队伍名词：trash_fish
队伍 ID：#0x00000c
队伍 Token: NYQkBpXOorAyC6xc2oudn
解出题目：
签到：TEST NC，从这里开始的序章。
CRYPTO： suprimeRSA， ezBag，sieve
MISC：Hakuya Want A Girl Friend，Level 314 线性走廊中的双生实体
RE：Compress dot new，Turtle
WEB：Level 24 Pacman，Level 47 BandBomb

签到：
TEST NC：
nc ip
cat /flag
拿到 flag
hgame{yOUR-C@N-CoNNECt_to_the-remote-3nvIRoNMenT_to-GeT-fIag0}

从这里开始的序章：
您必须注意，您所使用的键盘应该符合由 TC28（全国信息技术标准化技术委员会）归口，TC28SC28（全国信息技术标准化技术委员会办公机器、外围设备和耗材分会）执行的《信息处理用键盘通用规范》，然后使用鼠标选中题目中 hgame 开头的那串 flag（注意不是上面那个，不要被它迷惑了，真正的 flag 应该是 hgame 开头的），按下快捷键 Ctrl+C 来将其复制，然后在下方的提交框中，按下 Ctrl+Shift+V 来将其粘贴，最后按下回车键即可提交
hgame{Now-I-knOw-how-to-submlt-my-fl4gs!}

CRYPTO：
suprimeRSA：
刚开始想了一天没想出来，后来是在 lazzaro 的博客上发现了这其实是 ROCA
RSA | Lazzaro（https://lazzzaro.github.io/2020/05/06/crypto-RSA/#Return-of-Coppersmith%E2%80%99s-attack-ROCA）
然后用 GKCTF2020 Crypto 复现 仿射密码在线解密器-CSDN 博客（https://blog.csdn.net/qq_51999772/article/details/123146784）这篇文章中的 babyCtypto 提到的脚本进行攻击即可，里面还有有关 ROCA 的博客什么的
解出来
p=954455861490902893457047257515590051179337979243488068132318878264162627
之后正常解 RSA 即可

ezBag：
第一眼 AES，再看几眼很明显的 Knapsack Problem 的变式多子集问题（MultipleSubsetSumProblem）根据 A Gentle Tutorial for Lattice-Based Cryptanalysis 论文来构造格即可。

这里面用到的 BKZ 是 LLL 算法的改进，LLL 出不来，BKZ 能出来。但 BKZ 只能在 ZZ 上使用。然后找到前 n 个只含有 1 和-1 的解，这时只有两种情况，1 代表 1，或者 1 代表 0，-1 反之，两个都试一下就可以了。

