2018 CSAW

原创

Risker_GML 于 2018-10-14 21:00:42 发布 532 收藏
分类专栏： wp 文章标签： CTF Crypto Web Misc

欢迎关注我的新博客地址：http://mmmmmmlei.cn

之前9月份事情很多没有打CSAW，听说题质量不错就去做了做，不会的就去复现了一波，跟着师傅们的wp学到了不少东西。

Web

Ldab

这道题开始做的时候试了一波注入姿势，没有什么卵用，后来才知道考察的是 LDAP 的注入。

关于 LDAP 及注入的知识点，可以参考这几篇博客：

> https://www.jianshu.com/p/d94673be9ed0
>
> http://blog.51cto.com/407711169/1439623
>
> LDAP 注入常用payload

看这道题，flag 应该就藏在某一项中，试一下查出所有项：

```
*)(uid=*
```

没有 flag，应该是做了什么过滤，猜测后台查询语句可能是这样的：

```
(&(name=输入)(name!=flag))
```

在&的条件下，我们的 payload 带入，变为

```
(&(name=*)(uid=*)(name!=flag))
```

flag 依然无法查询到。

所以思路是闭合掉前面的 &，再构造一个查询语句，payload：

> *)(uid=*))(|(uid=*

代入查询语句变为：

> (&(name=*)(uid=*))(|(uid=*)(name!=flag))

这样过滤flag前面限制变成了"|"，查询到flag：

Employees	BrentBarlow	Barlow	Brent	BrentBarlow
Employees	MaiaMcneil	Mcneil	Maia	MaiaMcneil
Employees	QuinnHaney	Haney	Quinn	QuinnHaney
Employees	JenettePacheco	Pacheco	Jenette	JenettePacheco
Employees	flag{ld4p_inj3ction_i5_a_th1ng}	Man	Flag	fman

> flag{ld4p_inj3ction_i5_a_th1ng}

sso

题目描述：

```
Don't you love undocumented APIs
Be the admin you were always meant to be
http://web.chal.csaw.io:9000
Update chal description at: 4:38 to include solve details
Aesthetic update for chal at Sun 7:25 AM
```

查看源码：

```
<h1>Welcome to our SINGLE SIGN ON PAGE WITH FULL OAUTH2.0!</h1>
<a href="/protected">.</a>
<!--
Wish we had an automatic GET route for /authorize... well they'll just have to POST from their own clients I gue
ss
POST /oauth2/token
POST /oauth2/authorize form-data TODO: make a form for this route
--!>
```

应该是基于 OAuth2.0 协议的身份验证，去补一波知识...

关于OAuth2.0的知识，可以参考以下几篇博客：

阮一峰的博客

OAuth 2.0攻击方法

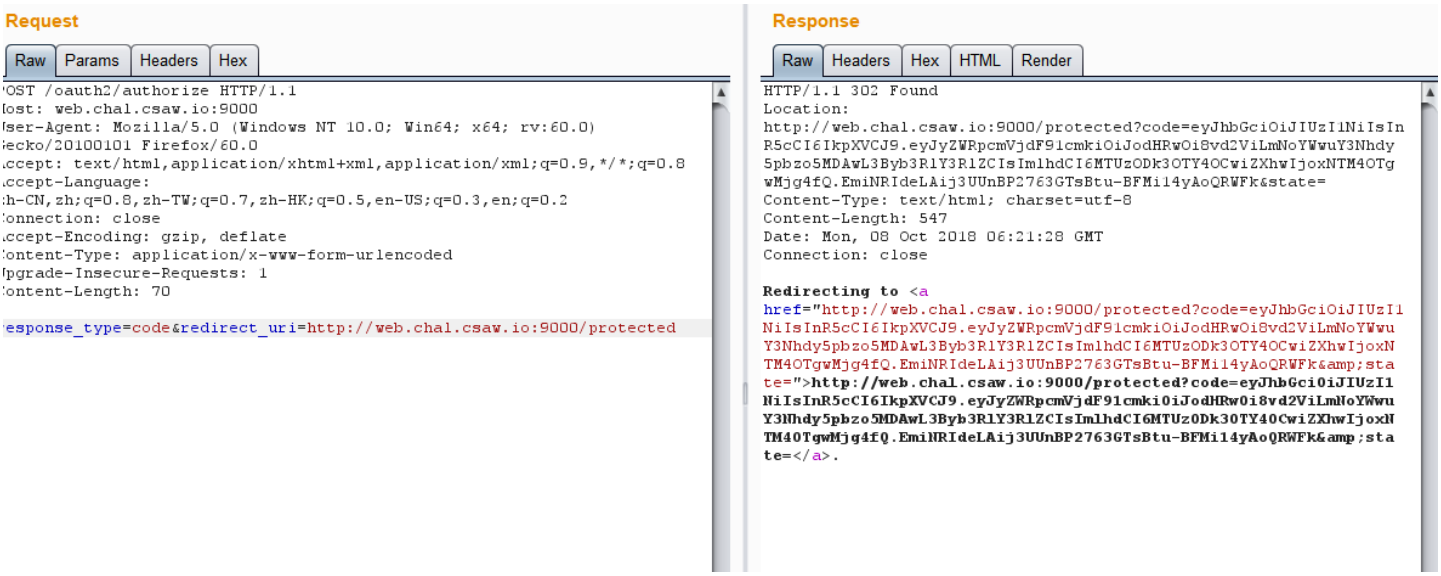
这道题涉及的验证模式是授权码模式。流程如下：

1. 用户访问客户端，后者将前者导向认证服务器。
2. 用户选择是否给予客户端授权。
3. 假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向 URI "（redirection URI），同时附上一个授权码。
4. 客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。

