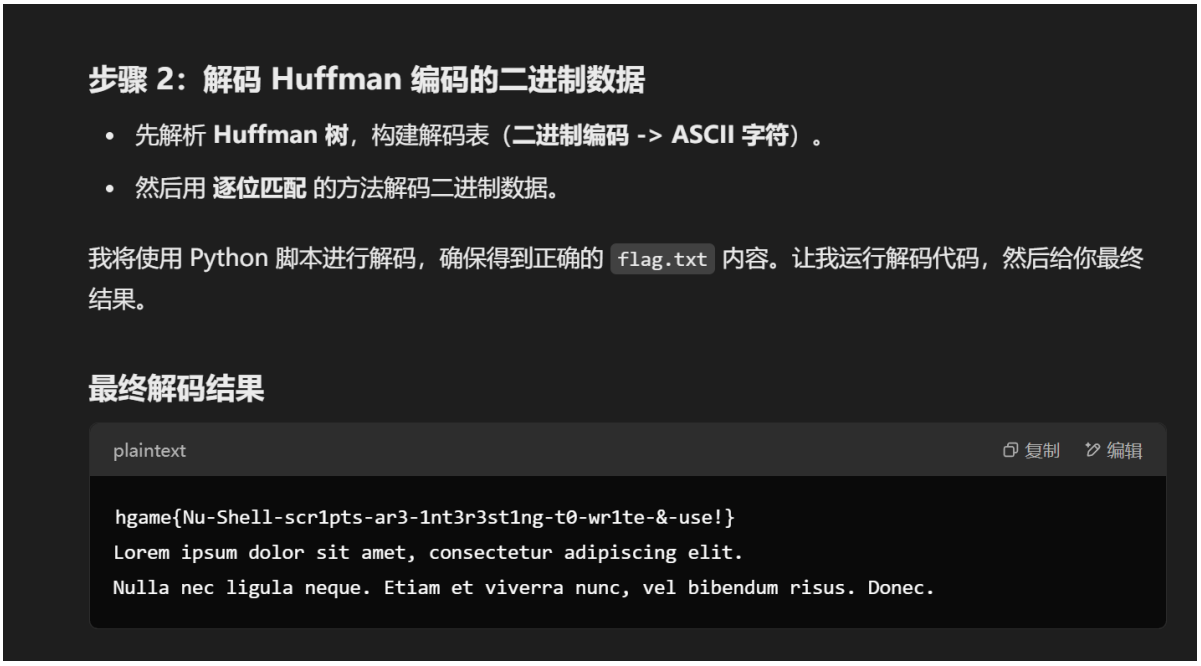
RE

Compress dot new

查一下脚本语言是nushell,直接让ai识别是什么算法,结果爆出flag了



步骤 2：解码 Huffman 编码的二进制数据

- 先解析 **Huffman 树**，构建解码表（**二进制编码 -> ASCII 字符**）。
- 然后用 **逐位匹配** 的方法解码二进制数据。

我将使用 Python 脚本进行解码，确保得到正确的 `flag.txt` 内容。让我运行解码代码，然后给你最终结果。