Exp:

```
list=[[2826962231, 3385780583, 3492076631, 3387360133, 2955228863,
2289302839, 2243420737, 4129435549, 4249730059, 3553886213,
3506411549, 3658342997, 3701237861, 4279828309, 2791229339,
4234587439, 3870221273, 2989000187, 2638446521, 3589355327,
3480013811, 3581260537, 2347978027, 3160283047, 2416622491,
2349924443, 3505689469, 2641360481, 3832581799, 2977968451,
4014818999, 3989322037, 4129732829, 2339590901, 2342044303,
3001936603, 2280479471, 3957883273, 3883572877, 3337404269,
2665725899, 3705443933, 2588458577, 4003429009, 2251498177,
2781146657, 2654566039, 2426941147, 2266273523, 3210546259,
4225393481, 2304357101, 2707182253, 2552285221, 2337482071,
3096745679, 2391352387, 2437693507, 3004289807, 3857153537,
3278380013, 3953239151, 3486836107, 4053147071], [2241199309,
3658417261, 3032816659, 3069112363, 4279647403, 3244237531,
2683855087, 2980525657, 3519354793, 3290544091, 2939387147,
3669562427, 2985644621, 2961261073, 2403815549, 3737348917,
2672190887, 2363609431, 3342906361, 3298900981, 3874372373,
4287595129, 2154181787, 3475235893, 2223142793, 2871366073,
3443274743, 3162062369, 2260958543, 3814269959, 2429223151,
3363270901, 2623150861, 2424081661, 2533866931, 4087230569,
2937330469, 3846105271, 3805499729, 4188683131, 2804029297,
2707569353, 4099160981, 3491097719, 3917272979, 2888646377,
3277908071, 2892072971, 2817846821, 2453222423, 3023690689,
3533440091, 3737441353, 3941979749, 2903000761, 3845768239,
2986446259, 3630291517, 3494430073, 2199813137, 2199875113,
3794307871, 2249222681, 2797072793], [4263404657, 3176466407,
3364259291, 4201329877, 3092993861, 2771210963, 3662055773,
3124386037, 2719229677, 3049601453, 2441740487, 3404893109,
3327463897, 3742132553, 2833749769, 2661740833, 3676735241,
2612560213, 3863890813, 3792138377, 3317100499, 2967600989,
2256580343, 2471417173, 2855972923, 2335151887, 3942865523,
2521523309, 3183574087, 2956241693, 2969535607, 2867142053,
2792698229, 3058509043, 3359416111, 3375802039, 2859136043,
3453019013, 3817650721, 2357302273, 3522135839, 2997389687,
3344465713, 2223415097, 2327459153, 3383532121, 3960285331,
3287780827, 4227379109, 3679756219, 2501304959, 4184540251,
3918238627, 3253307467, 3543627671, 3975361669, 3910013423,
3283337633, 2796578957, 2724872291, 2876476727, 4095420767,
3011805113, 2620098961], [2844773681, 3852689429, 4187117513,
3608448149, 2782221329, 4100198897, 3705084667, 2753126641,
```

```
3477472717, 3202664393, 3422548799, 3078632299, 3685474021,
3707208223, 2626532549, 3444664807, 4207188437, 3422586733,
2573008943, 2992551343, 3465105079, 4260210347, 3108329821,
3488033819, 4092543859, 4184505881, 3742701763, 3957436129,
4275123371, 3307261673, 2871806527, 3307283633, 2813167853,
2319911773, 3454612333, 4199830417, 3309047869, 2506520867,
3260706133, 2969837513, 4056392609, 3819612583, 3520501211,
2949984967, 4234928149, 2690359687, 3052841873, 4196264491,
3493099081, 3774594497, 4283835373, 2753384371, 2215041107,
4054564757, 4074850229, 2936529709, 2399732833, 3078232933,
2922467927, 3832061581, 3871240591, 3526620683, 2304071411,
3679560821]]
bag=[123342809734, 118191282440, 119799979406, 128273451872]
ciphertext=b'\x1d6\xcc}\x07\xfa7G\xbd\x01\xf0P4^Q"\x85\x9f\xac\x98\x8
f#\xb2\x12\xf4+\x05`\x80\x1a\xfa !\x9b\xa5\xc7g\xa8b\x89\x93\x1e\xedz
\xd2M;\xa2'

B=[[0 for i in range(len(list[0])+5)] for i in range(len(list[0])+1)]
de=2
for i in range(len(list[0])):
    B[i][i]=de
    B[i][len(list[0])+ 1]=list[0][i]
    B[i][len(list[0])+ 2]=list[1][i]
    B[i][len(list[0])+ 3]=list[2][i]
    B[i][len(list[0])+ 4]=list[3][i]
    B[len(list[0])][i]=de/2
B[-1][-1]=bag[-1]
B[-1][-2]=bag[-2]
B[-1][-3]=bag[-3]
B[-1][-4]=bag[-4]
B[-1][len(list[0])]=de/2
B=matrix(ZZ,B)
B_=B.BKZ()
for i in range(len(list[0]) + 1):
    M = B_.row(i).list()
    flag = True
    for m in M[:-5]:
        if m != de/2 and m != -de/2:
            flag = False
            break
    if flag:
        print(M)
a=[1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1, -1, -1,
1, -1, -1, 1, 1, -1, 1, 1, -1, 1, -1, 1, 1, 1, -1, 1, -1, 1, 1, 1, -
```

```
1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, -1, 1, 1,
-1, 1, 1, 1, 1][::-1]
c=''
for i in a:
    if i==-1:
        c+='0'
    else:
        c+='1'


p=int(c,2)
#p=17739748707559623655
key = hashlib.sha256(str(p).encode()).digest()
cipher = AES.new(key, AES.MODE_ECB)
print(cipher.decrypt(ciphertext))
b'hgame{A_S1mple_Modul@r_Subset_Sum_Problem}\x06\x06\x06\x06\x06\x06'
```

