

## 灵活奋斗

做累了题目就来小摆一会吧

扫码得到

一眼base32, 解得

emoji加密就那么几种，尝试后发现是base100，解得

一眼base64, 解得

看见现场很多选手被三层base折磨，出题人很愧疚，下次一定套更多（

# 英雄联盟

这是一张平平无奇的战绩图，刚入坑的\$2zz用寒冰打出了0-14的逆天战绩，但其中好像还藏着什么秘密。好的题目，爱来自瓷器，英雄联盟

小小分析一手，010 editor 或者 winhex 打开图片，会发现末尾附加了一个压缩包

LOL.png x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
6:B050h:	2D	6D	69	4B	5B	DA	D2	96	B6	B4	A5	2D	6D	79	CE	A5	-miK[ÚÒ-¶'¥-myÎ¥
6:B060h:	BD	18	6D	4B	5B	DA	D2	96	B6	B4	A5	2D	6D	69	4B	5B	½.mK[ÚÒ-¶'¥-miK[
6:B070h:	DA	D2	96	B6	3C	E7	F2	FF	01	8A	2C	8B	78	22	47	77	ÚÒ-¶<çòÿ.Š,«x"Gw
6:B080h:	B3	00	00	00	00	49	45	4E	44	AE	42	60	82	50	4B	03	³....IEND®B',PK.
6:B090h:	04	14	00	01	00	08	00	18	58	6F	55	3E	FA	F2	4C	32	.....XoU>úòL2
6:B0A0h:	01	00	00	1D	02	00	00	08	00	00	00	66	6C	61	67	2E	.....flag.
6:B0B0h:	74	78	74	7E	8B	A8	40	6F	1D	A7	F0	48	F7	95	79	95	txt~<"@o.ŠðH÷•y•
6:B0C0h:	F9	FD	6E	F1	E4	E5	81	FF	82	1F	69	0D	5E	10	B2	E4	ùýññää.ÿ,.i.^.^ä
6:B0D0h:	8D	37	CB	9B	C6	0D	4E	D7	FA	8A	56	18	06	DE	2E	76	.7Ë>Æ.N×úŠV..p.v
6:B0E0h:	81	83	25	0D	C8	76	94	1A	44	BD	89	20	B7	5C	AF	55	.f%.Èv".D½% .\¯U
6:B0F0h:	5A	65	2D	4E	21	6E	C2	39	C5	E7	EC	AE	3D	BB	F9	80	Ze-N!nÂ9Âçî@=>ù€
6:B100h:	A1	37	5B	B8	5C	80	FC	CF	19	D1	2D	D9	49	4E	F3	51	¡7[.€üİ.N-ÛINóQ
6:B110h:	49	A8	D9	82	2A	1F	46	74	7A	43	8E	54	A0	4C	83	74	I"U,*.FtzCŽT Lft
6:B120h:	FE	1A	83	48	2B	42	69	4D	2A	09	9E	21	BE	8B	A7	E2	p.fH+BiM*.Ž!¾«Šâ
6:B130h:	E3	01	87	95	02	F5	19	35	9B	12	2C	16	09	5A	73	60	ä.‡•.ð.5>...Zs`
6:B140h:	7F	85	DD	A0	70	5D	94	99	CC	E2	9E	BC	B7	5A	02	DB	...Ý p]"™İâž¼·Z.Û
6:B150h:	A5	FC	D2	15	F4	60	28	1B	83	21	C7	AE	50	54	D1	20	¥ÜÖ.ð`(.f!Ç@PTÑ
6:B160h:	62	26	34	C6	E4	3F	91	A9	F8	3B	A5	33	D8	5C	CF	2A	b&4Æä?'@ø;¥3Ø\İ*
6:B170h:	9D	BE	F5	7B	35	3C	AF	3C	28	64	AD	A7	C0	57	5A	66	.¾ð{5<¯<(d-ŠÂWzf
6:B180h:	D5	63	AD	0B	C4	01	95	2C	F7	15	87	05	36	B8	5E	14	Öc-.Ä.·,÷.‡.6.^.
6:B190h:	FB	50	D9	D9	21	0E	3B	36	0D	9D	A8	4F	FA	67	3D	D8	ûPUÛ!.;6..°Oúg=Ø
6:B1A0h:	52	6A	5A	67	A5	1D	6C	8C	42	63	F1	DB	35	D7	39	4C	RjZg¥.lÆBcñÜ5×9L
6:B1B0h:	A2	F2	CD	AF	4C	E9	56	46	31	F0	06	8C	0B	3E	36	91	çòİ`LéVF1ð.Æ.>6'
6:B1C0h:	C1	0D	43	66	82	2D	83	9E	0A	74	83	7C	F0	DF	02	09	Á.Cf,-fž.tf ðß..
6:B1D0h:	0E	12	40	1A	5F	BD	25	C0	8A	77	4C	6E	05	6C	93	72	..@.½ÅŠwLn.l"r
6:B1E0h:	E4	F0	EF	10	9F	50	4B	01	02	3F	00	14	00	01	00	08	äðİ.ŸPK..?.....
6:B1F0h:	00	18	58	6F	55	3E	FA	F2	4C	32	01	00	00	1D	02	00	..XoU>úòL2.....
6:B200h:	00	08	00	24	00	00	00	00	00	00	00	20	00	00	00	00	...\$......
6:B210h:	00	00	00	66	6C	61	67	2E	74	78	74	0A	00	20	00	00	...flag.txt... ..
6:B220h:	00	00	00	01	00	18	00	F3	F7	4B	75	9E	F8	D8	01	F3	.....ó÷KužøØ.ó
6:B230h:	F7	4B	75	9E	F8	D8	01	6E	BB	27	1F	AB	EE	D8	01	50	÷KužøØ.n»'«îø.P
6:B240h:	4B	05	06	00	00	00	01	00	01	00	5A	00	00	00	58		K.....Z...X
6:B250h:	01	00	00	00	00												.....