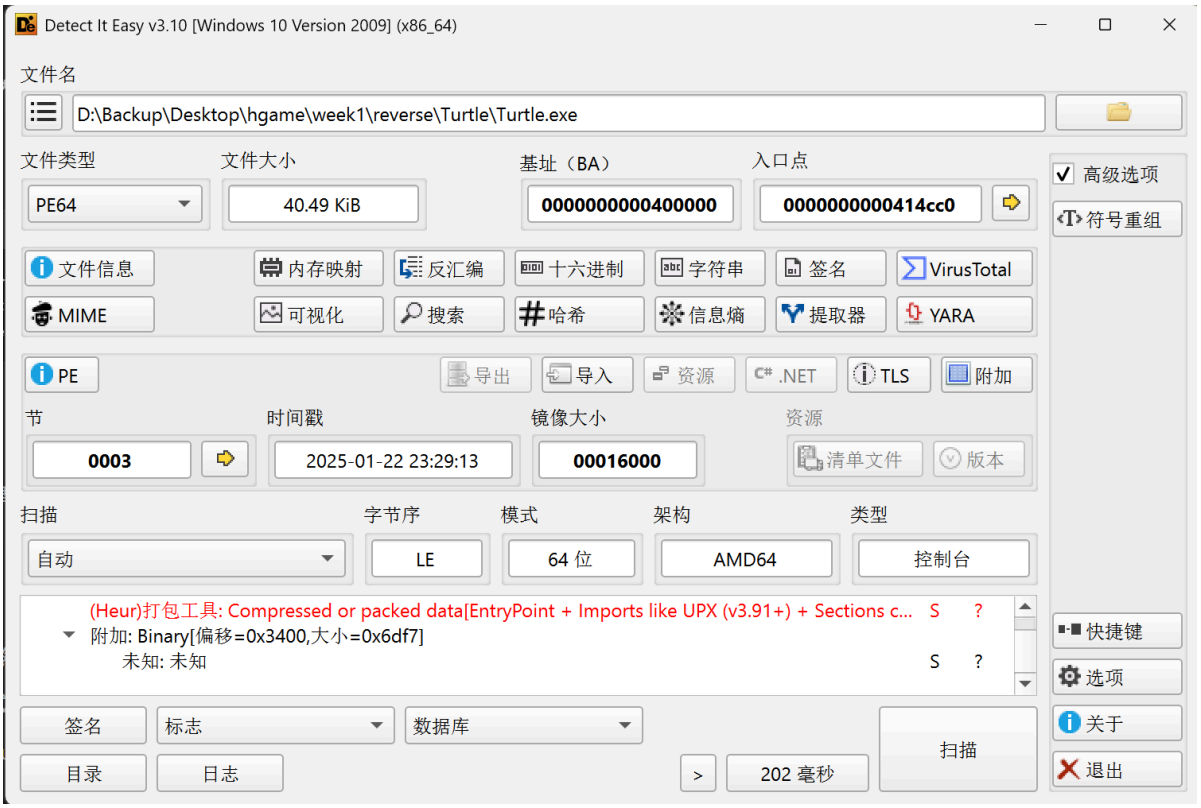
最终解码结果

plaintext 复制 编辑

```
hgame{Nu-Shell-scr1pts-ar3-1nt3r3st1ng-t0-wr1te-&-use!}
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nulla nec ligula neque. Etiam et viverra nunc, vel bibendum risus. Donec.
```

Turtle

开局用die查壳,魔改upx且魔改较多,用dbg脱壳(esp定律)