sieve：
之前没见过什么筛什么筛，觉得很难，后来看看再结合一下 AI，觉得可以去试试线性筛
题目是要我们求 e**2//6 以内的所有数的欧拉函数的和，如果是质数再额外+1
线性筛将就一下，然后正常解 RSA 即可
Exp:

```
from Crypto.Util.number import *
from sympy import nextprime
import time

def sieve_phi(n):
    global phi,prime,__sum
    __sum=1
    phi = [0] * (n + 1)
    prime = []
    phi[1] = 1
    print(1)
    for i in range(2, n + 1):
        if i%100000==0:
            print(i,'/',n,'|',i/n)
        if phi[i] == 0:
            __sum+=1
            phi[i] = i - 1
            prime.append(i)
        for p in prime:
            if i * p > n:
                break
```

```
            if i%p==0:
                phi[p*i]=phi[i]*p
                break
            phi[p*i]=phi[i]*(p-1)
        __sum+=phi[i]
    return __sum

e = 65537
n=e**2//6
c=244929409747471413653014009978459273276644448166527803806948446666550615396785106320940233602506547617261737654 6
result=sieve_phi(n)
print(result)
#155763335447735055
p=nextprime(result<<128)
n=p**2
m=p**2-p
d=inverse(e,m)
plain=pow(c,d,n)
print(long_to_bytes(plain))
#hgame{sieve_is_n0t_that_HArd}
```

MISC：

Hakuya Want A Girl Friend：

打开 txt 发现全是 16 进制，去在线 16 进制字符串转文件工具 - 在线工具网（https://tool.hiofd.com/hex-convert-file-online/）上转成文件，发现是压缩包。

丢进随波逐流里面发现压缩包后面还有。搜索文件头尾后没什么有用的东西，但仔细看会发现这其实是 png 文件的逆序，提取出来得到学长的帅照，丢进随波逐流里面，发现文件高宽被修改过，修复后可以在照片底部发现一串文字，是压缩包的密码，解压后得到 flag（我之前提取压缩包的时候一直没带上最后的 0，然后一直提示压缩包损坏不给解压缩…）

hgame{h4kyu4_w4nt_girlf3nd_+q_931290928}

Level 314 线性走廊中的双生实体：

不知道什么东西，丢给豆包分析一下

估计是要输入一个正确的张量，然后会返回 flag

那就开爆吧:））

期间发现张量的第二个维度必须是 10

豆包给了个随机张量的爆破

Exp:

```
import torch

# 加载模型
```

```
entity = torch.jit.load('Model.pt')

# 假设输入张量第一个维度为 1，第二个维度为 10，数据类型为 float32
input_shape = (1, 10)
dtype = torch.float32

# 设定目标字符串
target_string = "your_special_string"

# 设定最大尝试次数
max_trials = 10000

for trial in range(max_trials):
    # 生成随机输入张量，这里使用标准正态分布生成
    input_tensor = torch.randn(*input_shape, dtype=dtype)

    try:
        # 运行模型得到输出
        output = entity(input_tensor)

        # 尝试将输出转换为字符串
        if isinstance(output, str):
            result_string = output
        elif isinstance(output, torch.Tensor):
            try:
                # 假设输出张量是一维且元素可解释为字符编码
                result_string = ''.join(chr(int(x)) for x in
output.flatten() if 0 <= int(x) <= 127)
            except ValueError:
                continue
        else:
            try:
                result_string = str(output)
            except ValueError:
                continue

        # 检查输出是否为目标字符串
        if result_string == target_string:
            print(f"找到目标字符串！尝试次数: {trial + 1}")
            print(f"输入张量: {input_tensor}")
            break

    except Exception as e:
        print(f"第 {trial + 1} 次尝试时发生错误: {e}")
```

```
else:
    print(f"在 {max_trials} 次尝试后未找到目标字符串。")
```

输出：

```
Hidden: flag{s0_th1s_1s_r3al_s3cr3t}
Decoy: flag{fake_flag}
Decoy: flag{fake_flag}
Hidden: flag{s0_th1s_1s_r3al_s3cr3t}
Hidden: flag{s0_th1s_1s_r3al_s3cr3t}
Decoy: flag{fake_flag}
在 10000 次尝试后未找到目标字符串。
```