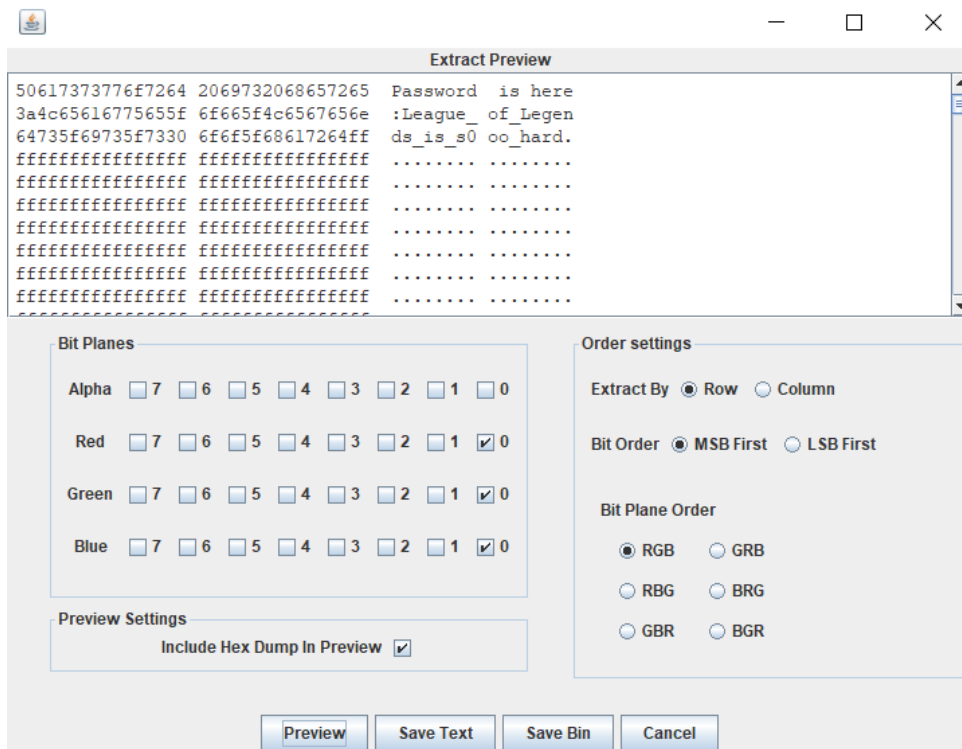
一般png图片结尾是 AE426082，而这里正常png结尾之后多了一串 504B0304 开头的内容，而 504B0304是zip压缩包的文件头

所以把后面这一部分提取出来，保存为zip压缩包

或者以上这些步骤，也可以在kali虚拟机里，用binwalk+foremost完成分离

总之，分离出zip压缩包之后，里面是flag.txt，但是需要密码

stegsolve查看图片RGB通道的最低位，发现密码 `League_of_Legends_is_s0oo_hard`



这一步也可以用kali虚拟机中的ztec来完成，原理都是一样的，就是查看图片各个通道的情况  
用密码解压之前的压缩包，得到一串奇怪的字符

```
HAI NUAACTF code
VISIBLE "HERE IS YOUR FLAG:)"
I HAS A CODE ITZ "LSBBDRGyJmjD1b4]2q]dsl6w{"
I HAS A MSG ITZ ""
I HAS A INDEX ITZ 0 BTW ITZ INDEX
I HAS A NUM ITZ
IM IN YR CYCLE UPPIN YR INDEX WILE INDEX SMALLR THAN LEN OF CODE
I HAS A C ITZ CODE!INDEX
NUM R ORD OF C
NUM R SUM OF NUM AN -3
I HAS A TEMP
NUM BIGGER THAN 69, O RLY?
YA RLY
NUM R SUM OF NUM AN 5
NO WAI
NUM R SUM OF NUM AN 2
OIC
TEMP R CHR OF NUM
OBTW NUM R CHR OF NUM
ITZ JUST A COMMENT
TLDR
MSG R SMOOSH MSG AN TEMP
IM OUTTA YR CYCLE
VISIBLE MSG
KTHXBYE
```

尝试搜索里面的某些语句，会发现这是 lolcode 语言，这是一种esolang（奇特的编程语言）

也会搜索到相似的ctf题 [第四届“安洵杯”网络安全挑战赛 CyzCC loves LOL](#)

找到解码网站 [LOLCODE Language - Compiler - Online Decoder, Encoder, Translator \(dcode.fr\)](https://dcode.fr) 运行即可