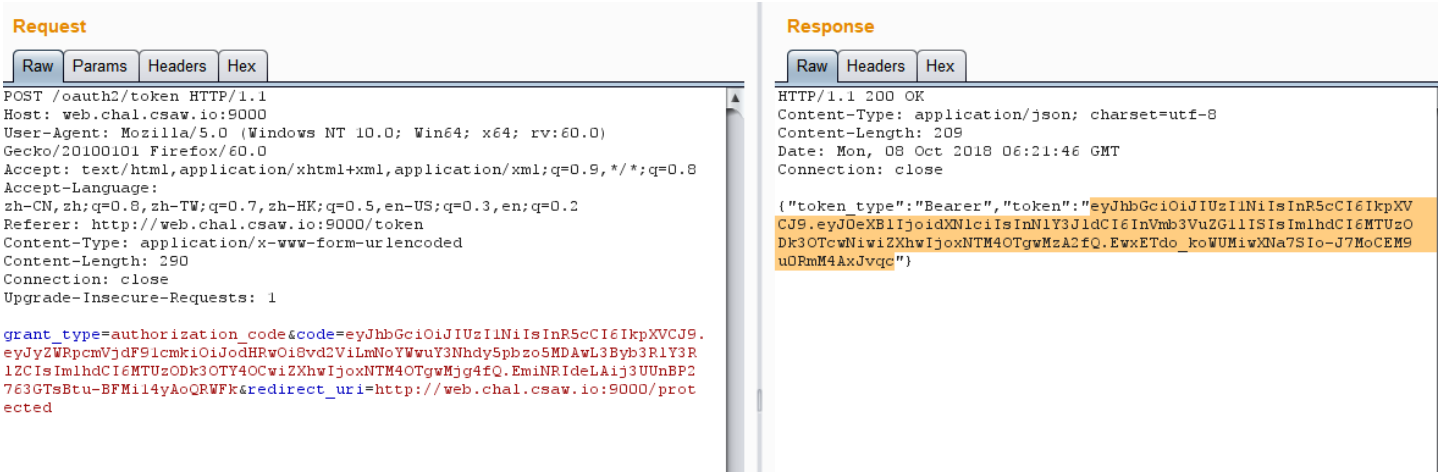
分析此题，我们要拿到 token 去访问 http://web.chal.csaw.io:9000/protected 得到flag，过程基本如下：

> 先去访问 /oauth2/authorize，构造 Post 请求拿到 code，带着这个code去访问 /oauth2/token，得到token去访问 /oauth2/protected 去找 flag。

首先访问 /oauth2/authorize，注意要在 Post 请求中加 “response_type=code”。redirect_uri 是重定向的url，与后面填的一致就可以。



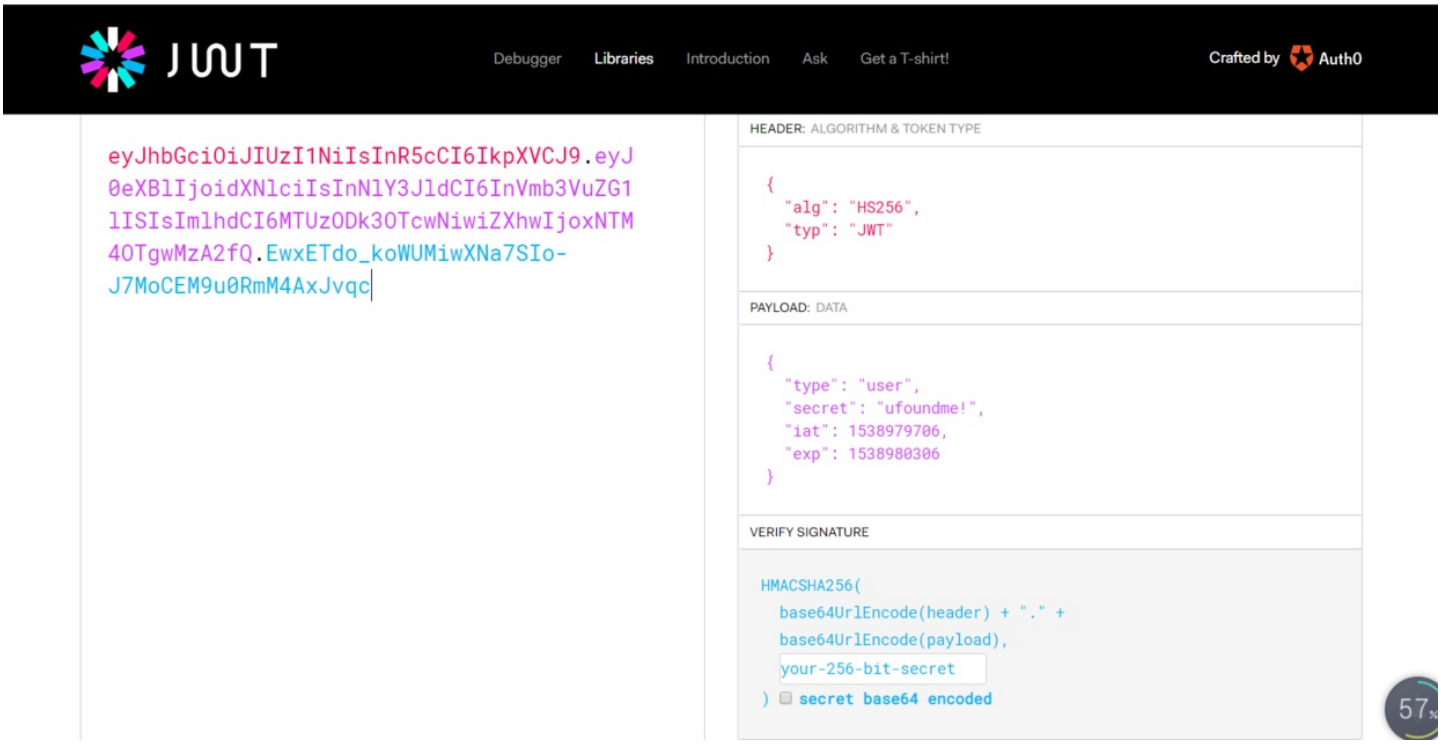
拿着这个 code 去访问 /oauth2/token，注意 Post 参数的格式：