里面一看是个rc4,exp:

```
#include <stdio.h>
void aa(unsigned __int8 *a1, int a2, unsigned __int8 *a3)
{
    unsigned __int8 v3; // [rsp+3h] [rbp-Dh]
    int i;              // [rsp+4h] [rbp-Ch]
    int v5;             // [rsp+8h] [rbp-8h]
    int v6;             // [rsp+Ch] [rbp-4h]

    v6 = 0;
    v5 = 0;
    for (i = 0; i < a2; ++i)
    {
        v6 = (v6 + 1) % 256;
        v5 = (a3[v6] + v5) % 256;
        v3 = a3[v6];
        a3[v6] = a3[v5];
        a3[v5] = v3;
        a1[i] += a3[(unsigned __int8)(a3[v6] + a3[v5])];
    }
}
int main()
{
    unsigned char enc[] =
        {
```

```
            0xF8, 0xD5, 0x62, 0xCF, 0x43, 0xBA, 0xC2, 0x23, 0x15, 0x4A,
            0x51, 0x10, 0x27, 0x10, 0xB1, 0xCF, 0xC4, 0x09, 0xFE, 0xE3,
            0x9F, 0x49, 0x87, 0xEA, 0x59, 0xC2, 0x07, 0x3B, 0xA9, 0x11,
            0xC1, 0xBC, 0xFD, 0x4B, 0x57, 0xC4, 0x7E, 0xD0, 0xAA, 0x0A};
    unsigned char box[] =
        {
            0x65, 0xC9, 0xDC, 0x3A, 0xCE, 0x59, 0xC0, 0x24, 0x48, 0xA0,
            0x41, 0x62, 0x8F, 0x20, 0x26, 0xF8, 0x7C, 0xB4, 0xBA, 0x96,
            0xE0, 0x5A, 0x2C, 0x19, 0x9D, 0x22, 0x93, 0xE4, 0x10, 0xE5,
            0xC7, 0xBD, 0x3E, 0x76, 0xBE, 0xC6, 0x01, 0xFC, 0x86, 0x4F,
            0xDD, 0xD9, 0xD4, 0x83, 0xD3, 0x77, 0x63, 0x97, 0xFD, 0x4A,
            0xF7, 0xD5, 0xFA, 0x60, 0xF3, 0x6E, 0x32, 0x9E, 0x5C, 0x73,
            0x61, 0xB5, 0x40, 0xDF, 0xE8, 0xF6, 0x80, 0x28, 0xCA, 0x45,
            0xF0, 0xBC, 0xB8, 0xD7, 0x58, 0xCF, 0x9C, 0x69, 0x25, 0x52,
            0x15, 0xCC, 0x70, 0x07, 0x7E, 0x06, 0x2E, 0x54, 0x1A, 0x35,
            0x3B, 0x6F, 0x3C, 0x31, 0x7F, 0x1D, 0xF4, 0xE3, 0x82, 0xA7,
            0x37, 0xF9, 0x50, 0x6D, 0x13, 0x46, 0x8D, 0x95, 0xAB, 0xB7,
            0xAF, 0x72, 0xA8, 0xBB, 0x94, 0xAE, 0x5B, 0x67, 0xC1, 0xB3,
            0xA4, 0x1C, 0x8C, 0x36, 0x14, 0xC4, 0xA5, 0xB2, 0x8A, 0xB0,
            0x2D, 0x0B, 0x34, 0xCD, 0xA6, 0xFF, 0x21, 0x8B, 0xC8, 0x43,
            0x00, 0x09, 0xF1, 0xD0, 0xB6, 0x23, 0x53, 0x84, 0x57, 0x64,
            0xA2, 0x4B, 0x18, 0x0D, 0x5D, 0x78, 0x05, 0x02, 0x44, 0x92,
            0x29, 0x7D, 0xFE, 0x08, 0x8E, 0xC3, 0x90, 0xE2, 0x1E, 0xE6,
```

```
            0x81, 0x49, 0xE7, 0x6B, 0x12, 0x79, 0x0C, 0x33, 0xE1, 0x68,
            0x27, 0xD1, 0x99, 0x03, 0x5F, 0xD2, 0xED, 0x0E, 0xB9, 0xCB,
            0xEC, 0x4E, 0x56, 0x42, 0xDA, 0x87, 0xFB, 0x3D, 0xA1, 0x6A,
            0x3F, 0x89, 0x0F, 0x51, 0x9B, 0x1B, 0x7A, 0x88, 0xEE, 0x30,
            0x16, 0xEF, 0xC5, 0x9F, 0x74, 0x4C, 0xEB, 0x66, 0xB1, 0xDB,
            0x6C, 0xD8, 0x47, 0x4D, 0xA9, 0x7B, 0x71, 0x2F, 0x1F, 0xAA,
            0xD6, 0x2A, 0x2B, 0x91, 0x0A, 0x38, 0x85, 0xBF, 0xA3, 0x9A,
            0x75, 0x55, 0x11, 0x98, 0x17, 0xC2, 0xF5, 0x39, 0xF2, 0xE9,
            0xDE, 0x04, 0x5E, 0xEA, 0xAC, 0xAD};
    aa(enc, 40, box);
    printf("%s\n", enc);
    return 0;
}
```