这里有些人可能需要代理才能上这个网站，有些人不需要代理直接能上，很玄学（

其实不用上这个解码网站也能解出题目，有一队学妹直接对着lolcode手搓出了对应的python脚本（tql

lolcode 是用网络上一些缩写流行语来完成编写的，比如BTW 是 by the way 的缩写，by the way 经常用于换个话题，相当于常说的“对了...”，在lolcode里就代表注释。

题目描述中“好的题目，爱来自瓷器，英雄联盟”也是玩了翻译梗，因为 lol 不仅有英雄联盟的意思，在网络流行语中还代表“笑”的意思，这里考点就是 lolcode。

所以 lolcode 可阅读性很高，可以直接阅读然后猜出里面的意思和逻辑。

出题人是直接仿照上面 **第四届“安洵杯”网络安全挑战赛 CyzCC\_loves\_LOL** 那题，稍微改动一下出的。但是原题写的和最正统的lolcode就有区别，很多语法官文档里没有，所以很多选手找到了 lolcode在线运行的网站，但是无法运行，必须得用上面的解密网站才能正确运行。

给出一个对应的python脚本，可以和 lolcode 对应一下，可以发现是真的简单易懂

```
print("HERE IS YOUR FLAG\n")
code = "LSBBDRGyJmjD1b4]2q]ds16w{"
msg = ""
index = 0
# 定义变量num
while(index < len(code)):
    c = code[index]
    num = ord(c)
    num = num + (-3)
    # 定义变量temp
    if(num > 69):
        num = num + 5
    else:
        num = num + 2
    temp = chr(num)
    msg = msg + temp

    index += 1

print(msg)
# HERE IS YOUR FLAG
# NUAACTF{Lo1C0d3_1s_fun5y}
```

description:

卢浮宫的深处是蒙娜丽莎寂静的目光

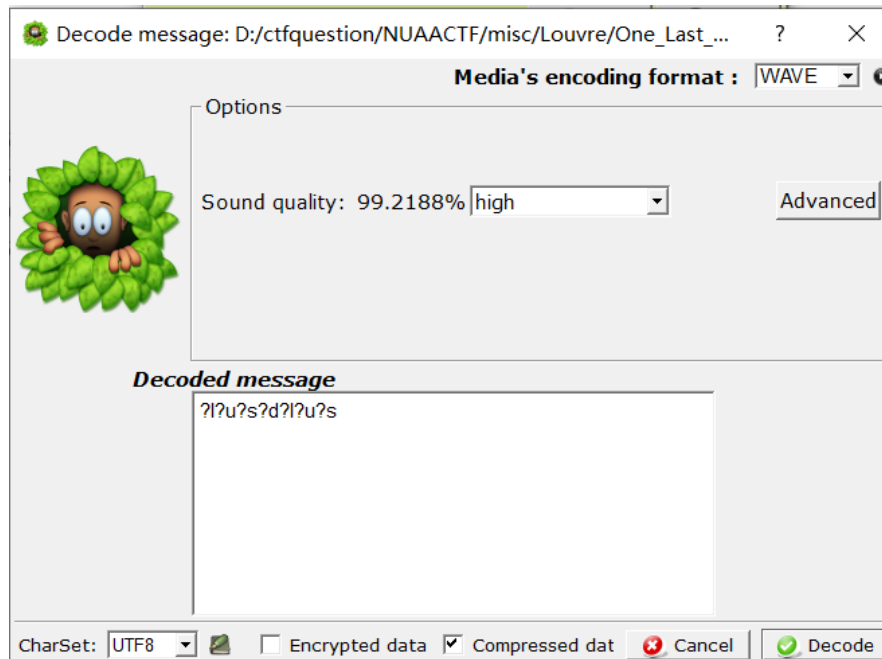
hint:

约翰的英文是?