payload:

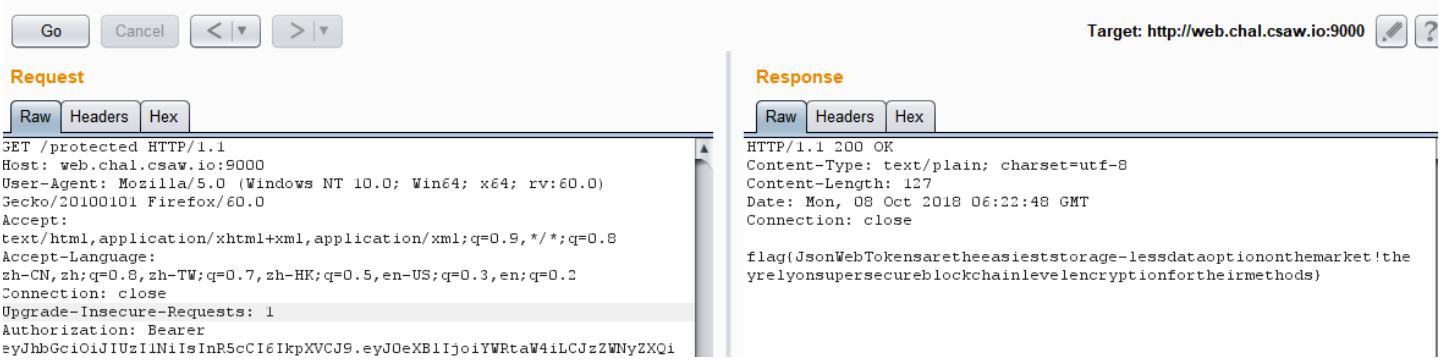
```
grant_type=authorization_code&code=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZWRpcmVjdF91cmkiOiJodHRwOi8vd2ViLmNoYWwuY3Nhdy5pbzo5MDAwL3Byb3R1Y3R1ZCIsImlhdCI6MTUzODk3OTY4OCwiZXhwIjoxNTM4OTgwMjg4fQ.EmiNRIdeLAij3UUnBP2763GTsBtu-BFMi14yAoQRWFk&redirect_uri=http://web.chal.csaw.io:9000/protected
```

得到 token，进行 jwt 在线解码:



这里的 ufoundme! 就是加密的密钥，根据题目提示把 user 改为 admin ,下面密钥换成 ufoundme!

拿着这个新的 token 去访问 http://web.chal.csaw.io:9000/protected 得到flag:




```
flag: flag{JsonWebTokensaretheeasieststorage-lessdataoptiononthemarket!theyrelyonsupersecureblockchainlevelencry
ptionfortheirmethods}
```

参考：

https://www.secpulse.com/archives/75785.html

https://delcoding.github.io/2018/09/csaw-writeup2/

Crypto

babycrypto

题目描述：

> yeeeeeeeeeeeeeeeeeeeeeeeeeeeet
> single yeet yeeted with single yeet == 0
> yeeet
> what is yeet?
> yeet is yeet
> Yeetdate: yeeted yeet at yeet: 9:42 pm

密文给了一串base64:

```
s5qQkd+WjN+e34+NkJiNnpKSmo3fiJeQ356Mj5aNmozfi5DfnI2anoua34+NkJiNnpKM34uXnovfl5qTj9+PmpCPk5rfm5Dfk5qMjNHft5rfiJ6R
i4zfi5Dfj4qL356Ki5CSnouWkJHfmZaNjIvT356Rm9+MnJ6Tnp2Wk5aLht+ek5CRmIyWm5rR37ea35uNmp6SjN+Qmd+e34iQjZOb34iXmo2a34uX
mt+akZuTmoyM356Rm9+Ll5rflpGZlpGWi5rfnZqckJKa342anpOWi5aajN+LkN+SnpGUlpGb09+ekZvfiJeajZrfi5ea34uNiprfiZ6TiprfkJnf
k5aZmt+WjN+PjZqMmo2JmpvRmZOemISblpmZlprSl5qTk5KekdKYz4+XzI2FjZ6wps61npPLnLeeuabGrKithr6uyZ63gg==
```

