**《软件安全》实验报告**

姓名：胡博浩 学号：2212998 班级：信息安全

**一、实验名称：**

复现反序列化漏洞

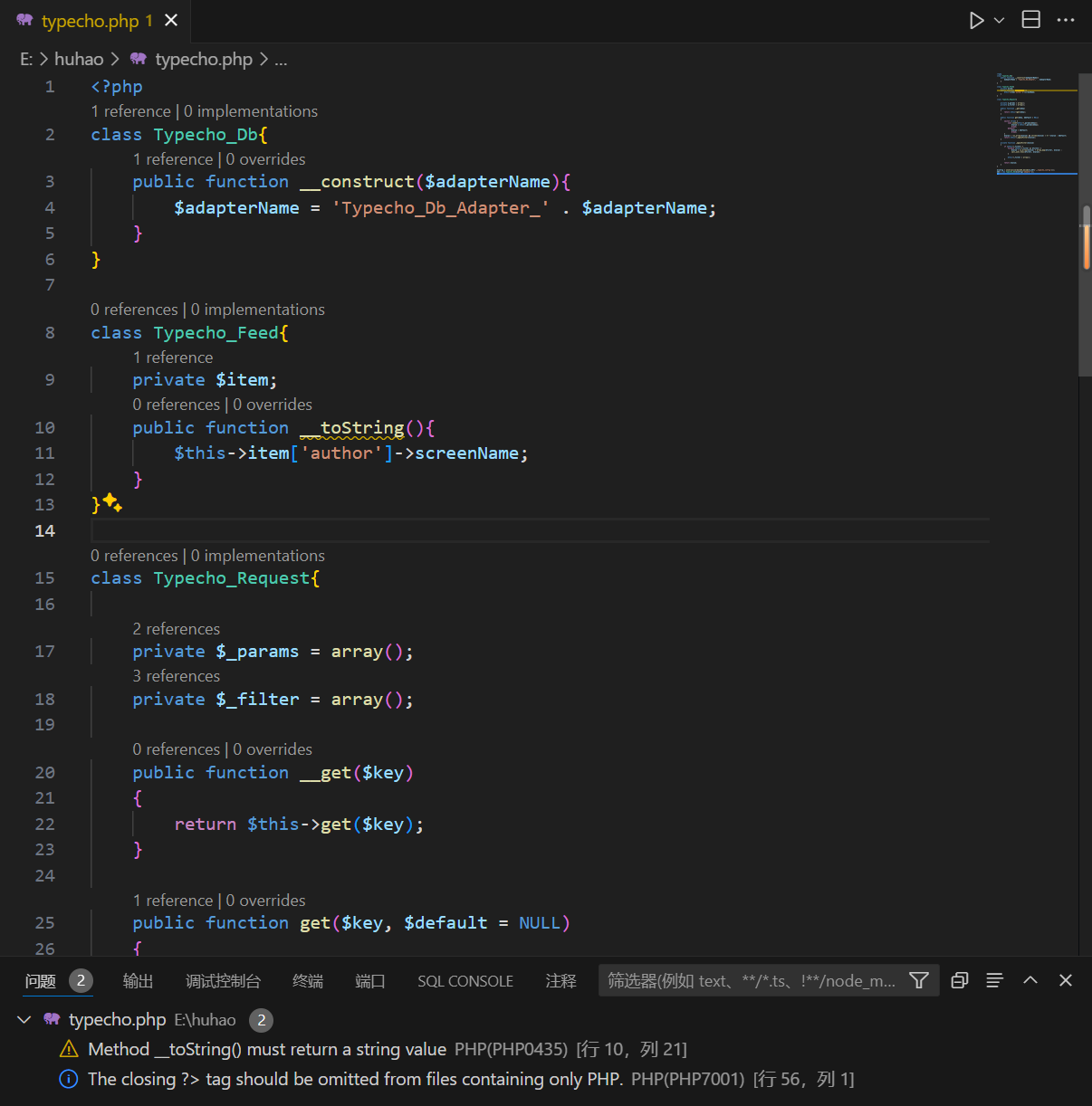
**二、实验要求：**

复现12.2.3中的反序列化漏洞，并执行其他的系统命令。