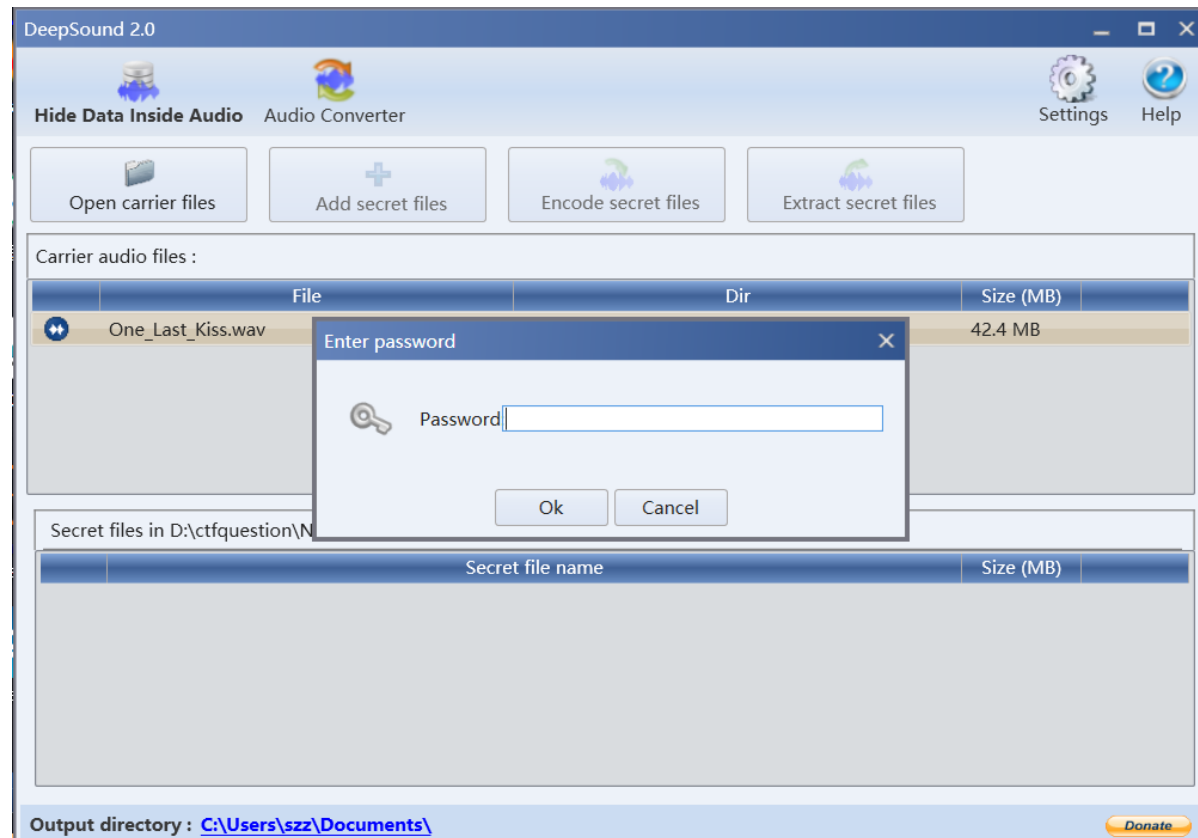
拿到一个One\_Last\_Kiss.mp3，但其实是个wav音频文件，小改一下后缀名

然后根据题目描述，很容易想到两个wav音频的隐写工具，silenteye 和 deepsound

先用silenteye解密，发现 `?l?u?s?d?l?u?s`



然后拖入deepsound发现要密码



那么用silenteye解出的那一串东西，肯定是和deepsound的密码有关

hint给出了john

搜索后发现了john爆破deepsound密码的操作，例题可参考 [INSHack2018] (not) so deep

那一串东西是掩码，因此是john掩码爆破deepsound密码

john是一种密码破解工具，kali上是自带的

然后如下操作，先用 [deepsound2john.py](#) 提取john要爆破的hash，然后直接掩码爆破

```
python3 deepsound2john.py One_Last_Kiss.wav > 1.txt
john --mask=?l?u?s?d?l?u?s 1.txt
```

```
(kali㉿kali)-[~/桌面/ctftools/john]
$ python3 deepsound2john.py One_Last_Kiss.wav > 1.txt

(kali㉿kali)-[~/桌面/ctftools/john]
$ cat 1.txt
One_Last_Kiss.wav:$dynamic_1529$220d65201c1c4712cd53b50b12176eaa75767b4d

(kali㉿kali)-[~/桌面/ctftools/john]
$ john --mask=?l?u?s?d?l?u?s 1.txt
Using default input encoding: UTF-8
Loaded 1 password hash (dynamic_1529 [sha1($p null_padded_to_len_32) (DeepSound) 128/128 AVX 4x1])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
oQ#7tG! (One_Last_Kiss.wav)
1g 0:00:00:36 DONE (2022-12-04 22:34) 0.02721g/s 11101Kp/s 11101Kc/s 11101Kc/s kB 7tG!..rQ#7tG!
Use the "--show --format=dynamic_1529" options to display all of the cracked passwords reliably
Session completed
```

得到密码 oQ#7tG!，deepsound解一下就行

这题本来出题人只是想简单地出一个silenteys + deepsound，silenteys解出来的就是deepsound的密码

但是后来发现南邮新生赛0xGame2022有一个相同的

所以引入了john爆破变化了一下，但是难度似乎太大了（

## ez\_Forensics

.raw文件，上内存取证

kali2020及之后不再自带volatility，可以自行安装

可以选择从源码安装，也可以选择直接下载release的二进制文件 [Volatility 2.6 Release](#) ([volatilityfoundation.org](#))

先查看imageinfo，得到 Win7SP1x64，是个win7系统

```
(kali㉿kali)-[~/桌面/ctfquestion]
$ vol.py -f mem.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000
7SP1x64_24000, Win7SP1x64_23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/kali/桌面/ctfquestion/mem.raw)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf800040020a0L
Number of Processors : 2
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xfffff80004003d00L
KPCR for CPU 1 : 0xfffff800009ef000L
KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2022-11-14 06:44:02 UTC+0000
Image local date and time : 2022-11-14 14:44:02 +0800
```



```
vol.py -f mem.raw --profile=win7SP1x64 filescan|grep "Desktop"
```

发现桌面上有一个flag.zip，注释里提供了5位掩码，爆破一下，得到

```
NUAACTF{this_is_fake_flag}
```

It's fake! I wouldn't put a secret in such an obvious place 🤔

I hid it somewhere on the Internet and no one can find it 😏

flag是假的，但是提示了 `hid it somewhere on the Internet`

```
vol.py -f mem.raw --profile=win7SP1x64 netscan
```

