《软件安全》实验报告

姓名：冯思程 学号：2112213 班级： 计算机科学与技术二班

**实验名称：**

反序列化漏洞

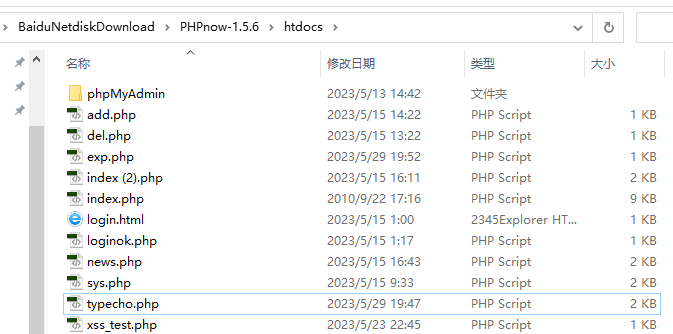
**实验要求：**

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：**

1. **首先进行网页代码的编写：**

这里我选择在本机windows10操作系统下进行实验，首先找到phpnow安装路径下的htdocs文件夹，新建两个文本文件，然后分别重命名为typecho.php、exp.php，如下：



然后用记事本打开方式打开文本文档，将教材上给出的示例代码粘贴到上面，如下：



其中typecho.php文件是从现实世界精简出来的一个简单实例。

根据typecho.php文件的代码进行分析：

该 web 应用通过$\_GET[‘\_\_typecho\_config’]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，unserialize()的参数可控，这里是漏洞的入口点。

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，而在 PHP 魔术方法中，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。

在类 Typecho\_Feed 的\_\_toString()方法中，会访问类中私有变量$item[‘author’]中 的 screenName，这里又有一个 PHP 反序列化的知识点，如果$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。

而在exp.php文件的代码中用到了 PHP 的 assert()函数，如果该函数的参数是字符串，那么该字符串会被 assert()当做 PHP 代码执行，这一点和 PHP 一句话木马常用的 eval()函数有相似之处。

保存文件后，用dreamweaver打开，如下：



现在提前的准备工作已经做好，可以开始下一步的实验部分。

1. **首先访问exp.php：获取到payload：**

访问网址：[127.0.0.1/exp.php](http://127.0.0.1/exp.php)，结果如下：

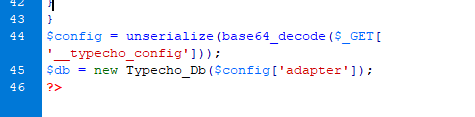


Payload的具体内容如下：

YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fX19

1. **将payload在url中通过get请求的方式传递给typecho.php：**

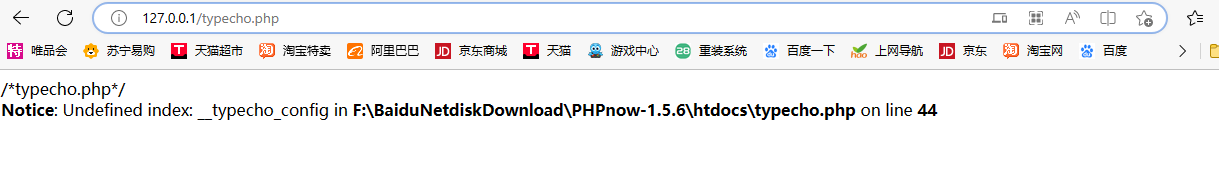
这里我们首先来搞清楚这里面的原理，打开typecho.php文件，找到第44行代码，如下：



代码内容：$config = unserialize(base64\_decode($\_GET['\_\_typecho\_config']));

这部分是漏洞的关键点，通过$\_GET[‘\_\_typecho\_config’]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，unserialize()的参数可控，这里就是漏洞的入口。

然后我去访问[127.0.0.1/typecho.php](http://127.0.0.1/tyecho.php)，界面显示如下：



说明现在payload没有被传入，还有无法识别的。

然后我们通过get请求方式将我们上文获得到的payload传入到typecho.php缺失处：

在网页的地址处输入：http://127.0.0.1/typecho.php?\_\_typecho\_config=，后面加上我们刚才获得到的payload，就可以了，结果如下：

