12.2.3中的反序列化漏洞，并执行其他系统命令。

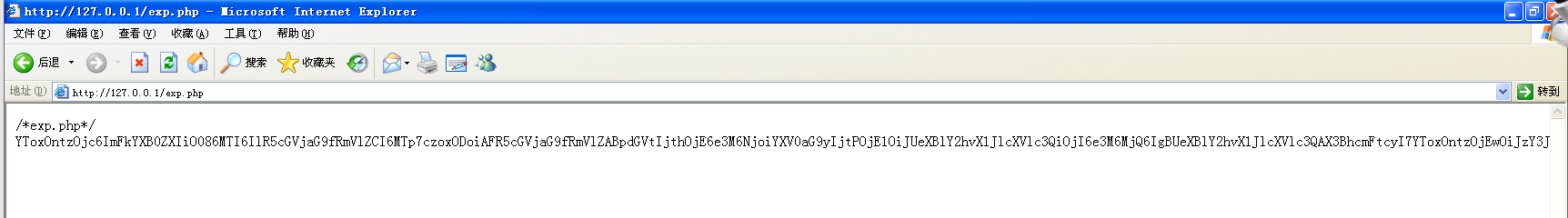
**实验过程：**

1. **复现反序列化漏洞**

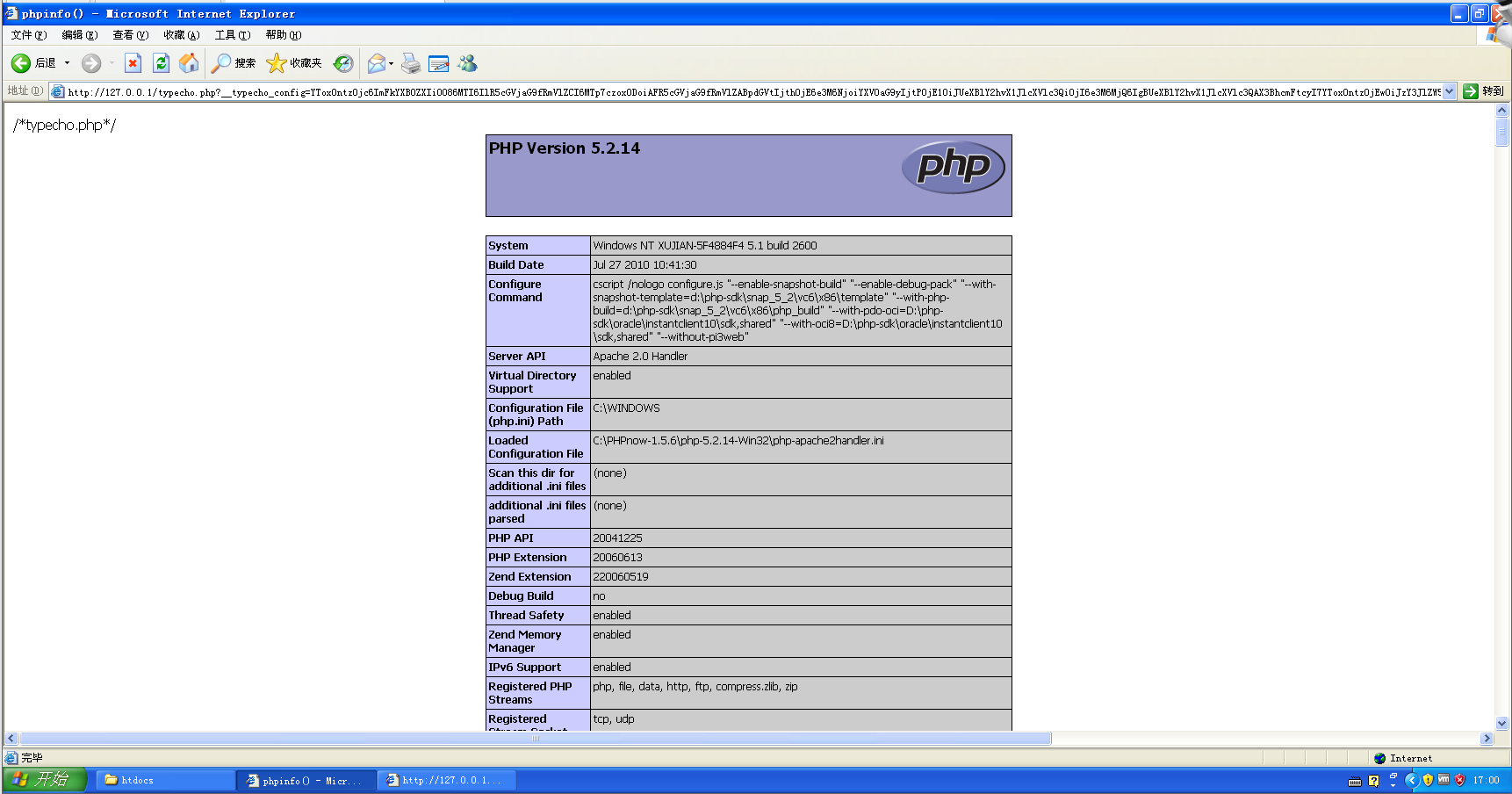
typecho.php页面初始化



exp.php得到反序列化字符串



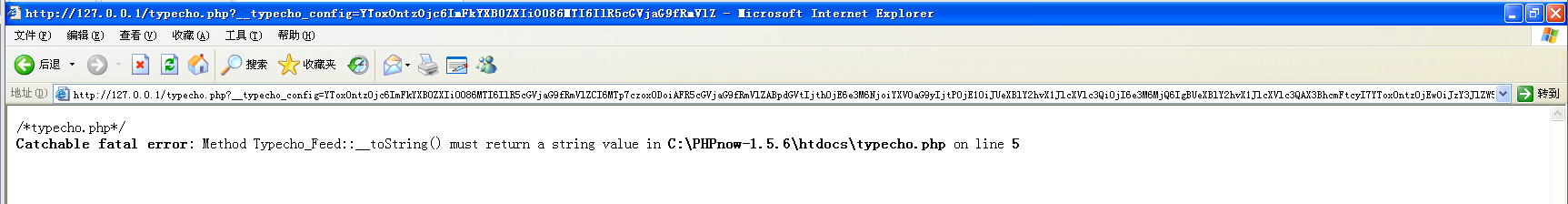
使用get的方式传入参数，成功运行phpinfo()



1. **执行其他系统命令**

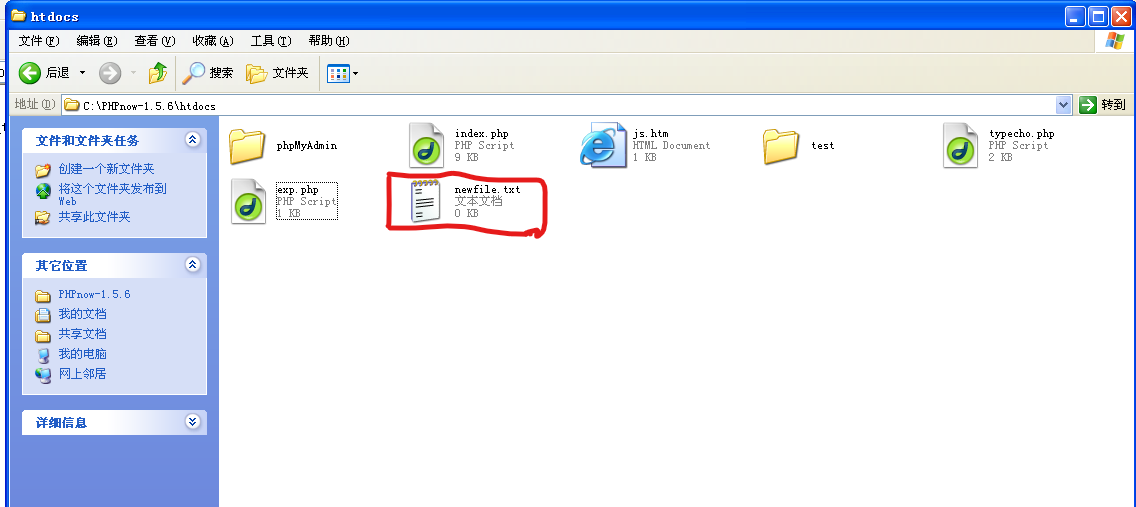
将’phpinfo()’更换为’fopen(\'newfile.txt\', \'w\');’

传入新的反序列化字符串



报错，提示方法要返回一个string类型值

但是系统命令已经被成功执行，查看php文件目录，生成了newfile.txt。



1. **反序列化原理解析**
2. $\_GET[‘\_\_typecho\_config’]获取反序列化的对象。
3. 类Typecho\_Db的构造函数进行了字符串拼接的操作。如果一个类被当做字符串处理，那么类中的\_\_toString()方法将会被调用。全局搜索，发现类Typecho\_Feed中存在\_\_toString()方法。
4. 类Typecho\_Feed的\_\_toString()方法会访问类中私有变量$item[‘author’]中的screenName。如果该对象没有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。
5. 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。
6. **构造反序列化字符串**
7. 初始化一个数组，key属性 'adapter' 对应的value属性是 new Typecho\_Feed()
8. 构造类 Typecho\_Feed()，对于它的私有属性要初始化为数组，key属性'author' 对应的value属性是 new Typecho\_Request()。
9. 构造类 Typecho\_Request()。其中\_filter[0] = 'assert';对应的是被调用的函数。\_params['screenName'] = 'phpinfo()';对应的是被调用的函数的参数。这都是我们可以控制的，也是反序列化漏洞的来源。
10. 对变量exp序列化并编码，对应的代码：echo base64\_encode(serialize($exp));
11. 将得到的反序列化字符串作为get的参数传递。