密文解码是乱码

说了这么多，yeet 到底是什么，通过第二句猜测可能是异或(自己和自己 yeet 为0)

逐字节异或 只有 256 种情况，写出爆破脚本：

```
import base64
ciphertext="s5qQkd+WjN+e34+NkJiNnpKSmo3fiJeQ356Mj5aNmozfi5DfnI2anoua34+NkJiNnpKM34uXnovfl5qTj9+PmpCPk5rfm5Dfk5qM
jNHft5rfiJ6Ri4zfi5Dfj4qL356Ki5CSnouWkJHfmZaNjIvT356Rm9+MnJ6Tnp2Wk5aLht+ek5CRmIyWm5rR37ea35uNmp6SjN+Qmd+e34iQjZOb
34iXmo2a34uXmt+akZuTmoyM356Rm9+Ll5rflpGZlpGWi5rfnZqckJKa342anpOWi5aajN+LkN+SnpGUlpGb09+ekZvfiJeajZrfi5ea34uNiprf
iZ6TiprfkJnfk5aZmt+WjN+PjZqMmo2JmpvRmZOemISblpmZlprSl5qTk5KekdKYz4+XzI2FjZ6wps61npPLnLeeuabGrKithr6uyZ63gg=="
cipher=base64.b64decode(ciphertext)
for i in range(0,256):
    result=""
    for s in cipher:
        result+=chr(ord(s)^i)
    if "flag" in result:
        print result
```

> flag: flag{diffie-hellman-g0ph3rzraOY1Jal4cHaFY9SWRyAQ6aH}

lowe

> I personally prefer Home Depot
> XOR Passes are the easiest way to use numbers to encrypt!
> By Kris Kwiatkowski, Cloudflare

附件有三个：

- file.enc
- key.enc
- pubkey.pem

file.enc 是一串 base64 编码，解码是一串乱码，看起来像是加密过。

> kStoynmN5LSniue0nDxli9csSrBgexZ/YOo5e+MUkfJKwvht8hHsYyMGVYzMlOp9sAFBrPCbm4UA4n7oMr2zlg==

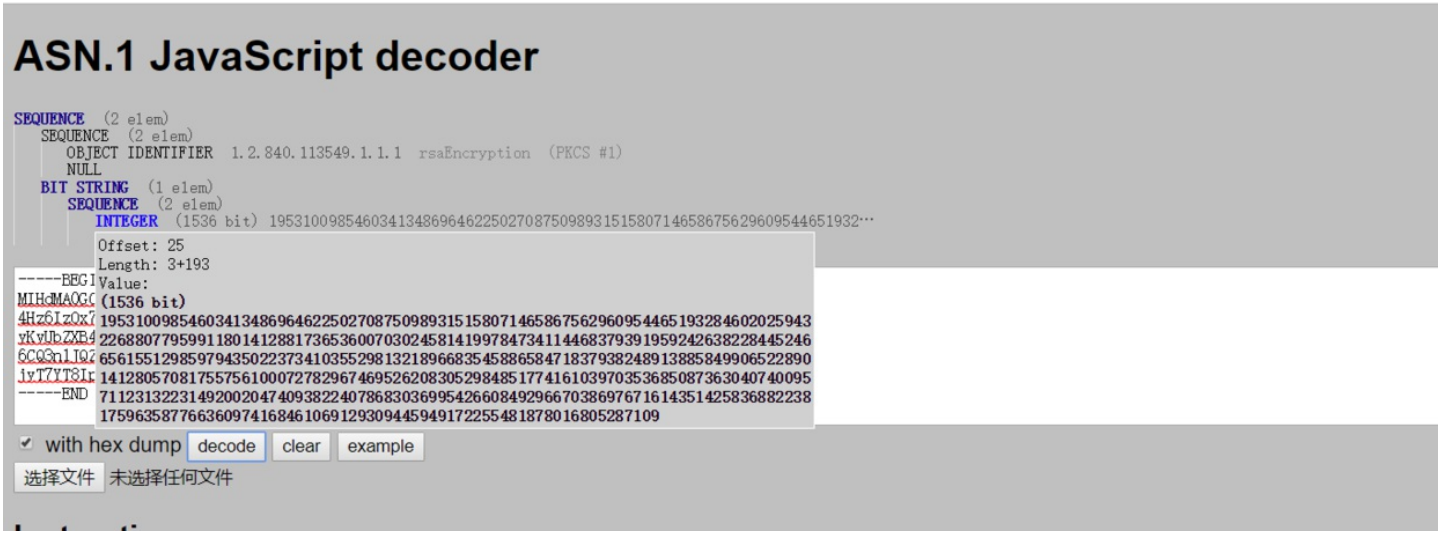
key.enc是 1533bit 的一个数。

```
21913599310960777800120184508415060222737614108219565784476266250808448108998605604853213376779260047012344460579568326804728134747449940967966078337062765256314425828464847480738161169413831435208742927112894278644560746231105244201561855835250650258684366009747174837219604826994258859772262396740274927966291344230398348043592674987944016723619770561365763102292049090691179042544319178164674454256222182931950931940442079514653286139333431038551783838407751 82
```