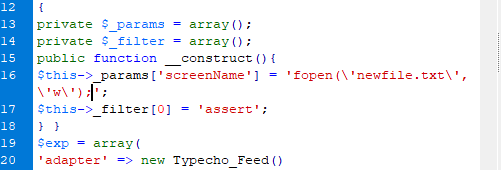


Phpinfo()成功执行，获得到复现结果。

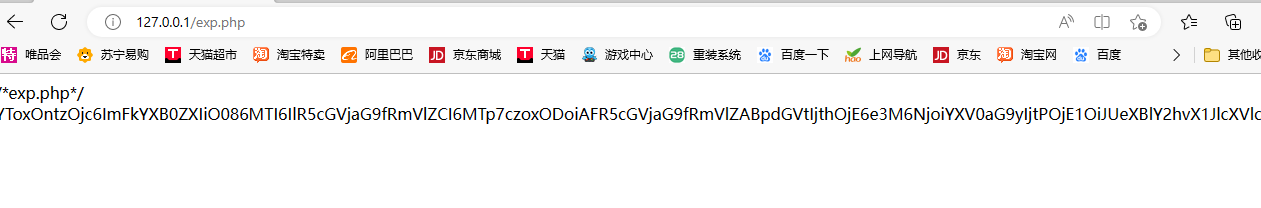
1. **执行其他系统命令：**
2. 执行fopen(\'newfile.txt\', \'w\');

这里首先要对exp文件进行修改，如果想要执行系统命令，将 phpinfo();替换为

，如下：



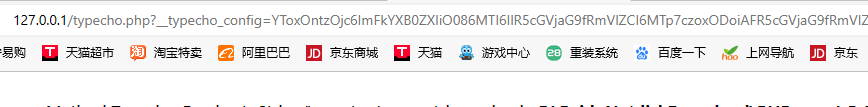
然后同理上文访问网址：[127.0.0.1/exp.php](http://127.0.0.1/exp.php)，获取到新的payload，结果如下：



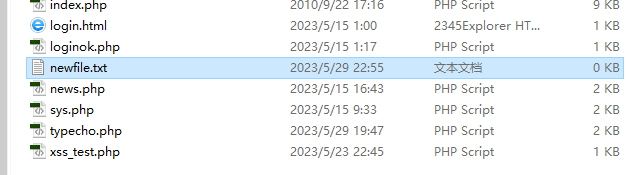
Payload内容：

YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjI2OiJmb3BlbignbmV3ZmlsZS50eHQnLCAndycpOyI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fX19

然后同理上文将payload在url中通过get请求的方式传递给typecho.php，如下：



然后观察F:\BaiduNetdiskDownload\PHPnow-1.5.6\htdocs路径下突然出现了一个新文件,如下：



对这个fopen(\'newfile.txt\', \'w\');命令的解释说明：

函数原型：FILE \*fopen(const char \*filename, const char \*mode);

参数：

i.filename-- 这是 C 字符串，包含了要打开的文件名称。

ii.mode-- 这是 C 字符串，包含了文件访问模式。

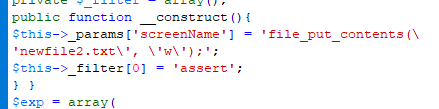
功能：使用给定的模式mode打开filename所指向的文件。

命令中的w关键字的功能：打开只写文件，若文件存在则文件长度清为零，即该文件内容会消失；若文件不存在则创建该文件。在上文中成功创建了一个名为newfile的txt文件，说明成功执行了fopen(\'newfile.txt\', \'w\');这个系统命令。

1. 执行file\_put\_contents(\'newfile2.txt\', \'w\');

这里首先要对exp文件进行修改，如果想要执行系统命令，将 phpinfo();替换为

file\_put\_contents(\'newfile2.txt\', \'w\');，如下：



然后同理上文访问网址：[127.0.0.1/exp.php](http://127.0.0.1/exp.php)，获取到新的payload，结果如下：



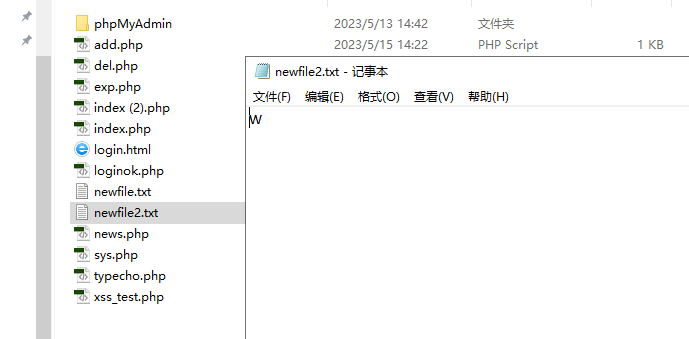
Payload内容：

YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjM5OiJmaWxlX3B1dF9jb250ZW50cygnbmV3ZmlsZTIudHh0JywgJ3cnKTsiO31zOjI0OiIAVHlwZWNob19SZXF1ZXN0AF9maWx0ZXIiO2E6MTp7aTowO3M6NjoiYXNzZXJ0Ijt9fX19fQ==

然后同理上文将payload在url中通过get请求的方式传递给typecho.php，如下：



然后观察F:\BaiduNetdiskDownload\PHPnow-1.5.6\htdocs路径下突然又出现了一个新文件,如下：（内容是一个字符w）



对这个file\_put\_contents(\'newfile2.txt\', \'w\');命令的解释说明：

函数原型：FILE \*file\_put\_contents(const char \*filename, const char charone);