，查看网络连接状况

0x7ddcd010	TCPv6	:::49152	:::0	LISTENING	408	wininit.exe	
0x7eb47710	UDPv4	127.0.0.1:52790	::*		2688	ieexplore.exe	2022-11-14 06:43:45 UTC+0000
0x7eb9e010	UDPv4	0.0.0.0:3702	::*		1308	svchost.exe	2022-11-14 06:43:41 UTC+0000
0x7eb9e010	UDPv6	:::3702	::*		1308	svchost.exe	2022-11-14 06:43:41 UTC+0000
0x7ed7fe0	TCPv4	0.0.0.0:49152	0.0.0.0:0	LISTENING	408	wininit.exe	
0x7ea254d0	TCPv4	192.168.20.137:49160	23.203.70.208:80	ESTABLISHED	2688	ieexplore.exe	
0x7f067550	TCPv4	192.168.20.137:49163	42.192.224.58:8002	CLOSED	2652	ieexplore.exe	
0x7fc3e3a0	UDPv4	0.0.0.0:52310	::*		1060	svchost.exe	2022-11-14 06:43:55 UTC+0000
0x7fccb3a0	UDPv4	0.0.0.0:52310	::*		1060	svchost.exe	2022-11-14 06:43:55 UTC+0000
0x7fd583a0	UDPv4	0.0.0.0:52310	::*		1060	svchost.exe	2022-11-14 06:43:55 UTC+0000

发现 ieexplore.exe 浏览器曾经和 42.192.224.58:8002 建立过连接，很可疑

查看ie浏览器历史记录，抓取一下这个网址

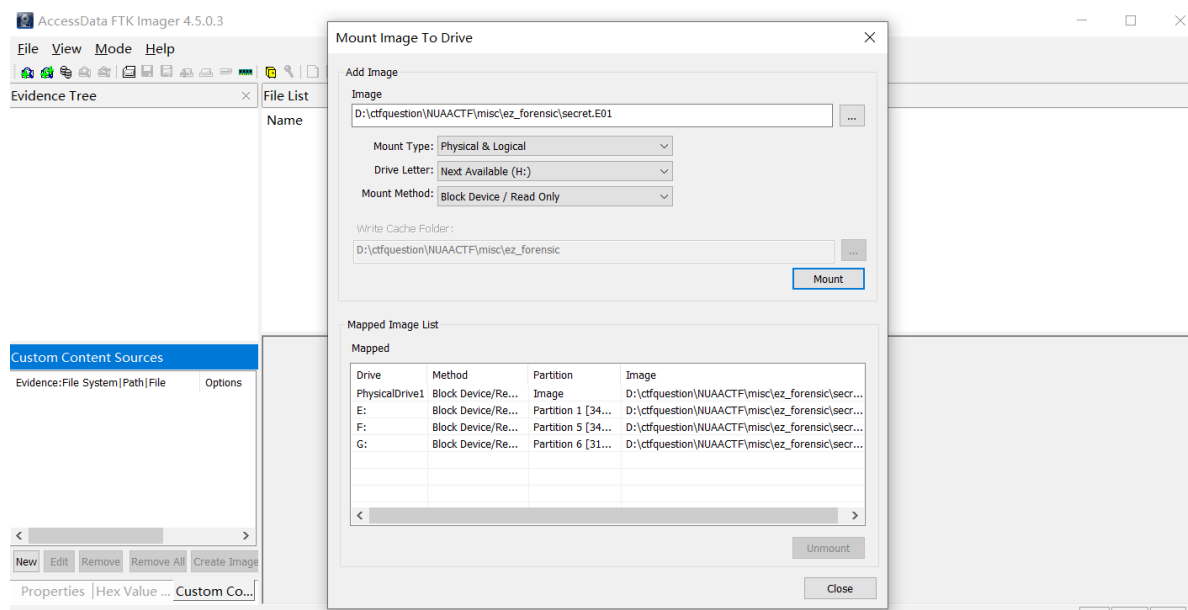
```
(kali㉿kali)-[~/桌面/ctfquestion]
└─$ vol.py -f mem.raw --profile=Win7SP1x64 iehistory | grep "42.192.224.58:8002"
Volatility Foundation Volatility Framework 2.6.1
URL: S2zz@http://42.192.224.58:8002
Location: Visited: S2zz@http://42.192.224.58:8002/favicon.ico
Location: Visited: S2zz@http://42.192.224.58:8002/favicon.ico
Location: Visited: S2zz@http://42.192.224.58:8002/real_target.zip
Location: Visited: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002/real_target.zip
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
Location: Visited: S2zz@http://42.192.224.58:8002/favicon.ico
Location: Visited: S2zz@http://42.192.224.58:8002/favicon.ico
Location: Visited: S2zz@http://42.192.224.58:8002/real_target.zip
Location: Visited: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002/real_target.zip
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
Location: Visited: S2zz@http://42.192.224.58:8002
Location: Visited: S2zz@http://42.192.224.58:8002/real_target.zip
Location: Visited: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
Location: :2022111420221115: S2zz@http://42.192.224.58:8002/real_target.zip
Location: :2022111420221115: S2zz@http://42.192.224.58:8002
```