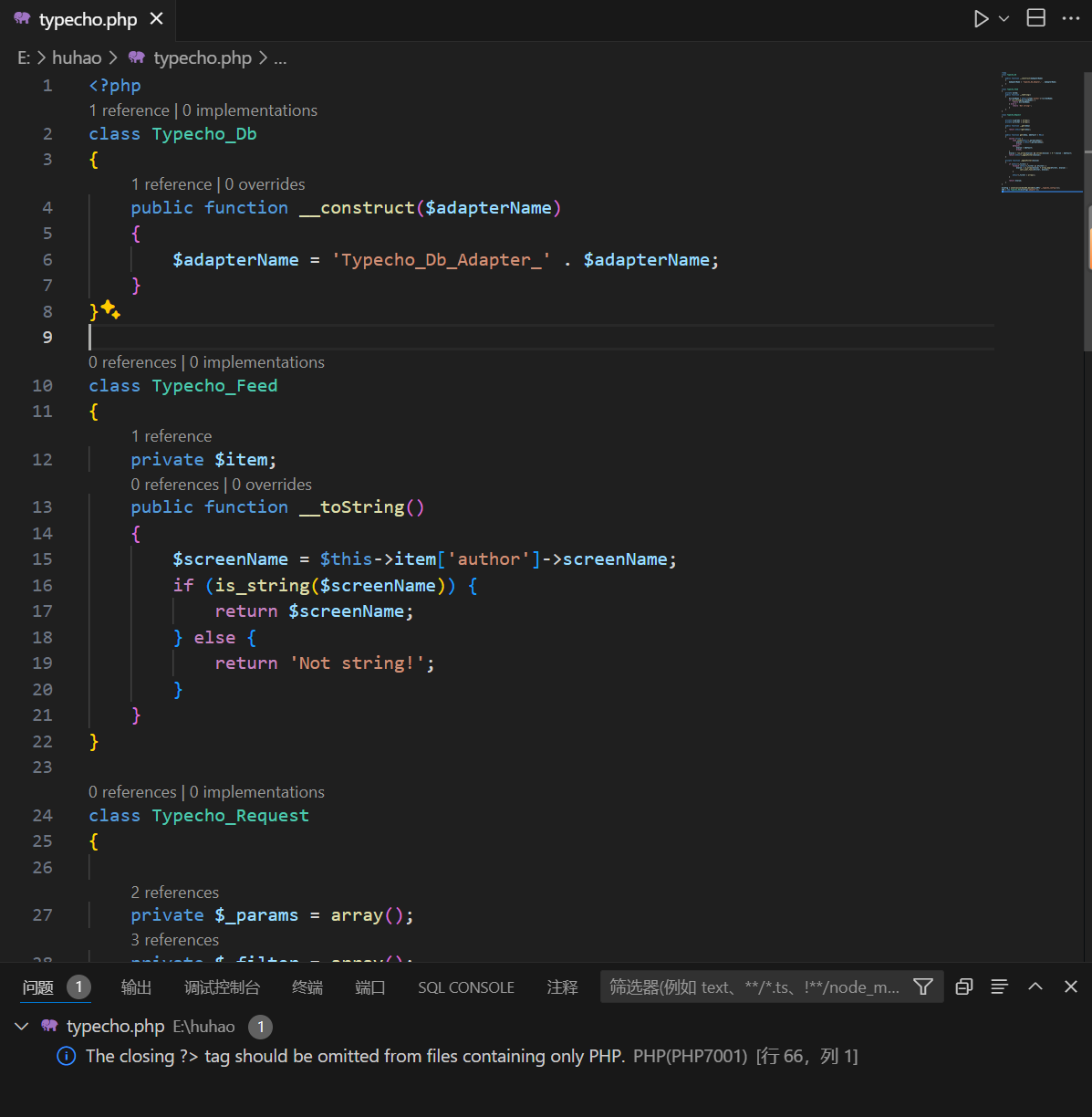
**三、实验过程：**

1. 构建环境条件：

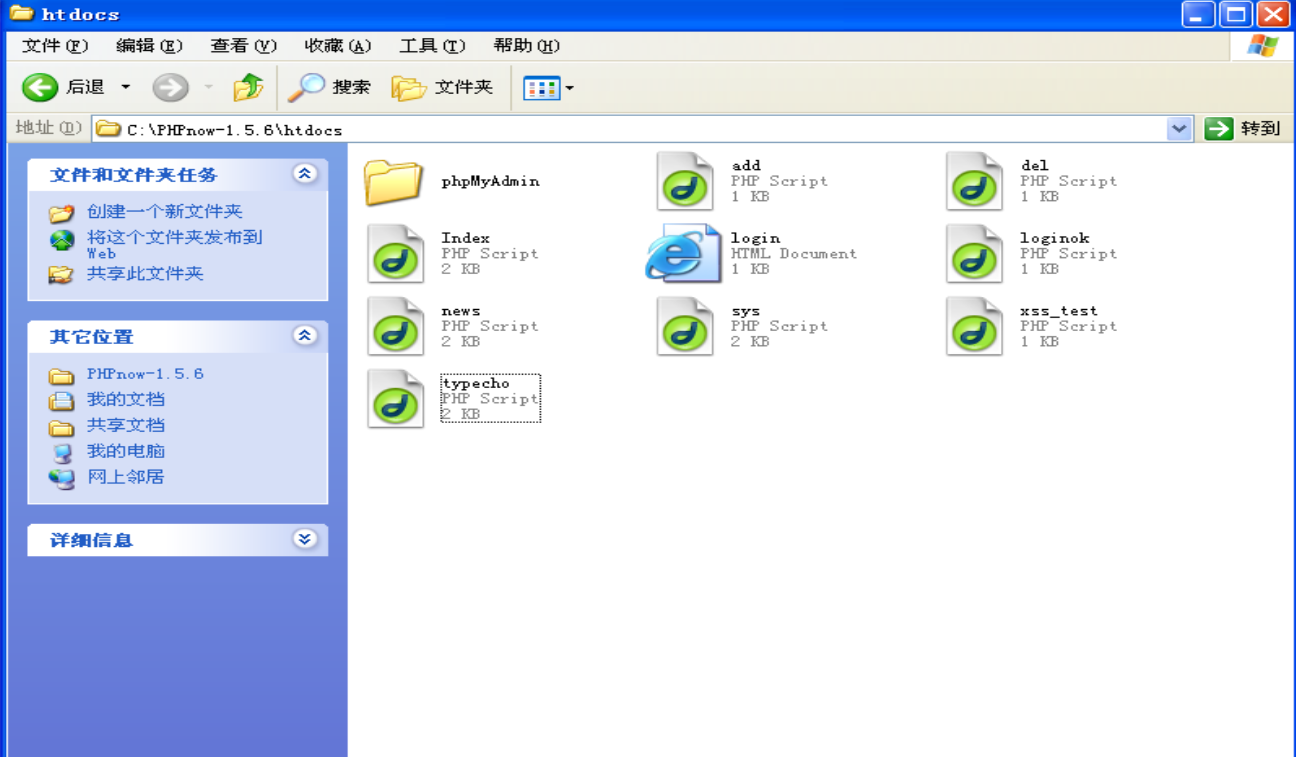
为了构建环境条件，需要创建一个能够被利用反序列化漏洞的php页面，新建文件typecho.php，代码即书上给出的。发现报错：



根据错误描述，检查 typechp.php 后发现问题出在 Typecho\_Feed 类的 \_\_toString() 方法中。在 PHP中， \_\_toString() 方法必须返回一个字符串值。但在代码中，这个方法并没有返回任何值，因此PHP 抛出了一个错误。修正代码如下：