pubkey.pem 是RSA公钥文件。

```
-----BEGIN PUBLIC KEY-----
MIHdMA0GCSqGSIb3DQEBAQUAA4HLADCBxwKBwQDPcH7tl5AXt/b0dv87plVZrbGC
4Hz6IzOx7AVrf3uWEkBU8fV0iwTDaU6Q8Nmf7gWEqHpwgXWA1JOTMhuyCAf/3iWk
yKvUbZXB43QNnmQf53+bls7K6RjmeiSJUrXaga53Qr2uUbEpJFlzQVBXrnXft1p4
6CQ3nlJQZZLDdQ6aHH5wG+6N38epynJTTNOwlXn4ek6zdvkmfNGhbh5XkJXFuG9L
jyT7YT8Ip+DkddJVVq5ByM7iSOkNrJZdxH3btMUCAQM=
-----END PUBLIC KEY-----
```

首先在线提取出 RSA 公钥的n和e (选择 openssl 提取也可以)。发现 n 是1536bit，而 e 仅仅为 3。



分析以上条件，不难得出 key.enc 是 RSA 加密的 c，位数相近。 此题 e 这么小，猜想可能直接开三次方开出来，或者 **c + i* n **能够直接开三次方，其中 i 穷举。

试了一下发现成功，得到 RSA 明文 m，根据题目提示和 file.enc 进行异或，得到 flag，解题脚本如下：

```
import gmpy2
import base64
import libnum
n=19531009854603413486964622502708750989315158071465867562960954465193284602025943226880779599118014128817365360
07030245814199784734114468379391959242638228445246656155129859794350223734103552981321896683545886584718379382 48
91388584990652289014128057081755756100072782967469526208305298485177416103970353685087363040740095711231322314 92
00204740938224078683036995426608492966703869767161435142583688223817596358776636097416846106912930944594917225 54
81878016805287109
e=3
key=21913599310960777800120184508415060222737614108219565784476266250808448108998605604853213376779260047012344 4
60579568326804728134747449940967966078337062765256314425828464847480738161169413831435208742927112894278644560 74
62311052442015618558352506502586843660097471748372196048269942588597722623967402749279662913442303983480435926 74
98794401672361977056136576310229204909069117904254431917816467445425622218293195093194044207951465328613933343 10
385517838840775182
data=base64.b64decode(open("file.enc","rb").read())
data=int(data.encode("hex"),16)
for i in range(10000):
    if gmpy2.iroot(key+i*n,e)[1]:
        m=gmpy2.iroot(key+i*n,e)[0]
plain=libnum.n2s(m^data)
print plain
```

> flag: flag{saltstacksaltcomit5dd304276ba5745ec21fc1e6686a0b28da29e6fc}

Misc

bin_t

题目描述：

```
Binary trees let you do some interesting things. Can you balance a tree?
nc misc.chal.csaw.io 9001
Equal nodes should be inserted to the right of the parent node. You should balance the tree as you add nodes.
```

ACM入门级，AVL树的生成和树的前序遍历，贴上python代码：

```
#coding:utf-8
class TreeNode(object):
    def __init__(self):
        self.data=0
        self.left=None
        self.right=None
        self.height=0
class BTree(object):
    def __init__(self):
        self.root=None
        self.result=""
    def __Max(self,h1,h2):
        if h1>h2:
            return h1
        elif h1<=h2:
            return h2
    def __LL(self,r):#左左情况，向右旋转
        node=r.left
        r.left=node.right
        node.right=r
        r.height=self.__Max(self.getHeight(r.right),self.getHeight(r.left))+1
        node.height=self.__Max(self.getHeight(node.right),self.getHeight(node.left))+1
        return node
    def __RR(self,r):#右右，左旋
        node = r.right
        r.right = node.left
        node.left = r
        r.height = self.__Max(self.getHeight(r.right), self.getHeight(r.left)) + 1
        node.height = self.__Max(self.getHeight(node.right), self.getHeight(node.left)) + 1
        return node
    def __LR(self,r):#左右，先左旋再右旋
        r.left=self.__RR(r.left)
        return self.__LL(r)
    def __RL(self,r):#右左，先右旋再左旋
        r.right=self.__LL(r.right)
        return self.__RR(r)
    def __insert(self,data,r):
        if r==None:
            node=TreeNode()
            node.data=data
            return node
        elif data==r.data:
            return r
        elif data<r.data:
            r.left=self.__insert(data,r.left)
            if self.getHeight(r.left)-self.getHeight(r.right)>=2:
                if data<r.left.data:
                    r=self.__LL(r)
                else:
                    r=self.__LR(r)
        else:
            r.right=self.__insert(data,r.right)
            if self.getHeight(r.right)-self.getHeight(r.left)>=2:
                if data>r.right.data:
                    r=self.__RR(r)
                else:
                    r=self.__RL(r)
        r.height=self.__Max(self.getHeight(r.left),self.getHeight(r.right))+1
```

```
        return r
    # 删除data节点
    def __delete(self,data,r):
        if r==None:
            print "don't have %d"%data
            return r
        elif r.data==data:
            if r.left==None:#如果只有右子树，直接将右子树赋值到此节点
                return r.right
            elif r.right==None:#如果只有左子树，直接将左子树赋值到此节点
                return r.left
            else:#如果同时有左右子树
                if self.getHeight(r.left)>self.getHeight(r.right):#左子树高度大于右子树
                    #找到最右节点 返回节点值 并删除该节点
                    node=r.left
                    while(node.right!=None):
                        node=node.right
                    r=self.__delete(node.data,r)
                    r.data=node.data
                    return r
                else:#右子树高度大于左子树
                    node=r.right
                    while node.left!=None:
                        node=node.left
                    r=self.__delete(node.data,r)
                    r.data=node.data
                    return r
        elif data<r.data:
            r.left=self.__delete(data,r.left)#在左子树中删除
            if self.getHeight(r.right)-self.getHeight(r.left)>=2:#右子树高度与左子树高度相差超过1
                if self.getHeight(r.right.left)>self.getHeight(r.right.right):
                    r=self.__RL(r)
                else:
                    r=self.__RR(r)
        elif data>r.data:
            r.right=self.__delete(data,r.right)#右子树中删除
            if self.getHeight(r.left)-self.getHeight(r.right)>=2:#左子树与右子树高度相差超过1
                if self.getHeight(r.left.right)>self.getHeight(r.left.left):
                    r=self.__LR(r)
                else:
                    r=self.__LL(r)
        r.height=self.__Max(self.getHeight(r.left),self.getHeight(r.right))+1
        return r
    #先序遍历
    def __show(self,root):
        if root!=None:
            self.result+= str(root.data)+","
            self.__show(root.left)
            self.__show(root.right)
        else:
            return 0
    def Insert(self,data):
        self.root=self.__insert(data,self.root)
        return self.root
    def Delete(self,data):
        self.root=self.__delete(data,self.root)
    #求结点的高度
    def getHeight(self,node):
        if node==None:
```