发现从这个网站下载了一个real\_target.zip，那就访问一下



发现确实有一个real\_target.zip，下载解压后得到 secret.E01

接下来是磁盘取证分析，用FTK Imager挂载



挂载出来的盘里，可以得到一个real\_flag.zip

real_flag.zip - ZIP 压缩文件, 解包大小为 68 字节					
名称	大小	压缩后大小	类型	修改时间	CRC32
..			文件夹		
key1.txt *	5	17	文本文档	2022/10/30 1...	DEEC9B53
key2.txt *	6	18	文本文档	2022/10/30 1...	EC2E190E
key3.txt *	6	18	文本文档	2022/10/30 1...	1E71510D
real_flag.txt *	51	63	文本文档	2022/11/9 21:...	65C41AA0

key被分成了3部分，但是每个部分都不超过6字节大小，太小了，可以利用CRC32的值进行爆破

可以利用github上现有的脚本: [theonlypwner/crc32: CRC32 tools: reverse, undo/rewind, and calculate hashes \(github.com\)](https://github.com/theonlypwner/crc32)

比如跑key1.txt，得到第一段key1: it\_1s

```
(kali㉿kali)-[~/桌面/ctftools/crc32]
$ python crc32.py reverse 0xDEEC9B53
4 bytes: {0x05, 0x69, 0x5d, 0x95}
verification checksum: 0xdeec9b53 (OK)
5 bytes: it_1s (OK)
6 bytes: Dyc8Hi (OK)
6 bytes: EemURd (OK)
6 bytes: Fd86x8 (OK)
6 bytes: Jj3Tus (OK)
6 bytes: Kjrenj (OK)
6 bytes: Ms4j_X (OK)
6 bytes: TTGvUf (OK)
6 bytes: bsYRCJ (OK)
6 bytes: jy010b (OK)
6 bytes: sBsQDH (OK)
```

也可以自己手搓一个crc32爆破脚本（

三个跑完，得到完整密码 it\_1s\_safest\_p4ss，解密real\_flag.txt，得到flag



这题就是简单的内存取证和磁盘取证合在了一块，有点小坑但是并不是很难，可惜没有人做，出题人很伤心

## Blockchain

## Forever Love

description:

\$2zz的女朋友听说区块链上的数据永远不可能被篡改，缠着他在区块链上写了一个浪漫的表白话语，你能找到是什么吗 0x0823E02C66fff3A3C3b75DAa04199ba6Fc7b2CCd@Goerli

可以看看题目地址上交易的具体信息，可能秘密就藏在交易里

这题只需要一个浏览器就能做

区块链浏览器 [etherscan.io](https://etherscan.io) 能够查看链上的各种信息

直接去etherscan.io，切换 [goerli测试网](#)，搜索这个地址

Contract

0x023E02C66fff3A3C3b75DAa04199ba6Fc7b2CCd

Contract Overview

Balance:

0 Ether

More Info

My Name Tag:

Not Available

Contract Creator:

0xc6ccf770af762bb7ce3... at txn 0xf1e8e0b552c1d24a66...

Transactions

Erc20 Token Txns

Contract

Events

☰

Latest 3 from a total of 3 transactions

⋮

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x018f51cf95716f363ae...	Transfer	8070968	1 day 1 hr ago	0x3b5ba470a9e76d91cc...	IN	0x0823e02c66fff3a3c3b7...	0.001 Ether <div>0.00003151</div>
0x0609f8dc847b04773a...	0xb10b108b	7903303	28 days 16 hrs ago	0xc6ccf770af762bb7ce3...	IN	0x0823e02c66fff3a3c3b7...	0 Ether <div>0.00011333</div>
0xf1e8e0b552c1d24a66...	0x60806040	7903296	28 days 16 hrs ago	0xc6ccf770af762bb7ce3...	IN	Contract Creation	0 Ether <div>0.00058238</div>

[ Download ]
[CSV Export](#)

怎么还有人给这个地址转账 (

这是个合约地址，不是账户地址，转账显然会失败（

根据hint, 看每一笔交易的具体信息, 其中第二笔交易, 即0x0609f8... 那一笔交易

查看input data, 直接utf-8格式查看, 就能得到flag

⑦ Value:	0 Ether (\$0.00)
⑦ Transaction Fee:	0.000113334096105531 Ether (\$0.00)
⑦ Gas Price:	0.000000002502022123 Ether (2.502022123 Gwei)
⑦ Gas Limit & Usage by Txn:	45,297   45,297 (100%)
⑦ Gas Fees:	Base: 0.002022123 Gwei   Max: 2.50387693 Gwei   Max Priority: 2.5 Gwei
⑦ Burnt & Txn Savings Fees:	Burnt: 0.000000091596105531 Ether (\$0.00) Txn Savings: 0.000000084017192679 Ether (\$0.00)
⑦ Other Attributes:	Txn Type: 2 (EIP-1559)   Nonce: 4   Position In Block: 5
⑦ Input Data:	<div>±0000 0xJAACTF{N0w_It_is_F4nta2y_T1m3}</div> <div>View Input As ▾<ul style="list-style-type: none"><li>Default View</li><li>UTF-8</li><li>Original</li></ul></div>

[Click to see Less ↑](#)

A transaction is a cryptographically signed instruction from an account or smart contract to change the state of the blockchain. Block explorers track the details of all transactions in the network. Learn more about transactions in our [Knowledge Base](#).