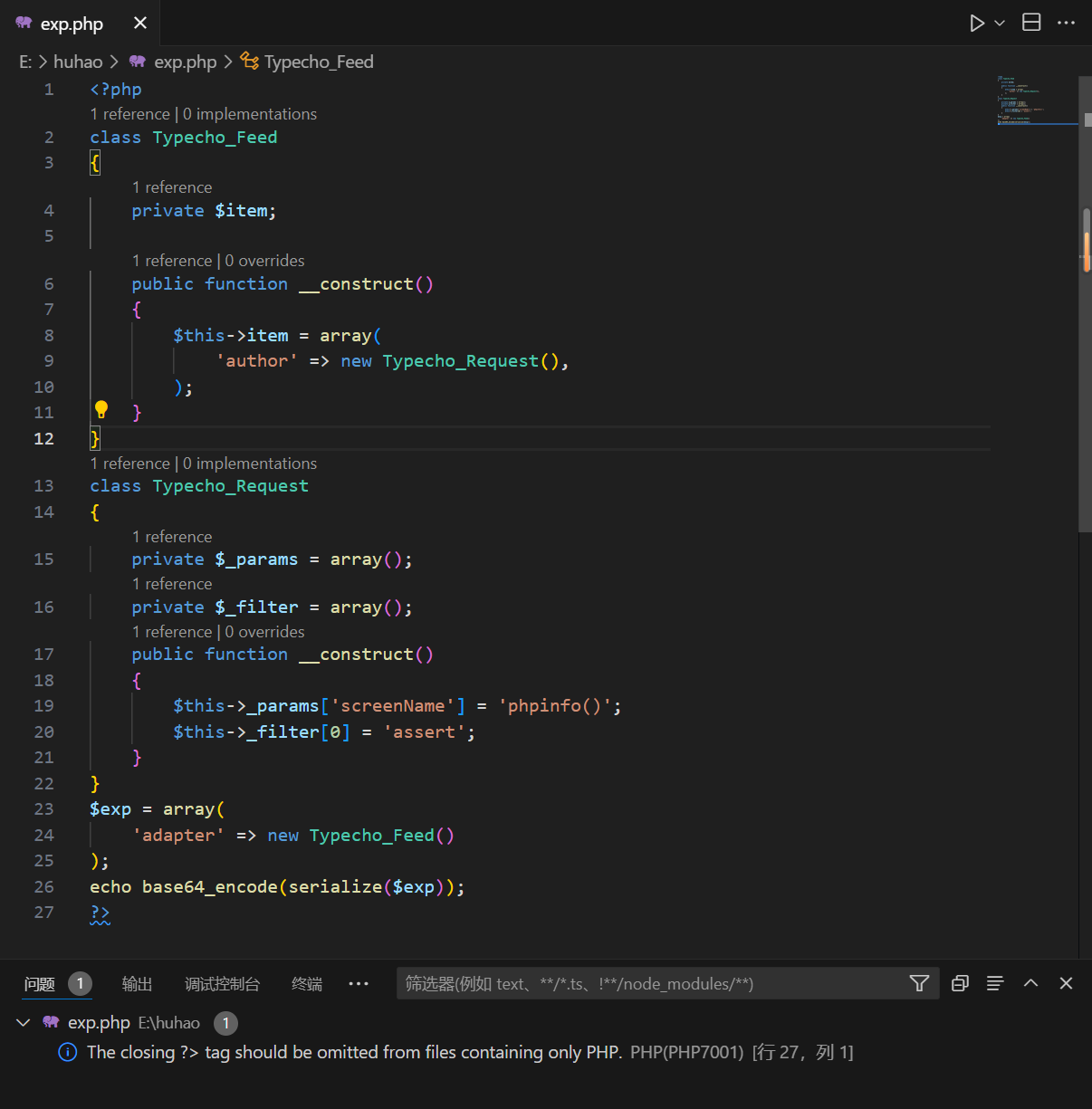


将文件放在PHPnow的htdocs目录下，在本次实验中我仍旧使用虚拟机WindowsXP：



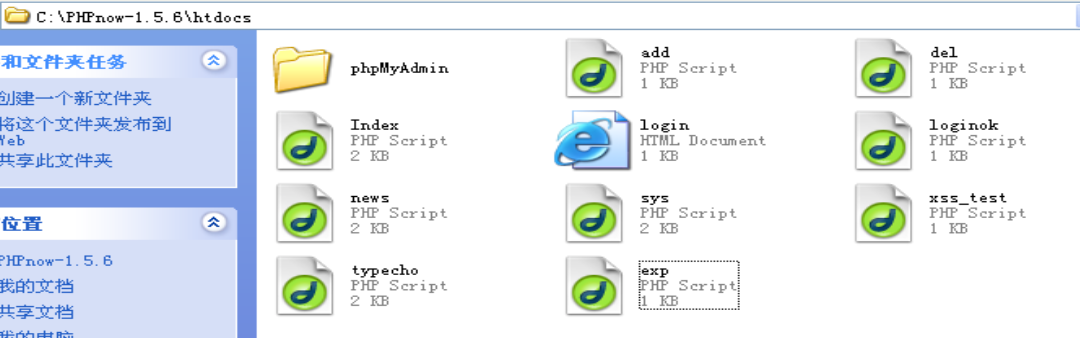
2. 创建PHP对象注入攻击文件：

新建文件exp.php，写入以下代码：



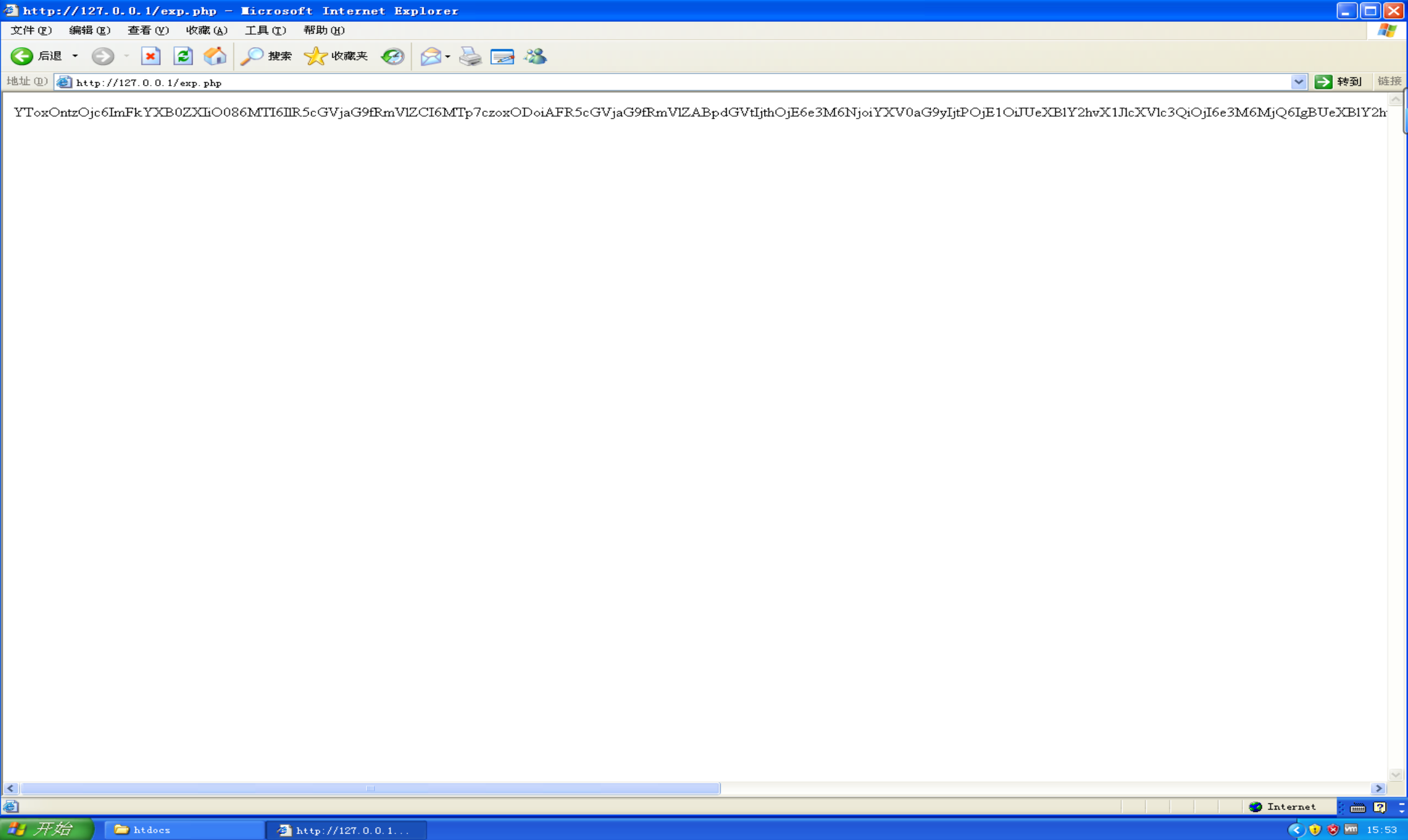
代码逻辑为：构造一个Typecho\_Request对象，然后给对象的\_params和\_filter属性赋值，其中\_filter为想要执行的函数，这里可以使用assert函数进一步执行我们最终想要执行的函数，\_params中则存放我们最终想要执行的函数。然后将这个Typecho\_Request对象作为Typecho\_Feed中item的值，并最后将一个包含以’adapter’为key，Typecho\_Feed为value的键值对进行序列化。