得到结果`hgame{YOu'r3_re4l1y_g3t_OUt_of_th3_upX!}`

Delta Erro0000ors

```
qword_140005180 = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD))GetProcAddress(LibraryA, "ApplyDeltaB");
DeltaFree = (BOOL (__stdcall *)(LPVOID))GetProcAddress(hLibModule, "DeltaFree");
```

引入了一个windows的函数,查了下是增量补丁相关的函数,根据其他人的介绍可以得知两点:

1. 有hash校验
2. 可以在md5比较前获得目标文件

先运行,发现有一次换md5,伪代码没有,改为汇编一看,果然是异常处理.

```
0001327                 call    cs:puts
000132D                 call    cs:GetLastError
0001333                 mov     ecx, eax        ; dwExceptionCode
0001335                 xor     r9d, r9d        ; lpArguments
0001338                 xor     r8d, r8d        ; nNumberOfArguments
000133B                 mov     edx, 1          ; dwExceptionFlags
0001340                 call    cs:RaiseException
0001346
0001346 loc_140001346:                         ; CODE XREF: main+1DE↑j
0001346                 jmp     loc_140001494
0001346 ;   } // starts at 1400012C0
000134B ; ---------------------------------------------------------------------------
000134B
000134B loc_14000134B:                         ; DATA XREF: .rdata:0000000140003A2C
000134B ;   __except(1) // owned by 1400012C0
000134B                 lea     rcx, aSevenEatsTheHa ; "Seven eats the hash and caus
0001352                 call    cs:puts
0001358                 lea     rcx, aSevenWantsToMa ; "Seven wants to make up for t
000135F                 call    cs:puts
0001365                 lea     rcx, aInputYourMd5 ; "input your MD5:"
000136C                 call    sub_140001020
0001371                 lea     rdx, [rsp+158h+Buffer]
0001379                 lea     rcx, a32s       ; "%32s"
0001380                 call    sub_140001080
0001385                 lea     rbx, [rsp+158h+Buffer]
000138D                 lea     rdi, unk_1400050B4
0001394                 mov     esi, 10h
0001399                 nop     dword ptr [rax+00000000h]
00013A0
00013A0 loc_1400013A0:                         ; CODE XREF: main+27D↓j
00013A0                 mov     r8, rdi
00013A3                 lea     rdx, a02x       ; "%02x"
00013AA                 mov     rcx, rbx        ; Buffer
00013AD                 call    sub_1400010E0
00013B2                 inc     rdi
00013B5                 add     rbx, 2
```

然后直接强行建立函数看伪代码.

```
__int64 v16; // [rsp+1D0h] [rbp+78h]
__int64 v17; // [rsp+208h] [rbp+B0h] BYREF
__int64 v18; // [rsp+230h] [rbp+D8h]

puts("Seven eats the hash and causes the program to appear to have some kind of error.");
puts("Seven wants to make up for the mistake, so she's giving you a chance to patch the hash.");
sub_140001020("input your MD5:");
sub_140001080("%32s");
v3 = (char *)&v17;
v4 = (char *)&unk_1400050B4;
v5 = 16i64;
do
{
  sub_1400010E0(v3, "%02x");
  ++v4;
  v3 += 2;
  --v5;
}
while ( v5 );
word_1400050C4 = 31233;
v15 = v9;
v16 = v10;
v13 = v11;
v14 = v12;
if ( !qword_140005180(0i64, &v13, &v15, &lpMemory) )
{
  puts("You didn't take advantage of this opportunity.");
  sub_1400014E0();
  exit(0);
}
v6 = 0;
v7 = 0i64;
do
{
  if ( byte_140003438[v7] != (*((_BYTE *)&v18 + v7) ^ *((_BYTE *)lpMemory + v6 % (unsigned __int64)qword_140005198)) )
  {
    puts("Flag is error!!");
    sub_1400014E0();
    exit(0);
  }
  ++v6;
  ++v7;
```

既然给改md5的机会,那么第二个思路就比较好了.