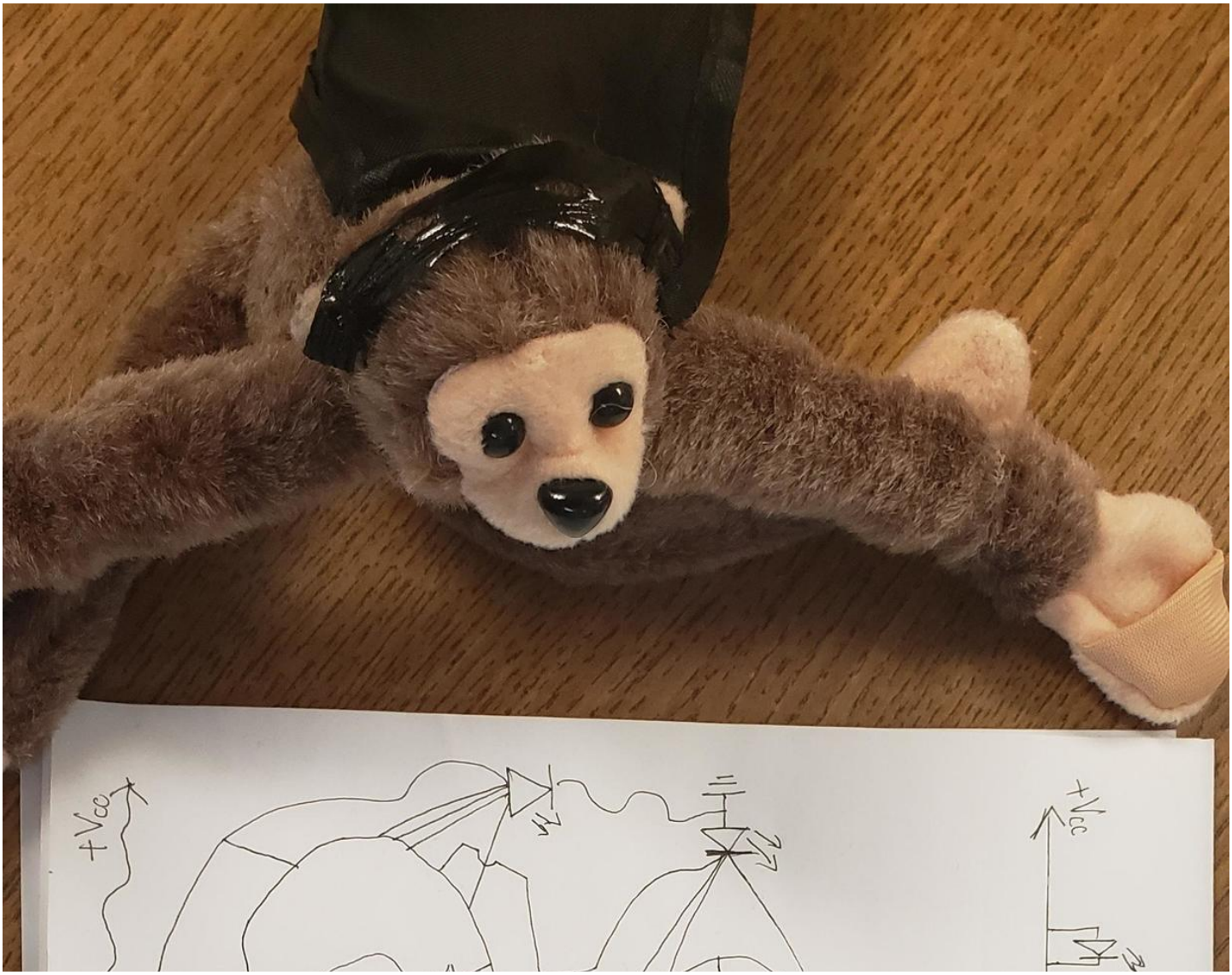
```
            return -1
        #print node
        return node.height
    def Show(self):
        self.__show(self.root)
if __name__=='__main__':
    bi=BTree()
    array=[40,84,21,16,50,28,32,10,46,55,22,4,50,76,45,56,42,6,89,99,21,16,72,47,96,29,46,97,10,58,63,83,40,65,2
4,65,72,100,11,52,57,77,49,61,5,85,73,49,98,90,81,39,12,73,80,19,69,78,27,25,84,92,79,98,21,8,48,2,34,49,35,21,2
4,86,10,60,39,28,84,80,61,6,73,67,38,78,70,12,3,94,46,10,91,12,14,69,1,81,56,21]
    for i in array:
        bi.Insert(i)
    bi.Show()
    print bi.result
```