同理，将文件放入htdocs目录下：

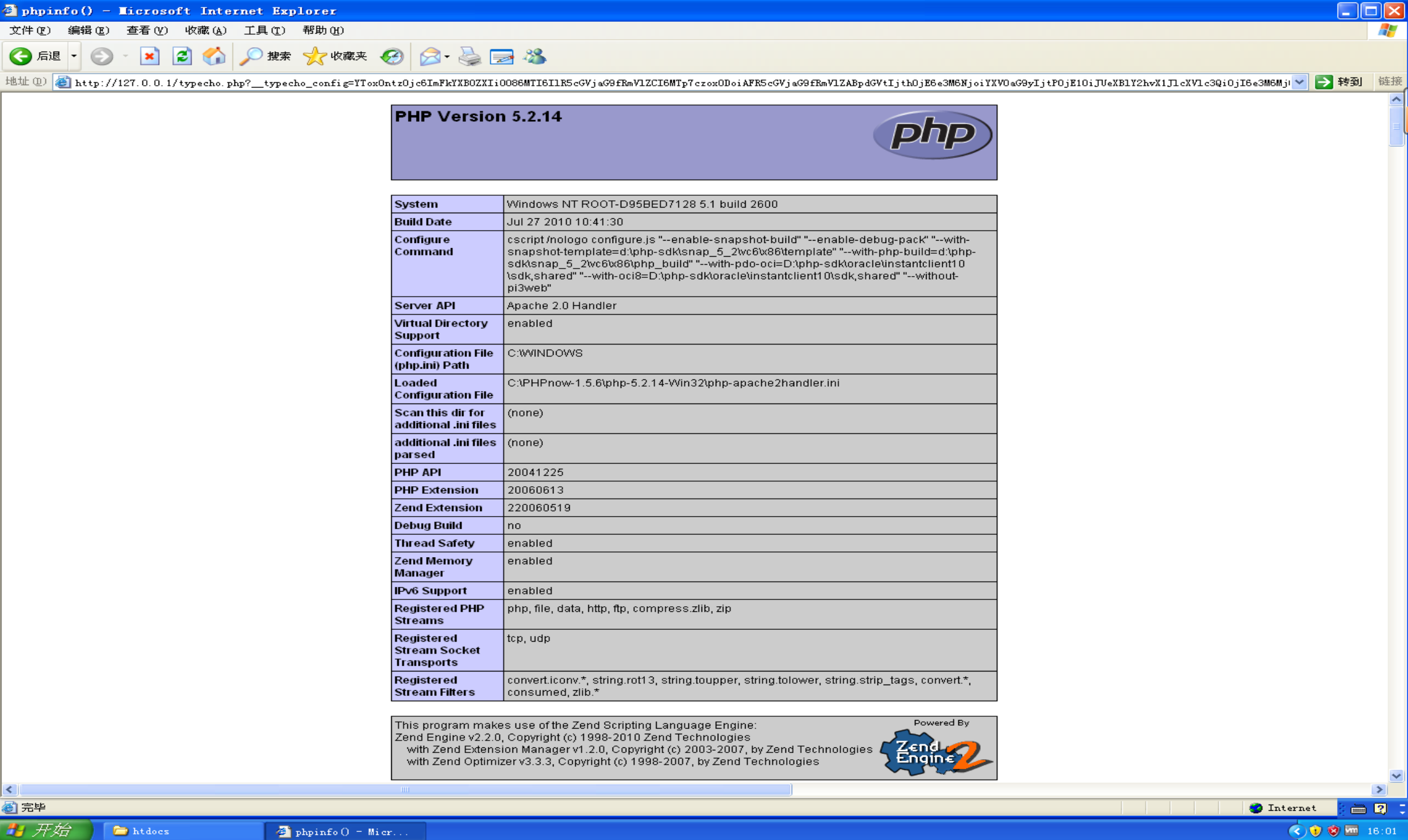


3. 复现反序列化漏洞：

首先尝试执行phpinfo()，访问URL：http://127.0.0.1/exp.php , 