在md5的地方下硬件断点跟踪dump出md5值然后输入

接下来写exp:

```
#include<stdio.h>
```

```
#include<string.h>
unsigned char ida_chars[] =
{
  0x53, 0x65, 0x76, 0x65, 0x6E, 0x20, 0x73, 0x61, 0x79, 0x73,
  0x20, 0x79, 0x6F, 0x75, 0x27, 0x72, 0x65, 0x20, 0x72, 0x69,
  0x67, 0x68, 0x74, 0x21, 0x21, 0x21, 0x21,0x00
};

unsigned char enc[] =
    {
        0x3B, 0x02, 0x17, 0x08, 0x0B, 0x5B, 0x4A, 0x52, 0x4D,
0x11,
        0x11, 0x4B, 0x5C, 0x43, 0x0A, 0x13, 0x54, 0x12, 0x46,
0x44,
        0x53, 0x59, 0x41, 0x11, 0x0C, 0x18, 0x17, 0x37, 0x30,
0x48,
        0x15, 0x07, 0x5A, 0x46, 0x15, 0x54, 0x1B, 0x10, 0x43,
0x40,
        0x5F, 0x45, 0x5A,0x00};

int main(){
    for (int i = 0; i < strlen(enc); i++){
        enc[i] ^= ida_chars[i%0x1c];
    }
    printf("%s", enc);
    return 0;
}
```