参数：

i.filename-- 这是 C 字符串，包含了要打开的文件名称。

ii.charone-- 这是 C 字符串，表示要写入文件的字符。

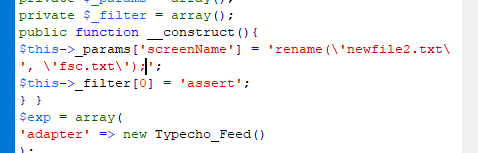
功能：打开filename所指向的文件并写入charone给定的字符，如果没有目标文件则会自动创建一个。

在上文中的流程程序成功创建了一个名为newfile2的txt文件，说明这个程序成功地执行了fopen(\'newfile.txt\', \'w\');这个系统命令。

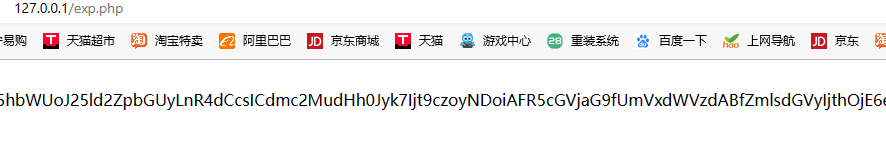
1. 执行rename(\'newfile2.txt\', \'fsc.txt\');

这里首先要对exp文件进行修改，如果想要执行系统命令，将 phpinfo();替换为

rename(\'newfile2.txt\', \'fsc.txt\');，如下：



然后同理上文访问网址：[127.0.0.1/exp.php](http://127.0.0.1/exp.php)，获取到新的payload，结果如下：



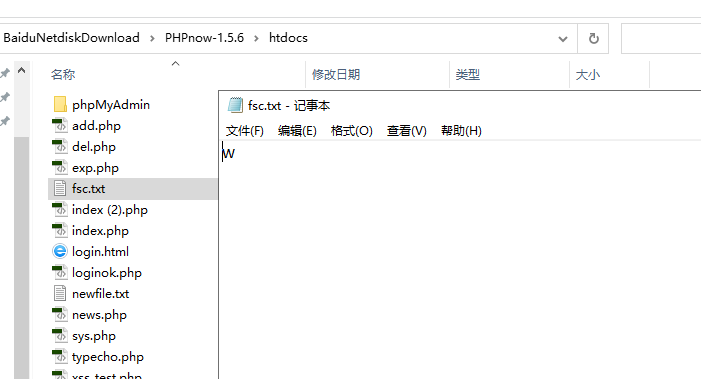
Payload内容：

YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjM0OiJyZW5hbWUoJ25ld2ZpbGUyLnR4dCcsICdmc2MudHh0Jyk7Ijt9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0=

然后同理上文将payload在url中通过get请求的方式传递给typecho.php，如下：



然后观察F:\BaiduNetdiskDownload\PHPnow-1.5.6\htdocs路径下原来的newfile2.txt文件没有了，出现了一个新的名为fsc.txt文件,如下：（内容是一个字符w）



对这个rename(\'newfile2.txt\', \'fsc.txt\');命令的解释说明：

函数原型：FILE \*rename(const char \*filename1, const char \*filename2);

参数：

i.filename1-- 这是 C 字符串，包含了要改名的文件名称。

ii.filename2-- 这是 C 字符串，表示改名后的新名字。

功能：将目标文件重命名。

在上文中的流程程序成功将一个名为newfile2的txt文件改名成了一个名为fsc的txt文件，说明这个程序成功地执行了rename(\'newfile2.txt\', \'fsc.txt\');命令。

**心得体会：**

在进行反序列化漏洞的实验过程中，我对漏洞的利用原理有了更深入的理解。通过构造恶意的反序列化对象，并利用特定的代码执行路径，可以实现对目标系统的攻击。在实验中，我了解到以下几点：

1反序列化漏洞的基本原理：通过操纵用户输入，构造恶意的序列化对象，并利用目标系统在反序列化过程中的弱点来执行恶意代码。

2关键函数和类的分析：通过分析目标系统的代码，我了解到关键的类和函数，如unserialize()、\_\_toString()、\_\_get()等，这些函数和类的结合使用，形成了漏洞的利用链。

3利用payload生成恶意代码：通过构造恶意的反序列化对象和payload，我成功实现了执行恶意代码的目标。在实验中，我使用了assert()函数来执行系统命令，实现了进一步的攻击。

4安全意识的重要性：反序列化漏洞是一种严重的安全问题，能够导致远程代码执行。在开发过程中，我们应该始终注意输入验证和安全编码实践，以防止此类漏洞的出现。

5通过这个实验，我对反序列化漏洞的工作原理有了更深入的理解，并意识到了防范这类漏洞的重要性。我将继续学习和探索更多关于软件安全的知识，以提高自己的安全意识和技能。