即可获取到对应的Payload，如下图所示：



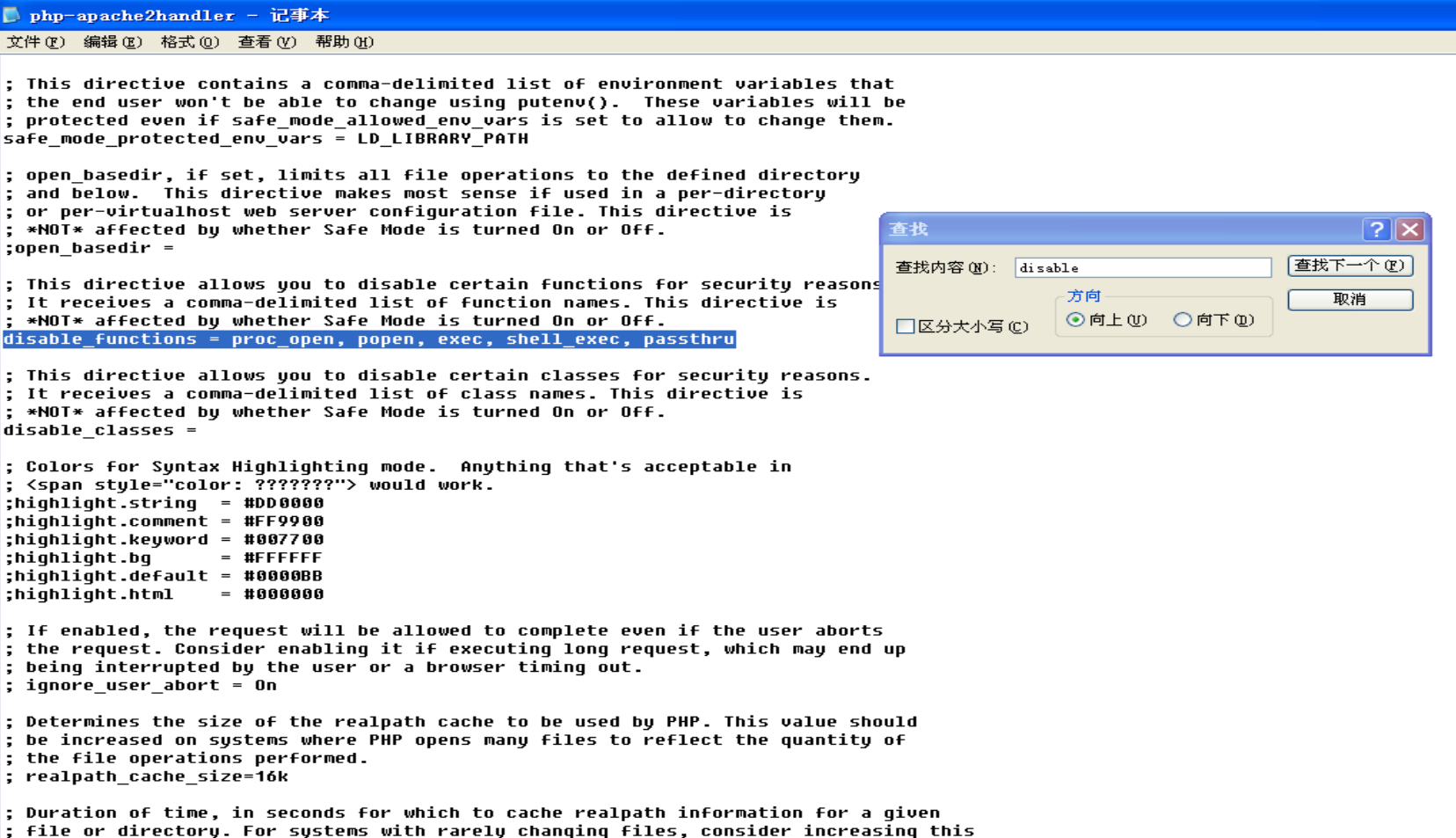
将Payload拼接到typecho.php的\_\_typecho\_config参数中，访问对应的URL：