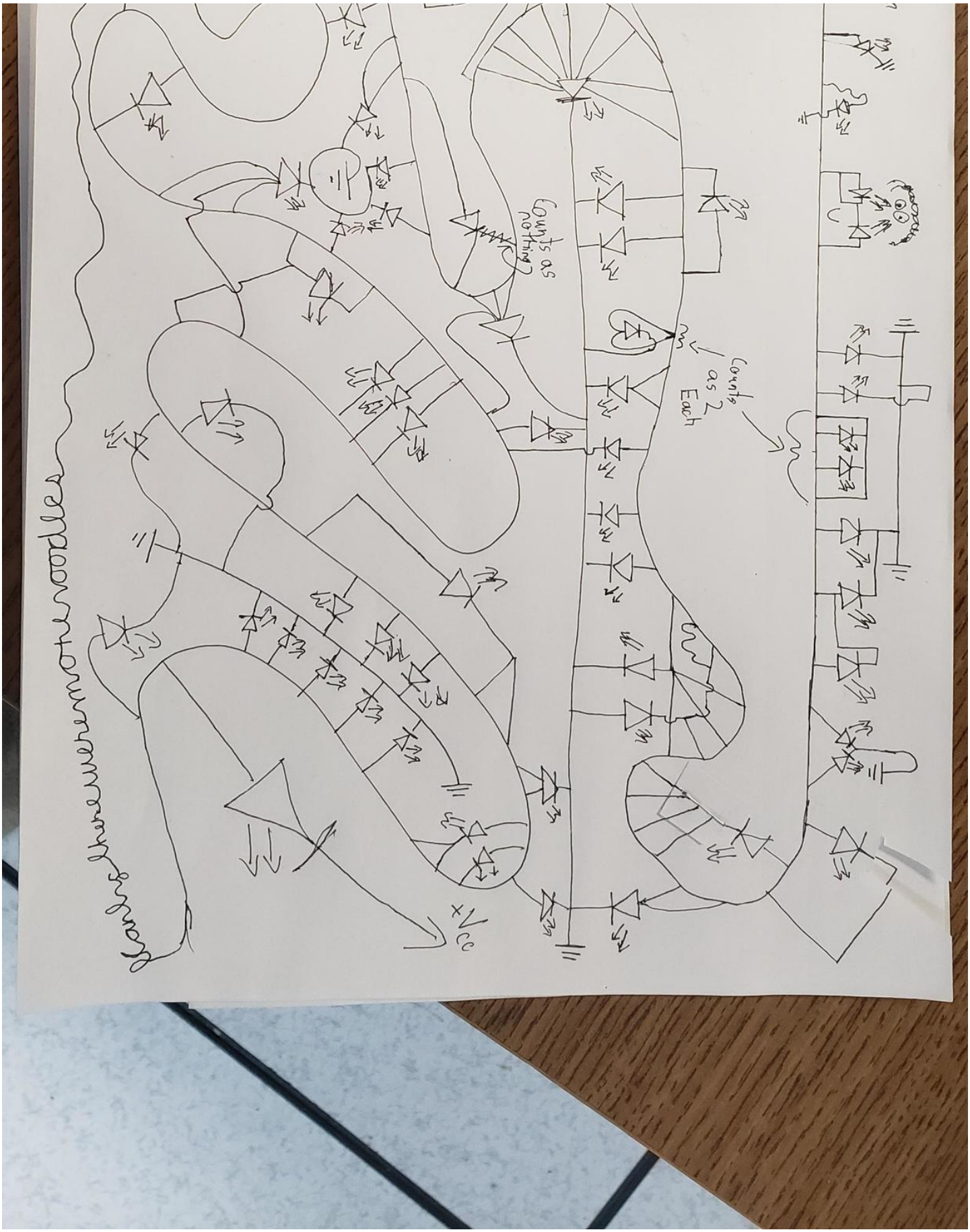
> flag: flag{HOW_WAS_IT_NAVIGATING_THAT_FOREST?}

Short Circuit

> Start from the monkey's paw and work your way down the high voltage line, for every wire that is branches off has an element that is either on or off. Ignore the first bit. Standard flag format.