其实爆率还挺高？

RE：

Compress dot new：

~~第一反应先丢进 IDA 里面看看~~

直接给源码了，那就丢给豆包分析一下，

代码功能概述

你提供的这段代码是用 Nu 编程语言编写的，它实现了一个数据压缩的功能，主要基于哈夫曼编码算法。哈夫曼编码是一种用于无损数据压缩的熵编码算法，通过构建哈夫曼树，将出现频率较高的字符用较短的编码表示，出现频率较低的字符用较长的编码表示，从而达到压缩数据的目的。

那么可以直接让豆包写个脚本解压缩，稍微改动一下即可

```
def decode_huffman(encoded_text, huffman_dict):
    decoded_text = []
    current_dict = huffman_dict
    for bit in encoded_text:
        if 's' in current_dict:
            # 如果当前是叶子节点，记录值并回到根节点
            decoded_text.append(current_dict['s'])
            current_dict = huffman_dict
        if bit == '0':
            current_dict = current_dict.get('a')
```

```
            else:
                current_dict = current_dict.get('b')

            if current_dict is None:
                print(f"错误：在编码 {bit} 处未找到对应的子节点，当前字典状态：{current_dict}")
                break

    # 处理最后一个叶子节点
    if 's' in current_dict:
        decoded_text.append(current_dict['s'])

    return ''.join(map(chr, decoded_text))


huffman_dict = {
    "a": {
        "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}
                }
            }
        }
    }
}

encoded_text =
"00010001110111111010010000011100010111000100111000110000
100010111001110010011011010101111011101100110100011101101
001110111110111011011001110110011110011110110111011101101
```

```
0110011110110011110001110011011110000110011000010110111011
0001110010100111001011100111100001100010100101000000010010
1000100010011111110110010111010101000111101000110110001
1101010110100111111110011111110110101011000011011101011011
1111010010011110010001011010111111111111001100010101011011
1001001111100011011010110111101000001111010000001101101010
1100011111100011010100101110000011011110000000100101000100
0101110001110011100101110101111100010101011010111100000110
0111100011100101110101111100010110101110000010100000001011
0001111011100011101111110101010010011101011100100011110
0100101101111011101110101111101100011110101011100100010111
0010010111000101101010000111010100010111101010011000111010
1011101100011011011000011010000000101100011101111111110001
0101010111000000"
```

```
decoded_text = decode_huffman(encoded_text, huffman_dict)
print("解码后的文本:", decoded_text)
```

```
解码后的文本: hgame{Nu-Shell-scr1pts-ar3-1nt3r3st1ng-t0-wr1te-&-use!}

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nulla nec ligula neque. Etiam et viverra nunc, vel bibendum risus.
Donec.
```

Turtle：
从题目描述来看应该是有壳了，IDA 也是这么说的

Re 题的壳只做过 UPX 的，丢进 UPX 的查壳工具发现确实有 UPX 壳，但脱壳程序失败了，提示 NotCompressibleException，去网上搜索发现可以尝试手动脱壳[如何用 x64dbg UPX 手动脱壳（64 位）_x64dbg 脱壳-CSDN 博客](https://blog.csdn.net/Pisces50002/article/details/135169522)（https://blog.csdn.net/Pisces50002/article/details/135169522），跟着教程一步一步来，要下断点的地方什么的都是可以找到的。脱完壳后丢进 IDA 里面，然后让豆包一通分析，知道了是 RC4，加密解密脚本一样，直接让豆包把它转成 python，稍微修改一下，把第二个加密函数的-改为+，发现解不出来，key 输入原文件也说是错的，反复质疑豆包，尝试自己加密和解密后，函数都是没问题的，但 flag 就是出不来，GPT 也问过了，那时已经凌晨 2，3 点了，就先去睡觉了。第二天仔细研究的时候发现 key 的长度是 5，但是传进去的长度是 7，很奇怪，发现 key 和另一个长度为 2 的数组一起定义了，那么猜测存储空间应该是连续的，把另一个数组加上后解出正确的 flag