可以看到成功执行了phpinfo()代码，访问到了服务器上的php信息！

3. 尝试执行系统命令：

为了执行系统命令，需要先在php-apache2handler.ini将系统命令system解除禁用，以便系统命令可以正常执行。即将system从disable\_functions去掉，修改如下：

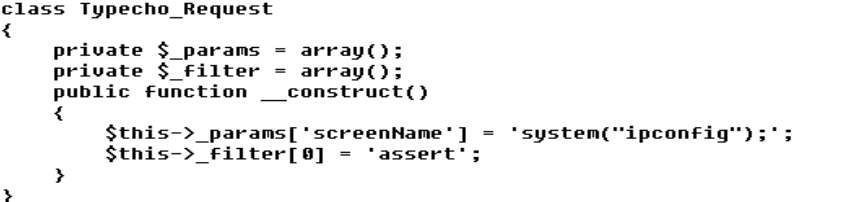


然后将上述 exp.php 中要执行的代码进行替换，即把：

$this->\_params['screenName'] = 'phpinfo()';

替换成：

$this->\_params['screenName'] = 'system("ipconfig");';



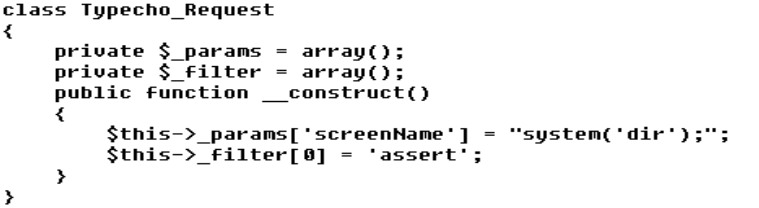
再次访问http://127.0.0.1/exp.php ,得到如下Payload:

YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjE5OiJzeXN0ZW0oImlwY29uZmlnIik7Ijt9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0

将其同样拼接到 \_\_typecho\_config 参数中，访问后执行结果如下：

表明执行命令成功！

接下来再试另一个命令，将 exp.php 执行代码那一行替换为：$this->\_params['screenName'] = "system('dir');";



然后同理进行上述操作，最后执行结果如图所示：



可见该命令也执行成功！

因此，三个命令phpinfo()、ipconfig、dir均执行成功，实验圆满完成！

**四、心得体会：**

在这次实验中，我深入学习了PHP反序列化漏洞的基本概念。这种漏洞的核心问题在于，当服务器进行对象反序列化操作时，由于缺少必要的验证措施，攻击者有可能借此执行恶意代码。

通过实验，我发现利用反序列化漏洞并不直接，这需要对反序列化过程和PHP魔法函数的执行有深入理解，以便找到可利用的突破口。在实验初期，我学会了如何构建恶意对象并将其序列化，随后探索了如何在服务器的反序列化环节触发这些对象中的恶意代码，特别是借助PHP的魔法函数来实现这一目标。我意识到，在反序列化过程中自动调用的这些魔法函数，可以在对象生命周期的特定时刻被恶意使用，以执行预设的攻击代码。

然而，挖掘反序列化漏洞是一项复杂的工作，它要求对目标应用的代码结构、对象序列化与反序列化流程有透彻的了解。实验让我认识到，要结合代码审计、动态调试及日志分析等多种手段，系统性地识别反序列化过程中的潜在威胁。

为了提升应用的安全性，我们可以通过修改php.ini配置文件来禁用高风险函数，并设置open\_basedir来限制文件操作的范围。

总结来说，这次实验不仅加深了我对PHP反序列化漏洞的认识，还让我掌握了发掘及防御这类漏洞的有效方法。这些宝贵的知识和技巧将助力我在未来的软件开发和安全测试中更准确地识别和防范潜在的安全风险，从而增强代码的安全性与稳定性。