图片：

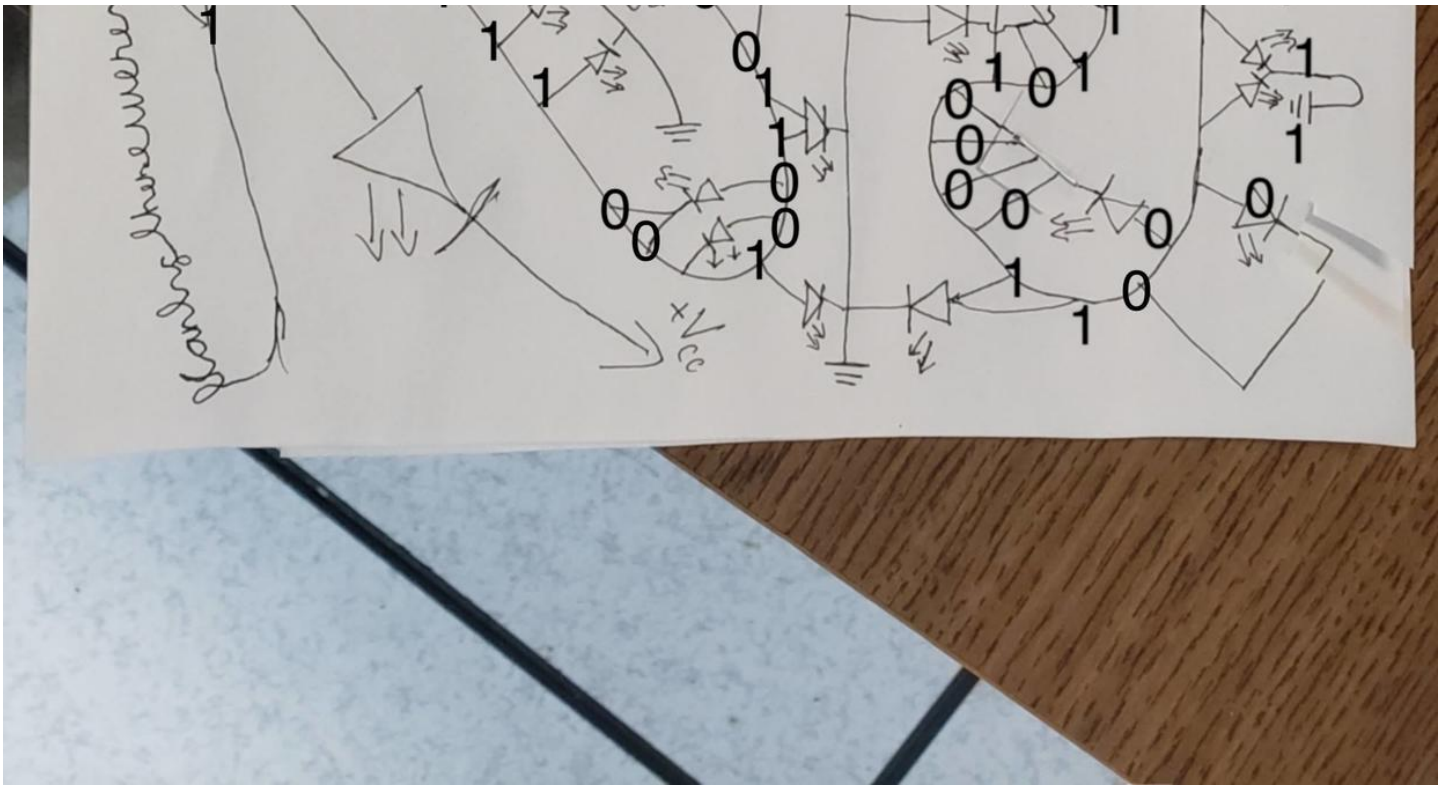




题都没看懂... 太真实了

网上搜到师傅的 wp，就是一张图片，绝了：

+Vcc
Counts as nothing
Counts as 2 Each
+Vcc
mothernoodles



打扰了。

> flag:flag{owmyhand}

Algebra

计算一元一次方程，开始想一个一个手算，实在算不动，太多了。写个python脚本：

因为我的那个计算算法可能会出错...所以加了捕获异常，捕获到异常直接重新连接...

```python
# -*- coding:utf-8 -*-
import socket
def solve(eq, var="X"):
    eq1 = eq.replace("=", "-(") + ")"
    c = eval(eq1, {var: 1j})
    if (-c.real == 0):
        return 0
    else:
        return -c.real / c.imag
while 1:
    try:
        time = 1
        sc = socket.socket()
        host = "misc.chal.csaw.io"
        port = 9002
        addr = (host, port)
        sc.connect(addr)
        print sc.recv(1024)
        while 1:
            print "-----------round {}--------------".format(time)
            data = sc.recv(1024)
            print data
            if "flag" in data:
                exit()
            if time == 1:
                shizi = data.split('\n')[0]
            else:
                shizi = data.split('\n')[1]
            result = str(solve(shizi))
            print "result:", result
            sc.send(result + "\n")
            time += 1
    except BaseException:
        if "flag" in data:
            break
        sc.shutdown(2)
        sc.close()
        continue
```

> flag: flag{y0u_s0_60od_aT_tH3_qU1cK_M4tH5}