得flag:`hgame{934b1236-a124-4150-967c-cb4ff5bcc900}`

WEB

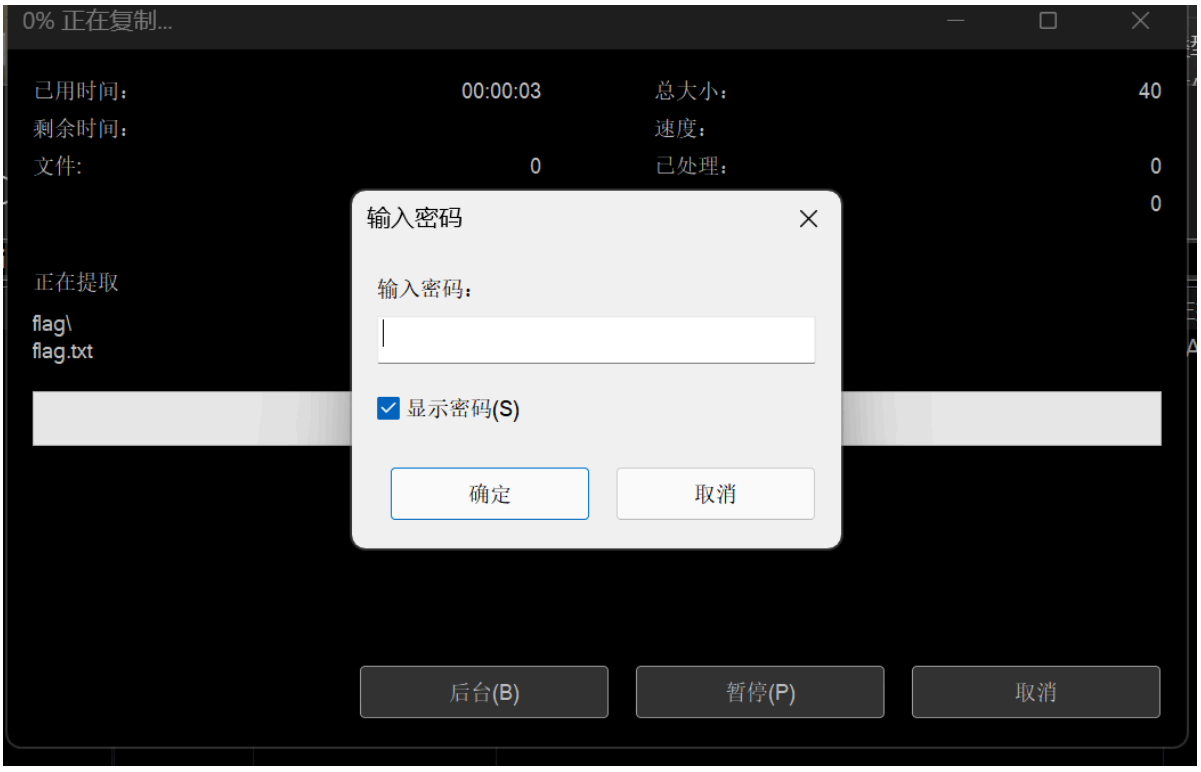
Level 24 Pacman

在index.js中搜索找到两个疑似是base64的文字,转化一下,顺序乱了说明是栅栏密码,获得flag.一个是假的,还被嘲讽了.

(打不开靶机了,所以就没有图片了)

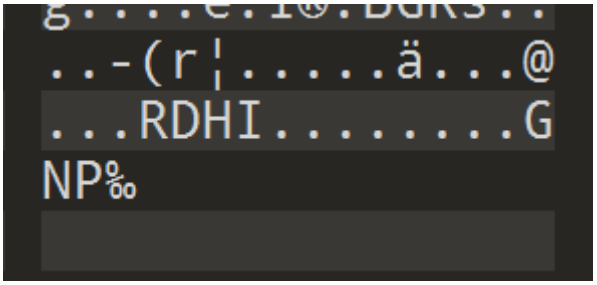
misc

拿到文件是二进制码转化成zip压缩包,提示密码:



看了加密方式AES,不可能破解.

看看文件,结尾有个GNP,把倒置的GNP图片dump出来并导致回来.




```python
with open('output.bin','rb') as f:
    with open('flag','wb') as g:
        g.write(f.read()[::-1])
```

获得学长的帅照,然后猜了下是高宽也没看crc用一把梭工具试了下,出了

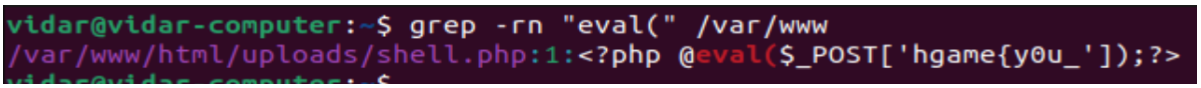
这是密码,输入zip可获得flag:`hagme{h4kyu4_w4nt_gir1f3nd_+q_931290928}`(难绷)

Computer cleaner

ai是好东西,先把题目要求告诉他,他给了我这个:

```
grep -rn "eval(" /var/www
```

获得第一部分



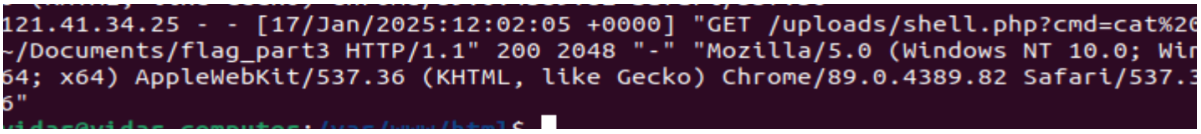
搜了下好像这个地方就是ubuntu服务器用来联网的地方,那就搜搜记录吧,遍历以后发现

`121.41.34.25`这个IP,原本以为要社工,后面发现直接访问即可(大道至简)

可以看到

```
Are you looking for me
Congratulations!!!
hav3_cleaned_th3
```

接着回到log:



这里执行了一个cmd命令去看看找到第三部分(但我是先找到才来看的,随便一翻就看到了)

`_cOmput3r!}`

flag:`hgame{y0u_hav3_cleaned_th3_cOmput3r!}`