NUAACTF{N0w\_It\_1s\_F4nta2y\_T1m3} 现在是，幻想时间！

如此简单清纯，居然只有三支队伍写了，我很伤心

## SecretNumber 1

description:

猜中这个神秘的数字 Goerli水龙头: <https://goerlifaucet.com/> 也可以找出题人领取测试币

nc 42.192.224.58 20000

hint:

怎么查看链上合约的公共变量呢？

这里稍微需要一点solidity的知识，一点remix交互的知识。我会写的很详细。

nc连接上之后，先查看源码

```
pragma solidity 0.8.7;

contract HiddenSecret1 {
    uint8 nonce;
    uint256[] public secrets;

    event isSolved();

    constructor(uint256 _secret) public {
        nonce = 0;
        secrets.push(_secret);
    }

    function guess(uint256 _answer) public {
        nonce = nonce + 1;
        require( nonce <= 3 );
        if( keccak256(abi.encodePacked(secrets[0])) ==
keccak256(abi.encodePacked(_answer)) ) {
            emit isSolved();
        }
    }
}
```

简单解释一下，constructor构造函数，把输入 \_secret 压入了secrets数组

guess函数，如果 \_answer 和 secrets[0] 相等，就会触发isSolved()事件，我们的目标就是触发这个事件

所以要得到 secrets[0] 这个值，而 secrets 是 public 属性，是能够直接交互获取的

先nc连接选择1，给deployer account转点钱，这样题目就能部署合约用于挑战（metamask钱包的操作就不教了，可以自己搜索

转账完成后选择2，会给出部署好的合约地址

```

(kali@kali)-[~]
$ nc 42.192.224.58 20000
I hide my secret number in the contract. I think it's very easy for you!
Your goal is to make isSolved() event emitted!

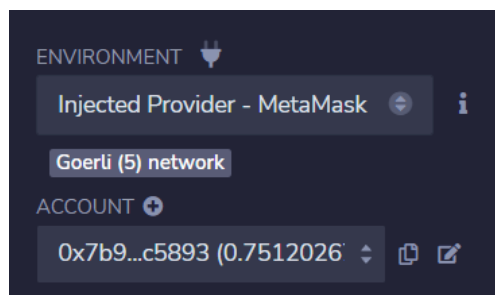
[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 1
[+] deployer account: 0xb28107C47433C20aFF2A6790De12770E7D3C3C3c
[+] token: v4.local.nbM5dLZYtdJKurG2EDLWp8S8UjYt3YEFVjxNZdsv57BdwQd396f9SpwDuxd07rxDVsoP1a3TubSqIUx9iejsjU_s-PxE1pi0hKxhs6rb_WUG
TuwBwgIR3iCNF1HR43R3IRjCFEteJysVNh1Wzz4KFwbokRQDJFb3r0YGFrd5Mu2zw
[+] please transfer 0.001 test ether to the deployer account for next step

(kali@kali)-[~]
$ nc 42.192.224.58 20000
I hide my secret number in the contract. I think it's very easy for you!
Your goal is to make isSolved() event emitted!

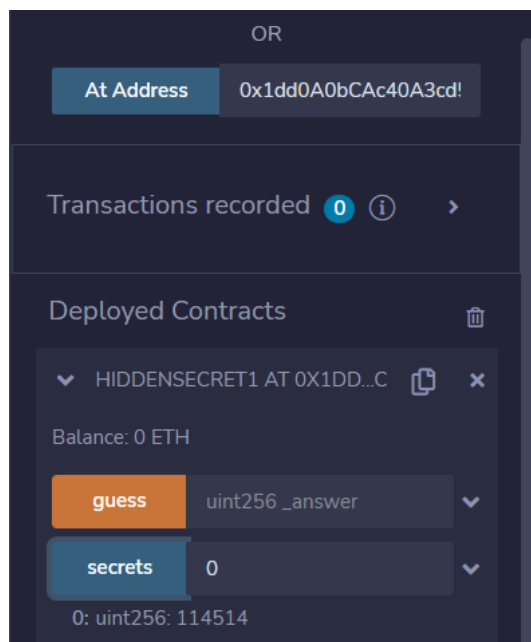
[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 2
[-] input your token: v4.local.nbM5dLZYtdJKurG2EDLWp8S8UjYt3YEFVjxNZdsv57BdwQd396f9SpwDuxd07rxDVsoP1a3TubSqIUx9iejsjU_s-PxE1pi0h
Kxhs6rb_WUGTuwBwgIR3iCNF1HR43R3IRjCFEteJysVNh1Wzz4KFwbokRQDJFb3r0YGFrd5Mu2zw
[+] contract address: 0x1dd0A0bCac40A3cd513C2571De97937d2BFcc082
[+] transaction hash: 0xeba563db9680df5f04d52d395b6782baf96acfab11bb538b424d2b4b152896d7

```

这时候用 [remix](#) 编译一下题目合约，然后使用 **At Address** 功能，直接和链上已部署的合约交互  
记得要选择测试网环境，从metamask钱包获取



然后直接交互，获取 secrets[0] 的值，是114514



然后guess，metamask会跳出交易确认，看到remix跳出这个说明已执行完毕

```

[✓] [block:8078497 txIndex:8] from: 0x7b9...c5893 to: 0x1dd0A0bCac40A3cd513C2571De97937d2BFcc082 0x1dd...Cc082 value: 0 wei data: 0x918...1bf52
logs: 0 hash: 0xdfd...8a12 Debug

status      true Transaction mined and execution succeed
transaction hash  0x795ed922a95447354a952ed943bac0de0ca1099e14bfd1f333c2751d11b5b2e5
from        0x7b9b459E82deFedf76261947DfeC18A6a7fc5893
to          0x1dd0A0bCac40A3cd513C2571De97937d2BFcc082

