WP

1. 签到题：将推文内所有尊都假都复制，放入转换器
2. 导航的时空奶龙防御系统：明显的爆破，爆破发现时间不一致，请求包里有额外一random 参数提示错误检查网页代码发现 rsa 加密的时间戳

```
const timestamp = Math.floor(Date.now() / 1000).toString();
console.log("[LOG] Captured Time:", timestamp);
const encrypt = new JSEncrypt();
encrypt.setPublicKey(PUBLIC_KEY);
const encrypted = encrypt.encrypt(timestamp);
```

用 py 发送 request 并返回响应代码

```
def randompart(public_key):

    timestamp = str(int(time.time()))
    rsa_key = RSA.import_key(public_key)
    cipher = PKCS1_v1_5.new(rsa_key)
    encrypted = cipher.encrypt(timestamp.encode('utf-8'))
    encrypted_b64 = base64.b64encode(encrypted).decode('utf-8')
    return encrypted_b64
```

生成的请求包包含用户名 admin 密码题目已给爆破文件，已及 random

3.魔王的宝藏

一眼反序列化

不瞒隐瞒，交给 ai 的 相关反序列化知识想到寒假系统性学习

Post 的 body 如下

```
Brave=O:5:"Start":4:{s:7:"isbrave";s:7:"brave!!";s:9:"adventure";O:3:"Elf":1:{s:2:"go";O:5:"Demon
":2:{s:6:"result";O:8:"Treasure":1:{s:6:"reward";O:6:"Reward":0:{}}s:3:"end";s:5:"0x7e9";}}s:6:"skill
1";s:3:"abc";s:6:"skill2";s:32:"900150983cd24fb0d6963f7d28e17f72";}
```

线代导论

加密过程

1 生成两个 1024 位随机素数 p，q

2n=p*q

3 公钥 e=0x10001

4 计算 gift=p^q

5flag 改成整数 m

6 计算密文 c=m^emodn

对于 pq 的每一位有两个约束一个是异或等于 gift 里的对应位一个是乘积约束

所以对 pq 每一位有四种可能，通过异或筛去两种，用乘积保留到最后一种

得到 pq 后就是常规的 rsa 解密

```
def pq(n, gift):
#给出了 g 的二进制数
    g_bits = []
    for i in range(1024):
        g_bits.append((gift >> i) & 1)
    res= [(1, 1)]
    #遍历每一位
```

```
    for i in range(1, 1024):
        new_res = []
        mod = 1 << (i+1)
        n_mod = n % mod
对四种情况遍历
        for (p, q) in res:
            for p_i in [0, 1]:
                for q_i in [0, 1]:
                    if p_i ^ q_i == g_bits[i]:
                        p_new = p | (p_i << i)
                        q_new = q | (q_i << i)
#满足异或和乘法约束的加入 newres
                        if (p_new * q_new) % mod == n_mod:
                            new_res.append((p_new, q_new))
        res = new_res
    return res
最后尝试所有 pq 的可能性
res = pq(n, gift)

for (p, q) in res:
    if p * q == n:
        phi = (p-1) * (q-1)
        d = pow(e, -1, phi)
        m = pow(c, d, n)
        flag = long_to_bytes(m)
        print(flag.decode())
        break
```

Vigenere++

这个加密过程中可以发现密文有周期即为 ab 的最小公倍数又 ab<100 尝试遍历 ab 之积

对于每一个可能的解密后的文件用卡方判断是否为英文短文（该卡方函数来自 ai）

1 计算函数：

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def modinv(a, m):
计算 a 在模 m 下的逆元
    if gcd(a, m) != 1:
        return None
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
```

```
    while v3 != 0:
        q = u3 // v3
        v1, v2, v3, u1, u2, u3 = (
            u1 - q * v1,
            u2 - q * v2,
            u3 - q * v3,
            v1,
            v2,
            v3,
        )

    return u1 % m
def lcm(a, b):
    return abs(a * b) // gcd(a, b) if a and b else 0
```

字符分析函数

```
def frequency_score(text):
    """简单的英文字母频率分析评分"""
    if not text:
        return float('inf')

    english_freq = {
        'a': 0.08167, 'b': 0.01492, 'c': 0.02782, 'd': 0.04253,
        'e': 0.12702, 'f': 0.02228, 'g': 0.02015, 'h': 0.06094,
        'i': 0.06966, 'j': 0.00153, 'k': 0.00772, 'l': 0.04025,
        'm': 0.02406, 'n': 0.06749, 'o': 0.07507, 'p': 0.01929,
        'q': 0.00095, 'r': 0.05987, 's': 0.06327, 't': 0.09056,
        'u': 0.02758, 'v': 0.00978, 'w': 0.02360, 'x': 0.00150,
        'y': 0.01974, 'z': 0.00074
    }

    text = text.lower()
    total_chars = sum(1 for c in text if c in string.ascii_lowercase)
    if total_chars == 0:
        return float('inf')

    score = 0
    for char in string.ascii_lowercase:
        observed = text.count(char) / total_chars
        expected = english_freq[char]
        score += (observed - expected) ** 2

    return score
```

仿射解密函数

```
def affine (ciphertext, k1, k2):


    table = string.ascii_lowercase
    k1_inv = mod_inverse(k1, 26)

    plaintext = []
    for char in ciphertext:
        if char not in table:
            plaintext.append(char)
            continue

        c_idx = table.index(char)
        p_idx = (c_idx - k2) * k1_inv % 26
        plaintext.append(table[p_idx])

    return ''.join(plaintext)
```

主解密函数

```
def decrypt_with_params(ciphertext, a, b):
    """使用给定的 a 和 b 进行解密"""
    L = lcm(a, b)
    if L == 0 or L > len(ciphertext):
        return None, float('inf')

    # 可能的 k1 值（必须与 26 互质）
    possible_k1 = [1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25]
    possible_k2 = list(range(26))

    # 将密文分成 L 组
    groups = [''] * L
    for i, char in enumerate(ciphertext):
        if char in string.ascii_lowercase:
            groups[i % L] += char

    # 对每组找到最佳密钥
    group_keys = []
    total_chi2 = 0

    for group in groups:


        best_chi2 = float('inf')
        best_key = None
```

```
            for k1 in possible_k1:
                for k2 in possible_k2:
                    decrypted = affine_decrypt(group, k1, k2)
                    if decrypted is None:
                        continue

                    chi2 = chi_squared_score(decrypted)
                    if chi2 < best_chi2:
                        best_chi2 = chi2
                        best_key = (k1, k2)

            if best_key is None:
                return None, float('inf')

            group_keys.append(best_key)
            total_chi2 += best_chi2
        plaintext = []
        for i, char in enumerate(ciphertext):
            if char not in string.ascii_lowercase:
                plaintext.append(char)
                continue

            k1, k2 = group_keys[i % L]
            c_idx = string.ascii_lowercase.index(char)
            k1_inv = mod_inverse(k1, 26)
            p_idx = (c_idx - k2) * k1_inv % 26
            plaintext.append(string.ascii_lowercase[p_idx])

        return ''.join(plaintext), total_chi2 / L
```

得到解密后文本直接字符串搜索 MD5 爆破

图寻，绝大部分有清新的标志很简单，冰岛那题用的 ai