Exp:

```
def sub_401550(key):
    s_box = list(range(256))
    v5 = 0
    key_length = len(key)
    for j in range(256):
        v5 = (s_box[j] + v5 + key[j % key_length]) % 256
        s_box[j], s_box[v5] = s_box[v5], s_box[j]
    return s_box

def sub_40163E(data, s_box):
    v7 = 0
    v6 = 0
    result = data[:]
    for i in range(len(data)):
        v7 = (v7 + 1) % 256
        v6 = (s_box[v7] + v6) % 256
        s_box[v7], s_box[v6] = s_box[v6], s_box[v7]
        index = (s_box[v7] + s_box[v6]) % 256
        result[i] ^= s_box[index]
    return result

def sub_40175A(data, data_length, s_box):
    v7 = 0
    v6 = 0
    result = data[:]
    for i in range(data_length):
        # 更新 v7 并取模 256
        v7 = (v7 + 1) % 256
        # 更新 v6 并取模 256
        v6 = (s_box[v7] + v6) % 256
        # 交换 s_box 中 v7 和 v6 位置的元素
```

```
            s_box[v7], s_box[v6] = s_box[v6], s_box[v7]
            # 计算伪随机字节的索引
            index = (s_box[v7] + s_box[v6]) % 256
            # 用输入数据当前位置的字节减去伪随机字节，并取模 256
            result[i] = (result[i] + s_box[index]) % 256
    return bytes(result)

# 假设这些值对应代码中的实际值
v11 = [ord(c) for c in "yekyek"]
Buf2 = [-51, -113, 37, 61, -31 ,ord('Q'),ord('J')]  #[205, 143, 37,
61, 225]
#Buf2 = [101, 99, 103, 52, 97]
# 将 Buf2 转换为无符号字节
Buf2 = [b % 256 for b in Buf2]

# 处理 v5 为无符号字节
v5 = [-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]
#v5=[196, 124, 171, 201, 197, 219, 3, 135, 113, 89, 133, 106, 36, 64,
101]
v5 = [b % 256 for b in v5]

# 初始化 S 盒
s_box = sub_401550(v11)
# 对 Buf2 进行解密
decrypted_key = sub_40163E(Buf2, s_box)

print("用户应输入的密钥:", decrypted_key,bytes(decrypted_key))

s_box=sub_401550(decrypted_key)
flag=sub_40175A(v5,len(v5),s_box)
print(flag)
用户应输入的密钥: [101, 99, 103, 52, 97, 98, 54] b'ecg4ab6'
b"hgame{YOu'r3_re4lly_g3t_OUt_of_th3_upX!}"
```

WEB：

Level 24 Pacman：

无需多盐，先开一把再说

中道崩殂后给了串 base64，解完后非常明显的栅栏，然后成功得到了假的 flag

~~菜，就多练~~

接着开始分析，刚开始以为是要更改游戏分数来直接达到 1W 分，然后把三个相关的文件全部丢给豆包分析了一通无果。

然后开始尝试搜索 flag,hgame,gift，在 gift 那边能发现还有一串不一样的 base64，按照上面的步骤解出来就是真的 flag
hgame{u_4re_pacman_m4ster}

Level 47 BandBomb：
先丢豆包分析着
刚开始搜索的是文件上传的 web 题，全是教你一句话木马的…
根据锤出题人得到的信息和豆包给出的各种函数的用法得知，网站的模板文件名应该是 mortis.ejs，在使用 rename 功能把模板文件转移到了 public 文件夹后下载下来，尝试注入点什么东西
原本一直以为 flag 是在里面的一个文件夹里（文件上传题全责），看看能不能让他输出路径或者环境变量什么的，然后 GPT 构造出以下模板

```
<!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>
<div class="meme-section">
            <h3>环境变量:</h3>
            <div id="envVariables" class="file-list">
                <% for (let key in process.env) { %>
                    <div class="file-item">
                        <span class="file-name"><%= key %>: <%=
process.env[key] %></span>
                    </div>
```

```
                <% } %>
            </div>
</div>
</body>
</html>
```

然后就意外的在里面发现了 flag，一直以为要找 flag 的文件…
结果是作为 FLAG 这个环境变量…
hgame{4VE_MUJ1C4-H4S_bROK3N_UP-6UT_w3-HAVE_uM1T4kI2O}