```

然后nc连接，选择3，把 transaction hash 发过去就能获取flag了

```

(kali@kali)-[~]
$ nc 42.192.224.58 20000
I hide my secret number in the contract. I think it's very easy for you!
Your goal is to make isSolved() event emitted!

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 3
[-] input your token: v4.local.nbm5dLZYtdJKurG2EDlWp8S8UjYT3YefLVjxNZdsv57BdwQd396f9SpwDuxd07rxDVsoP1a3TUbSqiUX9iejsjU_s-PxE1pi0h
kxhs6rb_WUGTuwBwgIR3iCNf1HR43R3IRjCFeteJysVNH1Wzz4KFwbokRQDJFb3r0YGfRD5Mu2zw
[-] input tx hash that emitted isSolved event: 0x795ed922a95447354a952ed943bac0de0ca1099e14bfd1f333c2751d11b5b2e5
[+] flag: NUAACtF{N3ver_Put_y0ur_s3crets_in_th3_public}

```

这里其实还有另一个比较取巧的方法，直接看合约创建的那一笔交易，会有constructor构造函数输入参数的信息



input data的最后是0x1bf52，就是114514

## SecretNumber 2

description:

众所周知，私有变量是不可以被外部读取的，所以藏在里面一定很安全吧

nc 42.192.224.58 20001

源码

```

pragma solidity 0.8.7;

contract HiddenSecret2 {
    uint8 nonce;
    uint256[] private secrets;

    event isSolved();

    constructor(uint256 _secret) public {
        nonce = 0;

        secrets.push(uint256(keccak256(abi.encodePacked(_secret+block.number))));
    }

    function guess(uint256 _answer) public {
        require(tx.origin != msg.sender);
        nonce = nonce + 1;
        require( nonce <= 3 );
        if( keccak256(abi.encodePacked(secrets[0])) ==
keccak256(abi.encodePacked(_answer)) ) {
            emit isSolved();
        }
    }
}

```

和上一题的区别就是，这里的secrets数组是private私有的，不能直接通过remix上交互来获取里面的值。

顺便还在constructor构造函数里把secrets[0]复杂化了一下，防止上一题的取巧做法出现（当然也是能做，但是比较费劲

还有guess里多了一个 `require(tx.origin != msg.sender);` 条件

主要是考察合约私有变量的读取，需要web3js或者web3py的一点知识，以及对以太坊变量存储规则的一些了解。

虽说是私有的，但是链上所有数据都是公开的，仍然是可以读取的。

solidity变量存储规则可以看看文档: [状态变量在储存中的布局 — Solidity中文文档](#)

数据的存储方式是从位置 0 开始连续放置在 存储槽slot 中，对于每个变量，根据其类型确定字节大小

假设 动态数组根据上述存储规则最终可确定某个位置 p，数组的元素会从 `keccak256(p)` 开始

比如这题，secrets变量在第1个，占用slot1

第0个变量是nonce，nonce只有1字节，虽然没有把slot0占满，但是下一个变量是数组，数组本身要占32字节来保存数组长度，因此secrets占用slot1

那么元素 secrets[i] 的位置在 `slot(keccak256(1)+i)` 处，知道了数据的位置就可以进行数据读取

先是和上一题一样，正常转钱等待部署，拿到合约地址

然后读取secrets[0]的位置，即 `keccak256(1)+0` 的存储槽

这里给出一个web3py读取数据的脚本

```
from web3 import web3

w3 = Web3(Web3.HTTPProvider('https://goerli.infura.io/v3/xxx'))
# infura提供公开以太坊和测试节点，可以利用infura提供的api访问以太坊，快速接入以太坊
# 可以去infura.io注册一个账号，xxx填你的API KEY

address = w3.toChecksumAddress("contract address")
slot = 1
sload = w3.solidityKeccak(['uint256'], [slot]).hex()
print(sload) # sload是secrets[0]的存储位置
s = w3.eth.getStorageAt(address, sload).hex()
print(s)     # s是secrets[0]的值
```

得到secrets[0]的值之后，进行guess调用

但是需要绕过 `require(tx.origin != msg.sender);`

这里很简单，tx.origin是整笔交易链的第一发起人，msg.sender是当前交易的发送者

可以编写一个合约，通过这个合约来调用题目合约的guess函数，这样tx.origin是你的账户地址，msg.sender是你编写的合约地址

```
pragma solidity 0.8.7;

contract HiddenSecret2 {
    //...the content of the HiddenSecret2 contract
}

contract Exp {
    HiddenSecret2 instance;
    constructor() {
        instance = HiddenSecret2(0xf181781980ceac49b129847c3531fc7922f1944e);
        //the address of the HiddenSecret2 contract
    }
}
```

```
}  
function hack(uint256 _answer) public {  
    instance.guess(_answer);  
}  
}
```

把Exp合约用remix部署，然后调用hack函数，输入参数是之前读取出来的secrets[0]的值

最后nc连接，提交 transaction hash，就能获取flag了